Linear Algebra

StartOnAI Tutorial 2

This is a tutorial that develops a firm foundation for understanding concepts in the mathematical world of Linear Algebra and its relevance and applications to Machine Learning (ML). Linear Algebra itself is extremely important to machine learning as it is a concise way to describe coordinates and interactions of different planes when involving a multitude of dimensions when operations are performed. This tutorial will cover concepts such as linear algebra fundamentals, vectors, matrix multiplication, and eigenvectors.

1. What is the Linear Algebra?

Linear Algebra is essentially an extended version of fundamental algebra that is taught at the baseline level of mathematics all around the planet. It involves solving for unknown attributes with the information that is given, except linear algebra takes it into some arbitrary number of dimensions in space. The data represented in linear equations in basic algebra is now transformed into matrices and vectors to be able to solve for more complex and useful attributes. On another note, Linear Algebra is extremely useful for finding and implementing various ways to read and enter statistics into an organized fashion due to its vector and scalar "storage" networks. For example, regressions, which are measurements of the strength and relationship between multiple variables, are an extremely useful tool that is provided within Linear Algebra and Machine Learning itself. Tasks that Linear Algebra make much easier include Regularization, Singular-Value Decompositions, Latent Semantic Analysis, Recommender System, One-Hot Encoding and may more!

2. Understanding Vectors, Matrices, and Eigenvectors

Introduction – Our journey of Linear Algebra begins with us putting together the basic concepts beginning with what basic algebra is, and the definition of vectors and matrices.

$$\begin{aligned} 4x_1 &- 5x_2 = -13 \\ -2x_1 &+ 3x_2 = 9. \end{aligned}$$

Take this basic system of linear equations defined by the 2 variables x1 and x2 (we normally would see x and y but this is the same concept). We see how we can manipulate either equation to be able to find out the values of both x1 and x2 which will result in a satisfied system of linear equations. Now we progress to the world of matrices, doing the same exact calculation with a matrix makes the mathematics so much easier and applicable to real-life scenarios.

$$Ax = b$$

$$A = \begin{bmatrix} 4 & -5 \\ -2 & 3 \end{bmatrix}, \quad b = \begin{bmatrix} -13 \\ 9 \end{bmatrix}.$$

We can write out this system with much more ease using this creating a matrix A with our coefficients, and a matrix b that holds our real number results. We can now define an arbitrary matrix x that will hold the solutions to both x1 and x2 which when matrix properties are applied to, yield us the same answer while using substitution or elimination of simple linear equations. The interesting part about matrices is that if we had said 5 unknown variables with 5 equations this method works just as well, however if one tried to solve that by hand using substitution or elimination, that would take much longer.

Let us now define some key symbols that will help us establish ground for different types of vectors and matrices.

$$\cdot A \in \mathbb{R}^{m \times n}$$

This symbol here tells us that matrix A is composed of m rows and n columns, and the values of this matrix are all real numbers, which is why we have the bold R.

$$x \in \mathbb{R}^n,$$

This symbol denotes that a vector that we choose to call x, has said n number of entries, and these entries are all real numbers as well. This n-dimension vector is generalized to have n rows and 1 column, however if we want to flip the orientation of the vector to be sideways with n columns and one row, we would use the notation below.

$$x^T$$

This denotes the exact same vector x that retains all the same values, except this time, the vector is transposed (denoted by the T) in such a way that it has only 1 row.

Now we can move onto the format of a matrix.

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}.$$

Displayed above is a matrix A that contains values in the i-th row and j-th column. We can also translate this to a number system where 12 represents the an "a" value on the first row second column and vice versa for a point on the matrix noted as "a" sub 21. This lets us organize the data efficiently in 2 dimensions so we can easily select the required data point in the matrix at any given instance.

Matrix and Vector Multiplication – We will now begin understanding conceptually how 2 vectors and or matrices are combined together in many formats.

Vector-Vector Products – Vector products are the combination of 2 different 1-dimensional matrices or vectors, that are combined together to get what is called the dot product.

$$x^T y \in \mathbb{R} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \sum_{i=1}^{n} x_i y_i.$$

Here we can see that we are multiplying a transposed vector x being multiplied by a vector y, and the result is the complicated expression on the far right. Let us dissect this, as it seems that all that is happening is that for every x term, we are multiplying it by its corresponding y term which means for the 5^{th} x value, we would multiply that by the 5^{th} y value and so on. The sigma tells us that we will eventually be summing up all the xy combination pairs until we get one result that will be some sort of real number, and this result is our dot product.

