L09MatrixOperations

September 29, 2015

1 Matrix Operations

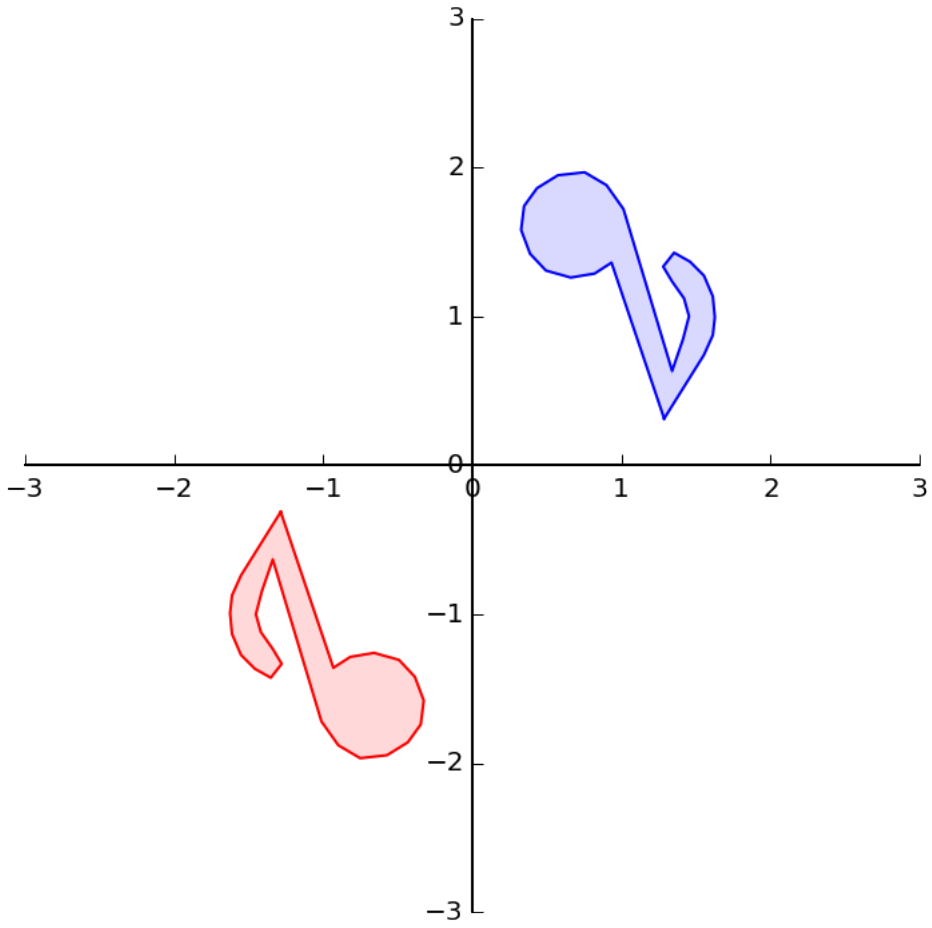
1.1 Composing Linear Transformations

Recall the case of reflection through the origin:

In [9]:
```
note = dm.mnote()
A = np.array(
    [[-1, 0],
     [ 0,-1]])
dm.plotSetup()
dm.plotShape(note)
dm.plotShape(A.dot(note),'r')
Latex(r'Reflection through the origin')
```

Out[9]:

