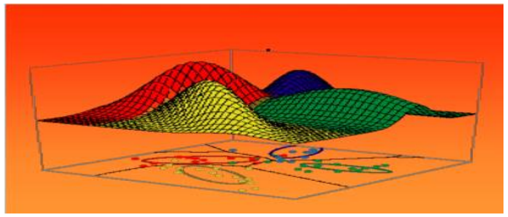


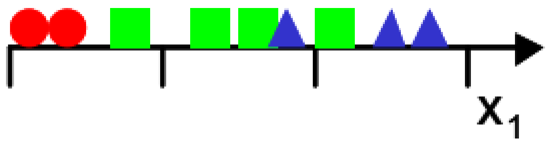
Pattern Recognition (CSE4213)

Faisal Muhammad Shah
Associate Professor,Dept Of CSE, AUST

- **Principal Components Analysis (PCA)**

The curse of dimensionality (1)

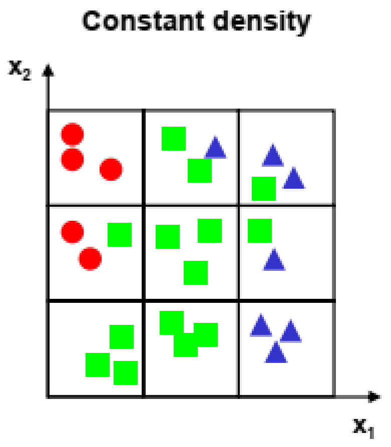
- **The curse of dimensionality**
  - A term coined by Bellman in 1961
  - Refers to the problems associated with multivariate data analysis as the dimensionality increases
  - We will illustrate these problems with a simple example
- **Consider a 3-class pattern recognition problem**
  - A simple approach would be to
    - Divide the feature space into uniform bins
    - Compute the ratio of examples for each class at each bin and,
    - For a new example, find its bin and choose the predominant class in that bin
  - In our toy problem we decide to start with one single feature and divide the real line into 3 segments

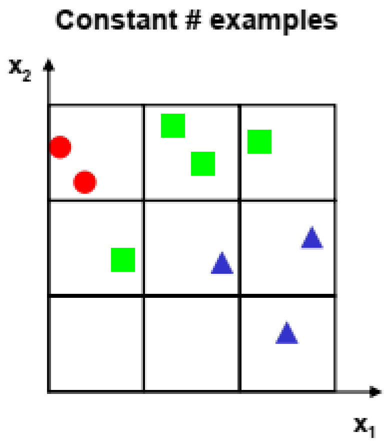


  - After doing this, we notice that there exists too much overlap among the classes, so we decide to incorporate a second feature to try and improve separability

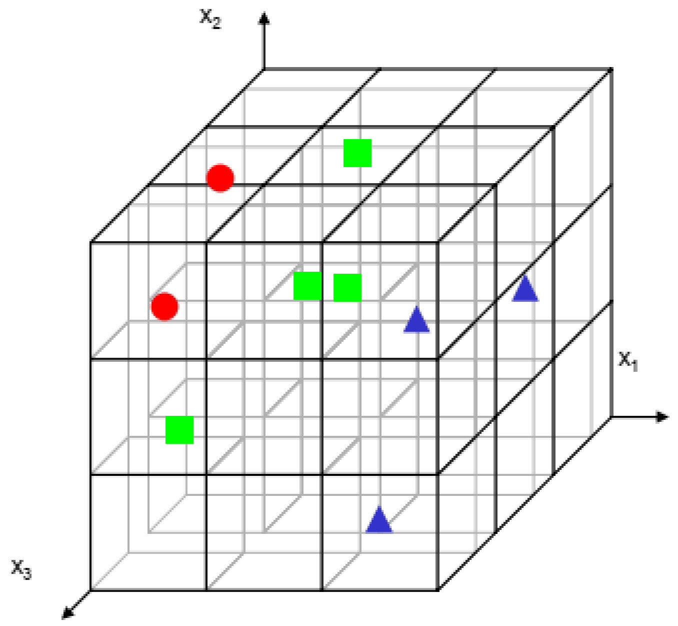
The curse of dimensionality (2)

- **We decide to preserve the granularity of each axis, which raises the number of bins from 3 (in 1D) to 3^2=9 (in 2D)**
  - At this point we need to make a decision: do we maintain the density of examples per bin or do we keep the number of examples had for the one-dimensional case?
    - Choosing to maintain the density increases the number of examples from 9 (in 1D) to 27 (in 2D)
    - Choosing to maintain the number of examples results in a 2D scatter plot that is very sparse



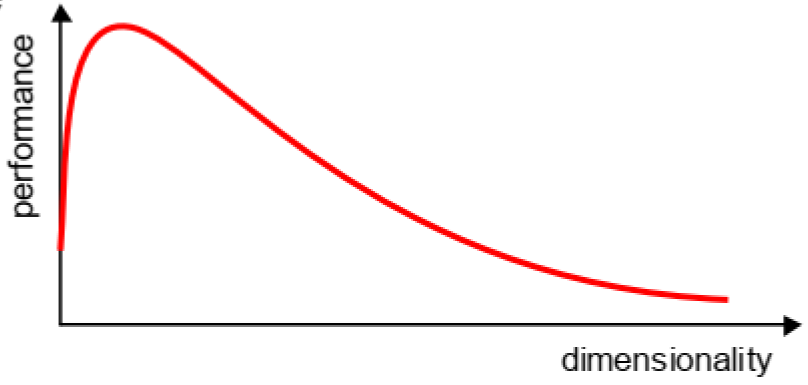


- **Moving to three features makes the problem worse:**
  - The number of bins grows to 3^3=27
  - For the same density of examples the number of needed examples becomes 81
  - For the same number of examples, well, the 3D scatter plot is almost empty