Before we dive further and into true matrix multiplication with 2 dimensional matrices, let us observe some key properties of matrices so that we can conceptually understand what happens behind the scenes when they are multiplied.

Matrix multiplication is associative: $(AB)C = A(BC)$.

Matrix multiplication is distributive: $A(B + C) = AB + AC$.

Matrix multiplication is, in general, *not* commutative; that is, it can be the case that $AB \neq BA$. (For example, if $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times q}$, the matrix product BA does not even exist if m and q are not equal!)

First of all, matrices are multiplied associatively which means that the order in which we combine two matrices does not affect the end result in any way. Similarly, we see the distributive property also being valid for matrices, as we can easily distribute a matrix A to B + C without changing the end result of the outcome. Finally, we see an interesting property that matrix multiplication is in fact not commutative, which means the order in which we multiply 2 matrices actually does affect our end result, and this is a very important concept to keep in mind. This being said, let us move on to Matrix-Vector Products.

Matrix-Vector Products – We will now discover that if given a matrix A and a vector x, we are able to combine them together using a matrix equation that is shown below.

$$y = Ax = \begin{bmatrix} — & a_1^T & — \\ — & a_2^T & — \\ & \vdots & \\ — & a_m^T & — \end{bmatrix} x = \begin{bmatrix} a_1^T x \\ a_2^T x \\ \vdots \\ a_m^T x \end{bmatrix}.$$

To debrief what is truly happening here let us first look at the first part of our equation, y = Ax. Now what we realize is obviously our vector is only one dimensional, so what we can do is denote matrix A in terms of rows by transposing it, as we can see from the first part of the equation. Now we are able to normally multiply together the i-th a value by the i-th x value to combine them into our product. The conversion is highlighted above is shown in the formula below.

$$y_i = a_i^T x.$$

To put this into mathematical terms, the i-th y term is calculated by the inner product of the i-th row of A and x respectively.

We can also manipulate this matrix-vector combination to be in terms of columns as we can see below.

$$y = Ax = \begin{bmatrix} | & | & & | \\ a_1 & a_2 & \cdots & a_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \\ a_1 \\ \\ \end{bmatrix} x_1 + \begin{bmatrix} \\ a_2 \\ \\ \end{bmatrix} x_2 + \ldots + \begin{bmatrix} \\ a_n \\ \\ \end{bmatrix} x_n \ .$$

This shows us that even if we arrange the matrix in a different format than was originally established, we are returning the same product. In this scenario, we are simply creating a "dot product" like structure except instead of multiplying a value by a value from 2 separate vectors and summing them up for the n terms that there are, we are multiplying a matrix by a vector value for the n matrices and vector values that were stored originally.
This result is called the linear combination result because the coefficients of this linear combination are given by the values of x in the vector.

Matrix-Matrix Products – We transition now to combining 2 matrices using matrix multiplication as a set of 2 vector products. This means that from our definition of matrix multiplication, the (i, j)th entry of our resulting matrix C is simplify the inner product from the i-th row of matrix A multiplied by the j-th column of matrix B. We can visualize this symbolically with the following diagram below.

$$C = AB = \begin{bmatrix} — & a_1^T & — \\ — & a_2^T & — \\ & \vdots & \\ — & a_m^T & — \end{bmatrix} \begin{bmatrix} | & | & & | \\ b_1 & b_2 & \cdots & b_p \\ | & | & & | \end{bmatrix} = \begin{bmatrix} a_1^T b_1 & a_1^T b_2 & \cdots & a_1^T b_p \\ a_2^T b_1 & a_2^T b_2 & \cdots & a_2^T b_p \\ \vdots & \vdots & \ddots & \vdots \\ a_m^T b_1 & a_m^T b_2 & \cdots & a_m^T b_p \end{bmatrix}.$$

We can see that we are simply taking the traversed matrix A by the column version of matrix B and combining them to form matrix C with the specifications given the definitions of what matrix multiplication truly are, which is where the number of column in matrix A must equal the number of rows in matrix C and vice versa for matrix B and C.

