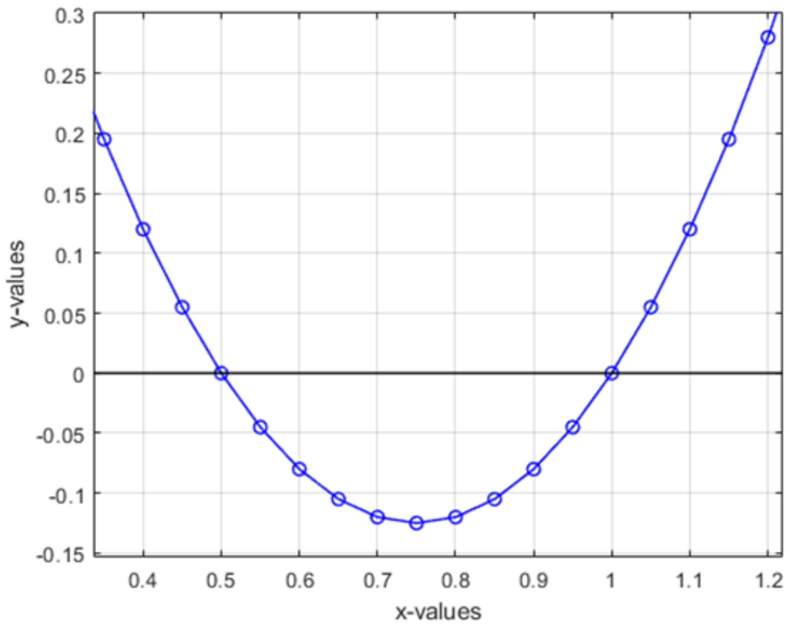
Constraint Optimization

Optimization

- Consider the problem of finding the roots of the polynomial $2x^2 - 3x + 1$
- The roots of a polynomial represent the values of x, for which the f(x) = 0



You can see from the above plot that the roots are 0.5 & 1.0. Such kind of “deterministic” problems **are not considered optimisation problems.**

Optimization

What is Optimization ?

Any problem that deals with maximizing or minimizing any physical quantity (which can be mathematically computed/expressed in the form of some variables) are defined as an Optimization Problem.

Mathematically,

$$\min f(x), \text{ where } f(x) = 2x^2 - 3x + 1$$

is read as **"find the minimum value of f(x)",** is an optimization problem.

Here f(x) is called the **Objective function** (which is referred to as the Loss function or Cost Function in our machine learning), and the variable "x" here is called the "**Decision Variable".**

Optimization

Also the below

$$\underset{x}{\operatorname{argmin}}\ f(x), \text{ where } f(x) = 2x^2 - 3x + 1$$

is read as **“find the argument x for which f(x) has minimum value”,** is also an optimization problem. ALL our machine-learning optimizations are of this type!

Different types Optimization Problems

There are 2 types of Optimization Problems in general: Constrained & Unconstrained.

(A) Unconstrained Optimization

$$\min f(x), \text{ where } f(x) = 2x^2 - 3x + 1$$

$$\underset{x}{\operatorname{argmin}}\ f(x), \text{ where } f(x) = 2x^2 - 3x + 1$$

Where there are neither any "**bounds**" on the values of x (e.g. x has to be a positive number, or it should be between 0 & 10, etc.) nor any type of constraints that need to be satisfied.

Linear Regression is an example of unconstrained optimization, given by:

$$\boxed{\underset{w_j}{\arg\min}\ L(w_j \mid X, Y)} \quad where,\ \hat{Y} = w_0 + w_1 X_1$$

Says, Find the **optimal weights (wj)** for which the **MSE** Loss function (given in eq. 3 above) has **min value,** for a **GIVEN X, Y data** (refer to very first the table at the start of the article). L(wj) represents the MSE Loss, a function of the model weights, not X or Y. Remember, X & Y is your DATA and is supposed to be CONSTANT! The subscript "j" represents the jth model coefficient/weight.

Different types Optimization Problems

There are 2 types of Optimization Problems in general: Constrained & Unconstrained.

(B) **Constrained Optimization:** for example

$$\min\ (x_1-2)^2+(x_2-1)^2 \qquad \text{subject to} \begin{cases} x_1^2 - x_2 & \leq 0, \\ x_1 + x_2 & \leq 2. \end{cases}$$

says “minimize (x1-2)^2 + (x2-1)^2, subject to the constraints” as given above.

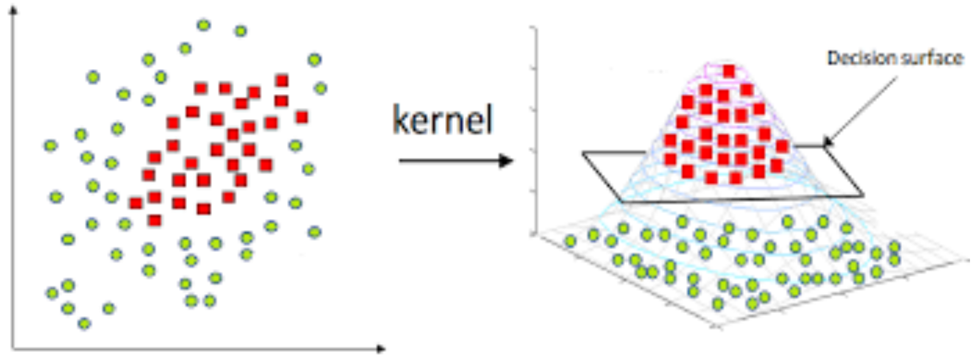
The range of values of the decision variable which satisfy all the constraints constitutes the “**Feasible Region**”, and the final optimal value will lie in these feasible regions. If the constraints are very “hard” and contradicting, there may not be any “feasible solution” to the given optimization problem.

