kNN, Curse of Dimensionality and PCA

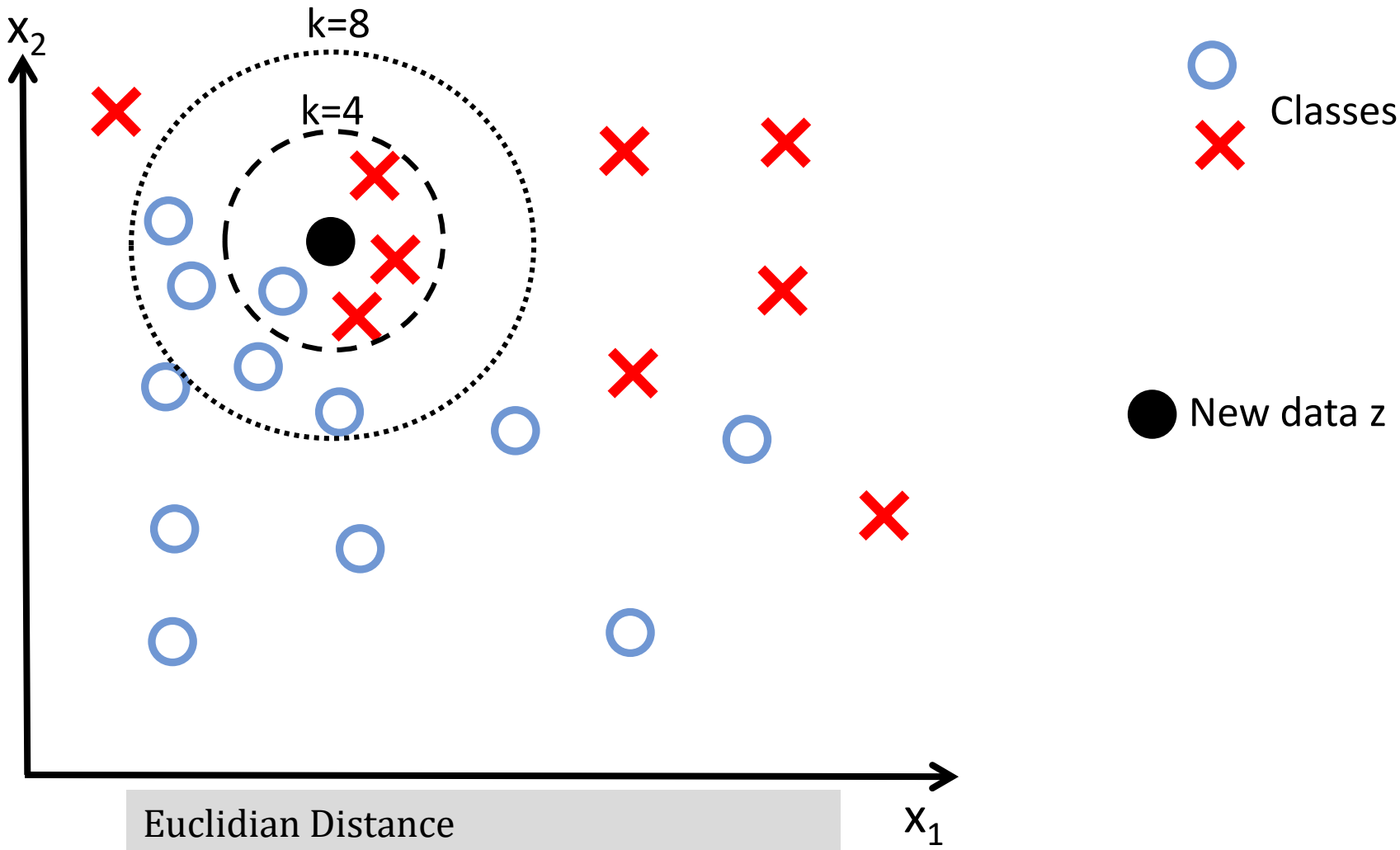
Prof. Dr. Stephan Trahasch

Offenburg University of Applied Sciences

Outline

- Curse of Dimensionality
- Principal Components Analysis - PCA
- Examples
- Summary

k-Nearest Neighbor (kNN) Classifier



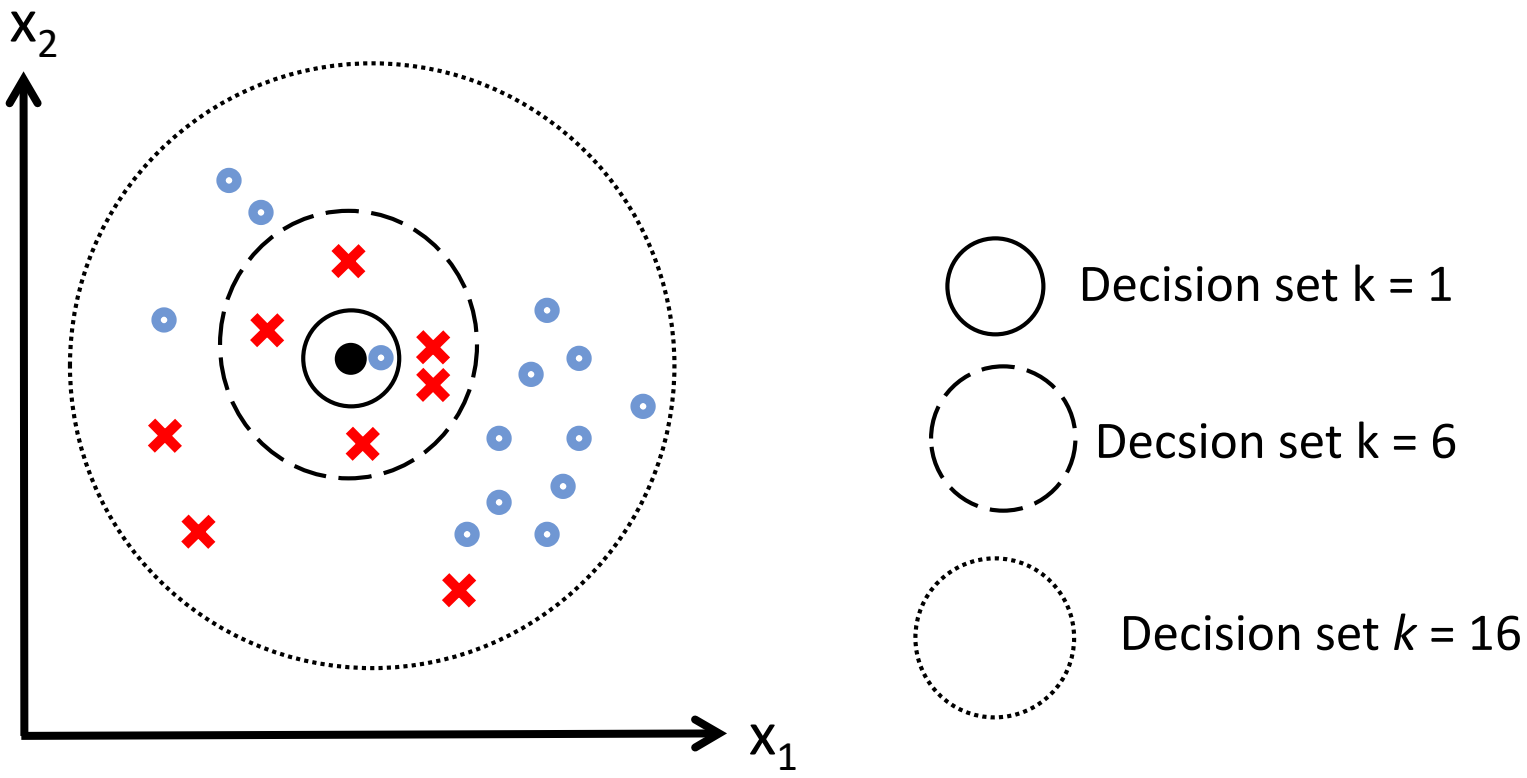
Euclidian Distance

$$d(x,z) = \sqrt{(x_1 - z_1)^2 + \cdots + (x_n - z_n)^2}$$

How to choose k?

k too small: Sensitive against outlier

k too large: Bad prediction, no local information



Main Idea of kNN

Find the k-nearest neighbors to a new data instance z

- rank the feature vectors according to Euclidean distance
- select the k vectors which are have smallest distance to z

Classification

- ranking yields k feature vectors and a set of k class labels
- pick the class label which is most common in this set (“voting”)
- classify z as belonging to this class

“Training” is trivial: just use training data as a lookup table, and search to classify a new datum

→ “Lazy Learner”

