高维数据的降维

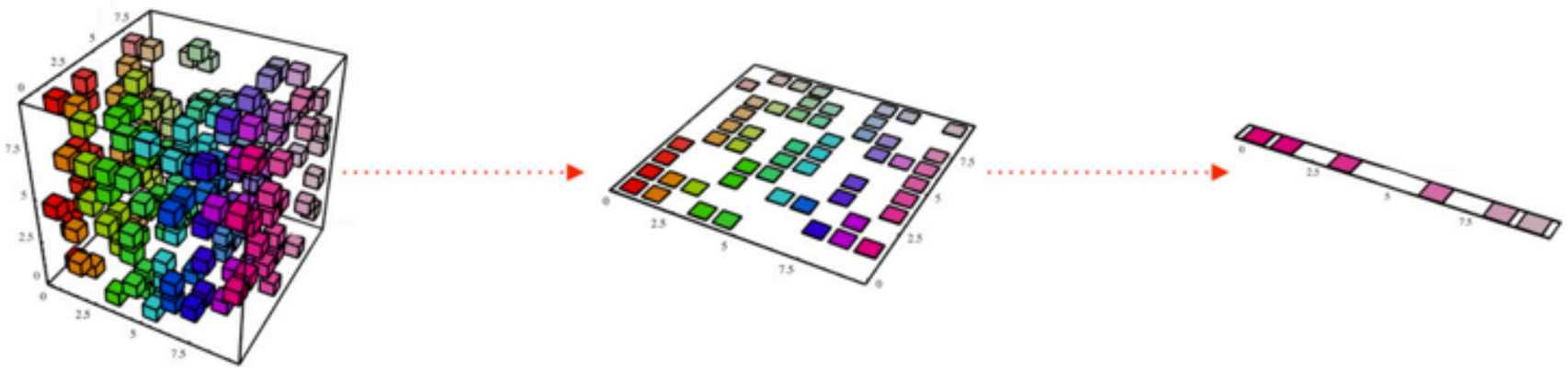
- 什么是数据降维?
- 为什么要对数据降维?
- 线性与非线性方法

什么是数据降维?

Dimensionality reduction or dimension reduction is the process of reducing the number of random variables under consideration, via obtaining a set of "uncorrelated" principal variable.

- Wikipedia: Dimensionality reduction

- 采用某种映射方法，将原高维空间中的数据点映射到低维度的空间中
- 降维的本质是学习一个映射函数 $f: X \to Y$，其中X是原始数据点，用n维向量表示。 Y是数据点映射后的r维向量，其中$n > r$



3 dimensions: 1000 positions!

2 dimensions: 100 positions

1 dimension: 10 positions

如何对数据降维?

- “数据降维”可通过特征选择和特征提取两种方式实现

特征选择:

从原始特征中挑选出一些最有代表性、分类性能最好的指标

特征提取（**本讲重点**）:

利用映射或变换的方法把原始特征变换为较少的特征

- “数据和特征决定了机器学习的上限，而模型和算法只是逼近这个上限而已”。

- 特征工程包括特征抽取和特征选择两部分。特征选择和提取其中一个方案是依据相关领域知识提取特征，依靠业务知识，另外一个就是数据驱动的方法。

为什么要进行降维?

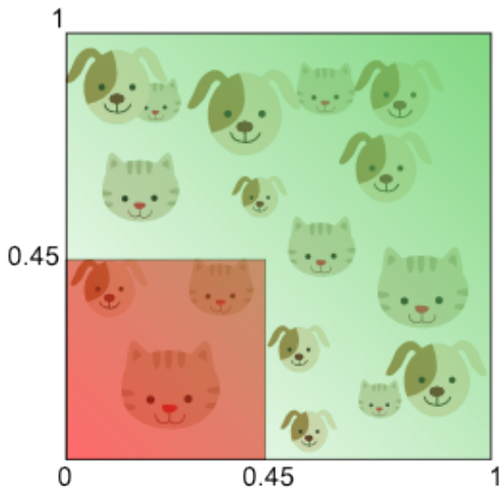
若想要覆盖20%的特征空间，需要多少数据？

高维度带来的问题：

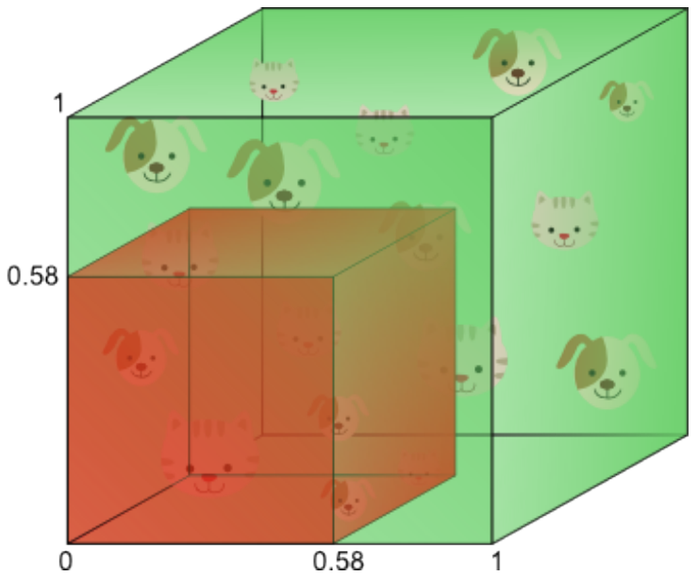
1. 维数灾难

   （Curse of Dimensionality）

一个特征
20%

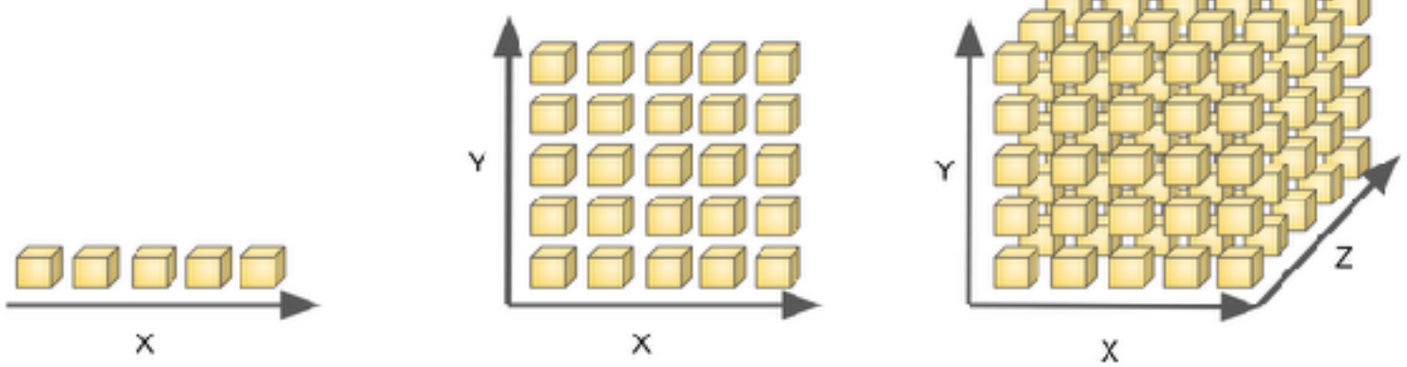


二个特征
45%



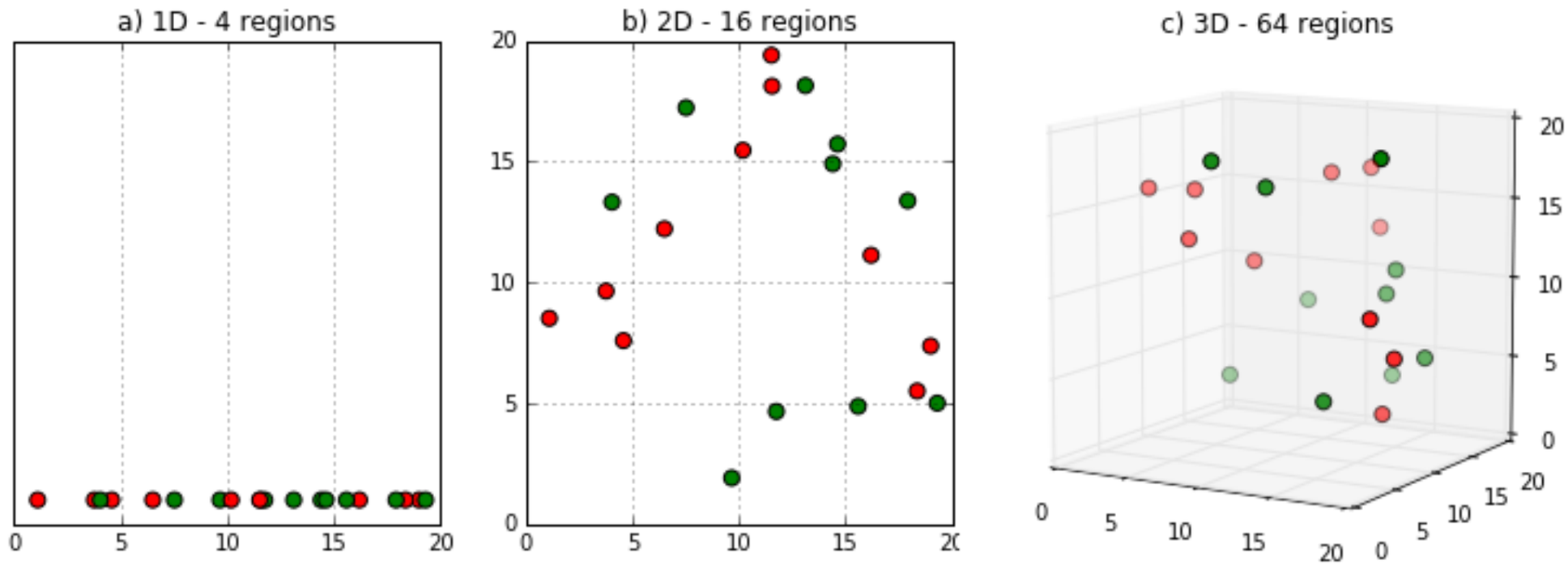
三个特征
58%

收集并计算大量的样本是困难的！



为什么要进行降维?

训练数据在高维空间变得稀疏，用于训练模型的样本量不足，易产生过拟合问题。



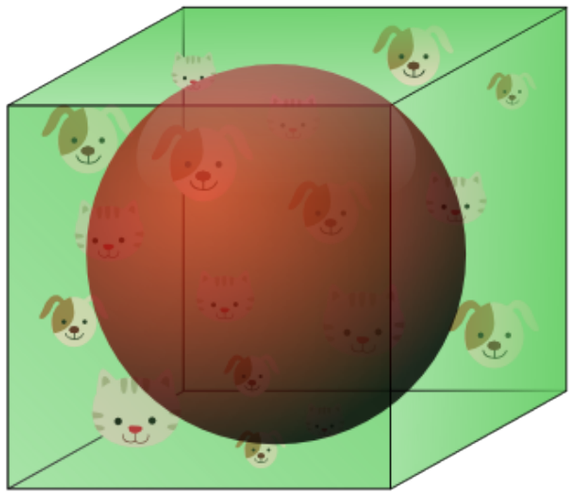
密度：$20/4 = 5$ $20/16 = 1.25$ $20/64 \approx 0.31$

为什么要进行降维?

