syde675 pattern recognition

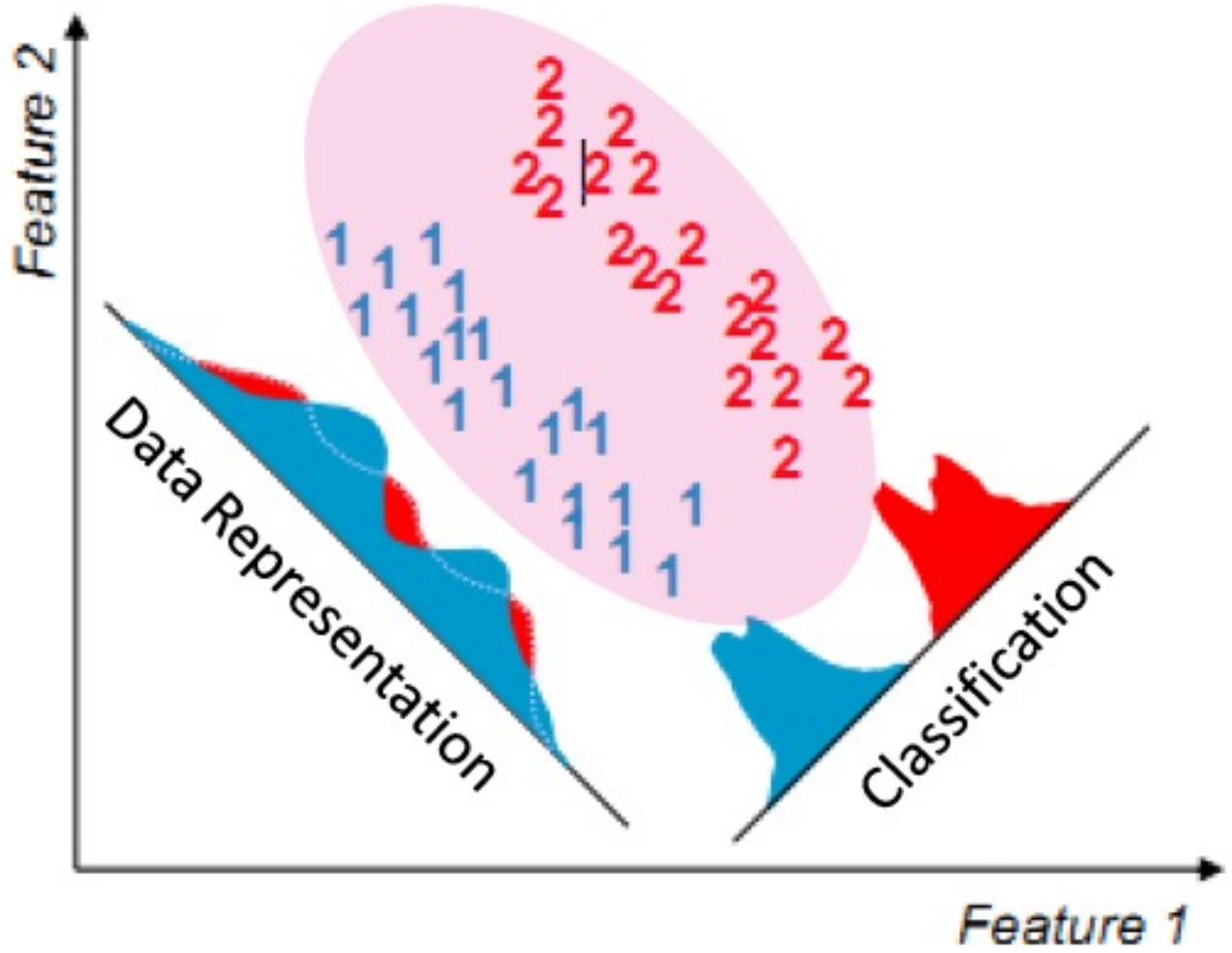
Principle Component Extraction

J.Zelek 2022

Topics

- Introduction
- Feature Selection vs. Feature Extraction
- Principle Component Analysis (PCA)
  - PCA Intuition
  - PCA Derivation
  - PCA Algorithm
  - PCA Applications

Principle Component Analysis



Principle Component Analysis

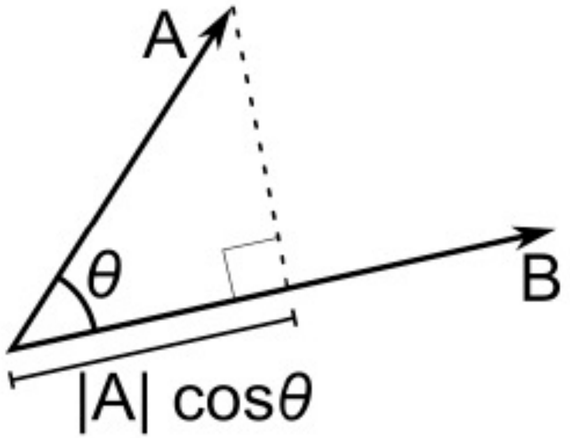
- PCA is an unsupervised dimensionality reduction technique
- Given a matrix of data points, it finds one or more orthogonal directions that capture the largest amount of variance in the data.
- Intuitively, the directions with less variance contain less info & can be discarded

Projection

Let us first review the meaning of scalar projection of one vector onto another. If $\mathbf{v} \in \mathbb{R}^d$ is a unit vector, i.e. $\|\mathbf{v}\| = 1$, then the scalar projection of another vector $\mathbf{x} \in \mathbb{R}^d$ onto $\mathbf{v}$ is given by $\mathbf{x}^\top \mathbf{v}$. This quantity tells us roughly how much of the projected vector $\mathbf{x}$ lies along the direction given direction $\mathbf{v}$. Why does this expression make sense? Recall the slightly more general formula which holds for vectors of any length:

$$\mathbf{x}^\top \mathbf{v} = \|\mathbf{x}\|\|\mathbf{v}\| \cos\theta$$

where θ is the angle between the vectors. In this case, since $\|\mathbf{v}\| = 1$, the expression simplifies to $\mathbf{x}^\top \mathbf{v} = \|\mathbf{x}\| \cos\theta$. But since cosine gives the ratio of the adjacent side (the projection we want to find) to the hypotenuse ($\|\mathbf{x}\|$), this is exactly what we want:



The first principal component

Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ be our matrix of data, where each row is a d-dimensional datapoint. These are to be thought of as i.i.d. samples from some random vector $\mathbf{x}$.

We will assume that the data points have mean zero; if this is not the case, we can make it so by subtracting the average of all the rows, $\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$, from each row. The motivation for this is that we want to find directions of high variance within the data, and variance is defined relative to the mean of the data. If we did not zero-center the data, the directions found would be heavily influenced by where the data lie relative to the origin, rather than where they lie relative to the other data, which is more useful.

Since $\mathbf{X}$ is zero-mean, the sample variance of the datapoints' projections onto a unit vector $\mathbf{v}$ is given by

$$\frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i^\top \mathbf{v})^2 = \frac{1}{n}\|\mathbf{X}\mathbf{v}\|^2 = \frac{1}{n}\mathbf{v}^\top \mathbf{X}^\top \mathbf{X}\mathbf{v}$$

The first principal component

where $\mathbf{v}$ is constrained to have unit norm.[1]

With this motivation, we define the **first loading vector** $\mathbf{v}_1$ as the solution to the constrained optimization problem

$$\max_{\mathbf{v}} \mathbf{v}^T\mathbf{X}^T\mathbf{X}\mathbf{v} \quad \text{subject to} \quad \mathbf{v}^T\mathbf{v} = 1$$

Note that we have discarded the positive constant factor $1/n$ which does not affect the optimal value of $\mathbf{v}$.

To reduce this constrained optimization problem to an unconstrained one, we write down its Lagrangian:

$$\mathcal{L}(\mathbf{v}) = \mathbf{v}^T\mathbf{X}^T\mathbf{X}\mathbf{v} - \lambda(\mathbf{v}^T\mathbf{v} - 1)$$

First-order necessary conditions for optima imply that

$$\mathbf{0} = \nabla\mathcal{L}(\mathbf{v}_1) = 2\mathbf{X}^T\mathbf{X}\mathbf{v}_1 - 2\lambda\mathbf{v}_1$$

Hence $\mathbf{X}^T\mathbf{X}\mathbf{v}_1 = \lambda\mathbf{v}_1$, i.e. $\mathbf{v}_1$ is an eigenvector of $\mathbf{X}^T\mathbf{X}$ with eigenvalue λ. Since we constrain $\mathbf{v}_1^T\mathbf{v}_1 = 1$, the value of the objective is precisely

$$\mathbf{v}_1^T\mathbf{X}^T\mathbf{X}\mathbf{v}_1 = \mathbf{v}_1^T(\lambda\mathbf{v}_1) = \lambda\mathbf{v}_1^T\mathbf{v}_1 = \lambda$$

so the optimal value is $\lambda = \lambda_{\max}(\mathbf{X}^T\mathbf{X})$, which is achieved when $\mathbf{v}_1$ is a unit eigenvector of $\mathbf{X}^T\mathbf{X}$ corresponding to its largest eigenvalue.

The first principal component - aside

[1] To make sense of the sample variance, recall that for any random variable Z,

$$\mathrm{Var}(Z) = \mathbb{E}[(Z - \mathbb{E}[Z])^2]$$

so if $\mathbb{E}[Z] = 0$ then $\mathrm{Var}(Z) = \mathbb{E}[Z^2]$. In practice we will not have the true random variable Z, but rather i.i.d. observations $z_1, \ldots, z_n$ of Z. The expected value can then be approximated by a sample average, i.e.

$$\mathbb{E}[Z^2] \approx \frac{1}{n} \sum_{i=1} z_i^2$$

which is justified by the law of large numbers, which states that (under mild conditions) the sample average converges to the expected value as $n \to \infty$. In our case the random variable Z is the principal component $\mathbf{v}^\top \mathbf{x}$, and the i.i.d. observations are the projections of our datapoints, i.e. $z_i = \mathbf{v}^\top \mathbf{x}_i$.

Finding more principal components

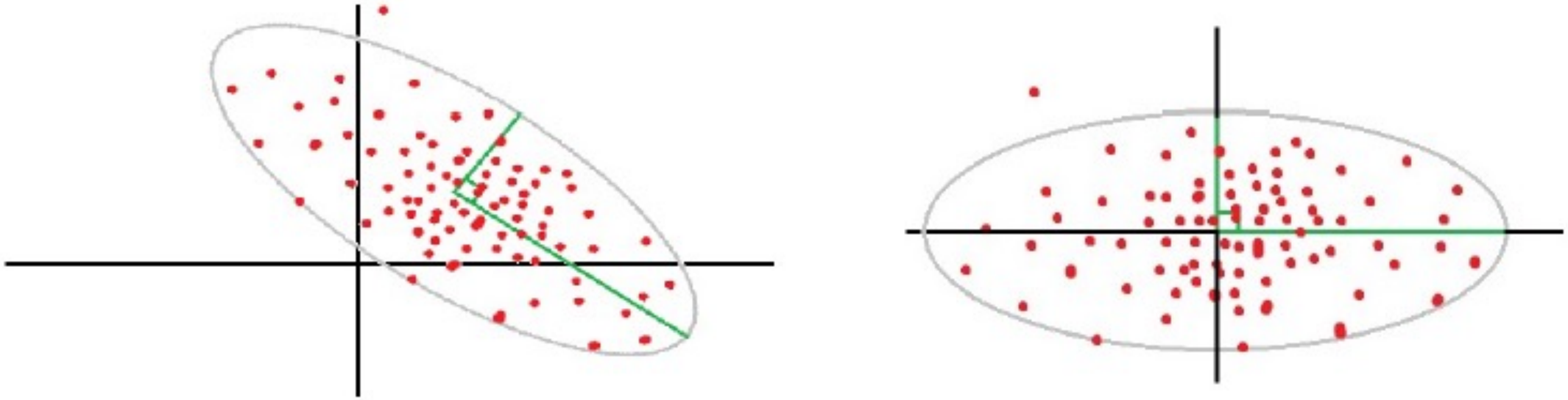
- we define the kth loading vector v_k as the solution to the constrained optimization problem:

$$\max_{\mathbf{v}} \mathbf{v}^T\mathbf{X}^T\mathbf{X}\mathbf{v} \quad \text{subject to} \quad \mathbf{v}^T\mathbf{v} = 1$$

$$\mathbf{v}^T\mathbf{v}_i = 0, \quad i = 1, \ldots, k-1$$

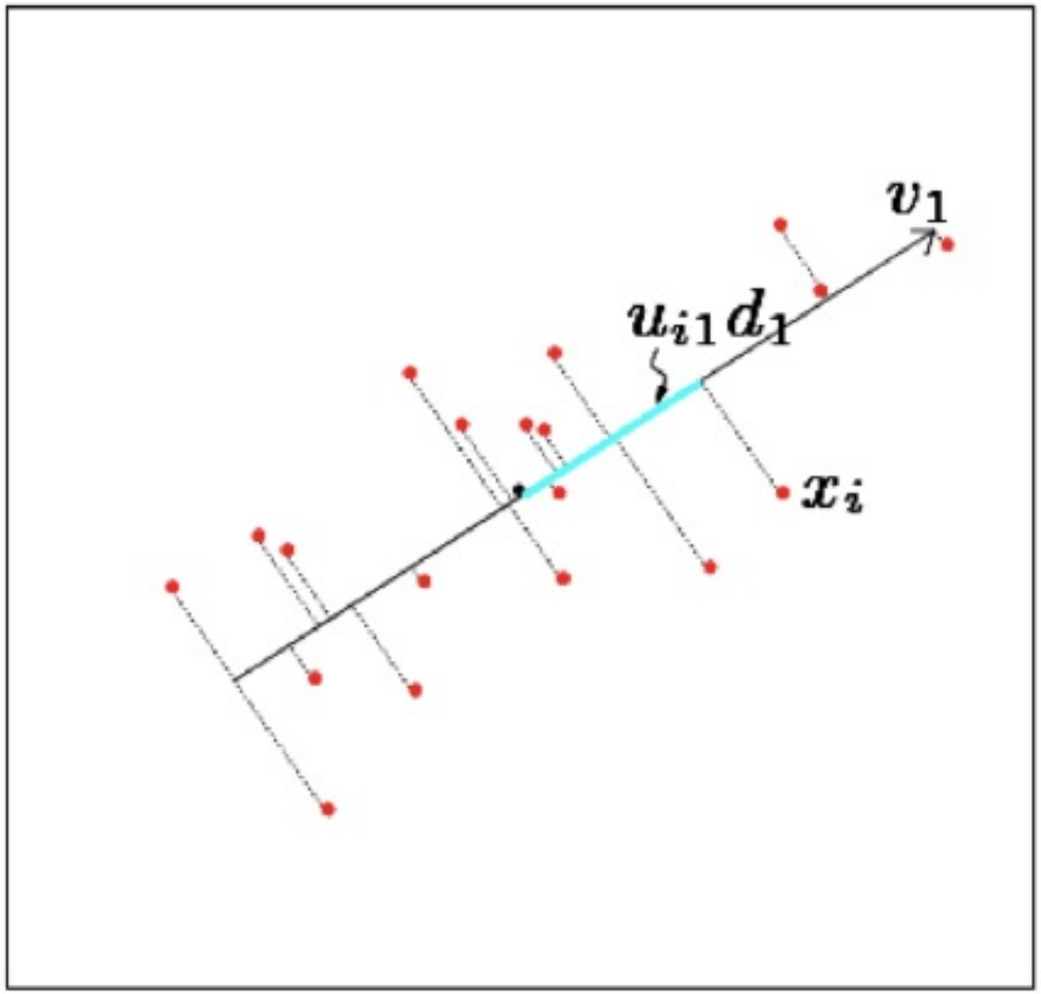
We claim that $\mathbf{v}_k$ is a unit eigenvector of $\mathbf{X}^T\mathbf{X}$ corresponding to its kth largest eigenvalue.

