Machine Learning Diploma

DSPrep1: Linear Algebra

AMIT

Agenda

- ➔ Introduction
- ➔ Linear equation
- ➔ Vectors
- ➔ Matrices
- ➔ Linear algebra applications in machine learning.

AMIT

1. Introduction to Linear Algebra

AMIT

Introduction:

- Linear algebra is the mathematics of data. Modern statistics is described using the notation of linear algebra and modern statistical methods harness the tools of linear algebra.

- Linear algebra is a large field of study that has tendrils into engineering, physics and quantum physics. Only a specific subset of linear algebra is required, though you can always go deeper once you have the basics.

Linear Equation:

- A linear equation is just a series of terms and mathematical operations where some terms are unknown; for example: $y = 4 \times x + 1$

- A mathematical statement that has an equal to the "=" symbol is called an equation. Linear equations are equations of degree 1, meaning the highest power is 1.
  X square is not linear.

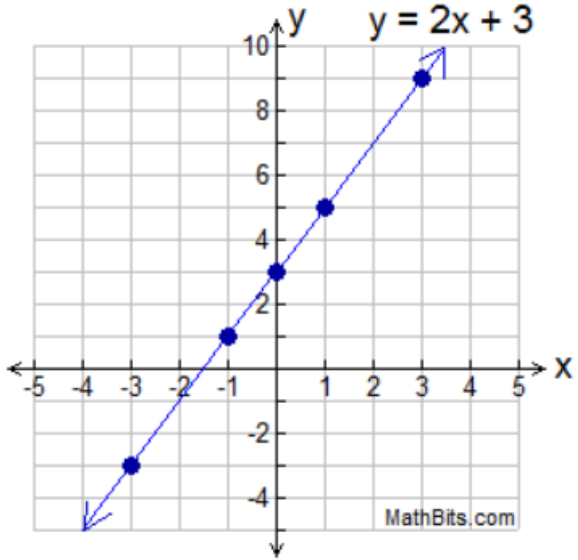
  This equation describes a line on a two-dimensional graph. The line comes from plugging in different values into the unknown x to find out what the equation or model does to the value of y.

AMIT

Linear Equation:

➔ Example:

x	$y = 2x + 3$	y
-3	2(-3) + 3	-3
-1	2(-1) + 3	1
0	2(0) + 3	3
1	2(1) + 3	5
3	2(3) + 3	9



Quiz:

➔ Which is linear and which is nonlinear?

Equations
$y = 8x - 9$
$y = x^2 - 7$
$\sqrt{y} + x = 6$
$y + 3x - 1 = 0$
$y^2 - x = 9$

AMIT

Quiz(Solution):

➔ Which is linear and which is nonlinear?

Equations	Linear or Non-Linear
$y = 8x - 9$	Linear
$y = x^2 - 7$	Non-Linear
$\sqrt{y} + x = 6$	Non-Linear
$y + 3x - 1 = 0$	Linear
$y^2 - x = 9$	Non-Linear

AMIT

2. Vectors

AMIT

Vectors:

- ➔ Vectors are objects that can be added together to form new vectors and that can be multiplied by scalars (i.e., numbers), also to form new vectors.
- ➔ If you have the heights, weights, and ages of a large number of people, you can treat your data as three-dimensional vectors [height, weight, age]
- ➔ The simplest from-scratch approach is to represent vectors as lists of numbers. A list of three numbers corresponds to a vector in three dimensional space, and vice versa.

```
height_weight_age = [70,  # inches,
                     170, # pounds,
                     40 ] # years
```


AMIT

Vectors addition/subtraction:

- ➔ We'll also want to perform arithmetic on vectors. Because Python lists aren't vectors (and hence provide no facilities for vector arithmetic)
- ➔ To begin with, we'll frequently need to add two vectors. Vectors add componentwise. This means that if two vectors v and w are the same length, their sum is just the vector whose first element is v[0] + w[0], whose second element is v[1] + w[1], and so on. (If they're not the same length, then we're not allowed to add them.)

