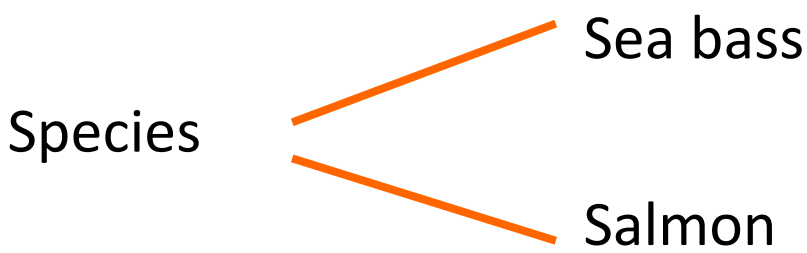
# Computer Vision
# CSC9010-005

Instructor: Dr. Edward Kim

# Classification in general

- “Sorting incoming Fish on a conveyor according to species using optical sensing”

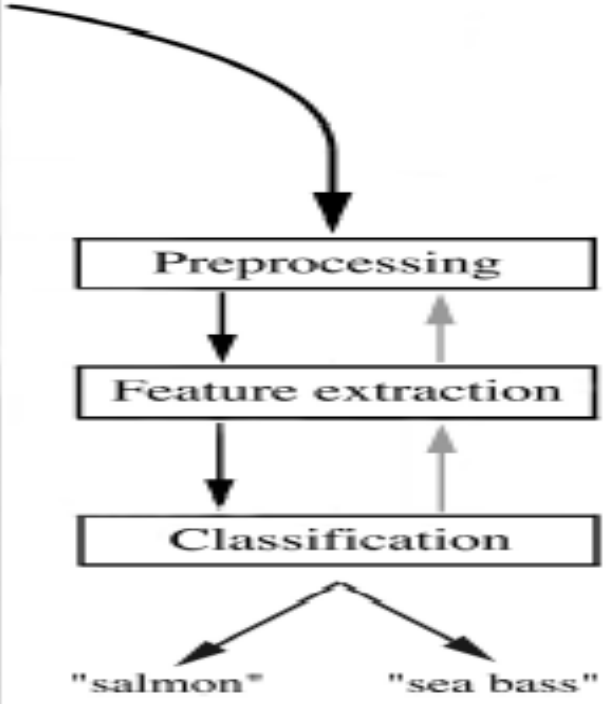


# What differentiates the two?

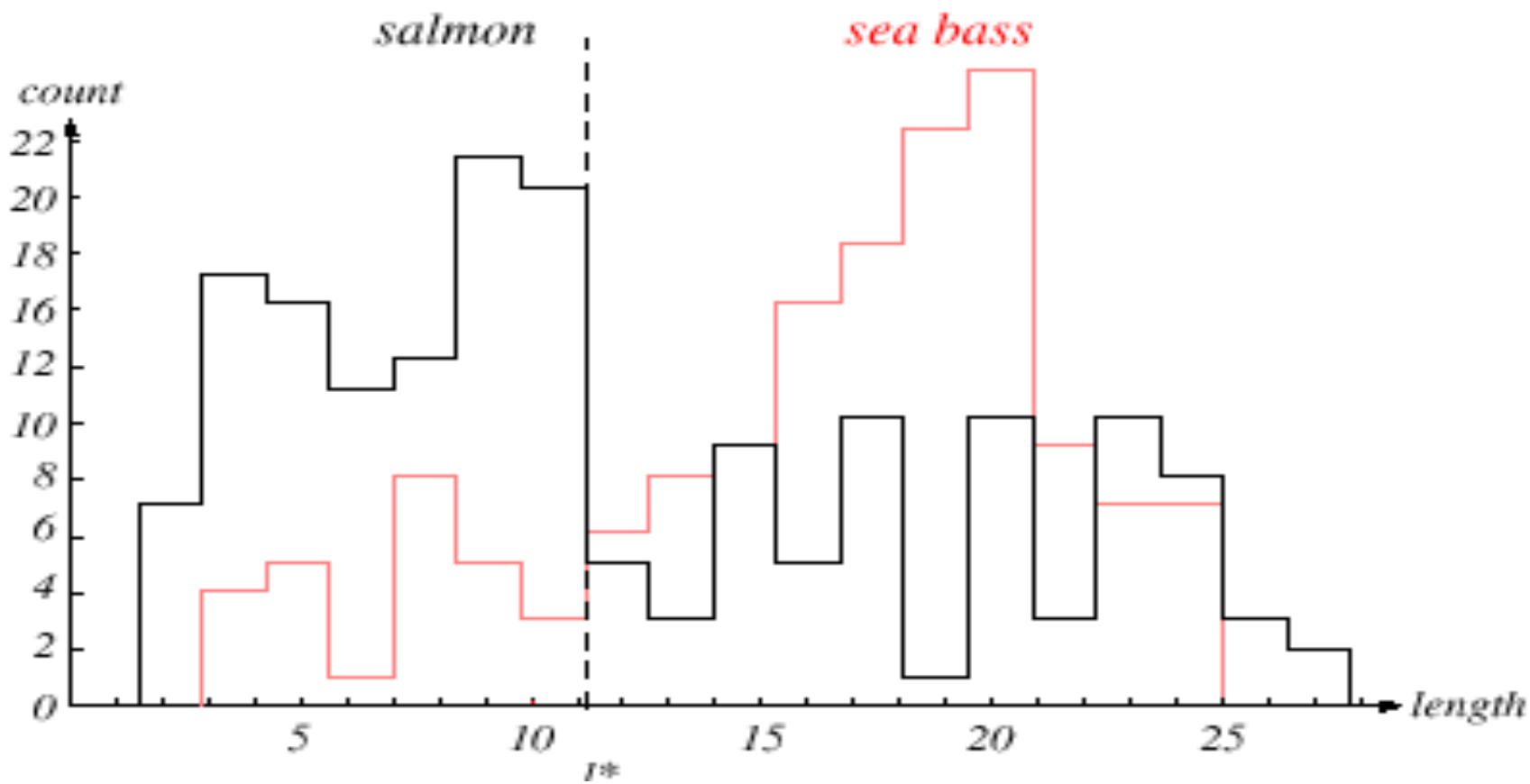
# Problem Analysis

- Set up a camera and take some sample images to extract features
  - Length
  - Lightness
  - Width
  - Number and shape of fins
  - Position of the mouth, etc…
  - This is the set of all suggested features to explore for use in our classifier!

- Preprocessing
  - Use a segmentation operation to isolate fishes from one another and from the background
- Information from a single fish is sent to a feature extractor whose purpose is to reduce the data by measuring certain features
- The features are passed to a classifier

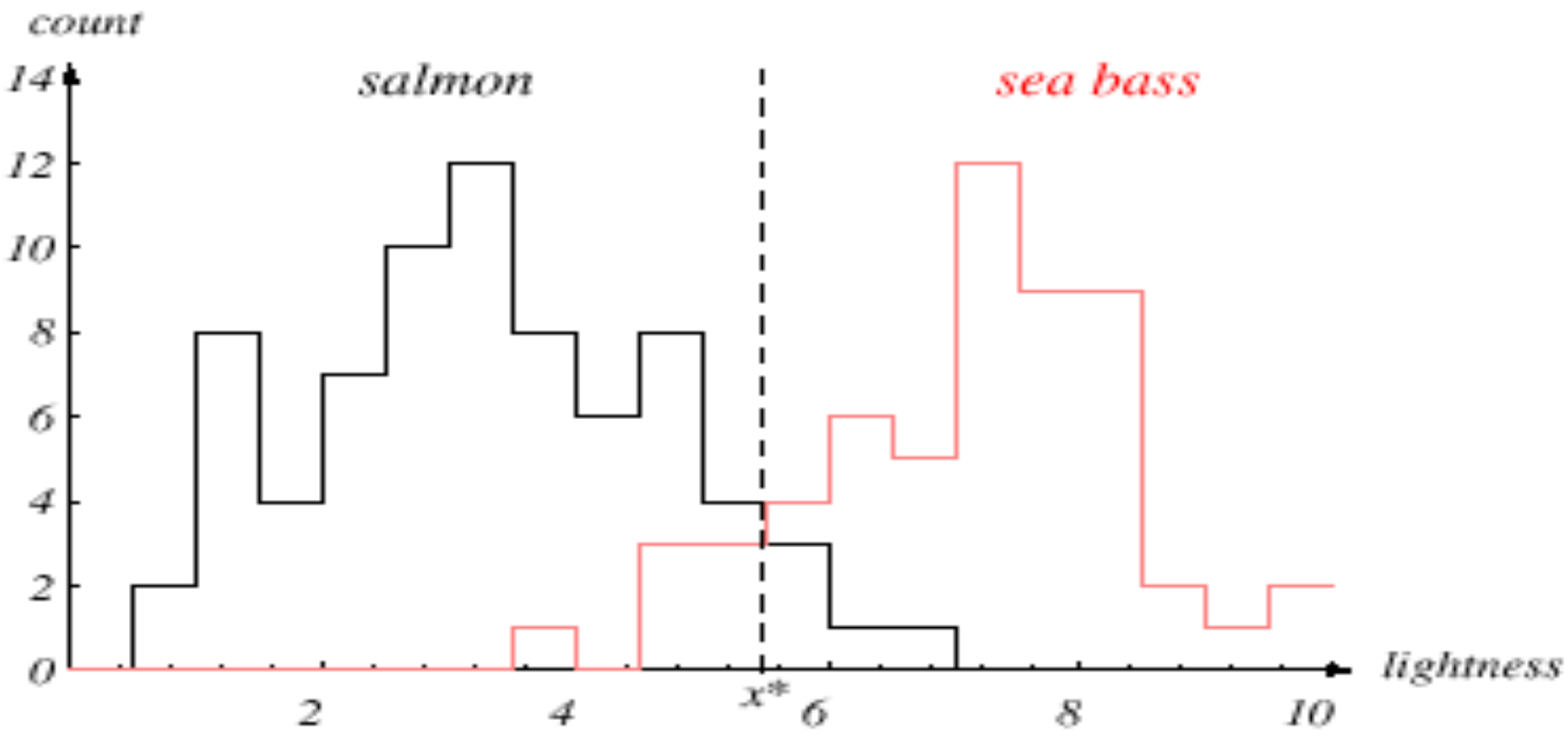

Preprocessing
Feature extraction
Classification
"salmon"
"sea bass"

- Classification
  - Select the length of the fish as a possible feature for discrimination


salmon
sea bass
count
22
20
18
16
12
10
8
6
4
2
0
5
10
15
20
25
length
l*

The length is a poor feature alone!

Select the lightness as a possible feature.


count
salmon
sea bass
14
12
10
8
6
4
2
0
2
4
x*
6
8
10
lightness

- Threshold decision boundary and cost relationship
  - Move our decision boundary toward smaller values of lightness in order to minimize the cost (reduce the number of sea bass that are classified salmon!)

