1 Linear algebra and machine learning

Linear algebra is a subfield of mathematics concerned with vectors, matrices and linear transformations.

It is a key foundation to the field of Data Science from notations used to describe the operation of algorithms, to the implementation of algorithms in code. Although linear algebra is integral to the field of Data Science, the tight relationship is often left unexplained.

When people think of the field of Data Science in general, or of specific areas of it, such as Natural Language Processes, Machine Learning, or Computer Vision, they rarely take Linear Algebra in serious.

The reason linear algebra is often overlooked is that tools used today to implement data science algorithms do an excellent job in hiding the underlying mathematics that make everything come true naturally.

Most of the time, people avoid getting into linear algebra because it's "difficult" or "hard to understand." Although partly true, **being familiar with linear algebra is an essential skill for data scientists**.

One of the essential ingredients of Data Science is Machine Learning.

What is Machine Learning ?
Or what is Machine Learning about ?

ML is about discovering structures and patterns that exist in a set of 'things'. This is done using the language of mathematics, so we have to translate each 'thing' into numbers somehow.

Machine learning is, without a doubt, the most known application of artificial intelligence. The main idea behind machine learning is giving systems the power to automatically learn and improve from experience without being explicitly programmed to do so. Machine learning functions through building programs that have access to data (constant or updated) to analyze, find patterns and learn from. Once the programs discover relationships in the data, it applies this knowledge to new sets of data.

For example a single number can't sum up all the relevant facts about a thing very well; normally 'interesting' things are more complex. Instead we take some number of different measurements on each thing, and collect them into a vector of numbers that stands in for the thing itself.

1.1 Storing an image

What do you see when you look at the image above? You most likely said flower, leaves – not too difficult. But, if you are asked to write that logic so that a computer can do the same for you it will be a very difficult task (to say the least).

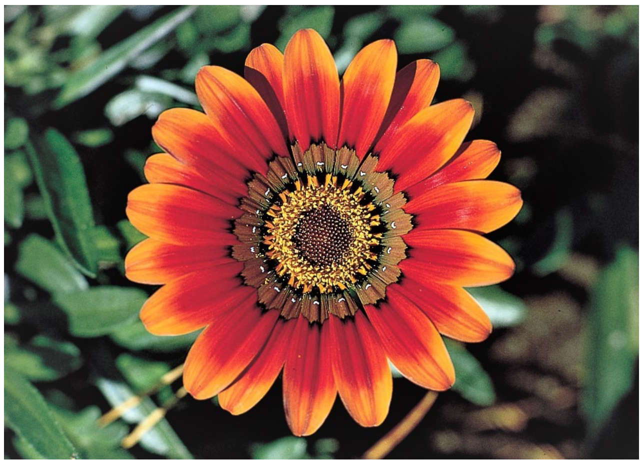


Figure 1: A flower

You were able to identify the flower because the human brain has gone through million years of evolution. We do not understand what goes in the background to be able to tell whether the colour in the picture is red or black. We have somehow trained our brains to automatically perform this task.

But making a computer do the same task is not an easy task, and is an active area of research in Machine Learning and Computer Science in general. Let us ponder over a particular question:

How does a machine stores this image?

You probably know that computers of today are designed to process only 0 and 1. So how can an image such as above with multiple attributes like colour be stored in a computer? This is achieved by storing the pixel intensities in a construct called Matrix. Then, this matrix can be processed to identify colours etc.

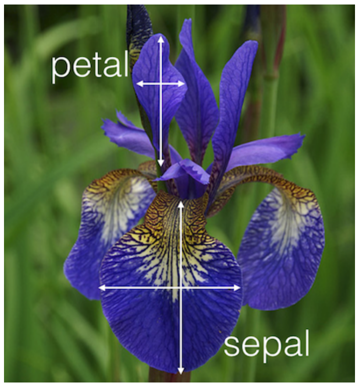
So any operation which you want to perform on this image would likely use Linear Algebra and matrices at the backstage.

