Unit 1: Introduction

Prof.Dr.P.M.Bajracharya

School of Mathematical Sciences
T.U., Kirtipur

February 8, 2025

Summary

1. **Linear algebra and data science**
   - Storing an image
   - Dealing with Text
   - Representation of problems in Linear Algebra
   - Representing data as flat tables versus matrices and graphs
   - Graphs and connections with matrices
   - Games
   - Hopfield Network
   - Symbolic dynamics
   - Big data statistics
   - Markov processes

- *Linear Algebra for Data*,
  Michael W. Mahoney, University of California Berkeley, 2018
- *Basics of Linear Algebra for Machine Learning*,
  Jason Brownlee

Linear algebra
and
data science

Linear algebra

It is a branch of mathematics concerned with vectors, matrices and linear transformations.

Linear algebra

It is a branch 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.

Although linear algebra is integral to the field of Data Science, the tight relationship is often left unexplained.

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.

Although linear algebra is integral to the field of Data Science, the tight relationship is often left unexplained.

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.

Being familiar with linear algebra is an essential skill for data scientists.

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

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

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

Machine Learning

ML is about discovering structures and patterns from data and make predictions or decisions without being explicitly programmed.

Machine Learning

ML is about discovering structures and patterns from data and make predictions or decisions without being explicitly programmed.

This is done using the language of mathematics.

Types of Machine Learning

1. **Supervised Learning** – The algorithm learns from labeled data (i.e., input-output pairs).

Types of Machine Learning

1. **Supervised Learning** – The algorithm learns from labeled data (i.e., input-output pairs).
   Examples:
   - Spam email classification
   - Stock price prediction

Types of Machine Learning

1. **Supervised Learning** – The algorithm learns from labeled data (i.e., input-output pairs).
   Examples:
   - Spam email classification
   - Stock price prediction
2. **Unsupervised Learning** – The algorithm identifies patterns in unlabeled data.

Types of Machine Learning

1. **Supervised Learning** – The algorithm learns from labeled data (i.e., input-output pairs).
   Examples:
   - Spam email classification
   - Stock price prediction
2. **Unsupervised Learning** – The algorithm identifies patterns in unlabeled data.
   Examples:
   - Customer segmentation
   - Anomaly detection

Types of Machine Learning

1. **Supervised Learning** – The algorithm learns from labeled data (i.e., input-output pairs).
   Examples:
   - Spam email classification
   - Stock price prediction
2. **Unsupervised Learning** – The algorithm identifies patterns in unlabeled data.
   Examples:
   - Customer segmentation
   - Anomaly detection
3. **Reinforcement Learning** – The algorithm learns through trial and error, receiving rewards or penalties.

Types of Machine Learning

1. **Supervised Learning** – The algorithm learns from labeled data (i.e., input-output pairs).
   Examples:
   - Spam email classification
   - Stock price prediction
2. **Unsupervised Learning** – The algorithm identifies patterns in unlabeled data.
   Examples:
   - Customer segmentation
   - Anomaly detection
3. **Reinforcement Learning** – The algorithm learns through trial and error, receiving rewards or penalties.
   Examples:
   - Self-driving cars
   - Game-playing AI (e.g., AlphaGo)

We see that in all types of ML, what is the first important thing is **data representation**.

We see that in all types of ML, what is the first important thing is **data representation**.

Data representation

Consider

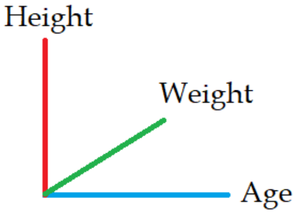
Height	Age	Weight
170	24	70
165	45	65
190	28	102
180	34	83
182	30	79
178	67	85

