Lecture 11: Dimensionality Reduction

Shuai Li

John Hopcroft Center, Shanghai Jiao Tong University

https://shuaili8.github.io

https://shuaili8.github.io/Teaching/VE445/index.html

Outline

- Motivation
- PCA
- SVD
- Autoencoder
- Feature selection

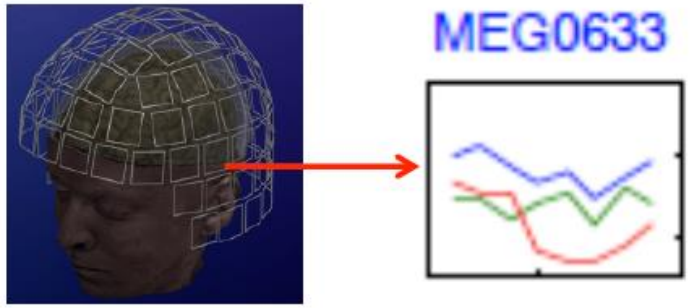
Motivation

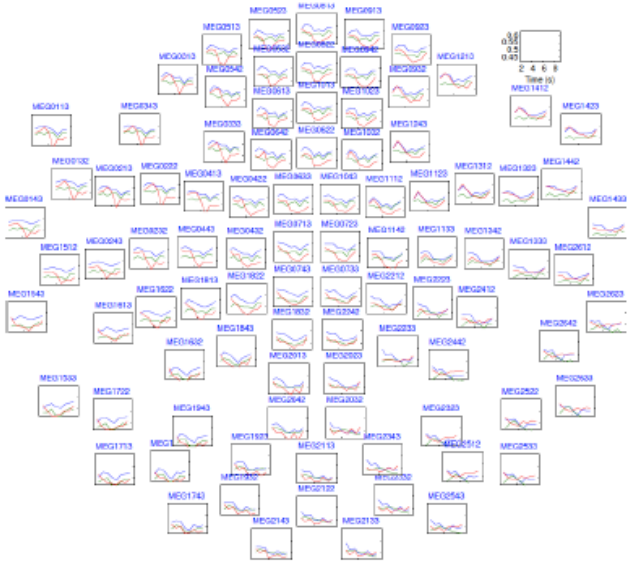
- Suppose we want to predict the health condition of some students, and the features for the students includes:
  - Weight in kilogram
  - Height in inch
  - Height in cm
  - Hours of sports per day
  - Favorite color
  - Scores in math

- Some features are irrelevant, e.g. favorite color and scores in math
- Some features are redundant, e.g. height in inch and cm

High dimensional data

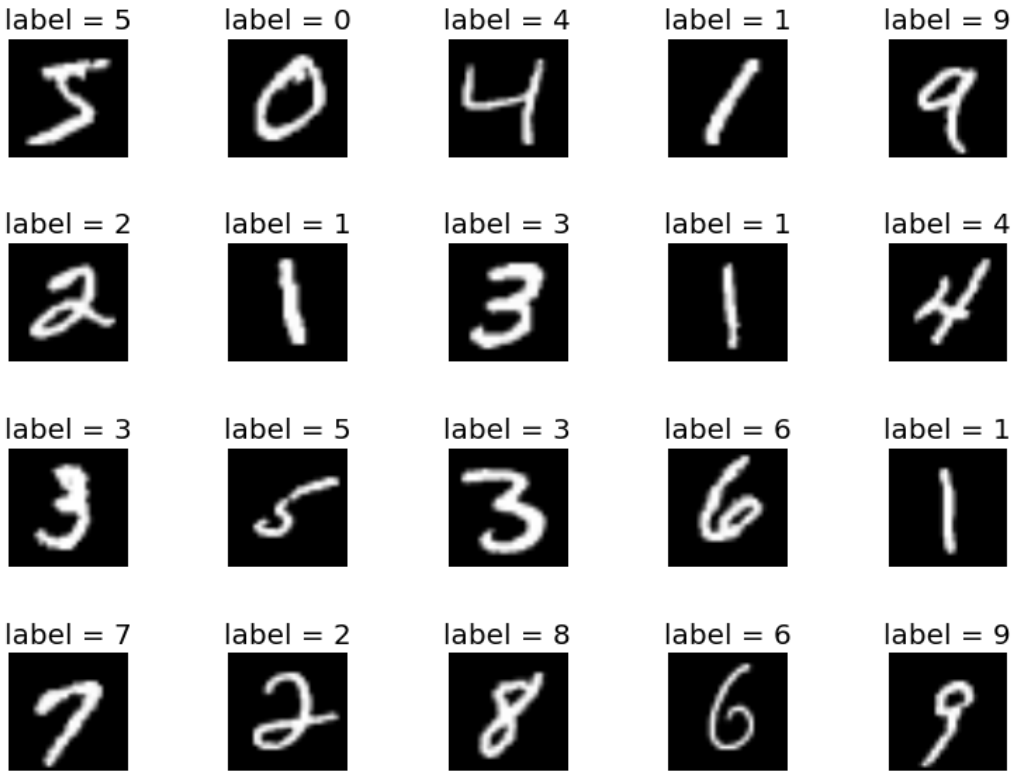
- In the era of big data, the dimensionality increases dramatically
  - E.g. there are many features for the electroencephalogram data

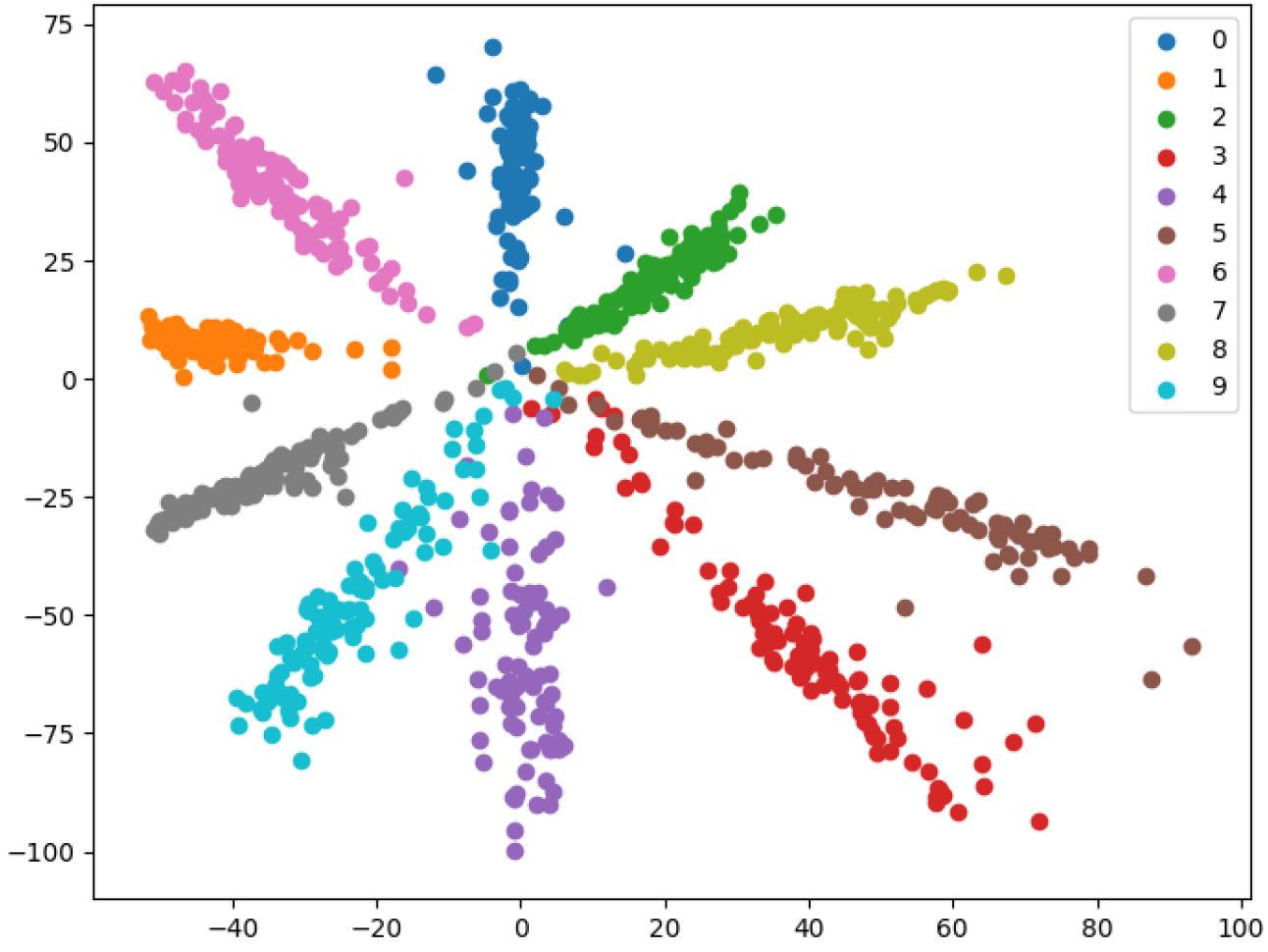




- It becomes very important to reduce the dimensionality, or select the most important features, or find the most representative features

Example – MNIST dataset





Principal Components Analysis

PCA

Principal components analysis (PCA)

- Principal components analysis (PCA) is a technique that can be used to simplify a dataset
- It is usually a linear transformation that chooses a new coordinate system for the data set such that
  - greatest variance by any projection of the dataset 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

Example

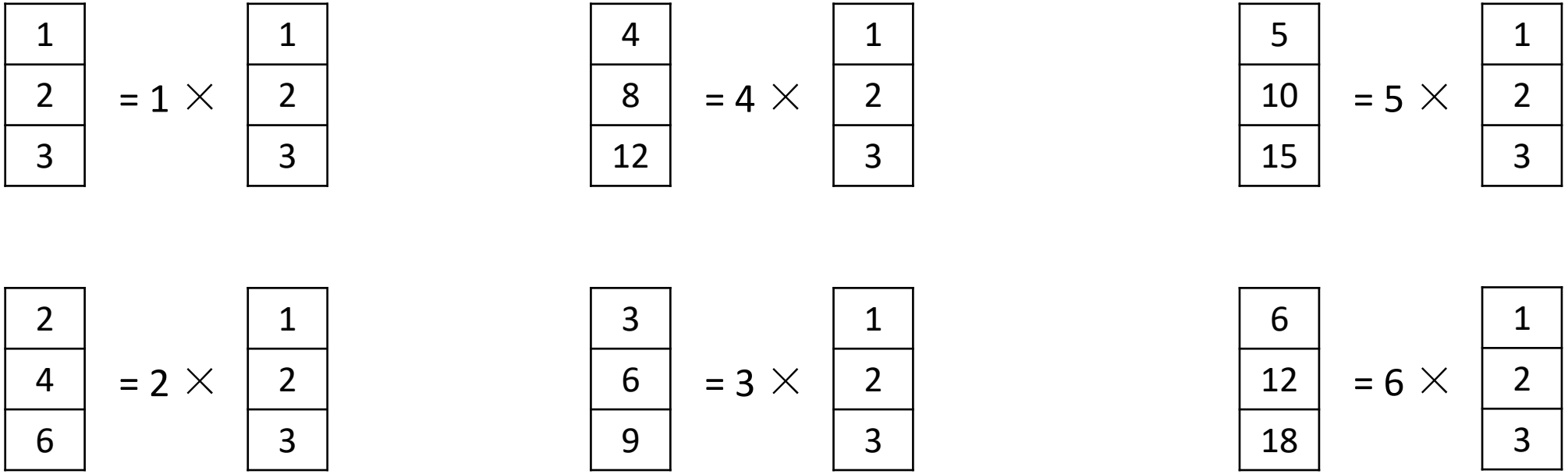
- Consider the following 3D points

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

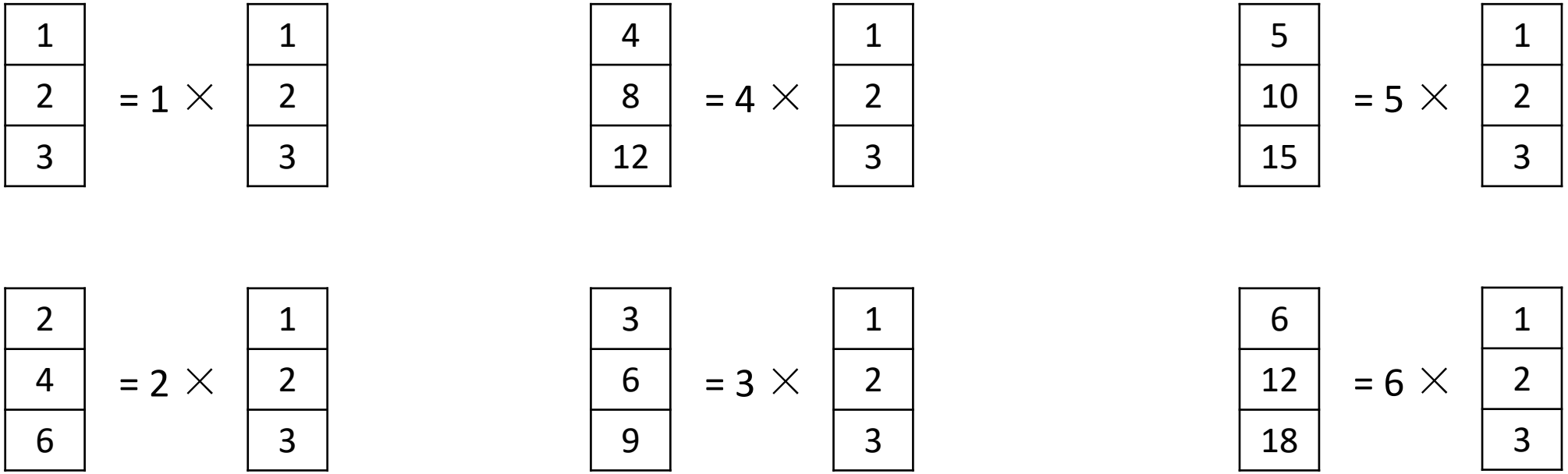
- If each component is stored in a byte, we need $18 = 3 \times 6$ bytes

Example (cont.)