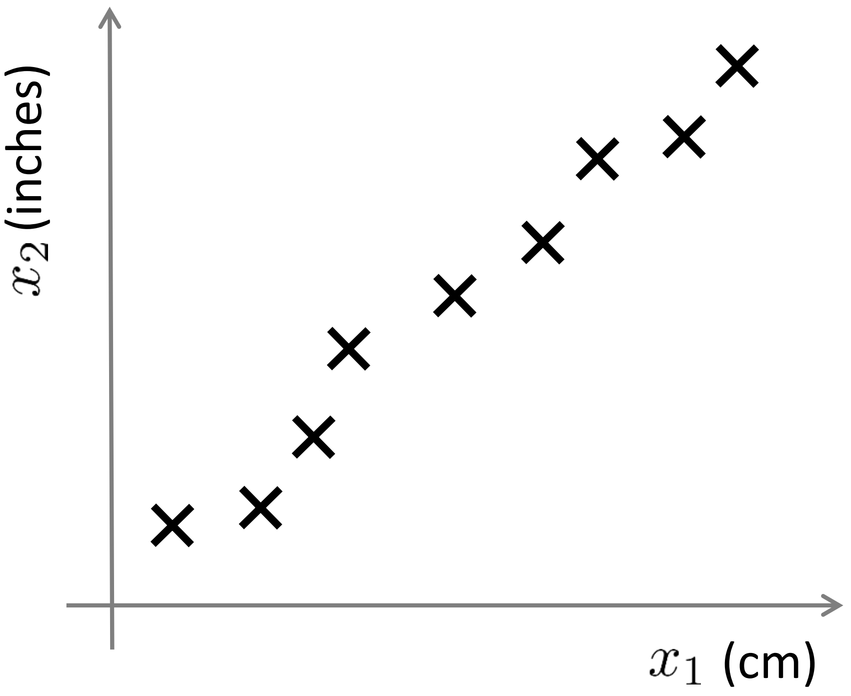


The curse of dimensionality (3)

- **Obviously, our approach to divide the sample space into equally spaced bins was quite inefficient**
  - There are other approaches that are much less susceptible to the curse of dimensionality, **but the problem still exists**
- **How do we beat the curse of dimensionality?**
  - By incorporating prior knowledge
  - By providing increasing smoothness of the target function
  - By reducing the dimensionality
- **In practice, the curse of dimensionality means that, for a given sample size, there is a maximum number of features above which the performance of our classifier will degrade rather than improve**
  - In most cases, the additional information that is lost by discarding some features is (more than) compensated by a more accurate mapping in the lower-dimensional space

