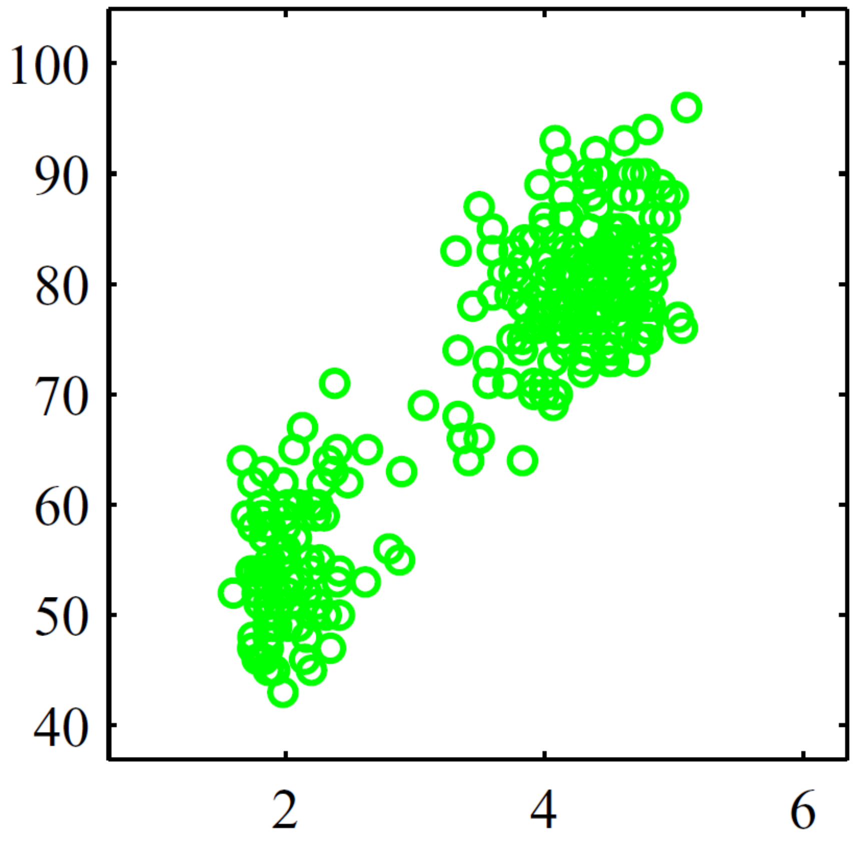
Principal component analysis

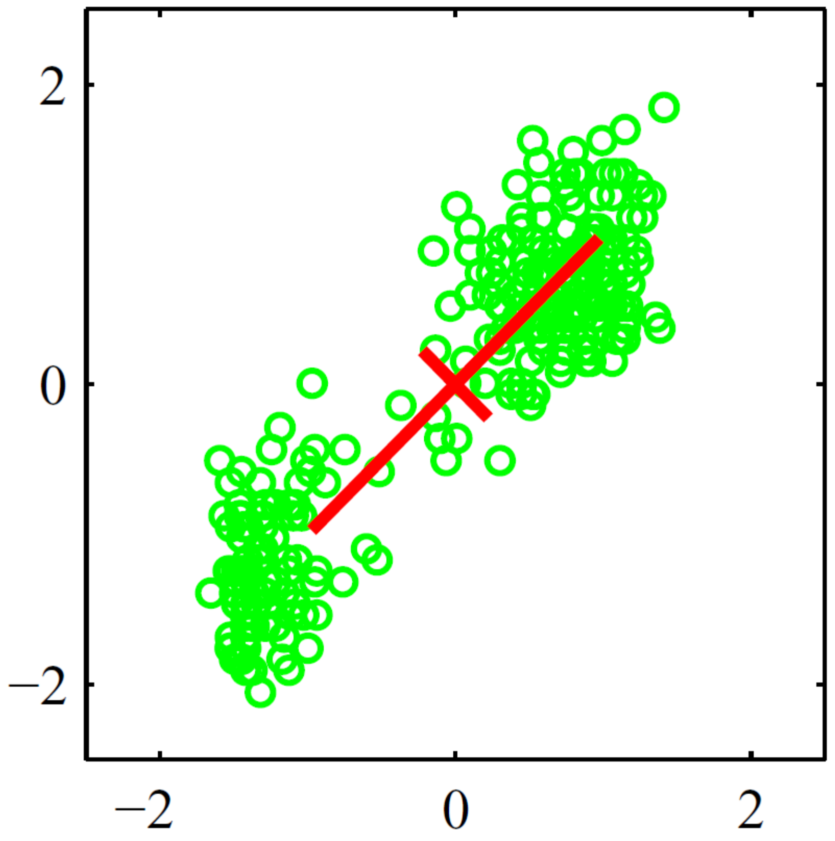
Bram van Ginneken

Feature extraction

- So far we have seen many classifiers that can map input (feature vectors) to output (class label), after a training procedure
- We have seen that the original features may not be optimal for this purpose, e.g.
  - there are too many features
  - the scale of the features is different
  - a nonlinear combination of features may allow a simple linear classifier to be effective

Introductory example





- Two features, clearly correlated, with different ranges

- Same features, scaled to zero mean and unit variance (standardized). They are still correlated. We can get rid of this correlation by projecting on the red lines.

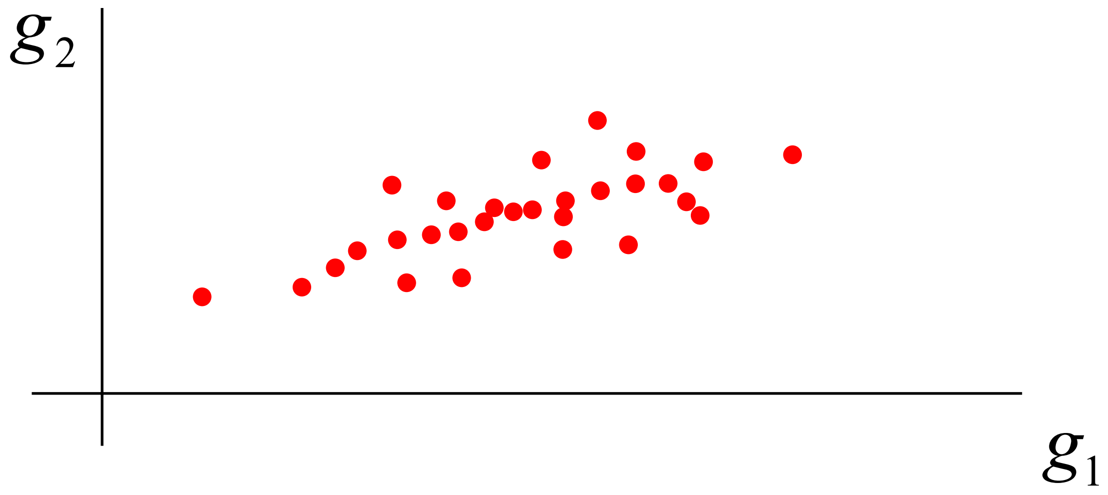
Feature extraction

- Today we consider the situation where we want to *reduce* the number of features so that the smaller number still represents the data accurately
- We will consider *linear combinations* of the original features
- Our definition of accurate representation is that we can reconstruct the large feature vector from the small one with *a small error in the least square sense*
- This leads to principal component analysis

Principal component analysis

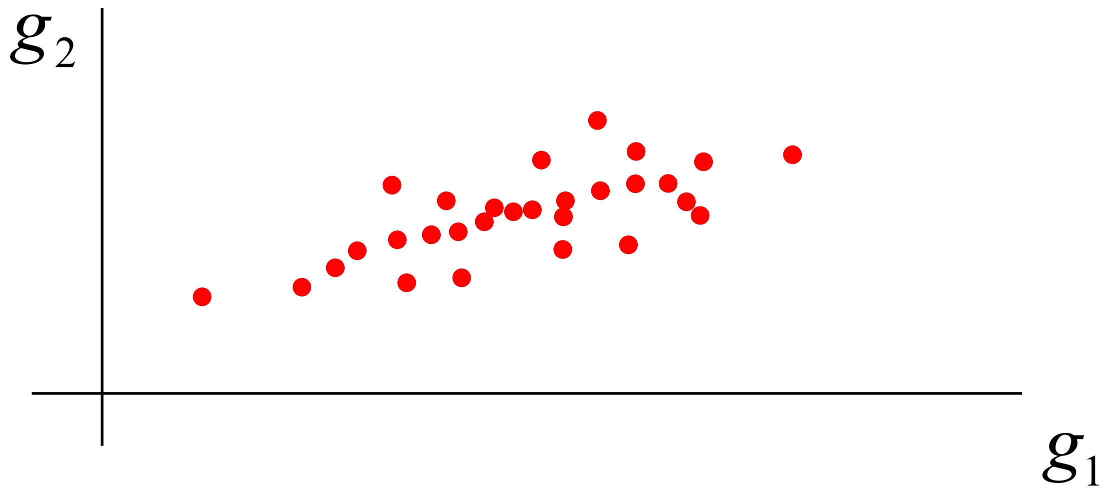
- We have D-dimensional feature vectors $\boldsymbol{x}$
- We transform these to $\boldsymbol{y} = \mathbf{A}\,(\boldsymbol{x} - \boldsymbol{\mu})$ where $\boldsymbol{\mu}$ is the mean of all $\boldsymbol{x}$
- $\mathbf{A}$ is a M x D orthogonal matrix, with M < D so $\boldsymbol{y}$ is an M-dimensional vector
- We have $\boldsymbol{x}_{\text{appr}} = \boldsymbol{\mu} + \mathbf{A}^T\boldsymbol{y}$
- For which $\mathbf{A}$ is $E[||\boldsymbol{x} - \boldsymbol{x}_{\text{appr}}||^2]$ minimal?

Principal component analysis



If we want to represent this 2-D data by 0-D data (a point), which point is the best?

