INT 303 BIG DATA ANALYTICS

Lecture9: Dimensionality Reduction

Jia WANG

Jia.wang02@xjtlu.edu.cn

Xi'an Jiaotong-Liverpool University
西交利物浦大學

OUTLINE

- High Dimensionality
- Principal Components Analysis
- Non-Negative Matrix Factorization (NMF)

WHAT IS 'BIG DATA'?

In the world of Data Science, the term *Big Data* gets thrown around a lot. What does *Big Data* mean?

A rectangular data set has two dimensions: number of observations
(n) and the number of predictors (p). Both can play a part in defining a problem as a *Big Data* problem.

What are some issues when:

- n is big (and p is small to moderate)?
- p is big (and n is small to moderate)?
- n and p are both big?

KEEP IN MIND, BIG *N* DOESN'T SOLVE EVERYTHING

The era of Big Data (aka, large *n*) can help us answer lots of interesting scientific and application-based questions, but it does not fix everything.

Remember the old adage: "**crap in = crap out**". That is to say, if the data are not representative of the population, then modeling results can be terrible.

Xiao-Li Meng does a wonderful job describing the subtleties involved (WARNING: it's a little technical, but digestible): https://www.youtube.com/watch?v=8YLdIDOMEZs

WHEN *N* IS BIG

When the sample size is large, this is typically not much of an issue from the statistical perspective, just one from the computational perspective.

- Algorithms can take forever to finish.
  - It may take some time to estimate the coefficients of regression models, especially models without closed form (such as LASSO).
  - Wait until we get to Neural Nets!

What can we do to fix this computational issue?

- Perform 'preliminary' steps (model selection, tuning, etc.) on a **subset of the training data set. 10% or less can be justified**

NYC TAXI VS. UBER

NYC TAXI VS. UBER

We'd like to compare Taxi and Uber rides in NYC (for example, how much the fare costs based on length of trip, time of day, location, etc.).

A public dataset has 1.9 million Taxi and Uber trips. Each trip is described by p = 29 useable predictors (and 1 ground truth variable).

In [11]:
```
print(nyc_cab_df.shape)
nyc_cab_df.head()
```

```
(1873671, 30)
```

Out[11]:

	AWND	Base	Day	Dropoff_latitude	Dropoff_longitude	Ehail_fee	Extra	Fare_amount	Lpep_dropoff_datetime	MTA_tax	...	TMIN	Tip_amount	Tolls_amou
0	4.7	B02512	1	NaN	NaN	NaN	NaN	33.863498	2014-04-01 00:24:00	NaN	...	39	NaN	Na
1	4.7	B02512	1	NaN	NaN	NaN	NaN	19.022892	2014-04-01 00:29:00	NaN	...	39	NaN	Na
2	4.7	B02512	1	NaN	NaN	NaN	NaN	25.498981	2014-04-01 00:34:00	NaN	...	39	NaN	Na
3	4.7	B02512	1	NaN	NaN	NaN	NaN	28.024628	2014-04-01 00:39:00	NaN	...	39	NaN	Na
4	4.7	B02512	1	NaN	NaN	NaN	NaN	12.083589	2014-04-01 00:40:00	NaN	...	39	NaN	Na

5 rows × 30 columns

WHEN *P* IS BIG

When the number of predictors is large (in any form: interactions, polynomial terms, etc.), then lots of issues can occur.

- Matrices may not be invertible (issue in LR)
- **Multicollinearity** is likely to be present

Multicollinearity：

occurrence of high intercorrelations among two or more independent variables in a multiple regression mode

INTERACTION TERMS

Recall that an interaction term between predictors X_1 and X_2 can be incorporated into a regression model by including the multiplicative (i.e. cross) term in the model, for example

$$Y = \beta_0 + \beta_1 \mathrm{X}_1 + \beta_2 \mathrm{X}_2 + \beta_3 (\mathrm{X}_1 + \mathrm{X}_2) + \mathrm{s}$$

Suppose X_1 is a binary predictor indicating whether a NYC ride pickup is a tax or an Uber, X_2 is the times of day of the pickup and Y is the length of the ride.

