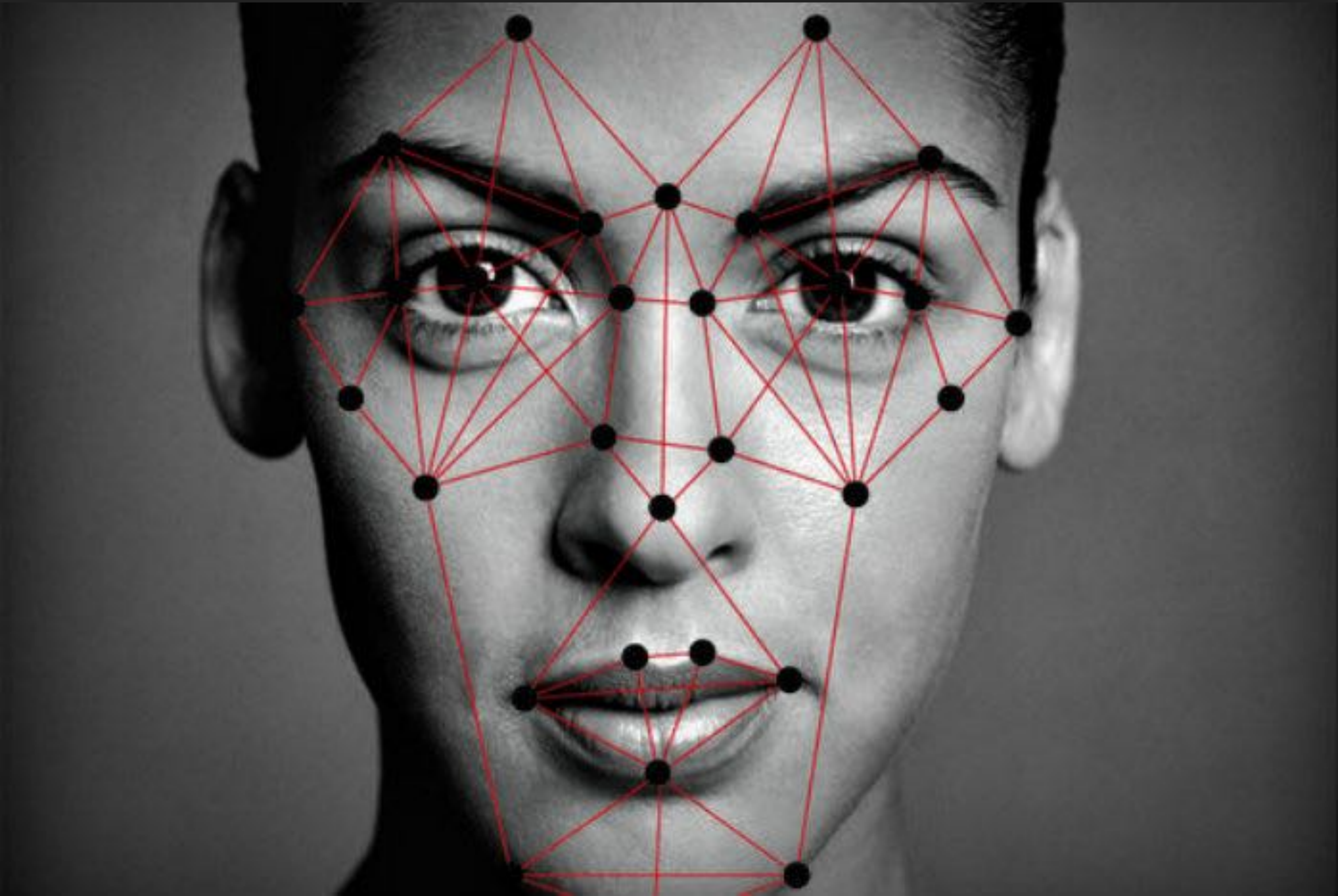
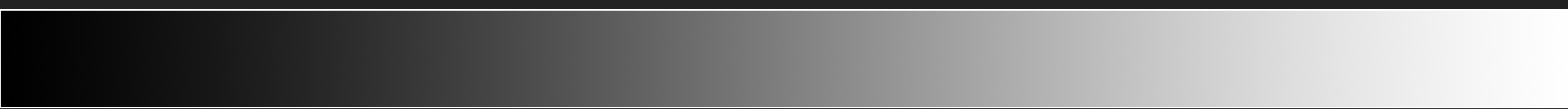
Pattern Recognition



what is the data?

