Contents

CHAPTER 1

Vectors and Matrices

Linear algebra is a specialized form of algebra that can represent and manipulate sets of variables that are linearly related to one another. One of the basic operations is the multiplication of a matrix by a vector. As you work through this module, you'll see that you already know the fundamentals (vectors, scalars, dot products) and how to apply these concepts to practical problems. You've also had some experience with matrices in the form of spreadsheets.

Matrices can represent:

- A linear transformation, such as rotation, scaling, and skewing. You apply a transformation to a vector by multiplying the vector by a matrix.
- A system of linear equations. Linear algebra provides various methods that you can use to find the solution vector.

1.1 Applications of Matrix-Vector Multiplication

Many areas in engineering and science rely on matrix-vector multiplication. These are just a few examples. As you encounter more topics in science and engineering, you will find that matrix-vector multiplication is crucial to many other fields.

1.1.1 Computer Graphics

When you play a video game or watch the latest CG animation, matrix operations transform objects in the scene to make them appear as if moving, getting closer, and so on. You can represent the vertices of objects as vectors, and then apply a transformation matrix.

1.1.2 Data Analysis

We live in an era in which it's easy to collect so much data that it's difficult to make sense of the data by just looking at it. You can represent the data in matrix form and then find a solution vector. For example, scientists use this technique to figure out the effectiveness of drug treatments on disease.

1.1.3 Economics

Take a look at financial section of any news organization and you'll see headlines such as "Economic Data Points to Faster Growth" or "Is the Inflation Battle Won?" Economists can use systems of linear equations to represent economic indicators, such as consumer consumption, government spending, investment rate, and gross national product. By using various methods that you'll learn about later, they can get a good idea of the state of the economy.

1.1.4 Engineering

Engineering couldn't do without vector-matrix multiplication. For example, the orbital dynamics of space travel relies on it. Engineers must predict and calculate the the motion of planetary bodies, satellites, and spacecraft. By solving systems of linear equations engineers can make sure that a spacecraft travels to its destination without running into a satellite or space rock.

1.1.5 Image Processing

An image is a matrix of pixel values. When you take a selfie and apply a filter, the image app applies a transformation to the image matrix. A simple operation would be to change the tint of the image. A more complex operation would be to skew the image to make it distorted, like a funhouse mirror.

1.2 Vector-Matrix Multiplication

Let's take a look at the general form of vector-matrix multiplication. Given a matrix A of size $m \times n$ and a vector x of size $n \times 1$, the product Ax is a new vector of size $m \times 1$.

You compute the i-th component of the product vector Av by taking the dot product of the i-th row of A and the vector v:

$$(Av)_i = \sum_{j=1}^{n} a_{ij} x_j$$

where a_{ij} is the element in the i-th row and j-th column of A, and v_j is the j-th element of v.

This is the general form of a matrix and a vector, written to show the specific components of each:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \dots \\ a_{m1} & a_{m2} & a_{m3} & \dots & a_{mn} \end{bmatrix}$$

$$v = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \dots \\ v_m \end{bmatrix}$$

$$Av = \begin{bmatrix} v_1 * a_{11} + v_2 * a_{12} + v_3 * a_{13} + \dots + v_m * a_{1n} \\ v_1 * a_{21} + v_2 * a_{22} + v_3 * a_{23} + \dots + v_m * a_{2n} \\ \dots \\ v_1 * a_{m1} + v_2 * a_{m2} + v_3 * a_{m3} + \dots + v_m * a_{mn} \end{bmatrix}$$

Let's look at a specific example.

$$A = \begin{bmatrix} 2 & 4 & 6 \\ 3 & 5 & 7 \\ 1 & 2 & 3 \\ 8 & 6 & 2 \end{bmatrix}$$

$$v = \begin{bmatrix} -2 \\ 1 \\ 3 \end{bmatrix}$$

Solution:

$$= \begin{bmatrix} -2*2+1*4+3*6 \\ -2*3+1*5+3*7 \\ -2*1+1*2+3*3 \\ -2*8+1*6+3*2 \end{bmatrix}$$

$$= \begin{bmatrix} 18 \\ 20 \\ 9 \\ -4 \end{bmatrix}$$

$$= (18, 20, 9, -4)$$

Exercise 1 Vector Matrix Multiplication

Multiply the array A with the vector v. Compute this by hand, and make sure to show your computations.

$$A = \begin{bmatrix} 1 & -2 & 3 & 5 \\ -4 & 2 & 7 & 1 \\ 3 & 3 & -9 & 1 \end{bmatrix}$$

$$v = \begin{bmatrix} 2 \\ 2 \\ 6 \\ -1 \end{bmatrix}$$

Working Space

Answer on Page 27

1.2.1 Vector-Matrix Multiplication in Python

Most real-world problems use very large matrices where it becomes impractical to perform calculations by hand. As long as you understand how matrix-vector multiplication is done, you'll be equipped to use a computing language, like Python, to do the calculations for you.