1.2 Dataset and Data Files

In machine learning, we fit a model on a dataset. This is the table like set of numbers where each row represents an observation and each column represents a feature of the observation. For example, below is a snippet of the Iris flowers dataset:

```
5.1,  3.5,  1.4,  0.2,  Iris-setosa
4.9,  3.0,  1.4,  0.2,  Iris-setosa
4.7,  3.2,  1.3,  0.2,  Iris-setosa
4.6,  3.1,  1.5,  0.2,  Iris-setosa
5.0,  3.6,  1.4,  0.2,  Iris-setosa
 ...
```

Sample output of the iris flowers dataset.



This data is in fact a matrix, a key data structure in linear algebra. We have, instead of a set of abstract 'things', a set of vectors and we can do math on them. Representing stuff as points in a vector space is convenient. It's easy to map attributes of the data to dimensions in the vector space.

In order to train a machine, you'll typically be using many multiple such training vectors. This creates a series of vectors next to each other, which is (drum roll) a matrix. If you are doing neural networks, you may have something like m training examples, each of which is a vector of length n. Then you have at least one layer of r hidden units, so to do the forward pass, you have to multiply the $[m \times n]$ input matrix by a $[n \times r]$ weight matrix, put the outputs through a sigmoid function, then multiply the $[m \times r]$ hidden layer results by an $[r \times 1]$ matrix (assuming you have one output). Then you might use backdrop so you have to be able to figure out the derivatives of these big matrixes and so forth.

1.3 Dealing with Text

Another active area of research in Machine Learning is **dealing with text** and the most common techniques employed are Bag of Words, Term Document Matrix etc. All these techniques in a very similar manner store counts(or something similar) of words in documents and store this frequency count in a Matrix form to perform tasks like Semantic analysis, Language translation, Language generation etc.

Lastly a lot of ML concepts are tied to linear algebra concepts. Some basic examples, PCA - eigenvalue, regression - matrix multiplication ... As most ML techniques deal with high dimensional data, they are often times represented as matrices. To do all the things mentioned above you need to understand some linear algebra. At a more advanced level, reproducing kernel Hilbert spaces are often used in machine learning (Gaussian process, support vector machine, kernel PCA, etc).

So, now you understand the importance of Linear Algebra in machine learning.

1.4 Representation of problems in Linear Algebra

Let's start with a simple problem. Suppose that price of 1 ball and 2 bat or 2 ball and 1 bat is Rs 100. We need to find price of a ball and a bat.

Suppose the price of a bat is Rs x and the price of a ball is Rs y. Values of x and y can be anything depending on the situation i.e. x and y are variables. Let's translate this in mathematical form

$$2x + y = 100 \tag{1}$$

Similarly, for the second condition

$$x + 2y = 100 \tag{2}$$

Now, to find the prices of bat and ball, we need the values of x and y such that it satisfies both the equations. The basic problem of linear algebra is to find these values of x and y i.e. the solution of a set of linear equations.

Broadly speaking, in linear algebra data is represented in the form of linear equations. These linear equations are in turn represented in the form of matrices and vectors as a single object.

$$\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 100 \\ 100 \end{pmatrix}.$$

1.5 Representing data as flat tables versus matrices and graphs

The idea of a flat table is a very general and common way to think about data. It is a table, in that it has several rows or records and one or more columns describing properties of the thing described by that row/record; and it is called **flat** since there is usually no structural relationships between the rows/records.

Here is an example of a flat table.

R.No.	Name	Major	HW1	HW2	HW3	HW4	Test1	Test2	Grade
102	Alice	Math	95	90	70		100		
143	Bob	Comp	Sci	80	65		80		
205	Bob	Undecided	50	60	60		70		
216	Charlotte	Stats	85	70	55		80		
...									
245&Zaccheus	Undecided	100	70	55		70			

Note that the elements of this table could be represented by two indices. So, if we wanted to call the table A, then we could represent an element of the table by a_{ij}. That can be helpful sometimes, but it's power is limited.

