College of Engineering & Applied Sciences

CSPB 2820

Linear Algebra With Computer Science Applications

Study Guide 1 - Vectors

Taylor Larrechea

2024

Sections

Study Guide 1

Vectors

Study Guide Instructions

- Submit your work in Gradescope as a PDF - you will identify where your "questions are."
- Identify the question number as you submit. Since we grade "blind" if the questions are NOT identified, the work WILL NOT BE GRADED and a 0 will be recorded. Always leave enough time to identify the questions when submitting.
- One section per page (if a page or less) - We prefer to grade the main solution in a single page, extra work can be included on the following page.
- Long instructions may be removed to fit on a single page.
- **Do not start a new question in the middle of a page.**
- Solutions to book questions are provided for reference.
- You may NOT submit given solutions - this includes minor modifications - as your own.
- Solutions that do not show individual engagement with the solutions will be marked as no credit and can be considered a violation of honor code.
- If you use the given solutions you must reference or explain how you used them, in particular...

Method Selection

For full credit, EACH book exercise in the Study Guides must use one or more of the following methods and FOR EACH QUESTION. Identify the number the method by number to ensure full credit.

- **Method 1** - Provide original examples which demonstrate the ideas of the exercise in addition to your solution.
- **Method 2** - Include and discuss the specific topics needed from the chapter and how they relate to the question.
- **Method 3** - Include original Python code, of reasonable length (as screenshot r text) to show how the topic or concept was explored.
- **Method 4** - Expand the given solution in a significant way, with additional steps and comments. All steps are justified. This is a good method for a proof for which you are only given a basic outline.
- **Method 5** - Attempt the exercise without looking at the solution and then the solution is used to check work. Words are used to describe the results.
- **Method 6** - Provide an analysis of the strategies used to understand the exercise, describing in detail what was challenging, who helped you or what resources were used. The process of understanding is described.

Problem 1

Problem Description

Reading the book carefully is essential in this class and in all advanced mathematics. This is an exercise in annotation.

For this first exercise, pick a section or page from Chapter one of VMLS.

Read straight through the section once for an overview. Include the following 3 items for #1.

(a) Write down your initial thoughts, questions, and first impressions ('whaaat???')

(b) Now, return to the section and slowly work through each line using a pencil or pen.

- Expand equations.
- Identify key concepts and explain in your own words.
- Make note - is that a vector or a scalar?
- Fill in missing ideas or steps.
- Include a screenshot of your annotation.

(c) How has your understanding of the section changed?

Solution - Part (a)

For this problem, I am going to utilize **Method 2**.

For this part of the problem, I will be reviewing **VMLS Page 19**, particularly the part about the inner product definition.

To begin, this page in the textbook gives a definition of what a inner product (dot or scalar product) is. This operation takes two vectors of equal length and 'multiplies' them together to get a scalar value.

Another way that an inner product is defined is with the use of the vectors magnitude and angle in between the vectors. If we let A and B represent the magnitudes of vectors a and b respectively and θ represent the angle in between these vectors, the dot product of two vectors can be defined as

$$a \cdot b = AB\cos(\theta). \tag{1}$$

My initial thoughts after reading this section was that this section skipped a lot of math in its definitions (as expected when talking about mathematical text books) and it could be very confusing to readers who do not have a more in depth mathematical background. With this being said, I did not know about some of the properties that were mentioned in this section. I also think this section of the textbook does not give an in depth explanation of what the transpose of a matrix is (vectors are indeed matrices) and that can also be confusing as to what that means.

Solution - Part (b)

For this problem, I am going to utilize **Method 2**.

I first want to begin with a more clear definition of what the transpose of a matrix is. In short, the transpose of a matrix takes the rows of matrix, beginning from the first row and takes the elements in those rows and puts them in the column of a matrix. The easiest example of this can be seen with an arbitrary 2×2 matrix. Lets say we have a matrix A, the transpose of this matrix (denoted as A^T) is then

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \implies A^T = \begin{bmatrix} a & c \\ b & d \end{bmatrix}. \tag{2}$$

Because of the simple nature of a 2×2 matrix, it may just seem that we are switching diagonal elements in a matrix. That is not the case when we are working with larger matrices.

If we now expand the definition of the transpose of a matrix in equation (2) to a vector, we can see that a

vector of length three's transpose (the vector is denoted as a) is then

$$a = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \implies a^T = \begin{bmatrix} a & b & c \end{bmatrix}. \tag{3}$$

The nature of which matrix multiplication occurs is that of 'row times column'. The necessity for why we have to transpose a vector in order to take the inner product of said vector with another vector is that one vector must be a row vector and the other must be a column vector.

