Lecture 5a

Linear Algebra Review

T. Gambill

Department of Computer Science
University of Illinois at Urbana-Champaign

February 2, 2010

Vector Space Example

The set of n-tuples in $\mathbb{R}^n$ form a vector space.

$$v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$$

By convention we will write vectors in column form.
The transpose operator T converts column vectors to row vectors and vice versa.

$$\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}^T = \begin{bmatrix} v_1 & v_2 & \dots & v_n \end{bmatrix}$$

Vector Space Example 2

The set of continuous functions defined on a closed interval e.g. $f \in C([a,b]), f : [a,b] \to \mathbb{R}$ form a vector space over the reals $\mathbb{R}$.

$$\text{If } f_1, f_2 \in C([a,b]) \text{ then } f_1 + f_2 \in C([a,b])$$

$$\text{If } r \in \mathbb{R} \text{ then } r * f_1 \in C([a,b])$$

Vector Operations

- Addition and Subtraction
- Multiplication by a scalar
- Transpose
- Linear Combinations of Vectors
- Inner Product
- Outer Product
- Vector Norms

Vector Addition and Subtraction

Addition and subtraction are element-by-element operations

$$c = a + b \iff c_i = a_i + b_i \quad i = 1, \ldots, n$$
$$d = a - b \iff d_i = a_i - b_i \quad i = 1, \ldots, n$$

$$a = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \qquad b = \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}$$

$$a + b = \begin{bmatrix} 4 \\ 4 \\ 4 \end{bmatrix} \qquad a - b = \begin{bmatrix} -2 \\ 0 \\ 2 \end{bmatrix}$$

Multiplication by a Scalar

Multiplication by a scalar involves multiplying each element in the vector by the scalar:

$$b = \sigma a \quad \Longleftrightarrow \quad b_i = \sigma a_i \quad i = 1, \ldots, n$$

$$a = \begin{bmatrix} 4 \\ 6 \\ 8 \end{bmatrix} \qquad b = \frac{a}{2} = \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix}$$

Vector Transpose

The *transpose* of a row vector is a column vector:

$$u = \begin{bmatrix}1, 2, 3\end{bmatrix} \qquad \text{then} \qquad u^T = \begin{bmatrix}1\\2\\3\end{bmatrix}$$

Likewise if v is the column vector

$$v = \begin{bmatrix}4\\5\\6\end{bmatrix} \qquad \text{then} \qquad v^T = \begin{bmatrix}4, 5, 6\end{bmatrix}$$

Linear Combinations

Combine scalar multiplication with addition

$$\alpha \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix} + \beta \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix} = \begin{bmatrix} \alpha u_1 + \beta v_1 \\ \alpha u_2 + \beta v_2 \\ \vdots \\ \alpha u_m + \beta v_m \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix}$$

$$r = \begin{bmatrix} -2 \\ 1 \\ 3 \end{bmatrix} \qquad s = \begin{bmatrix} 1 \\ 0 \\ 3 \end{bmatrix}$$

$$t = 2r + 3s = \begin{bmatrix} -4 \\ 2 \\ 6 \end{bmatrix} + \begin{bmatrix} 3 \\ 0 \\ 9 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \\ 15 \end{bmatrix}$$

Linear Combinations

Any one vector can be created from an infinite combination of other "suitable" vectors.

$$w = \begin{bmatrix}4\\2\end{bmatrix} = 4\begin{bmatrix}1\\0\end{bmatrix} + 2\begin{bmatrix}0\\1\end{bmatrix}$$

$$w = 6\begin{bmatrix}1\\0\end{bmatrix} - 2\begin{bmatrix}1\\-1\end{bmatrix}$$

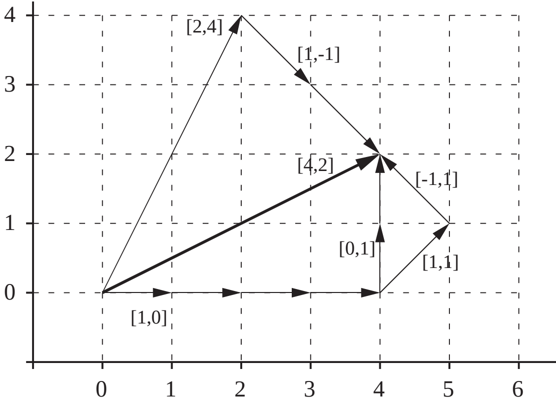
$$w = \begin{bmatrix}2\\4\end{bmatrix} - 2\begin{bmatrix}-1\\1\end{bmatrix}$$

$$w = 2\begin{bmatrix}4\\2\end{bmatrix} - 4\begin{bmatrix}1\\0\end{bmatrix} - 2\begin{bmatrix}0\\1\end{bmatrix}$$