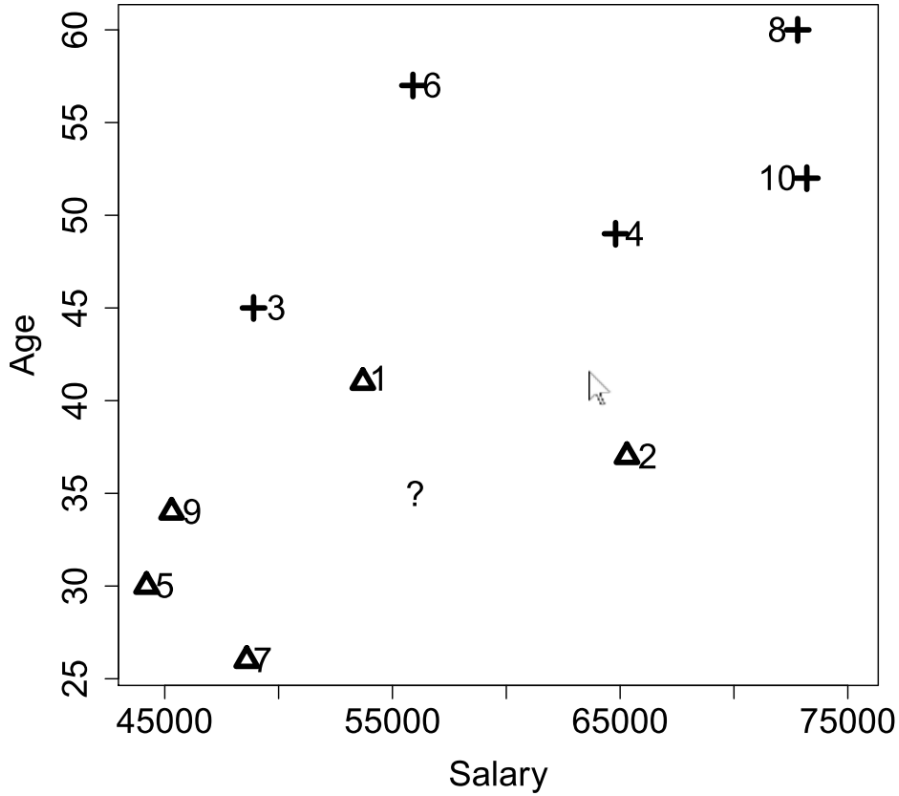
Basic Considerations

- k = number of neighbours considered
- Decision set the set of k-next neighbours considered for classification
- Decision rule How do you determine the class of the object to be classified from the classes of the decision set?
- Distance function defines the similarity for pairs of objects

Normalization

Marketing department wants to decide whether or not they should contact a customer with the following profile:

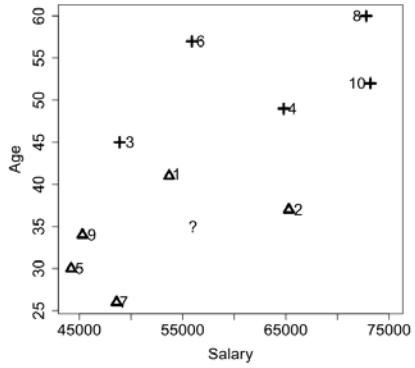
<Salary = 56, 000, age = 35>



Source: Fundamentals of machine learning for predictive data analytics.

Normalization

ID	Salary	Age	Purch.	Salary and Age Dist.	Salary and Age Neigh.	Salary Only Dist.	Salary Only Neigh.	Age Only Dist.	Age Only Neigh.
1	53700	41	No	2300.0078	2	2300	2	6	4
2	65300	37	No	9300.0002	6	9300	6	2	2
3	48900	45	Yes	7100.0070	3	7100	3	10	6
4	64800	49	Yes	8800.0111	5	8800	5	14	7
5	44200	30	No	11800.0011	8	11800	8	5	5
6	55900	57	Yes	102.3914	1	100	1	22	9
7	48600	26	No	7400.0055	4	7400	4	9	3
8	72800	60	Yes	16800.0186	9	16800	9	25	10
9	45300	34	No	10700.0000	7	10700	7	1	1
10	73200	52	Yes	17200.0084	10	17200	10	17	8



Source: Fundamentals of machine learning for predictive data analytics.

Normalization methods

- This odd prediction is caused by features taking different ranges of values, this is equivalent to features having different variances.
- We can adjust for this using normalization.

Normalization methods

z-transformation: $\forall x \in X: \quad z_i = \frac{x_i - \bar{x}}{s}$ bzw. $z_i = \frac{x_i - \mu}{\sigma}$

Normalization to interval [0, 1]: $\forall x \in X: \mathrm{x}' = \frac{x - \min(x)}{\max(x) - \min(x)}$

In general: $\mathrm{x}' = \frac{x - \min(x)}{\max(x) - \min(x)} * (\mathrm{high} - \mathrm{low}) + \mathrm{low}$

Normalizing data is an important thing to do for almost all machine learning algorithms

Normalized Dataset				Salary and Age		Salary Only		Age Only	
ID	Salary	Age	Purch.	Dist.	Neigh.	Dist.	Neigh.	Dist.	Neigh.
1	0.3276	0.4412	No	0.1935	1	0.0793	2	0.17647	4
2	0.7276	0.3235	No	0.3260	2	0.3207	6	0.05882	2
3	0.1621	0.5588	Yes	0.3827	5	0.2448	3	0.29412	6
4	0.7103	0.6765	Yes	0.5115	7	0.3034	5	0.41176	7
5	0.0000	0.1176	No	0.4327	6	0.4069	8	0.14706	3
6	0.4034	0.9118	Yes	0.6471	8	0.0034	1	0.64706	9
7	0.1517	0.0000	No	0.3677	3	0.2552	4	0.26471	5
8	0.9862	1.0000	Yes	0.9361	10	0.5793	9	0.73529	10
9	0.0379	0.2353	No	0.3701	4	0.3690	7	0.02941	1
10	1.0000	0.7647	Yes	0.7757	9	0.5931	10	0.50000	8

				Salary and Age		Salary Only		Age Only	
ID	Salary	Age	Purch.	Dist.	Neigh.	Dist.	Neigh.	Dist.	Neigh.
1	53700	41	No	2300.0078	2	2300	2	6	4
2	65300	37	No	9300.0002	6	9300	6	2	2
3	48900	45	Yes	7100.0070	3	7100	3	10	6
4	64800	49	Yes	8800.0111	5	8800	5	14	7
5	44200	30	No	11800.0011	8	11800	8	5	5
6	55900	57	Yes	102.3914	1	100	1	22	9
7	48600	26	No	7400.0055	4	7400	4	9	3
8	72800	60	Yes	16800.0186	9	16800	9	25	10
9	45300	34	No	10700.0000	7	10700	7	1	1
10	73200	52	Yes	17200.0084	10	17200	10	17	8

Source: Fundamentals of machine learning for predictive data analytics.

Examples of different values of k

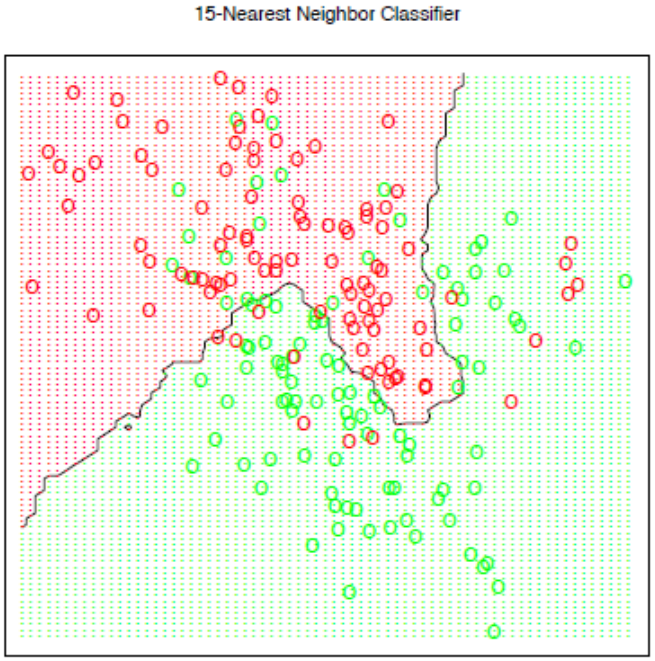


Figure 2.2: *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (GREEN $= 0$, RED $= 1$) *and then fit by* 15*-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the* 15-

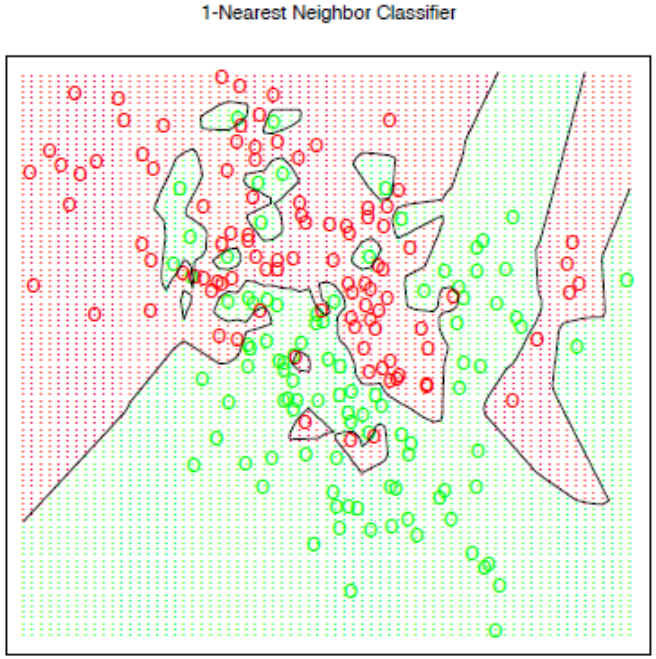


Figure 2.3: *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (GREEN $= 0$, RED $= 1$), *and then predicted by* 1*-nearest-neighbor classification.*