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



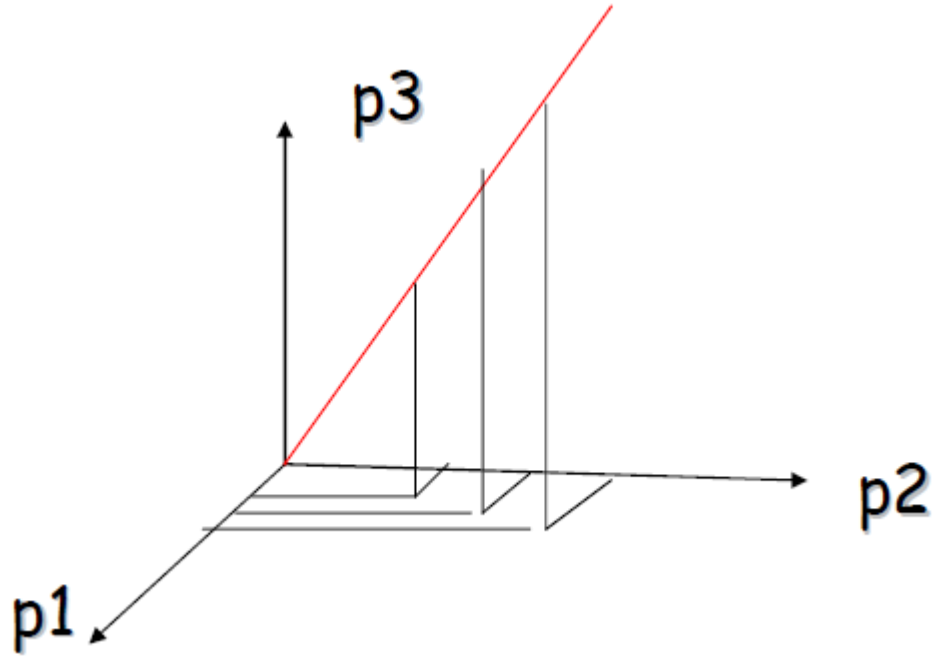
Example (cont.)



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

Geometrical interpretation

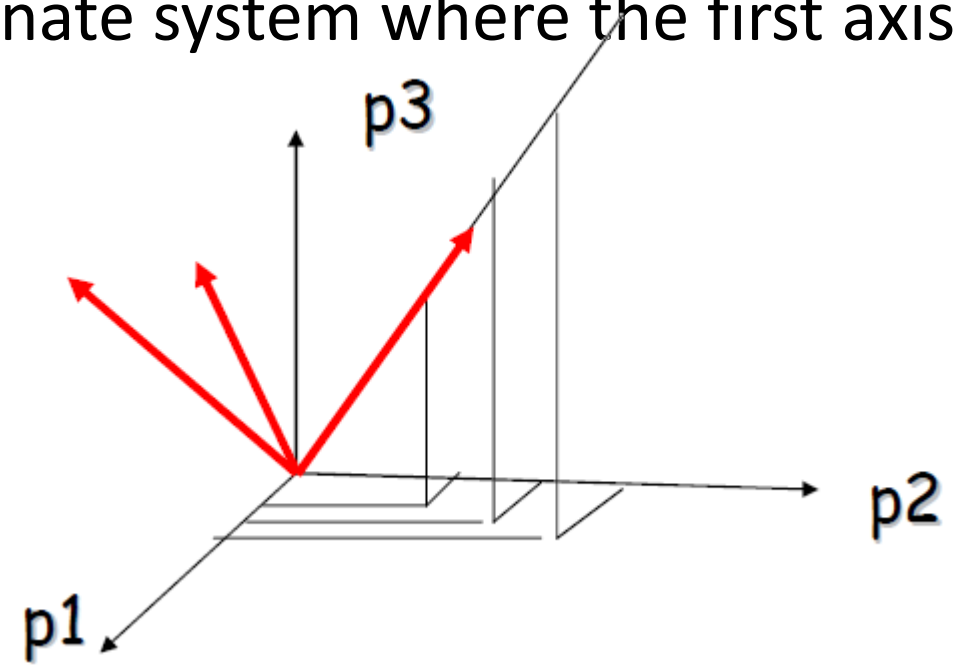
- View points in 3D space



- In this example, all the points happen to lie on one line
  - a 1D subspace of the original 3D space

Geometrical interpretation

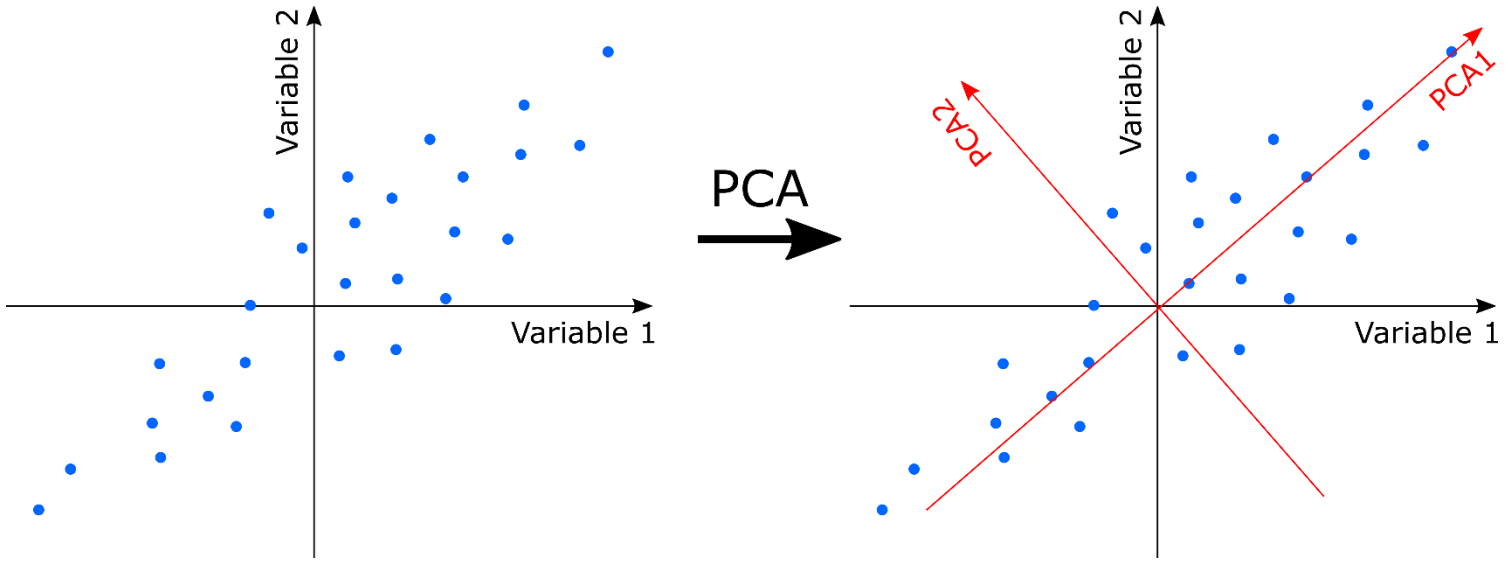
- Consider a new coordinate system where the first axis is along the direction of the line



- In the new coordinate system, every point has only one non-zero coordinate
  - we only need to store the direction of the line (a 3 bytes point) and the nonzero coordinates for each point (6 bytes)

Back to PCA

- Given a set of points, how can we know if they can be compressed similarly to the previous example?
  - We can look into the correlation between the points by the tool of PCA



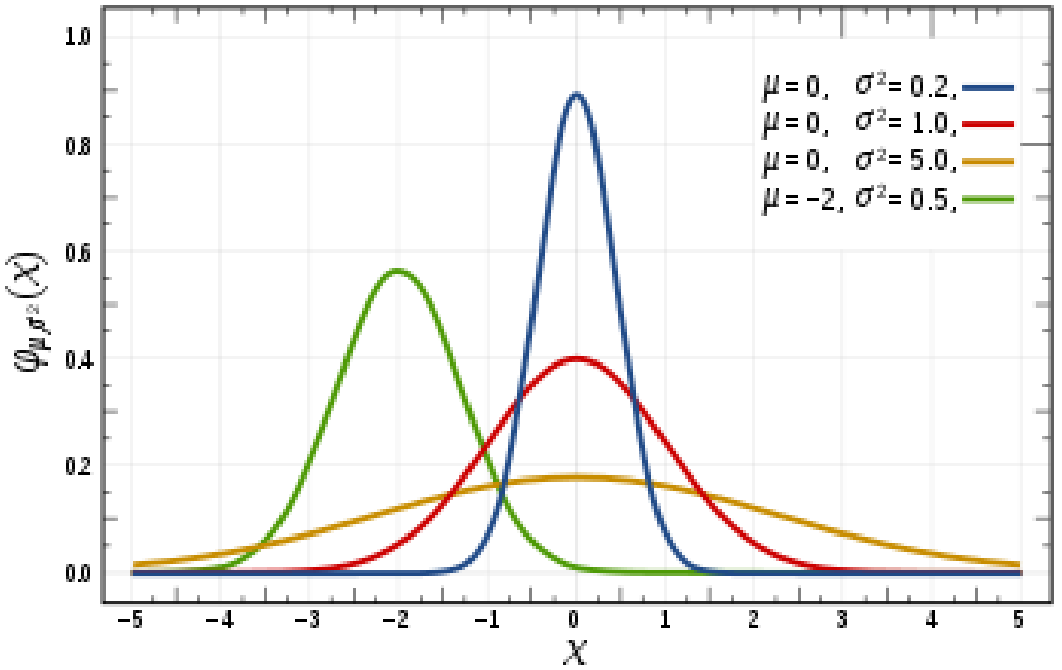
From example to theory

- In previous example, PCA rebuilds the coordination system for the data by selecting
  - the direction with largest variance as the first new base direction
  - the direction with the second largest variance as the second new base direction
  - and so on

- Then how can we find the direction with largest variance?
  - By the eigenvector for the covariance matrix of the data

Review – Variance

- Variance is the expectation of the squared deviation of a random variable from its mean
  - Informally, it measures how far a set of (random) numbers are spread out from their average value



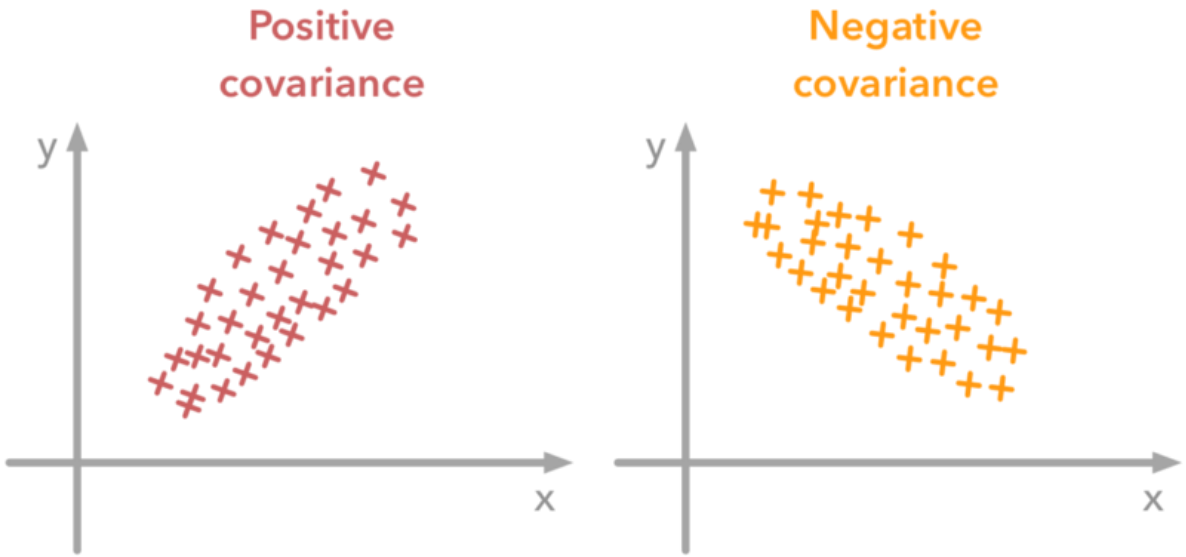
Review – Covariance

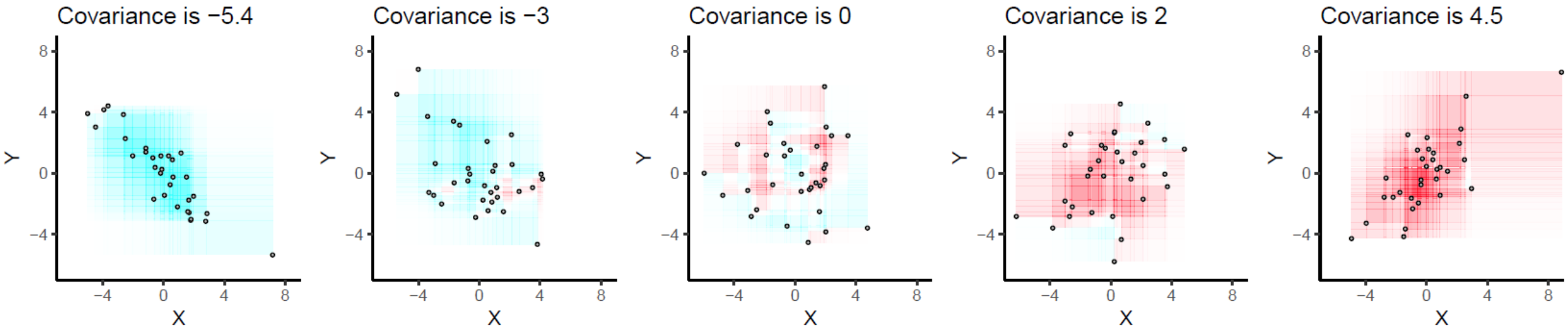
- Covariance is a measure of the joint variability of two random variables
  - If the greater values of one variable mainly correspond with the greater values of the other variable, and the same holds for the lesser values, (i.e., the variables tend to show similar behavior), the covariance is positive
    - E.g. as the number of hours studied increases, the marks in that subject increase
  - In the opposite case, when the greater values of one variable mainly correspond to the lesser values of the other, (i.e., the variables tend to show opposite behavior), the covariance is negative
  - The sign of the covariance therefore shows the tendency in the linear relationship between the variables
  - The magnitude of the covariance is not easy to interpret because it is not normalized and hence depends on the magnitudes of the variables. The normalized version of the covariance, the correlation coefficient, however, shows by its magnitude the strength of the linear relation

Review – Covariance (cont.)