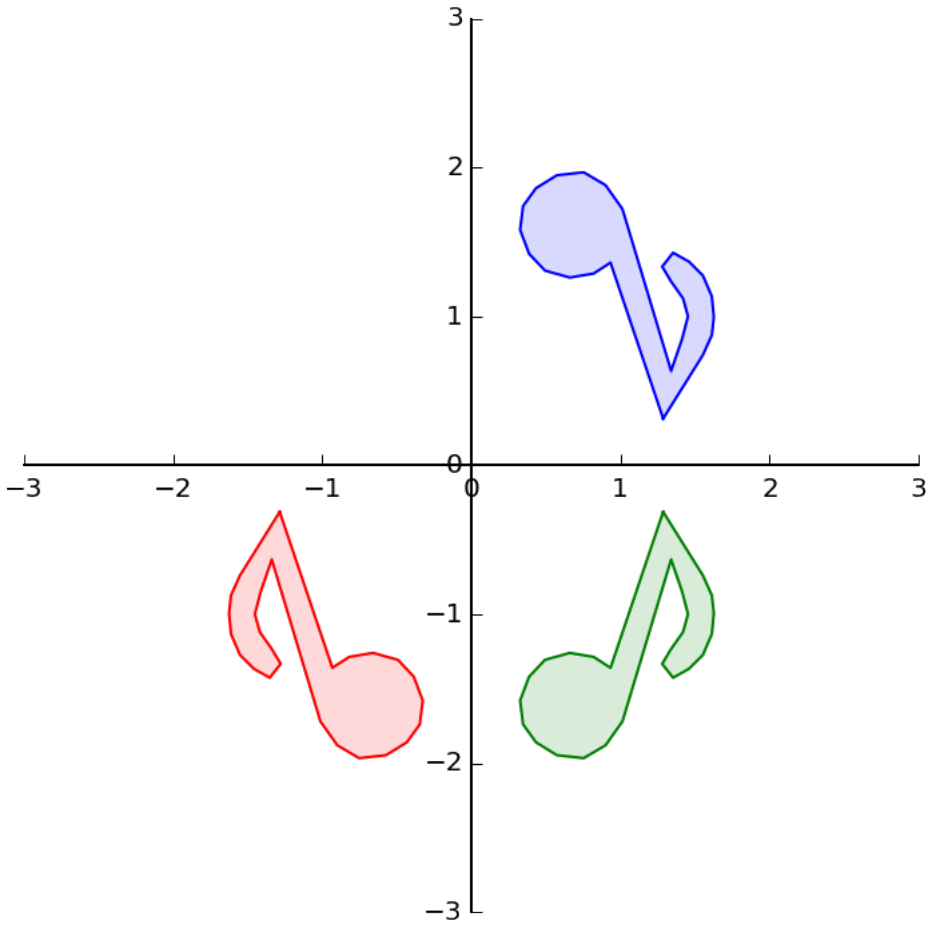
Reflection through the origin



We determined that the matrix $C = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$ implements this linear transformation. But notice that we could have accomplished this another way:

- First reflect through the x_1 axis
- Then reflect through the x_2 axis

```
In [15]: A = np.array(
             [[ 1, 0],
              [ 0,-1]])
         B = np.array(
             [[-1, 0],
              [ 0, 1]])
         dm.plotSetup()
         dm.plotShape(note)
         dm.plotShape(A.dot(note),'g')
         dm.plotShape(B.dot(A.dot(note)),'r')
```



As we saw, to reflect a point through the x_1 axis, we multiply it by matrix $A = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$.

Likewise, to reflect a point through the x_2 axis, we multiply it by matrix $B = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$.

So, another way to reflect point $\mathbf{u}$ through the origin would be:

- $\mathbf{v} = A\mathbf{u}$
- Followed by $\mathbf{w} = B\mathbf{v}$.

In other words, $\mathbf{w} = B(A\mathbf{u})$.

But it is clear that $B(A\mathbf{x})$ and $C\mathbf{x}$ are the *same* linear transformation. So, using C we can go directly to the solution using one multiplication, rather than having to multiply twice (once for A and once for B).

So a natural question is: given A and B, could we find C directly?

In other words, for any A and B, could we find C such that:

$$A(B\mathbf{x}) = C\mathbf{x}?$$

Let's determine how to find C given A and B.

If A is $m \times n$, B is $n \times p$, and $\mathbf{x} \in \mathbb{R}^p$, denote the columns of B by $\mathbf{b_1}, \ldots, \mathbf{b_p}$, and the entries in $\mathbf{x}$ by $x_1, \ldots, x_p$.

Then:

$$B\mathbf{x} = x_1\mathbf{b_1} + \cdots + x_p\mathbf{b_p}.$$

and:

$$A(B\mathbf{x}) = A(x_1\mathbf{b_1} + \cdots + x_p\mathbf{b_p})$$

Since matrix-vector multiplication is a linear transformation:

$$= x_1A\mathbf{b_1} + \cdots + x_pA\mathbf{b_p}.$$

So the vector $A(B\mathbf{x})$ is a linear combination of the vectors $A\mathbf{b_1}, \ldots, A\mathbf{b_p}$, using the entries in $\mathbf{x}$ as weights.