Principal component analysis



$$\text{Minimize } J = \sum_{i=1}^{N} ||\mathbf{x}_i - \mathbf{m}||^2 = \sum_{i=1}^{N} (x_i - m_x)^2 + (y_i - m_y)^2$$

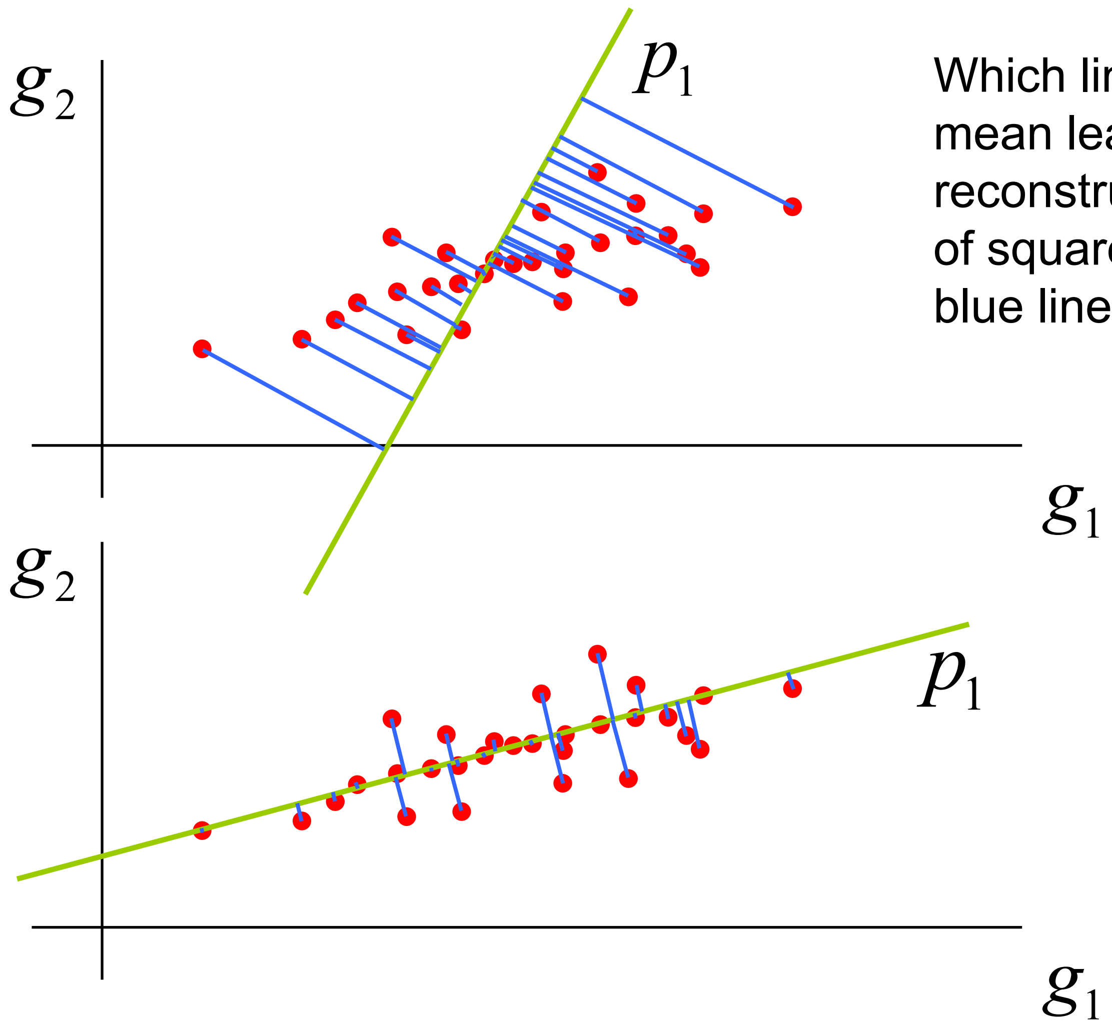
$$\frac{\partial J}{\partial m_x} = -\sum_{i=1}^{N} 2(x_i - m_x) = 0 \rightarrow \sum_{i=1}^{N} x_i = \sum_{i=1}^{N} m_x = N m_x \rightarrow m_x = \frac{1}{N}\sum_{i=1}^{N} x_i$$

and the same for m_y

Principal component analysis

- So the 0-D best representation of a data set, in terms of least square error, is the mean
- A 1-D linear representation (projection on a line) must be a line going through the mean (prove yourself)

Principal component analysis



Which line through the mean leads to smallest reconstruction error (sum of squared lengths of the blue lines)?

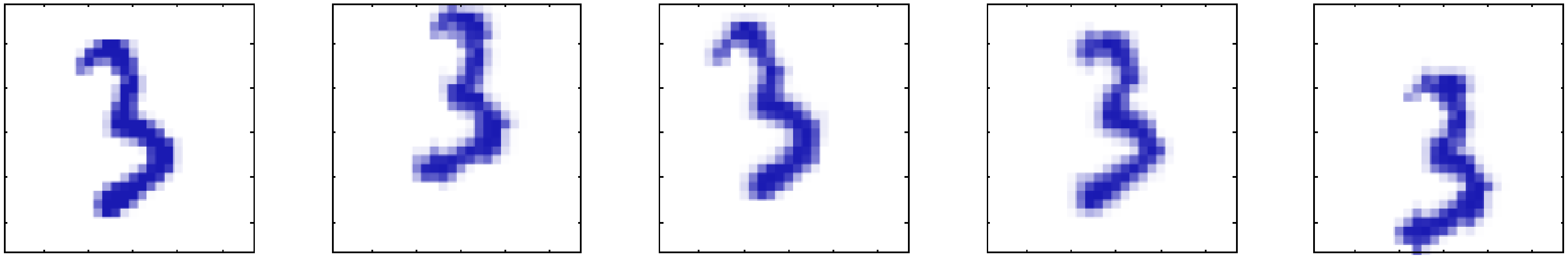
Principal component analysis

- Once the best line p_1 is found, we can encode every 2D point with 1 number, the projection on p and make only a small error
- p_1 is the principal component of the data
- We can look for p_2, perpendicular to p_1, that, combined with p_1 has the smallest error, etc.

Usefulness of PCA

- Many high dimensional signals have a much lower ‘intrinsic’ dimensionality; PCA ‘discovers’ this lower dimensionality

Digits



- 28 x 28 pixel images → dimensionality of signal is 784
- These examples were constructed from translating and rotating one image → intrinsic dimensionality is only 3 → images live on a 3D manifold (curved space) in a 728D space
- Approximating this with a linear subspace will give only a few components → enormous data reduction
- In general, images of the digit 3 look all similar, their intrinsic dimensionality is probably much smaller than 728. In other words, not every 28 x 28 image looks like a 3, only a very small subset does

International Journal of Computer Vision, 8:2, 99–111 (1992)
© 1992 Kluwer Academic Publishers, Manufactured in The Netherlands.

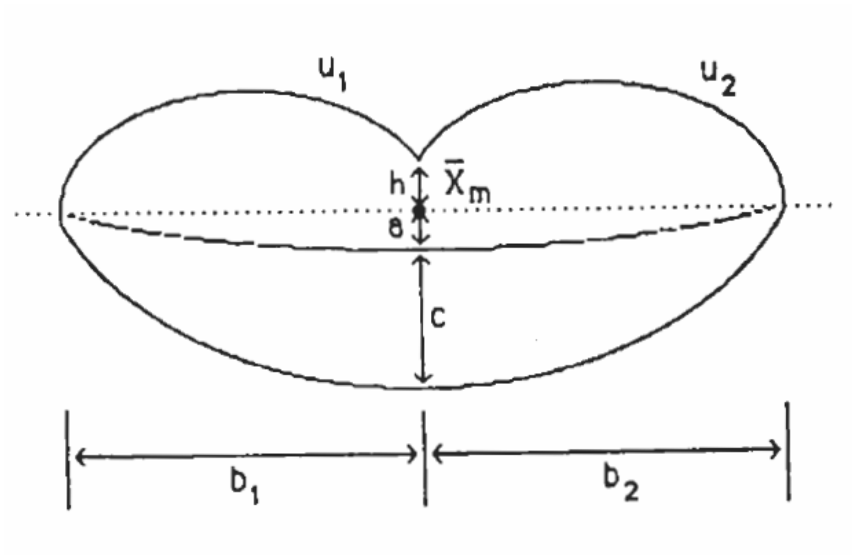
Feature Extraction from Faces Using Deformable Templates

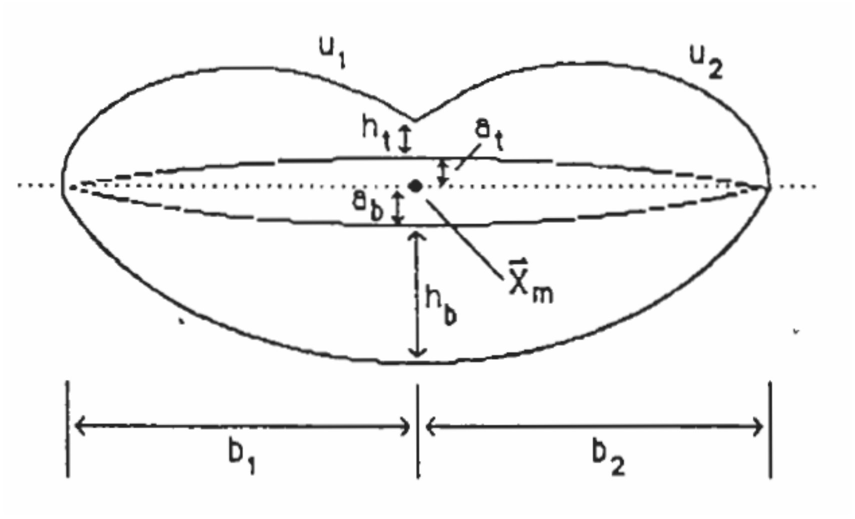
ALAN L. YUILLE, PETER W. HALLINAN, AND DAVID S. COHEN
Division of Applied Sciences, Harvard University, 29 Oxford St., Cambridge, MA 02138

Received February 11, 1992.