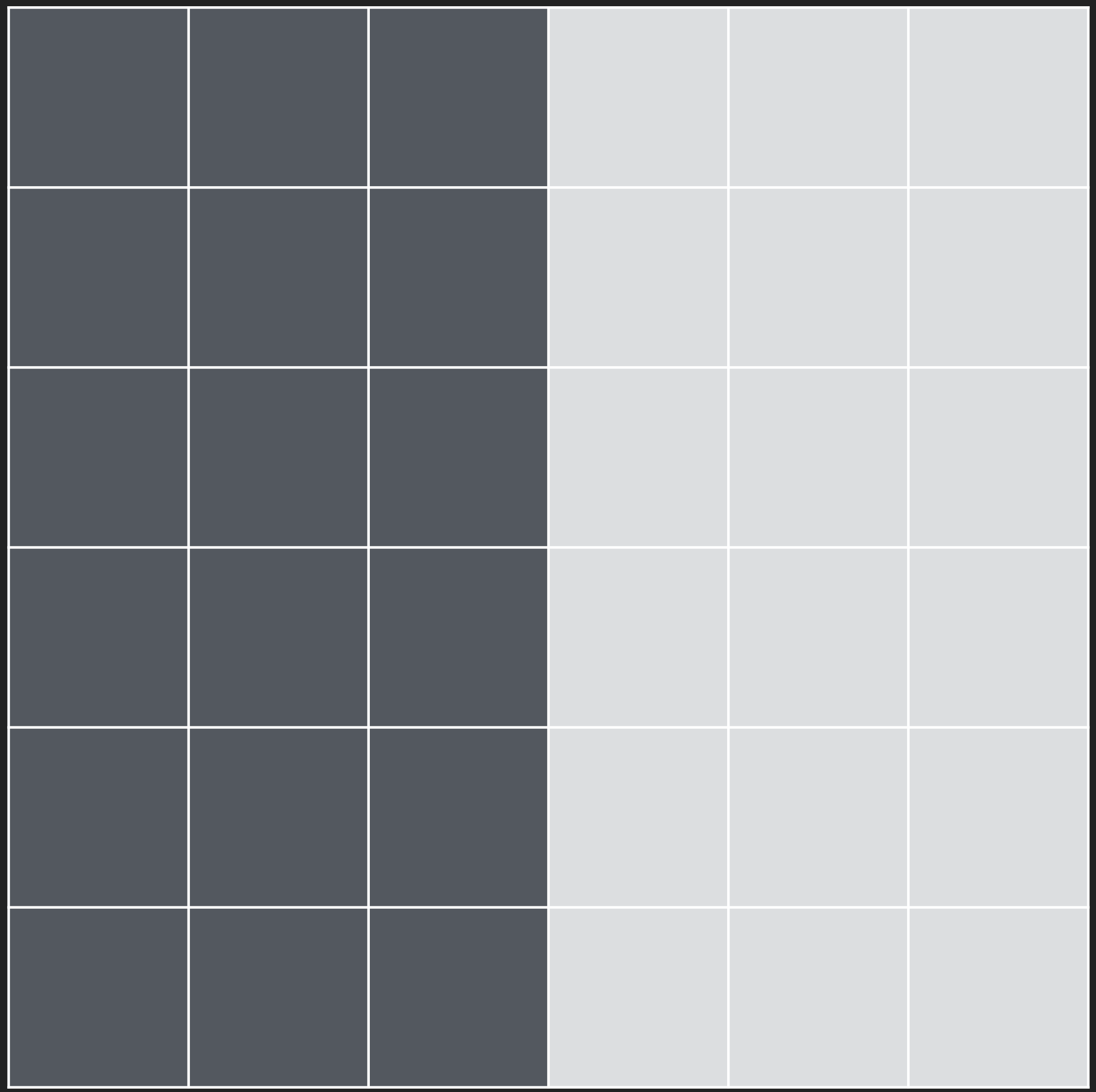
pixels

pixels



0 128 255

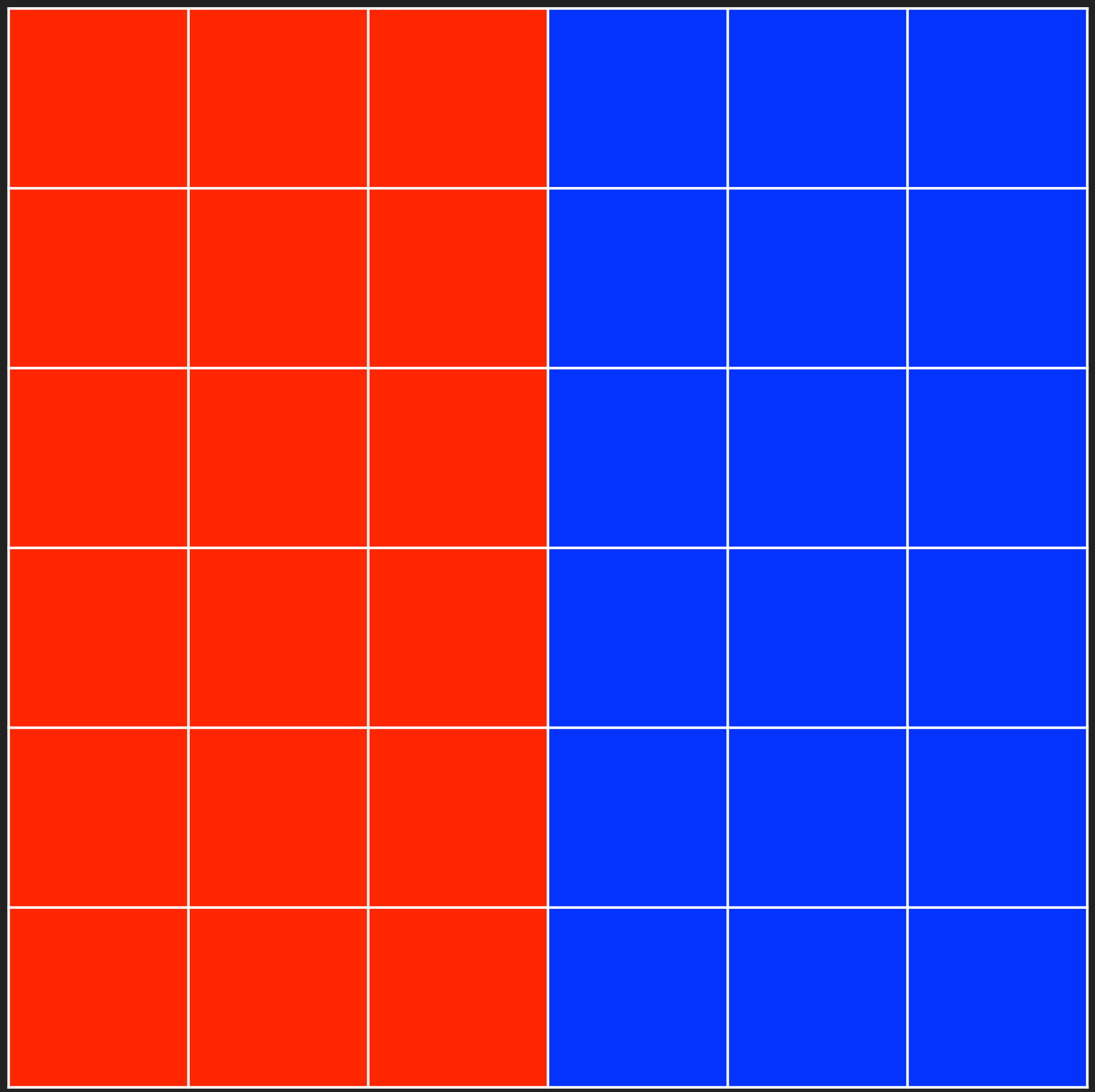
pixels



32	32	32	223	223	223
32	32	32	223	223	223
32	32	32	223	223	223
32	32	32	223	223	223
32	32	32	223	223	223
32	32	32	223	223	223

color pixels

[255,0,0]



[0,0,255]

what is the data?

a color (RGB) image is represented as 3 matrices (arrays) each containing numeric values typically in the range [0,255] with 0 corresponding to “black” and 255 corresponding to “white” how do we compare two images?