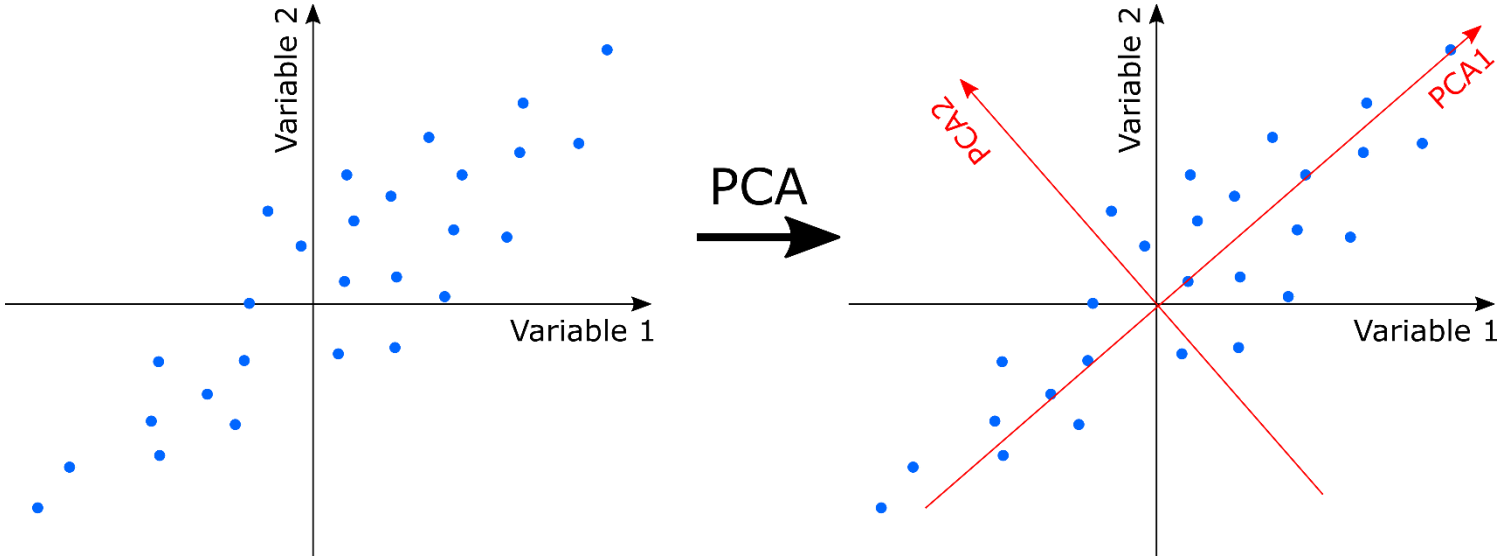
- Sample covariance

$$\text{covariance}(X, Y) = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{N - 1}$$





PCA



- PCA tries to identify the subspace in which the data approximately lies in
- PCA uses an orthogonal transformation on the coordinate system to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components
  - The number of principal components is less than or equal to $\min\{d, N\}$

Covariance matrix

- Suppose there are 3 dimensions, denoted as X, Y, Z. The covariance matrix is

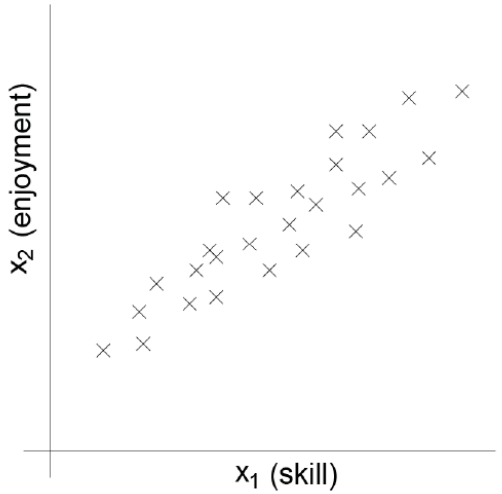
$$COV = \begin{bmatrix} COV(X,X) & COV(X,Y) & COV(X,Z) \\ COV(Y,X) & COV(Y,Y) & COV(Y,Z) \\ COV(Z,X) & COV(Z,Y) & COV(Z,Z) \end{bmatrix}$$

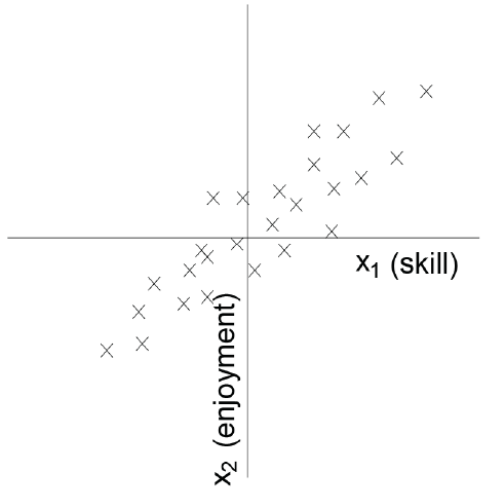
- Note the diagonal is the covariance of each dimension with respect to itself, which is just the variance of each random variable
- Also $COV(X,Y) = COV(Y,X)$
  - hence matrix is symmetric about the diagonal
- d-dimensional data will result in a $d \times d$ covariance matrix

Covariance in the covariance matrix

- Diagonal, or the variance, measures the deviation from the mean for data points in one dimension
- Covariance measures how one dimension random variable varies w.r.t. another, or if there is some linear relationship among them

Data processing





- Given the dataset $D = \left\{x^{(i)}\right\}_{i=1}^{N}$
- Let $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x^{(i)}$

$$X = \begin{bmatrix} \left(x^{(1)} - \bar{x}\right)^{\mathsf{T}} \\ \left(x^{(2)} - \bar{x}\right)^{\mathsf{T}} \\ \vdots \\ \left(x^{(N)} - \bar{x}\right)^{\mathsf{T}} \end{bmatrix} \in \mathbb{R}^{N \times d}$$

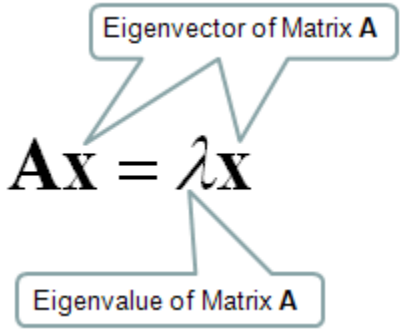
  - Move the center of the data set to 0

Data processing (cont.)

- $Q = X^\top X = [x^{(1)} - \bar{x} \quad x^{(2)} - \bar{x} \quad \dots \quad x^{(N)} - \bar{x}] \begin{bmatrix} (x^{(1)} - \bar{x})^\top \\ (x^{(2)} - \bar{x})^\top \\ \vdots \\ (x^{(N)} - \bar{x})^\top \end{bmatrix}$
  - Q is square with d dimension
  - Q is symmetric
  - Q is the covariance matrix [aka scatter matrix]
  - Q can be very large (in vision, d is often the number of pixels in an image!)
    - For a 256×256 image, $d = 65536$!!
    - Don't want to explicitly compute Q

PCA

- 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 variation in the dataset
- This is the principal component



- Application:
  - face recognition, image compression
  - finding patterns in data of high dimension

PCA theorem

- Theorem:
- Each $x^{(i)}$ can be written as: $x^{(i)} = \bar{x} + \sum_{j=1}^{d} g_{ij} e_j$
  where e_j are the d eigenvectors of Q with non-zero eigenvalues

- **Notes:**
  1. The eigenvectors $e_1 e_2 \cdots e_d$ span an **eigenspace**
  2. $e_1 e_2 \cdots e_d$ are $d \times 1$ orthonormal vectors (directions in d-Dimensional space)
  3. The scalars g_{ij} are the coordinates of $x^{(i)}$ in the space

$$g_{ij} = \left\langle x^{(i)} - \bar{x}, e_j \right\rangle$$

Using PCA to compress data

- Expressing x in terms of $e_1 e_2 \cdots e_d$ doesn't change the size of the data
- However, if the points are highly correlated, many of the new coordinates of x will become zero or close to zero
- Sort the eigenvectors e_i according to their eigenvalue

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$$

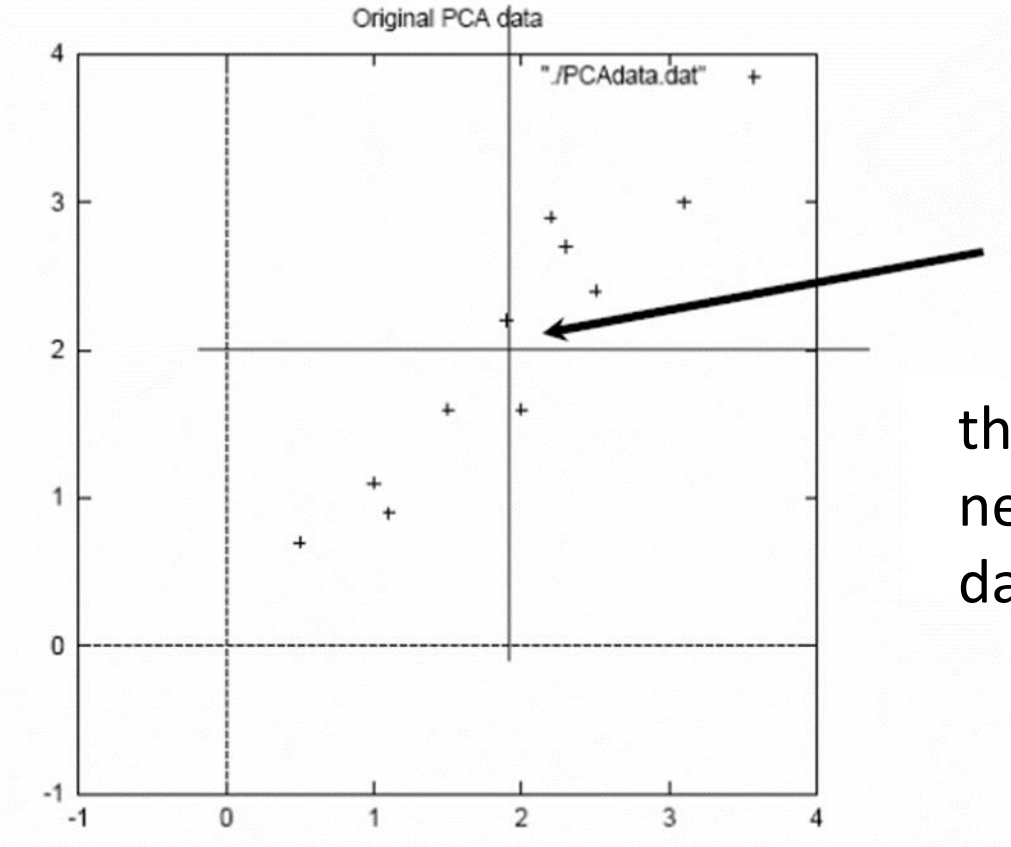
- Assume $\lambda_j \approx 0$ if $j > k$. Then

$$x^{(i)} \approx \bar{x} + \sum_{j=1}^{k} g_{ij} e_j$$

Example – STEP 1

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

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

Example – STEP 2

- Calculate the covariance matrix

$$\text{Cov} = \begin{bmatrix} 0.616555556 & 0.6154444444 \\ 0.6154444444 & 0.716555556 \end{bmatrix}$$

- since $\text{cov}(X, Y)$ is positive, it is expect that x and y increase together

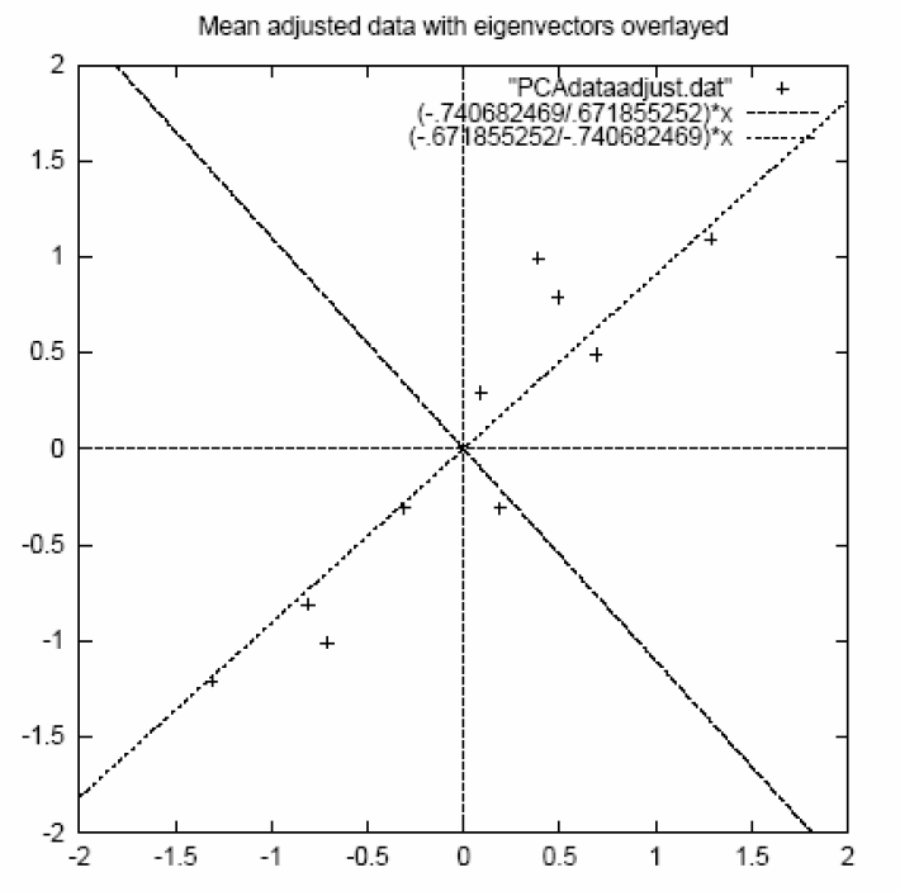
Example – STEP 3

- Calculate the eigenvectors and eigenvalues of the covariance matrix

- $\text{eigenvalues} = \begin{bmatrix} 0.0490833989 \\ 1.28402771 \end{bmatrix}$

- $\text{eigenvectors} = \begin{bmatrix} -0.735178656 & -0.677873399 \\ 0.677873399 & -0.735178656 \end{bmatrix}$

Example – STEP 3 (cont.)



