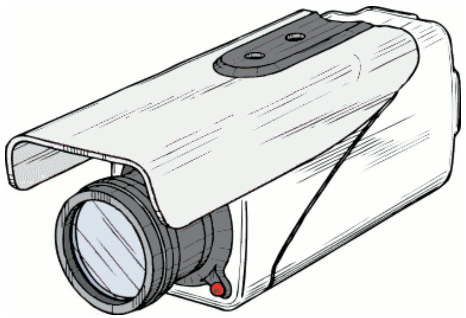
Univerza v Ljubljani

Machine perception
Recognition 1

Matej Kristan



Laboratorij za Umetne Vizualne Spoznavne Sisteme,
Fakulteta za računalništvo in informatiko,
Univerza v Ljubljani

Recognition

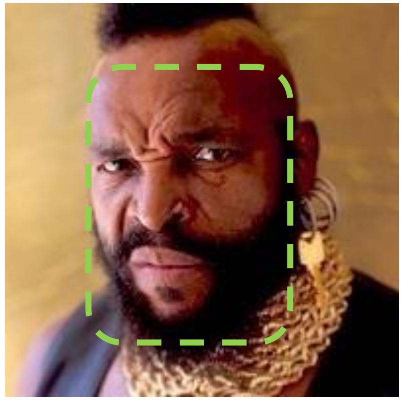
- Assume we have tagged an object with a bounding box

Whose face is it?

Browse your facebook friends....

Marilyn Manson?

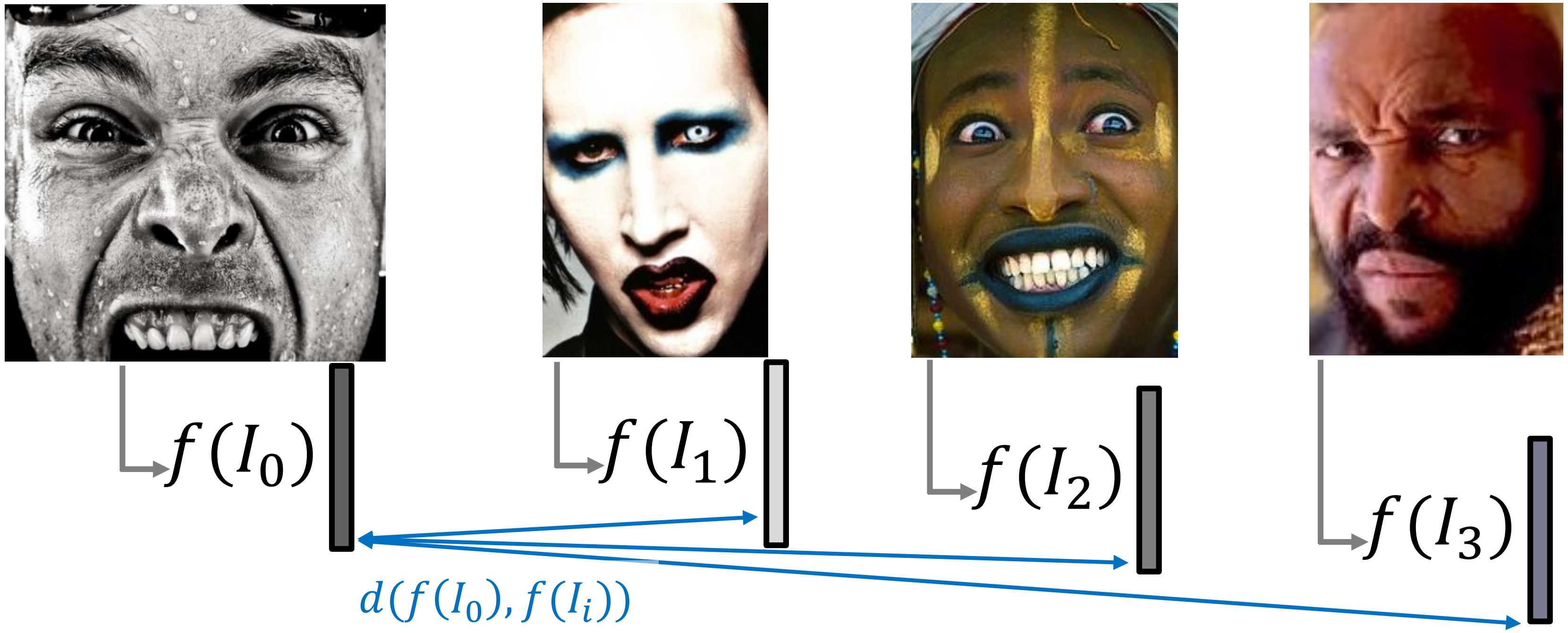
A man from Woodabe tribe?



Mr T?

...

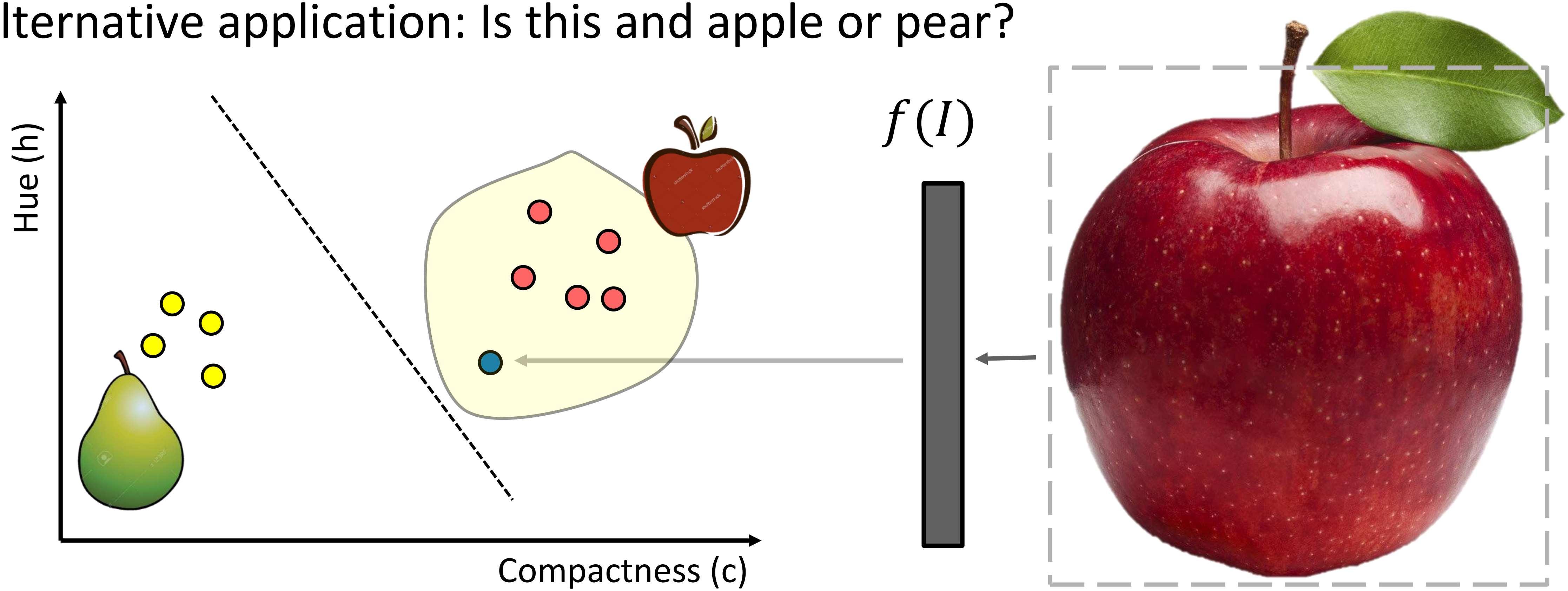
Recognition depends on representation



- Select the one with minimal distance?
- Select the one(s) whose distance is sufficiently small?

Recognition depends on representation

- Alternative application: Is this and apple or pear?



- Quality of recognition will depend on the quality of image representation – features.

How to come up with features?

1. Natural (linear) coordinate systems:
   For some applications, it is enough just to linearly transform the input data.
2. Handcrafted nonlinear transforms:
   Nonlinear transforms improve feature robustness.
3. Feature selection:
   Machine learning to select optimal features from a pool of several handcrafted transforms.
4. End-to-end learning of feature transform:
   Have machine learn entire feature extraction and selection pipeline.

Machine Perception

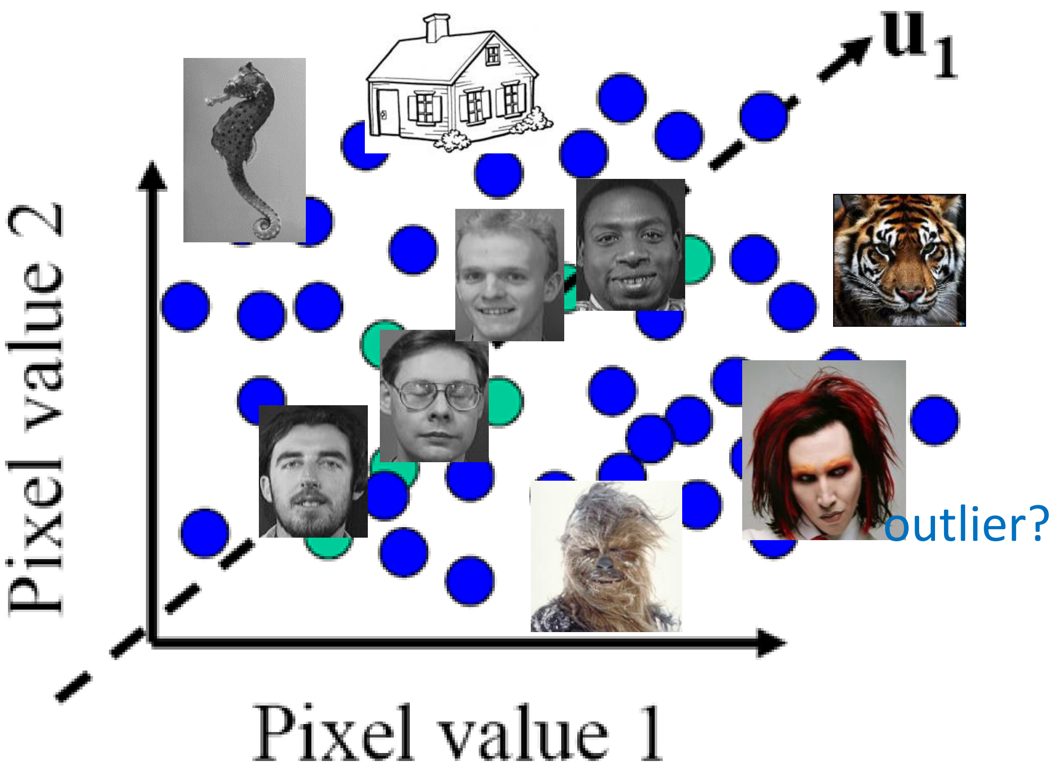
LEARNING NATURAL COORDINATE SYSTEMS BY SUBSPACE METHODS

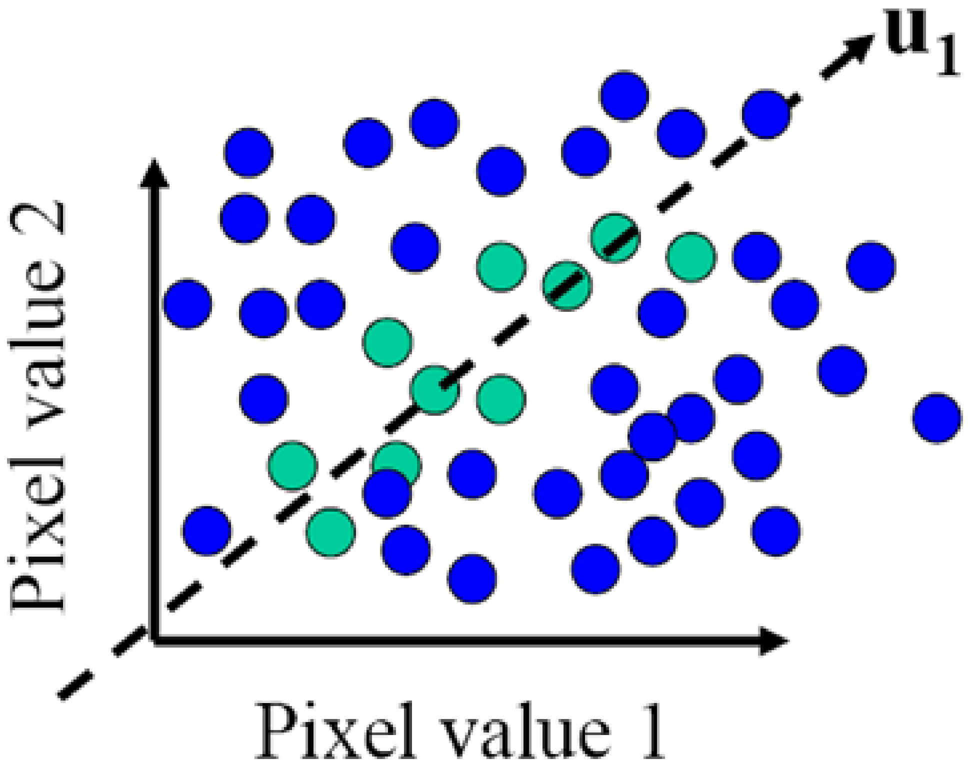
Motivation: A space of all faces

- Image of a face can be treated as a high-dimensional gray-level vector (e.g., stack columns one on top of the other), giving:
  - 100x100 image = 10,000x1 dim vector