If we take a further look at matrix multiplication, it can possibly help us understand the inner product a little bit more. Say we have two matrices (A) and (B), in general these matrices are

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad B = \begin{bmatrix} e & f \\ g & h \end{bmatrix}. \tag{4}$$

If we now multiply the matrices found in equation (4), we will have

$$A \times B = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} a(e) + b(g) & a(f) + b(h) \\ c(e) + d(g) & c(f) + d(h) \end{bmatrix}. \tag{5}$$

Now, we can observe that each element in the resulting matrix is indeed a scalar value. In the context of vectors, we do not get a vector (or matrix) as the resulting value that comes from an inner product, we get a scalar value. This is because when we take an inner product of two vectors, we are performing matrix multiplication of a $(1 \times n)$ matrix with that of a $(n \times 1)$ matrix.

This means, a better way to define the inner product of two vectors would then look something like

$$a = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \implies a \cdot b = a^T b = \begin{bmatrix} a_1 & a_2 & \dots & a_n \end{bmatrix} \times \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n. \tag{6}$$

Equation (6) in my eyes gives a more in depth explanation of the inner product and how it is calculated. Filling in these gaps for how to carry out these calculations can be very beneficial to understanding more general or complicated examples of the same topic.

Now I would like to discuss the properties of inner products. The first property that we encounter is the ***commutativity*** property. We can do a pseudo proof of this property

$$a = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \implies a \cdot b = a^T b = \begin{bmatrix} a_1 & a_2 \end{bmatrix} \times \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = a_1(b_1) + a_2(b_2), \tag{7}$$

where now we do the reverse

$$b \cdot a = b^T a = \begin{bmatrix} b_1 & b_2 \end{bmatrix} \times \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = b_1(a_1) + b_2(a_2). \tag{8}$$

Because of the commutativity property of real numbers we can say $a_1(b_1) = b_1(a_1)$ and vice versa to further say

$$a_1(b_1) + a_2(b_2) = b_1(a_1) + b_2(a_2) \implies a^T b = b^T a. \tag{9}$$

We now move on to ***associativity with scalar multiplication***. Starting with the same vectors defined in equation (7) we can do another pseudo proof of this property

$$(\gamma a^T)b = \gamma \begin{bmatrix} a_1 & a_2 \end{bmatrix} \times \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} \gamma(a_1) & \gamma(a_2) \end{bmatrix} \times \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \gamma(a_1)b_1 + \gamma(a_2)b_2 \tag{10}$$

where we can then do

$$\gamma(a^T b) = \gamma\left(\begin{bmatrix} a_1 & a_2 \end{bmatrix} \times \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \right) = \gamma(a_1(b_1) + a_2(b_2)) = \gamma a_1(b_1) + \gamma a_2(b_2) \implies (\gamma a^T)b = \gamma(a^T b). \tag{11}$$

Equations (10) and (11) are thus equivalent as their only differences lie in the formatting of the multiplication of the elements.

We now move on to our last property shown on this page, the ***distributivity with vector addition*** property. We first start with three vectors

$$a = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \quad c = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}. \tag{12}$$

We then first show

$$(a+b)^T c = \left(\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \right)^T \times \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \left(\begin{bmatrix} a_1 + b_1 \\ a_2 + b_2 \end{bmatrix} \right)^T \times \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} a_1 + b_1 & a_2 + b_2 \end{bmatrix} \times \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \tag{13}$$

$$= (a_1 + b_1)c_1 + (a_2 + b_2)c_2 = a_1(c_1) + b_1(c_1) + a_2(c_2) + b_2(c_2). \tag{14}$$

We can now go on to show that the result from equation (14) is equal to

$$a^T c + b^T c = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}^T \times \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}^T \times \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \end{bmatrix} \times \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + \begin{bmatrix} b_1 & b_2 \end{bmatrix} \times \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \tag{15}$$

$$= a_1(c_1) + a_2(c_2) + b_1(c_1) + b_2(c_2). \tag{16}$$

Using the commutativity property of addition we can then say

$$(a+b)^T c = a^T c + b^T c \tag{17}$$

and thus the property has been proved.

Solution - Part (c)

Before reading this section, I did not know of the properties that pertained to the inner product. Such as the commutativity property. After reading this chapter I gained a lot of insight into these other properties and how they can be applied to vectors. From my prior experience to linear algebra, I had a good understanding of the inner product but I didn't really know of these properties all too well. These properties can be utilized in certain scenarios to make problems easier and it has provided me some more tools that I can possibly use in the future.

Problem 1 Summary

Procedure

- This problem encapsulates summarizing a section / page of a textbook and highlighting important aspects from the chapter

Key Concepts

- Important aspects from this section are what an inner product is and how a transpose works on a matrix
- Inner products are commutative, associative, and distributive

Variations

- This particular example doesn't have much variations since it is annotating a chapter from the textbook

Problem 2

Problem Description

Solve the Random exercise from the video and Piazza in your own words here.

