Linear Algebra and Python Applications

Klein
carlj.klein@gmail.com

1 Acknowledgments

The majority of these notes follow closely with Krista King's Udemy course "Become a Linear Algebra Master", found here.

2 Learning Objectives

These notes are to be used in reviewing the basics of linear algebra, and providing an analogue in programming the concepts in Python. We'll try to keep this relevant to the data science pathway, but some asides may be taken for pleasure in mathematics and necessity!

1. Matrix & Vector Basics
2. Dot Products & Cross Products
3. Matrix-Vector Products

3 Matrix & Vector Basics

We'll begin with the basics of linear algebra, which are rooted in a discussion about vectors and matrices.

- Operations on a Single Matrix
- Matrix Addition & Subtraction
- Matrix Multiplication (Dot Product)
- Vector Basics
- Standard Basis Vector (SBV)
- Span
- Linear Dependence & Independence

- Linear Subspaces
- Spans as Subspaces
- Basis

3.1 Operations on a Single Matrix

For somewhat of an analogue into data science and for review's sake, we could mention Gauss-Jordan elimination (or Gaussian elimination) to solve a system of equations. Essentially a series of row operations "mimicking" algebra. We'll skip the details and any coding in this section as the solutions available with the tools we'll build throughout the sections will be much more direct and less computationally expensive.

However, we should touch on number of solutions. Systems of equations can have a few different solutions:

- single, unique solution
- no solutions
- infinitely many solutions

3.2 Matrix Addition & Subtraction

For matrix addition and subtraction, matrices must be the same size, and then the operation is performed on corresponding entries between the matrices.
For example, say matrices A and B have addition or subtraction performed between them to form a matrix C. All matrices will have the same mxn (rows by columns) dimensions.

$$A = \begin{bmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \cdots & a_{m,n} \end{bmatrix} \tag{1}$$

$$B = \begin{bmatrix} b_{1,1} & \cdots & b_{1,n} \\ \vdots & \ddots & \vdots \\ b_{m,1} & \cdots & b_{m,n} \end{bmatrix} \tag{2}$$

$$\begin{bmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \cdots & a_{m,n} \end{bmatrix} \pm \begin{bmatrix} b_{1,1} & \cdots & b_{1,n} \\ \vdots & \ddots & \vdots \\ b_{m,1} & \cdots & b_{m,n} \end{bmatrix} = \begin{bmatrix} c_{1,1} & \cdots & c_{1,n} \\ \vdots & \ddots & \vdots \\ c_{m,1} & \cdots & c_{m,n} \end{bmatrix} \tag{3}$$

In other words...

$$A \pm B = C \Longrightarrow a_{m,n} \pm b_{m,n} = c_{m,n} \, \forall m, n \tag{4}$$

For python, NumPy arrays will add correspondingly (element-wise), as is shown above. However, don't make the mistake of thinking lists, or lists of lists, will perform in this manner.

The following are always properties for Addition, but not always true for Subtraction:

- Commutative: A + B = B + A
- Associative: (A + B) + C = A + (B + C)

3.3 Matrix Multiplication (Dot Product)

Before diving into the dot product, we'll quickly mention scalar multiplication. Suppose we have a constant c and a Matrix A, then by performing scalar multiplication of A by c will multiply each element of A by c:

$$cA = \begin{bmatrix} c * a_{1,1} & \cdots & c * a_{1,n} \\ \vdots & \ddots & \vdots \\ c * a_{m,1} & \cdots & c * a_{m,n} \end{bmatrix} \tag{5}$$

$$cA \Longrightarrow c * a_{m,n} \; \forall m, n \tag{6}$$

Now to explain multiplying a matrix by another matrix. Dimensions and order of the matrices come into play here.

