5.1 - NumPy: Linear Algebra

Ha Khanh Nguyen (hkmguyen)

---

1. Why Linear Algebra?

- Linear algebra, like matrix multiplication, decompositions, determinants, and other square matrix math, is an important part of any array library.
- Why is it so important?
  - Multivariate statistics is built on linear algebra!
  - Think Principal Component Analysis (PCA), Linear Regression, etc.

2. Matrix Multiplication & Transpose

- First, note that NumPy array by default operates using element-wise operation.
  - That is multiplying 2 two-dimensional arrays with `*` is an element-wise product instead of a matrix dot product.
- To perform a matrix dot product, we use the `dot` function.

```
import numpy as np

x = np.array([[1., 2., 3.], [4., 5., 6.]])
y = np.array([[6., 23.], [-1, 7], [8, 9]])
```

```
x
```

```
## array([[1., 2., 3.],
##        [4., 5., 6.]])
```

```
y
```

```
## array([[ 6., 23.],
##        [-1.,  7.],
##        [ 8.,  9.]])
```

```
x.dot(y)
```

```
## array([[ 28.,  64.],
##        [ 67., 181.]])
```

- `x.dot(y)` is equivalent to `np.dot(x, y)`:

```
np.dot(x, y)
```

```
## array([[ 28.,  64.],
##        [ 67., 181.]])
```

- The `@` symbol (as of Python 3.5) also works as a matrix multiplication operator:

```
x @ y
```

```
## array([[ 28.,  64.],
##        [ 67., 181.]])
```

- With transpose, all NumPy objects have a transpose attribute named `T`.

```
x.T
```

```
## array([[1., 4.],
##        [2., 5.],
##        [3., 6.]])
```

3. Functions in `numpy.linalg`

- The sub-module `linalg` inside NumPy has a standard set of matrix decompositions and functions like finding inverse matrix and determinant.

```
from numpy.linalg import inv, qr

np.random.seed(430)
X = np.random.randn(5, 5)
mat = X.T.dot(X)
```

- So `mat` is the dot product of `X` and the transpose of `X`.

```
inv(mat)
```

```
## array([[ 3.68212553,  2.65173213, -2.13005389, -1.12192032, -0.86993906],
##        [ 2.65173213,  2.06091811, -1.59494517, -0.95436695, -0.73828164],
##        [-2.13005389, -1.59494517,  2.10818009,  0.27693373,  0.68048432],
##        [-1.12192032, -0.95436695,  0.27693373,  2.41643666,  1.84011535],
##        [-0.86993906, -0.73828164,  0.68048432,  1.84011535,  2.09115177]])
```

Function	Description
`diag`	Return the diagonal (or off-diagonal) elements of a square matrix as a 1D array, or convert a 1D array into a square matrix with zeros on the off-diagonal
`dot`	Matrix multiplication
`trace`	Compute the sum of the diagonal elements
`det`	Compute the matrix determinant

Function	Description
`eig`	Compute the eigenvalues and eigenvectors of a square matrix
`inv`	Compute the inverse of a square matrix
`pinv`	Compute the Moore-Penrose pseudo-inverse of a matrix
`qr`	Compute the QR decomposition
`svd`	Compute the singular value decomposition (SVD)
`solve`	Solve the linear system Ax = b for x, where A is a square matrix
`lstsq`	Compute the least-squares solution to `Ax = b`

4. Exercise

We are given the following matrices:

```
X = np.array([[1, 2, 3], [1, 5, 2], [1, 8, 1]])
y = np.array([5, 4, 6])
```

Evaluate the following expression using NumPy matrix functions:

$$(X^T X)^{-1} X^T y$$

This lecture note is modified from Chapter 4 of Wes McKinney's Python for Data Analysis 2nd Ed (https://www.oreilly.com/library/view/python-for-data/9781491957653/).