From the representation we just made, we can invert it to have a columnar representation of matrix A and a row representation of matrix B which gives us insight on how the matrices can be transposed without a change in result.

$$C = AB = \begin{bmatrix} | & | & & | \\ a_1 & a_2 & \cdots & a_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} — & b_1^T & — \\ — & b_2^T & — \\ & \vdots & \\ — & b_n^T & — \end{bmatrix} = \sum_{i=1}^{n} a_i b_i^T$$

From viewing this representation we can see how all that we have changes is the fact that now we are taking the transposed version of matrix b and summing all the possible combinations of the a sub i-th term and b sub i-th terms similar to the dot product expression we had done at the very beginning of this tutorial.

Amazingly we actually have 2 more representations that we can use to computer matrix multiplication. The first of the two is displayed below.

$$C = AB = A \begin{bmatrix} | & | & & | \\ b_1 & b_2 & \cdots & b_p \\ | & | & & | \end{bmatrix} = \begin{bmatrix} | & | & & | \\ Ab_1 & Ab_2 & \cdots & Ab_p \\ | & | & & | \end{bmatrix}.$$

What we notice here is that the value of matrix A times matrix B is the sum of the outer product of the i-th column of matrix A with the i-th row of matrix B. We notice that the only thing here that is appearing different is the symbolism that we show our matrix multiplication in. Rather than showing the true distribution in terms of the values "a" and "b" we are simply placing matrix A into the columns of matrix B and using the distributive property that we talked about earlier for matrices. Let us look at this property in a new light with row orientation.

$$C = AB = \begin{bmatrix} — & a_1^T & — \\ — & a_2^T & — \\ & \vdots & \\ — & a_m^T & — \end{bmatrix} B = \begin{bmatrix} — & a_1^T B & — \\ — & a_2^T B & — \\ & \vdots & \\ — & a_m^T B & — \end{bmatrix}.$$

As we see here our representation transposed and instead of distributing matrix A we are now distributing matrix B, which makes sense because we are viewing our matrix product of C as the matrix-vector product between the rows of matrix A and matrix C, which results in us having to distribute matrix B to satisfy the equation the we begun with (C = AB). Here is i-th row of matrix C is the matrix-vector product with the vector on the left side as shown below.

$$c_i^T = a_i^T B.$$

Eigenvectors – Simply put, an eigenvector is a vector that remains on its own span, which is a line passing through the origin and tip of a resultant vector between two matrices being combined together as vectors. For most vectors, this resultant vector would not lie on the span line, but when this does occur, the result is something called an eigenvector that has an associated eigenvalue attached to it. This eigenvalue is the factor that any vectors along the span line will

be stretched by, for example if we had a matrix Y with top row 3 and bottom row 0, the eigenvalue on the vector formed would be 3 because any vector lying on the x-axis (which is where the vector created by Y) will have to be multiplied by this eigenvalue. The generalized equation for eigenvectors is given below.

$$Ax = \lambda x, \quad x \neq 0.$$

Dissecting this we understand that A is a matrix that has a certain linear transformation to it while x represents a certain eigenvector with lambda a corresponding eigenvalue for that eigenvector. What this equality represents is that a matrix A representing some transformation multiplied by an eigenvector is the same as scaling an eigenvector itself (multiplying by the eigenvalue lambda). This seems awkward at first since the left-hand side is representing matrix multiplication while the right-hand side is scalar multiplication.

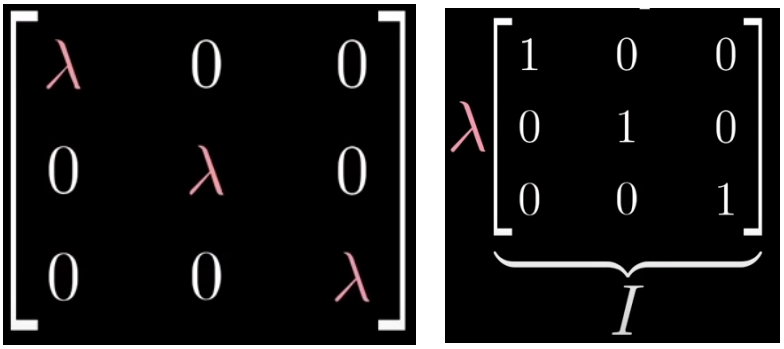
Let us rewrite the equation we just created to state that (λ, x) is an eigenvalue-eigenvector pair of A if the following equation yields true.

$$(\lambda I - A)x = 0, \quad x \neq 0.$$

This equation leads us to the property that in fact lambda times the I value – the matrix A is equal to 0 which is represented below.

$$|(\lambda I - A)| = 0.$$

If the last 2 equations did not come intuitively to you, let us dissect how we got from the generalized eigen-matrix equation to this final equality here.