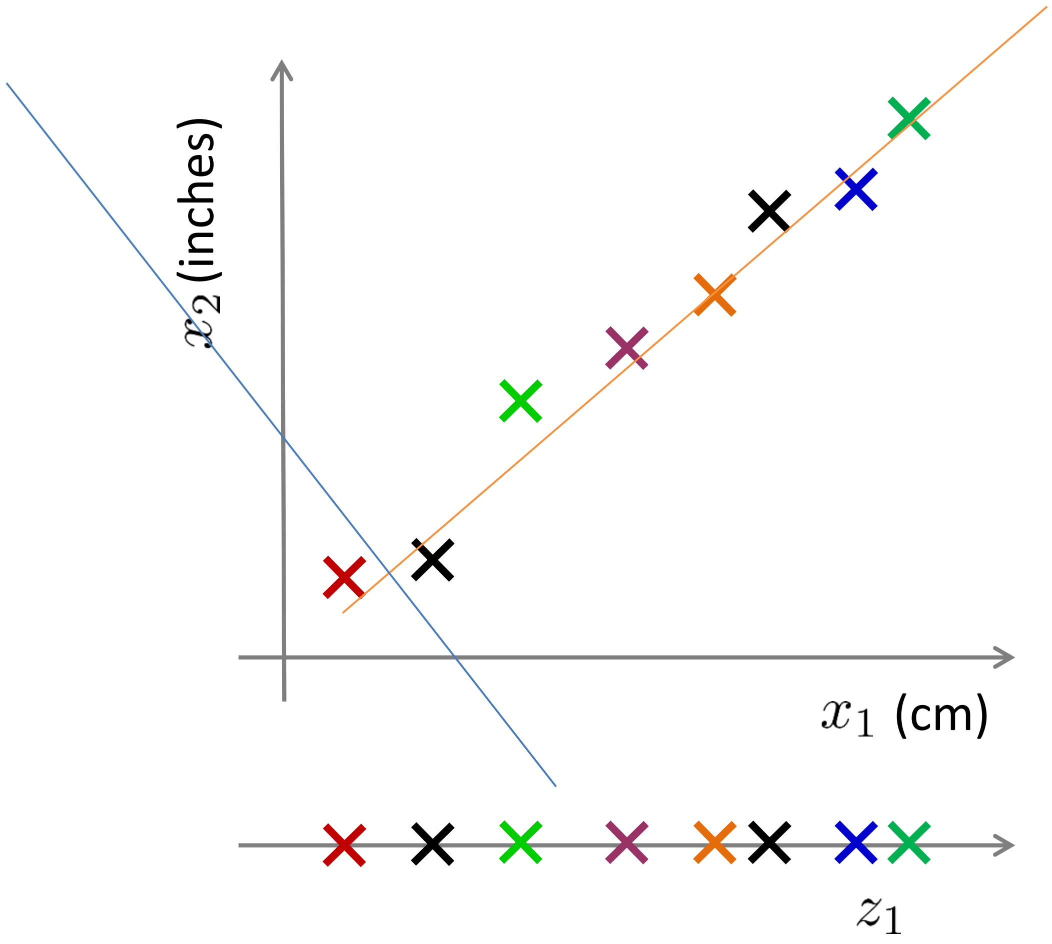


Data Compression



Reduce data from 2D to 1D

Data Compression



Reduce data from 2D to 1D

$x^{(1)} \rightarrow z^{(1)}$

$x^{(2)} \rightarrow z^{(2)}$

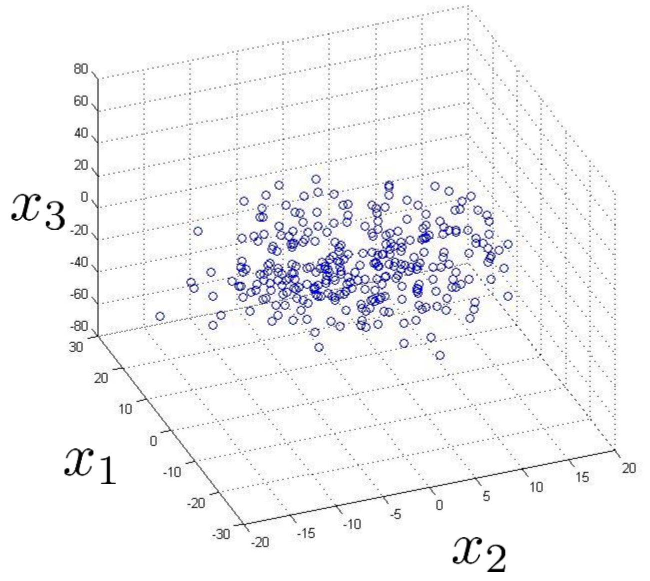
$\vdots$

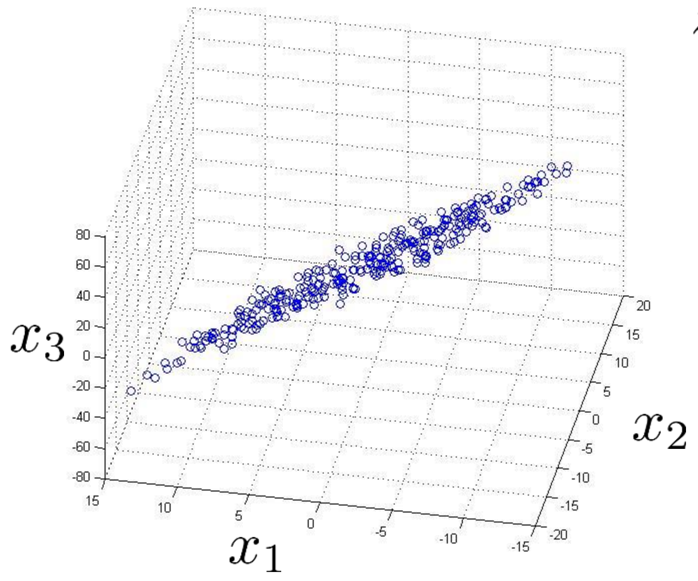
$x^{(m)} \rightarrow z^{(m)}$

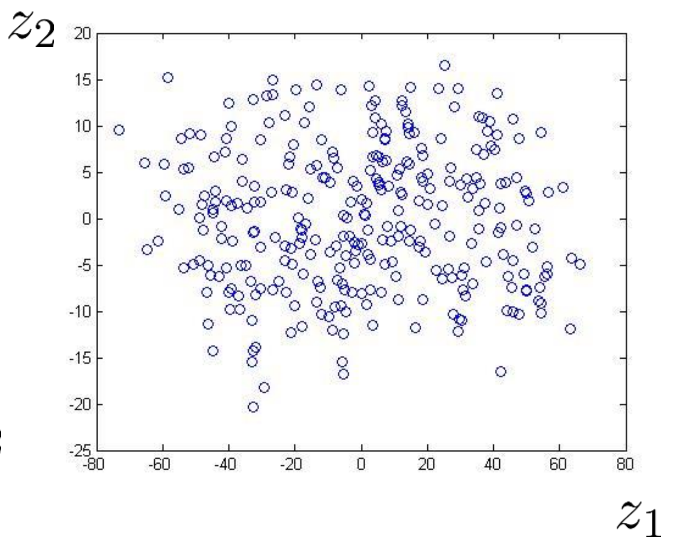
Andrew Ng

Data Compression

Reduce data from 3D to 2D







Andrew Ng

Principal Component Analysis (PCA) is a method of dimensionality reduction/feature extraction that transforms the data from a d-dimensional space into a new coordinate system of dimension p, where $p \leq d$ (the worst case would be to have $p = d$).

- The goal is to preserve as much of the variance in the original data as possible in the new coordinate systems.
- Give data on d variables, the hope is that the data points will lie mainly in a linear subspace of dimension lower than d.
- In practice, the data will usually not lie precisely in some lower dimensional subspace.
- The new variables that form a new coordinate system are called **principal components** (PCs).

- PCs are denoted by $\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_d$.
- The principal components form a basis for the data.
- Since PCs are orthogonal linear transformations of the original variables there is at most d PCs.
- Normally, not all of the d PCs are used but rather a subset of p PCs, $\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_p$
- In order to approximate the space spanned by the original data points $\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix}$ We can choose p based on what percentage of the variance of the original data we would like to maintain.

The first PC, $\mathbf{u}_1$ is called **first principal component** and has the maximum variance, thus it accounts for the most significant variance in the data.

The second PC, $\mathbf{u}_2$ is called **second principal component** and has the second highest variance and so on until PC $\mathbf{u}_d$ which has the minimum variance.

- In order to capture as much of the variability as possible, let us choose the first principal component, denoted by $\boldsymbol{u}_1$, to capture the maximum variance.
- Suppose that all centred observations are stacked into the columns of a $d \times n$ matrix X, where each column corresponds to a d-dimensional observation and there are n observations.
- The projection of n, d-dimensional observations on the first principal component $\boldsymbol{u}_1$ is $\boldsymbol{u}_1{}^T X$.

We want projection on this first dimension to have maximum variance.

$$var(\boldsymbol{u_1}^T X) = \boldsymbol{u_1}^T S \boldsymbol{u_1}$$

where S is the $d \times d$ sample covariance matrix of X.

- Clearly $var(\boldsymbol{u_1}^T X)$ can be made arbitrarily large by increasing the magnitude of $\boldsymbol{u_1}$.
- $var(\boldsymbol{u_1}^T X) = \boldsymbol{u_1}^T S \boldsymbol{u_1}$ where S is sample covariance matrix of sample data X.
- This means that the variance stated above has no upper limit and so we can not find the maximum.
- To solve this problem, we choose $\boldsymbol{u_1}$ to maximize $\boldsymbol{u_1}^T S \boldsymbol{u_1}$ while constraining $\boldsymbol{u_1}$ to have unit length.
- Therefore, we can rewrite the above optimization problem as:

$$\begin{aligned} \max \ \boldsymbol{u_1}^T S \boldsymbol{u_1} \\ subject\ to\ \boldsymbol{u_1}^T \boldsymbol{u_1} = 1 \end{aligned}$$

To solve this optimization problem a Lagrange multiplier λ is introduced:

$$L(u_1, \lambda) = {u_1}^T S u_1 - \lambda({u_1}^T \boldsymbol{u} - \mathbf{1})$$

Lagrange multipliers are used to find the maximum or minimum of a function $f(x, y)$ subject to constraint $g(x, y) = c$

Suppose we want to maximize the function $f(x, y) = x - y$ subject to the constraint $x^2 + y^2 = 1$.