What is the interpretation of β_3?

HOW MANY INTERACTION TERMS?

This NYC taxi and Uber dataset has 1.9 million Taxi and Uber trips. Each trip is described by p = 29 useable predictors (and 1 response variable). How many interaction terms are there?

- Two-way interactions: $\binom{p}{2} = 253$
- Three-way interactions: $\binom{p}{3} = 1771$
- Etc.

The total number of all possible interaction terms (including main effects) is.

$$\sum_{k\,1\,2}^{p} \binom{p}{k} = 2^p \approx 8.3\text{million}$$

What are some problems with building a model that includes all possible interaction terms?

HOW MANY INTERACTION TERMS?

In order to wrangle a data set with over 1 billion observations, we could use random samples of 100k observations from the dataset to build our models. If we include all possible interaction terms, our model will have 8.3 mil parameters.

We will not be able to uniquely determine 8.3 mil parameters with only 100k observations.

In this case, we call the model ***unidentifiable***.

WHEN *P* IS BIG

When the number of predictors is large (in any form: interactions, polynomial terms, etc.), then lots of issues can occur.

- Matrices may not be invertible (issue in LR).
- Multicollinearity is likely to be present
- Models are susceptible to overfitting

This situation is called *High Dimensionality*, and needs to be accounted for when performing data analysis and modeling.

IN PRACTICE, WE CAN:

- increase the number of observation
- consider only scientifically important interaction terms
- perform variable selection
- perform another ***dimensionality reduction*** technique like PCA

DIMENSIONALITY REDUCTION

- Generate a low-dimensional encoding of a high-dimensional space.

- Purposes:
  - Data compression/ visualization
  - Robustness to noise and uncertainty
  - Potentially easier to interpret

A FRAMEWORK FOR DIMENSIONALITY REDUCTION (CONT.)

A method of dimensionality reduction includes 2 steps:

- Determine a optimal set of new predictors $Z_1,\ldots, Z_m$, for $m < p$.
- Express each observation in the data in terms of these new predictors.

The transformed data will have m columns rather than p.

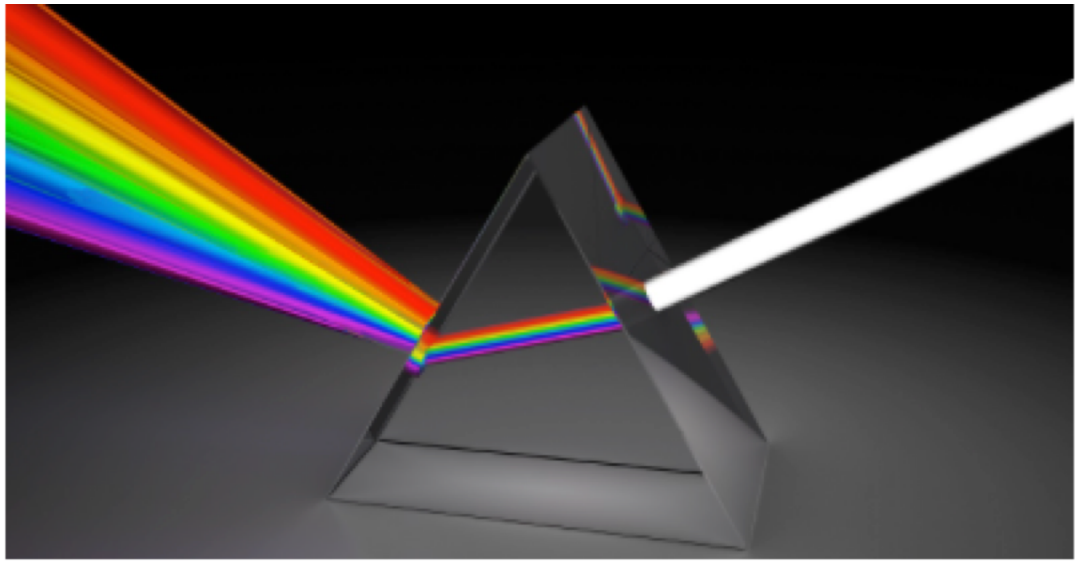
Thereafter, we can fit a model using the new predictors.

