Dimensionality Reductions

Is it important to standardize the data before applying PCA?

- Usually, the aim of standardization is to assign equal weights to all the variables.
- PCA finds new axes based on the covariance matrix of original variables.
- As the covariance matrix is sensitive to the standardization of variables therefore if we use features of different scales, we often get misleading directions.

Moreover, if all the variables are on the same scale, then there is no need to standardize the variables.

What is variance, covariance, covariance matrix?

1. The variance explains how the values vary in a variable. It depends on how the values far from each other.

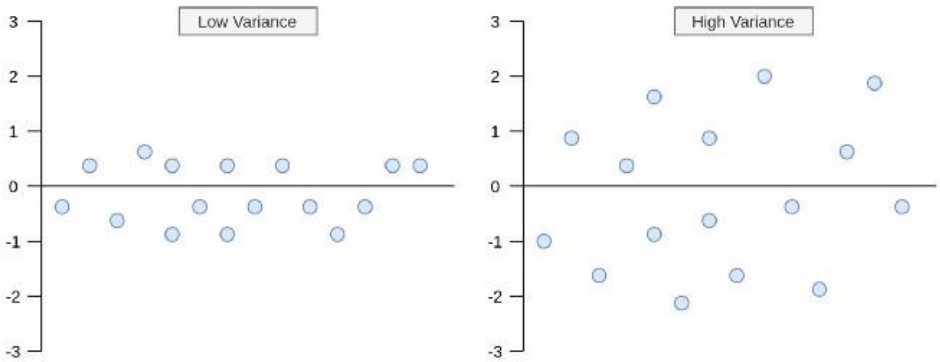
Population mean is known

$$var(x) = \frac{\sum_i^n (x_i - \mu)^2}{N}$$

Population mean is unknown

$$var(x) = \frac{\sum_i^n (x_i - \overline{x})^2}{N - 1}$$

   a. In the formula, each value in the variable subtracts from the mean of that variable.
   b. After the differences are squared, it gets divided by the number of values (N) in that variable.
   c. Okay, what happens when the variance is low or high. You can see Figure 1 to understand what happens if the variance value is low or high.



2. Now, it is time to have a look at the covariance formula. It is as simple as the variance formula.
   a. Unlike the variance, covariance is calculated between two different variables.
   b. Its purpose is to find the value that indicates how these two variables vary together.
   c. In the covariance formula, the values of both variables are multiplied by taking the difference from the mean. You can see Formula 2 to understand it clearly.

Population mean is known

$$cov(x, y) = \frac{\sum_i^n (x_i - \mu) \cdot (y_i - \mu)}{N}$$

Population mean is unknown

$$cov(x) = \frac{\sum_i^n (x_i - \overline{x}) \cdot (y_i - \overline{y})}{N - 1}$$

The only difference between variance and covariance is using the values and means of two variables instead of one.

Note:

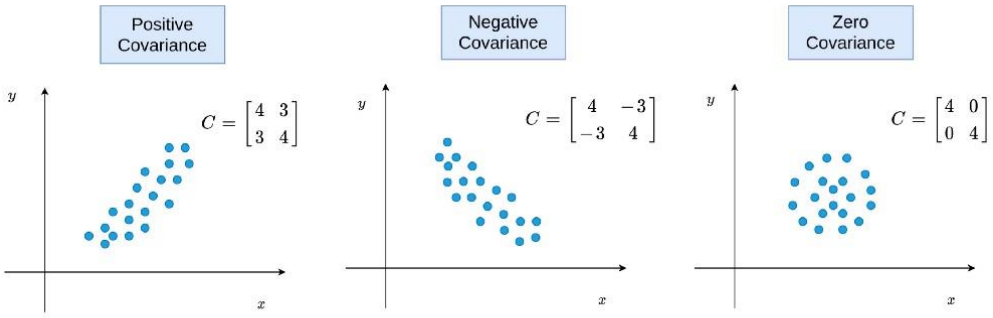
1. As you can see from Formula 1 and Formula 2, there are two different formulas as population known and unknown.
2. When we work on sample data, we don't know the population mean, we know only the sample mean.
3. That's why we should use the formula with N-1. When we have the all population of the subject, we can you the with N.

3. The Covariance Matrix
    1. The second thing that you should know is the covariance matrix.
    2. Because covariance can only be calculated between two variables, covariance matrices stand for representing covariance values of each pair of variables in multivariate data.
    3. Also, the covariance between the same variables equals variance, so, the diagonal shows the variance of each variable.
    4. Suppose there are two variables as x and y in our data set.

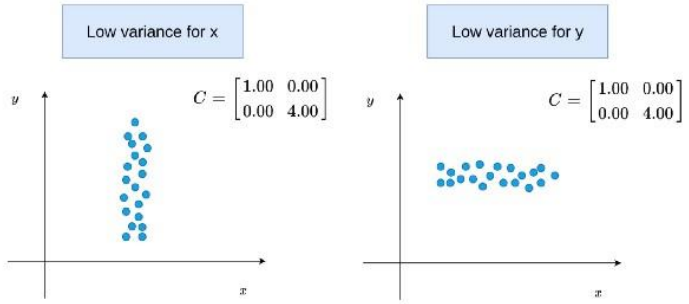
$$\begin{array}{c} \\ x \\ y \end{array}\begin{array}{c} \begin{array}{cc} x & y \end{array} \\ \begin{bmatrix} var(x) & cov(x,y) \\ cov(x,y) & var(y) \end{bmatrix} \end{array} \qquad \begin{array}{c} \\ x \\ y \\ z \end{array}\begin{array}{c} \begin{array}{ccc} x & y & z \end{array} \\ \begin{bmatrix} var(x) & cov(x,y) & cov(x,z) \\ cov(x,y) & var(y) & cov(y,z) \\ cov(x,z) & cov(y,z) & var(z) \end{bmatrix} \end{array}$$

It is a symmetric matrix that shows covariances of each pair of variables. These values in the covariance matrix show the distribution magnitude and direction of multivariate data in multidimensional space. By controlling these values we can have information about how data spread among two dimensions.