Symptoms vector. A 20-vector s records whether each of 20 different symptoms is present in a medical patient, with $s_i = 1$ meaning the patient has the symptom and $s_i = 0$ meaning she does not. Express the following using vector notation.

Original Questions

(a) The total number of symptoms the patient has.

(b) The patient exhibits five out of the first ten symptoms.

Solution - Part (a)

For this solution, I a going utilize **Method 5** for this problem.

The vector S encapsulates the symptoms that a patient is showing. Inside this vector, a given index is denoted with a 1 if they have the symptom and a 0 if they do not have the symptom. To accurately count the number of symptoms that a patient has, we must take the inner product of the symptoms vector S with the the unit vector $\mathbf{1}$. Let σ denote the total number of symptoms that a patient has. Mathematically this will look like

$$\sigma = \sum_i S_i \mathbf{1}_i = S^T \mathbf{1}. \tag{1}$$

The solution found in one can be explained further by looking at this simple example. For simplicity, I will simplify our vector down to only length 3.

$$\sigma = \sum_i S_i \mathbf{1}_i = S^T \mathbf{1} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}^T \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = 1(1) + 0(1) + 1(1) = 1 + 0 + 1 = 2. \tag{2}$$

In the case of equation (2), we see that there are a max of 3 possible symptoms and the current patient only has 2 symptoms. The unit vector $\mathbf{1}$ helps accurately count the number of symptoms that a patient possesses because when the inner product of the two matrices are taken, if a patient is not showing a symptom then the multiplication between the symptom and unit vector's for the given index that represents that symptom will be zero. In the example of (2) above, we see that the patient is only showing signs for symptom 1 and 3. This in turn means when the calculation is carried out the patient will only come back with showing 2 symptoms.

For brevity's sake, I simplified the problem as shown in equation (2) to be that of only three possible symptoms. This specific scenario can be generalized to meet any arbitrary number of symptoms that a patient may have. In our case, the patient may show up to 20 possible symptoms. The general case for the number of possible symptoms can be seen in equation (1) above.

Solution - Part (b)

For this solution, I a going utilize **Method 5** for this problem.

If a patient is showing five out of the first ten symptoms, this means the values for S in the range of indices 1 to 10 must either be zero or one. But the total number of symptoms (1's) that show up in the first ten possible symptoms ($S_{1:10}$) must add up to be 5. In general, the number of possible combinations for these can be calculated. In our case, for the patient is showing 5 out of the first 10 symptoms. The total number of combinations is

$$C(10,5) = \binom{10}{5} = \frac{10!}{5!(10-5)!} = \frac{10 \cdot 9 \cdot 8 \cdot 7 \cdot 6}{5 \cdot 4 \cdot 3 \cdot 2 \cdot 1} = 252. \tag{3}$$

Since I would be spending the rest of the semester creating all of these possible combinations, we can generalize this scenario. Let $\bar{S}$ represent one of the 252 possible combinations of a patient showing 5 out of the first 10 symptoms where the values of $S_{11:20}$ are zero. This in turn means that we can generalize equation (1) to now be

$$\sigma = \sum_i \bar{S}_i \mathbf{1_i} = \bar{S}^T \mathbf{1} = 5 \tag{4}$$

Further more, if we generalize equation (4) to encapsulate all 252 combinations, we now have

$$\sum_j \sum_i \bar{S}_{j_i} \mathbf{1} = \sum_j \bar{S}_j^T \mathbf{1}. \tag{5}$$

Here, j represents one of the 252 combinations for how a patient can have 5 out of the first 10 symptoms and i represents the index of that given index of that particular vector j. The resulting value of equation (4) will always be 5.

Comparison To Solution

The answer that I arrived to for part (a) of this problem is identical to that of the one in the solution manual and the one that is in the video on Moodle.

For the answer to part (b), I feel like the solution manual is missing the fact that there are a number of potential possibilities that can represent the vector a that they have in the solution manual. I feel like my solution to this problem is more general in that I have taken into account the number of possibilities there are for a patient showing 5 of the first 10 symptoms that are possible. In comparing my solution to that of the solution manual, we had the same answer for the other 10 symptoms ($S_{11:20}$) in that they were all zero.

Problem 2 Summary

Procedure

(a) Part (a)

- Create a variable that represents the number of symptoms that a patient is currently showing with the use of the **1** vector and a boolean vector that represents the symptom(s) that a current patient is exhibiting
- Take the inner product between these two vectors to count the number of symptoms that someone is currently showing
- Because S is a boolean vector, any symptom that a patient is not showing will have a 0 for that element

(b) Part (b)

- Because there are 252 possibilities for this answer, we represent all possibilities for a patient showing 5 out of the first 10 symptoms with

$$\sum_j \sum_i \bar{S}_{ji}\mathbf{1} = \sum_j \bar{S}_j^T\mathbf{1}.$$

This expression will sum over each possibility of a patient showing 5 out of the first 10 symptoms.