The method for determining the set of new variables (what do we mean by an optimal predictors set) can differ according to application. We will explore a way to create new data representation that captures the variations in the observed data.

PRINCIPAL COMPONENTS ANALYSIS (PCA)

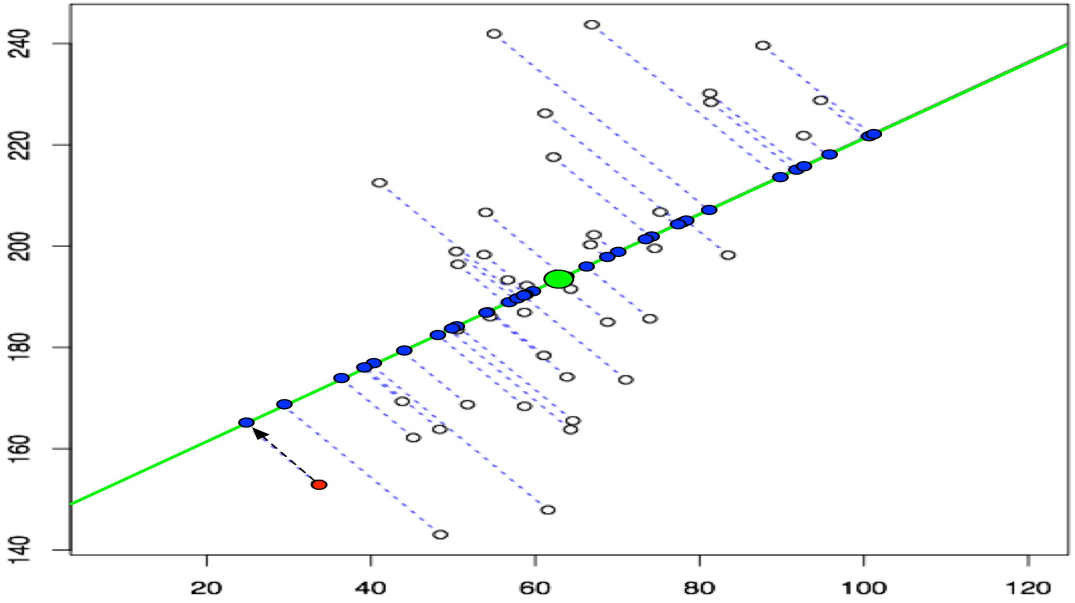
PRINCIPAL COMPONENTS ANALYSIS (PCA)

Principal components analysis (PCA) is a statistical technique that allows identifying underlying linear patterns in a data set so it can be expressed in terms of another data set of a significative lower dimension without much loss of information.



THE INTUITION BEHIND PCA (CONT.)

Transforming our observed data means projecting our dataset onto the space defined by the top *m* PCA components, these components are our new predictors.



IMPLEMENTATION OF PCA USING LINEAR ALGEBRA

To implement PCA yourself using this linear algebra result, you can perform the following steps:

- Take the whole dataset consisting of d+1 dimensions and ignore the labels such that our new dataset becomes d dimensional.
- Compute the mean for every dimension of the whole dataset.
- Compute the covariance matrix of the whole dataset.
- Compute eigenvectors and the corresponding eigenvalues.
- Sort the eigenvectors by decreasing eigenvalues and choose k eigenvectors with the largest eigenvalues to form a d $\times$ k dimensional matrix W.
- Use this d $\times$ k eigenvector matrix to transform the samples onto the new subspace.

