Notes for Matrix Algebra
TMATH 308

Ben Foster*
benf94@uw.edu

March 31, 2015

Abstract

This document contains the notes from the course TMATH 308 and does not necessarily contain all the information provided by the instructor or textbook.

*Institute of Technolgy, University of Washington Tacoma

Contents

1 Vectors

1.1 Geometry and Algebra of Vectors

A vector is a way of describing direction and magnitude. Vectors can be written in the following forms:

$$\vec{b} = \mathbf{b} = [-1, 3] =< -1, 3 >= \begin{bmatrix} -1 \\ 3 \end{bmatrix}$$

There is also such a thing as the **zero vector** where the vector has no direction and zero magnitude.

Vector addition is easy:

$$\vec{u} + \vec{v} =< u_1 + v_1, u_2 + v_2 >$$

The head to tail rule: Given vectors $\vec{u}$ and $\vec{v}$ in $\mathbb{R}^2$, translate $\vec{v}$ so that its tail coincides with the head of $\vec{u}$. The sum of $\vec{u}$ and $\vec{v}$ is the vector from the tail of $\vec{u}$ to the head of $\vec{v}$.

Subtraction works very similar to addition and you can multiply vectors by scalars:

$$c\vec{v} = c < v_1, v_2 >=< cv_1, cv_2 >$$

1.1.1 Theorem 1.1

Algebraic Properties of Vectors in $\mathbb{R}^n$
Let $\vec{u}, \vec{v}, and \vec{w}$ be vectors in $\mathbb{R}^n$ and let c and d be scalars.

- $\vec{u} + \vec{v} = \vec{v} + \vec{y}$
- $(\vec{u} + \vec{v}) + \vec{w} = \vec{u} + (\vec{v} + \vec{w}$
- $\vec{u} + \vec{0} = \vec{u}$
- $\vec{u} + (-\vec{u}) = \vec{0}$
- $c(\vec{u} + \vec{v}) = c\vec{u} + c\vec{v}$
- $(c + d)\vec{u} = c\vec{u} + d\vec{u}$
- $c(d\vec{u}) = (cd)\vec{u}$
- $1\vec{u} = \vec{u}$

Linear Combinations and Coordinates A vector that is the sum of scalar multiples of other vectors is said to be a linear combination of those vectors. The formal definition follows:
A vector $\vec{v}$ is a ***linear combination*** of vectors $\vec{v_1}, \vec{v_2}, ..., \vec{v_k}$ if there are scalars $c_1, c_2, ..., c_k$ such that $\vec{v} = c_1\vec{v_1} + c_2\vec{v_2} + ... + c_k\vec{v_k}$. The scalars $c_1, c_2, ..., c_k$ are called the ***coefficients*** of the linear combination.

n-ary vectors When writing $\mathbb{Z}_n^k$ we are saying the vectors of size k with integers modulo n.

1.2 The Dot Product

Definition:
If $\vec{u} =< u_1, u_2, ..., u_n >$ and $\vec{v} =< v_1, v_2, ..., v_n >$ then the **dot product** of $\vec{u}$ and $\vec{v}$ is defined by:

$$\vec{u} \cdot \vec{v} = u_1v_1 + u_2v_2 + ... + u_nv_n$$

1.2.1 Theorem 1.2

Let $\vec{u}, \vec{v} and \vec{w}$ be vectors in $\mathbb{R}^n$ and let c be a scalar.

- $\vec{u} \cdot \vec{v} = \vec{v} \cdot \vec{u}$
- $\vec{u} \cdot (\vec{v} + \vec{w}) = \vec{u} \cdot \vec{v} + \vec{u} \cdot \vec{w}$
- $(c\vec{u}) \cdot \vec{v} = c(\vec{u} \cdot \vec{v})$
- $\vec{u} \cdot \vec{u} \geq 0$ and $\vec{u} \cdot \vec{u} = 0$ if and only if $\vec{u} = \vec{0}$

Definition The **length** of a vector $\vec{v} =< v_1, v_2, ..., v_n >$ in $\mathbb{R}^n$ is the non-negative scalar $||\vec{v}||$ defined by:

$$||\vec{v}|| = \sqrt{\vec{v} \cdot \vec{v}} = \sqrt{v_1^2 + v_2^2 + ... + v_n^2}$$

1.2.2 Theorem 1.3

Let $\vec{v}$ be a vector in $\mathbb{R}^n$ and let c be a scalar. Then

- $||\vec{v}|| = 0$ if and only if $\vec{v} = \vec{0}$
- $||c\vec{v}|| = (|c|)||\vec{v}||$

The **unit vector** of a vector is a vector with magnitude 1 and direction of the original vector. It is found by doing this for a vector $\vec{v}$:

$$\vec{u} = \frac{\vec{v}}{||\vec{v}||}$$

1.2.3 Theorem 1.4

The **Cauchy=Schwarz Inequality**:
For all vectors $\vec{u}$ and $\vec{v}$ in $\mathbb{R}^n$

$$|\vec{u} \cdot \vec{v}| \leq ||\vec{u}||||\vec{v}||$$

1.2.4 Theorem 1.5

The **Triangle Inequality**:
For all vectors $\vec{u}$ and $\vec{v}$ in $\mathbb{R}^n$

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

Distance The distance between two vectors is the direct analogue of the distance between two points on the real number line or two points in the Cartesian plane.