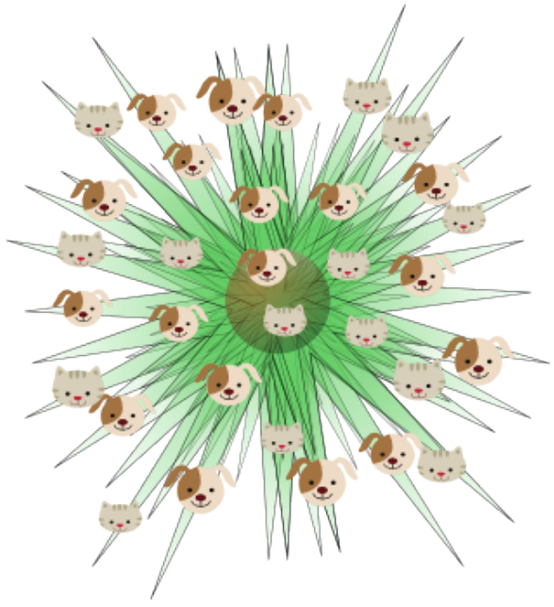
2. 模型表现变差

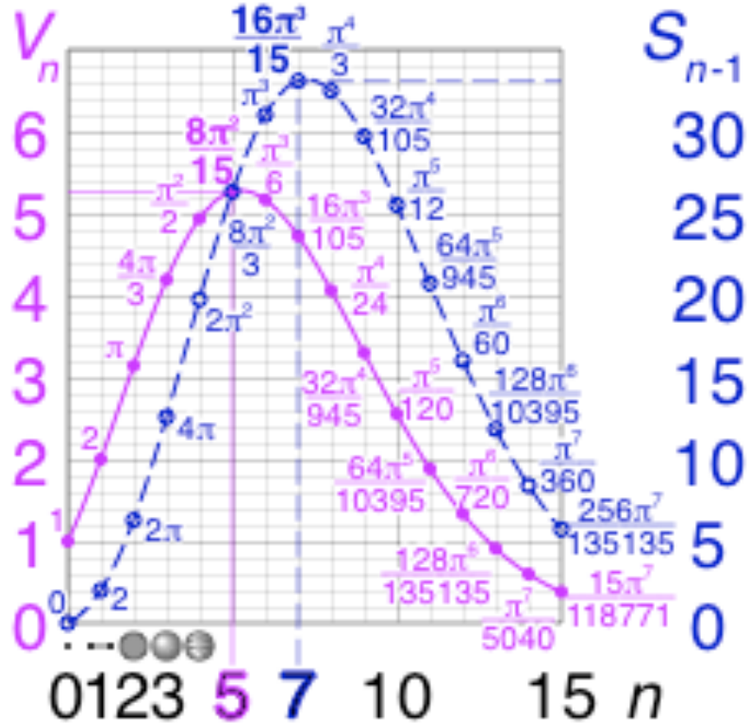
2维情况下
角落数量：$2^2 = 4$

3维情况下
角落数量：$2^3 = 8$



4维情况下
角落数量：$2^8 = 256$





随着维数增加，超球体(内接于超立方)的体积越来越小，落在角落的数据越来越多。因为不同角落的值差别很大，落在角落的数据很难进行分类

The volume of sphere $\dfrac{2r^d \pi^{d/2}}{d\,\Gamma(d/2)}$

The volume of cube $(2r)^d$

$$\frac{V_{\text{hypersphere}}}{V_{\text{hypercube}}} = \frac{\pi^{d/2}}{d2^{d-1}\Gamma(d/2)} \to 0$$

https://en.wikipedia.org/wiki/Hypercube

为什么要进行降维?

- 降维的目的：
  - 压缩数据
  - 去除噪声
  - 提取特征
- 降维的作用：
  - 降低空间复杂度：更少的参数
  - 降低时间复杂度：更少的计算量
  - 便于数据可视化

PCA on MNIST dataset

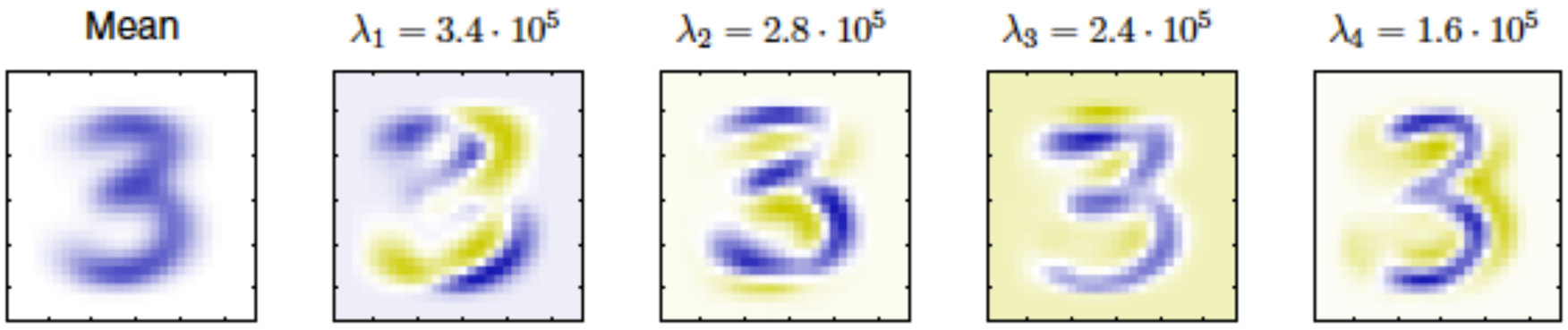


Figure 12.3 The mean vector $\overline{\mathbf{x}}$ along with the first four PCA eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_4$ for the off-line digits data set, together with the corresponding eigenvalues.

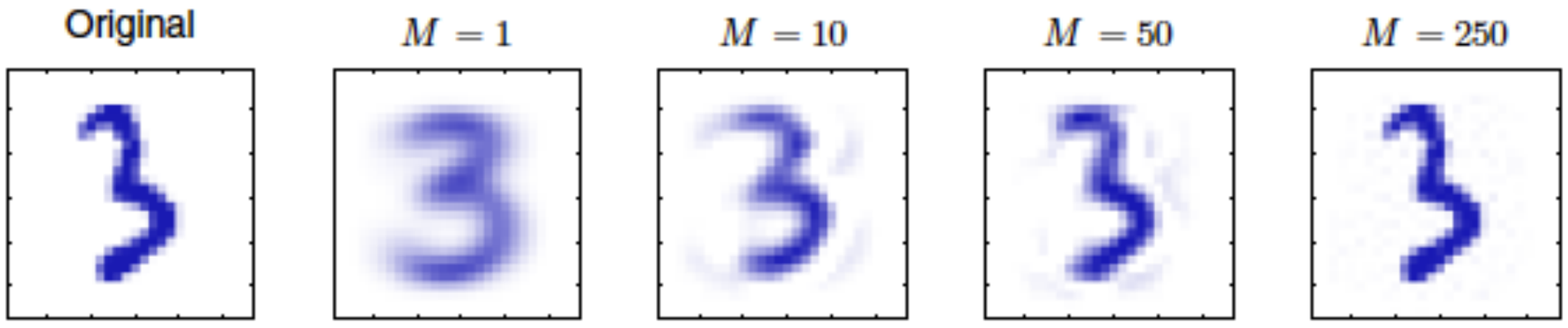


Figure 12.5 An original example from the off-line digits data set together with its PCA reconstructions obtained by retaining M principal components for various values of M. As M increases the reconstruction becomes more accurate and would become perfect when $M = D = 28 \times 28 = 784$.

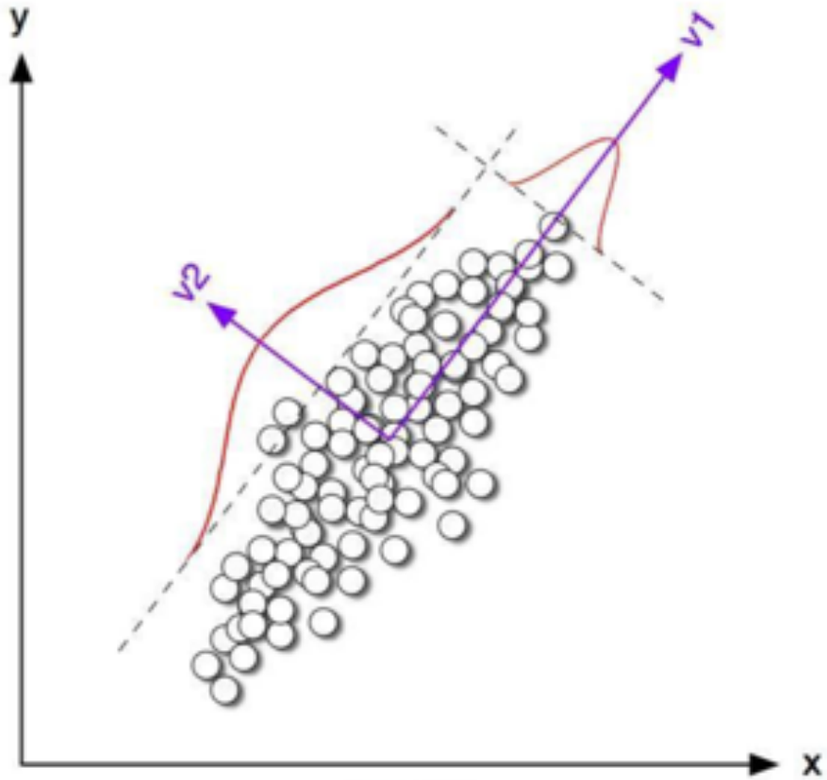
Bishop. Pattern Recognition and Machine Learning

降维的方法

- 线性降维方法：
  - Principle Component Analysis (PCA)
  - Linear Discriminant Analysis (LDA)
  - 其他： Independent Component Analysis (ICA)
  - Canonical Correlation Analysis (CCA)
- 非线性降维方法:
  - 基于核函数的降维方法
  - 基于流形学习的降维方法

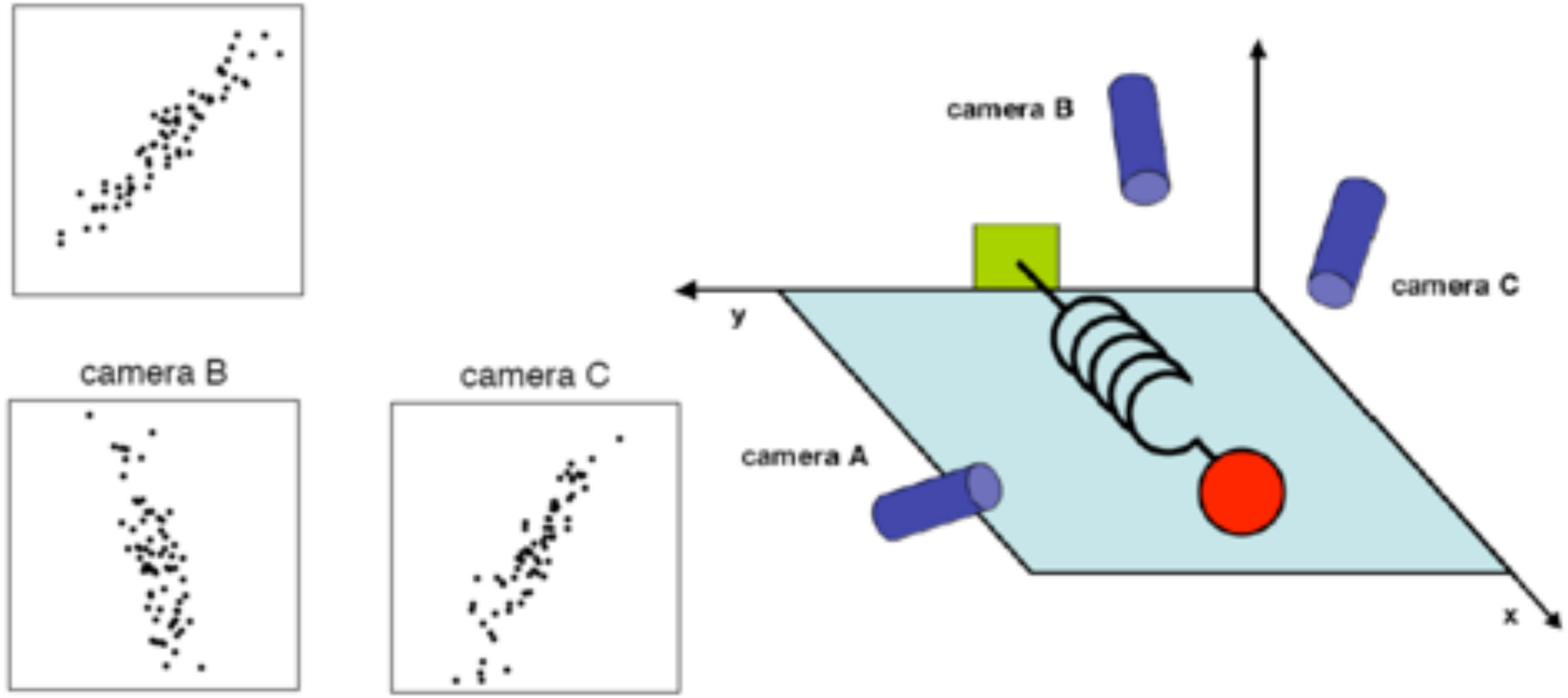
主成分分析 (PCA)

- 主成分分析（Principal Component Analysis，PCA)通过正交变换将一组可能存在相关性的变量转换为一组线性不相关的变量，转换后的这组变量叫主成分。
- 基本思想：构造原变量的一系列线性组合形成几个综合指标，以去除数据的相关性，并使低维数据最大程度保持原始高维数据的方差信息。



为什么需要PCA?