- Dimensions: the number of columns of the first matrix must be equal to the number of rows of the second matrix. The dimensions of the resulting matrix will be the number of rows of the first matrix by the number of columns of the second matrix.
- Order: A · B ≠ B · A

The dot product constructs a new matrix with each element being the sum of the first matrix's rows by the second matrix's columns.

For the dot product, the matrices can obviously be different dimensions as long as the requirements are met, so we'll slightly alter our generic matrices, A and B, with more specified labels.

$$\begin{bmatrix} a_{1,1} & \cdots & a_{1,n_a} \\ \vdots & \ddots & \vdots \\ a_{m_a,1} & \cdots & a_{m_a,n_a} \end{bmatrix} \tag{7}$$

$$\begin{bmatrix} b_{1,1} & \cdots & \cdots & b_{1,n_b} \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ b_{m_b,1} & \cdots & \cdots & b_{m_b,n_b} \end{bmatrix} \tag{8}$$

Returning dimensions discussion for a dot product:

- $n_a = m_b$ must be true
- The resulting matrix will have dimensions m_a x n_b

An easy way to look at this is to examine the inner and outer dimensions of the matrices:

- $A_{dimensions} = m_a$ x n_a
- $B_{dimensions} = m_b$ x n_b
- Dimensions side by side: (m_a x n_a) (m_b x n_b)
- Required Dimensions: Inner Pair
- Resulting Dimensions: Outer Pair

The resulting matrix will be of the form:

$$\begin{bmatrix} \sum_{i=1}^{n_a} \sum_{j=1}^{m_b} a_{1,i} \cdot b_{j,1} & \cdots & \cdots & \sum_{i=1}^{n_a} \sum_{j=1}^{m_b} a_{1,i} \cdot b_{j,n_b} \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ \sum_{i=1}^{n_a} \sum_{j=1}^{m_b} a_{m_a,i} \cdot b_{j,1} & \cdots & \cdots & \sum_{i=1}^{n_a} \sum_{j=1}^{m_b} a_{m_a,i} \cdot b_{j,n_b} \end{bmatrix} \tag{9}$$

Python requires a NumPy command, as using straight multiplication will multiply element-wise.

Here's an example of the dot product between two 2x2 matrices:

```
import numpy as np

A = np.array([[2, 6], [3, -1]])
B = np.array([[-4, -2], [1, 0]])
C = np.dot(A, B)

"""
array([[ -2,  -4],
       [-13,  -6]])
"""
```

Properties of matrix multiplication (dot product):

- NOT Commutative: AB $\neq$ BA
- Associative: (AB)C = A(BC)
- Distributive: A(B $\pm$ C) = AB $\pm$ AC
- Zero Matrix: Any matrix multiplied by the zero matrix will return a zero matrix, but depending on order, the dimensions could change

3.4 Vector Basics

A vector can be thought of as a special type of matrix in which one of it's dimensions is 1. It can be represented as a column vector (column dimension = 1), or a row vector (row dimension = 1). We can break down or describe any matrix in terms of its vectors, via rows or columns. Similarly, we can build a matrix out of a collection of same length vectors.

Since a vector is a special case of a matrix, as long as dimensional requirements are met, algebraic operations (addition, subtraction, matrix multiplication) between other vectors and matrices is possible.

Vectors contain two pieces of information:

- Direction
- Magnitude (length)

Magnitude is simply the length of a vector, which can be calculated using the distance formula extended to the length of the vector.

Given a vector, v, with dimensions n:

$$\vec{v} = \begin{bmatrix} v_1 \\ \vdots \\ \vdots \\ \vdots \\ v_n \end{bmatrix} \tag{10}$$

We can calculate the magnitude as follows:

$$\|\vec{v}\| = \sqrt{v_1^2 + ... + v_n^2} \tag{11}$$

Given we now know how to find the magnitude of a vector, we can introduce the concept of the unit vector. Every vector has a vector of magnitude 1 which points in the same direction as it, this is the unit vector.