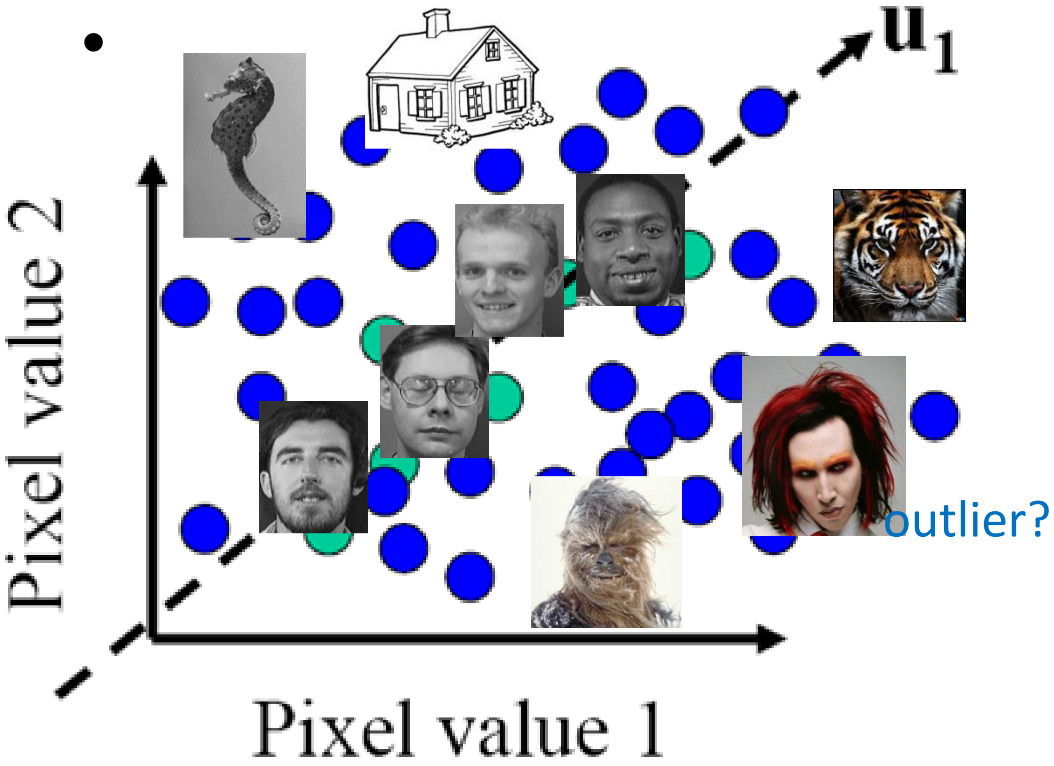
- Still only a fraction of 10,000-dim vectors of natural images really correspond to faces.
- Constrain our representation such that the vectors form a subspace spanned only by faces!

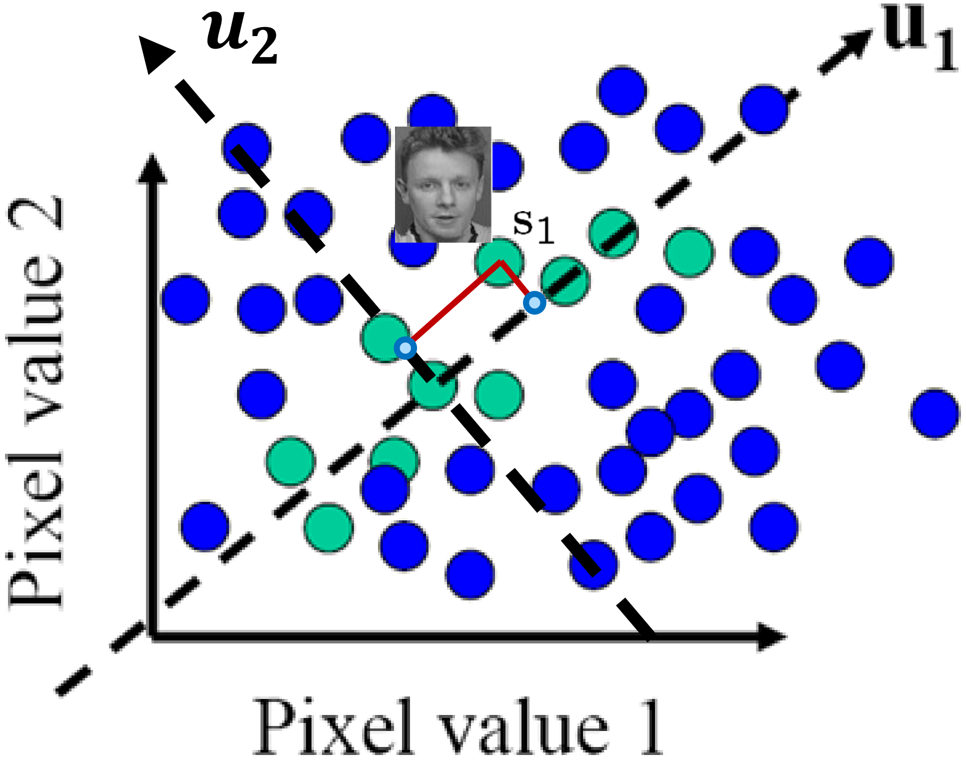
A face sub-space in a nutshell

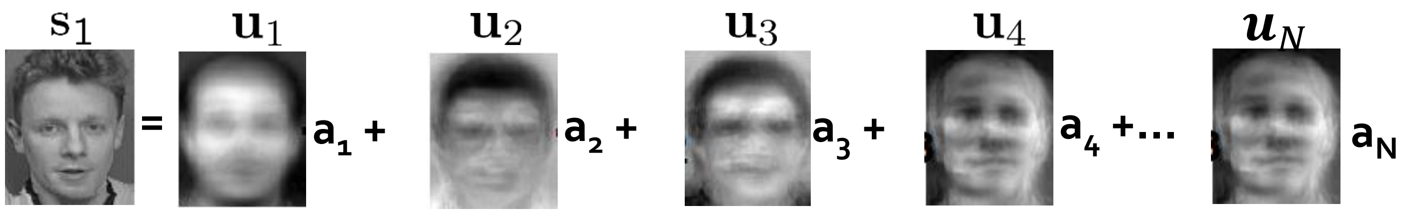




A face sub-space in a nutshell

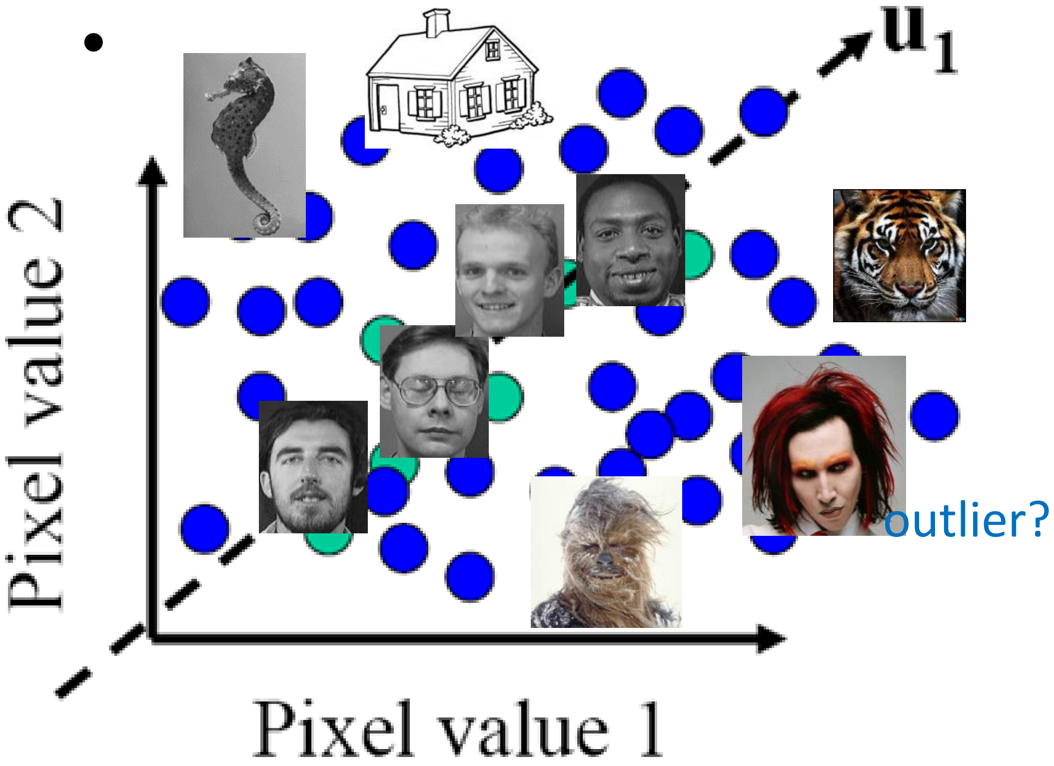
- 

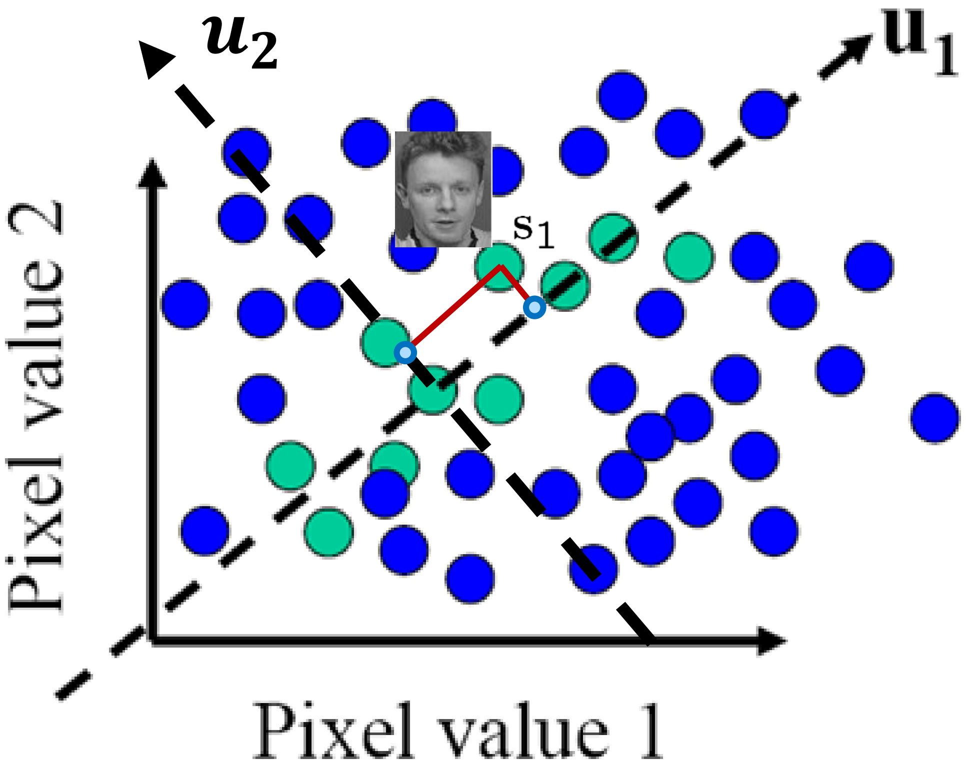


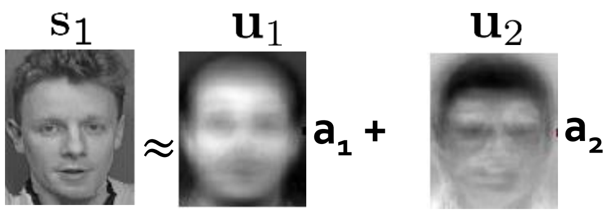


Any point in high dimensional system can be written as a sum of projections to basis vectors u_i.

A face sub-space in a nutshell





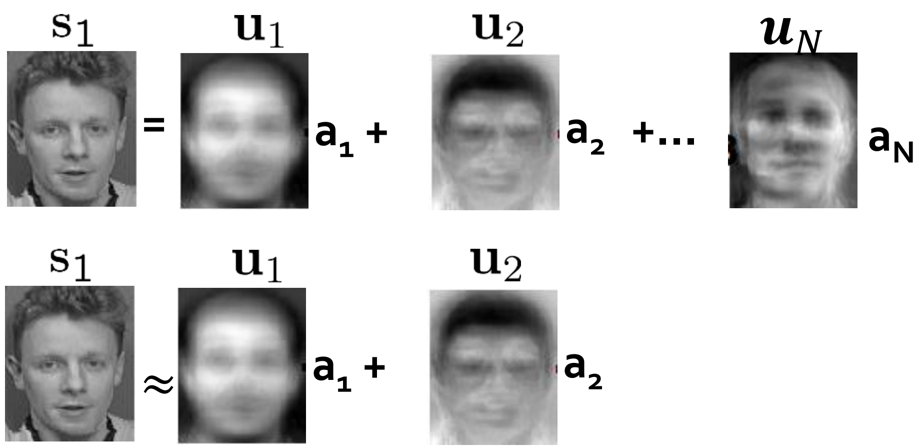


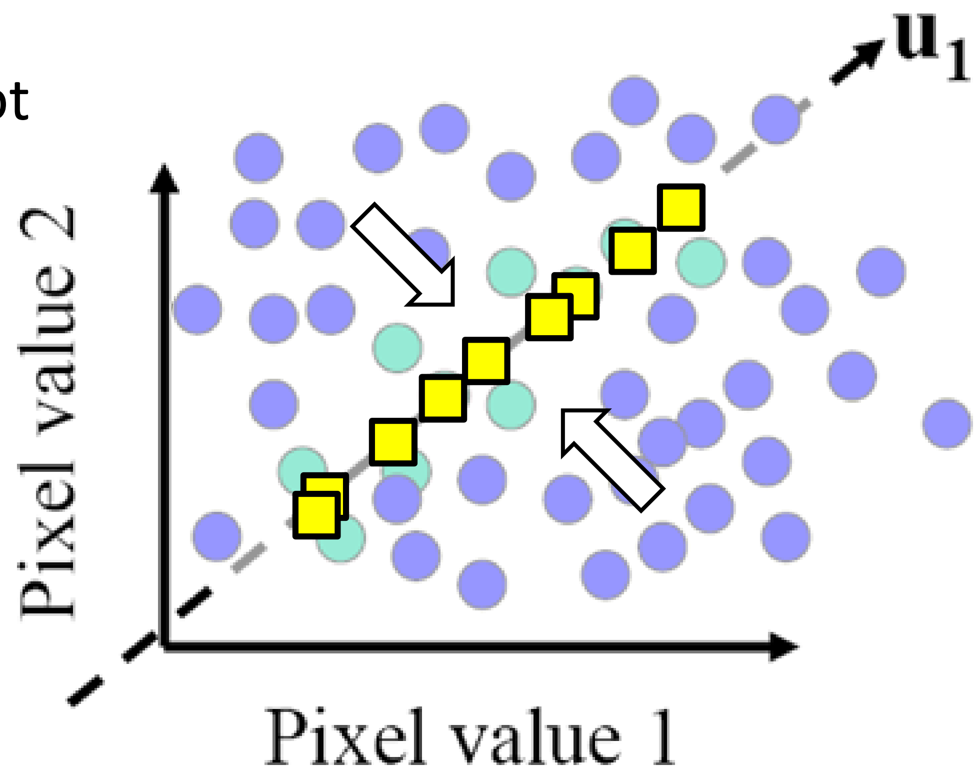
Projection to the right subspace does not distort the data significantly.