However, PCA is easy to perform in Python using the `decomposition.PCA` function in the `sklearn` package.

LET OUR DATA MATRIX X BE THE SCORE OF THREE STUDENTS :

Student	Math	English	Art
1	90	60	90
2	90	90	30
3	60	60	60
4	60	60	90
5	30	30	30

COMPUTE THE MEAN OF EVERY DIMENSION OF THE WHOLE DATASET

$$\mathbf{A} = \begin{bmatrix} 90 & 60 & 90 \\ 90 & 90 & 30 \\ 60 & 60 & 60 \\ 60 & 60 & 90 \\ 30 & 30 & 30 \end{bmatrix}$$

Matrix A

So, The mean of matrix **A** would be

$$\bar{\mathbf{A}} = [\ 66 \quad 60 \quad 60\ ]$$

Mean of Matrix A

COMPUTE THE *COVARIANCE MATRIX*

$$cov(X,Y) = \frac{1}{n-1}\sum_{i=1}^{n}(Xi - \bar{x})(Yi - \bar{y})$$

	Math	English	Arts
1	90	60	90
2	90	90	30
3	60	60	60
4	60	60	90
5	30	30	30

Matrix A

	Math	English	Art
Math	504	360	180
English	360	360	0
Art	180	0	720

Covariance Matrix of A

COMPUTE EIGENVECTORS AND CORRESPONDING EIGENVALUES

> *Let* $\mathbf{A}$ *be a square matrix,* $\mathbf{v}$ *a vector and* λ *a scalar that satisfies* $\mathbf{A}v = \lambda v$*, then* λ *is called eigenvalue associated with eigenvector* $\mathbf{v}$ *of* $\mathbf{A}$*.*

$$\det(A-\lambda I) = 0$$

$$\det\left(\begin{pmatrix} 504 & 360 & 180 \\ 360 & 360 & 0 \\ 180 & 0 & 720 \end{pmatrix} - \lambda \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}\right) \qquad \det\begin{pmatrix} 504-\lambda & 360 & 180 \\ 360 & 360-\lambda & 0 \\ 180 & 0 & 720-\lambda \end{pmatrix}$$

$$-\lambda^3 + 1584\lambda^2 - 641520\lambda + 25660800 = 0$$

$$\lambda \approx 44.81966\ldots,\ \lambda \approx 629.11039\ldots,\ \lambda \approx 910.06995\ldots$$

Eigenvalues

EIGENVECTORS

eigenvectors

$$\begin{pmatrix} -3.75100\ldots \\ 4.28441\ldots \\ 1 \end{pmatrix}, \begin{pmatrix} -0.50494\ldots \\ -0.67548\ldots \\ 1 \end{pmatrix}, \begin{pmatrix} 1.05594\ldots \\ 0.69108\ldots \\ 1 \end{pmatrix}$$

$$\lambda \approx 44.81966\ldots, \lambda \approx 629.11039\ldots, \lambda \approx 910.06995\ldots$$

Eigenvalues

SORT THE EIGENVECTORS BY DECREASING EIGENVALUES

So, after sorting the eigenvalues in decreasing order, we have

$$\begin{pmatrix} 910.06995 \\ 629.11039 \\ 44.81966 \end{pmatrix}$$

So, *eigenvectors* corresponding to two maximum eigenvalues are :

$$\mathbf{W} = \begin{bmatrix} 1.05594 & -0.50494 \\ 0.69108 & -0.67548 \\ 1 & 1 \end{bmatrix}$$

PRACTICAL NOTES

PCA

- Implemented within `sklearn.decomposition.PCA`
  - `PCA.fit_transform(X)` fits the model to `X`, then provides the data in principal component space
  - `PCA.components_` provides the "loadings matrix", or directions of maximum variance
  - `PCA.explained_variance_` provides the amount of variance explained by each component