Projecting onto the PCA coordinate system



- left: data points; right: PCA projection of data points

Projecting onto the PCA coordinate system



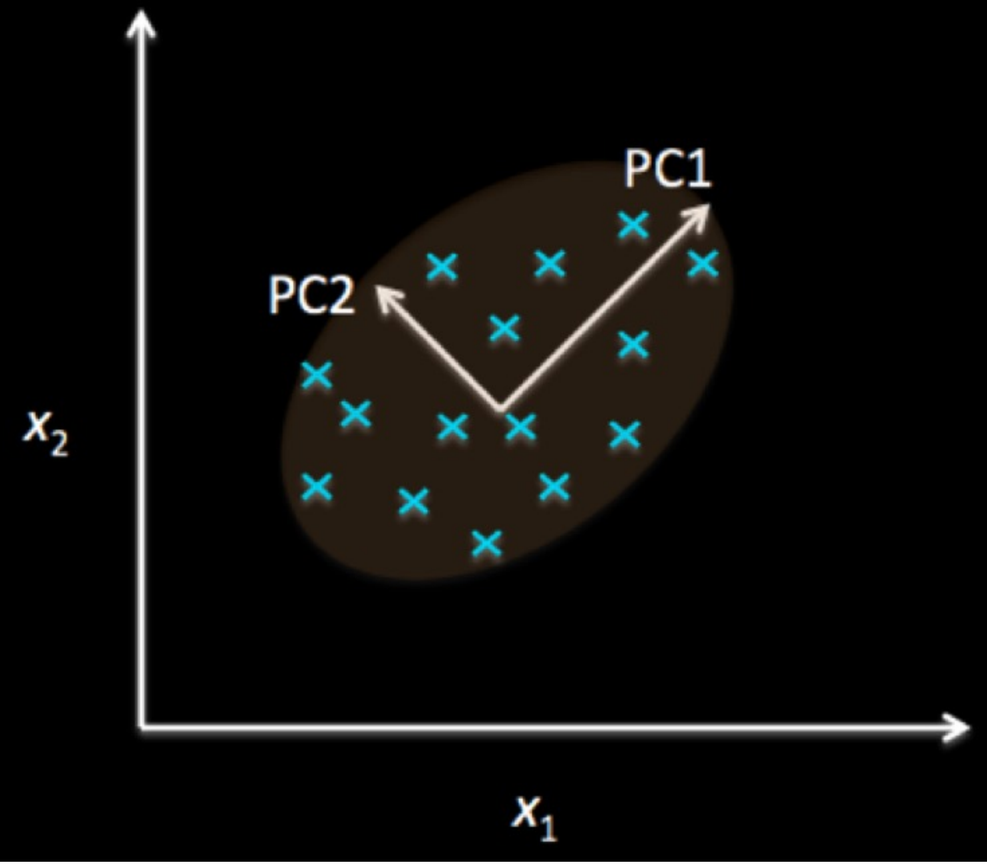
- The first linear principal component of a set of data. The line minimizes the total squared distance from each point to its orthogonal projection onto the line.

PCA

- Clustering may also be viewed as an unsupervised dimensionality reduction technique; in that it assigns possibly high dimensional feature vectors to one of K tokens (cluster labels)
- Clumpy data is useful for clustering; PCA is useful for manifoldy data (preferably linear) which is well represented in low dimension coordinate system
- Often problems well suited to k-means are ill suited to PCA & vice versa

PCA

orthogonal dimensions that maximize variance of X



PCA (simple intro): Covariance

- Variance and Covariance are a measure of the "spread" of a set of points around their center of mass (mean)
- Variance – measure of the deviation from the mean for points in one dimension e.g. heights
- Covariance as a measure of how much each of the dimensions vary from the mean with respect to each other.
- Covariance is measured between 2 dimensions to see if there is a relationship between the 2 dimensions e.g. number of hours studied & marks obtained.
- The covariance between one dimension and itself is the variance

PCA (simple intro): Covariance

$$\text{covariance}(X,Y) = \frac{\sum_{i=1}^{n} (\overline{X_i} - X)(\overline{Y_i} - Y)}{(n-1)}$$

- So, if you had a 3-dimensional data set (x,y,z), then you could measure the covariance between the x and y dimensions, the y and z dimensions, and the x and z dimensions. Measuring the covariance between x and x , or y and y , or z and z would give you the variance of the x , y and z dimensions respectively.

PCA (simple intro): Covariance Matrix

- Representing Covariance between dimensions as a matrix e.g. for 3 dimensions:

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

Variances

- Diagonal is the variances of x, y and z
- cov(x,y) = cov(y,x) hence matrix is symmetrical about the diagonal
- N-dimensional data will result in NxN covariance matrix

PCA (simple intro): Covariance

- What is the interpretation of covariance calculations?

  e.g.: 2 dimensional data set

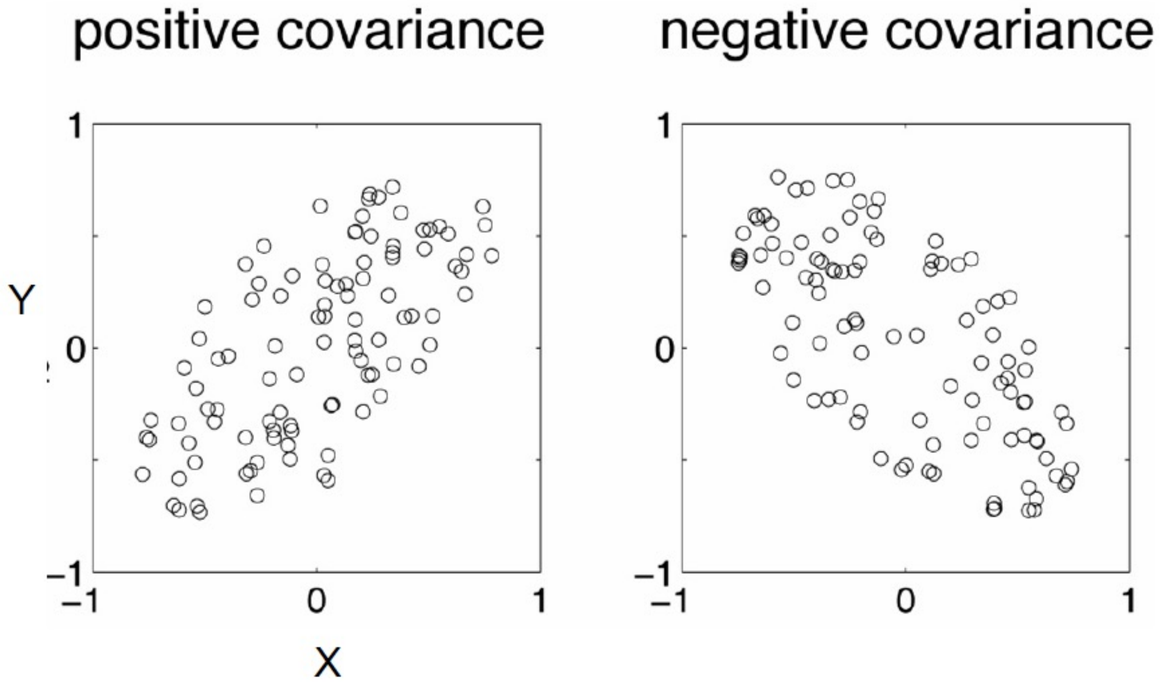
  x: number of hours studied for a subject

  y: marks obtained in that subject

  covariance value is say: 104.53

  what does this value mean?

PCA (simple intro): Covariance Examples



PCA (simple intro): Covariance

- Exact value is not as important as it's sign.
- A positive value of covariance indicates both dimensions increase or decrease together e.g. as the number of hours studied increases, the marks in that subject increase.
- A negative value indicates while one increases the other decreases, or vice-versa e.g. active social life at PSU vs performance in CS dept.
- If covariance is zero: the two dimensions are independent of each other e.g. heights of students vs the marks obtained in a subject

PCA (simple intro): Covariance

- Why bother with calculating covariance when we could just plot the 2 values to see their relationship?

  Covariance calculations are used to find relationships between dimensions in high dimensional data sets (usually greater than 3) where visualization is difficult.

PCA (simple intro):

- **principal components analysis (PCA)** is a technique that can be used to simplify a dataset
- It is a linear transformation that chooses a new coordinate system for the data set such that

  greatest variance by any projection of the data set comes to lie on the first axis (then called the first principal component),

  the second greatest variance on the second axis, and so on.
- PCA can be used for reducing dimensionality by eliminating the later principal components.

PCA (simple intro): PCA toy example

Consider the following 3D points

1	2	4	3	5	6
2	4	8	6	10	12
3	6	12	9	15	18

If each component is stored in a byte,
we need 18 = 3 x 6 bytes

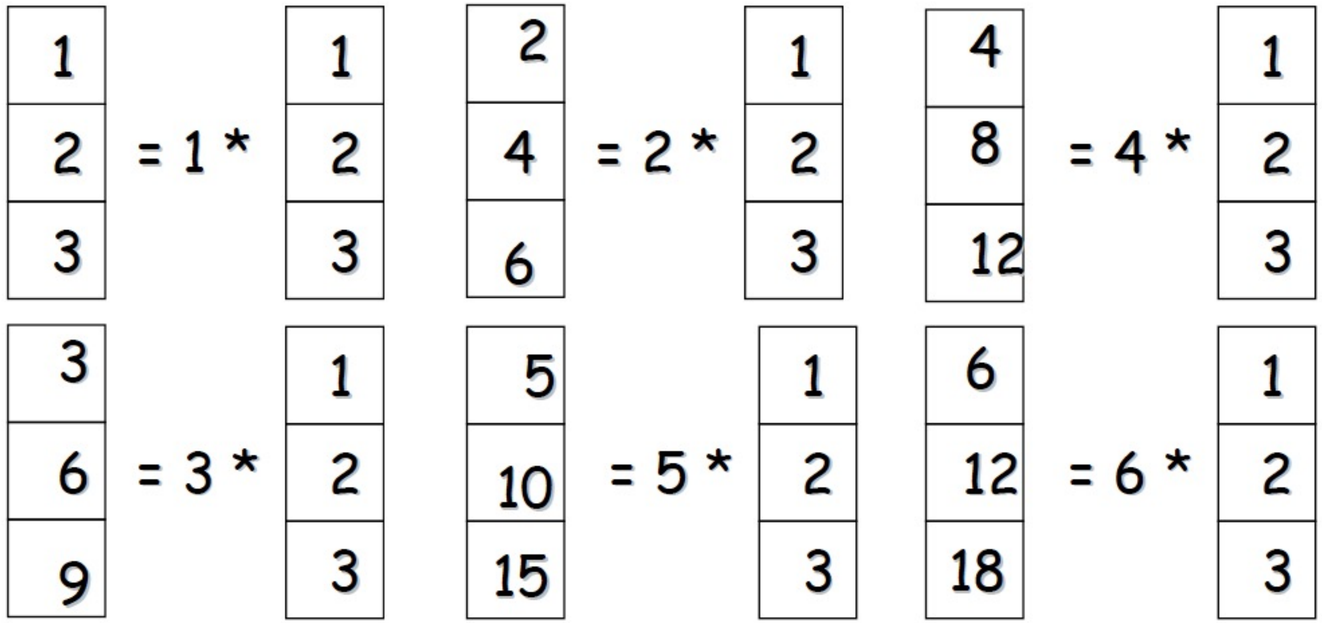
PCA (simple intro): PCA toy example

Looking closer, we can see that all the points are related geometrically: they are all the same point, scaled by a factor:

1		1	2		1	4		1	3		1
2	= 1 *	2	4	= 2 *	2	8	= 4 *	2	6	= 3 *	2
3		3	6		3	12		3	9		3

5		1	6		1						
10	= 5 *	2	12	= 6 *	2						
15		3	18		3						

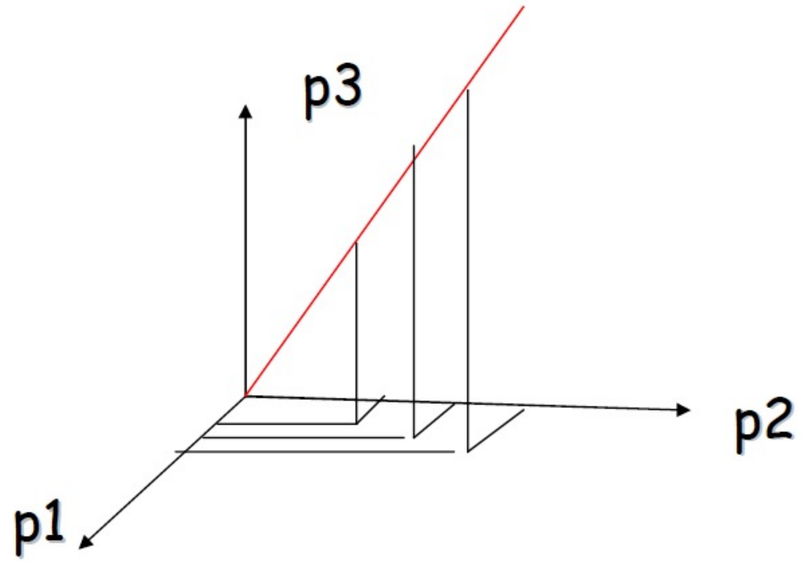
PCA (simple intro): PCA toy example



They can be stored using only 9 bytes (50% savings!):
Store one point (3 bytes) + the multiplying constants (6 bytes)

PCA (simple intro): Geometrical interpretation

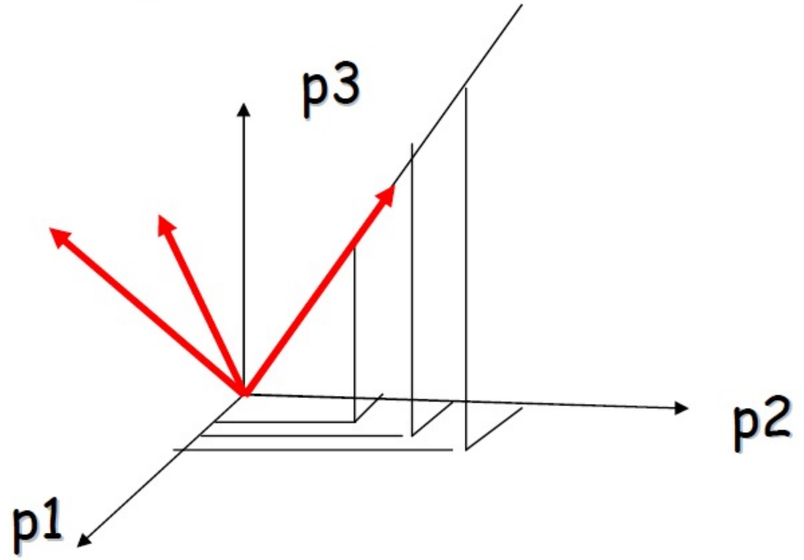
View each point in 3D space.



But in this example, all the points happen to belong to a line: a 1D subspace of the original 3D space.

PCA (simple intro): Geometrical interpretation

Consider a new coordinate system where one of the axes is along the direction of the line:



In this coordinate system, every point has only one non-zero coordinate: we *only* need to store the direction of the line (a 3 bytes image) and the non-zero coordinate for each of the points (6 bytes).

PCA (simple intro):

- Given a set of points, how do we know if they can be compressed like in the previous example?
  - The answer is to look into the correlation between the points
  - The tool for doing this is called PCA

PCA (simple intro):

- By finding the eigenvalues and eigenvectors of the covariance matrix, we find that the eigenvectors with the largest eigenvalues correspond to the dimensions that have the strongest correlation in the dataset.
- This is the principal component.
- PCA is a useful statistical technique that has found application in:
  - fields such as face recognition and image compression
  - finding patterns in data of high dimension.

PCA (simple intro): PCA Theorem

Let x_1 x_2 … x_n be a set of n N x 1 vectors and let $\bar{x}$ be their average:

$$\mathbf{x}_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{iN} \end{bmatrix} \qquad \bar{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^{i=n} \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{iN} \end{bmatrix}$$

PCA (simple intro): PCA Theorem

Let X be the N x n matrix with columns
$x_1 - \bar{x}, x_2 - \bar{x}, \ldots x_n - \bar{x}$:

$$X = \left[\mathbf{x}_1 - \bar{\mathbf{x}} \quad \mathbf{x}_2 - \bar{\mathbf{x}} \quad \cdots \quad \mathbf{x}_n - \bar{\mathbf{x}} \right]$$

Note: subtracting the mean is equivalent to translating the coordinate system to the location of the mean.

PCA (simple intro): PCA Theorem

Let Q = X X^T be the N x N matrix:

$$Q = XX^T = \left[\begin{array}{cccc} \mathbf{x}_1 - \bar{\mathbf{x}} & \mathbf{x}_2 - \bar{\mathbf{x}} & \cdots & \mathbf{x}_n - \bar{\mathbf{x}} \end{array} \right] \left[\begin{array}{c} (\mathbf{x}_1 - \bar{\mathbf{x}})^T \\ (\mathbf{x}_2 - \bar{\mathbf{x}})^T \\ \vdots \\ (\mathbf{x}_n - \bar{\mathbf{x}})^T \end{array} \right]$$

Notes:

1. Q is square
2. Q is symmetric
3. Q is the *covariance* matrix [aka scatter matrix]
4. Q can be very large (in vision, N is often the number of pixels in an image!)