- ➔ For example, [1, 2] + [2, 1] = [1 + 2, 2 + 1] = [3, 3]

AMIT

Quiz:

➔ Write a function that adds two vectors together

AMIT

Quiz (Solution):

➔ Write a function that adds two vectors together

```
def add(v, w):
    """Adds corresponding elements"""
    result =[]
    for i in range(len(v)):
        result.append(v[i]+w[i])
    return result

print(add([1, 2, 3], [4, 5, 6]))
```


AMIT

Quiz (Solution 2):

➔ Write a function that adds two vectors together

Notes:

zip() is a built-in function that pairs together items from the passed sequences

assert() is a built-in function that raises error if the condition is false

```
def add(v, w):
    """Adds corresponding elements"""
    assert len(v) == len(w), "vectors must be the same length"
    return [v_i + w_i for v_i, w_i in zip(v, w)]

assert add([1, 2, 3], [4, 5, 6]) == [5, 7, 9]
```


AMIT

Quiz (Solution 2):

➔ Write a function that adds two vectors together

Notes:

We here checked first if the lengths are equal because we can't add two vectors with different sizes. (Data validation step)

Instead of printing the result of adding those two vectors we used assert to test our function, to test it adds correctly.

```
def add(v, w):
    """Adds corresponding elements"""
    assert len(v) == len(w), "vectors must be the same length"
    return [v_i + w_i for v_i, w_i in zip(v, w)]

assert add([1, 2, 3], [4, 5, 6]) == [5, 7, 9]
```


AMIT

Quiz (Solution 2):

➔ Write a function that adds two vectors together
Notes:
We also have provided a sentence after the assertion condition to explain what happened. For example “vectors must be the same length”

```
def add(v, w):
    """Adds corresponding elements"""
    assert len(v) == len(w), "vectors must be the same length"
    return [v_i + w_i for v_i, w_i in zip(v, w)]

assert add([1, 2, 3], [4, 5, 6]) == [5, 7, 9]
```


AMIT

Vectors Multiplication by a scalar:

- A Scaler is simply any number. Multiplying a vector by a scalar multiplies all the elements by that scaler.

- For example:

  2 * [2,4,6] = [4,8,12]

AMIT

Quiz:

- ➔ Write a function that multiplies a scalar and a vector.
  - Inputs: scaler_num , vector
  - Outputs: vector

AMIT

Quiz (Solution):

➔ Write a function that multiplies a scalar and a vector.

- Inputs: scaler_num , vector
- Outputs: vector

```
def scalar(s, v):
    result = []
    for num in v:
        result.append(num*s)
    return result

print(scalar(2,[2,4,6]))
```


AMIT

Quiz (Solution2):

➔ Write a function that multiplies a scalar and a vector.

Notes:

We used here list Comprehension. It just makes it shorter syntax instead of using for loops on lists. It's just more pythonic but it's the same

Syntax: newlist = [expression for item in iterable if condition

```
def scalar_multiply(c, v):
    """Multiplies every element by c"""
    return [c * v_i for v_i in v]

assert scalar_multiply(2, [1, 2, 3]) == [2, 4, 6]
```


AMIT

Vectors Multiplication Dot product:

- The dot product of two vectors is the sum of their componentwise products
- For example:

  [1,2,3] * [3,4,5] = 1*3 + 2*4 + 3*5 = 27

AMIT

Quiz:

➔ Write a function that multiplies two vectors.

AMIT

Quiz (Solution):

➔ Write a function that multiplies two vectors.

```python
def dot(v, w):
    """Computes v_1 * w_1 + ... + v_n * w_n"""
    assert len(v) == len(w), "vectors must be same length"
    return sum(v_i * w_i for v_i, w_i in zip(v, w))

assert dot([1, 2, 3], [4, 5, 6]) == 32 # 1 * 4 + 2 * 5 + 3 * 6
```


AMIT

3. Matrices

AMIT

Matrices:

- ➔ A matrix is a two-dimensional collection of numbers. We will represent matrices as lists of lists, with each inner list having the same size and representing a row of the matrix.
- ➔ If A is a matrix, then A[i][j] is the element in the ith row and the jth column.

```
A = [[1, 2, 3],  # A has 2 rows and 3 columns
     [4, 5, 6]]

B = [[1, 2],     # B has 3 rows and 2 columns
     [3, 4],
     [5, 6]]
```

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

AMIT

Matrices:

➔ Given this list-of-lists representation, the matrix A has len(A) rows and len(A[0]) columns. which we consider its shape.

