CPE 490 590: Machine Learning for Engineering Applications

02 Mathematical Preliminaries: Linear Algebra Review

Rahul Bhadani

Electrical & Computer Engineering, The University of Alabama in Huntsville

Outline

Logistics

Announcement

- Homework 1 Due Date: Jan 22, 2025, 11:59 PM
- Quiz 1 Deadline: Jan 26, 2025, 11:59 PM

Linear Algebra Refresher

Linearity

Linear Combination:

$$\sum_{i}^{n} w_i x_i \tag{1}$$

Linearity

A System of Linear Equation:

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\
&\vdots \\
a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m \\
a_{ij}, b_i &\in \mathbb{R}
\end{aligned}
$$

Vectors

Simplistic Definition:

A vector is a tuple of one or more values called scalars. We will denote a vector by boldface lowercase letters, such as $\mathbf{v}$.

"Vectors are built from components, which are ordinary numbers. You can think of a vector as a list of numbers, and vector algebra as operations performed on the numbers in the list."
– No Bullshit Guide To Linear Algebra, 2017.

Vectors

Column Vectors and Row Vectors

Column vectors can be written as $\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$ and row vectors can be written as $\mathbf{v} = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}$

Vectors in Python

Column Vector

```
import numpy as np
v = np.array([[1], [2],[3]]) # 3 rows 1 column
print(v.shape)
```

Output:

```
(3, 1)
```

Vectors in Python

Row Vector

```
np.array([[1, 2, 3]]) # 1 row 3 columns
print(v.shape)
```

Output:

```
(1, 3)
```

Alternative Implementation of Vectors in Python

Vector as an array

We can have vector implementation in Python like a 1-D array where there is no distinction between column vectors and row vectors.

We will use this form for any operations on vectors when we don't need to worry about whether they are column vectors or row vectors.

```
import numpy as np
v = np.array([1, 2,3]) # 3-tuple vector
print(v.shape)
```

Output:

```
(3, )
```

Operations on Vectors

Elementwise Vector Multiplication

```
import numpy as np
v = np.array([[1], [2],[3]]) # 3 rows 1 column
w = np.array([[3], [4],[5]]) # 3 rows 1 column
u = v*w
print(u)
```

Output:

```
array([[ 3],
       [ 8],
       [15]])
```

Operations on Vectors

Vector Dot Product (also known as inner product)

Dot product of two vectors $\mathbf{v} = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}$ and $\mathbf{w} = \begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix}$ is defined as $u = v \cdot w = v_1 w_1 + v_2 w_2 + v_3 w_3$

```
import numpy as np
v = np.array([1, 2, 3]) # 3 rows 1 column
w = np.array([3, 4, 5]) # 3 rows 1 column
u = v.dot(w)
print(u)
```

Output:

```
26
```

Vector Norms

We often need to calculate the length or magnitude of a vector. Different ways to calculate vector norms are called vector norms. The length or magnitude is a non-negative number. We will consider three kind of vector norms: L1 norm, L2 norm, and Max norm.

Vector Norms

L1 Norm (Manhattan Norm)

The L1 norm of a vector $\mathbf{v} = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}$ is defined as $\|v\|_1 = |v_1| + |v_2| + |v_3|$.

```
import numpy as np
v = np.array([1, 2, 3])
norm_v = np.linalg.norm(v, ord=1)
print(norm_v)
```

Output:

```
6.0
```

Vector Norms

L2 Norm (Euclidean Norm)

The L2 norm of a vector $\mathbf{v} = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}$ is defined as $\|v\|_2 = \sqrt{v_1^2 + v_2^2 + v_3^2}$.

```
import numpy as np
v = np.array([1, 2, 3])
norm_v = np.linalg.norm(v)
print(norm_v)
```

Output:

```
3.7416573867739413
```

Vector Norms

Max Norm

The max norm (also known as infinity norm) of a vector $\mathbf{v} = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}$ is defined as $\|v\|_\infty = \max(|v_1|, |v_2|, |v_3|)$.

```
import numpy as np
v = np.array([1, 2, 3]) # 3 rows 1 column
norm_v = np.linalg.norm(v, np.inf)
print(norm_v)
```

Output:

```
3.0
```

Matrices

Simplistic Definition:

For a positive integer $m, n \in \mathbb{R}$, a matrix $\mathbf{A}$ is $m \times n$ tuple of elements $a_{i,j}$, $i = 1, \cdot, m$, $j = 1, \cdot n$ which is ordered according to a rectangular scheme with m rows and n columns such that we can write:

(2)

We will set a convention to use boldface uppercase letters to denote a matrix. You can see that a matrix can be constructed by placing multiple row vectors or column vectors as well.

Matrix in Python

A rectangular matrix

```
import numpy as np
A = np.array([[1, 2, 3], [4, 5, 6]])
print(A.shape)
```

Output:

```
(2, 3)
```

Elementwise Matrix Multiplication (Hadamard Product)

Two matrices with the same size can be multiplied together, and this is often called element-wise matrix multiplication or the Hadamard product. It is denoted using a small circle $\circ$:

$$\mathbf{C} = \mathbf{A} \circ \mathbf{B}$$

.