Different types Optimization Problems

An example of constrained optimization is Support Vector Machines (SVM), which is an ML algorithm typically used for classification (although it has a regressor variant too).

By default, the Support Vector Classifier (SVC) tries to find a separating hyperplane with maximum margin, with a “hard constraint” that all the data points must be classified correctly.

However, if the data is not linearly separable, it may not lead to any solution. In such cases, we use the “Kernel Trick” and/or use the soft-margin variant of SVM (controlled by the hyperparameter C in the algorithm), with the loss function



Ways to solve Optimization Problems

Typically there are 2 approaches to solving most of the most complex numerical/scientific problems:

- **Analytical Method**
- **Numerical Methods Approach**

Analytical Method

This is the "pen-and-paper" theoretical method normally taught to us in usual school/university. These methods will give the "exact" solution for any well-posed mathematical problem which is solvable.

For example, if we are asked to find the values of x for which the function f(x) = 2x^2 – 3x + 1 values equal to zero (normally called the "roots" of the polynomial), we know have been taught to solve for the roots of a generic 2-degree polynomial ax^2 + bx + c = 0, given by:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = 0.5 \; and \; 1.0$$

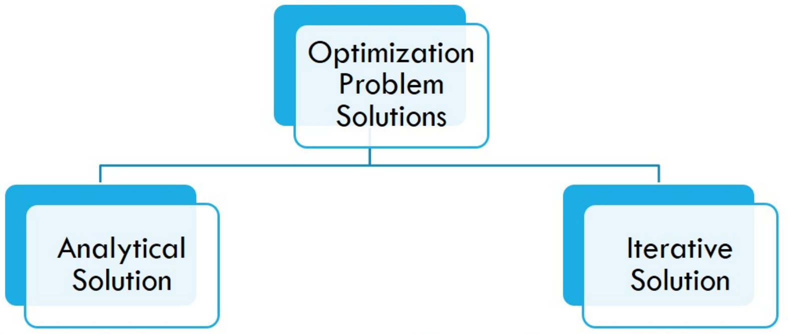
Ways to solve Optimization Problems

Numerical Methods Approach

Normally, in this approach, we start with a guess value as the solution and then perform some calculations (in this case, evaluate the function f(x), and then "improve" the solution with every iteration until we get the desired result (called as **Convergence**).

For example, if the tolerance is set to 0.0001, then the iterations will stop when the value of the f(x) <= 0.0001, and the whole optimization process is said to have Converged to an optimal solution.

Optimization Problem Solution Methods:



- **Theoretical Solution**, which gives the "exact" solution to the problem, provided the optim. problem has a "closed-form solution"

-**Approximate Solution** to the optim. problem, based on some iterative algorithm
- Can solve all types of optim. problems.

Use of Optimization in Machine learning and Deep learning

We have a fair idea about optimization by now. Now let us see where all this is used in ML & DL.

Linear Regression

It uses the OLS method for solving the optimal model coefficients.

Logistic Regression :

The optimal values of the model coefficients are obtained by minimizing the BinaryCrossEntropy Loss function. The Logistic Regression class of scikit-learn uses the “solver” option (default value of ‘lbfgs’) and “max_iter” (default value of 100) as one of the hyperparameters.

The LogisticRegression class in sklearn uses iterative solvers, and the other supported solvers are ‘liblinear’, ‘sag’, ‘saga’ and ‘newton-cg’.

In Deep Learning : we normally use the advanced variants of the MiniBatch Gradient Descent algorithm such RMSprop, Adam, Adadelta, Adagrad, Adamax & Nadam to train our neural networks.

Other Applications of Optimization

- Apart from the Data Science domain, optimisation problems are abundant in the industry. A manufacturing company trying to minimize the cost of manufacturing a product to **maximize its profit.** This total cost will be a function of the manpower cost, raw materials cost, costs of inventory, storage, transportation and other logistics. And furthermost of these variables come along with their own set of constraints. So you can find the **"optimal number of units"** to be manufactured to have **minimum manufacturing cost.**

- Another example of a **Scheduling problem** is finding the optimal timetable (occupancy and/or availability table) of employees in a company for doing specific tasks in different shifts so that the **manpower cost is minimized**, and overtime pay may not be needed.

- Chemical engineers, want to have the **optimal design (shape, size, diameter, length etc.)** of the pressure vessels for chemical reactions, boilers, and pipes for transporting the fluids and even storage.

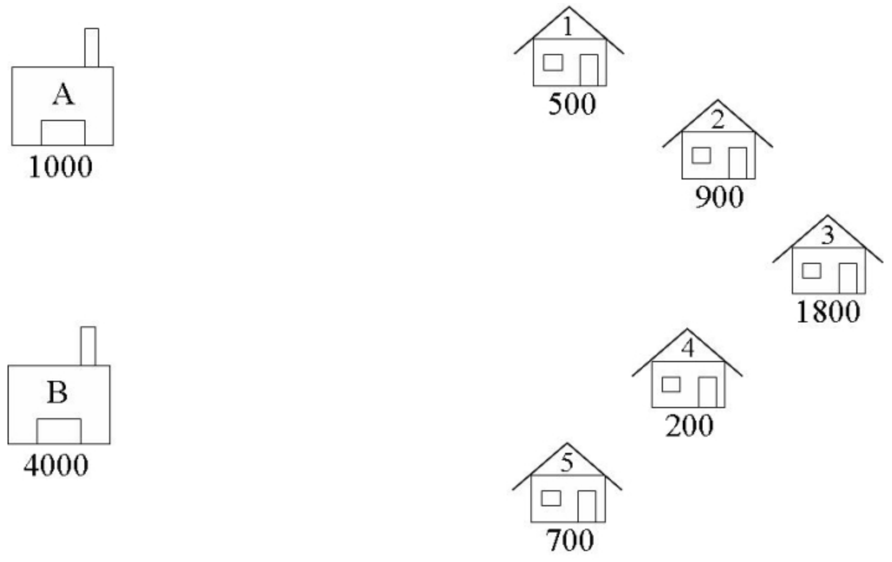
Optimization Examples

Examples of Optimization Problems

A boutique brewery has two warehouses from which it distributes beer to five carefully chosen bars. At the start of every week, each bar sends an order to the brewery's head office for so many crates of beer, which is then dispatched from the appropriate warehouse to the bar.

