1.10 *The k-means algorithm.* In this exercise we apply the k-means algorithm to the example in §4.4.1 of the textbook.

Download the file `mnist_train.mat` from the course website and load it in MATLAB or Octave using the command `load mnist_train`. This creates two variables: a 784×60000 matrix `digits` and a 1×60000 matrix `labels`. We will not need `labels`. Each column of `digits` is a 28×28 grayscale image, stored as a vector of length $28^2 = 784$ with elements between 0 and 1 (0 denotes a black pixel and 1 a white pixel). Figure 4.6 in the book shows the first 25 images. To display the image in the ith column of `digits` you can use the commands

```
X = reshape(digits(:,i), 28, 28);
imshow(X);
```

The first line converts column i of `digits` to a 28×28 matrix. The second command displays the matrix as an image. To speed up the computations we will use only the first 10000 digits:

```
digits = digits(:, 1:10000);
```

You are asked to apply the k-means algorithm to this set of $N = 10000$ vectors, with $k = 20$ groups, and starting from a random initial group assignment (as opposed to starting from 20 randomly generated group representatives, as in Algorithm 4.1 (page 72–73).

In the following list of MATLAB hints and comments we assume that the 20 group representatives are stored as columns of a 784×20 matrix `Z`, and that the group assignment is represented by a 1×10000 matrix (*i.e.*, row vector) `group`. The ith element `group(i)` is an integer between 1 and 20, with the index of the group that column i of `digits` is assigned to.

- You can create an initial group assignment using the `randi` function:

  ```
  group = randi(20, 1, 10000);
  ```

  This generates a pseudorandom 1×10000 matrix of integers between 1 and 20.
- Since we start from a random partition, the order of the two steps in algorithm 4.1 is switched. The first step in each cycle is to compute the group representatives for the current assignment. We can find the columns of `digits` that are assigned to group i using the function `find`. The command
  ```
  I = find(group == i);
  ```
  defines an index vector I such that `digits(:,I)` is the submatrix of `digits` with the columns assigned to class i. To find the average of the columns in this matrix, you can use for-loops, or the MATLAB functions `sum` or `mean`. (Be sure to look up what `sum` or `mean` do when applied to a matrix; see `help sum` and `help mean`.)
- We evaluate the quality of the clustering using the clustering objective

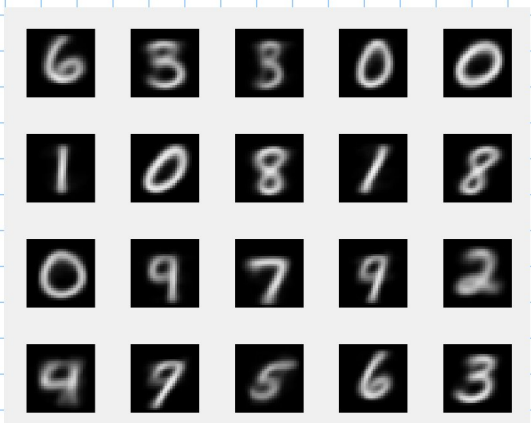
$$J = \frac{1}{N} \sum_{i=1}^{N} \min_{j=1,\ldots,k} \|x_i - z_j\|^2.$$

  The algorithm is terminated when J is nearly equal in two successive iterations (*e.g.*, we terminate when $|J - J_{\text{prev}}| \leq 10^{-5} J$, where J_{prev} is the value of J after the previous iteration).
- After running the k-means algorithm you can display the representative vectors of the 20 groups as follows:

  ```
  for k=1:20
      subplot(4,5,k)
      imshow(reshape(Z(:,k), 28, 28));
  end
  ```

  This produces a figure similar to figures 4.8 and 4.9 in the textbook. Your results will be different because figures 4.8 and 4.9 were computed using the full set of 60000 digits and, moreover, the result of the k-means algorithm depends on the starting point.
- Include the MATLAB code and a figure of a typical set of group representatives with your solution.

```
load ('mnist_train.mat');
digits = digits(:,1:10000);
group = randi(20,1,10000);
Z=zeros(284,20);
D= zeros(20,10000);
k=20;
Jprev = NaN;
for iter = 1:200
    for k = 1:20
        I = find(class == k);
        Z(:,k) = mean(digits(:,I), 2);
    end;
    for k =1:20
        D(k,:) =sqrt(sum((digits-Z(:,k*ones(1,10000))).^2));
    end;
    [d,group]=min(D);
    J = (1/10000) * norm(d)^2;
    if iter >1
        if abs(J-Jprev) < (10^(-5))*J, break;
        end;
        Jprev=J;
    end;
end;
for k= 1:20
    subplot(4,5,k);
    imshow(reshape(Z(L,k),28,28));
end;
```



6.8 *Cash flow to bank account balance.* The T-vector c represents the cash flow for an interest bearing bank account over T time periods. Positive values of c indicate a deposit, and negative values indicate a withdrawal. The T-vector b denotes the bank account balance in the T periods. We have $b_1 = c_1$ (the initial deposit or withdrawal) and

$$b_t = (1+r)b_{t-1} + c_t, \quad t = 2, \ldots, T,$$

where $r > 0$ is the (per-period) interest rate. (The first term is the previous balance plus the interest, and the second term is the deposit or withdrawal.)

