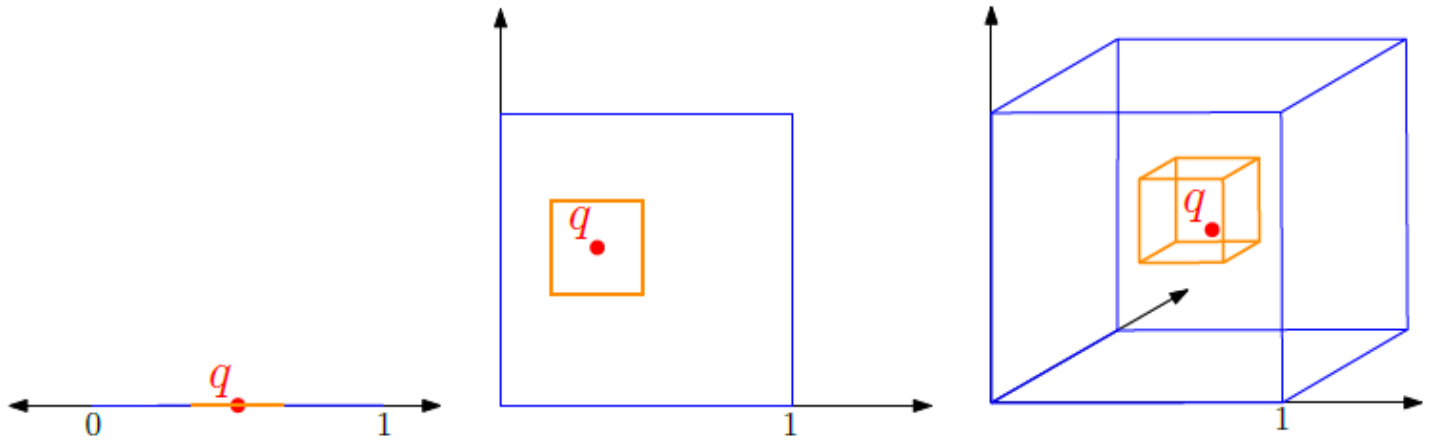
THE UNIVERSITY of TENNESSEE KNOXVILLE

Lecture 15: Linear Dimensionality Reduction Methods

Instructor: Sergei V. Kalinin

Reminder: dimensionality is a problem

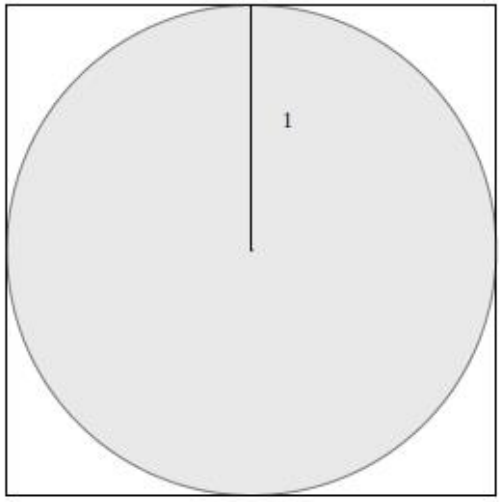


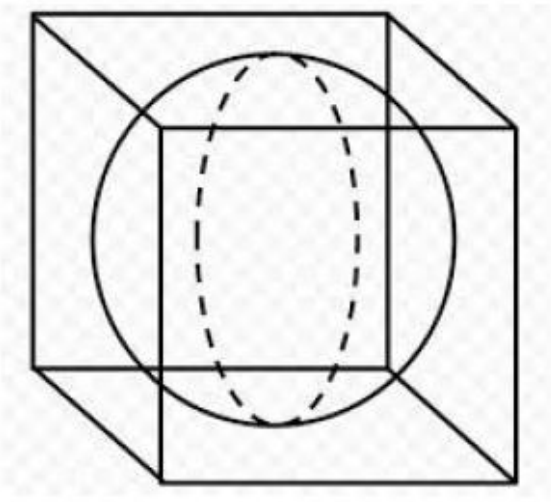
Suppose we have 5000 points:

- In 1d we have to explore 0.001 on average to capture 5 NN
- In 2d, on average we must explore 0:031 units along both dimensions to get 5 nearest neighbors points (about 3% of the whole cube).
- In 3d, on average we must go 10% of the total (unit) length in each of the 3 dimensions
- In 4d, we must explore 17:7% of unit length
- In 10d, we must go 50.1% of unit length along each dimension

From “Curse of Dimensionality”, Lecture Notes for Big Data Analytics, Faizad Ullah, February 2019

Reminder: dimensionality is a problem





dim m	volume of m-ball	volume of m-cube	ratio
2	π	2^2	~ 0.785
3	$4/3\pi$	2^3	~ 0.523
4	$\pi^2/2$	2^4	~ 0.308
6	$\pi^3/6$	2^6	~ 0.080
m	$\frac{\pi^{m/2}}{m/2!}$	2^m	$\to 0$

From “Curse of Dimensionality”, Lecture Notes for Big Data Analytics, Faizad Ullah, February 2019

Reminder: dimensionality is a problem

However if a dataset exhibit this phenomenon that the issue has be overcome by getting a larger training set (exponential in m). One way to look at this is as follows.

To cover $[-1,1]^m$ with $B_{m,1}$'s, the number of balls n must be

$$n \geq \frac{2^m}{V_m(1)} = \frac{2^m}{\pi^{m/2}/_{m/2!}} = \frac{m/2!\, 2^m}{\pi^{m/2}} \overset{m\to\infty}{\sim} \sqrt{m\pi}\left(\frac{m2^{m/2}}{2\pi e}\right)^{m/2}$$

For $m = 16$ (a very small number) this n is substantially larger than 2^{58}

- In higher dimensions all the volume is in `corners'
- Points in high dimensional spaces are isolated (empty surrounding)
- The probability that a randomly generated point is within r radius of q approaches 0 as dimensionality increases
- The probability of a close nearest neighbor in a data set is very small

From “Curse of Dimensionality”, Lecture Notes for Big Data Analytics, Faizad Ullah, February 2019

High Dimensional Data – what should we do?