We can apply the Lagrange multiplier method to find the maximum value for the function f ; the Lagrangian is:

$$L(x, y, \lambda) = x - y - \lambda(x^2 + y^2 - 1)$$

We want the partial derivatives equal to zero:

$$\frac{\partial L}{\partial x} = 1 + 2\lambda x = 0$$

$$\frac{\partial L}{\partial y} = -1 + 2\lambda y = 0$$

$$\frac{\partial L}{\partial \lambda} = x^2 + y^2 - 1$$

$$L(u_1, \lambda) = {u_1}^T S u_1 - \lambda({u_1}^T u_1 - 1) \tag{1}$$

Differentiating with respect to u_1 gives d equations,

$$S u_1 = \lambda u_1$$

Premultiplying both sides by ${u_1}^T$ we have:

$${u_1}^T S u_1 = \lambda {u_1}^T u_1 = \lambda$$

${u_1}^T S u_1$ is maximized if λ is the largest eigenvalue of S.

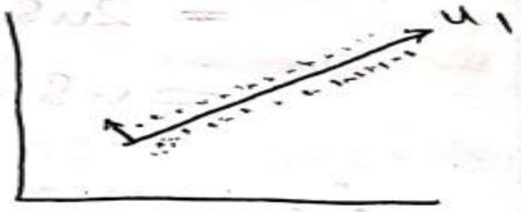
Clearly λ and $\boldsymbol{u_1}$ are an eigenvalue and an eigenvector of S. Differentiating (1) with respect to the Lagrange multiplier λ gives us back the constraint:

$$\boldsymbol{u_1}^T \boldsymbol{u_1} = 1$$

This shows that the first principal component is given by the eigenvector with the largest associated eigenvalue of the sample covariance matrix S. A similar argument can show that the p dominant eigenvectors of covariance matrix S determine the first p principal components.

⊛ Derive PCA ?

$x = [x_1, x_2, \dots x_n]_{d \times n}$



→ want to compute first PC_1 (u_1)

→ project the data on that direction that gives max^m variance.

→ u is an unknown vector.

→ project all of points on u.

→ $\underbrace{u^T}_{1\times d} \underbrace{x}_{d \times n}$ [data project in vector]

(1xn)

→ $var(u^T x) = \underbrace{u^T}_{1\times d} \underbrace{S}_{d\times d} \underbrace{u}_{d\times 1}$ S = Sample covariance matrix of x given formula.

(1x1)

→ $\max (u^T S u)$

→ $\max\limits_{u} (u^T S u)$

subject to : $u^T u = 1$ →

$$\boxed{\begin{array}{l} \text{Lagrange : } \max f(x,y) \\ \quad st\ g(x,y) = c \\ L(x,y,\lambda) = f(x,y) - \lambda g(x,y) \\ \nabla f(x,y) = \lambda \nabla g(x,y) \\ \text{So, from } \max(u^T S u) \\ \quad st : u^T u = 1 \end{array}}$$

→ $L(u, \lambda) = u^T S u - \lambda (u^T u - 1)$ [F Lagrange]

We know, $u^T u = u^2$

$$\frac{\partial L}{\partial u} = \frac{\partial}{\partial u}(u^2 S - \lambda u^2 - \lambda)$$