Key Concepts

(a) Part (a)

- This part of the problem is showing how a scalar value can be constructed with the use of two vectors
- We use a boolean vector to resemble if someone is showing a symptom and the **1** vector to represent all of the 20 symptoms that a patient can exhibit
- The inner product between these two vectors will resemble the number of symptoms that a patient is showing since inner products produce scalar values

(b) Part (b)

- This part of the problem is trying to find a vector representation of a patient exhibiting 5 out of the first 10 symptoms
- To achieve this, we have to sum over all the possibilities of a patient exhibiting 5 out of the first 10 symptoms
- To represent this mathematically, we use a double sum to show all possibilities and take the inner product of that one possibility with the **1** vector

Variations

- For this problem to retain its core structure, we could change the number of symptoms that a patient is possibly showing
  - This wouldn't change the procedure for how to answer the question, it would just simply change the size of the vectors
- For part (b), we could potentially change the number of symptoms that a patient would show out of the total number of first symptoms
  - This again wouldn't change how we would answer the problem since the answer provided is a generalized version of this problem

Problem 3

Problem Description

Explain the solution to 1.8 here in your own words. (Since you are given a solution, you will be graded on your ability to explain).

Original Question: Profit and sales vectors. A company sells n different products or items. The n-vector p gives the profit, in dollars per unit, for each of the n items. (The entries of p are typically positive, but a few items might have negative entries. These items are called loss leaders, and are used to increase customer engagement in the hope that the customer will make other, profitable purchases.) The n-vector s gives the total sales of each of the items, over some period (such as a month), i.e., s_i is the total number of units of item i sold. (These are also typically nonnegative, but negative entries can be used to reflect items that were purchased in a previous time period and returned in this one.) Express the total profit in terms of p and s using vector notation.

Solution

For this solution, I am going to utilize **Method 4** for this problem.

Original Solution: The profit for item i is $p_i s_i$, so the total profit is

$$\sum_i p_i s_i = p^T s. \tag{1}$$

In other words, the total profit is just the inner product of p and s.

My Explanation: Intuitively, the total profit in this case is going to be the sum of the profits (or loss leaders) for each individual item that is being sold. In order to do this, we have to know the total number of times that item was sold. In our case, the profit (or loss leader) for a given item is represented as p and the total number of times that corresponding item was sold is represented by s. So, if we had one item in our shop we could simply calculate the profit as

$$\text{Profit} = p \cdot s. \tag{2}$$

But, since we have n number of products, we denote each product with a corresponding subscript i. This means the total profit of the store can be calculated with

$$\text{Profit} = \sum_i p_i s_i. \tag{3}$$

In the context of linear algebra, we can depict the number of total products we have with a vector p and the total number of times that object was sold with another vector s. These vectors must be of the same length. After they have been constructed, to achieve the result of (3) we must execute an inner product of these vectors to get a scalar value at the end of the day that represents the total profit of the shop. This mathematically looks like

$$\sum_i p_i s_i = p^T s. \tag{4}$$

Equation (4) adequately expresses how one would solve this problem in the context of linear algebra.

Problem 3 Summary

Procedure

- To represent the profit of this company we have to use an inner product of two vectors
- The first vector p represents the profit or loss for a given item
- The second vector s represents the number of times this loss or profit was achieved for a given item

Key Concepts

- Inner products represent a scalar value and can be used to represent quantities like the total profit of a company and the products that they sell

Variations

- We could be asked to construct an inner product for a different value
  - We would then construct two vectors in a similar manner that could be used in an inner product to represent this new quantity

Problem 4

Problem Description

Explain the solution to 1.16 here in your own words. (Since you are given a solution, you will be graded on your ability to explain).

Original Questions:

(a) Explain why the inner product of two nonnegative vectors is nonnegative.

(b) Suppose the inner product of two nonnegative vectors is zero. What can you say about them? Your answer should be in terms of their respective sparsity patterns, i.e., which entries are zero and nonzero.

Solution - Part (a)

For this solution, I am going to utilize **Method 4** for this problem.

Original Solution: Let x and y be nonnegative n-vectors. The inner product

$$x^T y = x_1 y_1 + x_2 y_2 + ... + x_n y_n \tag{1}$$

is nonnegative because each term in the sum is nonnegative.

