8. Decision Tree [8. PCA] - YT - 5-minutes-Engineering

Principal Component Analysis:

→ What is PCA and why it is needed?

As we all know, including too many features during the training phase confuses the model which ultimately leads to overfitting of the model.

Now, to solve this overfitting problem, we generally do the dimensionality reduction. There are lots of ways of doing the dimensionality reduction to overcome the overfitting; 'PCA' is one of those.

→ How do we reduce the dimensions in PCA?

It is all about the views.

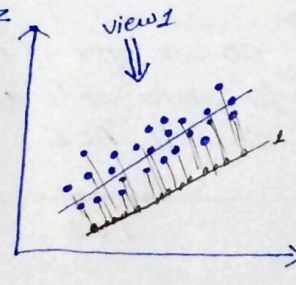
Suppose we have a dataset, which contains 2 independent variables.

Note that we do not consider PCA for dependent variable.

Here for the sake of simplicity, we are taking only 2 features/variables.

X	Z
2	5
6	7
7	3
9	5
2	10
⋮	⋮

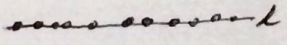
plot →



Now, how to reduce the dimensions (X & Z) to one dimension.

#Note: We will see all the mathematical details shortly. For now, only concentrate only on concepts.

→ So if we look at the above graph from 'view 1', then we get a line which has few datapoints projected on it.



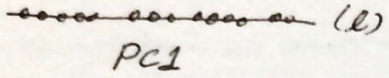
What we did here is basically, we projected our datapoints, which were plotted (distributed) across the dimensions (feature) and reduced the dimensions from two to one, which will eventually reduce overfitting as there is one less dimension now.

→ So by projecting these data points:

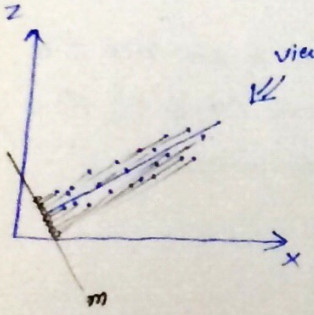
① We reduced the dimensions
② We reduced the complexities
③ We reduced the confusion for the model.

That is the actual motive of PCA.

• **PC1**: Now the component or new line (l) that we got after projecting the data points is called as a 'Principal Component'.
This can be called as Principal Component 1 if it explains the most of the variability.
We will later see, how to choose a PC as PC1 or PC2 or PC or so on mathematically. For now assume this is 'PC1'.

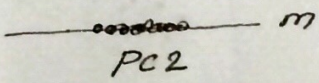


Another PC can be generated. Let's look the same original graph from view 2.



so we get

So we get a new line (m) on which the data sets are projected.
This is 'PC2'



Hence we have generated two principal components here.
From this we can conclude that:

> No. of principal components ≤ No. of features/variable/attributes

↳ axiom 1.

Note: Here in this example, we had only 2 attributes. However if we have more than 2 attributes, we use the same approach and generate multiple principal components.

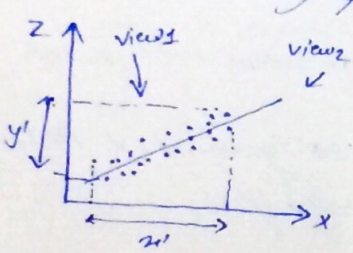
Ily, we can have PC3, PC4, PC5... etc. It all depends on number of features as per axiom 1.

Priority of Principal Components:

- PC1 has the highest priority as it explains/contributes maximum to the model.
- It is followed up by PC2, PC3, PC4 etc..
- Which component to assign as PC1?

Conceptually: We check which feature explains the maximum variance/ which component/attribute is ~~is~~ varies the most. (This is why standardization/normalization of your data is important).

eg. see the below graph:



→ If we see here, the x variable/feature is more widely distributed. It varies more than 'z' feature.

$x' > y'$

Hence, whatever PC we will get by projecting datapoints from 'view 1' will be our 'PC1'. Here, we are compromising with our 'z' attribute as it is varying less when compared to 'x' attribute.

Ily, whatever component we get from view 2 will be our PC2 as it is less varying.

Mathematically: We calculate the values of λ (lambda). It is also called as 'eigen value'. Which ever value of λ among all lambdas (λ_1, λ_2, λ_3 etc...) is greatest, the corresponding 'eigen vectors' ~~are~~ is assigned as 'PC1', then the second largest as PC2 and so on....