PCA

```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.decomposition import PCA

iris = datasets.load_iris()

X = iris.data
y = iris.target
target_names = iris.target_names

pca = PCA(n_components=2)
X_r = pca.fit(X).transform(X)

# Print PC1 loadings
print(pca.components_[:, 0])
# ...
```

PCA

```
# ...
pca = PCA(n_components=2)
X_r = pca.fit(X).transform(X)

# Print PC1 loadings
print(pca.components_[:, 0])

# Print PC1 scores
print(X_r[:, 0])

# Percentage of variance explained for each component
print(pca.explained_variance_ratio_)
# [ 0.92461621  0.05301557]
```

Practical Notes

In [24]:
```
pca_all = PCA()
pca_all.fit(X_non_test)

print('First 4 principal components:\n', pca_all.components_[0:4])
print('Explained variance ratio:\n', pca_all.explained_variance_ratio_)
```

```
First 4 principal components:
 [[  2.39129056e-02  -1.52589992e-03   2.87820181e-03   8.15273795e-01
    5.30856898e-01  -1.66247307e-01  -1.57138583e-01   1.63510006e-02
   -9.18124937e-03  -0.00000000e+00   0.00000000e+00  -1.59625033e-02]
 [ -4.81556857e-02  -1.64875776e-03   8.71237863e-03  -1.95383366e-01
   -1.16363766e-01  -6.82620696e-01  -6.92730264e-01  -2.87939549e-03
    2.04933364e-03  -0.00000000e+00  -0.00000000e+00   5.33823735e-03]
 [  9.62900731e-01  -5.74858553e-03   2.52808711e-01  -2.66439844e-02
   -2.31060283e-02  -4.15207358e-02  -1.18366711e-02   6.63443885e-02
   -3.68951248e-02  -0.00000000e+00  -0.00000000e+00  -2.48879974e-03]
 [ -4.24750484e-03  -1.28064076e-03  -8.79072218e-03   5.43141125e-01
   -8.38916835e-01   2.59209473e-04  -1.26252332e-02   1.35518981e-02
   -7.80620455e-03  -0.00000000e+00  -0.00000000e+00  -2.67441606e-02]]
Explained variance ratio: [  3.52498964e-01   3.12127677e-01   2.40830333e-01   4.55437699e-02
   3.29216180e-02   7.43824785e-03   4.94773071e-03   2.95866750e-03
   7.23712321e-04   9.27988146e-06   0.00000000e+00   0.00000000e+00]
```

```
mglearn.plots.plot_pca_faces(X_train, X_test, image_shape)
```

Figure 3-11. Reconstructing three face images using increasing numbers of principal components

A FEW NOTES ON USING PCA

- PCA is an unsupervised algorithm. Meaning? It is done independent of the outcome variable.
- PCA is not so good because:
    1. Will not improve predictive ability of a model.
    2. Do it if interpretation is important
- PCA is great for:
    1. Reducing dimensionality in very high dimensional settings.
    2. Visualizing how predictive your features can be of your response.

NON-NEGATIVE MATRIX FACTORIZATION (NMF)

NON-NEGATIVE MATRIX FACTORIZATION

- Particularly helpful for data that is created as the addition (or overlay) of several independent sources:
  - Audio track of multiple people speaking.
  - Music with many instruments.
- In these situations, NMF can identify the original components that make up the combined data.
- NMF leads to more interpretable components than PCA.