My Explanation: The inner product of two vectors is the sum of the multiplication of the elements between the two vectors. The elements that are multiplied together correspond to the indices of the given vectors. It is a given fact that for two nonnegative integers, the multiplication of those integers is also nonnegative. Mathematically this means

$$\text{When } a \geq 0 \text{ and } b \geq 0 \implies ab \geq 0. \tag{2}$$

The sum of positive integers will always be positive. For a given set of n number of nonnegative integers, say a_i, we can extrapolate to further say

$$\sum_{i}^{n} a_i \geq 0. \tag{3}$$

In the case of an inner product of two vectors, the corresponding scalar value is a sum of elements being multiplied together. If the vectors are nonnegative, this means the inner product will then be a sum of nonnegative values. This in turn means that when two vectors x and y are both nonnegative, their inner product will always be

$$x^T y \geq 0. \tag{4}$$

Solution - Part (b)

For this solution, I am going to utilize **Method 4** for this problem.

Original Solution: For each k, $x_k = 0$ or $y_k = 0$ (or both). Therefore only the following three combinations of zero-positive patterns are positive (here + stands for a positive entry):

x_k	y_k
$+$	0
0	$+$
0	0

My Explanation: For the inner product of two nonnegative vectors to be zero, the corresponding elements of the two vectors must have the following relationship.

- If the element in one vector is positive, then the corresponding element of the other vector must be zero.
- If the element in one vector is zero, then the corresponding element of the other vector can be either positive or zero.

The above observation can be applied for any and all indices of two vectors. In short, if one element is zero, the other element can either be positive or zero. If one element is positive, then the corresponding element at the same index of the other vector must be zero.

Problem 4 Summary

Procedure

(a) Part (a)

- Since inner products are the sum of the elements of two vectors being multiplied together, if the elements are nonnegative, then the total inner product must be nonnegative
- Show this by showing that the multiplication of two nonnegative integers will always be nonnegative

(b) Part (b)

- Sow the three possibilities for how the multiplication of two values can be zero (seen in the table of values for a given index of the vectors)

Key Concepts

(a) Part (a)

- The inner product of two nonnegative vectors will always be nonnegative

(b) Part (b)

- The sparsity of the two elements in corresponding vectors must be either positive or zero or both elements are zero

Variations

- We could be asked to show different properties of two vectors
  - We would show this by using a similar procedure of determining what the values of each multiplication of elements must be

Problem 5

Problem Description

Create a Jupyter Notebook of the following:

The Python Companion(linked at the top of the course) includes examples of code that can help you throughout the course and provides the basis for the fundamentals of vector operations we will use throughout the course.

Every week you can use these code elements to illustrate concepts and ideas from the weeks.

This week we guide you through an example.

1. Find the Python Companion and sections 1.1 - 1.4.
2. Select at least 10 code blocks (you may want to do all of them) that look useful.
3. Rewrite (best) or copy and paste each code cell into your own Jupyter Notebook.
4. Add notes in your own words in a text block above about the code using the notes given AND incorporate ideas from the book (cite the book page).
5. For each section of code include at least one additional specific example of using the code.
6. **Attach a PDF of your Jupyter Notebook here** to include in the single PDF study guide.

The pdf from my Jupyter notebook can be seen on the next page.

Vectors Jupyter Notebook

Code Block 1 - For the first code block I am defining a vector with the use of numpy. This original code block can be found on page 2 of the python companion and the theory of this can be found on page 3 of VMLS. The original code block can be seen below.

In [21]:
```
import numpy as np
import random
x = [-1.1, 0.0, 3.6, -7.2]
len(x)
```

Out[21]:
```
4
```

For my example, I am generating a random integer value between 1 and 100 for each element of the vector and then printing the vector itself along with the length of the vector.

In [22]:
```
import numpy as np
import random
# Vector definition
cool_vector = np.array([random.randint(1, 100), random.randint(1, 100), random.
# Print statements
print("Random Cool Vector:", cool_vector)
print("Random Cool Vector Length:", len(cool_vector))
```

```
Random Cool Vector: [39 61 16]
Random Cool Vector Length: 3
```

Code Block 2 - For the second code block I am copying a vector with the use of the "copy()" method. Once the vector has been copied, we re-assign an element with indexing one of the elements and assign it to a new value. This original code block can be found on page 4 of the python companion. The theory of how to index a vector can be found on page 5 of VMLS. The original code block can be found below.

In [23]:
```
import numpy as np
x = np.array([-1.1, 0.0, 3.6, -7.2])
y = x.copy()
x[2] = 20.0
print(y)
```

```
[-1.1  0.   3.6 -7.2]
```

For my rendition of this concept, I first create a vector and then copy it to another vector. I then re-assign a random element from the copied vector with the use of "random" and then assign it a random integer that ranges between 7 and 10.

In [24]:
```
import numpy as np
import random
# Vector definition
another_cool_vector = np.array([1, 3, 5, 6])
# Copy the vector
copy_of_another_cool_vector = another_cool_vector.copy()
copy_of_another_cool_vector[random.randint(0, len(copy_of_another_cool_vector)
# Print the original and the copied vector that has been modified
```

```
print("Original Cool Vector:", another_cool_vector)
print("Copy Of Cool Vector:", copy_of_another_cool_vector)
```

```
Original Cool Vector: [1 3 5 6]
Copy Of Cool Vector: [ 1  3  5 10]
```

Code Block 3 - For the third code block, I am looking into concatenation of two vectors. The way that this is done in python can be found in python companion on page 6 and the theory of stacked vectors can be found on page 4 on VMLS. The original code block can be found below.