Linear Combinations

Graphical interpretation:

- Vector tails can be moved to convenient locations
- Magnitude and direction of vectors is preserved



Vector Inner Product

In physics, analytical geometry, and engineering, the **dot product** has a geometric interpretation

$$\sigma = x \cdot y \quad \Longleftrightarrow \quad \sigma = \sum_{i=1}^{n} x_i y_i$$

$$x \cdot y = \|x\|_2 \, \|y\|_2 \cos\theta$$

Vector Inner Product

The inner product of x and y *requires* that x be a row vector y be a column vector

$$\begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = x_1y_1 + x_2y_2 + x_3y_3 + x_4y_4$$

Vector Inner Product

For two n-element *column* vectors, u and v, the inner product is

$$\sigma = u^T v \quad \Longleftrightarrow \quad \sigma = \sum_{i=1}^{n} u_i v_i$$

The inner product is commutative so that
(for two column vectors)

$$u^T v = v^T u$$

Computing the Inner Product in Python

The * operator performs the inner product if two vectors are compatible.

```
>>> import numpy as np
>>> u = np.arange(0,4,1,(float)).reshape(4,1) # u and v are column vectors
>>> v = np.arange(3,-1,-1,(float)).reshape(4,1)
>>> print(u*v)
[[ 0.]
 [ 2.]
 [ 2.]
 [ 0.]]
>>> print(u.reshape(1,4) * v)
[[ 0.  3.  6.  9.]
 [ 0.  2.  4.  6.]
 [ 0.  1.  2.  3.]
 [ 0.  0.  0.  0.]]
>>> print(np.dot(u.reshape(1,4), v))
[[ 4.]]
```

Vector Outer Product

The inner product results in a scalar.
The *outer product* creates a rank-one matrix:

$$A = uv^T \iff a_{i,j} = u_i v_j$$

Example

Outer product of two 4-element column vectors

$$uv^T = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \begin{bmatrix} v_1 & v_2 & v_3 & v_4 \end{bmatrix}$$

$$= \begin{bmatrix} u_1v_1 & u_1v_2 & u_1v_3 & u_1v_4 \\ u_2v_1 & u_2v_2 & u_2v_3 & u_2v_4 \\ u_3v_1 & u_3v_2 & u_3v_3 & u_3v_4 \\ u_4v_1 & u_4v_2 & u_4v_3 & u_4v_4 \end{bmatrix}$$

Computing the Outer Product in Matlab

The * operator performs the outer product if two vectors are compatible.

```
>>> u = np.arange(0,4,1,(float)).reshape(4,1) # u is a column vector
>>> v = np.arange(3,-1,-1,(float)).reshape(1,4) # v is a row vector
>>> print(u * v)
[[ 0.  0.  0.  0.]
 [ 3.  2.  1.  0.]
 [ 6.  4.  2.  0.]
 [ 9.  6.  3.  0.]]
>>> _
```

Vector Norms

Compare magnitude of scalars with the *absolute value*

$$|\alpha| > |\beta|$$

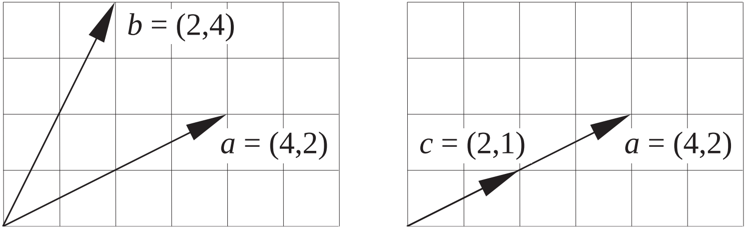
Compare magnitude of vectors with *norms*

$$\|x\| > \|y\|$$

There are several ways to compute $\|x\|$. In other words the size of two vectors can be compared with different norms.

