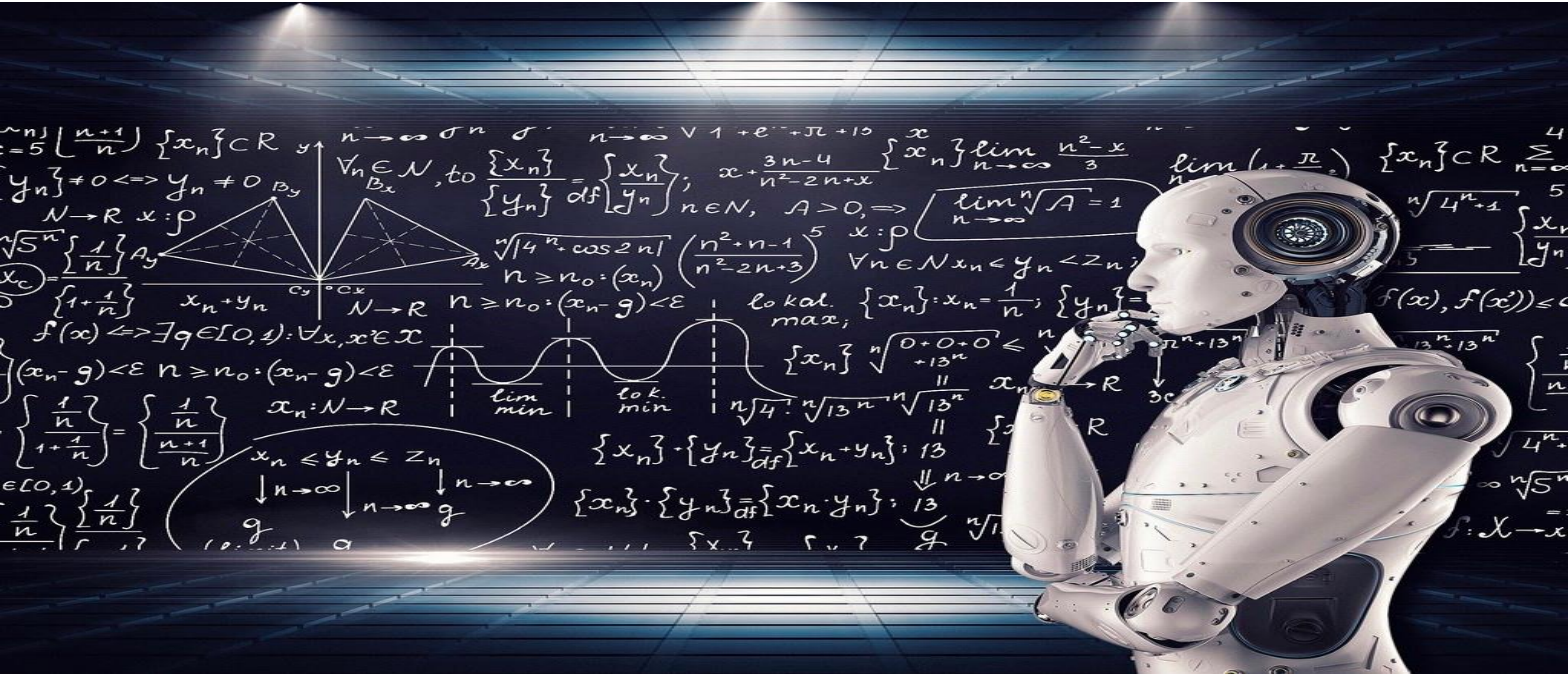
Dimensionality Reduction



IDC HERZLIYA

Agenda

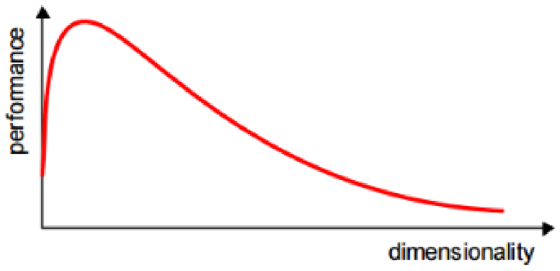
- Why?
- Methods
  - Feature selection
    - Filter
    - Wrapper
    - Embedded
  - Feature extraction
    - PCA

Why Reduce Dimensionality?

- The computation perspective
  - Reduces time complexity – Less computation
  - Reduces space complexity – Less parameters
- The modeling perspective
  - Information is less scattered – in a high dimension everything is far
  - Reduces sample complexity
  - More interpretable; simpler explanation

Why Reduce Dimensionality?