Examples of high D data:

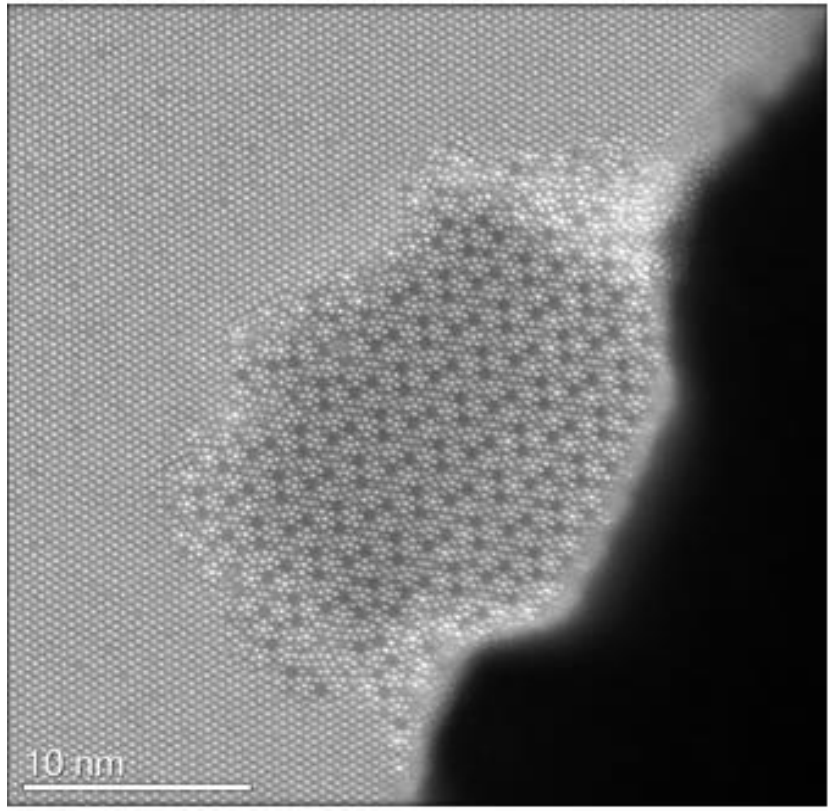
- Face recognition
- Image compression
- Gene expression analysis
- Spectroscopy
- 4D STEM
- X-Ray scattering

What do we want to accomplish?

- Reduce number of dimensions in data
- Find patterns in high-dimensional data
- Visualize data of high dimensionality

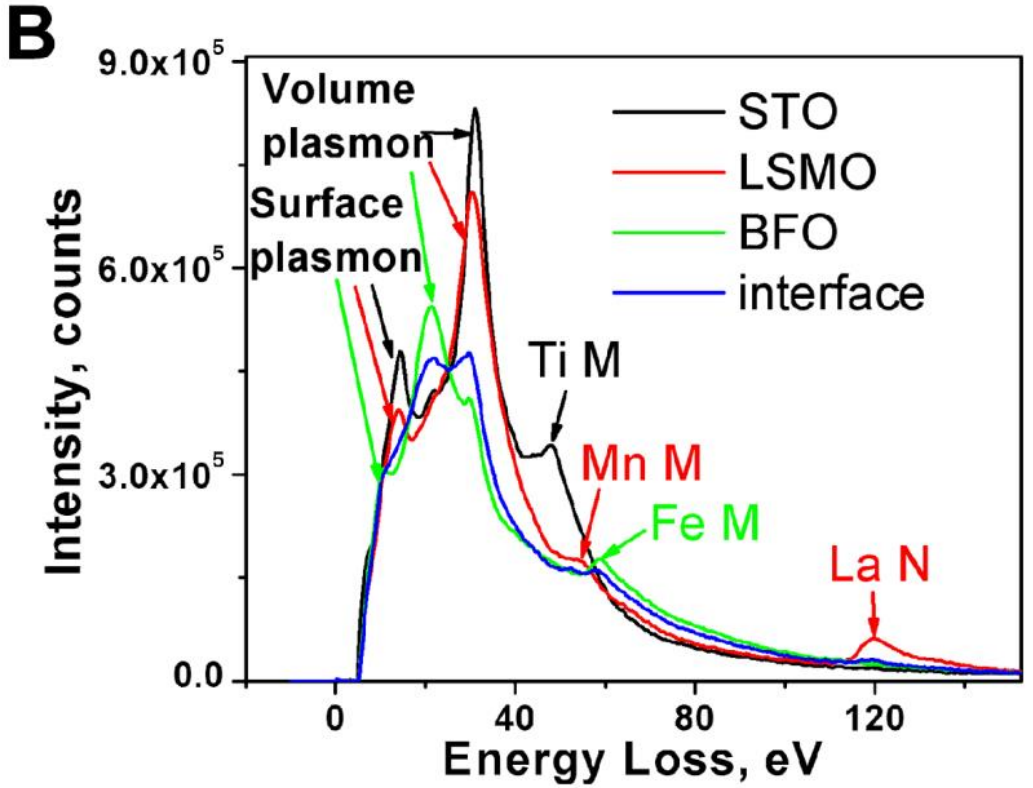
High D data in materials science?

Mo-V-Ta complex oxide



Q. He et al, ACS Nano 9, 3470-3478

Low-loss EELS spectra



A.Y. Borisevich et al., Phys. Rev. Lett. **105**, 087204 (2010).

- How many dimensions are in this data?
- Are all these dimensions necessary?
- For given acquisition time, how would the noise and signal balance?
- How do we extract “useful” information?