PCA (simple intro): PCA Theorem

Theorem:

Each x_j can be written as: $$\mathbf{x}_j = \bar{\mathbf{x}} + \sum_{i=1}^{i=n} g_{ji}\mathbf{e}_i$$

where e_i are the n eigenvectors of Q with non-zero eigenvalues.

Notes:

1. The eigenvectors e_1 e_2 … e_n span an ***eigenspace***
2. e_1 e_2 … e_n are N x 1 orthonormal vectors (directions in N-Dimensional space)
3. The scalars g_{ji} are the coordinates of x_j in the space.

$$g_{ji} = (\mathbf{x}_j - \bar{\mathbf{x}}).\mathbf{e}_i$$

PCA (simple intro): using PCA to compress data

- Expressing x in terms of $e_1 \ldots e_n$ has not changed the size of the data
- However, if the points are highly correlated many of the coordinates of x will be zero or closed to zero.

note: this means they lie in a lower-dimensional linear subspace

PCA (simple intro): using PCA to compress data

- Sort the eigenvectors e_i according to their eigenvalue:

$$\lambda_1 \geq \lambda_2 \geq \ldots \lambda_n$$

- Assuming that $\lambda_i \approx 0$ if $i > k$

- Then

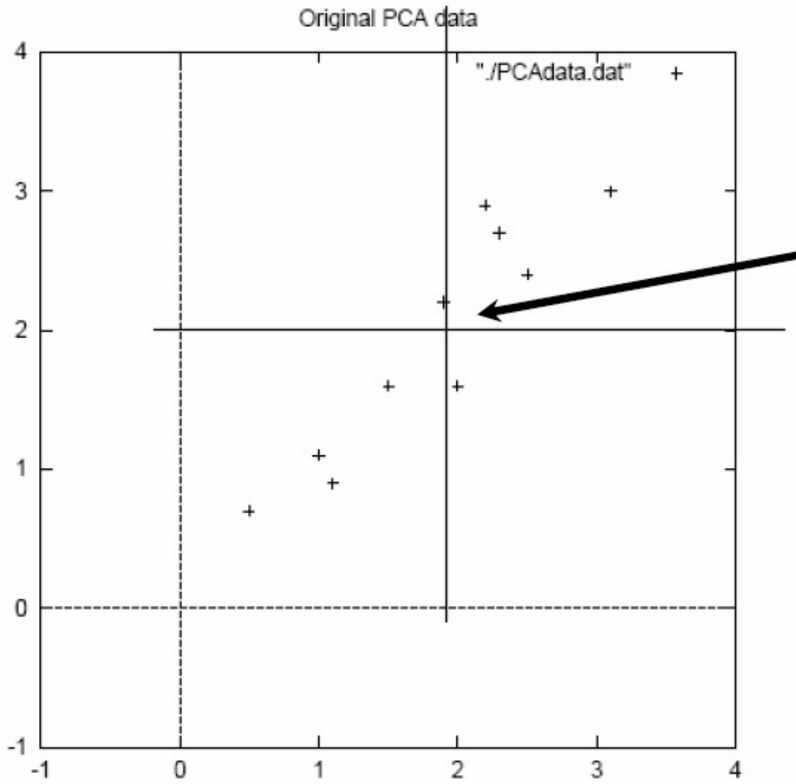
$$\mathbf{x}_j \approx \bar{\mathbf{x}} + \sum_{i=1}^{i=k} g_{ji}\mathbf{e}_i$$

PCA (simple intro): step 1

http://kybele.psych.cornell.edu/~edelman/Psych-465-Spring-2003/PCA-tutorial.pdf

- 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



mean

this becomes the new origin of the data from now on

Figure 3.1: PCA example data, original data on the left, data with the means subtracted on the right, and a plot of the data

PCA (simple intro): step 2

- Calculate the covariance matrix

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

- since the non-diagonal elements in this covariance matrix are positive, we should expect that both the x and y variable increase together.

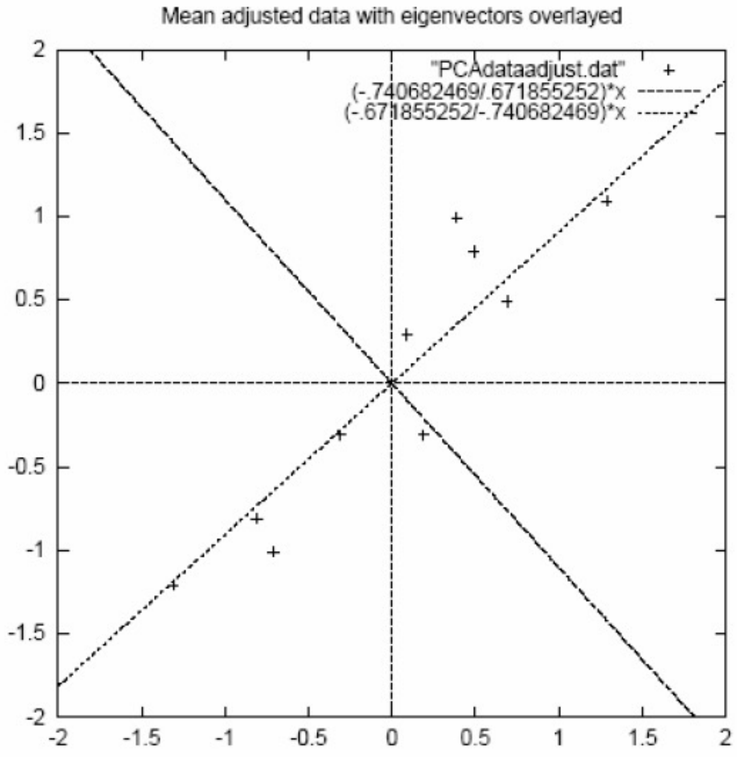
PCA (simple intro): step 3

- Calculate the eigenvectors and eigenvalues of the covariance matrix

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

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

PCA (simple intro): step 3



gure 3.2: A plot of the normalised data (mean subtracted) with the eigenvectors of e covariance matrix overlayed on top.

- eigenvectors are plotted as diagonal dotted lines on the plot.
- Note they are perpendicular to each other.
- Note one of the eigenvectors goes through the middle of the points, like drawing a line of best fit.
- The second eigenvector gives us the other, less important, pattern in the data, that all the points follow the main line, but are off to the side of the main line by some amount.

PCA (simple intro): step 4

- Feature Vector

  FeatureVector = (eig_1 eig_2 eig_3 … eig_n)

  We can either form a feature vector with both of the eigenvectors:

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

  or, we can choose to leave out the smaller, less significant component and only have a single column:

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

PCA (simple intro): step 5

- Deriving new data coordinates

FinalData = RowFeatureVector x RowZeroMeanData

RowFeatureVector is the matrix with the eigenvectors in the columns *transposed* so that the eigenvectors are now in the rows, with the most significant eigenvector at the top

RowZeroMeanData is the mean-adjusted data *transposed*, ie. the data items are in each column, with each row holding a separate dimension.

Note: his is essential Rotating the coordinate axes so higher-variance axes come first.

PCA (simple intro): step 5

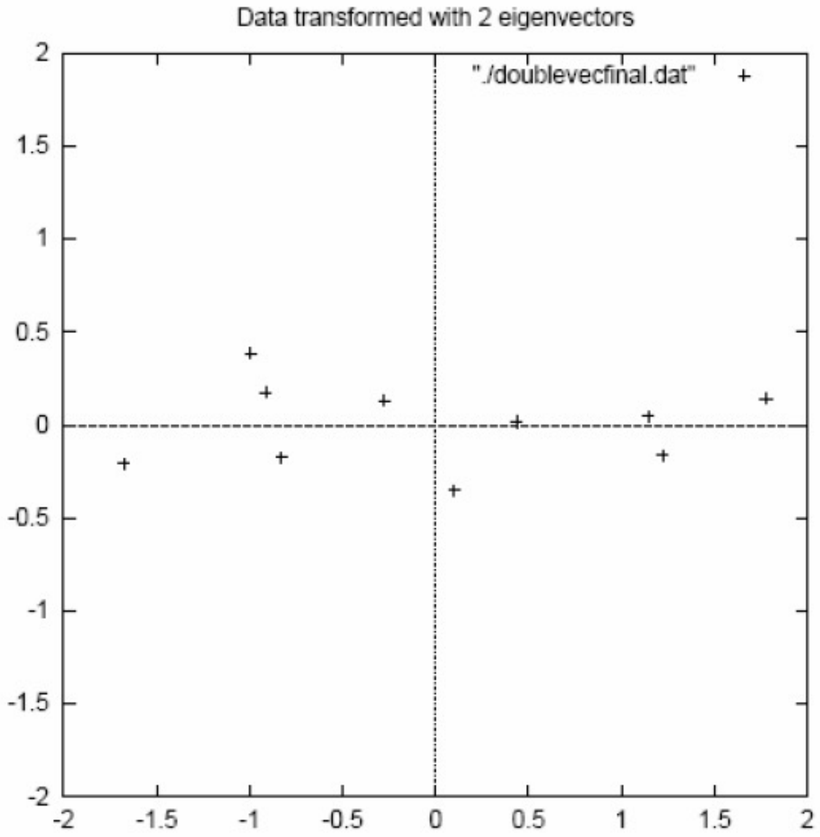
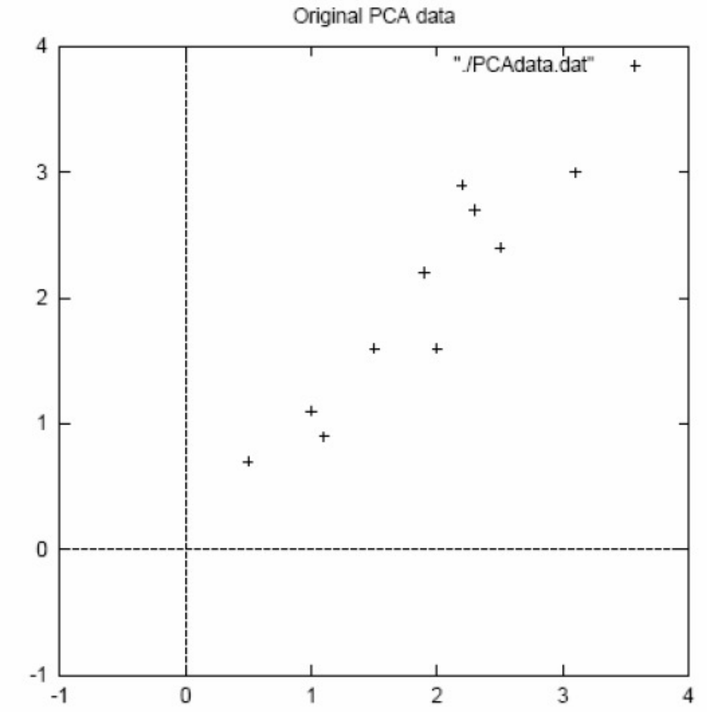


Figure 3.3: The table of data by applying the PCA analysis using both eigenvectors, and a plot of the new data points.

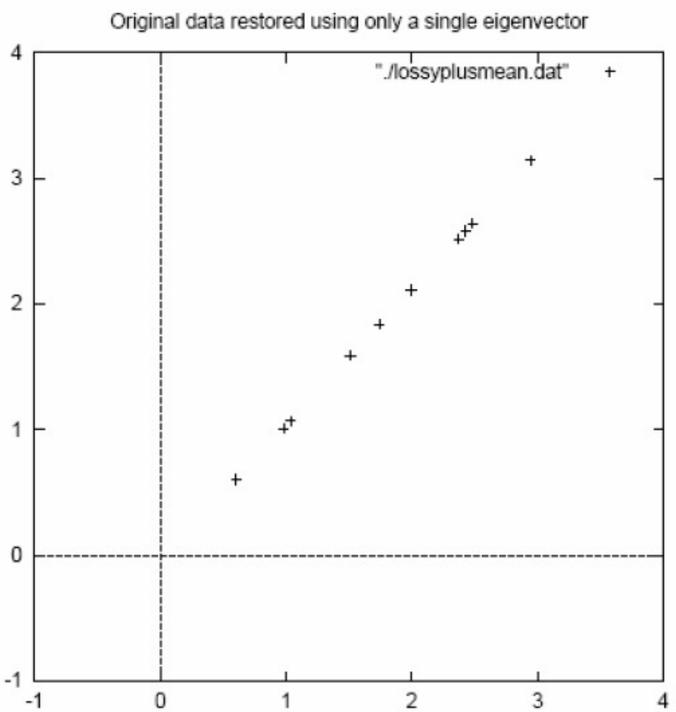
PCA (simple intro): example - approximation

- If we reduced the dimensionality, obviously, when reconstructing the data we would lose those dimensions we chose to discard. In our example let us assume that we considered only the x dimension…

PCA (simple intro): example – final approximation



2D point cloud



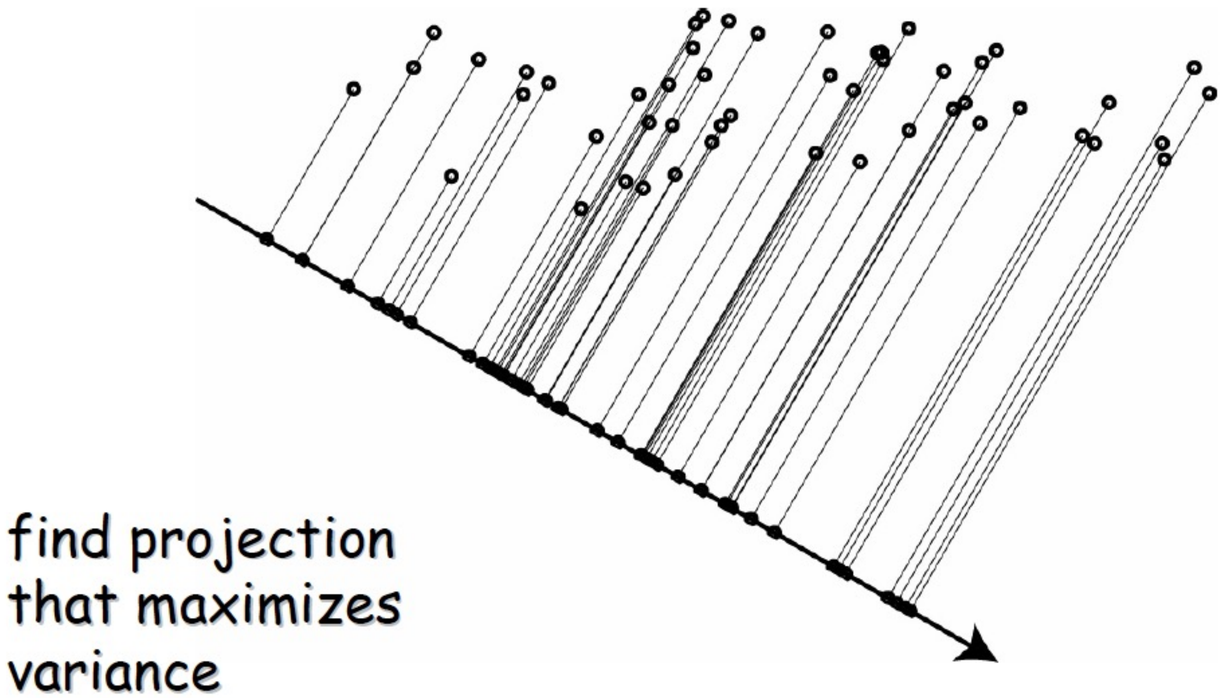
Approximation using one eigenvector basis

PCA (simple intro):

Another way of thinking about Principal component

- direction of maximum variance in the input space
- happens to be same as the principal eigenvector of the covariance matrix

PCA (simple intro): 1 dimensional projection



PCA (simple intro): covariance to variance

- From the covariance, the variance of any projection can be calculated.
- Let *w* be a unit vector

$$\left\langle \left(w^T x\right)^2 \right\rangle - \left\langle w^T x \right\rangle^2 = w^T C w$$

$$= \sum_{ij} w_i C_{ij} w_j$$

PCA (simple intro): maximizing variance

- Principal eigenvector of C
  - the one with the largest eigenvalue.

$$w^* = \arg\max_{w:|w|=1} w^T C w$$

$$\lambda_{\max}(C) = \max_{w:|w|=1} w^T C w$$

$$= w^{*T} C w^*$$

PCA (simple intro): implementing PCA

- Need to find "first" k eigenvectors of Q:

$$Q = XX^T = \begin{bmatrix} \mathbf{x}_1 - \bar{\mathbf{x}} & \mathbf{x}_2 - \bar{\mathbf{x}} & \cdots & \mathbf{x}_n - \bar{\mathbf{x}} \end{bmatrix} \begin{bmatrix} (\mathbf{x}_1 - \bar{\mathbf{x}})^T \\ (\mathbf{x}_2 - \bar{\mathbf{x}})^T \\ \vdots \\ (\mathbf{x}_n - \bar{\mathbf{x}})^T \end{bmatrix}$$

Q is N x N (Again, N could be the number of pixels in an image. For a 256 x 256 image, N = 65536 !!)
Don't want to explicitly compute Q!!!!

