Math 1600B Lecture 4, Section 2, 13 Jan 2014

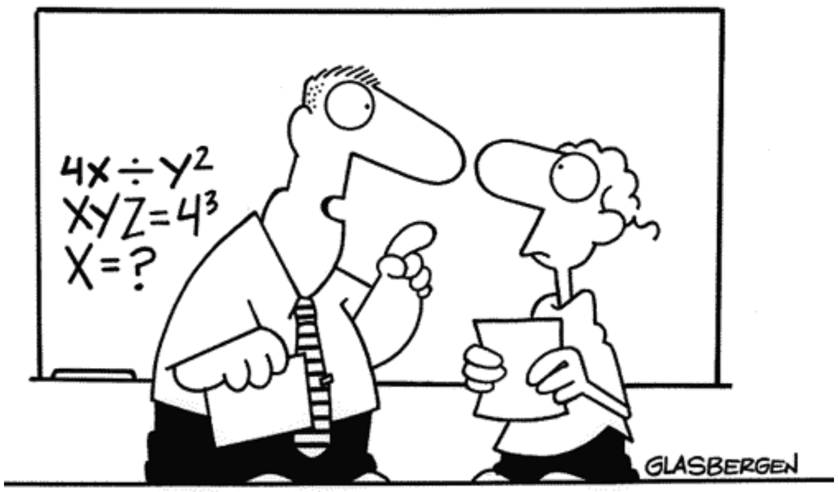
Announcements:

Read Section 1.3 for next class. Work through recommended homework questions.

Tutorials start **this week**, and include a **quiz** covering Sections 1.1, 1.2 as well as the code vectors part of 1.4. It does not cover Section 1.3 or the Exploration after Section 1.2.
The quizzes last 20 minutes, and are at the end of the tutorial, so you have time for questions at the beginning.
Questions are *similar* to homework questions, but may be slightly different.
There will be two true/false questions, for which you must explain your answers.
You must write in the tutorial you are registered in.
Different sections have different quizzes, but it is still considered an academic offense to share information about quizzes.

Office hour: today, 1:30-2:30, MC103B.
My office hour on Wednesday is cancelled this week.

Lecture notes (this page) available from course web page.



"Algebra class will be important to you later in life because there's going to be a test six weeks from now."

(Actually, a quiz this week and a midterm in 6 1/2 weeks...)

Partial review of last lecture:

Section 1.2: Length and Angle: The Dot Product

Definition: The **dot product** or **scalar product** of vectors $\vec{u}$ and $\vec{v}$ in $\mathbb{R}^n$ is the real number defined by

$$\vec{u} \cdot \vec{v} := u_1 v_1 + \cdots + u_n v_n .$$

This has familiar properties; see Theorem 1.2.

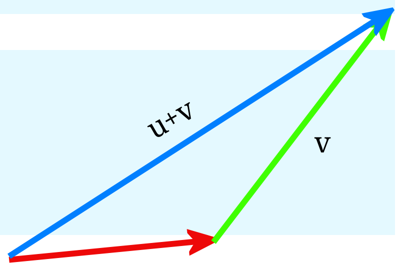
Definition: The **length** or **norm** of $\vec{v}$ is the scalar $\|\vec{v}\|$ defined by

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

A vector of length 1 is called a **unit** vector.

Theorem 1.5: The Triangle Inequality: For all $\vec{u}$ and $\vec{v}$ in $\mathbb{R}^n$,

$$\|\vec{u} + \vec{v}\| \le \|\vec{u}\| + \|\vec{v}\|.$$



We define the **distance** between vectors $\vec{u}$ and $\vec{v}$ by the formula u

$$d(\vec{u},\vec{v}) := \|\vec{u}-\vec{v}\| = \sqrt{(u_1-v_1)^2+\cdots+(u_n-v_n)^2}.$$

Angles from dot product

Theorem 1.4: The Cauchy-Schwarz Inequality: For all $\vec{u}$ and $\vec{v}$ in $\mathbb{R}^n$,

$$|\vec{u}\cdot\vec{v}| \le \|\vec{u}\|\,\|\vec{v}\|.$$

We can therefore use the dot product to *define* the **angle** between two vectors $\vec{u}$ and $\vec{v}$ in $\mathbb{R}^n$ by the formula

$$\cos\theta = \frac{\vec{u}\cdot\vec{v}}{\|\vec{u}\|\,\|\vec{v}\|}, \quad \text{i.e.,} \quad \theta := \arccos\left(\frac{\vec{u}\cdot\vec{v}}{\|\vec{u}\|\,\|\vec{v}\|}\right),$$

where we choose $0 \le \theta \le 180^\circ$. This makes sense because the fraction is between -1 and 1.