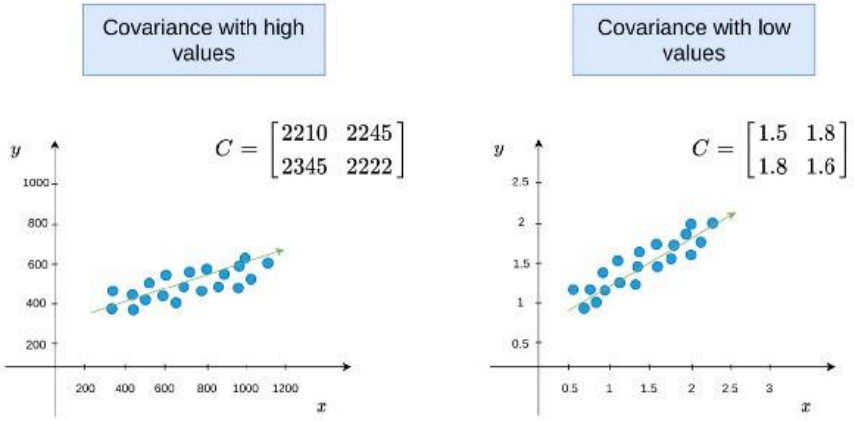
5. Positive, Negative, and Zero States of The Covariance.
    i. When Xi-Xmean and Yi-Ymean are both negative or positive at the same time, multiplication returns a positive value. If the sum of these values is positive, covariance gets found as positive.
    ii. It means variable X and variable Y variate in the same direction. In other words, if a value in variable X is higher, it is expected to be high in the corresponding value in variable Y too.
    iii. In short, there is a positive relationship between them.
    iv. If there is a negative covariance, this is interpreted right as the opposite.
    v. That is, there is a negative relationship between the two variables.
    vi. The covariance can only be zero when the sum of products of Xi-Xmean and Yi-Ymean is zero.
    vii. However, the products of Xi-Xmean and Yi-Ymean can be near-zero when one or both are zero. In such a scenario, there aren't any relations between variables.



As another possible scenario, we can have a distribution something like in Figure 3. It happens while the covariance is near zero and the variance of variables are different.



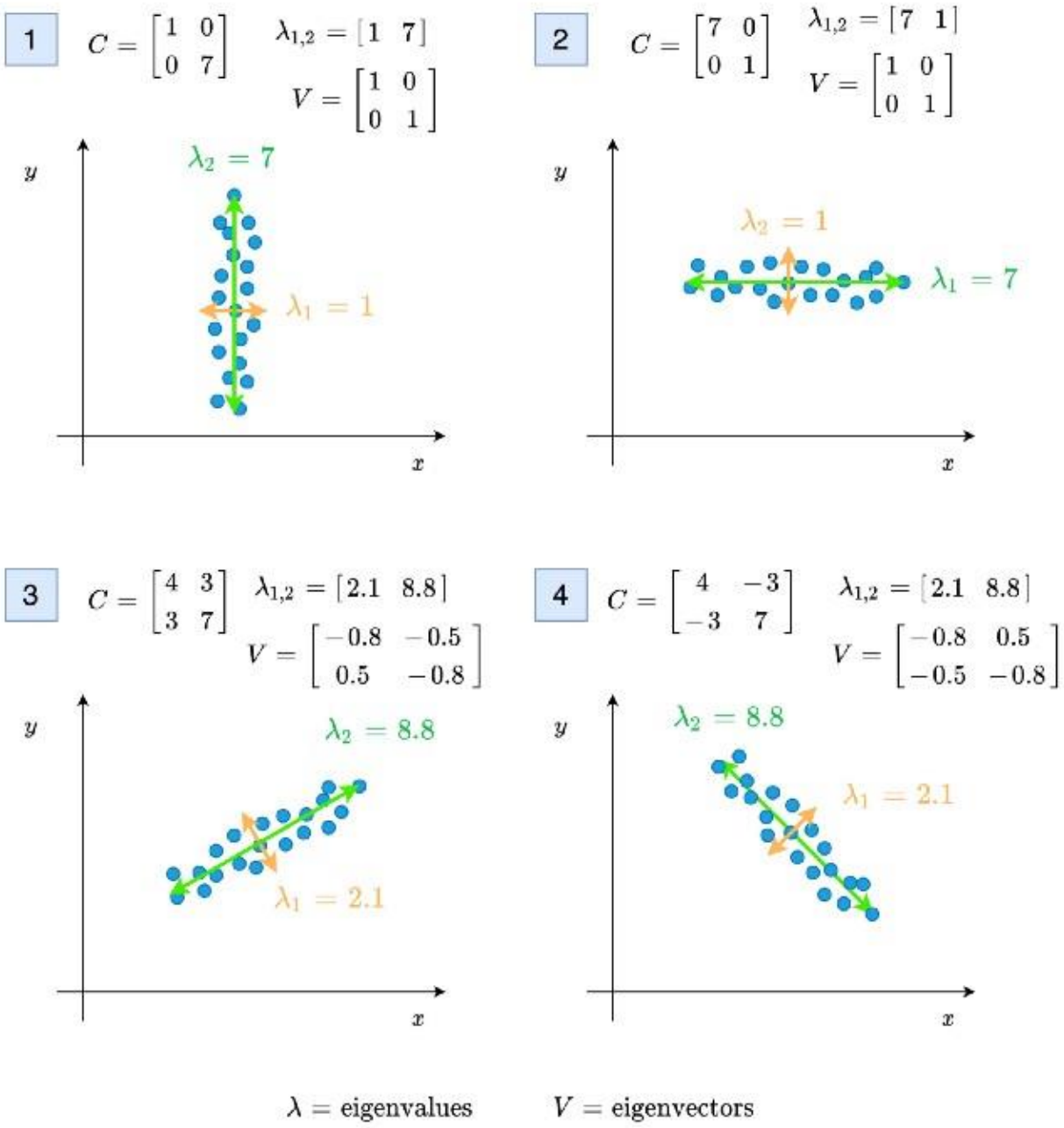
4. The size of covariance value
    a. Unlike correlation, covariance values do not have a limit between -1 and 1.
    b. Therefore, it may be wrong to conclude that there might be a high relationship between variables when the covariance is high.
    c. The size of covariance values depends on the difference between values in variables.
    d. For instance, if the values are between 1000 and 2000 in the variable, it possible to have high covariance.
    e. However, if the values are between 1 and 2 in both variables, it is possible to have a low covariance.
    f. Therefore, we can't say the relationship in the first example is stronger than the second.
    g. The covariance stands for only the variation and relation direction between two variables.
    h. Although the covariance in the first figure is very large, the relationship can be higher or the same in the second figure.