➔ A has a shape of (2, 3)
➔ B has a shape of (3,2)

```
A = [[1, 2, 3],  # A has 2 rows and 3 columns
     [4, 5, 6]]

B = [[1, 2],     # B has 3 rows and 2 columns
     [3, 4],
     [5, 6]]
```


AMIT

Quiz:

➔ Write a function that takes a matrix and gives you its shape.

AMIT

Quiz (Solution):

➔ Write a function that takes a matrix and gives you its shape.

```python
def shape(A):
    """Returns (# of rows of A, # of columns of A)"""
    num_rows = len(A)
    num_cols = len(A[0]) if A else 0 # number of elements in first row
    return num_rows, num_cols
assert shape([[1, 2, 3], [4, 5, 6]]) == (2, 3) # 2 rows, 3 columns
```


AMIT

Matrices:

➔ If a matrix has n rows and k columns, we will refer to it as an n × k matrix. We can (and sometimes will) think of each row of an n × k matrix as a vector of length k, and each column as a vector of length n:.

➔ A is 2*3 matrix
➔ B is 3*2 matrix

```
A = [[1, 2, 3],  # A has 2 rows and 3 columns
     [4, 5, 6]]

B = [[1, 2],     # B has 3 rows and 2 columns
     [3, 4],
     [5, 6]]
```


AMIT

Matrices:

➔ We can use a matrix to represent a dataset consisting of multiple vectors, simply by considering each vector as a row of the matrix. For example, if you had the heights, weights, and ages of 1,000 people, you could put them in a 1,000 × 3 matrix

```
data = [[70, 170, 40],
        [65, 120, 26],
        [77, 250, 19],
        # ....
       ]
```


AMIT

Matrices Addition/Subtraction:

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

$$C = A + B$$

- The scalar elements in the resulting matrix are calculated as the addition of the elements in each of the matrices being added.

$$C = \begin{pmatrix} a_{1,1} + b_{1,1} & a_{1,2} + b_{1,2} \\ a_{2,1} + b_{2,1} & a_{2,2} + b_{2,2} \\ a_{3,1} + b_{3,1} & a_{3,2} + b_{3,2} \end{pmatrix}$$

AMIT

Quiz:

➔ Write a function that adds two matrices, using this notation:

$$
\begin{aligned}
C[0,0] &= A[0,0] + B[0,0] \\
C[1,0] &= A[1,0] + B[1,0] \\
C[2,0] &= A[2,0] + B[2,0] \\
C[0,1] &= A[0,1] + B[0,1] \\
C[1,1] &= A[1,1] + B[1,1] \\
C[2,1] &= A[2,1] + B[2,1]
\end{aligned}
$$

AMIT

Quiz (Solution):

➔ Write a function that adds two matrices

```
A = [[1,2,3],[4,5,6]]
B = [[1,2,3],[4,5,6]]
def matrix_add(A,B):
    c = []
    for i in range(len(A)):
        c.append(add(A[i],B[i]))
    return c

print(matrix_add(A,B))
```


AMIT

Matrices Multiplication:

- Two matrices with the same size can be multiplied together, and this is often called element-wise matrix multiplication or the Hadamard product.

$$C = A \circ B$$

- As with element-wise subtraction and addition, element-wise multiplication involves the multiplication of elements from each parent matrix to calculate the values in the new matrix.

$$C = \begin{pmatrix} a_{1,1} \times b_{1,1} & a_{1,2} \times b_{1,2} \\ a_{2,1} \times b_{2,1} & a_{2,2} \times b_{2,2} \\ a_{3,1} \times b_{3,1} & a_{3,2} \times b_{3,2} \end{pmatrix}$$

