[3] The Matrix

What is a matrix? Traditional answer

Neo: What is the Matrix?
Trinity: The answer is out there, Neo, and it's looking for you, and it will find you if you want it to. *The Matrix*, 1999

Traditional notion of a matrix: two-dimensional array.

$$\begin{bmatrix} 1 & 2 & 3 \\ 10 & 20 & 30 \end{bmatrix}$$

- Two rows: $[1, 2, 3]$ and $[10, 20, 30]$.
- Three columns: $[1, 10]$, $[2, 20]$, and $[3, 30]$.
- A 2×3 matrix.

For a matrix A, the i, j element of A

- is the element in row i, column j
- is traditionally written $A_{i,j}$
- but we will use $A[i, j]$

List of row-lists, list of column-lists (Quiz)

- One obvious Python representation for a matrix: a list of row-lists:
  $\begin{bmatrix} 1 & 2 & 3 \\ 10 & 20 & 30 \end{bmatrix}$ represented by `[[1,2,3],[10,20,30]]`.

- Another: a list of column-lists:
  $\begin{bmatrix} 1 & 2 & 3 \\ 10 & 20 & 30 \end{bmatrix}$ represented by `[[1,10],[2,20],[3,30]]`.

List of row-lists, list of column-lists

Quiz: Write a nested comprehension whose value is list-of-*row*-list representation of a 3×4 matrix all of whose elements are zero:

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Hint: first write a comprehension for a typical row, then use that expression in a comprehension for the list of lists.

Answer:

```
>>> [[0 for j in range(4)] for i in range(3)]
[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
```

List of row-lists, list of column-lists (Quiz)

Quiz: Write a nested comprehension whose value is list-of-*column*-lists representation of a 3×4 matrix whose i, j element is $i - j$:

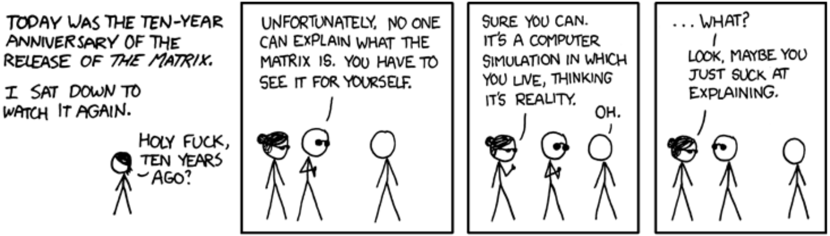
$$\begin{bmatrix} 0 & -1 & -2 & -3 \\ 1 & 0 & -1 & -2 \\ 2 & 1 & 0 & -1 \end{bmatrix}$$

Hint: First write a comprension for column j, assuming j is bound to an integer. Then use that expression in a comprehension in which j is the control variable.

Answer:

```
>>> [[i-j for i in range(3)] for j in range(4)]
[[0, 1, 2], [-1, 0, 1], [-2, -1, 0], [-3, -2, -1]]
```

The matrix revealed



The Matrix Revisited (excerpt) `http://xkcd.com/566/`

Definition: For finite sets R and C, an $R \times C$ *matrix over* $\mathbb{F}$ is a function from $R \times C$ to $\mathbb{F}$.

	@	#	?
a	1	2	3
b	10	20	30

- $R = \{\texttt{a}, \texttt{b}\}$ and $C = \{\texttt{@}, \texttt{\#}, \texttt{?}\}$.
- R is set of row labels
- C is set of column labels

In Python, the function is represented by a dictionary:

```
{('a','@'):1, ('a','#'):2, ('a', '?'):3,
 ('b', '@'):10, ('b',  '#'):20, ('b','?'):30}
```

Rows, columns, and entries

	@	#	?
a	1	2	3
b	10	20	30

Rows and columns are vectors, e.g.

- Row `'a'` is the vector `Vec({'@', '#', '?'}, {'@':1, '#':2, '?':3})`
- Column `'#'` is the vector `Vec({'a','b'}, {'a':2, 'b':20})`

Dict-of-rows/dict-of-columns representations

	@	#	?
a	1	2	3
b	10	20	30

One representation: *dictionary of rows:*

```
{'a': Vec({'#', '@', '?'}, {'@':1, '#':2, '?':3}),
 'b': Vec({'#', '@', '?'}, {'@':10, '#':20, '?':30})}
```

Another representation: *dictionary of columns:*

```
{'@': Vec({'a','b'}, {'a':1, 'b':10}),
 '#': Vec({'a','b'}, {'a':2, 'b':20}),
 '?': Vec({'a','b'}, {'a':3, 'b':30})}
```

Our Python implementation

	@	#	?
a	1	2	3
b	10	20	30

```
>>> M=Mat(({'a','b'}, {'@', '#', '?'}),
            {('a','@'):1, ('a','#'):2,('a','?'):3,
             ('b','@'):10, ('b','#'):20, ('b','?'):30})
```

A class with two fields:

- D, a *pair* (R, C) of sets.
- f, a dictionary representing a function that maps pairs $(r, c) \in R \times C$ to field elements.

```
class Mat:
    def __init__(self, labels, function):
        self.D = labels
        self.f = function
```

We will later add lots of matrix operations to this class.

Identity matrix

```
        a b c
      -------
 a   |  1 0 0
 b   |  0 1 0
 c   |  0 0 1
```

Definition: $D \times D$ *identity matrix* is the matrix $\mathbb{1}_D$ such that $\mathbb{1}_D[k,k] = 1$ for all $k \in D$ and zero elsewhere.

Usually we omit the subscript when D is clear from the context.
Often letter `I` (for "identity") is used instead of $\mathbb{1}$

```
Mat(({'a','b','c'},{'a','b','c'}),{('a','a'):1,('b','b'):1,('c','c'):1})
```

Quiz: Write procedure `identity(`D`)` that returns the $D \times D$ identity matrix over $\mathbb{R}$ represented as an instance of `Mat`.

Answer:

```
>>> def identity(D): return Mat((D,D), {(k,k):1 for k in D})
```

Converting between representations

Converting an instance of `Mat` to a column-dictionary representation:

	@	#	?
a	1	2	3
b	10	20	30

```
Mat(({'a','b'}, {'@', '#', '?'}), {('a','@'):1, ('a','#'):2,
          ('a','?'):3, ('b','@'):10, ('b','#'):20, ('b','?'):30})
```

$\Downarrow$

```
{'@': Vec({'a','b'}, {'a':1, 'b':10}),
 '#': Vec({'a','b'}, {'a':2, 'b':20}),
 '?': Vec({'a','b'}, {'a':3, 'b':30})}
```

Quiz: Write the procedure `mat2coldict(A)` that, given an instance of `Mat`, returns the column-dictionary representation of the same matrix.

Answer:

```
def mat2coldict(A):
 return {c:Vec(A.D[0],{r:A[r,c] for r in A.D[0]}) for c in A.D[1]}
```

Module `matutil`

We provide a module, `matutil`, that defines several conversion routines:

- `mat2coldict(A)`: from a `Mat` to a dictionary of columns represented as `Vec`s)
- `mat2rowdict(A)`: from a `Mat` to a dictionary of rows represented as `Vec`s
- `coldict2mat(coldict)` from a dictionary of columns (or a list of columns) to a `Mat`
- `rowdict2mat(rowdict)`: from a dictionary of rows (or a list of rows) to a `Mat`
- `listlist2mat(L)`: from a list of list of field elements to a `Mat` the inner lists turn into rows

and also:

- `identity(D)`: produce a `Mat` representing the $D \times D$ identity matrix

The `Mat` class

We gave the definition of a rudimentary matrix class:

```
class Mat:
    def __init__(self,
          labels, function):
           self.D = labels
           self.f = function
```

The more elaborate class definition allows for more concise vector code, e.g.

```
>>> M['a', 'B'] = 1.0
>>> b = M*v
>>> B = M*A
>>> print(B)
```

More elaborate version of this class definition allows operator overloading for element access, matrix-vector multiplication, etc.

operation	syntax
Matrix addition and subtraction	`A+B` and `A-B`
Matrix negative	`-A`
Scalar-matrix multiplication	`alpha*A`
Matrix equality test	`A == B`
Matrix transpose	`A.transpose()`
Getting a matrix entry	`A[r,c]`
Setting a matrix entry	`A[r,c] = alpha`
Matrix-vector multiplication	`A*v`
Vector-matrix multiplication	`v*A`
Matrix-matrix multiplication	`A*B`

You will code this class starting from a template we provide.

Using Mat

You will write the bodies of named procedures such as `setitem(M, k, val)` and `matrix_vector_mul(M, v)` and `transpose(M)`.

However, in actually using Mats in other code, you must use operators and methods instead of named procedures, e.g.

```
>>> M['a', 'b'] = 1.0
>>> v = M*u
>>> b_parallel =
      Q*Q.transpose()*b
```

instead of

```
>>> setitem(M, ('a','B'), 1.0)
>>> v = matrix_vector_mul(M, u)
>>> b_parallel =
    matrix_vector_mul(matrix_matrix_mul(Q,
                                  transpose(Q)), b)
```

In fact, in code outside the `mat` module that uses `Mat`, you will import just `Mat` from the `mat` module:

```
from mat import Mat
```

so the named procedures will not be imported into the namespace. Those named procedures in the `mat` module are intended to be used *only* inside the `mat` module itself.

> **In short:** Use the operators [ ], +, *, - and the method `.transpose()` when working with Mats

Assertions in Mat

For each procedure you write, we will provide the stub of the procedure, e.g. for `matrix_vector_mul(M, v)`, we provide the stub

```
def matrix_vector_mul(M, v):
    "Returns the product of matrix M and vector v"
    assert M.D[1] == v.D
    pass
```

You are supposed to replace the pass statement with code for the procedure.

The first line in the body is a documentation string.

The second line is an assertion. It asserts that the second element of the pair `M.D`, the set of column-labels of `M`, must be equal to the domain of the vector `v`. If the procedure is called with arguments that violate this, Python reports an error.

The assertion is there to remind us of a rule about matrix-vector multiplication.

Please keep the assertions in your `mat` code while using it for this course.

Testing Mat with doctests

Because you will use Mat a lot, making sure your implementation is correct will save you from lots of pain later.

Akin to Vec, we have provided doctests

```
def getitem(M, k):
    """
    Returns value of entry k in M
    >>> M = Mat(({1,3,5}, {'a'}),
             {(1,'a'):4, (5,'a'): 2})
    >>> M[1,'a']
    4
    """
    pass
```

You can test each of these examples while running Python in interactive mode by importing Mat from the module mat and then copying the example from the docstring and pasting:

```
>>> from vec import Mat
>>> M = Mat(({1,3,5}, {'a'}), ...
>>> M[1,'a']
4
```

You can also run all the tests at once from the console (outside the Python interpreter) using the following command:

```
python3 -m doctest mat.py
```

This will run the doctests in mat.py, and will print messages about any discrepancies that arise. If your code passes the tests, nothing will be printed.

Column space and row space

One simple role for a matrix: packing together a bunch of columns or rows

Two vector spaces associated with a matrix M:

Definition:

- *column space* of M = Span {columns of M}
  Written Col M

- *row space* of M = Span {rows of M}
  Written Row M

Examples:

- Column space of $\begin{bmatrix} 1 & 2 & 3 \\ 10 & 20 & 30 \end{bmatrix}$ is Span $\{[1, 10], [2, 20], [3, 30]\}$.
  In this case, the span is equal to Span $\{[1, 10]\}$ since $[2, 20]$ and $[3, 30]$ are scalar multiples of $[1, 10]$.

- The row space of the same matrix is Span $\{[1, 2, 3], [10, 20, 30]\}$.
  In this case, the span is equal to Span $\{[1, 2, 3]\}$ since $[10, 20, 30]$ is a scalar multiple of $[1, 2, 3]$.

Transpose

Transpose swaps rows and columns.

```
          @   #   ?
      ----------
a   |    2   1   3
b   |   20  10  30
```

$\longrightarrow$

```
           a   b
       ------
@   |    2  20
#   |    1  10
?   |    3  30
```

Transpose (and Quiz)

Quiz: Write `transpose(M)`

Answer:

```
def transpose(M):
    return Mat((M.D[1], M.D[0]), {(q,p):v for (p,q),v in M.f.items()})
```

Matrices as vectors

Soon we study true matrix operations. But first....

A matrix can be interpreted as a vector:

- an $R \times S$ matrix is a function from $R \times S$ to $\mathbb{F}$,
- so it can be interpreted as an $R \times S$-vector:
  - *scalar-vector multiplication*
  - *vector addition*
- Our full implementation of `Mat` class will include these operations.

Matrix-vector and vector-matrix multiplication

Two ways to multiply a matrix by a vector:

- matrix-vector multiplication
- vector-matrix multiplication

For each of these, two *equivalent definitions*:

- in terms of linear combinations
- in terms of dot-products

Matrix-vector multiplication in terms of linear combinations

Linear-Combinations Definition of matrix-vector multiplication: Let M be an $R \times C$ matrix.