$$\vec{u} = \frac{1}{\|\vec{v}\|} \cdot \vec{v} \tag{12}$$

Syntactically, $\vec{u} = \hat{u}$, which represents a vector has length 1.
Calculating an example in python introduces another NumPy command (and sub-module). Introducing numpy.linalg and np.linalg.norm:

```
v = [1, 2, 3]
v_magnitude = np.linalg.norm(v)
# 3.7416573867739413
u = v / v_magnitude
# [0.26726124 0.53452248 0.80178373]
u_magnitude = np.linalg.norm(u)
# 1.0
```

3.5 Standard Basis Vector (SBV)

The standard basis vectors are a set of vectors which can build every vector in their own respective n-dimensional space by scaling each one. These scaled combinations are called linear combinations.

Set of standard basis vectors in $\mathbb{R}^n$:

$$S = s_1, s_2, ..., s_n = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ \vdots \\ \vdots \\ 0 \end{bmatrix}, ..., \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \vdots \\ 1 \end{bmatrix} \tag{13}$$

There are n vectors, each of which has dimension n.

A linear combination can build any vector in their respective n-dimensional space:

$$\begin{bmatrix} v_1 \\ \vdots \\ \vdots \\ \vdots \\ v_n \end{bmatrix} = \vec{v} = v_1 \cdot s_1 + ... + v_n \cdot s_n \tag{14}$$

3.6 Span

Span of a vector set: collection of all vectors which can be represented by some linear combination of the set.

Two generic examples:

- SBVs in $\mathbb{R}^n$: spans the entirety of $\mathbb{R}^n$
- Zero Vector, $\vec{0}$: spans the origin

3.7 Linear Dependence & Independence

Span is closely related to Linear Dependence & Independence:

- Any n-dimensional linearly independent vectors will span $\mathbb{R}^n$
- It takes n vectors to span $\mathbb{R}^n$
- Vectors must be "different enough"
  - Not "different": when n n-dimensional vectors line in the same (n-1)-dimensional space, they are linearly dependent and won't span $\mathbb{R}^n$
  - Example: Parallel lines in $\mathbb{R}^2$
- Scaled vectors will always lie on the same line (i.e. parallel, collinear, coplanar, etc.)

- Linear Dependence: when one vector in the set can be represented by a linear combination of the other vectors in the set

We'll quickly cover the concept of determining linear (in)dependence, just know that alternative and possibly more efficient methods will be presented.

Given $\vec{0} = c_1 \cdot v_1 + ... + c_n \cdot v_n$, if $c_i = 0 \, \forall i$ is the only solution, then the set is linear independent. Otherwise, the set is linearly dependent.

To clarify a previous point, if a set contains (n+1) vectors in an n-dimensional space, the set must be linearly dependent.

3.8 Linear Subspaces

A subspace must meet 3 criteria:

1. Set includes the zero vector, $\vec{0}$
2. Set is closed under multiplication
3. Set is clsoed under addition

Helpful Hint: If a set is closed under multiplication, then the set automatically includes the zero vector, hence only 2 and 3 are active requirements.

Some examples:

2) Closed under Multiplication:
Given a set of vectors, $V = \{v_1, ..., v_n\}$, the set is closed under multiplication if $c \cdot v_i \in V$ for any c.

3) Closed under Addition
Given a set of vectors, $V = \{v_1, ..., v_n\}$, the set is closed under addition if $v_i + v_j \in V \, \forall i, j$ such that $i \neq j$.

3.9 Spans as Subspaces

Some interesting observations:

- $\mathbb{R}^n$ itself is a subspace
- any hyperplane through the origin, and the $\vec{0}$, itself are subspaces
- Any span (all linear combinations of the set's vectors) is a subspace

3.10 Basis

A linearly independent vector set that spans a vector space like $\mathbb{R}^n$ forms a basis for $\mathbb{R}^n$.