Create a file called `vectors_matrices.py` and enter this code:

```
// import the python module that supports matrices
import numpy as np

// create an array
a = np.array([[ 5, 1 ,3, -2],
              [ 1, -1 ,8, 4],
              [ 6, 2 ,1, 3]])

// create a vector
b = np.array([1, 2, 3,-8])

//calculate the dot product
print(a.dot(b))
```

When you run it, you should see:

```
[16  6  8]
```

1.3 Where to Learn More

Watch this video from Khan Academy about matrix-vector products: https://rb.gy/frga5

CHAPTER 2

Linear Combinations

A linear combination of vectors is the addition of two or more scaled vectors. For example, given two vectors, v_1, v_2 and two scalars a_1, a_2, you'd write their linear combination as:

$$x\mathbf{w} = a_1\mathbf{v}_1 + a_2\mathbf{v}_2$$

The scalars can be any real number. The vectors can be of any dimension.

Let's take a more generalized approach. Given vectors $\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n \in \mathbb{R}^m$ and scalars $a_1, a_2, ..., a_n \in \mathbb{R}$, a linear combination of these vectors is any vector of the form

$$\mathbf{w} = a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + ... + a_n\mathbf{v}_n$$

Each scalar a_i scales the corresponding vector $\mathbf{v}_i$, and added together, the results are produce a new vector $\mathbf{w}$.

Let's look at an example that has 4 vectors and their scalars.

$$a_1 = 1, v_1 = [9, 1, 2]$$
$$a_2 = -1, v_2 = [8, -3, 4]$$
$$a_3 = 3, v_3 = [6, 0, 1]$$
$$a_4 = -4, v_4 = [3, 7, 2]$$

As a linear combination:

$$\mathbf{w} = 1 * [9, 1, 2] + (-1) * [8, -3, 4] + 3 * [6, 0, 1] + (-4) * [3, 7, 2]$$

After multiplying each vector by its associated scalar.

$$\mathbf{w} = [9, 1, 2] + [-8, 3, -4] + [18, 0, 3] + [-12, -28, -8]$$

When combined:

$$\mathbf{w} = [7, -24, -7]$$

Exercise 2 Linear Combination

Calculate the linear combination for vectors v_1, v_2, v_3 and scalars a_1, a_2, a_3 where:

$$a1 = 2, v1 = [2, 4, 8]$$
$$a2 = -2, v2 = [8, -6, 3]$$
$$a3 = 4, v3 = [7, 9, 2]$$

Make sure to show all your work.

Working Space

Answer on Page 27

2.1 Weighted Averages of Vectors

A weighted average of vectors is a specific type of linear combination where the coefficients (or weights) a_i are non-negative and sum to 1:

$$\sum_{i=1}^{n} a_i = 1, \quad a_i \geq 0$$

A weighted average of vectors $\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n$ is then defined as

$$\mathbf{w} = a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + ... + a_n\mathbf{v}_n$$

In this case, each a_i not only scales the corresponding vector $\mathbf{v}_i$, but also represents the proportion of that vector in the final average vector $\mathbf{w}$.

Weighted averages are useful when you want to attribute the contribution of one feature or item over another. For example, a teacher might figure a student's final grade using exam scores, class participation, and a final project. The exam scores might make up 65% of the final grade, class participation 10%, and a final project 25%. Thus giving the formula for a grade as:

$$\mathbf{Grade} = .65 * ExamScores + .10 * Participation + .25 * FinalProject$$

The teacher defines the weights, making sure they sum to 1.0.

