# Computing for Data Sciences

# Lecture 5

## Least Squares :

The method of least squares is a way of “solving” an overdetermined system of linear equations

$$Ax = b,$$

i.e., a system in which A is a rectangular m × n-matrix with more equations than unknowns (when $m > n$).

Historically, the method of least square was used by Gauss and Legendre to solve problems in astronomy and geodesy. The method was first published by Legendre in 1805 in a paper on methods for determining the orbits of comets.

The reason why more equations than unknowns arise in such problems is that repeated measurements are taken to minimize errors. This produces an overdetermined and often inconsistent system of linear equations.

For example, Gauss solved a system of eleven equations in six unknowns to determine the orbit of the asteroid Pallas.

As a concrete illustration, suppose that we observe the motion of a small object, assimilated to a point, in the plane. From our observations, we suspect that this point moves along a straight line, say of equation

$$y = dx + c.$$

Suppose that we observed the moving point at three different
locations (x1, y1), (x2, y2), and (x3, y3).
Then, we should have

$c + dx1 = y1,$
$c + dx2 = y2,$
$c + dx3 = y3.$

If there were no errors in our measurements, these equations
would be compatible, and c and d would be determined
by only two of the equations.

However, in the presence of errors, the system may be inconsistent. Yet, we would like to find c and d. The idea of the method of least squares is to determine (c, d) so that it minimizes the sum of the squares of the errors, namely

$$(c + dx1 - y1)2 + (c + dx2 - y2)2 + (c + dx3 - y3)2$$

In general, for an overdetermined m×n system $Ax = b$, what Gauss and Legendre discovered is that there are solutions x minimizing

$$||Ax - b||2$$

and that these solutions are given by the square n × n system

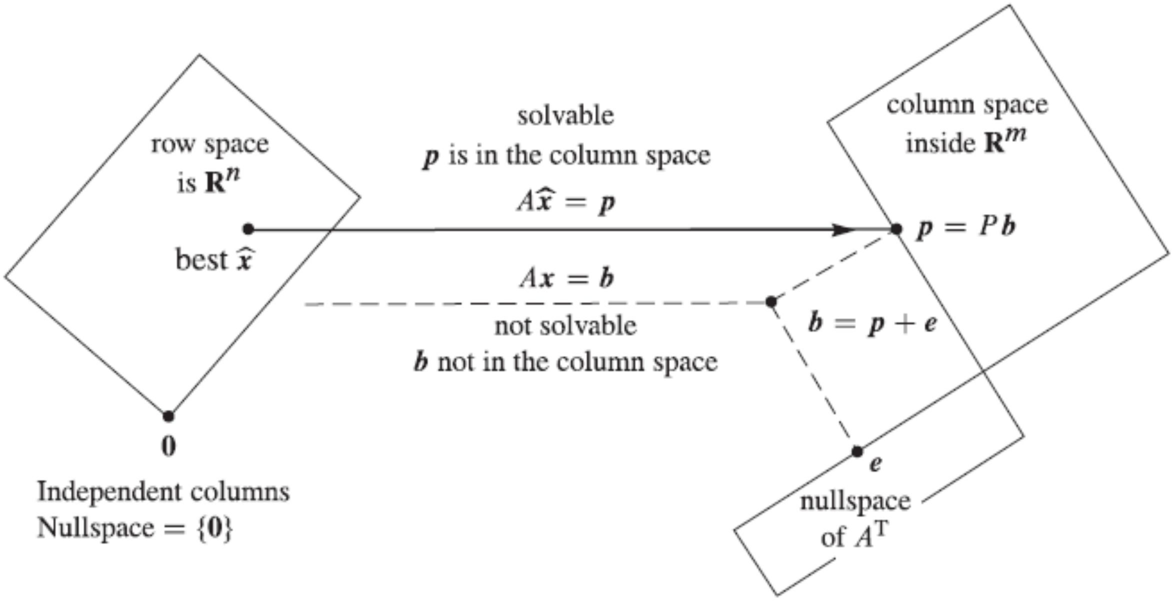
$$ATAx = ATb$$

Furthermore, when the columns of A are linearly independent, it turns out that ATA is invertible, and so x is unique and given by

$$x = (AT A)\text{-}1 AT b.$$

There are no solutions to the equation $Ax=b$ because b is not present in the column space of x, so in order to find a closest possible solution we split b, as shown in the figure. ‘b’ is split into parts p and e, p is called projection and e is called the error. Instead of solving for $Ax = b$, we now solve for $Ax = p$, by minimizing the error.