Our data can be represented using vectors. The first row in this data is represented by a vector called a **feature vector** which has 3 elements or components representing 3 different dimensions

$$\begin{pmatrix} 170 \\ 24 \\ 70 \end{pmatrix}$$

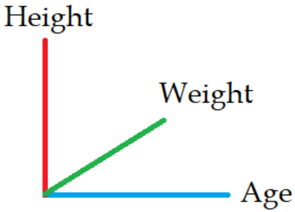
Our data can be represented using vectors. The first row in this data is represented by a vector called a **feature vector** which has 3 elements or components representing 3 different dimensions

$$\begin{pmatrix} 170 \\ 24 \\ 70 \end{pmatrix} \longrightarrow$$



Our data can be represented using vectors. The first row in this data is represented by a vector called a **feature vector** which has 3 elements or components representing 3 different dimensions

$$\begin{pmatrix} 170 \\ 24 \\ 70 \end{pmatrix} \longrightarrow$$



n-entries in a vector makes it n-dimensional vector space and in this case, we can see 3-dimensions. Thus, we can enter our data as a 3-dimensional vector space.

- The main idea behind ML is giving systems the power to automatically learn and improve from experience without being explicitly programmed to do so.

- The main idea behind ML is giving systems the power to automatically learn and improve from experience without being explicitly programmed to do so.
- ML functions through building programs that have access to data (constant or updated) to analyze, find patterns and learn from.

- The main idea behind ML is giving systems the power to automatically learn and improve from experience without being explicitly programmed to do so.
- ML 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.

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.

- **Deep learning techniques** are popularly used in unstructured data such as text data or image data.

- **Deep learning techniques** are popularly used in unstructured data such as text data or image data.
- Before working on any type of data, one should have a good understanding of it.

- **Deep learning techniques** are popularly used in unstructured data such as text data or image data.
- Before working on any type of data, one should have a good understanding of it.
- So, we will now discuss images and see how they get stored on a computer.

- **Deep learning techniques** are popularly used in unstructured data such as text data or image data.
- Before working on any type of data, one should have a good understanding of it.
- So, we will now discuss images and see how they get stored on a computer.
- We will learn about pixel values and cover two popular formats of images – Grayscale and RGB.

What do you see when you look at the image above?

What do you see when you look at the image above?
You most likely said flower, leaves – not too difficult.

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).

You were able to identify the flower because the human brain has gone through million years of evolution.

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.

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.

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?

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.

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.

How Are B & W or Grayscale Images Stored in a Computer?

How Are B & W or Grayscale Images Stored in a Computer?

Let's take an example. Here we have taken a black-and-white image, also known as a **Grayscale image**.

This is the image of the number 8. In storing images, an imortant term is **pixel**

What is pixel

Now, we zoom in further.

If you look closely, you can see that the images are getting distorted, and you will see some small square boxes on this image.

These small boxes are called Pixels. We often use the term- the **dimension of the image** is

$$X \times Y.$$

What does that actually mean?

These small boxes are called Pixels. We often use the term- the **dimension of the image** is

$$X \times Y.$$

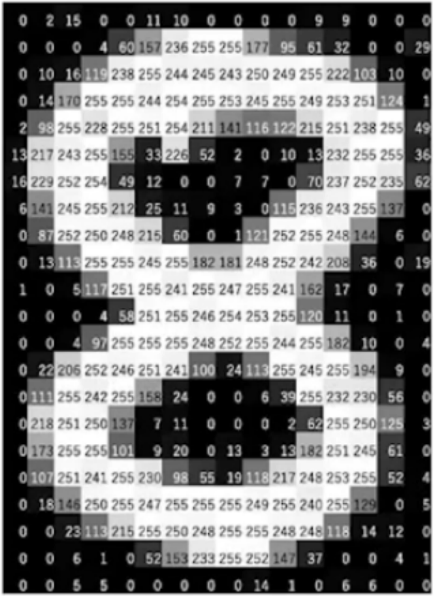
What does that actually mean?
This means that the dimension of the image is simply the number of pixels across the image's height(x) and width(y).

In this case, if you count, it would be 24 pixels across the height and 16 pixels across the width.

In this case, if you count, it would be 24 pixels across the height and 16 pixels across the width. Hence the dimension of this image will be

$$24 \times 16.$$

Although we see an image in this format, the computer store image in the form of numbers.