AMIT

Quiz:

➔ Write a function that multiply two matrices element-wise, using this notation:

$$
\begin{aligned}
C[0,0] &= A[0,0] \times B[0,0] \\
C[1,0] &= A[1,0] \times B[1,0] \\
C[2,0] &= A[2,0] \times B[2,0] \\
C[0,1] &= A[0,1] \times B[0,1] \\
C[1,1] &= A[1,1] \times B[1,1] \\
C[2,1] &= A[2,1] \times B[2,1]
\end{aligned}
$$

AMIT

Quiz (Solution):

➔ Write a function that multiply two matrices element-wise:

```
def mult_element_wise(v, w):
    """Adds corresponding elements"""
    assert len(v) == len(w), "vectors must be the same length"
    return [v_i * w_i for v_i, w_i in zip(v, w)]

def matrix_mult_element_wise(A,B):
    c = []
    for i in range(len(A)):
        c.append(mult_element_wise(A[i],B[i]))
    return c

print(matrix_mult_element_wise(A,B))
```


AMIT

Matrices Multiplication 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.

$$C = A \cdot B$$

- The number of columns (n) in the first matrix (A) must equal the number of rows (m) in the second matrix (B).
- This rule applies for a chain of matrix multiplications where the number of columns in one matrix in the chain must match the number of rows in the following matrix in the chain.

$$C(m,k) = A(m,n) \cdot B(n,k)$$

AMIT

Matrices Multiplication Dot product:

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

$$B = \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix}$$

$$C = \begin{pmatrix} a_{1,1} \times b_{1,1} + a_{1,2} \times b_{2,1}, a_{1,1} \times b_{1,2} + a_{1,2} \times b_{2,2} \\ a_{2,1} \times b_{1,1} + a_{2,2} \times b_{2,1}, a_{2,1} \times b_{1,2} + a_{2,2} \times b_{2,2} \\ a_{3,1} \times b_{1,1} + a_{3,2} \times b_{2,1}, a_{3,1} \times b_{1,2} + a_{3,2} \times b_{2,2} \end{pmatrix}$$

AMIT

Matrix-Vector Multiplication:

➔ A matrix and a vector can be multiplied together as long as the rule of matrix multiplication is observed. Specifically, that the number of columns in the matrix must equal the number of items in the vector.

$$c = A \cdot v$$

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

$$v = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$$

$$c = \begin{pmatrix} a_{1,1} \times v_1 + a_{1,2} \times v_2 \\ a_{2,1} \times v_1 + a_{2,2} \times v_2 \\ a_{3,1} \times v_1 + a_{3,2} \times v_2 \end{pmatrix}$$

AMIT

Quiz:

➔ Write a function that multiply a vector with a matrix. Using this notation:

$$c[0] = A[0,0] \times v[0] + A[0,1] \times v[1]$$
$$c[1] = A[1,0] \times v[0] + A[1,1] \times v[1]$$
$$c[2] = A[2,0] \times v[0] + A[2,1] \times v[1]$$

AMIT

Quiz:

➔ Write a function that multiply a vector with a matrix.

```
def dot(v, w):
    """Adds corresponding elements"""
    assert len(v) == len(w), "vectors must be the same length"
    return sum([v_i * w_i for v_i, w_i in zip(v, w)])

def matrix_mult_vector(A,b):
    c = []
    for i in range(len(A)):
        c.append(dot(A[i],b))
    return c

A = [[1, 2], [3, 4],[5, 6]]
b = [0.5,0.5]
print(matrix_mult_vector(A,b))
```


AMIT

Matrix-Scalar Multiplication:

- A matrix can be multiplied by a scalar. This can be represented using the dot notation between the matrix and the scalar.

$$C = A \cdot b$$

- The result is a matrix with the same size as the parent matrix where each element of the matrix is multiplied by the scalar value.