Any point in high dimensional system can be written as a sum of projections to basis vectors u_i.

Reconstruction subspace

Projection to the right subspace does not distort the data significantly…





- Instead of memorizing all pixels in an image, remember just:
  1. The subspace vectors (e.g., u_1, u_2, …)
  2. And projections of the images onto the subspace (e.g., a_1, a_2, …)

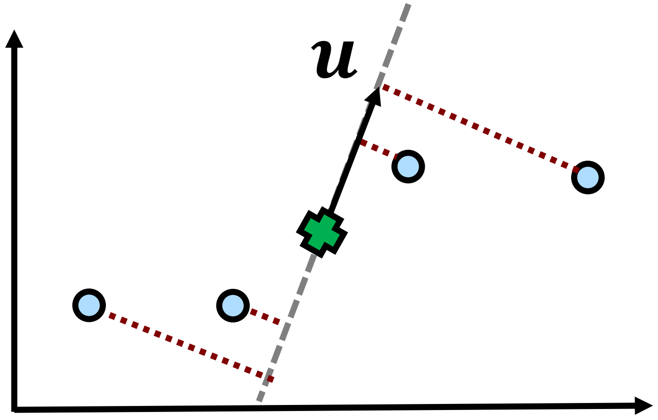
Machine perception

RECONSTRUCTION SUBSPACE: PRINCIPAL COMPONENT ANALYSIS (PCA)

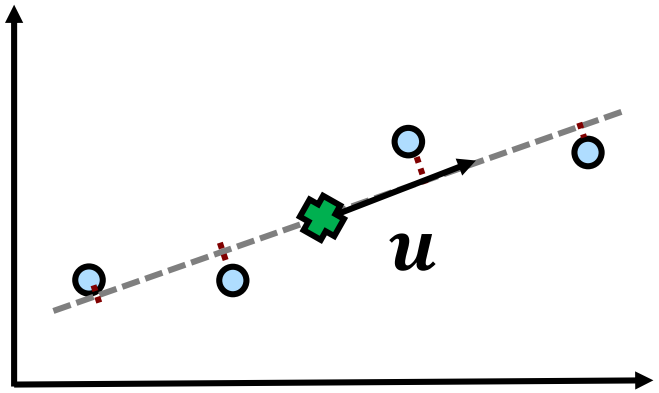
Minimization of reconstruction error

- Find a low-dimensional subspace that efficiently compresses data
- We are given N M-dimensional points (images): $x_1, \dots, x_N$; $x_i \in R^M$
- A toy example of finding 1D subspace in 2D data:

  find a unit vector $\boldsymbol{u} \in R^M$ *such that projection to this vector will minimize the reconstruction error.*



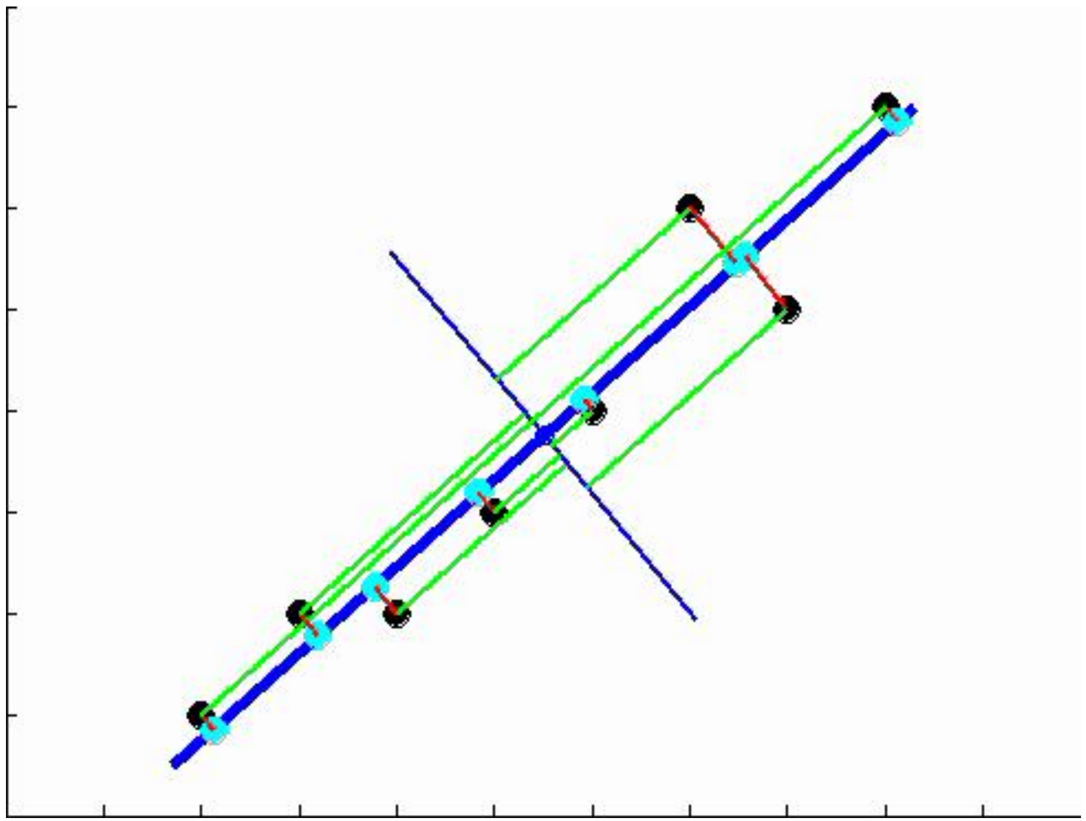
Pretty bad choice of $\boldsymbol{u}$



Much better choice of $\boldsymbol{u}$

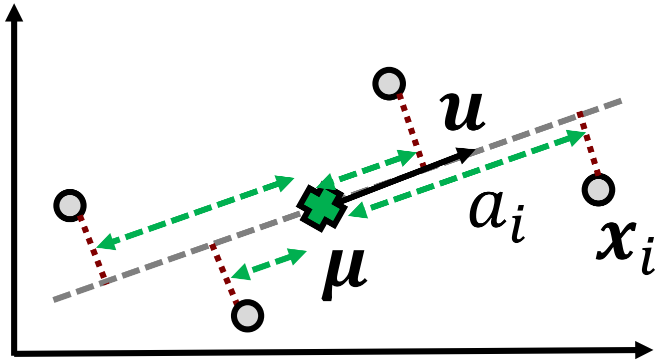
Reconstruction subspace – intuition

- Reconstruction error minimization is equivalent to maximization of variance of projected points

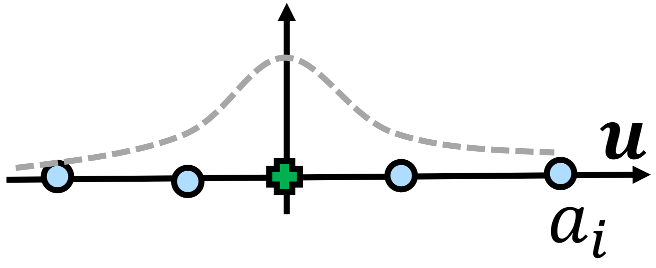


Projection equation:

$$a_i = \boldsymbol{u}^T(\boldsymbol{x}_i - \boldsymbol{\mu})$$

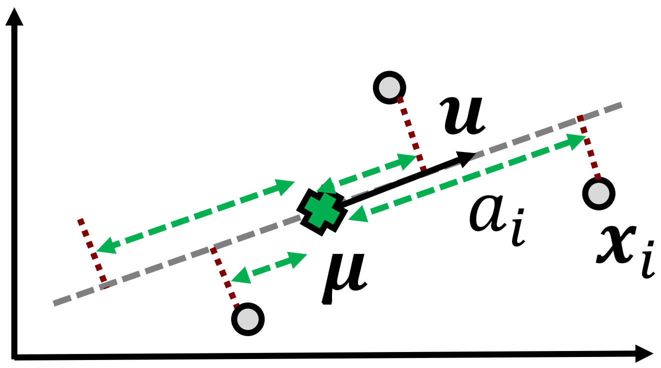


Projection to $\boldsymbol{u}$ visualized:

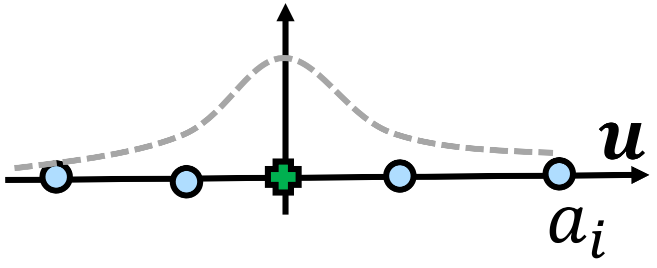


PCA derivation (variance maximization)

- Variance of the projections $\mathrm{a}_i = \boldsymbol{u}^T(\boldsymbol{x}_i - \boldsymbol{\mu})$.



Projection to $\boldsymbol{u}$ visualized:



$$\mathrm{var}(\mathrm{a}) = \frac{1}{N}\sum_{i=1}^{N} a_i^{\,2} \quad = \frac{1}{N}\sum_{i=1}^{N} \mathbf{u}^T(\mathrm{x}_i - \mu)\left[\mathbf{u}^T(\mathrm{x}_i - \mu)\right]^T$$

$$= \mathbf{u}^T \underbrace{\left[\frac{1}{N}\sum_{i=1}^{N}(\mathrm{x}_i - \mu)(\mathrm{x}_i - \mu)^T\right]}_{\text{Data covariance matrix } \boldsymbol{\Sigma}} \mathbf{u}$$

$$\mathrm{var}(\mathrm{a}) = \mathbf{u}^T \boldsymbol{\Sigma} \mathbf{u} \longrightarrow$$ PCA task: Find u that maximizes $\mathrm{var}(a)$!

PCA – variance maximization!

- Task: Find ***u***, that maximizes the following cost function

$$E(\mathbf{u}) = \mathbf{u}^T \mathbf{\Sigma} \mathbf{u}$$ (the variance along vector ***u***)

- Need to constrain our solution: $||\mathbf{u}||^2 = 1$
- Write a Lagrangian for constrained optimization:

$$F(\mathbf{u}) = \mathbf{u}^T \mathbf{\Sigma} \mathbf{u} - \lambda(\mathbf{u}^T \mathbf{u} - 1)$$

$$\frac{\partial F(\mathbf{u})}{\mathbf{u}} \triangleq 0$$

$$\mathbf{\Sigma} \mathbf{u} = \lambda \mathbf{u}$$

- We have obtained a standard equation whose solutions for ***u*** are the eigenvectors of Σ.

PCA – eigenvector maximizes the variance?