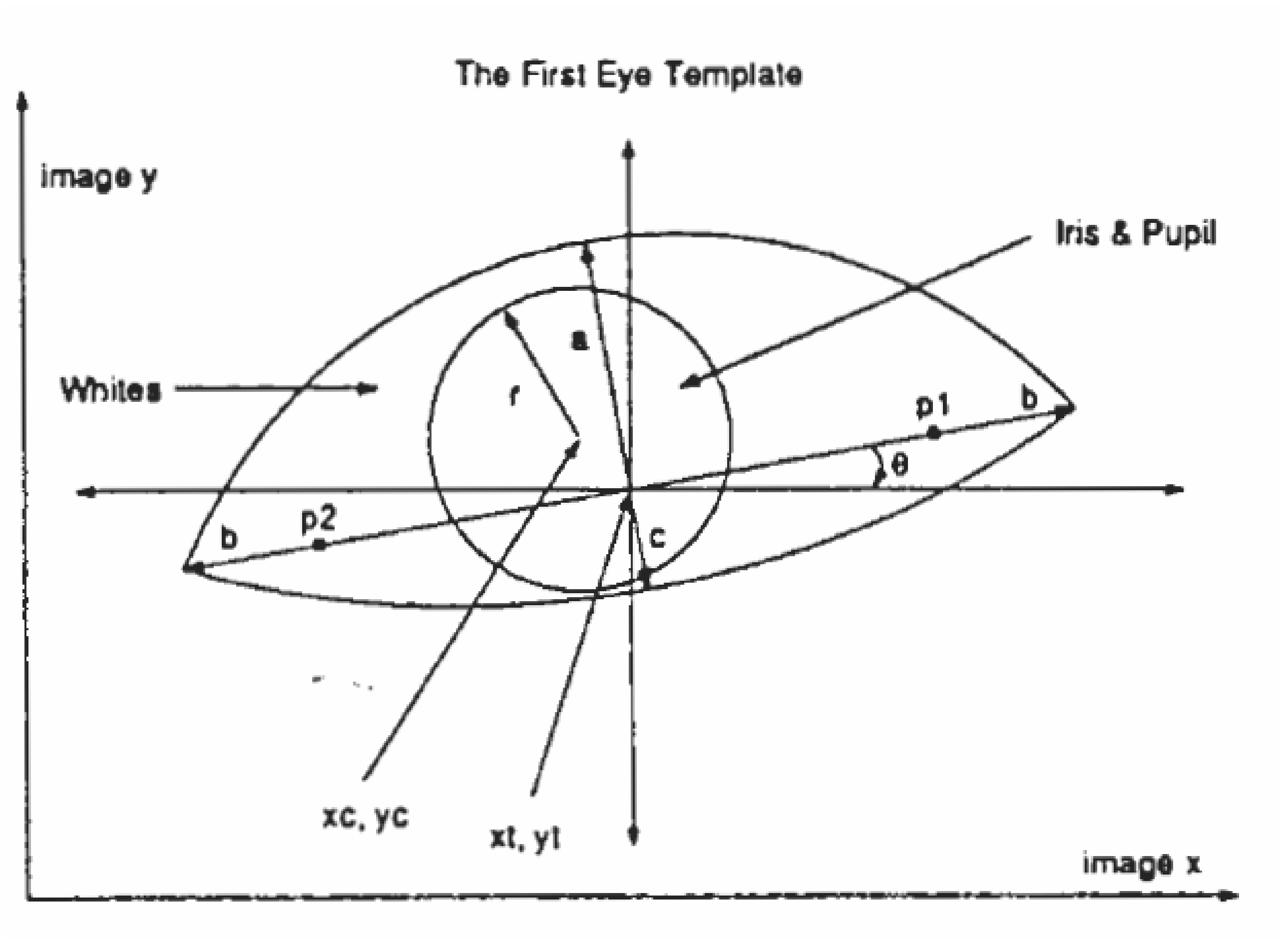
Abstract
We propose a method for detecting and describing features of faces using deformable templates. The feature of interest, an eye for example, is described by a parameterized template. An energy function is defined which links edges, peaks, and valleys in the image intensity to corresponding properties of the template. The template then interacts dynamically with the image by altering its parameter values to minimize the energy function, thereby deforming itself to find the best fit. The final parameter values can be used as descriptors for the feature. We illustrate this method by showing deformable templates detecting eyes and mouths in real images. We demonstrate their ability for tracking features.

Mouth templates

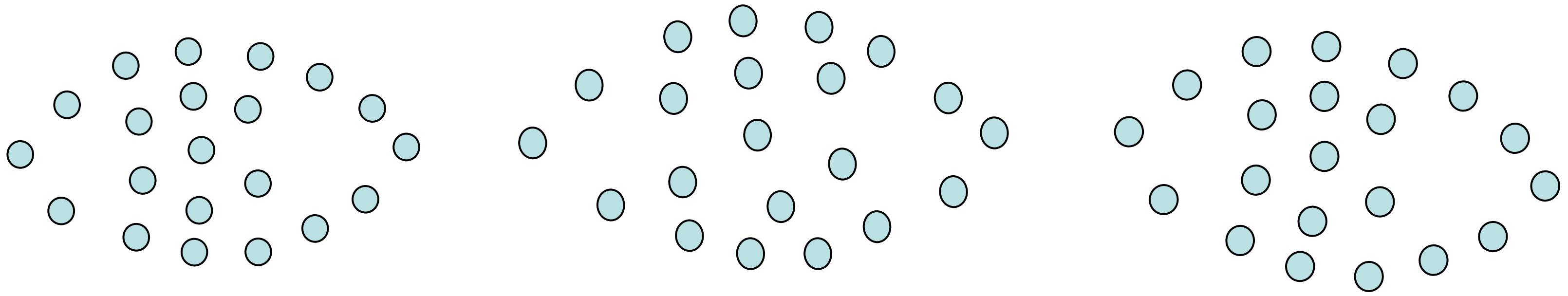




Eye template

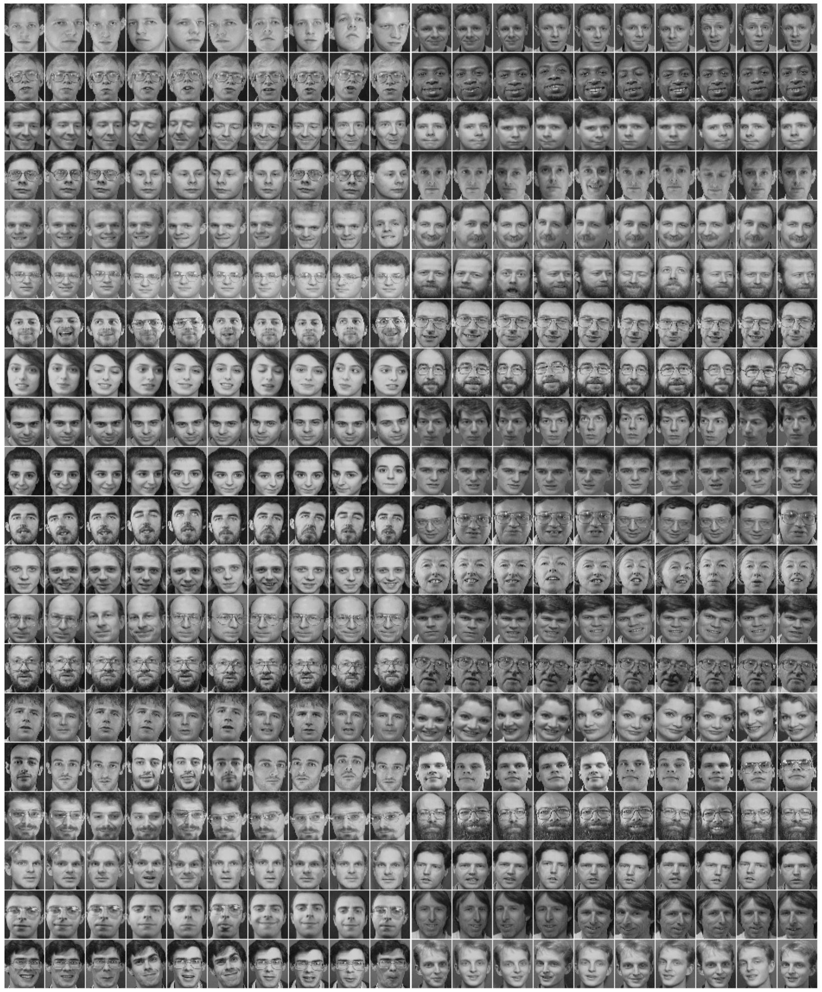


An alternative eye model



Specify the eye by 21 points (or more points if more detail is needed). This ‘model’ contains 42 parameters. That’s too much: not every configuration of 21 points is a ‘valid’ eye. How to make a more compact model?

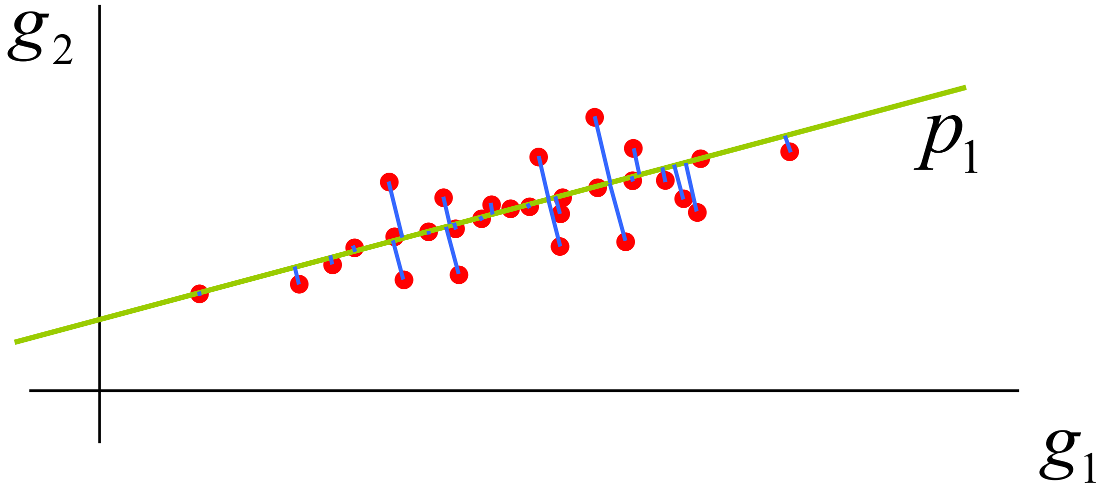
Another example



Images of a fixed dimension, e.g. containing 15000 pixels, all representing an image. This ‘model’ contains 15000 parameters. That’s too much: not every image of this dimension constitutes a face. How to make a more compact model?