PCA (simple intro): SVD

Any m x n matrix X can be written as the product of 3 matrices:

$$X = UDV^T$$

Where:

- U is m x m and its columns are orthonormal vectors
- V is n x n and its columns are orthonormal vectors
- D is m x n diagonal and its diagonal elements are called the singular values of X, and are such that:

$$\sigma_1 , \sigma_2 , \ldots \sigma_n , 0$$

PCA (simple intro): SVD properties

$$X = UDV^T$$

- The columns of U are the eigenvectors of XX^T
- The columns of V are the eigenvectors of X^TX
- The squares of the diagonal elements of D are the eigenvalues of XX^T and X^TX

Linear Algebra: Basic Definitions

- For a square matrix $A_{n\times n}$:
  - **A** is **positive-definite** if $x^T Ax > 0$ for all $x \neq 0$
    - A covariance matrix is a positive-definite matrix
  - **A** is **positive-semidefinite** if $x^T Ax \geq 0$ for all $x \neq 0$

Linear Function

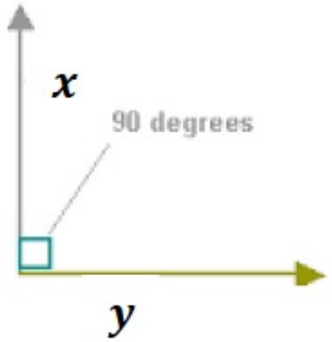
- Linear function
  - $f(\boldsymbol{x}+\boldsymbol{y}) = f(\boldsymbol{x}) + f(\boldsymbol{y}) \quad \forall \boldsymbol{x}, \boldsymbol{y} \in RV$
  - $f(a\boldsymbol{x}) = af(\boldsymbol{x}) \quad \forall \boldsymbol{x} \in RV, a \in \mathbb{R}$

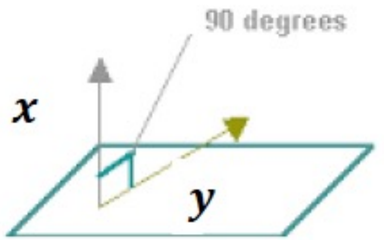
$$f(a\boldsymbol{x}+b\boldsymbol{y}) = af(\boldsymbol{x}) + bf(\boldsymbol{y}) \qquad \forall \boldsymbol{x}, \boldsymbol{y} \in RV,\ a, b \in \mathbb{R}$$

- A linear function can be express as:
  - $f(\boldsymbol{x}) = w_1x_1 + w_2x_2 + \cdots + w_nx_n = [w_1 \quad w_2 \quad \dots \quad w_n]\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \boldsymbol{w}^T\boldsymbol{x}$

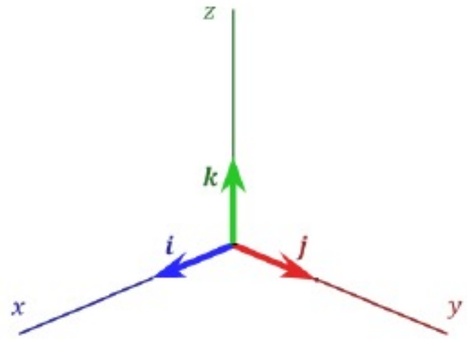
Orthogonal and Orthonormal Vectors

- **Orthogonal:** two vectors x and y are orthogonal if $x^T y = 0$





- **Orthonormal:** two vectors x and y are orthonormal $x^T y = 0$ and $|x| = |y| = 1$
  - **Orthogonal unit vectors**
  - Example: i, j and k, the basis in a Cartesian space



Orthonormal Transformation

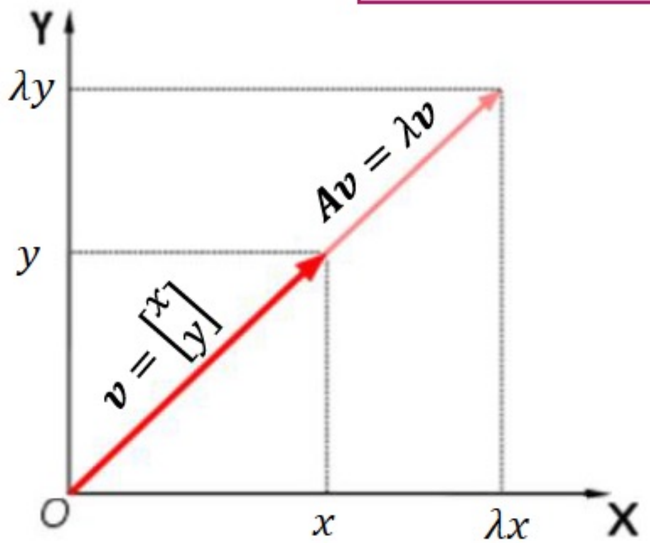
- **Orthonormal Transformation:** in $Y = AX$, a square matrix A is said to be orthonormal if $A^T A = AA^T = I$
  - It implies that $A^T = A^{-1}$
  - An orthonormal transformation has the property of preserving the magnitude of the vectors:

$$|y| = \sqrt{y^T y} = \sqrt{(Ax)^T (Ax)} = \sqrt{x^T A^T A x} = \sqrt{x^T x} = |x|$$

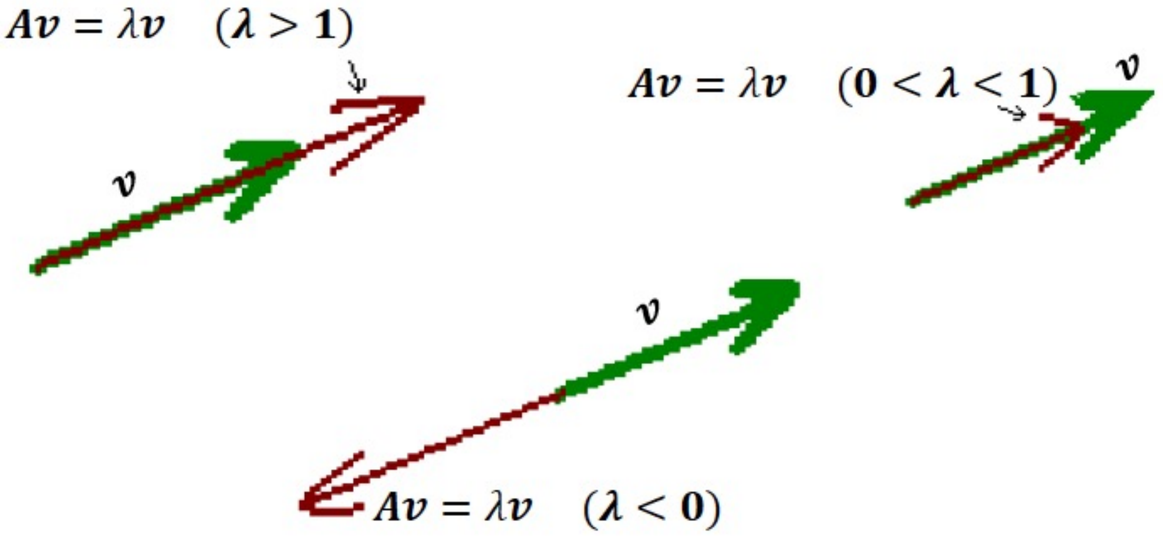
Eigenvalues and Eigenvectors

- Let A be an $n \times n$ matrix, the number (scalar) λ is the eigenvalue of A if there exist a non-zero vector $\boldsymbol{v}$ such that:

$$\boldsymbol{Av} = \boldsymbol{\lambda v} \quad \Leftrightarrow \quad \begin{cases} \lambda \text{ is an eigenvalue} \\ \boldsymbol{v} \text{ is the corresponding eigenvector} \end{cases}$$

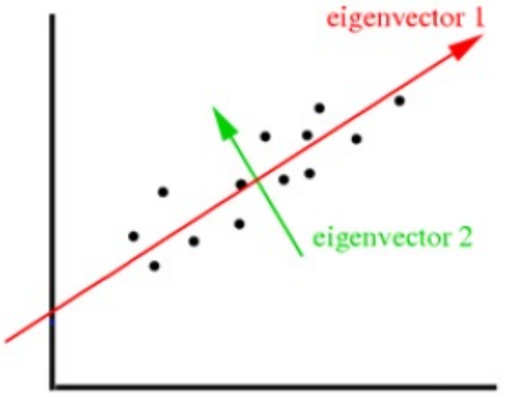


The vector $\boldsymbol{v}$ is an eigenvector of the matrix $\boldsymbol{A}$

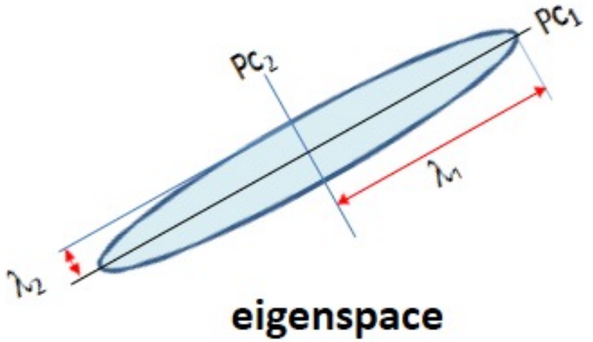


Eigenvectors of a Covariance Matrix

- A covariance Σ of a n-dimensional dataset is a square matrix $n \times n$ matrix



- The eigenvectors of Σ show the **directions in which the data varies the most**.
  - In other words, the largest **eigenvector** always points into the direction of the largest variance of the data.
  - The magnitude of this vector equals the corresponding **eigenvalue**.



eigenspace

- The eigenvectors of a covariance matrix are called the **Principle Component** (PC) of the dataset
  - They are orthogonal to each other

Computation of Eigenvalues and Eigenvectors

- Computation of the **eigenvalues** of A an $n \times n$ matrix:

$$Av = \lambda v \Rightarrow Av - \lambda v = 0 \Rightarrow (A - \lambda I)v = 0 \Rightarrow \begin{cases} v = 0 & \text{trivial solution} \\ (A - \lambda I) = 0 & \text{non-trivial solution} \end{cases}$$

$$(A - \lambda I) = 0 \Rightarrow \underbrace{|A - \lambda I|}_{\text{determinant}} = 0 \Rightarrow \underbrace{\lambda^n + a_1\lambda^{n-1} + \cdots a_{n-1}\lambda + a_0}_{\text{Characteristic Equation}} = 0$$

- Computation the **eigenvectors:**
  - The corresponding eigenvector of each eigenvalues is computed by substituting the eigenvalues in $\boldsymbol{Av} = \lambda \boldsymbol{v}$ and solving it

Example - Computation of Eigenvalues and Eigenvectors

Let $\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix}$

Computation of characteristic equation

$$|\mathbf{A} - \lambda \cdot \mathbf{I}| = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} = 0 \Rightarrow \begin{bmatrix} -\lambda & 1 \\ -2 & -3-\lambda \end{bmatrix} = \lambda^2 + 3\lambda + 2 = 0$$

Calculate eigenvalues

$$(\lambda + 2)(\lambda + 1) = 0$$

$$\lambda_1 = -1 \qquad \lambda_2 = -2$$

Find eigenvector v_1 associated with $\lambda_1 = -1$

$$(\mathbf{A} - \lambda_1) \cdot \mathbf{v}_1 = 0 \Leftarrow \mathbf{A} \cdot \mathbf{v}_1 = \lambda_1 \cdot \mathbf{v}_1$$

$$\Downarrow$$

$$\begin{bmatrix} -\lambda_1 & 1 \\ -2 & -3-\lambda_1 \end{bmatrix} \cdot \mathbf{v}_1 = 0$$

$$\Downarrow$$

$$\begin{bmatrix} 1 & 1 \\ -2 & -2 \end{bmatrix} \cdot \mathbf{v}_1 = \begin{bmatrix} 1 & 1 \\ -2 & -2 \end{bmatrix} \cdot \begin{bmatrix} v_{1,1} \\ v_{1,2} \end{bmatrix} = 0$$

$$\Downarrow$$

from the 1st row: $v_{1,1} + v_{1,2} = 0$

or from the 2nd row: $-2 \cdot v_{1,1} + -2 \cdot v_{1,2} = 0$

Eigenvector v_1

$$v_{1,1} = -v_{1,2}$$

Any 2-element column vector in which the two elements have equal magnitude and opposite sign

$$\mathbf{v}_1 = k_1 \begin{bmatrix} +1 \\ -1 \end{bmatrix}$$

k_1 is an arbitrary constant

eigenvector v_2

same procedure to find eigenvector v_2 associated with $\lambda_2 = -2$

...

Properties of Eigenvalues and Eigenvectors

- If A is non-singular (invertible)
  - All eigenvalues are non-zero

- If A is real and symmetric
  - All eigenvalues are real
  - The eigenvectors associated with distinct eigenvalues are orthogonal

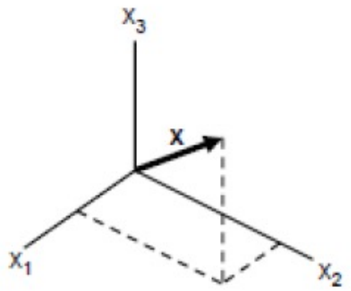
- If A is positive definite
  - All eigenvalues are positive $\forall i,\ \lambda_i > 0$

Introduction

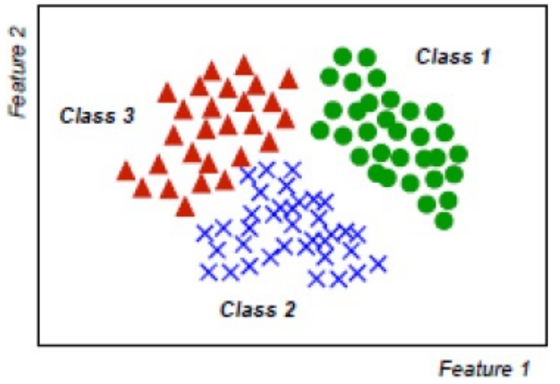
- **Feature or Attribute**
  - Feature is any distinctive aspect, quality or characteristic
- **Feature vector**
  - The combination of d features represented as a d-dimensional column vector
- **Feature space**
  - The d-dimensional space defined by the feature vector
  - Objects are represented as points in feature space. This representation is called a **Scatter plot**

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \\ x_d \end{bmatrix}$$

Feature vector



Feature space (3D)



Scatter plot (2D)

Introduction

- The complexity of a learning algorithms includes:
  - Time complexity
  - Space complexity
  - Samples complexity (speech, image, video, …)

- In most learning methods, these complexities depends on the number of feature dimensions (D) as well as the size of training set (N).

- As a rule of thumb, the number of training samples must be 10 to 20 times of the number of features

Why Dimensionality Reduction

- In most pattern recognition and machine leering methods, we are interested to reduce the feature dimension as a separate pre processing step. This is because:
  - Decreasing the time complexity
  - Decreasing the cost of collecting/extracting unnecessary features
  - Decreasing the space complexity
  - Having simpler description and visualization of the learning algorithms
    - Humans are usually interested in 1D, 2D and 3D datasets.
  - Developing a **simpler model** for the problem.

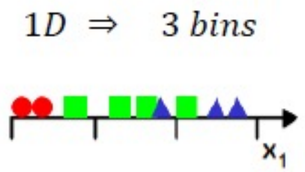
Why Dimensionality Reduction

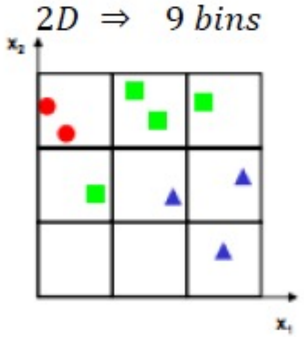
- Developing a **simpler model** for the problem.
  - A simpler model is less prone to overfitting problem (and has less variance/ is more robust).
  - A simpler model has more generalization on unseen samples.
  - A simpler model needs smaller number of training samples to be learned.
  - A simpler model is less sensitive to noise and outliers.