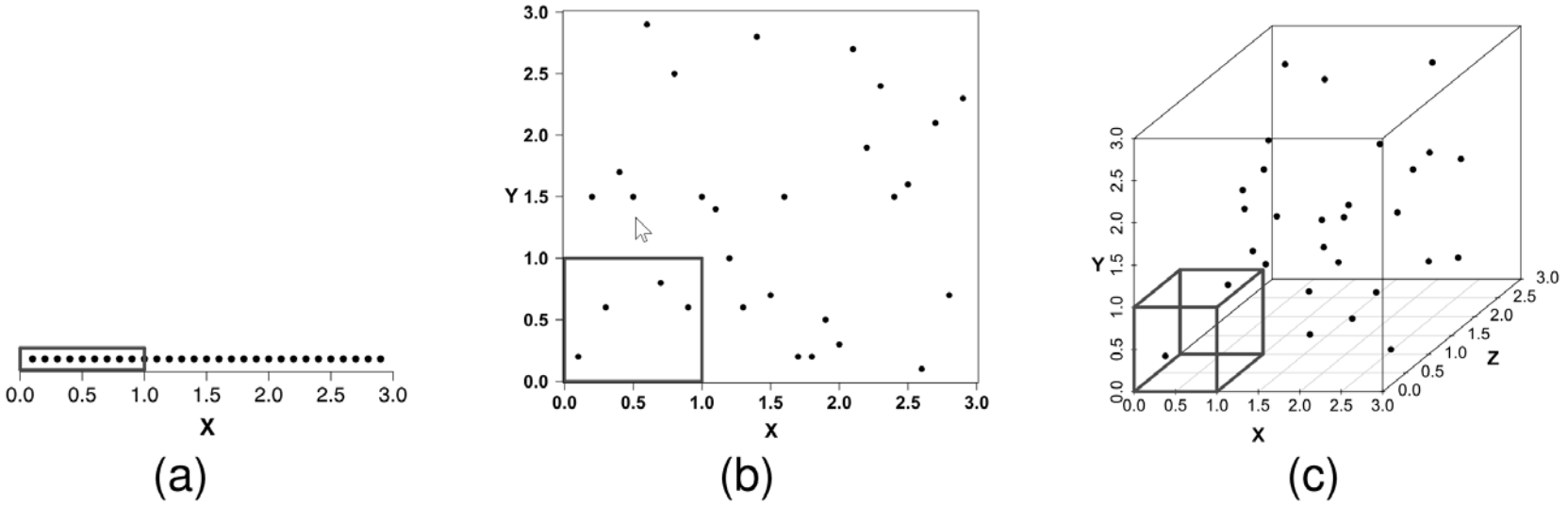
Source: Elements of Statistical Learning

Curse of Dimensionality (Bellman, 1961)

10^2 data points in the unit interval have a distance of 0.01;

Equal distribution in a 10-dimensional unit cube with a distance of 0.01 requires 10^{2*10} data points!

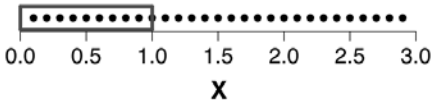
If 100 examples at p = 1 result in a dense space, you have to collect 100^{10} examples for the same density at p = 10.



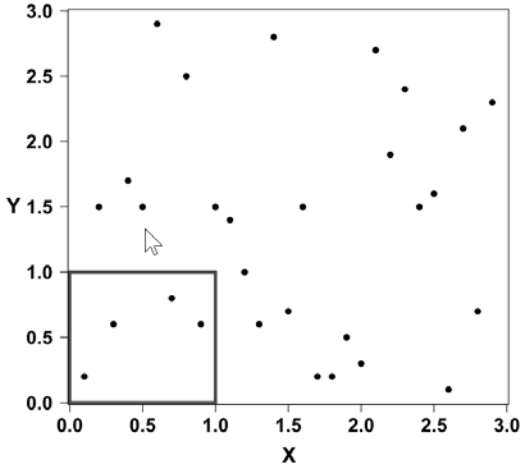
Across figures (a), (b) and (c) the density of the marked unit hypercubes decreases as the number of dimensions increases.

Source: Pattern recognition and machine learning

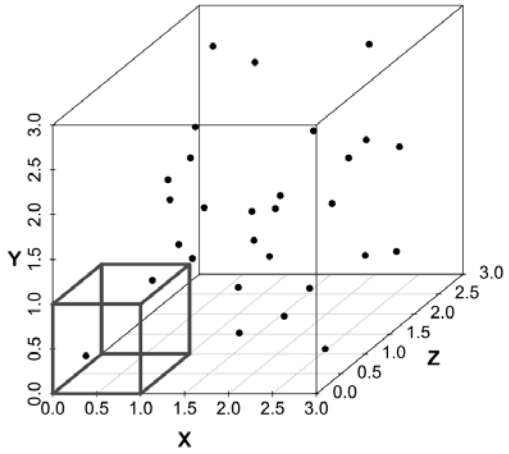
Curse of Dimensionality



(a)

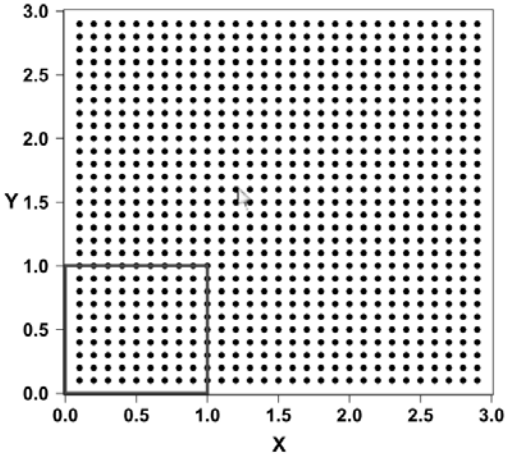


(b)

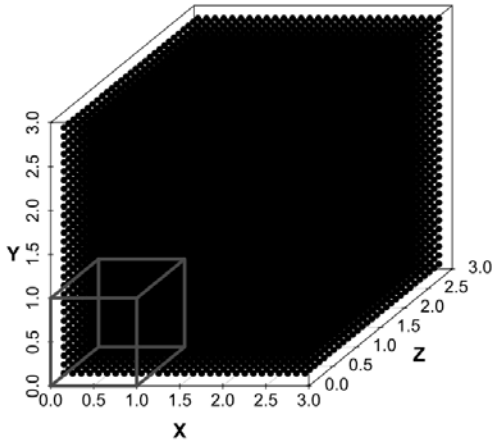


(c)

Figures (d) and (e) illustrate the cost we must incur if we wish to maintain the density of the instances in the feature space as the dimensionality of the feature space increases.



(d)



(e)

Curse of Dimensionality

Data in p-dimensional space are normally distributed.

The r-th part of the data is to be taken into account for kNN.

r-th part of the data around a point x is located in a cube with the length $e_p(r) = r^{\frac{1}{p}}$

With 10 dimensions me r = 0.01: $e_{10}(r) = r^{\frac{1}{10}} = 0.63$

With 10 dimensions me r = 0.1: $e_{10}(r) = r^{\frac{1}{10}} = 0.80$

This means that we need 80% of the possible values of each attribute Xi for at least 10% of the data in a neighbourhood, even at p = 10 dimensions! The density of data points in the neighbourhood is sparse with high dimensions.

Outline

- Curse of Dimensionality
- Principal Components Analysis - PCA
- Examples
- Summary

Motivation

- High-dimensional data
  - Images of faces
  - Text from articles
  - Sensor data
  - DAX stocks
- Can we describe them in a “simpler” way?

- Ex: Dax– vector of 30 (change in) values per day
- Some elements tend to “change together”
- Maybe we only need a few values to approximate it?
- “Tech stocks up 2x, manufacturing up 1.5x, …” ?

How can we access that structure?