In [25]:
```
import numpy as np
x = np.array([1, -2])
y = np.array([1,1,0])
z = np.concatenate((x,y))
print(z)
```

```
[ 1 -2  1  1  0]
```

For my rendition of this, I first create two vectors. The first vector contains integer values from 1 to 4 and the second is an addition of four to each element of the original vector. I then concatenate the two vectors and print all of the vectors.

In [26]:
```
import numpy as np
# Vector definition
x = np.array([1, 2, 3, 4])
y = np.array([x[0] + 4, x[1] + 4, x[2] + 4, x[3] + 4])
# Concatenate vector
z = np.concatenate((x,y))
# Print the vectors
print("Vector X:", x)
print("Vector Y:", y)
print("Concatenated Vector, Z:", z)
```

```
Vector X: [1 2 3 4]
Vector Y: [5 6 7 8]
Concatenated Vector, Z: [1 2 3 4 5 6 7 8]
```

Code Block 4 - For the fourth code block we are examining the addition of two vectors. The original code block can be found in the python companion on page 10. The theory for this can be found on page 11 of VMLS. The original code block can be found below.

In [27]:
```
import numpy as np
x = np.array([1,2,3])
y = np.array([100,200,300])
print('Sum of arrays:', x+y)
print('Difference of arrays:', x-y)
```

```
Sum of arrays: [101 202 303]
Difference of arrays: [ -99 -198 -297]
```

My rendition of this example involves two vectors. The first vector is defined with set values, the second vector then has its elements assigned with random values that are predicated on the original vector. For example, for the first element of b, we take the first element of a as the minimum in the random calculation and then the maximum is calculated by taking a

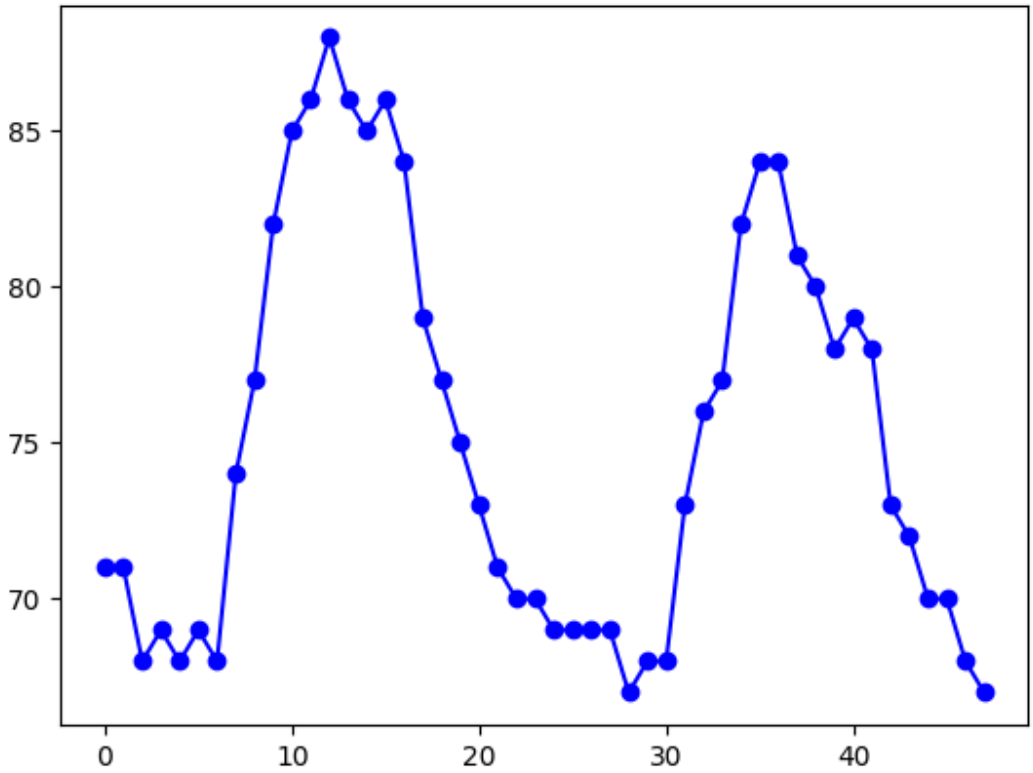
multiple of the second element from a and then multiplying it with the first element of a. This patten is repeated for all of the succeeding values.