MATRIX FACTORIZATION

Many (most?) dimensionality reduction methods involve matrix factorization

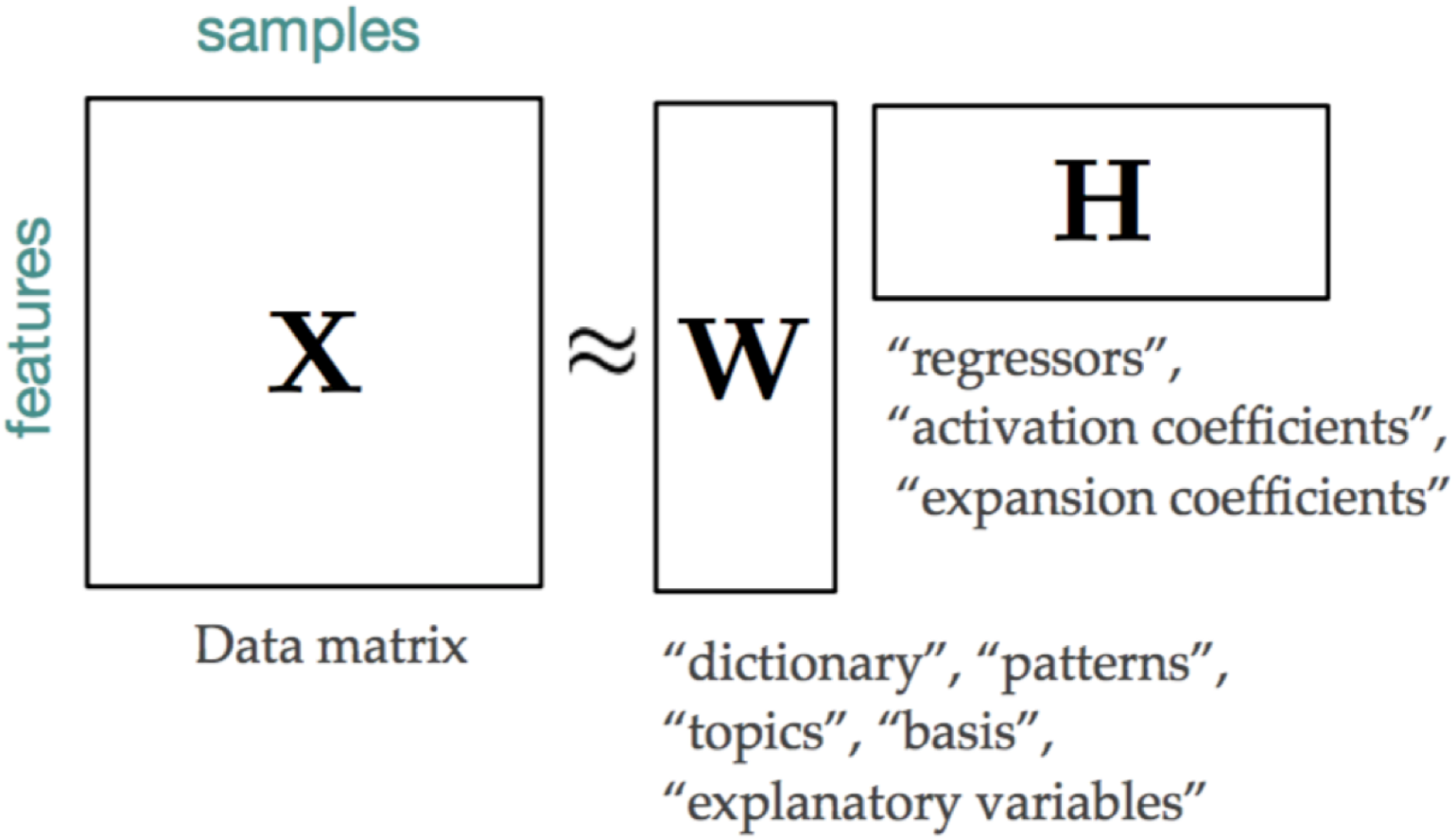
Basic Idea: Find two (or more) matrices whose product best approximate the original matrix

Low rank approximation to original $N \times M$ matrix:

$$\mathbf{X} \approx \mathbf{W}\mathbf{H}^T$$

where $\mathbf{W}$ is $N \times R$, $\mathbf{H}^T$ is $M \times R$, and $R \ll N$.

MATRIX FACTORIZATION



Generalization of many methods (e.g., SVD, QR, CUR, Truncated SVD, etc.)