5. Eigenvalues and Eigenvectors of Covariance Matrix
    a. The methods that require a covariance matrix to find the magnitude and direction of the data points use eigenvalues and eigenvectors.
    b. For example, the eigenvalues represent the magnitude of the spread in the direction of the principal components in PCA.
    c. the first and second plots show the distribution of points when the covariance is near zero. When the covariance is zero, eigenvalues will be directly equal to the variance values.
    d. The third and fourth plots represent the distribution of points when the covariance is different from zero. Unlike the first two, eigenvalues and eigenvectors should be calculated for these two.
    e. The eigenvalues represent the magnitude of the spread for both variables x and y.
    f. The eigenvectors show the direction.
    g. It is possible to find the angle of propagation from the arccosine of the value v[0,0] when the covariance is positive. If the covariance is negative, the cosine of the valuev[0,0]gives the spread direction.

How do you find eigenvalues and eigenvectors from the covariance matrix?

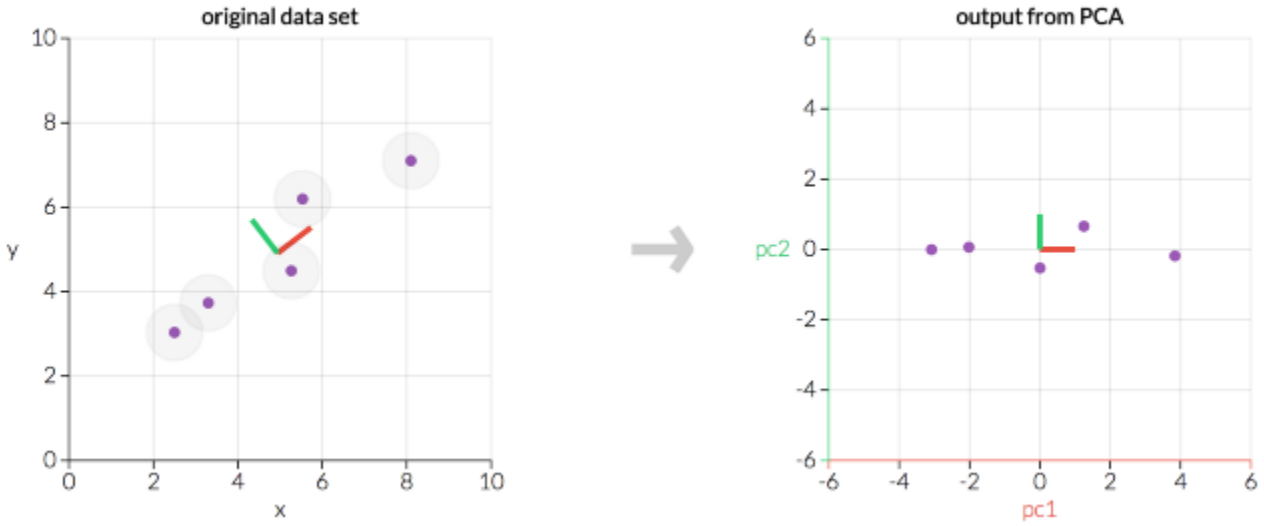
- You can find both eigenvectors and eigenvalues using NumPY in Python. First thing you should do is to find covariance matrix using method numpy.cov(). After you found the covariance matrix you can use the method numpy.linalg.eig(M) to find eigenvectors and eigenvalues.



List down the steps of a PCA algorithm.

1. If a Y variable exists and is part of your data, then separate your data into Y and X
2. Take the matrix of independent variables X and, for each column, subtract the mean of that column from each entry. (This ensures that each column has a mean of zero.)
3. Decide whether or not to standardize.

   a. Given the columns of X, are features with higher variance more important than features with lower variance, or is the importance of features independent of the variance?
   b. In this case, importance means how well that feature predicts Y
   c. If the importance of features is independent of the variance of the features, then divide each observation in a column by that column's standard deviation.
   d. This, combined with step 2, standardizes each column of X to make sure each column has mean zero and standard deviation 1.
   e. Call the centered (and possibly standardized) matrix Z.
4. Take the matrix Z, transpose it, and multiply the transposed matrix by Z. (Writing this out mathematically, we would write this as Z^TZ.) The resulting matrix is the covariance matrix of Z, up to a constant.
5. Calculate the eigenvectors and their corresponding eigenvalues of Z^TZ.
   a. This is quite easily done in most computing package
   b. in fact, the eigen decomposition of Z^TZ is where we decompose Z^TZ into PDP^{-1}, where P is the matrix of eigenvectors and D is the diagonal matrix with eigenvalues on the diagonal and values of zero everywhere else.
   c. The eigenvalues on the diagonal of D will be associated with the corresponding column in P — that is, the first element of D is λ_1 and the corresponding eigenvector is the first column of P.
   d. This holds for all elements in D and their corresponding eigenvectors in P.
   e. We will always be able to calculate PDP^{-1} in this fashion. (Bonus: for those interested, we can always calculate PDP^{-1} in this fashion because Z^TZ is a symmetric, positive semidefinite matrix.)
6. Take the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda p$ and sort them from largest to smallest.
   a. In doing so, sort the eigenvectors in P accordingly.
   b. For example, if λ_2 is the largest eigenvalue, then take the second column of P and place it in the first column position.
   c. Depending on the computing package, this may be done automatically. Call this sorted matrix of eigenvectors P*.
   d. The columns of P* should be the same as the columns of P, but perhaps in a different order Note that these eigenvectors are independent of one another.
7. Calculate Z* = ZP*.
   a. This new matrix, Z*, is a centered/standardized version of X but now each observation is a combination of the original variables, where the weights are determined by the eigenvector.
   b. As a bonus, because our eigenvectors in P* are independent of one another, each column of Z* is also independent of one another!



Note two things in this graphic:

1. The two charts show the exact same data, but the right graph reflects the original data transformed so that our axes are now the principal components.
2. **In both graphs, the principal components are perpendicular to one another.**
3. In fact, every principal component will ALWAYS be orthogonal (a.k.a. official math term for perpendicular) to every other principal component.
4. Because our principal components are orthogonal to one another, they are statistically linearly independent of one another… which is why our columns of Z* are linearly independent of one another!

8. Finally, we need to determine how many features to keep versus how many to drop. There are three common methods to determine this, discussed below and followed by an explicit example:
    a. Calculate the proportion of variance explained for each feature, sort features by proportion of variance explained and plot the cumulative proportion of variance explained as you keep more features.
    b. This plot is called a scree plot, shown below.
    c. One can pick how many features to include by identifying the point where adding a new feature has a significant drop in variance explained relative to the previous feature and choosing features up until that point. (I call this the "find the elbow" method, as looking at the "bend" or "elbow" in the scree plot determines where the biggest drop in proportion of variance explained occurs.)
    d. Because each eigenvalue is roughly the importance of its corresponding eigenvector, the proportion of variance explained is the sum of the eigenvalues of the features you kept divided by the sum of the eigenvalues of all features.