- 我们在处理数据的时候经常会遇到每个样本包含很多个变量。变量太多，会增加分析问题的难度与复杂性。
- 在许多实际问题中，多个变量之间具有一定的相关关系(多重共线性)。因此，研究的重点是，能否在各个变量之间相关关系研究的基础上，用较少的新变量代替原来较多的变量，而且使这些较少的新变量尽可能多地保留原来较多的变量所反映的信息。



为什么需要PCA?

- 美国统计学家斯通在1947年关于国民经济的研究中，得到了17个反映国民收入与支出的变量要素，例如雇主补贴、消费资料、生产资料、纯公共支出、股息、利息、外贸平衡等。
- 在进行主成分分析后，以97.4%的精度，用3个新变量就取代了原来的17个变量。根据经济学知识，斯通给这3个新变量分别命名为总收入F1，总收入变化率F2和经济发展或衰退趋势F3。

Sir John Richard Nicholas Stone (30 August 1913 – 6 December 1991) was an eminent British economist, educated at Westminster School, Cambridge University (Caius and King's), who in 1984 received the Nobel Memorial Prize in Economic Sciences for developing an accounting model that could be used to track economic activities on a national and, later, an international scale.

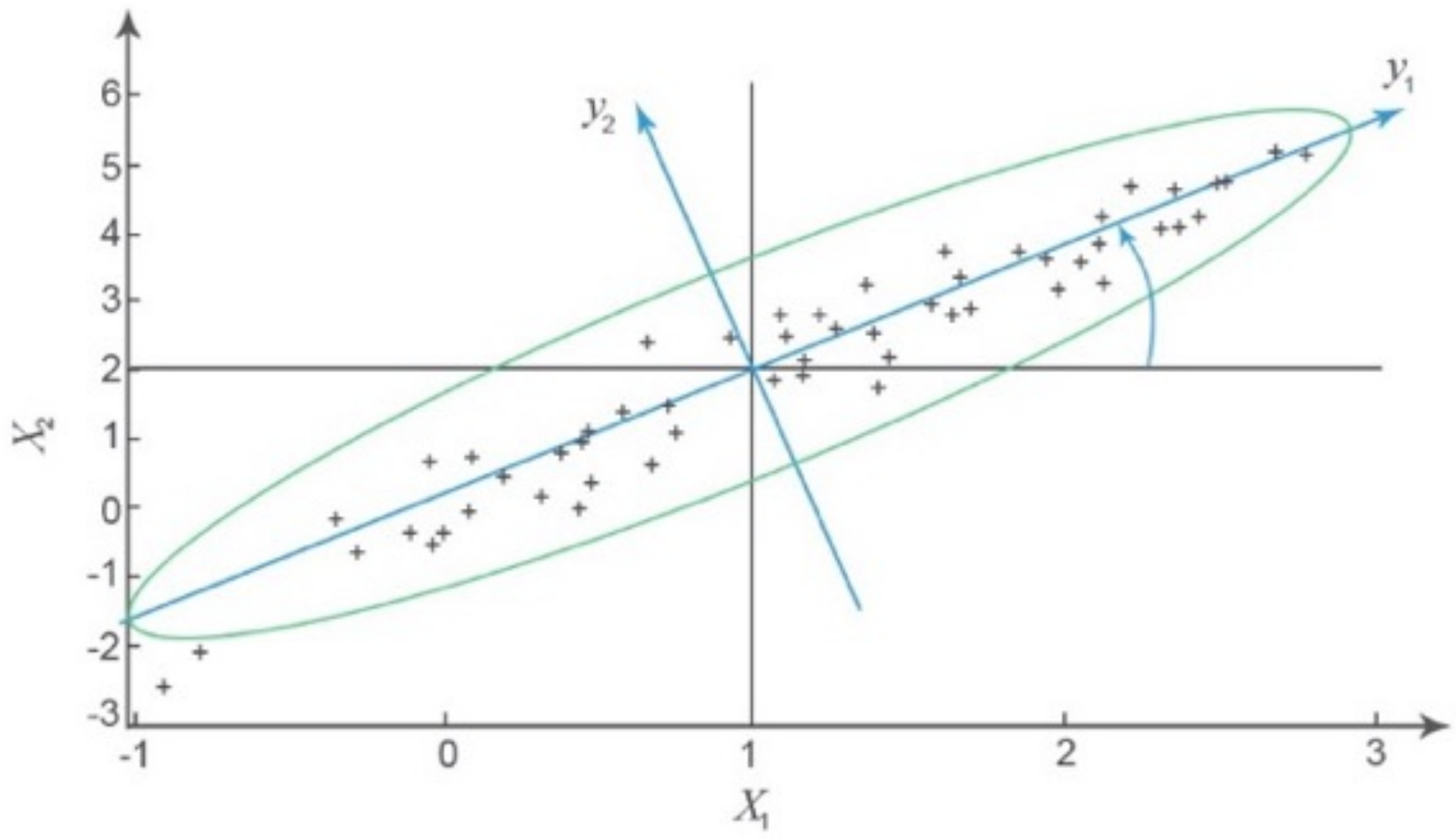
约翰·理查德·尼古拉斯·斯通，著名经济学家，国民经济统计之父，在国民帐户体系的发展中做出了奠基性贡献，极大地改进了经济实证分析的基础，是1984年的诺贝尔经济学奖获得者。

PCA的几何解释

首先我们从集合的角度直观理解什么是主成分?

- 图中的点为一些2维样本向量分布而成，原始坐标轴为x_1, x_2。这些数据形成一个椭圆形状的点阵（在样本的二维正态分布的假定下），椭圆有一个长轴y_1和一个短轴y_2

如果我们想把这些样本向量降为1维表示，理想情况是这个1维新向量包含原始数据最多的信息。于是我们选择y_1方向，因为y_1方向离散程度最大，方差最大，包含的信息量最多；而短轴方向上的数据变化很少，对数据的解释能力弱



PCA的几何解释（续）

- 对于多变量的情况和2维类似，可以推广为高维椭球。
- 首先把高维椭球的主轴找出来，再用代表大多数数据信息的最长的几个轴作为新变量，这样主成分分析就完成了。
- 和2维情况类似，高维椭球的主轴也是互相垂直的。这些互相正交的新变量是原先变量的线性组合，称为主成分。

如何选择主成分?

- 2维椭圆有两个主轴，3维椭球有三个主轴，有几个变量就有几个候选的主成分。主成分选择的标准就是被选的主成分所代表的主轴长度之和占了主轴长度总和的大部分（通常为85%）。具体选择几个主成分还要根据实际情况而定。

PCA的基本假设

- 大方差代表重要结构
  - 降维后尽量保留更多的原始数据的方差
- 线性投影
  - 降维后的数据是原始数据经过线性投影得到
- 主成分之间正交，

$$w_j^T w_i = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

假设：原始数据X每一维的平均值是0

$$Y = W^T X$$

$X \in R^{n,m}$ 原始数据, m个n维样本

$W^T \in R^{r,n}$ 投影矩阵，r组投影基, r < n

$Y \in R^{r,m}$ 投影后数据, m个r维样本

$$W^T = \begin{pmatrix} w_1^T \\ w_2^T \\ \cdots \\ w_r^T \end{pmatrix}$$

PCA的目标

- 将一组n维向量降为r维（0<r<n），其目标是选择r个单位正交基，使得原始数据变换到这组基上后，各变量两两间协方差为0（让两个变量尽可能表示更多的原始信息，不希望它们之间存在相关性），而同一变量的方差则尽可能大（让某一变量尽可能携带更多的信息）。
- 等价于将降维后的协方差矩阵对角化，即除对角线外的其它元素化为0，并且在对角线上将元素按大小从上到下排列，这样我们就达到了PCA的目标。

PCA的目标

- 假设数据变量的协方差是

$$C = \frac{1}{m-1} X X^T$$

- 投影后的协方差矩阵S_y,目标是使S_y对角化。

$$Y = W^T X$$

$$\begin{aligned} S_y &= \frac{1}{m-1} Y Y^T \\ &= \frac{1}{m-1} W^T X (W^T X)^T \\ &= \frac{1}{m-1} W^T X X^T W \\ &= W^T \left(\frac{1}{m-1} X X^T \right) W \\ &= W^T C W \end{aligned}$$

PCA的求解

- 由之前的特征值分解知识，我们有

$$U^{-1}CU = U^{\mathrm{T}}CU = \Lambda = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix}$$

- 对比前式，有$\mathrm{W} = \mathrm{U}$，即W是协方差矩阵的特征向量单位化后按列排列出的矩阵U，其中每一列都是C的一个特征向量。如果设W按照Λ中特征值从大到小，将特征向量从上到下排列，则用W^T的前r行组成的矩阵乘以原始数据矩阵X，就得到了我们需要的降维后的数据矩阵Y

主成分数量r的选取

- 记第k个主成分的方差贡献率为 $e_k = \frac{\lambda_k}{\sum_{k=1}^{n} \lambda_k}$
- 若只取 $r(r < n)$ 个主成分，那么前r个主成分的累计方差贡献率为：

$$E_r = \frac{\sum_{k=1}^{r} \lambda_k}{\sum_{k=1}^{n} \lambda_k}$$

- 累计方差贡献率表明 $Y_1, Y_2, \ldots, Y_r$ 综合描述原数据 $X_1, X_2, \ldots, X_n$ 的能力，通常情况下，我们选取r个主成分使得E_r达到一个较高的百分数（如85%以上）

$$\frac{\sum_{k=1}^{r} \lambda_k}{\sum_{k=1}^{n} \lambda_k} > \text{阈值}(0.85\text{或者}0.9)$$

PCA算法

设有m个n维数据样本X

- 1 将原始数据按列组成n行m列矩阵X
- 2 将X的每一行进行零均值化，即减去这一行的均值
- 3 求出协方差矩阵

$$C = \frac{1}{m-1} X X^T$$

- 4 求出协方差矩阵的特征值及对应的特征向量
- 5 将特征向量按对应特征值大小从上到下按行排列成矩阵，取前r行组成矩阵W^T
- 6 Y=W^T X即为降维到r维后的数据

主成分分析示例

- 为了分析汽车市场的销售状况，我们以三款车为例，判断哪款车的销售量是影响汽车市场销售状况主要成分
- 假设市场上三款汽车价格（Jeep:x_1,Toyota:x_2;Benz:x_3）的月份数据的协方差矩阵为：

$$C = \begin{pmatrix} 1 & 2/\sqrt{10} & -2/\sqrt{10} \\ 2/\sqrt{10} & 1 & -4/5 \\ -2/\sqrt{10} & -4/5 & 1 \end{pmatrix}$$