In particular, there aren't many mathematical operations defined on this table A or on the rows i or columns j that describe interactions between parts of the table in a rich way. For example, one could add the numbers in the HW1 column field, e.g., to get the average score on that homework, but it doesn't make much sense to add "Alice" and "Bob" and so on.

Truly speaking, while this table looks like a matrix, it isn't "really" a matrix – since it isn't defined as a thing that has two subscripts, but instead by the operations that are allowed on it. This is true for a flat table, but there the operations were rather limited. For matrices and graphs, the operations will be much richer.

Question: For the tables above, what operations are allowed?

Answer: There are many, but they are typically combinations of a small number of primitive operations.

Examples of those primitives are the following.

- **Query.** Here, e.g., we might want to ask if a word appeared in a row or it appeared more than a certain number of times.
- **Filter.** Here, e.g., we might want to select just those rows where a given word appeared or appeared more than a certain number of times.
- **Join.** Here, e.g., we might want to combine two rows into one, which might be of interest if we mistakenly split data about an object into two.
- **Count or Sum.** Here, e.g., we might want to count the number of words or the number of times a given word appears.

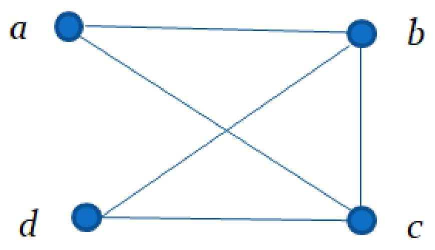
Flat tables and extensions of them are very important in data science. Here are two places in particular where they are widely used.

- **In databases.** In databases, i.e., that area of computer science that studies how to store, manipulate, etc. data, they are very common, in particular when the data are very large. In particular, if the data are really large then they are often stored somewhere in a database that the data scientist can access with queries and related operations.
- **For simple operations.** When you are first exploring a new data set, even if it is rather small. Often you want to see what you have, do a few simple operations to determine the size and shape of the data, do some initial visualization, etc., to determine whether there is anything crazy, outliers, etc.

1.6 Graphs and connections with matrices

Definition. *A graph $G = (V, E)$ consists of a nonempty set V of* ***vertices*** *(or* ***nodes****) and a set E of* ***edges****. Each edge has either one or two vertices associated with it, called its* ***endpoints****. An edge is said to* ***connect its endpoints****.*

Example:
This is a graph with four vertices and five edges.



Using graph models, we can

- determine whether it is possible to walk down all the streets in a city without going down a street twice,

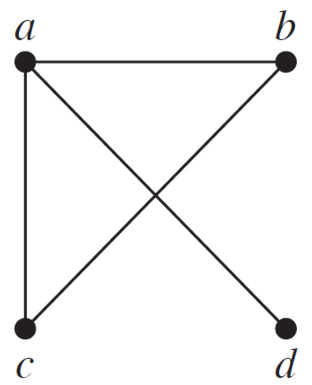
- find the number of colors needed to color the regions of a map.
- determine whether a circuit can be implemented on a planar circuit board.
- distinguish between two chemical compounds with the same molecular formula but different structures using graphs.
- determine whether two computers are connected by a communication link.

The so called adjacency Matrices can be used to represent graphs. Put

$$a_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \text{ is an edge of } G \\ 0 & \text{otherwise.} \end{cases}$$

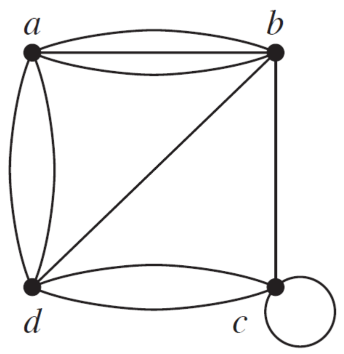
Then the matrix $A = \{a_{ij}\}$ represents the graph G.