Once we've dropped the transformed variables we want to drop, we're done! That's PCA.

Why does PCA work?

1. First, the covariance matrix Z^TZ is a matrix that contains estimates of how every variable in Z relates to every other variable in Z. Understanding how one variable is associated with another is quite powerful.
2. Second, eigenvalues and eigenvectors are important. Eigenvectors represent directions. Think of plotting your data on a multidimensional scatterplot. Then one can think of an individual eigenvector as a particular "direction" in your scatterplot of data. Eigenvalues represent magnitude, or importance. Bigger eigenvalues correlate with more important directions.
3. Finally, we make an assumption that more variability in a particular direction correlates with explaining the behavior of the dependent variable. Lots of variability usually indicates signal, whereas little variability usually indicates noise. Thus, the more variability there is in a particular direction is, theoretically, indicative of something important we want to detect. (The setosa.io PCA applet is a great way to play around with data and convince yourself why it makes sense.)

Thus, PCA is a method that brings together:

1. A measure of how each variable is associated with one another. (Covariance matrix.)
2. The directions in which our data are dispersed. (Eigenvectors.)
3. The relative importance of these different directions. (Eigenvalues.)
4. PCA combines our predictors and allows us to drop the eigenvectors that are relatively unimportant.

What are the assumptions taken into consideration while applying PCA?

The assumptions needed for PCA are as follows:

1. PCA is based on Pearson correlation coefficients. As a result, there needs to be a linear relationship between the variables for applying the PCA algorithm.
2. For getting reliable results by using the PCA algorithm, we require a large enough sample size i.e., we should have sampling adequacy.

3. Your data should be suitable for data reduction i.e., we need to have adequate correlations between the variables to be reduced to a smaller number of components.
4. No significant noisy data or outliers are present in the dataset.

What are the properties of Principal Components in PCA?

The properties of principal components in PCA are as follows:

1. These Principal Components are linear combinations of original variables that result in an axis or a set of axes that explain/s most of the variability in the dataset.
2. All Principal Components are orthogonal to each other.
3. The first Principal Component accounts for most of the possible variability of the original data i.e, maximum possible variance.
4. The number of Principal Components for n-dimensional data should be at utmost equal to n(=dimension). For Example, there can be only two Principal Components for a two-dimensional data set.

What does a Principal Component in a PCA signify? How can we represent them mathematically?

- The Principal Component represents a line or an axis along which the data varies the most and it also is the line that is closest to all of the n observations in the dataset.
- In mathematical terms, we can say that the first Principal Component is the eigenvector of the covariance matrix corresponding to the maximum eigenvalue.
- Accordingly,
    - Sum of squared distances = Eigenvalue for PC-1
    - Sqrt of Eigenvalue = Singular value for PC-1

Can PCA be used for regression-based problem statements? If Yes, then explain the scenario where we can use it.

Yes, we can use Principal Components for regression problem statements.

- PCA would perform well in cases when the first few Principal Components are sufficient to capture most of the variation in the independent variables as well as the relationship with the dependent variable.
- The only problem with this approach is that the new reduced set of features would be modeled by ignoring the dependent variable Y when applying a PCA and while these features may do a good overall job of explaining the variation in X, the model will perform poorly if these variables don't explain the variation in Y.

Can we use PCA for feature selection?

- Feature selection refers to choosing a subset of the features from the complete set of features.
- No, PCA is not used as a feature selection technique because we know that any Principal Component axis is a linear combination of all the original set of feature variables which defines a new set of axes that explain most of the variations in the data.
- Therefore while it performs well in many practical settings, it does not result in the development of a model that relies upon a small set of the original features.

Comment whether PCA can be used to reduce the dimensionality of the non-linear dataset.

- PCA does not take the nature of the data i.e, linear or non-linear into considerations during its algorithm run but PCA focuses on reducing the dimensionality of most datasets significantly. PCA can at least get rid of useless dimensions.
- However, reducing dimensionality with PCA will lose too much information if there are no useless dimensions.

How can you evaluate the performance of a dimensionality reduction algorithm on your dataset?

- A dimensionality reduction algorithm is said to work well if it eliminates a significant number of dimensions from the dataset without losing too much information. Moreover, the use of dimensionality reduction in preprocessing before training the model allows measuring the performance of the second algorithm.
- We can therefore infer if an algorithm performed well if the dimensionality reduction does not lose too much information after applying the algorithm.

What are the Advantages of Dimensionality Reduction?

1. Less misleading data means model accuracy improves.
2. Fewer dimensions mean less computing. Less data means that algorithms train faster.
3. Less data means less storage space required.
4. Removes redundant features and noise.
5. Dimensionality Reduction helps us to visualize the data that is present in higher dimensions in 2D or 3D.

What are the Disadvantages of Dimensionality Reduction?

1. While doing dimensionality reduction, we lost some of the information, which can possibly affect the performance of subsequent training algorithms.
2. It can be computationally intensive.
3. Transformed features are often hard to interpret.
4. It makes the independent variables less interpretable.

Consider a set of 2D points {(-3,-3), (-1,-1),(1,1),(3,3)}. We want to reduce the dimensionality of these points by 1 using PCA algorithms. Assume sqrt(2)=1.414.

1. The eigenvalue of the data matrix XXT is equal to _____?
2. The weight matrix W will be equal to_____?
3. The reduced dimensionality data will be equal to_____?

Here the original data resides in R2 i.e, two-dimensional space, and our objective is to reduce the dimensionality of the data to 1 i.e, 1-dimensional data ⇒ K=1

1. Get the Dataset
    a. Here data matrix X is given by [[-3, -1, 1 ,3], [-3, -1, 1, 3]]
2. Compute the mean vector (μ)
    a. Mean Vector: [{-3+(-1)+1+3}/4, {-3+(-1)+1+3}/4] = [0, 0]
3. Subtract the means from the given data
    a. Since here the mean vector is 0, 0 so while subtracting all the points from the mean we get the same data points.
4. Compute the covariance matrix
    a. Therefore, the covariance matrix becomes XXT since the mean is at the origin.
    b. Therefore, XXT becomes [[-3, -1, 1 ,3], [-3, -1, 1, 3]] ([[-3, -1, 1 ,3], [-3, -1, 1, 3]])T
    c. = [[20, 20], [20, 20]]
5. Determine the eigenvectors and eigenvalues of the covariance matrix
    a. det(C-λI)=0 gives the eigenvalues as 0 and 40.
    b. Now, choose the maximum eigenvalue from the calculated and find the eigenvector corresponding to λ = 40 by using the equations CX = λX :
    c. Accordingly, we get the eigenvector as (1/√ 2) [1, 1]
    d. Therefore, the eigenvalues of matrix XXT are 0 and 40.
6. Choosing Principal Components and forming a weight vector
    a. Here, U = R^2×1 and equal to the eigenvector of XXT corresponding to the largest eigenvalue.