0 2 15 0 0 11 10 0 0 0 0 9 9 0 0 0
0 0 0 4 60 157 236 255 255 177 95 61 32 0 0 29
0 10 16 119 238 255 244 245 243 250 249 255 222 103 10 0
0 14 170 255 255 244 254 255 253 245 255 249 253 251 124 1
2 98 255 228 255 251 254 211 141 116 122 215 251 238 255 49
13 217 243 255 155 33 226 52 2 0 10 13 232 255 255 36
16 229 252 254 49 12 0 0 7 7 0 70 237 252 235 62
6 141 245 255 212 25 11 9 3 0 115 236 243 255 137 0
0 87 252 250 248 215 60 0 1 121 252 255 248 144 6 0
0 13 113 255 255 245 255 182 181 248 252 242 208 36 0 19
1 0 5 117 251 255 241 255 247 255 241 162 17 0 7 0
0 0 0 4 58 251 255 246 254 253 255 120 11 0 1 0
0 0 4 97 255 255 255 248 252 255 244 255 182 10 0 4
0 22 206 252 246 251 241 100 24 113 255 245 255 194 9 0
0 111 255 242 255 158 24 0 0 6 39 255 232 230 56 0
0 218 251 250 137 7 11 0 0 0 2 62 255 250 125 3
0 173 255 255 101 9 20 0 13 3 13 182 251 245 61 0
0 107 251 241 255 230 98 55 19 118 217 248 253 255 52 4
0 18 146 250 255 247 255 255 255 249 255 240 255 129 0 5
0 0 23 113 215 255 250 248 255 255 248 248 118 14 12 0
0 0 6 1 0 52 153 233 255 252 147 37 0 0 4 1
0 0 5 5 0 0 0 0 0 14 1 0 6 6 0 0

Each of these pixels is denoted as a numerical value, and these numbers are called **Pixel Values**. These pixel values denote the intensity of the pixels.

Each of these pixels is denoted as a numerical value, and these numbers are called **Pixel Values**. These pixel values denote the intensity of the pixels. For a grayscale or b&w image, we have pixel values ranging from 0 to 255.

The smaller numbers closer to zero represent the darker shade while the larger numbers closer to 255 represent the lighter or the white shade.

The smaller numbers closer to zero represent the darker shade while the larger numbers closer to 255 represent the lighter or the white shade.

So every image in a computer is saved in this form where you have a matrix of numbers, and this matrix is also known as a **Channel**.

Now can you guess the shape of this matrix?

Now can you guess the shape of this matrix? Well, it will be the same as the number of pixel values across the height and width of the image.

Now can you guess the shape of this matrix? Well, it will be the same as the number of pixel values across the height and width of the image. In this case, the shape of the matrix would be 24×16

0	2	15	0	0	11	10	0	0	0	0	9	9	0	0	0
0	0	0	4	60	157	236	255	255	177	95	61	32	0	0	29
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	49
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0
0	87	252	250	248	215	60	0	1	121	252	255	248	144	6	0
0	13	113	255	255	245	255	182	181	248	252	242	208	36	0	19
1	0	5	117	251	255	241	255	247	255	241	162	17	0	7	0
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4
0	22	206	252	246	251	241	100	24	113	255	245	255	194	9	0
0	111	255	242	255	158	24	0	0	6	39	255	232	230	56	0
0	218	251	250	137	7	11	0	0	0	2	62	255	250	125	3
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4
0	18	146	250	255	247	255	255	255	249	255	240	255	129	0	5
0	0	23	113	215	255	250	248	255	255	248	248	118	14	12	0
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1
0	0	5	5	0	0	0	0	0	14	1	0	6	6	0	0

Now let's quickly summarize the points that we've learned so far.

- Images are stored in a computer as a matrix of numbers known as pixel values.

Now let's quickly summarize the points that we've learned so far.