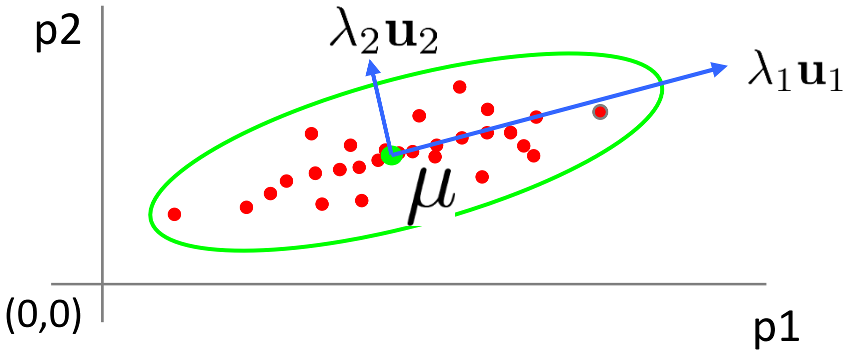
- Recall the projected data variance we want to maximize: $var(a) = \boldsymbol{u}^T \boldsymbol{\Sigma} \boldsymbol{u}$.
- Variance is maximized by $\boldsymbol{u}$ that satisfies $\boldsymbol{\Sigma u} = \boldsymbol{\lambda u}$, i.e., eigenvector of $\boldsymbol{\Sigma}$.
- But *which* eigenvector maximizes the variance?
- Multiplying both sides by $\boldsymbol{u}^T$ yields: $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{\Sigma u} = \lambda \boldsymbol{u}^{\mathrm{T}}\boldsymbol{u} = \lambda \cdot 1$.
- Therefore the maximum of $var(a) = \lambda$ is reached at the largest eigenvalue.
- This means that the variance $var(a)$ is maximized by the eigenvector that corresponds to the largest eigenvalue.
- *A similar argument can be made to prove that the eigenvectors corresponding to large eigenvalues are directions of dominant variance in the data.*

PCA – geometric interpretation

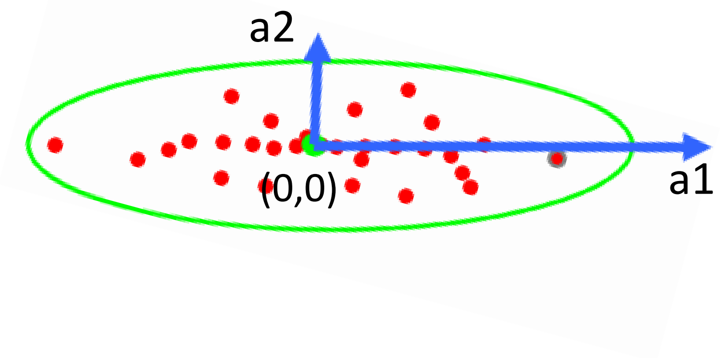
- Calculate eigenvectors and eigenvalues of covariance matrix $\mathbf{\Sigma}$
- Eigenvectors: main directions of variance, perpendicular to each other.

$$\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2]$$

- Eigenvalues: variance of data in direction of eigenvectors



1. Translate to origin by $\mathbf{t} = \boldsymbol{\mu}$
2. Rotate by $\boldsymbol{R} = \boldsymbol{U}$

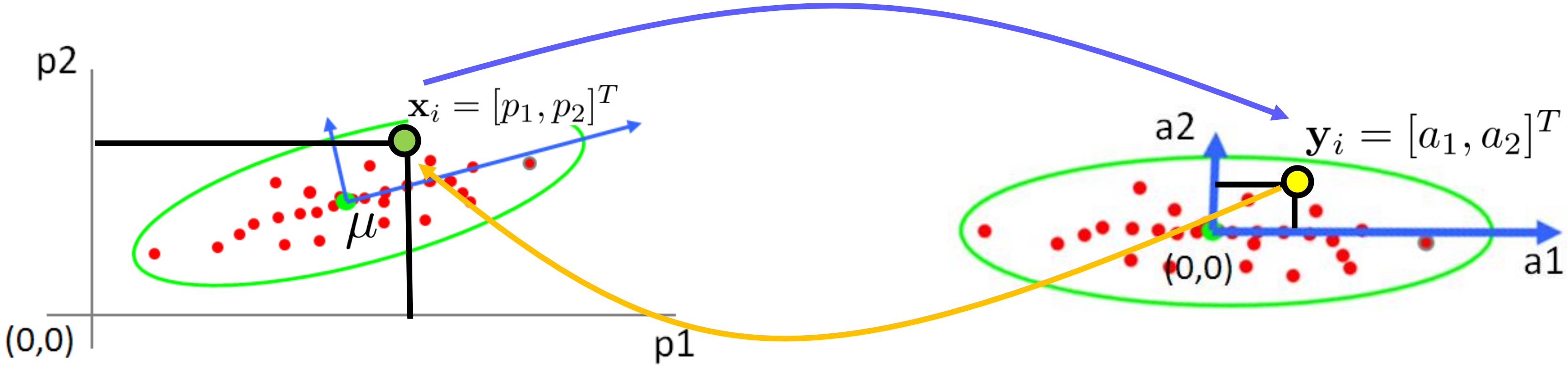


- PCA is actually: change of coordinate system that captures major directions of variance in the data.

Projection and reconstruction

- We know the covariance matrix $\boldsymbol{\Sigma}$ and the mean value $\boldsymbol{\mu}$
- Concatenate M first (this case all) eigenvectors into a rotation matrix ***U***:

$$\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_M]$$



- Projection of data $\boldsymbol{x}_i$ into the new coordinate system:

$$\mathbf{y}_i = \mathbf{U}^T(\mathbf{x}_i - \mu)$$

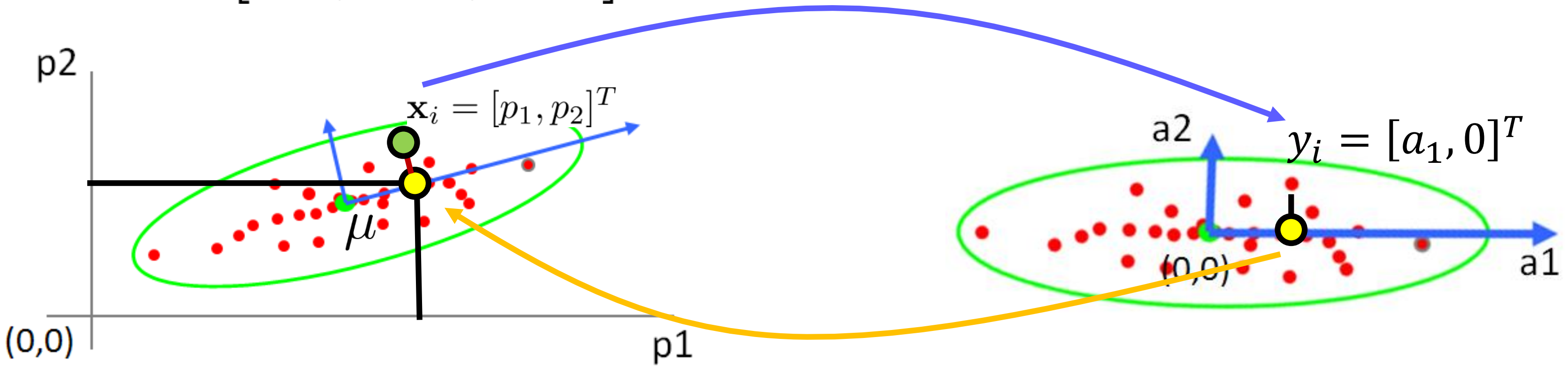
- Projection of $\boldsymbol{y}_i$ back into the original coordinate system:

$$\mathbf{x}_i = \mathbf{U}\mathbf{y}_i + \mu$$

Projection and reconstruction

- Similar holds also for K < M !
- Create ***U*** from just the first K eigen vectors:

$$\tilde{\mathbf{U}} = [\mathbf{u}_1, \ldots, \mathbf{u}_K]$$



- Projection to subspace:

$$\mathbf{y}_i = \tilde{\mathbf{U}}^T(\mathbf{x}_i - \mu)$$

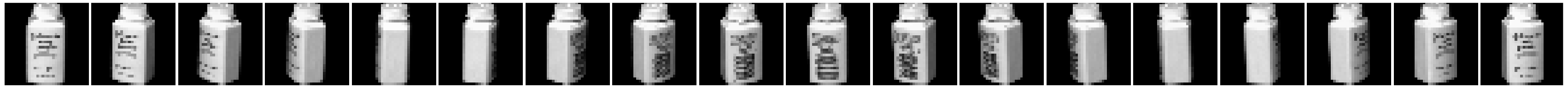
- Reconstruction:

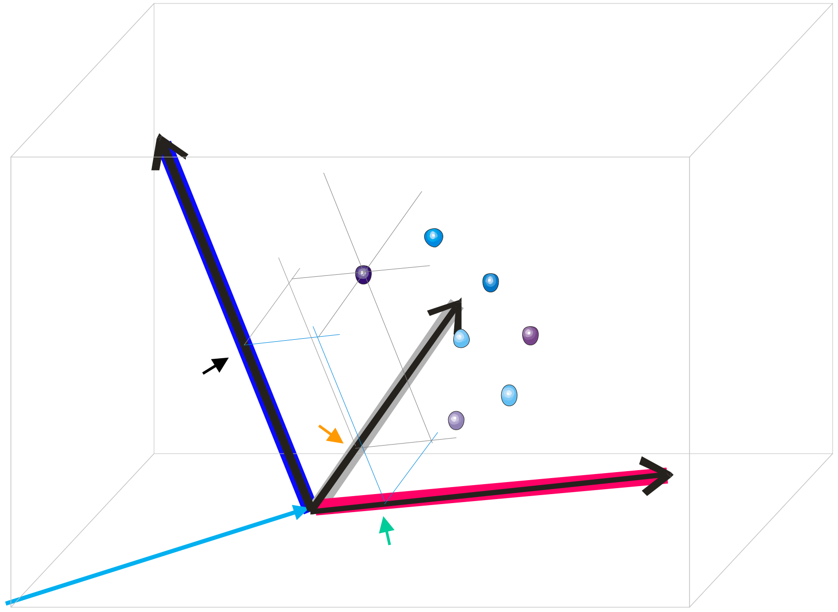
$$\tilde{\mathbf{x}}_i = \tilde{\mathbf{U}}\mathbf{y}_i + \mu$$

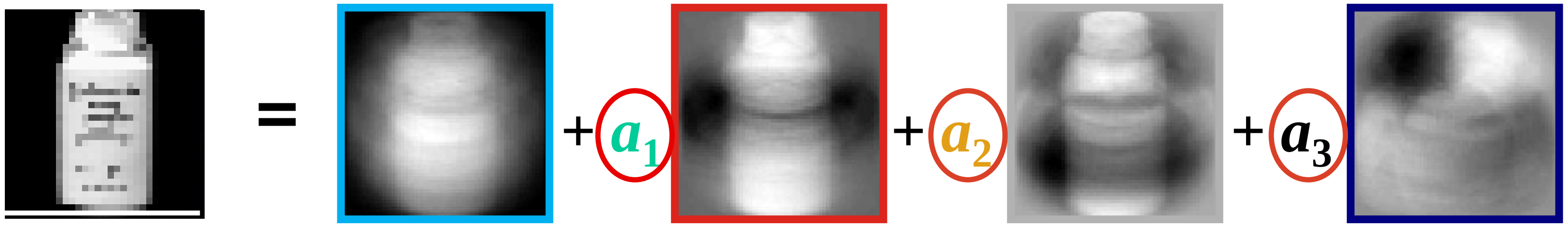
Reconstruction error:

$$||\mathbf{x}_i - \tilde{\mathbf{x}}_i||^2$$

Example: Object representation







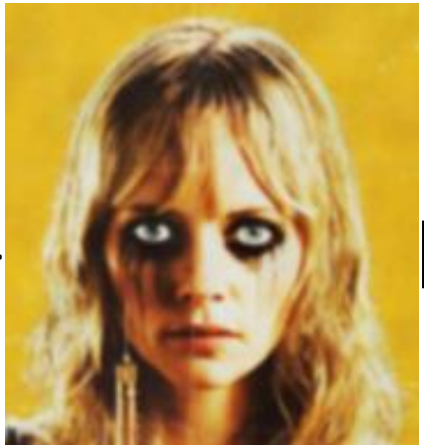
Q: How many of a_i should you retain?

How many eigenvectors for reconstruction?

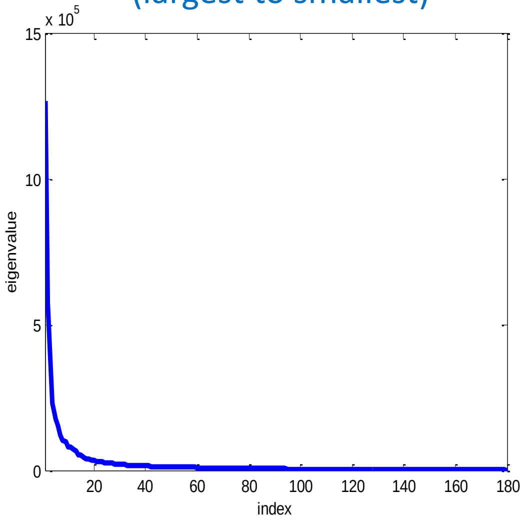
- Can show that the sum of squared differences $\epsilon(m)$ between training images $\{\boldsymbol{x}_i\}_{i=1:N}$ and their reconstructions using only first m eigen vectors is given by:

Original

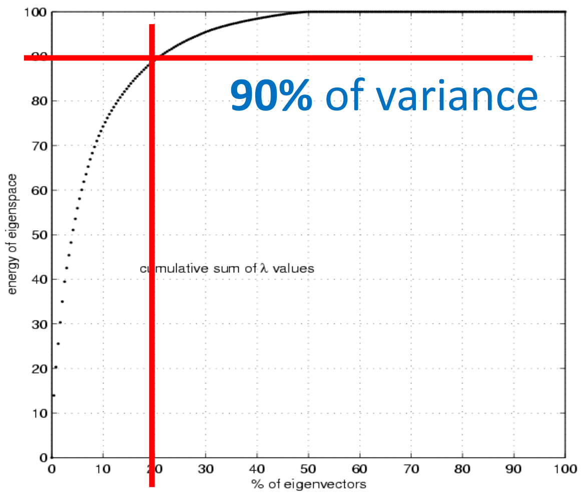
Reconstruction with m eigenvectors

$\epsilon(m) = \|$  -  $\|^2$

Eigenvalues (largest to smallest)



Cumulative sum of eigenvalues (explained variance)



m eigen vectors

$$\epsilon(m) = \sum_{j=1}^{N} \lambda_j - \sum_{j=1}^{m} \lambda_j = \sum_{j=m+1}^{N} \lambda_j$$

Build you own subspace!