The formula can also be written

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

New material

Orthogonal Vectors

How can we tell whether two vectors are orthogonal / perpendicular?
Easy: $\theta = 90^\circ$ is the only angle for which $\cos\theta = 0$.
So $\vec{u}$ and $\vec{v}$ are **orthogonal** if and only if $\vec{u}\cdot\vec{v}=0$.

Example: If $\vec{u}=[1,2,3]$ and $\vec{v}=[1,1,-1]$ in $\mathbb{R}^3$, then $\vec{u}\cdot\vec{v} = 1\cdot1+2\cdot1+3\cdot(-1) = 1+2-3=0$, so $\vec{u}$ and $\vec{v}$ are orthogonal.

Also, $\vec{u}=[1,2,3]$ and $\vec{v}=[1,1,1]$ in $\mathbb{Z}_3^3$ are orthogonal, since $\vec{u}\cdot\vec{v} = 1+2+3=6=0 \pmod 3$.

Pythagorean theorem in $\mathbb{R}^n$: If $\vec{u}$ and $\vec{v}$ are orthogonal, then

$$\|\vec{u}+\vec{v}\|^2 = \|\vec{u}\|^2 + \|\vec{v}\|^2.$$

Explain on board, using Theorem 1.2.

Projections

Use board to derive formula for **the projection of $\vec{v}$ onto $\vec{u}$**:

$$\text{proj}_{\vec{u}}(\vec{v}) = \left(\frac{\vec{u}\cdot\vec{v}}{\vec{u}\cdot\vec{u}}\right)\vec{u}.$$

Here $\vec{u}$ must not be $\vec{0}$, but $\vec{v}$ can be any vector. To help remember the formula, note that the denominator ensures that the answer does not depend on the length of $\vec{u}$.

The applet we saw before is useful for understanding projections as well. Java version.

Example: If $\vec{u} = [-1,1,0]$ and $\vec{v} = [1,2,3]$ then

$$\begin{aligned}\text{proj}_{\vec{u}}(\vec{v}) = \frac{\vec{u}\cdot\vec{v}}{\vec{u}\cdot\vec{u}}\,\vec{u} &= \frac{-1+2+0}{1+1+0}\,[-1,1,0] \\ &= \frac{1}{2}\,[-1,1,0] = [-\frac{1}{2}\,,\frac{1}{2}\,,0]\end{aligned}$$

Questions

True/false: If $\vec{u}$, $\vec{v}$ and $\vec{w}$ are vectors in $\mathbb{R}^n$ such that $\vec{u}\cdot\vec{v} = \vec{u}\cdot\vec{w}$ and $\vec{u} \neq \vec{0}$, then $\vec{v} = \vec{w}$.

False. For example, if $\vec{u} = [1,0]$, $\vec{v} = [0,1]$ and $\vec{w} = [0,2]$, then $\vec{u}\cdot\vec{v} = 0$ and $\vec{u}\cdot\vec{w} = 0$ but $\vec{v} \neq \vec{w}$.

True/false: If $\vec{u}$ is orthogonal to both $\vec{v}$ and $\vec{w}$, then $\vec{u}$ is orthogonal to $2\vec{v}+3\vec{w}$.

True, because $\vec{u} \cdot (2\vec{v} + 3\vec{w}) = 2\,\vec{u} \cdot \vec{v} + 3\,\vec{u} \cdot \vec{w} = 2(0) + 3(0) = 0\,.$

You only answer **true** if a statement is *always* true. You justify this answer by giving a general explanation of why it is always true, not just an example where it happens to be true.

You answer **false** if a statement can in *some case* be false. You justify this answer by giving an explicit example where the statement is false.

Question: Suppose I tell you that $\vec{u} \cdot \vec{v} = 1/2$ and $\vec{u} \cdot \vec{w} = -1$. What is $\vec{u} \cdot (2\vec{v} + 3\vec{w})$?

Solution: $\vec{u} \cdot (2\vec{v} + 3\vec{w}) = 2\,\vec{u} \cdot \vec{v} + 3\,\vec{u} \cdot \vec{w} = 2(1/2) + 3(-1) = -2\,.$

Question: Does $\text{proj}_{\vec{u}}(\vec{v})$ always point in the same direction as $\vec{u}$?

Solution: No. It is always parallel, but might point in the opposite direction. For example, if $\vec{u} = [1, 0]$ and $\vec{v} = [-1, 1]$ then $\text{proj}_{\vec{u}}(\vec{v}) = [-1, 0] = -\vec{u}$.