Let's look at an example where the weights don't sum to 1.0. A store that sells umbrellas might have to get the umbrella stock from three different manufacturers. The store owner buys 100 umbrellas at a cost of \$2.10 each, 50 umbrellas cost \$1.85 each, and 200 umbrellas cost \$2.00.

$$\mathbf{TotalCost} = 2.10 * 100 + 1.85 * 50 + 2.00 * 200 = 702.5$$

To calculate the weighted average, divide the total cost by the number of items.

$$\mathbf{WeightedAverage} = 702.5/350 = 2.01$$

Exercise 3 Weighted Average

A concert sells 300 tickets in the balcony at $50 each, 100 tickets on the main floor at $75 each, and 50 tickets in the section closest to the stage at $150 each. What's the weighted average?

Working Space

Answer on Page 28

2.2 Weighted Averages of Vectors in Python

Create a file called `linearCombos.py` and enter this code:

```
// import the python module that supports matrices
import numpy as np

// an array for number of umbrellas by manufacturer
items = np.array([100, 50, 200])

// weights are the cost of item by manufacturer
weights = np.array([2.10, 1.85, 2.00])

// create an array for total cost for each manufacturer
costPerManufacturer=items * weights

// sum the individuals costs to get the total
totalCost = np.sum(costPerManufacturer)

// get number of items
numItems = np.sum(items)

// you are ready to calculated the weighted average
weightedAverage = totalCost/numItems
print(weightedAverage)
```

When you run this code, you should get a weighted average of $2.01 when rounded to the nearest cent.

CHAPTER 3

Vector Spans and Independence

A vector span is the collection of vectors obtained by scaling and combining the original set of vectors in all possible proportions. Formally, if the set $S = \{v_1, v_2, ..., v_n\}$ contains vectors from a vector space V, then the span of S is given by:

$$\mathrm{Span}(S) = \{a_1 v_1 + a_2 v_2 + ... + a_n v_n : a_1, a_2, ..., a_n \in \mathbb{R}\} \tag{3.1}$$

This means that any vector in the Span(S) can be written as a linear combination of the vectors in S.

Vector spans have practical applications in a number of fields. Computer graphics and physics are two of them. For example, in space travel, knowing the vector span is essential to calculating a slingshot maneuver that will give spacecraft a gravity boost from a planet. For this, you'd need to know the gravity vector of the planet relative to the sun and the velocity vectors that characterize the spacecraft. Engineers would use this information to figure out the trajectory angle that would allow the spacecraft to achieve a particular velocity in the desired direction.

3.1 Vector Independence

A set of vectors $S = \{v_1, v_2, ..., v_n\}$ is linearly independent if the only solution to the equation $a_1v_1 + a_2v_2 + ... + a_nv_n = 0$.

is $a_1 = a_2 = ... = a_n = 0$. This means that no vector in the set can be written as a linear combination of the other vectors.

If there exists a nontrivial solution (i.e., a solution where some $a_i \neq 0$), then the vectors are said to be linearly dependent. This means that at least one vector in the set can be written as a linear combination of the other vectors.

The concept of vector independence is fundamental to the study of vector spaces, bases, and rank. You'll learn more about these concepts in future modules.

3.1.1 Dependent Vectors

Let's start by looking at two vectors.

$$v_1 = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$
$$v_2 = \begin{bmatrix} -14 \\ -28 \end{bmatrix}$$

These two vectors are dependent because $v_2 = -7 * v_2$. This is an obvious example but let's show it mathematically. If linearly independent, the two vectors must satisfy:

$$v_1a_1 + v_1a_2 = 0$$
$$v_2a_1 + v_2a_2 = 0$$

which is:

$$2a_1 - 14a_2 = 0$$
$$4a_1 - 28a_2 = 0$$

To solve, multiply the top equation by -2 and add it to the bottom:

$$2a_1 - 14a_2 = 0$$
$$0 + 0 = 0$$

The bottom equation drops out. Now olve for a_1 in the remaining equation:

$$a_1 = -7a_2$$

As you can see, one vector is a multiple of another.

$$a_1 \neq a_2 \neq 0$$

3.1.2 Independent Vectors

Let's see if these two vectors are independent.

$$v_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$v_2 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

To be independent,the two vectors must satisfy:

$$v_1 a_1 + v_1 a_2 = 0$$

$$v_2 a_1 + v_2 a_2 = 0$$

which is:

$$\begin{bmatrix} a_1 + 0 * a_2 \\ 0 * a_1 + a_2 \end{bmatrix}$$

So: $a_1 = a_2 = 0$ These vectors are not only independent, but they are orthogonal (perpendicular) to one another. You'll learn more about orthogonality later.