Vector Norms

Consider two element vectors, which lie in a plane



Use geometric lengths to represent the magnitudes of the vectors

$$\ell_a = \sqrt{4^2 + 2^2} = \sqrt{20}, \qquad \ell_b = \sqrt{2^2 + 4^2} = \sqrt{20}, \qquad \ell_c = \sqrt{2^2 + 1^2} = \sqrt{5}$$

We conclude that

$$\ell_a = \ell_b \quad \text{and} \quad \ell_a > \ell_c$$

or

$$\|a\| = \|b\| \quad \text{and} \quad \|a\| > \|c\|$$

The L_2 Norm

The notion of a geometric length for 2D or 3D vectors can be extended vectors with arbitrary numbers of elements.
The result is called the *Euclidian* or L_2 norm:

$$\|x\|_2 = \left(x_1^2 + x_2^2 + \ldots + x_n^2\right)^{1/2} = \left(\sum_{i=1}^{n} x_i^2\right)^{1/2}$$

The L_2 norm can also be expressed in terms of the inner product

$$\|x\|_2 = \sqrt{x \cdot x} = \sqrt{x^T x}$$

p-Norms

For any positive integer p

$$\|x\|_p = \left(|x_1|^p + |x_2|^p + \ldots + |x_n|^p\right)^{1/p}$$

The L_1 norm is sum of absolute values

$$\|x\|_1 = |x_1| + |x_2| + \ldots + |x_n| = \sum_{i=1}^{n} |x_i|$$

The L_∞ norm or *max norm* is

$$\|x\|_\infty = \max\left(|x_1|, |x_2|, \ldots, |x_n|\right) = \max_i\left(|x_i|\right)$$

Although p can be any positive number, $p = 1, 2, \infty$ are most commonly used.

Application of Norms

Are two vectors (nearly) equal?

Floating point comparison of two scalars with absolute value:

$$\frac{|\alpha - \beta|}{|\alpha|} < \delta$$

where δ is a small tolerance.

Comparison of two vectors with norms:

$$\frac{\|y - z\|}{\|z\|} < \delta$$

Application of Norms

Notice that

$$\frac{\|y - z\|}{\|z\|} < \delta$$

is **not equivalent to**

$$\frac{\|y\| - \|z\|}{\|z\|} < \delta.$$

This comparison is important in convergence tests for sequences of vectors.

Application of Norms

Creating a Unit Vector
Given $u = [u_1, u_2, \ldots, u_m]^T$, the unit vector in the direction of u is

$$\hat{u} = \frac{u}{\|u\|_2}$$

Proof:

$$\|\hat{u}\|_2 = \left\| \frac{u}{\|u\|_2} \right\|_2 = \frac{1}{\|u\|_2} \|u\|_2 = 1$$

The following are *not* unit vectors

$$\frac{u}{\|u\|_1} \qquad \frac{u}{\|u\|_\infty}$$

Orthogonal Vectors

From geometric interpretation of the inner product

$$u \cdot v = \|u\|_2 \, \|v\|_2 \cos\theta$$

$$\cos\theta = \frac{u \cdot v}{\|u\|_2 \, \|v\|_2} = \frac{u^T v}{\|u\|_2 \, \|v\|_2}$$

Two vectors are orthogonal when $\theta = \pi/2$ or $u \cdot v = 0$.
In other words

$$u^T v = 0$$

if and only if u and v are *orthogonal*.

Orthonormal Vectors

Orthonormal vectors are **unit vectors** that are *orthogonal*.
A **unit** vector has an L_2 norm of one.
The unit vector in the direction of u is

$$\hat{u} = \frac{u}{\|u\|_2}$$

Since

$$\|u\|_2 = \sqrt{u \cdot u}$$

it follows that $u \cdot u = 1$ if u is a unit vector.

Matrices

- Columns and Rows of a Matrix are Vectors
- Addition and Subtraction
- Multiplication by a scalar
- Transpose
- Linear Combinations of Vectors
- Matrix–Vector Product
- Matrix–Matrix Product

Notation

The matrix A with m rows and n columns looks like:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & & a_{2n} \\ \vdots & & & \vdots \\ a_{m1} & & \cdots & a_{mn} \end{bmatrix}$$

a_{ij} = element in **row** i, and **column** j

In Python we can define a matrix with

```
>> A = numpy.array([ [...] , [...] , [...] ])
```

where commas separate lists of row elements.
The $a_{2,3}$ element of the Python array `A` is `A[1,2]`.