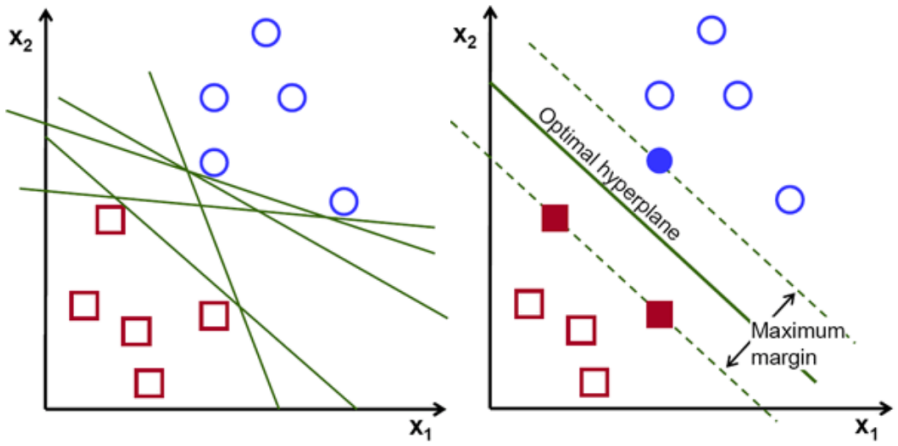


On the left hand-side we can imagine that since we are just scaling a vector, we can simply scale an original matrix by the value lambda and showcase that inside the actual matrix as each basis vector inside the matrix is being multiplied by lambda. Now transition to the right-hand side and notice that we are factoring out our lambda and multiply this lambda by I, which is the identity matrix with 1's down the diagonal. This can be seen in second eigen-matrix equation we created which we set equal to 0 using some basic level algebra.
Finally, we can use these definitions to expand into a large polynomial expression in terms of lambda, where lambda's maximum degree is n. We can then find the n (possible complex) roots of this polynomial to find th n eigenvalues λ1. . .. λn. Then to find the corresponding eigenvector using these eigenvalues, we simplify have to solve the linear equation below.

$$(\lambda_i I - A)x = 0.$$

3. Concepts of Linear Algebra Useful to Machine Learning

Concept 1 – Vectors and ML – Vectors are an extremely useful tool in terms of organizing data due to their convenient structures. When one begins to learn about machine learning, the first thing they will tend to do is make the data vectorized. Vectors are used in machine learning as support vector machines, which is a collection of supervised learning algorithms that use hyperplane graphing to analyze new possible scenarios for any given data. Essentially, this support vector machine attempts to find a line that has the optimal distance between data sets of 2 classes, namely this optimal distance is a maximum distance as highlighted in the diagram below.



Concept 2 – Matrices and ML – Matrices are extremely useful ways to store and manipulate data in machine learning. This task of data manipulation is extremely important because that is the basis of what machine learning revolves around. The Python coding language involved many libraries where these matrix transformations can be made, and this is absolutely necessary for a machine learning algorithm to store and process data as efficiently as possible. These matrices are effectively data structures which allow an organized fashion in completing any sort of computational transaction.

Concept 3 – Eigenvectors and ML – Let us first realize that the word Eigen is derived from a German word that means characteristic. This precisely describes what and eigenvector and eigenvalue are as they are derived characteristics from vectors and matrices. Eigenvectors help with processes such as Principal Component Analysis which is useful for data compression so that the machine learning processes can learn to deal with images of all sorts. Another useful implementation is Spectral Clustering in which the most popular algorithm is the K-Means algorithm. Eigenvectors of a matrix are used to find "K" clusters of data. These processes truly make data analysis much more practical and innate.

4. References

A. Brownlee, J. (2019, August 9). Introduction to Matrices and Matrix Arithmetic for Machine Learning.

B. Brownlee, J. (2019, August 9). 10 Examples of Linear Algebra in Machine Learning.

C. Vector. (2019, May 17).

D. Colter, Z., & Do, C. Linear Algebra Review and Reference (2015, September 30).

E. Donges, N. (2019, October 8). Basic Linear Algebra for Deep Learning.

F. Singh, R. (2020, January 4). The essence of eigenvalues and eigenvectors in Machine Learning.