- Reshape all training images into column vectors: $[\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]$
- Calculate the average image: $\mu = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i$
- Center data: $\mathbf{X}_d = [\mathbf{x}_1 - \mu, \mathbf{x}_2 - \mu, \ldots, \mathbf{x}_N - \mu]$
- Calculate the covariance matrix: $\mathbf{C} = \frac{1}{N}\sum_{i=1}^{N} (\mathbf{x_i} - \mu)(\mathbf{x_i} - \mu)^T = \frac{1}{N}\mathbf{X}_d\mathbf{X}_d^T$
- Calculate eigenvector matrix ***U*** and eigenvalue matrix ***S*** (using, e.g., svd): $\mathbf{C} = \mathbf{U}\mathbf{S}\mathbf{V}^T$
- Construct a matrix using only first K eigen vectors: $\tilde{\mathbf{U}} = [\mathbf{u}_1, \ldots, \mathbf{u}_K]$
- For each test image ***x***:
  - Project to subspace: $\mathbf{y} = \tilde{\mathbf{U}}^T(\mathbf{x} - \mu)$
  - Reconstruct: $\tilde{\mathbf{x}} = \tilde{\mathbf{U}}\mathbf{y} + \mu$

Important!

- Do not implement PCA as shown in the previous slide!

$$\mathbf{C} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x_i} - \mu)(\mathbf{x_i} - \mu)^T = \frac{1}{N}\mathbf{X}_d\mathbf{X}_d^T$$

1. Consider the size of the covariance matrix C
   - *The size is M×M*, where *M* is the number of pixels in the image!
   - But, we have only N training examples, typically N<<M.

   $\Rightarrow$ So C will have at most rank N!

2. In any case, we need only first k eigen vectors!

The inner-product matrix

- For *a large* M, the SVD of C becomes inefficient.

$$\mathbf{C} = \frac{1}{N}\mathbf{X}_d\mathbf{X}_d^T$$

- For *N* <<*M*, it is better to compose the $N \times N$ inner product matrix $\mathbf{C}'$:

$$\mathbf{C}' = \frac{1}{N}\mathbf{X}_d^T\mathbf{X}_d$$

- Eigenvectors and eigenvalues of matrix $\boldsymbol{C}$ are calculated from the eigenvectors $\boldsymbol{u}_i'$ and eigenvalues λ_i' of $\boldsymbol{C}'$:

$$\lambda_i = \lambda_i'$$
$$\mathbf{u}_i = \frac{\mathbf{X}_d\mathbf{u}_i'}{\sqrt{N\lambda_i'}},\ i = 1 \ldots N$$

This is called
“the dual PCA”

A general PCA algorithm

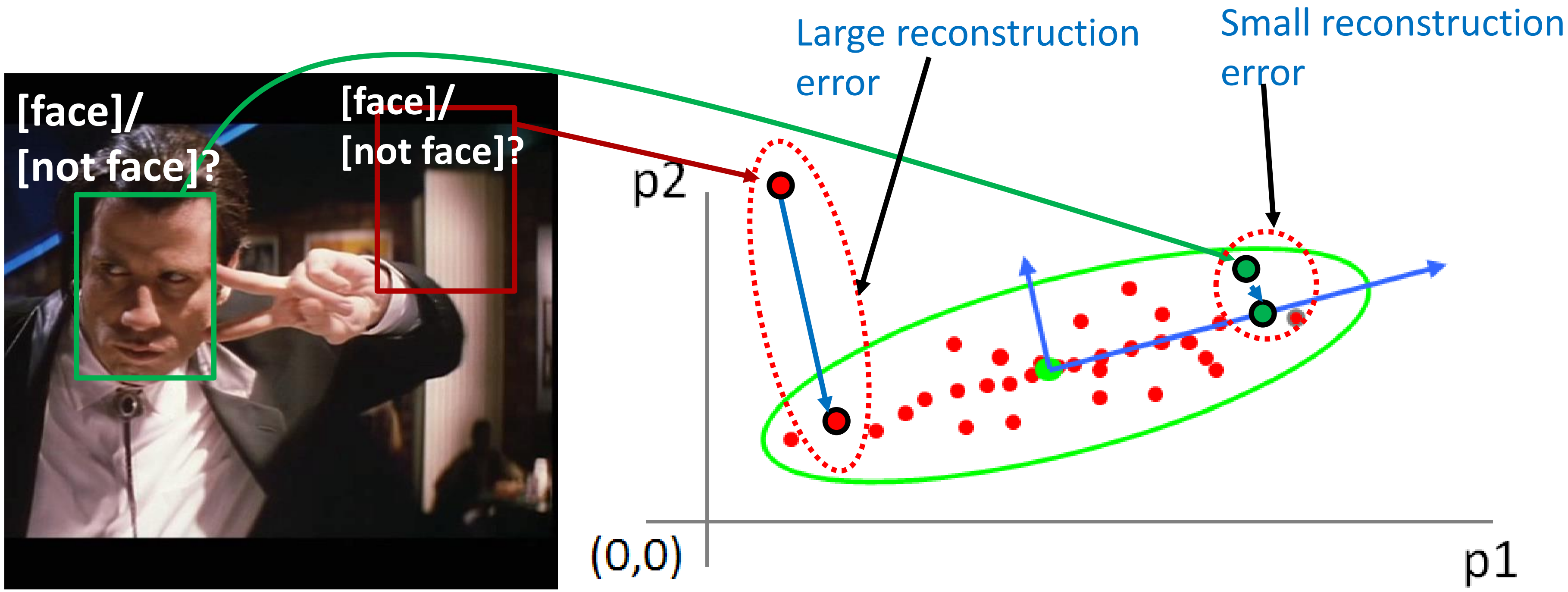
Input: data matrix $\mathbf{X}$

Output: mean value $\boldsymbol{\mu}$, eigenvectors $\mathbf{U}$, eigenvalues $\boldsymbol{\lambda}$.

1. Estimate the mean vector: $\boldsymbol{\mu} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i$
2. Center the input data around the mean: $\hat{\mathbf{X}}_i = \mathbf{X}_i - \mu$
3. **if** $M \leq N$ **then**
4. Estimate the covariance matrix : $\mathbf{C} = \frac{1}{N} \hat{\mathbf{X}} \hat{\mathbf{X}}^\top$
5. Perform SVD on $\mathbf{C}$. Obtain eigenvectors $\mathbf{U}$ and eigenvalues $\boldsymbol{\lambda}$.
6. **else**
7. Estimate the inner product matrix: $\mathbf{C}' = \frac{1}{N} \hat{\mathbf{X}}^\top \hat{\mathbf{X}}$
8. Perform SVD on $\mathbf{C}'$. Obtain eigenvectors $\mathbf{U}'$ and eigenvalues $\boldsymbol{\lambda}'$.
9. Determine the eigenvectors $\mathbf{U}$: $\mathbf{u}_i = \frac{\hat{\mathbf{X}} \mathbf{u}'_i}{\sqrt{N \lambda'_i}}$, $i = 1 \ldots N$
10. Determine the eigenvalues $\boldsymbol{\lambda} = \boldsymbol{\lambda}'$
11. **end if**

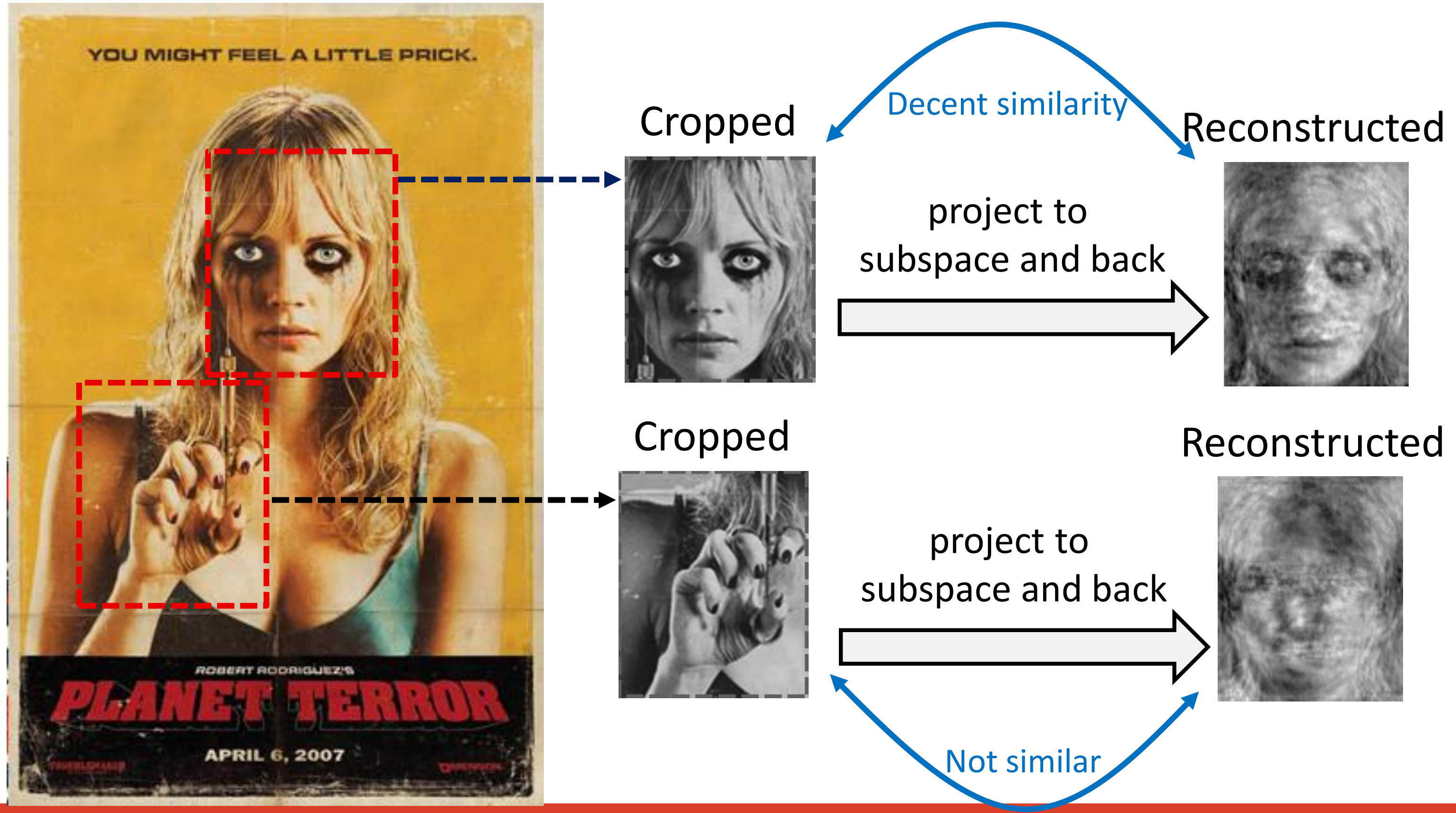
Classification by subspace reconstruction

- Assume we have used a large collection of faces to construct the subspace.
- Assumption: faces will be well reconstructed by the subspace!



Classification by subspace reconstruction

- If the window contains a category for which the subspace was constructed, the reconstruction will work well, otherwise not!
- A real-life example

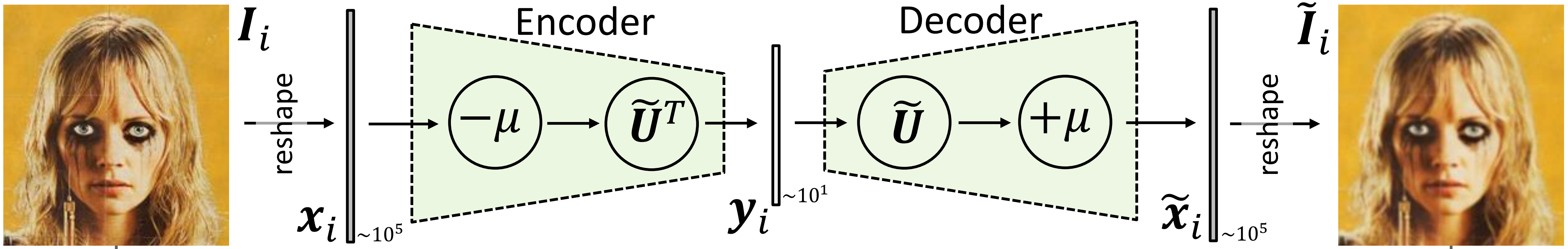


Detection by distance from subspace

- Use a subspace learned on faces to detect a face.
- Approach: slide a window over each image position and calculate the reconstruction error.
- Repeat for all scales. Makes sense to normalize the window intensity |W|=1.
- Low reconstruction error indicates a face. (i.e., apply a threshold)

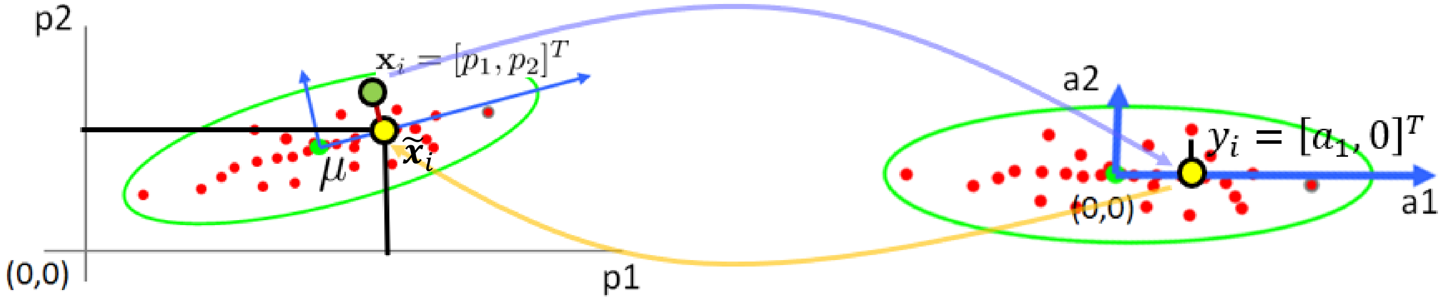
$$\left\|\tilde{\mathbf{x}}_i - \mathbf{x}_i\right\|^2 < \theta$$