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \\ a_{3,1} & a_{3,2} \end{pmatrix} \qquad C = \begin{pmatrix} a_{1,1} \times b + a_{1,2} \times b \\ a_{2,1} \times b + a_{2,2} \times b \\ a_{3,1} \times b + a_{3,2} \times b \end{pmatrix}$$

AMIT

Quiz:

➔ Write a function that multiplies a matrix by a scalar, using this notation:

$$
\begin{aligned}
C[0,0] &= A[0,0] \times b \\
C[1,0] &= A[1,0] \times b \\
C[2,0] &= A[2,0] \times b \\
C[0,1] &= A[0,1] \times b \\
C[1,1] &= A[1,1] \times b \\
C[2,1] &= A[2,1] \times b
\end{aligned}
$$

AMIT

Quiz (Solution):

➔ Write a function that multiplies a matrix by a scalar:

```
def scalar_multiply(c,v):
    return [c * v_i for v_i in v]

def matrix_mult_scalar(A,b):
    c = []
    for i in range(len(A)):
        c.append(scalar_multiply(b,A[i]))
    return c
A = [[1, 2], [3, 4], [5, 6]]
b = 0.5
print(matrix_mult_scalar(A,b))
```


AMIT

4. Types of Matrices

Types of Matrices:

- ➔ Square Matrix
- ➔ Symmetric Matrix
- ➔ Triangular Matrix
- ➔ Diagonal Matrix
- ➔ Identity Matrix

AMIT

Square Matrix:

- ➔ A square matrix is a matrix where the number of rows (n) is equivalent to the number of columns (m).
- ➔ The square matrix is contrasted with the rectangular matrix where the number of rows and columns are not equal.
- ➔ Square matrices are readily added and multiplied together and are the basis of many simple linear transformations, such as rotations (as in the rotations of images)

$$M = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}$$

AMIT

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. Below is an example of a 5 × 5 symmetric matrix.

$$M = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & 2 & 3 & 4 \\ 3 & 2 & 1 & 2 & 3 \\ 4 & 3 & 2 & 1 & 2 \\ 5 & 4 & 3 & 2 & 1 \end{pmatrix}$$

AMIT

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.
- Below is an example of a 3 × 3 upper triangular matrix.

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

- Below is an example of a 3 × 3 lower triangular matrix.

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

AMIT

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.
- ➔ Diagonal matrices consist mostly of zeros and have non-zero entries only along the main diagonal.
- ➔ A diagonal matrix does not have to be square. In the case of a rectangular matrix, the diagonal would cover the dimension with the smallest length; for example:

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix} \qquad D = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

AMIT

Identity Matrix

- An identity matrix is a square matrix that does not change a vector when multiplied. The values of an identity matrix are known. All of the scalar values along the main diagonal (top-left to bottom-right) have the value one, while all other values are zero.
- An identity matrix is a matrix that does not change any vector when we multiply that vector by that matrix.

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

AMIT

5. Linear Algebra Examples

AMIT

Linear Algebra in ML Examples:

1. Dataset and Data Files
2. Images and Photographs
3. One Hot Encoding
4. Linear Regression
5. Regularization
6. Principal Component Analysis
7. Singular-Value Decomposition
8. Latent Semantic Analysis
9. Recommender Systems

AMIT

1- Dataset and Data Files

This is the iris flower dataset. the table consists of a set of numbers where:

each row represents an observation

each column represents a feature of the observation.

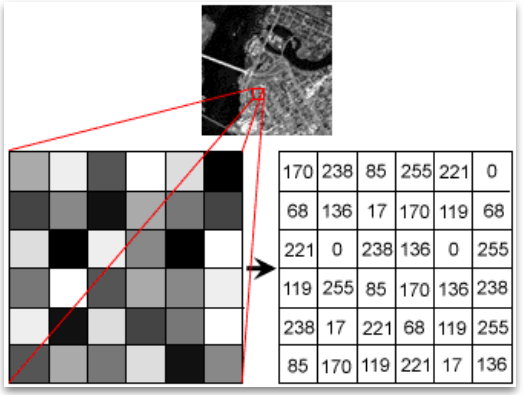
```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
```

- This data is in fact a matrix (Data structure in linear algebra).
- when you split the data into inputs and outputs to fit a supervised machine learning model, such as the measurements and the flower species, you have a matrix (X) and a vector (y).
- The vector is another key data structure in linear algebra

AMIT

2- Images and Photographs

- Perhaps you are more used to working with images or photographs in computer vision applications.
- Each image that you work with is itself a table structure with a width and height and one pixel value in each cell for black and white images or 3 pixel values in each cell for a color image. A photo is yet another example of a matrix from linear algebra.