b. Now, the eigenvalue decomposition of C=XXT
c. And W (weight matrix) is the transpose of the U matrix and given as a row vector.
d. Therefore, the weight matrix is given by [1 1]/1.414

7. Deriving the new data set by taking the projection on the weight vector
    a. Now, reduced dimensionality data is obtained as xi = UT Xi = WXi
        1. x1 = WX1= (1/√ 2) [1, 1] [-3, -3]T = – 3√ 2
        2. x2 = WX2= (1/√ 2) [1, 1] [-1, -1]T = – √ 2
        3. x3 = WX3= (1/√ 2) [1, 1] [1, 1]T = – √ 2
        4. x4 = WX4= (1/√ 2) [1, 1] [3, 3]T = – 3√ 2

Therefore, the reduced dimensionality will be equal to {-3*1.414, -1.414,1.414, 3*1.414}.

Imagine, you have 1000 input features and 1 target feature in a machine learning problem. You have to select 100 most important features based on the relationship between input features and the target features. Do you think, this is an example of dimensionality reduction?

Yes

It is not necessary to have a target variable for applying dimensionality reduction algorithms.

TRUE (LDA is an example of supervised dimensionality reduction algorithm.)

I have 4 variables in the dataset such as – A, B, C & D. I have performed the following actions:

1. Step 1: Using the above variables, I have created two more variables, namely E = A + 3 * B and F = B + 5 * C + D.
2. Step 2: Then using only the variables E and F I have built a Random Forest model.

Could the steps performed above represent a dimensionality reduction method?

True (Yes, Because Step 1 could be used to represent the data into 2 lower dimensions.)

Principal Component Analysis (PCA). Which of the following is/are true about PCA?

1. PCA is an unsupervised method
2. It searches for the directions that data have the largest variance
3. Maximum number of principal components <= number of features
4. All principal components are orthogonal to each other

Suppose we are using dimensionality reduction as pre-processing technique, i.e, instead of using all the features, we reduce the data to k dimensions with PCA. And then use these PCA projections as our features. Which of the following statement is correct?

Higher ‘k’ means less regularization. Higher k would lead to less smoothening as we would be able to preserve more characteristics in data, hence less regularization.

t-SNE cost function?

Cost function of SNE is asymmetric in nature. Which makes it difficult to converge using gradient decent. A symmetric cost function is one of the major differences between SNE and t-SNE.

What is the importance of using PCA before the clustering?

In order to improve your clustering efficiency, you need to find which dimension of data maximize the features variance, also with the result you can find the explained variance for each dimension. As more variance you have, less data you loss.

What will happen when eigenvalues are roughly equal?

PCA will perform badly. When all eigen vectors are same in such case you won't be able to select the principal components because in that case all principal components are equal.

The below snapshot shows the scatter plot of two features (X1 and X2) with the class information (Red, Blue). You can also see the direction of PCA and LDA.

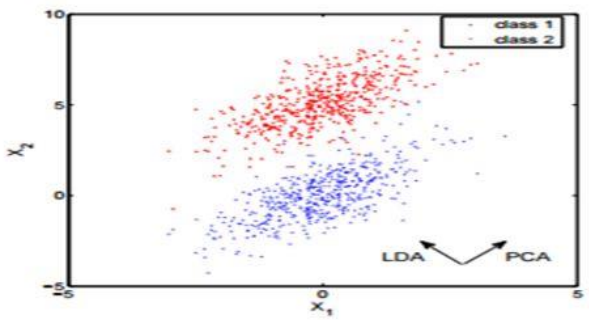
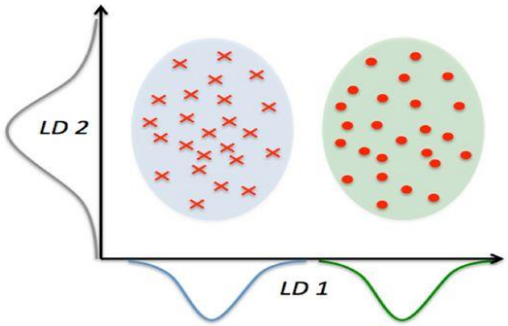


Figure 3: PCA vs LDA.

Which of the following method would result into better class prediction?

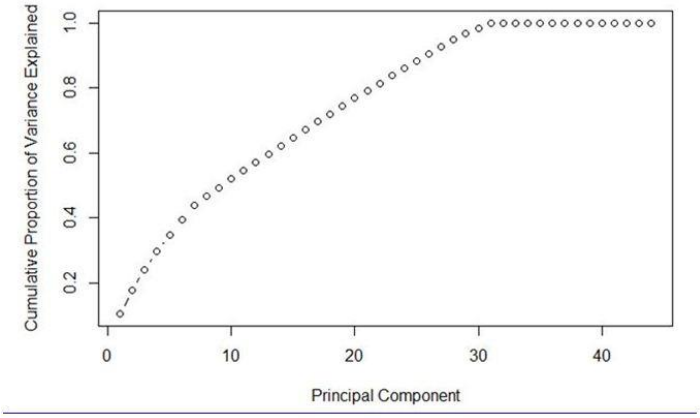
Building a classification algorithm with LDA. If our goal is to classify these points, PCA projection does only more harm than good—the majority of blue and red points would land overlapped on the first principal component. Hence PCA would confuse the classifier.

In LDA, the idea is to find the line that best separates the two classes. In the given image which of the following is a good projection?



LD1 Is a good projection because it best separates the class.

What are the optimum number of principle components in the below figure?



We can see in the above figure that the number of components = 30 is giving highest variance with lowest number of components.

What is the difference between Factor Analysis Vs PCA

Both are dimension reduction techniques, but, the main difference between Factor Analysis and PCA is the way they try to reduce the dimensions.

