Basics of Linear Algebra for Machine Learning

7-Day Crash-Course

Jason Brownlee

MACHINE LEARNING MASTERY

Disclaimer

Basics of Linear Algebra for Machine Learning Crash Course

Edition: v1.7

Find the latest version of this guide online at: http://MachineLearningMastery.com

Contents

Before We Get Started...

Linear algebra is a field of mathematics that is universally agreed to be a prerequisite for a deeper understanding of machine learning. Although linear algebra is a large field with many esoteric theories and findings, the nuts and bolts tools and notations taken from the field are required for machine learning practitioners. With a solid foundation of what linear algebra is, it is possible to focus on just the good or relevant parts. In this crash course, you will discover how you can get started and confidently read and implement linear algebra notation used in machine learning with Python in 7 days. Let's get started.

Who Is This Crash-Course For?

Before we get started, let's make sure you are in the right place. This course is for developers that may know some applied machine learning. Maybe you know how to work through a predictive modeling problem end-to-end, or at least most of the main steps, with popular tools. The lessons in this course do assume a few things about you, such as:

You need to know:

- You know your way around basic Python for programming.
- You may know some basic NumPy for array manipulation.
- You want to learn linear algebra to deepen your understanding and application of machine learning.

You do NOT need to know:

- You do not need to be a math wiz!
- You do not need to be a machine learning expert!

This crash course will take you from a developer that knows a little machine learning to a developer who can navigate the basics of linear algebra. This crash course assumes you have a working Python3 SciPy environment with at least NumPy installed. If you need help with your environment, you can follow the step-by-step tutorial here:

- How to Setup a Python Environment for Machine Learning and Deep Learning

Crash-Course Overview

This crash course is broken down into 7 lessons. You could complete one lesson per day (recommended) or complete all of the lessons in one day (hardcore). It really depends on the time you have available and your level of enthusiasm. Below is a list of the 7 lessons that will get you started and productive with linear algebra for machine learning in Python:

- **Lesson 01**: Linear Algebra for Machine Learning.
- **Lesson 02**: Linear Algebra.
- **Lesson 03**: Vectors.
- **Lesson 04**: Matrices.
- **Lesson 05**: Matrix Types and Operations.
- **Lesson 06**: Matrix Factorization.
- **Lesson 07**: Singular-Value Decomposition.

Each lesson could take you 60 seconds or up to 30 minutes. Take your time and complete the lessons at your own pace. Ask questions and even post results online. The lessons expect you to go off and find out how to do things. I will give you hints, but part of the point of each lesson is to force you to learn where to go to look for help on and about the linear algebra and the NumPy API and the best-of-breed tools in Python (hint, I have all of the answers directly on this blog, use the search box). Post your results online, I'll cheer you on!

Hang in there, don't give up!

Lesson 01: Linear Algebra for Machine Learning

In this lesson, you will discover the 5 reasons why a machine learning practitioner should deepen their understanding of linear algebra.

You Need to Learn Linear Algebra Notation

You need to be able to read and write vector and matrix notation. Algorithms are described in books, papers and on websites using vector and matrix notation.

You Need to Learn Linear Algebra Arithmetic

In partnership with the notation of linear algebra are the arithmetic operations performed. You need to know how to add, subtract, and multiply scalars, vectors, and matrices.

You Need to Learn Linear Algebra for Statistics

You must learn linear algebra in order to be able to learn statistics. Especially multivariate statistics. In order to be able to read and interpret statistics, you must learn the notation and operations of linear algebra. Modern statistics uses both the notation and tools of linear algebra to describe the tools and techniques of statistical methods. From vectors for the means and variances of data, to covariance matrices that describe the relationships between multiple Gaussian variables.

You Need to Learn Matrix Factorization

Building on notation and arithmetic is the idea of matrix factorization, also called matrix decomposition.You need to know how to factorize a matrix and what it means. Matrix factorization is a key tool in linear algebra and used widely as an element of many more complex operations in both linear algebra (such as the matrix inverse) and machine learning (least squares).

You Need to Learn Linear Least Squares

You need to know how to use matrix factorization to solve linear least squares. Problems of this type can be framed as the minimization of squared error, called least squares, and can be recast in the language of linear algebra, called linear least squares. Linear least squares problems can be solved efficiently on computers using matrix operations such as matrix factorization.