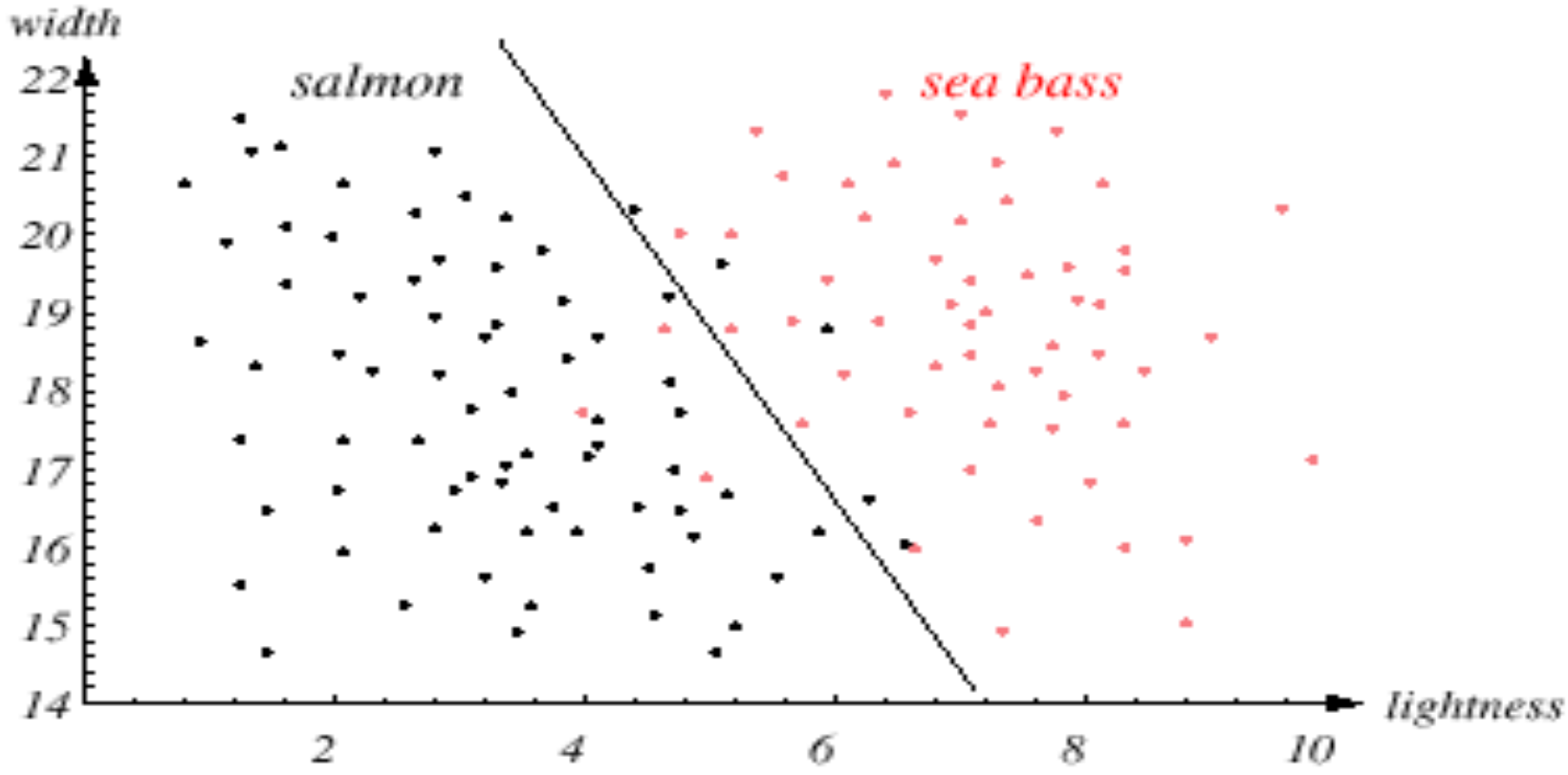
Task of decision theory

- Adopt the lightness and add the width of the fish

Fish → $x^T = [x_1, x_2]$

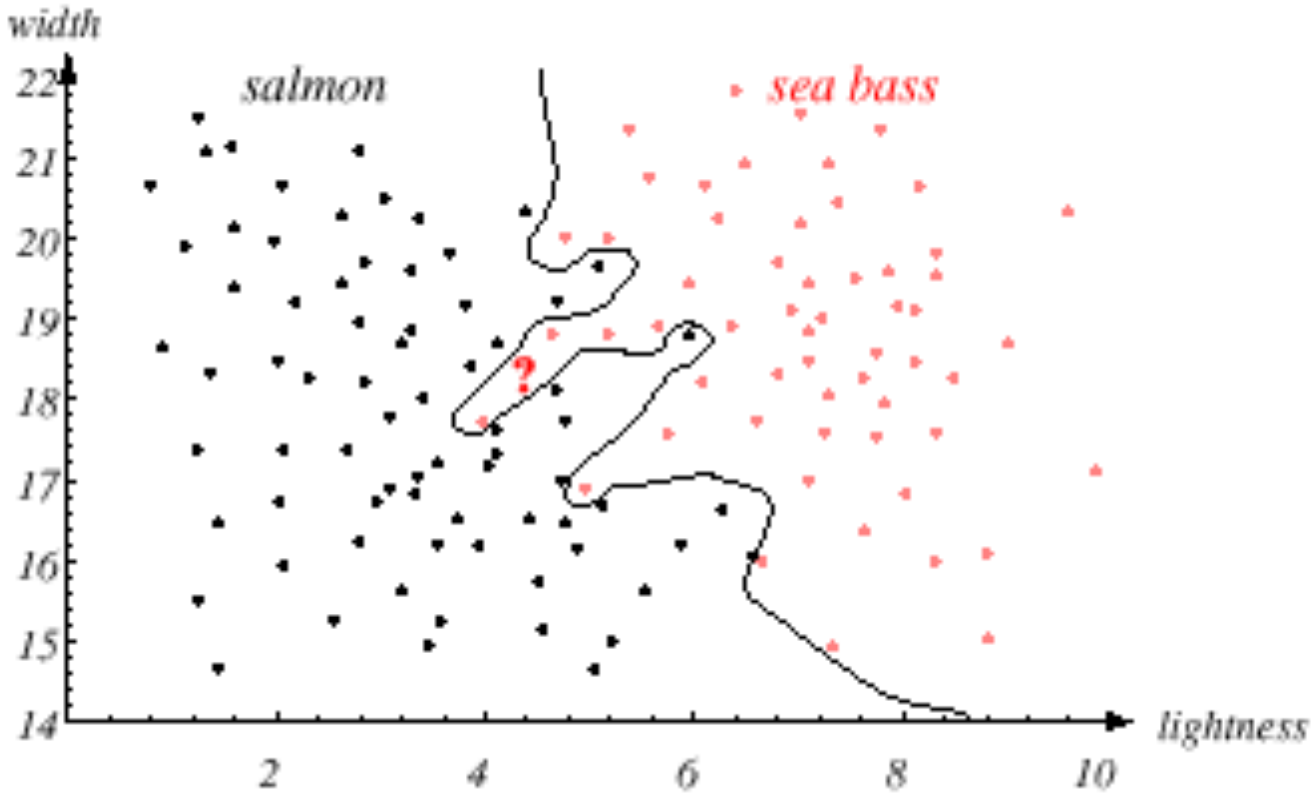
$x_1$ → **Lightness**

$x_2$ → **Width**


width
22
21
20
19
18
17
16
15
14
salmon
sea bass
2
4
6
8
10
lightness

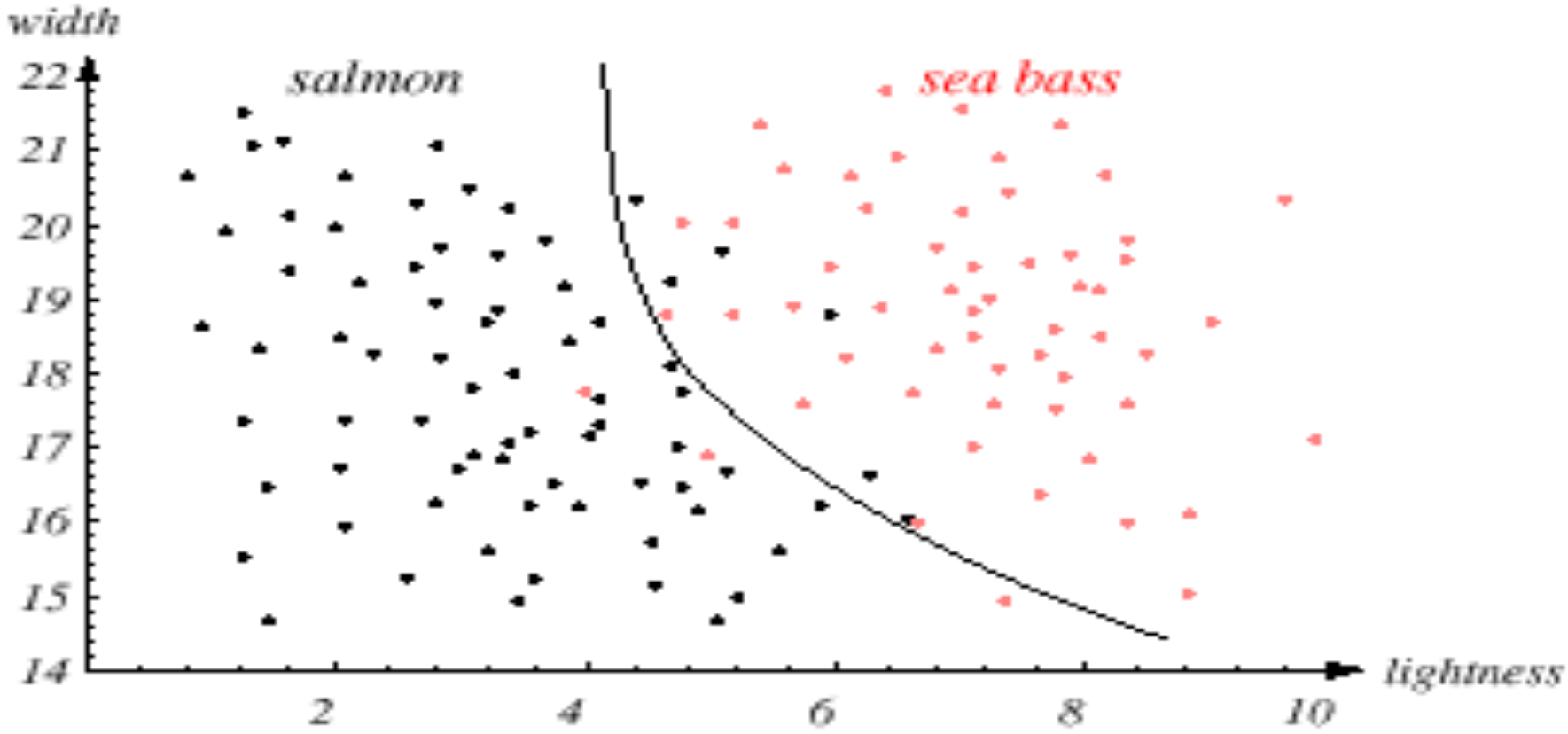
- We might add other features that are not correlated with the ones we already have. A precaution should be taken not to reduce the performance by adding such “noisy features”

- Ideally, the best decision boundary should be the one which provides an optimal performance such as in the following figure:


width
salmon
sea bass
?
22
21
20
19
18
17
16
15
14
2
4
6
8
10
lightness

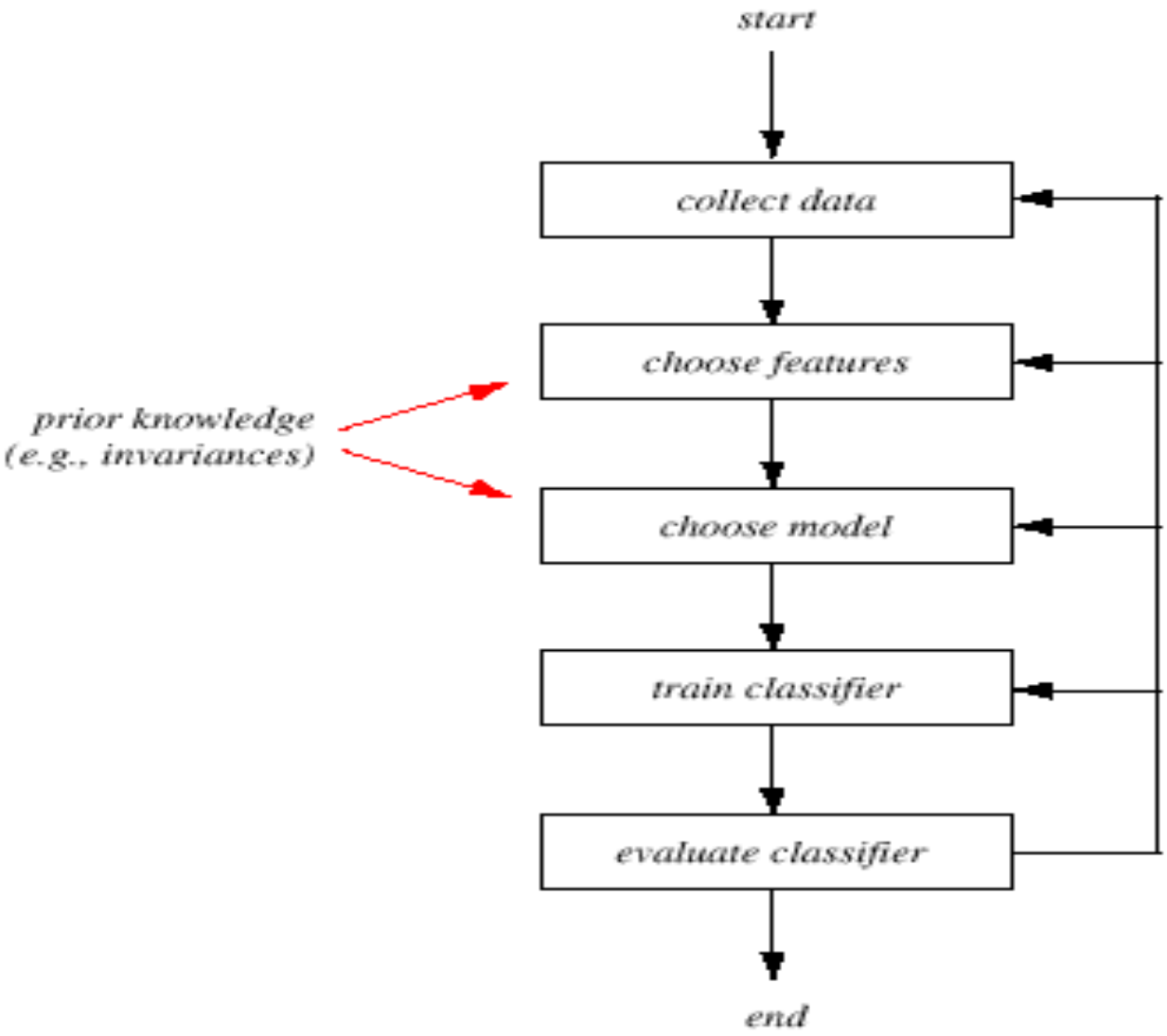
- However, our satisfaction is premature because the central aim of designing a classifier is to correctly classify novel input

Issue of generalization!


width
salmon
sea bass
22
21
20
19
18
17
16
15
14
2
4
6
8
10
lightness

# The Design Cycle

- Data collection
- Feature Choice
- Model Choice
- Training
- Evaluation
- Computational Complexity


start
collect data
choose features
prior knowledge
(e.g., invariances)
choose model
train classifier
evaluate classifier
end

- Data Collection
  - How do we know when we have collected an adequately large and representative set of examples for training and testing the system?