The brewery would like to have an interactive computer program which they can run week by week to tell them which warehouse should supply which bar so as to minimize the costs of the whole operation.

For example, suppose that at the start of a given week the brewery has 1000 cases at warehouse A, and 4000 cases at warehouse B, and that the bars require 500, 900, 1800, 200, and 700 cases respectively. Which warehouse should supply which bar?

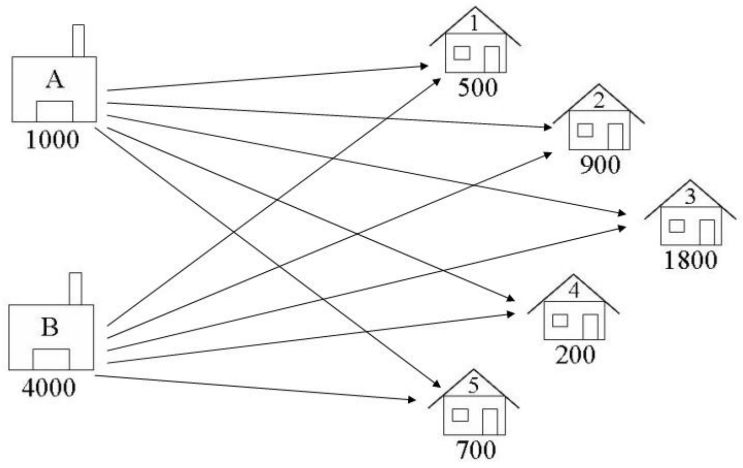


Examples of Optimization Problems

Identify the Decision Variables

In transportation problems we are deciding how to transport goods from their supply nodes to their demand nodes. The decision variables are the Arcs connecting these nodes, as shown in the diagram below. We are deciding how many crates of beer to transport from each warehouse to each pub.

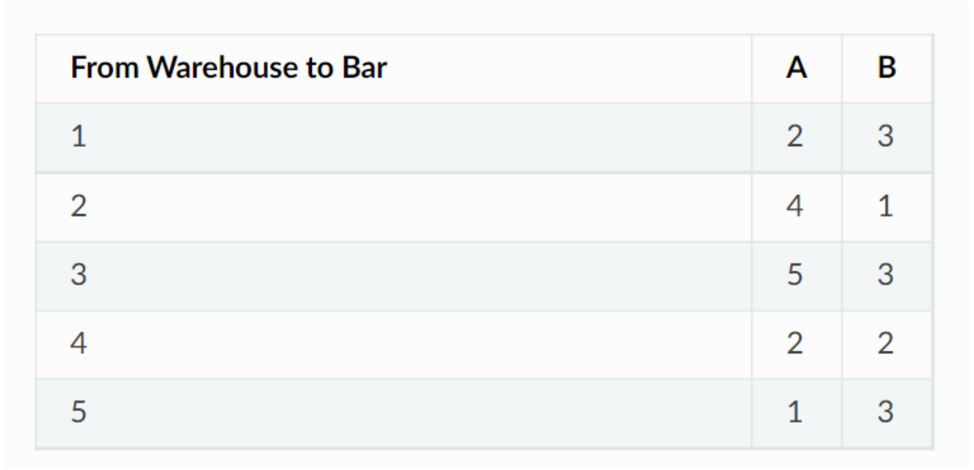
•A1 = number of crates of beer to ship from Warehouse A to Bar 1
•A5 = number of crates of beer to ship from Warehouse A to Bar 5
•B1 = number of crates of beer to ship from Warehouse B to Bar 1
•B5 = number of crates of beer to ship from Warehouse B to Bar 5



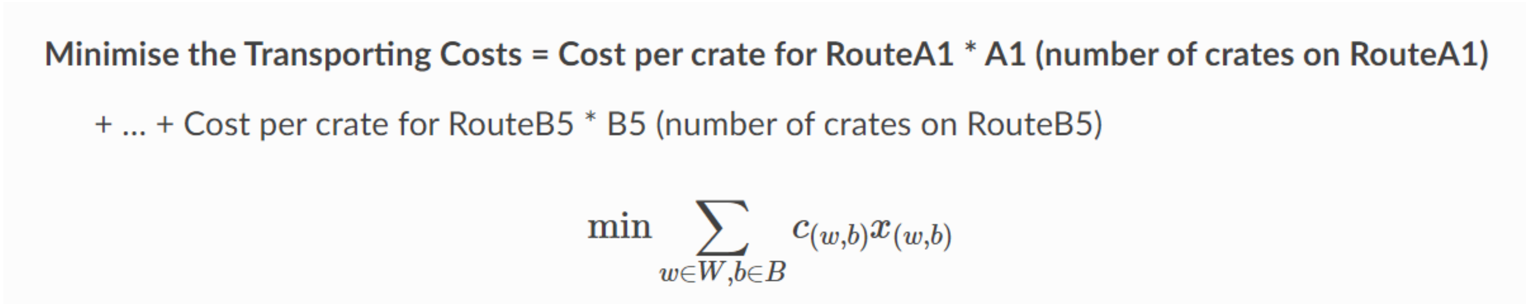
Examples of Optimization Problems

Objective Function

We shall assume then that there is a fixed transportation cost per crate. (If the capacity of a truck is small compared with the number of crates that must be delivered then this is a valid assumption). Lets assume we talk with the financial manager, and she gives us the following transportation costs (dollars per crate):