How many PCs should we have?

→ well, whatever number of PCs explains the 75-80% of the variance, those many PCs are taken as optimised number of Principal components.

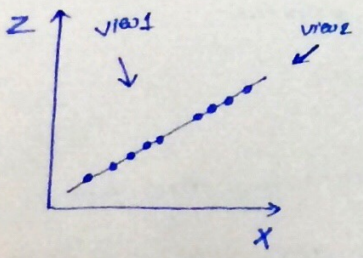
→ You can check the variance by plotting 'scree plot'.

→ **Orthogonal property:** The PCs that we get after PCA, there should be an orthogonal property between them.
Basically, each 'PC' should be orthogonal to to the other.
In simple words, PC1 should be totally independent of PC2, and vice versa.

Each PC should be independent of each other.

See, there is a logical reasoning behind it, which we shall see, using a graph.

Let's say, we have X & Z perfectly correlated with each other.



→ In this case, we generated PC1, which is perfectly fine.

→ But now, if we generate 'PC2' using view2, then it will project down to a single point.

Hence, there the sum of squared distance, on which basically the PCA works will be zero and hence our PC2 will be of no use.
Basically, because of this reason only, we say the Principal Components should be independent of each other.

PCA - How it works? - Step by Step.

Step 1: Standardization of your features (inputs)

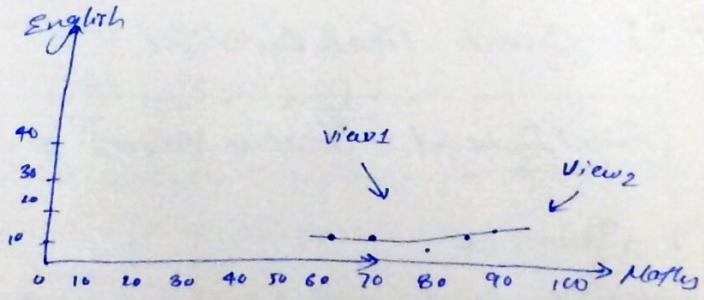
PCA is quite sensitive regarding the variances of the initial variables. If there are large differences between the ranges of initial variables, those variables with larger ranges will dominate over those with small ranges, which will lead to bias results.

Hence data transformation to common scale is necessary.

Let me give you an example: Let say we have maths subject marks out of 100 and English subject marks out of 10. Both subjects are important and contributes almost equally to the marks.

Without Standardization:

Maths	English
70	7
80	6
95	8.5
99	9.5
50	7.5



If we see from 'view 1', we can clearly see that maths is contributing a lot towards variance, while English is almost negligible (see on English axis, it shows very less variation, view 2).

However it is not the case in reality.

Both the subjects are equally contributing to the variation.

Hence standardization of data is important.

~~Step 2: Calculo~~

Step 1: Standardization of features (inputs)

step 2: Covariance Matrix Computation

Step 3: Compute Eigen values & Eigen vectors of the co-variance matrix
Assign the PCA's accordingly

step 4: Keep Important PCs and discard the less significant ones

Step 5: Create the final feature vector by taking eigen vectors kept in above step (according to PCs)

Step 6: Create Final Data set.

$$\text{Final Data Set} = (\text{Feature Vector})^T * (\text{Standardized Original Dataset})^T$$

T → transpose.

Note: Standardised Original Dataset does not contains target variable.

PCA - Example: For the sake of simplicity, we are taking only 2-d features (X & Y). However the same steps and concepts are applied for more than 2 features as well:

(1) Data we have is:

X	Y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9
$\bar{x}$	$\bar{y}$
1.81	1.91

→ total 10 observations

→ attributes are x & y

→ Standardization, we are skipping here, but if you do it programatically, make sure you do the standardization.

mean (x) i.e. $\boxed{\bar{x} = 1.81 \quad \& \quad \bar{y} = 1.91}$

Step 2: Calculating Co-variance matrix.

$$C = \begin{bmatrix} cov(x,x) & cov(x,y) \\ cov(y,x) & cov(y,y) \end{bmatrix}$$

Note: We had only 2 dimensions of data here, hence we have 2×2 covariance matrix, If we have higher dimension data, then we have higher dimension matrix.

eg: If we have 3 dimension data, then (x, y, z), the covariance matrix would have been:

$$C = \begin{bmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{bmatrix}$$

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

Applying the formula, we get:

$cov(x,x) = 0.6165$ $\quad$ $cov(y,x) = 0.6154$

$cov(x,y) = 0.6154$ $\quad$ $cov(y,y) = 0.7165$

∴ Covariance matrix is:

$$C = \begin{bmatrix} 0.6165 & 0.6154 \\ 0.6154 & 0.7165 \end{bmatrix}$$

Step 3: Compute Eigen value & Eigen vectors.