- 试求三款汽车月份价格的所有主成分

主成分分析示例

- 由协方差矩阵可得特征多项式为：

$$|\lambda I - C| = \begin{pmatrix} \lambda - 1 & -2/\sqrt{10} & 2/\sqrt{10} \\ -2/\sqrt{10} & \lambda - 1 & 4/5 \\ 2/\sqrt{10} & 4/5 & \lambda - 1 \end{pmatrix}$$

- 令特征多项式为零可解得特征根为：$\lambda_1 = 2.38, \lambda_2 = 0.42, \lambda_3 = 0.2$

- 将特征根代入特征方程可得到相应的特征向量：

$$U_1 = (0.54, 0.59, -0.59)$$
$$U_2 = (0.84, -0.39, 0.39)$$
$$U_3 = (0, 0.71, 0.71)$$

主成分分析示例

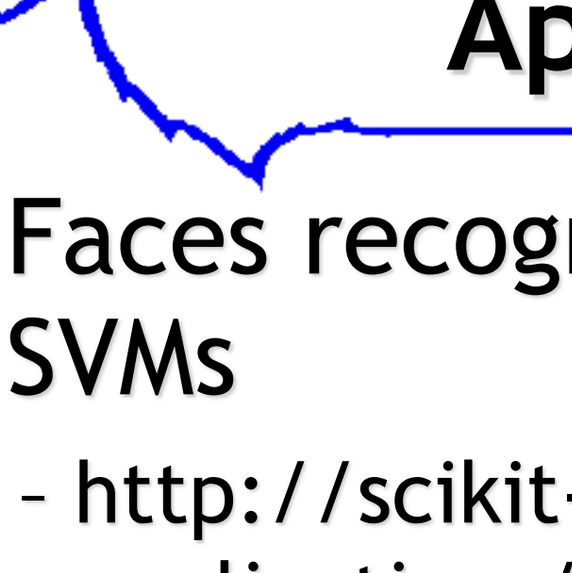
- 因此，三款汽车价格的三个主成分分别为：

$$Y_1 = U_1^T X = 0.54X_1 + 0.59X_2 - 0.59X_3$$
$$Y_2 = U_2^T X = 0.84X_1 - 0.39X_2 + 0.39X_3$$
$$Y_3 = U_3^T X = 0.71X_2 + 0.71X_3$$

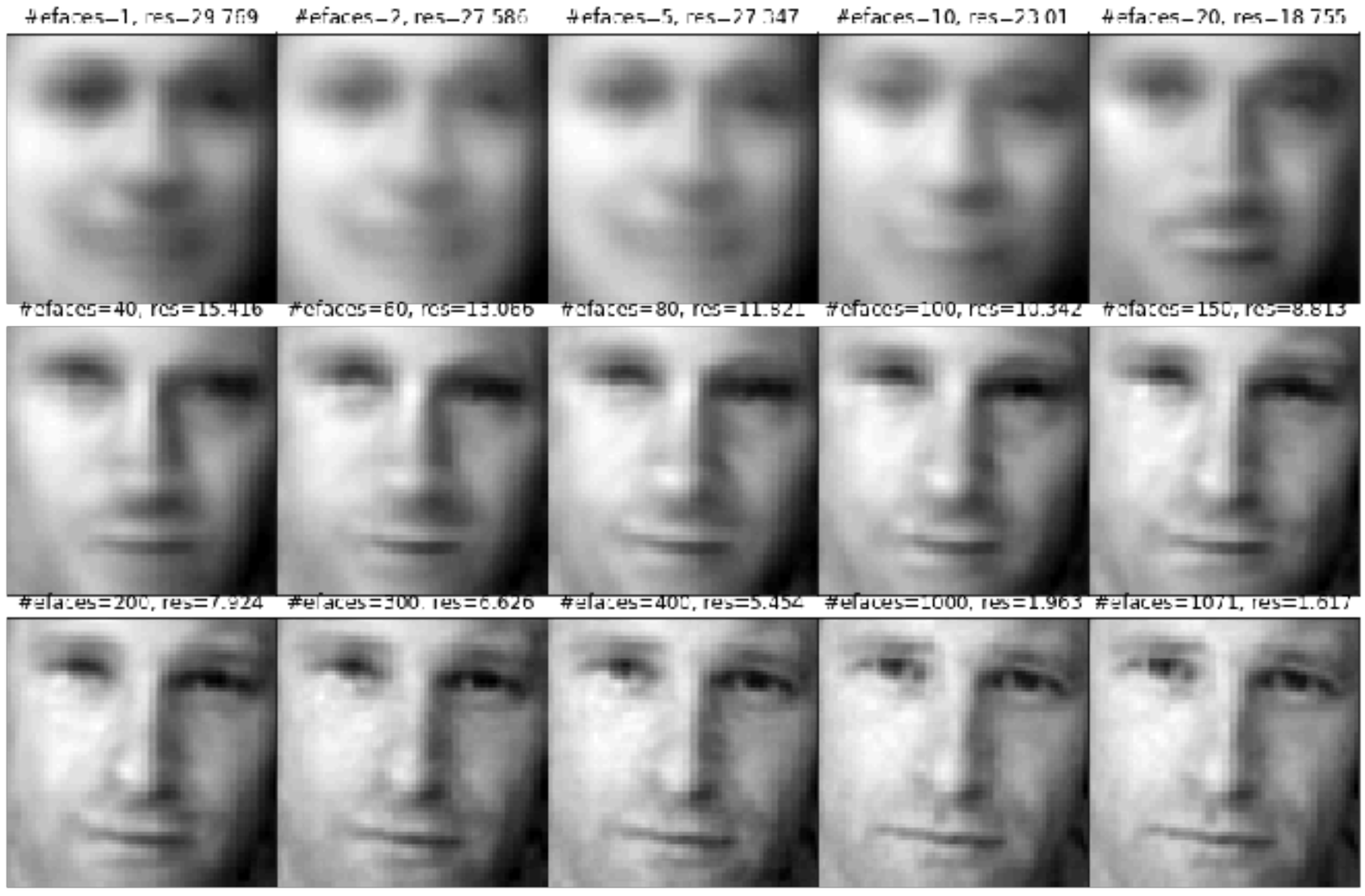
- 其中，第一个主成分的方差占了原始变量总方差的大部分，所以第一主成分综合反映了三款汽车价格的绝大部分变动

Application :PCA + SVM

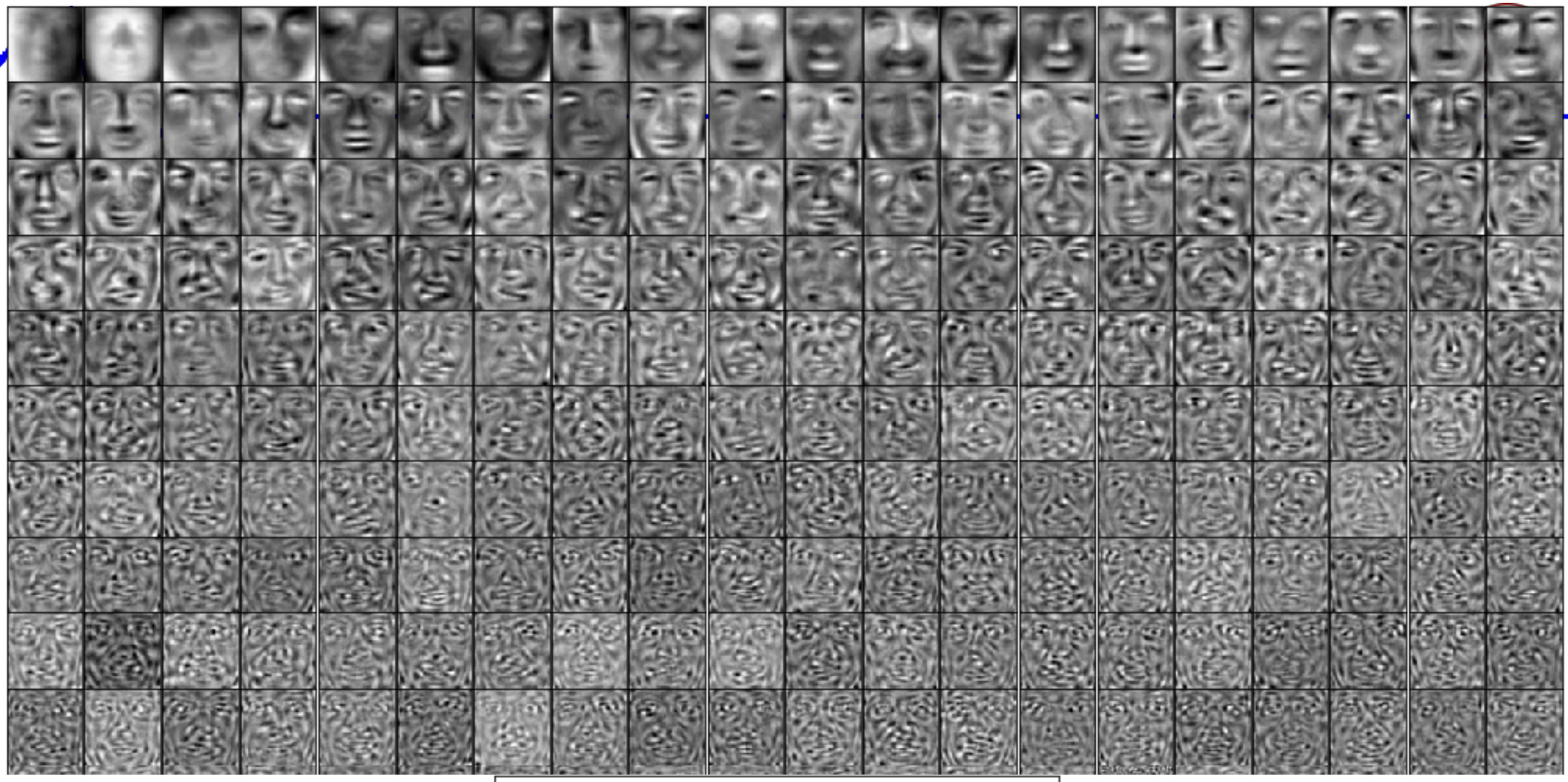
- Faces recognition example using eigenfaces and SVMs
  - http://scikit-learn.org/stable/auto_examples/applications/plot_face_recognition.html