- The curse of dimensionality cause the data to be sparse in high features space
- In order to maintain the density we need to increase the training example exponentially
- In practice, the curse of dimensionality means that, for a given sample size, there is a maximum number of features above which the performance of our classifier will degrade rather than improve

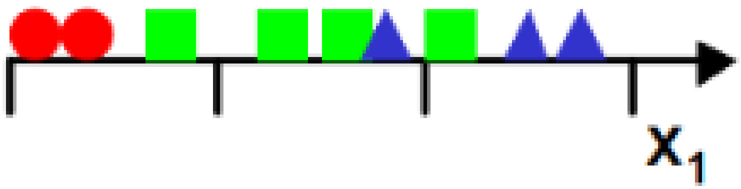


Example I

X1	X2	X3	X4	Y
10	14	24	1	1
50	2	5	5	1
5	16	30	0.5	2
30	10	2	3	3
10	23	21	1	2

Example II

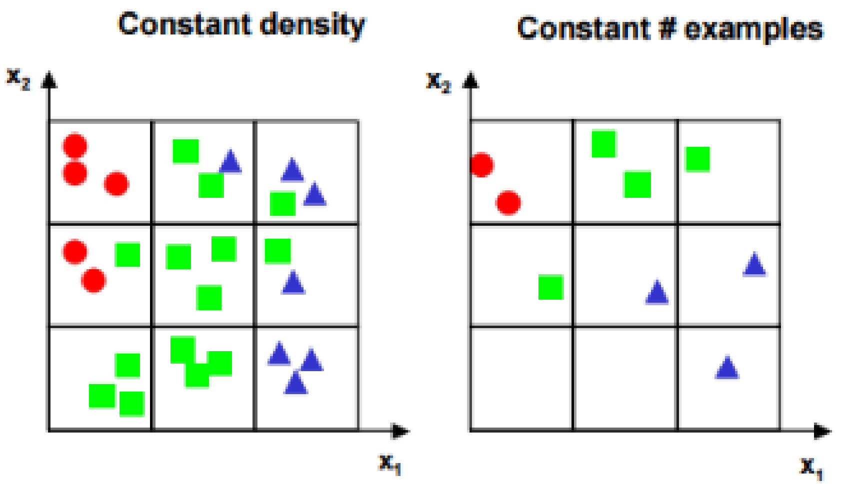
- Consider a 3-class classification problem
- Simple approach:
- Divide the feature space into uniform bins
- Compute the ratio of examples for each class at each bin
- For a new example, find its bin and choose the predominant class in that bin
- In 1 feature dimension with 9 samples:



- But there is too much overlap – bad classification

Example II

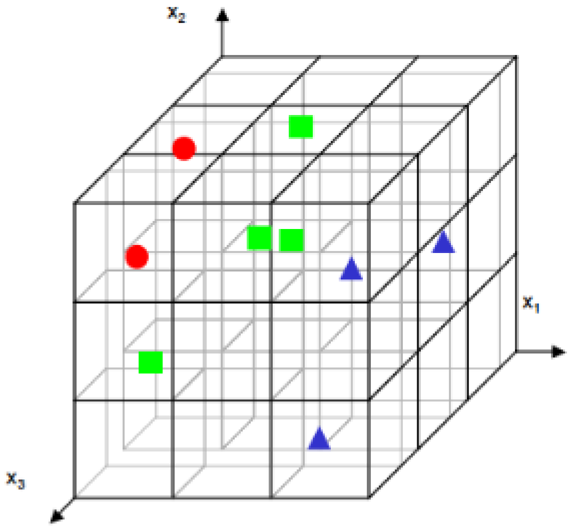
- We decide to incorporate a second feature (9 bins instead of 3):



- Maintain the number of examples results in a 2D scatter plot that is very sparse
- Maintain the density increases the number of examples from 9 (in 1D) to 27 (in 2D)