In [28]:
```
import numpy as np
import random
# Vector definition
a = np.array([2, 4, 6, 8])
b = np.array([random.randint(a[0], a[1] * a[0]), random.randint(a[1], a[0] * a[
# Sum and difference
print("Sum of a and b:", a + b)
print("Difference of arrays:", a - b)
```

```
Sum of a and b: [ 9  8 15 22]
Difference of arrays: [-5  0 -3 -6]
```

Code Block 5 - For the fifth code block, I am taking the plot example from pages 9 and 10 of the python companion. The theory of this can be found on page 15 of VMLS. The original code block can be seen below.

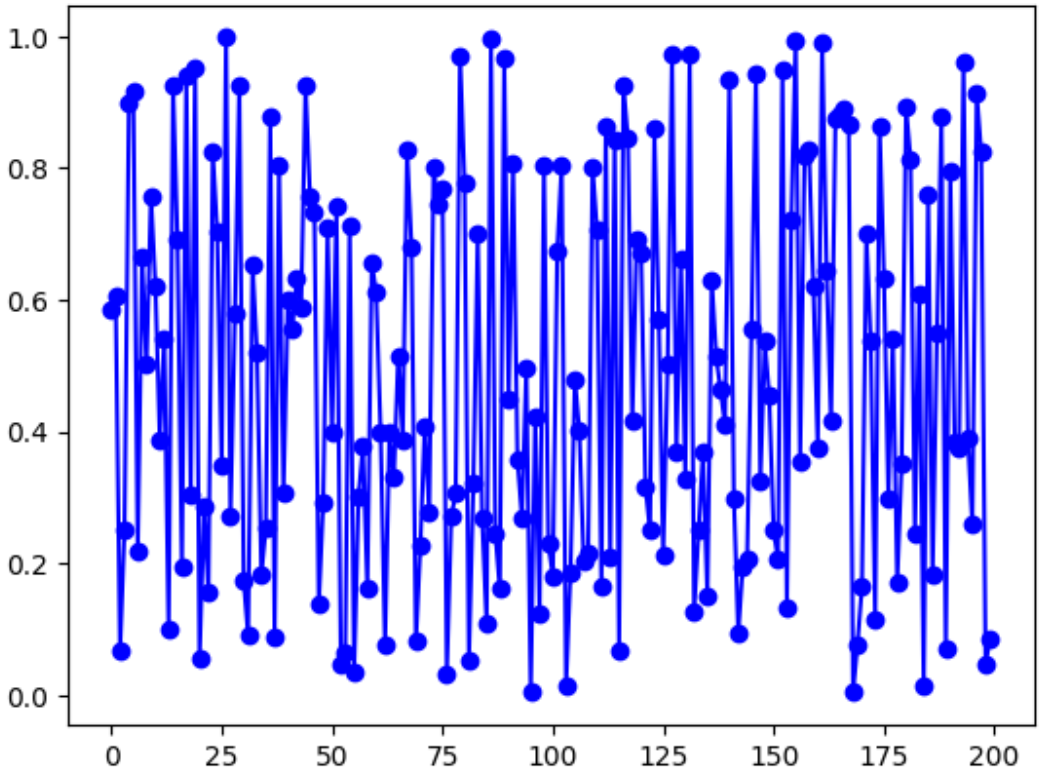
In [29]:
```
import matplotlib.pyplot as plt
plt.ion()
temps = [ 71, 71, 68, 69, 68, 69, 68, 74, 77, 82, 85, 86, 88, 86,
85, 86, 84, 79, 77, 75, 73, 71, 70, 70, 69, 69, 69, 69, 67,
68, 68, 73, 76, 77, 82, 84, 84, 81, 80, 78, 79, 78, 73, 72,
70, 70, 68, 67 ]
plt.plot(temps, '-bo')
plt.savefig('temperature.pdf', format = 'pdf')
plt.show()
```



Now, for my rendition I am going to plot the values of two vectors that have been added together. These vectors are 100 elements in length and the elements in them are random.

This code block uses the "append" method found later on the python companion.

In [30]:
```
import numpy as np
import random
import matplotlib.pyplot as plt
# Vector definitions
a = []
b = []
# Assign random values to elements in vector
for i in range(100):
    a.append(random.random())
    b.append(random.random())
# Add vectors
c = a + b
# Plot
plt.plot(c, '-bo')
plt.show()
```



Code Block 6 - For this code block, I am going to take a look at scalar-vector multiplication. The original code block can be found on page 11 of the python companion. The theory of this operation can be found on page 15 of VMLS. The original code block can be found below.

In [31]:
```
import numpy as np
x = np.array([1,2,3])
print(2.2*x)
```

```
[2.2 4.4 6.6]
```

For my rendition, I am going to perform scalar-vector multiplication based off of the vector size.