- 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

Example – STEP 4

- Feature vector $= [e_1 \quad e_2 \quad \cdots \quad e_d]$

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

$$\begin{bmatrix} -0.735178656 & -0.677873399 \\ 0.677873399 & -0.735178656 \end{bmatrix}$$

- or, we can choose to delete the smaller, less significant component:

$$\begin{bmatrix} -0.677873399 \\ -0.735178656 \end{bmatrix}$$

Example – STEP 5

$$\text{FinalData}_{N\times d} = \begin{bmatrix} g(x^{(1)})^{\mathsf{T}} \\ \vdots \\ g(x^{(N)})^{\mathsf{T}} \end{bmatrix}_{N\times d} \begin{bmatrix} e_1^{\mathsf{T}} \\ \vdots \\ e_d^{\mathsf{T}} \end{bmatrix}_{d\times d}$$

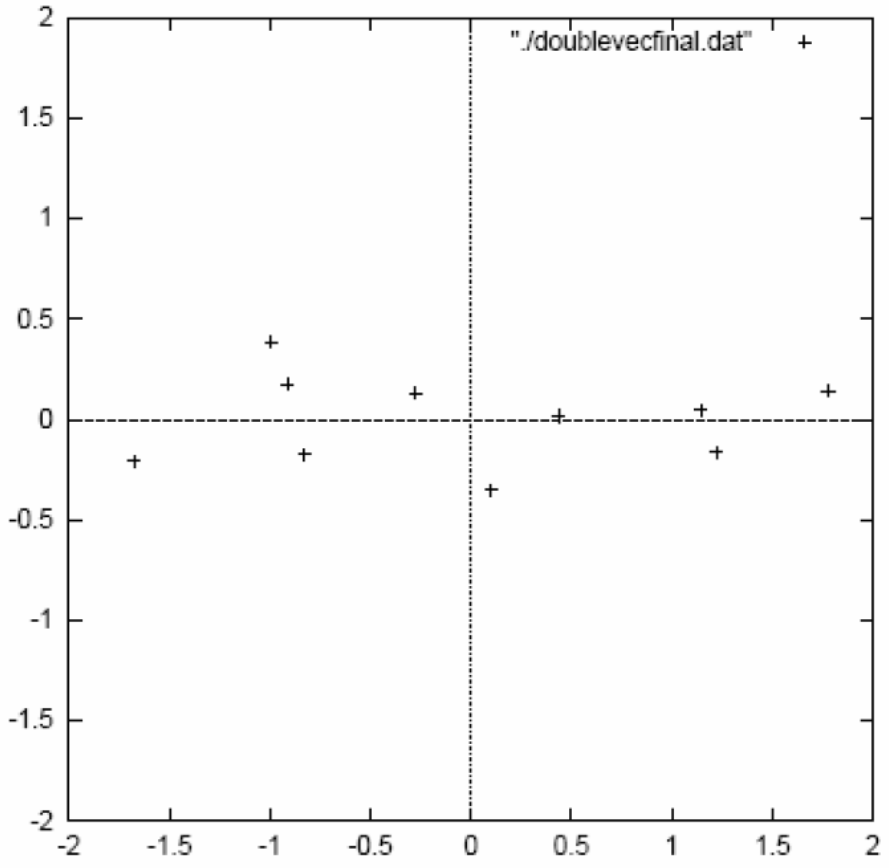
- Deriving new data coordinates

FinalData = RowZeroMeanData x RowFeatureVector

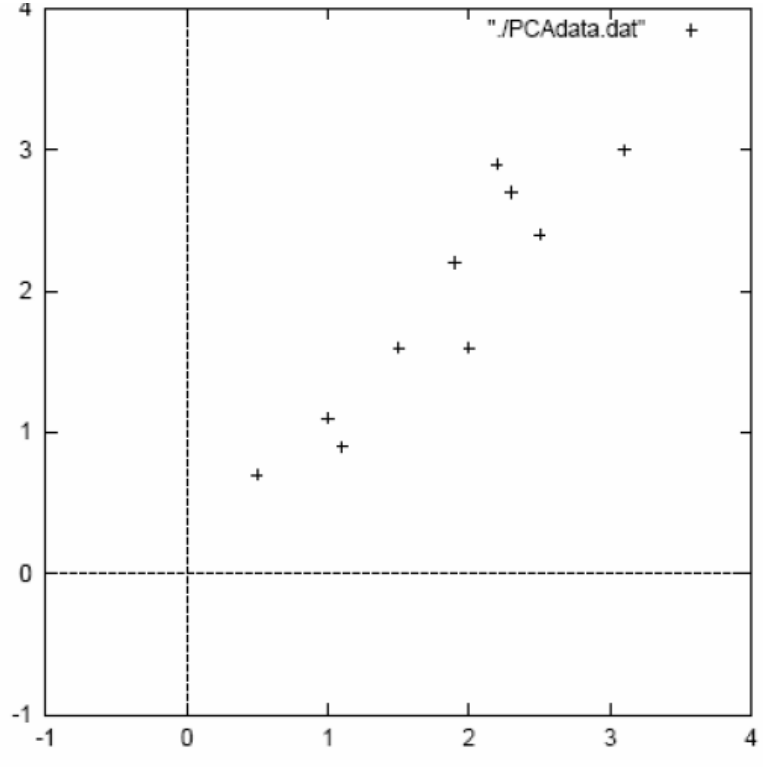
- RowZeroMeanData is the mean-adjusted data, i.e. the data items are in each row, with each column representing a separate dimension
- RowFeatureVector is the matrix with the eigenvectors in the columns transposed so that the eigenvectors are now in the rows, with the most significant eigenvector at the top
- Note: We rotate the coordinate axes so high-variance axis comes first

Example – STEP 5 (cont.)

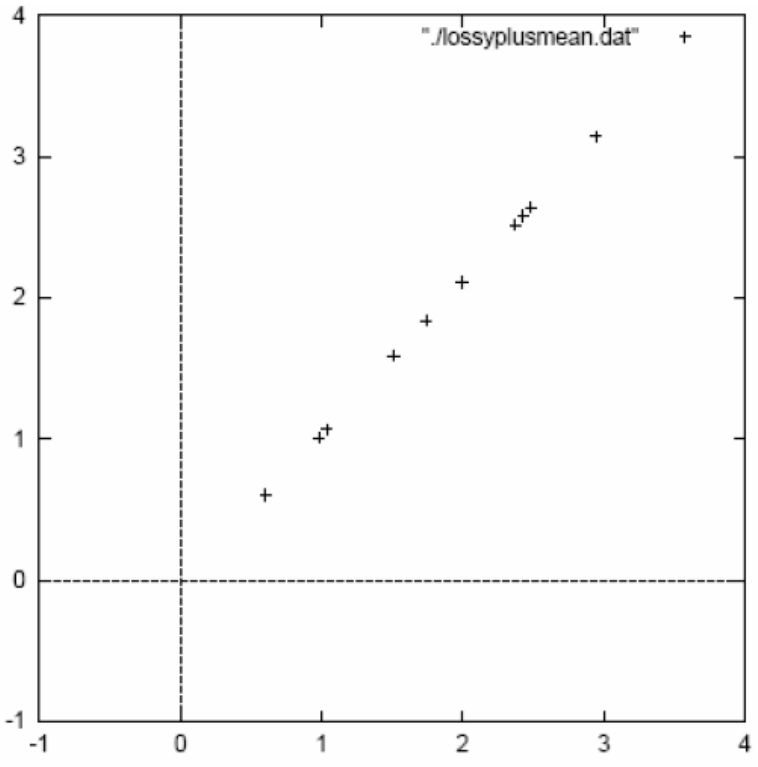
- The plot of the PCA results using both the two eigenvector



Example – Final approximation



2D point cloud



Approximation using one eigenvector basis

Example – Final approximation

$$\text{FinalData}_{N\times d} = \begin{bmatrix} g\left(x^{(1)}\right)^{\top} \\ \vdots \\ g\left(x^{(N)}\right)^{\top} \end{bmatrix}_{N\times d} \begin{bmatrix} e_1^{\top} \\ \vdots \\ e_d^{\top} \end{bmatrix}_{d\times d}$$

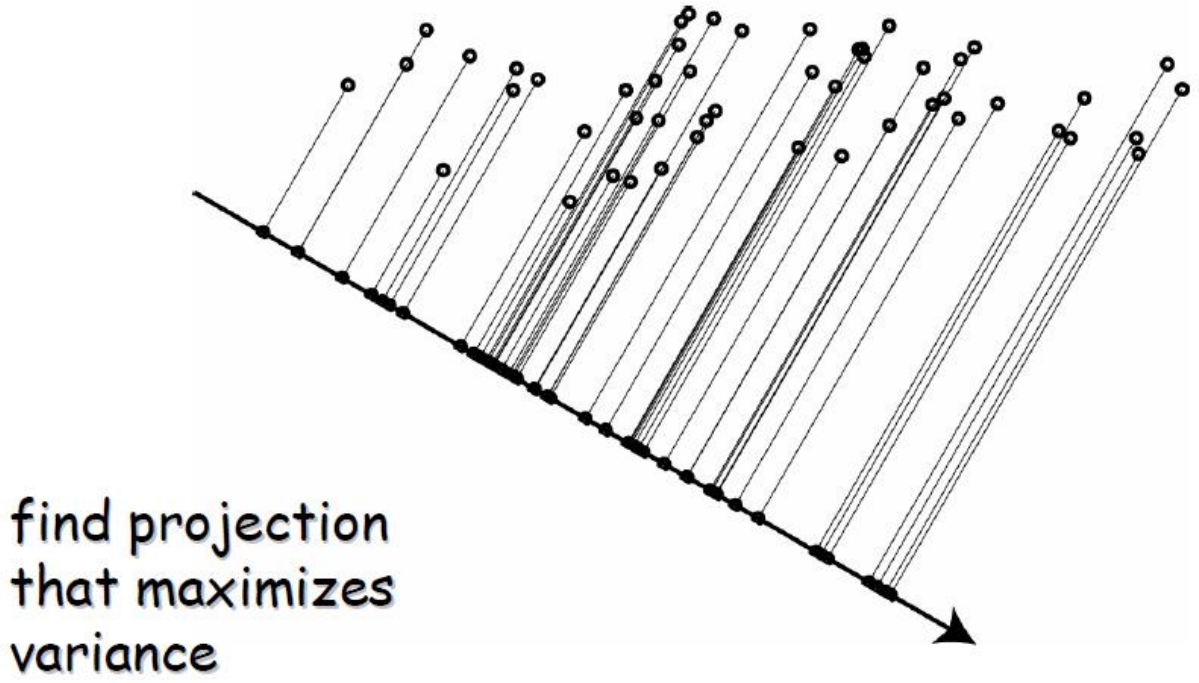
$$\approx \begin{bmatrix} g\left(x^{(1)}\right)_1 & \dots & g\left(x^{(1)}\right)_k & \cdots & \cancel{g\left(x^{(1)}\right)_d} \\ & & \vdots & & \\ g\left(x^{(N)}\right)_1 & \dots & g\left(x^{(N)}\right)_k & \cdots & \cancel{g\left(x^{(N)}\right)_d} \end{bmatrix}_{N\times d} \begin{bmatrix} e_1^{\top} \\ \vdots \\ e_k^{\top} \\ \vdots \\ e_d^{\top} \end{bmatrix}_{d\times d}$$

Zero!!

Revisit the eigenvectors in PCA

- It is critical to notice that the *direction of maximum variance* in the input space happens to be same as the *principal eigenvector of the covariance matrix*

- Why?



Revisit the eigenvectors in PCA (cont.)

- The projection of each point x to a direction u (with $\|u\| = 1$) is
$$x^\top u$$
- The variance of the projection is
$$\sum_{i=1}^{N} \left(\left(x^{(i)} - \bar{x} \right)^\top u \right)^2 = u^\top Q u$$
which is maximized when u is the eigenvector with the largest eigenvalue
- $Q = \sum_{j=1}^{d} \lambda_j e_j e_j^\top = E \Lambda E^\top$ with $\Lambda = \begin{bmatrix} \lambda_1 & \cdots & \\ \vdots & \ddots & \vdots \\ & \cdots & \lambda_d \end{bmatrix}$

Review – Total/Explained variance

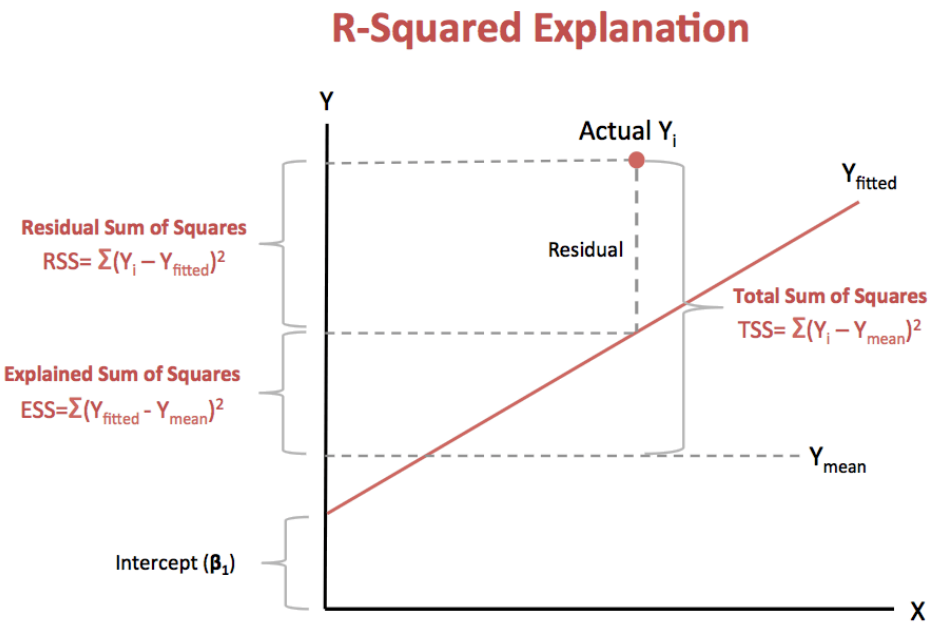
- $R^2 = \frac{explained\ variance}{total\ variance}$

- Total variance: $SS_{\text{tot}} = \sum_i (y_i - \bar{y})^2$

- Explained variance: $SS_{\text{reg}} = \sum_i (f_i - \bar{y})^2$

- Or, it can be computed as:

$R^2 \equiv 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$ where $SS_{\text{res}} = \sum_i (y_i - f_i)^2 = \sum_i e_i^2$



$R_{Sq} = 1 - \frac{RSS}{TSS}$

Total variance and PCA

- Note that $I = e_1 e_1^\top + \cdots + e_d e_d^\top$
- Total variance is
- $\sum_{i=1}^{N} \left(x^{(i)} - \bar{x}\right)^\top \left(x^{(i)} - \bar{x}\right)$
- $= \sum_{i=1}^{N} \left(x^{(i)} - \bar{x}\right)^\top \left(e_1 e_1^\top + \cdots + e_d e_d^\top\right)\left(x^{(i)} - \bar{x}\right)$
- $= \sum_{j=1}^{d} e_j^\top Q\, e_j = \lambda_1 + \cdots + \lambda_d$

Total variance and PCA (cont.)