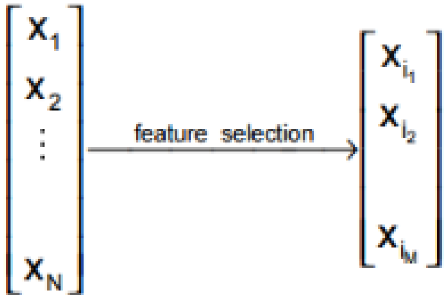
Example II

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

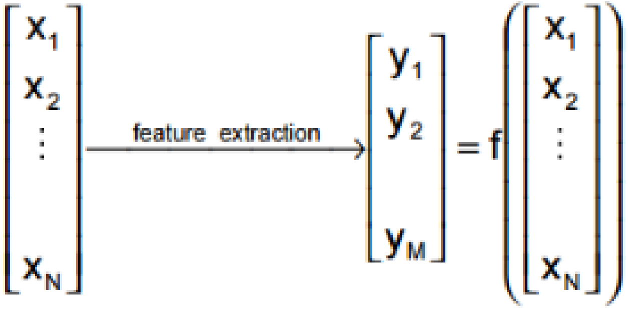


Reducing features space

- We will talk about 2 processes:
  - Feature selection
    - The process of selecting a subset of relevant features for use in model construction



  - Feature extraction
    - Creates new set of features that better represent the data (usually in lower dimension)



Feature selection

- We want to find a way to select features in reasonable time (polynomial)
- 3 approaches for feature selection:
  - Wrapper
    - In every step try to learn with a subset of features and select the subset with the best accuracy (less error)
    - Must be greedy selection – otherwise will be exponentially
  - Filter
    - Run a test for each feature and select the ones with the highest scores
  - Embedded
    - During the learning phase the algorithm calculate ‘importance score’

Feature selection - Wrapper

- How many feature subset possible? 2^n
- Forward
  - Start with empty set
  - In each iteration add the feature that most increases the accuracy
  - Stop when you reach a predefined accuracy / number of features
- Backward
  - Start with a full set
  - In each iteration remove the feature that minimally reduces the accuracy
  - Stop when you reach a predefined accuracy / number of features

Feature selection - Wrapper

- More search strategies
  - Greedy search on part of the space, exponential search on the other (small part)
  - Hybrid – Plus L Minus R
  - Bidirectional
  - Random

Feature selection - Filter

- Select subset of variables, independently of the learning model
- Provide generic selection of features, not tuned by a given learner (universal) – it can be drawback, selected features are not optimized for the used classifier

Feature selection - Filter

- One of the most basic idea is to find correlation to the target function
  - Pearson Correlation – Measures linear dependency
  - Mutual Information – Measures non-linear dependency

Feature selection - Filter

- Pearson Correlation

$$\rho(x_k, y) = \frac{\sum_{d=1}^{m}(x_k^{(d)} - \mu_k)(y^{(d)} - \mu_y)}{\sqrt{\sum_{d=1}^{m}(x_k^{(d)} - \mu_k)^2 \sum_{d=1}^{m}(y^{(d)} - \mu_y)^2}} = \frac{\sigma_{x_k y}}{\sigma_{x_k}\sigma_y}$$

- $x_k^{(d)}$ – the *d* value of the feature *k*
- $y^{(d)}$ – the *d* value of the function target *y*
- μ_k – the mean of the feature *k*
- μ_y – the mean of the function target *y*

Pearson Correlation – example I

$$\rho = \frac{\sum_{d=1}^{m}(x_k^{(d)} - \mu_k)(y^{(d)} - \mu_y)}{\sqrt{\sum_{d=1}^{m}(x_k^{(d)} - \mu_k)^2 \sum_{d=1}^{m}(y^{(d)} - \mu_y)^2}} = \frac{\sigma_{x_k y}}{\sigma_{x_k}\sigma_y}$$

X_k	Y
0.5377	0
1.8339	0
-2.2588	1
0.8622	1
0.3188	0
-1.3077	0
-0.4336	0
0.3426	1
3.5784	1
2.769	1

$$\mu_{\boldsymbol{k}} = 0.6243$$
$$\mu_y = 0.5$$

X_k-μ_k	Y-μ_y
-0.0866	-0.5
1.2096	-0.5
-2.8831	0.5
0.2379	0.5
-0.3055	-0.5
-1.932	-0.5
-1.0579	-0.5
-0.2817	0.5
2.9541	0.5
2.1447	0.5

$$\rho = 0.2587$$

Pearson Correlation – example II

$$\rho = \frac{\sum_{d=1}^{m}(x_k^{(d)} - \mu_k)(y^{(d)} - \mu_y)}{\sqrt{\sum_{d=1}^{m}(x_k^{(d)} - \mu_k)^2 \sum_{d=1}^{m}(y^{(d)} - \mu_y)^2}} = \frac{\sigma_{x_k y}}{\sigma_{x_k}\sigma_y}$$