High Dimensional Data is often redundant

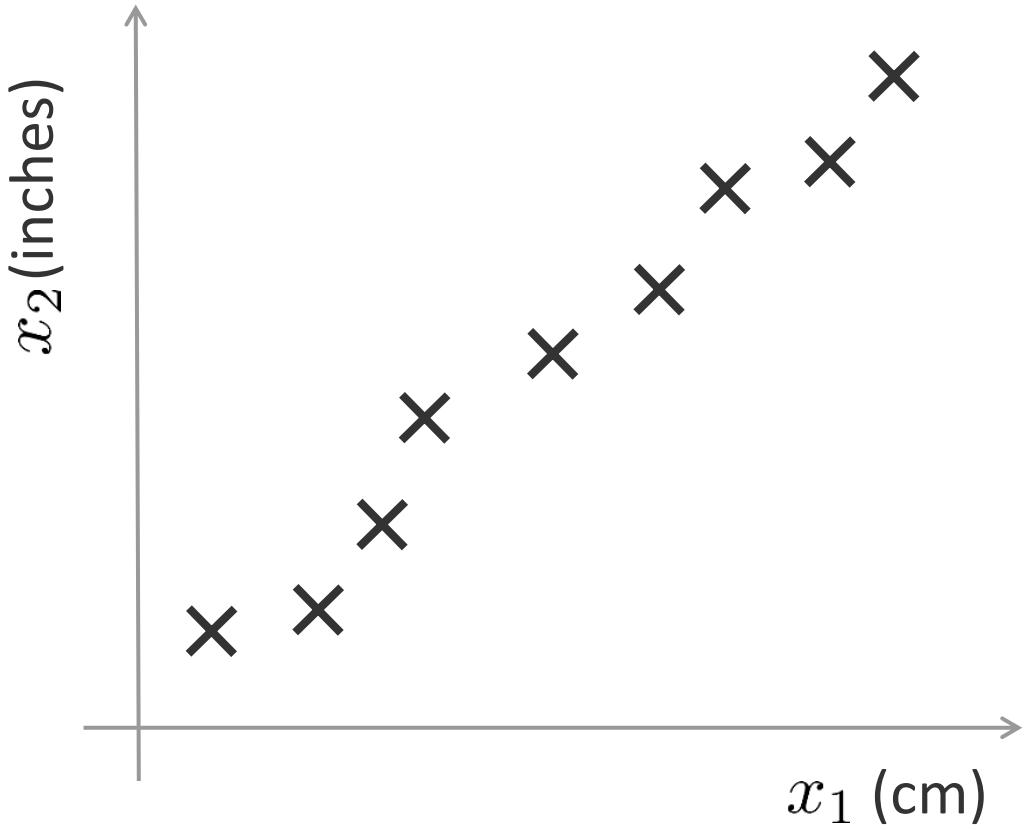
- We often need a method to simplify data on a large number of variables, and believe that there is some redundancy in those variables.
- Redundancy means that some of the variables are correlated with one another, possibly because they are measuring the same object or phenomenon.
- Because of redundancy, we believe that it should be possible to reduce the observed variables into a smaller number of artificial variables that will account for most of the variance in the observed variables.

Dimensionality Reduction Methods

- PCA (Principal Component Analysis):
  - Find projection that maximize the variance
- ICA (Independent Component Analysis):
  - Very similar to PCA except that it assumes non-Guassian features
- Multidimensional Scaling:
  - Find projection that best preserves inter-point distances
- LDA(Linear Discriminant Analysis):
  - Maximizing the component axes for class-separation
- Bayesian Linear Unmixing
  - Linear unmixing, non-negative, sum to one
- … constrained linear unmixing methods

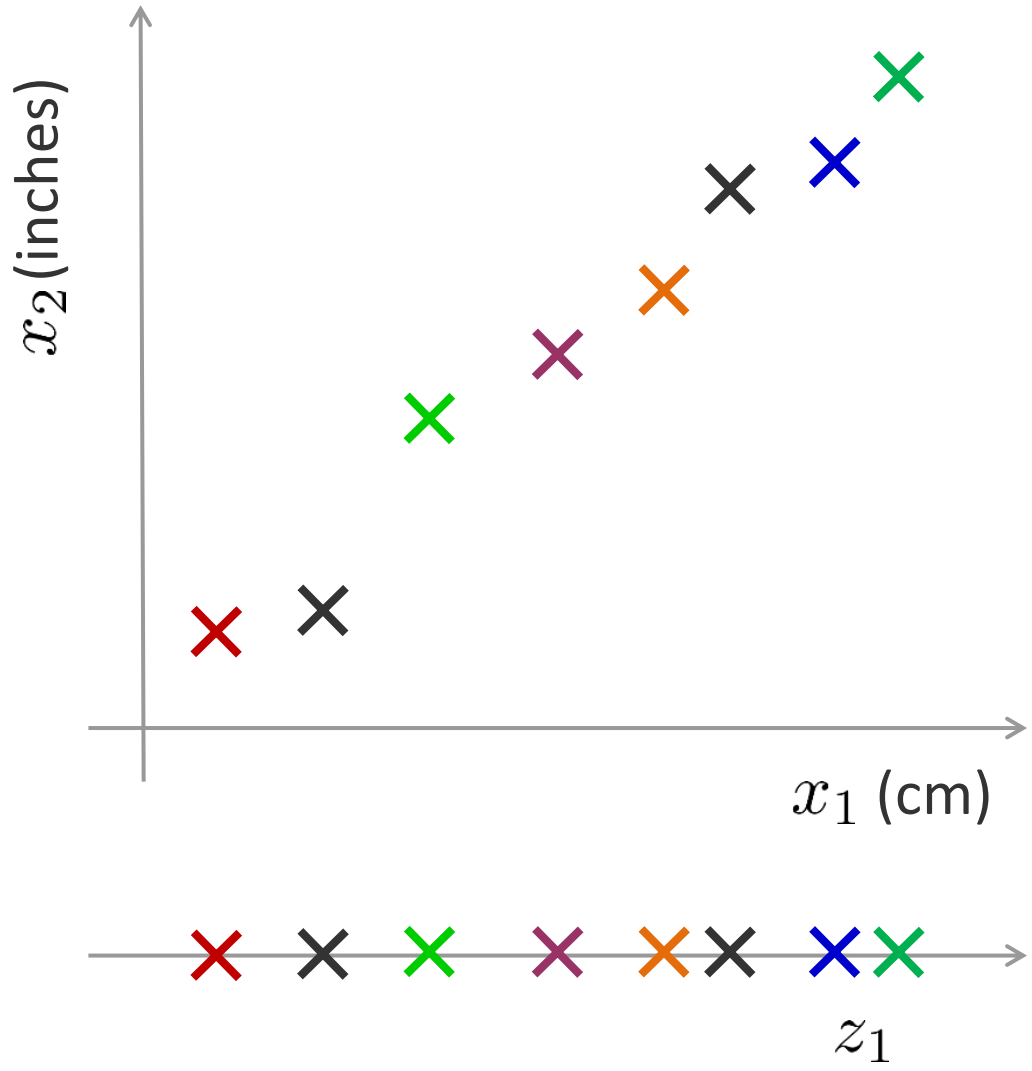
Manifold Hypothesis!