The **distance** between vectors $\vec{u}$ and $\vec{v}$ in $\mathbb{R}^n$ is defined by:

$$d(\vec{u}, \vec{v}) = ||\vec{u} - \vec{v}||$$

Angles The dot product can also be used to calculate the angle between a pair of vectors. For non-zero vectors $\vec{u}$ and $\vec{v}$ in $\mathbb{R}^n$,

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

θ	$0°$	$30°$	$45°$	$60°$	$90°$
$cos\theta$	$\frac{\sqrt{4}}{2} = 1$	$\frac{\sqrt{3}}{2}$	$\frac{\sqrt{2}}{2} = \frac{1}{\sqrt{2}}$	$\frac{\sqrt{1}}{2} = \frac{1}{2}$	$\frac{\sqrt{0}}{2} = 0$

Table 1: Cosines of Special Angles

Orthogonal Vectors The concept of perpendicularity is fundamental to geometry.
The two vectors $\vec{u}$ and $\vec{v}$ in $\mathbb{R}^n$ are **orthogonal** to each other if $\vec{u} \cdot \vec{v} = 0$.

1.2.5 Theorem 1.6

Pythagoras' Theorem
For all vectors $\vec{u}$ and $\vec{v}$ in $\mathbb{R}^n$, $||\vec{u} + \vec{v}||^2 = ||\vec{u}||^2 + ||\vec{v}||^2$ if and only if $\vec{u}$ and $\vec{v}$ are orthogonal.

Projections We will now consider the problem of finding the distance from a point to a line in the context of vectors.
If $\vec{u}$ and $\vec{v}$ are vectors in $\mathbb{R}^n$ and $\vec{u} \neq \vec{0}$, then the **projection of** $\vec{v}$ **onto** $\vec{u}$ is the vector $proj_{\vec{u}}(\vec{v})$ defined by:

$$proj_{\vec{u}}(\vec{v}) = \left(\frac{\vec{u} \cdot \vec{v}}{\vec{u} \cdot \vec{u}}\right) \vec{u}$$

1.3 Lines and Planes

The **normal form** of the equation of line ℓ in $\mathbb{R}$ is

$$\vec{n} \cdot (\vec{x} - \vec{p}) = 0 \text{ or } \vec{n} \cdot \vec{x} = \vec{n} \cdot \vec{p}$$

where $\vec{p}$ is a specific point on ℓ and $\vec{n} \neq \vec{0}$ is a normal vector for ℓ. The **General Form** of the equation of ℓ is $ax + by = c$ where $\vec{n} =< a, b >$.

1.4 Applications

1.4.1 Force Vectors

We can use vectors to model force. It is often the case that multiple forces act on an object. In such situations, the net result of all the forces acting together is a single force called the ***resultant***, which is simply the vector sum of the individual forces. When several forces act on an object, it is possible that the resultant force is zero. In this case, the object is clearly not moving in any direction and we say that it is in ***equilibrium***. When an object is in equilibrium, and the force vectors acting on it are arranged head-to-tail, the result is a closed polygon.
We can use basic trigonometry to find the forces acting in a particular direction. For example, if the angle between the force vector and the x-axis is $20°$ then the amount of force acting in the x direction is $||f_x|| = ||f||cos(20°)$ and the amount of force acting in the y direction is $||f_y|| = ||f||sin(20°)$.

We can also use the law of sines to determine forces. For example, given the figure: And then using the law of sines, we get:

$$\frac{||f_1||}{sin45°} = \frac{||f_2||}{sin30°} = \frac{||r||}{sin105°}$$

1.4.2 Code Vectors

Sometimes the intent of transmitting information using codes is to disguise the message being sent. In this section we will use vectors to design codes for detecting errors that can not only detect but also correct errors. The vectors that arise in the study of codes are not vectors in $\mathbb{R}^n$ but vectors in $\mathbb{Z}_m^n$. In practice, we have a message we wish to transmit. We begin by encoding each "word" of the message as a binary vector.

Definition A ***binary code*** is a set of binary vectors (of the same length) called ***code vectors***. The process of converting a message into code vectors is called ***encoding***, and the reverse process is called ***decoding***.

The message to be transmitted may itself consist of binary vectors. In this case, a simple but useful error-detecting code is a ***parity check code***, which is created by appending an extra component called a ***check digit*** to each vector so that the parity (the total number of 1s) is even.