One More Reason

If I could give one more reason, it would be: because it is fun. Seriously.

Your Task

For this lesson, you must list 3 reasons why you, personally, want to learn linear algebra.

Next

In the next lesson, you will discover a concise definition of linear algebra.

Lesson 02: Linear Algebra

In this lesson, you will discover a concise definition of linear algebra.

Linear Algebra

Linear algebra is a branch of mathematics, but the truth of it is that linear algebra is the mathematics of data. Matrices and vectors are the language of data. Linear algebra is about linear combinations. That is, using arithmetic on columns of numbers called vectors and 2D arrays of numbers called matrices, to create new columns and arrays of numbers.

Numerical Linear Algebra

The application of linear algebra in computers is often called numerical linear algebra. It is more than just the implementation of linear algebra operations in code libraries; it also includes the careful handling of the problems of applied mathematics, such as working with the limited floating point precision of digital computers.

Applications of Linear Algebra

As linear algebra is the mathematics of data, the tools of linear algebra are used in many domains.

- **Matrices in Engineering**, such as a line of springs.
- **Graphs and Networks**, such as analyzing networks.
- **Markov Matrices**, Population, and Economics, such as population growth.
- **Linear Programming**, the simplex optimization method.
- **Fourier Series**, Linear Algebra for functions, used widely in signal processing.
- **Linear Algebra for statistics and probability**, such as least squares for regression.
- **Computer Graphics**, such as the various translation, scaling and rotation of images.

Your Task

For this lesson, you must find five quotes from research papers, blogs or books that define the field of linear algebra.

Next

In the next lesson, you will discover vectors and simple vector arithmetic.

Lesson 03: Vectors

In this lesson, you will discover vectors and simple vector arithmetic.

What is a Vector?

A vector is a tuple of one or more values called scalars. Vectors are often represented using a lowercase character such as v; for example:

$$v = (v_1, v_2, v_3) \tag{1}$$

Where v_1, v_2 and v_3 are scalar values, often real values.

Defining a Vector

We can represent a vector in Python as a NumPy array. A NumPy array can be created from a list of numbers. For example, below we define a vector with the length of 3 and the integer values 1, 2 and 3.

```
# create a vector
from numpy import array
v = array([1, 2, 3])
print(v)
```

Listing 1: Example of creating a vector.

Vector Multiplication

Two vectors of equal length can be multiplied together.

$$c = a \times b \tag{2}$$

As with addition and subtraction, this operation is performed element-wise to result in a new vector of the same length.

$$a \times b = (a_1 \times b_1, a_2 \times b_2, a_3 \times b_3) \tag{3}$$

We can perform this operation directly in NumPy.

```
# multiply vectors
from numpy import array
a = array([1, 2, 3])
print(a)
b = array([1, 2, 3])
print(b)
c = a * b
print(c)
```

Listing 2: Example of vector multiplication.

Your Task

For this lesson, you must implement other vector arithmetic operations such as addition, division, subtraction and the vector dot product.

Next

In the next lesson, you will discover matrices and simple matrix arithmetic.

Lesson 04: Matrices

In this lesson, you will discover matrices and simple matrix arithmetic.

What is a Matrix?

A matrix is a two-dimensional array of scalars with one or more columns and one or more rows. The notation for a matrix is often an uppercase letter, such as A, and entries are referred to by their two-dimensional subscript of row (i) and column (j), such as $a_{i,j}$. For example:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \\ a_{3,1} & a_{3,2} \end{pmatrix} \tag{4}$$

Defining a Matrix

We can represent a matrix in Python using a two-dimensional NumPy array. A NumPy array can be constructed given a list of lists. For example, below is a 2 row, 3 column matrix.

```
# create matrix
from numpy import array
A = array([[1, 2, 3], [4, 5, 6]])
print(A)
```

Listing 3: Create a matrix.

