Dimension Reduction → **PCA** — Feature Extraction → [d = d']
↳ spread or variance

Principal Component Analysis

* Feature selection works in general by selecting features | columns with high variance or high spread over the axis of graph.
↳ variance threshold ← [d > d']

* PCA → Reduces high Dimension data which are Imp to low Dimension Data for better understanding and visualization. [2D → 1D] [3D → 2D or 1D]

No. of PCA's = or < n

Shifting of Axis

Based on Variance of each PC line

Transform the data acc to max value

PCA

Find unit vector

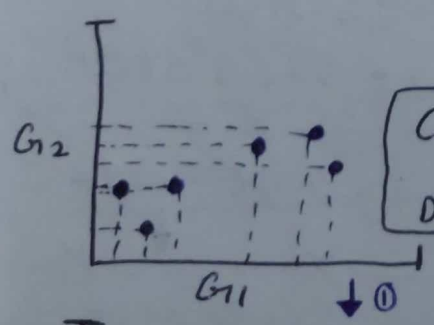
• Project of one vector onto Another (one point (Dist from (0,0) to (x,y)) vector ($\vec{x}$ and $\vec{u}$) → $\frac{\vec{u}.\vec{x}}{|u|}$ as u is unit vector so

Projected vector = $\vec{u}.\vec{x} = u^T x$

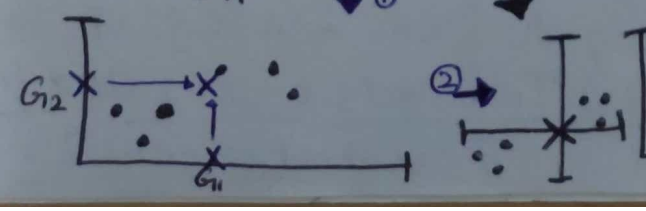
PCA retains the valuable Insights after Reducing the Dimensions
PCA Also Tells which feature is more important for clustering
PCA Also Tells us about the accuracies of the Reduced Graph

Graphical Understanding [2D - 1D]

TYPICAL DATA → [G1, G2] MOUSES



Center the Data

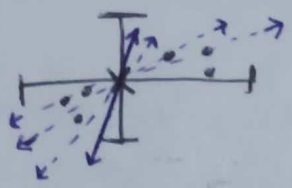


① avg measurement for Gene 1
② avg measurement for Gene 2
∴ with avg values – calculate the center of the Data

→ shift the center of the Data To the origins

→ shifting the Data donot change the position of Datapoints relativly

→ Try to fit a line that goes through Origin → Rotate the line until it fits the Data as well as goes through Origin



But how to decide the Best fit line ?

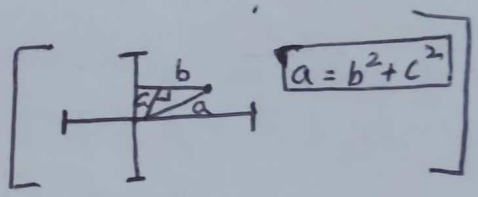
→ 2 Factors to Decide Best fitness of line

↳ a] measure the Distances from the data to the line that minimizes those Distances — (a)

b] maximizes the Distance from The Projected Points to the Origin. — (b)

- PCA projects the Data onto it

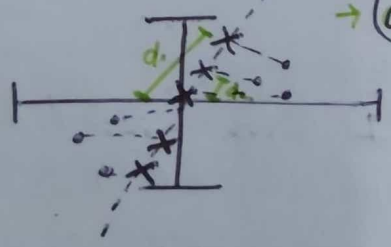
* Best fit line → (a) minimizes
(b) maximizes



What PCA Finally do ?

as it is easier to calculate c (or here b) so PCA FINDS the Best fitting line By → Maximizing The sum of The squared Distances from the projected points to the origin.

→ $(d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2 + d_6^2) = \max \rightarrow SS$ (Distance)



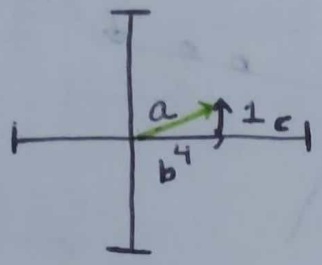
∴ decide the line with largest SS (Distance)

The line is known as PC1. (let's say slope of PC1 = 0.25)

for every 4 units that we go out along the Gene 1, we go up 1 unit along Gene 2. Data is more spreaded Towards Gene 1 and less Towards Gene 2.