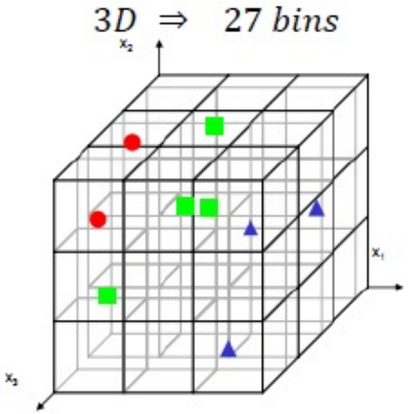
Cures of Dimensionality

- **Cures of dimensionality**
  - Refers to the problems associated with multivariate data analysis as the dimensionality increases

- For example, suppose we want to develop a histogram with 3 bins at each axis:







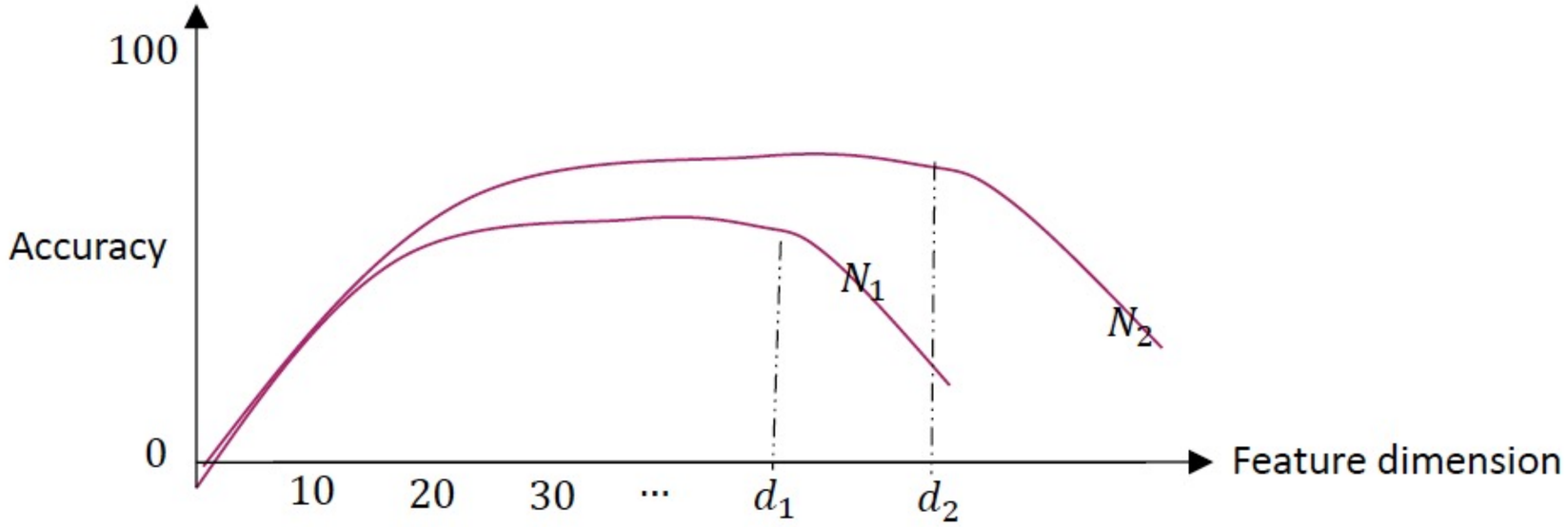
Most bins are sparse or even empty!

- Usually, increasing dimensionality:
  - Exponentially increases the required training examples.
  - Increase the time/space complexity
  - May need more complex function to model the problem ⇒ complex function also needs denser sample points
  - …

Peaking Phenomena

Peaking Phenomena:

- In practice, for a given finite number of samples (N), there is a maximum number of features above which the performance of our classifier will degrade rather than improve



- If the number of samples increases ($N_2 \gg N_1$), the peaking phenomenon occurs for larger number of features ($d_2 \gg d_1$).

Dimensionality Reduction Methods

- **Feature selection:**
  - These methods find a new feature set of d $(d < D)$ by selecting d features from existing features

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix} \longrightarrow \begin{bmatrix} x_{i_1} \\ x_{i_2} \\ \vdots \\ x_{i_d} \end{bmatrix}$$

feature selection

- **Feature extraction:**
  - These methods select d $(d < D)$ features out of D dimensions and discard $D - d$ dimensions.
    - Given a feature space $\boldsymbol{x} \in \mathbb{R}^D$ find a mapping $\boldsymbol{y} = f(\boldsymbol{x}) : \mathbb{R}^D \rightarrow \mathbb{R}^d$ with $d < D$ such that transformed feature vector $\boldsymbol{y} \in \mathbb{R}^d$ optimises an objective function

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix} \xrightarrow[f : \mathbb{R}^D \rightarrow \mathbb{R}^d]{\text{feature extraction}} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_d \end{bmatrix} = f\left(\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix}\right)$$

Feature Extraction

Feature Extraction Methods

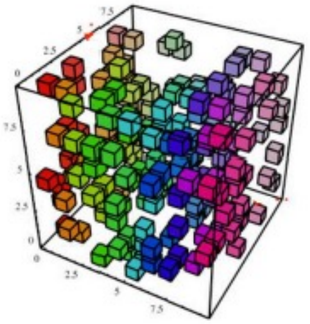
- **Feature extraction:**
  - These methods select d $(d < D)$ features out of D dimensions and discard $D - d$ dimensions.
    - Given a feature space $\boldsymbol{x} \in \boldsymbol{R^D}$ find a mapping $\boldsymbol{y} = f(\boldsymbol{x}) : \mathbb{R}^D \rightarrow \mathbb{R}^d$ with $d < D$ such that transformed feature vector $\boldsymbol{y} \in \mathbb{R}^d$ optimises an objective function

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix} \xrightarrow{\text{feature extraction}} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_d \end{bmatrix} = f\left(\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix}\right)$$

- Feature extraction methods can be:
  - Supervised / Unsupervised
  - Linear / Non-Linear

Linear Feature Extraction Methods

- The mapping $\boldsymbol{y} = f(\boldsymbol{x}) : \mathbb{R}^D \rightarrow \mathbb{R}^d$ can be either a linear or a non-linear function
- We focus on linear mapping $\boldsymbol{y} = W\boldsymbol{x}$, commonly used in different applications

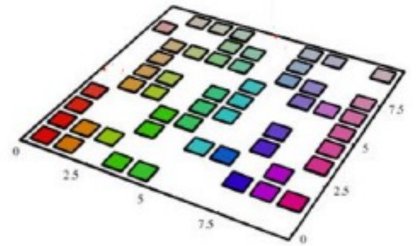


$\boldsymbol{x}$: inpute samples in a 3D feature space

Linear Feature Extraction

$$\boldsymbol{y} = W\boldsymbol{x}$$

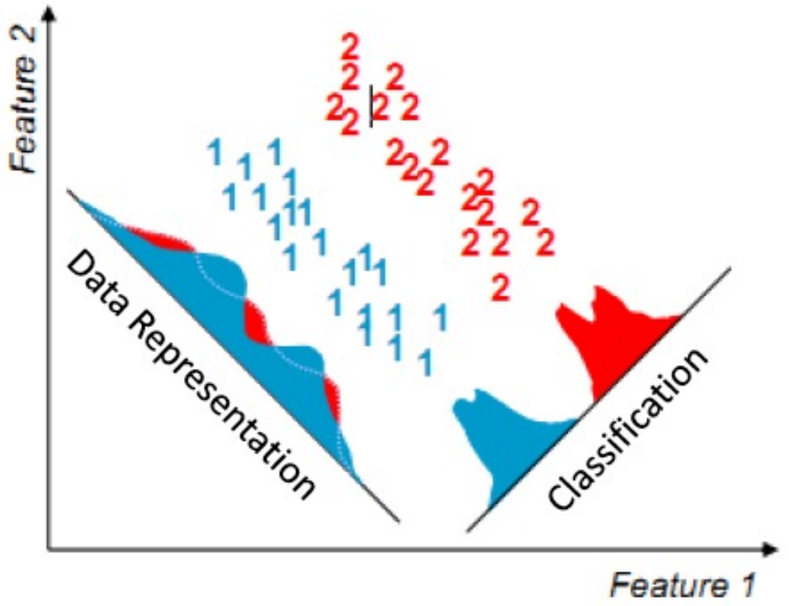
$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_d \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & \cdots & & w_{1D} \\ w_{21} & w_{22} & \cdots & & w_{1D} \\ \vdots & \vdots & & \ddots & \vdots \\ w_{d1} & w_{d2} & \cdots & \cdots & w_{dD} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \\ \\ x_D \end{bmatrix}$$



$\boldsymbol{y}$: transformed Samples in a 2D feature space

Data Representations vs. Classification

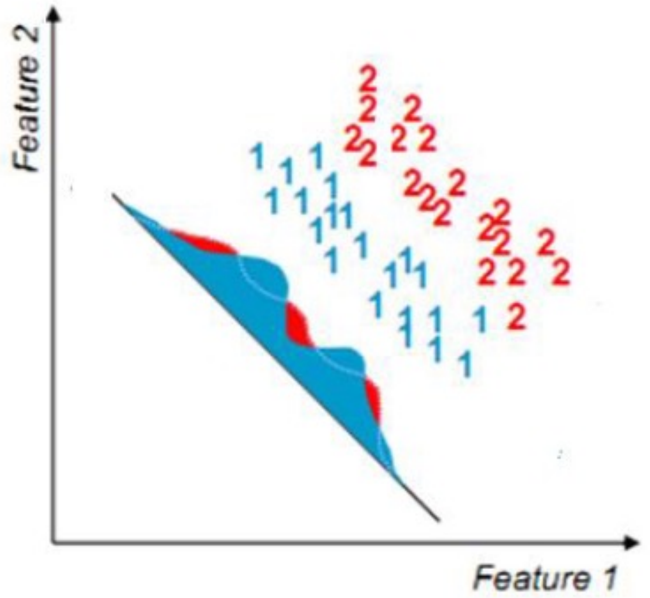
- The selection of the feature extraction mapping $\boldsymbol{y} = f(\boldsymbol{x})$ is guided by an objective function that we seek to maximize (or minimize)
- Depending on the objective function, feature extraction methods can be grouped into:
  - **Data Representation:** The goal of feature extraction mapping is to represent samples accurately in a lower-dimensional space
    - Here, we study Principal Components Analysis (PCA)
  - **Classification:** The goal of feature extraction mapping is to improve class-discriminatory information in a lower-dimensional space
    - Here, we study Linear Discriminant Analysis (LDA)



Principle Components Analysis (PCA)

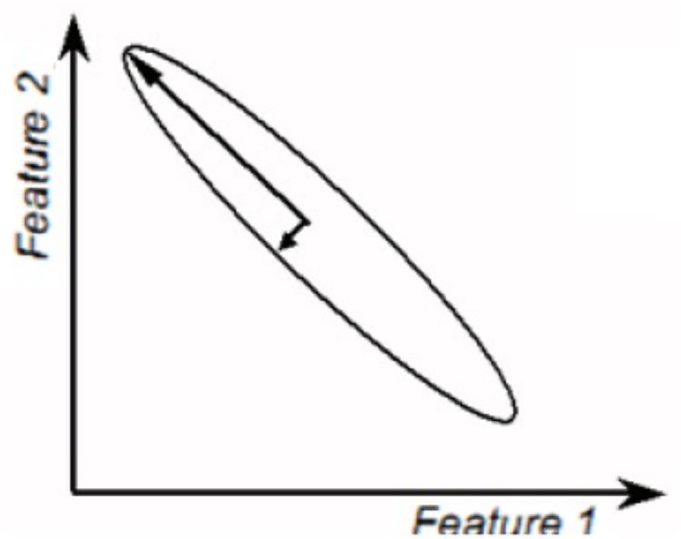
- Main idea:
  - Seek most accurate data representation in a lower-dimensional space
  - For this, PCA performs dimensionality reduction while **preserving largest variances in the data**

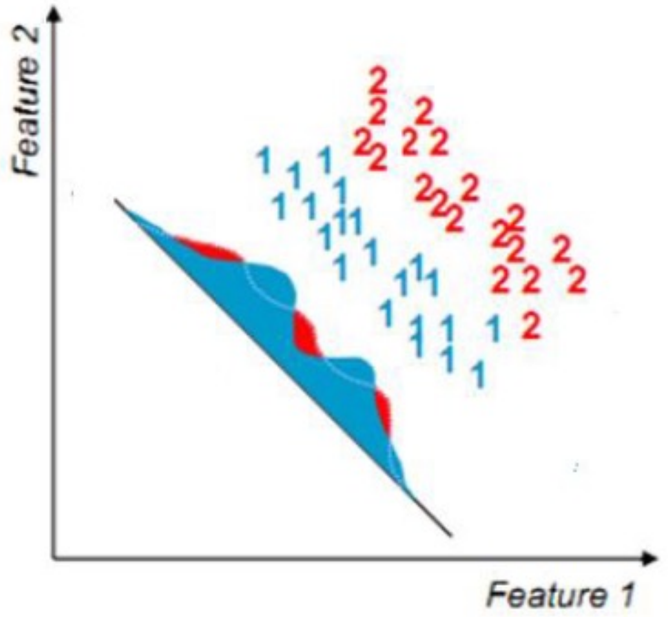
- Example in a 2D problem:
  - Project data to 1D subspace (a line) which maximize the largest variances
  - Note that the good line minimize the projection error



Principle Components Analysis (PCA)

- How find the direction of largest variance in data?
  - If x has multivariate Gaussian distribution N(μ, Σ), direction of largest variance is given by eigenvector corresponding to the largest eigenvalue of Σ





PCA goal

- PCA tries to find $W_{D\times d}$ such that the mapping results losses minimum information in lower-dimension:

$$Y = W^T X$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_d \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1D} \\ w_{21} & w_{22} & \cdots & w_{1D} \\ \vdots & \vdots & \ddots & \vdots \\ w_{d1} & w_{d2} & \cdots & w_{dD} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix} = \begin{bmatrix} & \leftarrow & \boldsymbol{w_1^T} & \rightarrow & \\ & \leftarrow & \boldsymbol{w_2^T} & \rightarrow & \\ \vdots & \vdots & \ddots & & \vdots \\ & \leftarrow & \boldsymbol{w_d^T} & \rightarrow & \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix}$$

- Equivalently, PCA tries to **maximize the variances over each transformed element** y_k, i.e., maximize $Var(y_i)$

PCA Derivation

- First, let consider w_1 a principle component of mapping function W. We have:

$$y_1 = w_1^T \boldsymbol{x}$$

- PCA tries to find the principle component w_1 such that the sample after projection onto w_1 is most spread out (maximum variance), i.e. the variance in the projected sample points becomes maximum:

$$w_1 = \underset{w_1}{\operatorname{argmax}}[Var(y_1)]$$

- Besides, in order to have a unique solution, we find the solution with $||w_1|| = 1$
- Consequently, the problem becomes:

$$w_1 = \underset{w_1}{\operatorname{argmax}}[Var(y_1)] \qquad \text{subject to } w_1^T w_1 = 1$$

- Therefore, we must solve the following Lagrange problem:

$$w_1 = \underset{w_1}{\operatorname{argmax}}[Var(y_1) - \alpha(w_1^T w_1 - 1)]$$

PCA Derivation

- Now, let calculate the $Var(\mathrm{y}_1)$:

$$Var(\mathrm{y}_1) = \mathrm{E}\left[\left(y_1 - \mu_{y_1}\right)^2\right] \qquad \Sigma_x = E[(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^T]$$

$$\xrightarrow{y_1 = w_1^T \boldsymbol{x}} = E[(w_1^T \boldsymbol{x} - w_1^T \boldsymbol{\mu})(\boldsymbol{x}^T w_1 - \boldsymbol{\mu}^T w_1))]$$

$$= E[w_1^T (\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^T w_1]$$

$$= w_1^T \, E[(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^T] w_1$$

$$= w_1^T \Sigma \, w_1$$

- Therefore, the problem become:

$$w_1 = \underset{w_1}{\mathrm{argmax}} [\underbrace{w_1^T \Sigma \, w_1 - \alpha(w_1^T w_1 - 1)}_{\mathrm{J}(w_1)}]$$

PCA Derivation

- The maxima of $J(w_1)$ is computed by the taking derivative with respect to w_1 and setting it equal to 0:

$$\frac{\partial J(w_1)}{\partial w_1} = \frac{\partial \left(w_1^T \Sigma\, w_1 - \alpha(w_1^T w_1 - 1) \right)}{\partial w_1} = 0$$

- Note: $\frac{\partial (x^T A x)}{\partial x} = (A + A^T)x \xrightarrow{if\ A\ is\ symetric} = 2Ax$

$$2\,\Sigma\,\mathrm{w}_1 - 2\alpha w_1 = 0 \qquad \Rightarrow \qquad \Sigma w_1 = \alpha w_1$$

- Hence, w_1 is an eigenvector of Σ and α its corresponding eigenvalue.