In the figure the point p is closest to b, and will hence give the least error. So x minimizes $E= ||b - Ax||2$



**The Singular Value Decomposition :**

**Suppose we have set of data point observations, then we like to convert this into set of values of linearly uncorrelated variables called principal components. There exists one principle component (PC) such that it has or takes maximum projection component of original data and minimizes error component.**

**Every set of data has two components :**

**1. Variance**

**2. Error.**

**Variance is the projection and Error is left over of the original data set.**

**Then next comes outliners, whose contribution towards the principal component is zero. Outliners are that whose dot product with principle component will be zero. They lie in component perpendicular to principle component. They are other components whose contribution will be high enough that they can't be neglected. These are called as Principal component 2(PC 2), Principal component 3(PC3) and so on. Then consider every other components then some of them will be neglected due to less significant contribution compared to others.**

**Theorem 1.1** *(The Singular Value Decomposition, SVD). Let $A$ be an $(m \times n)$ matrix with $m \geq n$. Then there exist orthogonal matrices $U$ $(m \times m)$ and $V$ $(n \times n)$ and a diagonal matrix $\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_n)$ $(m \times n)$ with $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0$, such that*

$$A = U\Sigma V^T$$

*holds. If $\sigma_r > 0$ is the smallest* singular value *greater than zero then the matrix $A$ has rank $r$.*

**Definition 1.1** *The column vectors of $U = [u_1, \ldots, u_m]$ are called the* left singular vectors *and similarly $V = [v_1, \ldots, v_n]$ are the* right singular vectors. *The values $\sigma_i$ are called the* singular values *of $A$.*

*If $A = U\Sigma V^T$. The column vectors of $V$ are the eigenvectors of the matrix $A^T A$ to the eigenvalues $\sigma_i^2$, $i = 1, \ldots, n$. The column vectors of $U$ are the eigenvectors of the matrix $AA^T$.*

$$A^T A = (U\Sigma V^T)^T U\Sigma V^T = VDV^T, \quad D = \Sigma^T\Sigma = \operatorname{diag}(\sigma_1^2, \ldots, \sigma_n^2).$$

Thus $A^T AV = VD$ and $\sigma_i^2$ is an eigenvalue of $A^T A$. Similarly

$$AA^T = U\Sigma V(U\Sigma V^T)^T = U^T\Sigma\Sigma^T U^T,$$

where $\Sigma\Sigma^T = \operatorname{diag}(\sigma_1^2, \ldots, \sigma_n^2, 0, \ldots, 0)$.

## Algorithm to find square root of a number :

**Algorithm describes actions on the input instance**
**Infinitely many correct algorithms for the same algorithmic problem.**

**For an algorithm to be good it should have efficient running time & space used. Also efficient in the no. of input data elements.**

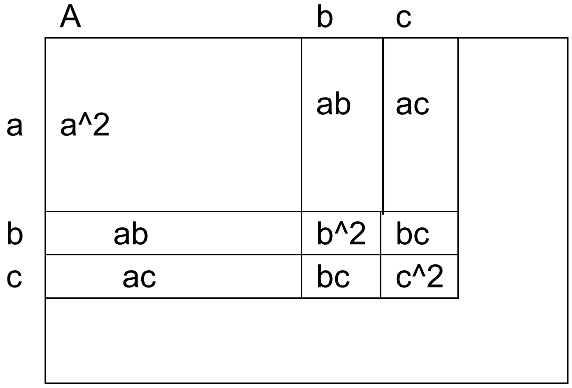
**Algorithm effiency of running time is clearly depicted in the below the Cases.**

**Methods for computing square root**

**Method 1: Finding square root of a number in the general method**

**In solving step first** a^2 **term gets cancelled which leaves the remainder is** b^2+2ab **for the next step.**

**After solving second step** 2ab **term gets cancelled leaving b^2 term as remainder.**
**This b^2 can be restated as (c+d)^2**
**And same steps are re-iterated.**



**After each step it is getting reduced by a factor of 10 following a log N base 10 complexity which is generally fast algorithm.**

**Algorithm to find square root of given number with precision :**

1. **Accept the number(n) whose square root is to be found**
2. **Accept the precision value**
3. **If the number is greater than 1 goto step 5**
4. **Assign n as lower bound and 1 as upper bound goto step 7**
5. **Assign n as upper bound and 1 as lower bound**
6. **Calculate the arithmetic mean(AM) of upper bound(UB) and lower bound(LB)**
7. **Check UB-Sqrt (n) < precision value. If yes goto step 9 else step 8**
8. **Assign recently calculatedAM(UB and LB) as upper bound and N/AM as lower bound and goto step 6**
9. **Print and Exit.**