- Approximation of each $x^{(i)} - \bar{x} \approx \sum_{j=1}^{k} g_{ij} e_j =: \tilde{x}^{(i)} - \bar{x}$
- Then the explained variance is
- $\sum_{i=1}^{N} \left(\tilde{x}^{(i)} - \bar{x}\right)^{\mathsf{T}} \left(\tilde{x}^{(i)} - \bar{x}\right)$
- $= \sum_{i=1}^{N} \left(\tilde{x}^{(i)} - \bar{x}\right)^{\mathsf{T}} \left(e_1 e_1^{\mathsf{T}} + \cdots + e_d e_d^{\mathsf{T}}\right) \left(\tilde{x}^{(i)} - \bar{x}\right)$
- $= \sum_{j=1}^{d} e_j^{\mathsf{T}} \tilde{Q} e_j = \lambda_1 + \cdots + \lambda_k$
- where $\tilde{Q} = \sum_{i=1}^{N} \left(\tilde{x}^{(i)} - \bar{x}\right) \left(\tilde{x}^{(i)} - \bar{x}\right)^{\mathsf{T}} = E \tilde{\Lambda} E^{\mathsf{T}}$ with

- $\tilde{\Lambda} = \begin{bmatrix} \lambda_1 & & & & \\ & \ddots & & & \\ & & \lambda_k & & \\ & & & 0 & \\ & & & & 0 \end{bmatrix}$

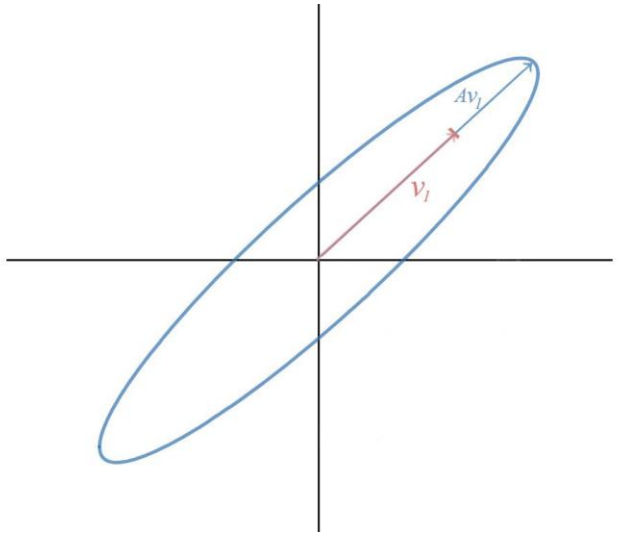
Eigenvalues and Eigenvectors

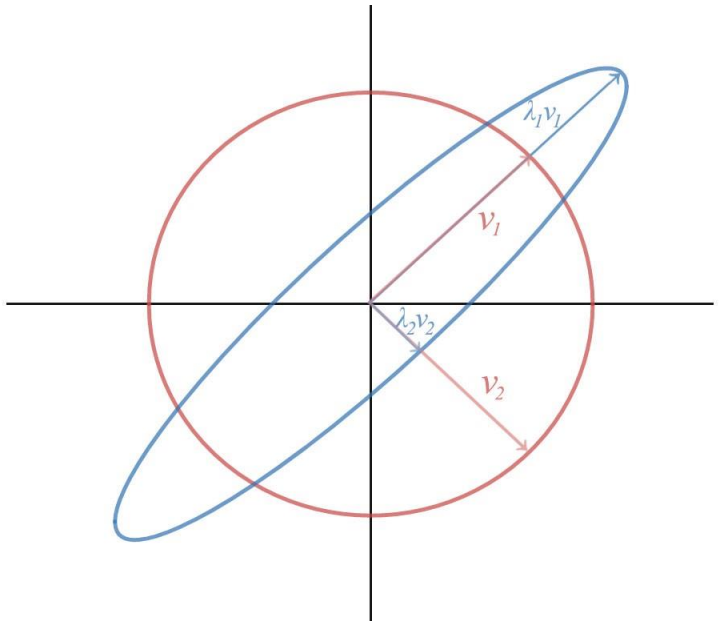
Properties

- Scalar λ and vector v are eigenvalues and eigenvectors of A

$$Av = \lambda v$$

- Visually, Av lies along the same line as the eigenvector of v





Example

$$\begin{bmatrix} 1 & -3 & 3 \\ 3 & -5 & 3 \\ 6 & -6 & 4 \end{bmatrix} \begin{bmatrix} 1/2 \\ 1/2 \\ 1 \end{bmatrix} = 4 \begin{bmatrix} 1/2 \\ 1/2 \\ 1 \end{bmatrix}$$

A — *eigenvalue* — *eigenvector*

$$\begin{bmatrix} 1 & -3 & 3 \\ 3 & -5 & 3 \\ 6 & -6 & 4 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = -2 \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

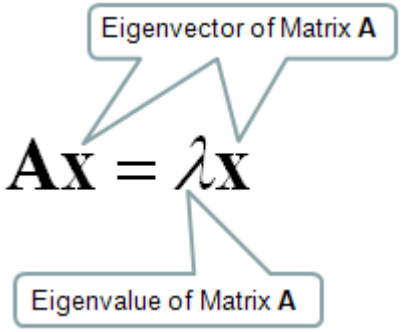
$$\begin{bmatrix} 1 & -3 & 3 \\ 3 & -5 & 3 \\ 6 & -6 & 4 \end{bmatrix} \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} = -2 \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

How to solve eigenvalues and eigenvectors?

- If $(A - \lambda I)x = 0$ has a nonzero solution for λ, then $A - \lambda I$ is not invertible. Then the determinant of $A - \lambda I$ must be zero

- λ is an eigenvalue of A if and only if $A - \lambda I$ is singular:

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



Example

- Find the eigenvalues and eigenvectors of $A = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}$

- Need to solve $\det(A - \lambda I) = 0$
- $A - \lambda I = \begin{bmatrix} 1-\lambda & 2 \\ 2 & 4-\lambda \end{bmatrix}$
- $\det(A - \lambda I) = (1-\lambda)(4-\lambda) - 2 \times 2 = \lambda^2 - 5\lambda$
- $\det(A - \lambda I) = 0$ would imply $\lambda = 0$ and $\lambda = 5$

Example (cont.)

- Then solve $(A - \lambda I)x = 0$ to get the eigenvectors for each of the eigenvalues
- $Ax = 0$ has a nonzero solution of $\begin{bmatrix} 2 \\ -1 \end{bmatrix}$, or $\begin{bmatrix} 2/\sqrt{5} \\ -1/\sqrt{5} \end{bmatrix}$
- $(A - 5I)x = 0$ has a nonzero solution of $\begin{bmatrix} 1 \\ 2 \end{bmatrix}$, or $\begin{bmatrix} 1/\sqrt{5} \\ 2/\sqrt{5} \end{bmatrix}$
- Note: Eigenvectors for different eigenvalues are orthogonal

Singular Value Decomposition

SVD

SVD

- Singular Value Decomposition (SVD) is a factorization method of matrix. It states that any $m \times n$ matrix A can be written as the product of 3 matrices:

$$A = U S V^T$$

- Where:
  - U is $m \times m$ and its columns are orthonormal eigenvectors of $AA^\top$
  - V is $n \times n$ and its columns are orthonormal eigenvectors of $A^\top A$
  - S is $m \times n$ is a diagonal matrix with r elements equal to the root of the positive eigenvalues of $AA^\top$ or $A^\top A$ (both matrices have the same positive eigenvalues anyway)

In full matrix form

$$A_{m\times n} = U_{m\times m} S_{m\times n} V^T_{n\times n}$$

$$\underset{m\times n}{\mathrm{A}} = \underset{m\times m}{\mathrm{U}} \; \underset{m\times n}{\mathrm{S}} \; \underset{n\times n}{\mathrm{V^T}}$$

$$\begin{pmatrix} x_{11} & x_{12} & & x_{1n} \\ & & \ddots & \\ x_{m1} & & & x_{mn} \end{pmatrix} = \begin{pmatrix} u_{11} & & u_{m1} \\ & \ddots & \\ u_{1m} & & u_{mm} \end{pmatrix} \begin{pmatrix} \sigma_1 & \ddots & & 0 \\ & & \sigma_r & \\ & & & \ddots \\ 0 & & & 0 \end{pmatrix} \begin{pmatrix} v_{11} & & v_{1n} \\ & \ddots & \\ v_{n1} & & v_{nn} \end{pmatrix}$$

Example

- Let's assume:

$$A = \begin{pmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{pmatrix}$$

- We can have:

$$AA^T = \begin{pmatrix} 17 & 8 \\ 8 & 17 \end{pmatrix} \qquad A^T A = \begin{pmatrix} 13 & 12 & 2 \\ 12 & 13 & -2 \\ 2 & -2 & 8 \end{pmatrix}$$

Example (cont.)

- Compute U and V respectively:

$$AA^T = \begin{pmatrix} 17 & 8 \\ 8 & 17 \end{pmatrix}$$

eigenvalues: $\lambda_1 = 25, \lambda_2 = 9$

eigenvectors

$$u_1 = \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix} \quad u_2 = \begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix}$$

$$A^T A = \begin{pmatrix} 13 & 12 & 2 \\ 12 & 13 & -2 \\ 2 & -2 & 8 \end{pmatrix}$$

eigenvalues: $\lambda_1 = 25, \lambda_2 = 9, \lambda_3 = 0$

eigenvectors

$$v_1 = \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \end{pmatrix} \quad v_2 = \begin{pmatrix} 1/\sqrt{18} \\ -1/\sqrt{18} \\ 4/\sqrt{18} \end{pmatrix} \quad v_3 = \begin{pmatrix} 2/3 \\ -2/3 \\ -1/3 \end{pmatrix}$$

Example (cont.)

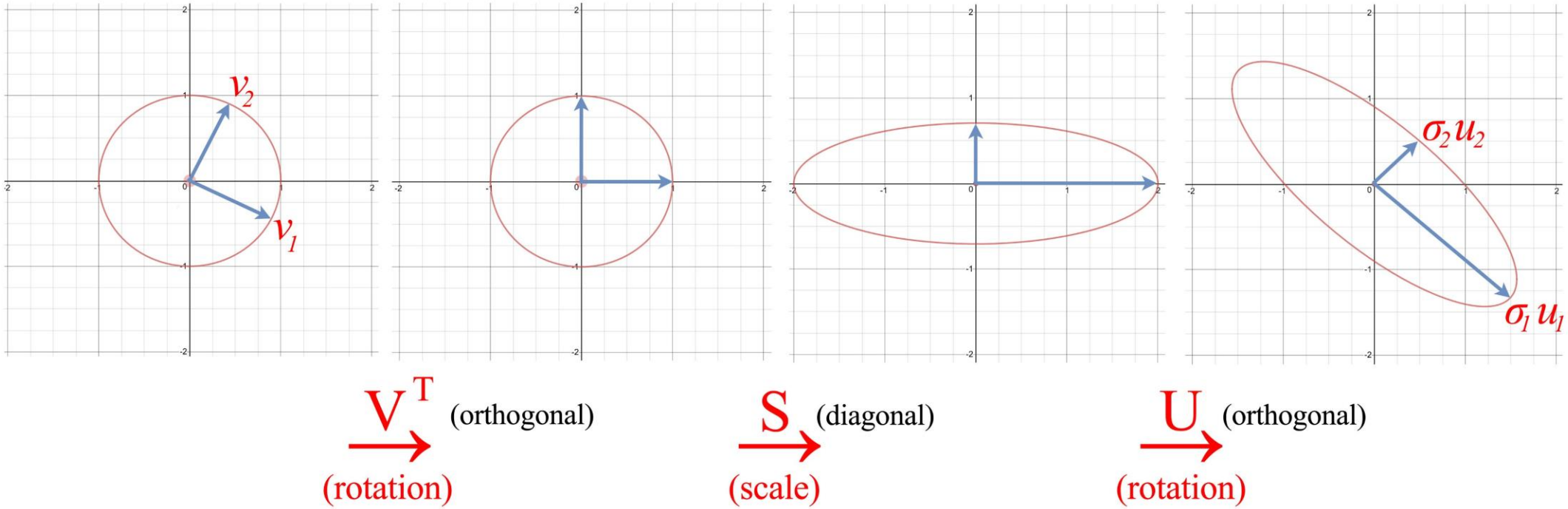
- Finally, we have:

$$A = USV^T = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix} \begin{pmatrix} 5 & 0 & 0 \\ 0 & 3 & 0 \end{pmatrix} \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 1/\sqrt{18} & -1/\sqrt{18} & 4/\sqrt{18} \\ 2/3 & -2/3 & -1/3 \end{pmatrix}$$

Visualization

- The eigenvector v_j of V is transformed into $Av_j = \sigma_j u_j$

$$Ax = USV^T x$$



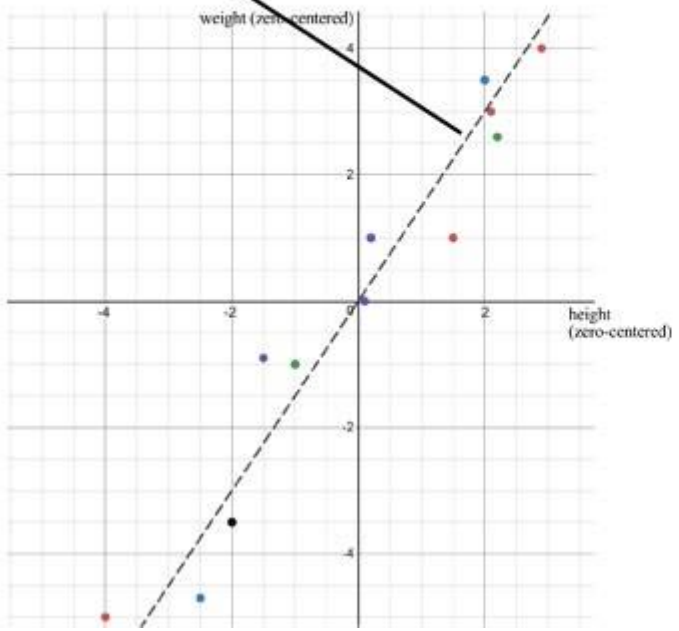
Insight

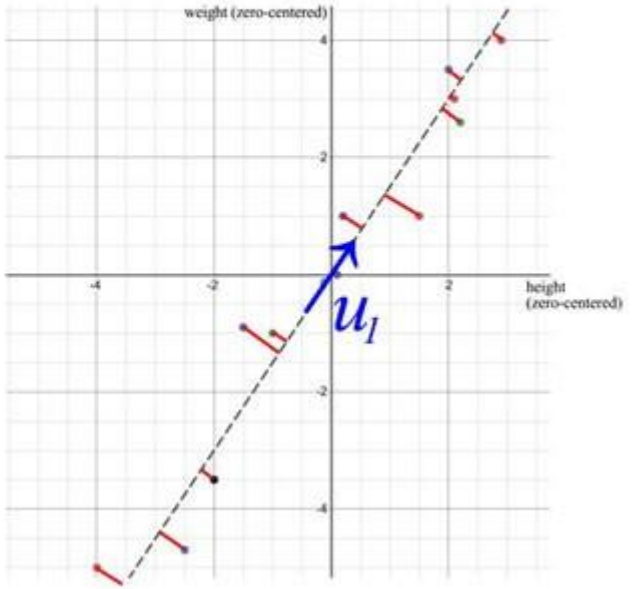
more significant — less significant

$$A = U S V^T = \sigma_1 u_1 v_1^T + \ldots + \sigma_r u_r v_r^T$$

$m \times n$ = $m \times 1$ $1 \times n$ + ... + $m \times n$

$$S = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T$$





SVD and PCA

$$(S^\top S)_{d\times d} = \begin{bmatrix} \sigma_1^2 & \cdots & \\ \vdots & \ddots & \vdots \\ & \cdots & \sigma_d^2 \end{bmatrix}$$