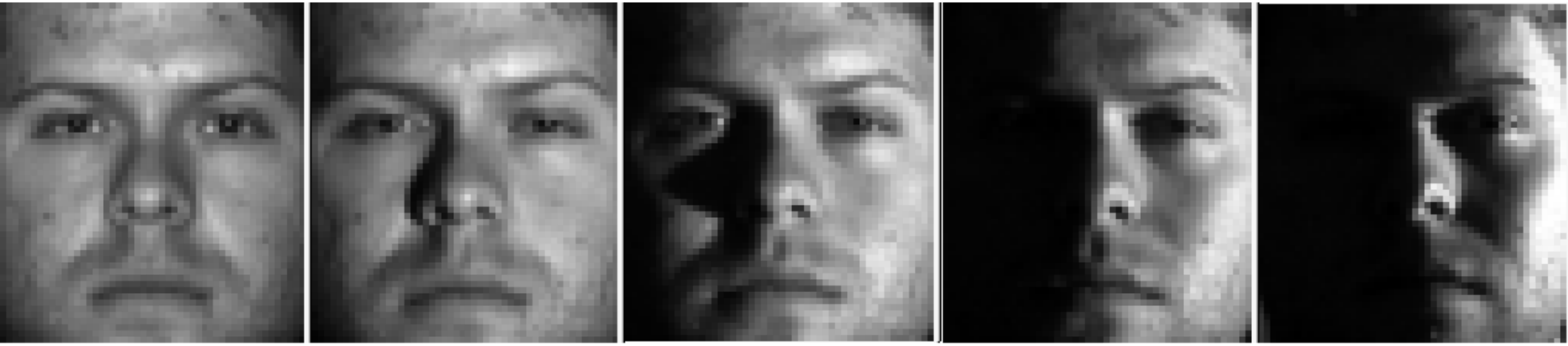
- Feature Choice
  - Depends on the characteristics of the problem domain.
  - (obsolete with DL) Find features that are: Simple to extract, invariant to irrelevant transformations, insensitive to noise, and useful in discriminating patterns in different categories.
  - (obsolete with DL) Prior knowledge plays a major role- how to combine prior knowledge and empirical data to find relevant and effective features?

- Model Choice
  - Unsatisfied with the performance of our fish classifier and want to jump to another class of model
    - Based on some function of the number and position of the fins, the color of the eyes, the weight, shape of the mouth …

- Training
  - Use data to determine the classifier. Many different procedures for training classifiers and choosing models

- Evaluation
  - Measure the error rate (or performance and switch from one set of features to another one

- Computational Complexity
  - What is the trade-off between computational ease and performance?
  - (How an algorithm scales as a function of the number of features, patterns or categories?)

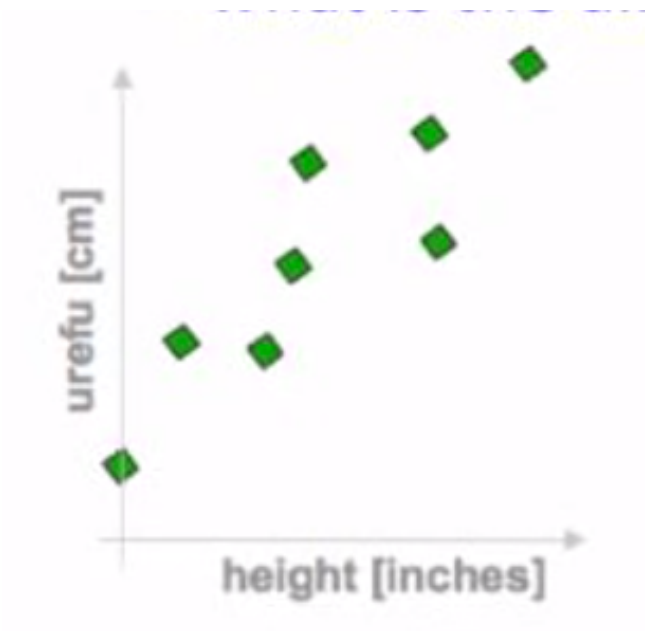


# Dimensionality

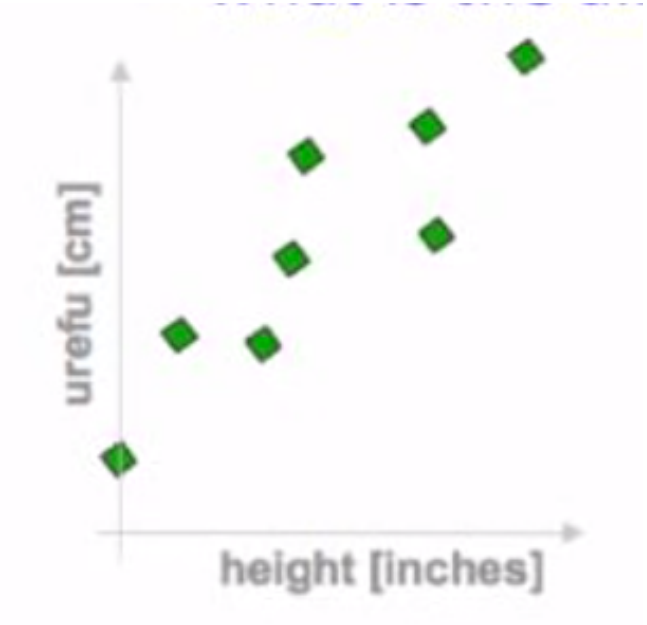
- Adding more and more features…

- Get a population, predict some properties
  - Instances represented as {urefu, height} pairs
  - What is the dimensionality of this data?

- Get a population, predict some properties
  - Instances represented as {urefu, height} pairs
  - What is the dimensionality of this data?

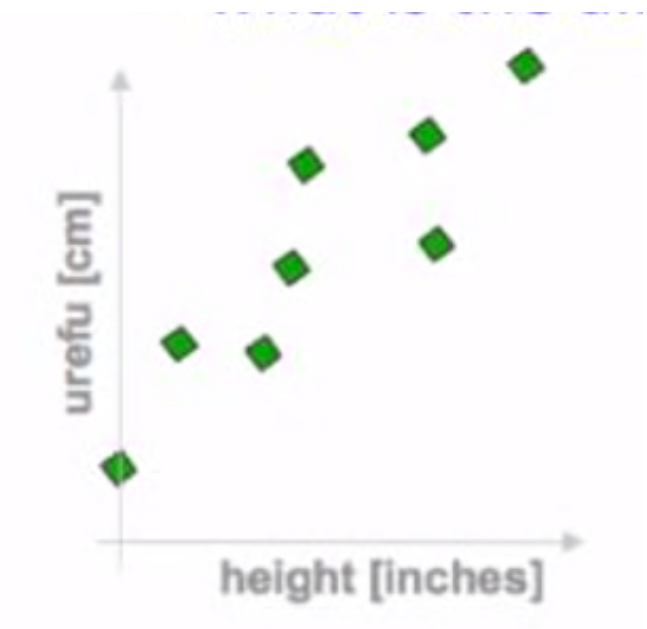


- Get a population, predict some properties
  - Instances represented as {urefu, height} pairs
  - What is the dimensionality of this data?




“height” = ‘urefu” in Swahili

- Get a population, predict some properties
  - Instances represented as {urefu, height} pairs
  - What is the dimensionality of this data?

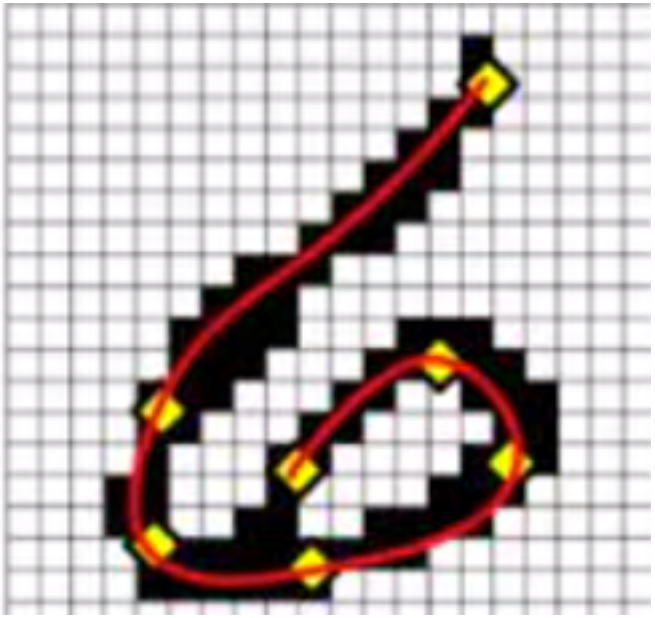



- Data points over time from different geographic areas over time:
  - $X_1$: # of skidding accidents
  - $X_2$: # of burst water pipes
  - $X_3$: snow-plow expenditures
  - $X_4$: # of school closures
  - $X_5$: # patients with heat stroke

# Curse of dimensionality

- Datasets typically high dimensional
- But the true dimensionality is often much lower

- Total dimensionality
  - 20x20 bitmap
  - 2 ^ 400 events



# Curse of dimensionality

- Datasets typically high dimensional
- But the true dimensionality is often much lower

- Total dimensionality
  - 20x20 bitmap
  - 2 ^ 400 events
  - True dimensionality is lower

# Dealing with high dimensionality

- Harris, SIFT, HOG reduce the space
  - Manual

- Or you can reduce the dimensions
  - Automatically

# Dimensionality Reduction

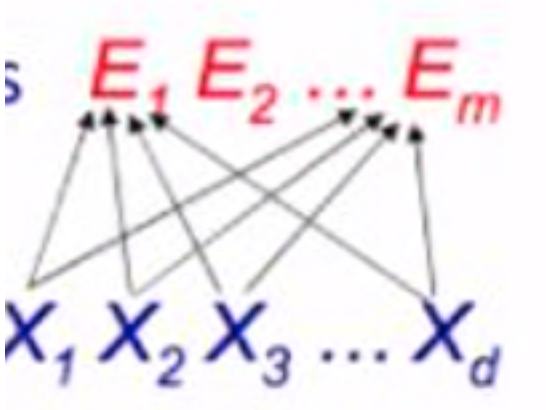
- Goal: represent the “numbers” or “faces” with fewer variables
  - Preserve the structure of the data
  - Be discriminative, only structure that affects class separability

- Feature selection
  - Pick a subset of dimensions, X1, X2, X3, X4
    - Use only X2, X4 (height, width)