- If $\mathbf{v}$ is a C-vector then

$$M * \mathbf{v} = \sum_{c \in C} \mathbf{v}[c] \text{ (column } c \text{ of } M)$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 10 & 20 & 30 \end{bmatrix} * [7, 0, 4] = 7\,[1, 10] + 0\,[2, 20] + 4\,[3, 30]$$

- If $\mathbf{v}$ is *not* a C-vector then

$$M * \mathbf{v} = \text{ERROR!}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 10 & 20 & 30 \end{bmatrix} * [7, 0] = \text{ERROR!}$$

Matrix-vector multiplication in terms of linear combinations

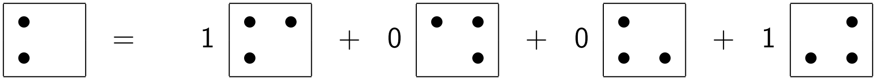
$$\begin{array}{c|ccc} & @ & \# & ? \\ \hline a & 2 & 1 & 3 \\ b & 20 & 10 & 30 \end{array} \; * \; \begin{array}{ccc} @ & \# & ? \\ \hline 0.5 & 5 & -1 \end{array} \; = \; \begin{array}{c|c} a & 3 \\ b & 30 \end{array}$$

$$\begin{array}{c|ccc} & @ & \# & ? \\ \hline a & 2 & 1 & 3 \\ b & 20 & 10 & 30 \end{array} \; * \; \begin{array}{ccc} \% & \# & ? \\ \hline 0.5 & 5 & -1 \end{array} \; = \; \text{ERROR!}$$

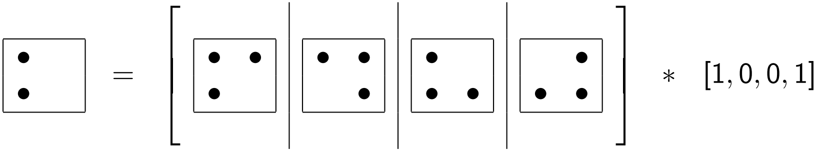
Matrix-vector multiplication in terms of linear combinations: *Lights Out*

A solution to a *Lights Out* configuration is a linear combination of "button vectors."

For example, the linear combination

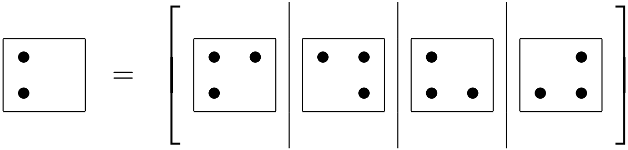


can be written as



Solving a matrix-vector equation: *Lights Out*

$\boxed{\text{Solving an instance of } \textit{Lights Out}} \quad \Rightarrow \quad \boxed{\text{Solving a matrix-vector equation}}$



$* \quad [\alpha_1, \alpha_2, \alpha_3, \alpha_4]$

Solving a matrix-vector equation

Fundamental Computational Problem: *Solving a matrix-vector equation*

- *input:* an $R \times C$ matrix A and an R-vector $\mathbf{b}$
- *output:* the C-vector $\mathbf{x}$ such that $A * \mathbf{x} = \mathbf{b}$

Solving a matrix-vector equation: 2×2 special case

Simple formula to solve

$$\begin{bmatrix} a & c \\ b & d \end{bmatrix} * [x, y] = [p, q]$$

if $ad \neq bc$:

$$x = \frac{dp-cq}{ad-bc} \text{ and } y = \frac{aq-bp}{ad-bc}$$

For example, to solve

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} * [x, y] = [-1, 1]$$

we set

$$x = \frac{4 \cdot -1 - 2 \cdot 1}{1 \cdot 4 - 2 \cdot 3} = \frac{-6}{-2} = 3$$

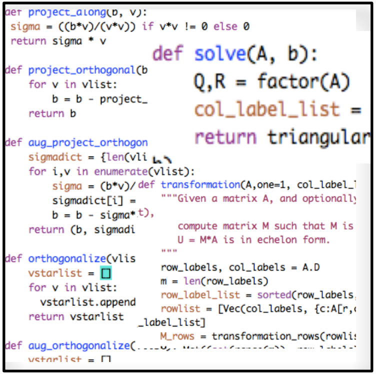
and

$$y = \frac{1 \cdot 1 - 3 \cdot -1}{1 \cdot 4 - 2 \cdot 3} = \frac{4}{-2} = -2$$

Later we study algorithms for more general cases.

The `solver` module

We provide a module `solver` that defines a procedure `solve(A, b)` that tries to find a solution to the matrix-vector equation $A\mathbf{x} = \mathbf{b}$
Currently `solve(A, b)` is a black box



but we will learn how to code it in the coming weeks.

Let's use it to solve this *Lights Out* instance...

Vector-matrix multiplication in terms of linear combinations

Vector-matrix multiplication is different from matrix-vector multiplication:

Let M be an $R \times C$ matrix.
Linear-Combinations Definition of matrix-vector multiplication: If $\mathbf{v}$ is a C-vector then

$$M * \mathbf{v} = \sum_{c \in C} \mathbf{v}[c] \text{ (column } c \text{ of } M)$$

Linear-Combinations Definition of vector-matrix multiplication: If $\mathbf{w}$ is an R-vector then

$$\mathbf{w} * M = \sum_{r \in R} \mathbf{w}[r] \text{ (row } r \text{ of } M)$$

$$[3,4] \quad * \quad \begin{bmatrix} 1 & 2 & 3 \\ 10 & 20 & 30 \end{bmatrix} \quad = \quad 3\,[1,2,3] \quad + \quad 4\,[10,20,30]$$

Vector-matrix multiplication in terms of linear combinations: JunkCo

Let M =

	metal	concrete	plastic	water	electricity
garden gnome	0	1.3	.2	.8	.4
hula hoop	0	0	1.5	.4	.3
slinky	.25	0	0	.2	.7
silly putty	0	0	.3	.7	.5
salad shooter	.15	0	.5	.4	.8

$$\text{total resources used} = [\alpha_{\text{gnome}}, \alpha_{\text{hoop}}, \alpha_{\text{slinky}}, \alpha_{\text{putty}}, \alpha_{\text{shooter}}] * \text{M}$$

Suppose we know total resources used and we know M.

To find the values of $\alpha_{\text{gnome}}, \alpha_{\text{hoop}}, \alpha_{\text{slinky}}, \alpha_{\text{putty}}, \alpha_{\text{shooter}}$,

solve a *vector-matrix* equation $\mathbf{b} = \mathbf{x} * M$
where **b** is vector of total resources used.

Solving a matrix-vector equation

Fundamental Computational Problem: *Solving a matrix-vector equation*

- *input:* an $R \times C$ matrix A and an R-vector $\mathbf{b}$
- *output:* the C-vector $\mathbf{x}$ such that $A * \mathbf{x} = \mathbf{b}$

If we had an algorithm for solving a *matrix-vector* equation,
could also use it to solve a *vector-matrix* equation,
using transpose.

The `solver` module, and floating-point arithmetic

For arithmetic over $\mathbb{R}$, Python uses floats, so round-off errors occur:

```
>>> 10.0**16 + 1 == 10.0**16
True
```

Consequently algorithms such as that used in `solve(A, b)` do not find exactly correct solutions.

To see if solution $\mathbf{u}$ obtained is a reasonable solution to $A * \mathbf{x} = \mathbf{b}$, see if the vector $\mathbf{b} - A * \mathbf{u}$ has entries that are close to zero:

```
>>> A = listlist2mat([[1,3],[5,7]])
>>> u = solve(A, b)
>>> b - A*u
Vec({0, 1},{0: -4.440892098500626e-16, 1: -8.881784197001252e-16})
```

The vector $\mathbf{b} - A * \mathbf{u}$ is called the *residual*. Easy way to test if entries of the residual are close to zero: compute the dot-product of the residual with itself:

```
>>> res = b - A*u
>>> res * res
9.860761315262648e-31
```

Checking the output from `solve(A, b)`

For some matrix-vector equations $A * \mathbf{x} = \mathbf{b}$, there is no solution.
In this case, the vector returned by `solve(A, b)` gives rise to a largeish residual:

```
>>> A = listlist2mat([[1,2],[4,5],[-6,1]])
>>> b = list2vec([1,1,1])
>>> u = solve(A, b)
>>> res = b - A*u
>>> res * res
0.24287856071964012
```

Later in the course we will see that the residual is, in a sense, as small as possible.

Some matrix-vector equations are *ill-conditioned*, which can prevent an algorithm using floats from getting even approximate solutions, even when solutions exists:

```
>>> A = listlist2mat([[1e20,1],[1,0]])
>>> b = list2vec([1,1])
>>> u = solve(A, b)
>>> b - A*u
Vec({0, 1},{0: 0.0, 1: 1.0})
```

We will not study conditioning in this course.

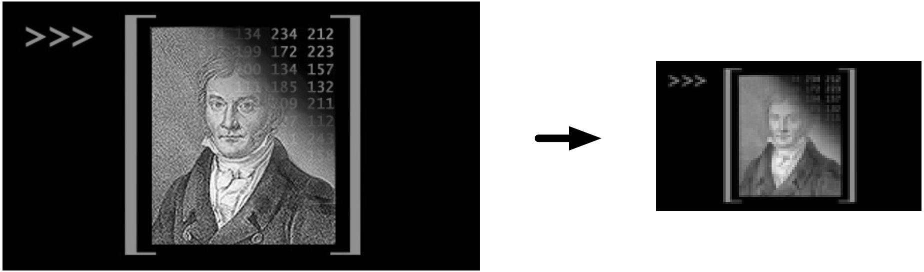
Matrix-vector multiplication in terms of dot-products

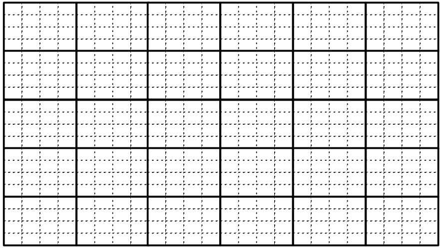
Let M be an $R \times C$ matrix.

Dot-Product Definition of matrix-vector multiplication: $M * \mathbf{u}$ is the R-vector $\mathbf{v}$ such that $\mathbf{v}[r]$ is the dot-product of row r of M with $\mathbf{u}$.

$$
\begin{aligned}
\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 10 & 0 \end{bmatrix} \quad * \quad [3,-1] \quad &= \quad [\quad [1,2]\cdot[3,-1], \quad [3,4]\cdot[3,-1], \quad [10,0]\cdot[3,-1] \quad] \\
&= \quad [1,5,30]
\end{aligned}
$$

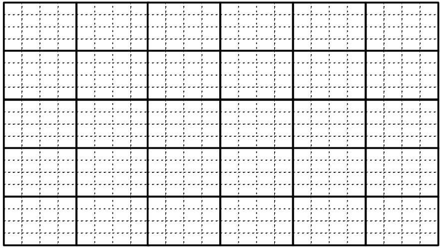
Applications of dot-product definition of matrix-vector multiplication: Downsampling





- Each pixel of the low-res image corresponds to a little grid of pixels of the high-res image.
- The intensity value of a low-res pixel is the *average* of the intensity values of the corresponding high-res pixels.

Applications of dot-product definition of matrix-vector multiplication: Downsampling



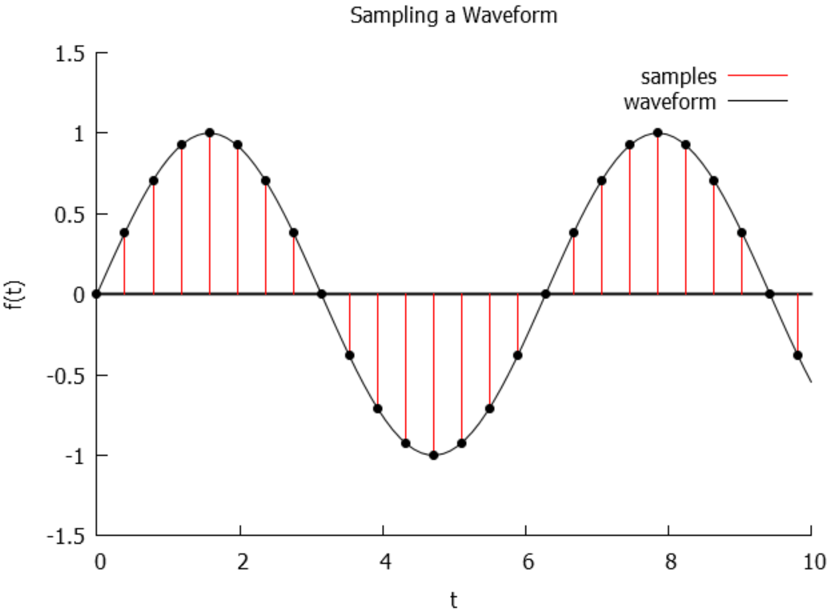
- Each pixel of the low-res image corresponds to a little grid of pixels of the high-res image.
- The intensity value of a low-res pixel is the *average* of the intensity values of the corresponding high-res pixels.