From Warehouse to Bar	A	B
1	2	3
2	4	1
3	5	3
4	2	2
5	1	3



Minimise the Transporting Costs = Cost per crate for RouteA1 * A1 (number of crates on RouteA1)

+ ... + Cost per crate for RouteB5 * B5 (number of crates on RouteB5)

$$\min \sum_{w \in W, b \in B} c_{(w,b)} x_{(w,b)}$$

Examples of Optimization Problems

Formulate the Constraints

The constraints come from considerations of supply and demand. The supply of beer at warehouse A is 1000 cases. The total amount of beer shipped from warehouse A cannot exceed this amount. Similarly, the amount of beer shipped from warehouse B cannot exceed the supply of beer at warehouse B. The sum of the values on all the arcs leading out of a warehouse, must be less than or equal to the supply value at that warehouse:

Such that:

- A1 + A2 + A3 + A4 + A5 <= 1000
- B1 + B2 + B3 + B4 + B5 <= 4000

The demand for beer at bar 1 is 500 cases, so the amount of beer delivered there must be at least 500 to avoid lost sales. Similarly, considering the amounts delivered to the other bars must be at least equal to the demand at those bars.

- A1 + B1 >= 500
- A2 + B2 >= 900
- A3 + B3 >= 1800
- A4 + B4 >= 200
- A5 + B5 >= 700

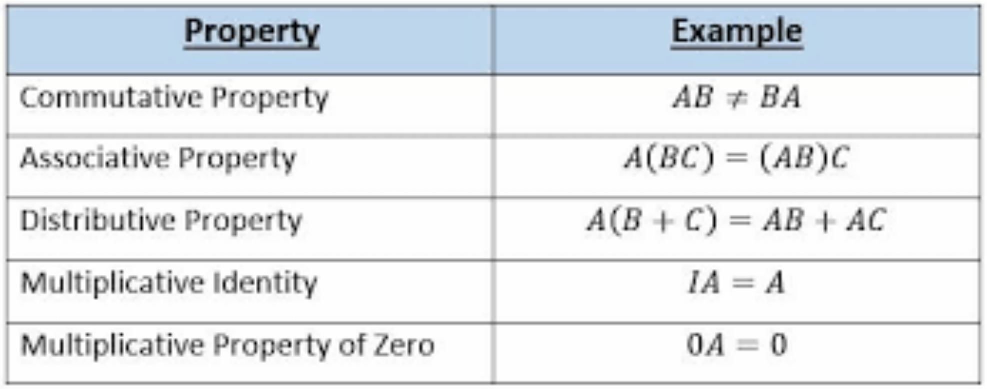
Covariance Matrix ,
Matrix Calculus

Matrix calculus

Linear algebra is essential in Machine Learning (ML) and Deep Learning (DL).

Basic matrix behavior

Distributive, associative & communicative properties.



Property	Example
Commutative Property	$AB \neq BA$
Associative Property	$A(BC) = (AB)C$
Distributive Property	$A(B + C) = AB + AC$
Multiplicative Identity	$IA = A$
Multiplicative Property of Zero	$0A = 0$

As we can see from the above image that matrices are not commutative.

Matrix calculus

Transpose

The transpose of a matrix is found by interchanging its rows into columns or columns into rows.

The transpose of the matrix is denoted by using the letter “T” in the superscript of the given matrix.

For example, if “A” is the given matrix, then the transpose of the matrix is represented by A' or A^T.

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \\ x_n \end{bmatrix}, \qquad \boldsymbol{x}^T = \left[x_1, x_2, \ldots, x_n\right]$$

$$\begin{bmatrix} 7 & -2 & 4 \\ 8 & 3 & 6 \\ 2 & -4 & 7 \end{bmatrix}^T = \begin{bmatrix} 7 & 8 & 2 \\ -2 & 3 & -4 \\ 4 & 6 & 7 \end{bmatrix}$$

Properties

$$(AB)^T = B^T A^T$$
$$(A + B)^T = A^T + B^T$$

Matrix calculus

Symmetric matrix

A square matrix that is equal to the transpose of that matrix is called a symmetric matrix.

i.e. A = A^T

$$A = \begin{bmatrix} 9 & 2 & 3 \\ 2 & -1 & -8 \\ 3 & -8 & 0 \end{bmatrix} \qquad A' = \begin{bmatrix} 9 & 2 & 3 \\ 2 & -1 & -8 \\ 3 & -8 & 0 \end{bmatrix}$$

Skew Symmetric matrix

A square matrix that is equal to the negative of transpose of that matrix is called a skew symmetric matrix.

i.e. A = $-A^T$

$$A = \begin{pmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{pmatrix} \qquad A^T = \begin{pmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{pmatrix}$$

$$-A = \begin{pmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{pmatrix}$$

are equal

Matrix calculus

Inner Product

The inner product ⟨a, b⟩ (or a ▪ b) is a scalar function. The inner product for two vectors is defined as:

$$\boldsymbol{a}^{\mathrm{T}} = [a_1, a_2, \ldots, a_n] \ , \quad \boldsymbol{b}^{\mathrm{T}} = [b_1, b_2, \ldots, b_n]$$

$$\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \boldsymbol{a}^T \boldsymbol{b} = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

Example :

$$r = \begin{bmatrix} 1 \\ -1 \\ 2 \end{bmatrix} \quad s = \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{aligned} \langle r, s \rangle &= \begin{bmatrix} 2 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 2 \end{bmatrix} \\ &= 2 \cdot 1 + 0 \cdot (-1) + 1 \cdot 2 \\ &= 4 \end{aligned}$$

Matrix calculus

Inner Product

The transpose of any scalar value equals itself.

$$s^T = s \qquad \text{transpose of a scalar } s$$

Inner product of two vectors remains the same irrespective of order of vectors in multiplication.

$$x^T y = y^T x \qquad \text{inner product for vectors}$$

Magnitude of a Vector

A vector has both a direction and a magnitude.