Example.



$$\begin{aligned} v_{11} &= (a,a) \\ v_{21} &= (b,a) \\ v_{31} &= (c,a) \\ v_{41} &= (d,a) \end{aligned} \qquad \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Example (Graph with multiple edges)



$$v_{11} = (a,a) \quad v_{21} = (b,a) \quad v_{31} = (c,a) \quad v_{41} = (d,a)$$

$$\begin{pmatrix} 0 & 3 & 0 & 2 \\ 3 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 2 & 1 & 2 & 0 \end{pmatrix}$$

The matrix A helps to understand the network: assume that we want to find the number of walks of length n in the network which start a vertex i and end at the vertex j. It is given by A^n_{ij} , where A_n is the n-th power of the matrix A.

An other application is the “page rank”. The network structure of the web allows to assign a “relevance value” telling how important each node is. This is the bread and butter for a multibillion dollar enterprise.

1.7 Games

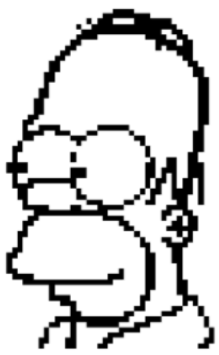
To move around in a computer game requires rotations and translations to be implemented efficiently. Hardware acceleration can help to handle this. We live in a time where graphics processor power grows at a tremendous speed. Virtual reality again tries to get a foothold in the consumer market. Rotations are represented by special matrices. For example, if an object centered at (0, 0, 0) is turned around the y-axes by an angle ϕ, every point in the object gets transformed by the matrix

$$\begin{pmatrix} \cos\phi & 0 & -\sin\phi \\ 0 & 1 & 0 \\ \sin\phi & 0 & \cos\phi \end{pmatrix}$$

1.8 Hopfield Network.

One input image should first be stored and then be retrieved. The input image is: Since an associative memory

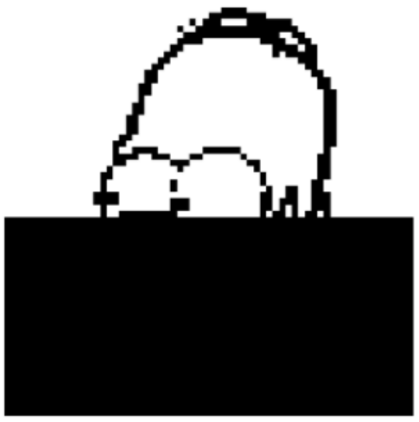
has polar states and patterns (or binary states and patterns), we convert the input image to a black and white image:



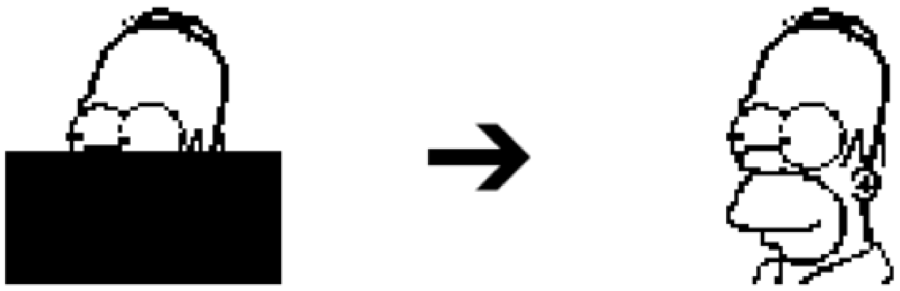
The weight matrix W is the outer product of this black and white image x_{Homer}:

$$W = x_{Homer} x_{Homer}^T, \quad x_{Homer} \in \{-1, 1\}^d,$$

where for this example $d = 64 \times 64$. Can the original image be restored if half of the pixels are masked out? The masked image is:



