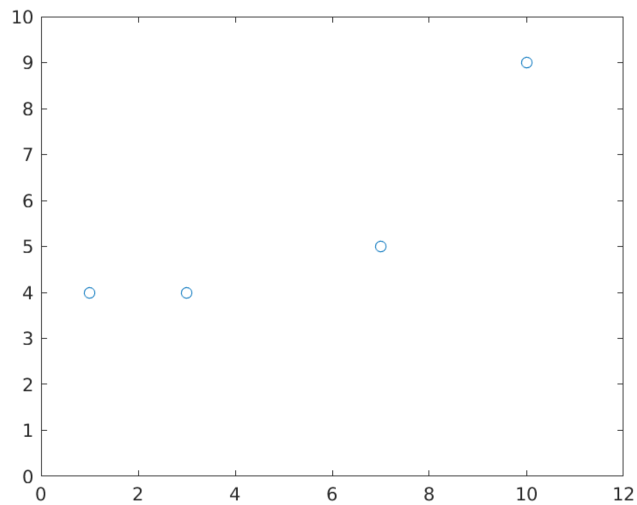
Exercise 14.1

1.

$$\boldsymbol{D} = \begin{bmatrix} -1 & 3 \\ 1 & 4 \\ 3 & 4 \\ 7 & 5 \\ 10 & 9 \end{bmatrix}$$

Create a plot of $\boldsymbol{D}$ as a set of points on the xy plane.

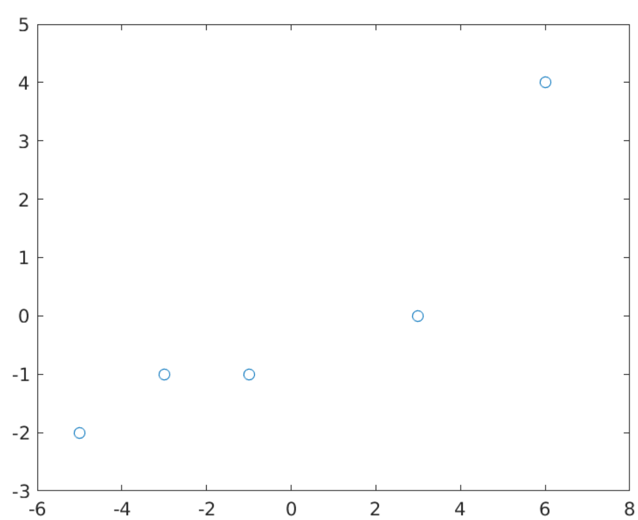
Solution.



■

2. Define a matrix $\tilde{\boldsymbol{D}}$ which is the mean-centered version of $\boldsymbol{D}$ and plot $\tilde{\boldsymbol{D}}$ as a set of points in the xy-plane.

Solution.



■

3. The principal components (p_1 and p_2) are the eigenvectors of the covariance matrix of the mean-centered $\tilde{\boldsymbol{D}}$. Compute p_1 and p_2.

   Solution.

$$p_1 = \begin{bmatrix} 0.4435 \\ -0.8963 \end{bmatrix}$$
$$p_2 = \begin{bmatrix} -0.8963 \\ 0.4435 \end{bmatrix}$$

■

4. Compute the projection of your data onto the eigenvector which corresponds to the largest eigenvalue.

   Solution.

$$\boldsymbol{B} = \begin{bmatrix} 5.3684 \\ 3.1323 \\ 1.3398 \\ -2.6889 \\ -7.1516 \end{bmatrix}$$

■

Exercise 14.2

1. Can you recreate $\boldsymbol{D}$ perfectly from $\boldsymbol{B}$?

   Solution. No. The data along p_1 is lost. ■

2. What would have happened if you had created $\boldsymbol{B}$ using only information about the values along p_1 instead of p_2?

   Solution. Because the direction of p_2 is where the most variance lies, the projection onto p_1 would have less resolution than the projection onto p_2. ■

3. How might you quantify how well you can represent $\boldsymbol{D}$ in this reduced dimensionality form?

   Solution. The error $\boldsymbol{B} - \boldsymbol{D}$ ■

4. If you received a new piece of data, how would you go about representing this as a linear combination of p_1 and p_2?

Solution.

$$(d \cdot p_1)p_1 + (d \cdot p_2)p_2$$

■

Exercise 14.3

1. Create a covariance matrix R using the 10 years worth of temperature data from Boston, Washington DC and Sao Paolo.

   Solution.

```
load('data/avg_temperatures_pt2.mat')

A = (1/sqrt(17304)) * [(b_tr-mean(b_tr))/std(b_tr); (w_tr-mean(w_tr))/
    ↪ std(w_tr) (s_tr-mean(s_tr))/std(s_tr)]
R = A' * A
```

$$\boldsymbol{R} = \begin{bmatrix} 0.4221 & 0.3999 & -0.2286 \\ 0.3999 & 0.4221 & -0.2291 \\ -0.2286 & -0.2291 & 0.4221 \end{bmatrix}$$

■

2. Perform an eigendecomposition of the matrix $\boldsymbol{R}$, and make a new matrix $\boldsymbol{V}_p$ which has the 2 eigenvectors corresponding to the 2 largest eigenvalues of $\boldsymbol{R}$.

   Solution.

```
[v, lambda] = eig(R);
V_p = [v(:, 3) v(:,2)]
```

$$\boldsymbol{V}_p = \begin{bmatrix} 0.5786 & 0.4995 \\ 0.5723 & -0.8119 \\ 0.5811 & 0.3022 \end{bmatrix}$$

■

3. Create centered versions of the new temperature data vectors, and create a 3×365 matrix $\boldsymbol{T}$ which has the centered temperatures of Boston, Washington DC, and Sao Paolo as its rows.