# Feature extraction

- Construct a new set of dimensions

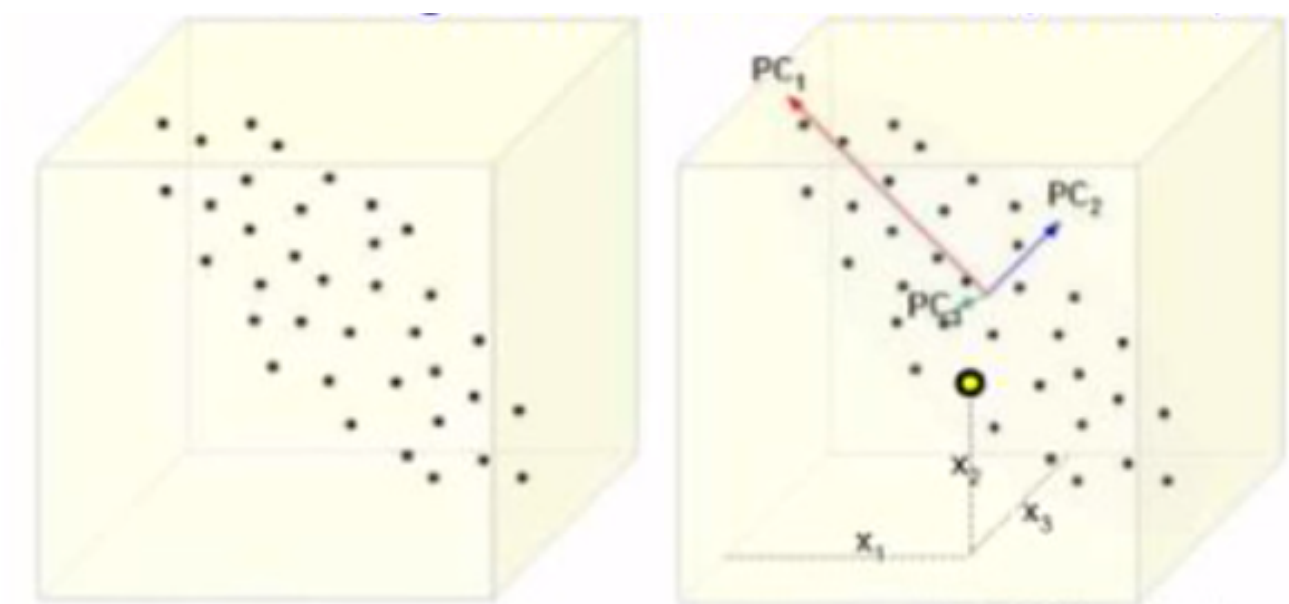
$$E_i = f(X_1 \ldots X_d)$$




- Linear combination of original

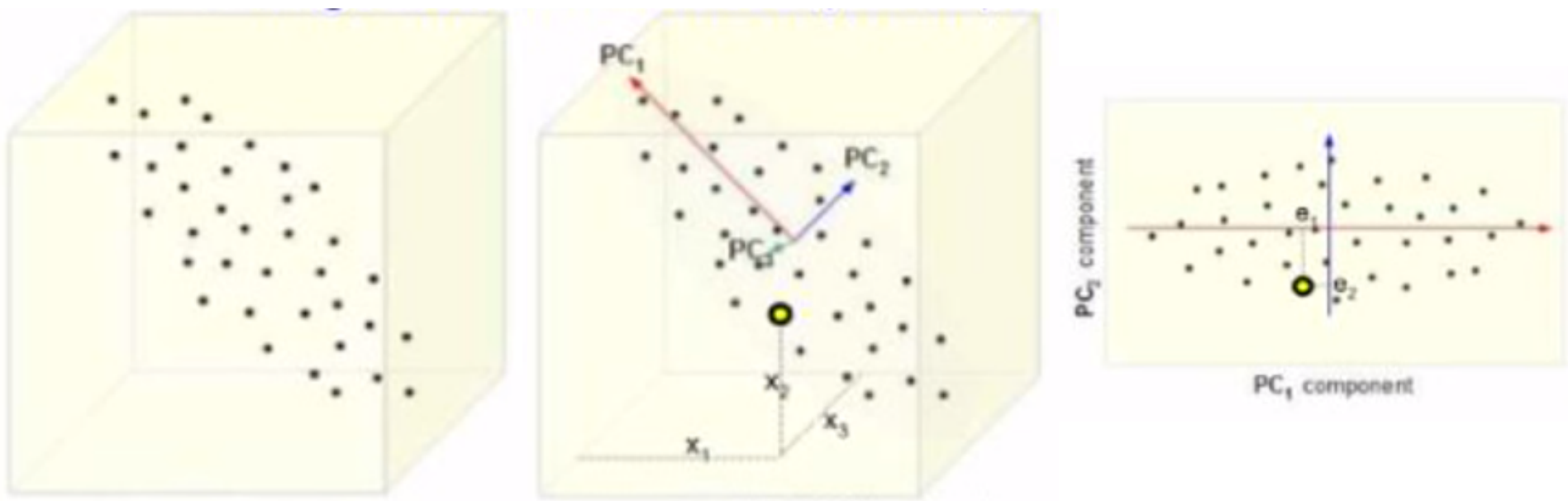
# Principal Component Analysis

- Defines a set of principal components
  - Pick a line of greatest variability in the data
  - Next perpendicular to first, greatest variability of what is left… until d (original dimensionality)



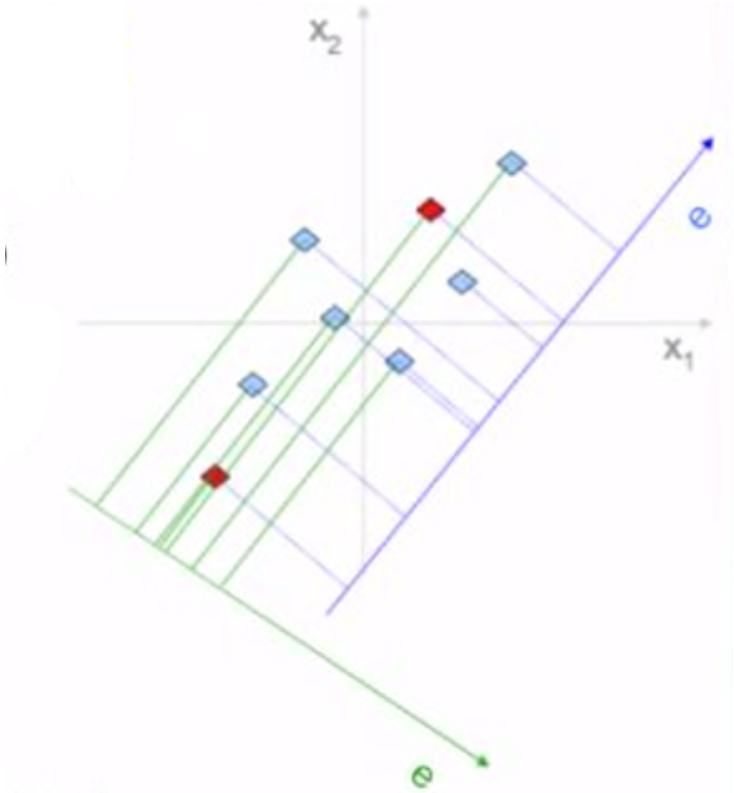
# Principal Component Analysis

- Defines a set of principal components
- M components become m new dimensions
  - Change coordinates of every data point in these dimensions



# Why greatest variability

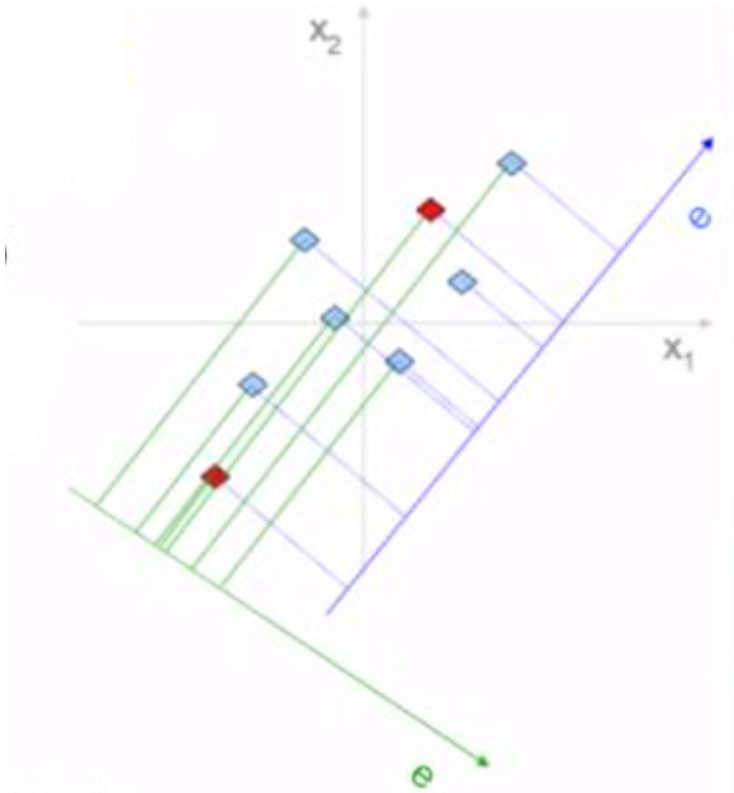
- Example: reduce 2-dimensional data to 1-d
- Pick e to maximize variability



# Why greatest variability

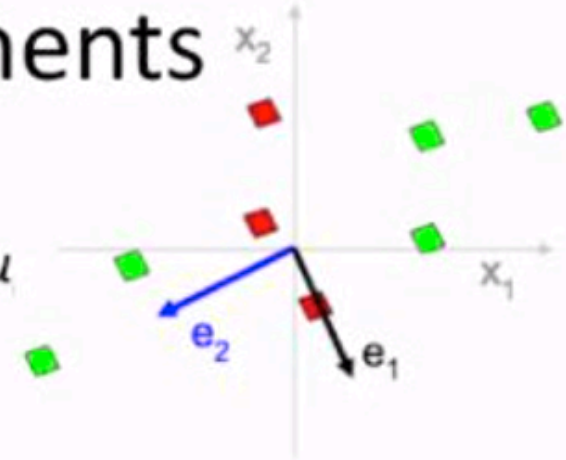
- Example: reduce 2-dimensional data to 1-d
- Pick e to maximize variability

- Preserve distances
  - In projected space
  - And original space

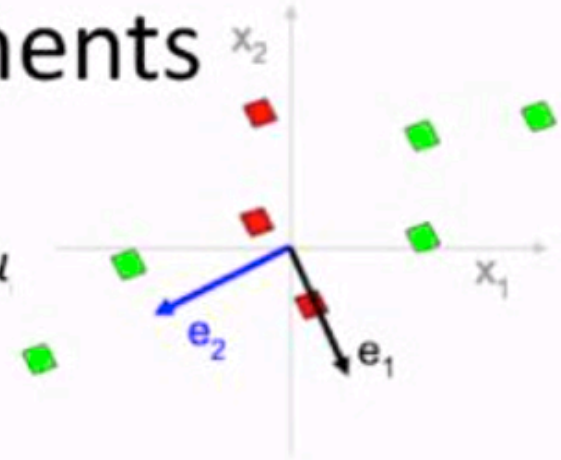


# Principal components

- "Center" the data at zero: $x_{i,a} = x_{i,a} - \mu$
  - subtract mean from each attribute

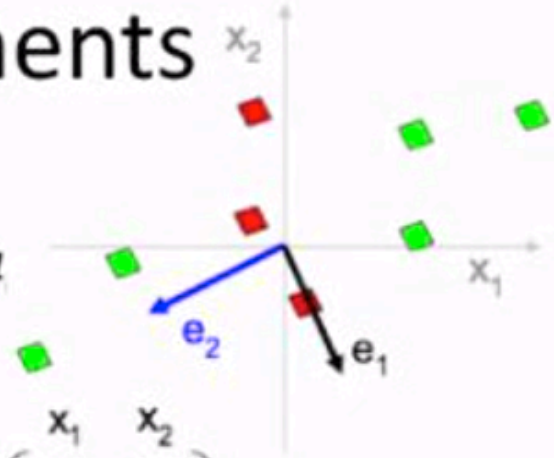


# Principal components




- "Center" the data at zero: $x_{i,a} = x_{i,a} - \mu$
  - subtract mean from each attribute
- Compute covariance matrix $\Sigma$
  - covariance of dimensions $x_1$ and $x_2$:
    - do $x_1$ and $x_2$ tend to increase together?
    - or does $x_2$ decrease as $x_1$ increases?

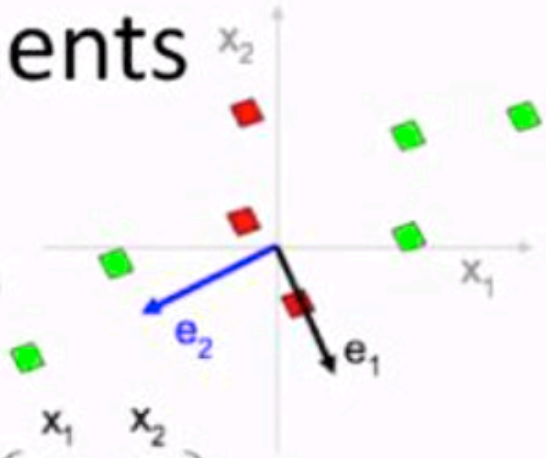
# Principal components




- "Center" the data at zero: $x_{i,a} = x_{i,a} - \mu$
  - subtract mean from each attribute
- Compute covariance matrix Σ
  - covariance of dimensions $x_1$ and $x_2$:
    - do $x_1$ and $x_2$ tend to increase together?
    - or does $x_2$ decrease as $x_1$ increases?

$$\begin{array}{c c} & \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{array}{c} x \\ x_2^1 \end{array} & \begin{bmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{bmatrix} \end{array}$$

# Principal components

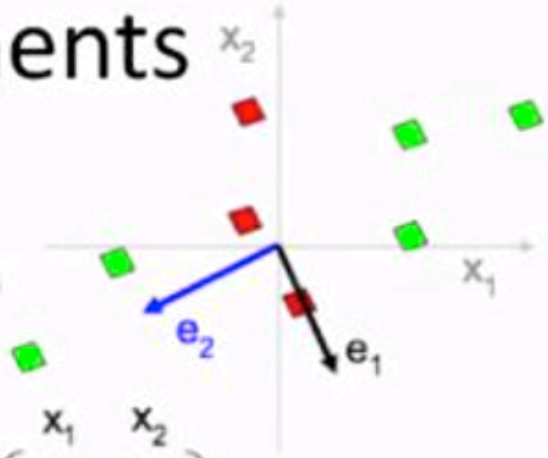



- "Center" the data at zero: $x_{i,a} = x_{i,a} - \mu$
  - subtract mean from each attribute
- Compute covariance matrix Σ
  - covariance of dimensions $x_1$ and $x_2$:
    - do $x_1$ and $x_2$ tend to increase together?
    - or does $x_2$ decrease as $x_1$ increases?

$$\begin{array}{c|cc} & x_1 & x_2 \\ \hline x & 2.0 & 0.8 \\ x_2^1 & 0.8 & 0.6 \end{array}$$

$$\mathrm{cov}(b,a) = \frac{1}{n}\sum_{i=1}^{n} x_{ib} x_{ia}$$

# Principal components



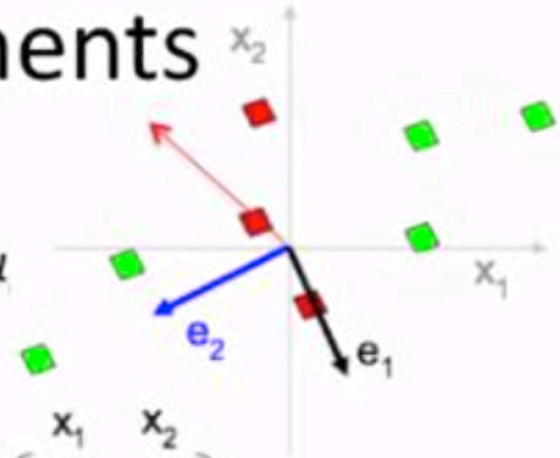

- "Center" the data at zero: $x_{i,a} = x_{i,a} - \mu$
  - subtract mean from each attribute
- Compute covariance matrix Σ
  - covariance of dimensions $x_1$ and $x_2$:
    - do $x_1$ and $x_2$ tend to increase together?
    - or does $x_2$ decrease as $x_1$ increases?