Matrix Addition

Two matrices with the same dimensions can be added together to create a new third matrix.

$$C = A + B \tag{5}$$

The scalar elements in the resulting matrix are calculated as the addition of the elements in each of the matrices being added. We can implement this in Python using the plus operator directly on the two NumPy arrays.

```
# add matrices
from numpy import array
A = array([[1, 2, 3], [4, 5, 6]])
print(A)
B = array([[1, 2, 3], [4, 5, 6]])
```

```
print(B)
C = A + B
print(C)
```

Listing 4: Add matrices.

Matrix Dot Product

Matrix multiplication, also called the matrix dot product is more complicated than the previous operations and involves a rule as not all matrices can be multiplied together.

$$C = A \times B \tag{6}$$

The rule for matrix multiplication is as follows: The number of columns (n) in the first matrix (A) must equal the number of rows (m) in the second matrix (B). For example, matrix A has the dimensions m rows and n columns and matrix B has the dimensions n and k. The n columns in A and n rows b are equal. The result is a new matrix with m rows and k columns.

$$C(m, k) = A(m, n) \times B(n, k) \tag{7}$$

The intuition for the matrix multiplication is that we are calculating the dot product between each row in matrix A with each column in matrix B. For example, we can step down rows of column A and multiply each with column 1 in B to give the scalar values in column 1 of C. The matrix multiplication operation can be implemented in NumPy using the `dot()` function.

```
# matrix dot product
from numpy import array
A = array([[1, 2], [3, 4], [5, 6]])
print(A)
B = array([[1, 2], [3, 4]])
print(B)
C = A.dot(B)
print(C)
```

Listing 5: Multiply matrices.

Your Task

For this lesson, you must implement more matrix arithmetic operations such as subtraction, division, the Hadamard product and vector-matrix multiplication.

Next

In the next lesson, you will discover the different types of matrices and matrix operations.

Lesson 05: Matrix Types and Operations

In this lesson, you will discover the different types of matrices and matrix operations.

Transpose

A defined matrix can be transposed, which creates a new matrix with the number of columns and rows flipped. This is denoted by the superscript T next to the matrix.

$$C = A^T \tag{8}$$

We can transpose a matrix in NumPy by calling the T attribute.

```
# transpose matrix
from numpy import array
A = array([[1, 2], [3, 4], [5, 6]])
print(A)
C = A.T
print(C)
```

Listing 6: Transpose a matrix.

Inversion

The operation of inverting a matrix is indicated by a -1 superscript next to the matrix; for example, A^{-1}. The result of the operation is referred to as the inverse of the original matrix; for example, B is the inverse of A.

$$B = A^{-1} \tag{9}$$

Not all matrices are invertible. A matrix can be inverted in NumPy using the `inv()` function.

```
# invert matrix
from numpy import array
from numpy.linalg import inv
# define matrix
A = array([[1.0, 2.0], [3.0, 4.0]])
print(A)
# invert matrix
B = inv(A)
print(B)
```

Listing 7: Invert a matrix.

Square Matrix

A square matrix is a matrix where the number of rows (n) equals the number of columns (m).

$$n = m \tag{10}$$

The square matrix is contrasted with the rectangular matrix where the number of rows and columns are not equal.

Symmetric Matrix

A symmetric matrix is a type of square matrix where the top-right triangle is the same as the bottom-left triangle. To be symmetric, the axis of symmetry is always the main diagonal of the matrix, from the top left to the bottom right. A symmetric matrix is always square and equal to its own transpose.

$$M = M^T \tag{11}$$

Triangular Matrix

A triangular matrix is a type of square matrix that has all values in the upper-right or lower-left of the matrix with the remaining elements filled with zero values. A triangular matrix with values only above the main diagonal is called an upper triangular matrix. Whereas, a triangular matrix with values only below the main diagonal is called a lower triangular matrix.

Diagonal Matrix

A diagonal matrix is one where values outside of the main diagonal have a zero value, where the main diagonal is taken from the top left of the matrix to the bottom right. A diagonal matrix is often denoted with the variable D and may be represented as a full matrix or as a vector of values on the main diagonal.

Your Task

For this lesson, you must develop examples for other matrix operations such as the determinant, trace and rank.

Next

In the next lesson, you will discover matrix factorization.

Lesson 06: Matrix Factorization

In this lesson, you will discover the basics of matrix factorization also called matrix decomposition.

What is a Matrix Decomposition?