- Operations on the image, such as cropping, scaling, shearing and so on are all described using the notation and operations of linear algebra

AMIT

2- Images and Photographs

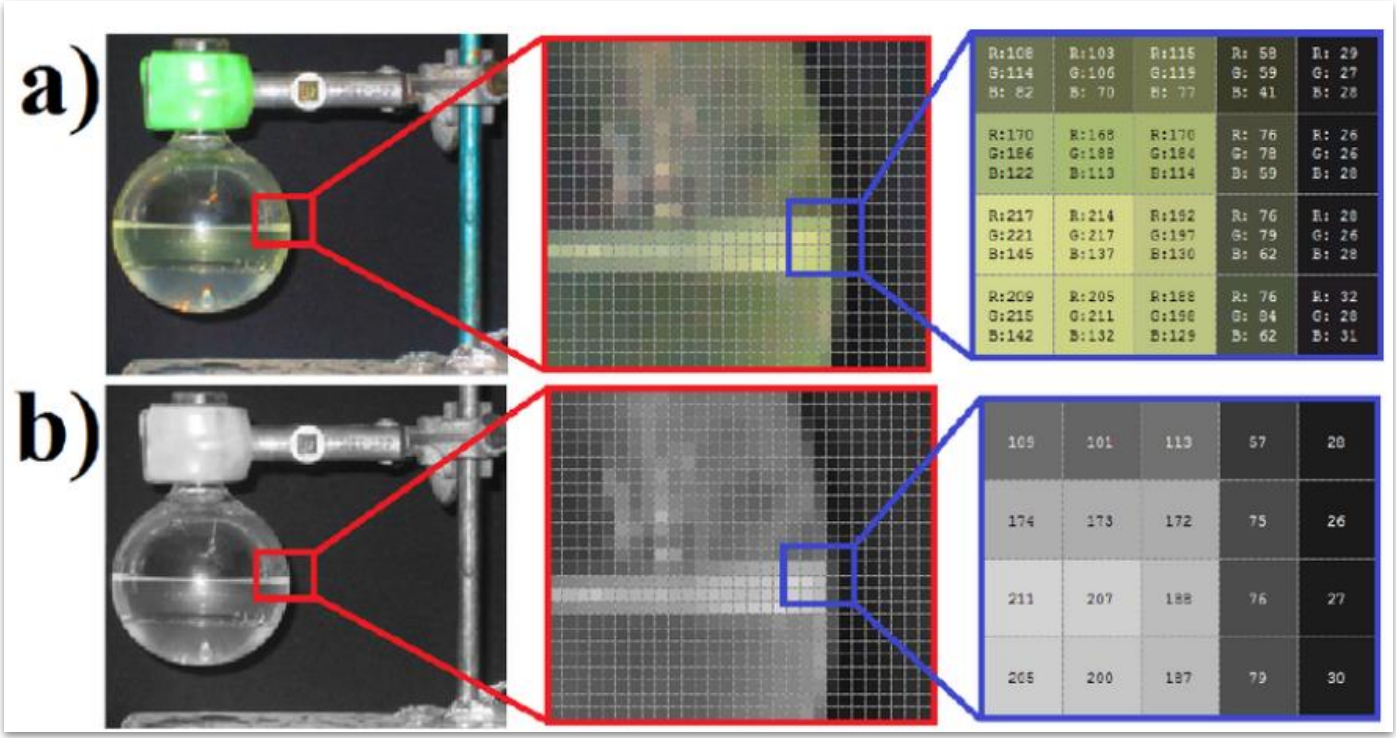
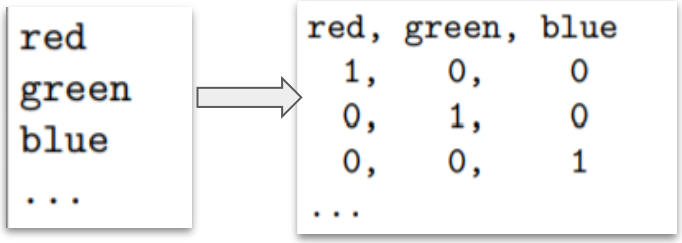