MATRIX FACTORIZATION IS ALSO COMPRESSION

- The quality of the back-transformed data is similar to when using PCA, but slightly worse.
- This is expected, as PCA finds the optimum directions in terms of reconstruction.
- NMF is usually not used for its ability to reconstruct or encode data, but **rather for finding interesting patterns within the data**.

```
mglearn.plots.plot_nmf_faces(X_train, X_test, image_shape)
```

Figure 3-14. Reconstructing three face images using increasing numbers of components found by NMF

PRACTICAL NOTES - NMF

```
from sklearn.decomposition import NMF
nmf = NMF(n_components=15, random_state=0)
nmf.fit(X_train)
X_train_nmf = nmf.transform(X_train)
X_test_nmf = nmf.transform(X_test)
```

SUMMARY

PCA

- Preserves the covariation within a dataset
- Therefore mostly preserves axes of maximal variation
- Number of components can vary—in practice more than 2 or 3 rarely helpful

NMF

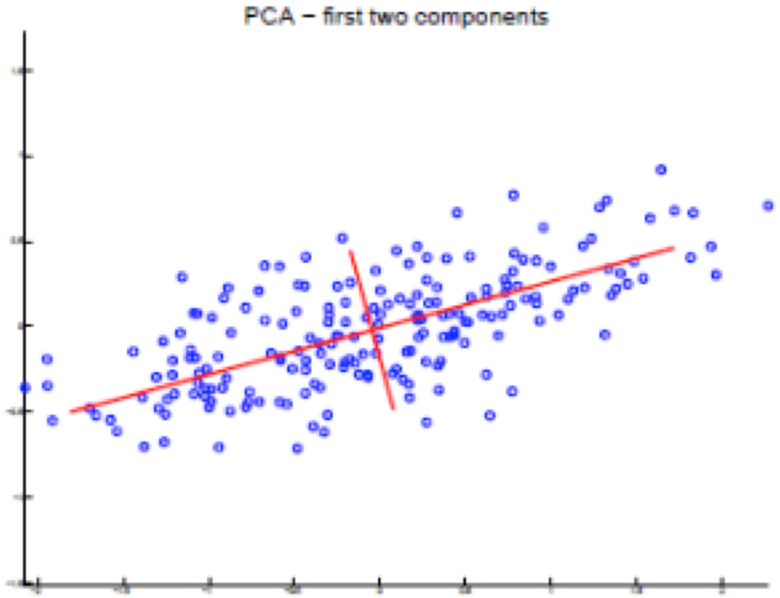
- Explains the dataset through factoring into two **non-negative** matrices
- Much more
- Excellent for separating out additive factors

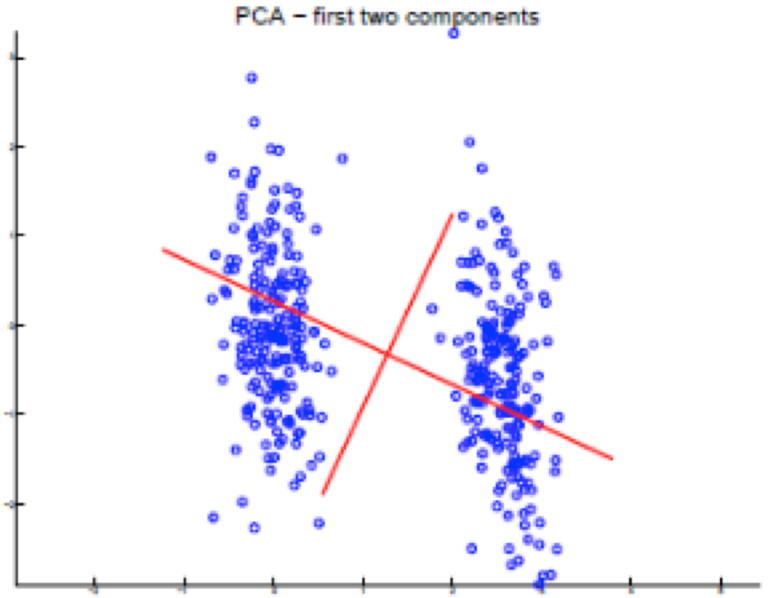
CLOSING

- **As always, selection of the appropriate method depends upon the question being asked.**