The magnitude of a vector formula summarises the individual measures of the vector along the x-axis, y-axis, and z-axis.

The magnitude of a vector gives the numeric value for a given vector.

The magnitude of a vector **A** is denoted by |**A**|

example, if **A** = 1**i** + 2**j** + 3**k**,

then |**A**|

= √($1^2+2^2+2^2$)

= √9 = 3 units.

Matrix calculus

Unit Vector

A unit vector is a vector with a unit length.

Angle between two vectors

The angle between the two vectors *u* and *v* is

$$\cos\theta = \frac{u \cdot v}{||u||\,||v||}$$

Vectors obey the following inequality

$$|u \bullet v| \leq \| u \| \| v \|$$

$$\| u + v \| \leq \| u \| + \| v \|$$

Matrix calculus

Other Properties of inner product

A unit vector is a vector with a unit length.

$$\begin{aligned}\langle ax, y\rangle &= a\langle x, y\rangle \\ \langle x + y, z\rangle &= \langle x, z\rangle + \langle y, z\rangle \\ \langle Ax, y\rangle &= \langle x, A^{\mathrm{T}}y\rangle\end{aligned}$$

Matrix calculus

Outer Product

If u and v are column vectors of any size,then

UV^T is the outer product of u and v.

Note : The inner product $u^T v$ produces a scalar but the **outer product** uv^T produces a matrix.

Example :

If U = $\begin{bmatrix} 1 \\ 3 \end{bmatrix}$ and V = $\begin{bmatrix} 5 \\ 6 \end{bmatrix}$

then outer product of U and V is

$$\begin{pmatrix} 1 \\ 3 \end{pmatrix} \begin{pmatrix} 5 & 6 \end{pmatrix} = \begin{pmatrix} 5 & 6 \\ 15 & 18 \end{pmatrix}$$

Matrix calculus

Outer Product

The multiplication of two matrices is the sum of the outer product of column *i* of *A* with row *i* of *B*.

$$\underbrace{\begin{pmatrix} | & & | \\ \boldsymbol{u}_1 & \cdots & \boldsymbol{u}_m \\ | & & | \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} — \boldsymbol{v}_1 — \\ \vdots \\ — \boldsymbol{v}_n — \end{pmatrix}}_{B} = \begin{pmatrix} | \\ \boldsymbol{u}_1 \\ | \end{pmatrix} \begin{pmatrix} — \boldsymbol{v}_1 — \end{pmatrix} + \ldots + \begin{pmatrix} | \\ \boldsymbol{u}_n \\ | \end{pmatrix} \begin{pmatrix} — \boldsymbol{v}_n — \end{pmatrix}$$

Example

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \end{pmatrix} \begin{pmatrix} 5 & 6 \end{pmatrix} + \begin{pmatrix} 2 \\ 4 \end{pmatrix} \begin{pmatrix} 7 & 8 \end{pmatrix}$$

Matrix calculus

Matrix Derivative

In machine learning, we use matrices to store data.

Objective functions and errors are often expressed with matrices and vectors.

To optimize a solution or to minimize an error, we need to know how to differentiate them.

the notation for the derivative of a **scalar y (f(x))** w.r.t. x as

$$\frac{\partial y}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y}{\partial x_1} \\ \frac{\partial y}{\partial x_2} \\ \vdots \\ \frac{\partial y}{\partial x_n} \end{bmatrix}$$

Example

$y = ax$

$y = a_1 x_1 + a_2 x_2 + \ldots + a_n x_n$

$$\frac{\partial y}{\partial \mathbf{x}} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = a$$

Matrix calculus

Matrix Derivative

the notation we will use for differentiating **a vector y w.r.t. x**

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_2}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_1} \\ \frac{\partial y_1}{\partial x_2} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_m}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_1}{\partial x_n} & \frac{\partial y_2}{\partial x_n} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}$$

Example

$$\underset{\mathbf{y}}{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}} = A \underset{\mathbf{x}}{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}} = \begin{bmatrix} \overset{a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n}{A_1 x} \\ A_2 x \\ \vdots \\ A_m x \end{bmatrix}$$

$$y_1 = a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n$$

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = A^T$$

Matrix calculus

Singular Matrix

A singular (degenerate) matrix is a square matrix whose determinant is zero.

For example:

$$A = \begin{pmatrix} 3 & 12 \\ 2 & 8 \end{pmatrix}$$

The Determinant is

$(3 \times 8) - (12 \times 2)$

$24 - (24) = 0$

Inverse

The **matrix inverse** of matrix *A* is defined as:

$$\underline{A^{-1}} A = I$$

Matrix calculus

Properties of Matrix Inverse

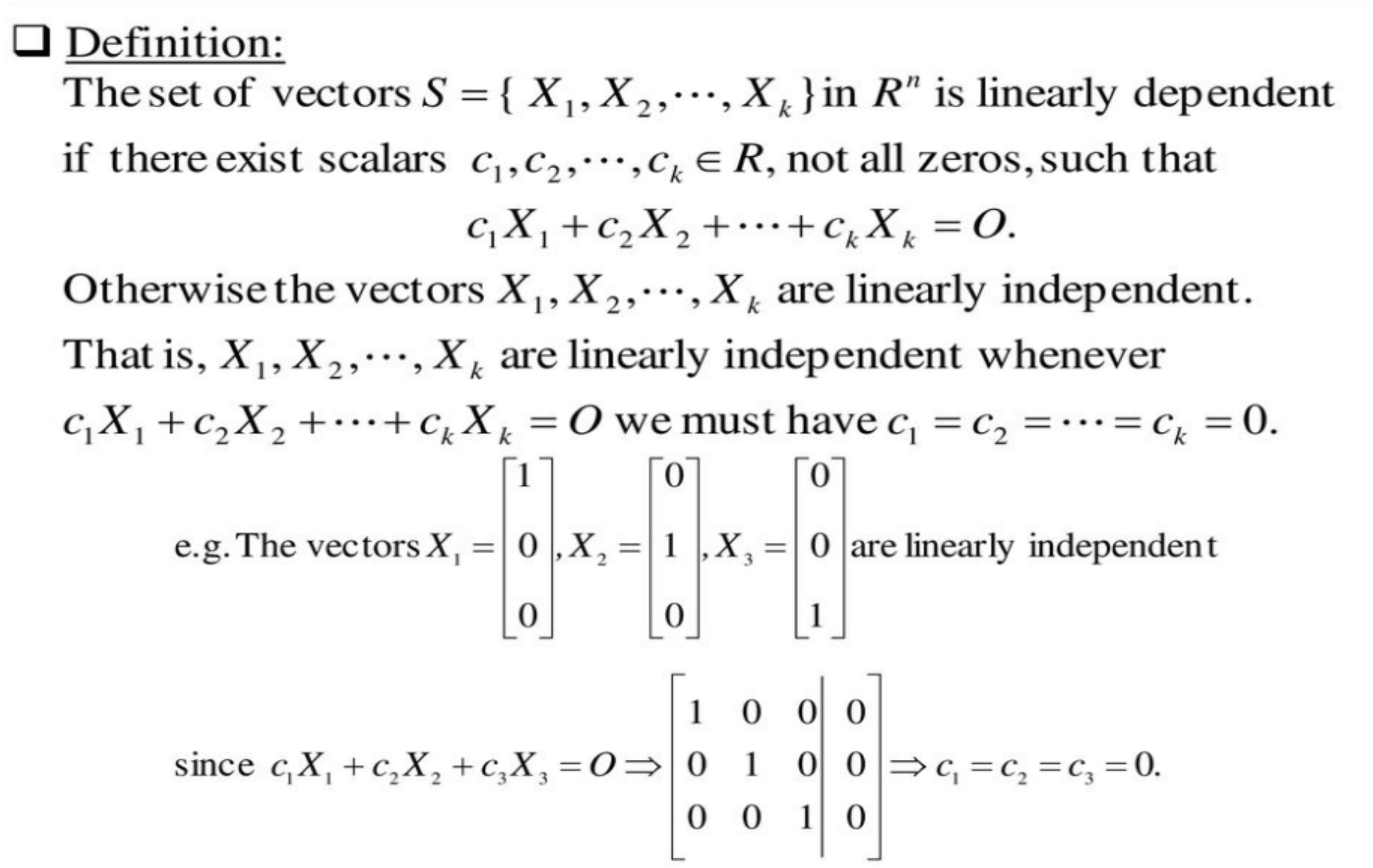
$$(AB)^{-1} = B^{-1}A^{-1}$$
$$(A^T)^{-1} = (A^{-1})^T$$

Note:

1) The property above is true only if *A* and *B* are invertible. (i.e. the inverse of the matrix can be computed.)
2) Inverse of a singular matrix can not be computed.

Matrix calculus

Linear dependency



❑ Definition:

The set of vectors $S = \{X_1, X_2, \cdots, X_k\}$ in R^n is linearly dependent if there exist scalars $c_1, c_2, \cdots, c_k \in R$, not all zeros, such that

$$c_1 X_1 + c_2 X_2 + \cdots + c_k X_k = O.$$

Otherwise the vectors $X_1, X_2, \cdots, X_k$ are linearly independent.

That is, $X_1, X_2, \cdots, X_k$ are linearly independent whenever $c_1 X_1 + c_2 X_2 + \cdots + c_k X_k = O$ we must have $c_1 = c_2 = \cdots = c_k = 0$.

e.g. The vectors $X_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, X_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, X_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ are linearly independent

since $c_1 X_1 + c_2 X_2 + c_3 X_3 = O \Rightarrow \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right] \Rightarrow c_1 = c_2 = c_3 = 0.$

Matrix calculus

Rank

Rank measures the linear independence of the columns/rows of *A*.

The maximum number of its linearly independent columns (or rows) of a matrix is called the rank of a matrix.

$$\underset{\text{rank } 2}{\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}} \quad \underset{\text{rank } 2}{\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}} \quad \underset{\text{rank } 3}{\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 8 \end{pmatrix}}$$

row 3 = 2 × row 2 - row 1

Matrix calculus

Orthogonal matrix

Two vectors or matrices are **orthogonal** if their inner product ⟨*x, y*⟩ equals zero.

$$\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \mathbf{y} = \mathbf{I}$$

Trace

The trace is the sum of the diagonal elements of *A*.

$$Tr(A) = \sum_i A_{ii}$$

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Covariance Matrix

The concepts of covariance and correlation are very useful in many machine learning applications.

Algorithms, like PCA for example, depend heavily on the computation of the covariance matrix, which plays a vital role in obtaining the principal components.

Covariance and the covariance matrix

Assume that we have a dataset with two features and we want to describe the different relations within the data.

Covariance measures the variance between two variables.

Formula to compute covariance is given by

$$C_{x,y} = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})$$

Covariance Matrix

Consider a dataset with three features x, y, and z.