In other words, a vector set is a basis for the space if:

1. Spans the space
2. Linearly independent

Standard Basis: set of unit vectors for $\mathbb{R}^n$.

Any set of n linearly independent vectors forms a basis for $\mathbb{R}^n$.

A quick check for span:
Given a set of vectors, $V = \{v_1, ..., v_n\}$, there exists a c which satisfies:
$c \cdot V = \begin{bmatrix} x_1 \\ \vdots \\ \vdots \\ x_n \end{bmatrix}$
for all combinations of x_i's.

4 Dot Products & Cross Products

We'll take another look into multiplication, more specifically multiplication between vectors, by further investigating properties of the dot product and introducing the cross product.

- Dot Product vs. Cross Products
- Dot Products Between Vectors
- Cauchy-Schwarz Inequality
- Vector Triangle Inequality
- Angle Between Vectors
- Cross Product

4.1 Dot Product vs. Cross Products

- Dot Product: How much two vectors point in the same direction.
- Cross Product: How much two vectors point in opposite directions.

4.2 Dot Products Between Vectors

Relationship with vector length: "The square of the length of a vector is equal to the vector dotted with itself."

Recall the distance formula (magnitude of a vector): $\|\vec{v}\| = \sqrt{v_1^2 + ... + v_n^2}$
Although we haven't explicitly defined the dot product for just vectors, if we follow the definition for dot products between matrices, the dot product for vectors produces a single value.

$$\vec{v}\cdot\vec{v} = v_1\cdot v_1 + ... + v_n\cdot v_n = v_1^2 + ... + v_n^2 \longrightarrow \vec{v}\cdot\vec{v} = \left(\sqrt{v_1^2 + ... + v_n^2}\right)^2 = \|\vec{v}\|^2 \quad (15)$$

A interesting relationship, but let's jump into some more generic properties of the dot product between vectors:

- commutative: $\vec{u} \cdot \vec{v} = \vec{v} \cdot \vec{u}$
- distributive: $(\vec{u} \pm \vec{v}) \cdot \vec{w} = \vec{u} \cdot \vec{w} \pm \vec{v} \cdot \vec{w}$
- associate: $(c \cdot \vec{u}) \cdot \vec{v} = c(\vec{u} \cdot \vec{v})$

Once again, to hit home the general vector dot product equation:

$$\vec{a} \cdot \vec{b} = \sum_{i=1}^{n} a_i \cdot b_i \quad (16)$$

4.3 Cauchy-Schwarz Inequality

The absolute value of a dot product is always less than or equal to the product of their lengths.

$$|\vec{u} \cdot \vec{v}| \leq \|\vec{u}\| \cdot \|\vec{v}\| \quad (17)$$

The inequality is equivalent only when the vectors are collinear (i.e $\vec{u} = c \cdot \vec{v}$).

Given that equivalence means collinearity, the C-S Inequality gives us another way to test for linear independence, since collinearity implies linear dependence.

4.4 Vector Triangle Inequality

The sum of the lengths of two sides of a triangle will always be greater than or equal to the length of the third side.

Given sides of a triangle a, b, and c:

$$c \leq a + b \quad (18)$$

Written in terms of vectors:

$$\|\vec{u} + \vec{v}\| \leq \|\vec{u}\| + \|\vec{v}\| \tag{19}$$

Once again, the inequality is equivalent only when the vectors are collinear and implies linear dependence.

4.5 Angle Between Vectors

So far we've been preoccupied with magnitude, but what other relationships can we find between vectors. How about angle?

Given the following relationship:

$$\vec{u} \cdot \vec{v} = \|\vec{u}\| \cdot \|\vec{v}\| \cdot \cos\theta \tag{20}$$

What further information can solving for θ provide?

One of the more crucial relationships is finding when vectors are perpendicular.

$$\theta = 90^\circ \longrightarrow \cos(90^\circ) = 0 \longrightarrow \vec{u} \cdot \vec{v} = 0 \tag{21}$$

In other words, in the case neither of the vectors are the zero vector, a dot product between vectors resulting in 0 implies the vectors are perpendicular.