Simple Example



Reduce data from 2D to 1D

Simple Example



Reduce data from 2D to 1D

$$x^{(1)} \rightarrow z^{(1)}$$

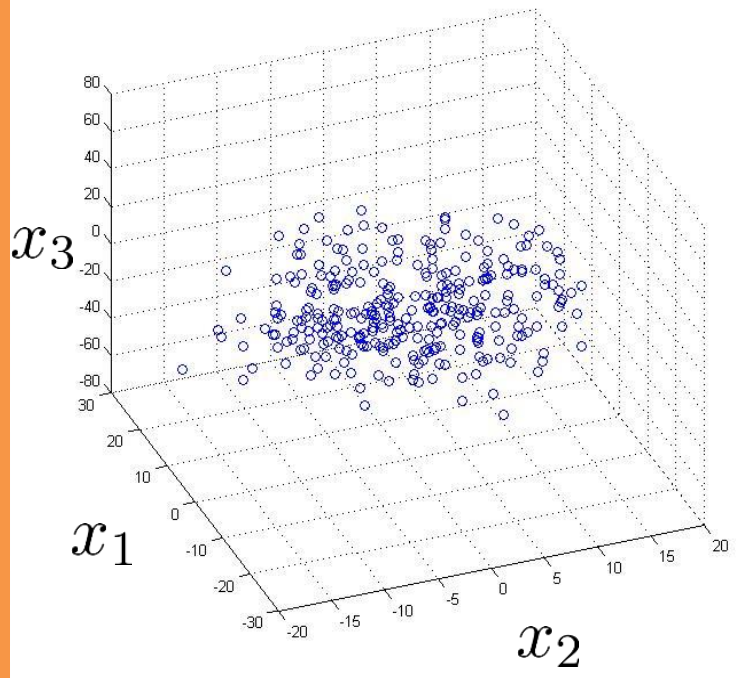
$$x^{(2)} \rightarrow z^{(2)}$$

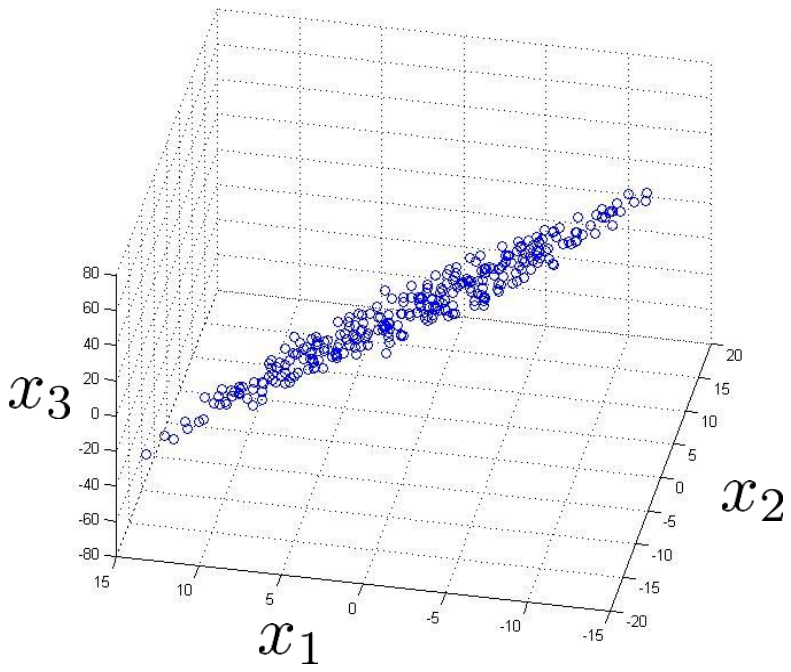
$$\vdots$$

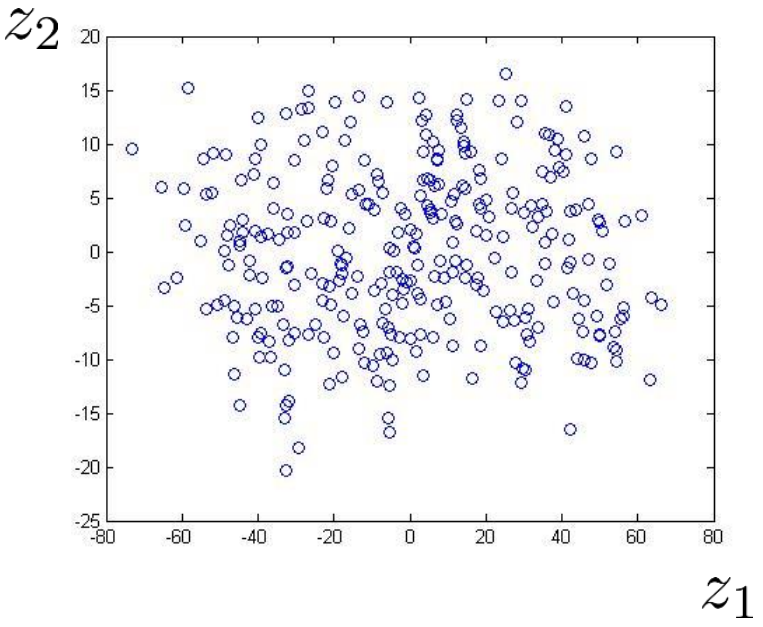
$$x^{(m)} \rightarrow z^{(m)}$$

Another Simple Example

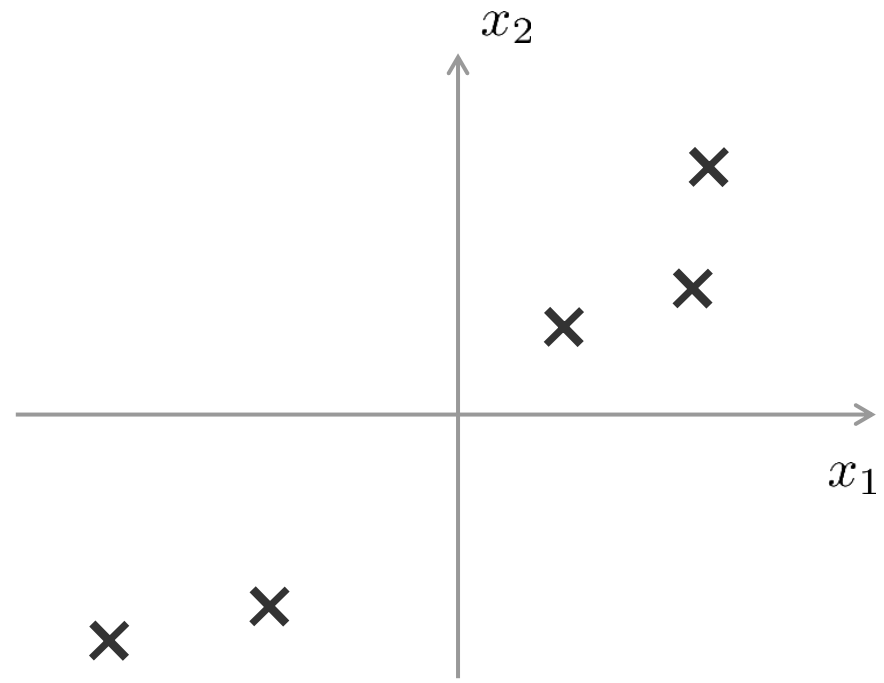
Reduce data from 3D to 2D



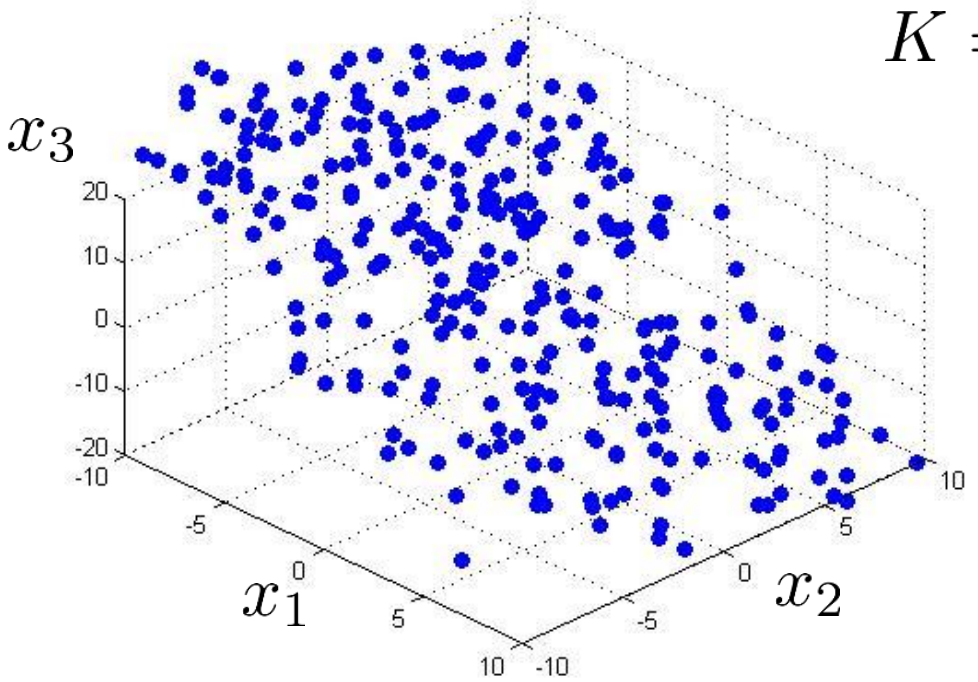




Generalize the problem



$3D \rightarrow 2D$
$K = 2$



Reduce from 2-dimension to 1-dimension: Find a direction (a vector $u^{(1)} \in \mathbb{R}^n$) onto which to project the data so as to minimize the projection error.

Reduce from n-dimension to k-dimension: Find k vectors $u^{(1)}, u^{(2)}, \ldots, u^{(k)}$ onto which to project the data, so as to minimize the projection error.