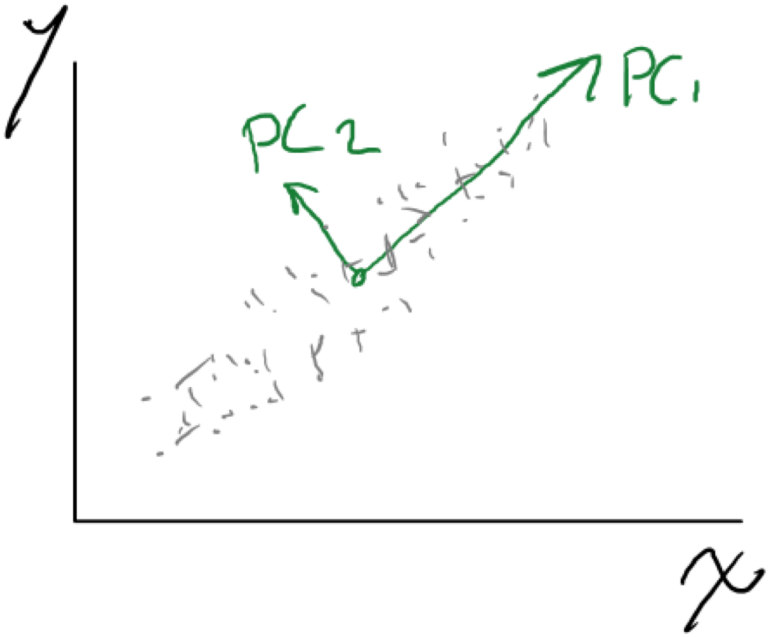
QUESTION

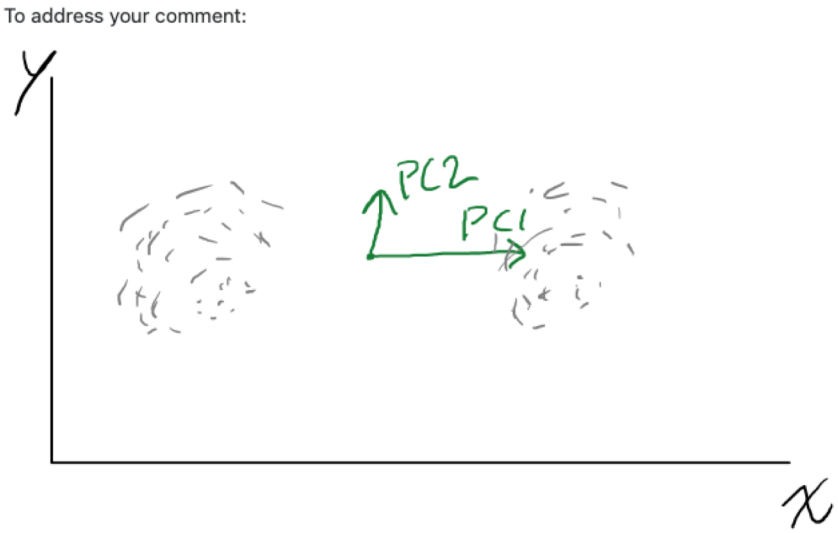
- Assume I have some data in 2D. How to draw the first and second principal component in PCA method?





- In the case of 2d data, the first principal component will be aligned along the direction of major continuity, and the second will be perpendicular to that, aligned along the least continuous direction. Generally just the eigen vectors at the mean x / mean y.





THANK YOU

VISIT US

WWW.XJTLU.EDU.CN

FOLLOW US

@XJTLU

Xi'an Jiaotong-Liverpool University
西交利物浦大学