$$\begin{array}{c|cc} & x_1 & x_2 \\ \hline x & 2.0 & 0.8 \\ x_2^1 & 0.8 & 0.6 \end{array}$$

$$\mathrm{var}(a) = \frac{1}{n}\sum_{i=1}^{n} x_{ia}^2$$

$$\mathrm{cov}(b,a) = \frac{1}{n}\sum_{i=1}^{n} x_{ib}x_{ia}$$

# Principal components



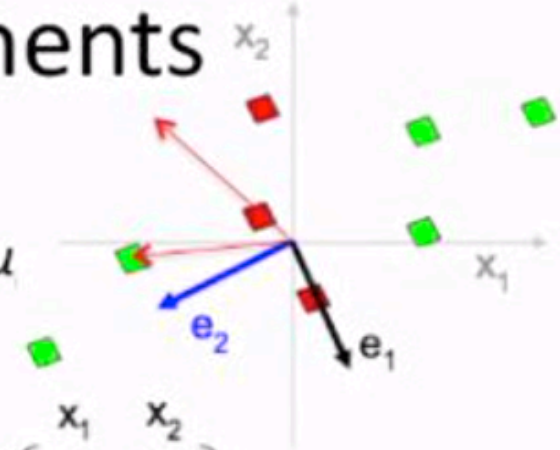

- "Center" the data at zero: $x_{i,a} = x_{i,a} - \mu$
  - subtract mean from each attribute
- Compute covariance matrix Σ
  - covariance of dimensions $x_1$ and $x_2$:
    - do $x_1$ and $x_2$ tend to increase together?
    - or does $x_2$ decrease as $x_1$ increases?
- Multiply a vector by Σ: $\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix}\begin{pmatrix} -1 \\ +1 \end{pmatrix} \rightarrow \begin{pmatrix} -1.2 \\ -0.2 \end{pmatrix}$ again
  - turns towards direction of variance

$$\begin{array}{c|cc} & x_1 & x_2 \\ \hline x & 2.0 & 0.8 \\ x_2^1 & 0.8 & 0.6 \end{array}$$

$$\mathrm{var}(a) = \frac{1}{n}\sum_{i=1}^{n} x_{ia}^2$$

$$\mathrm{cov}(b,a) = \frac{1}{n}\sum_{i=1}^{n} x_{ib} x_{ia}$$

# Principal components



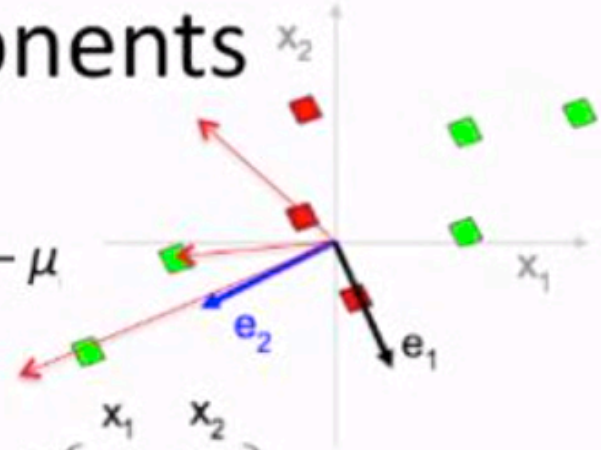

- "Center" the data at zero: $x_{i,a} = x_{i,a} - \mu$
  - subtract mean from each attribute
- Compute covariance matrix Σ
  - covariance of dimensions $x_1$ and $x_2$:
    - do $x_1$ and $x_2$ tend to increase together?
    - or does $x_2$ decrease as $x_1$ increases?
- Multiply a vector by Σ: $\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix}\begin{pmatrix} -1 \\ +1 \end{pmatrix} \rightarrow \begin{pmatrix} -1.2 \\ -0.2 \end{pmatrix}$ again
  - turns towards direction of variance

$$\begin{array}{c|cc} & x_1 & x_2 \\ \hline x & 2.0 & 0.8 \\ x_2^1 & 0.8 & 0.6 \end{array}$$

$$\mathrm{var}(a) = \frac{1}{n}\sum_{i=1}^{n} x_{ia}^2$$

$$\mathrm{cov}(b,a) = \frac{1}{n}\sum_{i=1}^{n} x_{ib}x_{ia}$$

# Principal components




- "Center" the data at zero: $x_{i,a} = x_{i,a} - \mu$
  - subtract mean from each attribute
- Compute covariance matrix Σ
  - covariance of dimensions $x_1$ and $x_2$:
    - do $x_1$ and $x_2$ tend to increase together?
    - or does $x_2$ decrease as $x_1$ increases?

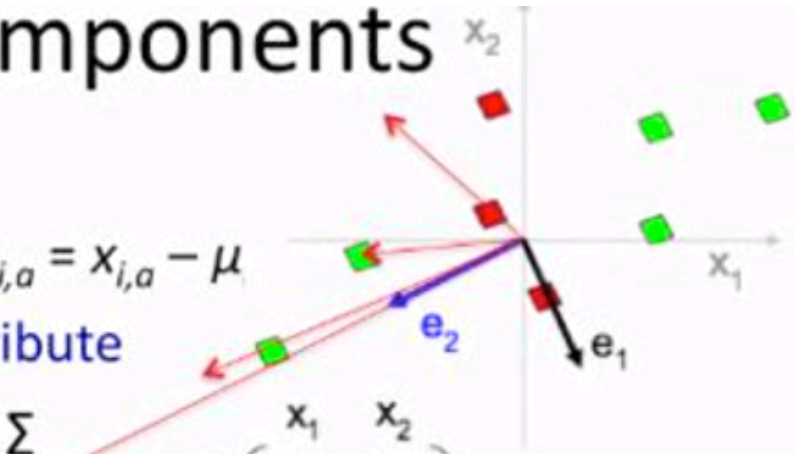
$$\begin{array}{c|cc} & x_1 & x_2 \\ \hline x & 2.0 & 0.8 \\ x_2^1 & 0.8 & 0.6 \end{array}$$

$$\mathrm{var}(a) = \frac{1}{n}\sum_{i=1}^{n} x_{ia}^2$$

$$\mathrm{cov}(b,a) = \frac{1}{n}\sum_{i=1}^{n} x_{ib}x_{ia}$$

- Multiply a vector by Σ: $\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix}\begin{pmatrix} -1 \\ +1 \end{pmatrix} \rightarrow \begin{pmatrix} -1.2 \\ -0.2 \end{pmatrix}$ again $\rightarrow \begin{pmatrix} -2.5 \\ -1.0 \end{pmatrix} \rightarrow \begin{pmatrix} -6.0 \\ -2.7 \end{pmatrix}$
  - turns towards direction of variance

# Principal components



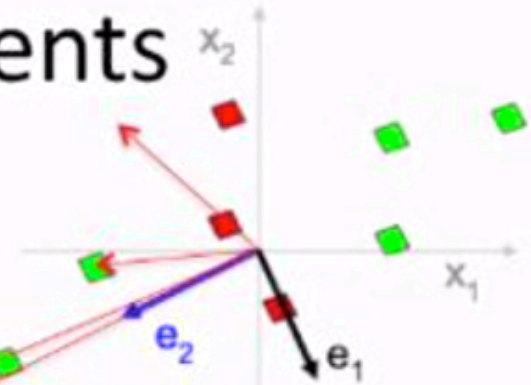

- "Center" the data at zero: $x_{i,a} = x_{i,a} - \mu$
  - subtract mean from each attribute
- Compute covariance matrix Σ
  - covariance of dimensions $x_1$ and $x_2$:
    - do $x_1$ and $x_2$ tend to increase together?
    - or does $x_2$ decrease as $x_1$ increases?
- Multiply a vector by Σ: $\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix}\begin{pmatrix} -1 \\ +1 \end{pmatrix} \rightarrow \begin{pmatrix} -1.2 \\ -0.2 \end{pmatrix}$ again $\rightarrow \begin{pmatrix} -2.5 \\ -1.0 \end{pmatrix} \rightarrow \begin{pmatrix} -6.0 \\ -2.7 \end{pmatrix}$
  - turns towards direction of variance

$$\begin{array}{c|cc} & x_1 & x_2 \\ \hline x & 2.0 & 0.8 \\ x_2^1 & 0.8 & 0.6 \end{array}$$

$$\mathrm{var}(a) = \frac{1}{n}\sum_{i=1}^{n} x_{ia}^2$$

$$\mathrm{cov}(b,a) = \frac{1}{n}\sum_{i=1}^{n} x_{ib}x_{ia}$$

# Principal components




- "Center" the data at zero: $x_{i,a} = x_{i,a} - \mu$
  - subtract mean from each attribute
- Compute covariance matrix Σ
  - covariance of dimensions $x_1$ and $x_2$:
    - do $x_1$ and $x_2$ tend to increase together?
    - or does $x_2$ decrease as $x_1$ increases?
- Multiply a vector by Σ: $\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix}\begin{pmatrix} -1 \\ +1 \end{pmatrix} \rightarrow \begin{pmatrix} -1.2 \\ -0.2 \end{pmatrix}$ again $\rightarrow \begin{pmatrix} -2.5 \\ -1.0 \end{pmatrix} \rightarrow \begin{pmatrix} -6.0 \\ -2.7 \end{pmatrix} \rightarrow \begin{pmatrix} -14.1 \\ -6.4 \end{pmatrix} \rightarrow \begin{pmatrix} -33.3 \\ -15.1 \end{pmatrix}$

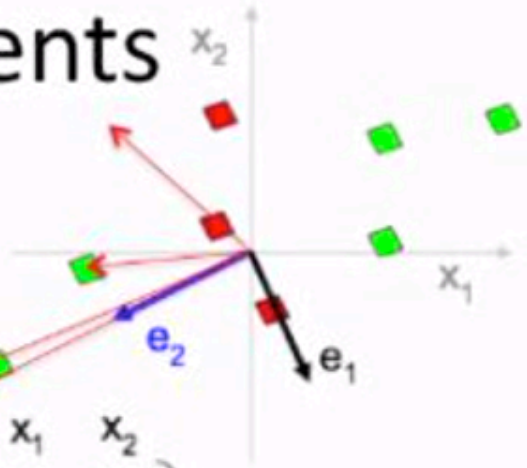
  slope: 0.400 0.450 0.454 0.454
  - turns towards direction of variance

$$\begin{array}{c|cc} & x_1 & x_2 \\ \hline x & 2.0 & 0.8 \\ x_2^1 & 0.8 & 0.6 \end{array}$$

$$\mathrm{var}(a) = \frac{1}{n}\sum_{i=1}^{n} x_{ia}^2$$

$$\mathrm{cov}(b,a) = \frac{1}{n}\sum_{i=1}^{n} x_{ib}x_{ia}$$

# Principal components




- "Center" the data at zero: $x_{i,a} = x_{i,a} - \mu$
  - subtract mean from each attribute
- Compute covariance matrix Σ
  - covariance of dimensions $x_1$ and $x_2$:
    - do $x_1$ and $x_2$ tend to increase together?
    - or does $x_2$ decrease as $x_1$ increases?

$$\begin{array}{c|cc} & x_1 & x_2 \\ \hline x & 2.0 & 0.8 \\ x_2^1 & 0.8 & 0.6 \end{array}$$

$$\mathrm{var}(a) = \frac{1}{n}\sum_{i=1}^{n} x_{ia}^2$$

$$\mathrm{cov}(b,a) = \frac{1}{n}\sum_{i=1}^{n} x_{ib} x_{ia}$$

- Multiply a vector by Σ: $\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix}\begin{pmatrix} -1 \\ +1 \end{pmatrix} \to \begin{pmatrix} -1.2 \\ -0.2 \end{pmatrix}$ again $\to \begin{pmatrix} -2.5 \\ -1.0 \end{pmatrix} \to \begin{pmatrix} -6.0 \\ -2.7 \end{pmatrix} \to \begin{pmatrix} -14.1 \\ -6.4 \end{pmatrix} \to \begin{pmatrix} -33.3 \\ -15.1 \end{pmatrix}$

  slope: 0.400 0.450 0.454 0.454
  - turns towards direction of variance
- Want vectors **e** which aren't turned: Σ **e** = λ **e**
  - e … eigenvectors of Σ, λ … corresponding eigenvalues
  - principal components = eigenvectors w. largest eigenvalues

# Finding Principal Components

1. find eigenvalues by solving: $\det(\Sigma - \lambda I) = 0$

$$\det\begin{pmatrix} 2.0-\lambda & 0.8 \\ 0.8 & 0.6-\lambda \end{pmatrix} = (2-\lambda)(0.6-\lambda) - (0.8)(0.8) = \lambda^2 - 2.6\lambda + 0.56 = 0$$

$$\{\lambda_1, \lambda_2\} = \tfrac{1}{2}\left(2.6 \pm \sqrt{2.6^2 - 4 * 0.56}\right) = \{2.36, 0.23\}$$