X	Y
0.1	0
0.2	0
0.3	0
0.4	0
0.5	0
0.6	1
0.7	1
0.8	1
0.9	1
1	1

X	Y
1	0
2	0
3	0
4	0
5	0
6	1
7	1
8	1
9	1
10	1

X	Y
-1	0
-2	0
-3	0
-4	0
-5	0
-6	1
-7	1
-8	1
-9	1
-10	1

X	Y
-1	1
-2	1
-3	1
-4	1
-5	1
-6	0
-7	0
-8	0
-9	0
-10	0

$\rho 1 = 0.8704$

$\rho 2 = 0.8704$

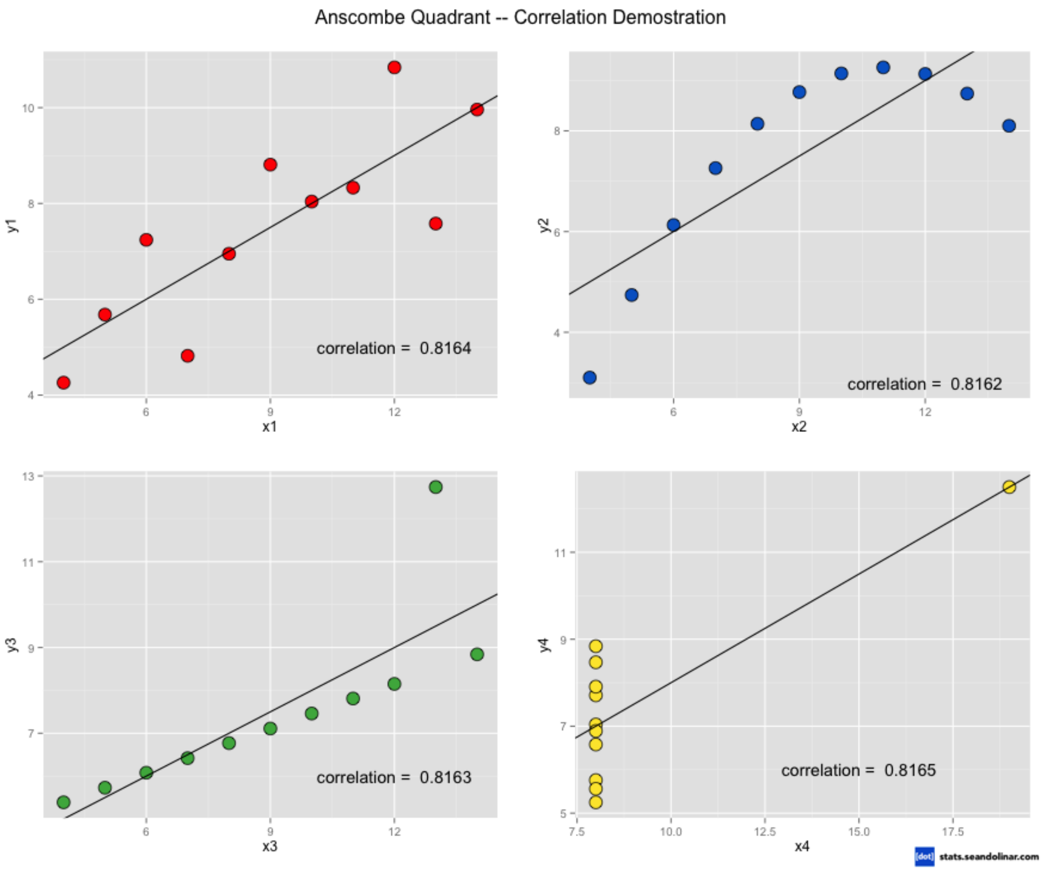
$\rho 3 = -0.8704$

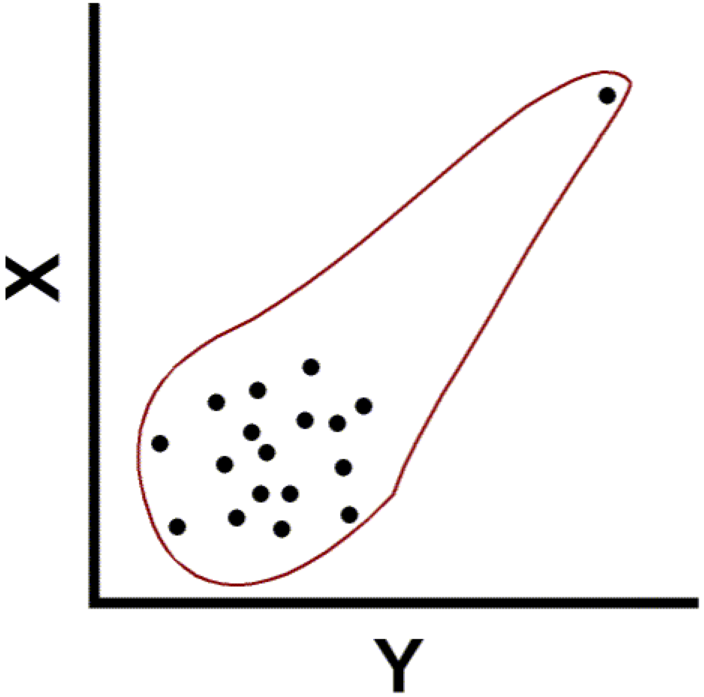
$\rho 4 = 0.8704$

Feature selection - Filter

- Pearson Correlation
  - 1 – total positive correlation
  - 0 – no correlation
  - -1 – is total negative correlation

Pearson Correlation





From Pearson Correlation to Spearman rank correlation

- Pearson correlation is very sensitive to the actual values of the data and to outliers
- Pearson (and its relatives) measure linearity of the data. This is not always the correct model or desired observation/assessment.
  (a strong non-linear relationship can exist but Pearson might not get too excited…)
- It is often much more robust to use rank based correlation measures

Spearman rank correlation

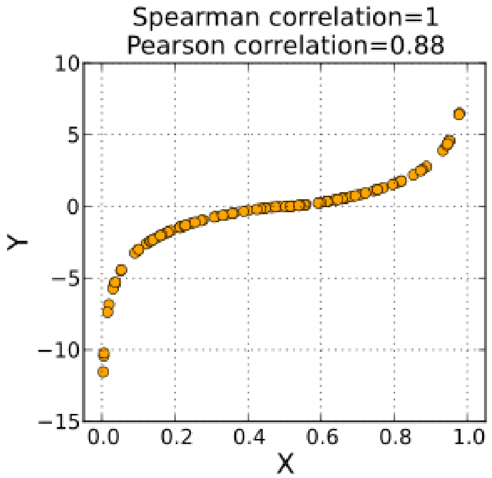
- Transform the data into ranks
- Calculate the correlation on the ranks
- What are the possible values? $-1 \leq SRC \leq 1$

X_k	Y
0.5377	0
1.8339	0
-2.2588	1
0.8622	1
0.3188	0
-1.3077	0
-0.4336	0
0.3426	1
3.5784	1
2.769	1

X_k	Y
6	0
8	0
1	1
7	1
4	0
2	0
3	0
5	1
10	1
9	1

$\rho = 0.2587$

$r_s = 0.3133$



Embbeded