Image reference [link]

3- One Hot Encoding

Perhaps the class labels for classification problems, or perhaps categorical (data type: string) input variables. It is common to encode categorical variables to make their easier to work with and learn by some techniques.

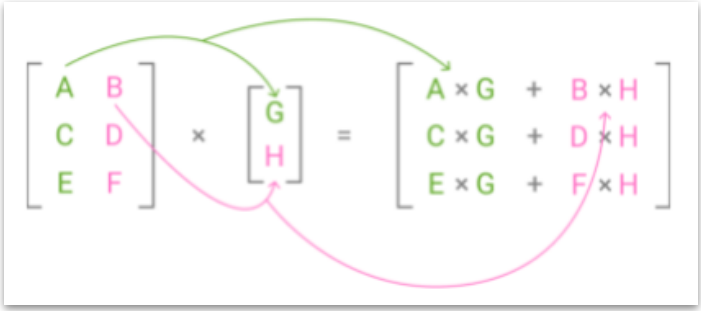


- A one hot encoding is where a table is created to represent the variable with one column for each category and a row for each example in the dataset.
- This is an example of a sparse representation, a whole sub-field of linear algebra.

AMIT

4- Linear Regression

Linear regression is an old method from statistics for describing the relationships between variables. It is often used in machine learning for predicting numerical values.



- The common way of summarizing the linear regression equation uses linear algebra notation Where y is the output variable A is the dataset and b are the model coefficients

$$y = A \cdot b$$

AMIT

5- Regularization

In applied machine learning, we often seek the simplest possible models that achieve the best skill on our problem. Simpler models are often better at generalizing from specific to unseen data.

L1 Regularization

$$\text{Cost} = \sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2 + \lambda \sum_{j=0}^{M} |W_j|$$

L2 Regularization

$$\text{Cost} = \sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2 + \lambda \sum_{j=0}^{M} W_j^2$$

Loss function

Regularization Term

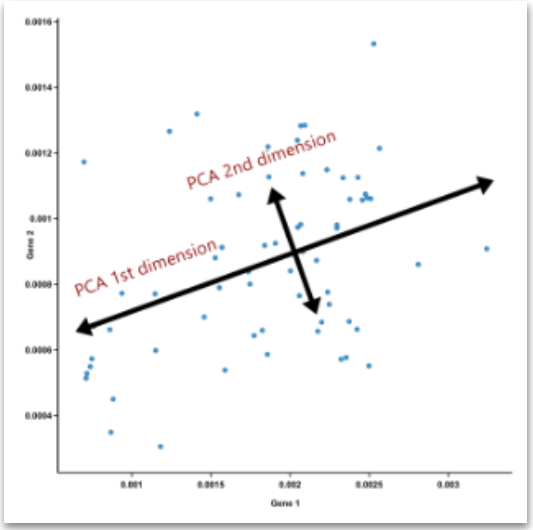
- Simpler models are often characterized by models that have smaller coefficient values.
- Common implementations include the L 2 and L 1 forms of regularization. Both of these forms of regularization are in fact a measure of the magnitude or length of the coefficients as a vector and are methods lifted directly from linear algebra called the vector norm.

AMIT

6- Principal Component Analysis

Often a dataset has many columns, perhaps thousands or more. Modeling data with many features is challenging, and models built from data that include irrelevant features are often less skillful than models trained from the most relevant data.

Methods for automatically reducing the number of columns of a dataset are called dimensionality reduction.