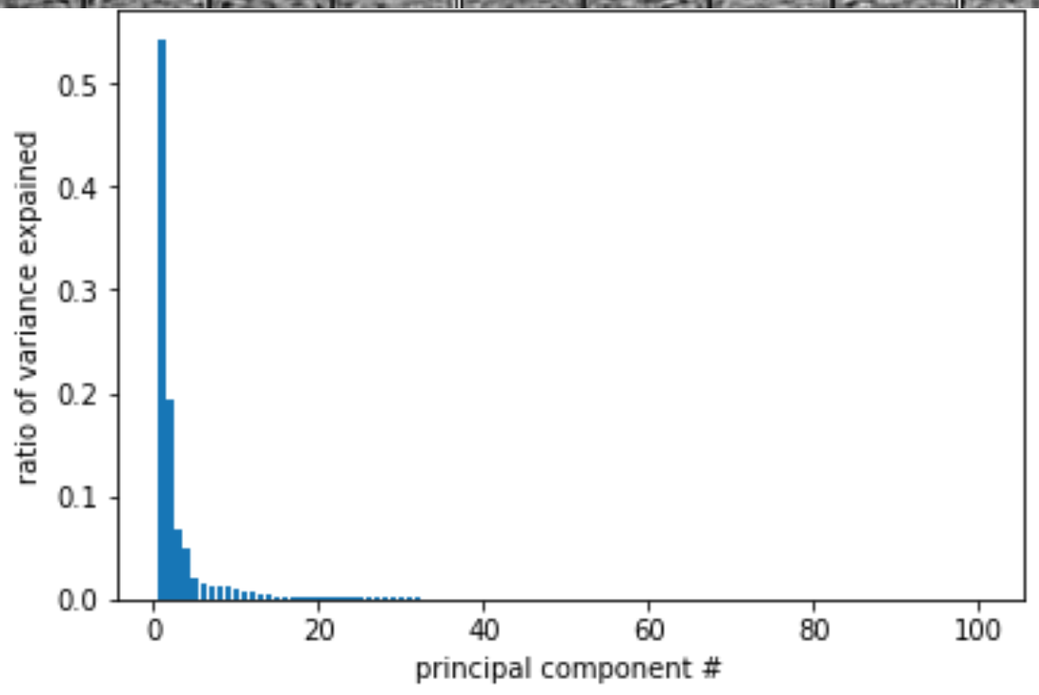


Original face



Reconstructed face with truncated number of principal components

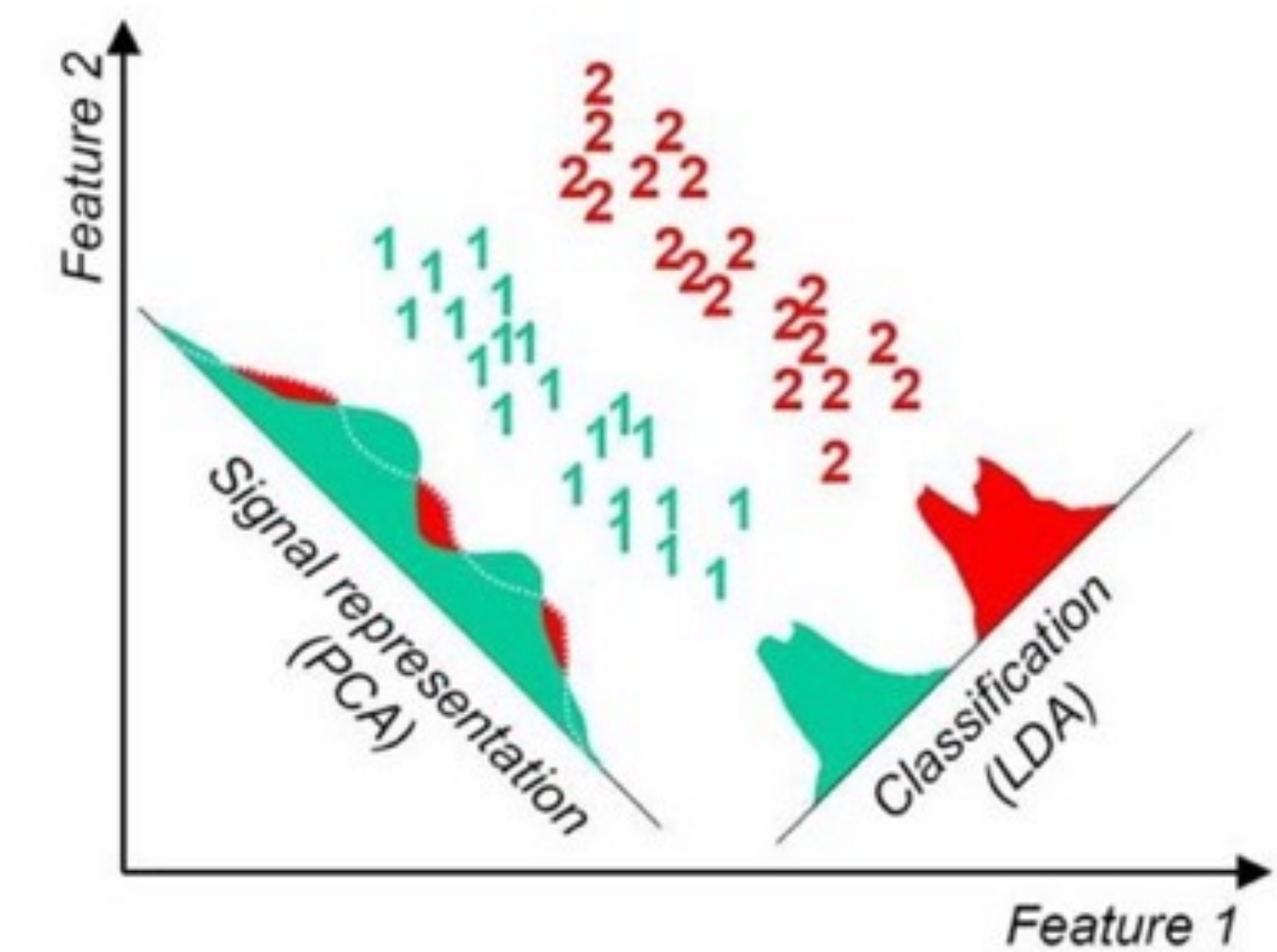



ratio of variance expained
principal component #
0.0
0.1
0.2
0.3
0.4
0.5
0
20
40
60
80
100

线性判别分析 (LDA)

●线性判别分析(Linear Discriminant Analysis):有监督的线性降维

基本思想：使用数据的类别信息，将高维的样本线性投影到低维空间中，使得数据样本在低维空间中，数据的类别区分度最大



线性判别分析 (LDA)

- Study a special case of LDA for binary classification:
  - Take n dimensional input data x and project it into one dimension using

$$y = w^T x$$

  - If y is larger or equal than 0, classify x as positive; otherwise classify x as negative.

线性判别分析 (LDA)

- Suppose there are m data samples

$$(x_1, y_1), \cdots, (x_m, y_m)$$

- The center of the positive and negative samples is

$$\mu_+ = \frac{1}{|\mathcal{C}_+|} \sum_{i \in \mathcal{C}_+} x_i$$

$$\mu_- = \frac{1}{|\mathcal{C}_-|} \sum_{i \in \mathcal{C}_-} x_i$$

$\mathcal{C}_+$ is the index set of positive samples

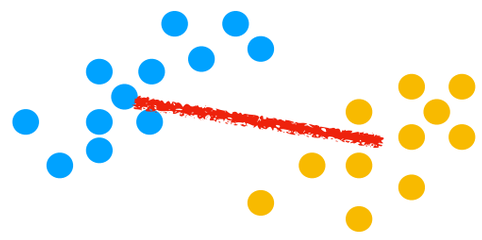
$\mathcal{C}_-$ is the index set of negative samples

线性判别分析 (LDA)

- Try to maximize the distance between the two classes projected center

$$||w^T(\mu_+ - \mu_-)||$$

- But need to normalize the norm of w.
- However, this can result in small "margin".



线性判别分析 (LDA)

- Key idea: to maximize the large separation between the projected class means, while to minimize the variance within each class: thereby minimizing the class overlap.

线性判别分析 (LDA)

- Compute the within class variance

$$s_+^2 = \frac{1}{|\mathcal{C}_+|} \sum_{i \in \mathcal{C}_+} (w^T(x_i - \mu_+))^2$$

$$s_-^2 = \frac{1}{|\mathcal{C}_-|} \sum_{i \in \mathcal{C}_-} (w^T(x_i - \mu_-))^2$$

- Goal:

$$\max_w J(w) = \frac{(w^T(\mu_+ - \mu_-))^2}{s_+^2 + s_-^2}$$

线性判别分析 (LDA)

- Rewrite

$$s_+^2 = \frac{1}{|\mathcal{C}_+|} \sum_{i \in \mathcal{C}_+} (w^T(x_i - \mu_+))^2$$

- as

$$s_+^2 = w^T \Sigma_+ w \qquad \Sigma_+ = \frac{1}{|\mathcal{C}_+|} \sum_{i \in \mathcal{C}_+} (x_i - \mu_+)(x_i - \mu_+)^T$$

- Similarly

$$s_-^2 = w^T \Sigma_- w \qquad \Sigma_- = \frac{1}{|\mathcal{C}_-|} \sum_{i \in \mathcal{C}_-} (x_i - \mu_-)(x_i - \mu_-)^T$$

线性判别分析 (LDA)

- Goal:

$$\max_w J(w) = \frac{(w^T(\mu_+ - \mu_-))^2}{s_+^2 + s_-^2}$$

- is equivalent to

$$\max_w J(w) = \frac{w^T A w}{w^T(\Sigma_+ + \Sigma_-)w}$$

- with

$$A = (\mu_+ - \mu_-)(\mu_+ - \mu_-)^T$$

线性判别分析 (LDA)

- To solve

$$\max_{w} J(w) = \frac{w^T A w}{w^T (\Sigma_+ + \Sigma_-) w}$$

- compute the derivative and set it to zero:

$$(w^T (\Sigma_+ + \Sigma_-) w) A w = (w^T A w)(\Sigma_+ + \Sigma_-) w$$

- thus

$$w \propto (\Sigma_+ + \Sigma_-)^{-1} A w \propto (\Sigma_+ + \Sigma_-)^{-1} (\mu_+ - \mu_-)$$

LDA与PCA

- 思想不同

PCA主要是从特征的协方差角度，去找到比较好的投影方式，即选择样本点投影具有最大方差的方向，最大化保留了原始数据本身的内部信息；而LDA则更多的是考虑了分类标签信息，寻求投影后不同类别之间数据点距离更大化以及同一类别数据点距离最小化，即选择分类性能最好的方向。

- 学习模式不同

PCA属于无监督式学习，因此大多场景下只作为数据预处理过程的一部分，需要与其他算法结合使用，例如将PCA与聚类、判别分析、回归分析等组合使用；而LDA是一种监督式学习方法，除了可以降维外，还可以进行预测应用，因此既可以组合其他模型一起使用，也可以独立使用。

其他方法

- Independent Component Analysis (ICA)
  - PCA是寻找不相关的主成分
  - ICA寻找独立的主成分
  - 独立→不相关
  - 不相关↛独立

- Demo

  Isomura, Takuya; Toyoizumi, Taro (2016). "A local learning rule for independent component analysis". *Scientific Reports*.

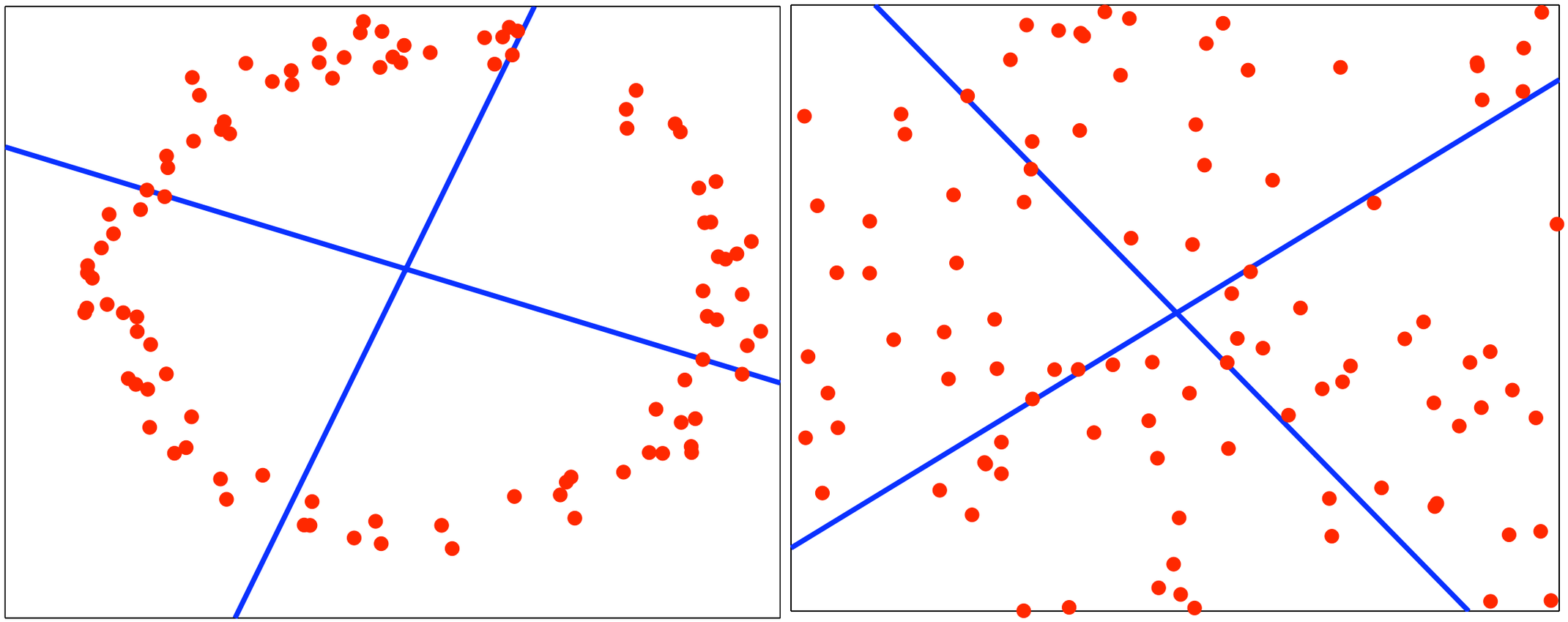
- Canonical Correlation Analysis (CCA)