Computing the covariance matrix will yield us a 3 by 3 matrix. This matrix contains the covariance of each feature with all the other features and itself. We can visualize the covariance matrix like this:

$$C(x, y, z) = \begin{bmatrix} var_x & covar_{x,y} & covar_{x,z} \\ covar_{y,x} & var_y & covar_{y,z} \\ covar_{z,x} & covar_{z,y} & var_z \end{bmatrix}$$

The covariance matrix is symmetric. The diagonal contains the variance of a single feature, whereas the non-diagonal entries contain the covariance.

Principal Component Analysis

PCA and Dimensionality Reduction

Why PCA and why Dimensionality Reduction?

Imagine we have a large dataset, which has multiple dimensions or features. Just by looking at it and trying to perform EDA and analyzing the hidden patterns in the data is a tedious job.

It is difficult to try to understand hidden patterns just by looking at the data.

This is where the PCA comes to our rescue. The idea behind PCA is simply to find a low-dimension set of axes that summarize data.

Example

We have a dataset composed of a set of properties from Laptops. These properties describe each Laptop by its size, color, screen size, weight, processor, number of USB slots, O.S., Touch screen or no Touch screen, and so on. However, many of those features will **measure related properties** and thus are going to be redundant. Therefore, we should remove this redundancy and describe each laptop with fewer properties. This is exactly what PCA does.

For example, think about the screen size as a feature of Laptops, almost every example has a screen size almost similar, **hence we can conclude that this feature has a low variance** (On average, most popular laptops have screen sizes that range between 13 to 15 inches.), so this feature will make all laptops look the same, but they are pretty different from each other.

PCA and Dimensionality Reduction

Why PCA and why Dimensionality Reduction?

Do not think that PCA is just dropping the feature of the screen size of Laptops and thus by considering the processor feature and other features which has significant variance in it is reducing the dimensions of the given dataset.

PCA does not drop any feature present in the original dataset and arrives at Principal Components.

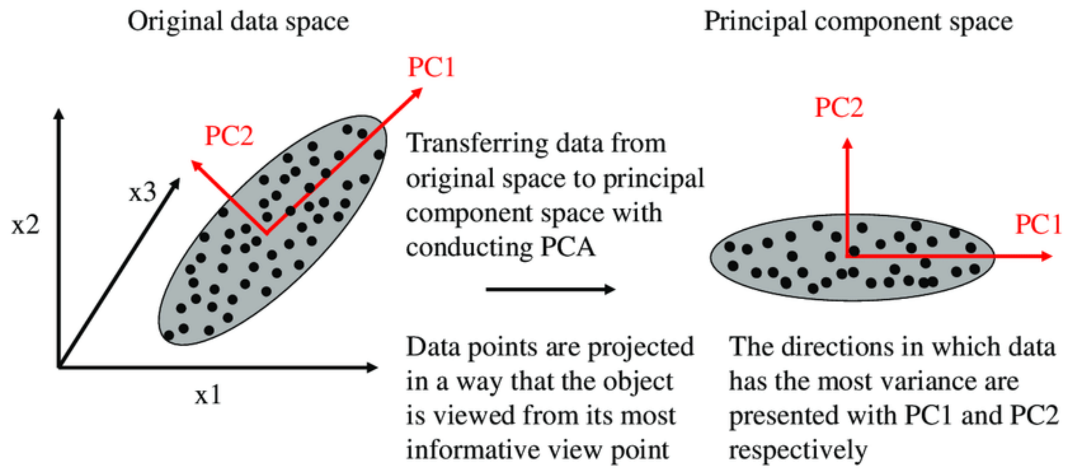
The Algorithm actually constructs a new set of properties which is a combination of the old ones. Mathematically PCA performs a **Linear Transformation moving the original set of features to a new space composed by the Principal Component**. These new features do not have any real meaning except algebraic, therefore do not think that by combining features, linearly you will find new features that you have never thought could exist.

PCA and Dimensionality Reduction

How does the new PCA space look like?

In the new feature space, we are trying to find some properties that strongly differ across the classes. As discussed in the above example, some properties that present low variance are not useful, it will make the laptops look the same. On the other hand, PCA looks for properties that show the maximum amount of variation across classes as possible to create the Principal Component Space. The algorithm uses the concepts of Variance Matrix, Covariance Matrix, Eigenvector, and EigenValues pairs to perform PCA, providing a set of eigenvectors and its respective eigenvalues as a result.

The concepts of Covariance Matrix, Eigenvalues, and Eigenvectors are explained in the section below on Mathematical concepts behind PCA.


Original data space
Principal component space
PC1
PC2
x3
x2
x1
Transferring data from original space to principal component space with conducting PCA
PC2
PC1
Data points are projected in a way that the object is viewed from its most informative view point
The directions in which data has the most variance are presented with PC1 and PC2 respectively

Steps in PCA

The steps followed in PCA are as mentioned below:

1. Standardization of the continuous variables of the dataset.

2. Computing the Co-variance Matrix to identify Co-relations.

3. Computing the Eigen Values and Eigenvectors of the covariance Matrix to identify the Principal Components.

4. Deciding on the Principal Components to be kept for further analysis based on the variation in the Components using the Scree Plot.

Steps in PCA

Step 1: Standardization

- The reason why it is important to perform standardization before PCA is that the latter is extremely sensitive regarding the variances of the variables. i.e. if there are large differences between the ranges of initial variables, those variables with larger ranges will dominate over those with small ranges(For example, a variable that ranges between 0 and 100 will dominate over a variable that ranges between 0 and 1), which can cause biased results.
- So transforming the data to comparable scales will prevent this problem.

Steps in PCA

Step 2: Covariance Matrix Computation

In this step, we will try to identify if there is any relationship between the variables in the dataset. As sometimes, variables are highly correlated in a way such that the information contained in them is redundant. So as to identify these co-relations, we compute the Covariance Matrix.