Matrices Consist of Row and Column Vectors

As a collection of column vectors

$$A = \left[\begin{array}{c|c|c|c} & & & \\ a_{(1)} & a_{(2)} & \cdots & a_{(n)} \\ & & & \end{array} \right]$$

As a collection of row vectors

$$A = \left[\begin{array}{c} a'_{(1)} \\ \hline a'_{(2)} \\ \hline \vdots \\ \hline a'_{(m)} \end{array} \right]$$

A prime is used to designate a row vector on this and the following pages.

Preview of the Row and Column View

Matrix and
vector operations

$\updownarrow$

Row and column
operations

$\updownarrow$

Element-by-element
operations

Matrix Operations

- Addition and subtraction
- Multiplication by a Scalar
- Matrix Transpose
- Matrix–Vector Multiplication
- Vector–Matrix Multiplication
- Matrix–Matrix Multiplication

Matrix Operations

Addition and subtraction

$$C = A + B$$

or

$$c_{i,j} = a_{i,j} + b_{i,j} \quad i = 1, \ldots, m; \quad j = 1, \ldots, n$$

Multiplication by a Scalar

$$B = \sigma A$$

or

$$b_{i,j} = \sigma a_{i,j} \quad i = 1, \ldots, m; \quad j = 1, \ldots, n$$

Note

Commas in subscripts are necessary when the subscripts are assigned numerical values. For example, $a_{2,3}$ is the row 2, column 3 element of matrix A, whereas a_{23} is the 23rd element of vector a. When variables appear in indices, such as a_{ij} or $a_{i,j}$, the comma is optional

Matrix Transpose

$$B = A^T$$

or

$$b_{i,j} = a_{j,i} \quad i = 1, \ldots, m; \;\; j = 1, \ldots, n$$

Matrix Transpose

In Python

```
>>> import numpy
>>> A = numpy.array([[0., 0., 0.], [0., 0., 0.], [1., 2., 3.],
    [0., 0., 0.]])
>>> A
array([[ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 1.,  2.,  3.],
       [ 0.,  0.,  0.]])

>>> B = A.transpose()
>>> B
>>> B
array([[ 0.,  0.,  1.,  0.],
       [ 0.,  0.,  2.,  0.],
       [ 0.,  0.,  3.,  0.]])
```

Matrix–Vector Product

- The Column View
  - gives mathematical insight
- The Row View
  - easy to do by hand
- The Vector View
  - A square matrix rotates and stretches a vector

Column View of Matrix–Vector Product

Consider a **linear combination of a set of column vectors** $\{a_{(1)}, a_{(2)}, \ldots, a_{(n)}\}$. Each $a_{(j)}$ has m elements
Let x_i be a set (a vector) of scalar multipliers

$$x_1 a_{(1)} + x_2 a_{(2)} + \ldots + x_n a_{(n)} = b$$

or

$$\sum_{j=1}^{n} a_{(j)} x_j = b$$

Expand the (hidden) row index

$$x_1 \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{m1} \end{bmatrix} + x_2 \begin{bmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{m2} \end{bmatrix} + \cdots + x_n \begin{bmatrix} a_{1n} \\ a_{2n} \\ \vdots \\ a_{mn} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

Column View of Matrix–Vector Product

Form a matrix with the $a_{(j)}$ as columns

$$\begin{bmatrix} \Bigg| & \Bigg| & & \Bigg| \\ a_{(1)} & a_{(2)} & \cdots & a_{(n)} \\ \Bigg| & \Bigg| & & \Bigg| \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \\ b \\ \\ \end{bmatrix}$$

Or, writing out the elements

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

Column View of Matrix–Vector Product

Thus, the matrix-vector product is

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

Save space with matrix notation

$$Ax = b$$

Column View of Matrix–Vector Product

The matrix–vector product $b = Ax$ produces a vector b from a linear combination of the columns in A.

$$b = Ax \quad \Longleftrightarrow \quad b_i = \sum_{j=1}^{n} a_{ij} x_j$$

where x and b are column vectors

Column View of Matrix–Vector Product

Listing 1: Matrix–Vector Multiplication by Columns

```
initialize:  b = zeros(m,1)
for j = 1,...,n
   for i = 1,...,m
      b(i) = A(i,j)x(j) + b(i)
   end
end
```