Principal component analysis

- How to compute the subspace?
- Minimizing the squared projection error is equivalent to maximizing the variance along the retained directions



Principal component analysis

- The mean of the data projected on *p_1* is

$$\bar{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i$$

- The variance of the projected data is

$$\frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{p}_1'\mathbf{x}_i - \mathbf{p}_1'\bar{\mathbf{x}}\right)^2 = \frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{p}_1'(\mathbf{x}_i - \bar{\mathbf{x}})\right)^2$$

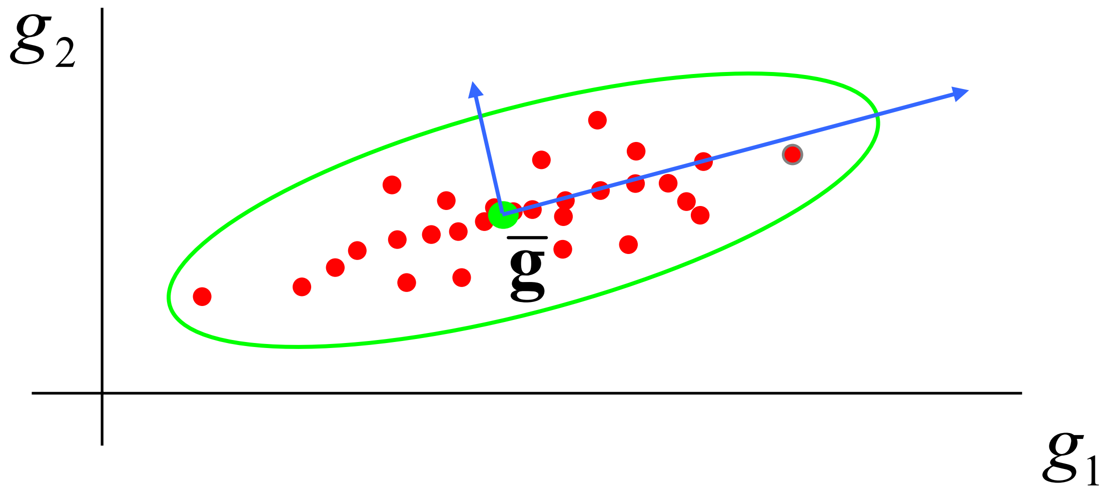
- This is equal to

$$\mathbf{p}_1'\mathbf{S}\mathbf{p}_1 \text{ where } \mathbf{S} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})'$$

- So all we need is the mean and covariance!

Principal component analysis

- Interestingly, the mean and the covariance specify a Gaussian model
- So PCA seems to be related to fitting a Gaussian model to the data



Fitting a Gaussian model

- The mean and the covariance completely determine the Gaussian:

$$\bar{\mathbf{x}} = \tfrac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$$

$$\mathbf{S} = \tfrac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T$$

$$G(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\mathbf{S}|^{-1/2}}\exp[-\tfrac{1}{2}(\mathbf{x}-\bar{\mathbf{x}})^T\mathbf{S}^{-1}(\mathbf{x}-\bar{\mathbf{x}})]$$

Covariance matrix

- Is symmetric
- Has positive values on the diagonal
- Is a diagonal matrix if variables are statistically independent
- Is positive semidefinite: $\mathbf{e}^T\mathbf{S}\mathbf{e} > 0$
- $\mathbf{e}^T\mathbf{S}\mathbf{e}$ is the variance in the direction $\mathbf{e}$
- Has only real and positive eigenvalues

Eigenvector Decomposition

- If **A** is a square matrix then an eigenvector of **A** is a vector, **p**, such that

$$\mathbf{A}\mathbf{p} = \lambda\mathbf{p}$$

$$\lambda \text{ is the eigenvalue associated with } \mathbf{p}$$

- Usually **p** is scaled to have unit length

Fitting a Gaussian model

- You can always rotate the coordinate system so that the covariance matrix is diagonal

$$\mathbf{S} = \mathbf{P}\mathbf{D}\mathbf{P}^T \qquad \mathbf{P}^T\mathbf{P} = \mathbf{P}\mathbf{P}^T = \mathbf{I}$$

$$\mathbf{S}^{-1} = \mathbf{P}\mathbf{D}^{-1}\mathbf{P}^T$$

$$\mathbf{D}^{-1} = \begin{pmatrix} \lambda_1^{-1} & 0 & \cdots & 0 \\ 0 & \lambda_2^{-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n^{-1} \end{pmatrix}$$

Fitting a Gaussian model

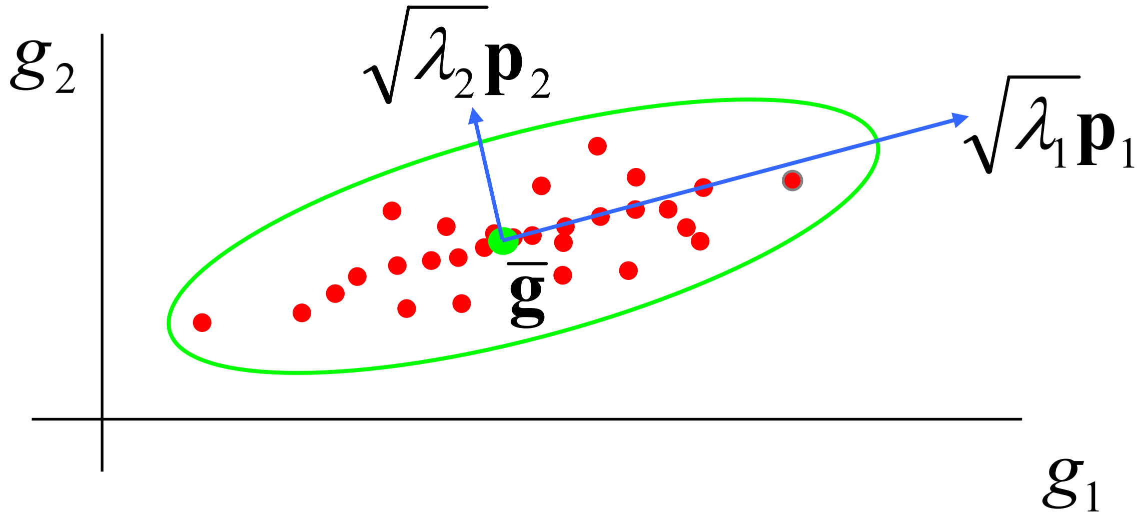
- For a diagonal covariance matrix **S**, the principal directions are (1,0,…), (0,1,0,…) and so on.
- These are eigenvectors of **S**

Fitting a Gaussian model

- Rotating the coordinate system only rotates the eigenvectors
- So the eigenvectors of the covariance matrix are, in general, the principal directions

Principal Component Analysis

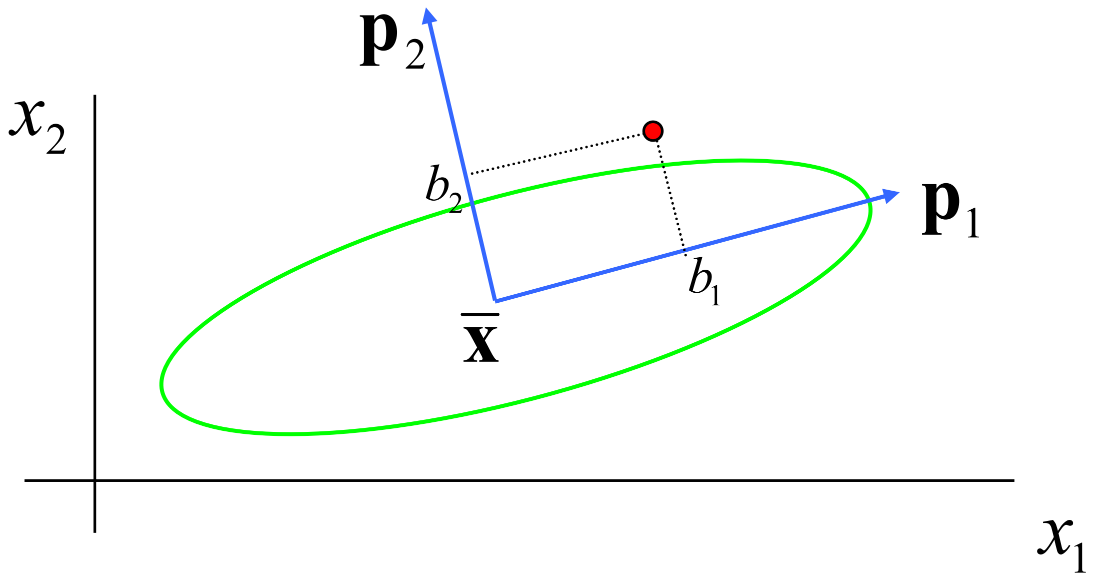
- Compute eigenvectors of covariance, **S**
- Eigenvectors : main directions
- Eigenvalue : variance along eigenvector


g_2
$\sqrt{\lambda_2}\mathbf{p}_2$
$\sqrt{\lambda_1}\mathbf{p}_1$
$\overline{\mathbf{g}}$
g_1

Eigenvector Decomposition

- Consider the transformation

$$\mathbf{b} = \mathbf{P}^T(\mathbf{x} - \overline{\mathbf{x}})$$



Principal component analysis

- Fit a Gaussian model to the data
- Find eigenvectors/values of covariance matrix
- Transform coordinate system by translation and rotation:
  - Mean becomes origin
  - Eigenvector with largest eigenvalue becomes 1st axis
  - Eigenvector with next largest eigenvalue becomes 2nd axis
  - And so on