A linear combination of vectors is the same as a matrix-vector multiplication. In matrix terms, this linear combination is written:

$$A(B\mathbf{x}) = [A\mathbf{b_1} \ \ldots \ A\mathbf{b_p}]\mathbf{x}.$$

So this matrix is what we are looking for!

Definition. If A is an $m \times n$ matrix and B is $n \times p$ matrix with columns $\mathbf{b_1}, \ldots, \mathbf{b_p}$, then the product AB is defined as the $m \times p$ matrix whose columns are $A\mathbf{b_1}, \ldots, A\mathbf{b_p}$. That is,

$$AB = A[\mathbf{b_1} \ \ldots \ \mathbf{b_p}] = [A\mathbf{b_1} \ \ldots \ A\mathbf{b_p}].$$

This definition means that for any A and B for which AB is defined, then if $C = AB$,

$$C\mathbf{x} = A(B\mathbf{x}).$$

That is: *multiplication of matrices* corresponds to *composition of linear transformations.*

Note that when $C = AB$, $C\mathbf{x}$ is a vector *in the span of the columns of A.*

Example. Compute AB where $A = \begin{bmatrix} 2 & 3 \\ 1 & -5 \end{bmatrix}$ and $B = \begin{bmatrix} 4 & 3 & 6 \\ 1 & -2 & 3 \end{bmatrix}$.

Solution. Write $B = [\mathbf{b_1} \ \mathbf{b_2} \ \mathbf{b_3}]$, and compute:

$$A\mathbf{b_1} = \begin{bmatrix} 2 & 3 \\ 1 & -5 \end{bmatrix}\begin{bmatrix} 4 \\ 1 \end{bmatrix}, \quad A\mathbf{b_2} = \begin{bmatrix} 2 & 3 \\ 1 & -5 \end{bmatrix}\begin{bmatrix} 3 \\ -2 \end{bmatrix}, \quad A\mathbf{b_3} = \begin{bmatrix} 2 & 3 \\ 1 & -5 \end{bmatrix}\begin{bmatrix} 6 \\ 3 \end{bmatrix},$$

$$= \begin{bmatrix} 11 \\ -1 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 13 \end{bmatrix} \quad \begin{bmatrix} 21 \\ -9 \end{bmatrix}.$$

So:

$$AB = [A\mathbf{b_1} \ A\mathbf{b_2} \ A\mathbf{b_3}] = \begin{bmatrix} 11 & 0 & 21 \\ -1 & 13 & -9 \end{bmatrix}.$$

Example. Verify that reflection through the x_1 axis followed by reflection through the x_2 axis is the same as reflection through the origin.

$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}.$$

Note that this is a valid proof because every linear transformation of vectors is defined by its standard matrix.

Example. If A is a 3×5 matrix, and B is a 5×2 matrix, what are the sizes of AB and BA, if they are defined?

$$\underset{3\times 5}{A} \quad \underset{5\times 2}{B} \quad = \quad \underset{3\times 2}{AB}$$

$$\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix}\begin{bmatrix} * & * \\ * & * \\ * & * \\ * & * \\ * & * \end{bmatrix} = \begin{bmatrix} * & * \\ * & * \\ * & * \end{bmatrix}$$

What about BA?

It is not defined, because the number of columns of B does not match the number of rows of A.

Facts.

If A is $m \times n$, and B is $p \times q$, then AB is defined if and only if $n = p$. If AB is defined, then it is $m \times q$.

$$\begin{array}{ccccc} A & & B & = & AB \\ 3 \times \boxed{5} & & \boxed{5} \times 2 & & 3 \times 2 \end{array}$$

1.2 Inner Product View of Matrix Multiplication.

Recall that the inner product of two vectors or sequences $\mathbf{u}$ and $\mathbf{v}$ is $\sum_k u_k v_k$.

Also recall that one way to define the matrix vector product is $(A\mathbf{x})_i$ = inner product of $\mathbf{x}$ and row i of A.

This immediately shows another way to think of matrix multiplication:

$(AB)_{ij}$ = inner product of row i of A and column j of $B = \sum_k A_{ik} B_{kj}$.