- $X = US_{N\times d}V^\top$
- $Q = X^\top X = V(S^\top)_{d\times N}U^\top \cdot US_{N\times d}V^\top = V(S^\top S)_{d\times d}V^\top$

- This explains why singular value σ_j is the square root of the eigenvalue λ_j of Q
- $\lambda_j = \sigma_j^2$
- Also V contains eigenvectors of $X^\top X$
- Similarly, $XX^\top = US_{N\times d}V^\top \cdot V(S^\top)_{d\times N}U^\top = U(SS^\top)_{N\times N}U^\top$

$$(SS^\top)_{N\times N} = \begin{bmatrix} \sigma_1^2 & \cdots & & & \\ \vdots & \ddots & & \vdots & \\ & & \sigma_d^2 & & \\ & \cdots & & \ddots & \\ & & & & 0 \end{bmatrix}$$

Application

- Matrix factorization in recommendation system

- Suppose in the database of Taobao, there are m users and n items, and an $m \times n$ binary matrix A
  - Each entry indicates whether a user has bought an item or not
  - As each user only buys very few items among all items, the matrix is very sparse
- As the manager of Taobao, how can you predict the likelihood that a user will buy a given item?
- SVD could help

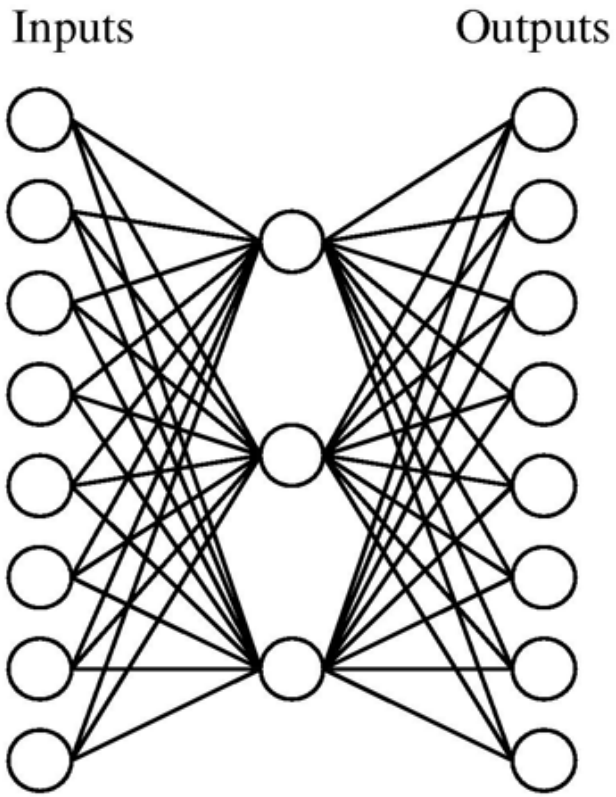
Autoencoder

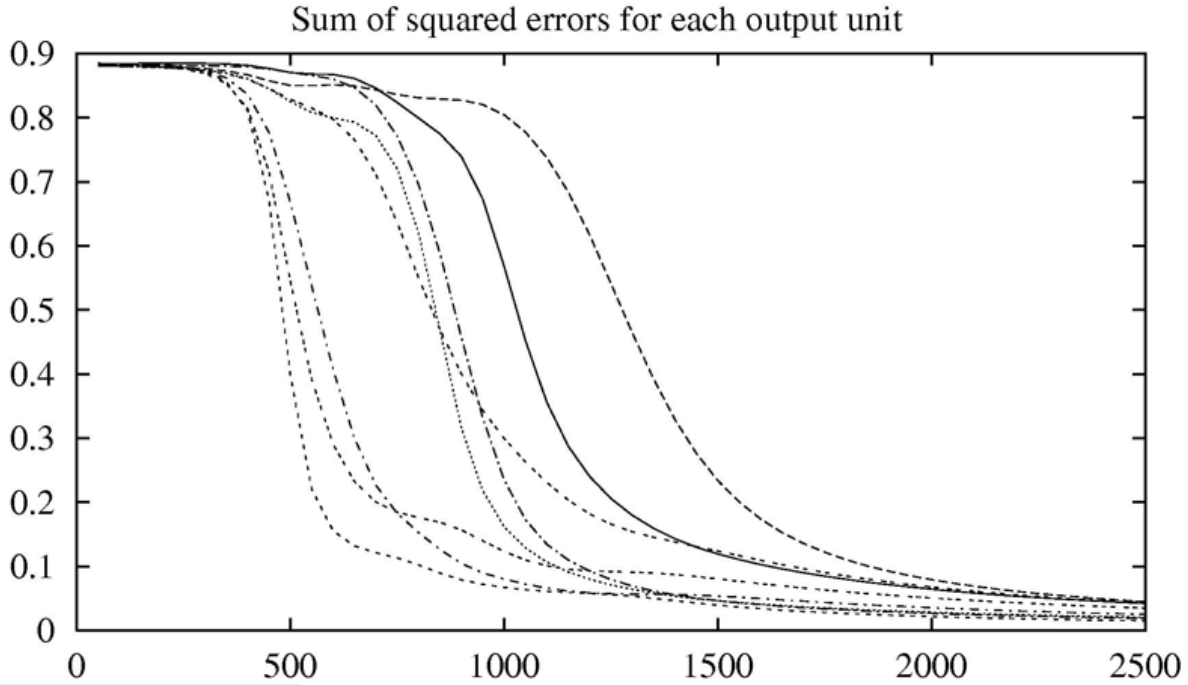
https://www.edureka.co/blog/autoencoders-tutorial/

Autoencoder

- An autoencoder **neural network** is an **unsupervised Machine learning** model that applies backpropagation, setting the target values to be equal to the inputs
- Autoencoders are used to reduce the size of inputs into a smaller representation
- If anyone needs the original data, they can reconstruct it from the compressed data

Example visited

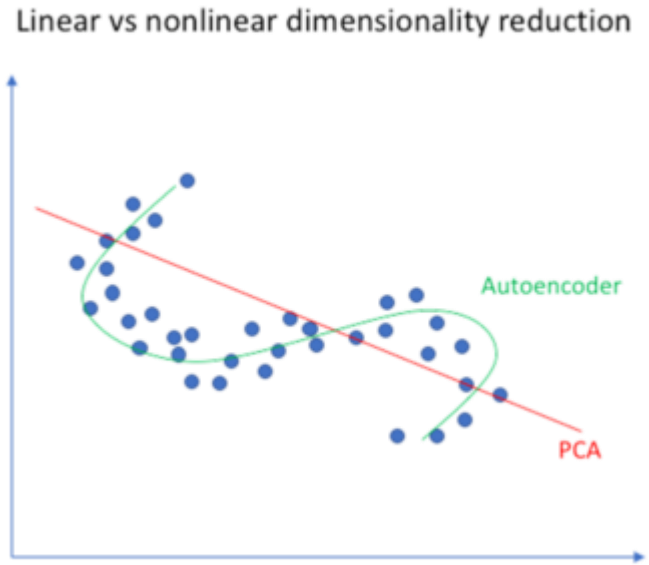




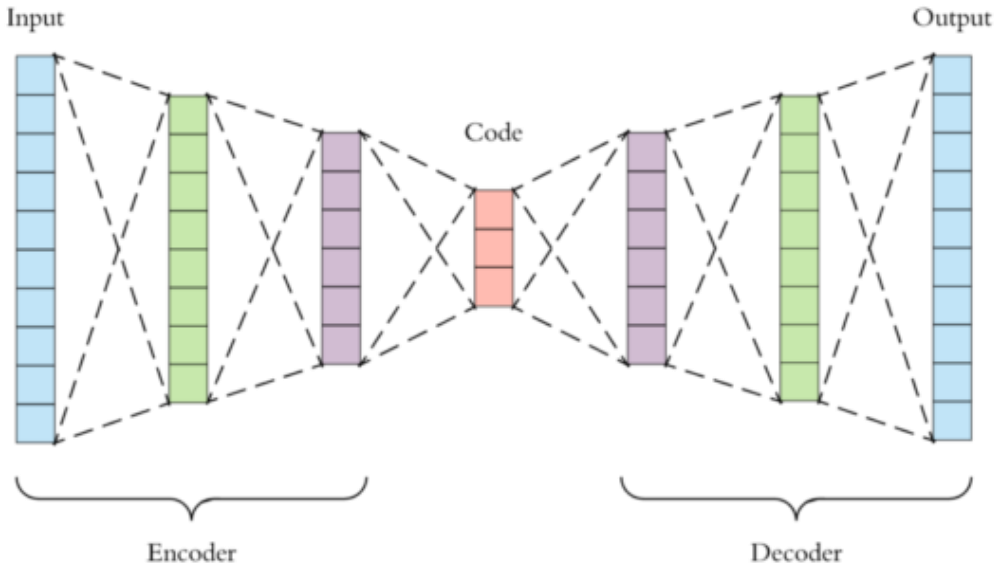
Input		Output
10000000	→	10000000
01000000	→	01000000
00100000	→	00100000
00010000	→	00010000
00001000	→	00001000
00000100	→	00000100
00000010	→	00000010
00000001	→	00000001

Benefits of Autoencoder over PCA

- An autoencoder can learn non-linear transformations with a non-linear activation function and multiple layers
- doesn't have to learn dense layers. Can use convolutional/recurrent layers to learn for video, image and series data
- more efficient to learn several layers at the same time with an autoencoder rather than learn one huge transformation with PCA
- Each layer is a representation
- can make use of pre-trained layers from another model to apply transfer learning to enhance the encoder/decoder

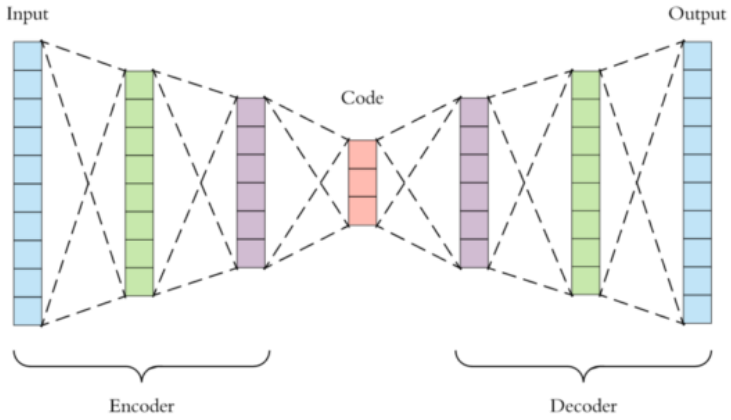


Architecture of Autoencoders



- Encoder: This part of the network compresses the input into a latent space representation. The encoder layer encodes the input image as a compressed representation in a reduced dimension
  - E.g. The compressed image is the distorted version of the original image
- Code: This part of the network represents the compressed input which is fed to the decoder
- Decoder: This layer decodes the encoded image back to the original dimension. The decoded image is a lossy reconstruction of the original image and it is reconstructed from the latent space representation

Training the Autoencoder



- Let f be the function of the encoder

$$f(input) = code$$

- Let g be the function of the decoder

$$g(code) = output$$

- Autoencoder is trying approximate the input using the output, or trying to find f and g that minimize the following loss:

$$J = \sum (output - input)^2$$

- The above loss can be minimized using backpropagation

Hyperparameters

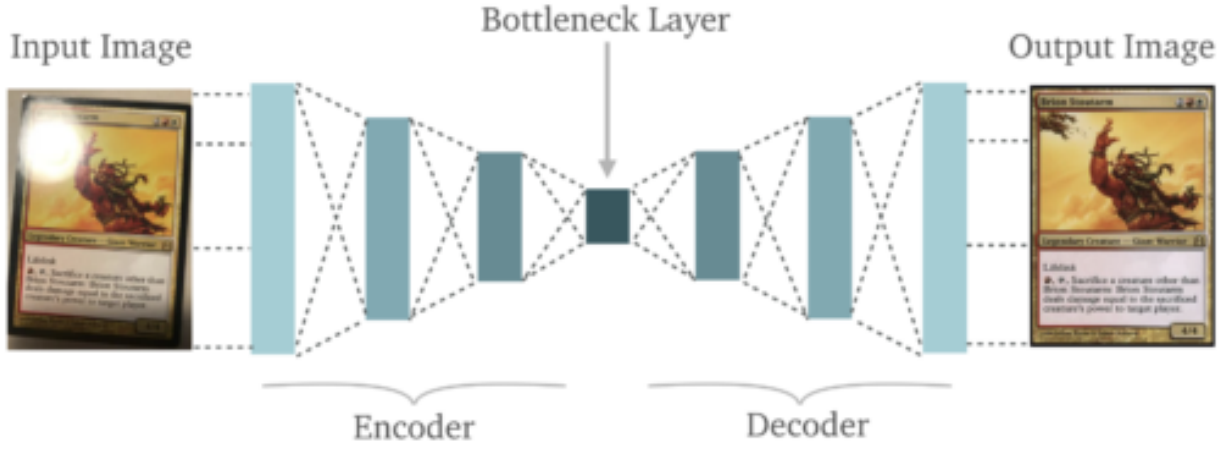
- Code size: It represents the number of nodes in the middle layer. Smaller size results in more compression
- Number of layers: The autoencoder can consist of as many layers as we want
- Number of nodes per layer: The number of nodes per layer decreases with each subsequent layer of the encoder, and increases back in the decoder
  - The decoder is symmetric to the encoder in terms of the layer structure (if possible)
- Loss function: We either use mean squared error or binary cross-entropy
  - If the input values are in the range [0, 1] then we typically use cross-entropy, otherwise, we use the mean squared error

Application – Image coloring

- Autoencoders are used for converting any black and white picture into a colored image. Depending on what is in the picture, it is possible to tell what the color should be

Application – Feature variation

- It extracts only the required features of an image and generates the output by removing any noise or unnecessary interruption