Principal component analysis

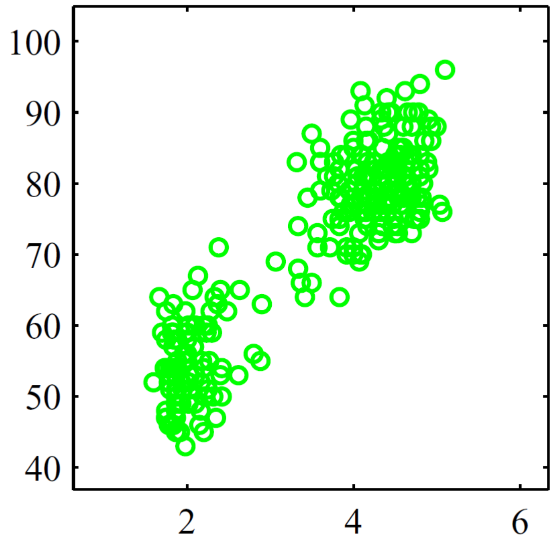
- Discarding eigenvectors with smaller eigenvalues gives a projection on the principal components
- Best linear approximation to the data in least square sense

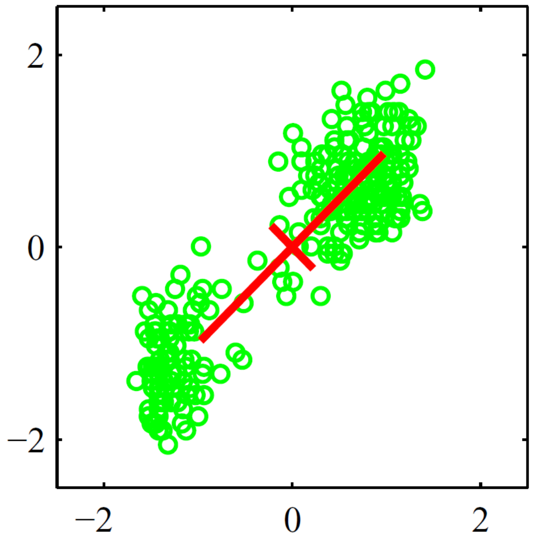
Demos

PRML12.1.4

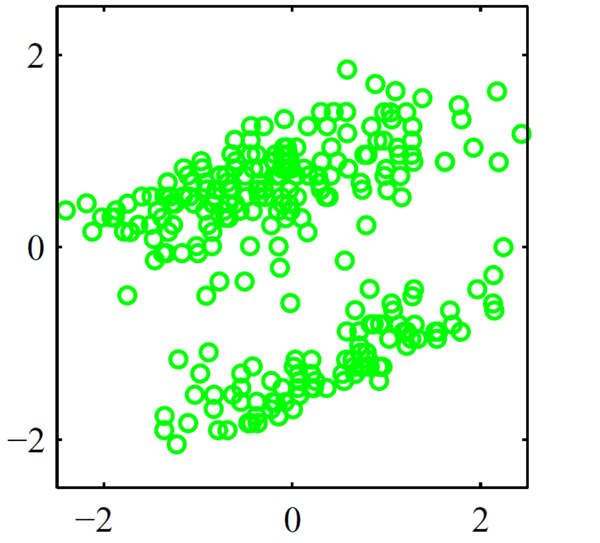
- Store zero-mean data in data matrix **X**, N rows (points), D columns (features)
- Normally: PCA from eigensystem **S** = **X'X**
- When N < D, it is faster to get eigensystem of **K** = **XX'**, these eigenvectors are also eigenvectors of **S**
- Akin to kernel methods

Whitening



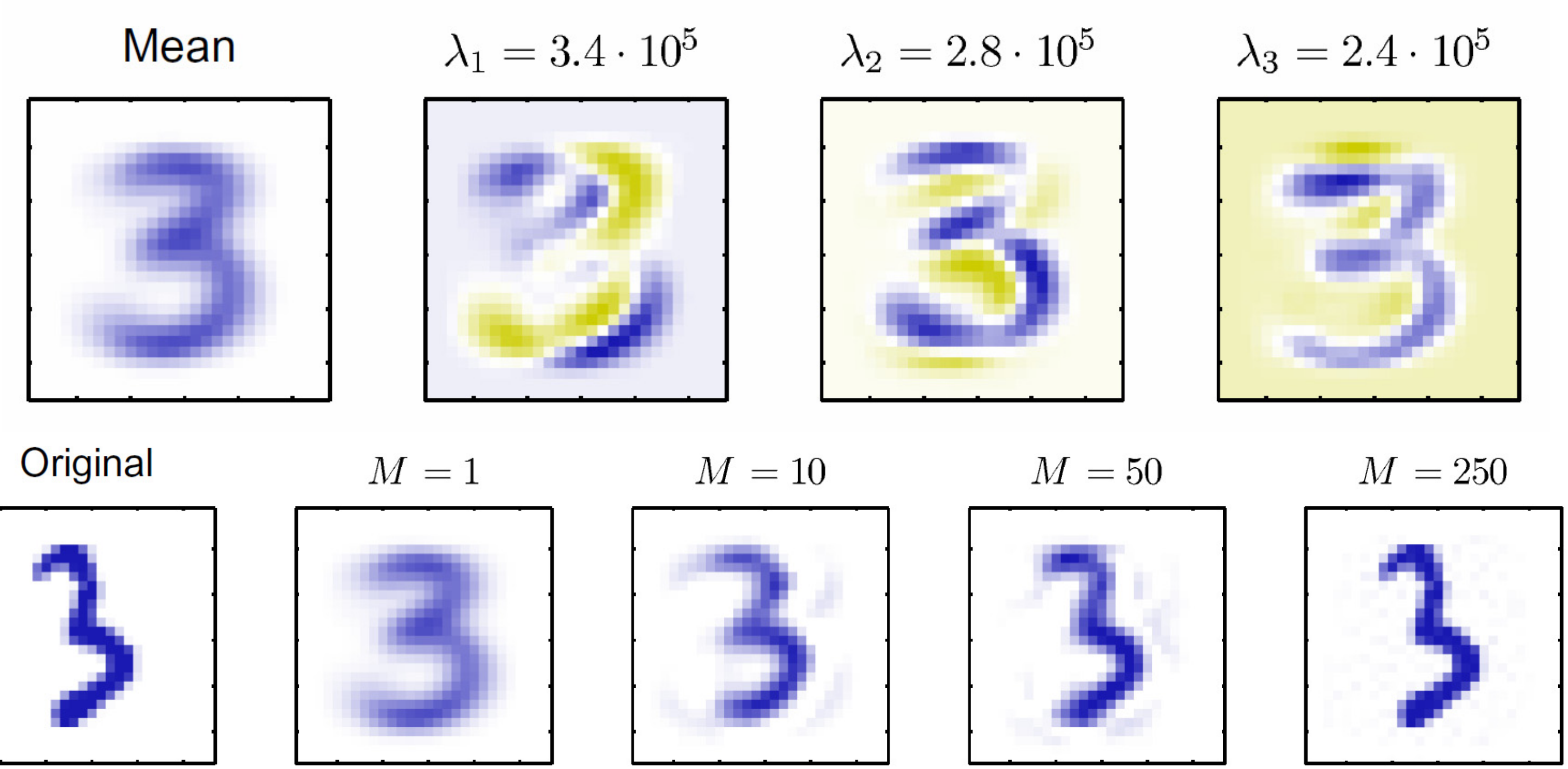


After standardizing, each feature has zero mean and unit variance, but features are still correlated



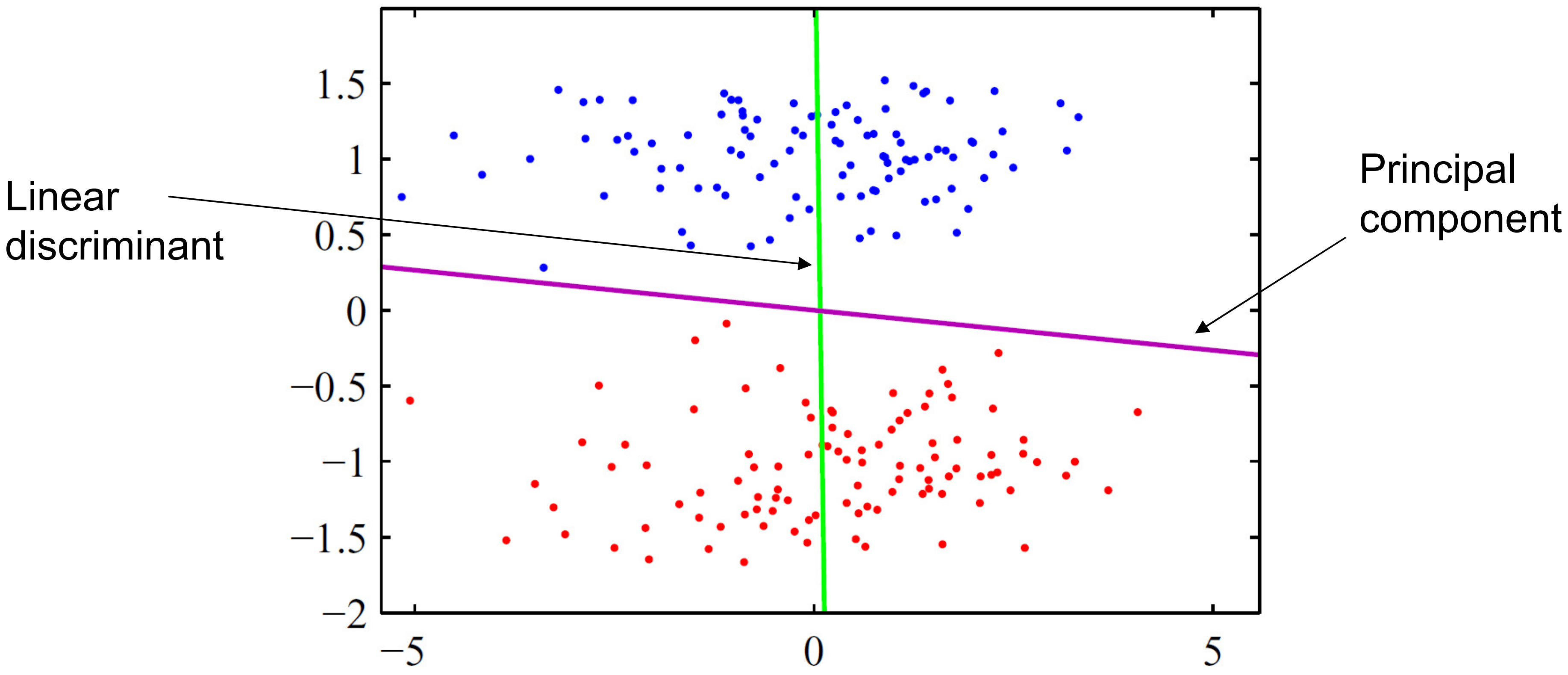
After whitening, each feature has zero mean, unit variance and is decorrelated with other features (covariance is a unit matrix)