Here is an example whose solution isn't as obvious. You can solve using Gaussian elmination.

$$v_1 = [2, 1]$$

$$v_2 = [1, -6]$$

Rewrite as a system of equations:

$$a_1 * 2 + a_2 * 1 = 0$$

$$a_1 * 1 + a_2 * (-6) = 0$$

First swap the equations to that the the top equation has a coefficient of 1 for a_1:

$$a_1 - 6a_2 = 0$$

$$2a_1 + a_2 = 0$$

Next multiply row 1 by -2 and add it to row 2:

$$a_1 - 6a_2 = 0$$

$$0 - 11a_2 = 0$$

Multiply row 2 by 1 divided by 11.

$$a_1 - 6a_2 = 0$$

$$0 + a_2 = 0$$

Back substitute a_2 solution into the first equation:

$$a_1 = 0$$

$$a_2 = 0$$

Therefore

$$a_1 = a_2 = 0$$

and the two vectors are linearly independent.

Exercise 4 Vector Independence

Are these vectors independent?

$$[2, 1, 4]$$

$$[2, -1, 2]$$

$$[0, 1, -2]$$

Show your work.

Working Space

Answer on Page 28

3.2 Checking for Linear Independence Using Python

One way to use python to check for linear independence is to use the linalg.solve() function to solve the system of equations. You need to create an array that contains the coefficients of the variable and a vector that contains the values on the right-side of each equation. So far, you've either been given equations that equal 0 or you've manipulated each equation to be equal to 0.

Let's first see how to use python to solve the equations in the previous exercise. If the equations are linearly independent, then $a_1 = a_2 = a_3 = 0$

Create a file called linear_independence.Python and enter this code:

```
import numpy as np

A = np.array([[ 2, 2, 0],
              [ 1, -1,1],
              [4, 2, -2]])
b = np.array([0,0,0])
c = np.linalg.solve(A,b)
print(c)
```

Your should get this result, which shows the equations are linearly independent.

```
[ 0. -0.  0.]
```

But what happens if the equations are not independent? Let's make the first two equations dependent by making equation 1 two times equation 2. Enter this code into your file:

```
import numpy as np

D = np.array([[ 2, -2, 2],
              [ 1, -1,1],
              [4, 2, -2]])
e = np.array([0,0,0])
f = np.linalg.solve(D,f)
print(f)
```

You should get many lines indicating an error. Among the spew, you should see:

```
raise LinAlgError("Singular matrix")
```

So while the linalg.solve() function is quite useful for solving a system of independent linear equations, raising an error is not the most elegant way to figure out if the equations are dependent. That's where the concept of a determinant comes in. You'll learn about that in the next section. But for now, let's use the linalg.solve() function to find a solution for a set of equations known to be linearly independent.

$$4x_1 + 3x_2 - 5x_3 = 2$$

$$-2x_1 - 4x_2 - 5x_3 = 5$$

$$8x_2 + 8x_3 = -3$$

You will create a matrix that contains all the coeffients and a vector that contains the values on the right-side of the equations.

Enter this code into your file.

```
G = np.array([[4, 3, -5],
              [-2, -4, 5],
              [8, 8, 0]])
h = np.array([2, 5, -3])

j = np.linalg.solve(G, h)
print(j)
```

You should get this answer:

```
[ 2.20833333 -2.58333333 -0.18333333]
```

3.3 Determinants

The determinant of a matrix is a scalar value that can be calculated for a square matrix. If a matrix has linearly dependent rows or columns, the determinant is 0. One way to figure out if a set of equations are independent is to calculate the determinant.

For a matrix that is 2 by 2, the calculation is easy:

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

the determinant is:

$$\det(A) = (a * d) - (b * c)$$

For a larger matrix, finding the determinant is more complex and requires breaking down the matrix into smaller matrices until you reach te 2x2 form. The process is called expansion by minors. For our purposes, we simply want to first check to see if a matrix contains linearly independent rows and columns before using our Python code to solve. Modify your code so that is uses the $np.linalg.det()$ function. If the determinat is not zero, then you can call the $np.linalg.solve()$ function. Your code should look like this:

```
if (np.linalg.det(D) != 0):
    j = np.linalg.solve(D,e)
    print(j)
else:
     print("Rows and columns are not independent.")
```

3.4 Where to Learn More

Watch this video on *Linear Combinations and Vector Spans from Khan Academy*: `http://rb.gy/g1snk`

If you curious about the *Expansion of Minors*, see: `https://mathworld.wolfram.com/DeterminantExpansionbyMinors.html`

CHAPTER 4

Matrices

You've already had experience with matrices earlier in this module and also when you"'ve used spreadsheets. In this chapter you'll learn the types of matrices and get an introduction to some of the special matrices used for various calculations.