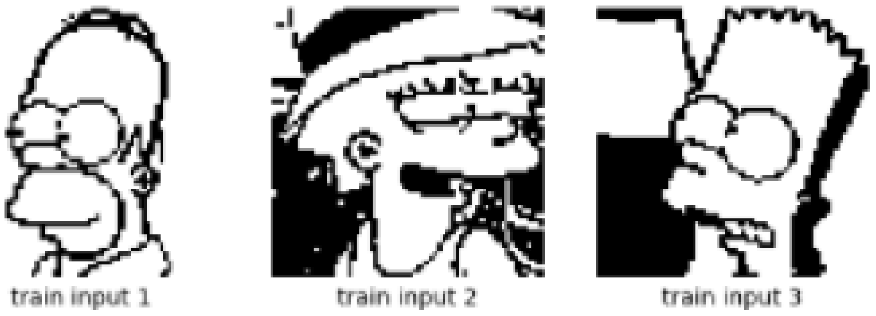
which is our initial state. This initial state is updated via multiplication with the weight matrix W. It takes one update until the original image is restored.



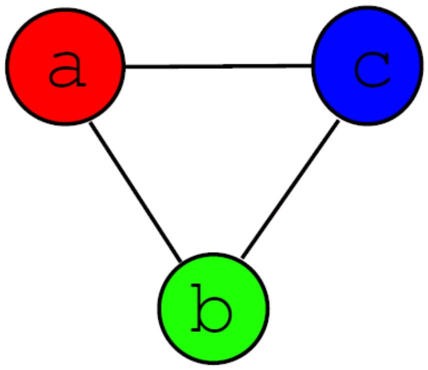
What happens if we store more than one pattern? The weight matrix is then built from the sum of outer products of three stored patterns (three input images):

$$W = \sum_{1}^{3} x_i x_i^T, \quad x_i \in \{-1, 1\}^d.$$

The following figure shows the three stored patterns:



1.9 Symbolic dynamics



Assume that a system can be in three different states a, b, c and that transitions

$$a \mapsto b,\ \ b \mapsto a,\ \ b \mapsto c,\ \ c \mapsto c,\ \ c \mapsto a$$

are allowed. A possible evolution of the system is then

$$a, b, a, b, a, c, c, c, a, b, c, a, ...$$

One calls this a description of the system with symbolic dynamics. This language is used in information theory or in dynamical systems theory. The dynamics of the system is coded with a symbolic dynamical system. The transition matrix is

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix} .$$

Information theoretical quantities like the “entropy” can be read off from this matrix.

1.10 Big data statistics

When analyzing data statistically, one is often interested in the correlation matrix $A_{ij} = E[Y_i Y_j]$ of a random vector $X = (X_1, ..., X_n)$ with $Y_i = X_i E[X_i]$. This matrix can be derived from data. It sometimes even determines the random variables, if the type of the distribution is fixed.

For example, if the random variables have a Gaussian (=Bell shaped) distribution, the correlation matrix together with their expectations $E[X_i]$ determines the random variables. To show this, one requires linear algebra. The magic of linear algebra is one can reduce a complex system of random variables to pairwise independent quantities. The method allows so to see what is important.

1.11 Markov processes

Suppose we have three bags containing 100 balls each. Every time, when a 5 shows up, we move a ball from bag 1 to bag 2, if the dice shows 1 or 2, we move a ball from bag 2 to bag 3, if 3 or 4 turns up, we move a ball from bag 3 to bag 1 and a ball from bag 3 to bag 2. After some time, how many balls do we expect to have in each bag? The problem defines a Markov chain described by a matrix

$$\begin{pmatrix} 5/6 & 1/6 & 0 \\ 0 & 2/3 & 1/3 \\ 1/6 & 1/6 & 2/3 \end{pmatrix}$$

From this matrix, the equilibrium distribution can be read off as an eigenvector of a matrix. Eigen-vectors will play an important role throughout the course.