非线性降维方法

- 现实生活中所获取到的数据集往往呈现出非线性结构，为了弥补线性降维方法的不足，人们提出了许多有效的非线性降维方法。

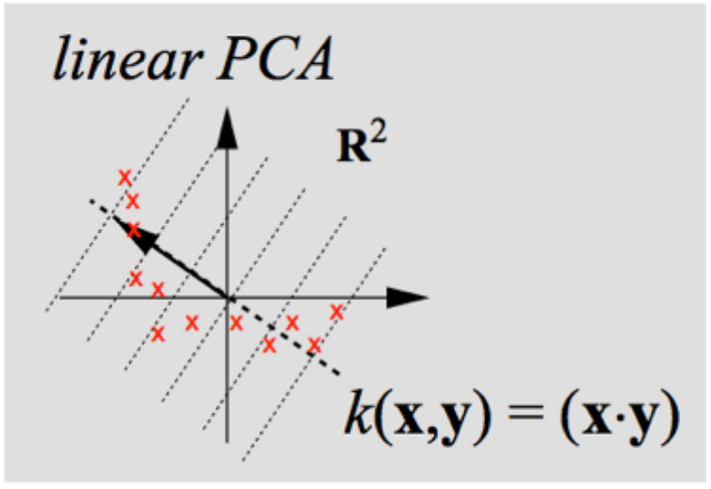
- 非线性降维方法主要可以分成两大类：
  - 基于核函数的降维方法（e.g. KPCA）
  - 基于流形学习的降维方法（e.g. ISOMAP、LLE、MDS、t-SNE）

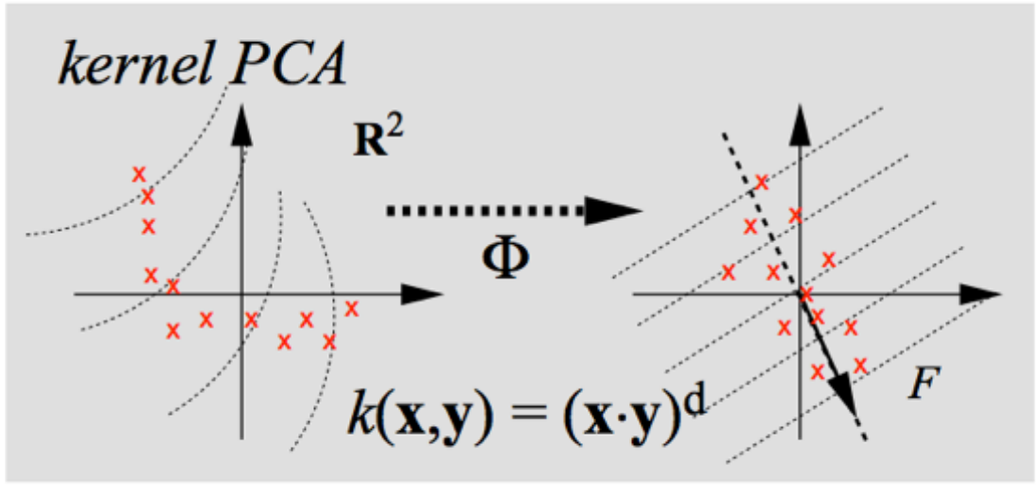
Difficult example



- PCA will make no difference between these examples, because the structure on the left is not linear
- Are there ways to find non-linear, low-dimensional manifolds?

Kernel PCA: idea





- linear PCA assumes Gaussian distribution
- idea: 把数据映射到空间ϕ(x) (similar to kernel SVM),使得在该空间下服从高斯分布
- 在ϕ(x)空间，应用linear PCA

PCA in feature space (I)

- Suppose for the moment that the mean of the data in feature space is 0, so: $\sum_{i=1}^{m} \phi(\mathbf{x}_i) = 0$
- The covariance matrix is:

$$\mathbf{C} = \frac{1}{m} \sum_{i=1}^{m} \phi(\mathbf{x}_i) \phi(\mathbf{x}_i)^T$$

- The eigenvectors are:

$$\mathbf{C}\mathbf{v}_j = \lambda_j \mathbf{v}_j, j = 1, \ldots N$$

- We want to avoid explicitly going to feature space - instead we want to work with *kernels*:

$$K(\mathbf{x}_i, \mathbf{x}_k) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_k)$$

PCA in feature space (II)

- Re-write the PCA equation:

$$\frac{1}{m}\sum_{i=1}^{m}\phi(\mathbf{x}_i)\phi(\mathbf{x}_i)^T\mathbf{v}_j = \lambda_j\mathbf{v}_j, j = 1,\ldots N$$

- So the eigenvectors can be written as a linear combination for features:

$$\mathbf{v}_j = \sum_{i=1}^{m} a_{ji}\phi(\mathbf{x}_i)$$

- Finding the eigenvectors is equivalent to finding the coefficients $a_{ji}, j = 1,\ldots N, i = 1,\ldots m$

PCA in feature space (III)

- By substituting this back into the equation we get:

$$\frac{1}{m}\sum_{i=1}^{m}\phi(\mathbf{x}_i)\phi(\mathbf{x}_i)^T\left(\sum_{l=1}^{m}a_{jl}\phi(\mathbf{x}_l)\right)=\lambda_j\sum_{l=1}^{m}a_{jl}\phi(\mathbf{x}_l)$$

- We can re-write this as:

$$\frac{1}{m}\sum_{i=1}^{m}\phi(\mathbf{x}_i)\left(\sum_{l=1}^{m}a_{jl}K(\mathbf{x}_i,\mathbf{x}_l)\right)=\lambda_j\sum_{l=1}^{m}a_{jl}\phi(\mathbf{x}_l)$$

- A small trick: multiply this by $\phi(\mathbf{x}_k)^T$ to the left:

$$\frac{1}{m}\sum_{i=1}^{m}\phi(\mathbf{x}_k)^T\phi(\mathbf{x}_i)\left(\sum_{l=1}^{m}a_{jl}K(\mathbf{x}_i,\mathbf{x}_l)\right)=\lambda_j\sum_{l=1}^{m}a_{jl}\phi(\mathbf{x}_k)^T\phi(\mathbf{x}_l)$$

PCA in feature space (IV)

- We plug in the kernel again:

$$\frac{1}{m}\sum_{i=1}^{m} K(\mathbf{x}_k, \mathbf{x}_i)\left(\sum_{l=1}^{m} a_{jl} K(\mathbf{x}_i, \mathbf{x}_l)\right) = \lambda_j \sum_{l=1}^{m} a_{jl} K(\mathbf{x}_k, \mathbf{x}_l), \forall j, k$$

- By rearranging we get: $\mathbf{K}^2\mathbf{a}_j = m\lambda_j \mathbf{K}\mathbf{a}_j$

- We can remove a factor of $\mathbf{K}$ from both sides of the matrix (this will only affect eigenvectors with eigenvalues 0, which will not be principle components anyway):

$$\mathbf{K}\mathbf{a}_j = m\lambda_j \mathbf{a}_j$$

PCA in feature space (V)

- We have a normalization condition for the $\mathbf{a}_j$ vectors:

$$\mathbf{v}_j^T \mathbf{v}_j = 1 \Rightarrow \sum_{k=1}^{m} \sum_{l=1}^{m} a_{jl} a_{jk} \phi(\mathbf{x}_l)^T \phi(\mathbf{x}_k) = 1 \Rightarrow \mathbf{a}_j^T \mathbf{K} \mathbf{a}_j = 1$$

- Plugging this into:

$$\mathbf{K}\mathbf{a}_j = m\lambda_j \mathbf{a}_j$$

  we get: $\lambda_j m \mathbf{a}_j^T \mathbf{a}_j = 1, \forall j$

- For a new point $\mathbf{x}$, its projection onto the principal components is:

$$\phi(\mathbf{x})^T \mathbf{v}_j = \sum_{i=1}^{m} a_{ji} \phi(\mathbf{x})^T \phi(\mathbf{x}_i) = \sum_{i=1}^{m} a_{ji} K(\mathbf{x}, \mathbf{x}_i)$$

Normalizing the feature space

- In general, the features $\phi(\mathbf{x}_i)$ may not have mean 0
- We want to work with:

$$\tilde{\phi}(\mathbf{x}_i) = \phi(\mathbf{x}_i) - \frac{1}{m}\sum_{k=1}^{m}\phi(\mathbf{x}_k)$$

- The corresponding kernel matrix entries are given by:

$$\tilde{K}(\mathbf{x}_k, \mathbf{x}_l) = \tilde{\phi}(\mathbf{x}_l)^T\tilde{\phi}(\mathbf{x}_j)$$

- After some algebra, we get:

$$\tilde{\mathbf{K}} = \mathbf{K} - 2\mathbf{1}_{1/m}\mathbf{K} + \mathbf{1}_{1/m}\mathbf{K}\mathbf{1}_{1/m}$$

where $\mathbf{1}_{1/m}$ is the matrix with all elements equal to $1/m$

Summary of kernel PCA

1. Pick a kernel
2. Construct the normalized kernel matrix $\tilde{\mathbf{K}}$ of the data (this will be of dimension $m \times m$)
3. Find the eigenvalues and eigenvectors of this matrix λ_j, $\mathbf{a}_j$
4. For any data point (new or old), we can represent it as the following set of features:

$$y_j = \sum_{i=1}^{m} a_{ji} K(\mathbf{x}, \mathbf{x}_i), j = 1, \ldots m$$

5. We can limit the number of components to $k < m$ for a more compact representation (by picking the $\mathbf{a}$'s corresponding to the highest eigenvalues)

Representation obtained by kernel PCA

- Each y_j is the coordinate of $\phi(\mathbf{x})$ along one of the feature space axes $\mathbf{v}_j$
- Remember that $\mathbf{v}_j = \sum_{i=1}^{m} a_{ji}\phi(\mathbf{x}_i)$ (the sum goes to k if $k < m$)
- Since $\mathbf{v}_j$ are orthogonal, the projection of $\phi(\mathbf{x})$ onto the space spanned by them is:

$$\Pi\phi(\mathbf{x}) = \sum_{j=1}^{m} y_j \mathbf{v}_j = \sum_{j=1}^{m} y_j \sum_{i=1}^{m} a_{ji}\phi(\mathbf{x}_i)$$

(again, sums go to k if $k < m$)

- The *reconstruction error in feature space* can be evaluated as:

$$\|\phi(\mathbf{x}) - \Pi\phi(\mathbf{x})\|^2$$

This can be re-written by expanding the norm; we obtain dot-products which can all be replaced by kernels

- Note that the error will be 0 on the training data if enough $\mathbf{v}_j$ are retained