$= 2uS - 2u\lambda - 0$

$= 2uS - 2u\lambda = 0$

$= uS - \lambda u$

$\therefore \quad Su = \lambda u$ $\quad \boxed{Av = \lambda V}$

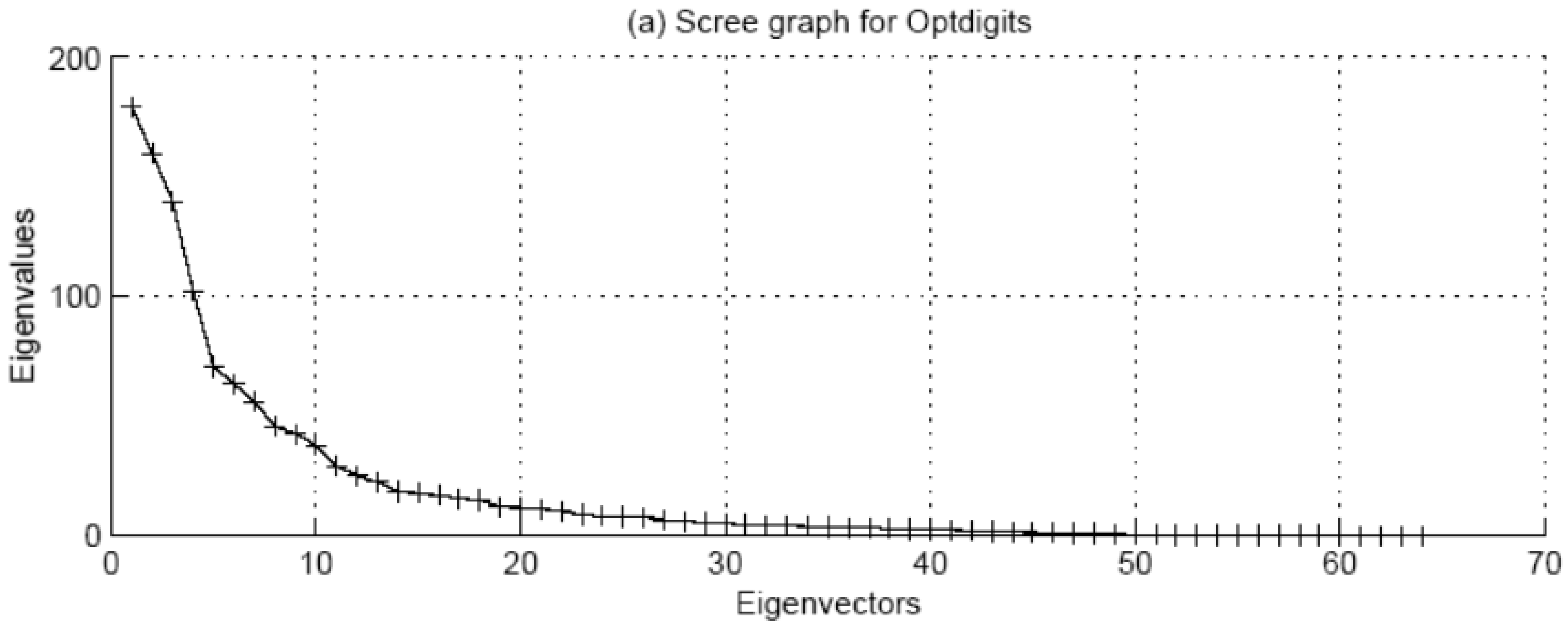
Here,
S = matrix
u = vector
λ = scalar

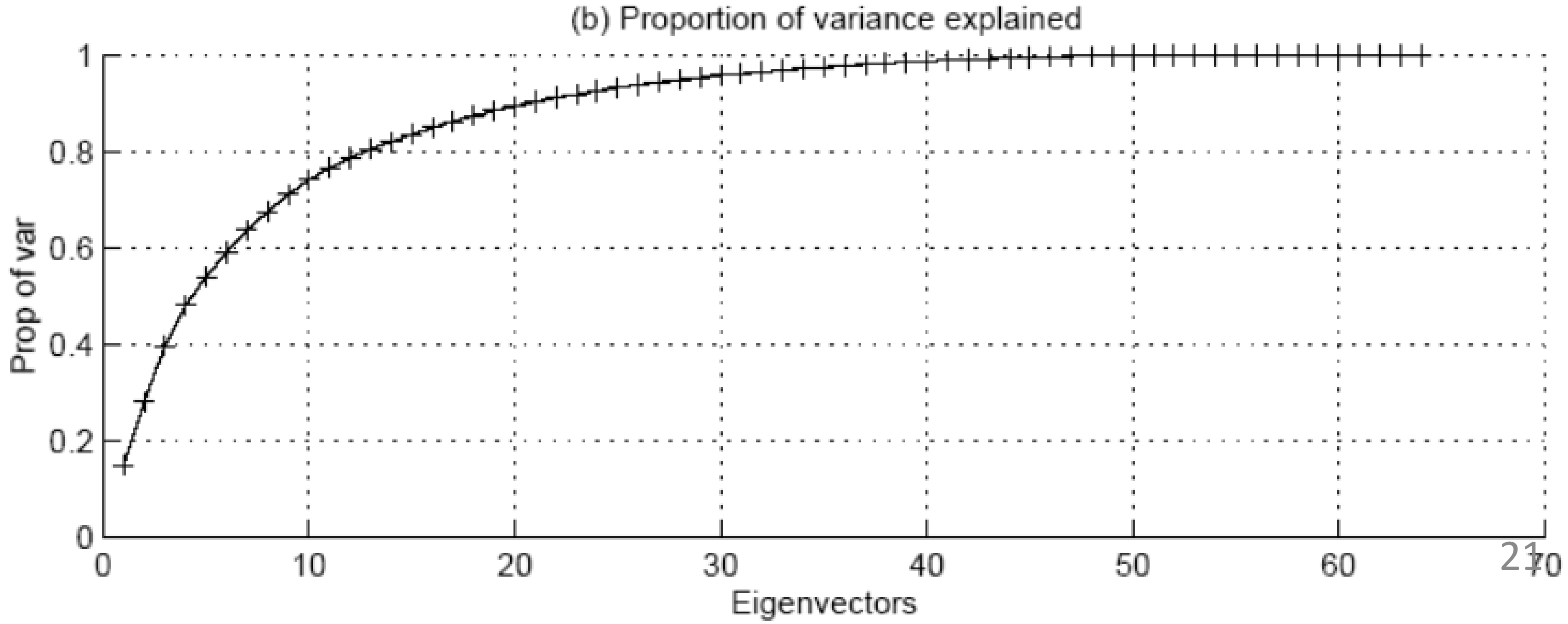
u is eigenvector of S
λ " eigenvalues of u.

$\rightarrow \quad u^T S u = u^T \lambda u$
$= \lambda u^T u \quad [u^T u = 1]$
$= \lambda$

The first PC (Principal Component)

PC is the eigen vector of S that has the max^m eigen value.


(a) Scree graph for Optdigits
Eigenvalues
200
100
0
0
10
20
30
40
50
60
70
Eigenvectors


(b) Proportion of variance explained
Prop of var
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40
50
60
70
Eigenvectors

Principal Components Analysis (PCA)

→ PCA is a useful statistical technique for finding patterns in data of high dimension.

→ It is used to express the data in such a way as to highlight their similarities and differences.

Background Statistics

→ Mean
$$\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n}$$

→ Standard Deviation
$$s = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{(n-1)}}$$

→ Variance
$$s^2 = \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{(n-1)}$$

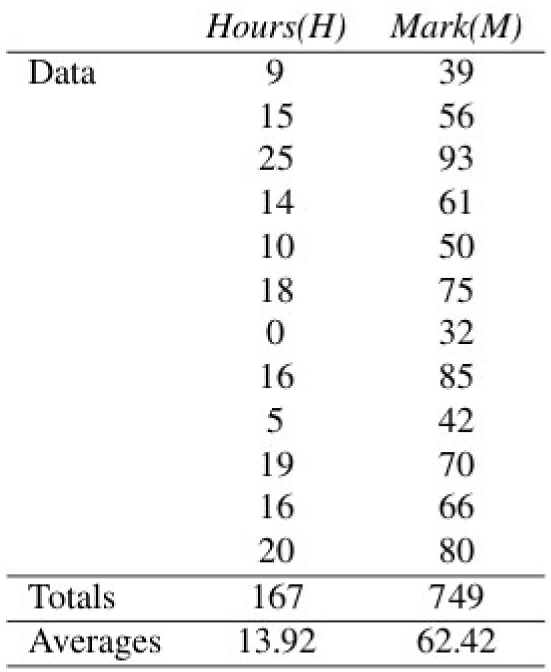
Covariance

→ Standard deviation and variance can only operate one dimensional data.
→ Covariance is a measure of how much two dimensions vary from the mean with respect to each other.