- Images are stored in a computer as a matrix of numbers known as pixel values.
- These pixel values represent the intensity of each pixel.

Now let's quickly summarize the points that we've learned so far.

- Images are stored in a computer as a matrix of numbers known as pixel values.
- These pixel values represent the intensity of each pixel.
- In grayscale images, a pixel value of 0 represents black, and 255 represents white.

Now let's quickly summarize the points that we've learned so far.

- Images are stored in a computer as a matrix of numbers known as pixel values.
- These pixel values represent the intensity of each pixel.
- In grayscale images, a pixel value of 0 represents black, and 255 represents white.
- A channel is a matrix of pixel values, and we have only one channel in the case of a grayscale image.

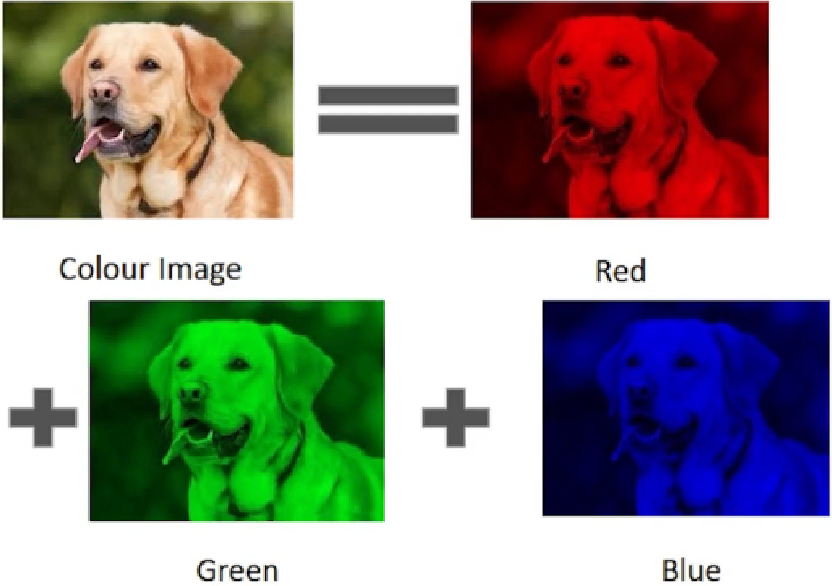
How Are Colored Images Stored on a Computer?

Now that we know how grayscale images are stored in a computer, let's look at an example of a colored image. So let's take an example of a colored image. This is an image of a dog-

This image comprises many different colors.

This image comprises many different colors. Almost all colors can be generated from the three primary colors – Red, Green, and Blue.

This image comprises many different colors. Almost all colors can be generated from the three primary colors – Red, Green, and Blue. Therefore, we can say that each colored image is a unique composition of these three colors or 3 channels – Red, Green, and Blue.


Colour Image
Red
Green
Blue

This means that in a colored image, the number of matrices or the number of channels will be more.

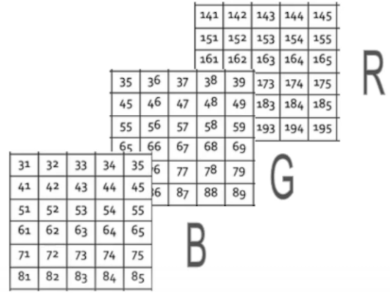
This means that in a colored image, the number of matrices or the number of channels will be more. In this particular example, we have 3 matrices – 1 matrix for red, known as the Red channel,

141	142	143	144	145
151	152	153	154	155
161	162	163	164	165
171	172	173	174	175
181	182	183	184	185
191	192	193	194	195

R

another metric for green, known as the Green channel,

and finally, a third matrix for the blue color, known as the Blue channel.



Each of these matrices would again have values ranging from 0 to 255, where each of these numbers represents the intensity of the pixels. Or in other words, these values represent different shades of red, green, and blue.

All of these channels or matrices superimpose over one another to form the shape of the image when loaded into a computer.