Compatibility Requirement

Inner dimensions must agree

$$\begin{array}{cccc} A & x & = & b \\ [m \times n] & [n \times 1] & = & [m \times 1] \end{array}$$

Row View of Matrix–Vector Product

Consider the following matrix–vector product written out as a linear combination of matrix columns

$$\begin{bmatrix} 5 & 0 & 0 & -1 \\ -3 & 4 & -7 & 1 \\ 1 & 2 & 3 & 6 \end{bmatrix} \begin{bmatrix} 4 \\ 2 \\ -3 \\ -1 \end{bmatrix}$$

$$= 4 \begin{bmatrix} 5 \\ -3 \\ 1 \end{bmatrix} + 2 \begin{bmatrix} 0 \\ 4 \\ 2 \end{bmatrix} - 3 \begin{bmatrix} 0 \\ -7 \\ 3 \end{bmatrix} - 1 \begin{bmatrix} -1 \\ 1 \\ 6 \end{bmatrix}$$

This is the column view.

Row View of Matrix–Vector Product

Now, group the multiplication and addition operations by row:

$$4\begin{bmatrix}5\\-3\\1\end{bmatrix}+2\begin{bmatrix}0\\4\\2\end{bmatrix}-3\begin{bmatrix}0\\-7\\3\end{bmatrix}-1\begin{bmatrix}-1\\1\\6\end{bmatrix}$$

$$=\begin{bmatrix}(5)(4) & + & (0)(2) & + & (0)(-3) & + & (-1)(-1)\\(-3)(4) & + & (4)(2) & + & (-7)(-3) & + & (1)(-1)\\(1)(4) & + & (2)(2) & + & (3)(-3) & + & (6)(-1)\end{bmatrix}=\begin{bmatrix}21\\16\\-7\end{bmatrix}$$

Final result is identical to that obtained with the column view.

Row View of Matrix–Vector Product

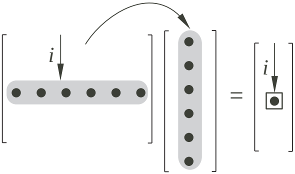
Product of a 3×4 matrix, A, with a 4×1 vector, x, looks like

$$\begin{bmatrix} a'_{(1)} \\ \hline a'_{(2)} \\ \hline a'_{(3)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} a'_{(1)} \cdot x \\ a'_{(2)} \cdot x \\ a'_{(3)} \cdot x \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

where $a'_{(1)}$, $a'_{(2)}$, and $a'_{(3)}$, are the *row vectors* constituting the A matrix.

The matrix–vector product $b = Ax$ produces elements in b by forming inner products of the rows of A with x.

Row View of Matrix–Vector Product



$a'_{(i)}$ $\quad x \quad$ y_i

Vector View of Matrix–Vector Product

If A is square, the product Ax has the effect of stretching and rotating x.
Pure stretching of the column vector

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix}$$

Pure rotation of the column vector

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

Vector–Matrix Product

Matrix–vector product

$$\underset{m\times n}{\begin{bmatrix} \quad & \quad \\ & \end{bmatrix}} \underset{n\times 1}{\begin{bmatrix} \\ \end{bmatrix}} = \underset{m\times 1}{\begin{bmatrix} \\ \end{bmatrix}}$$

Vector–Matrix product

$$\underset{1\times m}{\begin{bmatrix} \quad & \quad \end{bmatrix}} \underset{m\times n}{\begin{bmatrix} \quad & \quad \\ & \end{bmatrix}} = \underset{1\times n}{\begin{bmatrix} \quad & \quad \end{bmatrix}}$$

Vector–Matrix Product

Compatibility Requirement: Inner dimensions must agree

$$\begin{array}{cccc} u & A & = & v \\ [1 \times m] & [m \times n] & = & [1 \times n] \end{array}$$