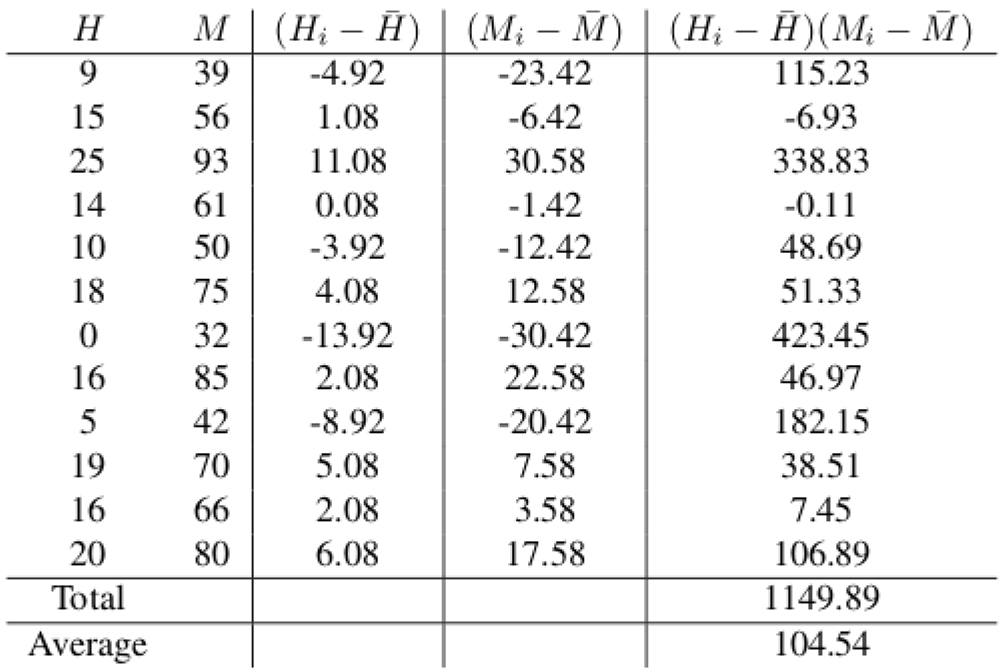
$$var(X) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(X_i - \bar{X})}{(n-1)}$$

$$cov(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)}$$

Covariance: An Example



	Hours(H)	*Mark(M)*
Data	9	39
	15	56
	25	93
	14	61
	10	50
	18	75
	0	32
	16	85
	5	42
	19	70
	16	66
	20	80
Totals	167	749
Averages	13.92	62.42



H	M	$(H_i - \bar{H})$	$(M_i - \bar{M})$	$(H_i - \bar{H})(M_i - \bar{M})$
9	39	-4.92	-23.42	115.23
15	56	1.08	-6.42	-6.93
25	93	11.08	30.58	338.83
14	61	0.08	-1.42	-0.11
10	50	-3.92	-12.42	48.69
18	75	4.08	12.58	51.33
0	32	-13.92	-30.42	423.45
16	85	2.08	22.58	46.97
5	42	-8.92	-20.42	182.15
19	70	5.08	7.58	38.51
16	66	2.08	3.58	7.45
20	80	6.08	17.58	106.89
Total				1149.89
Average				104.54

→ **Positive value** indicates both dimensions increase together.

→ **Negative value** indicates as one increases, the other decreases.

→ Zero indicates dimensions are independent.

Covariance matrix

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

Eigenvector and Eigenvalue

→ An eigenvector or characteristic vector of a square matrix is a vector that does not change its direction under the associated linear transformation.

→ If v is a vector that is not zero, then it is an eigenvector of a square matrix A if A**v** is a scalar multiple of v. This condition could be written as the equation:

$$A\mathbf{v} = \lambda\mathbf{v} \text{ where } \lambda \text{ is the eigenvalue}$$

Self Study

→ Find the eigenvector and eigenvalue of the following matrix.

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix}$$

Steps of PCA

→ Step 1: Get some data

→ Step 2: Subtract the mean

→Step 3: Calculate the covariance matrix

→Step 4: Calculate eigenvectors and eigenvalues of covariance matrix

→Step 5: Choosing components and forming a feature vector

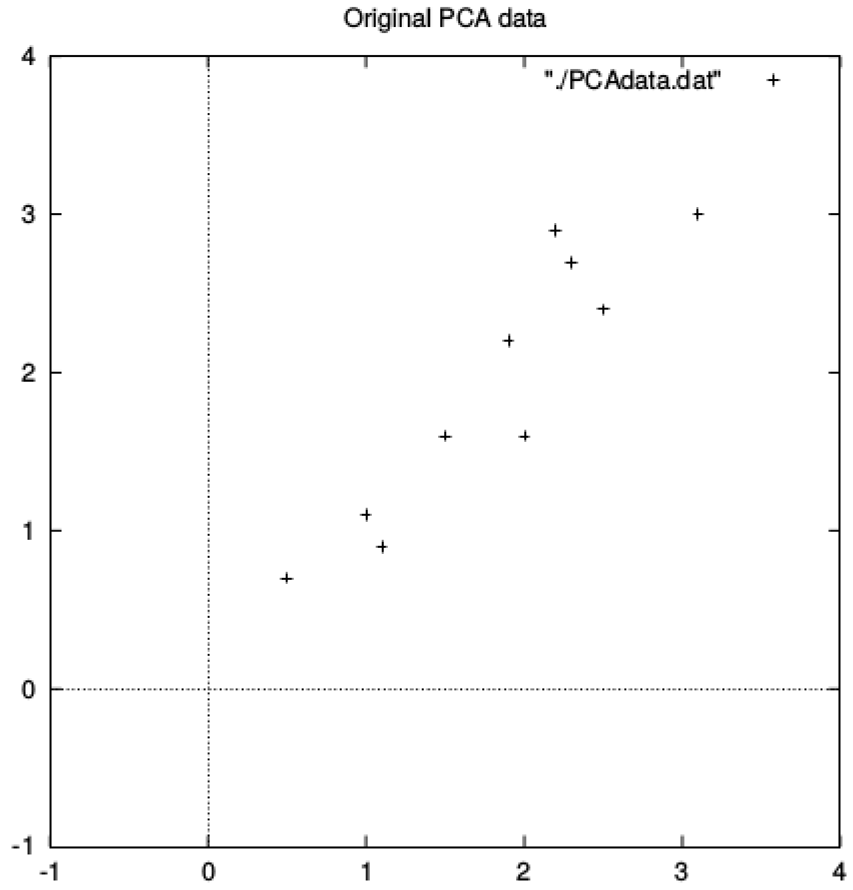
Dataset and subtracting the mean

Data =

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

DataAdjust =

x	y
.69	.49
-1.31	-1.21
.39	.99
.09	.29
1.29	1.09
.49	.79
.19	-.31
-.81	-.81
-.31	-.31
-.71	-1.01