Variance and covariance

1D: Variance=(Standard deviation)^2

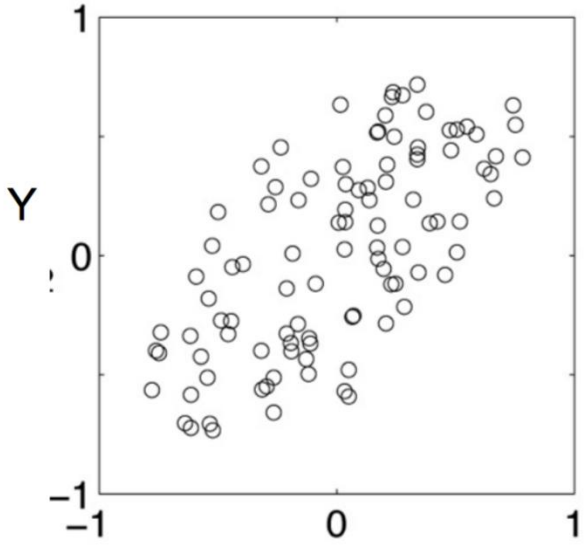
$$s^2 = \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{(n-1)}$$

2D: Covariance: measures the correlation between X and Y

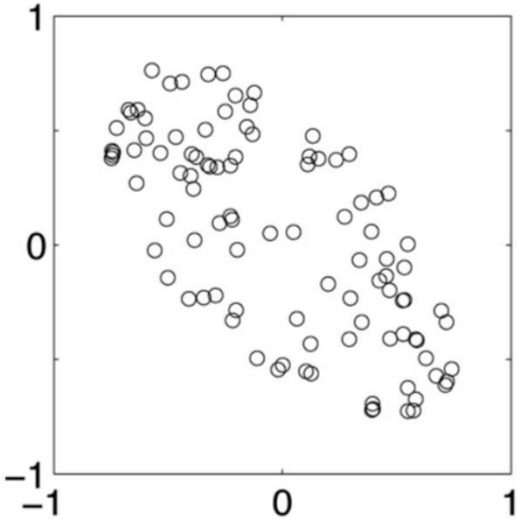
- Cov(X,Y)=0: independent
- Cov(X,Y)>0: move in the same direction
- Cov(X,Y)<0: move in opposite direction

$$\mathrm{cov}(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)}$$

positive covariance



negative covariance



Multidimensional data: covariance matrix

- Contains covariance values between all possible dimensions (=attributes):

$$C^{nxn} = (c_{ij} \mid c_{ij} = \mathrm{cov}(Dim_i, Dim_j))$$

- Example for three attributes (x,y,z):