Example. Start with the same matrices as the last example, $A = \begin{bmatrix} 2 & 3 \\ 1 & -5 \end{bmatrix}$ and $B = \begin{bmatrix} 4 & 3 & 6 \\ 1 & -2 & 3 \end{bmatrix}$. Compute the entry in row 1 and column 3 of C.

$$AB = \begin{bmatrix} \boxed{2} & \boxed{3} \\ 1 & -5 \end{bmatrix} \begin{bmatrix} 4 & 3 & \boxed{6} \\ 1 & -2 & \boxed{3} \end{bmatrix} = \begin{bmatrix} * & * & 2(6)+3(3) \\ & * & * \end{bmatrix} = \begin{bmatrix} * & * & 21 \\ & * & * \end{bmatrix}.$$

This agrees with the result of the last example, and we could reproduce the whole solution by repeating this for each element of the result matrix.

1.3 Question Time! Q9.1

1.4 Matrix Algebra

We've defined multiplication of two matrices. What about addition of two matrices?

This is straightfoward: if A and B are the same shape, we get $A + B$ by adding the corresponding elements. (Just like adding vectors.)

That is,

$$(A + B)_{ij} = A_{ij} + B_{ij}.$$

If A and B are not the same shape, $A + B$ is undefined.

Furthermore, we define scalar-matrix multiplication just as for vectors:

$$(rA)_{ij} = r(A_{ij}).$$

So, just as we did for vectors, we can show that the standard properties of addition apply, and that scalar multiplication distributes over addition:

1. $A + B = B + A$
2. $(A + B) + C = A + (B + C)$
3. $A + 0 = A$
4. $r(A + B) = rA + rB$
5. $(r + s)A = rA + sA$
6. $r(sA) = (rs)A$

Furthermore, we find that **some** of the familiar properties of multiplication apply to matrix multiplication (assume that all sums and products are defined):

1. $A(BC) = (AB)C$
    - multiplication of matrices is associative

2. $A(B+C) = AB + AC$
   - multiplication on the left distributes over addition
3. $(B+C)A = BA + CA$
   - multiplication on the right distributes over addition
4. $r(AB) = (rA)B = A(rB)$
   - for any scalar r
5. $IA = A = AI$

Note that property 1 means that we can write ABC without bothering about parentheses.
Now, here is where things get different!

- In general, AB is **not** equal to BA. Multiplication is **not commutative!**
  - Consider $A = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ and $B = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}$.
- In fact, even if AB is defined, BA may not be defined.
  - AB and BA are both defined only if A and B are square.
- On the other hand, sometimes A and B **do** commute.
  - Consider A and B as the reflections through the x_1 and x_2 axis. Then AB and BA both implement reflection through the origin (i.e., the same transformation.) So in this case $AB = BA$.
- You cannot, in general, cancel out matrices in a multiplication. That is, if $AC = AB$, it does not follow that $C = B$.
  - Consider the case where A is the projection onto one of the axes.
- If AB is the zero matrix, you cannot in general conclude that either A or B must be a zero matrix.
  - Consider $A = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ and $B = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$.

Study and remember these rules. You will use them!

1.5 Powers of a Matrix

Equipped now with matrix-matrix multiplication, we can define the powers of a matrix in a straightforward way. For an integer $k > 0$:

$$A^k = \overbrace{A \cdots A}^{k}.$$

Obviously, A must be a square matrix for A^k to be defined.
What should A^0 be?
$A^0\mathbf{x}$ should be the result of multiplying $\mathbf{x}$ with A zero times. So we define $A^0 = I$.

1.6 Question Time! Q9.2

1.7 The Transpose of a Matrix

Given an $m \times n$ matrix A, the *transpose* of A is the matrix we get by interchanging its rows and columns.

It is denoted A^T. Its shape is $n \times m$.

For example, if:

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \quad B = \begin{bmatrix} -5 & 2 \\ 1 & -3 \\ 0 & 4 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -3 & 5 & -2 & 7 \end{bmatrix}$$

Then:

$$A^T = \begin{bmatrix} a & c \\ b & d \end{bmatrix}, \quad B^T = \begin{bmatrix} -5 & 1 & 0 \\ 2 & -3 & 4 \end{bmatrix}, \quad C^T = \begin{bmatrix} 1 & -3 \\ 1 & 5 \\ 1 & -2 \\ 1 & 7 \end{bmatrix}$$

The definition can be stated succinctly:

$$A^T_{ij} = A_{ji}.$$

Rules for Transposes:

1. $(A^T)^T = A$
2. $(A + B)^T = A^T + B^T$
3. For any scalar r, $(rA)^T = r(A^T)$
4. $(AB)^T = B^T A^T$

The first three are pretty obvious.