Hadamard Product

```
import numpy as np
A = np.array([[1, 2, 3],[4, 5, 6]])
B = np.array([[1, 2, 3],[4, 5, 6]])
C = A * B
```

Matrix-Matrix Multiplication (Or, Matrix Multiplication)

Matrix multiplication is a binary operation that takes a pair of matrices and produces another matrix. If $\mathbf{A}$ is an $m \times n$ matrix and $\mathbf{B}$ is an $n \times k$ matrix, then their matrix product $\mathbf{AB}$ is an $m \times k$ matrix.
The element at the i-th row and j-th column of $\mathbf{C}$, denoted as c_{ij}, can be calculated as follows:

$$c_{ij} = \sum_{r=1}^{n} a_{ir} b_{rj} \tag{3}$$

where a_{ir} is the element at the i-th row and r-th column of $\mathbf{A}$, and b_{rj} is the element at the r-th row and j-th column of $\mathbf{B}$. The summation runs over the r index from 1 to n.
In Python, you can use the 'numpy' library to perform matrix multiplication using the 'dot' function.

Matrix-Matrix Multiplication (Or, Matrix Multiplication)

Matrix Multiplication

```
import numpy as np
A = np.array([[1, 2], [3, 4]]) # 2x2 matrix
B = np.array([[5, 6], [7, 8]]) # 2x2 matrix
C = A.dot(B)
print(C)
```

Output:

```
[[19 22]
 [43 50]]
```

Types of Matrices

Triangular Matrices

A triangular matrix is a special kind of square matrix where all the entries above the main diagonal are zero (lower triangular) or all the entries below the main diagonal are zero (upper triangular).
Example in Python:

```
import numpy as np
A = np.array([[1, 2, 3], [0, 4, 5], [0, 0, 6]]) # Upper triangular
B = np.array([[1, 0, 0], [4, 5, 0], [7, 8, 9]]) # Lower triangular
print("A = ", A)
print("B = ", B)
```

Types of Matrices

Identity Matrices

An identity matrix **I** is a square matrix in which all the elements of the principal diagonal are ones and all other elements are zeros.
Example in Python:

```
import numpy as np
I = np.eye(3) # 3x3 identity matrix
print("I = ", I)
```

Types of Matrices

Diagonal Matrices

A diagonal matrix is a matrix in which the entries outside the main diagonal are all zero.
Example in Python:

```
import numpy as np
D = np.diag([1, 2, 3]) # Diagonal matrix with diagonal elements 1, 2, 3
print("D = ", D)
```

Types of Matrices

Orthogonal Matrices

An orthogonal matrix is a square matrix whose rows and columns are orthogonal unit vectors (i.e., orthonormal vectors), making it a column and row-stochastic matrix. Orthogonal matrices have the property that their transpose is equal to their inverse, i.e.

$$Q^\top = Q^{-1} \quad \text{or} \quad Q \cdot Q^\top = I$$

Types of Matrices

Orthogonal Matrices: Example in Python

```
import numpy as np
Q = np.array([[1, 0], [0, -1]]) # An example of orthogonal matrix
print("Q = ", Q)
print("Q Transpose = ", Q.T)
print("Q Inverse = ", np.linalg.inv(Q))
```

Operations on Matrices

Transpose

The transpose of a matrix is obtained by interchanging its rows into columns or columns into rows. It is denoted by $\mathbf{A}^T$. For example, if

Operations on Matrices

Transpose: Example

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \tag{4}$$

then,

$$A^\top = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix} \tag{5}$$

Operations on Matrices

Transpose: Example in Python

```
import numpy as np
A = np.array([[1, 2, 3], [4, 5, 6]])
print("A = ", A)
print("Transpose of A = ", A.T)
```

Operations on Matrices

Determinant

The determinant is a special number that can be calculated from a square matrix. The determinant helps us find the inverse of a matrix, tells us things about the matrix that are useful in systems of linear equations, calculus and more.

Operations on Matrices

Determinant

The determinant of a matrix gives us important information about the matrix. For example, a matrix is invertible (i.e., has an inverse) if and only if its determinant is non-zero. The determinant of a matrix also gives us the scale factor by which area (or volume, in higher dimensions) is transformed under the linear transformation represented by the matrix.
It's important to note that the determinant is only defined for square matrices. For non-square matrices, related concepts like the rank or the pseudoinverse are often used instead.

Operations on Matrices

A 2x2 Determinant

The determinant of a matrix $\mathbf{A}$ is often denoted as $|\mathbf{A}|$ or $\det(\mathbf{A})$.
For a 2x2 matrix:

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

The determinant is calculated as:

$$|\mathbf{A}| = ad - bc$$

Operations on Matrices

Determinant

For a 3x3 matrix:

$$\mathbf{A} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

The determinant is calculated as:

$$|\mathbf{A}| = a(ei - fh) - b(di - gf) + c(dh - eg)$$

For larger square matrices (4x4 and above), the determinant is usually computed using more advanced methods like the Laplace expansion, or LU decomposition. The determinant of a matrix can also be calculated using the product of its eigenvalues.

Operations on Matrices

Determinant: Example in Python

```
A = np.array([[1, 2], [3, 4]])
print("A = ", A)
print("Determinant of A = ", np.linalg.det(A))
```

Operations on Matrices

Inverse

The inverse of a square matrix $\mathbf{A}$ is denoted as $\mathbf{A}^{-1}$, and it is the matrix such that when it is multiplied by $\mathbf{A}$, the result is the identity matrix.

Example in Python:

```
A = np.array([[4, 7], [2, 6]])
print("A = ", A)
print("Inverse of A = ", np.linalg.inv(A))
```

Operations on Matrices

Trace

The trace of a square matrix $\mathbf{A}$, denoted as $\text{tr}(\mathbf{A})$, is the sum of the elements on the main diagonal.

Example in Python:

```
A = np.array([[1, 2], [3, 4]])
print("A = ", A)
print("Trace of A = ", np.trace(A))
```

Operations on Matrices

Rank

The rank of a matrix **A** is the maximum number of linearly independent row vectors in the matrix.
Example in Python:

```
A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print("A = ", A)
print("Rank of A = ", np.linalg.matrix_rank(A))
```

The End