- Which algorithm can calculate feature importance during the learning?
  - Decision Trees – weighted goodness of split
  - Regression – when the features have the same scale

Feature extraction

- We want to reduce the dimension, but to preserve most of the information
- In other word, we want to take the original features space, probably with correlation between the features, and summarizes it by uncorrelated axes
- First, how we measure information?
  - One way is the Variance
- So, we want to project the data to new axes and preserve the variance

Feature extraction - projection

- A projection is a transformation of data points from one axes system to another
  - Translate – Subtracting the mean vector from all data points, and this is equivalent to moving the center of the data to the origin

$$\begin{pmatrix} x_{1,1} - \mu_1 & \cdots & x_{1,d} - \mu_d \\ \vdots & \ddots & \vdots \\ x_{N,1} - \mu_1 & \cdots & x_{N,d} - \mu_d \end{pmatrix}$$

  - Rotate – Project to the new axes using a dot product $(x_i - \mu) \cdot a_j$

PCA - Principle Component Analysis

- Projection Matrix
  - A full projection is defined by a matrix in which each column is a vector defining the direction of one of the new axes

$$A = [a_1, a_2, \cdots, a_k] \in \mathbb{R}^{dxk}$$

  - We further require that these vectors will be orthogonal, and often also orthonormal (of a unit length), thus $A^T A = I$