PCA can reduce dimensionality

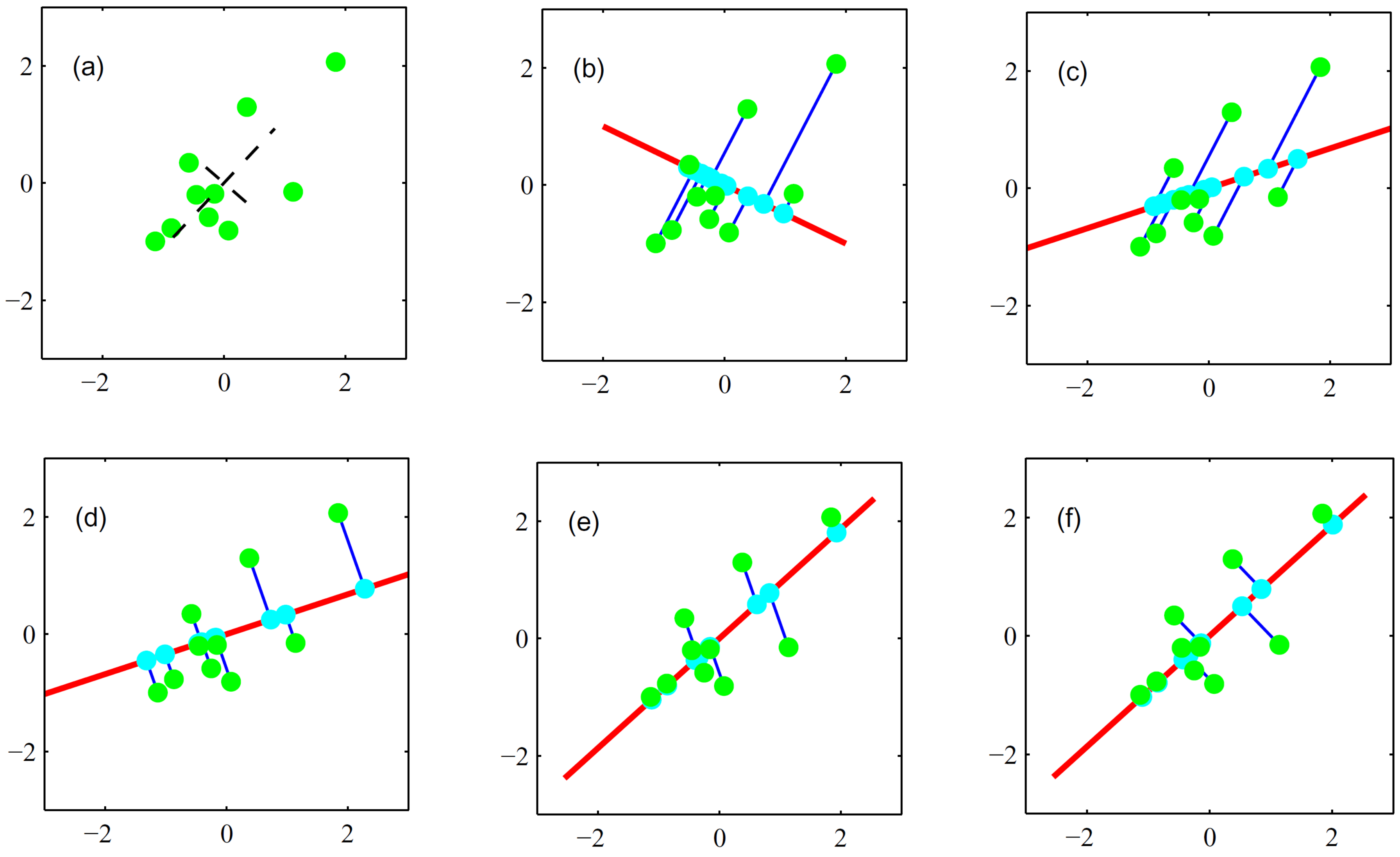

Mean
$\lambda_1 = 3.4 \cdot 10^5$
$\lambda_2 = 2.8 \cdot 10^5$
$\lambda_3 = 2.4 \cdot 10^5$
Original
$M = 1$
$M = 10$
$M = 50$
$M = 250$

Note that this model is built with only 3's. Reconstruction for all digits would not work so well. You could classify digits by projecting them on individual digit models and take the model that reconstructs best with a small set of components only (e.g. 20). Or build one model and use principal components as features instead of the 728 dimensions. But is one PCA model for ten digits a good idea?

PCA does not use class labels!

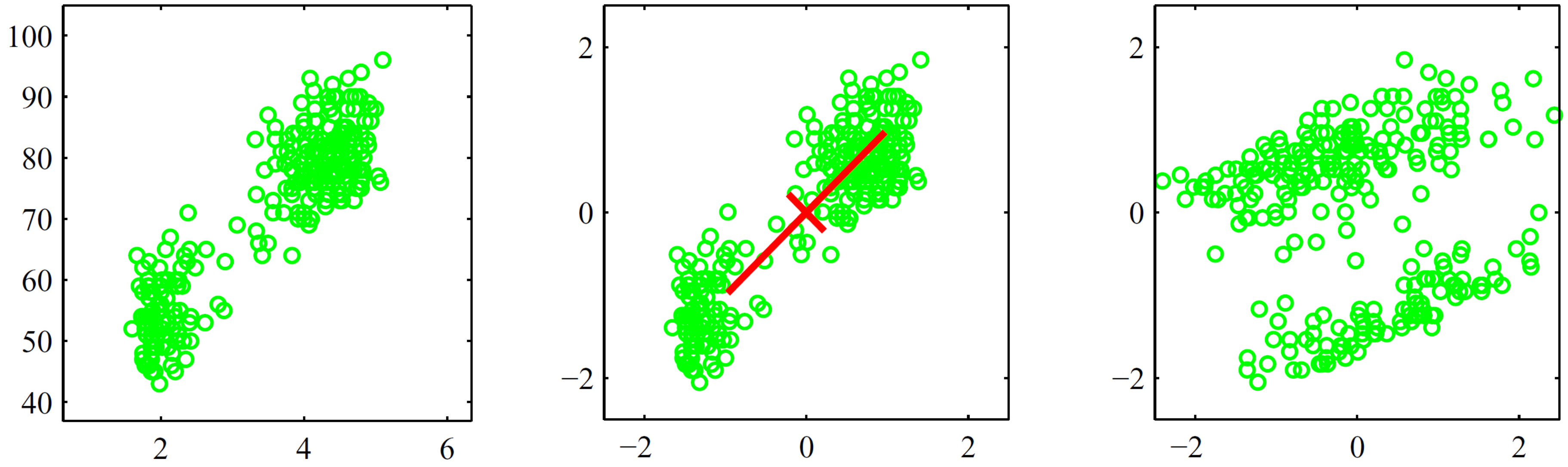


The direction(s) of maximum variance may not be the directions that separate classes best. Be careful when using PCA for classification. Linear discriminant analysis finds only one direction, other techniques find multiple directions that separate data best, e.g. M. Loog, R.P.W. Duin, "Linear Dimensionality Reduction via a Heteroscedastic Extension of LDA : the Chernoff Criterion", IEEE Transactions on Pattern Analysis and Machine Intelligence, 2004, vol. 26, pp. 723-739.



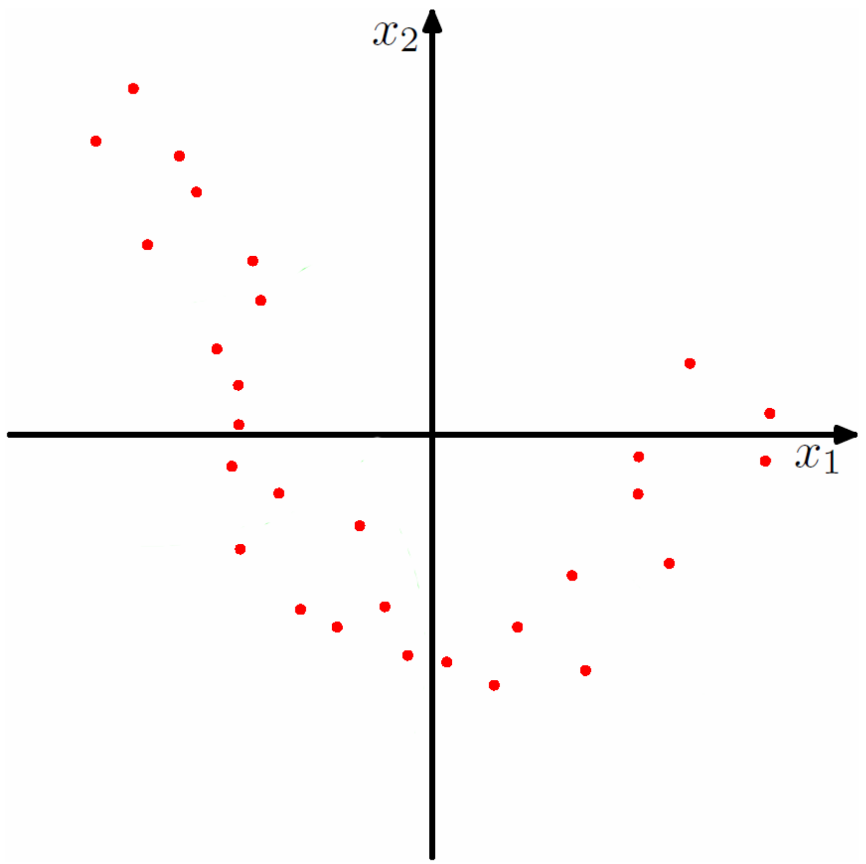
- EM algorithm for PCA; can be faster in certain situations and predict missing values. Variations of this algorithm can be used for non-linear PCA