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

- Eigenvalues of covariance matrix contain information on the independent factors of variability
- Eigenvectors of covariance matrix provide the information on direcitons

Principal Component Analysis

- Center the data (subtract the mean $\boldsymbol{\mu} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n$ from each data point)
- Compute the $D \times D$ covariance matrix $\mathbf{S}$ using the centered data matrix $\mathbf{X}$ as

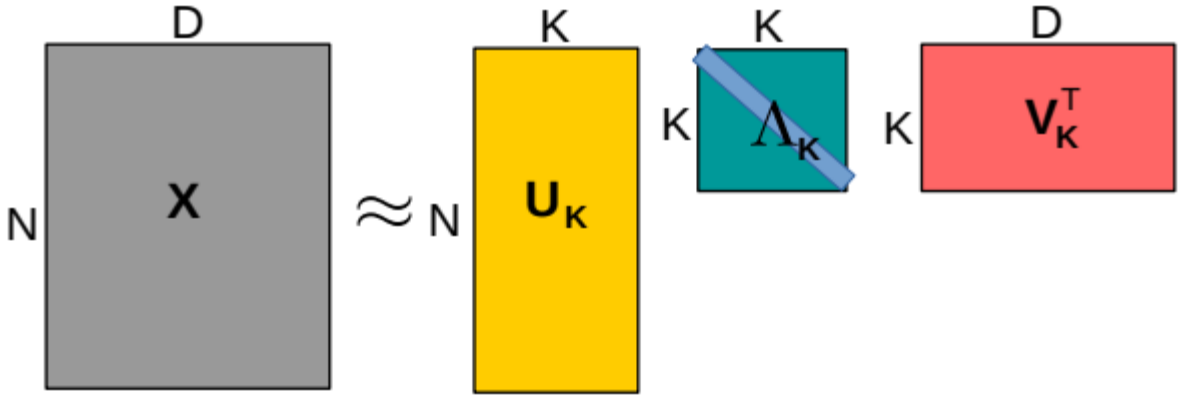
$$\mathbf{S} = \frac{1}{N}\mathbf{X}^\top\mathbf{X}$$ (Assuming $\mathbf{X}$ is arranged as $N \times D$)

- Do an eigen decomposition of the covariance matrix $\mathbf{S}$ (many methods exist)
- Take top $K < D$ leading eigvectors $\{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_K\}$ with eigen values $\{\lambda_1, \lambda_2, \ldots, \lambda_K\}$
- The K-dimensional projection/embedding of each input is $\boldsymbol{z}_n \approx \mathbf{W}_K^\top \boldsymbol{x}_n$
- Where $\mathbf{W}_\mathrm{K} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_K]$ is projection matrix of size $D \times K$

Adapted from Introduction to Machine Learning by Nisheeth

Singular Value Decomposition

- If we just use the top $K < \min\{N, D\}$ singular values, we get a rank-K SVD



$$\mathbf{X} \approx \hat{\mathbf{X}} = \sum_{k=1}^{K} \lambda_k \boldsymbol{u}_k \boldsymbol{v}_k^\top = \mathbf{U}_K \Lambda_K \mathbf{V}_K^\top$$

reconstruction error $\|\boldsymbol{X} - \hat{\boldsymbol{X}}\|$'

- Fact: SVD gives the best rank-K approximation of a matrix

- PCA is done by doing SVD on the covariance matrix **S** (left and right singular vectors are the sar and become eigenvectors, singular values become eigenvalues)

Adapted from Introduction to Machine Learning by Nisheeth

Principal Component Analysis

PCA: orthogonal transformation converting possibly correlated variables into linearly uncorrelated *principal components*

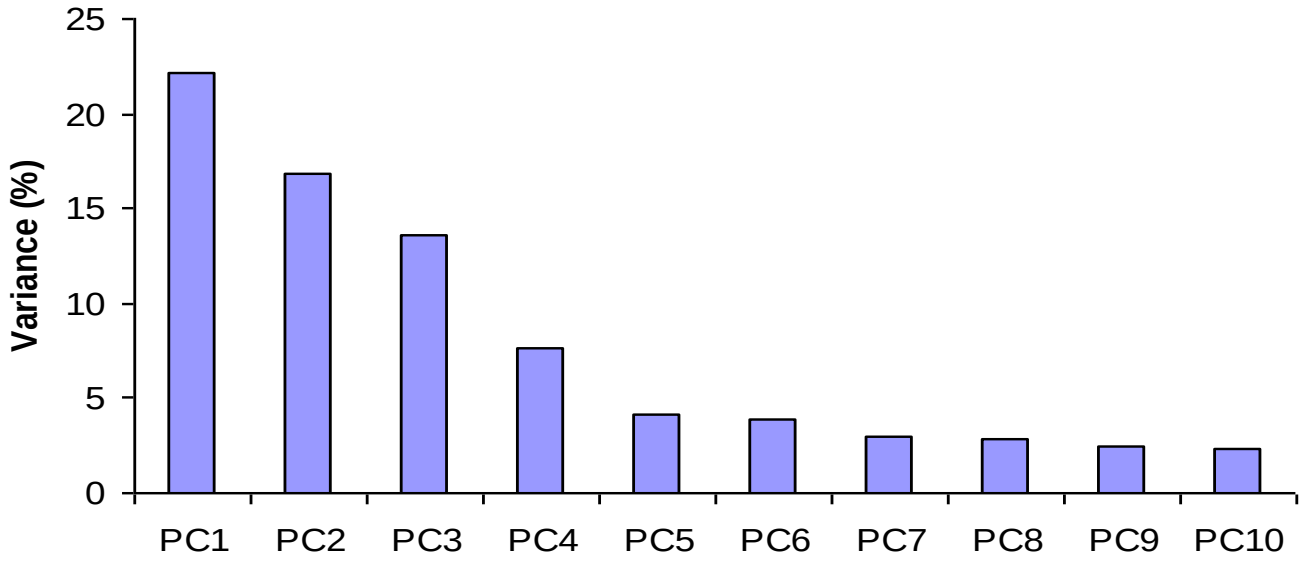
- PCA was invented by Karl Pearson in 1901, however the Singular Value Decomposition was independently derived some half a century earlier in Italy, Germany and France
- PCA transforms the data such that the greatest variance by any projection lies on the first coordinate
- Reveals internal structure of the data that best explains variance in the data set
- Since data often moves in clusters, PCA reveals those variables that drive the variance