Linear Combination of Gene 1 & 2 { To make PC 1: mix 4 Parts Gene 1 with 1 Part Gene 2 } Gene 1 is more significant when it is about Describing the spread of Data

4-1 → Linear Combⁿ



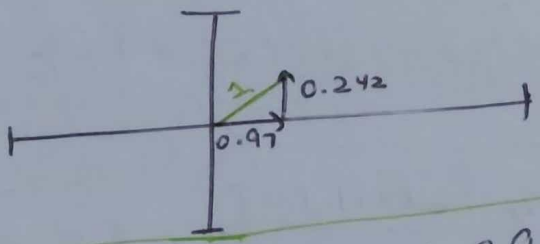
→ $a^2 = b^2 + c^2$

$a^2 = 16 + 1$

$a^2 = 17$ → $\boxed{a = 4.12}$

Length 'a' which forms as Hypotenous with the help of Linear Combⁿ is scaled to = 1 → unit vector

$\frac{a}{4.12}, \frac{b}{4.12}, \frac{c}{4.12} = \boxed{1, 0.97, 0.242}$ ratio of G_1 to G_2 Remain same



This unit vector consisting of 0.97 parts of Gene 1 and 0.242 parts Gene 2 | Linear Combⁿ of Gene 1 and Gene 2 is known as eigen vector for PC 1

and proportion of each gene/feature is known as LOADING SCORES

X X X

SS (Distance) = sum of square of Projected Points to origin of PC 1

$= d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2 + d_6^2$ = Eigenvalue for PC 1

X X X

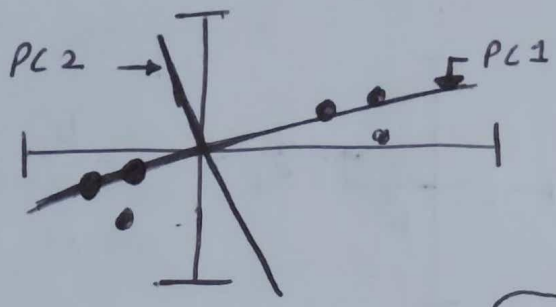
SS (Distance for PC 1) = Eigenvalue for PC 1

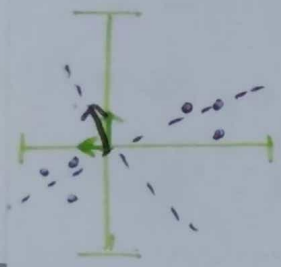
$\sqrt{\text{Eigen value for PC1}}$ = Singular value for PC 1

X X X

$\boxed{\text{PC-2}}$ →

PC 2 is simply the line through the origin that is ⊥ar to PC 1, without any further optimization that has to be done.





PC 2 → −1 Parts of Gene 1, 4 Parts of Gene 2 } loading scores of PC2: −0.242 parts of Gene 1, 0.96 parts of Gene 2

unit eigenvector of PC_2

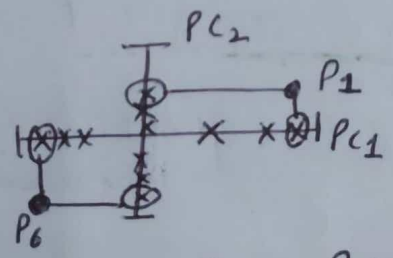
SS (Distance of PC2) = Eigenvalues for PC2

—— X —— X —— X ——

FINAL PCA PLOT

① SIMPLY Rotate everything so that PC 1 is horizontal

② Use Projected Points to find where the samples go in the PCA plot.



That's the general way of Doing PCA using Singular Value Decomposition (SVD). —

$\therefore \dfrac{SS(\text{Distance for } PC1)}{n-1}$ = Variation for PC1 —— X

$\therefore \dfrac{SS(\text{Distance for } PC2)}{n-1}$ = Variation for PC2 —— Y

—— X —— X —— X —— X ——

$X + Y$ = Total

$\left(\frac{X}{2}\right)100\%$ accounts for PC1

$\left(\frac{Y}{2}\right)100\%$ accounts for PC2

Variation around PCs

A graphical representation of the % of variation that each PC accounts for = **SCREE PLOT**

$\left(\frac{x}{z}\right)\times 100 \quad \left(\frac{y}{z}\right)\times 100$

PCA with 3 variables

Follows General steps

1. Center the Data
2. Find the Best fitting line that goes through the origin - PC1
   ↳ Recipe or Loading Score = no. of variables.
3. PC2 → Best fitting line that gives through the origin and ⊥ar to PC1
   ↳ Recipe or Loading Score = no. of variables
4. PC3 → Best fitting line that goes through the origin and ⊥ar to PC1 and PC2.

∴ If we have more genes, we would just keep on adding ⊥ar lines and rotating them. finding more and more principal components by

→ unit vector arised from Loading Scores normalized or scaled = eigenvector of that PC.

5. USE eigenvalues (SS(Distance)) to determine the proportion of variance (x, y, z) (Divide by n-1)

$(x+y+z) \quad \left(\left(\frac{x}{x+y+z}\right)100\right)$ = proportion of variance.