As you know, a matrix is a rectangular array of numbers arranged in rows and columns. The individual numbers in the matrix are called elements or entries. Matrices can be described by their dimensions. For example, a matrix with 2 rows and 3 columns is a 2 by 3 matrix.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

More generally, a matrix with m rows and n columns is referred to as an $m \times n$ matrix or simply an m-by-n matrix, and m and n are its dimensions.

The general form of a 2×3 matrix A is:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}$$

4.1 Types of Matrices

Matrices can be described by their shape:

- **Row Matrix:** has only one row.
- **Column Matrix:** has only one column.
- **Square Matrix:** has the same number of rows and columns.
- **Rectangular Matrix:** has an unequal number of rows and columns.

They can also be described by their unique numerical properties. Special matrices tht come in handy for certains types of computations. These are a few of the most common special matrices.

- **Zero Matrix:** contains only entries that are zero.
- **Identity Matrix:** sometimes called the unit matrix, is a square matrix whose diagonal entries are 1 and all other entries are 0.
- **Symmetric Matrix:** a square matrix that equals its transpose.
- **Diagonal Matrix:** has nonzero elements on the main diagonal, but all other elements are zero
- **Triangular Matrix:** This is a special square matrix that can be upper triangular or lower triangular. If upper, the main diagonal and all entries above it are nonzero while the lower entries are all zero. If lower, the main diagonal and all the entries below it are nonzero while the upper entries are all zero.

4.1.1 Symmetric Matrices

If you want to find out if a square matrix is symmetric, you need to transpose it. If the transpose is equal to the original matrix, then the matrix is symmetric.

To transpose a matrix, flip it over its diagonal so that the rows and columns are switched, like this:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

After transposing:

$$A^T = \begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{bmatrix}$$

Note that A^T means the transpose of A.

Let's see how this works for the following square matrix, A.

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix}$$

Switch the rows and columns to get the transpose:

$$A^T = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix}$$

Notice that $A = A^T$, so the matrix is symmetric.

What about matrix B?

$$B = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 7 & 8 & 9 \end{bmatrix}$$

Switch the rows and columns to get the transpose:

$$B^T = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

Note that $B \neq B^T$. So B is not symmetrical.

Exercise 5 Matrix Transposition

Find the transpose of this matrix. Is it symmetric?

$$A = \begin{bmatrix} 3 & -2 & 4 \\ -2 & 6 & 2 \\ 4 & 2 & 3 \end{bmatrix}$$

Working Space

Answer on Page 29

4.1.2 Creating Matrices in Python

Create a file called `matrices_creation.py` and enter this code:

```
// import the python module that supports matrices
import numpy as np
// Use the function np.array to define a matrix that
// contains specific values that you supply.
A = np.array([[ 5, 1 ,3],
              [ 1, -1 ,8],
              [ 6, 2 ,1]])
// The transpose function returns
A.transpose()
```

When you run it, you should see:

```
array([[ 5, 1 ,6],
       [ 1, -1 ,2],
       [ 3, 8 ,1]])
```

As you can see, $A \neq A^T$ so A is not symmetric. Try another:

```
// create a matrix, B
B = np.array([[ 5, 1 ,6],
              [ 1, -1 ,2],
              [ 6, 2 ,1]])
B.transpose()
```

When you run it, you should see:

```
array([[ 5, 1, 6],
       [ 1, -1, 2],
       [ 6, 2, 1]])
```

B is symmetric. You can actually transpose any matrix using this function. But a matrix cannot be symmetric unless it is square.