All of these channels or matrices superimpose over one another to form the shape of the image when loaded into a computer. The computer reads this image as

$$N \times M \times 3,$$

where N is the number of pixels across the height, M is the number of pixels across the width, and 3 represents the number of channels.

In this case, we have 3 channels R, G, and B.

In this case, we have 3 channels R, G, and B. So, in our example, the shape of the colored image would be

$$6 \times 5 \times 3$$

since we have 6 pixels across the height, 5 across the width, and there are 3 channels present.

Dealing with Texts

What is a Term-Document Matrix?

- Another active area of research in Machine Learning is **dealing with text**.

What is a Term-Document Matrix?

- Another active area of research in Machine Learning is **dealing with text**.
- One of the most common techniques employed is **Term Document Matrix**.

What is a Term-Document Matrix?

- Another active area of research in Machine Learning is **dealing with text**.
- One of the most common techniques employed is **Term Document Matrix**.
- This technique store counts of words in documents and store this frequency count in a Matrix form.

Let's understand it with an example.

Consider the following sentences.

Index	Sentences
1	I love football
2	Messi is a great football player
3	Messi has won seven Ballon d'Or awards

Here, we can see a set of text responses. The term-document matrix of these responses will look like this:

Here, we can see a set of text responses. The term-document matrix of these responses will look like this:

	I	love	football	Messi	is	a	great	player
Sen. 1	1	1	1	0	0	0	0	0
Sen. 2	0	0	1	1	1	1	1	1
Sen. 3	0	0	0	1	0	0	0	0

This table is a representation of the term-document matrix.

- From this matrix, we can get the total number of occurrences of any word in the whole corpus and by analyzing them we can reach many fruitful results.

- From this matrix, we can get the total number of occurrences of any word in the whole corpus and by analyzing them we can reach many fruitful results.
- Term document matrices are one of the most common approaches which need to be followed during natural language processing and analyzing the text data.

- From this matrix, we can get the total number of occurrences of any word in the whole corpus and by analyzing them we can reach many fruitful results.
- Term document matrices are one of the most common approaches which need to be followed during natural language processing and analyzing the text data.
- More formally we can say that it is the way to represent the relationship between words and sentences presented in the corpus.

Application of Term-Document Matrix

- We can say that making a term-document matrix from the text data is one of the tasks which comes in between the whole NLP project.

Application of Term-Document Matrix

- We can say that making a term-document matrix from the text data is one of the tasks which comes in between the whole NLP project.
- Term document matrix can be used in various types of NLP tasks, some of the tasks we can perform using the term-document matrix are as follows.

- By performing the *singular value decomposition* on the term-document matrix, search results can be improved to an extent. Using it on the search engine, we can improve the results of the searches by disambiguating polysemous words and searching for synonyms of the query.

- By performing the *singular value decomposition* on the term-document matrix, search results can be improved to an extent. Using it on the search engine, we can improve the results of the searches by disambiguating polysemous words and searching for synonyms of the query.
- Most of the NLP processes are focused on *mining one or more behavioural data* from the corpus of text. Term document matrices are very helpful in extracting the data. By performing multivariate analysis on the document term matrix we can reach the different themes of the data.

Using Term Document Matrix we can also perform tasks like Semantic analysis, Language translation, Language generation etc.

Lastly a lot of ML concepts are tied to linear algebra concepts.

Lastly a lot of ML concepts are tied to linear algebra concepts. Some basic examples, PCA - eigenvalue, regression - matrix multiplication ...

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.

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).

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.

Representation of problems in Linear Algebra

Consider a simple problem.

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.

Solution. Suppose the price of a bat is Rs x and the price of a ball is Rs y.

Solution. 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.

Solution. 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

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

Solution. 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.

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.

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.

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.

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

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.

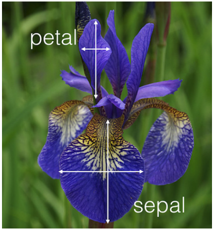
The idea of a flat table is a very general and common way to think about data. It is a table, in that it has several rows or records and one or more columns describing properties of the thing described by that row/record; and it is called **flat** since there is usually no structural relationships between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

In machine learning, we fit a model on a dataset.