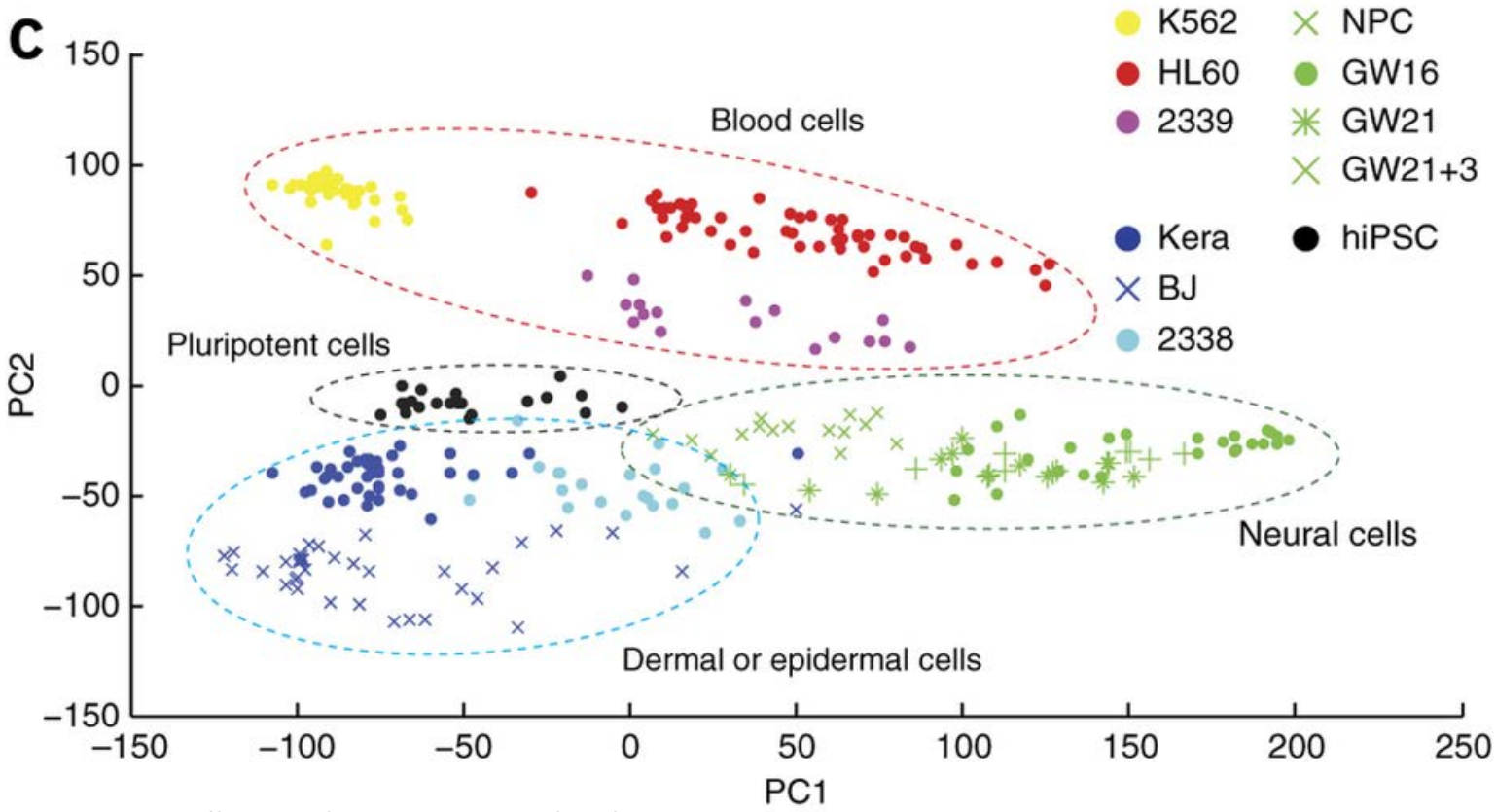
Motivation

- dimensionality reduction transforms a n-dimensional dataset to a k-dimensional dataset with k < n
- dataset compression
  - less memory storage consumption
  - machine learning algorithms run faster on low-
  - dimensional data

- data visualization
  high-dimensional data can be transformed to 2D or 3D for plotting

10,000 genes get compressed to a single dot in 2 dimensions

This graph was drawn from single-cell RNA-seq. There were about 10,000 transcribed genes in each cell. Each dot represents a single-cell and its transcription profile. The general idea is that cells with similar transcription should cluster.



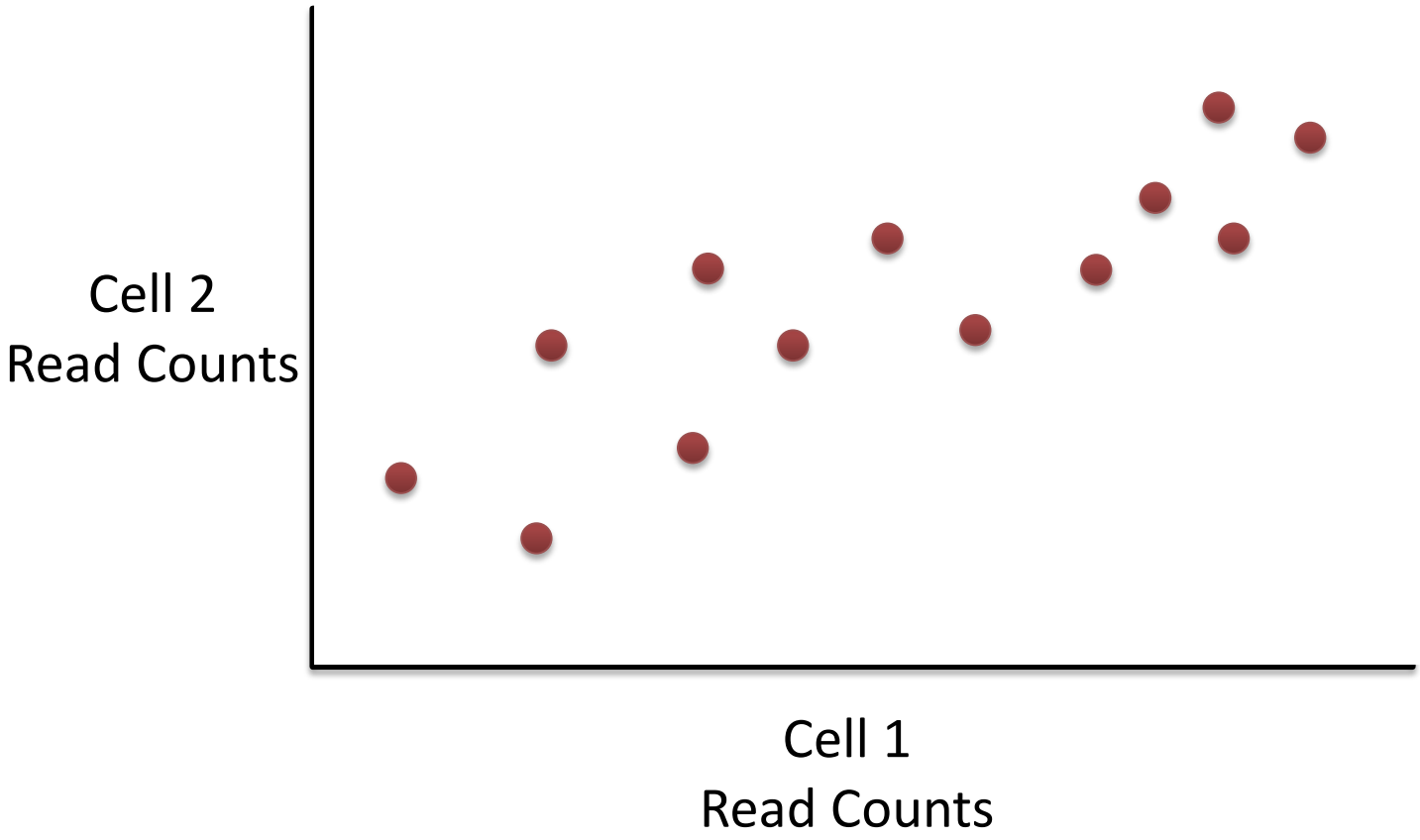
Pollen et al. Nature Biotechnology 2014

PCA example

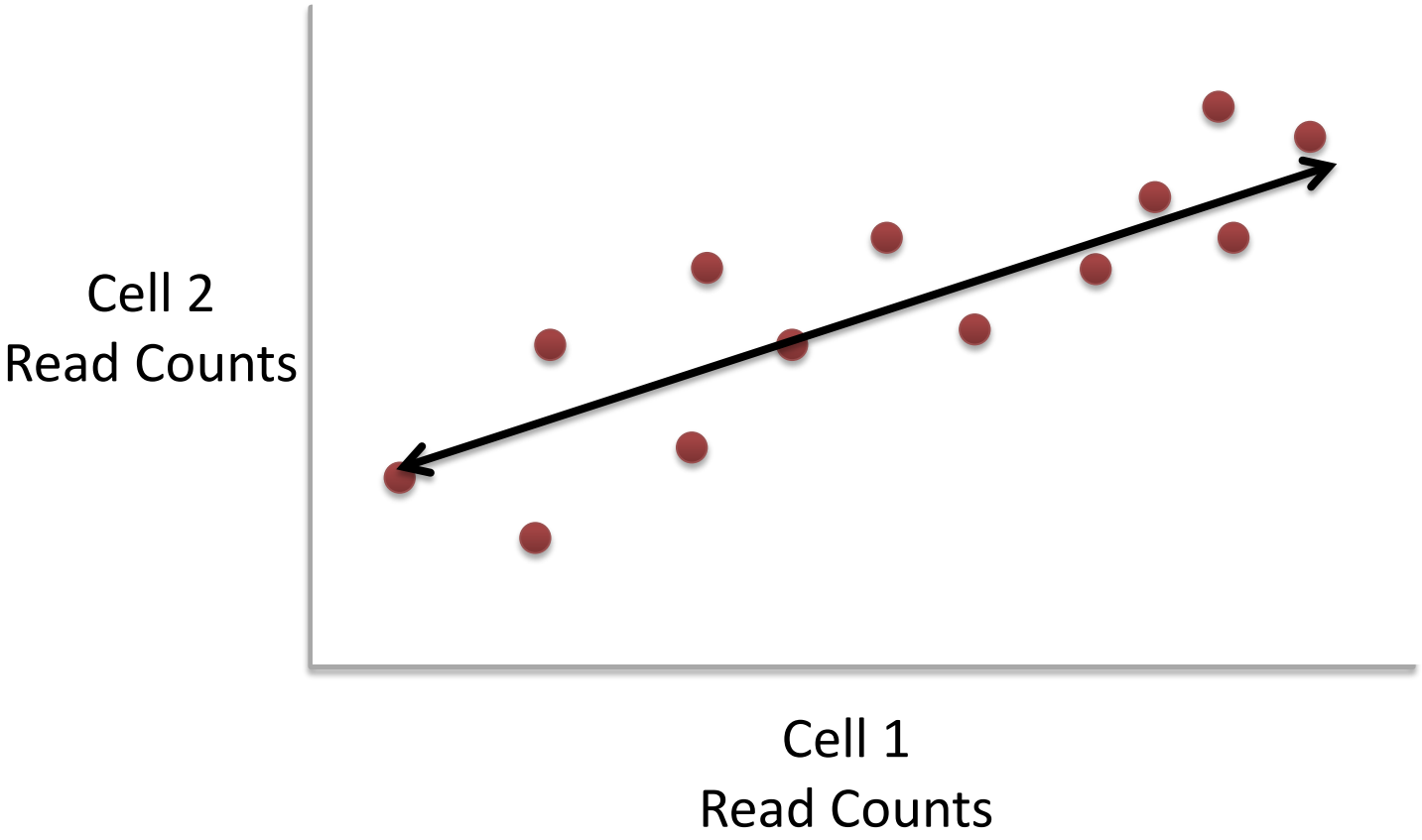
Just two cells …

Gene	Cell1 reads	Cell2 reads
a	10	8
b	0	2
c	14	10
d	33	45
e	50	42
f	80	72
g	95	90
h	44	50
i	60	50
… (etc)	… (etc)	… (etc)