Find the $T \times T$ matrix A for which $b = Ac$. That is, the matrix A maps a cash flow sequence into a bank account balance sequence. Your description must make clear what all entries of A are.

$b_t = (1+r)\,b_{t-1} + c_t \qquad t = 2, \ldots, T$

$b_1 = c_1$

$b_2 = (1+r)\,b_1 + c_2 \rightarrow b_2 = (1+r)\,c_1 + c_2$

$b_3 = (1+r)\,b_2 + c_3 \rightarrow (1+r)\left[(1+r)\,c_1 + c_2\right] + c_3 \rightarrow (1+r)^2 c_1 + (1+r)\,c_2 + c_3$

$b_4 = (1+r)\,b_3 + c_4$

$\rightarrow (1+r)\left[(1+r)^2 c_1 + (1+r)\,c_2 + c_3\right] + c_4$

($t-1$, $t-2$, $t-3$, $t-4$)

$\rightarrow (1+r)^3 c_1 + (1+r)^2 c_2 + (1+r)\,c_3 + c_4$

$\vdots$

$b_T = (1+r)\,b_{T-1} + c_T$

($T-(T-1) = 1$, $T-T = 0$)

$\rightarrow (1+r)^{T-1} c_1 + (1+r)^{T-2} c_2 + (1+r)^{T-3} c_3 + \ldots + (1+r)^1 c_{T-1} + (1+r)^0 c_T$

$b = Ac \rightarrow \begin{bmatrix} & A & \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_T \end{bmatrix}$ (b)

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ (1+r) & 1 & \cdots & 0 \\ (1+r)^2 & (1+r) & \cdots & \\ (1+r)^3 & (1+r)^2 & \cdots & \\ \vdots & (1+r)^3 & \cdots & \vdots \\ & \vdots & \ddots & 0 \\ (1+r)^{T-1} & (1+r)^{T-2} & \cdots & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ \\ \vdots \\ c_T \end{bmatrix} \quad \cdots \quad b_1 = c_1$$

6.9 *Multiple channel marketing campaign.* Potential customers are divided into m market segments, which are groups of customers with similar demographics, *e.g.*, college educated women aged 25–29. A company markets its products by purchasing advertising in a set of n channels, *i.e.*, specific TV or radio shows, magazines, web sites, blogs, direct mail, and so on. The ability of each channel to deliver impressions or views by potential customers is characterized by the *reach matrix*, the $m \times n$ matrix R, where R_{ij} is the number of views of customers in segment i for each dollar spent on channel j. (We assume that the total number of views in each market segment is the sum of the views from each channel, and that the views from each channel scale linearly with spending.) The n-vector c will denote the company's purchases of advertising, in dollars, in the n channels. The m-vector v gives the total number of impressions in the m market segments due to the advertising in all channels. Finally, we introduce the m-vector a, where a_i gives the profit in dollars per impression in market segment i. The entries of R, c, v, and a are all nonnegative.

(a) Express the total amount of money the company spends on advertising using vector/matrix notation.

(b) Express v using vector/matrix notation, in terms of the other vectors and matrices.

(c) Express the total profit from all market segments using vector/matrix notation.

(d) How would you find the single channel most effective at reaching market segment 3, in terms of impressions per dollar spent?

(e) What does it mean if R_{35} is very small (compared to other entries of R)?

m : market segment
n : channels
R : m×n ; R_{ij} , i = dollar spent on channel j

c : n vector ; ad purchases in n channel
v : m vector ; # of impressions on m market
a : m vector ; a_i is profit per impression in market m

a) $c^T 1$ or $1^T c$

b) $v = Rc \rightarrow \langle m \times 1 \rangle = \langle m \times n \rangle \langle n \times 1 \rangle$

c) $a^T 1$ or $1^T a$

d) inside our reach matrix R , segment = i

thus $R_{3,j}$ and we look into these values to find the largest.

e) it means it is a small demographic.
there are not many dollars spent and there are not many dollars spent there.