Let's look at this concept a bit more formally. Suppose the message is the binary vector $\vec{b} =< b_1, b_2, ..., b_n >$ in $\mathbb{Z}_2^n$. Then the parity check code vector is $\vec{v} =< b_1, b_2, ..., b_n, d >$ in $\mathbb{Z}_2^{n+1}$ where the check digit d is chosen so that:

$$b_1 + b_2 + ... + b_n + d = 0 \text{ in } \mathbb{Z}_2$$

or equivalently so that

$$\vec{1} \cdot \vec{v} = 0$$

Parity check codes are a special case of the more general ***check digit codes***, which we will consider after first extending the foregoing ideas to more general settings. Codes using m-ary vectors are called ***m-ary codes***. For example, let $\vec{b} =< b_1, b_2, ..., b_n >$ be a vector in $\mathbb{Z}_3^n$. Then a check digit code vector may be defined by $\vec{v} =< b_1, b_2, ..., b_n, d >$, with the check digit d chosen so that $\vec{1} \cdot \vec{v} = 0$.

While simple check digit codes will detect single errors, it is often important to catch other common types of errors as well, such as the accidental interchange, or *transposition*, of two adjacent components. For such purposes, other types of check digit codes have been designed. Many of these simply replace the check vector $\vec{1}$ by some other carefully chosen vector $\vec{c}$.

The Universal Product Code or UPC, is a code associated with the bar codes found on many types of merchandise. The black and white bars that are scanned by a laser at a store's checkout counter correspond to a 10-ary vector $\vec{u} =< u_1, u_2, ..., u_{11}, d >$ of length 12. The first 11 components form a vector in $\mathbb{Z}_{10}^{11}$ that gives manufacturer and product information; The last component d is a check digit chosen so that $\vec{c} \cdot \vec{u} = 0$ in $\mathbb{Z}_10$, where the check vector $\vec{c} =< 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1 >$.

The 10-digit International Standard Book Number (ISBN-10) code is another widely used check digit code. It is designed to detect more types of errors than the UPC and, consequently, is slightly more complicated. The code vector is a vector in $\mathbb{Z}_{11}^{10}$. The first nine components five country, publisher, and book information; the tenth component is the check digit. For the ISBN-10 code, the check vector is the vector $\vec{c} =< 10, 9, 8, 7, 6, 5, 4, 3, 2, 1 >$ and we require that $\vec{c} \cdot \vec{b} = 0$ in $\mathbb{Z}_{11}$.

2 Systems of Linear Equations

2.1 Intro to Systems of Linear Equations

2.2 Direct Methods for Solving

2.3 Spanning Sets and Linear Independence

2.4 Applications

2.4.1 Allocation of Resources

2.4.2 Balancing Chemical Equations

2.4.3 Network Analysis

2.4.4 Electrical Networks

2.4.5 Linear Economic Models

2.4.6 Finite Linear Games

2.5 Iterative Methods for Solving

3 Matrices

3.1 Intro to Matrices in Action

3.2 Matrix Operations

3.3 Matrix Algebra

3.4 The inverse of a Matrix

3.5 The LU Factorization

3.6 Subspaces, Basis, Dimension, and Rank

3.7 Intro to Linear Transformations

3.8 Applications

3.8.1 Markov Chains

3.8.2 Linear Economic Models

3.8.3 Population Growth

3.8.4 Graphs and Digraphs

3.8.5 Error-Correcting Codes

4 Eigenvalues and Eigenvectors

4.1 Introduction

4.2 Determinants

4.3 Eigenvalues and vectors of $n \times n$ Matrices

4.4 Similarity and Diagonalization

4.5 Iterative Methods for Computing Eigenvalues

4.6 Applications and the Perron-Frobenius Theorem

4.6.1 Markov Chains

4.6.2 Population Growth

4.6.3 The Perron-Frobenius Theorem

4.6.4 Linear Recurrence Relations

4.6.5 Systems of Linear Differential Equations

4.6.6 Discrete Linear Dynamical Systems

5 Orthogonality

5.1 Orthogonality in $\mathbb{R}^n$

5.2 Orthogonal Complements and Orthogonal Projections

5.3 The Gram-Schmidt Process and the QR Factorization

5.4 Orthogonal Diagonalization of Symmetric Matrices

5.5 Applications

5.5.1 Dual Codes

5.5.2 Quadratic Forms

5.5.3 Graphing Quadratic Equations

6 Vector Spaces

6.1 Vector Spaces and Subspaces

6.2 Linear Independence, Basis, and Dimension

6.3 Change of Basis

6.4 Linear Transformations

6.5 The Kernel and Range of a Linear Transformation

6.6 The Matrix of a Linear Transformation

6.7 Applications

6.7.1 Homogeneous Linear Differential Equations

6.7.2 Linear Codes

7 Distance and Approximation

7.1 Inner Product Spaces

7.2 Norms and Distance Functions

7.3 Least Squares Approximation

7.4 The Singular Value Decomposition

7.5 Applications

7.5.1 Approximation of Functions

7.5.2 Error-Correcting Codes