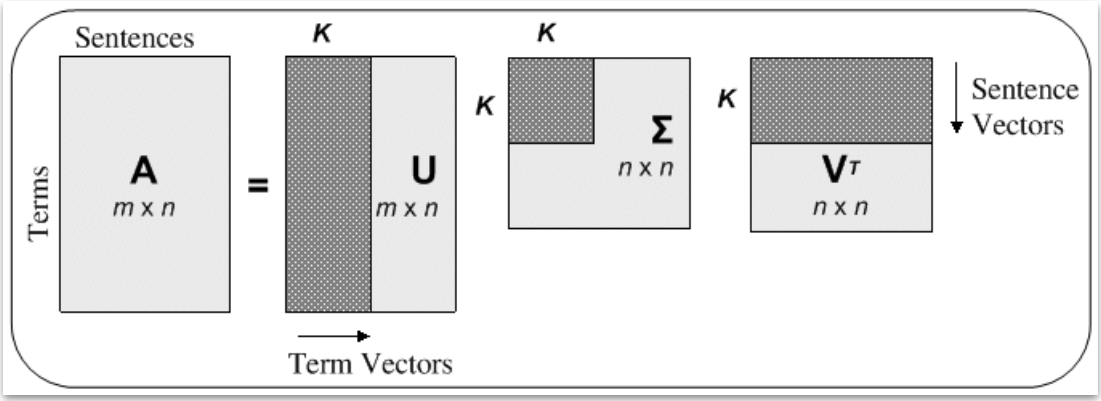
- The core of the PCA method is a matrix factorization method from linear algebra.

7- Singular-Value Decomposition

Another popular dimensionality reduction method is the singular-value decomposition method or SVD for short.

It is a matrix factorization method from the field of linear algebra.

It has wide use in linear algebra and can be used directly in applications such as feature selection, visualization, noise reduction and more.



8- Latent Semantic Analysis

In the sub-field of machine learning for working with text data called natural language processing, it is common to represent documents as large matrices of word occurrences.

For example, the columns of the matrix may be the known words in the vocabulary and rows may be sentences, paragraphs, pages or documents of text with cells in the matrix marked as the count or frequency of the number of times the word occurred.

	Quick	Brown	Fox	Jumps	Over	Lazy	Dog
The quick brown fox jumps over the lazy dog	1	1	1	1	1	1	1
If the fox is quick he can jump over the dog.	1	0	1	0	1	0	1
Foxes are quick. Dogs are lazy.	0	1	1	0	0	1	1
Can a fox jump over a dog?	0	0	1	1	1	0	1

AMIT

8- Latent Semantic Analysis

Matrix factorization methods such as the SVD can be applied to this sparse matrix which has the effect of distilling the representation down to its most relevant essence.

Documents processed in this way are much easier to compare, query and use as the basis for a supervised machine learning model.

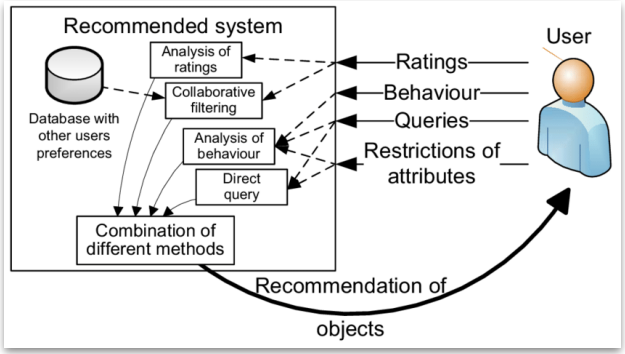
This form of data preparation is called Latent Semantic Analysis or LSA for short, and is also known by the name Latent Semantic Indexing or LSI.

AMIT

9- Recommender Systems

Predictive modeling problems that involve the recommendation of products are called recommender systems, a sub-field of machine learning.

The development of recommender systems is primarily concerned with linear algebra methods. Matrix factorization methods like the singular-value decomposition are used widely in recommender systems to distill item and user data to their essence for querying and searching and comparison.



Any Questions?

AMIT

THANK YOU!