Independent Component Analysis



• After whitening, features are uncorrelated but not independent
• If the signal consists of independent components these can be recovered with *independent component analysis*
• ICA finds the most non-Gaussian directions in feature space (but *not* the directions of maximum variance)
• See e.g. Hyvarinen, A. & Oja, E., “Independent component analysis: algorithms and applications”, *Neural Networks,* **2000***, 13*, 411-430

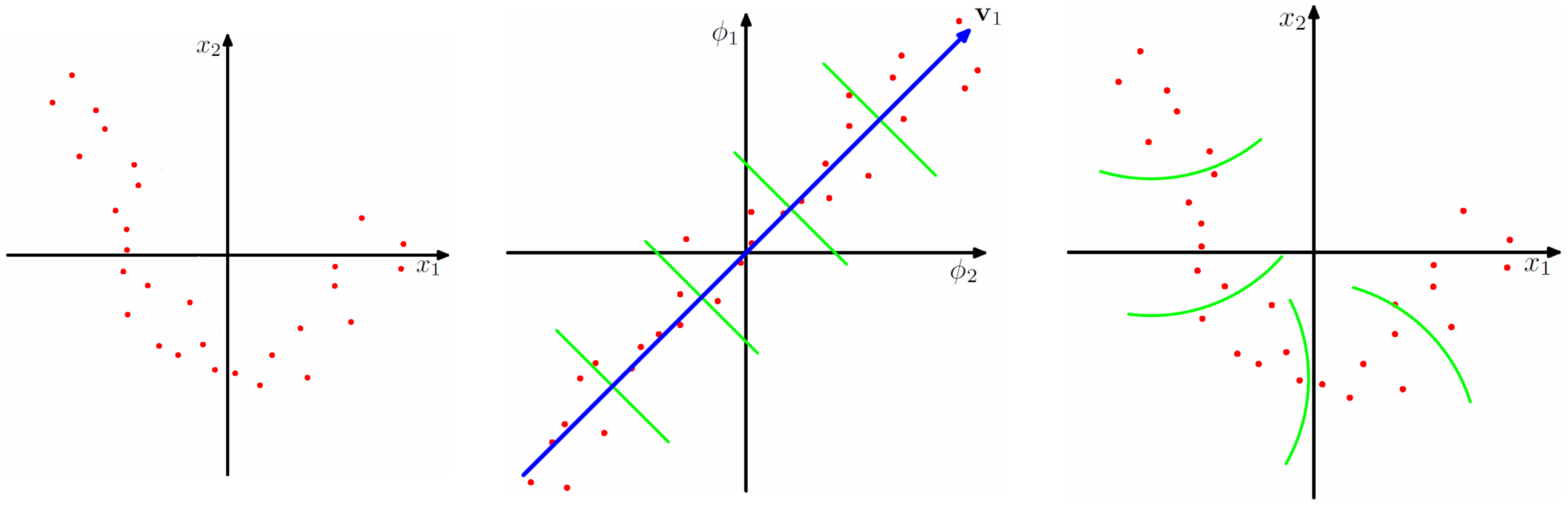
Kernel PCA



- If data is not normally distributed, it could be e.g. much more efficient to project the data on a curve
- Finding this curve is much harder than the linear case

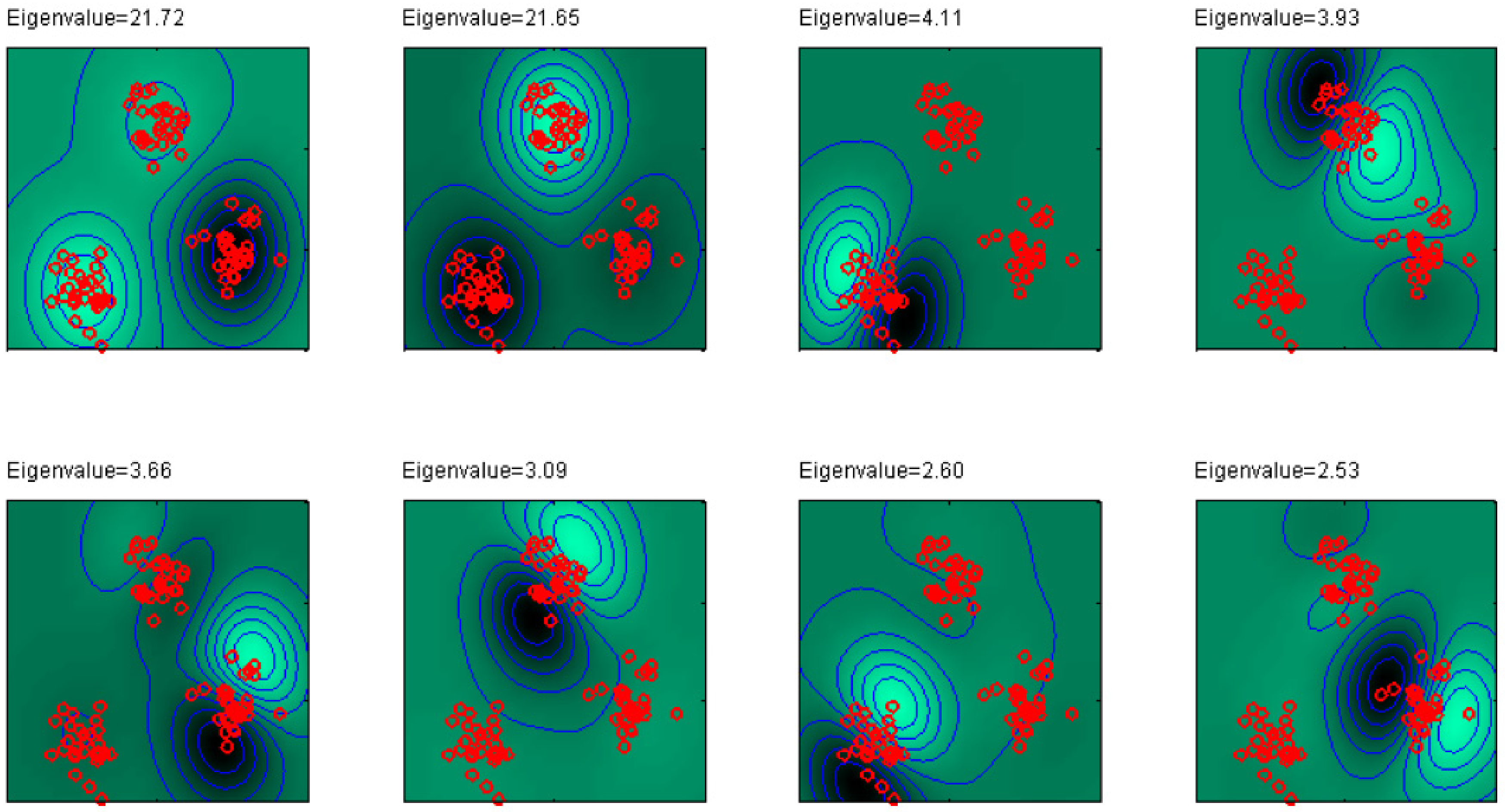
Kernels to the rescue

- Map data to higher dimensional space
- Do standard PCA there
- Corresponds to non-linear PCA in lower dimensional space



Kernels to the rescue

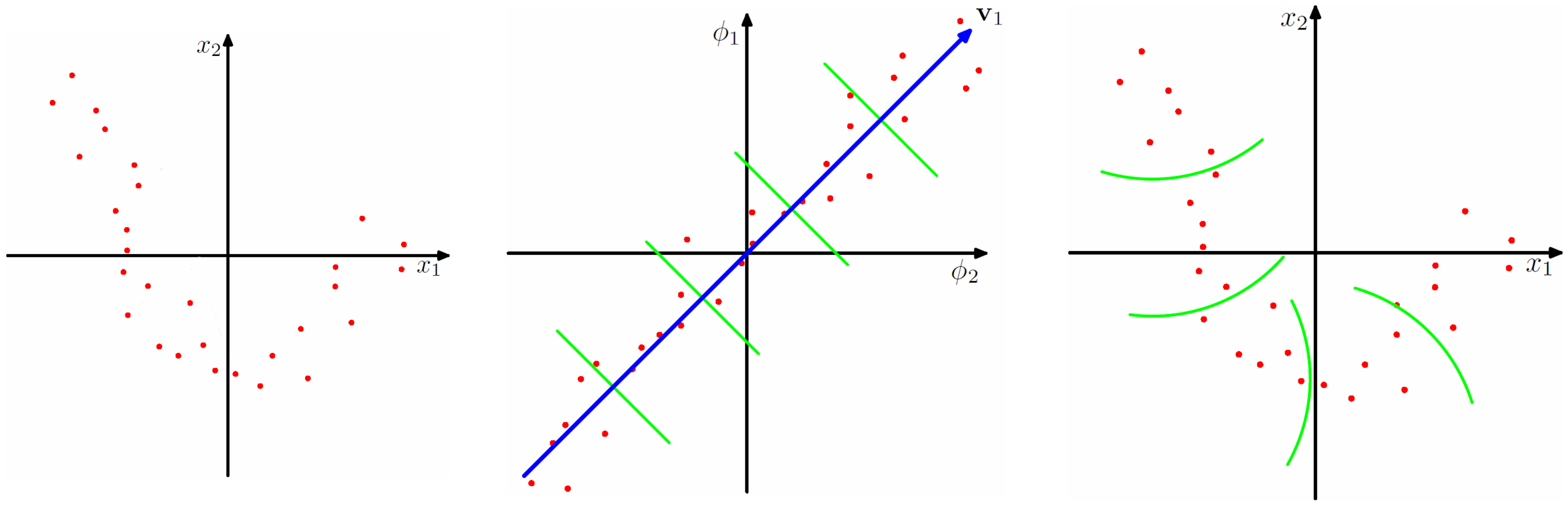
- The PCA procedure can be written in terms of inner products only
- Need to get eigensystem of an N x N matrix (instead of D x D) → problematic for large datasets
- Any kernel can be used