1. Factor Analysis
    a. It tries to find the hidden groups of variables, which is like a common driving factor for the type of values in all those variables,
    b. Example, customer survey, if you are unhappy with the taste of the coffee then all those questions about coffee taste in the survey will have low scores! Hence Bad Taste will be a Factor and questions like Coffee sweetness rating, Coffee bitterness rating, Coffee Freshness rating will represent the individual variables that will have low ratings.
    c. Factor Analysis will form equations like below:
        i. Coffee sweetness rating = β1 * (Bad Taste) +C1
        ii. Coffee bitterness rating = β2 * (Bad Taste) +C2
        iii. Coffee Freshness rating = β3 * (Bad Taste) +C3
2. PCA
    a. It tries to find fewer new variables called Principal Components which are linear combinations of the variables in the data, hence these fewer variables "represent" all the variables, while reducing the dimensions.
    b. Here the goal is to create a new dataset that has the same number of rows but, lesser number of columns (Principal Components) which explains the variance of all the original variables in the data.
    c. PCA will form equations like below:
        1. PC1=α1 * (Coffee sweetness rating) + α2(Coffee bitterness rating) + α3(Coffee Freshness rating)
        2. PC2=β1 * (Coffee sweetness rating) + β2(Coffee bitterness rating) + β3(Coffee Freshness rating)

Algorithm	**Principal Component Analysis**
Type	Unsuperised Machine Learning
Use	Represent large number of columns as fewer set of columns (Dimension Reduction)
Cost Function	SVD (Single Value Decomposition)
Goodness of Fit	explained_variance_ratio_
Hyperparameters	n_components=3, svd_solver='auto'
Function in R	PrincipalComponents=**prcomp**(DataForPCA,scale=T)
Function in Python	from sklearn.decomposition import **PCA** pca = **PCA**(n_components=5) FinalPrincipalComponents = pca.fit_transform(X)

Algorithm	t-SNE
Type	Unsuperised Machine Learning
Use	Represent large number of columns as fewer set of columns (Dimension Reduction)
Cost Function	Kullback-Leibler divergence(KL-Divergence)
Goodness of Fit	Shepard diagram correlation
Hyperparameters	n_components=2, perplexity=30.0, n_iter=1000

Function in R	library(**Rtsne**)
	# Changing the data in matrix format
	DataMatrix=as.matrix(DataForTSNE)
	
	# Reducing data using TSNE into 2-dimensions
	TSNE_Output=**Rtsne**(DataMatrix, dims = 2, perplexity=2)

Function in Python	from sklearn.manifold import **TSNE**
	# Reducing the original data into 2 dimensions
	tsne = **TSNE**(n_components=2)
	ComponentValues=tsne.fit_transform(X)

T-SNE stands for “t-distributed Stochastic Neighbor Embedding”. Since plotting a high dimension data (e.g 100 columns) is computationally intense as well as difficult to understand. If the same data can be represented in 2 or 3 dimensions, it can be plotted easily and the data patterns can be understood.

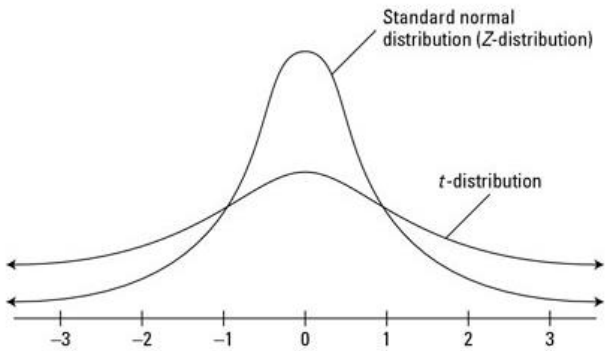
This can also be done using PCA, but, it is a linear method. If the relationship between the features is non-linear, then PCA will fail to provide good components that are efficient in explaining the variance. This is where t-SNA comes into the picture.

- t-SNE can find the non-linear relations between features and represent them in a lower dimension.
- t-SNE works by finding a probability distribution in low dimensional space “similar” to the distribution of original high dimensional data.
- t-SNE can find the non-linear relations between features and represent them in a lower dimension.

The working steps of t-SNE can be summarized as below

1. Find the euclidean distance of every point from every other point in the data in high dimension
2. Calculate the probability of two points being close to each other in a high dimension. If the distance is low then the probability is high.
3. Find the Euclidean distance of every point from every other point in the data in low dimension
4. Calculate the probability of two points being close to each other in low dimension. If the distance is low then the probability is high.
5. Minimize the difference in probabilities of high dimension and low dimension.
6. The “t” in t-SNE is t-distribution. The reason for using a t-distribution instead of the normal distribution is the long tails of t-distribution, which allows the representation of long distances in a better way.

t-distribution has long tails as compared to the normal distribution



SNE is Stochastic Neighbor Embedding, this concept tries to keep the clusters of data points in higher dimension intact when these data points are encoded in the lower dimension.

Basically if data row numbers 1, 10 and 13 are “similar” and close to each other (less euclidean distance) in the high dimension, then they will be “similar” and close to each other in the low dimension as well. Hence, retaining the non-linear properties of data.

Importance and limitation of Principal Component Analysis?

Following are the advantages of PCA

1. Removes Correlated Features – To visualize our all features in data, we must reduce the same in data, to do that we need to find out the correlation among the features (correlated variables). Finding correlation manually in thousands of features is nearly impossible, frustrating and time-consuming. PCA does this for you efficiently.
2. Improve algorithm performance – With so many features, the performance of your algorithm will drastically degrade. PCA is a very common way to speed up your Machine Learning algorithm by getting rid of correlated variables which don’t contribute in any decision making.
3. Improve Visualization – It is very hard to visualize and understand the data in high dimensions. PCA transforms a high dimensional data to low dimensional data (2 dimension) so that it can be visualized easily.

Following are the limitation of PCA

1. Independent variable become less interpretable – After implementing PCA on the dataset, your original features will turn into Principal Components. Principal Components are the linear combination of your original features. Principal Components are not as readable and interpretable as original features.
2. Data standardization is must before PCA – You must standardize your data before implementing PCA, otherwise PCA will not be able to find the optimal Principal Components.
3. Information loss – Although Principal Components try to cover maximum variance among the features in a dataset, if we don’t select the number of Principal Components with care, it may miss some information as compared to the original list of features.

What is t-SNE and How to apply t-SNE ?

t-Distributed Stochastic Neighbor Embedding (t-SNE) is an unsupervised, non-linear technique primarily used for data exploration and visualizing high-dimensional data. In simpler terms, t-SNE gives you a feel or intuition of how the data is arranged in a high-dimensional space.

Let’s understand each and every term in details.

1. Scholastic – Not definite but random probability
2. Neighborhood – Concerned only about retaining the structure of neighborhood points.
3. Embedding – It means picking up a point from high dimensional space and placing it into lower dimension

How to interpret t-SNE output?