- Therefore, to maximize $Var(y_1)$, we must choose eigenvector w_1 of Σ with largest eigenvalue $\alpha = \lambda_1$

PCA Derivation

- To find the second principle component, we should find w_2 such that:
  - Maximize variance,
  - Be unit length, and
  - w_2 be orthogonal to w_1 (y_1 and y_2 must be uncorrelated)
- In other words, the problem to find the second principle component becomes:

$$w_2 = \underset{w_2}{\operatorname{argmax}}[Var(y_2)] \qquad \text{subject to } w_2^T w_2 = 1 \text{ and } w_2^T w_1 = 0$$

- Therefore, we must solve the following Lagrange problem:

$$w_2 = \underset{w_2}{\operatorname{argmax}}[w_2^T \Sigma\, w_2 - \alpha(w_2^T w_2 - 1) - \beta(w_2^T w_1 - 0)]$$

- By taking derivative with respect to w_2 and setting it equal to 0:

$$2\Sigma w_2 - 2\alpha w_2 - \beta w_1 = 0$$

- Multiplying by w_1^T:

$$2w_1^T \Sigma w_2 - 2\alpha w_1^T w_2 - \beta w_1^T w_1 = 0$$

- We know $w_1^T w_2 = 0$ and $w_1^T w_1$=1, so:

$$2w_1^T \Sigma w_2 - 0 - \beta = 0 \qquad \Rightarrow \qquad \beta = 2w_1^T \Sigma w_2$$

PCA Derivation

- Also since $w_1^T \Sigma w_2$ is a scaler we have $w_1^T \Sigma w_2 = (w_1^T \Sigma w_2)^T$:

$$\beta = 2w_1^T \Sigma w_2 = 2w_2^T \Sigma w_1 \xrightarrow{\Sigma w_1 = \lambda_1 w_1} \beta = 2\lambda_1 w_2^T w_1 = 0$$

- Therefore:

$$2\Sigma w_2 - 2\alpha w_2 - \beta w_1 = 0 \xrightarrow{\beta = 0} \Sigma w_2 = \alpha w_2$$

- This implies that w_2 should be the eigenvector of Σ with the second largest eigenvalue $\alpha = \lambda_2$

- Similarly, it can be shown that the other components are given by the eigenvectors with decreasing eigenvalues.

- Recall that:
  - If Σ is positive definite ($x^T \Sigma x > 0$ for all vector x), then all its eigenvalues are positive.
  - If Σ is singular, Σ is called semi-positive definite ($x^T \Sigma x \geq 0$), and one or more eigenvalues of Σ are zero.

PCA Algorithm

- To maximize variances, PCA form the mapping function $W_{D\times d}$ by choosing d $(d < D)$ eigenvectors of Σ with largest eigenvalues (each column corresponds to an eigenvector)
  - value of eigenvalues gives importance of each component

PCA Algorithm

- **Input:** Data $X_{D\times N}$ (each column a D-dim sample)

1. $\boldsymbol{\mu}$ = sample mean of X, Σ = sample covariance of X
2. $\bar{X}$ = subtract sample mean $\boldsymbol{\mu}$ from each column sample ($\bar{X}$ has zero mean)
3. Find eigenvectors and eigenvalues of Σ
4. W = Using d ($d < D$) eigenvectors with largest eigenvalues form the mapping function as a $D \times d$ matrix, each column corresponds to an eigenvector
   - value of eigenvalues gives importance of each component
5. The transformed samples will be $Y = W^T \bar{X}$

- **Output:** Transformed data $Y_{d\times N}$ (each column a d-dim sample)

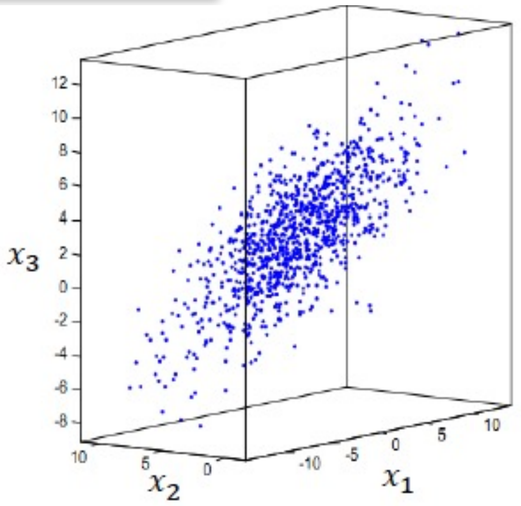
PCA Applications

- Dimensionality reduction and feature extraction/reduction
  - **A pre-processing step for other learning algorithms**
  - Such as face detection and face recognition (see Eigenface (Truk & Pentland 1991))
- Data compression
  - Such as image compression
- Data representation and visualization
  - Plotting data in low dimension
- Data denoising

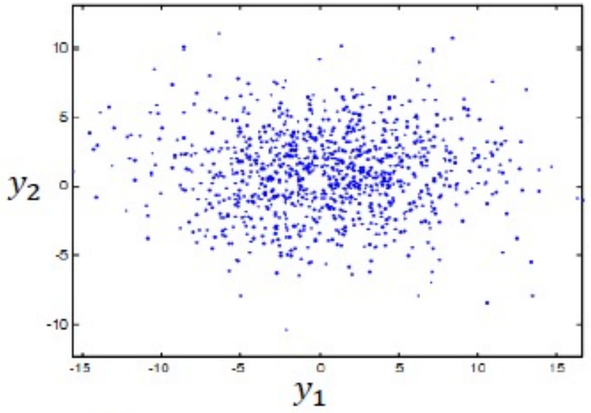
Example 1: PCA

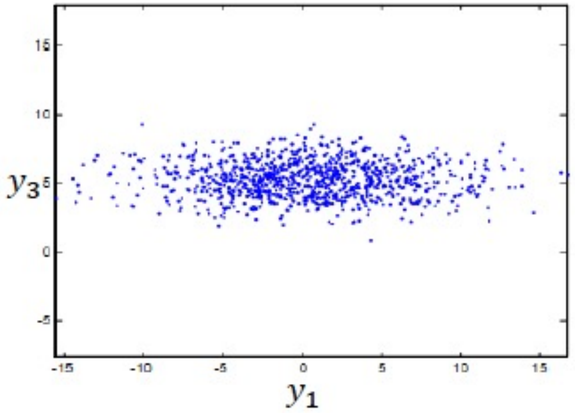
- Consider a three-dimensional Gaussian distribution with:

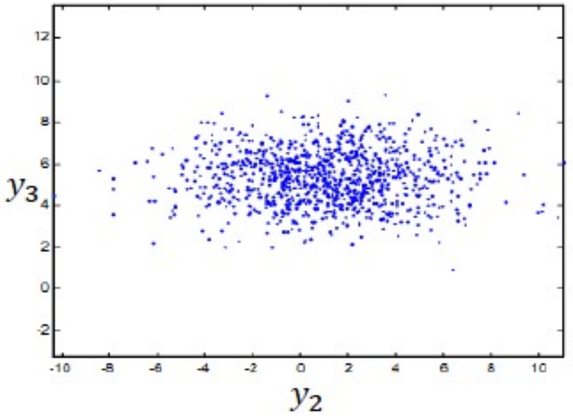
$$\mu = [0 \;\; 5 \;\; 2]^T \quad , \quad \Sigma = \begin{bmatrix} 25 & -1 & 7 \\ -1 & 4 & -4 \\ 7 & -4 & 10 \end{bmatrix}$$



- The three possible pairs of principle components are:



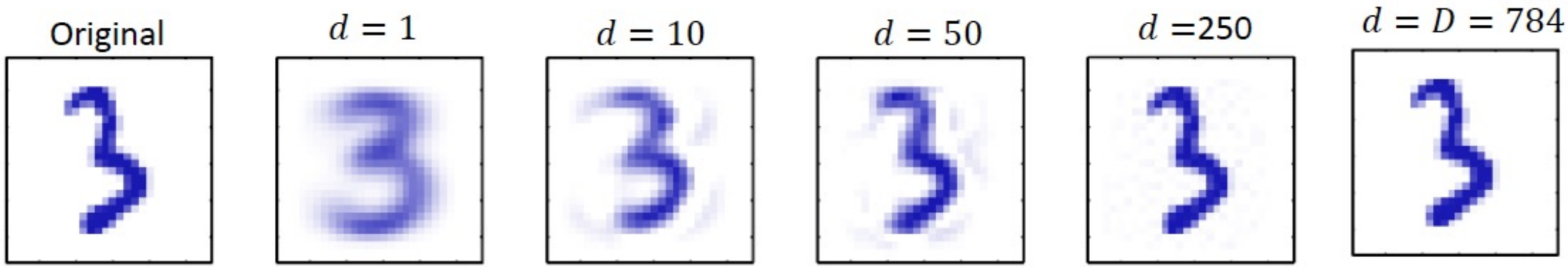




- Note that:
  - The first projection has the largest variance and the last projection has the smallest variance
  - The PCA projection de-correlates the axis

Example 2: PCA for Data Compression

- Consider a dataset consist of digits images. Image size $28 \times 28 = 784$
- Each image is vectorized as a column sample 784×1
- PCA reconstruction using different d.



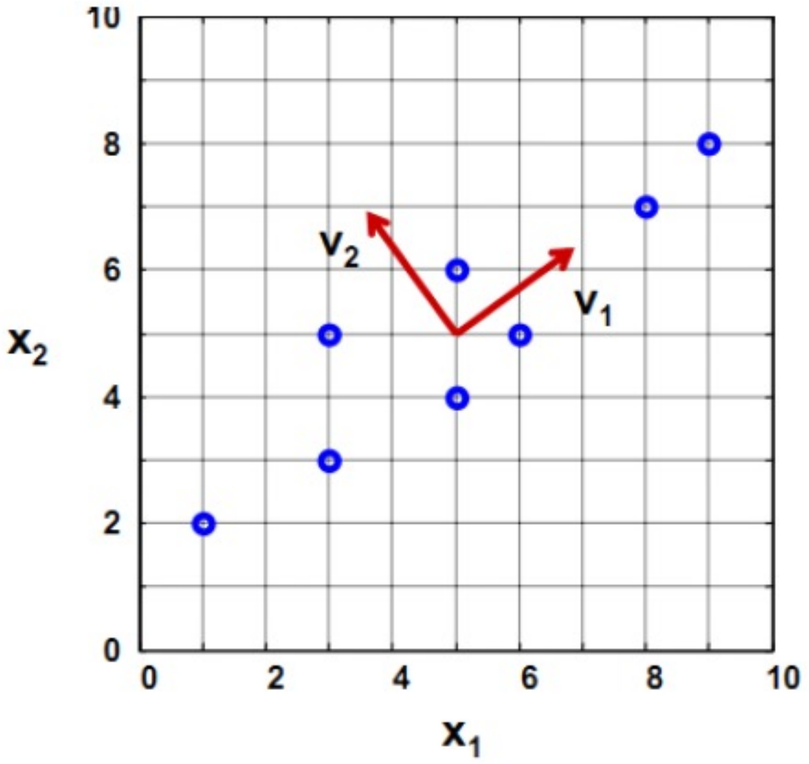
- As d increases the reconstruction becomes more accurate
- Reconstruction is perfect when $d = D = 784$

Exercise: PCA

- Compute the principal components for the following two-dimensional dataset:

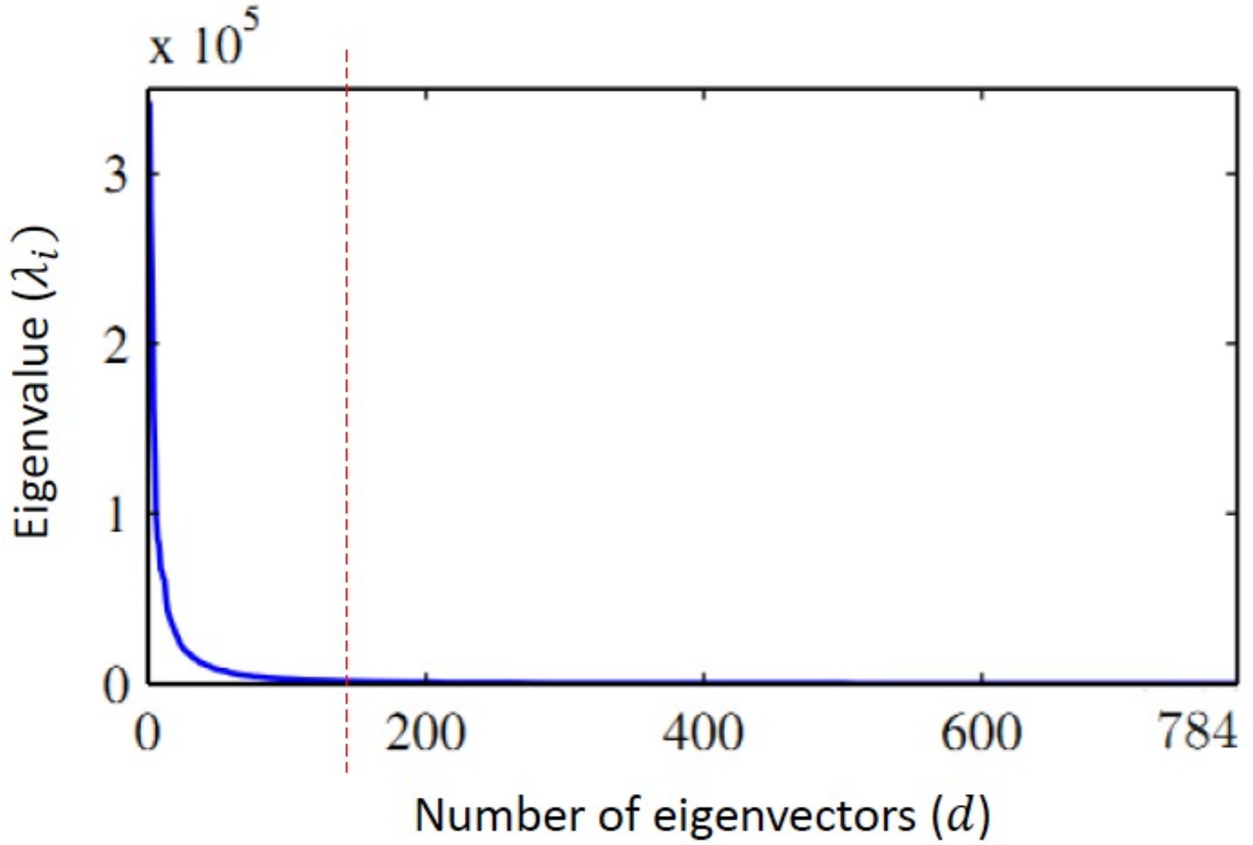
$$X = (x1, x2) = \{(1,2), (3,3), (3,5), (5,4), (5,6), (6,5), (8,7), (9,8)\}$$

- Calculate the projection of data into first eigenvector
- Calculate the projection of data into second eigenvector