Matrix–Matrix Product

Computations can be organized in **six different ways**
We'll focus on just two

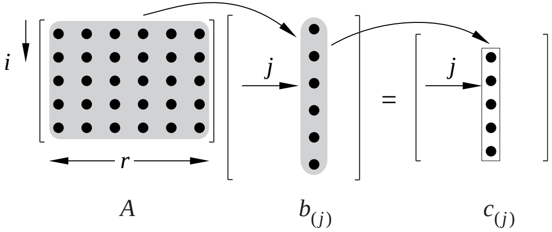
- Column View — extension of column view of matrix–vector product
- Row View — inner product algorithm, extension of column view of matrix–vector product

Column View of Matrix–Matrix Product

The product AB produces a matrix C. The columns of C are linear combinations of the columns of A.

$$AB = C \qquad \Longleftrightarrow \qquad c_{(j)} = Ab_{(j)}$$

$c_{(j)}$ and $b_{(j)}$ are column vectors.



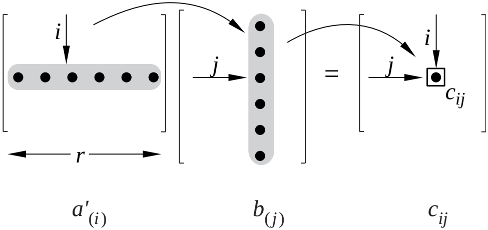
The column view of the matrix–matrix product $AB = C$ is helpful because it shows the relationship between the columns of A and the columns of C.

Inner Product (Row) View of Matrix–Matrix Product

The product AB produces a matrix C. The c_{ij} element is the *inner product* of row i of A and column j of B.

$$AB = C \qquad \Longleftrightarrow \qquad c_{ij} = a'_{(i)} b_{(j)}$$

$a'_{(i)}$ is a row vector, $b_{(j)}$ is a column vector.



The inner product view of the matrix–matrix product is easier to use for hand calculations.

Matrix–Matrix Product Summary

The **Matrix–vector product** looks like:

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

The **vector–Matrix product** looks like:

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

Matrix–Matrix Product Summary

The **Matrix–Matrix product** looks like:

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

Matrix–Matrix Product Summary

Compatibility Requirement

$$\begin{array}{cccc} A & B & = & C \\ [m \times r] & [r \times n] & = & [m \times n] \end{array}$$

Inner dimensions must agree
Also, in general

$$AB \neq BA$$

Mathematical Properties of Vectors and Matrices

- Linear Independence
- Vector Spaces
- Subspaces associated with matrices
- Matrix Rank

Linear Independence

Two vectors lying along the same line are not independent

$$u = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \qquad \text{and} \qquad v = -2u = \begin{bmatrix} -2 \\ -2 \\ -2 \end{bmatrix}$$

Any two independent vectors, for example,

$$v = \begin{bmatrix} -2 \\ -2 \\ -2 \end{bmatrix} \qquad \text{and} \qquad w = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

define a plane. Any other vector in this plane of v and w can be represented by

$$x = \alpha v + \beta w$$

x is **linearly dependent** on v and w because it can be formed by a linear combination of v and w.

Linear Independence

A set of vectors is linearly independent if it is impossible to use a linear combination of vectors in the set to create another vector in the set. Linear independence is easy to see for vectors that are orthogonal, for example,

$$\begin{bmatrix} 4 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ -3 \\ 0 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

are linearly independent.

Linear Independence

Consider two linearly independent vectors, u and v.
If a third vector, w, *cannot* be expressed as a linear combination of u and v, then the set $\{u, v, w\}$ is linearly independent.
In other words, if $\{u, v, w\}$ is linearly independent then

$$\alpha u + \beta v = \delta w$$

can be true *only if* $\alpha = \beta = \delta = 0$.
More generally, if the only solution to

$$\alpha_1 v_{(1)} + \alpha_2 v_{(2)} + \cdots + \alpha_n v_{(n)} = 0 \tag{1}$$

is $\alpha_1 = \alpha_2 = \ldots = \alpha_n = 0$, then the set $\{v_{(1)}, v_{(2)}, \ldots, v_{(n)}\}$ is **linearly independent**. Conversely, if equation (1) is satisfied by at least one nonzero α_i, then the set of vectors is **linearly dependent**.

Linear Independence

Let the set of vectors $\{v_{(1)}, v_{(2)}, \ldots, v_{(n)}\}$ be organized as the columns of a matrix. Then the condition of linear independence is

$$\left[\begin{array}{c|c|c|c} & & & \\ v_{(1)} & v_{(2)} & \cdots & v_{(n)} \\ & & & \end{array}\right] \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{2}$$

The columns of the $m \times n$ matrix, A, are linearly independent if and only if $x = (0, 0, \ldots, 0)^T$ is the only n element column vector that satisfies $Ax = 0$.