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

In machine learning, we fit a model on a dataset. This is the table like a 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.

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.

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.

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).

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 matrices and so forth.

Here is another example of a flat table.

R.No.	Name	Major	HW1	HW2	HW3	Test
102	Alice	Math	95	90	70	100
143	Bob	Comp	Sci	80	65	80
205	Bob	NA	50	60	60	70
216	Charlotte	Stats	85	70	55	80
...						
245	Zaccheus	NA	100	70	55	70

Note that the elements of this table could be represented by two indices.

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}.

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.

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.

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.

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?

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.

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.

- **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.

- **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 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,

For simple operations. When you are first exploring a new data set, even if it is rather small,

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,

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,

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.,

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.

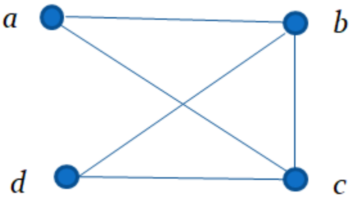
Graphs and connections with matrices

Definitions

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,

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.

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.

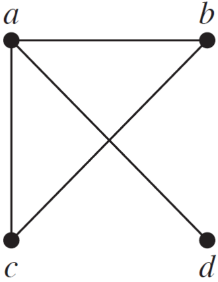
- distinguish between two chemical compounds with the same molecular formula but different structures using graphs.
- determine whether two computers are connected by a communication link.

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.

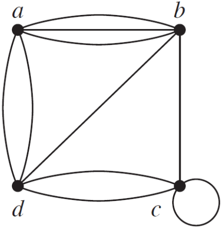
Example.



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

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

Example (Graph with multiple edges)



$$v_{11} = (a, a) \qquad \begin{pmatrix} 0 & 3 & 0 & 2 \\ 3 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 2 & 1 & 2 & 0 \end{pmatrix}$$
$$v_{21} = (b, a)$$
$$v_{31} = (c, a)$$
$$v_{41} = (d, a)$$

- 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.

- 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.
- Another 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.

Games

- To move around in a computer game requires rotations and translations to be implemented efficiently.

- To move around in a computer game requires rotations and translations to be implemented efficiently.
- Hardware acceleration can help to handle this.

- 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.

- 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.

- 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

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

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}:

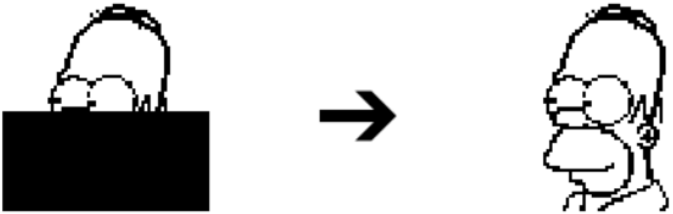
$$W = x_{Homer} x_{Homer}^T, \quad x_{Homer} \in \{-1, 1\}^d,$$

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):

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

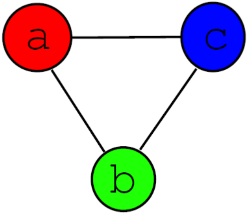
The following figure shows the three stored patterns:

train input 1

train input 2

train input 3

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.

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.

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

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

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.

Big data statistics

When analyzing data statistically, one is often interested in the correlation matrix

$$A_{ij} - E[Y_iY_j]$$

of a random vector $X = (X_1, ..., X_n)$ with

$$Y_i = X_i - E[X_i].$$

This matrix can be derived from data.

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.

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.

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.

Conclusion

- Matrices are a cornerstone of data science, facilitating the representation, transformation, and analysis of data.
- Understanding matrices and their operations is vital for proficiently applying machine learning algorithms, conducting statistical analyses, and extracting insights from complex datasets.
- By harnessing the power of matrices, data scientists can unlock the potential hidden within data and drive informed decision-making across various domains.