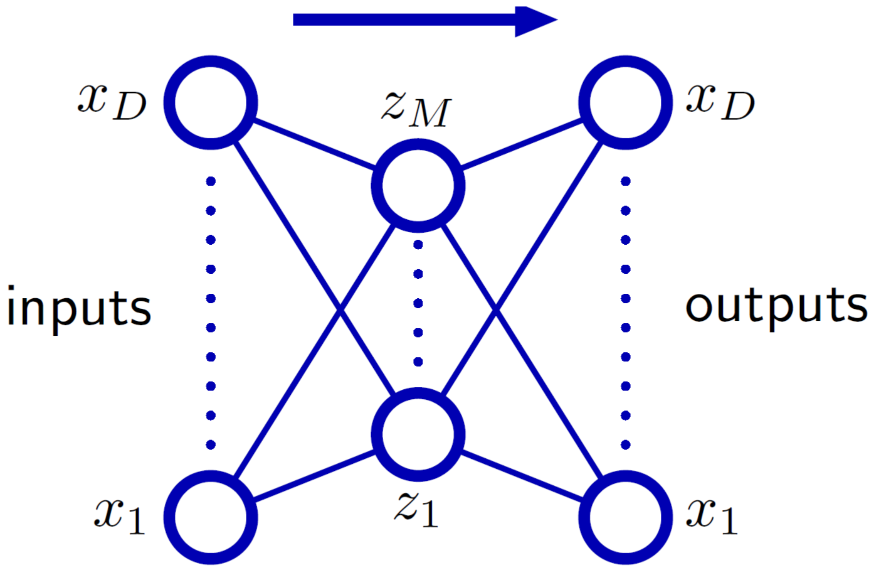
KernelPCA with Gaussian kernel. Data is clustered. First two components encode cluster. Higher components encode structure within clusters

Limitation of kernel PCA

- Curves with the same projections can be visualized in the original space, because they are the intersection of the manifold in the high dimensional space and the plane perpendicular to a point on the eigenvector
- The eigenvectors themselves are not in the manifold and cannot be visualized
- The projected data (points on blue line) is not known, only approximations can be computed (pre-images)

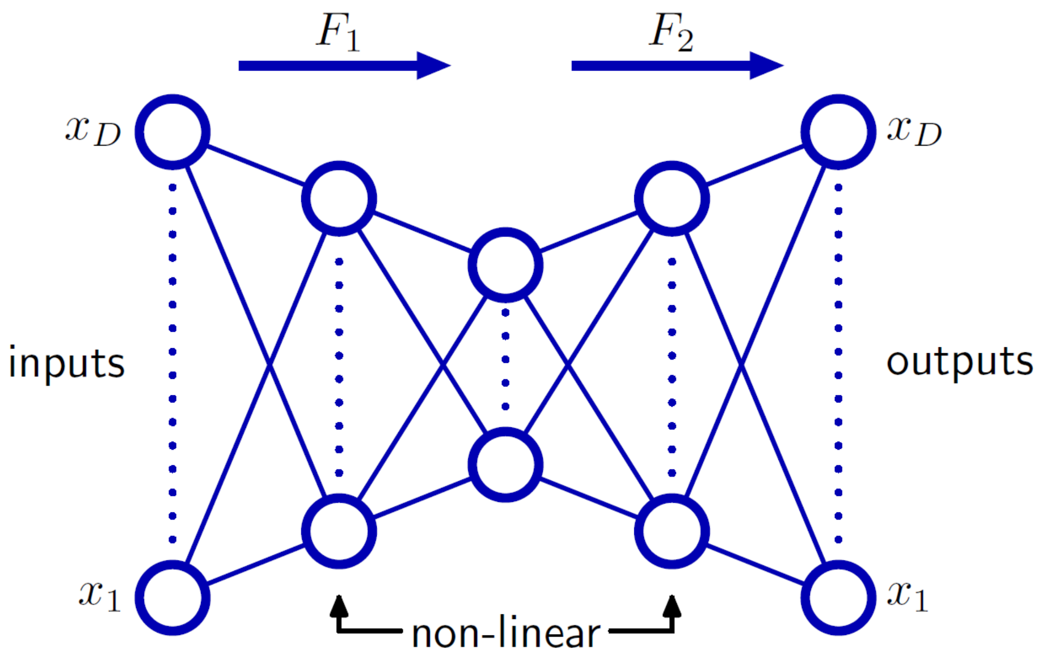


Autoassociative neural networks



- Such a network, trained with output-input and sum of squares error function finds principal components (not necessarily orthogonal)

Autoassociative neural networks



- Extra layers allow the network to do non-linear dimensionality reduction
- See Hinton & Salakhutdinov, “Reducing the Dimensionality of Data with Neural Networks”, *Science,* **2006**, *313*, 504-507 (They call them Restricted Boltzmann Machines, also used in collaborative filtering, e.g. Netflix)

From Hinton & Salakhutdinov

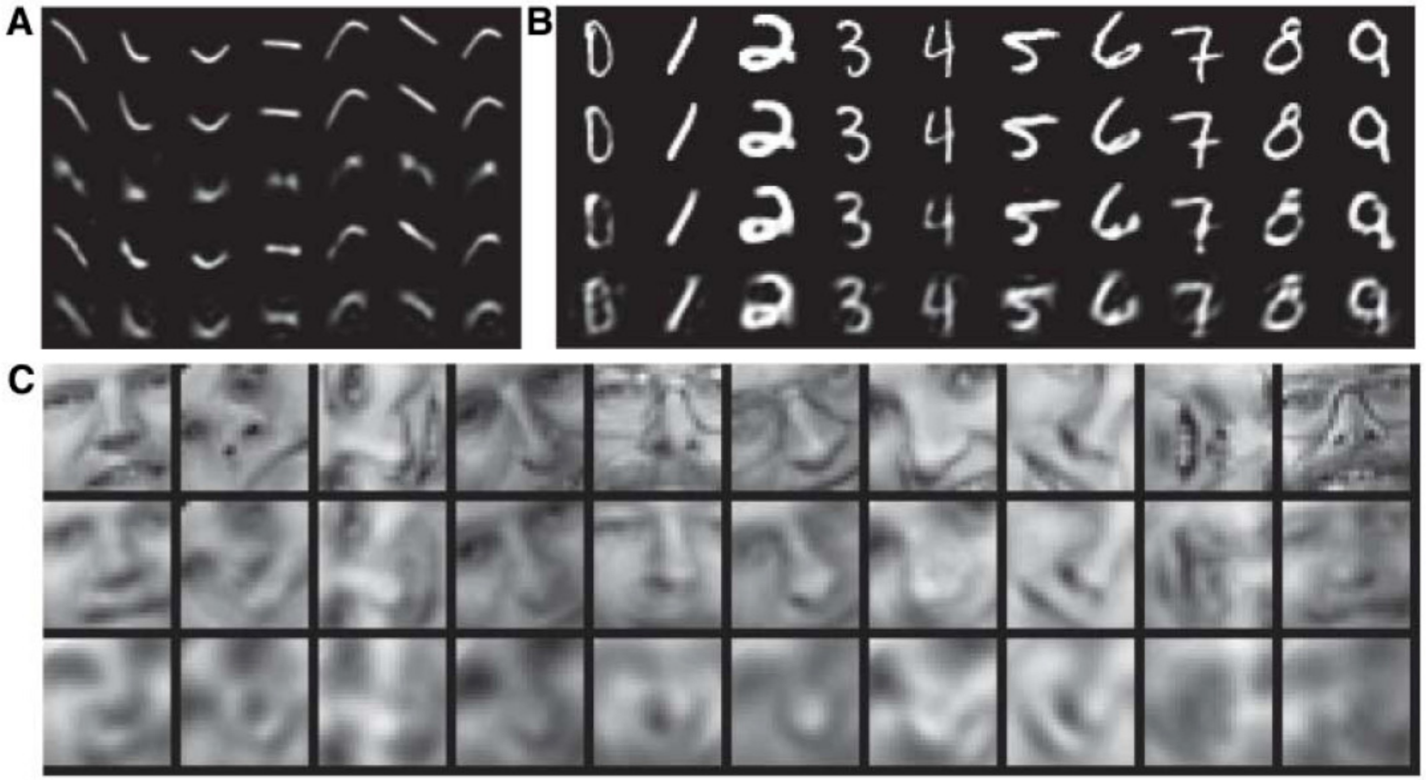


Fig. 2. (**A**) Top to bottom: Random samples of curves from the test data set; reconstructions produced by the six-dimensional deep autoencoder; reconstructions by "logistic PCA" (*8*) using six components; reconstructions by logistic PCA and standard PCA using 18 components. The average squared error per image for the last four rows is 1.44, 7.64, 2.45, 5.90. (**B**) Top to bottom: A random test image from each class; reconstructions by the 30-dimensional autoencoder; reconstructions by 30-dimensional logistic PCA and standard PCA. The average squared errors for the last three rows are 3.00, 8.01, and 13.87. (**C**) Top to bottom: Random samples from the test data set; reconstructions by the 30-dimensional autoencoder; reconstructions by 30-dimensional PCA. The average squared errors are 126 and 135.

Summary

- PCA is a powerful and simple technique to reduce the dimensionality of data
- Applications:
  - Compact generative models
  - Feature extraction
  - Many more…
- Nonlinear extensions are much more complex but may be more powerful