Calculating the Covariance matrix

$$cov = \begin{pmatrix} .616555556 & .615444444 \\ .615444444 & .716555556 \end{pmatrix}$$

$$cov(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)}$$

Eigenvalues and Eigenvectors

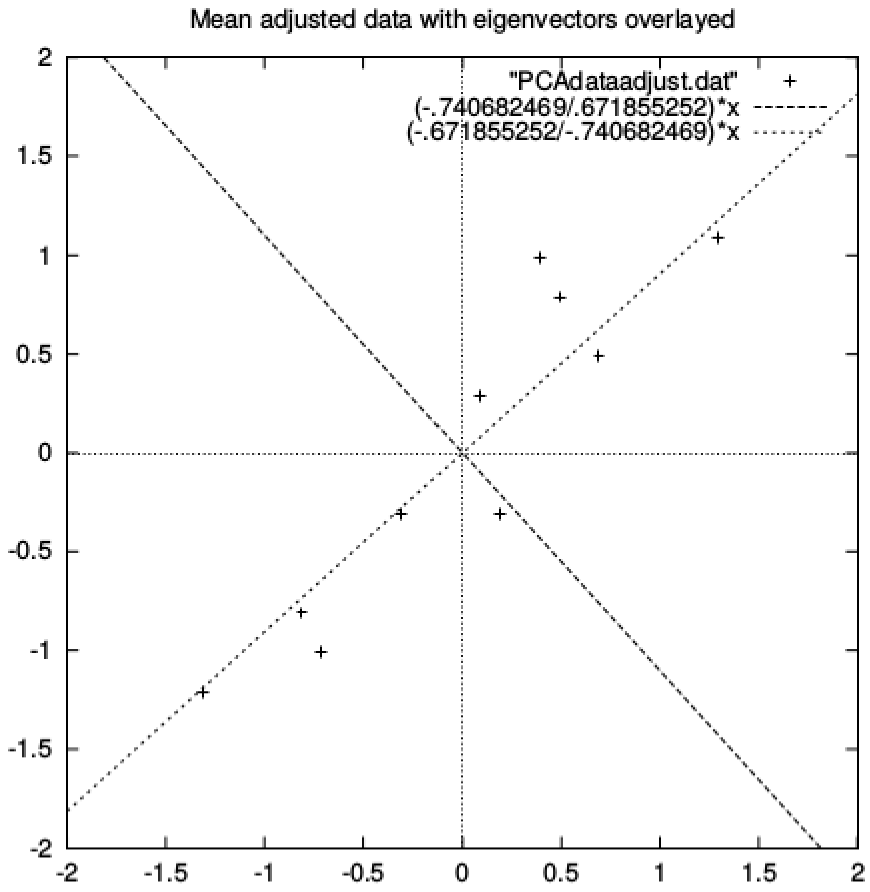
$$eigenvalues = \begin{pmatrix} .0490833989 \\ 1.28402771 \end{pmatrix}$$

$$eigenvectors = \begin{pmatrix} -.735178656 & -.677873399 \\ .677873399 & -.735178656 \end{pmatrix}$$

Eigenvalues and Eigenvectors

$$eigenvalues = \begin{pmatrix} .0490833989 \\ 1.28402771 \end{pmatrix}$$

$$eigenvectors = \begin{pmatrix} -.735178656 & -.677873399 \\ .677873399 & -.735178656 \end{pmatrix}$$



Forming a feature vector

→The eigenvector with the highest eigenvalue is the principle component of the data set.
→ Sort the eigenvalues in descending order.
→ From n eigenvalues, take top p eigenvalues.
→ Dimensions will be changed from n to p.

$$eigenvalues = \begin{pmatrix} .0490833989 \\ 1.28402771 \end{pmatrix}$$

$$\begin{pmatrix} -.677873399 \\ -.735178656 \end{pmatrix}$$

$$eigenvectors = \begin{pmatrix} -.735178656 & -.677873399 \\ .677873399 & -.735178656 \end{pmatrix}$$

$$FeatureVector = (eig_1\ eig_2\ eig_3\ ....\ eig_n)$$

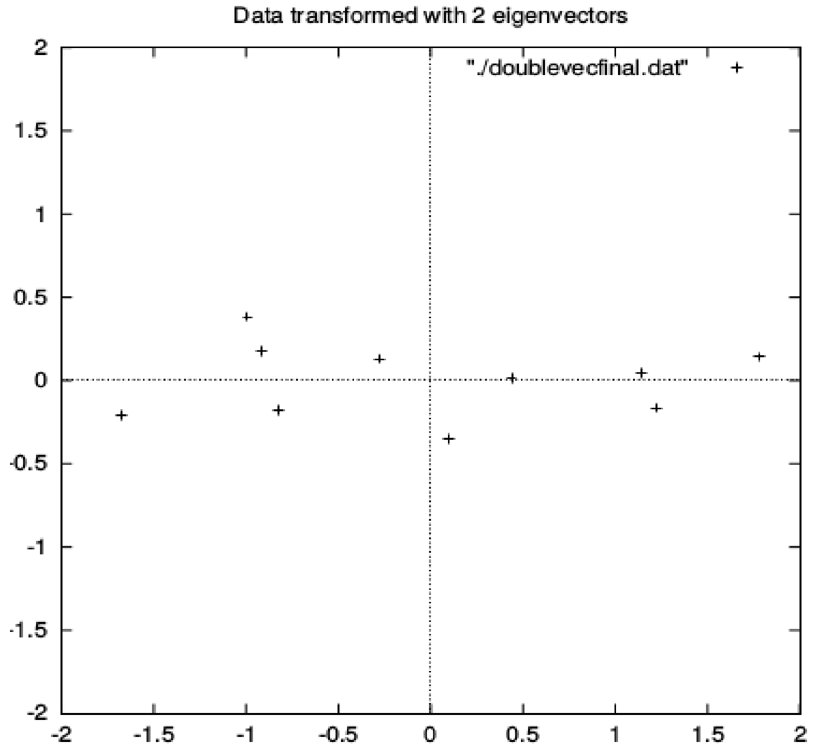
Deriving new dataset

→Take the transpose of the vector and multiply it on the left of the original data set, transposed.