Vector Spaces

- Spaces and Subspaces
- Basis of a Subspace
- Subspaces associated with Matrices

Spaces and Subspaces

Group vectors according to number of elements they have. Vectors from these different groups cannot be mixed.

$\mathbf{R}^1$ = Space of all vectors with one element.
These vectors define the points along a line.

$\mathbf{R}^2$ = Space of all vectors with two elements.
These vectors define the points in a plane.

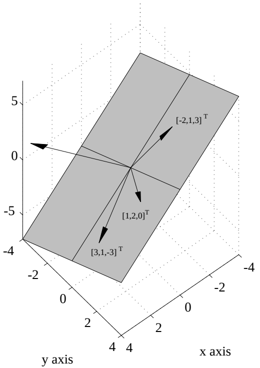
$\mathbf{R}^n$ = Space of all vectors with n elements.
These vectors define the points in an n-dimensional space (hyperplane).

Subspaces

The three vectors

$$u = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}, \quad v = \begin{bmatrix} -2 \\ 1 \\ 3 \end{bmatrix}, \quad w = \begin{bmatrix} 3 \\ 1 \\ -3 \end{bmatrix},$$

lie in the same plane. The vectors have three elements each, so they belong to $\mathbf{R}^3$, but they **span** a **subspace** of $\mathbf{R}^3$.



Basis and Dimension of a Subspace

- A **basis** for a subspace is a set of **linearly independent** vectors that **span** the subspace.
- Since a basis set must be linearly independent, it also must have the smallest number of vectors necessary to span the space. (Each vector makes a unique contribution to spanning some other direction in the space.)
- The number of vectors in a basis set is equal to the **dimension** of the **subspace** that these vectors span.
- Mutually orthogonal vectors (an orthogonal set) form convenient basis sets, but basis sets need not be orthogonal.

Subspaces Associated with Matrices

The matrix–vector product

$$y = Ax$$

creates y from a linear combination of the columns of A
The column vectors of A form a basis for the **column space** or **range** of A.

Matrix Rank

- The **rank** of a matrix, A, is the number of linearly independent columns in A.
- $\mathrm{rank}(A)$ is the dimension of the column space of A.
- Numerical computation of $\mathrm{rank}(A)$ is tricky due to roundoff.

Consider

$$u = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad v = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \qquad w = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

Do these vectors span $\mathbf{R}^3$?

Matrix Rank

- The **rank** of a matrix, A, is the number of linearly independent columns in A.
- $\mathrm{rank}(A)$ is the dimension of the column space of A.
- Numerical computation of $\mathrm{rank}(A)$ is tricky due to roundoff.

Consider

$$u = \begin{bmatrix} 1 \\ 0 \\ 0.00001 \end{bmatrix} \qquad v = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \qquad w = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

Do these vectors span $\mathbf{R}^3$?

Matrix Rank

- The **rank** of a matrix, A, is the number of linearly independent columns in A.
- $\operatorname{rank}(A)$ is the dimension of the column space of A.
- Numerical computation of $\operatorname{rank}(A)$ is tricky due to roundoff.

Consider

$$u = \begin{bmatrix} 1 \\ 0 \\ \varepsilon_m \end{bmatrix} \qquad v = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \qquad w = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

Do these vectors span $\mathbf{R}^3$?

Matrix Rank (2)

We can use Numpy's built-in **rank** function for exploratory calculations on (relatively) small matrices

```
>>> import numpy as np
>>> A = np.array([[1.,0.,0.],[0.,1.,0.],[0.,0.,0.]])
>>> A
array([[ 1.,  0.,  0.],
       [ 0.,  1.,  0.],
       [ 0.,  0.,  0.]])
>>> np.rank(A)
2
    _
```

Matrix Rank (2)