STEP 3: Computing Eigenvectors and Eigenvalues

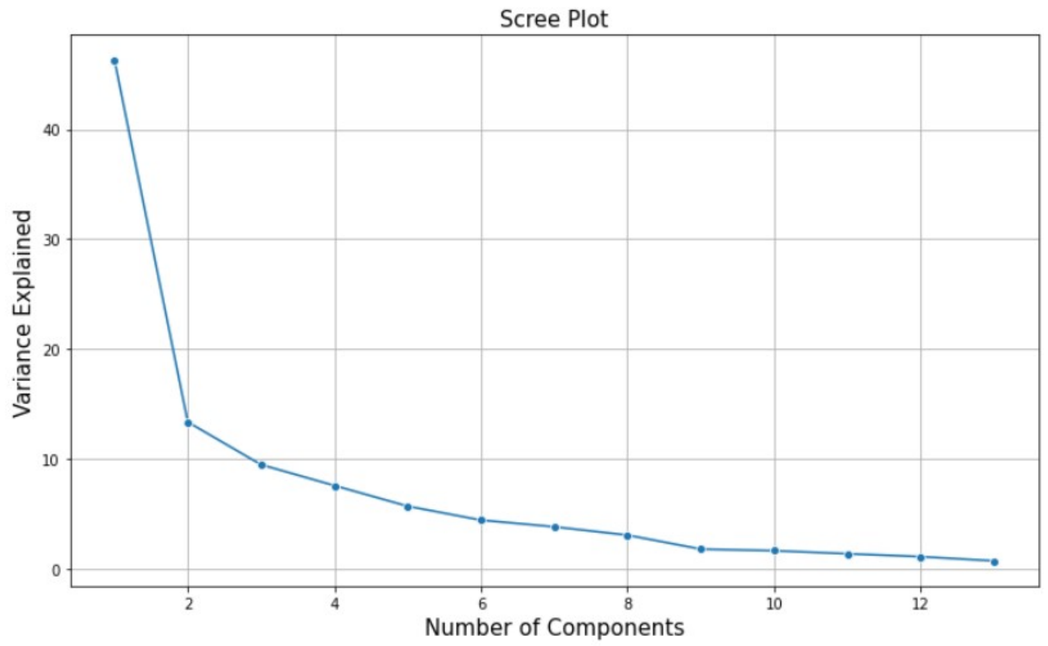
- Eigenvectors and Eigenvalues are the Mathematical Concepts of Linear Algebra that we've to calculate from the Covariance Matrix to find out the Principal Components of the dataset
- Principal Components represent the directions of the data that specify a maximum amount of variance i.e. the lines that capture most information present in the data.
- To put it in a simpler way, just consider Principal Components as new axes that provide the best angle to visualize and evaluate the data, in order that the difference between the observations is best visible.

Steps in PCA

Step 4: Scree Plot to decide on the Number of PCS

A Scree plot always displays the Eigenvalues in a downward curve, ordering the eigenvalues from largest to smallest. According to the Scree test, the ”elbow” of the graph where the Eigenvalues seem to level off is found and factors or components to the left of this point should be retained as significant.

The Scree Plot is used to determine the number of Principal Components to keep in a Principal Component Analysis(PCA).



Advantages and of PCA

Advantages of PCA

1. Removes correlated Features.
2. Improves Algorithm Performance
3. Reduces Overfitting
4. Improves visualization of the data.

Limitations of PCA

1) Model Performance: PCA can lead to a reduction in Model Performance on datasets with no or low feature co-relation or does not meet the assumptions of linearity.

2) **Outliers:** PCA is affected by outliers. Hence treating outliers is important before performing PCA.

3) **Interpretability:** After performing PCA on the dataset original features will turn into Principal Components which is a Linear Combination of the original features. Hence, these principal components are not as readable as the original features.

Singular Value Decomposition

Geometrical Interpretation of Eigen decomposition

The concepts of eigen decomposition is very important in many fields such as computer vision and machine learning using dimension reduction methods of PCA.

The geometrical explanation of the matrix eigen decomposition helps to make the tedious theory easier to understand.

Let’s look at the following equation

$$AX = \lambda X$$

$$A = \begin{bmatrix} 2 & 4 \\ 1 & 5 \end{bmatrix}$$

$$X = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$\lambda = 6$$

•Where A —Square Matrix; X — Eigenvector; λ — Eigenvalue.

•The right hand side plot is a simple example of the left equation.

Geometrical Interpretation of Eigen decomposition

- The two sides are still equal if we multiply any positive scalar on both sides

$$AsX = \lambda sX$$
$$AX' = \lambda X'$$

where X'=sX is another eigenvector

Both X and X' are corresponding to the same eigenvalue λ. Since s can be any non-zero scalar, we see this unique λ can have infinite number of eigenvectors.

Singular Value Decomposition

SVD is based on eigenvalues computation, it generalizes the eigen decomposition of the square matrix A to any matrix M of dimension m×n.

$$M = U\Sigma V' = \sigma_1 u_1 v_1' + \ \ldots \sigma_p u_p v_p'$$

- M is factorized into three matrices, U, Σ and V, it can be expended as linear combination of orthonormal basis directions (u and v) with coefficient σ.
- U and V are both orthonormal matrices which means UU′ = VV′ = I , I is the identity matrix. So in above equation:

$$u_i u_i = 1, u_i u_p = 0, p \neq i,$$
$$v_i v_i = 1, v_i v_p = 0, p \neq i$$

Singular Value Decomposition

Σ is a diagonal matrix with singular values lying on the diagonal. σ1 ≥ σ2 ≥ … ≥ σp ≥ 0 with a descending order, are very much like the stretching parameter λ in eigen decomposition.

$$\Sigma = \begin{pmatrix} \sigma 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma p \end{pmatrix}$$

Thank you!