In [32]:
```
import numpy as np
x = np.array([1, 2, 4, 5, 6, 7])
y = len(x) * x
print("Original Vector:", x)
print("Scalar Multiplied Vector:", y)
```

```
Original Vector: [1 2 4 5 6 7]
Scalar Multiplied Vector: [ 6 12 24 30 36 42]
```

Code Block 7 - For the seventh code block, I am going to dive into linear combination. The original code block can be found on page 12 of the python companion and the theory of this can be found on page 17 of VMLS. The original code block can be found below.

In [33]:
```
import numpy as np
a = np.array([1,2])
b = np.array([3,4])
alpha = -0.5
beta = 1.5
c = alpha*a + beta*b
print(c)
```

```
[4. 5.]
```

For my rendition, I am generating two random vectors that are comprised of 10 elements. These vectors are then scaled off of the middle element of the vector and then combined into a third vector.

In [34]:
```
import numpy as np
import random
# Vector definition
a = []
b = []
# Append random values
for i in range(10):
    a.append(random.random())
    b.append(random.random())
# Grab middle element
alpha = a[4]
beta = b[4]
# Create numpy arrays
a = np.array(a)
b = np.array(b)
# Calculate new vector
c = alpha*a + beta*b
print("Vector a:", a)
print("Vector b:", b)
print("Vector c:", c)
print("Alpha:", alpha)
print("Beta:", beta)
```

```
Vector a: [0.85226249 0.33766032 0.39193881 0.987842   0.84522741 0.07673927
 0.47095763 0.24721789 0.86380362 0.7840817 ]
Vector b: [0.22896021 0.65161428 0.09290719 0.6868429  0.51744293 0.52351399
 0.35781774 0.90120774 0.00718673 0.72531563]
Vector c: [0.83882945 0.62257295 0.37935159 1.19035314 0.98215656 0.33575074
 0.58321656 0.67527891 0.73382921 1.03803679]
Alpha: 0.8452274084651978
Beta: 0.5174429306003135
```

Code Block 8 - This code block is based off of the inner product example that can be found on page 14 of the python companion. The theory of this can be found on page 20 of VMLS. The original code block can be found below.

In [35]:
```
import numpy as np
c = np.array([0.1,0.1,0.1,1.1]) #cash flow vector
n = len(c)
r = 0.05 #5% per-period interest rate
d = np.array([(1+r)**-i for i in range(n)])
NPV = c @ d
print(NPV)
```

```
1.236162401468524
```

The only difference from the original code block to my rendition is that the interest rate is randomly generated. The minimum value of this random interest rate is 0.01 and the maximum is 0.10. The way that this number was generated can be found at the following link: https://www.w3schools.com/python/ref_random_uniform.asp

In [36]:
```
import numpy as np
import random
c = np.array([0.1,0.1,0.1,1.1]) #cash flow vector
n = len(c)
r = random.uniform(0.01, 0.1)
d = np.array([(1+r)**-i for i in range(n)])
NPV = c @ d
print(NPV)
```

```
1.3309583328199663
```

Code 9 - This code block examines slicing in python. The original code block can be found on page 7 of the python companion. The theory of slicing can be found on page 4 of VMLS. The original code block can be found below.

In [37]:
```
import numpy as np
x = np.array([1,8,3,2,1,9,7])
d = x[1:] - x[:-1]
print(d)
```

```
[ 7 -5 -1 -1  8 -2]
```

For my rendition, I am adding the values instead of subtracting them.

In [38]:
```
import numpy as np
x = np.array([1,8,3,2,1,9,7])
d = x[1:] + x[:-1]
print(d)
```

```
[ 9 11  5  3 10 16]
```

Code Block 10 - For the last code block, we are going to look at unit vectors. The application of this can be found on page 8 of the python companion. The theory of this can be found on page 5 of VMLS. The original code block can be found below.

In [39]:
```
import numpy as np
i = 2
n = 4
x = np.zeros(n)
x[i] = 1
print(x)
```

```
[0. 0. 1. 0.]
```

For my rendition, I am going to randomly create a unit vector. e.g. The length of the vector will be an arbitrary number of elements from 5 to 10 and location of i will be randomly selected based on the size of the vector.

In [40]:
```
import numpy as np
import random
n = random.randint(5, 10)
i = random.randint(0, n - 1)
x = np.zeros(n)
x[i] = 1
print(x)
```

```
[0. 0. 0. 0. 0. 0. 0. 1. 0.]
```

Most of my examples involved random number generations. These random numbers followed the method that is found on page 9 of the python companion. Most of theory that was taken place can be found in the corresponding pages of VMLS.

Problem 5 Summary

Procedure

- For this problem we showcase different examples from VMLS with the use of a Jupyter notebook

Key Concepts

- This problem showcases many properties of inner products with the use of Jupyter notebook and python

Variations

- We could be asked to copy code from a different section of VMLS