PCA is a *linear* autoencoder



$$\mathcal{L}(\mu, \tilde{U}) = \left\| \boldsymbol{I}_i - \tilde{\boldsymbol{I}}_i \right\|^2$$

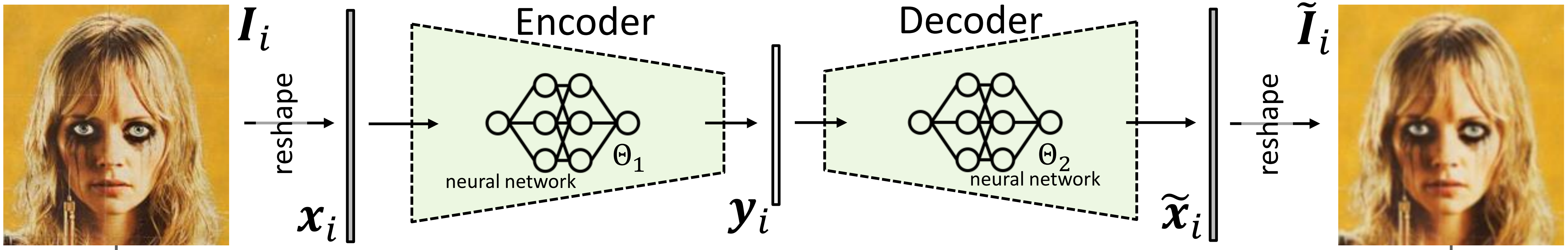
$$\mu, \tilde{U} = \underset{\mu^*, \tilde{U}^*}{\operatorname{argmin}} \, \mathcal{L}(\mu^*, \tilde{U}^*)$$



$$\mathbf{y}_i = \tilde{\mathbf{U}}^T (\mathbf{x}_i - \mu)$$

$$\tilde{\mathbf{x}}_i = \tilde{\mathbf{U}} \mathbf{y}_i + \mu$$

Encoder-Decoder does not have to be linear

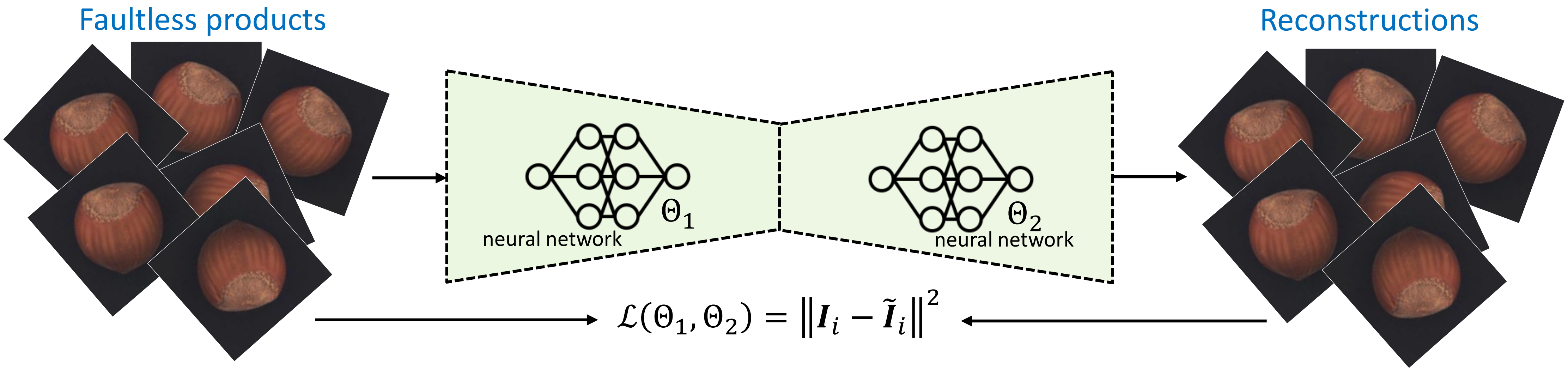


$$\mathcal{L}(\Theta_1, \Theta_2) = \left\| \boldsymbol{I}_i - \tilde{\boldsymbol{I}}_i \right\|^2$$

$$\Theta_1, \Theta_2 = \underset{\Theta_1^*, \Theta_2^*}{\operatorname{argmin}} \mathcal{L}(\Theta_1^*, \Theta_2^*)$$

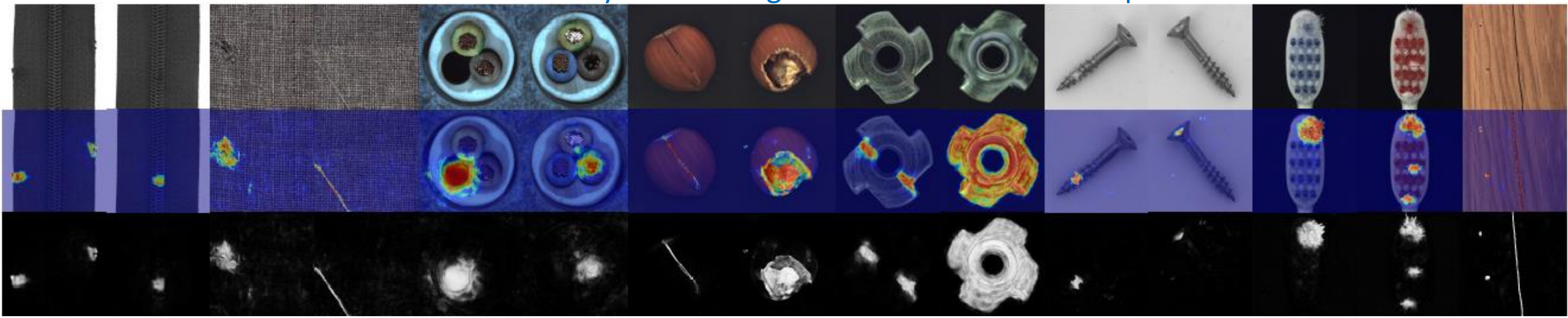
- Modern Autoencoders apply (convolutional) neural networks to map into a nonlinear subspace (latent space)

Autoencoders don’t have to just reconstruct



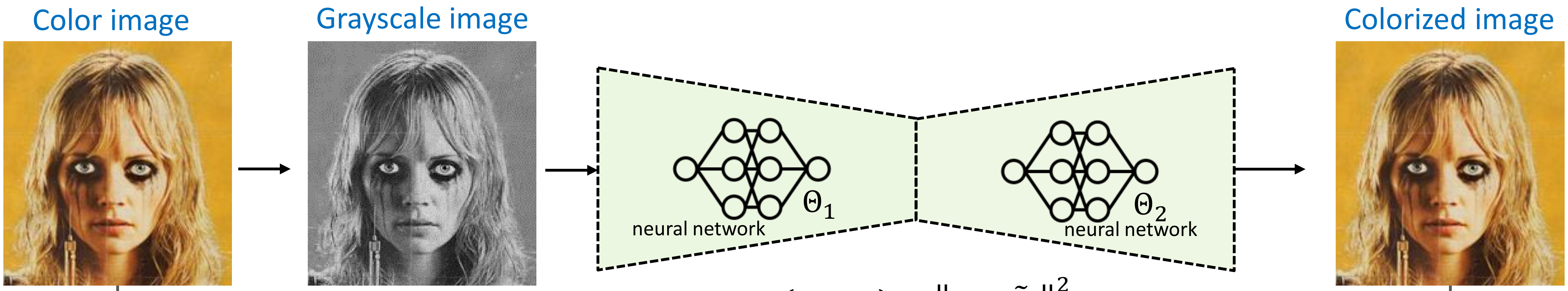
$$\mathcal{L}(\Theta_1, \Theta_2) = \left\| \boldsymbol{I}_i - \tilde{\boldsymbol{I}}_i \right\|^2$$

Anomalies detected by differencing the reconstruction and input

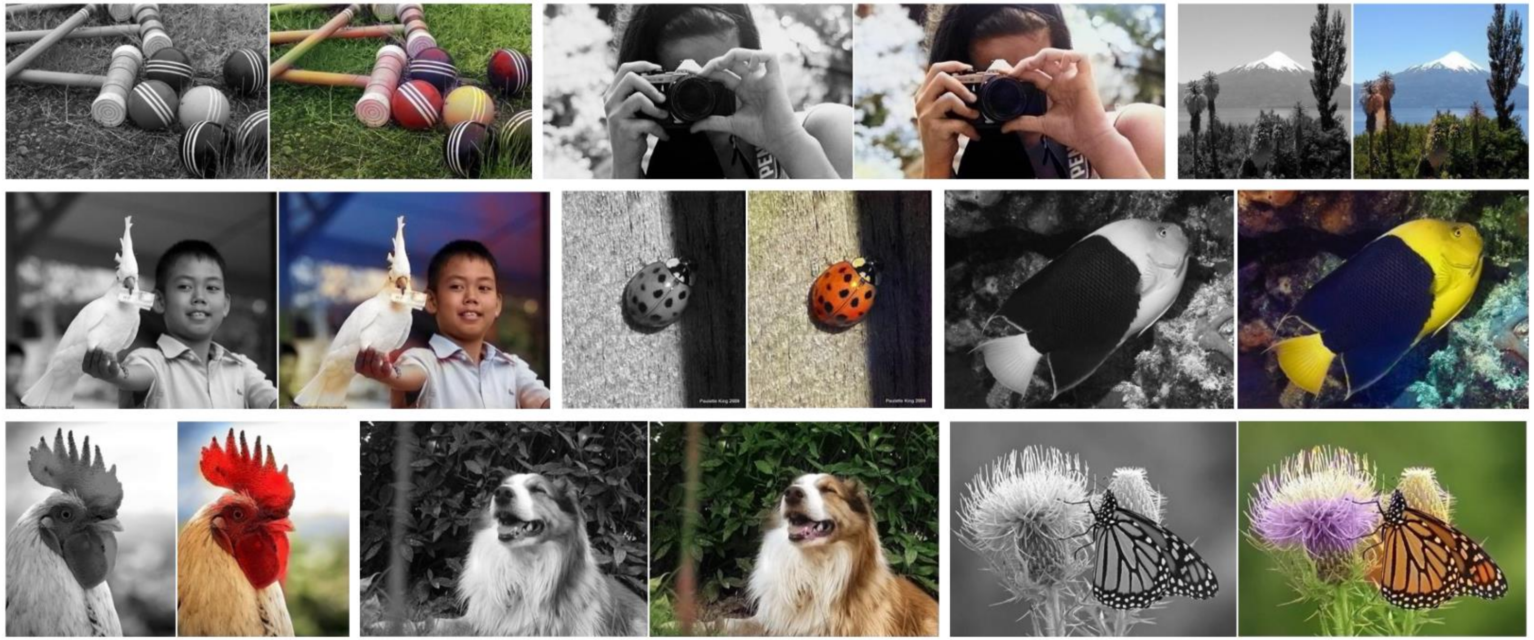


Zavrtanik, Kristan, Skočaj, DRÆM – A discriminatively trained reconstruction embedding for surface anomaly detection, ICCV 2021

Autoencoders don’t have to just reconstruct



$$\mathcal{L}(\Theta_1, \Theta_2) = \left\| \boldsymbol{I}_i - \tilde{\boldsymbol{I}}_i \right\|^2$$



Zhang et al., Colorful Image Colorization, ECCV 2016 [GIT]

Textbooks on PCA

- Szeliski, R., Computer vision – algorithms and applications, 2011, Section14.2.1 (available online)

- Forsyth, Ponce, Computer vision – a modern approach, second edition, 2012, Section 16.1.5

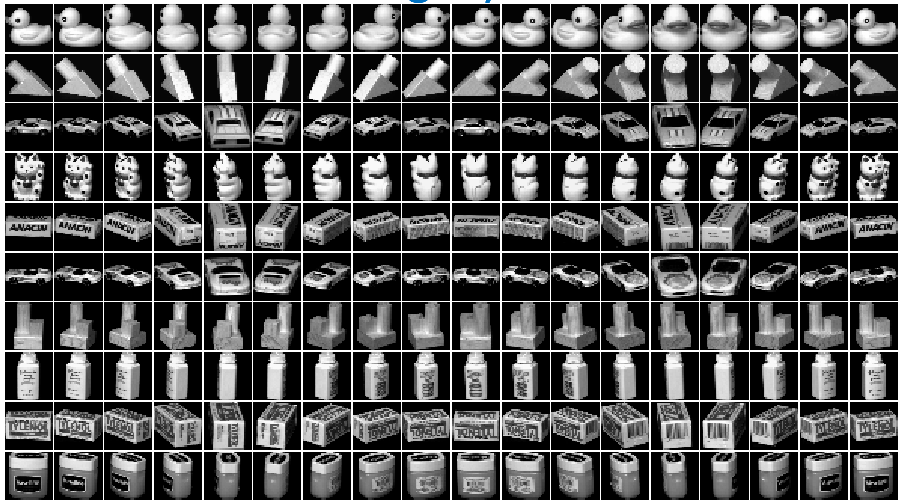
- Prince, S.J.D. Computer vision – modelling learning and inference, 2012, Section 13.4 (to 13.4.3) (available online)

Classification task

- Assume we are given some training data of two categories:

Category 1

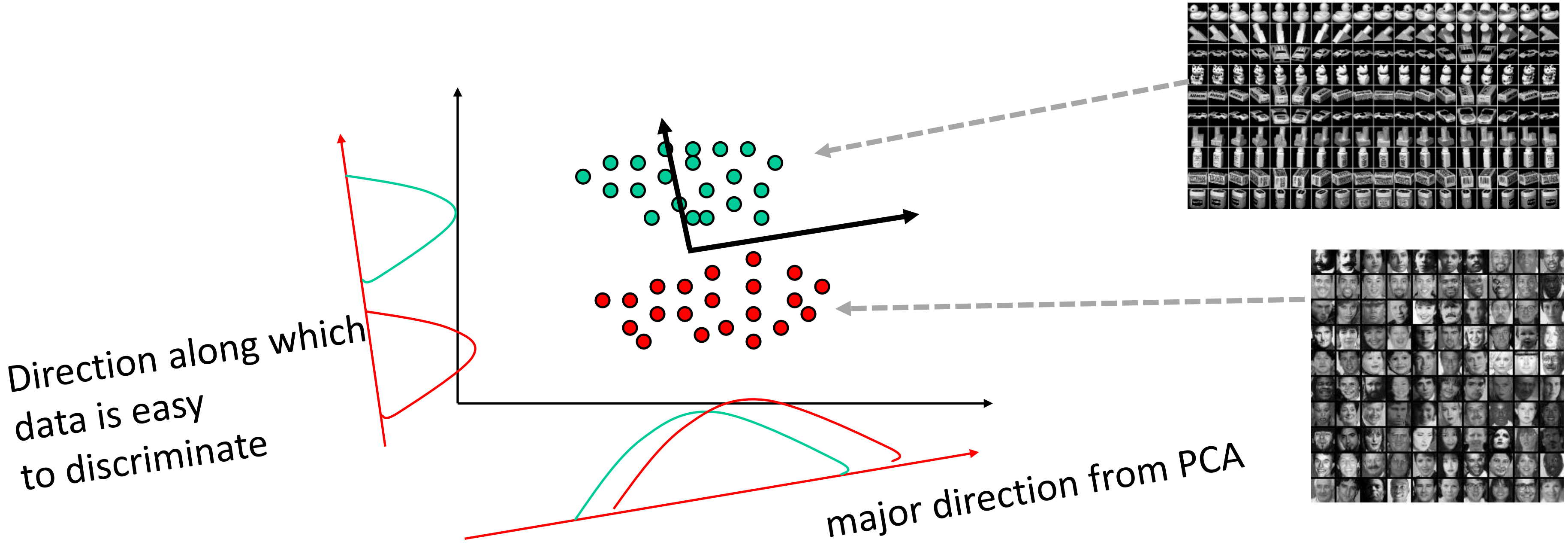
Category 2



- Task: Find a feature space in which these categories are most easily distinguishable (maximize discrimination).

Could we apply PCA?

- PCA minimizes reprojection (reconstruction) error

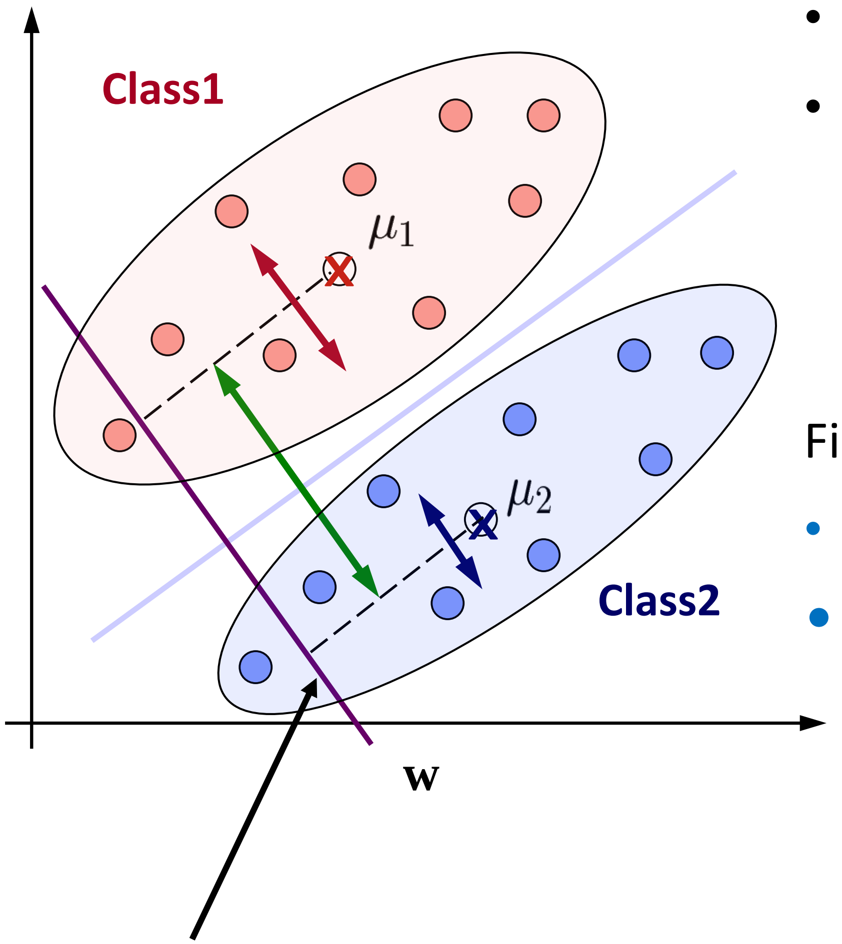


- PCA is „unsupervised“: does not use class-label information
- That is why the discriminative information is not necessarily preserved.

Machine perception

DISCRIMINATION SUBSPACE: LINEAR DISCRIMINANT ANALYSIS (LDA)

Linear Discriminant Analysis (LDA)



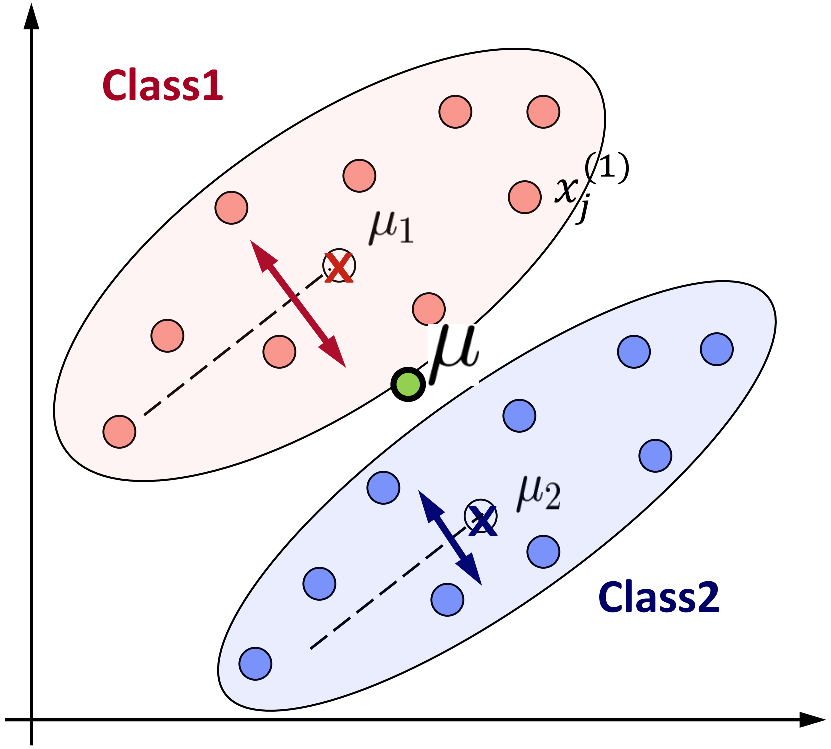
Subspace defined by $\boldsymbol{w}$.

- Assume we know the class labels.
- Task: derive an approach that takes the class-labels into consideration in subspace estimation.

Find a subspace that:

- Maximizes distances between classes
- Minimizes distances within classes

LDA – the mean image



- Let $X_1, X_2, \ldots, X_c$ be sets of images from c classes and let each set contain k images: $X_i = \left\{x_j^{(i)}\right\}_{j=1:k}$.

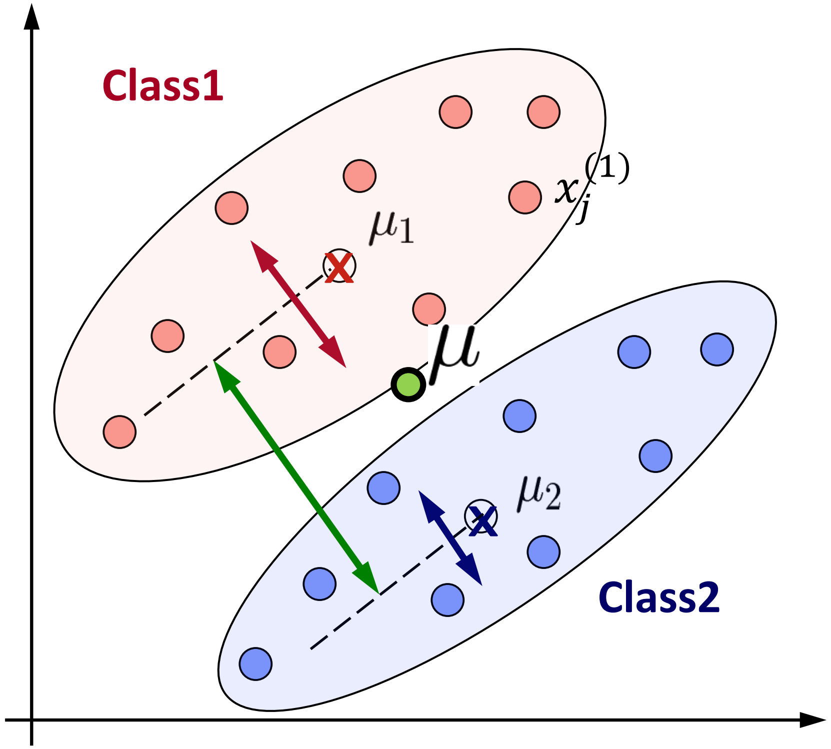
- For each class the mean image μ_i:

$$\mu_i = \frac{1}{k}\sum_{j=1}^{k} x_j^{(i)}$$

- The mean image over all classes:

$$\mu = \frac{1}{c}\sum_{i=1}^{c} \mu_i$$

LDA – scatter matrices (covariances)



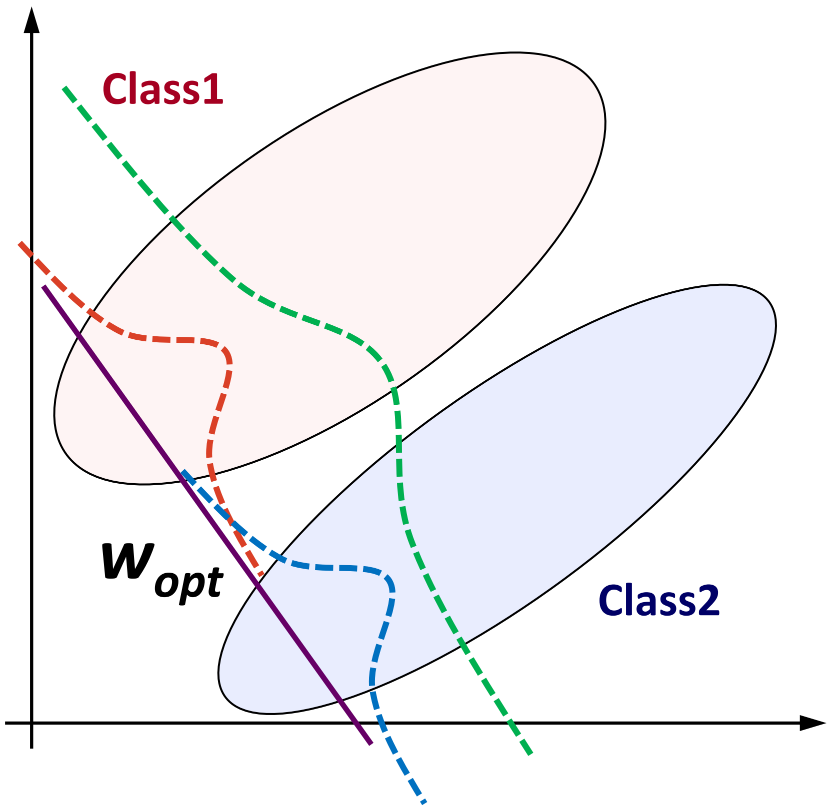
- The within-class scatter matrix:

$$S_W = \sum_{i=1}^{c} \sum_{j} (x_j^{(i)} - \mu_i)(x_j^{(i)} - \mu_i)^T$$

- The between-class scatter matrix:

$$S_B = \sum_{i=1}^{c} N_i (\mu_i - \mu)(\mu_i - \mu)^T$$

LDA – The cost function



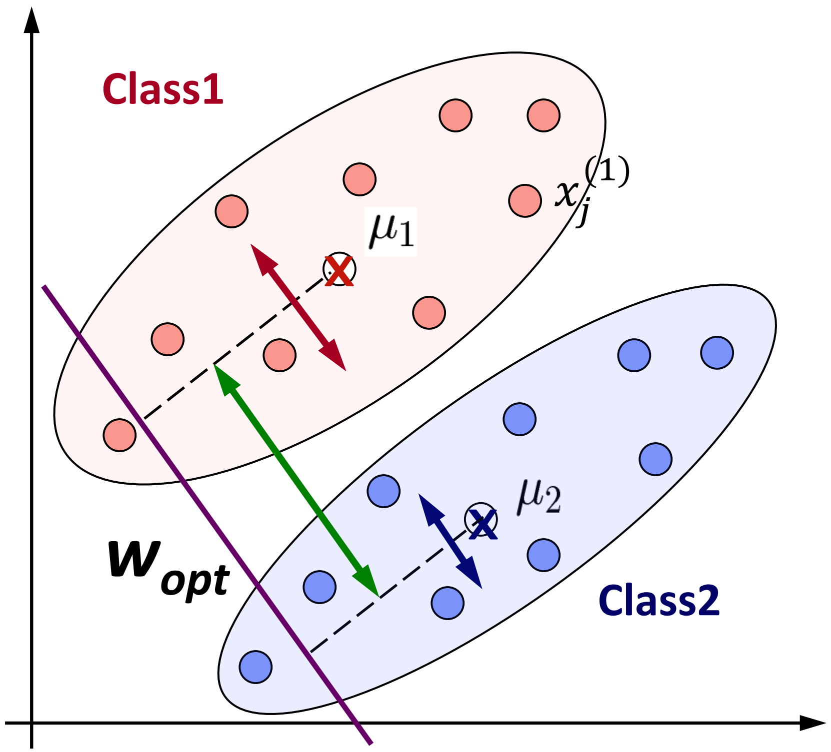
- Looking for a projection direction ***w***, such that the projection of:
  - within-class scatter matrix S_W is small (compact classes)
  - between-class scatter matrix S_B is large (classes far apart)
- Recall covariance matrix projection from PCA derivation:

$$\sigma^2(\mathbf{w}) = \mathbf{w}^T \boldsymbol{\Sigma} \mathbf{w}$$

- The projected scatter matrices:

$$\sigma_W^2 = \mathbf{w}^T S_W \mathbf{w}$$
$$\sigma_B^2 = \mathbf{w}^T S_B \mathbf{w}$$