Try this code to see what happens when you transpose a rectangular matrix.

```
// create a matrix, B
J = np.array([[ 5, 1 ,3, 0],
              [ 1, -1 ,8, 11],
              [ 6, 2 ,1,-7]])
J.transpose()
```

Note that transposing a rectangular matrix changes its dimension from 3 by 4 to 4 by 3. You should see a transposed matrix, but it's not symmetric.

```
array([[ 5,  1,  6],
       [ 1, -1,  2],
       [ 3,  8,  1],
       [ 0, 11, -7]])
```

4.1.3 Creating Special Matrices in Python

Use the same file to add this code for creating a zero matrix.

```
 // create an 8 by 10 Zero matrix.
 C = np.zeros((8, 10))
 C
```

When you run it, you should see:

```
array([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]])
```

Add the following code to create an 8 by 8 Identity matrix.

```
 // create an 8 by 8 Identity matrix
 D = np.eye(8)
 D
```

When you run it, you should see:

```
array([[1., 0., 0., 0., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0., 0., 0., 0.],
       [0., 0., 1., 0., 0., 0., 0., 0.],
       [0., 0., 0., 1., 0., 0., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0.],
       [0., 0., 0., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 0., 0., 0., 1.]])
```

Appendix A

Answers to Exercises

Answer to Exercise 1 (on page 6)

$$Av = (11, 37, -43)$$

Answer to Exercise 2 (on page 10)

$$\mathbf{w} = 2 * [2, 4, 8] + (-2) * [8, -6, 3] + 4 * [7, 9, 2]$$
$$\mathbf{w} = [4, 8, 16] + [-16, 12, -6] + [28, 36, 8]$$
$$\mathbf{w} = [16, 56, 18]$$

Answer to Exercise 3 (on page 12)

$$\begin{aligned}\textbf{TotalSales} &= 50 * 300 + 75 * 100 + 150 * 50 = 30,000\\ \textbf{NumberTickets} &= 300 + 100 + 50 = 450\\ \textbf{WeightedAverage} &= 30,000/450 = 66.67\end{aligned}$$

Answer to Exercise 4 (on page 16)

Rewrite as a system of equations:

$$\begin{gathered}2 * a_1 + 2 * a_2 + 0 * a_3 = 0\\ 1 * a_1 - 1 * a_2 + 1 * a_3 = 0\\ 4 * a_1 + 2 * a_2 - 2 * a_3 = 0\end{gathered}$$

Simplify

$$\begin{gathered}2a_1 + 2 * a_2 = 0\\ a_1 - a_2 + a_3 = 0\\ 4a_1 + 2a_2 - 2a_3 = 0\end{gathered}$$

Swap row 2 and 1:

$$\begin{gathered}a_1 - a_2 + a_3 = 0\\ 2a_1 + 2 * a_2 = 0\\ 4a_1 + 2a_2 - 2a_3 = 0\end{gathered}$$

Multiply row 1 by -2 and add to row 2:

$$\begin{gathered}a_1 - a_2 + a_3 = 0\\ 0 + 3 * a_2 - 2a_3 = 0\\ 4a_1 + 2a_2 - 2a_3 = 0\end{gathered}$$

Multiply row 1 by -4 and add to row 3:

$$\begin{gathered}a_1 - a_2 + a_3 = 0\\ 0 + 3 * a_2 - 2a_3 = 0\\ 0 + 6a_2 - 6a_3 = 0\end{gathered}$$

Multiply row 2 by -4 and add to row 3:

$$\begin{gathered}a_1 - a_2 + a_3 = 0\\ 0 + 3 * a_2 - 2a_3 = 0\\ 0 + 0 - 2a_3 = 0\end{gathered}$$

Multiply row 3 by -1 and add to row 2:

$$\begin{gathered}a_1 - a_2 + a_3 = 0\\ 0 + 3 * a_2 + 0 = 0\\ 0 + 0 - 2a_3 = 0\end{gathered}$$

Divide row 3 by -2 and row 2 by $\frac{1}{3}$:

$$\begin{gathered} a_1 - a_2 + a_3 = 0 \\ 0 + a_2 + 0 = 0 \\ 0 + 0 + a_3 = 0 \end{gathered}$$

Backsubstitute a_2 and a_3 into row 1:

$$\begin{gathered} a_1 + 0 + 0 = 0 \\ 0 + a_2 + 0 = 0 \\ 0 + 0a_3 = 0 \end{gathered}$$

Therefore

$$a_1 = a_2 = a_3 = 0$$

.

Answer to Exercise 5 (on page 23)

$$A = A^t = \begin{bmatrix} 3 & -2 & 4 \\ -2 & 6 & 2 \\ 4 & 2 & 3 \end{bmatrix}$$

Index