PCA – Finding the first PC

- Let $x'_1 = a_1^T X^T$ be the data projected on the first PC, a_1

$$Var[x'_1] = E[(x'_1 - \mu'_1)^2] = E\left[{x'_1}^2\right]$$

$$= E[(a_1^T X^T)^2] = a_1^T E[X^T X] a_1 = a_1^T S a_1$$

  - Where S (sometimes notated Σ) is the covariance matrix of X

- We want to find a_1 that maximizes $Var[x'_1]$ subject to the constraint $a_1^T a_1 = 1$
- This is optimization problem…

PCA – Finding the first PC

- We use Lagrange to find the max of:

$$a_1^T S a_1 - \lambda(a_1^T a_1 - 1)$$

- The derivative with respect to a_1:

$$2Sa_1 - 2\lambda a_1 = 0$$

$$Sa_1 = \lambda a_1$$

- This means that $a_1$is the eigenvector of S and λ is the corresponding eigenvalue
- Furthermore, we can substitute the result in the original equation:

$$Var[x'_1] = a_1^T S a_1 = a_1^T \lambda a_1 = \lambda a_1^T a_1 = \lambda$$

- And we got that $\max(Var[x'_1]) = \max(\lambda)$

PCA – Finding the PC's

- We can prove that the last equation hold for all PC's, and get:

$$Var[x'_i] = \lambda_i$$

$$\max(Var[x'_i]) = \max(\lambda_i)$$

- The PC's are the eigenvectors of S (the covariance matrix of X) and the λ are the corresponding eigenvalues

- The PC's that preserve most of the information are those with the largest λ

PCA – algorithm

- Build the covariance matrix *S*
- Find Eigenvectors and Eigenvalues of *S* by solving $(S - \lambda_i I)a_i = 0$
- Sort Eigenvectors by Eigenvalues
- Build the matrix A_j using j eigenvectors with the j greatest eigenvalues
- Transform $x' = A_j(x - \mu)$

PCA – example

- 2 features space, 10 instances

	x_1	x_2
	1.4	1.65
	1.6	1.975
	-1.4	-1.775
	-2	-2.525
	-3	-3.95
	2.4	3.075
	1.5	2.025
	2.3	2.75
	-3.2	-4.05
	-4.1	-4.85
mean	-0.45	-0.5675
var	6.422778	9.952785

Mean →

	$x_1 - \mu_1$	$x_2 - \mu_2$
	1.85	2.2175
	2.05	2.5425
	-0.95	-1.2075
	-1.55	-1.9575
	-2.55	-3.3825
	2.85	3.6425
	1.95	2.5925
	2.75	3.3175
	-2.75	-3.4825
	-3.65	-4.2825
mean	0	0
var	6.422778	9.952785

- The covariance matrix

$$S = X^T X = \begin{pmatrix} 6.4228 & 7.9876 \\ 7.9876 & 9.9528 \end{pmatrix}$$

PCA – example

- Finding the eigenvalues:

$$\det(S - \lambda_i I) = 7.9876^2 - (6.4228 - \lambda)(9.9528 - \lambda) = 0$$

$$\lambda^2 - 16.3756\lambda + 0.123 = 0$$

- We get $\lambda = 16.3681, 0.0075$
  - Note $\lambda_1 + \lambda_2 = 16.3756 = 6.4228 + 9.9528 = Var(x_1) + Var(x_2)$

PCA – example

- Finding the eigenvectors:

$$(S - \lambda_i I)a_i = 0$$

$$\begin{pmatrix} 6.4228 - 16.3681 & 7.9876 \\ 7.9876 & 9.9528 - 16.3681 \end{pmatrix} \begin{pmatrix} a_1^{(1)} \\ a_1^{(2)} \end{pmatrix} = 0$$

$$\begin{pmatrix} -9.9453 & 7.9876 \\ 7.9876 & -6.4153 \end{pmatrix} \begin{pmatrix} a_1^{(1)} \\ a_1^{(2)} \end{pmatrix} = 0$$

- We get

$$a_1^{(1)} = 0.6262, a_1^{(2)} = 0.7797$$

  - Note that $a_1^T a_1 = 1$

- We find a_2 with λ_2, and get

$$a_2^{(1)} = 0.7797, a_2^{(2)} = -0.6262$$

PCA – example

- Build A_j and transform $x' = A_j(x - \mu)$

	$x_1 - \mu_1$	$x_2 - \mu_2$
	1.85	2.2175
	2.05	2.5425
	-0.95	-1.2075
	-1.55	-1.9575
	-2.55	-3.3825
	2.85	3.6425
	1.95	2.5925
	2.75	3.3175
	-2.75	-3.4825
	-3.65	-4.2825
mean	0	0
var	6.422778	9.952785