- Averaging can be expressed as dot-product.
- We want to compute a dot-product for each low-res pixel.
- Can be expressed as matrix-vector multiplication.

Applications of dot-product definition of matrix-vector multiplication: blurring

- To blur a face, replace each pixel in face with average of pixel intensities in its neighborhood.
- Average can be expressed as dot-product.
- By dot-product definition of matrix-vector multiplication, can express this image transformation as a matrix-vector product.
- Gaussian blur: a kind of weighted average

Applications of dot-product definition of matrix-vector multiplication: Audio search



Applications of dot-product definition of matrix-vector multiplication: Audio search

Lots of dot-products!

5	-6	9	-9	-5	-9	-5	5	-8	-5	-9	9	8	-5	-9	6	-2	-4	-9	-1	-1	-9	-3
2	7	4	-3	0	-1	-6	4	5	-8	-9												

5	-6	9	-9	-5	-9	-5	5	-8	-5	-9	9	8	-5	-9	6	-2	-4	-9	-1	-1	-9	-3
	2	7	4	-3	0	-1	-6	4	5	-8	-9											

5	-6	9	-9	-5	-9	-5	5	-8	-5	-9	9	8	-5	-9	6	-2	-4	-9	-1	-1	-9	-3
		2	7	4	-3	0	-1	-6	4	5	-8	-9										

5	-6	9	-9	-5	-9	-5	5	-8	-5	-9	9	8	-5	-9	6	-2	-4	-9	-1	-1	-9	-3
			2	7	4	-3	0	-1	-6	4	5	-8	-9									

5	-6	9	-9	-5	-9	-5	5	-8	-5	-9	9	8	-5	-9	6	-2	-4	-9	-1	-1	-9	-3
				2	7	4	-3	0	-1	-6	4	5	-8	-9								

5	-6	9	-9	-5	-9	-5	5	-8	-5	-9	9	8	-5	-9	6	-2	-4	-9	-1	-1	-9	-3
					2	7	4	-3	0	-1	-6	4	5	-8	-9							

5	-6	9	-9	-5	-9	-5	5	-8	-5	-9	9	8	-5	-9	6	-2	-4	-9	-1	-1	-9	-3
						2	7	4	-3	0	-1	-6	4	5	-8	-9						

Applications of dot-product definition of matrix-vector multiplication: Audio search

Lots of dot-products!

- Represent as a matrix-vector product.
- One row per dot-product.

To search for $[0, 1, -1]$ in $[0, 0, -1, 2, 3, -1, 0, 1, -1, -1]$:

$$\begin{bmatrix} 0 & 0 & -1 \\ 0 & -1 & 2 \\ -1 & 2 & 3 \\ 2 & 3 & -1 \\ 3 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & -1 \\ 1 & -1 & -1 \end{bmatrix} \quad * \quad [0, 1, -1]$$

Formulating a system of linear equations as a matrix-vector equation

Recall the *sensor node* problem:

- In each of several test periods, measure total power consumed:

$$\beta_1, \beta_2, \beta_3, \beta_4, \beta_5$$

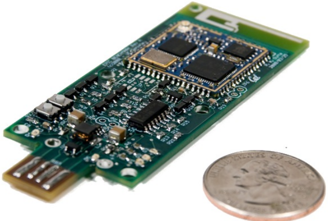
- For each test period, have a vector specifying how long each hardware component was operating during that period:

$$\mathbf{duration}_1, \mathbf{duration}_2, \mathbf{duration}_3, \mathbf{duration}_4, \mathbf{duration}_5$$

- Use measurements to calculate energy consumed per second by each hardware component.

  Formulate as system of linear equations

$$\begin{aligned}
\mathbf{duration}_1 \cdot \mathbf{x} &= \beta_1 \\
\mathbf{duration}_2 \cdot \mathbf{x} &= \beta_2 \\
\mathbf{duration}_3 \cdot \mathbf{x} &= \beta_3 \\
\mathbf{duration}_4 \cdot \mathbf{x} &= \beta_4 \\
\mathbf{duration}_5 \cdot \mathbf{x} &= \beta_5
\end{aligned}$$



Formulating a system of linear equations as a matrix-vector equation

Linear equations

$$\begin{aligned}
\mathbf{a}_1 \cdot \mathbf{x} &= \beta_1 \\
\mathbf{a}_2 \cdot \mathbf{x} &= \beta_2 \\
&\vdots \\
\mathbf{a}_m \cdot \mathbf{x} &= \beta_m
\end{aligned}$$

Each equation specifies the value of a dot-product.

Rewrite as

$$\begin{bmatrix} \mathbf{a}_1 \\ \hline \mathbf{a}_2 \\ \hline \vdots \\ \hline \mathbf{a}_m \end{bmatrix} * \mathbf{x} = [\beta_1, \beta_2, \ldots, \beta_m]$$

Matrix-vector equation for sensor node

Define `D = {'radio', 'sensor', 'memory', 'CPU'}`.

Goal: Compute a D-vector $\mathbf{u}$ that, for each hardware component, gives the current drawn by that component.

Four test periods:

- total milliampere-seconds in these test periods $\mathbf{b} = [140, 170, 60, 170]$
- for each test period, vector specifying how long each hardware device was operating:
  - $\mathbf{duration}_1 =$ `Vec(D, 'radio':.1, 'CPU':.3)`
  - $\mathbf{duration}_2 =$ `Vec(D, 'sensor':.2, 'CPU':.4)`
  - $\mathbf{duration}_3 =$ `Vec(D, 'memory':.3, 'CPU':.1)`
  - $\mathbf{duration}_4 =$ `Vec(D, 'memory':.5, 'CPU':.4)`

To get $\mathbf{u}$, solve $A * \mathbf{x} = \mathbf{b}$ where $A = \left[\begin{array}{c} \mathbf{duration}_1 \\ \hline \mathbf{duration}_2 \\ \hline \mathbf{duration}_3 \\ \hline \mathbf{duration}_4 \end{array}\right]$

Triangular matrix

Recall: We considered *triangular* linear systems, e.g.

$$
\begin{array}{rcl}
[\ 1,\ 0.5,\ -2,\ 4\ ] \cdot \mathbf{x} & = & -8 \\
[\ 0,\ 3,\ 3,\ 2\ ] \cdot \mathbf{x} & = & 3 \\
[\ 0,\ 0,\ 1,\ 5\ ] \cdot \mathbf{x} & = & -4 \\
[\ 0,\ 0,\ 0,\ 2\ ] \cdot \mathbf{x} & = & 6 \\
[\ 0,\ 0,\ 0,\ 2\ ] \cdot \mathbf{x} & = & 6
\end{array}
$$

We can rewrite this linear system as a matrix-vector equation:

$$
\begin{bmatrix} 1 & 0.5 & -2 & 4 \\ 0 & 3 & 3 & 2 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 2 \end{bmatrix} * \mathbf{x} = [-8, 3, -4, 6]
$$

The matrix is a *triangular* matrix.

Definition: An $n \times n$ *upper triangular* matrix A is a matrix with the property that $A_{ij} = 0$ for $j > i$. Note that the entries forming the triangle can be be zero or nonzero.

We can use backward substitution to solve such a matrix-vector equation.

Triangular matrices will play an important role later.

Computing sparse matrix-vector product

To compute matrix-vector or vector-matrix product,

- could use dot-product or linear-combinations definition.
  (You'll do that in homework.)
- However, using those definitions, it's not easy to exploit sparsity in the matrix.

"Ordinary" Definition of Matrix-Vector Multiplication: If M is an $R \times C$ matrix and $\mathbf{u}$ is a C-vector then $M * \mathbf{u}$ is the R-vector $\mathbf{v}$ such that, for each $r \in R$,

$$v[r] = \sum_{c \in C} M[r, c] u[c]$$

Computing sparse matrix-vector product

"Ordinary" Definition of Matrix-Vector Multiplication: If M is an $R \times C$ matrix and $\mathbf{u}$ is a C-vector then $M * \mathbf{u}$ is the R-vector $\mathbf{v}$ such that, for each $r \in R$,

$$v[r] = \sum_{c \in C} M[r,c]u[c]$$

Obvious method:

```
1 for i in R:
2     v[i] := ∑_{j∈C} M[i,j]u[j]
```

But this doesn't exploit sparsity!

Idea:

- Initialize output vector $\mathbf{v}$ to zero vector.
- Iterate over nonzero entries of M, adding terms according to ordinary definition.

```
1 initialize v to zero vector
2 for each pair (i,j) in sparse representation,
3     v[i] = v[i] + M[i,j]u[j]
```

Matrix-matrix multiplication

If

- A is a $R \times S$ matrix, and
- B is a $S \times T$ matrix

then it is legal to multiply A times B.

- In Mathese, written AB
- In our `Mat` class, written `A*B`

AB is different from BA.

In fact, one product might be legal while the other is illegal.

Matrix-matrix multiplication

We'll see two equivalent definitions:

- one in terms of vector-matrix multiplication,
- one in terms of matrix-vector multiplication.

Matrix-matrix multiplication: vector-matrix definition

Vector-matrix definition of matrix-matrix multiplication:
For each row-label r of A,

$$\text{row } r \text{ of } AB = \underbrace{(\text{row } r \text{ of } A)}_{\text{vector}} * B$$

$$\begin{bmatrix} 1 & 0 & 0 \\ \hline 2 & 1 & 0 \\ \hline 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} & & \\ & \text{B} & \\ & & \end{bmatrix} = \begin{bmatrix} [1,0,0] * B \\ \hline [2,1,0] * B \\ \hline [0,0,1] * B \end{bmatrix}$$

How to interpret $[1,0,0] * B$?

- *Linear combinations* definition of vector-matrix multiplication?
- *Dot-product* definition of vector-matrix multiplication?

Each is correct.

Matrix-matrix multiplication: vector-matrix interpretation

$$\begin{bmatrix} 1 & 0 & 0 \\ \hline 2 & 1 & 0 \\ \hline 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} & & \\ & \text{B} & \\ & & \end{bmatrix} = \begin{bmatrix} [1,0,0] * B \\ \hline [2,1,0] * B \\ \hline [0,0,1] * B \end{bmatrix}$$

How to interpret $[1,0,0] * B$? *Linear combinations* definition:

$$[1,0,0] * \begin{bmatrix} \mathbf{b}_1 \\ \hline \mathbf{b}_2 \\ \hline \mathbf{b}_3 \end{bmatrix} = \mathbf{b}_1 \qquad [0,0,1] * \begin{bmatrix} \mathbf{b}_1 \\ \hline \mathbf{b}_2 \\ \hline \mathbf{b}_3 \end{bmatrix} = \mathbf{b}_3$$

$$[2,1,0] * \begin{bmatrix} \mathbf{b}_1 \\ \hline \mathbf{b}_2 \\ \hline \mathbf{b}_3 \end{bmatrix} = 2\,\mathbf{b}_1 + \mathbf{b}_2$$

Conclusion:

$$\begin{bmatrix} 1 & 0 & 0 \\ \hline 2 & 1 & 0 \\ \hline 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{b}_1 \\ \hline \mathbf{b}_2 \\ \hline \mathbf{b}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \hline 2\,\mathbf{b}_1 + \mathbf{b}_2 \\ \hline \mathbf{b}_3 \end{bmatrix}$$

Matrix-matrix multiplication: vector-matrix interpretation

Conclusion:

$$\begin{bmatrix} 1 & 0 & 0 \\ \hline 2 & 1 & 0 \\ \hline 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{b}_1 \\ \hline \mathbf{b}_2 \\ \hline \mathbf{b}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \hline 2\,\mathbf{b}_1 + \mathbf{b}_2 \\ \hline \mathbf{b}_3 \end{bmatrix}$$

We call $\begin{bmatrix} 1 & 0 & 0 \\ \hline 2 & 1 & 0 \\ \hline 0 & 0 & 1 \end{bmatrix}$ an *elementary row-addition matrix*.

Matrix-matrix multiplication: matrix-vector definition

Matrix-vector definition of matrix-matrix multiplication:
For each column-label s of B,

$$\text{column } s \text{ of } AB = A * (\text{column } s \text{ of } B)$$

Let $A = \begin{bmatrix} 1 & 2 \\ -1 & 1 \end{bmatrix}$ and $B =$ matrix with columns $[4, 3]$, $[2, 1]$, and $[0, -1]$

$$B = \left[\begin{array}{c|c|c} 4 & 2 & 0 \\ 3 & 1 & -1 \end{array}\right]$$

AB is the matrix with column $i = A * (\text{ column } i \text{ of } B)$

$A * [4, 3] = [10, -1]$ $\quad$ $A * [2, 1] = [4, -1]$ $\quad$ $A * [0, -1] = [-2, -1]$

$$AB = \left[\begin{array}{c|c|c} 10 & 4 & -2 \\ -1 & -1 & -1 \end{array}\right]$$

Matrix-matrix multiplication: Dot-product definition

Combine

- *matrix-vector* definition of matrix-matrix multiplication, and
- *dot-product* definition of matrix-vector multiplication

to get...

