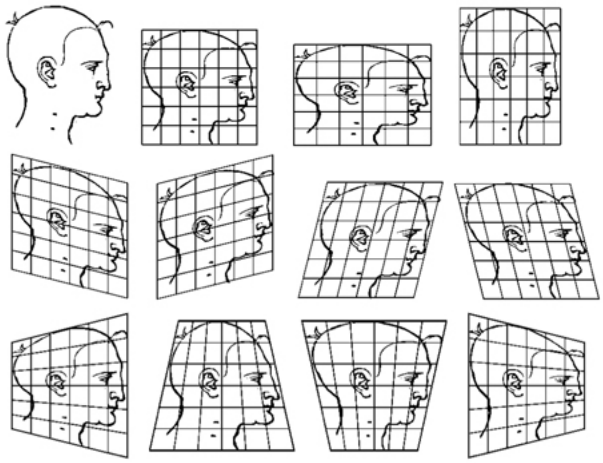
Linear Algebra

MATH1900: Machine Learning

Location: http://people.sc.fsu.edu/~jburkardt/classes/ml_2019/linear_algebra/linear_algebra.pdf

---



Matrices and vectors describe simple linear transformations

Machine learning looks for patterns and structures in data. Linear relationships are often the best simple approximation to complicated systems. Linear algebra is a collection of ideas and tools that we can use to construct these simple models of our observations.

In linear algebra, we study abstract objects called *vectors*; in machine learning, these are the individual observations of temperature, answers on a survey, medical records. While in machine learning, we might have a table whose rows are the observations, linear algebra thinks of this as a *matrix*. Linear algebra can help us decide whether

- two observations are very similar;
- your hospital bill can be approximately predicted by your sex, age, and smoking status;
- an image we have scanned is a picture of a cat or a dog.

The Linear Algebra Problem

How can we use the tools of linear algebra to analyze our data when we think of it as vectors? In particular, we want to:

- *initialize a vector;*
- *compute the norm (size) of a vector;*
- *compute the distance and angle between two vectors;*
- *initialize a matrix;*
- *multiply a vector by a matrix;*
- *solve a system of linear equations;*

1 One vector

We can think of a vector as a one-dimensional list of values. The number of values is the *extent* of the vector. We often refer to this extent as `n`. If v has extent n, we may express this by "v is an n-vector".

To measure the *magnitude* or *norm* of a vector, we combine its values in a certain way. Although there are many vector norms, the most common and useful is the *Euclidean* norm:

$$||v|| = \sqrt{\sum_{i=1}^{n} v_i^2}$$

In Python, the Euclidean norm of vector v can be computed by `np.linalg.norm(v)`:

```
1 import numpy as np
2 x = np.random.rand ( 2 )
3 print ( 'Norm of', x, ' is ', np.linalg.norm ( x ) )
4 y = np.array ( [ 3, 4 ] )
5 print ( 'Norm of', y, ' is ', np.linalg.norm ( y ) )
6 z = np.ones ( 2 )
7 print ( 'Norm of', z, ' is ', np.linalg.norm ( z ) )
```

The norm computes the length of a vector. If we divide a vector v by its norm, we get what is called a *unit vector*, sometimes written $\hat{v}$. We can think of $\hat{v}$ as the *direction* of the vector, while $||v||$ is the *strength* or *magnitude*

$$\hat{v} = \frac{v}{||v||}$$

This allows us to understand a vector as a product of a separate magnitude and direction:

$$v = ||v|| \, \frac{v}{||v||} = ||v|| \, \hat{v}$$

2 Two vectors

If we have two n-vectors v and w, there are some obvious questions we can ask:

- Is w equal to v?
- Is w simply a multiple of v?
- Is w at least partly a multiple of v?
- What is the angle between v and w?

It turns out that w is equal to v exactly if the norm of $v - w$ is zero. So this is easy to check! In fact, $||v - w||$ measures the magnitude or distance between the two vectors, so the size of this quantity is an indication of how close they are.

To answer the other questions, we need to know about the vector *dot product*:

$$< v, w >= \sum_{i=1}^{n} v(i) * w(i)$$

It turns out that

$$< v, w >= ||v|| \cdot ||w|| \cdot \cos(\alpha)$$

where α is the angle between the two vectors. In particular, this means that

$$\cos(\alpha) = \frac{< v, w >}{||v||\,||w||}$$

If w is simply a multiple of v, then

- $\cos(\alpha) = 1$ (the vectors point in the same direction), or
- $\cos(\alpha) = -1$ (the vectors point in opposite directions.)

On the other hand, if $\cos(\alpha) = 0$, then the vectors are perpendicular.

In general, two vectors will have $\cos(\alpha)$ somewhere between -1 and +1. Values near +1 or -1 mean the two directions are close, while values near 0 mean they have little relation. In our work, we will often need to use this measurement to decide whether two objects, represented by vectors, are closely related or not.

In Python, we can compute the vector dot product of v and w using `np.dot(v,w)`

```
1 import numpy as np
2 w = np.random.rand ( 2 )
3 x = np.random.rand ( 2 )
4 dwx = np.dot ( w, x )
5 print ( 'dot(w,x) = ', dwx )
```

and the cosine of α is also easy:

```
1 import numpy as np
2 w = np.random.rand ( 2 )
3 w_norm = np.linalg.norm ( w )
4 x = np.random.rand ( 2 )
5 x_norm = np.linalg.norm ( x )
6 cwx = np.dot ( w, x ) / w_norm / x_norm
7 print ( 'cos(w,x) = ', cwx )
```

3 A matrix

In linear algebra, a matrix is a two-dimensional table of numbers, with m rows and n columns. In machine learning, a matrix has two common uses. A matrix can be used to represent m examples of data, each having n measurements. It can also be used to represent a linear relationship that we discover between some of the measurements in a set of data. But this discussion will be useful, no matter which way we happen to be using a matrix in our machine learning application.