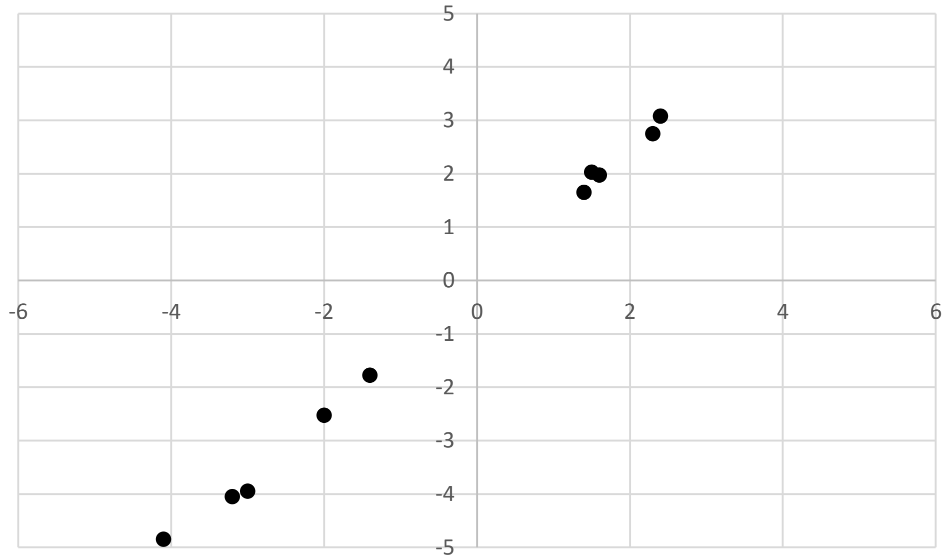
$$\xrightarrow{A_j = \begin{pmatrix} 0.6262 & 0.7797 \\ 0.7797 & -0.6262 \end{pmatrix}}$$

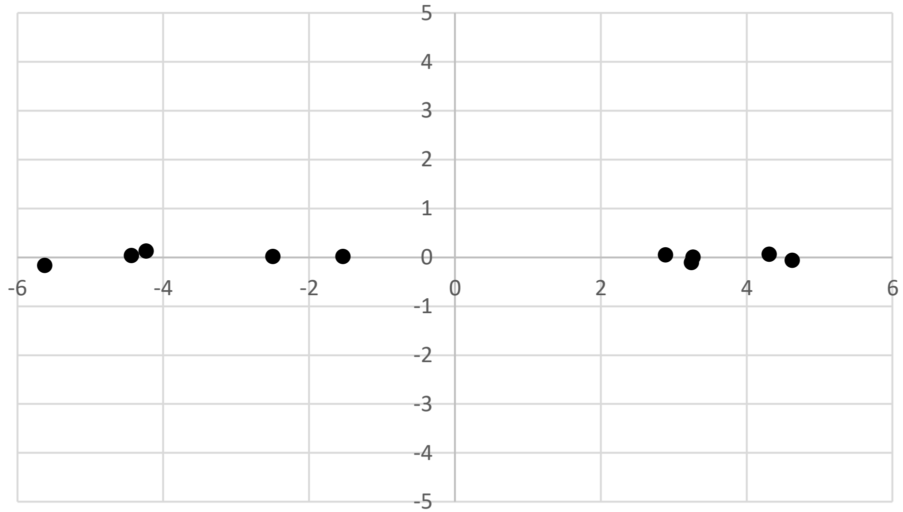
	x'_1	x'_2
	2.887455	0.053846
	3.266097	0.006272
	-1.53638	0.015422
	-2.49687	0.017252
	-4.23415	0.129887
	4.624727	-0.05879
	3.242462	-0.10301
	4.308705	0.066756
	-4.43736	0.036567
	-5.6247	-0.1642
mean	0	0
var	16.3681	0.0075