7.15 *Channel equalization.* We suppose that $u_1, \ldots, u_m$ is a signal (time series) that is transmitted (for example by radio). A receiver receives the signal $y = c * u$, where the n-vector c is called the channel impulse response. (See page 138.) In most applications n is small, *e.g.*, under 10, and m is much larger. An *equalizer* is a k-vector h that satisfies $h * c \approx e_1$, the first unit vector of length $n + k - 1$. The receiver equalizes the received signal y by convolving it with the equalizer to obtain $z = h * y$.

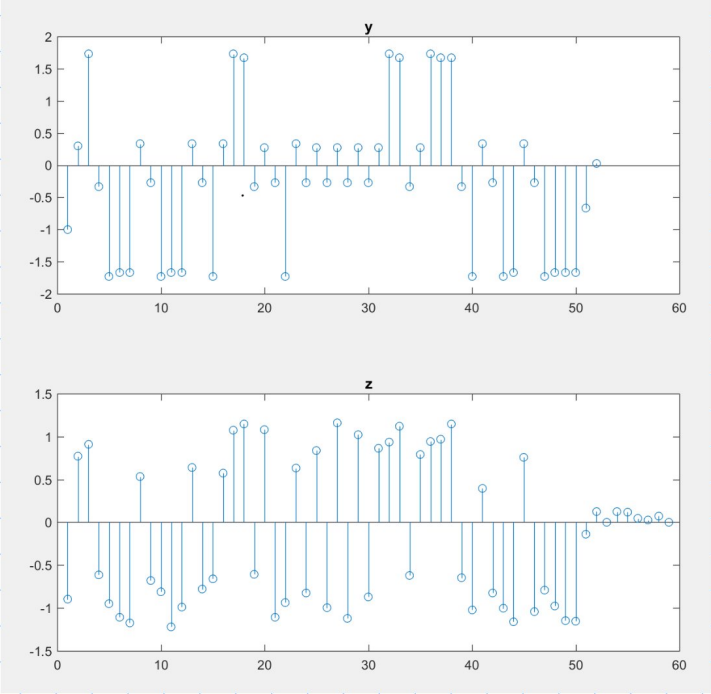
(a) How are z (the equalized received signal) and u (the original transmitted signal) related? *Hint.* Recall that $h * (c * u) = (h * c) * u$.

(b) *Numerical example.* Generate a signal u of length $n = 50$, with each entry a random value that is either -1 or $+1$. Plot u and $y = c * u$, with $c = (1, 0.7, -0.3)$. Also plot the equalized signal $z = h * y$, with

$$h = (0.9, -0.5, 0.5, -0.4, 0.3, -0.3, 0.2, -0.1).$$

a) $z = h * y = h * (c * u) = (h * c) * u \Rightarrow e_1 * u$

b)




```
%hw2pT715
k=50;
u=sign(randn(k,1));
c=[1;0.7; -0.03];
h=[0.9, -0.5, 0.5, -0.4, 0.3, -0.3, 0.2, -0.1];

y=conv(c,u);
subplot(2,1,1);
stem(y);
title('y');
z=conv(h,y);
subplot(2,1,2);
stem(z);
title('z');
```

8.3 *Cross-product.* The cross product of two 3-vectors $a = (a_1, a_2, a_3)$ and $x = (x_1, x_2, x_3)$ is defined as the vector

$$a \times x = \begin{bmatrix} a_2x_3 - a_3x_2 \\ a_3x_1 - a_1x_3 \\ a_1x_2 - a_2x_1 \end{bmatrix}.$$

The cross product comes up in physics, for example in electricity and magnetism, and in dynamics of mechanical systems like robots or satellites. (You do not need to know this for this exercise.)

Assume a is fixed. Show that the function $f(x) = a \times x$ is a linear function of x, by giving a matrix A that satisfies $f(x) = Ax$ for all x.

$$a \times x = \begin{bmatrix} a_2x_3 - a_3x_2 \\ a_3x_1 - a_1x_3 \\ a_1x_2 - a_2x_1 \end{bmatrix}$$

linear $\quad f(x) = a \times x = Ax$

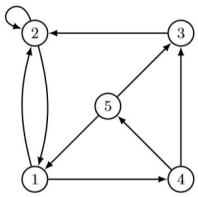
$$\begin{pmatrix} 0 \\ a_3 \\ -a_2 \end{pmatrix} x_1 + \begin{pmatrix} -a_3 \\ 0 \\ a_1 \end{pmatrix} x_2 + \begin{pmatrix} a_2 \\ -a_1 \\ 0 \end{pmatrix} x_3 = \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