2. find $i^{th}$ eigenvector by solving: $\Sigma\, \mathbf{e}_i = \lambda_i\, \mathbf{e}_i$

$$\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix}\begin{pmatrix} e_{1,1} \\ e_{1,2} \end{pmatrix} = 2.36\begin{pmatrix} e_{1,1} \\ e_{1,2} \end{pmatrix} \Rightarrow \begin{matrix} 2.0e_{1,1} + 0.8e_{1,2} = 2.36e_{1,1} \\ 0.8e_{1,1} + 0.6e_{1,2} = 2.36e_{1,2} \end{matrix} \Rightarrow e_{1,1} = 2.2e_{1,2}$$

$$\Downarrow$$

$$e_1 \sim \begin{bmatrix} 2.2 \\ 1 \end{bmatrix}$$

$$\Downarrow$$

want: $\|e_1\| = 1$

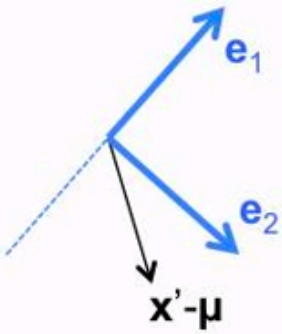
$$\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix}\begin{pmatrix} e_{2,1} \\ e_{2,2} \end{pmatrix} = 0.23\begin{pmatrix} e_{2,1} \\ e_{2,2} \end{pmatrix} \Rightarrow e_2 = \begin{bmatrix} -0.41 \\ 0.91 \end{bmatrix}$$

$$e_1 = \begin{bmatrix} 0.91 \\ 0.41 \end{bmatrix}$$

slope: 0.454

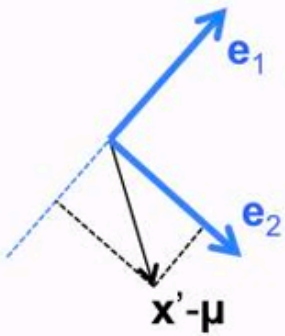
# Project data points to new dimensions

- $\mathbf{e}_1 \ldots \mathbf{e}_m$ are new dimension vectors
- Have instance $\mathbf{x} = \{x_1 \ldots x_d\}$ (original coordinates)
- Want new coordinates $\mathbf{x}' = \{x'_1 \ldots x'_m\}$:
  1. "center" the instance (subtract the mean): $\mathbf{x}'-\boldsymbol{\mu}$
  2. "project" to each dimension: $(\mathbf{x}'-\boldsymbol{\mu})^T\mathbf{e}_j$ for j=1…m



# Projecting to new dimensions

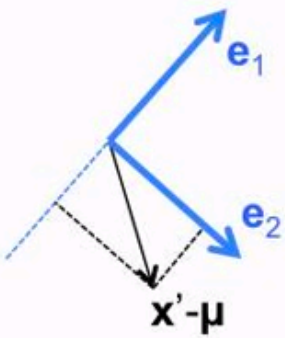
- $\mathbf{e}_1 \ldots \mathbf{e}_m$ are new dimension vectors
- Have instance $\mathbf{x} = \{x_1 \ldots x_d\}$ (original coordinates)
- Want new coordinates $\mathbf{x}' = \{x'_1 \ldots x'_m\}$:
  1. "center" the instance (subtract the mean): $\mathbf{x}'-\boldsymbol{\mu}$
  2. "project" to each dimension: $(\mathbf{x}'-\boldsymbol{\mu})^T\mathbf{e}_j$ for j=1…m




$$(\vec{x} - \vec{\mu}) = \begin{bmatrix} (x_1 - \mu_1) & (x_2 - \mu_2) & \cdots & (x_d - \mu_d) \end{bmatrix}$$

# Projecting to new dimensions

- $\mathbf{e}_1 \ldots \mathbf{e}_m$ are new dimension vectors
- Have instance $\mathbf{x} = \{x_1 \ldots x_d\}$ (original coordinates)
- Want new coordinates $\mathbf{x}' = \{x'_1 \ldots x'_m\}$:
  1. "center" the instance (subtract the mean): $\mathbf{x}'-\boldsymbol{\mu}$
  2. "project" to each dimension: $(\mathbf{x}'-\boldsymbol{\mu})^T\mathbf{e}_j$ for j=1…m

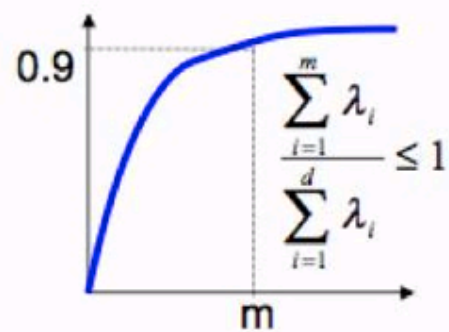



$$(\vec{x}-\vec{\mu}) = \begin{bmatrix}(x_1-\mu_1) & (x_2-\mu_2) & \cdots & (x_d-\mu_d)\end{bmatrix}$$

$$\begin{bmatrix} x_1' \\ x_2' \\ \vdots \\ x_m' \end{bmatrix} = \begin{bmatrix} (\vec{x}-\vec{\mu})^T\vec{e}_1 \\ (\vec{x}-\vec{\mu})^T\vec{e}_2 \\ \vdots \\ (\vec{x}-\vec{\mu})^T\vec{e}_m \end{bmatrix} = \begin{bmatrix} (x_1-\mu_1)e_{1,1}+(x_2-\mu_2)e_{1,2}+\cdots+(x_d-\mu_d)e_{1,d} \\ (x_1-\mu_1)e_{2,1}+(x_2-\mu_2)e_{2,2}+\cdots+(x_d-\mu_d)e_{2,d} \\ \vdots \\ (x_1-\mu_1)e_{m,1}+(x_2-\mu_2)e_{m,2}+\cdots+(x_d-\mu_d)e_{m,d} \end{bmatrix}$$

# How many eigenvectors

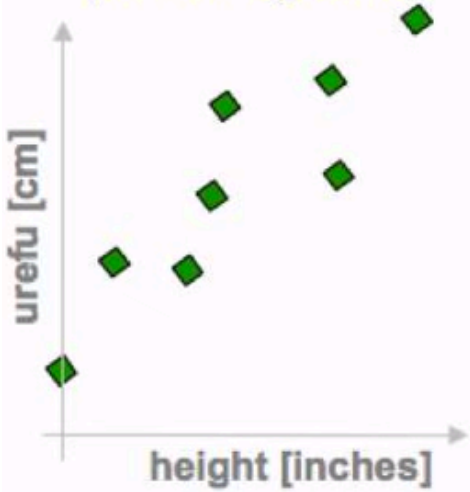
- Have: eigenvectors $\mathbf{e}_1 \ldots \mathbf{e}_d$ want: $m << d$
- Proved: eigenvalue $\lambda_i$ = variance along $\mathbf{e}_i$
- Pick $\mathbf{e}_i$ that "explain" the most variance
  - sort eigenvectors s.t. $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d$
  - pick first m eigenvectors which explain 90% or the total variance
    - typical threshold values: 0.9 or 0.95



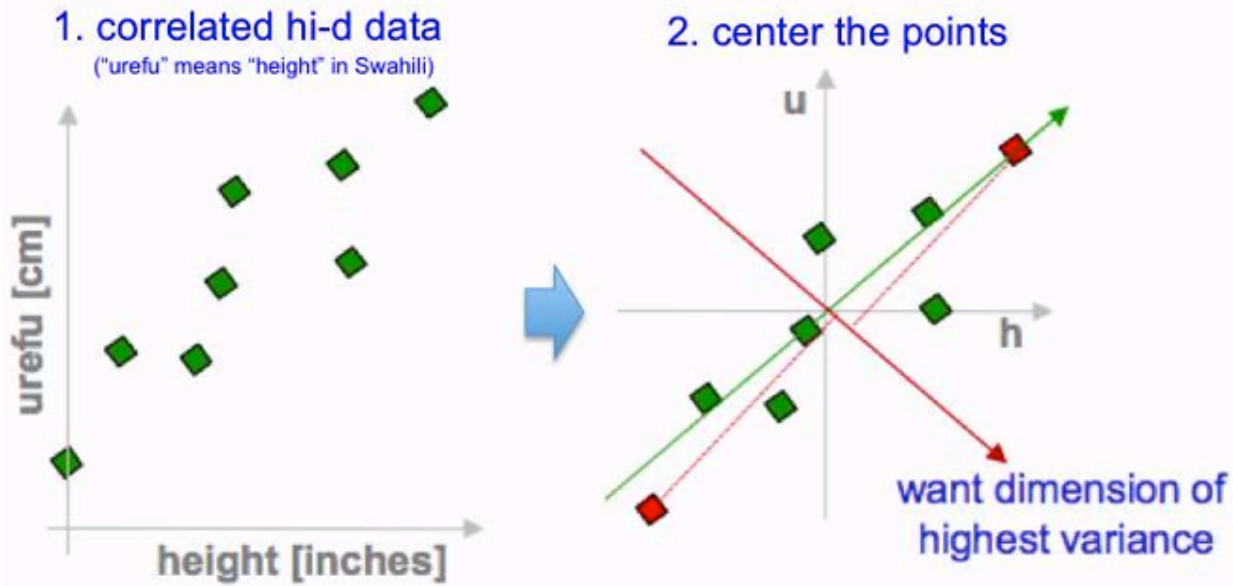
# PCA in a nutshell

1. correlated hi-d data

("urefu" means "height" in Swahili)



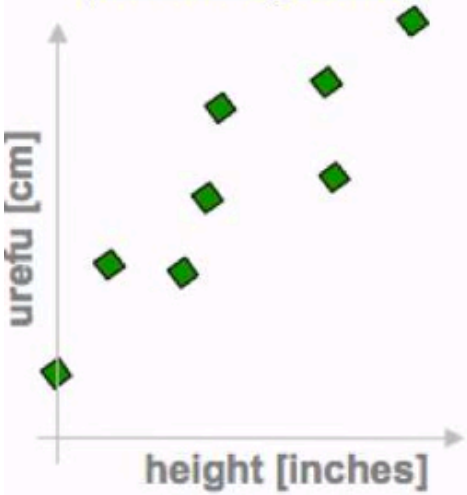
# PCA in a nutshell



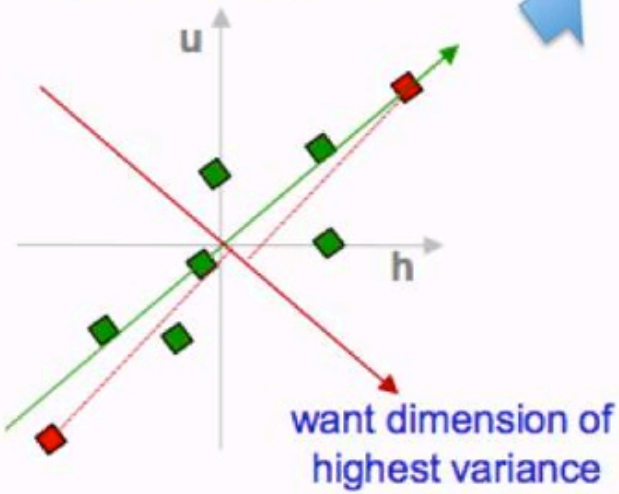
# PCA in a nutshell

1. correlated hi-d data

("urefu" means "height" in Swahili)




2. center the points



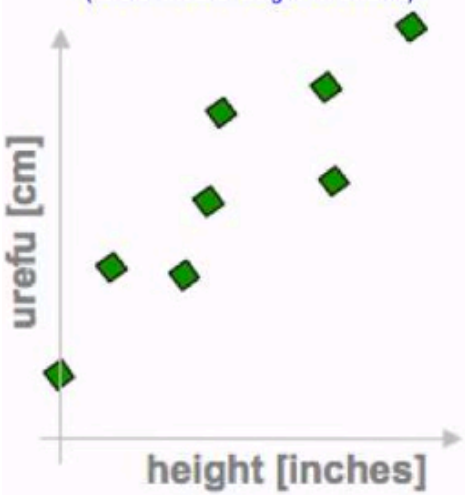

3. compute covariance matrix

$$\begin{array}{c|cc} & h & u \\ \hline h & 2.0 & 0.8 \\ u & 0.8 & 0.6 \end{array} \rightarrow \mathrm{cov}(h,u) = \frac{1}{n}\sum_{i=1}^{n} h_i u_i$$

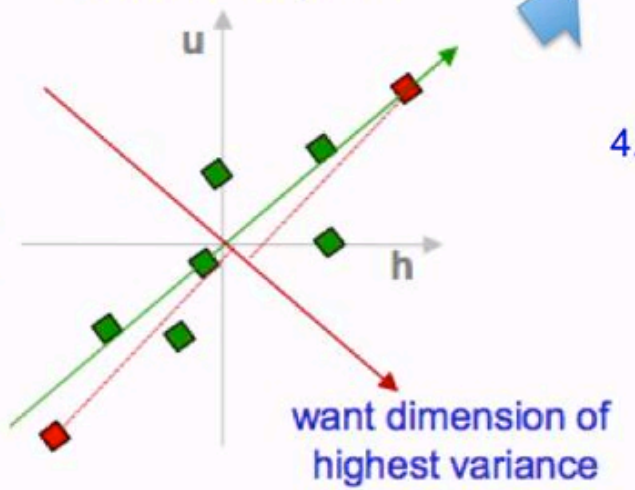
# PCA in a nutshell

1. correlated hi-d data

("urefu" means "height" in Swahili)




2. center the points




3. compute covariance matrix

$$\begin{array}{c|cc} & h & u \\ \hline h & 2.0 & 0.8 \\ u & 0.8 & 0.6 \end{array} \rightarrow \mathrm{cov}(h,u) = \frac{1}{n}\sum_{i=1}^{n} h_i u_i$$