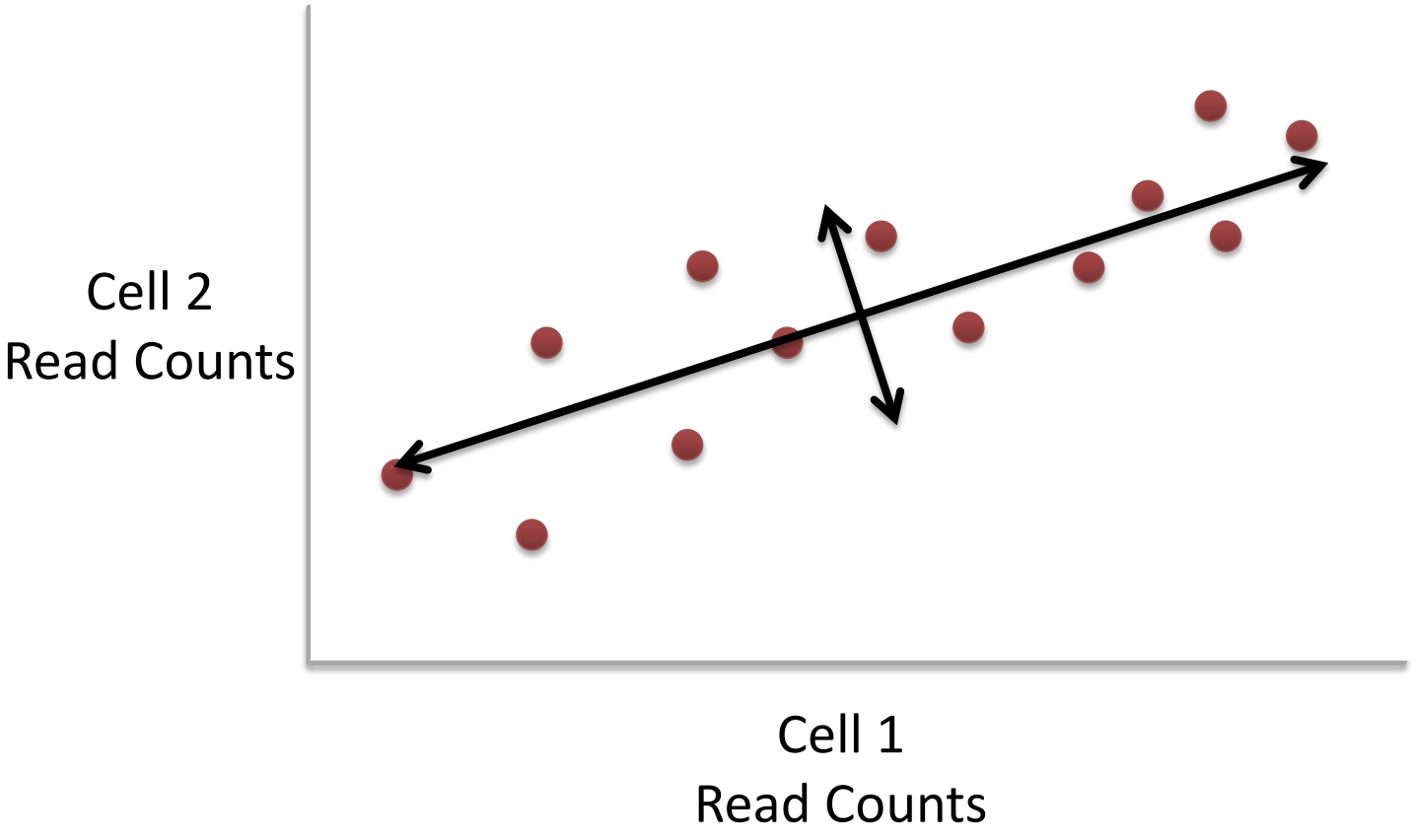
Here is a 2-D plot of the data from 2 cells.



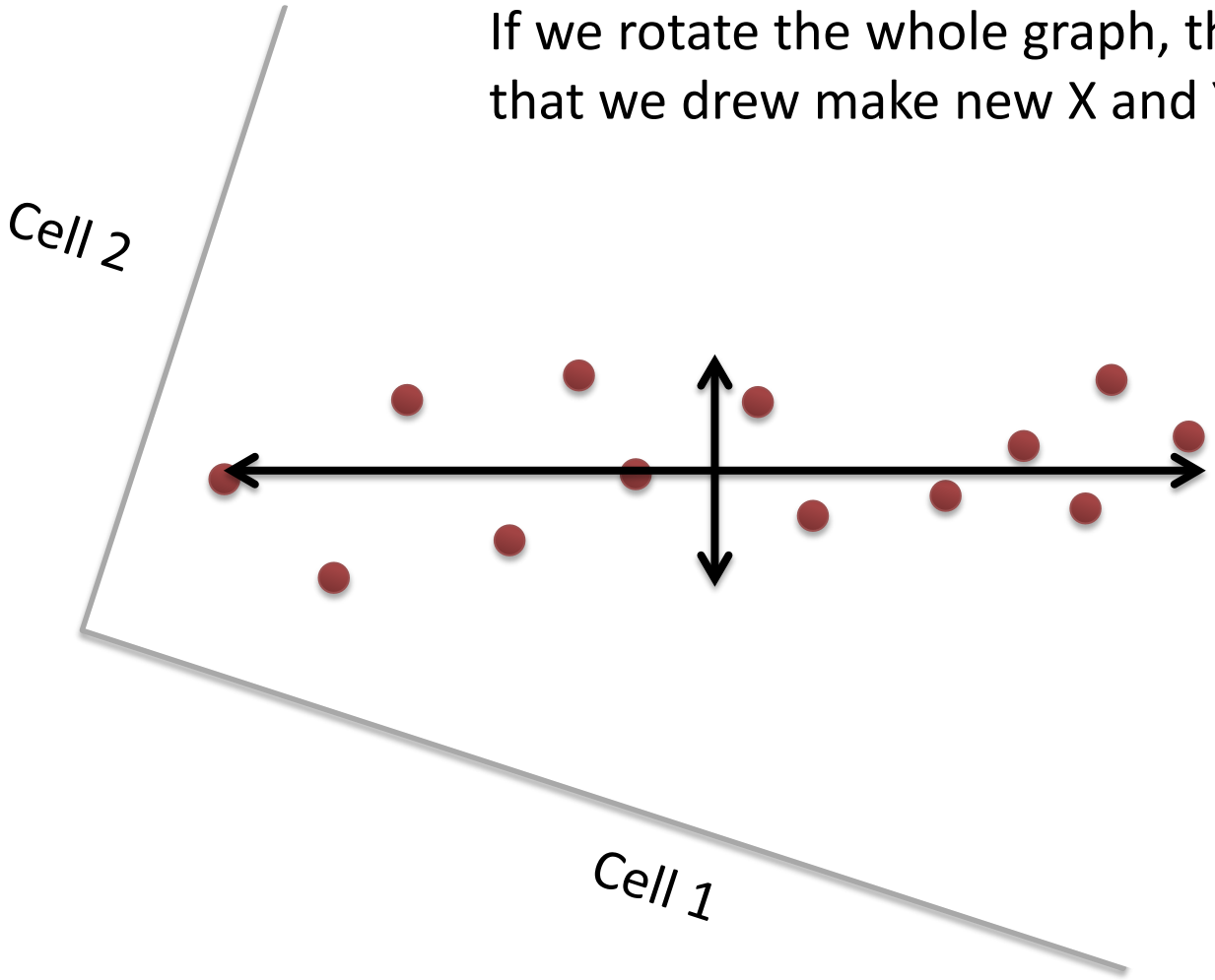
Generally speaking, the dots are spread out along a diagonal line. Another way to think about this is that the maximum variation in the data is between the two endpoints of this line.



Generally speaking, the dots are also spread out a little above and below the first line.
Another way to think about this is that the 2nd largest amount of variation is at the endpoints of the new line.

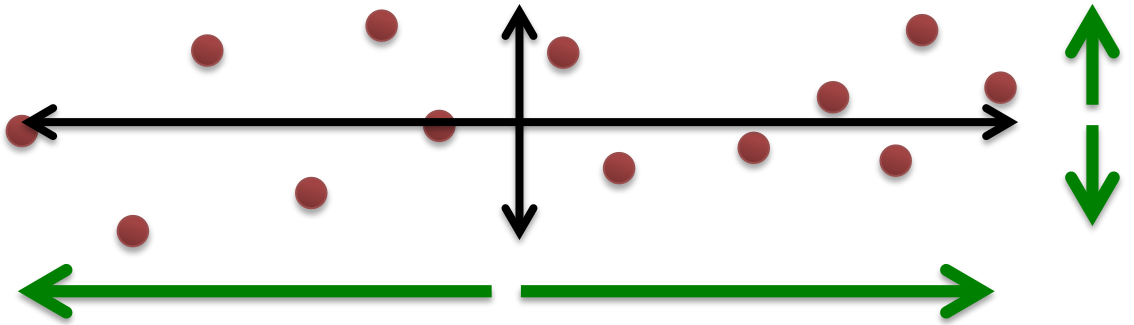


If we rotate the whole graph, the two lines that we drew make new X and Y axes.