   Solution.

```
T = [(b_new-mean(b_new))/std(b_new) (w_new-mean(w_new))/std(w_new) (
    ↪ s_new-mean(s_new))/std(s_new)]'
```

■

4. Take the dot product of each column of the matrix $\boldsymbol{T}$ with the two eigenvectors in matrix V_p.

$$\alpha_{1i} = \boldsymbol{v}_1^\intercal \boldsymbol{t}_i$$
$$\alpha_{1i} = \boldsymbol{v}_2^\intercal \boldsymbol{t}_i$$

Solution.

```
alpha = V_p' * T
```

■

5. You can now check how well your compression worked, by using the values of α_{1i} and α_{2i} to reconstruct 365 different 3×1 vectors each representing the temperatures for the three cities over the 365 days. Let $\hat{\boldsymbol{t}}_i$ represent the reconstructed temperature vector on the i-th day. Using what you know about projections onto orthonormal vectors, reconstruct $\boldsymbol{t}_i$ using α_{1i}, α_{2i}, v_1 and v_2. Repeat this for all 365 days.

Solution.

```
t_hat = V_p * alpha
```

■

6. On the same axes, plot the original and reconstructed temperature for Boston. Repeat this for Washington DC and Sao Paolo. Observe how close the reconstructions are, for the different data sets.

Solution.

```
hold on
plot3(T(1,:), T(2,:), T(3,:), 'b.')
plot3(t_hat(1,:), t_hat(2,:), t_hat(3,:), 'r.')
grid on
```

■

7. How accurately do you think you can represent the data if you used 3 eigenvectors instead of 2?

Solution. If all 3 eigenvectors are used, the data would be perfectly represented. ■

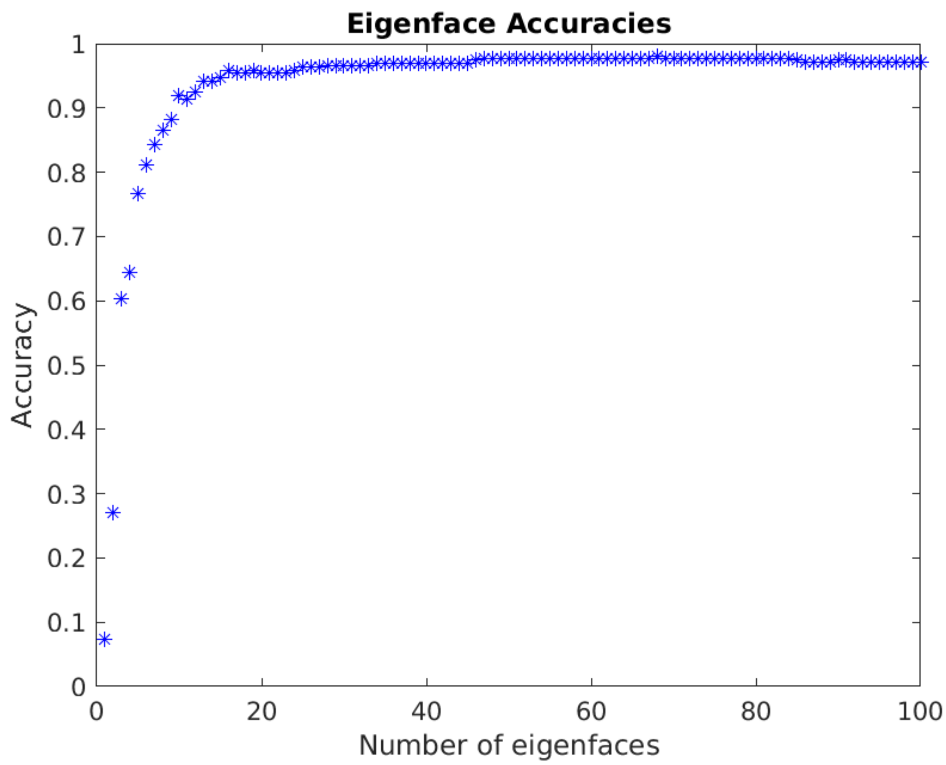
Exercise 14.5

1. Implement the eigenfaces algorithm.

   Solution.

```
function accuracy = face_recognition(num_eig, use_smiles, show_faces)
    if nargin < 1
        % specify number of eigenvectors to use
        num_eig = 16;
    end

    if nargin < 2 || use_smiles == 1
        load('data/classdata_smile.mat')
        load('data/classdata_no_smile.mat')
    else
        load('data/classdata_train.mat')
        load('data/classdata_test.mat')
    end

    if nargin < 3
        show_faces = 0;
    end

    % flatten images into vectors
    train = reshape(grayfaces_train, size(grayfaces_train, 1) * size(
        ↪ grayfaces_train, 2), size(grayfaces_train, 3));
    test = reshape(grayfaces_test, size(grayfaces_test, 1) * size(
        ↪ grayfaces_test, 2), size(grayfaces_test, 3));

    % get covariance matrix
    train_adj = train - mean(train);
    R_train = train_adj * train_adj';

    % get eigenvectors of the covariance matrix
    [v, ~] = eigs(R_train, num_eig);

    if show_faces
        for i = 1 : num_eig
            subplot(ceil(sqrt(num_eig)), ceil(sqrt(num_eig)), i);
            imagesc(reshape(v(:,i), [64 64]));
            colormap('gray');
        end
    end
```

```
    % get faces in face space
    train_facespace = v' * train;
    test_facespace = v' * test;

    % nearest neighbor search
    % grabbed this from the solutions
    % this compares the faces from the test set and finds the closest
    % representation from the training set
    NN = knnsearch(test_facespace', train_facespace');

    % check accuracy
    accuracy = mean(subject_train(NN) == subject_test);
end
```



■