4. eigenvectors + eigenvalues

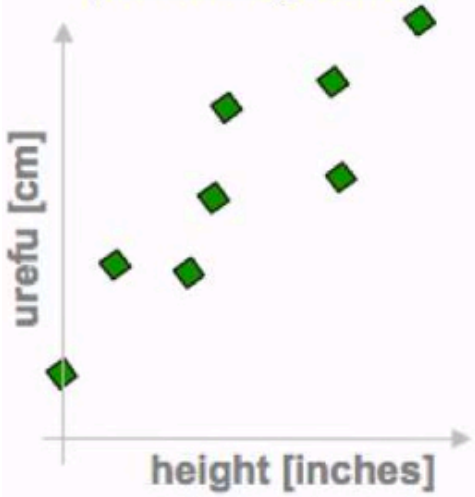
$$\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix} \begin{pmatrix} e_h \\ e_u \end{pmatrix} = \lambda_e \begin{pmatrix} e_h \\ e_u \end{pmatrix}$$

$$\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix} \begin{pmatrix} f_h \\ f_u \end{pmatrix} = \lambda_f \begin{pmatrix} f_h \\ f_u \end{pmatrix}$$

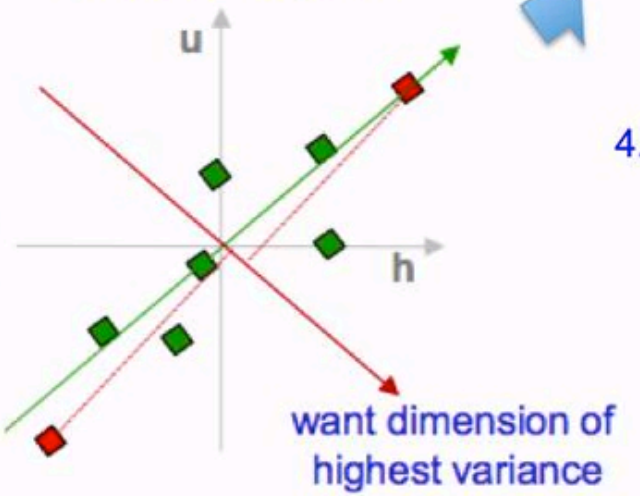
# PCA in a nutshell

1. correlated hi-d data

("urefu" means "height" in Swahili)




2. center the points




3. compute covariance matrix

$$\begin{array}{c|cc} & h & u \\ \hline h & 2.0 & 0.8 \\ u & 0.8 & 0.6 \end{array} \rightarrow \mathrm{cov}(h,u) = \frac{1}{n}\sum_{i=1}^{n} h_i u_i$$

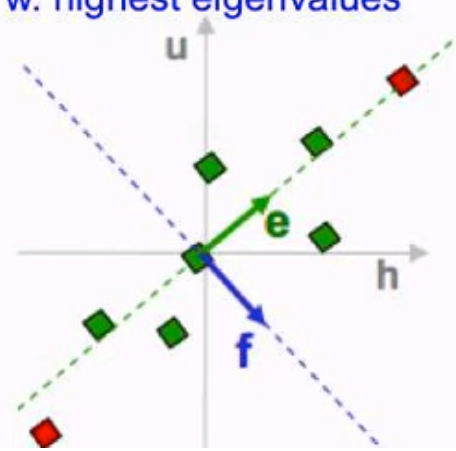
4. eigenvectors + eigenvalues

$$\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix} \begin{pmatrix} e_h \\ e_u \end{pmatrix} = \lambda_e \begin{pmatrix} e_h \\ e_u \end{pmatrix}$$

$$\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix} \begin{pmatrix} f_h \\ f_u \end{pmatrix} = \lambda_f \begin{pmatrix} f_h \\ f_u \end{pmatrix}$$

```
eig(cov(data))
```

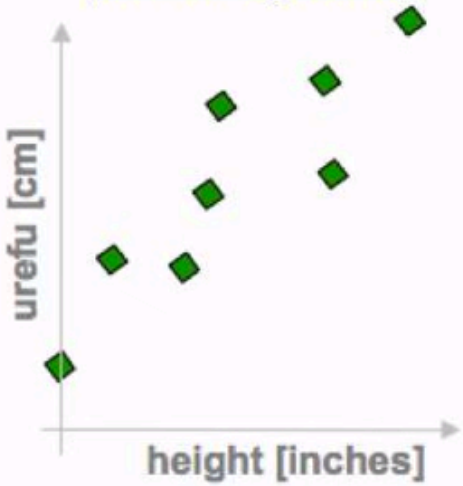
5. pick m<d eigenvectors w. highest eigenvalues



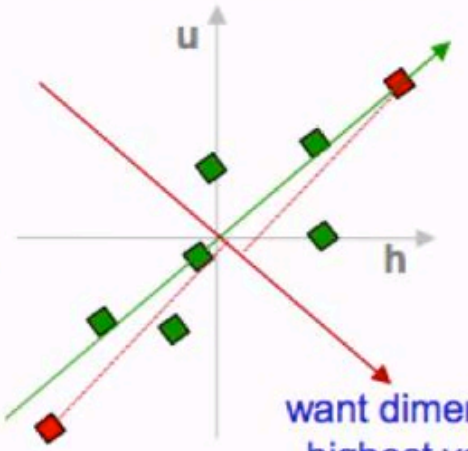
# PCA in a nutshell

1. correlated hi-d data

("urefu" means "height" in Swahili)




2. center the points




want dimension of highest variance

3. compute covariance matrix

$$\begin{array}{c|cc} & h & u \\ \hline h & 2.0 & 0.8 \\ u & 0.8 & 0.6 \end{array} \rightarrow \mathrm{cov}(h,u) = \frac{1}{n}\sum_{i=1}^{n} h_i u_i$$

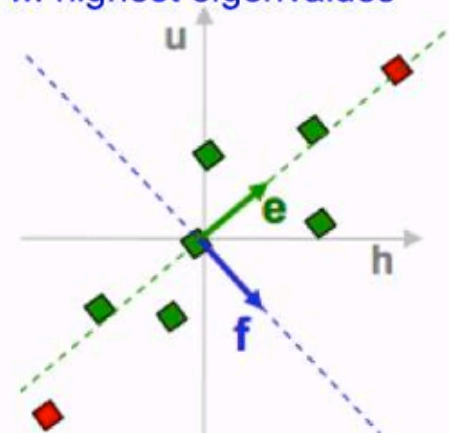
4. eigenvectors + eigenvalues

$$\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix} \begin{pmatrix} e_h \\ e_u \end{pmatrix} = \lambda_e \begin{pmatrix} e_h \\ e_u \end{pmatrix}$$

$$\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix} \begin{pmatrix} f_h \\ f_u \end{pmatrix} = \lambda_f \begin{pmatrix} f_h \\ f_u \end{pmatrix}$$

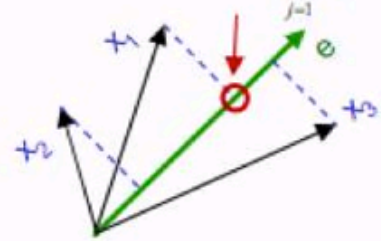
```
eig(cov(data))
```

5. pick m<d eigenvectors w. highest eigenvalues




6. project data points to those eigenvectors

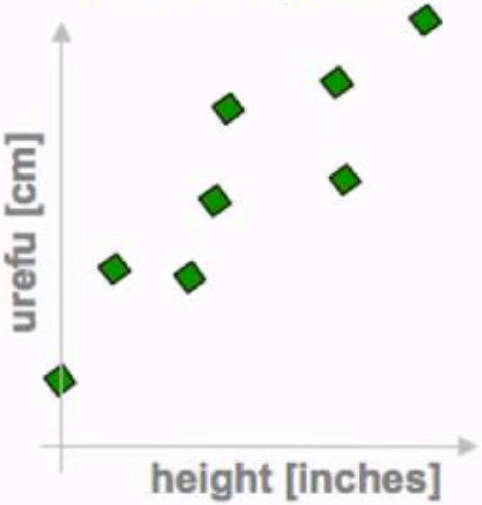
$$x_e' = x^T e = \sum_{j=1}^{d} x_{ij} e_j$$



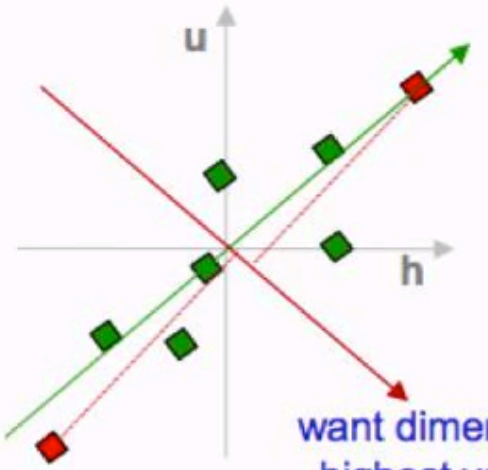
# PCA in a nutshell

1. correlated hi-d data

("urefu" means "height" in Swahili)




2. center the points




want dimension of highest variance

3. compute covariance matrix

$$\begin{array}{c|cc} & h & u \\ \hline h & 2.0 & 0.8 \\ u & 0.8 & 0.6 \end{array} \rightarrow \mathrm{cov}(h,u) = \frac{1}{n}\sum_{i=1}^{n} h_i u_i$$

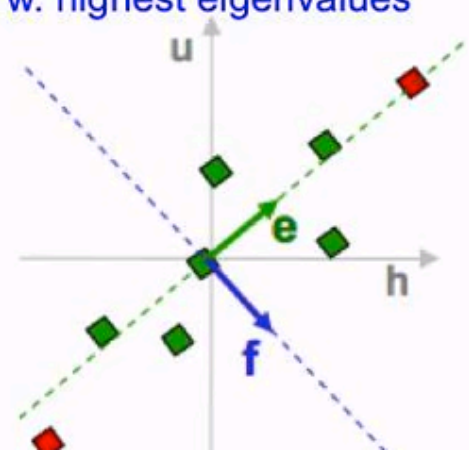
4. eigenvectors + eigenvalues

$$\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix} \begin{pmatrix} e_h \\ e_u \end{pmatrix} = \lambda_e \begin{pmatrix} e_h \\ e_u \end{pmatrix}$$

$$\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix} \begin{pmatrix} f_h \\ f_u \end{pmatrix} = \lambda_f \begin{pmatrix} f_h \\ f_u \end{pmatrix}$$

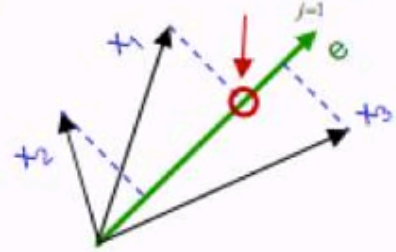
```
eig(cov(data))
```

5. pick m<d eigenvectors w. highest eigenvalues

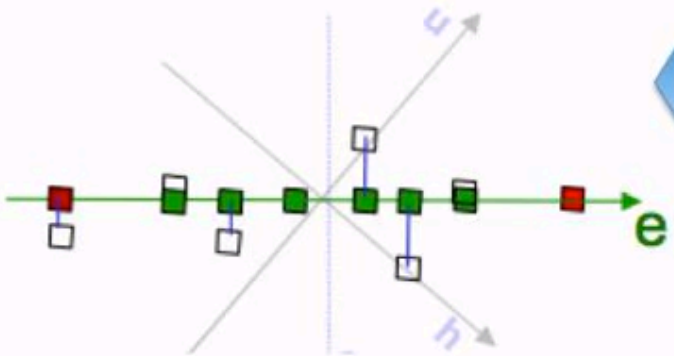



6. project data points to those eigenvectors

$$x_e' = x^T e = \sum_{j=1}^{d} x_{ij} e_j$$




7. uncorrelated low-d data



# Eigen Faces

input: dataset of N face images

input: dataset of N face images

face: K x K bitmap of pixels

input: dataset of N face images

face: K x K bitmap of pixels

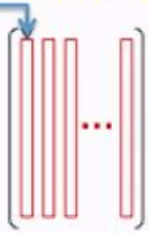
"unfold" each bitmap to $K^2$-dimensional vector

input: dataset of N face images

face: K x K bitmap of pixels

"unfold" each bitmap to $K^2$-dimensional vector

arrange in a matrix
each face = column



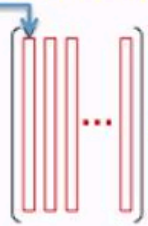
$K^2 \times N$

input: dataset of N face images

face: K x K bitmap of pixels

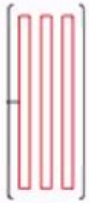
"unfold" each bitmap to $K^2$-dimensional vector