Orthogonality: The concept of perpendicular vectors makes sense in 2-dimensions, but to extend to n-dimensions requires the idea of orthogonality.

In general, $\vec{u} \cdot \vec{v} = 0$ (and provided neither of the vectors are the zero vector) implies the vectors are orthogonal.

4.6 Cross Product

The cross product of two vectors, a x b, gives an orthogonal vector to both a and b.
It's calculated by taking the determinant of:

$$\begin{bmatrix} i & j & k \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{bmatrix} \tag{22}$$

The dot product and cross product are opposite ideas:

- Dot Product: how much two vectors point in the same direction.
- Cross Product: how much two vectors point in opposite directions.

The cross product can also be defined as:

$$\left\|\vec{a}x\vec{b}\right\| = \|\vec{a}\| \cdot \left\|\vec{b}\right\| \cdot \sin\theta \tag{23}$$

where the following hold:

- $\theta = 90°$: orthogonal
- $\theta = 0°$ or $180°$: collinear

5 Matrix-Vector Products

We'll go deeper into the relationships between matrices and vectors, and combining and introducing new concepts.

- Fundamental Subspaces
- Complementary, Particular, and General Solutions
- Functions and Transformations
- Linear Transformations as Matrix-Vector Products
- Projections as Linear Transformations
- Compositions of Linear Transformations
- Inverses

5.1 Fundamental Subspaces

The four fundamental subspaces:

- Null: $N(A)$
- Left Null: $N(A^T)$
- Row: $C(A^T)$
- Column: $C(A)$

Null Space

The Null Space of a matrix A is defined as the set of all vectors which satisfy:

- $A \cdot \vec{x} = \vec{0}$
- $N(A) = N(rref(A))$

Column Space

The Column Space of a matrix A is defined as:

- $C(A)$ = linear combinations of columns of A
- = span of columns of A

To find the Column Space, $C(A)$, we can use $A \cdot \vec{x} = b \rightarrow b \in C(A)$, along with the $N(A)$ and the concept of a basis.

Recall:

- Basis: a collection of linearly independent vectors that span an entire vector space.
- Independence: None of the vectors can be expressed as a linear combination of the others.

We know that if $N(A) = \vec{0} \rightarrow$ columns of A are linearly independent (forms a basis). And if $N(A) = \vec{0}, \vec{v} \rightarrow$ columns of A are linearly dependent (do not form a basis).

A quick aside on dimensions. Given an mxn matrix A, $N(A)$ will is a subspace of $\mathbb{R}^n$, and $C(A)$ of A is a subspace in $\mathbb{R}^m$.

- $C(A)$: r-dimensional subspace where r is the number of pivot columns
- $N(A)$: (n-r)-dimensional subspace where n is the total number of columns, and would be the total number of non-pivot columns

For example, given the matrix:

$$\begin{bmatrix} 2 & 1 & 3 & 1 \\ 4 & -2 & 8 & 4 \\ 5 & 6 & -2 & -3 \end{bmatrix} \tag{24}$$

then,

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = x_4 * \begin{bmatrix} \frac{3}{31} \\ \frac{8}{31} \\ \frac{-15}{31} \\ 1 \end{bmatrix}$$

Therefor,

$$N(A) = N(rref(A)) = Span(\begin{bmatrix} \frac{3}{31} \\ \frac{8}{31} \\ \frac{-15}{31} \\ 1 \end{bmatrix})$$

And we also have $Dim(N(A)) = 1$.

Given that x_4 is a "free", meaning it is a scalar which can take on any value (infinitely many), this means $N(A)$ doesn't just contain the zero vector and there is more than one solution to $A\vec{x} = \vec{0}$, which means that the column vectors of A are a linearly dependent set.