Pearson, K. (1901). "On Lines and Planes of Closest Fit to Systems of Points is Space". *Philosophical Magazine Series 6* **2** (11): 559–572

PCA: Eigenvalues

Eigenvalues λ_j are used for calculation of [% of total variance] (V_j) for each component j:

$$V_j = 100 \cdot \frac{\lambda_j}{\sum_{x=1}^{n} \lambda_x} \qquad \sum_{x=1}^{n} \lambda_x = n$$



PCA: Components

- The first PC retains the greatest amount of variation in the sample
- The k^{th} PC retains the k^{th} greatest fraction of the variation in the sample
- The k^{th} largest eigenvalue of the correlation matrix C is the variance in the sample along the k^{th} PC
- The least-squares view: PCs are a series of linear least squares fits to a sample, each orthogonal to all previous ones

PCA Components and Loadings

- Technique useful for compression and classification of data
- Find new descriptors smaller than original variables
- Retain most of sample's information - correlation between original variables
- New descriptors are principal components (PCs)
- Loadings represent the “fraction” of PCs in initial data
- Uncorrelated, and ordered by fraction of total information retained in each PC

PCA in scikit-learn

sklearn.decomposition.PCA

class sklearn.decomposition.**PCA**(*n_components=None, *, copy=True, whiten=False, svd_solver='auto', tol=0.0, iterated_power='auto', n_oversamples=10, power_iteration_normalizer='auto', random_state=None*) [source]

Methods

fit(X[, y])	Fit the model with X.
fit_transform(X[, y])	Fit the model with X and apply the dimensionality reduction on X.
get_covariance()	Compute data covariance with the generative model.
get_feature_names_out([input_features])	Get output feature names for transformation.
get_metadata_routing()	Get metadata routing of this object.
get_params([deep])	Get parameters for this estimator.
get_precision()	Compute data precision matrix with the generative model.
inverse_transform(X)	Transform data back to its original space.
score(X[, y])	Return the average log-likelihood of all samples.
score_samples(X)	Return the log-likelihood of each sample.
set_output(*[, transform])	Set output container.
set_params(**params)	Set the parameters of this estimator.
transform(X)	Apply dimensionality reduction to X.

Examples: Eigenfaces

- When viewed as vectors of pixel values, face images are extremely high dimensional. Image of 100x100 pixels has 10,000 dimensions.
- However, very few of 100x100 vectors are valid face images
- We want to effectively represent the subspace of face images