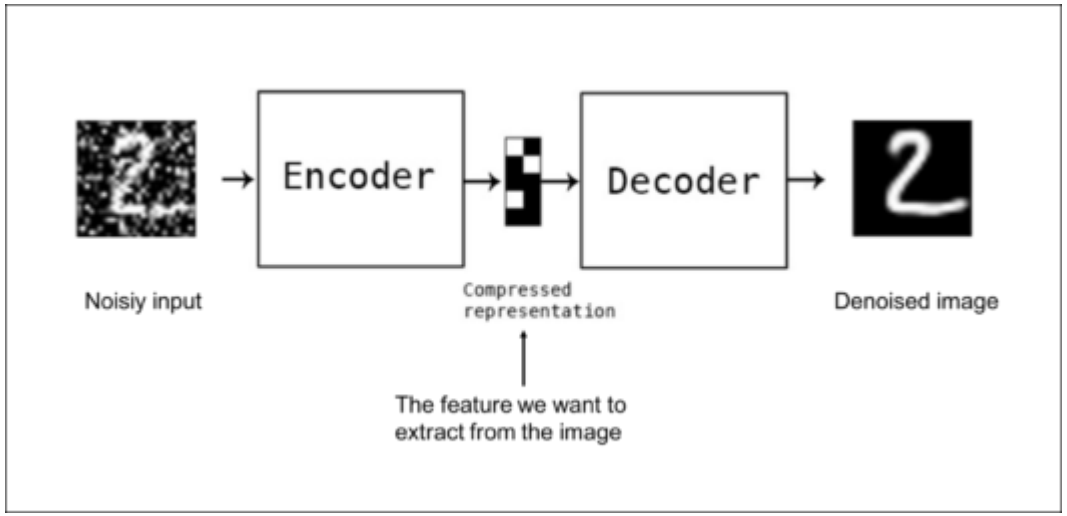


Application – Dimensionality reduction

- The reconstructed image is the same as our input but with reduced dimensions. It helps in providing the similar image with a reduced pixel value

Application – Denoising image

- The input seen by the autoencoder is not the raw input but a stochastically corrupted version. A denoising autoencoder is thus trained to reconstruct the original input from the noisy version

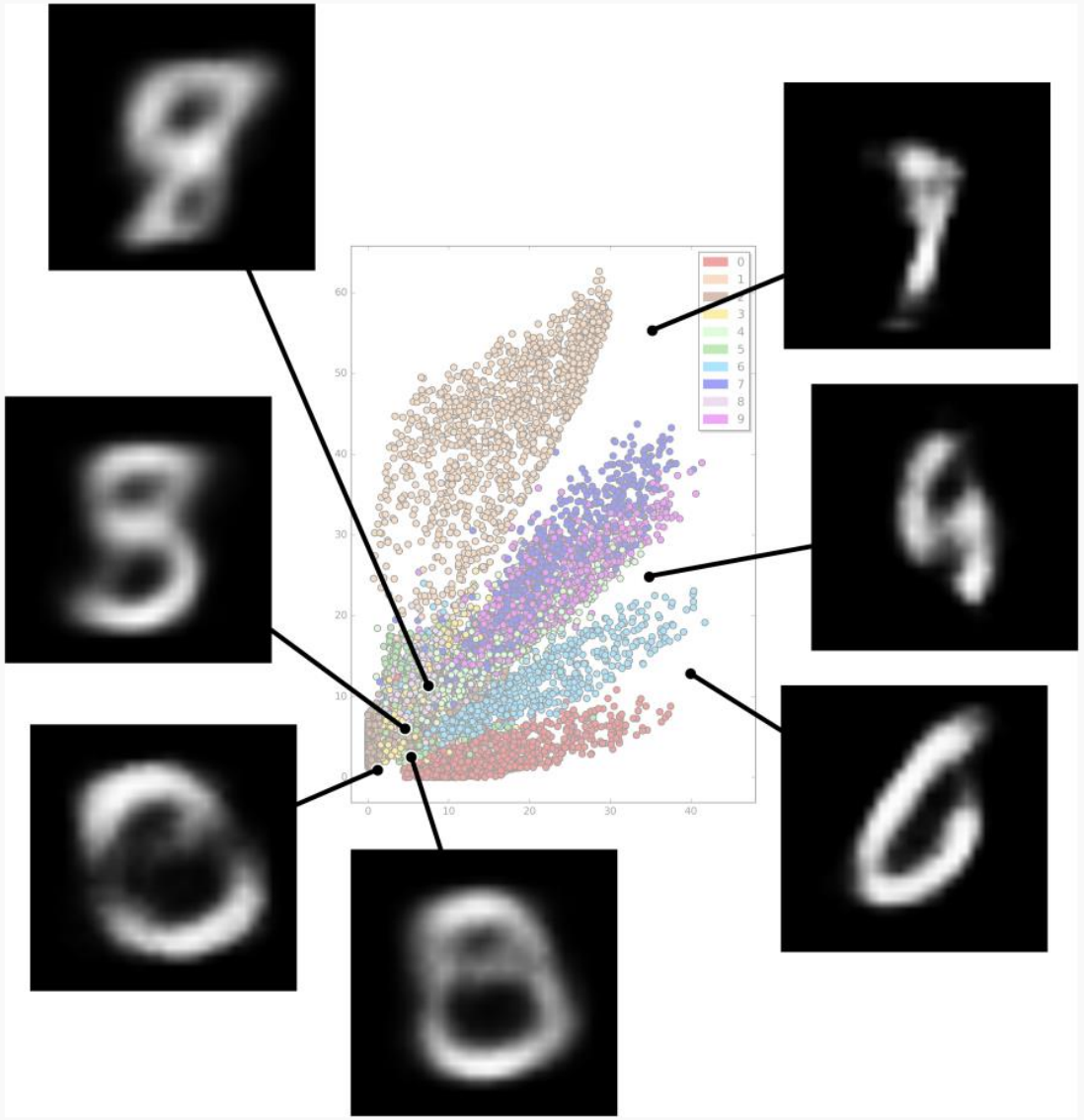


Application – Watermark removal

- It can also be used for removing watermarks from images or removing any object while filming a video or a movie

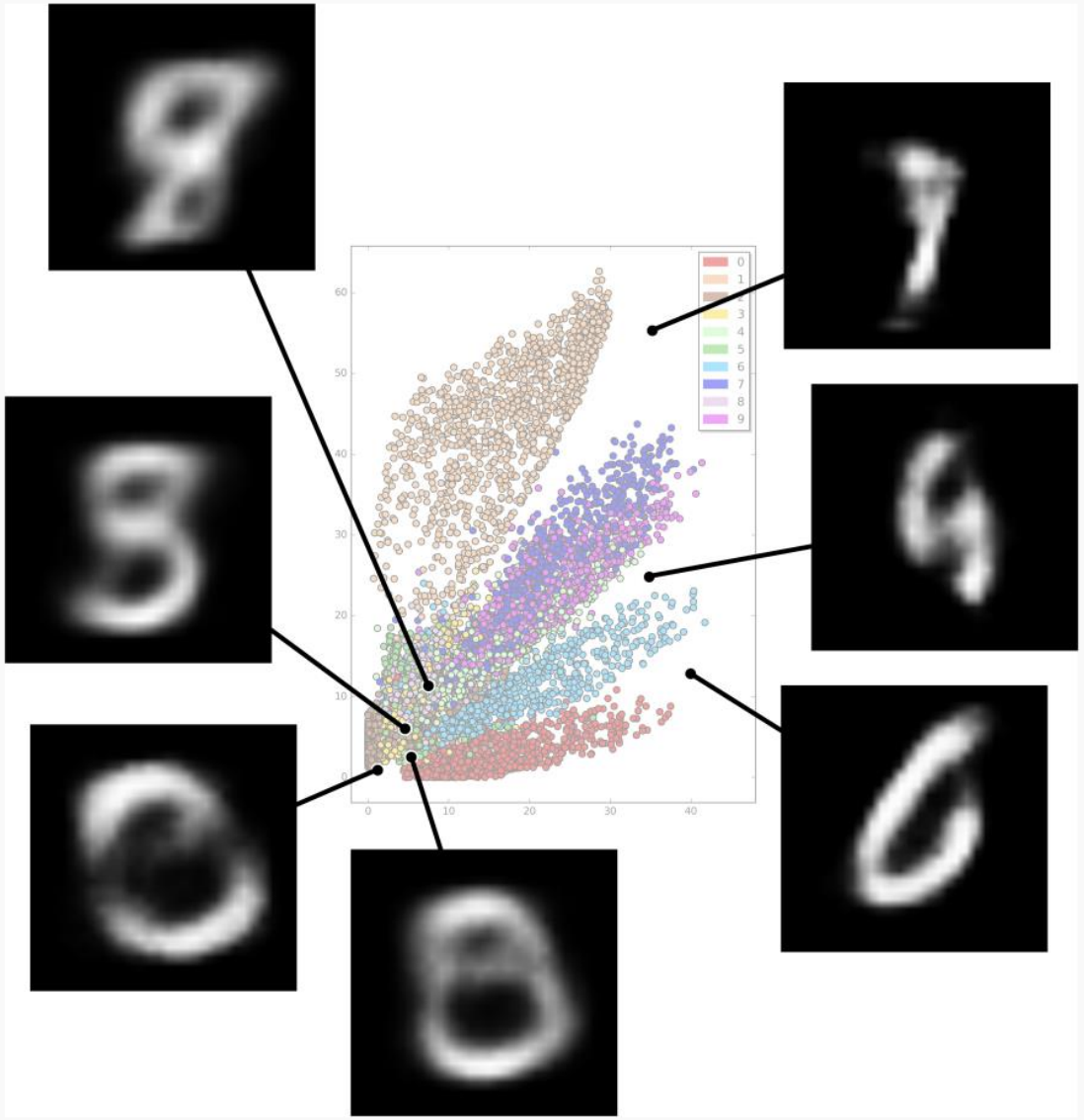
Challenges

- Gaps in the latent space or discrete latent space
- Separability in the latent space



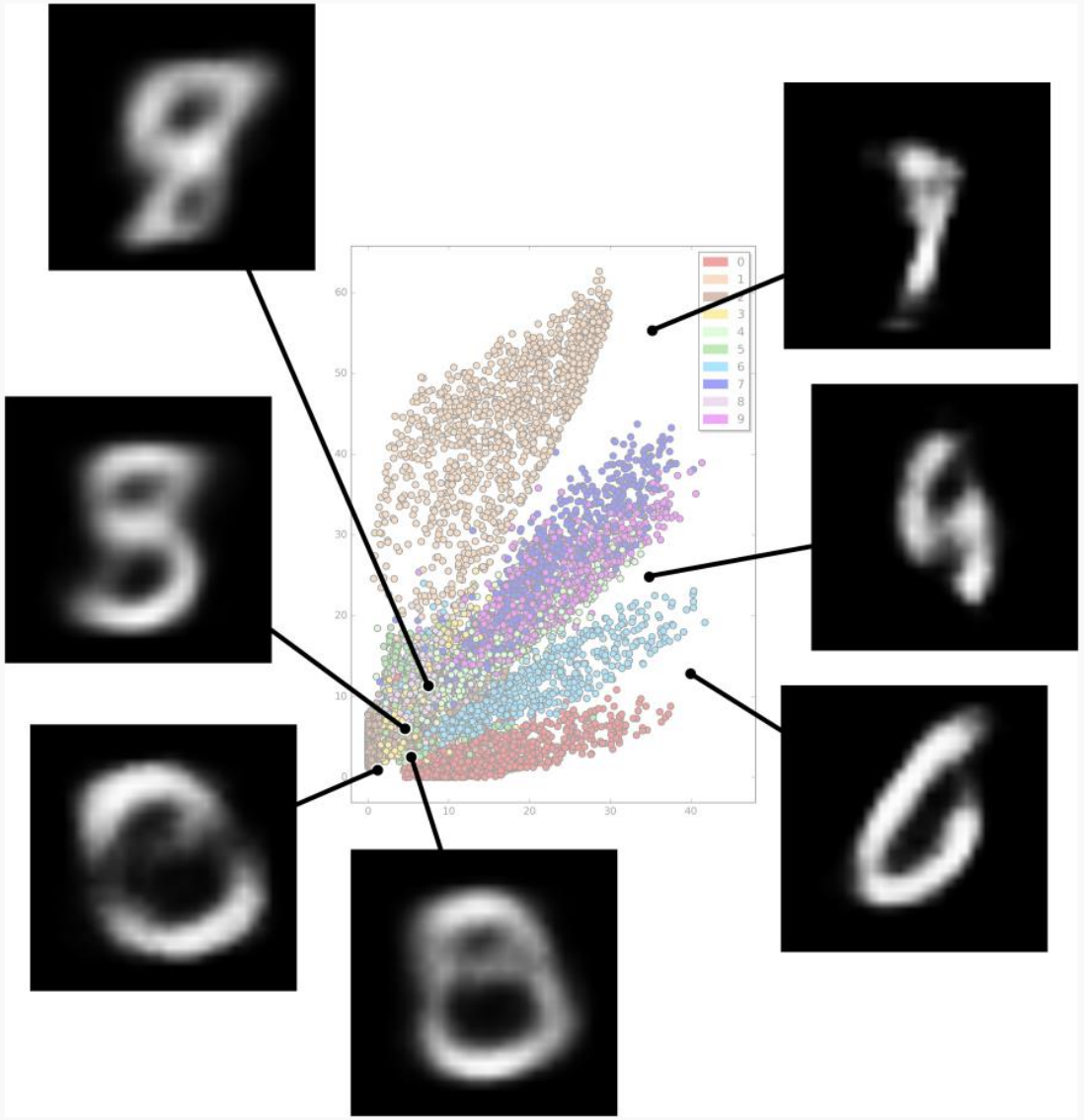
Challenges (cont.)