70	244	129	38	236	89
173	86	178	65	89	211
167	149	227	214	50	149
41	57	244	64	64	140
30	191	139	207	157	233
127	65	35	62	120	72

image 1

193	135	3	42	114	137
192	198	85	153	21	254
97	238	41	67	58	19
144	33	202	166	232	112
19	145	79	175	38	27
13	119	134	190	210	245

image 2

70	244	129	38	236	89
173	86	178	65	89	211
167	149	227	214	50	149
41	57	244	64	64	140
30	191	139	207	157	233
127	65	35	62	120	72

image 1

-

193	135	3	42	114	137
192	198	85	153	21	254
97	238	41	67	58	19
144	33	202	166	232	112
19	145	79	175	38	27
13	119	134	190	210	245

image 2

-123	109	126	-4	122	-48
19	-112	93	-88	68	-43
70	-89	186	147	-8	130
-103	24	42	-102	-168	28
11	46	60	32	119	206
114	-54	-99	-128	-90	-173

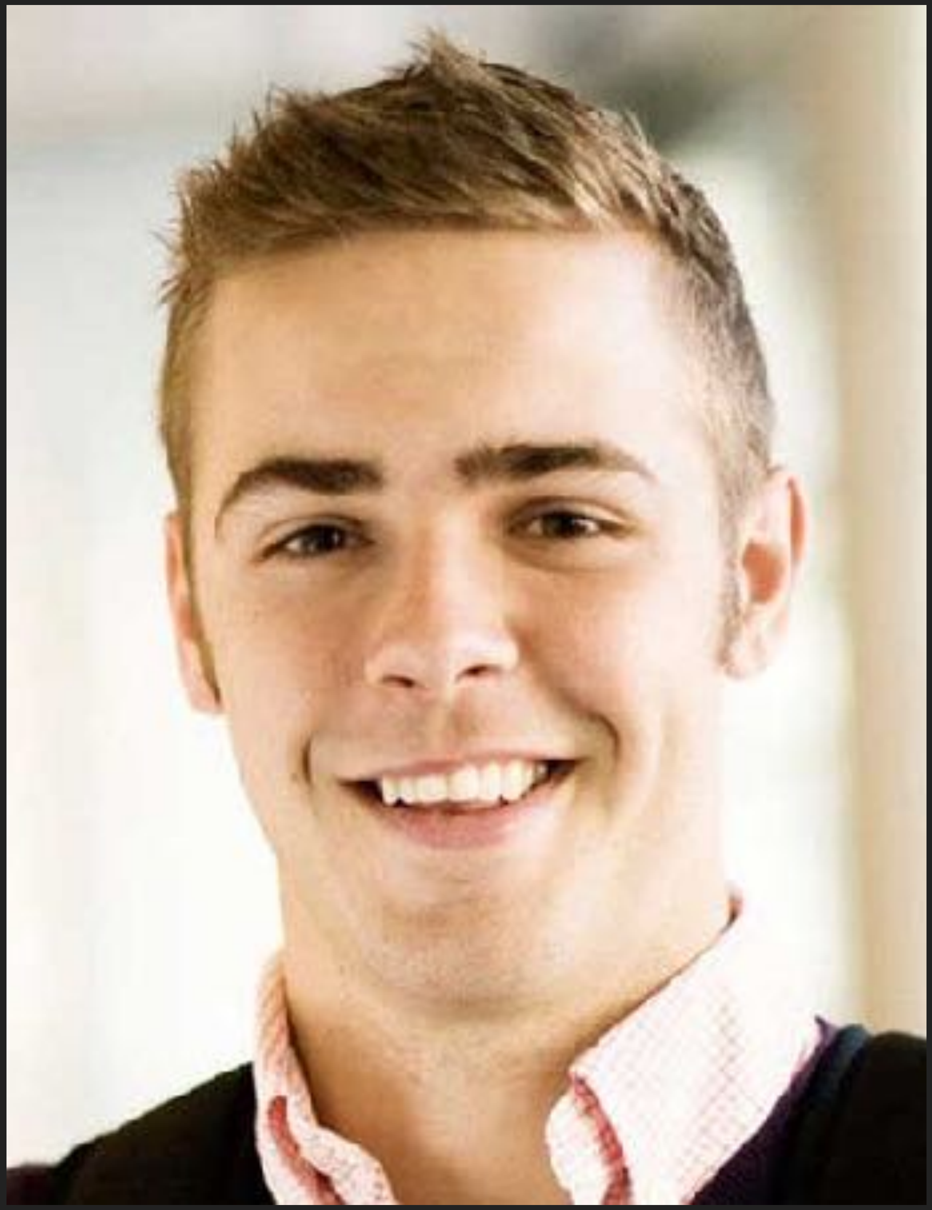
image 1 - image 2
(difference)