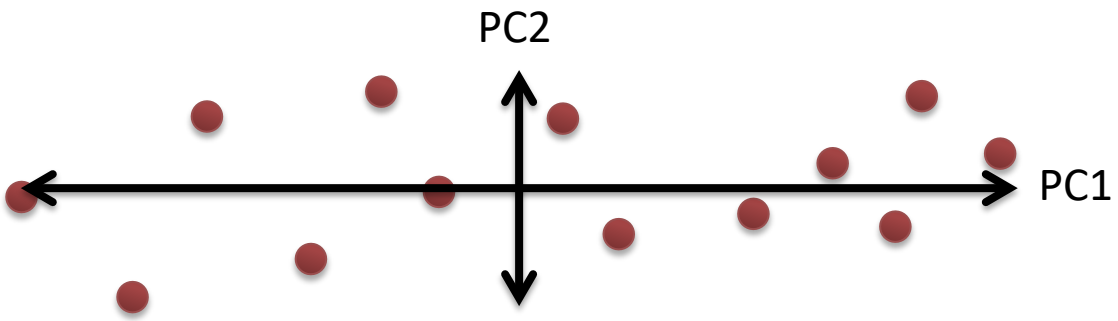


If we rotate the whole graph, the two lines that we drew make new X and Y axes.

This makes the left/right, above/below variation easier to see.

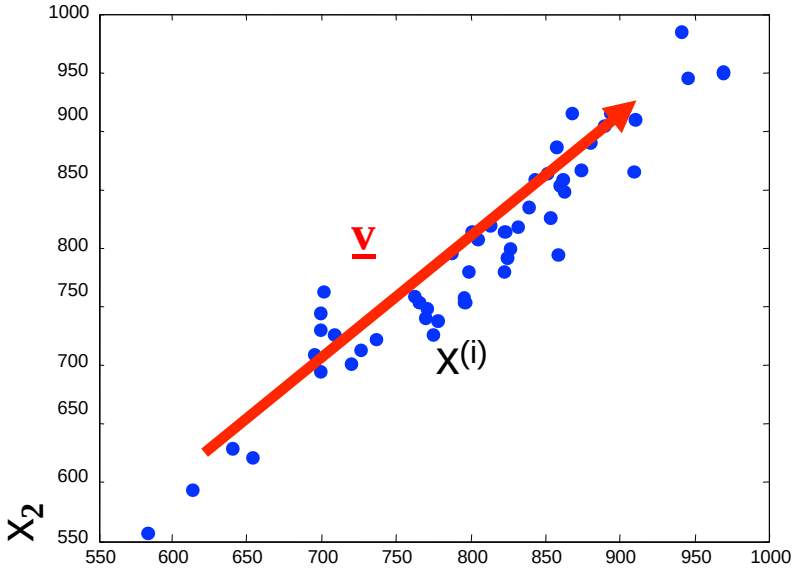


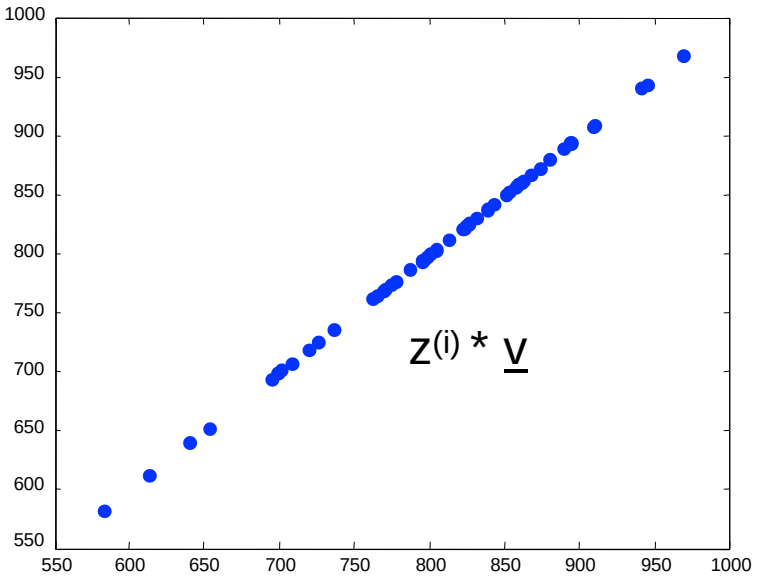
These two “new” (or “rotated”) axes that describe the variation in the data are “Principal Components” (PCs)



Dimensionality reduction

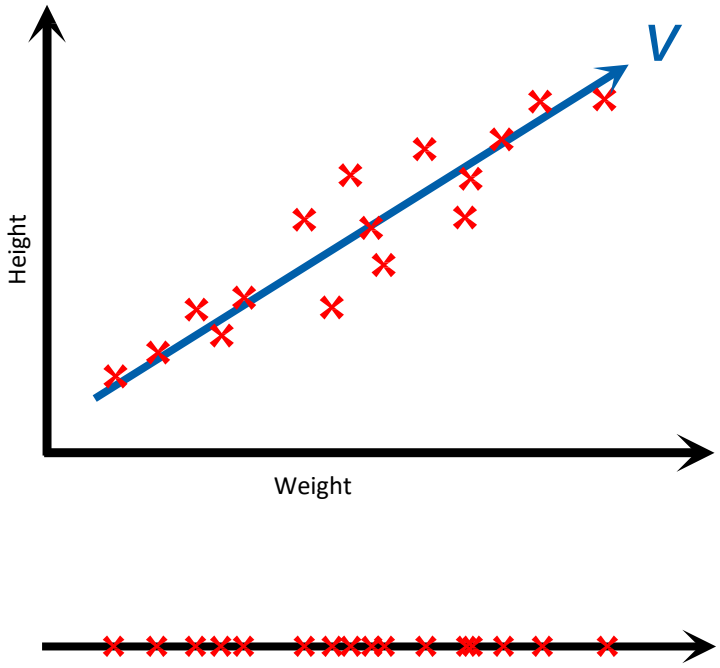
- Ex: data with two real values $[x_1,x_2]$
- We'd like to describe each point using only one value $[z_1]$
- We'll communicate a "model" to convert: $[x_1, x_2] \sim f(z_1)$
- Ex: linear function $f(z)$: $[x_1, x_2] = z * v * [v_1, v_2]$
- v is the same for all data points (communicate once)
- z tells us the closest point on v to the original point $[x_1,x_2]$





Dimensionality reduction

- most commonly used dimensionality reduction method
- projects the data on k orthogonal bases vectors v that minimize the projection error



Example:

- original 2D dataset containing features weight and height
- projection on vector v

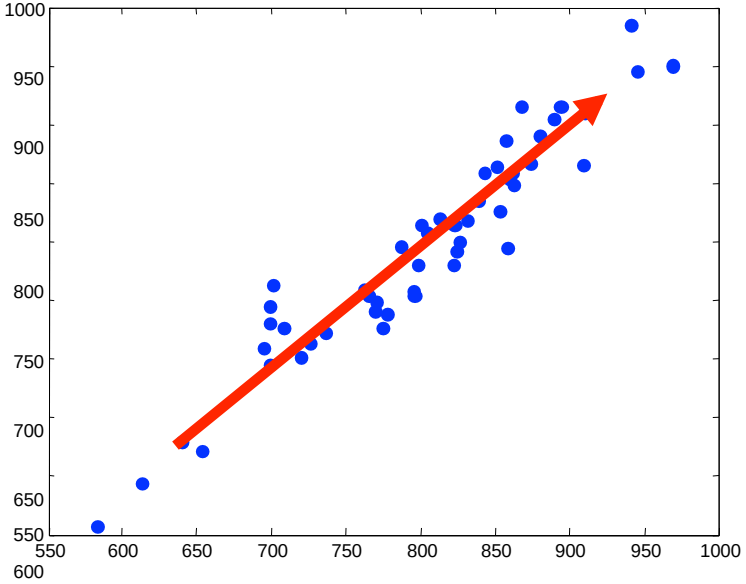
Principal Components Analysis

What is the vector that would most closely reconstruct X?

$$\min_{a,v} \sum_i \left(x^{(i)} - a^{(i)} \cdot v\right)^2$$

- Given v: $a^{(i)}$ is the projection of each point $x^{(i)}$ onto v
- v chosen to minimize the residual variance
- Equivalently, v is the direction of maximum variance
- Extensions: best two dimensions:

$$x_i = a_i v + b_i w + m$$



Geometry of Gaussian Distribution

$$\Delta^2 = (x-\mu)^T \Sigma^{-1} (x-\mu)$$

$$\Sigma = \mathrm{U}\Lambda U^T$$

Write Σ in terms of eigenvectors:

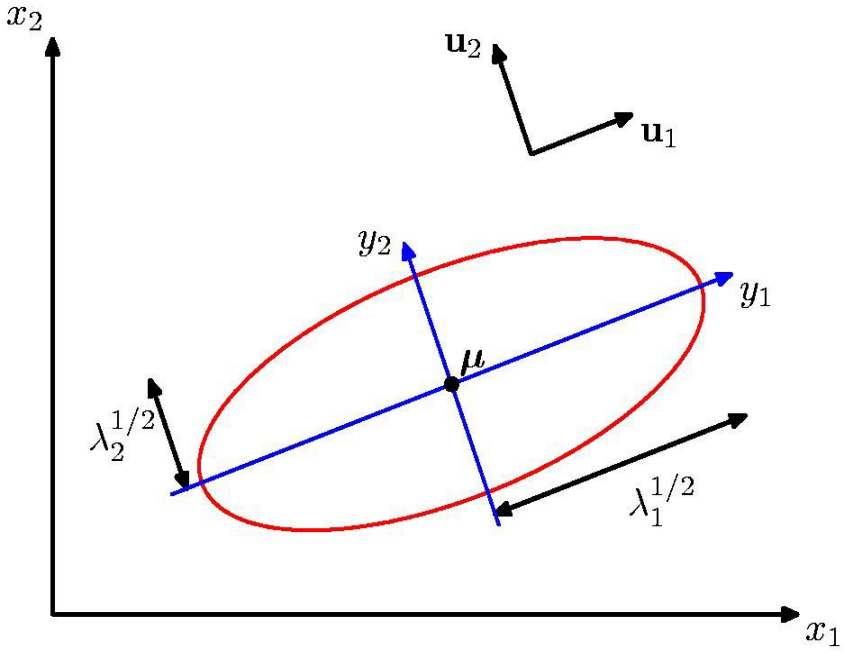
$$\Sigma^{-1} = \sum_{\mathrm{i}=1}^{D} \frac{1}{\lambda_i} u_i u_i^T$$

Then

$$\Delta^2 = \sum_{i=1}^{D} \frac{y_i^2}{\lambda_i}$$

$$y_i = u_i^T (x-\mu)$$

Oval shows constant Δ^2 value…



PCA Algorithm

Input: $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$ with Mean $\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}$

Subtract mean from each data point: $x_j^{(i)} \longleftarrow x_j^{(i)} - \mu_j$

(Typically) scale each dimension by its variance. Helps pay less attention to magnitude of the variable: $x_j^{(i)} \longleftarrow a_j x_j^{(i)}$

Compute covariance matrix $S = \frac{1}{n}\sum (x_i' - m)(x_i - m)$

Compute the k largest eigenvectors of S. $S = VDV^T$

PCA Algorithm

Compute the k largest eigenvectors of S. $S = VDV^T$

Dimensionality reduction from n to k dimensions:
Project the data onto the Eigenvectors corresponding to the k largest Eigenvalues.

$$v_1 = 1st \text{ principal component}$$
$$v_2 = 2nd \text{ principal component...}$$

Project data points $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$ to subspace defined by k principal components.

Scaling up PCA

Practical issue: covariance matrix is n x n.

- E.g. for image data Σ = 32,768 x 32,768
- Finding eigenvectors of such a matrix is slow.

Singular value decomposition (SVD) to the rescue!

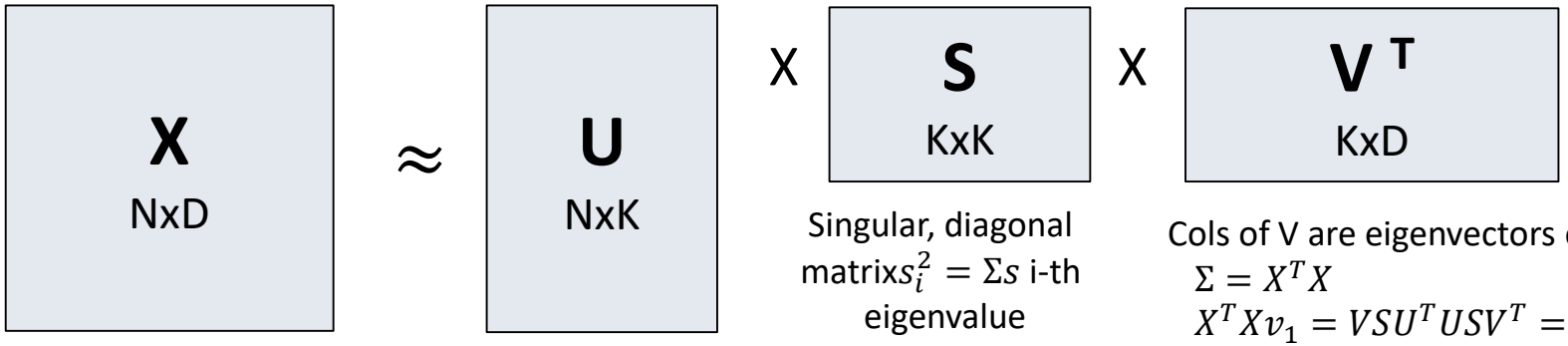
- Can be used to compute principal components.
- Efficient implementations available.
- Alternative method to calculate (still subtract mean 1st)

Singular Value Decomposition

Decompose $X = USV^T$

Orthogonal: $X^T X = VSSV^T = VDV^T$

$XX^T = USSU^T = UDU^T$



X: our data one row per data point

Each row of US gives coordinates of a data point in the projected space

Singular, diagonal matrix$s_i^2 = \Sigma s$ i-th eigenvalue

Cols of V are eigenvectors of $\Sigma = X^T X$

$X^T X v_1 = VSU^T USV^T = s_1^2 v_1$

If X is centered, then cols of V are the principal components.

$U * D$ matrix provides coefficients

Example $x_i = U_{i,1} S_{11} v_1 + U_{i,2} S_{22} v_2 + \ldots$

Gives the least-squares approximation to X of this form

SVD for PCA

- Subtract data mean from each point
- (Typically) scale each dimension by its variance
  Helps pay less attention to magnitude of the variable
- Compute the SVD of the data matrix

$$\mathrm{X} = USV^T$$

- Columns of **V** are the eigenvectors of $\boldsymbol{\Sigma}$ sorted from largest to smallest eigenvalues.
- Select the first *k* columns as our *k* principal components.

Outline

- Curse of Dimensionality
- Principal Components Analysis - PCA
- Examples
- Summary

Iris Data Set from Fisher, 1936

This is perhaps the best known database to be found in the pattern recognition literature. Fisher's paper is a classic in the field and is referenced frequently to this day. (See Duda & Hart, for example.) The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.

Predicted attribute: class of iris plant. This is an exceedingly simple domain.

This data differs from the data presented in Fishers article (identified by Steve Chadwick, spchadwick '@' espeedaz.net). The 35th sample should be: 4.9,3.1,1.5,0.2,"Iris-setosa" where the error is in the fourth feature. The 38th sample: 4.9,3.6,1.4,0.1,"Iris-setosa" where the errors are in the second and third features.

Attribute Information:

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm
5. class: Iris Setosa, Iris Versicolour, Iris Virginica

PCA on the Iris dataset

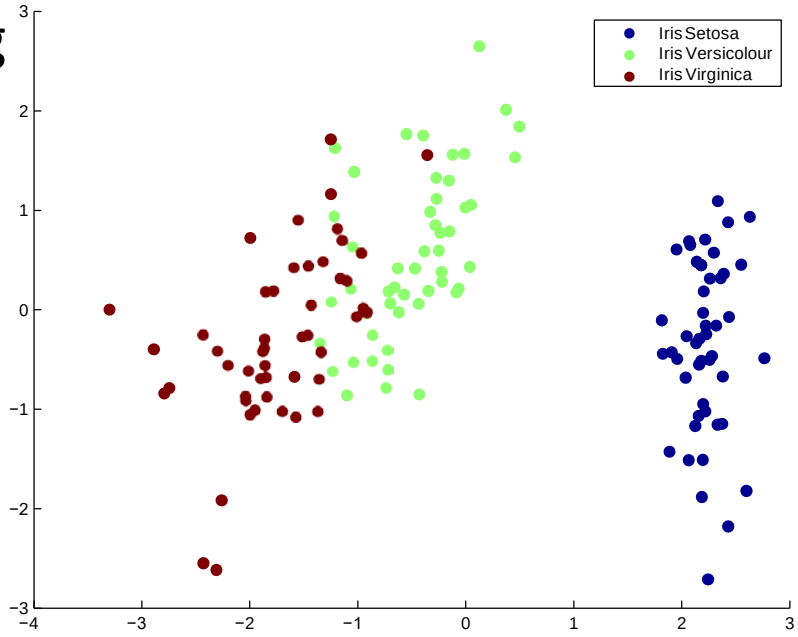
given: data matrix X

Preprocessing:

- mean normalization feature scaling
- compute covariance matrix
- $S = \frac{1}{n}\sum(x_i' - m)(x_i - m)$

Compute eigenvectors and eigenvalues:

$$V = \begin{pmatrix} -0.5224 & -0.3723 & 0.7210 & 0.2620 \\ 0.2634 & -0.9256 & -0.2420 & -0.1241 \\ -0.5813 & -0.0211 & -0.1409 & -0.8012 \\ -0.5656 & -0.0654 & -0.6338 & 0.5235 \end{pmatrix}$$