arrange in a matrix
each face = column



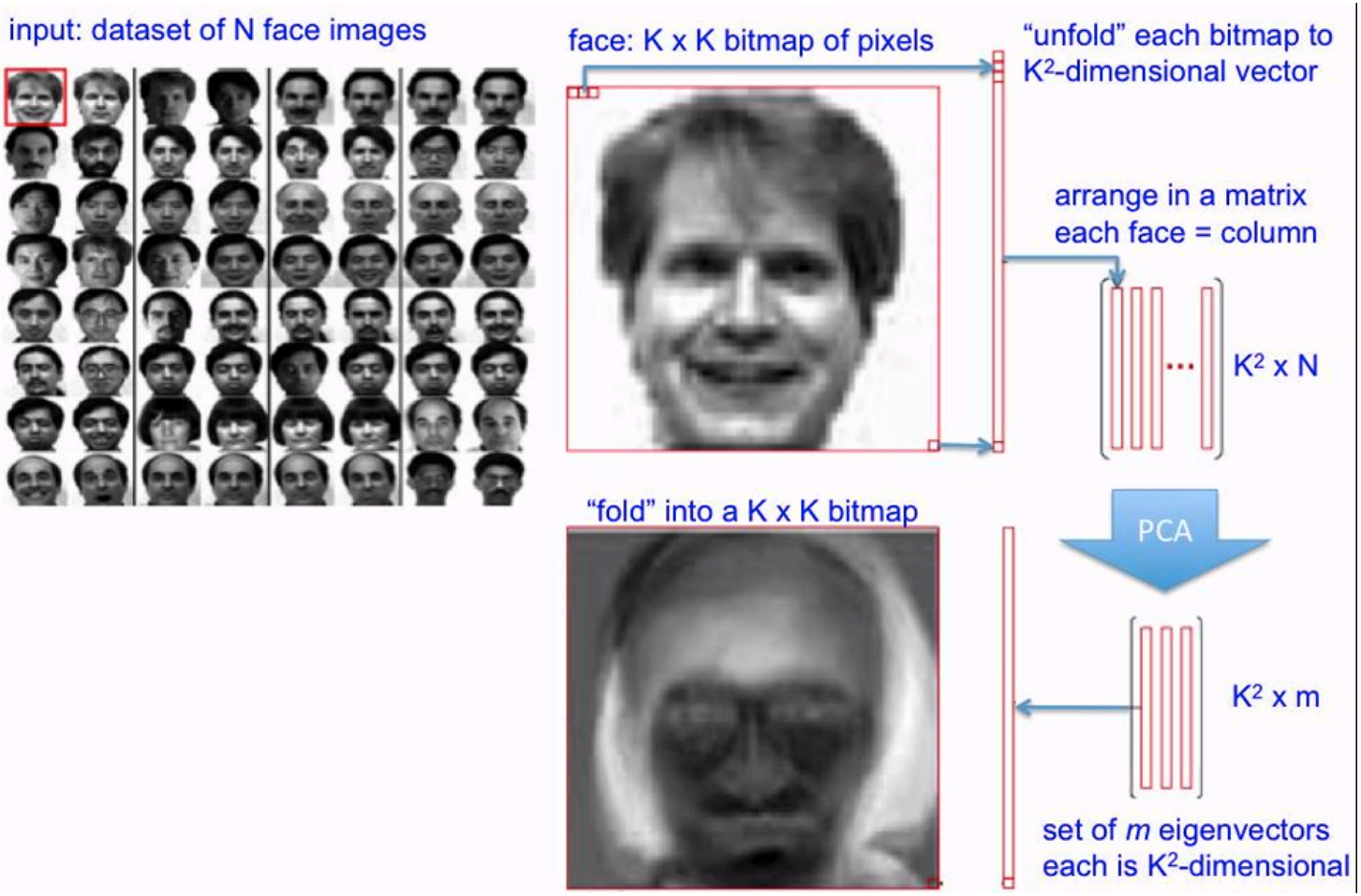
$K^2 \times N$

PCA



$K^2 \times m$

set of *m* eigenvectors
each is $K^2$-dimensional

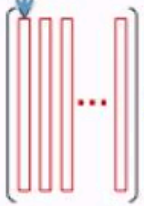

input: dataset of N face images
face: K x K bitmap of pixels
"unfold" each bitmap to
K²-dimensional vector
arrange in a matrix
each face = column
K² x N
PCA
"fold" into a K x K bitmap
K² x m
set of m eigenvectors
each is K²-dimensional

input: dataset of N face images

face: K x K bitmap of pixels

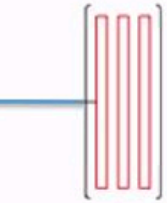
"unfold" each bitmap to $K^2$-dimensional vector

arrange in a matrix
each face = column



$K^2 \times N$

PCA



$K^2 \times m$

set of *m* eigenvectors
each is $K^2$-dimensional

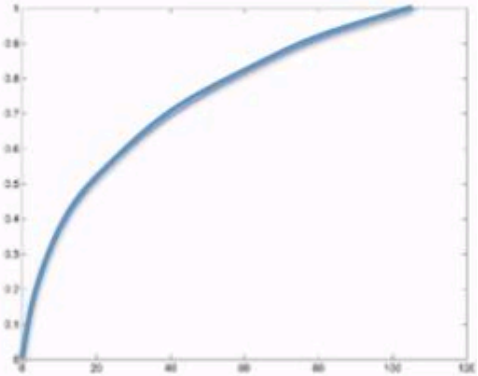
"fold" into a K x K bitmap

can visualize
eigenvectors:
*m* "aspects"
of prototypical
facial features

- Project new face to space of eigen-faces
- Represent vector as a linear combination of principal components
- How many do we need?


- Project new face to space of eigen-faces
- Represent vector as a linear combination of principal components
- How many do we need?


- Project new face to space of eigen-faces
- Represent vector as a linear combination of principal components
- How many do we need?



# applications

- Face similarity
  - in the reduced space
  - insensitive to lighting expression, orientation
- Projecting new "faces"
  - everything is a face

# Eigenface Recognition

- Face similarity
  - in the reduced space
  - insensitive to lighting expression, orientation
- Projecting new "faces"
  - everything is a face

new face

projected to eigenfaces

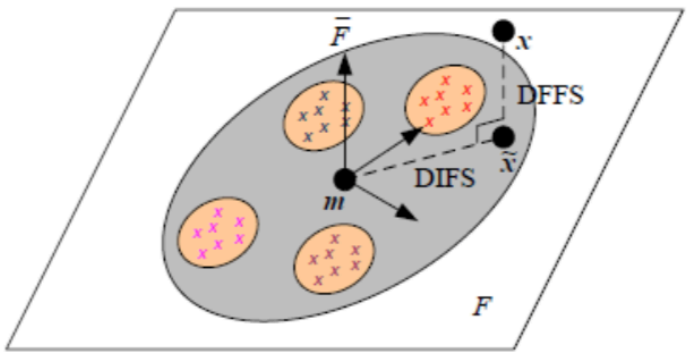
# Compute the Face distance

The distance of a projected face DIFS (Distance in face space)

$$DIFS = \|\tilde{x} - m\| = \sqrt{\sum_{i=0}^{M-1} a_i^2}$$

The distance between two faces

$$DIFS = \|\tilde{x} - \tilde{y}\| = \sqrt{\sum_{i=0}^{M-1} (a_i - b_i)^2}$$



We are not utilizing the eigen value information, compute Mahalonobis distance

$$DIFS' = \left\| \tilde{x} - m \right\|_{C^{-1}} = \sqrt{\sum_{i=0}^{M-1} a_i^2 \Big/ \lambda_i^2}$$

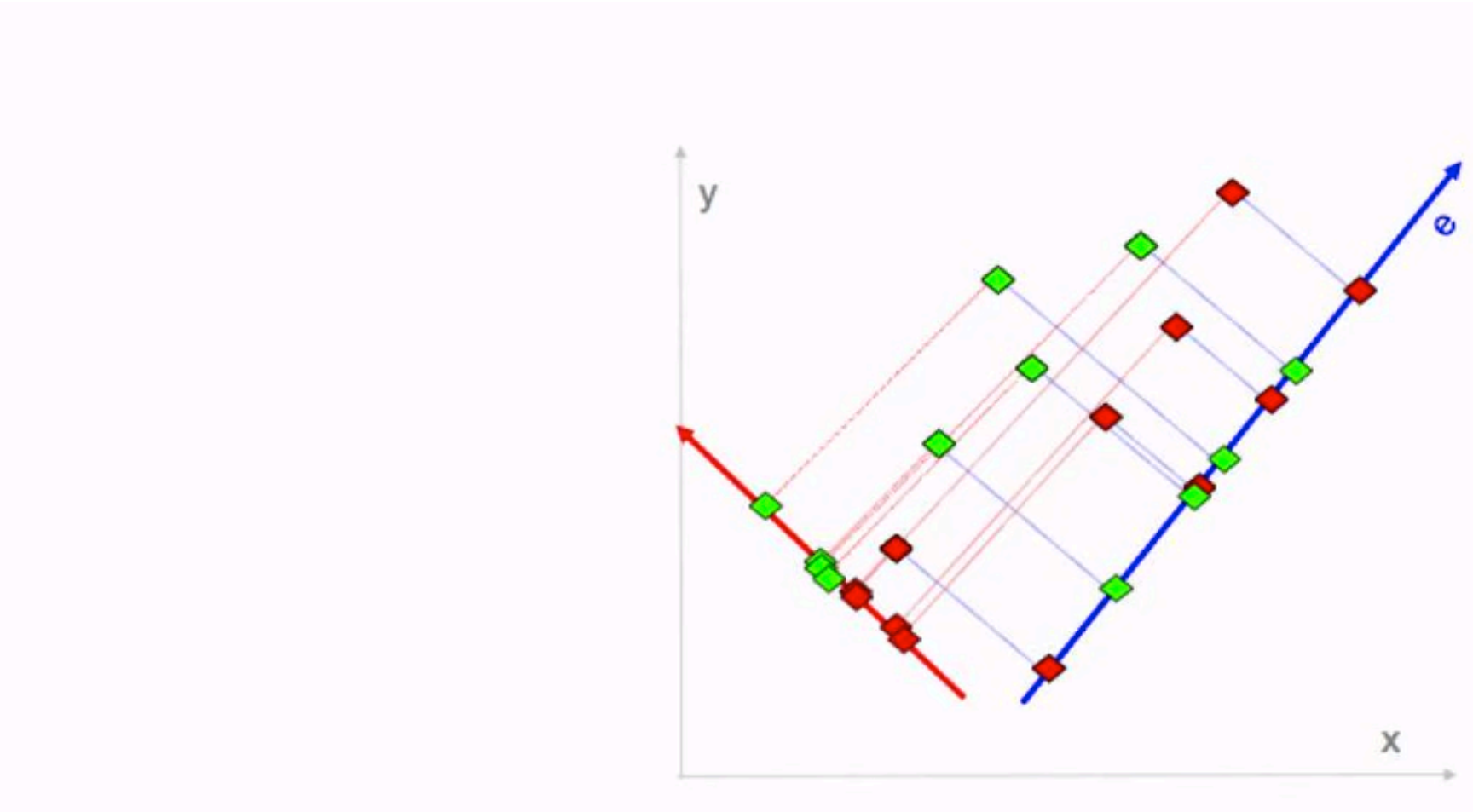
Pre-scale the eigen vectors by eigenvalues:

$$\hat{U} = U\mathrm{A}^{-1/2}$$

Euclidean $$\mathrm{d}(\mathbf{p},\mathbf{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2}.$$

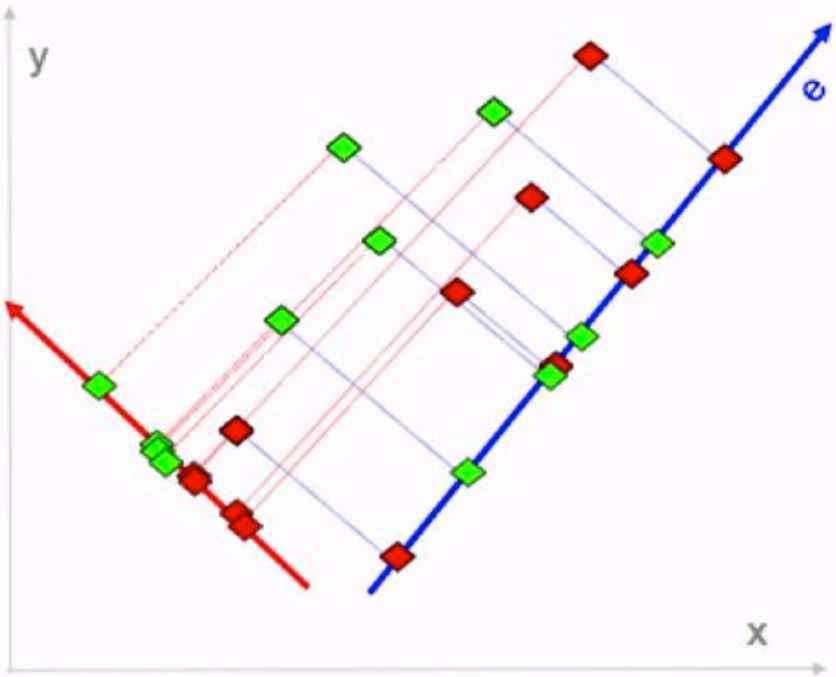
Mahalonobis $$d(\vec{x},\vec{y}) = \sqrt{(\vec{x} - \vec{y})^T S^{-1} (\vec{x} - \vec{y})}.$$

# Linear Discriminant Analysis



# Linear Discriminant Analysis

- PCA is unsupervised
  - maximizes overall variance of the data along a small set of directions
  - does not know anything about class labels
  - can pick direction that makes it hard to separate classes
- Discriminative approach
  - look for a dimension that makes it easy to separate classes



# Linear Discriminant Analysis

- LDA: pick a new dimension that gives:
  - maximum separation between means of projected classes
  - minimum variance within each projected class
- Solution: eigenvectors based on between-class and within-class covariance matrices