Dot-product definition of matrix-matrix multiplication:
Entry rc of AB is the dot-product of row r of A with column c of B.

Example:

$$\left[\begin{array}{ccc} 1 & 0 & 2 \\ \hline 3 & 1 & 0 \\ \hline 2 & 0 & 1 \end{array}\right] \left[\begin{array}{c|c} 2 & 1 \\ 5 & 0 \\ 1 & 3 \end{array}\right] = \left[\begin{array}{cc} [1,0,2]\cdot[2,5,1] & [1,0,2]\cdot[1,0,3] \\ [3,1,0]\cdot[2,5,1] & [3,1,0]\cdot[1,0,3] \\ [2,0,1]\cdot[2,5,1] & [2,0,1]\cdot[1,0,3] \end{array}\right] = \left[\begin{array}{cc} 4 & 7 \\ 11 & 3 \\ 5 & 5 \end{array}\right]$$

Matrix-matrix multiplication: transpose

$$(AB)^T = B^T A^T$$

Example:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 5 & 0 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 7 & 4 \\ 19 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 5 & 0 \\ 1 & 2 \end{bmatrix}^T \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^T = \begin{bmatrix} 5 & 1 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} = \begin{bmatrix} 7 & 19 \\ 4 & 8 \end{bmatrix}$$

You might think "$(AB)^T = A^T B^T$" but this is **false**.
In fact, doesn't even make sense!

- For AB to be legal, A's column labels $=$ B's row labels.
- For $A^T B^T$ to be legal, A's row labels $=$ B's column labels.

Example: $\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \begin{bmatrix} 6 & 7 \\ 8 & 9 \end{bmatrix}$ is legal but $\begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix} \begin{bmatrix} 6 & 8 \\ 7 & 9 \end{bmatrix}$ is not.

Matrix-matrix multiplication: Column vectors

Multiplying a matrix A by a one-column matrix B

$$\begin{bmatrix} & & \\ & A & \\ & & \end{bmatrix} \begin{bmatrix} \\ \mathbf{b} \\ \\ \end{bmatrix}$$

By matrix-vector definition of matrix-matrix multiplication, result is matrix with one column: $A * \mathbf{b}$

This shows that matrix-vector multiplication is subsumed by matrix-matrix multiplication.

Convention: Interpret a vector $\mathbf{b}$ as a one-column matrix ("column vector")

- Write vector $[1, 2, 3]$ as $\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$
- Write $A * [1, 2, 3]$ as $\begin{bmatrix} & & \\ & A & \\ & & \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$ or $A\,\mathbf{b}$

Matrix-matrix multiplication: Row vectors

If we interpret vectors as one-column matrices.... what about vector-matrix multiplication?

Use transpose to turn a column vector into a row vector: Suppose $\mathbf{b} = [1, 2, 3]$.

$$[1, 2, 3] * A = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} & & \\ & A & \\ & & \end{bmatrix} = \mathbf{b}^T A$$

Algebraic properties of matrix-vector multiplication

Proposition: Let A be an $R \times C$ matrix.

- For any C-vector $\mathbf{v}$ and any scalar α,

$$A * (\alpha\, \mathbf{v}) = \alpha\, (A * \mathbf{v})$$

- For any C-vectors $\mathbf{u}$ and $\mathbf{v}$,

$$A * (\mathbf{u} + \mathbf{v}) = A * \mathbf{u} + A * \mathbf{v}$$

Algebraic properties of matrix-vector multiplication

To prove

$$A * (\alpha\, \mathbf{v}) = \alpha\, (A * \mathbf{v})$$

we need to show corresponding entries are equal:

Need to show

$$\text{entry } i \text{ of} \quad A * (\alpha\, \mathbf{v}) = \text{entry } i \text{ of} \quad \alpha\, (A * \mathbf{v})$$

Proof: Write $A = \begin{bmatrix} \mathbf{a}_1 \\ \hline \vdots \\ \hline \mathbf{a}_m \end{bmatrix}$.

By dot-product def. of matrix-vector mult,

$$\begin{aligned} \text{entry } i \text{ of} \quad A * (\alpha\, \mathbf{v}) &= \mathbf{a}_i \cdot \alpha\, \mathbf{v} \\ &= \alpha\, (\mathbf{a}_i \cdot \mathbf{v}) \end{aligned}$$

by homogeneity of dot-product

By definition of scalar-vector multiply,

$$\begin{aligned} \text{entry } i \text{ of } \alpha\, (A * \mathbf{v}) &= \alpha\, (\text{entry } i \text{ of } A * \mathbf{v}) \\ &= \alpha\, (\mathbf{a}_i \cdot \mathbf{v}) \end{aligned}$$

by dot-product definition of matrix-vector multiply

QED

Algebraic properties of matrix-vector multiplication

To prove

$$A * (\mathbf{u} + \mathbf{v}) = A * \mathbf{u} + A * \mathbf{v}$$

we need to show corresponding entries are equal:

Need to show

$$\text{entry } i \text{ of } \quad A * (\mathbf{u} + \mathbf{v}) = \text{entry } i \text{ of } \quad A * \mathbf{u} + A * \mathbf{v}$$

Proof: Write $A = \left[\begin{array}{c} \mathbf{a}_1 \\ \hline \vdots \\ \hline \mathbf{a}_m \end{array}\right]$.

By dot-product def. of matrix-vector mult,

$$\begin{aligned} \text{entry } i \text{ of } \quad A * (\mathbf{u} + \mathbf{v}) &= \mathbf{a}_i \cdot (\mathbf{u} + \mathbf{v}) \\ &= \mathbf{a}_i \cdot \mathbf{u} + \mathbf{a}_i \cdot \mathbf{v} \end{aligned}$$

by distributive property of dot-product

By dot-product def. of matrix-vector mult,

$$\begin{aligned} \text{entry } i \text{ of } \quad A * \mathbf{u} &= \mathbf{a}_i \cdot \mathbf{u} \\ \text{entry } i \text{ of } \quad A * \mathbf{v} &= \mathbf{a}_i \cdot \mathbf{v} \end{aligned}$$

so

$$\text{entry } i \text{ of } \quad A * \mathbf{u} + A * \mathbf{v} = \mathbf{a}_i \cdot \mathbf{u} + \mathbf{a}_i \cdot \mathbf{v}$$

QED

Null space of a matrix

Definition: *Null space* of a matrix A is $\{\mathbf{u} \ : \ A * \mathbf{u} = \mathbf{0}\}$. Written *Null* A

Example:

$$\begin{bmatrix} 1 & 2 & 4 \\ 2 & 3 & 9 \end{bmatrix} * [0,0,0] = [0,0]$$

so the null space includes $[0,0,0]$

$$\begin{bmatrix} 1 & 2 & 4 \\ 2 & 3 & 9 \end{bmatrix} * [6,-1,-1] = [0,0]$$

so the null space includes $[6,-1,-1]$

By dot-product definition,

$$\left[\begin{array}{c} \mathbf{a}_1 \\ \hline \vdots \\ \hline \mathbf{a}_m \end{array}\right] * \mathbf{u} = [\mathbf{a}_1 \cdot \mathbf{u}, \ \ldots, \ \mathbf{a}_m \cdot \mathbf{u}]$$

Null space of a matrix

We just saw:

Null space of a matrix $\begin{bmatrix} \mathbf{a}_1 \\ \hline \vdots \\ \hline \mathbf{a}_m \end{bmatrix}$

equals the solution set of the homogeneous linear system $\boxed{\begin{array}{rcl} \mathbf{a}_1 \cdot \mathbf{x} & = & 0 \\ & \vdots & \\ \mathbf{a}_m \cdot \mathbf{x} & = & 0 \end{array}}$

This shows: $\boxed{\textit{Null space of a matrix is a vector space.}}$

Can also show it directly, using algebraic properties of matrix-vector multiplication:

Property V1: Since $A * \mathbf{0} = \mathbf{0}$,the null space of A contains $\mathbf{0}$

Property V2: if $\mathbf{u} \in \text{Null } A$ then $A * (\alpha\, \mathbf{u}) = \alpha\, (A * \mathbf{u}) = \alpha\, \mathbf{0} = \mathbf{0}$ so $\alpha \mathbf{u} \in \text{Null } A$

Property V3: If $\mathbf{u} \in \text{Null } A$ and $\mathbf{v} \in \text{Null } A$
then $A * (\mathbf{u} + \mathbf{v}) = A * \mathbf{u} + A * \mathbf{v} = \mathbf{0} + \mathbf{0} = \mathbf{0}$
so $\mathbf{u} + \mathbf{v} \in \text{Null } A$

Null space of a matrix

Definition: *Null space* of a matrix A is $\{\mathbf{u} \; : \; A * \mathbf{u} = \mathbf{0}\}$. Written *Null A*

Proposition: Null space of a matrix is a vector space.

Example:

$$\text{Null} \begin{bmatrix} 1 & 2 & 4 \\ 2 & 3 & 9 \end{bmatrix} = \text{Span } \{[6, -1, -1]\}$$

Solution space of a matrix-vector equation

Earlier, we saw:

If $\mathbf{u}_1$ is a solution to the linear system $\boxed{\begin{array}{rcl} \mathbf{a}_1 \cdot \mathbf{x} & = & \beta_1 \\ & \vdots & \\ \mathbf{a}_m \cdot \mathbf{x} & = & \beta_m \end{array}}$

then the solution set is $\mathbf{u}_1 + \mathcal{V}$,

where $\mathcal{V} =$ solution set of $\boxed{\begin{array}{rcl} \mathbf{a}_1 \cdot \mathbf{x} & = & 0 \\ & \vdots & \\ \mathbf{a}_m \cdot \mathbf{x} & = & 0 \end{array}}$

Restated: If $\mathbf{u}_1$ is a solution to $A * \mathbf{x} = \mathbf{b}$ then solution set is $\mathbf{u}_1 + \mathcal{V}$ where $\mathcal{V} = \text{Null } A$

Solution space of a matrix-vector equation

Proposition: If $\mathbf{u}_1$ is a solution to $A * \mathbf{x} = \mathbf{b}$ then solution set is $\mathbf{u}_1 + \mathcal{V}$ where $\mathcal{V} = \text{Null } A$

Example:

- Null space of $\begin{bmatrix} 1 & 2 & 4 \\ 2 & 3 & 9 \end{bmatrix}$ is Span $\{[6, -1, -1]\}$.
- One solution to $\boxed{\begin{bmatrix} 1 & 2 & 4 \\ 2 & 3 & 9 \end{bmatrix} * \mathbf{x} = [1, 1]}$ is $x = [-1, 1, 0]$.
- Therefore solution set is $[-1, 1, 0] +$ Span $\{[6, -1, -1]\}$
- For example, solutions include
  - $[-1, 1, 0] + [0, 0, 0]$
  - $[-1, 1, 0] + [6, -1, -1]$
  - $[-1, 1, 0] + 2\,[6, -1, -1]$

$$\vdots$$

Solution space of a matrix-vector equation

Proposition: If $\mathbf{u}_1$ is a solution to $A * \mathbf{x} = \mathbf{b}$ then solution set is $\mathbf{u}_1 + \mathcal{V}$ where $\mathcal{V} = \text{Null } A$

- If $\mathcal{V}$ is a trivial vector space then $\mathbf{u}_1$ is the only solution.
- If $\mathcal{V}$ is not trivial then $\mathbf{u}_1$ is *not* the only solution.

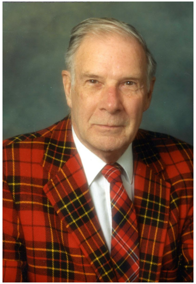
Corollary: $A * \mathbf{x} = \mathbf{b}$ has at most one solution iff Null A is a trivial vector space.

Question: How can we tell if the null space of a matrix is trivial?

Answer comes later...

Error-correcting codes

- Originally inspired by errors in reading programs on punched cards
- Now used in WiFi, cell phones, communication with satellites and spacecraft, digital television, RAM, disk drives, flash memory, CDs, and DVDs

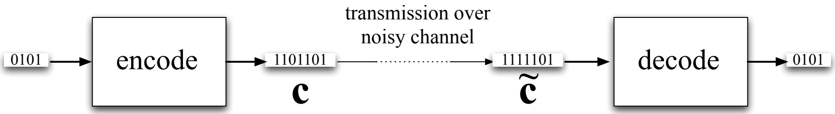


Richard Hamming

Hamming code is a *linear binary block code*:

- *linear* because it is based on linear algebra,
- *binary* because the input and output are assumed to be in binary, and
- *block* because the code involves a fixed-length sequence of bits.