Section 1.4: Applications: Code Vectors (we aren't covering force vectors)

We're going to study a way to encode data that allows us to detect transmission errors. Used on CDs, UPC codes, ISBN numbers, credit card numbers, etc.

Example 1.37: Suppose we want to send the four commands "forward", "back", "left" and "right" as a sequence of 0s and 1s. We could use the following code:

$$\text{forward} = [0, 0], \quad \text{back} = [0, 1], \quad \text{left} = [1, 0], \quad \text{right} = [1, 1].$$

But if there is an error in our transmission, the Mars rover will get the wrong message and will drive off of a cliff, wasting billions of dollars of taxpayer money (but making for some good NASA jokes).

Here's a more clever code:

$$\text{forward} = [0, 0, 0], \quad \text{back} = [0, 1, 1], \quad \text{left} = [1, 0, 1], \quad \text{right} = [1, 1, 0].$$

If any single *bit* (binary digit, a 0 or a 1) is flipped during transmission, the Mars rover will notice the error, since all of the **code vectors** have an **even** number of 1s. It could then ask for retransmission of the command.

This is called an **error-detecting code**. Note that it is formed by adding a bit to the end of each of the original code vectors so that the total number of 1s is even.

In vector notation, we replace a vector $\vec{b} = [v_1, v_2, \ldots, v_n]$ with the vector $\vec{v} = [v_1, v_2, \ldots, v_n, d]$ such that $\vec{1} \cdot \vec{v} = 0 \pmod{2}$, where $\vec{1} = [1, 1, \ldots, 1]$.

Exactly the same idea works for vectors in $\mathbb{Z}_3^n$; see Example 1.39 in the text.

Note: One problem with the above scheme is that **transposition** errors are not detected: if we want to send $[0, 1, 1]$ but the first two bits are exchanged, the rover receives $[1, 0, 1]$, which is also a valid command. We'll see codes that can detect transpositions.

Example 1.40 (UPC Codes): The Univeral Product Code (bar code) on a product is a vector in $\mathbb{Z}_{10}^{12}$, such as

$$\vec{u} = [6, 7, 1, 8, 6, 0, 0, 1, 3, 6, 2, 4].$$

6 71860 01362 4

Instead of using $\vec{1}$ as the **check vector**, UPC uses

$$\vec{c} = [3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1].$$

The last digit is chosen so that $\vec{c} \cdot \vec{u} = 0 \pmod{10}$.

For example, if we didn't know the last digit of $\vec{u}$, we could compute

$$\vec{c} \cdot [6, 7, 1, 8, 6, 0, 0, 1, 3, 6, 2, d] = \cdots = 6 + d \pmod{10}$$

and so we would find that we need to take $d = 4$, since $6 + 4 = 0 \pmod{10}$.

This detects any single error. The pattern in $\vec{c}$ was chosen so that it detects many transpositions, but it doesn't detect when digits whose difference is 5 are transposed. For example, $3 \cdot 5 + 1 \cdot 0 = 15$ and $3 \cdot 0 + 1 \cdot 5 = 5$, and these are the same modulo 10.

Example 1.41 (ISBN Codes): ISBN codes use vectors in $\mathbb{Z}_{11}^{10}$. The check vector is $\vec{c} = [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]$. Because 11 is a prime number, this code detects all single errors and *all* single transposition errors.

Summary: To create a code, you choose m (which determines the allowed digits), n (the number of digits in a code word), and a **check vector** $\vec{c} \in \mathbb{Z}_m^n$. Then the **valid words** $\vec{v}$ are those with $\vec{c} \cdot \vec{v} = 0$. If $\vec{c}$ ends in a 1, then you can always choose the last digit of $\vec{v}$ to make it valid.

Note: This kind of code can only reliably detect one error, but more sophisticated codes can detect multiple errors. There are even **error-correcting codes**, which can *correct* multiple errors in a transmission without needing it to be resent. In fact, you can drill small holes in a CD, and it will still play the entire content perfectly.

Question: The Dan code uses vectors in $\mathbb{Z}_4^3$ with check vector $\vec{c} = [3, 2, 1]$. Find the check digit d in the code word $\vec{v} = [2, 2, d]$.

Solution: We compute

$$\begin{aligned} \vec{c} \cdot \vec{v} = [3, 2, 1] \cdot [2, 2, d] &= 3 \cdot 2 + 2 \cdot 2 + 1 \cdot d \\ &= 10 + d = 2 + d \pmod{4} \end{aligned}$$

To make $\vec{c} \cdot \vec{v} = 0 \pmod{4}$, we choose $d = 2$.

This is the end of the material for quiz 1.