Repeat numerical calculation of rank with smaller diagonal entry

```
>>> import numpy as np
>>> A = np.eye(3)
>>> A
array([[ 1.,  0.,  0.],
       [ 0.,  1.,  0.],
       [ 0.,  0.,  1.]])
>>> A[2,2] = np.finfo(float).eps
>>> A
array([[  1.00000000e+00,   0.00000000e+00,   0.00000000e+00],
       [  0.00000000e+00,   1.00000000e+00,   0.00000000e+00],
       [  0.00000000e+00,   0.00000000e+00,   2.22044605e-16]])
>>> np.rank(A)
2
```

Even though `A(2,2)` is not identically zero, it is small enough that the matrix is *numerically* rank-deficient

Special Matrices

- Diagonal Matrices
- Tridiagonal Matrices
- The Identity Matrix
- The Matrix Inverse
- Symmetric Matrices
- Positive Definite Matrices
- Orthogonal Matrices
- Permutation Matrices

Diagonal Matrices

Diagonal matrices have non-zero elements only on the main diagonal.

$$C = \text{diag}\,(c_1, c_2, \ldots, c_n) = \begin{bmatrix} c_1 & 0 & \cdots & 0 \\ 0 & c_2 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & c_n \end{bmatrix}$$

The **diagflat** function is used to create a diagonal matrix from a vector.

```
>>> import numpy as np
>>> x = np.array([1.,-5.,2.,6.])
>>> A = np.diagflat(x)
>>> A
array([[ 1.,  0.,  0.,  0.],
       [ 0., -5.,  0.,  0.],
       [ 0.,  0.,  2.,  0.],
       [ 0.,  0.,  0.,  6.]])
```

Diagonal Matrices

The **diagflat** function can also be used to create a matrix with elements only on a specified *super*-diagonal or *sub*-diagonal. Doing so requires using the two-parameter form of **diagflat**:

```
>>> np.diagflat(np.array([1.,2.,3.]),k=1)
array([[ 0.,  1.,  0.,  0.],
       [ 0.,  0.,  2.,  0.],
       [ 0.,  0.,  0.,  3.],
       [ 0.,  0.,  0.,  0.]])
>>> np.diagflat(np.array([1.,2.,3.]),k=-1)
array([[ 0.,  0.,  0.,  0.],
       [ 1.,  0.,  0.,  0.],
       [ 0.,  2.,  0.,  0.],
       [ 0.,  0.,  3.,  0.]])
```

Identity Matrices

An identity matrix is a square matrix with ones on the main diagonal.

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

An identity matrix is special because

$$AI = A \quad \text{and} \quad IA = A$$

for *any* compatible matrix A. This is like multiplying by one in scalar arithmetic.

Identity Matrices

Identity matrices can be created with the built-in **eye** function.

```
>>> I = np.eye(4)
>>> I
array([[ 1.,  0.,  0.,  0.],
       [ 0.,  1.,  0.,  0.],
       [ 0.,  0.,  1.,  0.],
       [ 0.,  0.,  0.,  1.]])
```

Sometimes I_n is used to designate an identity matrix with n rows and n columns. For example,

$$I_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Identity Matrices

A non-square, *identity-like* matrix can be created with the two-parameter form of the `eye` function:

```
>>> J = np.eye(3,5)
>>> J
array([[ 1.,  0.,  0.,  0.,  0.],
       [ 0.,  1.,  0.,  0.,  0.],
       [ 0.,  0.,  1.,  0.,  0.]])
>>> K = np.eye(4,2)
>>> K
array([[ 1.,  0.],
       [ 0.,  1.],
       [ 0.,  0.],
       [ 0.,  0.]])
```

`J` and `K` are *not* identity matrices!

Functions to Create Special Matrices

Matrix	Matlab function
Diagonal	`diag`
Identity	`eye`
Inverse	`inv`

Symmetric Matrices

If $A = A^T$, then A is called a *symmetric* matrix.

$$\begin{bmatrix} 5 & -2 & -1 \\ -2 & 6 & -1 \\ -1 & -1 & 3 \end{bmatrix}$$

Note

$B = A^T A$ is symmetric for any (real) matrix A.

Tridiagonal Matrices

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

The diagonal elements need not be equal. The general form of a tridiagonal matrix is

$$A = \begin{bmatrix} a_1 & b_1 & & & & & \\ c_2 & a_2 & b_2 & & & & \\ & c_3 & a_3 & b_3 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & & & & \\ & & & & c_{n-1} & a_{n-1} & b_{n-1} \\ & & & & & c_n & a_n \end{bmatrix}$$