In Python, we have seen that a vector `v` can be initialized by listing its values using the `v=np.array([`*list of m values*`]))` function. Python thinks of a matrix as an *array of arrays*. Thus each value in the array definition is itself a list of values, that is a row vector. The matrix

$$A = \begin{pmatrix} 11 & 12 & 13 & 14 \\ 21 & 22 & 23 & 24 \\ 31 & 32 & 33 & 34 \end{pmatrix}$$

could be entered in Python by

```
1 import numpy as np
2 A = np.array ( [
3   [ 11, 12, 13, 14 ],
4   [ 21, 22, 23, 24 ],
5   [ 31, 32, 33, 34 ] ] )
```

Other useful matrix creation commands include

```
import numpy as np
B = np.zeros ( [ 4, 5 ] )     # A 4x5 array of 0's
C = np.ones ( [ 6, 3 ] )      # a 6x3 array of 1's
D = np.random.rand ( 2, 5 )   # 2x5 random values
I = np.identity ( 3 )         # 3x3 identity matrix
```

A matrix A can be measured using a norm, and `np.linalg.norm(A)` can compute it. The most interesting factor about the norm of a matrix is that it provides a limit on how much the matrix can transform a given vector. If the norm is 5, then the transformation can't make the vector any more than 5 times bigger than it was, for instance.

In Python, we can create a random 4×3 matrix and compute its size using norms:

```
import numpy as np
A = np.array ( [
   [ 1, 2, 3 ],
   [ 4, 5, 6 ],
   [ 7, 8, 0 ] ] )
A_norm = np.linalg.norm ( A )
print ( 'norm of', A, ' is ', A_norm )
```

4 $A * x = y$: A matrix can multiply a vector

Matrices affect vectors by multiplying them. To compute the matrix-vector product $A * x$, if A is an $m \times n$ matrix, then x must be an n-vector and y must be an m-vector. One way to think about this is that if you replace the equation $A * x = y$ by the shapes of the objects, you must have something like $(m \times n) \times n = m$. We can multiply a matrix times a vector using `np.matmul()`:

```
import numpy as np
A = np.array ( [
   [ 1, 2, 3 ],
   [ 4, 5, 6 ],
   [ 7, 8, 0 ] ] )
x = np.array ( [ 1, 2, 3 ] )
y = np.matmul ( A, x )
print ( A, '*', x, '=', y )
```

5 Solving $A * x = y$ for x

It turns out that many linear algebra problems seek to reverse the process of matrix multiplication. In other words, we know the matrix A and the right hand side y, and we wish to find a vector x so that $A * x = y$. This is called *solving a linear system.* For now, we will assume that the matrix A is square (that is, that $m = n$) and that the matrix A is not *singular.* When a matrix is singular, the solution process can break down. For now, we will just assume that's not going to happen.

As mentioned in class, Gauss elimination can be used to solve the system. Luckily for us, the function `np.linalg.solve(A,y)` will do this for us:

```
import numpy as np
A = np.array ( [
   [ 1, 2, 3 ],
   [ 4, 5, 6 ],
   [ 7, 8, 0 ] ] )
y = np.array ( [ 14, 32, 23 ] )
```

```
x = np.linalg.solve ( A, y )
print ( A, '*', x, '=', y )
#
#  Verify norm of error is zero or very small:
#
e = np.matmul ( A, x ) - y
e_norm = np.linalg.norm ( e )
print ( 'Error ||A*x-y|| = ', e_norm )
```

6 Matrix and vector norms estimate the size of $A * x$

When computing $A * x$, the norms of A and x give us a limit for how big the product $y = A * x$ can be. In particular:

$$y = A * x$$
$$||y|| \leq ||A|| * ||x||$$

Let us verify this for our example:

```
import numpy as np
A = np.array ( [
  [ 1, 2, 3 ],
  [ 4, 5, 6 ],
  [ 7, 8, 0 ] ] )
A_norm = np.linalg.norm(A)
x = np.array ( [ 1, 2, 3 ] )
x_norm = np.linalg.norm ( x )
y = np.matmul ( A, x )
y_norm = np.linalg.norm ( y )
print ( '||y|| = ', y_norm, '<=', A_norm, '*', x_norm '=', A_norm * x_norm )
```

7 Computing Assignment #1

Suppose that the matrix A and vector y are defined as:

$$A = \begin{pmatrix} 5 & 7 & 6 & 5 \\ 7 & 10 & 8 & 7 \\ 6 & 8 & 10 & 9 \\ 5 & 7 & 9 & 10 \end{pmatrix}, \qquad y = \begin{pmatrix} 58 \\ 81 \\ 77 \\ 69 \end{pmatrix}$$

Write a Python program called *hw1.py* in which you:

1. set the variables `A` and `y`;
2. solve for a vector `x` so that $A * x = y$;
3. compute $||A||$, $||x||$, $||y||$ and verify that $||y|| \leq ||A|| * ||x||$
4. compute the error $e = A * x - y$, and print the norm of the error, $||e||$, which should be zero or very small.

Email a copy of your program to Dr Schneier **mhs64@pitt.edu** by 11:59pm Friday, 13 September.