There are 3 parameters

a) Steps: number of iterations.
b) Perplexity: can be thought of as the number of neighboring points.
c) Epsilon: It is for data visualization and determines the speed which it should be changed

What is the difference between PCA and t-SNE?

t-SNE in comparison to PCA:

a) When the data is huge (in size), t-SNE may fail to produce better results.
b) t-SNE is nonlinear whereas PCA is linear.
c) PCA will preserve things that t-SNE will not.
d) PCA is deterministic; t-SNE is not
e) t-SNE does not scale well with the size of the dataset, while PCA does.

Should one remove highly correlated variables before doing PCA?

No, PCA loads out all highly correlated variables on the same Principal Component(Eigenvector), not different ones.

Is it important to standardize before applying PCA?

PCA finds new directions based on the covariance matrix of original variables. Since the covariance matrix is sensitive to the standardization of variables. Usually, we do standardization to assign equal weights to all the variables. If we use features of different scales, we get misleading directions. But, it is not necessary to standardize the variables, if all the variables are on the same scale.

What is the difference between Normalization and Standardization (both are part of feature scaling)?

Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling.

Here's the formula for normalization:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Here, Xmax and Xmin are the maximum and the minimum values of the feature respectively.

- When the value of X is the minimum value in the column, the numerator will be 0, and hence X' is 0
- On the other hand, when the value of X is the maximum value in the column, the numerator is equal to the denominator and thus the value of X' is 1
- If the value of X is between the minimum and the maximum value, then the value of X' is between 0 and 1

Standardization is another scaling technique where the values are centered around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation.

Here's the formula for standardization:

$$X' = \frac{X - \mu}{\sigma}$$

μ is the mean of the feature values and

σ is the standard deviation of the feature values. Note that in this case, the values are not restricted to a particular range.

When to Normalize and Standardize?

a) Normalization is good to use when you know that the distribution of your data does not follow a Gaussian distribution. This can be useful in algorithms that do not assume any distribution of the data like K-Nearest Neighbors and Neural Networks.

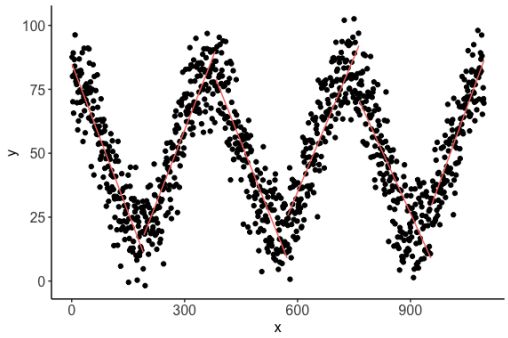
b) Standardization, on the other hand, can be helpful in cases where the data follows a Gaussian distribution. However, this does not have to be necessarily true. Also, unlike normalization, standardization does not have a bounding range. So, even if you have outliers in your data, they will not be affected by standardization.

Is rotation necessary in PCA? If yes, Why? What will happen if you don't rotate the components?

Yes, rotation (orthogonal) is necessary to account the maximum variance of the training set. If we don't rotate the components, the effect of PCA will diminish and we'll have to select more number of components to explain variance in the training set.

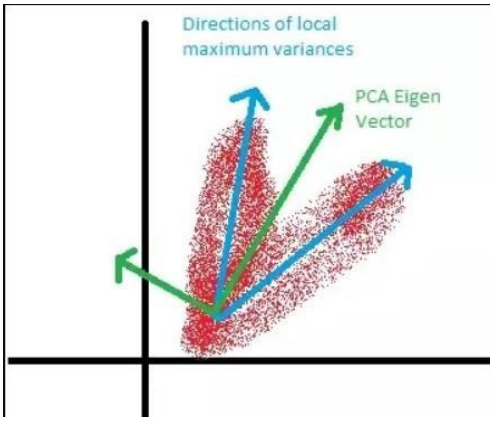
Limitations of PCA?

1. Doesn't work well for non linearly correlated data.



Here the data points are not linearly correlated. PCA won't work well.

2. PCA always finds orthogonal principal components. Sometimes, our data demands non-orthogonal principal components to represent the data.



The green colour vectors are principal components. But, the actual maximum variance directions are blue colour vectors. Standard PCA fails to find that vectors.

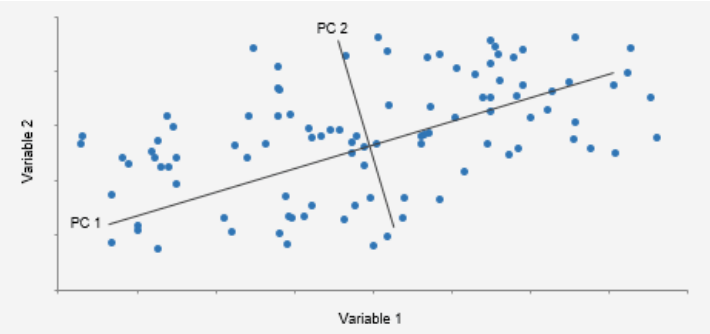
3. PCA always considered the low variance components in the data as noise and recommend us to throw away those components. But, sometimes those components play a major role in a supervised learning task.
4. If the variables are correlated, PCA can achieve dimension reduction. If not, PCA just orders them according to their variances

Can PCA be used to reduce the dimensionality of a highly nonlinear dataset?

PCA can be used to significantly reduce the dimensionality of most datasets, even if they are highly nonlinear because it can at least get rid of useless dimensions. However, if there are no useless dimensions, reducing dimensionality with PCA will lose too much information.

What does a PCA do? How is the first principal component axis selected?

1. PCA stands for principal component analysis. It is a dimensionality reduction technique which summarizes a large set of correlated variables (basically a high dimensional data) into a smaller number of representative variables, called the principal components, that explains most of the variability in the original set.
2. The first principal component axis is selected in a way such that it explains most of the variation in the data and is closest to all n observations.



What does a principal component in a PCA represent?

1. It represents a line or an axis along which the data varies the most and it also is the line that is closest to all n observations. OR
2. It is the linear combination of observed variables that results in an axis or a set of axes, that explain/s most of the variability in the dataset.
3. Mathematically speaking, it is the eigenvector of the first principal component. The sum of the squared distances is the eigenvalue for PC1 and the square root of the eigenvalue is the singular value for the PC1

Can we use PCA for regression? When is it advisable to use PCA for regression?

Yes, we can use principal components for regression setup. PCA would perform well in cases when the first few principal components are sufficient to capture most of the variation in the predictors as well as the relationship with the response. The only drawback to this approach is that the new reduced set of features would be modeled ignoring the response variable Y when applying a PCA and while these features may do a good overall job of explaining the variation in X, the model will perform poorly, if these variables don't explain the variation in Y.