Error-correcting codes: Block codes



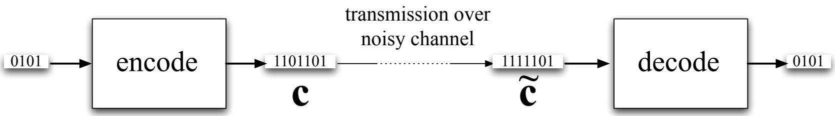
To protect an 4-bit block:

- Sender *encodes* 4-bit block as a 7-bit block $\mathbf{c}$
- Sender transmits $\mathbf{c}$
- $\mathbf{c}$ passes through noisy channel—errors might be introduced.
- Receiver receives 7-bit block $\tilde{\mathbf{c}}$
- Receiver tries to figure out original 4-bit block

The 7-bit encodings are called *codewords*.

$\mathcal{C}$ = set of permitted codewords

Error-correcting codes: Linear binary block codes



Hamming's first code is a *linear* code:

- Represent 4-bit and 7-bit blocks as 4-vectors and 7-vectors over $GF(2)$.
- 7-bit block received is $\tilde{\mathbf{c}} = \mathbf{c} + \mathbf{e}$
- $\mathbf{e}$ has 1's in positions where noisy channel flipped a bit
  ($\mathbf{e}$ is the *error vector*)
- **Key idea:** set $\mathcal{C}$ of codewords is the null space of a matrix H.

This makes Receiver's job easier:

- Receiver has $\tilde{\mathbf{c}}$, needs to figure out $\mathbf{e}$.
- Receiver multiplies $\tilde{\mathbf{c}}$ by H.

$$H * \tilde{\mathbf{c}} = H * (\mathbf{c} + \mathbf{e}) = H * \mathbf{c} + H * \mathbf{e} = \mathbf{0} + H * \mathbf{e} = H * \mathbf{e}$$

- Receiver must calculate $\mathbf{e}$ from the value of $H * \mathbf{e}$. *How?*

Hamming Code

In the Hamming code, the codewords are 7-vectors, and

$$H = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Notice anything special about the columns and their order?

- Suppose that the noisy channel introduces at most one bit error.
- Then $\mathbf{e}$ has only one 1.
- Can you determine the position of the bit error from the matrix-vector product $H * \mathbf{e}$?

Example: Suppose $\mathbf{e}$ has a 1 in its third position, $\mathbf{e} = [0, 0, 1, 0, 0, 0, 0]$.

Then $H * \mathbf{e}$ is the third column of H, which is $[0, 1, 1]$.

As long as $\mathbf{e}$ has at most one bit error, the position of the bit can be determined from $H * \mathbf{e}$. This shows that the Hamming code allows the recipient to correct one-bit errors.

Hamming code

$$H = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Quiz: Show that the Hamming code does not allow the recipient to correct two-bit errors: give two different error vectors, $\mathbf{e}_1$ and $\mathbf{e}_2$, each with at most two 1's, such that $H * \mathbf{e}_1 = H * \mathbf{e}_2$.

Answer: There are many acceptable answers. For example, $\mathbf{e}_1 = [1, 1, 0, 0, 0, 0, 0]$ and $\mathbf{e}_2 = [0, 0, 1, 0, 0, 0, 0]$ or $\mathbf{e}_1 = [0, 0, 1, 0, 0, 1, 0]$ and $\mathbf{e}_2 = [0, 1, 0, 0, 0, 0, 1]$.

Matrices and their functions

Now we study the relationship between a matrix M and the function $\mathbf{x} \mapsto M * \mathbf{x}$

- *Easy:* Going from a matrix M to the function $\mathbf{x} \mapsto M * \mathbf{x}$
- *A little harder:* Going from the function $\mathbf{x} \mapsto M * \mathbf{x}$ to the matrix M.

In studying this relationship, we come up with the fundamental notion of a *linear function*.

From matrix to function

Starting with a M, define the function $f(\mathbf{x}) = M * x$.

Domain and co-domain?
If M is an $R \times C$ matrix over $\mathbb{F}$ then

- domain of f is $\mathbb{F}^C$
- co-domain of f is $\mathbb{F}^R$

Example: Let M be the matrix

	#	@	?
a	1	2	3
b	10	20	30

and define $f(\mathbf{x}) = M * \mathbf{x}$

- Domain of f is $\mathbb{R}^{\{\#, @, ?\}}$.
- Co-domain of f is $\mathbb{R}^{\{a, b\}}$.

f maps

#	@	?
2	2	-2

to

a	b
0	0

Example: Define $f(\mathbf{x}) = \begin{bmatrix} 1 & 2 & 3 \\ 10 & 20 & 30 \end{bmatrix} * \mathbf{x}$.

- Domain of f is $\mathbb{R}^3$
- Co-domain of f is $\mathbb{R}^2$

f maps $[2, 2, -2]$ to $[0, 0]$

From function to matrix

We have a function $f : \mathbb{F}^A \longrightarrow \mathbb{F}^B$

We want to compute matrix M such that $f(\mathbf{x}) = M * \mathbf{x}$.

- Since the domain is $\mathbb{F}^A$, we know that the input $\mathbf{x}$ is an A-vector.
- For the product $M * \mathbf{x}$ to be legal, we need the column-label set of M to be A.
- Since the co-domain is $\mathbb{F}^B$, we know that the output $f(\mathbf{x}) = M * \mathbf{x}$ is B-vector.
- To achieve that, we need row-label set of M to be B.

Now we know that M must be a $B \times A$ matrix....

... but what about its entries?

From function to matrix

- We have a function $f : \mathbb{F}^n \longrightarrow \mathbb{F}^m$
- We think there is an $m \times n$ matrix M such that $f(\mathbf{x}) = M * \mathbf{x}$

How to go from the function f to the entries of M?

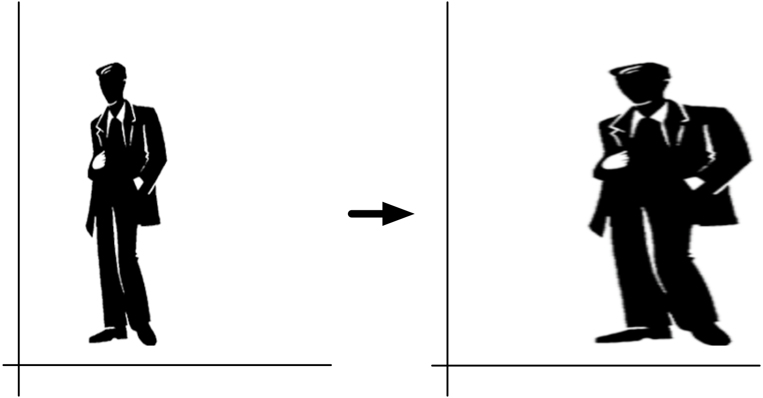
- Write mystery matrix in terms of its columns: $M = \left[\begin{array}{c|c|c} \mathbf{v}_1 & \cdots & \mathbf{v}_n \end{array}\right]$
- Use standard generators $\mathbf{e}_1 = [1, 0, \ldots, 0, 0], \ldots, \mathbf{e}_n = [0, \ldots, 0, 1]$ with *linear-combinations* definition of matrix-vector multiplication:

$$f(\mathbf{e}_1) = \left[\begin{array}{c|c|c} \mathbf{v}_1 & \cdots & \mathbf{v}_n \end{array}\right] * [1, 0, \ldots, 0, 0] = \mathbf{v}_1$$

$$\vdots$$

$$f(\mathbf{e}_n) = \left[\begin{array}{c|c|c} \mathbf{v}_1 & \cdots & \mathbf{v}_n \end{array}\right] * [0, 0, \ldots, 0, 1] = \mathbf{v}_n$$

From function to matrix: horizontal scaling

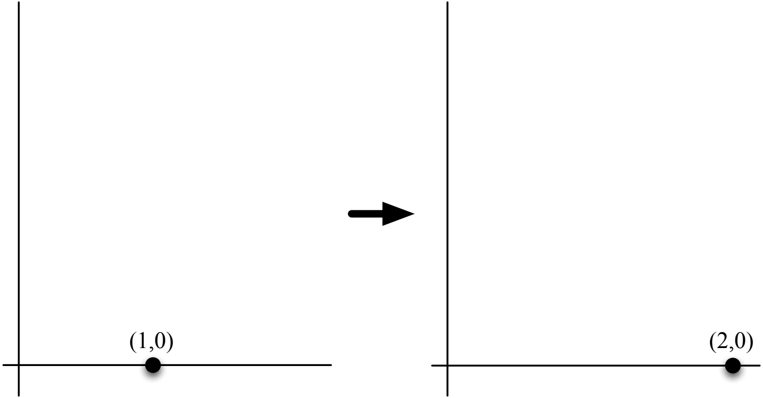


Define $s([x, y])$ = stretching by two in horizontal direction
Assume $s([x, y]) = M * [x, y]$ for some matrix M.

- We know $s([1, 0]) = [2, 0]$ because we are stretching by two in horizontal direction
- We know $s([0, 1]) = [0, 1]$ because no change in vertical direction.

Therefore $M = \left[\begin{array}{c|c} 2 & 0 \\ 0 & 1 \end{array}\right]$

From function to matrix: horizontal scaling

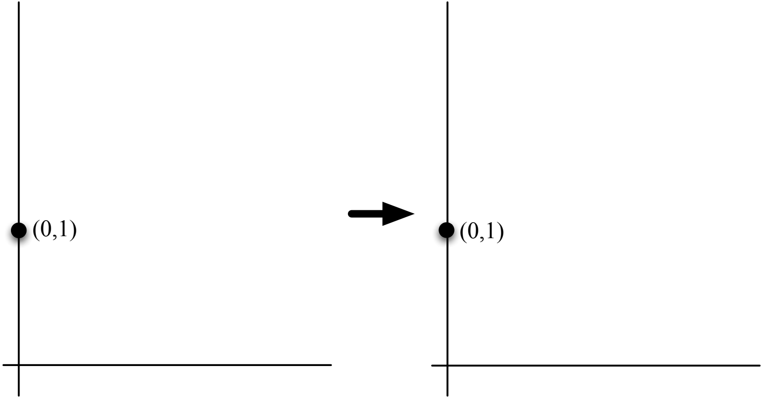


Define $s([x, y])$ = stretching by two in horizontal direction
Assume $s([x, y]) = M * [x, y]$ for some matrix M.

- We know $s([1, 0]) = [2, 0]$ because we are stretching by two in horizontal direction
- We know $s([0, 1]) = [0, 1]$ because no change in vertical direction.

Therefore $M = \left[\begin{array}{c|c} 2 & 0 \\ 0 & 1 \end{array}\right]$

From function to matrix: horizontal scaling



Define $s([x, y])$ = stretching by two in horizontal direction
Assume $s([x, y]) = M * [x, y]$ for some matrix M.

- We know $s([1, 0]) = [2, 0]$ because we are stretching by two in horizontal direction
- We know $s([0, 1]) = [0, 1]$ because no change in vertical direction.

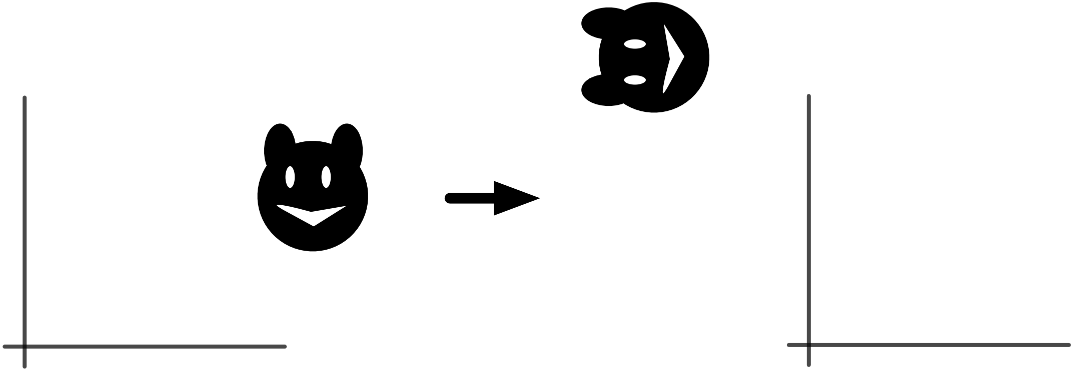
Therefore $M = \left[\begin{array}{c|c} 2 & 0 \\ 0 & 1 \end{array}\right]$

From function to matrix: rotation by 90 degrees

Define $r([x,y]) =$ rotation by 90 degrees
Assume $r([x,y]) = M * [x,y]$ for some matrix M.

- We know rotating $[1,0]$ should give $[0,1]$ so $r([1,0]) = [0,1]$
- We know rotating $[0,1]$ should give $[-1,0]$ so $r([0,1]) = [-1,0]$

Therefore $M = \left[\begin{array}{c|c} 0 & -1 \\ 1 & 0 \end{array}\right]$

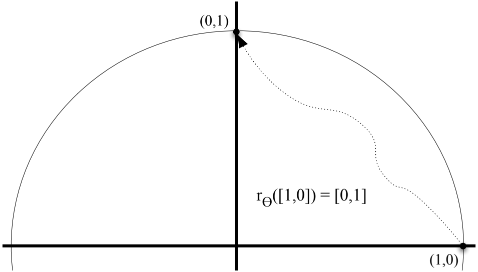


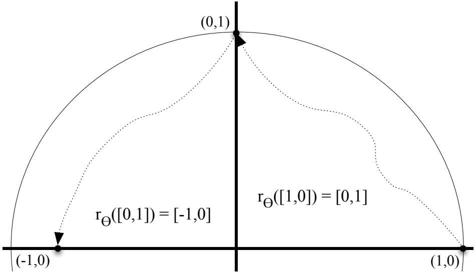
From function to matrix: rotation by 90 degrees

Define $r([x, y])$ = rotation by 90 degrees
Assume $r([x, y]) = M * [x, y]$ for some matrix M.