A matrix decomposition is a way of reducing a matrix into its constituent parts. It is an approach that can simplify more complex matrix operations that can be performed on the decomposed matrix rather than on the original matrix itself. A common analogy for matrix decomposition is the factoring of numbers, such as the factoring of 25 into 5×5. For this reason, matrix decomposition is also called matrix factorization. Like factoring real values, there are many ways to decompose a matrix, hence there are a range of different matrix decomposition techniques.

LU Matrix Decomposition

The LU decomposition is for square matrices and decomposes a matrix into L and U components.

$$A = L \cdot U \tag{12}$$

Where A is the square matrix that we wish to decompose, L is the lower triangle matrix and U is the upper triangle matrix. A variation of this decomposition that is numerically more stable to solve in practice is called the LUP decomposition, or the LU decomposition with partial pivoting.

$$A = P \cdot L \cdot U \tag{13}$$

The rows of the parent matrix are re-ordered to simplify the decomposition process and the additional P matrix specifies a way to permute the result or return the result to the original order. There are also other variations of the LU. The LU decomposition is often used to simplify the solving of systems of linear equations, such as finding the coefficients in a linear regression. The LU decomposition can be implemented in Python with the `lu()` function. More specifically, this function calculates an LPU decomposition.

```
# LU decomposition
from numpy import array
from scipy.linalg import lu
# define a square matrix
A = array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print(A)
# LU decomposition
```

```
P, L, U = lu(A)
print(P)
print(L)
print(U)
# reconstruct
B = P.dot(L).dot(U)
print(B)
```

Listing 8: LU Decomposition.

Your Task

For this lesson, you must implement small examples of other simple methods for matrix factorization, such as the QR decomposition, the Cholesky decomposition and the eigendecomposition.

Next

In the next lesson, you will discover the Singular-value Decomposition method for matrix factorization.

Lesson 07: Singular-Value Decomposition

In this lesson, you will discover the Singular-value Decomposition method for matrix factorization.

Singular-Value Decomposition

The Singular-Value Decomposition, or SVD for short, is a matrix decomposition method for reducing a matrix to its constituent parts in order to make certain subsequent matrix calculations simpler.

$$A = U \cdot \Sigma \cdot V^T \tag{14}$$

Where A is the real $m \times n$ matrix that we wish to decompose, U is an $m \times m$ matrix, Σ (sigma) is an $m \times n$ diagonal matrix, and V^T is the transpose of an $n \times n$ matrix where T is a superscript.

Calculate Singular-Value Decomposition

The SVD can be calculated by calling the `svd()` function. The function takes a matrix and returns the U, Σ and V^T elements. The Σ diagonal matrix is returned as a vector of singular values. The V matrix is returned in a transposed form, e.g. `V.T`.

```
# Singular-value decomposition
from numpy import array
from scipy.linalg import svd
# define a matrix
A = array([[1, 2], [3, 4], [5, 6]])
print(A)
# SVD
U, s, V = svd(A)
print(U)
print(s)
print(V)
```

Listing 9: Singular-value Decomposition.

Your Task

For this lesson, you must list 5 applications of the SVD. Bonus points if you can demonstrate each with a small example in Python.

Final Word Before You Go...

You made it. Well done! Take a moment and look back at how far you have come. You discovered:

- The importance of linear algebra to applied machine learning.
- What linear algebra is all about.
- What a vector is and how to perform vector arithmetic.
- What a matrix is and how to perform matrix arithmetic, including matrix multiplication.
- A suite of types of matrices, their properties, and advanced operations involving matrices.
- Matrix factorization methods and the LU decomposition method in detail.
- The popular Singular-Value decomposition method used in machine learning.

This is just the beginning of your journey with linear algebra for machine learning. Keep practicing and developing your skills. Take the next step and check out my book on Linear Algebra for Machine Learning.

How Did You Go With The Crash-Course?

Did you enjoy this crash-course?
Do you have any questions or sticking points?

Let me know, send me an email at: **jason@MachineLearningMastery.com**

Take the Next Step

Looking for more help with Linear Algebra for Machine Learning?

Grab my new book:
Basics of Linear Algebra for Machine Learning
https://machinelearningmastery.com/linear_algebra_for_machine_learning/