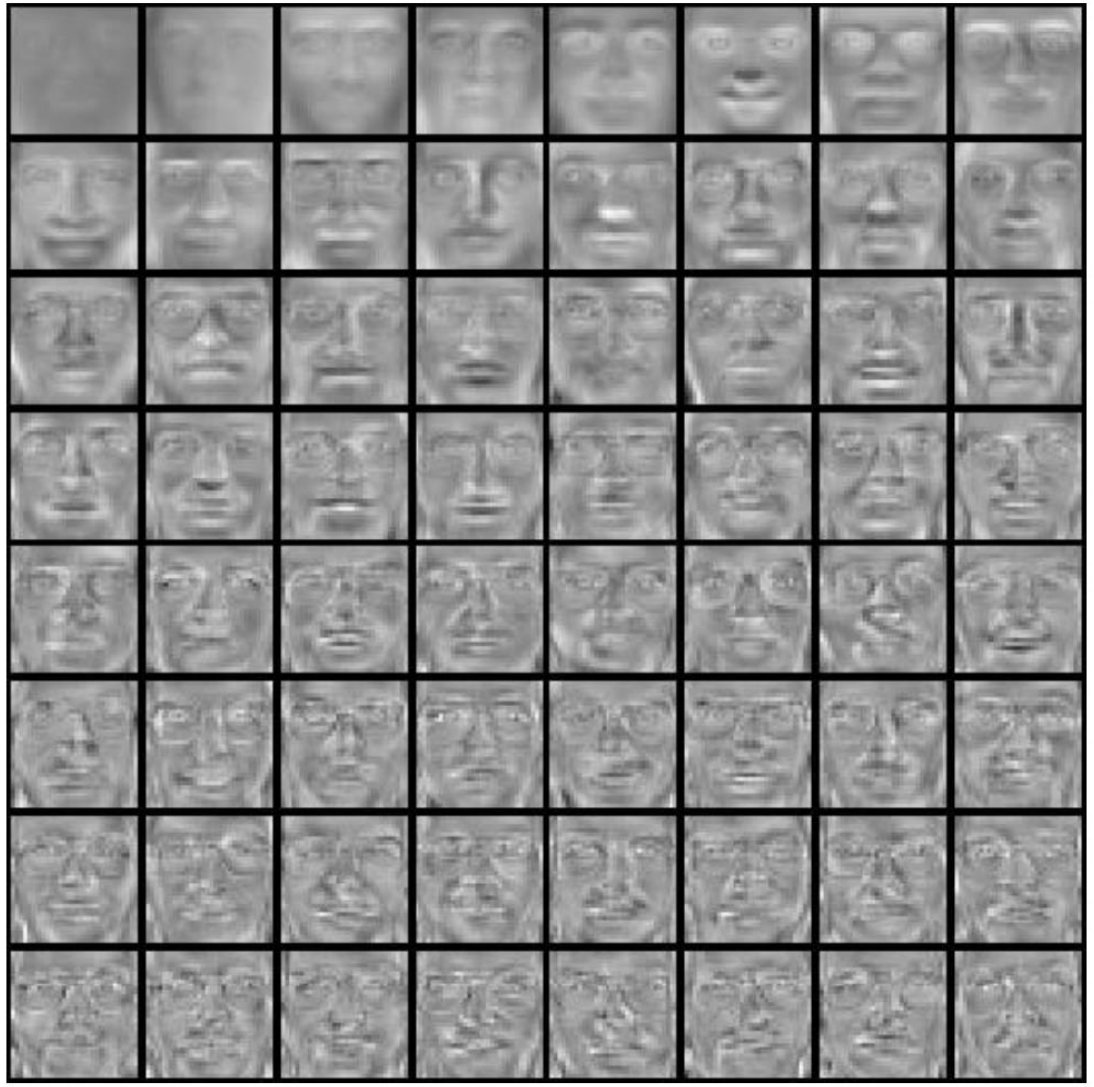
Adapted form Fereshteh Sadeghi

slide by Derek Hoiem

Examples: Eigenfaces

Mean: μ

Top eigenvectors: $u_1,\ldots u_k$



Adapted from Fereshteh Sadeghi slide by Derek Hoiem

Representation and reconstruction

- Face **x** in "face space" coordinates:

$$\mathbf{x} \rightarrow [\mathbf{u}_1^{\mathrm{T}}(\mathbf{x}-\mu), \ldots, \mathbf{u}_k^{\mathrm{T}}(\mathbf{x}-\mu)]$$

$$= \quad w_1, \ldots, w_k$$

- Reconstruction:

$$\hat{x} = \mu + w_1 u_1 + w_2 u_2 + w_3 u_3 + w_4 u_4 + \ldots$$

slide by Derek Hoiem

Adapted from Fereshteh Sadeghi

Reconstruction

P = 4

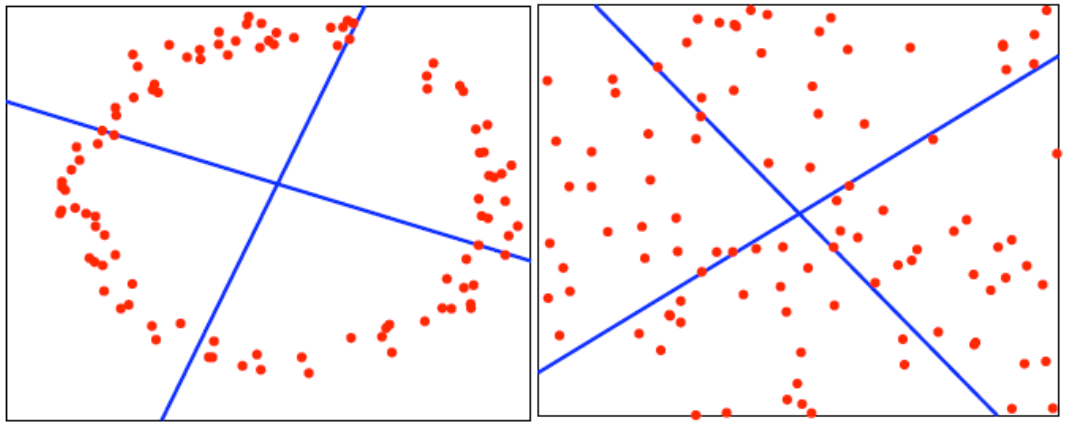
P = 200

P = 400

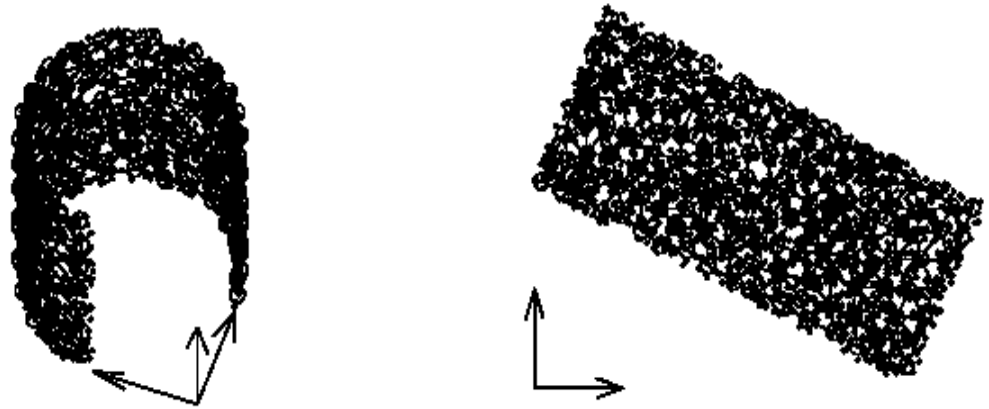
We can represent faces well by 400 components – rather then 10,000!

Adapted from Fereshteh Sadeghi

Limitations of PCA



PCA will make no difference between these examples



Non linear projection of a horseshoe

Non-linear PCA

- Suppose that instead of using the points $\mathbf{x}_i$ as is, we wanted to go to some different *feature space* $\phi(\mathbf{x}_i) \in \Re^N$
- E.g. using polar coordinates instead of cartesian coordinates would help us deal with the circle
- In the higher dimensional space, we can then do PCA
- The result will be non-linear in the original data space!
- Similar idea to support vector machines

Kernel PCA

Kernel PCA is an unsupervised manifold learning technique that maps data points to a generally lower-dimensional space. It generalizes the Principal Components Analysis approach to non-linear transformations using the kernel trick (Schölkopf, Smola and Müller, 1996; Schölkopf, Smola and Müller, 1998; Schölkopf, Burges and Smola, 1999). The algorithm implicitly finds the leading eigenvectors and eigenvalues of the covariance of the projection $\phi(x)$ of the data in "feature space", where $\phi(x)$ is such that the kernel $K_n(x, y) = \phi(x) \cdot \phi(y)$ (i.e. K_n must not have negative eigenvalues). If the data is

(http://research.microsoft.com/users/Cambridge/nicolasl/papers/eigen_dimred.pdf)