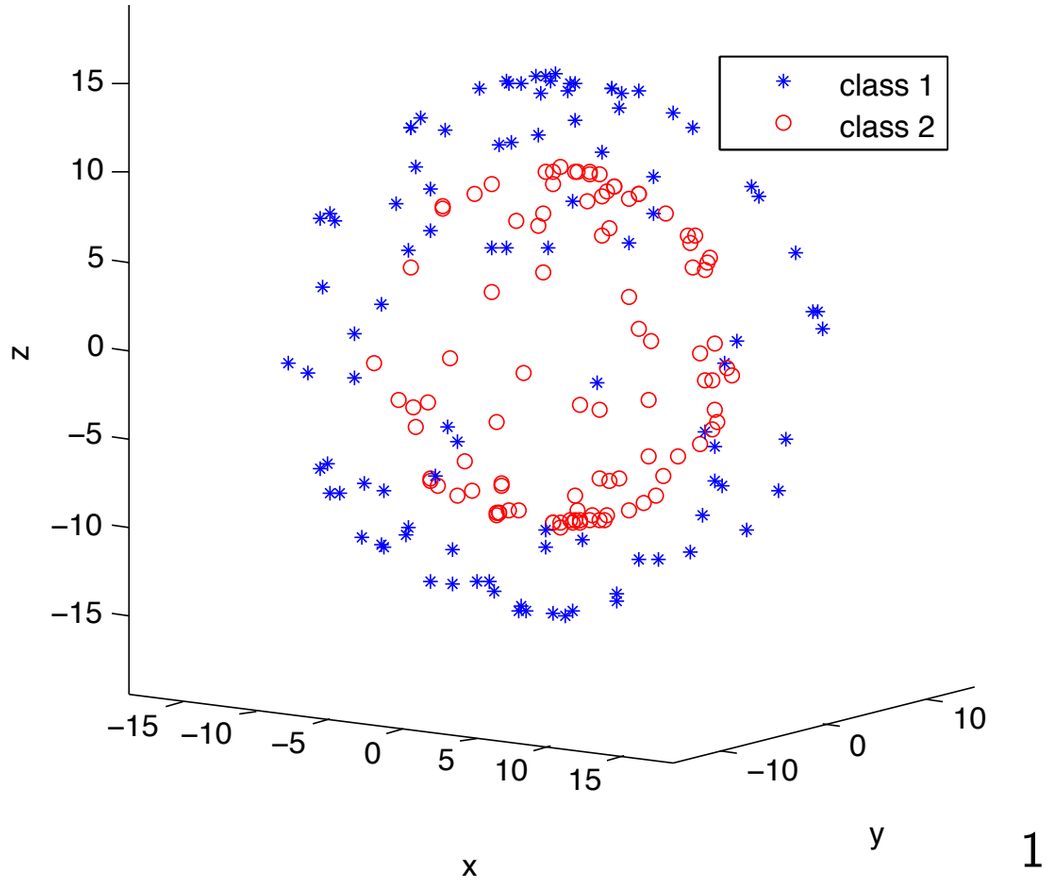
Alternative reconstruction error measures

- An alternative way of measuring performance is by looking at how well kernel PCA preserves distances between data points
- In this case, the Euclidian distance in kernel space between points $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x}_j)$, d_{ij}, is:

$$\|\phi\mathbf{x}_i) - \phi(\mathbf{x}_j)\| = K(\mathbf{x}_i, \mathbf{x}_i) + K(\mathbf{x}_j, \mathbf{x}_j) - 2K(\mathbf{x}_i, \mathbf{x}_j)$$

- The distance $\hat{d}_ij$ between the projected points in kernel space is defined as above, but with $\phi(\mathbf{x}_i)$ replaced by $\Pi\phi(\mathbf{x}_i)$.
- The average of $d_{ij} - \hat{d}_{ij}$ over all pairs of points is a measure of reconstruction error
- Note that reconstruction error in the original space of the $\mathbf{x}_i$ is very difficult to compute, because it requires taking $\Pi\phi(\mathbf{x})$ and finding its pre-image in the original feature space, which is not always feasible (though approximations exist)

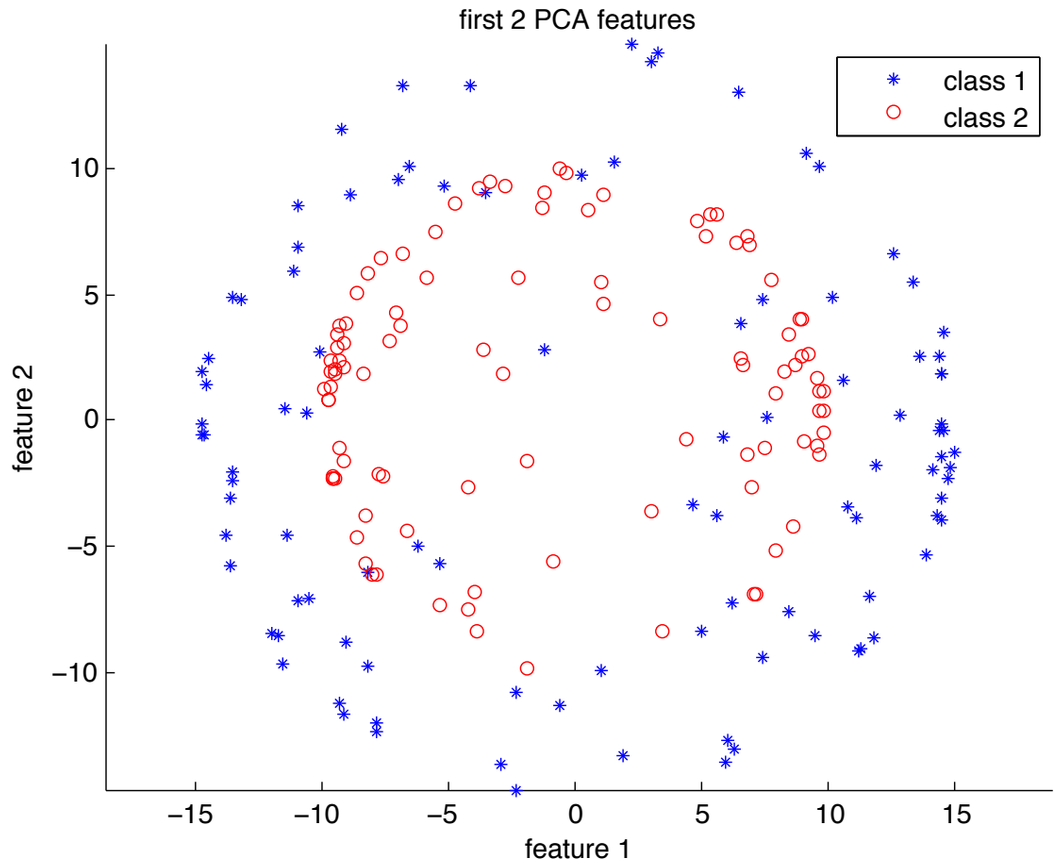
Example: Two concentric spheres



1

- Colours are used for clarity in the picture, but the data is presented unlabelled
- We want to project form 3D to 2D

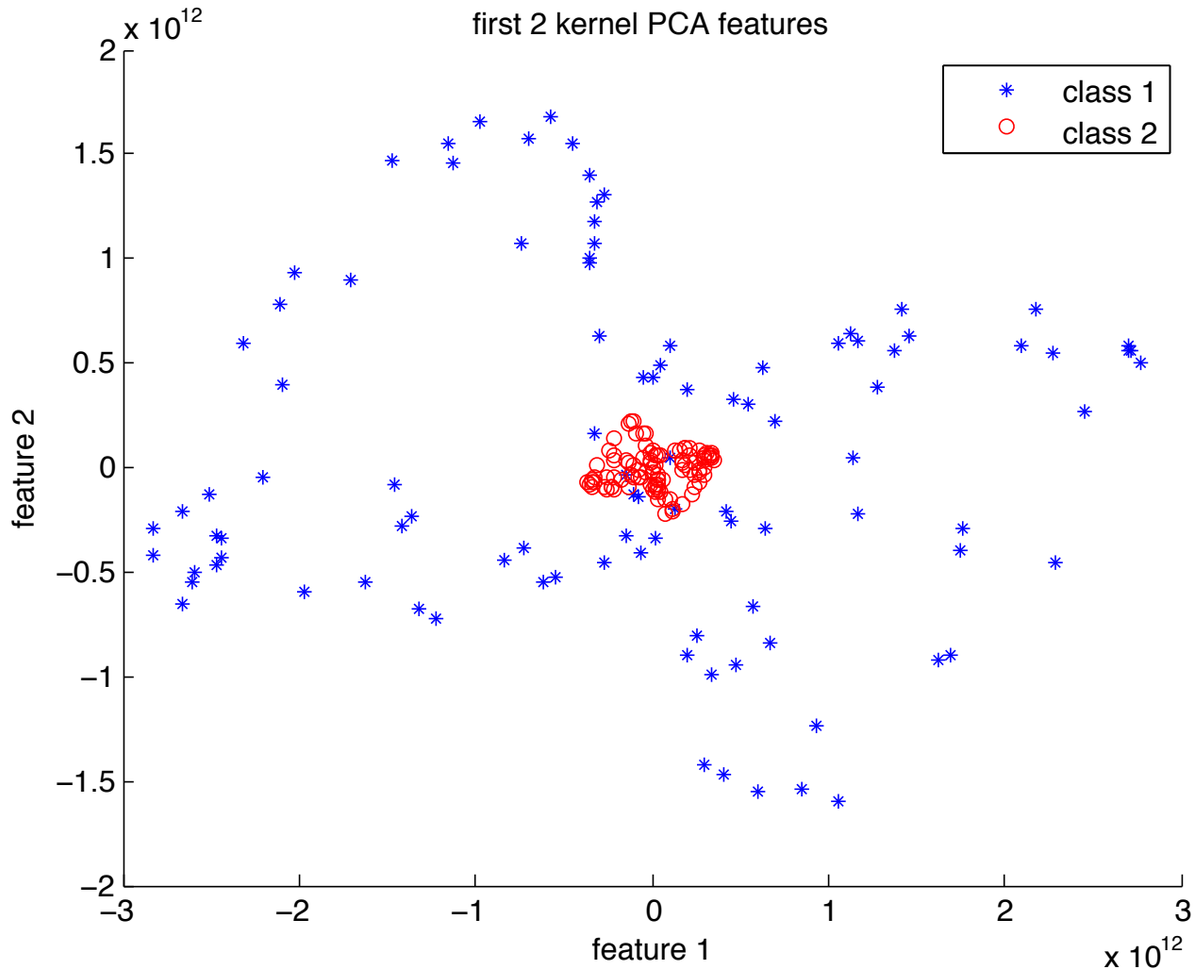
Example: Two concentric spheres - PCA



2

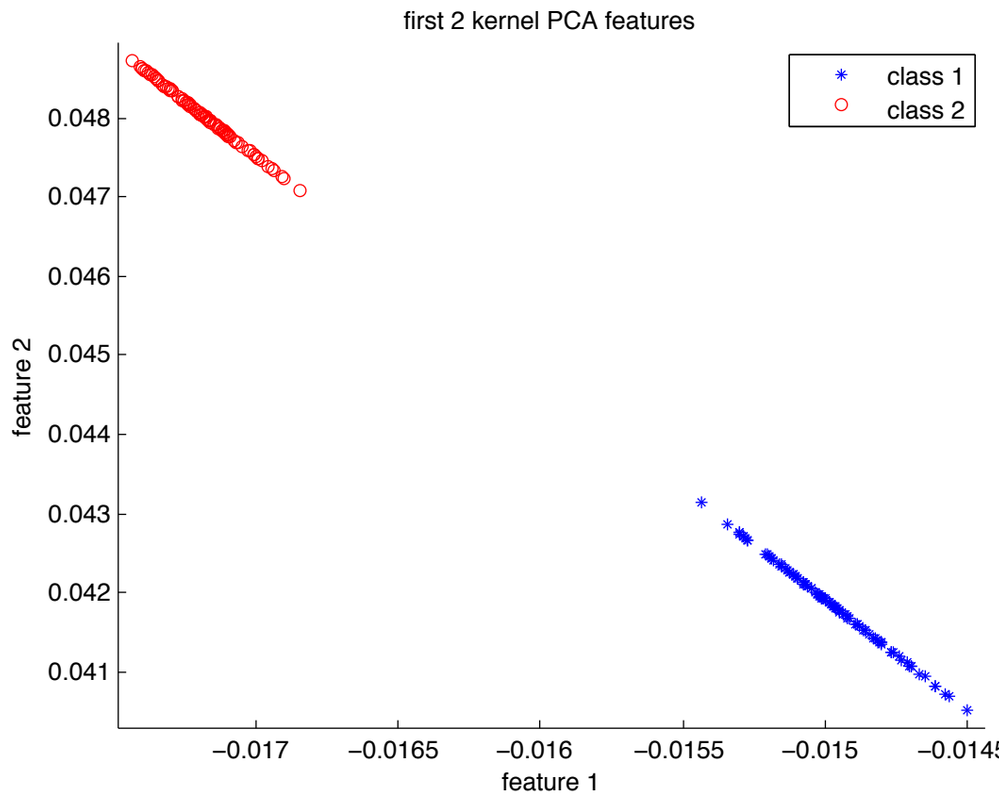
Note that PCA is unable to separate the points from the two spheres