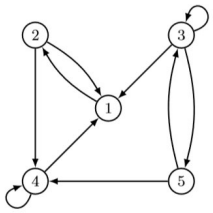
(A ↓)

$$f(x) = Ax$$

2.4 Let A be an $n \times n$ matrix with nonnegative elements ($A_{ij} \geq 0$ for all i, j). We define a directed graph G_A with vertices (nodes) $1, \ldots, n$, and an arc (directed edge) from vertex j to vertex i if only if $A_{ij} > 0$. The figure shows the graphs for the matrices

$$A_1 = \begin{bmatrix} 0 & 3 & 0 & 0 & 5 \\ 2 & 1 & 4 & 0 & 0 \\ 0 & 0 & 0 & 3 & 2 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \end{bmatrix}, \qquad A_2 = \begin{bmatrix} 0 & 3 & 1 & 2 & 0 \\ 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 1 \\ 0 & 2 & 0 & 1 & 4 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}.$$





Define $B = I + A$

$(B^2)_{ij} = \sum B_{ik} B_{kj} > 0$ if $B_{ik} > 0$ and $B_{kj} > 0$

- $k \neq j$ & $k \neq i \Rightarrow$ path $j \to k \to i$ w/ length = 2
- $k = j \neq i$ or $k \neq j = i \Rightarrow$ path $j \to i$ ($k \to j$) w/ length = 1
- $k = j = i$, $B_{ik} = B_{kk} > 0 \Rightarrow$ diagonal elements of B^2 are positive

$\therefore (B^m)_{ij} > 0$ iff path of length m or less from $j \to i$

$$(B^m)_{ij} = \sum_{k=1}^{n} (B^{m-1})_{ik} B_{kj} > 0 \text{ iff at least a } k \text{ s.t. } (B^{m-1})_{ik} > 0 \text{ and } B_{kj} > 0$$

- $k = j \to$ path of length $m-1$ or less from $j \to i$
- $k \neq j \to$ path of length m or less from $j \to i$

$B^{n-1} = (I + A)^{n-1}$ has "+" elements

2.6 A matrix C is defined as

$$C = Auv^T B$$

where A and B are $n \times n$-matrices, and u and v are n-vectors. The product on the right-hand side can be evaluated in many different ways, *e.g.*, as $A(u(v^T B))$ or as $A((uv^T)B)$, etc. What is the fastest method (requiring the least number of flops) when n is large?

$C = (Auv^T B)$ $\quad C = A(u(v^T B))$

$C = (Au)(v^T B)$ $\quad C = A((uv^T)B)$

$Au = 2n^2$ flops

$v^T B = 2n^2$ flops

$\therefore (Au)(v^T B) = n^2$ flops

$2n^2 + 2n^2 + n^2 = 5n^2$ flops

All other combinations will increase flop count by an extra n (i.e. n^3)

2.7 The *Kronecker product* of two $n \times n$ matrices A and B is the $n^2 \times n^2$ matrix

$$A \otimes B = \begin{bmatrix} A_{11}B & A_{12}B & \cdots & A_{1n}B \\ A_{21}B & A_{22}B & \cdots & A_{2n}B \\ \vdots & \vdots & & \vdots \\ A_{n1}B & A_{n2}B & \cdots & A_{nn}B \end{bmatrix}.$$

For example,

$$\begin{bmatrix} 1 & 3 \\ 2 & -1 \end{bmatrix} \otimes \begin{bmatrix} 3 & 4 \\ -5 & 6 \end{bmatrix} = \begin{bmatrix} 3 & 4 & 9 & 12 \\ -5 & 6 & -15 & 18 \\ 6 & 8 & -3 & -4 \\ -10 & 12 & 5 & -6 \end{bmatrix}.$$

Suppose the $n \times n$ matrices A and B, and an n^2-vector x are given. Describe an efficient method for the matrix-vector multiplication

$$y = (A \otimes B)x = \begin{bmatrix} A_{11}B & A_{12}B & \cdots & A_{1n}B \\ A_{21}B & A_{22}B & \cdots & A_{2n}B \\ \vdots & \vdots & & \vdots \\ A_{n1}B & A_{n2}B & \cdots & A_{nn}B \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}.$$

(On the right we partitioned x in subvectors x_i of size n.) What is the complexity of your method? How much more efficient is it than a general matrix-vector multiplication of an $n^2 \times n^2$ matrix and an n^2-vector?

Bx gives $2n^3$ flops giving Bx_1 & Bx_2

$$y = \begin{bmatrix} A_{11}Bx_1 + A_{12}Bx_2 + \ldots + A_{1n}Bx_n \\ A_{21}Bx_1 + A_{22}Bx_2 + \ldots + A_{2n}Bx_n \\ \vdots \\ A_{n1}Bx_1 + A_{n2}Bx_2 + \ldots + A_{nn}Bx_n \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

y is partioned into n vectors each with $(2n^2)$ flops $\therefore$ $(2n^3)$

y can be obtained by $\langle n \times n \rangle \langle n \times n \rangle$ matrix product