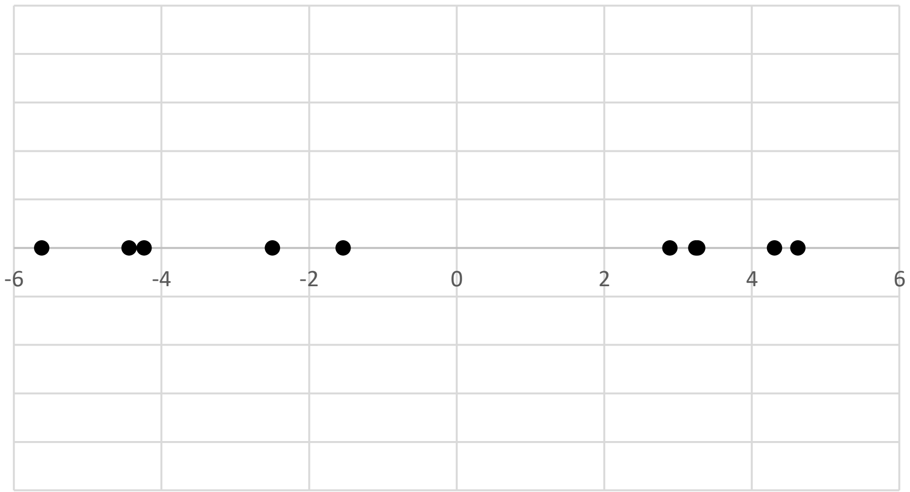
- So, if we measure the information with VAR, we can see that the first PC has

$$\frac{16.3681}{16.3681+0.0075} = 0.999$$

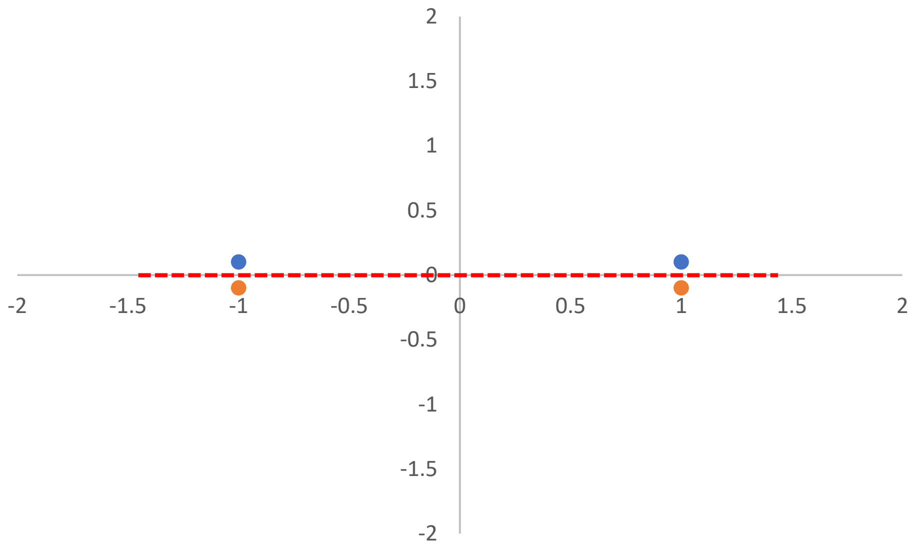
PCA – example visualization

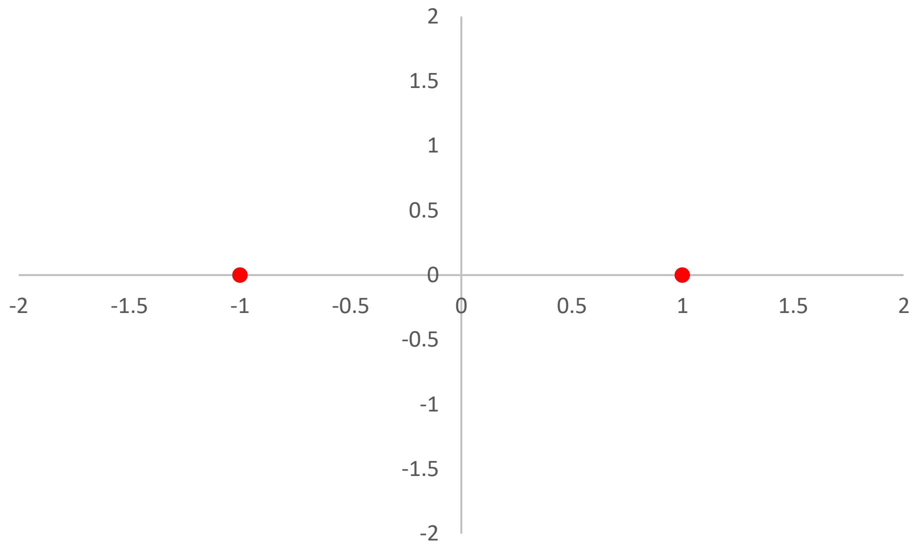






PCA – classification problem





Questions

?