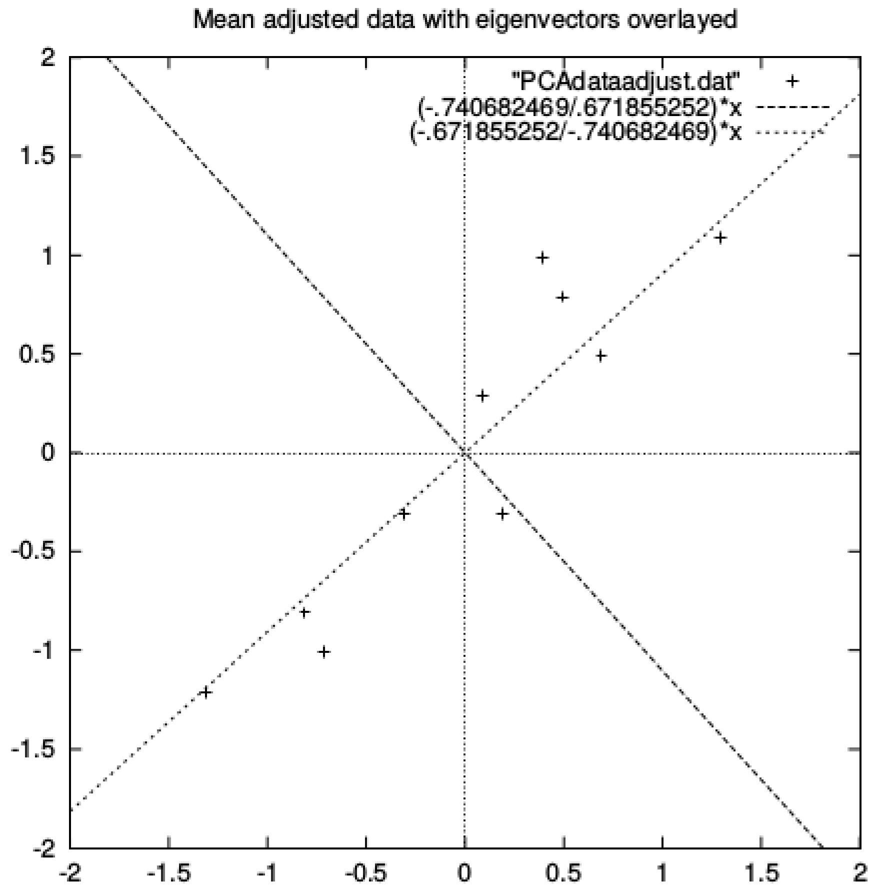
$$FinalData = RowFeatureVector \times RowDataAdjust,$$

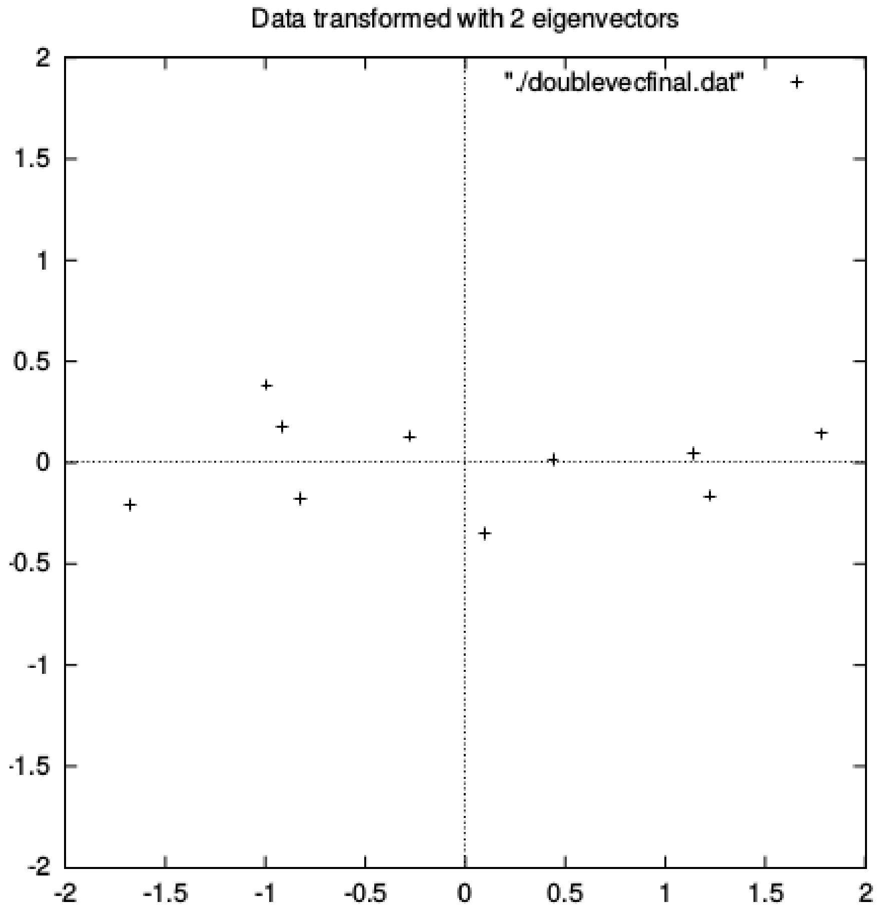
Transformed Data=

x	y
-.827970186	-.175115307
1.77758033	.142857227
-.992197494	.384374989
-.274210416	.130417207
-1.67580142	-.209498461
-.912949103	.175282444
.0991094375	-.349824698
1.14457216	.0464172582
.438046137	.0177646297
1.22382056	-.162675287



Old and new dataset





Getting Original Data back

→Getting back exact original data is only possible if we take all eigenvectors during transformation. Otherwise, we lose some information.

$$FinalData = RowFeatureVector \times RowDataAdjust,$$

$$RowDataAdjust = RowFeatureVector^{-1} \times FinalData$$

$$RowDataAdjust = RowFeatureVector^{T} \times FinalData$$

$$RowOriginalData = (RowFeatureVector^{T} \times FinalData) + OriginalMean$$

Thank You