- We know rotating $[1, 0]$ should give $[0, 1]$ so $r([1, 0]) = [0, 1]$
- We know rotating $[0, 1]$ should give $[-1, 0]$ so $r([0, 1]) = [-1, 0]$

Therefore $M = \left[\begin{array}{c|c} 0 & -1 \\ 1 & 0 \end{array}\right]$



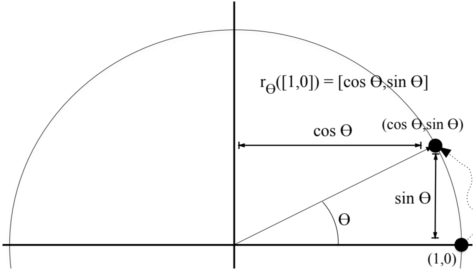


From function to matrix: rotation by θ degrees

Define $r([x, y]) =$ rotation by θ.
Assume $r([x, y]) = M * [x, y]$ for some matrix M.

- We know $r([1, 0]) = [\cos\theta, \sin\theta]$ so column 1 is $[\cos\theta, \sin\theta]$
- We know $r([0, 1]) = [-\sin\theta, \cos\theta]$ so column 2 is $[-\sin\theta, \cos\theta]$

Therefore $M = \left[\begin{array}{c|c} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{array}\right]$

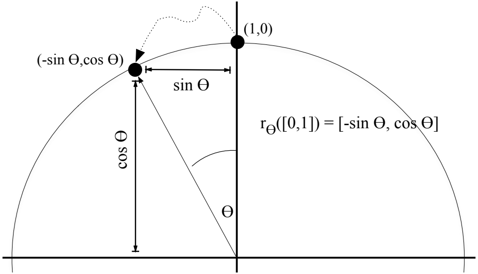


From function to matrix: rotation by θ degrees

Define $r([x, y])$ = rotation by θ.
Assume $r([x, y]) = M * [x, y]$ for some matrix M.

- We know $r([1, 0]) = [\cos\theta, \sin\theta]$ so column 1 is $[\cos\theta, \sin\theta]$
- We know $r([0, 1]) = [-\sin\theta, \cos\theta]$ so column 2 is $[-\sin\theta, \cos\theta]$

Therefore $M = \left[\begin{array}{c|c} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{array}\right]$



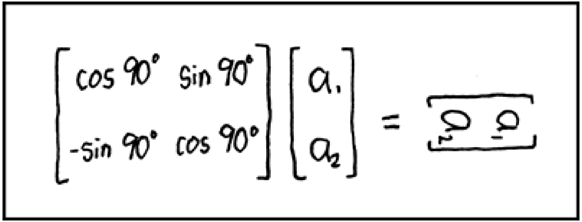
From function to matrix: rotation by θ degrees

Define $r([x, y])$ = rotation by θ.
Assume $r([x, y]) = M * [x, y]$ for some matrix M.

- We know $r([1, 0]) = [\cos\theta, \sin\theta]$ so column 1 is $[\cos\theta, \sin\theta]$
- We know $r([0, 1]) = [-\sin\theta, \cos\theta]$ so column 2 is $[-\sin\theta, \cos\theta]$

Therefore $M = \left[\begin{array}{c|c} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{array}\right]$

For clockwise rotation by 90 degrees, plug in θ = -90 degrees...



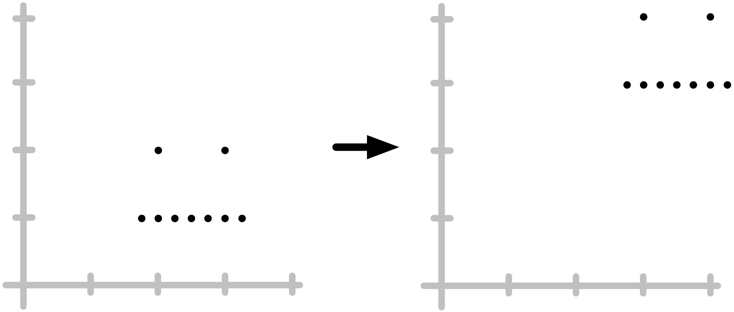
Matrix Transform (`http://xkcd.com/824`)

From function to matrix: translation

$t([x, y])$ = translation by $[1, 2]$. Assume $t([x, y]) = M * [x, y]$ for some matrix M.

- We know $t([1, 0]) = [2, 2]$ so column 1 is $[2, 2]$.
- We know $t([0, 1]) = [1, 3]$ so column 2 is $[1, 3]$.

Therefore $M = \left[\begin{array}{c|c} 2 & 1 \\ 2 & 3 \end{array}\right]$

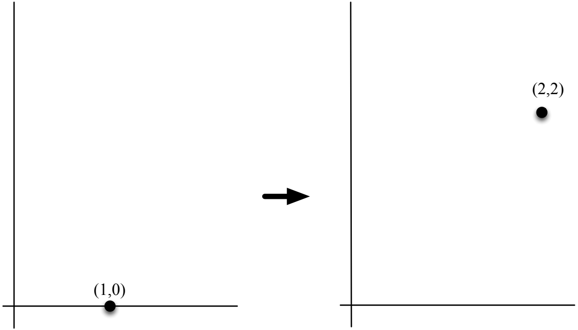


From function to matrix: translation

$t([x,y])$ = translation by $[1,2]$. Assume $t([x,y]) = M * [x,y]$ for some matrix M.

- We know $t([1,0]) = [2,2]$ so column 1 is $[2,2]$.
- We know $t([0,1]) = [1,3]$ so column 2 is $[1,3]$.

Therefore $M = \left[\begin{array}{c|c} 2 & 1 \\ 2 & 3 \end{array}\right]$

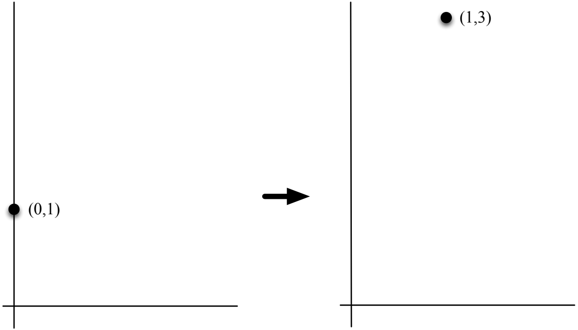


From function to matrix: translation

$t([x,y]) =$ translation by $[1,2]$. Assume $t([x,y]) = M * [x,y]$ for some matrix M.

- We know $t([1,0]) = [2,2]$ so column 1 is $[2,2]$.
- We know $t([0,1]) = [1,3]$ so column 2 is $[1,3]$.

Therefore $M = \left[\begin{array}{c|c} 2 & 1 \\ 2 & 3 \end{array}\right]$



From function to matrix: identity function

Consider the function $f : \mathbb{R}^4 \longrightarrow \mathbb{R}^4$ defined by $f(\mathbf{x}) = \mathbf{x}$
This is the identity function on $\mathbb{R}^4$.

Assume $f(\mathbf{x}) = M * \mathbf{x}$ for some matrix M.

Plug in the standard generators
$\mathbf{e}_1 = [1, 0, 0, 0], \mathbf{e}_2 = [0, 1, 0, 0], \mathbf{e}_3 = [0, 0, 1, 0], \mathbf{e}_4 = [0, 0, 0, 1]$

- $f(\mathbf{e}_1) = \mathbf{e}_1$ so first column is $\mathbf{e}_1$
- $f(\mathbf{e}_2) = \mathbf{e}_2$ so second column is $\mathbf{e}_2$
- $f(\mathbf{e}_3) = \mathbf{e}_3$ so third column is $\mathbf{e}_3$
- $f(\mathbf{e}_4) = \mathbf{e}_4$ so fourth column is $\mathbf{e}_4$

So $M = \left[\begin{array}{c|c|c|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right]$

Identity function $f(\mathbf{x})$ corresponds to identity matrix $\mathbb{1}$

Diagonal matrices

Let $d_1, \ldots, d_n$ be real numbers. Let $f : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ be the function such that $f([x_1, \ldots, x_n]) = [d_1 x_1, \ldots, d_n x_n]$. The matrix corresponding to this function is

$$\begin{bmatrix} d_1 & & \\ & \ddots & \\ & & d_n \end{bmatrix}$$

Such a matrix is called a *diagonal* matrix because the only entries allowed to be nonzero form a diagonal.

> **Definition:** For a domain D, a $D \times D$ matrix M is a *diagonal* matrix if $M[r, c] = 0$ for every pair $r, c \in D$ such that $r \neq c$.

Special case: $d_1 = \cdots = d_n = 1$. In this case, $f(\mathbf{x}) = \mathbf{x}$ (*identity function*)

The matrix $\begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix}$ is an identity matrix.

Linear functions: Which functions can be expressed as a matrix-vector product?

In each example, we *assumed* the function could be expressed as a matrix-vector product.

How can we verify that assumption?

We'll state two algebraic properties.

- If a function can be expressed as a matrix-vector product $\mathbf{x} \mapsto M * \mathbf{x}$, it has these properties.
- If the function from $\mathbb{F}^C$ to $\mathbb{F}^R$ has these properties, it can be expressed as a matrix-vector product.

Linear functions: Which functions can be expressed as a matrix-vector product?

Let $\mathcal{V}$ and $\mathcal{W}$ be vector spaces over a field $\mathbb{F}$.

Suppose a function $f : \mathcal{V} \longrightarrow \mathcal{W}$ satisfies two properties:

Property L1: For every vector $\mathbf{v}$ in $\mathcal{V}$ and every scalar α in $\mathbb{F}$,

$$f(\alpha\,\mathbf{v}) = \alpha\, f(\mathbf{v})$$

Property L2: For every two vectors $\mathbf{u}$ and $\mathbf{v}$ in $\mathcal{V}$,

$$f(\mathbf{u} + \mathbf{v}) = f(\mathbf{u}) + f(\mathbf{v})$$

We then call f a *linear function*.

Proposition: Let M be an $R \times C$ matrix, and suppose $f : \mathbb{F}^C \mapsto \mathbb{F}^R$ is defined by $f(\mathbf{x}) = M * \mathbf{x}$. Then f is a linear function.

Proof: Certainly $\mathbb{F}^C$ and $\mathbb{F}^R$ are vector spaces.

We showed that $M * (\alpha\,\mathbf{v}) = \alpha\, M * \mathbf{v}$. This proves that f satisfies Property L1.

We showed that $M * (\mathbf{u} + \mathbf{v}) = M * \mathbf{u} + M * \mathbf{v}$. This proves that f satisfies Property L2.

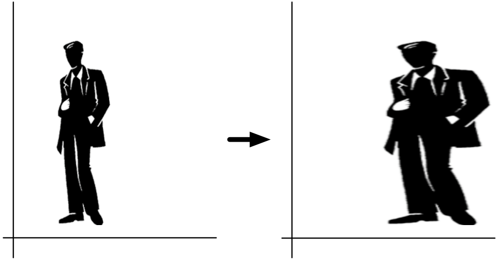
QED

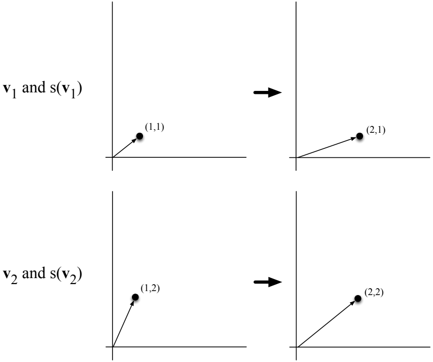
Which functions are linear?

Define $s([x, y])$ = stretching by two in horizontal direction

Property L1: $s(\mathbf{v}_1 + \mathbf{v}_2) = s(\mathbf{v}_1) + s(\mathbf{v}_2)$

Property L2: $s(\alpha\, \mathbf{v}) = \alpha\, s(\mathbf{v})$





Since the function $s(\cdot)$ satisfies Properties L1 and L2, it is a linear function.

Similarly can show rotation by θ degrees is a linear function.

What about translation?

$$t([x, y]) = [x, y] + [1, 2]$$

This function violates Property L1. For example:

$$t([4, 5] + [2, -1]) = t([6, 4]) = [7, 6]$$

but

$$t([4, 5]) + t([2, -1]) = [5, 7] + [3, 1] = [8, 8]$$

A linear function maps zero vector to zero vector

Lemma: If $f : \mathcal{U} \longrightarrow \mathcal{V}$ is a linear function then f maps the zero vector of $\mathcal{U}$ to the zero vector of $\mathcal{V}$.