Eg → PC1 (79.1%) } majority → 94% so a 2D graph would be better as 94% variation.
PC2 (15.1%)
PC3 (6%)

[IMP] on the basis of proportion of variation the Decision is made to reduce complexity of Graph.

here 3D-2D (Remove everything Except PC1, PC2 and data)

- Project the samples on PC1 and PC2
- Rotate so that PC1 is horizontal and PC2 is vertical
- Place the points on new plot.

4D→2D decided on the Basis of scree plot or proportion of variance accountable by Each PCs

---

If bars in scree plot are similar and not have high Difference then Plotting 4D to 2D is noisy But can be used to form CLUSTERS of Data

[IMP]

Dimension Reduction

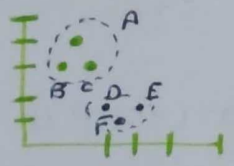
U-MAP

Uniform Manifold Approximation & Projection

- Create a low-dimension graph of this data that preserves the high Dimensional clusters and their rel^n the to each other.
- Initialize the low Dim points and then move the low-dimension points around untill they form clusters that have the same rel^n we saw in the high-dimensional Data.

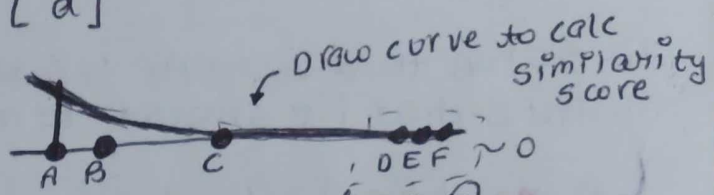
* Projecting Points will create mismash of Points irrespective of the axis
Increasing Dimensions will make things worse.

1 Calculate the SIMILARITY SCORES that help identify clustered points so it can try to preserve that clustering in the low-dimensional graph.



↳ First thing → Calculate the Distance B/w each pair of high dimensional points... [d]

✱ similarity scores for A →

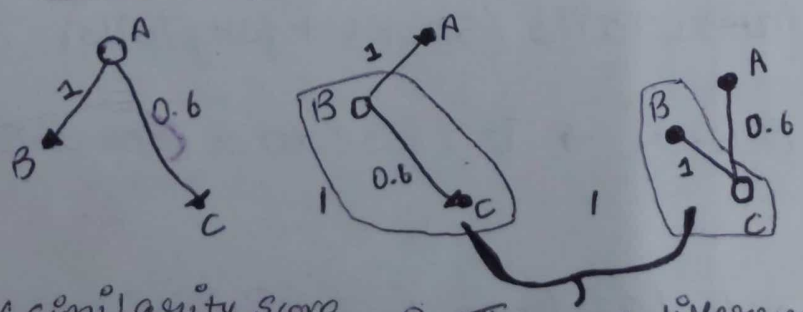


- plot A on graph
- plot Point B 'd' distance (eg 0.5) away from A.
- plot Point C 'd' distance (eg 2.4) away from A.
- Plot Points 'D', 'E' and 'F' far away from A, as they are far away from A in high Dimension.

NOTE The shape of the curve depends on the number of high-Dimensional neighbors that you want each point to have
↳ K → Default value = 15 But for small it can be 3 (it includes Point itself (K=3 so 2 neighbors) ←

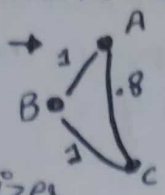
- $\log_2$ (neighbors) → $\log_2(3) = 1.6$ (defines shape of curve)
- curve shape in such a way that y-axis coordinate for the nearest neighbour (B, C) add up to the $\log_2$ (neighbors) = 1.6
  (B → 1, C → 0.6)
- These values of B & C (neighbors) are the similarity score rel to A.

✱ Similarly curves for B, C, D, E and F are created



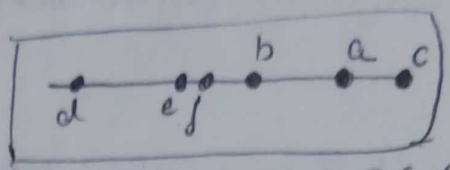
{ similarity score are not symmetrical } They are different ← They relative score of B & C { Because curve for each point are Different }

* UMAP makes them symmetrical using a Method similar to taking the average →



same for DEF

* THEN → UMAP initializes a low-dimensional graph



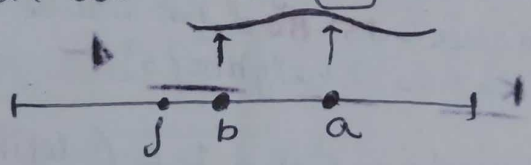
→ Not ideal as clusters are not similar

To solve this → UMAP picks 2 low-dimensional points that it should to move closer together

└ Does this By Randomly selecting a pair of points in a cluster ∝ to their high Dimensional score. [A and B]