PCA: how to select d

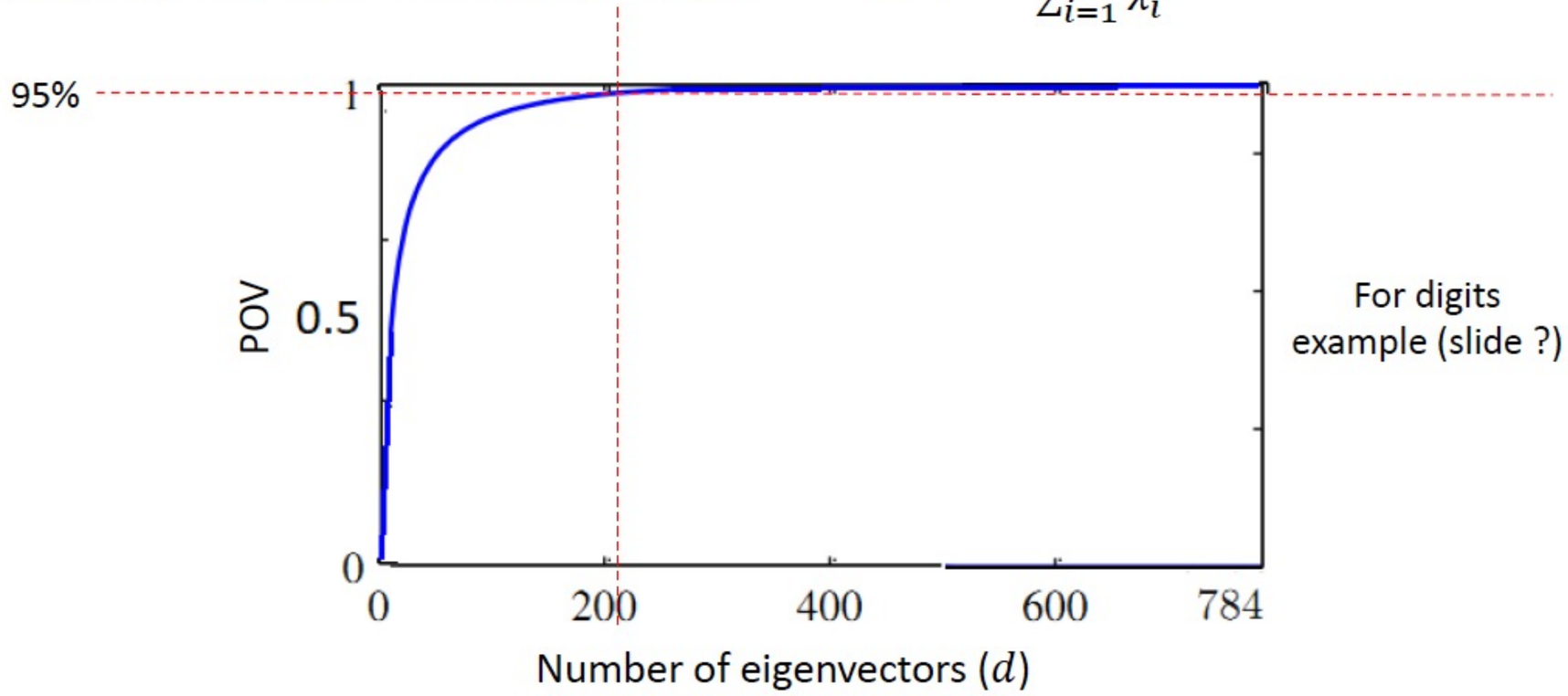
- Method1: Select eigenvector with large eigenvalue and discard the ones with small eigenvalues



For digits example (slide ?)

PCA: how to select d

- Method2: select d components that explain more than for example 95% of the variance.
- The proportion of variance (POV) is: $POV = \frac{\sum_{i=1}^{d} \lambda_i}{\sum_{i=1}^{D} \lambda_i}$



For digits example (slide ?)

Eigenface

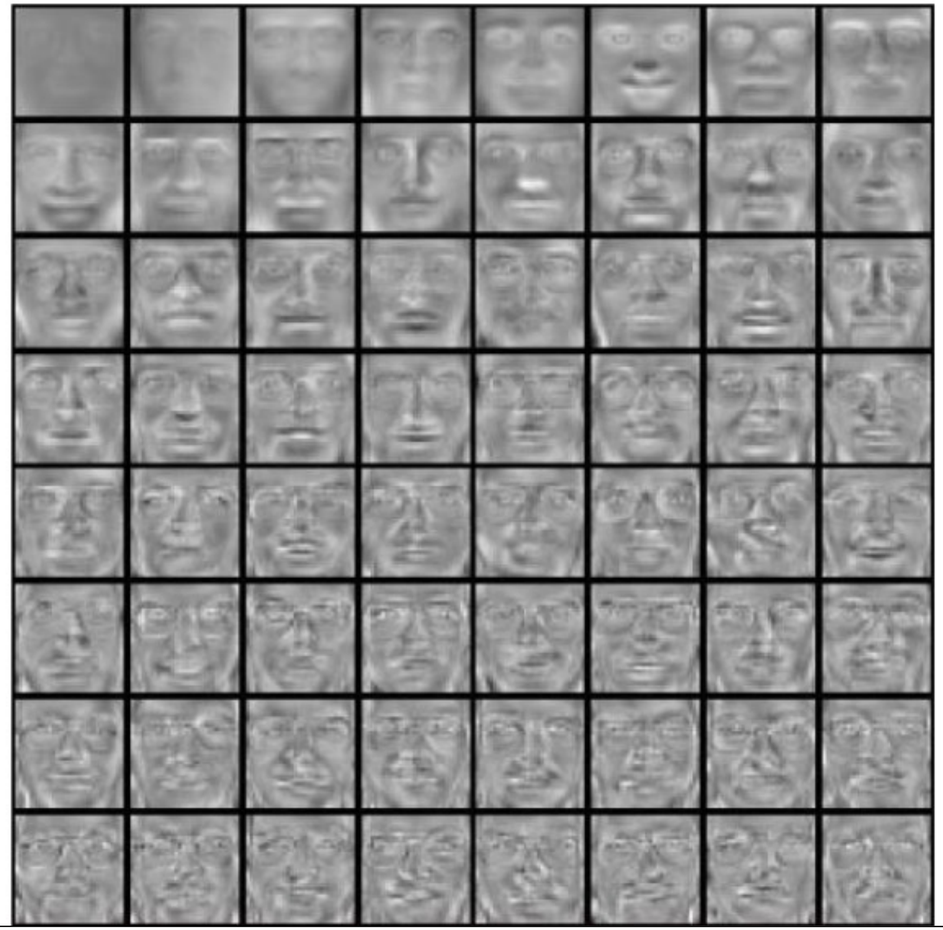
- Training image for face recognition application

- Compute covariance matrix of face images
- Compute the principal components **(Eigenfaces)**
- Store K eigenvectors with largest eigenvalues
- Represent all face images in the dataset as linear combinations of eigenfaces
- Perform nearest neighbor on these coefficients

Eigenface

Top eigenvectors: $u_1,\ldots u_k$

Mean: μ

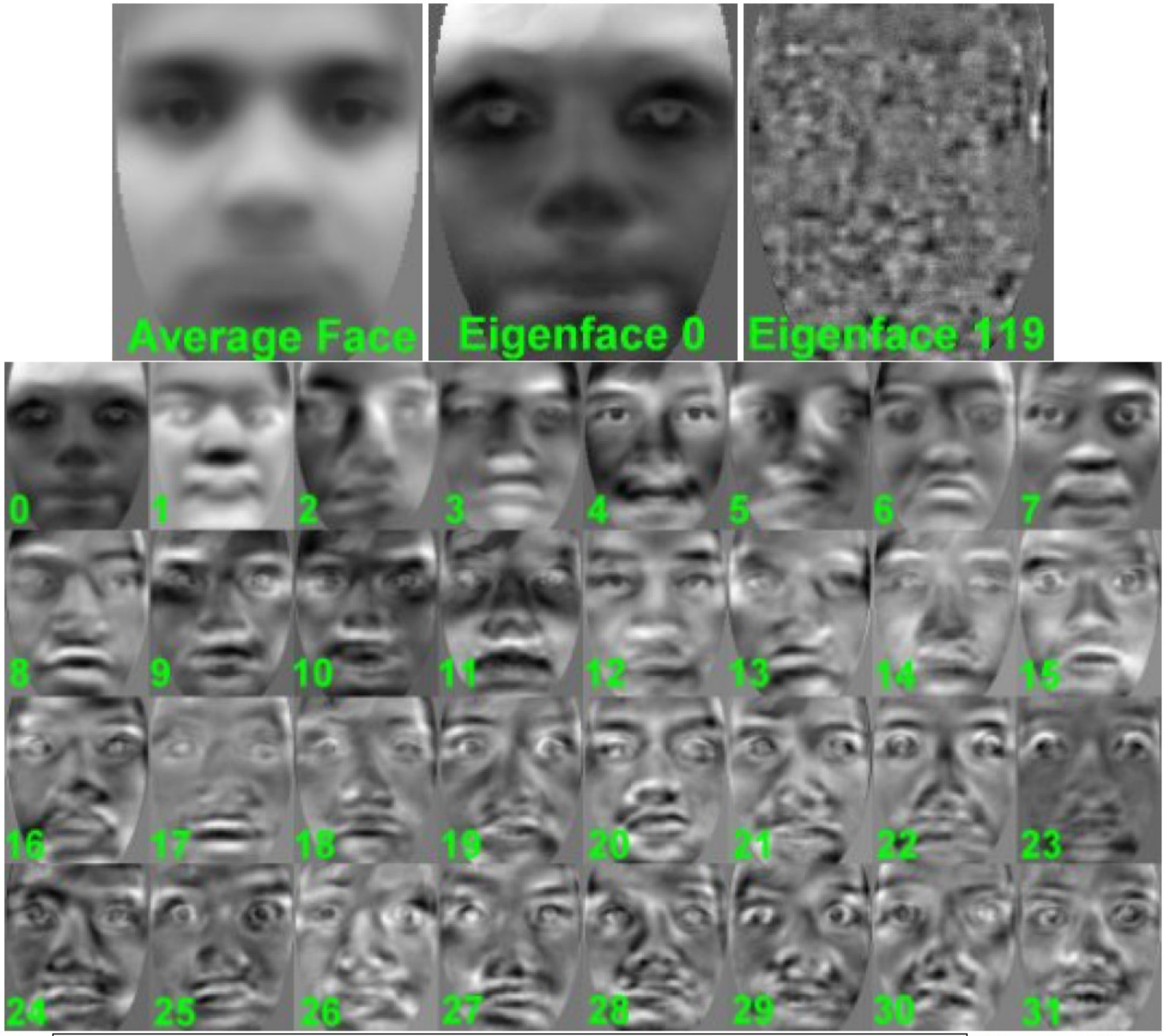


Reconstruction & Classification

$$\hat{x} = \mu + w_1u_1+w_2u_2+w_3u_3+w_4u_4+ \ldots$$

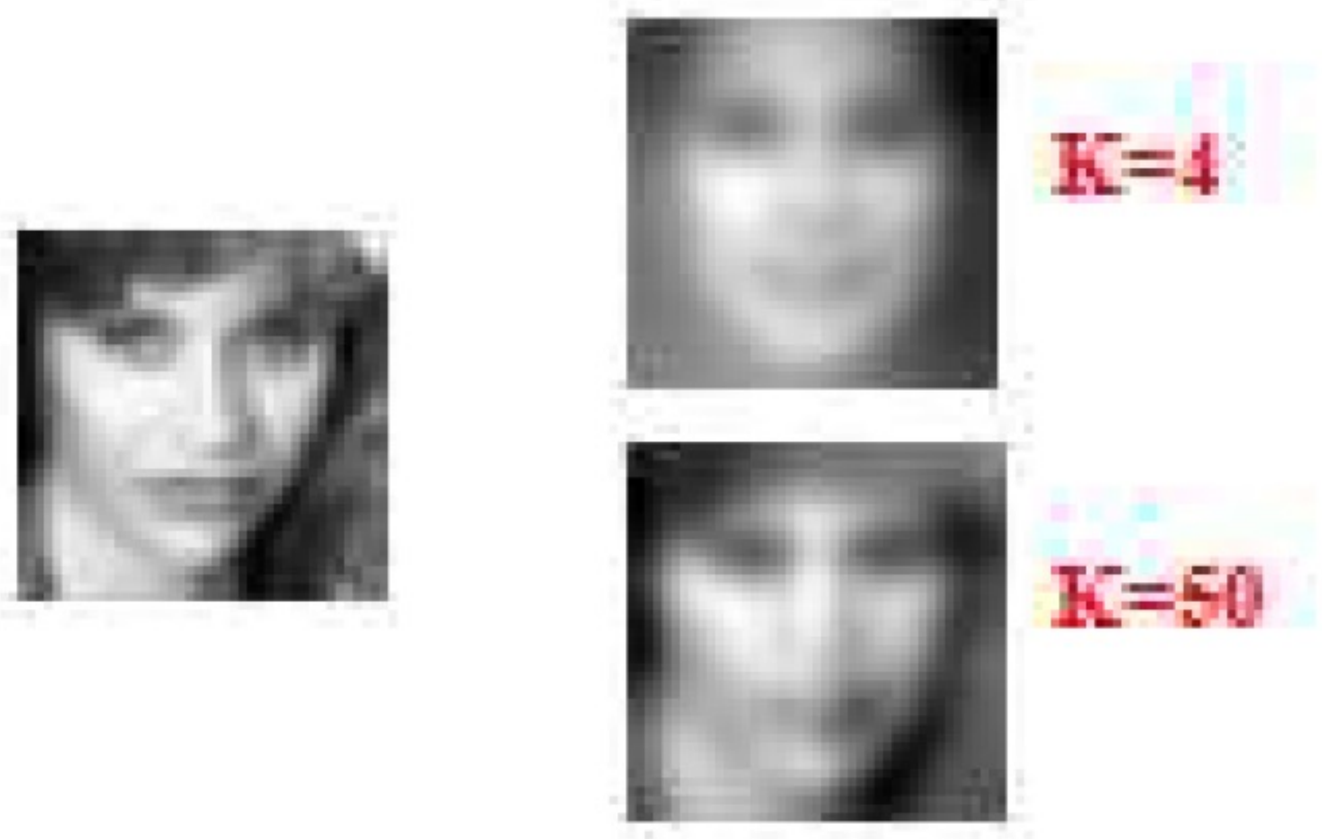
- Use vector $W = \{w_1, w_2, \ldots, w_n\}$ as the new feature vector and do the classification

Examples



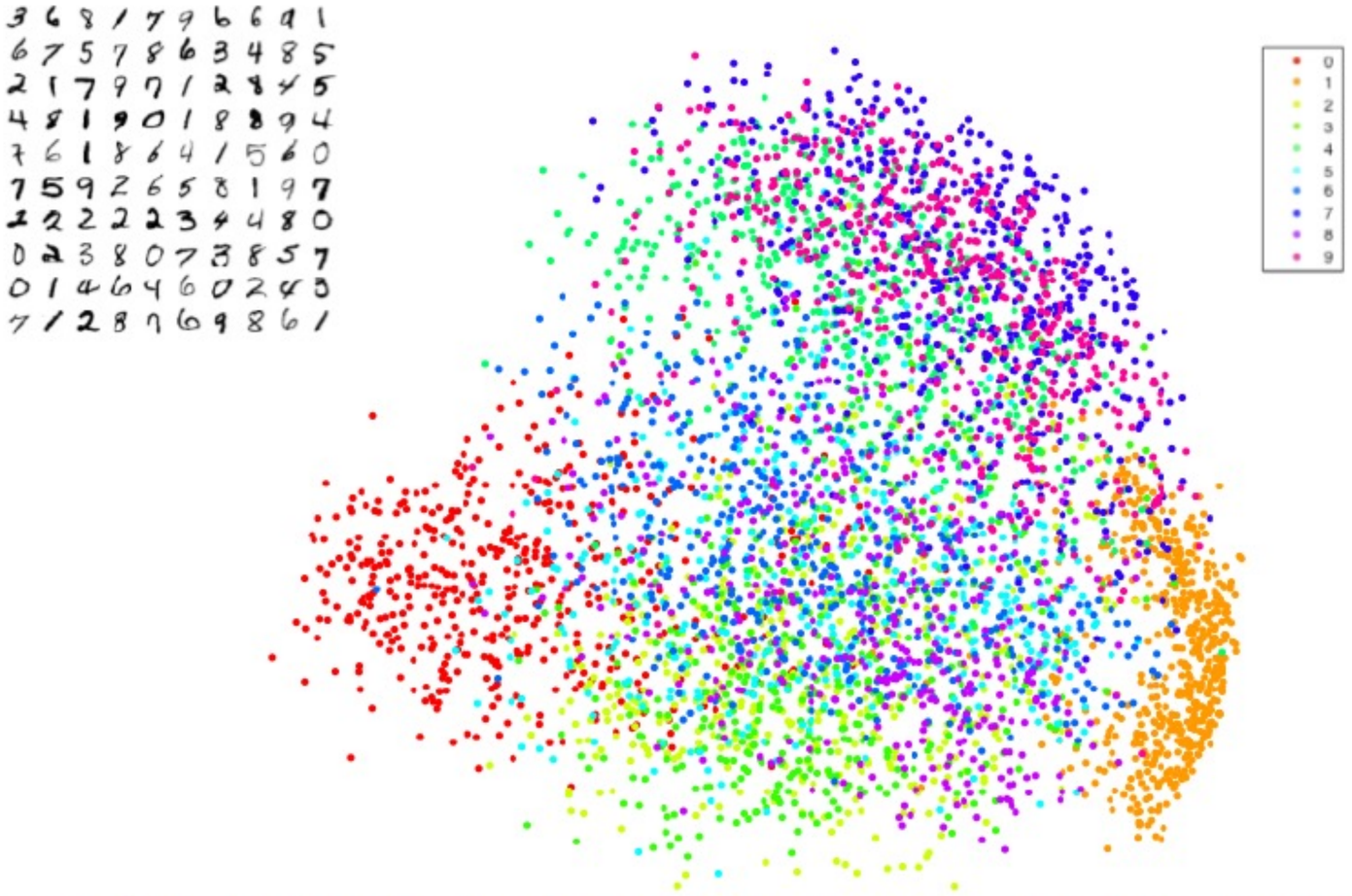
facerecaverage.jpg, facereceigen0.jpg, facereceigen119.jpg, facereceigen.jpg [Images of the the eigenfaces. The "average face" is the mean used to center the data.]