Example: Kernel PCA with Polynomial Kernel ($d = 5$)



- Points from one sphere are much closer together, the others are scattered
- The projected data is not linearly separable

Example: Kernel PCA with Gaussian Kernel ($\sigma = 20$)

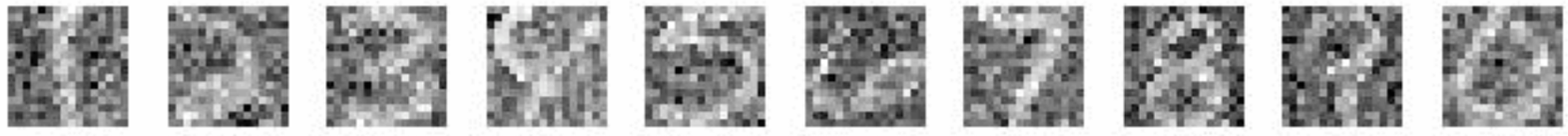


- Points from the two spheres are really well separated
- Note that the choice of parameter for the kernel matters!
- Validation can be used to determine good kernel parameter values

Example: De-noising images

Original data

Data corrupted with Gaussian noise

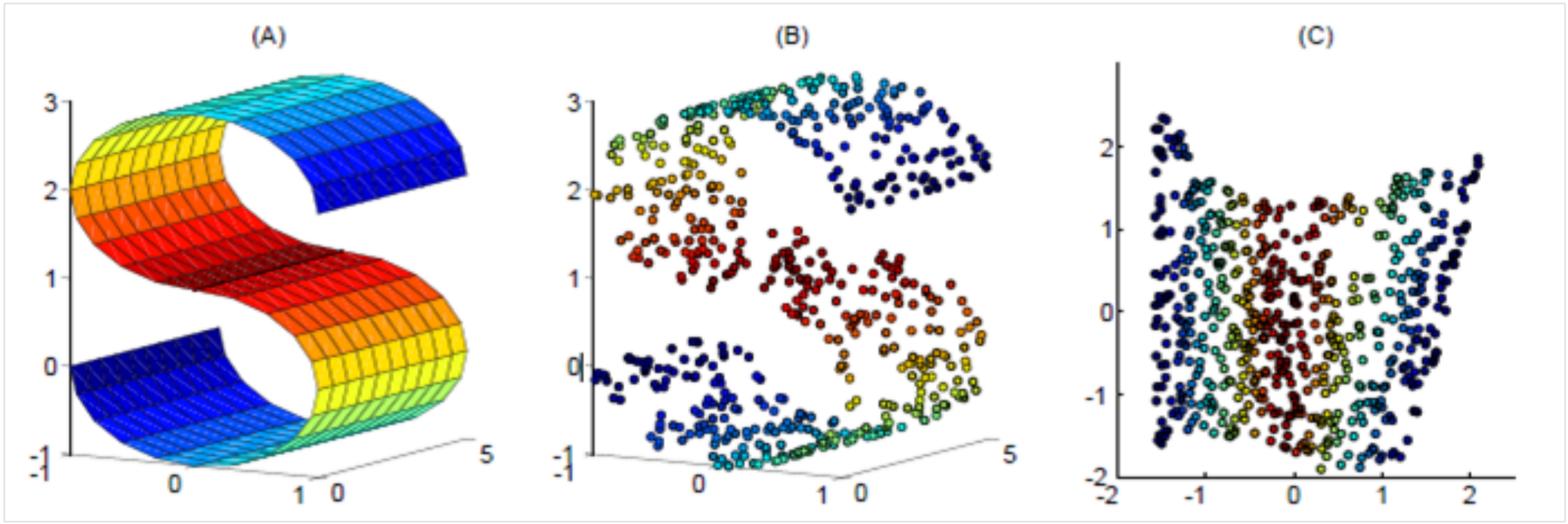


Result after linear PCA

Result after kernel PCA, Gaussian kernel

局部线性嵌入（LLE）

- LLE（Locally linear embedding，局部线性嵌入）的主要思路就是将数据降到低维空间中，但是保留数据局部的几何信息。对于每个点，有一组权系数对它的领域点进行加权，从而重构它，这组权系数会使得重构的误差最小。
- 使用LLE将三维数据（图B）映射到二维（图C）之后，映射的数据仍能保持原有的数据流形（红色的点相互靠近，蓝色的也相互靠近）。



https://cs.nyu.edu/~roweis/lle/

Locally Linear Embedding

- $\mathbf{x}_1, \cdots, \mathbf{x}_m \in \mathbb{R}^n$ lies on a k-dimensional manifold.

$\Rightarrow$ Each point and its neighbors lie close to a *locally linear* patch of the manifold.

- We try to reconstruct each point from its neighbors:

$$\min_{\mathbf{W}} \sum_i \|\mathbf{x}_i - \sum_j \mathbf{W}_{i,j}\mathbf{x}_j\|^2$$

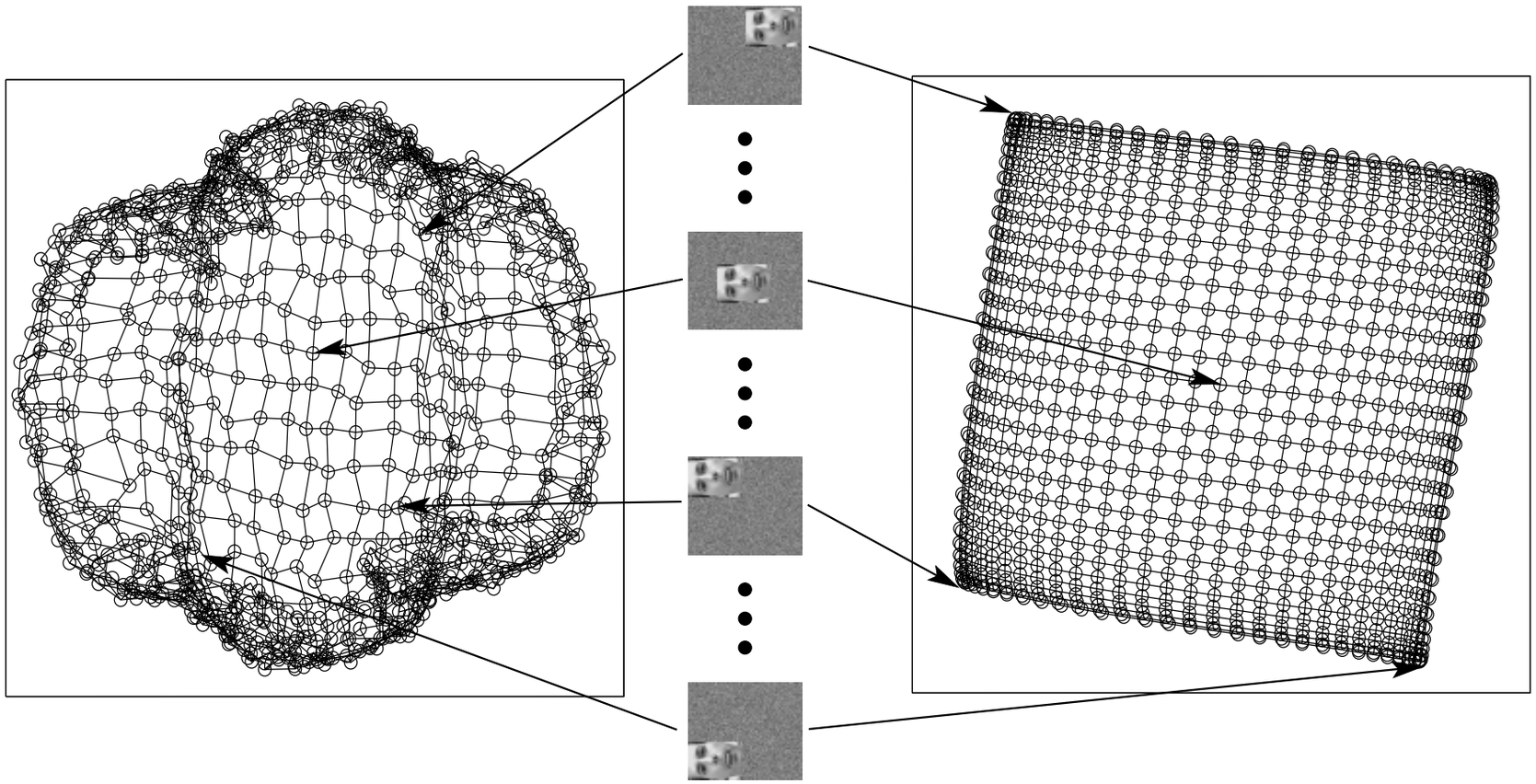
s.t. $\mathbf{W1} = \mathbf{1}$ and $\mathbf{W}_{i,j} = 0$ if $\mathbf{x}_j \notin neighbors(\mathbf{x}_i)$

$\Rightarrow$ For each point the weights are invariant to rotation, scaling and translations: the weights $\mathbf{W}_{i,j}$ capture intrinsic geometric properties of each neighborhood.

- These local properties of each neighborhood should be preserved by the embedding:

$$\min_{\mathbf{z}_1,\ldots,\mathbf{z}_m \in \mathbb{R}^k} \sum_i \|\mathbf{z}_i - \sum_j \mathbf{W}_{i,j}\mathbf{z}_j\|^2$$

PCA vs Locally Linear Embedding



[Saul, L. K., & Roweis, S. T. (2000). An introduction to locally linear embedding.]

Multi-dimensional scaling

- Input:
  - An $m \times m$ dissimilarity matrix d, where $d(i, j)$ is the distance between instances $\mathbf{x}_i$ and $\mathbf{x}_j$
  - Desired dimension k of the embedding.
- Output:
  - Coordinates $\mathbf{z}_i \in \mathbb{R}^k$ for each instance i that minimize a "stress" function quantifying the mismatch between distances as given by d and distances of the data representation in $\mathbb{R}^k$.

Stress functions

- Common stress functions include:
  - The least-squares or Kruskal-Shephard criterion:

$$\sum_{i=1}^{m}\sum_{j\neq i}(d(i,j)-\|\mathbf{z}_i-\mathbf{z}_j\|)^2$$

  - The Sammon mapping:

$$\sum_{i=1}^{m}\sum_{j\neq i}\frac{(d(i,j)-\|\mathbf{z}_i-\mathbf{z}_j\|)^2}{d(i,j)},$$

    which emphasizes getting small distances correct.
- Gradient-based optimization is usually used to find $\mathbf{z}_i$

A visualization method: t-SNE

- t-distributed Stochastic Neighborhood Embedding
- A popular visualization method and data explorative analysis tool
- Project data into 2D or 3D

Journal of Machine Learning Research 9 (2008) 2579-2605 Submitted 5/08; Revised 9/08; Published 11/08

Visualizing Data using t-SNE

Laurens van der Maaten LVDMAATEN@GMAIL.COM
TiCC
Tilburg University
P.O. Box 90153, 5000 LE Tilburg, The Netherlands

Geoffrey Hinton HINTON@CS.TORONTO.EDU
Department of Computer Science
University of Toronto
6 King's College Road, M5S 3G4 Toronto, ON, Canada

Editor: Yoshua Bengio