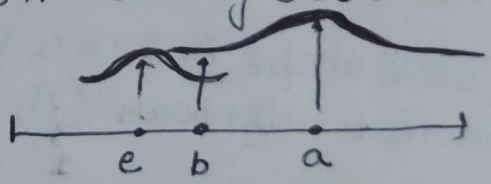
└ a and b closer, b far from f. → Randomly picks points that are not in B's high Dimensional cluster. [F]

Now → How much closer to a AND far from f



└ calculate low dimensional similarity scores y-axis coordinate on a curve ┘

└ fixed Bell shaped curve
└ same size



t-distribution

* t-distribution curve is like a Gaussian ~~or normal~~ Dist But have shorter and fatter Tails (Short + fatty Tails)

* As a and b are in same cluster → b close to a (maximise the low dim score)

* B and ~~E~~ F → Diff cluster → b far from F (minimize the low_Dim score).

similarly we do it for all remaining pair.

$\boxed{\text{IMP}}$ → In case where 

The score we want to minimize will still be pretty small + score - we want to minimize would be much larger

---

t-SNE similar to U-MAP

① T-SNE always starts with a random init of the low-dim graph

① BUT U-MAP uses Spectral Embedding to init the low-dim graph

② t-SNE moves every single point a little bit each iteration

② U-MAP JUST (move) one point - or a small subset of points, each time - helps it scale well with

---

Number of neighbors (K)

- A relatively low value for the number of neighbors (K) results in → small - independent clusters. [Details But not the Big Picture]
- A relatively large value for the no. of neighbors gives you more of the Big picture and less of details.

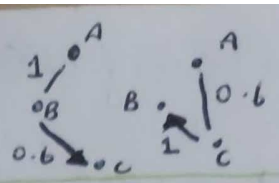
$\boxed{\text{The MATH}}$

The general eq for the similarity scores (y-axis coordinate)

$$\text{similarity score} = e^{-(\text{raw dist} - \text{dist to nearest neighbor})/\sigma}$$

∴ gets diff curve for DIFF POINT $\boxed{\sigma}$ ⇒ Change shape of curve → make $SS = \log_2(K)$.

∴ Asymetrical scores B/w points →



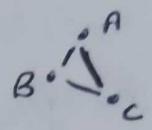
↳ averging to make them symmetrical → $\boxed{(S_1 + S_2) - S_1 S_2}$

S_1 (B→C), S_2 (C→B)

↳ $(0.6 + 1.0) - 0.6 \times 1.0 = 1.0$

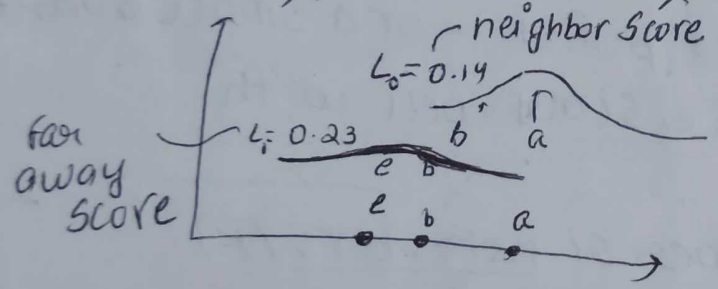
update the score to 1.0 ↵
(B and C)

Symmetrical scores → $(S_1 + S_2) - S_1 S_2$



low dimensional similarity scores $= \boxed{\dfrac{1}{1 + \alpha d^{2\beta}}} = \boxed{L}$

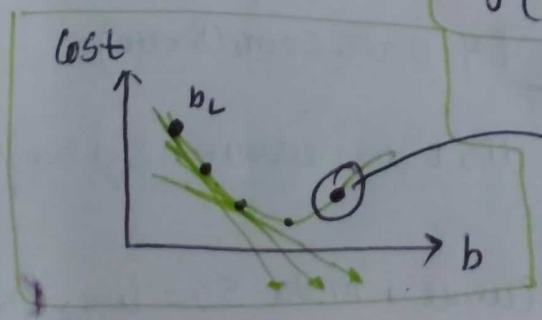
↳ α & β → How tightly the low-dimensional points can be packed together (default → $\alpha = 1.577$ | $\beta = 0.8951$)



→ As we move b to a & b frome the L_1 gets smaller & L_0 gets higher

→ Uses L_1 & L_0 to evaluate if b is at Right Place or not → Cost funcⁿ = $\log\left(\dfrac{1}{\text{neighbor}}\right) + \log\left(\dfrac{1}{1 - \text{not neighbor}}\right)$

↳ $\log\left(\dfrac{1}{L_0}\right) + \log\left(\dfrac{1}{1 - L_1}\right) = \boxed{b_L}$



b crosses a

Stohastic GD

→ Concept of gradient desent to acheive minima