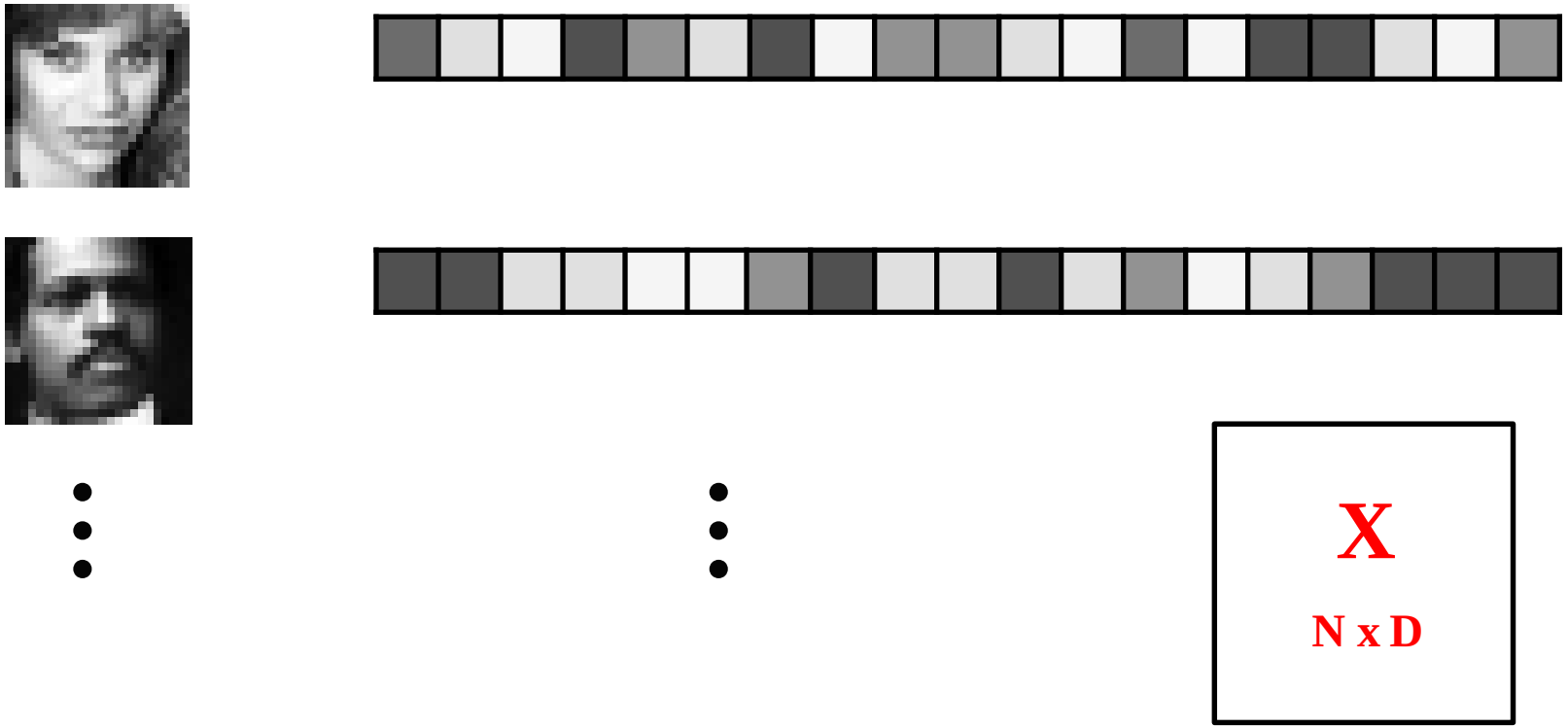
$$S = \begin{pmatrix} 2.8914 & 0 & 0 & 0 \\ 0 & 0.9151 & 0 & 0 \\ 0 & 0 & 0.1464 & 0 \\ 0 & 0 & 0 & 0.0205 \end{pmatrix}$$

reduce V to *k* components

$$z^{(i)} = V_{reduce}^{T} x^{(i)}$$

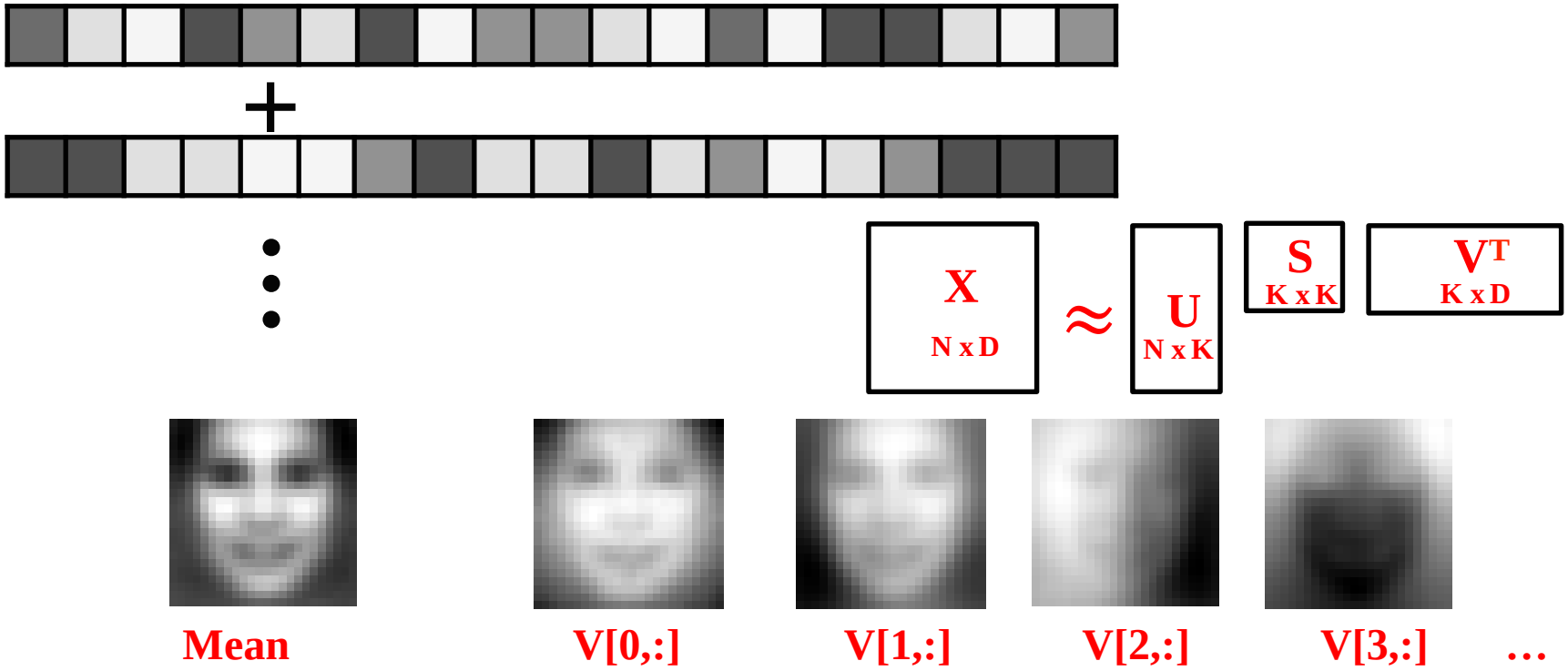
Eigen-faces (Example by Ihler)

- "Eigen-X" = represent X using PCA
- Ex: Viola Jones data set 24x24 images of faces = 576 dimensional measurements



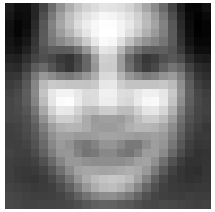
Eigen-faces

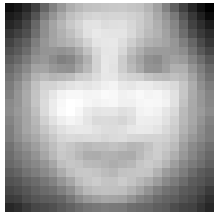
- "Eigen-X" = represent X using PCA
- Ex: Viola Jones data set 24x24 images of faces = 576 dimensional measurements
- Take first K PCA components

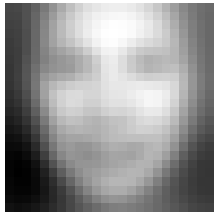


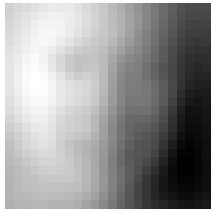
Eigen-faces

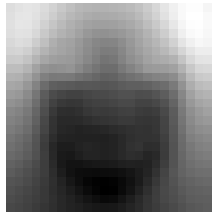
- "Eigen-X" = represent X using PCA
- Ex: Viola Jones data set 24x24 images of faces = 576 dimensional measurements
- Take first K PCA components

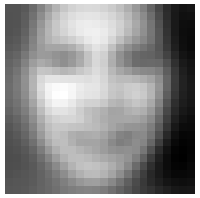

Mean


Dir 1

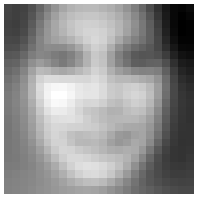

Dir 2

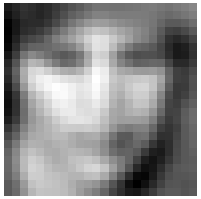

Dir 3


Dir 4 …


K=4


K=50


K=4


K=50

Outline

- Curse of Dimensionality
- Principal Components Analysis - PCA
- Examples
- Summary

Summary

- PCA is dimensionality reduction
- Representation: basis vectors & coefficients
- Linear decomposition
  - PCA / eigendecomposition
  - Singular value decomposition
- PCA can only realize linear transformations, there exist nonlinear extensions (Kernel PCA)
- PCA-transformed data is uncorrelated
- PCA assumes that most of the information is contained in the direction with the highest variance
- PCA is often used to reduce the noise in a signal
- PCA is an unsupervised method - when used as a preprocessing step for supervised learning theperformance can drop significantly