- The latent space may contain gaps
- There is no clue what the data points in these gaps might look like
- Similar with the problem of lacking data in supervised learning, as our model hasn’t seen these cases



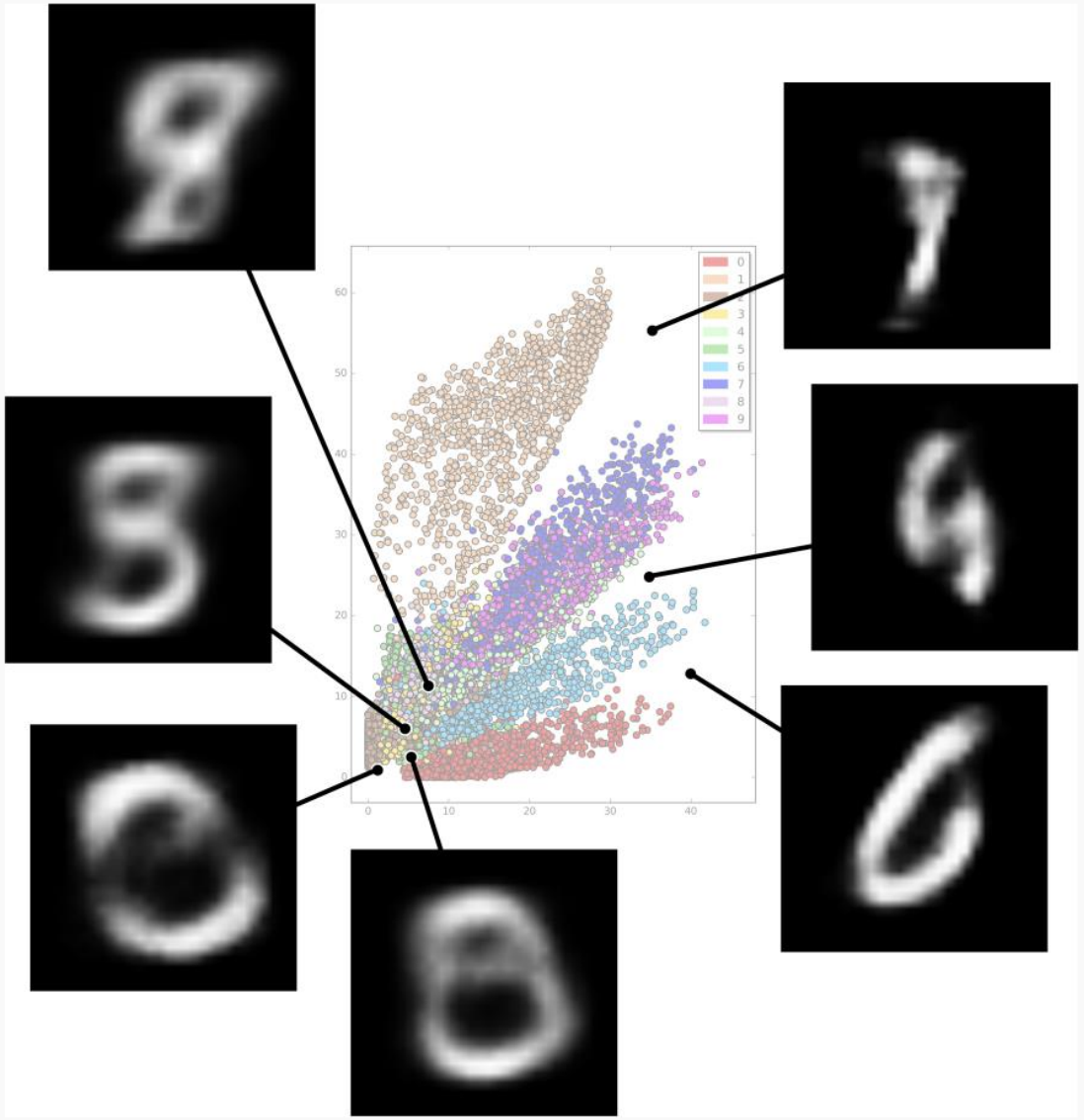
Challenges (cont.)

- The inability to study a continuous latent space
- E.g. We do not have a statistical model that has been trained for arbitrary input (and would not even if we closed all gaps in the latent space)
- A possible solution: generative models



Challenges (cont.)

- Separability of the spaces
- Some numbers are well separated but there are some overlapping regions with multiple possible labels, making it difficult to assign a unique feature to the data points



Feature Selection

Feature selection

- The most important problem with PCA and Autoencoder is that the new feature is some combination of existing features, which might be hard to interpret

- In the example of predicting the health condition of some students, we have the following feature:
  - Weight in kilogram
  - Height in inch
  - Height in cm
  - Hours of sport per day
  - Favorite color
  - Scores in math

Feature selection (cont.)

- In the example
  - The new feature produced by PCA could be some thing like $0.3 * height + 0.62 * weight$
  - The new feature produced by Autoencoder could be some thing like $(0.1 * height + weight/height)^2$
- All of these new features are hard to interpret
- We can *select* a subset of feature, instead of *transform* the original features into a smaller set

Feature selection (cont.)

- *Feature Selection* is a process to choose an optimal subset of features according to some criterion

- Why we need FS:
  1. to improve performance (in terms of speed, predictive power, simplicity of the model)
  2. to visualize the data for model selection
  3. To reduce dimensionality and remove noise
  4. Easy to understand

Feature types

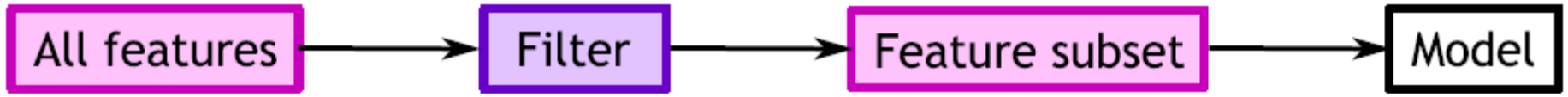
- Relevant features - those that we need to perform well
- Irrelevant features - those that are simply unnecessary
- Redundant features - those that become irrelevant in the presence of others
- Feature selection works by finding the relevant feature and eliminating the irrelevant feature and redundant feature

Feature selection algorithms

- Can be classified into 3 main categories:
  - Filters
  - Wrappers
  - Embedded methods

Filter

- selects a subset of variables independently of the model that shall use them



- It is a one-shot process (not iterative)
- It provides a set of “the most important” variables as the resulting subset *independently of the employed model*

Example – mRMR

- Minimum-redundancy-maximum-relevance (mRMR) algorithms tries to find a subset of features with the maximum relevance to the label while minimize the redundancy within the features

- The relevance of a feature set *S* for the class *c* is defined by the average value of all mutual information values between the individual feature f_i and the class c as follows

$$D(S,c) = \frac{1}{|S|} \sum_{f_i \in S} I(f_i;c).$$

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \left(\frac{p(x,y)}{p(x)\,p(y)} \right).$$

Mutual information

- Mutual information is the KL-divergence between probability distribution of $p(x, y)$ and $p(x)p(y)$
  - The latter is the probability density at x and y, if they are independent
- $I(X;Y) = \sum_x \sum_y p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$
- $= \sum_x \sum_y p(x|y)p(y) \log \frac{p(x|y)p(y)}{p(x)p(y)}$
- $= \sum_y p(y) \sum_x p(x|y) \log p(x|y) - \sum_x \left(\sum_y p(x|y)p(y) \right) \log p(x)$
- $= -H(X|Y) + H(X)$
- which is just the information gain

Example – mRMR (cont.)

- The redundancy of all features in the set *S* is the average value of all mutual information values between the feature f_i and the feature f_j

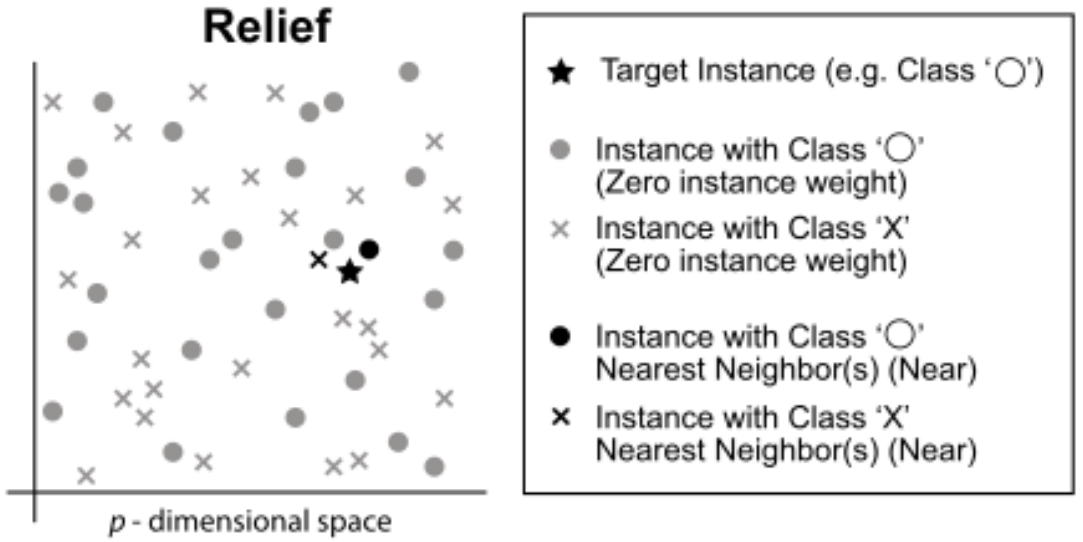
$$R(S) = \frac{1}{|S|^2} \sum_{f_i, f_j \in S} I(f_i; f_j)$$

- The mRMR criterion is a combination of two measures given above and is defined as follows:

$$\mathrm{mRMR} = \max_S \left[\frac{1}{|S|} \sum_{f_i \in S} I(f_i; c) - \frac{1}{|S|^2} \sum_{f_i, f_j \in S} I(f_i; f_j) \right]$$

Example – Relief

- Relief feature scoring is based on the identification of feature value differences between nearest neighbor instance pairs
  - If a feature value difference is observed in a neighboring instance pair with the same class (a 'hit'), the feature score decreases
  - If a feature value difference is observed in a neighboring instance pair with different class (a 'miss'), the feature score increases

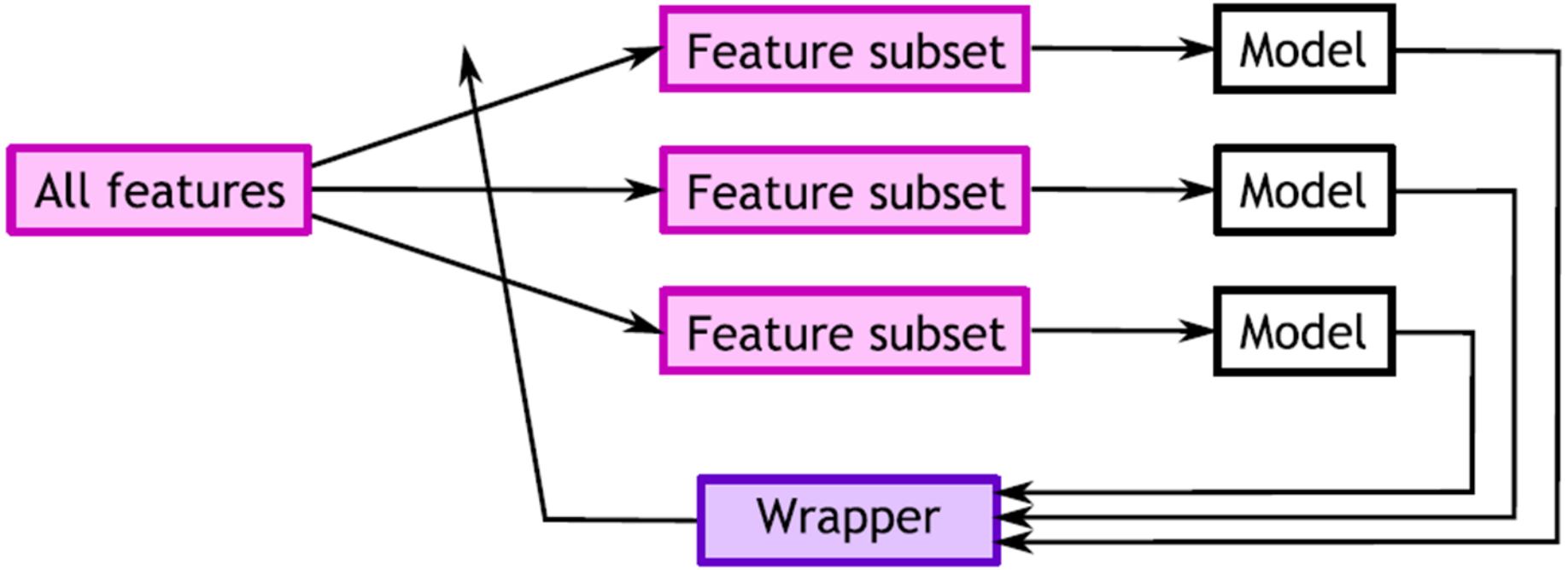


Example – Relief (cont.)

- The general workflow of Relief algorithm
- For each feature f_i, initialize its score w_i to be 0
- For each feature f_i
  - For each sample s_j, find its nearest hit h and nearest miss m
    - if s_j and h have different value at f_i, w_i decrease by 1
    - if s_j and m have different value at f_i, w_i increase by 1
- Return the top k features with highest score

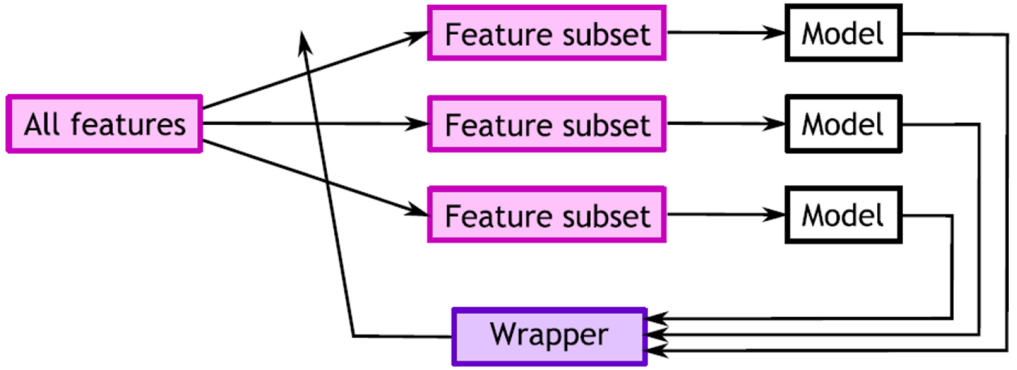
Wrapper

- **Wrapper:** selects a subset of variables taking into account the model that will use them



Wrapper

- **Wrapper:** selects a subset of variables taking into account the model that will use them



  - It is an iterative process
  - In each iteration, several subsets of input variables is generated and tested on the particular model type
  - According to the success of the model for individual feature subsets, it is chosen which subsets will be tested in the next iteration
  - Feature selection is here part of the model training; we need separate testing data to evaluate the final model error

Example – SFS

- Sequential Forward Selection (SFS) is a greedy search algorithm that attempts to find the “optimal” feature subset by iteratively selecting features based on the classifier performance
  - Start with an empty set S, represent all features as set F
  - Select the feature f from F, which can help the model achieve the highest score on S+f
  - S = S+f, F = F-f
  - Repeat the steps until there are k features in S

Embedded

- The embedded algorithm integrates the feature selection process into the learning tasks
- A famous example is the L1-regularization, or LASSO algorithm
- The feature selection process is embedded in the learning process