Since the vector set (columns) in our matrix A are not linearly independent, then they can't form a basis for the column space.

The column space is still, however, a linearly combination of the column vectors. In other words $C(A) = Span([A_1], [A_2], [A_3], [A_4])$.

We can take the span and create a basis, though!

Since x_4 is free, we can set $x_4 = 0$. If A is still our aforementioned matrix, then let A' be our matrix with $x_4 = 0$. If we solve for $N(A')$, then we'll find that the only solution to $N(A')$ is $\vec{0}$. Thus, $C(A') = Span([A'_1], [A'_2], [A'_3])$ is also a basis for the column space. Hence, $Dim(C(A)) = 3$.

5.2 Complementary, Particular, and General Solutions

If we let any $\vec{x}$ such that $A\vec{x} = \vec{0} \rightarrowtail \vec{x}$ be known as the complementary solution.

Then, if we let any $\vec{x}$ such that $A\vec{x} = \vec{b} \rightarrowtail \vec{x}$ be known as the particular solution.

Then, the general solution is the sum of the complementary and the particular solution.

For notation's sake, let

- x_n: complementary solution
- x_p: particular solution
- x: general solution

then $A\vec{x_n} + A\vec{x_p} = \vec{0} + \vec{b} \rightarrow$

$A(\vec{x_n} + \vec{x_p}) = \vec{b} = A\vec{x}$.

Applying this to solving for the complete set of solutions, if we set any free variables to 0 for both the complementary and particular solution, then we get the complete set of solutions.

A quick summary:

- complementary solution: $N(A)$

- particular solution: $C(A)$
- $dim(A) = mxn$
- rank: $rank(A) = dim(C(A)) = r$
- nullity: $nullity(A) = dim((N(A))) = n - r$

5.3 Functions and Transformations

A transformation is function which maps vectors. Some common notation and properties associated with transformations.
Given the transformation from $\mathbb{R}^m \to \mathbb{R}^n$:

- $T(\vec{v}) = \vec{v}_T$
- domain: $\mathbb{R}^m$
- codomain: $\mathbb{R}^n$
- range: specific points mapped inside the codomain (may not be all of $\mathbb{R}^n$)

A transformation matrix, M, contains instructions for how to alter a vector or set of vectors when M is applied to them.

- Preimage: the vector, or vector set, before a transformation
- Image: the vector, or vector set, after a transformation

Consider the transformation matrix

$$M = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \tag{25}$$

Given preimage vectors of the unit vectors, then then we can create corresponding image vectors:

$$m_i = \begin{bmatrix} a & b \\ c & d \end{bmatrix} * \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} a \\ c \end{bmatrix} \tag{26}$$

and

$$m_j = \begin{bmatrix} a & b \\ c & d \end{bmatrix} * \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} b \\ d \end{bmatrix} \tag{27}$$

This is extends to any dimensions and can be applied to any vector in a hyperspace.
A few more facts about transformations:

- Transformations are subspaces
- T^{-1} maps to the preimage in the domain
- Kernel: all vectors that result in $\vec{0}$ under T

5.4 Linear Transformations as Matrix-Vector Products

$T : \mathbb{R}^m \to \mathbb{R}^n$ is a linear transformation if:

- $T(\vec{u} + \vec{v}) = T(\vec{u}) + T(\vec{v})$
- $T(c\vec{u}) = cT(\vec{u})$ AND $T(c\vec{v}) = cT(\vec{v})$

Given a codomain equation, we can backwards solve to obtain the transformation. This is done by plugging I in.

$$T(\begin{bmatrix} a_1 \\ a_2 \end{bmatrix}) = \begin{bmatrix} -a_1 + 2a_2 \\ a_2 - 3a_1 \\ a_1 - a_2 \end{bmatrix} \tag{28}$$