Proof: Let $\mathbf{0}$ denote the zero vector of $\mathcal{U}$, and let $\mathbf{0}_{\mathcal{V}}$ denote the zero vector of $\mathcal{V}$.

$$f(\mathbf{0}) = f(\mathbf{0} + \mathbf{0}) = f(\mathbf{0}) + f(\mathbf{0})$$

Subtracting $f(\mathbf{0})$ from both sides, we obtain

$$\mathbf{0}_{\mathcal{V}} = f(\mathbf{0})$$

QED

Linear functions: Pushing linear combinations through the function

Defining properties of linear functions:

Property L1: $f(\alpha\,\mathbf{v}) = \alpha\, f(\mathbf{v})$

Property L2: $f(\mathbf{u}+\mathbf{v}) = f(\mathbf{u}) + f(\mathbf{v})$

Proposition: For a linear function f,
for any vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ in the domain of f and any scalars $\alpha_1, \ldots, \alpha_n$,

$$f(\alpha_1\,\mathbf{v}_1 + \cdots + \alpha_n\,\mathbf{v}_n) = \alpha_1\, f(\mathbf{v}_1) + \cdots + \alpha_n\, f(\mathbf{v}_n)$$

Proof: Consider the case of $n = 2$.

$$\begin{aligned} f(\alpha_1\,\mathbf{v}_1 + \alpha_2\,\mathbf{v}_2) &= f(\alpha_1\,\mathbf{v}_1) + f(\alpha_2\,\mathbf{v}_2) && \text{by Property L2} \\ &= \alpha_1\, f(\mathbf{v}_1) + \alpha_2\, f(\mathbf{v}_2) && \text{by Property L1} \end{aligned}$$

Proof for general n is similar. QED

Linear functions: Pushing linear combinations through the function

Proposition: For a linear function f,
$f(\alpha_1\,\mathbf{v}_1 + \cdots + \alpha_n\,\mathbf{v}_n) = \alpha_1\,f(\mathbf{v}_1) + \cdots + \alpha_n\,f(\mathbf{v}_n)$

Example: $f(\mathbf{x}) = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} * \mathbf{x}$

Verify that $f(10\,[1,-1] + 20\,[1,0]) = 10\,f([1,-1]) + 20\,f([1,0])$

$$\begin{aligned}
&\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}\left(10\,[1,-1] + 20\,[1,0]\right) \\
= \quad &\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}\left([10,-10] + [20,0]\right) \\
= \quad &\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}[30,-10] \\
= \quad &30\,[1,3] - 10[2,4] \\
= \quad &[30,90] - [20,40] \\
= \quad &[10,50]
\end{aligned}$$

$$\begin{aligned}
&10\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} * [1,-1]\right) + 20\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} * [1,0]\right) \\
= \quad &10\,([1,3] - [2,4]) + 20\,(1[1,3]) \\
= \quad &10\,[-1,-1] + 20\,[1,3] \\
= \quad &[-10,-10] + [20,60] \\
= \quad &[10,50]
\end{aligned}$$

From function to matrix, revisited

We saw a method to derive a matrix from a function:

Given a function $f : \mathbb{R}^n \longrightarrow \mathbb{R}^m$, we want a matrix M such that $f(\mathbf{x}) = M * \mathbf{x}$....

- Plug in the standard generators $\mathbf{e}_1 = [1, 0, \ldots, 0, 0], \ldots, \mathbf{e}_n = [0, \ldots, 0, 1]$
- Column i of M is $f(\mathbf{e}_i)$.

This works correctly whenever such a matrix M really exists:

Proof: If there is such a matrix then f is linear:

- (Property L1) $f(\alpha\, \mathbf{v}) = \alpha\, f(\mathbf{v})$ and
- (Property L2) $f(\mathbf{u} + \mathbf{v}) = f(\mathbf{u}) + f(\mathbf{v})$

Let $\mathbf{v} = [\alpha_1, \ldots, \alpha_n]$ be any vector in $\mathbb{R}^n$.

We can write $\mathbf{v}$ in terms of the standard generators.

$$
\begin{aligned}
\mathbf{v} &= \alpha_1\, \mathbf{e}_1 + \cdots + \alpha_n\, \mathbf{e}_n \\
\text{so} & \\
f(\mathbf{v}) &= f(\alpha_1\, \mathbf{e}_1 + \cdots + \alpha_n\, \mathbf{e}_n) \\
&= \alpha_1\, f(\mathbf{e}_1) + \cdots + \alpha_n\, f(\mathbf{e}_n) \\
&= \alpha_1\, (\text{column } 1 \text{ of } M) + \cdots + \alpha_n\, (\text{column} n \text{ of } M) \\
&= M * \mathbf{v} \qquad\qquad \textit{QED}
\end{aligned}
$$

Linear functions and zero vectors: Kernel

Definition: *Kernel* of a linear function f is $\{\mathbf{v} \,:\, f(\mathbf{v}) = \mathbf{0}\}$

Written Ker f

For a function $f(\mathbf{x}) = M * \mathbf{x}$,

$$\text{Ker } f = \text{Null } M$$

Kernel and one-to-one

One-to-One Lemma: A linear function is one-to-one if and only if its kernel is a trivial vector space.

Proof: Let $f : \mathcal{U} \longrightarrow \mathcal{V}$ be a linear function. We prove two directions.

- Suppose Ker f contains some nonzero vector $\mathbf{u}$, so $f(\mathbf{u}) = \mathbf{0}_{\mathcal{V}}$.
  Because a linear function maps zero to zero, $f(\mathbf{0}) = \mathbf{0}_{\mathcal{V}}$ as well,
  so f is not one-to-one.

- Suppose Ker $f = \{\mathbf{0}\}$.
  Let $\mathbf{v}_1, \mathbf{v}_2$ be any vectors such that $f(\mathbf{v}_1) = f(\mathbf{v}_2)$.
  Then $f(\mathbf{v}_1) - f(\mathbf{v}_2) = \mathbf{0}_{\mathcal{V}}$
  so, by linearity, $f(\mathbf{v}_1 - \mathbf{v}_2) = \mathbf{0}_{\mathcal{V}}$,
  so $\mathbf{v}_1 - \mathbf{v}_2 \in$ Ker f.
  Since Ker f consists solely of $\mathbf{0}$,
  it follows that $\mathbf{v}_1 - \mathbf{v}_2 = \mathbf{0}$, so $\mathbf{v}_1 = \mathbf{v}_2$.

QED

Kernel and one-to-one

One-to-One Lemma A linear function is one-to-one if and only if its kernel is a trivial vector space.

Define the function $f(\mathbf{x}) = A * \mathbf{x}$.

If Ker f is trivial (i.e. if Null A is trivial)

then a vector $\mathbf{b}$ is the image under f of at most one vector.

That is, at most one vector $\mathbf{u}$ such that $A * \mathbf{u} = \mathbf{b}$
That is, the solution set of $A * \mathbf{x} = \mathbf{b}$ has at most one vector.

Linear functions that are onto?

Question: How can we tell if a linear function is onto?

Recall: for a function $f : \mathcal{V} \longrightarrow \mathcal{W}$, the *image* of f is the set of all images of elements of the domain:

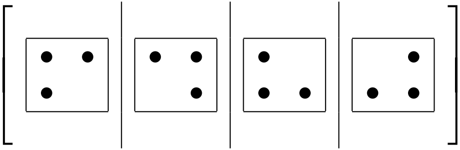
$$\{f(\mathbf{v}) \ : \mathbf{v} \in \mathcal{V}\}$$

(You might know it as the "range" but we avoid that word here.)

The image of function f is written $\mathrm{Im}\ f$

"Is function f is onto?" same as "is $\mathrm{Im}\ f$ = co-domain of f?"

Example: *Lights Out*

Define $f([\alpha_1, \alpha_2, \alpha_3, \alpha_4]) =$  $* [\alpha_1, \alpha_2, \alpha_3, \alpha_4]$

$\mathrm{Im}\ f$ is set of configurations for which 2×2 *Lights Out* can be solved,
so "f is onto" means "2×2 *Lights Out* can be solved for every configuration"

Can 2×2 *Lights Out* be solved for every configuration? What about 5×5?
Each of these questions amounts to asking whether a certain function is onto.

Linear functions that are onto?

$\boxed{\text{"Is function } f \text{ is onto?" same as "is Im } f = \text{co-domain of } f\text{?"}}$

First step in understanding how to tell if a linear function f is onto:

- study the image of f

Proposition: The image of a linear function $f : \mathcal{V} \longrightarrow \mathcal{W}$ is a vector space

The image of a linear function is a vector space

Proposition: The image of a linear function $f : \mathcal{V} \longrightarrow \mathcal{W}$ is a vector space

Recall: a set $\mathcal{U}$ of vectors is a vector space if

V1: $\mathcal{U}$ contains a zero vector,

V2: for every vector $\mathbf{w}$ in $\mathcal{U}$ and every scalar α, the vector $\alpha\,\mathbf{w}$ is in $\mathcal{U}$

V3: for every pair of vectors $\mathbf{w}_1$ and $\mathbf{w}_2$ in $\mathcal{U}$, the vector $\mathbf{w}_1 + \mathbf{w}_2$ is in $\mathcal{U}$

Proof:

V1: Since the domain $\mathcal{V}$ contains a zero vector $\mathbf{0}_{\mathcal{V}}$ and $f(\mathbf{0}_{\mathcal{V}}) = \mathbf{0}_{\mathcal{W}}$, the image of f includes $\mathbf{0}_{\mathcal{W}}$. This proves Property V1.

V2: Suppose some vector $\mathbf{w}$ is in the image of f.
That means there is some vector $\mathbf{v}$ in the domain $\mathcal{V}$ that maps to $\mathbf{w}$: $f(\mathbf{v}) = \mathbf{w}$.
By Property L1, for any scalar α, $f(\alpha\,\mathbf{v}) = \alpha\, f(\mathbf{v}) = \alpha\,\mathbf{w}$
so $\alpha\,\mathbf{w}$ is in the image. This proves Property V2.

V3: Suppose vectors $\mathbf{w}_1$ and $\mathbf{w}_2$ are in the image of f.
That is, there are vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ in the domain such that $f(\mathbf{v}_1) = \mathbf{w}_1$ and $f(\mathbf{v}_2) = \mathbf{w}_2$.
By Property L2, $f(\mathbf{v}_1 + \mathbf{v}_2) = f(\mathbf{v}_1) + f(\mathbf{v}_2) = \mathbf{w}_1 + \mathbf{w}_2$
so $\mathbf{w}_1 + \mathbf{w}_2$ is in the image. This proves Property V3. QED

Linear functions that are onto?

We've shown

$\boxed{\textbf{Proposition:} \text{ The image of a linear function } f : \mathcal{V} \longrightarrow \mathcal{W} \text{ is a vector space}}$

This proposition shows that, for a linear function f, Im f is always a subspace of the co-domain $\mathcal{W}$.

The function is onto if Im f includes all of $\mathcal{W}$.

In a couple of weeks we will have a way to tell.

Inner product

Let $\mathbf{u}$ and $\mathbf{v}$ be two D-vectors interpreted as matrices (column vectors).
Matrix-matrix product $\mathbf{u}^T\mathbf{v}$.

Example: $\begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 10 \end{bmatrix}$

- First "matrix" has one row.
- Second "matrix" has one column.
- Therefore product "matrix" has one entry.

By dot-product definition of matrix-matrix multiplication,
that one entry is the dot-product of $\mathbf{u}$ and $\mathbf{v}$.

Sometimes called *inner product* of matrices.
However, that term has taken on another meaning, which we study later.

Outer product

Another way to multiply vectors as matrices.
For any $\mathbf{u}$ and $\mathbf{v}$, consider $\mathbf{u}\mathbf{v}^T$.

Example: $\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \begin{bmatrix} v_1 & v_2 & v_3 & v_4 \end{bmatrix} = \begin{bmatrix} u_1 v_1 & u_1 v_2 & u_1 v_3 & u_1 v_4 \\ u_2 v_1 & u_2 v_2 & u_2 v_3 & u_2 v_4 \\ u_3 v_1 & u_3 v_2 & u_3 v_3 & u_3 v_4 \end{bmatrix}$

For each element s of the domain of $\mathbf{u}$ and each element t of the domain of $\mathbf{v}$, the s, t element of $\mathbf{u}\mathbf{v}^T$ is $\mathbf{u}[s]\ \mathbf{v}[t]$.

Called *outer product* of $\mathbf{u}$ and $\mathbf{v}$.