The last one is a bit different. **Memorize it.** You will use it: the transpose of a product is the product of the transposes **in reverse order**.

1.8 Question Time! Q9.3

Question: For a vector in $\mathbf{x} \in \mathbb{R}^n$, what is $\mathbf{x}^T$?

Answer: For the purposes of the definition, we treat $\mathbf{x}$ as a $n \times 1$ matrix. So its transpose is an $1 \times n$ matrix, i.e., a matrix with a single row.

Question: For two vectors $\mathbf{x}$ and $\mathbf{y}$, what is $\mathbf{x}^T\mathbf{y}$?

Answer: By the definition of matrix-vector multiplication, $\mathbf{x}^T\mathbf{y} = \sum_{i=1}^{n} x_i y_i$.

That is, $\mathbf{x}^T\mathbf{y}$ is the **inner product** of $\mathbf{x}$ and $\mathbf{y}$. This simple construction is a very useful one to remember.

1.9 The Computational Viewpoint

You recall in the last lecture I said that in Python/numpy:

```
C = A.dot(B)
```

was the same as:

```
for i in range(k):
    C[:,k] = AxIP(A, B[:,k])
```

So now you know: `A.dot(B)` is really *matrix multiplication* of `A` and `B.` :)

Matrix multiplication is a mainstay of computing. Thousands of applications rely heavily on matrix multiplication. Some examples include

- Computer graphics and animation
- Google's algorithm for ranking search results
- Modeling mechanical structures such as aircraft and buildings
- Compressing and decompressing audio signals
- Weather modeling and prediction
- Modeling quantum computing

So minimizing the time required to do matrix multiplication is immensely important.

Complexity.

What is the computational complexity of matrix multiplication?

For two $n \times n$ matrices, consider the definition that uses inner product:

$$(AB)_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}.$$

So each element of the product AB requires n multiplications and n additions.

There are n^2 elements of AB, so the overall computation requires

$$2n \cdot n^2 = 2n^3$$

operations.

That's not particularly good news; for two matrices of size 10,000 $\times$ 10,000 (which is not particularly large in practice), this is 2 trillion operations (2 teraflops).

What is the complexity of matrix-vector multiplication?

We know that matrix-vector multiplication requires n inner products of size n. So, it is $2n^2$.

So what is the most efficient way to compute $A^2\mathbf{x}$?

1. First compute A^2, then compute $A^2\mathbf{x}$: $2n^3 + 2n^2$.
2. First compute $A\mathbf{x}$, then compute $A(A\mathbf{x}) = 2 \cdot 2n^2 = 4n^2$.

Parallelization.

Although matrix multiplication is computationally demanding, it has a wonderful property: it is *highly parallel*. That is, the computation needed for each element does not require computing the other elements.

(This is not true, for example, for Gaussian elimination; think about the role of a pivot.)

This means that if we have multiple processors, and each has access to A and B, the work can be divided up very cleanly.

For example, let's say you have n processors. Then each processor can independently compute one column of the result, without needing to know anything about what the other processors are doing.

Specifically, processor i can compute its column as $A\mathbf{b_i}$.

In that case, since all processors are working in parallel, the elapsed time is reduced to $2n^2$.

Even better, say you have n^2 processors. Then each processor can compute a single element of the result, $(AB)_{ij}$ as $\sum_{k=1}^{n} A_{ik} B_{kj}$. Then the elapsed time is reduced to $2n$.

This sort of strategy is used for huge computations like web search and weather modeling.

Libraries.

The importance of matrix multiplication in practice means that very efficient and carefully constructed libraries have been developed for it.

An important issue for high performance is how the matrices are actually laid out in memory, and the order in which matrix elements are accessed.

The premier library is called LAPACK. LAPACK has been developed over the past 40 years and is updated frequently to tune it for new computer hardware.

Python's "numpy" uses LAPACK under the hood for its matrix computations.

Hence, even though Python is an interpreted language, for doing intensive matrix computations it is very fast, just as fast as compiled code.