LDA – The cost function



- The projected variances:

$$\sigma_W^2 = \mathbf{w}^T S_w \mathbf{w}$$

$$\sigma_B^2 = \mathbf{w}^T S_b \mathbf{w}$$

- *Ronald* A. *Fisher* formulated the *Linear Discriminant* (in 1936):

$$J(\mathbf{w}) = \frac{\sigma_B^2}{\sigma_W^2}$$

- Fisher criteria (cost function):

$$J(\mathbf{w}) = \frac{\mathbf{w}^T S_b \mathbf{w}}{\mathbf{w}^T S_w \mathbf{w}}$$

$$\mathbf{w}_{opt} = \arg\max_{\mathbf{w}} J(\mathbf{w})$$

LDA – optimal projection calculation

- Maximize cost function over ***w***: $$J(w) = \frac{w^T S_B w}{w^T S_W w}$$

  the solution as a generalized eigenvalue problem:

  $$S_B w = \lambda S_W w$$

- If S_W is full-rank, we have $S_W^{-1} S_B w = \lambda w$,
  which is in fact the standard eigenvalue problem.

- For *c* classes we get at most (*c*-1) projection directions.

LDA application

- Training:
  - Compute LDA on a set of images ***X***, and calculate the basis functions ***W*** as well as center means ($\boldsymbol{\mu_1}$, $\boldsymbol{\mu_2}$, **…**) and overall mean $\boldsymbol{\mu}$.

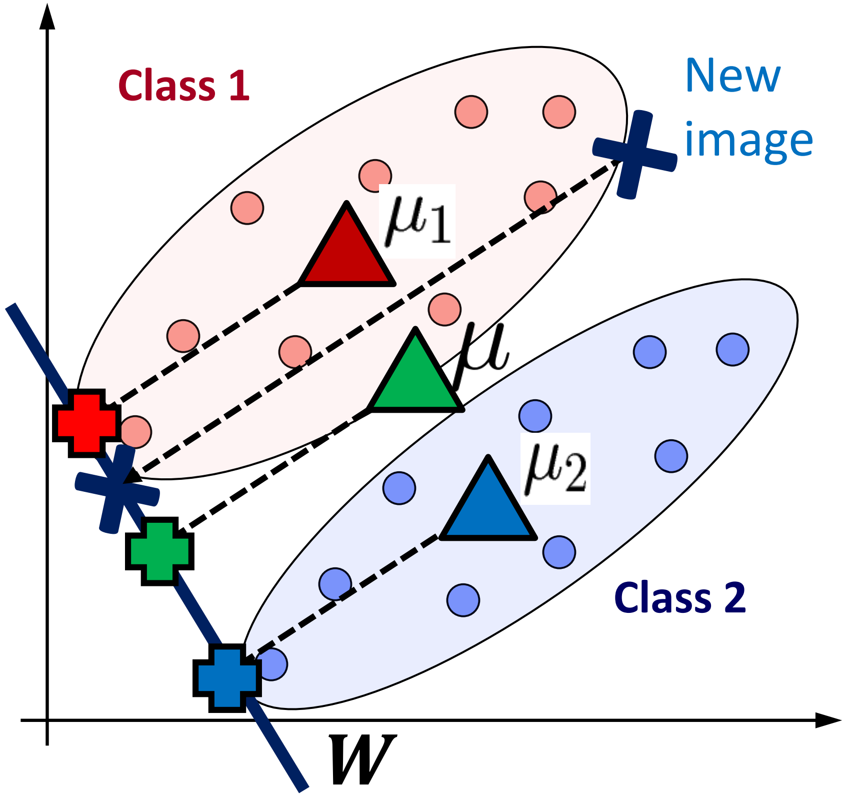
$$S_W^{-1} S_B w = \lambda w$$

- Recognition:
  - Project new image ***x*** into the LDA subspace using the matrix ***W***

$$\mathbf{y} = \mathbf{W}^T(\mathbf{x} - \mu)$$

  - Find the nearest class center among all (projected) class centers:

$$\{\mathbf{W}^T(\mu_i - \mu)\}_{i=1:c}$$



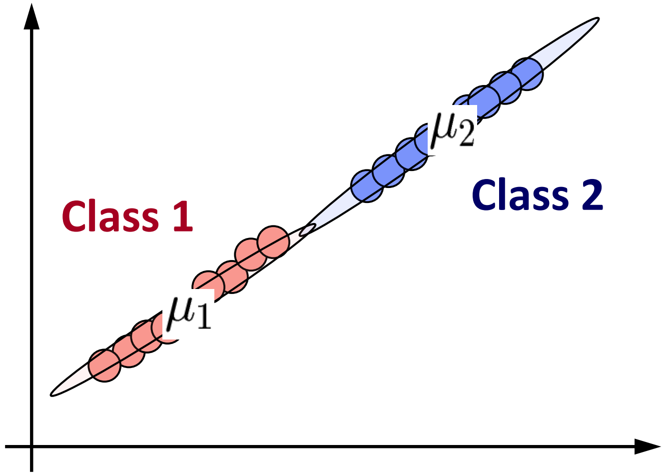
LDA application

- Problem of singularities with high-dimensional data

$$S_W^{-1} S_B w = \lambda w$$

- If the data lies on a subspace, the within-class scatter matrix S_W will be singular.

$$S_W = \sum_{i=1}^{c} \sum_{j} (x_j^{(i)} - \mu_i)(x_j^{(i)} - \mu_i)^T$$



- Solution: find a non-degenerated subspace on the training set using the PCA and perform LDA on the data projected to this subspace first.

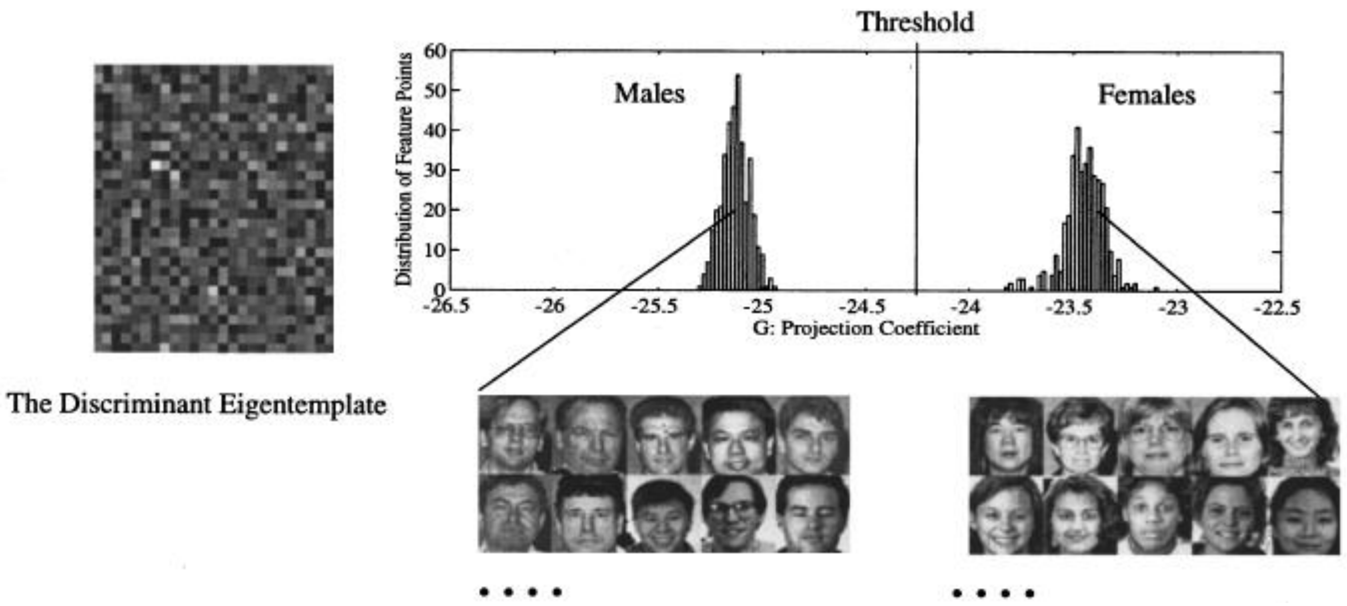
The LDA algorithm

```
function [W,Ms]=lda(X,c,n)
% X: input samples in columns, arranged by classes
% c, n: number of classes, number of samples per class
% W: LDA subspace basis vectors
% Ms: class means in the LDA subspace
SB=0; SW=0;
MM=mean(X')'; %overall mean
for i=1:c
  Ms(:,i)=mean(X(:,(i-1)*n+1:i*n)')';  %class means
  SB=SB+n*(Ms(:,i)-MM)*(Ms(:,i)-MM)';  % between class scatter m.
  for j=1:n % within class scatter matrix
    SW=SW+(X(:,(i-1)*n+j)-Ms(:,i))*(X(:,(i-1)*n+j)-Ms(:,i))';
  end;
end;
% the solution of a generalized eigenproblem:
[W, EIGD] = eigs(inv(SW)*SB, c-1,'LM',opts);
Ms=W'*Ms;  %map means into the LDA space
```

Example

- Male vs. Female faces
- Success rate in recognition:
  - Same persons: 95%
  - New persons: 92%

Projection into only a single most discriminative direction w.

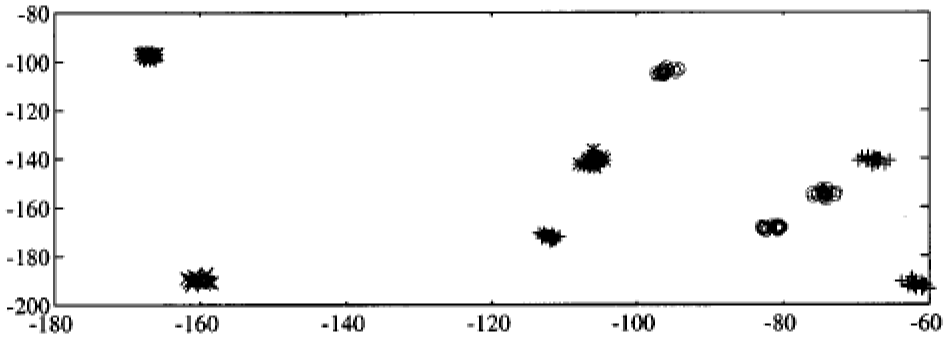


From K. Etemad, R. Chellapa, Discriminant analysis for recognition of human faces. J. Opt. Soc. Am. A,Vol. 14, No. 8, August 1997

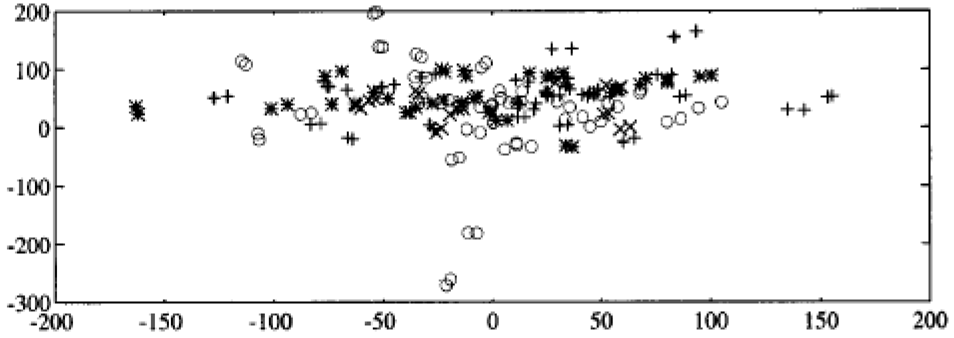
Example

- Images of ten persons projected onto the first two main directions using LDA and PCA:

LDA



PCA



From K. Etemad, R. Chellapa, Discriminant analysis for recognition of human faces. J. Opt. Soc. Am. A,Vol. 14, No. 8, August 1997

Many more subspace methods exist

- CCA, ICA, LLE, Robust variants, Kernel PCA, …
- Sparse reconstruction http://vcc.kaust.edu.sa/Pages/ICCV2013ShortCourse.aspx
- Deep learning era:
  - PCA equivalents : Autoecoders
    https://medium.com/swlh/introduction-to-autoencoders-56e5d60dad7f

References

- Ali Ghodsi, Dimensionality Reduction A Short Tutorial, 2006
- Gonzalez & Woods, Digital image processing, 2007
- David A. Forsyth, Jean Ponce, Computer Vision: A Modern Approach (2nd Edition), (prva izdaja dostopna na spletu)
- R. Szeliski,Computer Vision: Algorithms and Applications, 2010