Matrix-matrix multiplication and function composition

Corresponding to an $R \times C$ matrix A over a field $\mathbb{F}$, there is a function

$$f : \mathbb{F}^C \longrightarrow \mathbb{F}^R$$

namely the function defined by $f(\mathbf{y}) = A * \mathbf{y}$

Matrix-matrix multiplication and function composition

Matrices A and B $\Rightarrow$ functions $f(\mathbf{y}) = A * \mathbf{y}$ and $g(\mathbf{x}) = B * \mathbf{x}$ and $h(\mathbf{x}) = (AB) * \mathbf{x}$

$\boxed{\textbf{Matrix-Multiplication Lemma } f \circ g = h}$

Example:

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \Rightarrow f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1 + x_2 \\ x_2 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \Rightarrow g\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_1 + x_2 \end{bmatrix}$$

product $AB = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$

corresponds to function $h\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2x_1 + x_2, x_1 + x_2 \end{bmatrix}$

$f \circ g\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = f\left(\begin{bmatrix} x_1 \\ x_1 + x_2 \end{bmatrix}\right) = \begin{bmatrix} 2x_1 + x_2 \\ x_1 + x_2 \end{bmatrix}$ so $f \circ g = h$

Matrix-matrix multiplication and function composition

Matrices A and $B \Rightarrow$ functions $f(\mathbf{y}) = A * \mathbf{y}$ and $g(\mathbf{x}) = B * \mathbf{x}$ and $h(\mathbf{x}) = (AB) * \mathbf{x}$

Matrix-Multiplication Lemma: $f \circ g = h$

Proof: Let columns of B be $\mathbf{b}_1, \ldots, \mathbf{b}_n$. By the matrix-vector definition of matrix-matrix multiplication, column j of AB is $A *$ (column j of B).
For any n-vector $\mathbf{x} = [x_1, \ldots, x_n]$,

$$\begin{aligned} g(\mathbf{x}) &= B * \mathbf{x} && \text{by definition of } g \\ &= x_1\mathbf{b}_1 + \cdots + x_n\mathbf{b}_n && \text{by linear combinations definition} \end{aligned}$$

Therefore

$$\begin{aligned} f(g(\mathbf{x})) &= f(x_1\mathbf{b}_1 + \cdots x_n\mathbf{b}_n) \\ &= x_1(f(\mathbf{b}_1)) + \cdots + x_n(f(\mathbf{b}_n)) && \text{by linearity of } f \\ &= x_1(A * \mathbf{b}_1) + \cdots + x_n(A * \mathbf{b}_n) && \text{by definition of } f \\ &= x_1(\text{column 1 of } AB) + \cdots + x_n(\text{column } n \text{ of } AB) && \text{by matrix-vector def.} \\ &= (AB) * \mathbf{x} && \text{by linear-combinations def.} \\ &= h(\mathbf{x}) && \text{by definition of } h \end{aligned}$$

QED

Associativity of matrix-matrix multiplication

Matrices A and $B \Rightarrow$ functions $f(\mathbf{y}) = A * \mathbf{y}$ and $g(\mathbf{x}) = B * \mathbf{x}$ and $h(\mathbf{x}) = (AB) * \mathbf{x}$

$\boxed{\textbf{Matrix-Multiplication Lemma: } f \circ g = h}$

Matrix-matrix multiplication corresponds to function composition.

Corollary: Matrix-matrix multiplication is associative:

$$(AB)C = A(BC)$$

Proof: Function composition is associative. QED

Example:

$$\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \left(\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 \\ 1 & 2 \end{bmatrix} \right) = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 5 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 0 & 5 \\ 1 & 7 \end{bmatrix}$$

$$\left(\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \right) \begin{bmatrix} -1 & 3 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} -1 & 3 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 0 & 5 \\ 1 & 7 \end{bmatrix}$$

From function inverse to matrix inverse

Matrices A and $B \Rightarrow$ functions $f(\mathbf{y}) = A * \mathbf{y}$ and $g(\mathbf{x}) = B * \mathbf{x}$ and $h(\mathbf{x}) = (AB) * \mathbf{x}$

Definition If f and g are functional inverses of each other, we say A and B are matrix inverses of each other.

Example: An elementary row-addition matrix

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow \text{function } f([x_1, x_2, x_3]) = [x_1, x_2 + 2x_1, x_3])$$

Function adds twice the first entry to the second entry.

Functional inverse: subtracts the twice the first entry from the second entry:

$$f^{-1}([x_1, x_2, x_3]) = [x_1, x_2 - 2x_1, x_3]$$

Thus the inverse of A is

$$A^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This matrix is also an elementary row-addition matrix.

Matrix inverse

If A and B are matrix inverses of each other, we say A and B are *invertible* matrices.

Can show that a matrix has at most one inverse.

We denote the inverse of matrix A by A^{-1}.

(A matrix that is not invertible is sometimes called a *singular* matrix, and an invertible matrix is called a *nonsingular* matrix.)

Invertible matrices: why care?

Reason 1: Existence and uniqueness of solution to matrix-vector equations.

Let A be an $m \times n$ matrix, and define $f : \mathbb{F}^n \longrightarrow \mathbb{F}^m$ by $f(\mathbf{x}) = A\mathbf{x}$

Suppose A is an invertible matrix. Then f is an invertible function. Then f is one-to-one and onto:

- Since f is onto, for any m-vector $\mathbf{b}$ there is *some* vector $\mathbf{u}$ such that $f(\mathbf{u}) = \mathbf{b}$. That is, there is at least one solution to the matrix-vector equation $A\mathbf{x} = \mathbf{b}$.
- Since f is one-to-one, for any m-vector $\mathbf{b}$ there is *at most one* vector $\mathbf{u}$ such that $f(\mathbf{u}) = \mathbf{b}$. That is, there is at most one solution to $A\mathbf{x} = \mathbf{b}$.

If A is invertible then, for every right-hand side vector $\mathbf{b}$, the equation $A\mathbf{x} = \mathbf{b}$ has *exactly one solution*.

Example 1: Industrial espionage. Given the vector $\mathbf{b}$ specifying the amount of each resource consumed, figure out quantity of each product JunkCo has made.

Solve vector-matrix equation $\mathbf{x}^T M = \mathbf{b}$ where

M =

	metal	concrete	plastic	water	electricity
garden gnome	0	1.3	.2	.8	.4
hula hoop	0	0	1.5	.4	.3
slinky	.25	0	0	.2	.7
silly putty	0	0	.3	.7	.5
salad shooter	.15	0	.5	.4	.8

Invertible matrices: why care?

Reason 2: Algorithms for solving matrix-vector equation $A\mathbf{x} = \mathbf{b}$ are simpler if we can assume A is invertible.

Later we learn two such algorithms.

We also learn how to cope if A is not invertible.

Reason 3:
Invertible matrices play a key role in *change of basis*.

Change of basis is important part of linear algebra

- used e.g. in image compression;
- we will see it used in adding/removing perspective from an image.

Product of invertible matrices is an invertible matrix

Proposition: Suppose the matrix product AB is defined. Then AB is an invertible matrix if and only both A and B are invertible matrices.

Example:

$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ and $B = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$ correspond to functions

$f : \mathbb{R}^2 \longrightarrow \mathbb{R}^2$ and $g : \mathbb{R}^2 \longrightarrow \mathbb{R}^2$

$$f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1 + x_2 \\ x_2 \end{bmatrix}$$

$$g\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_1 + x_2 \end{bmatrix}$$

f is an invertible function.

g is an invertible function.

The functions f and g are invertible so the function $f \circ g$ is invertible.

By the Matrix-Multiplication Lemma, the function $f \circ g$ corresponds to the matrix product $AB = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$ so that matrix is invertible.

Proof: Define the functions f and g by $f(\mathbf{x}) = A\mathbf{x}$ and $g(\mathbf{x}) = B\mathbf{x}$.

Product of invertible matrices is an invertible matrix

Proposition: Suppose the matrix product AB is defined. Then AB is an invertible matrix if and only both A and B are invertible matrices.

Example:

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 4 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 5 & 0 & 1 \end{bmatrix}$$

Multiplication by matrix A adds 4 times first element to second element:

$$f([x_1, x_2, x_3]) = [x_1, x_2 + 4x_1, x_3])$$

This function is invertible.

Multiplication by matrix B adds 5 times first element to third element:

$$g([x_1, x_2, x_3]) = [x_1, x_2, x_3 + 5x_1]$$

This function is invertible

By Matrix Multiplication Lemma, multiplication by matrix AB corresponds to composition of functions $f \circ g$: $(f \circ g)([x_1, x_2, x_3]) = [x_1, x_2 + 4x_1, x_3 + 5x_1]$

The function $f \circ g$ is also an invertible function.... so AB is an invertible matrix.

Proof: Define the functions f and g by $f(\mathbf{x}) = A\mathbf{x}$ and $g(\mathbf{x}) = B\mathbf{x}$.

- Suppose A and B are invertible matrices.
  Then the corresponding functions f and g are invertible.
  Therefore $f \circ g$ is invertible

Product of invertible matrices is an invertible matrix

Proposition: Suppose the matrix product AB is defined. Then AB is an invertible matrix if and only both A and B are invertible matrices.

Example:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}$$

The product is $AB = \begin{bmatrix} 4 & 5 & 1 \\ 10 & 11 & 4 \\ 16 & 17 & 7 \end{bmatrix}$

which is *not* invertible

so at least one of A and B is not invertible

and in fact $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$ is *not* invertible.

Proof: Define the functions f and g by $f(\mathbf{x}) = A\mathbf{x}$ and $g(\mathbf{x}) = B\mathbf{x}$.

- Suppose A and B are invertible matrices.
  Then the corresponding functions f and g are invertible.
  Therefore $f \circ g$ is invertible
  so the matrix corresponding to $f \circ g$ (which is AB) is an invertible matrix

Product of invertible matrices is an invertible matrix

Proposition: Suppose the matrix product AB is defined. Then AB is an invertible matrix if and only both A and B are invertible matrices.

Proof: Define the functions f and g by $f(\mathbf{x}) = A\mathbf{x}$ and $g(\mathbf{x}) = B\mathbf{x}$.

- Suppose A and B are invertible matrices.
  Then the corresponding functions f and g are invertible.
  Therefore $f \circ g$ is invertible
  so the matrix corresponding to $f \circ g$ (which is AB) is an invertible matrix.

- Conversely, suppose AB is an invertible matrix.
  Then the corresponding function $f \circ g$ is an invertible function.
  It follows that f and g must be invertible functions,
  so the corresponding matrices A and B must be invertible matrices.

QED

Matrix inverse

Lemma: If the $R \times C$ matrix A has an inverse A^{-1} then AA^{-1} is the $R \times R$ identity matrix.

Proof: Let $B = A^{-1}$. Define $f(\mathbf{x}) = A\mathbf{x}$ and $g(\mathbf{y}) = B\mathbf{y}$.

- By the Matrix-Multiplication Lemma, $f \circ g$ satisfies $(f \circ g)(\mathbf{x}) = AB\mathbf{x}$.
- On the other hand, $f \circ g$ is the identity function,
- so AB is the $R \times R$ identity matrix.

QED

Matrix inverse

Lemma: If the $R \times C$ matrix A has an inverse A^{-1} then AA^{-1} is identity matrix.

What about the converse?

Conjecture: If AB is an indentity matrix then A and B are inverses...?

Counterexample:

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$AB = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$A * [0, 0, 1]$ and $A * [0, 0, 0]$ both equal $[0, 0]$, so null space of A is not trivial,
so the function $f(\mathbf{x}) = A\mathbf{x}$ is not one-to-one,
so f is not an invertible function.

Shows: $AB = I$ is *not* sufficient to ensure that A and B are inverses.

Matrix inverse

Lemma: If the $R \times C$ matrix A has an inverse A^{-1} then AA^{-1} is identity matrix.

What about the converse?

FALSE Conjecture: If AB is an indentity matrix then A and B are inverses...?

Corollary: Matrices A and B are inverses of each other if and only if both AB and BA are identity matrices.

Matrix inverse

Lemma: If the $R \times C$ matrix A has an inverse A^{-1} then AA^{-1} is identity matrix.

Corollary: A and B are inverses of each other iff both AB and BA are identity matrices.

Proof:

- Suppose A and B are inverses of each other. By lemma, AB and BA are identity matrices.
- Suppose AB and BA are both identity matrices.
  Define $f(\mathbf{y}) = A * \mathbf{y}$ and $g(\mathbf{x}) = B * \mathbf{x}$
  - Because AB is identity matrix, by Matrix-Multiplication Lemma, $f \circ g$ is the identity function.
  - Because BA is identity matrix, by Matrix-Multiplication Lemma, $g \circ f$ is the identity function.
  - This proves that f and g are functional inverses of each other, so A and B are matrix inverses of each other.

QED

Matrix inverse

Question: How can we tell if a matrix M is invertible?

Partial Answer: By definition, M is an invertible matrix if the function $f(\mathbf{x}) = M\mathbf{x}$ is an invertible function, i.e. if the function is one-to-one and onto.

- *One-to-one:* Since the function is linear, we know by the One-to-One Lemma that the function is one-to-one if its kernel is trivial, i.e. if the null space of M is trivial.
- *Onto:* We haven't yet answered the question *how we can tell if a linear function is onto?*

If we knew how to tell if a linear function is onto, therefore, we would know how to tell if a matrix is invertible.