3.1 → Compute Eigen values : To get the eigen values, use the below form $\boxed{C - \lambda I = 0}$, then solve for the ~~determinants~~ determinants.

Where,

C = covariance matrix

λ = constant / eigen value

I = Identity matrix. (Its dimensions depends on 'C'.)
If 'C' is 3×3 matrix, 'I' is also 3×3 matrix.

∴ $$\begin{bmatrix} 0.6165 & 0.6154 \\ 0.6154 & 0.7165 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = 0$$

∴ $$\begin{bmatrix} 0.6165 - \lambda & 0.6154 - 0 \\ 0.6154 - 0 & 0.7165 - \lambda \end{bmatrix} = 0$$

solve for ~~determinants~~ determinants, ~~this~~ ~~with~~ form we get quadratic equation in terms of lambda:

$$(0.6165 - \lambda) * (0.7165 - \lambda) - 0.6154 * 0.6154 = 0$$

Here the above equation is quadratic, hence it has 2 roots.
after solving we get the roots as:

$\boxed{\lambda_1 = 0.4908}$ $\boxed{\lambda_2 = 1.2840}$

3.2 → Compute Eigen vectors:

ⓐ Substitute λ_1 & λ_2 in: ~~(A-λI)x=0~~ $(AX=0)$

$$\begin{bmatrix} 0.6165-\lambda & 0.6154 \\ 0.6154 & 0.7165-\lambda \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} = 0$$

& solve for it.

Hence: $\lambda_1 = 0.4908$

$$\begin{bmatrix} 0.6165-0.4908 & 0.6154 \\ 0.6154 & 0.7165-0.4908 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = 0$$

$\therefore$ $0.1257\,x_1 + 0.6154\,y_1 = 0$

$0.6154\,x_1 + 0.2257\,y_1 = 0$

$\therefore$ we get $\boxed{x_1 = -0.735 \quad y_1 = 0.677}$ ---- vector ①

↳ Eigen vector for λ_1 $\therefore$ $v_1 = \begin{bmatrix} -0.735 \\ 0.677 \end{bmatrix}$

IInly for $\lambda_2 = 1.2840$

$$\begin{bmatrix} -0.6675 & 0.6154 \\ 0.6154 & -0.5675 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = 0$$

$\boxed{x_2 = -0.677 \quad y_2 = -0.735}$ → Eigen vectors for λ_2

--- vector ② $v_2 = \begin{bmatrix} -0.677 \\ -0.735 \end{bmatrix}$

Note: Here we had 2 eigen values. hence we got 2 eigen vectors if we had 3 eigen value we shall have got 3 eigen vectors and so on..

Assigning PC1 & PC2:

Here $\lambda_2 = 1.2840$ & $\lambda_1 = 0.4908$ $\boxed{\lambda_2 > \lambda_1}$

$\therefore$ corresponding eigen vector of λ_2 will be our PC1.

$\therefore$

$$\begin{bmatrix} -0.678 & -0.73 \end{bmatrix} \rightarrow PC1$$

$$\begin{bmatrix} -0.735 & 0.677 \end{bmatrix} \rightarrow PC2$$

Step 4&5: Keeping both here. However we can discard PC2 as well.

If we keep both:

Feature vector = $\begin{bmatrix} (v1) & (v2) \\ -0.735 & -0.677 \\ 0.677 & -0.735 \end{bmatrix}$

If we discard PC2: (v_1)

$$\begin{bmatrix} -0.735 \\ 0.677 \end{bmatrix}$$

However we keep both here.

Step 6:

$$\text{FinalDataSet} = (\text{Feature vector})^T * [\text{Standardization}(\text{Data})]^T$$