Can we use PCA for feature selection?

No! PCA is not a feature selection technique because if you think, any principal component axis is a linear combination of all the original set of feature variables which defines a new set of axes that explain most of the variability in the data. So while it performs well in many practical settings, it does not result in the development of a model that relies upon a small set of the original features.

What are some caveats of PCA?

1. Make sure data is on the same scale
2. Make sure data is centered
3. The maximum number of principal components equals the number of variables. There is a principal component or PC for each variable in the data set. However, if there are fewer samples than variables, the number of samples puts an upper bound on the number of PCs with eigenvalues greater than 0

Explain the Curse of Dimensionality?

The curse of dimensionality refers to all the problems that arise working with data in the higher dimensions. As the number of features increase, the number of samples increases, hence, the model becomes more complex. The more the number of features, the more the chances of overfitting. A machine learning model that is trained on a large number of features,

gets increasingly dependent on the data it was trained on and in turn overfitted, resulting in poor performance on real data, beating the purpose. The fewer features our training data has, the lesser assumptions our model makes and the simpler it will be.

Why do we need dimensionality reduction? What are its drawbacks?

Advantages of Dimensionality Reduction:

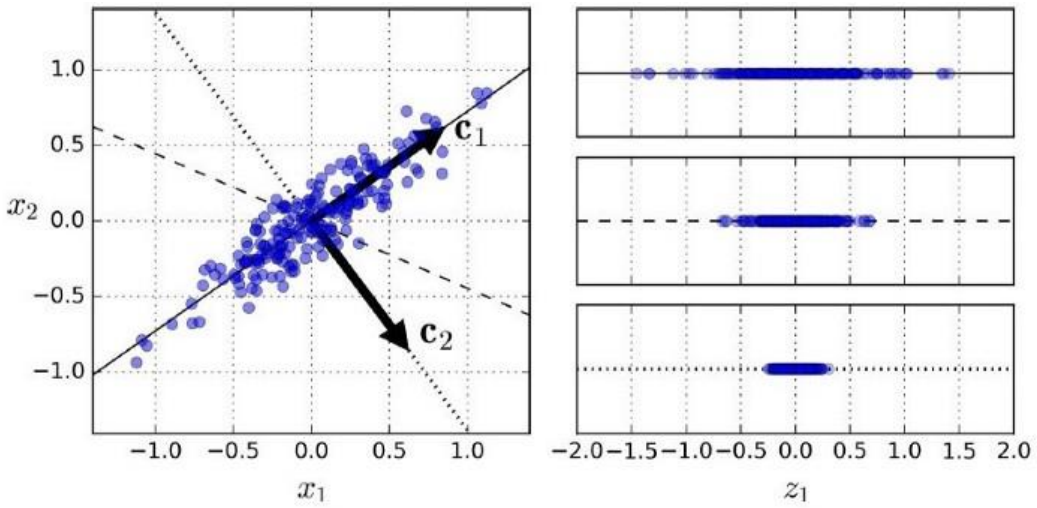
1. Less misleading data means model accuracy improves.
2. Fewer dimensions mean less computing. Less data means that algorithms train faster.
3. Less data means less storage space required.
4. Removes redundant features and noise.
5. Dimensionality Reduction helps us visualize the data on 2D plots or 3D plots.

Drawbacks of Dimensionality Reduction are:

1. Some information is lost, possibly degrading the performance of subsequent training algorithms.
2. It can be computationally intensive.
3. Transformed features are often hard to interpret.
4. It makes the independent variables less interpretable.

Explain Principal Component Analysis, assumptions, equations.

Principal Component Analysis (PCA) is an unsupervised dimensionality reduction algorithm. It identifies the hyperplane that lies closest to the data, and then it projects the data onto it preserving the variance.



Here is a result of the projection of a dataset onto each of these axes. As you can see, the projection onto the solid line preserves the maximum variance, while the projection onto the dotted line preserves very little variance, and the projection onto the dashed line preserves an intermediate amount of variance.

PCA selects the axis which preserves maximum variance in the training set. PCA finds as many axes as the number of dimensions such that every axis is orthogonal to each other.

Equations:

1. First, we calculate the covariance matrix

$$\begin{bmatrix} V_a & C_{a,b} & C_{a,c} & C_{a,d} & C_{a,e} \\ C_{a,b} & V_b & C_{b,c} & C_{b,d} & C_{b,e} \\ C_{a,c} & C_{b,c} & V_c & C_{c,d} & C_{c,e} \\ C_{a,d} & C_{b,d} & C_{c,d} & V_d & C_{d,e} \\ C_{a,e} & C_{b,e} & C_{c,e} & C_{d,e} & V_e \end{bmatrix}$$

2. Decompose the covariance matrix and find the eigenvalues and corresponding eigenvectors

$$A\,x = \lambda\,x \quad \text{——} \quad \lambda \text{ \& } x \text{ is the eigenvalue \& eigenvector of } x$$

Eigen Value and Eigen Vector

$$[Covariance\ \ matrix] \cdot [Eigenvector] = [eigenvalue] \cdot [Eigenvector]$$

Decompose the Covariance Matrix

3. We arrange the eigenvalues as follows

$$\textbf{If } \lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \ldots\ldots \geq \lambda_d$$

$$\mathbf{u_1} \textbf{ is the corresponding maximum variance direction vector of } \lambda_1$$

$$\textbf{Similarly for } \mathbf{u_2, u_3 \ldots u_d} \textbf{ then } \mathbf{u_i \perp u_j} \mapsto \mathbf{u_i^T u_j = 0}$$

Each Eigenvalue corresponds to a principal component. The highest eigenvalue corresponds to the 1st Principal Component and so on.

Assumptions:

1. There needs to be a linear relationship between all variables. The reason for this assumption is that a PCA is based on Pearson correlation coefficients, and as such, there needs to be a linear relationship between the variables
2. You should have sampling adequacy, which simply means that for PCA to produce a reliable result, large enough sample sizes are required.
3. Your data should be suitable for data reduction. Effectively, you need to have adequate correlations between the variables to be reduced to a smaller number of components.
4. There should be no significant outliers.

Can PCA be used to reduce the dimensionality of a highly nonlinear dataset?

PCA can be used to significantly reduce the dimensionality of most datasets, even if they are highly nonlinear because it can at least get rid of useless dimensions. However, if there are no useless dimensions, reducing dimensionality with PCA will lose too much information.