Substitute I in the following manner:

$$T(\begin{bmatrix} 1 \\ 0 \end{bmatrix}) = \begin{bmatrix} -1 \\ -3 \\ 1 \end{bmatrix} \tag{29}$$

and

$$T(\begin{bmatrix} 0 \\ 1 \end{bmatrix}) = \begin{bmatrix} 2 \\ 1 \\ -1 \end{bmatrix} \tag{30}$$

Hence,

$$T = \begin{bmatrix} -1 & 2 \\ -3 & 1 \\ 1 & -1 \end{bmatrix} \tag{31}$$

Linear Transformation as Rotations

- Use a linear transformation to rotate by a certain degree
- Use a matrix combination of sin and cos, changing with n for $\mathbb{R}^n$

Adding and Scaling Linear Transformations
Given the following transformations, $S(\vec{x}) = A\vec{x}$ and $T(\vec{x}) = B\vec{x}$, where both S and T are $\mathbb{R}^n \to \mathbb{R}^m$.

- Summations: $(S + T)\vec{x} = S\vec{x} + T\vec{x} = (A + B)\vec{x}$
- Scaling: $cT(\vec{x}) = c(B\vec{x}) = (cB)\vec{x}$

5.5 Projections as Linear Transformations

We can think of a projection in the following manner. Take a vector $\vec{v}$ casting a shadow on a line L, then

Projection of $vecv$ onto L: $Proj_L(\vec{v})$

And if $\vec{x}$ is any vector on L, $Proj_L(\vec{v})$ is a scaled version of $\vec{x}$, i.e.

$Proj_L(\vec{v}) = c\vec{x}$

Another property of projections is that
$v - Proj_L(\vec{v})$ is orthogonal to L,

thus,
$\rightarrow c = \frac{\vec{v}\vec{x}}{\vec{x}\vec{x}}$

$Proj_L(\vec{v}) = \rightarrow c = \frac{\vec{v}\vec{x}}{\vec{x}\vec{x}}\vec{x}.$

Recall the fact that $\vec{x}\vec{x} = ||\vec{x}||^2$, then

$Proj_L(\vec{v}) = (\frac{\vec{v}\vec{x}}{||\vec{x}||^2})\vec{x}$

If we normalize $\vec{x}$, then

$Proj_L(\vec{v}) = (\vec{v}\vec{u})\vec{u},$

Using this, it's easy to show that the projection is a linear transformation due to

$Proj_L(\vec{v}) = (\vec{v}\vec{u})\vec{u} = A\vec{v}$

5.6 Compositions of Linear Transformations

Given the following Linear Transformations:

- $T : X \rightarrow Y$
- $S : Y \rightarrow Z$

Then, we can summarize a composition with the following notation (equivalent):

- $T(S(\vec{x})) : X \rightarrow Z$
- $T \circ S : X \rightarrow Z$

Compositions, themselves, are also linear transformations.

$T \circ S(x + y) = T(S(X) + S(y))$
$= T(S(X)) + T(S(y))$

$= T \circ Sx + T \circ Sy$
and
$T \circ S(cx) = cT \circ S(x)$

Finally, because they are linear transformations, they can be written as matrix-vector products:

Give $S(x) = Ax$ and $T(x) = Bx$,

then $T \circ S(x) = T \circ Ax = BAx = Cx$

Note the ordering of the matrix multiplication here (it may almost seem counterintuitive).

5.7 Inverses

When specifically speaking of transformations and invertibility:

- The identity matrix, I, maps to itself $I \to I$
- Definition: A transformation is invertible if it has an inverse:
- $T : A \to B$
- $T^{-1} \circ T = I_A$
- $T \circ T^{-1} = I_B$

Invertibility and Uniqueness:

If a transformation, T, is invertible, then the following hold:

- T^{-1} is unique
- T maps $a \in A$ to a unique $b \in B$
- T^{-1} maps $b \in B$ back to a unique $a \in A$

... this brings us to the topics of **Surjective and Injective**!