Examples



eigenfaceproject.pdf [Images of a face (left) projected onto the first 4 and 50 eigenvectors, with the average face added back. These last image is blurry but good enough for face recognition.]

Examples



pcadigits.pdf [The (high-dimensional) MNIST digits projected to 2D (from 784D). Two dimensions aren't enough to fully separate the digits, but observe that the digits 0 (red) and 1 (orange) are well on their way to being separated.]

Examples

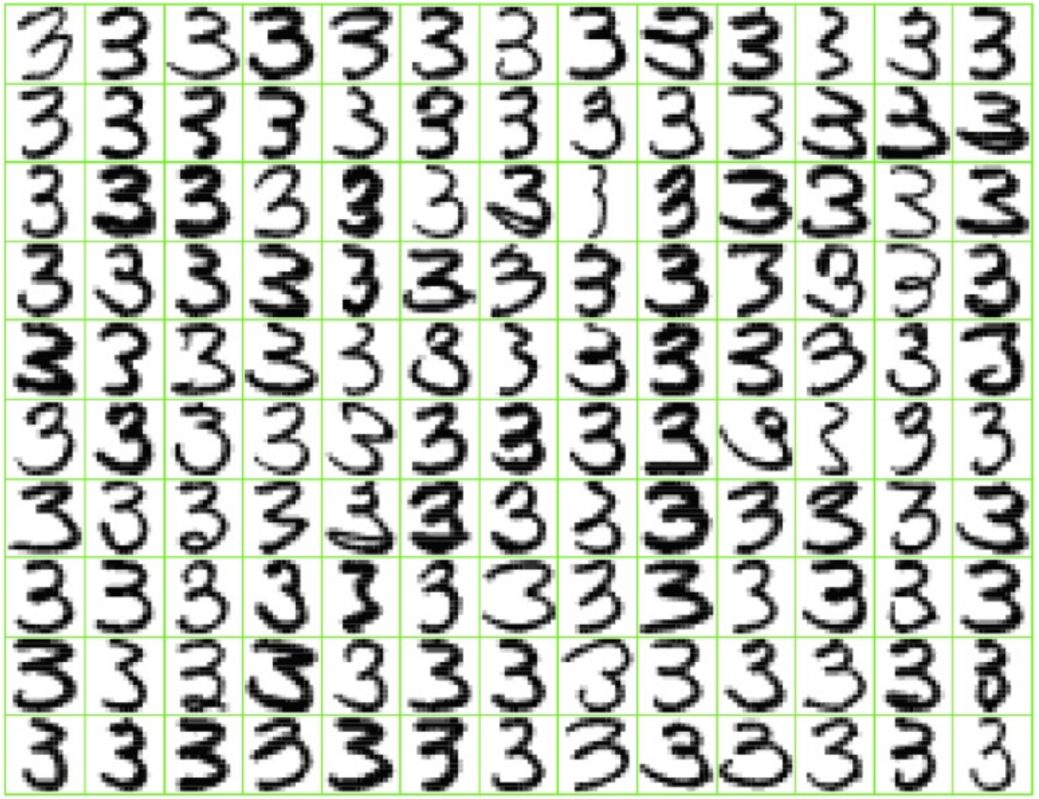


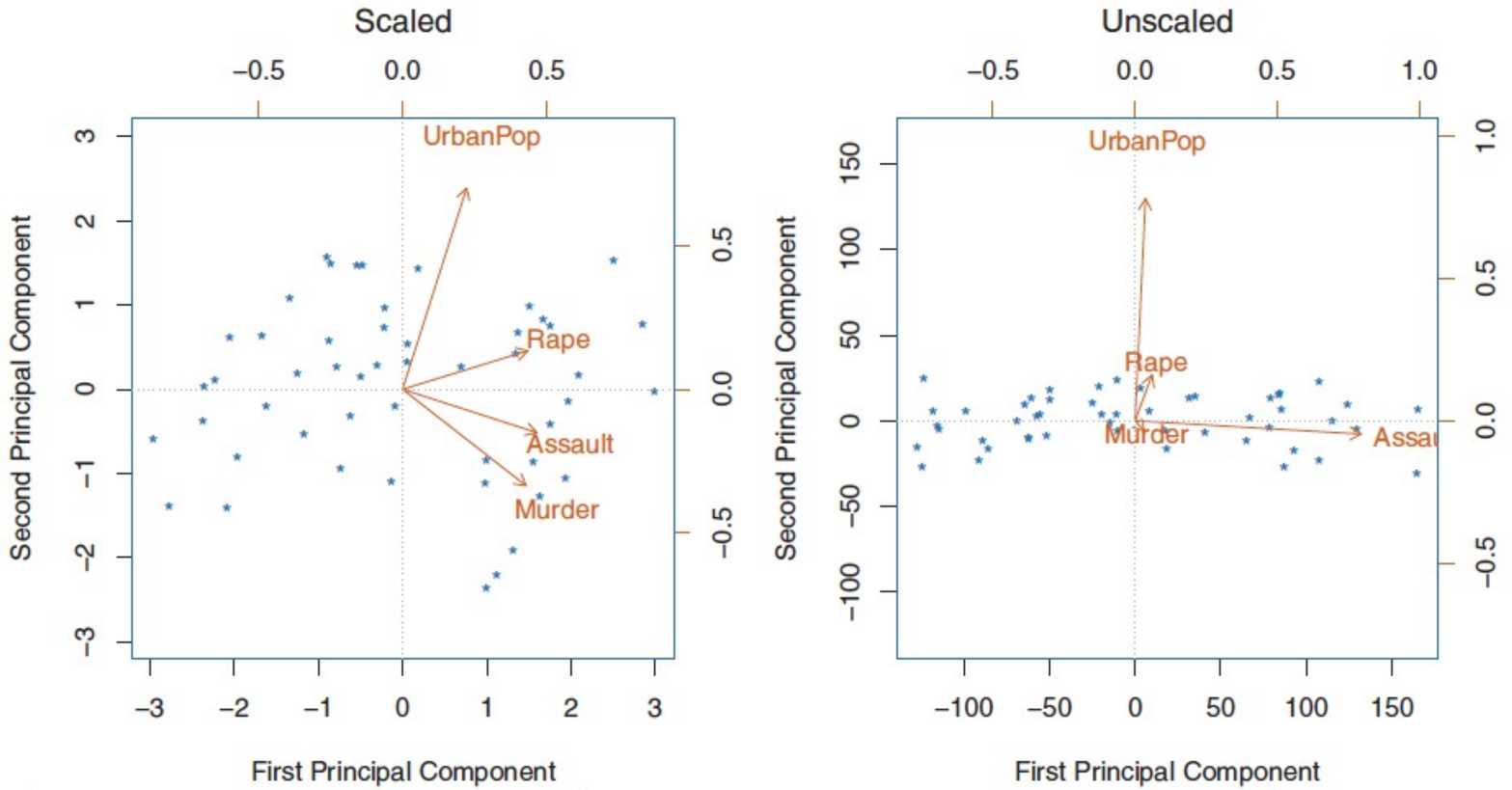
FIGURE 14.22. *A sample of* 130 *handwritten 3's shows a variety of writing styles.*

$$\hat{f}(\lambda) = \bar{x} + \lambda_1 v_1 + \lambda_2 v_2$$

$$= \quad \text{3} + \lambda_1 \cdot \text{3} + \lambda_2 \cdot \text{3}. \qquad (14.55)$$

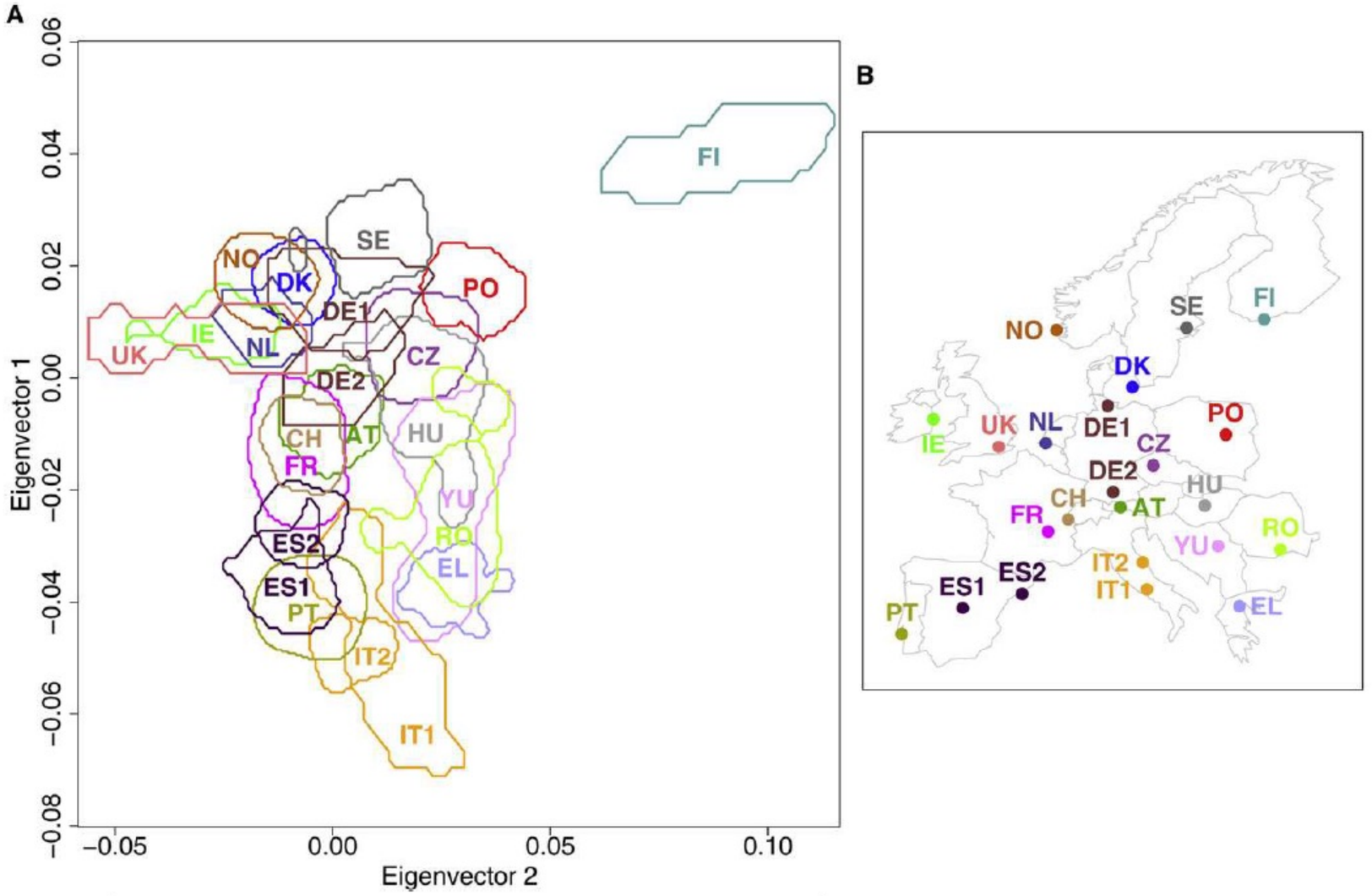
Figure 4:

Examples



normalize.pdf (ISL, Figure 10.3) [Projection of 4D data onto a 2D subspace. Each point represents one metropolitan area. Normalized data at left; unnormalized data at right. The arrows show the four original axes projected on the two principal components. When the data are not normalized, rare occurrences like murder have little influence on the principal directions. Which is better? It depends on whether you think that low-frequency events like murder and rape should have a disproportionate influence.]

Examples

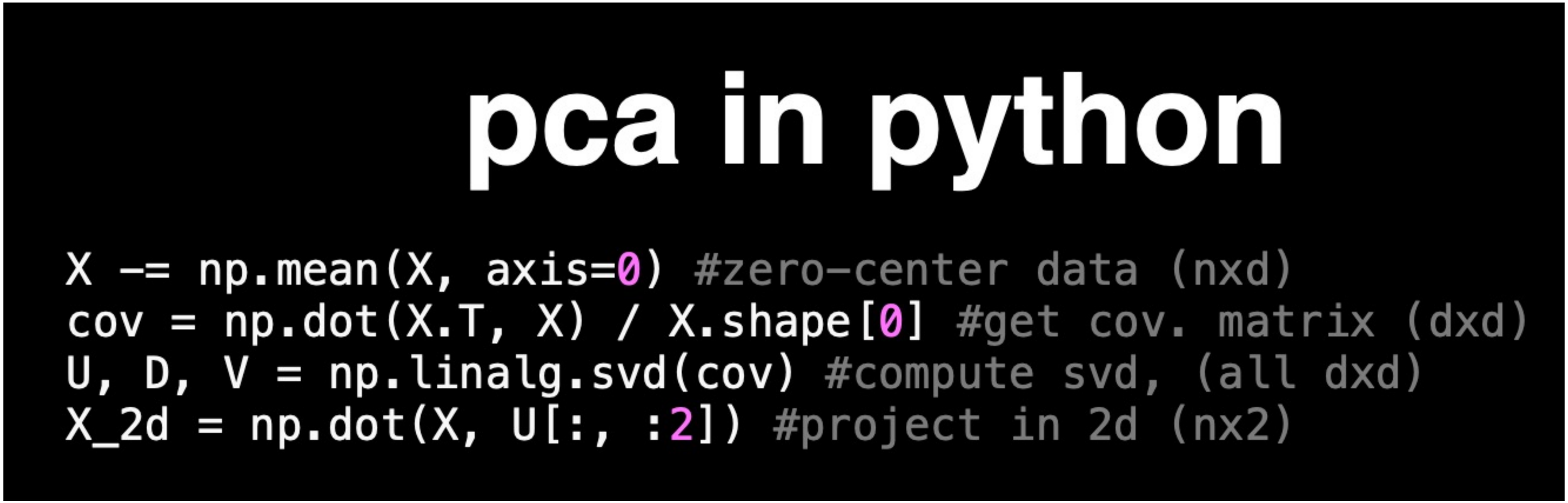


europegenetics.pdf (Lao et al., Current Biology, 2008.) [Illustration of the first two principal components of the single nucleotide polymorphism (SNP) matrix for the genes of various Europeans. The input matrix has 2,541 people from these locations in Europe (right), and 309,790 SNPs. Each SNP is binary, so think of it as 309,790 dimensions of zero or one. The output (left) shows spots on the first two principal components where there was a high density of projected people from a particular national type. What's amazing about this is how closely the projected genotypes resemble the geography of Europe.]

Summary

- PCA is an unsupervised (does not need label information) feature extraction method
- PCA project x into subspace of dimension d which has the largest variance
- PCA produced uncorrelated features
- PCA use the eigenvectors with largest eigenvalues as the basis in the sub-space
- PCA is sensitive to outliers. A few points distant from the center have large effect on the variances and thus eigenvectors

Pca in python



Pca in practice

- eigenvalue represents prop. of explained variance: $\sum \lambda_i = tr(\Sigma) = \sum Var(X_i)$
- use svd
- adaptive PCA is faster (sequential)

sources

Previous syde675 classes (Mohammad Javad Shafiee
-Octavia Camps, PSU
-www.cs.rit.edu/~rsg/BIIS_05lecture7.pp tby R.S.Gaborski Professor
-hebb.mit.edu/courses/9.641/lectures/pca.ppt by Sebastian Seung.
Cs189
Cs5785
Concise Machine Learning - Jonathan Richard Shewchuk
Russ Salakhutdinov – U. Toronto
B. Bernstein, NYU