123	109	126	4	122	48
19	112	93	88	68	43
70	89	186	147	8	130
103	24	42	102	168	28
11	46	60	32	119	206
114	54	99	128	90	173

| image 1 - image 2 |
(absolute value)

3184

$\sum$| image 1 - image 2 |
(sum of absolute values)



$\sum |\text{image 1} - \text{image 2}|$

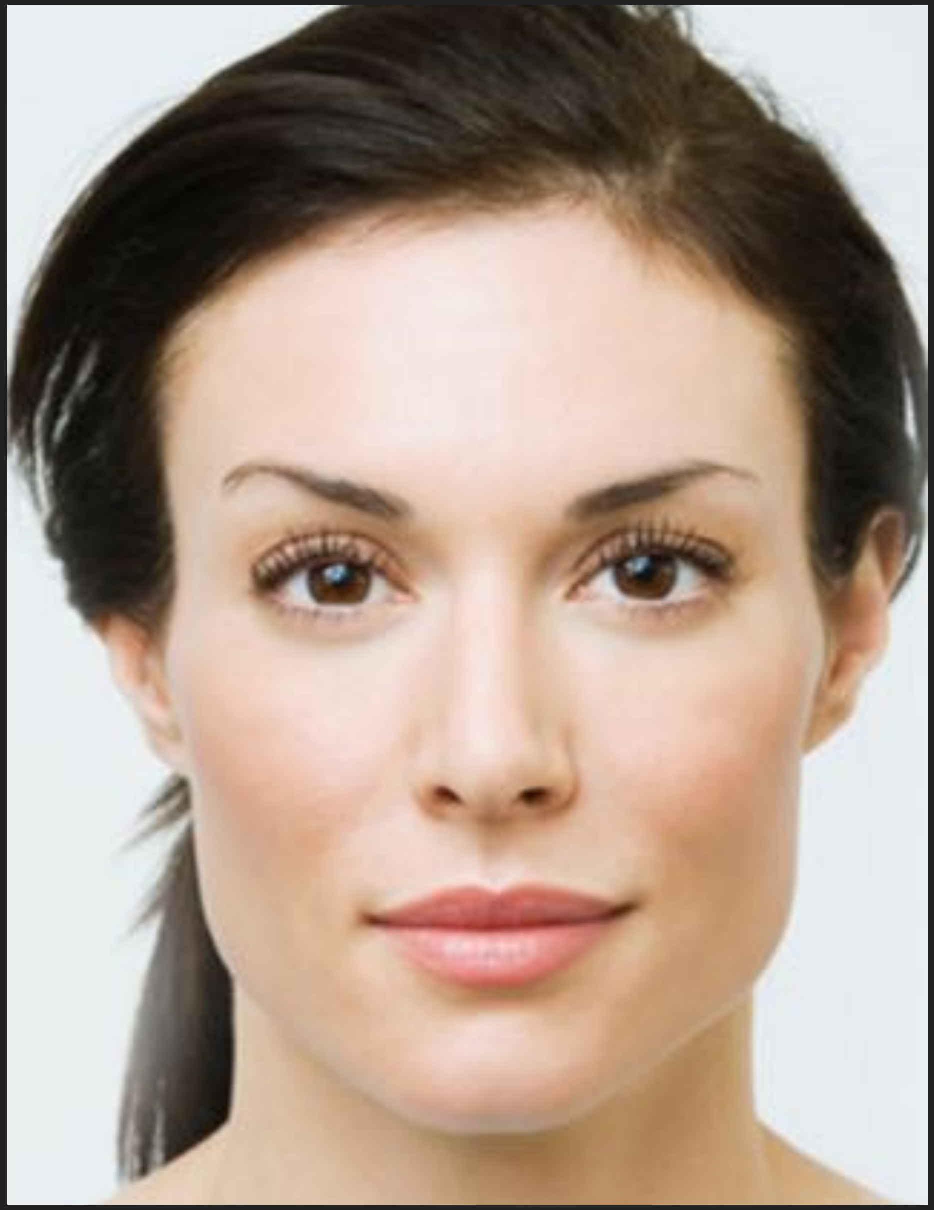
(sum of absolute values)

$\sum$| image 1 - image 2 |

(sum of absolute values)

$\sum$| image 1 - image 2 |

(sum of absolute values)



$\sum |\text{ image 1 - image 2 }|$

(sum of absolute values)

are pixels the right “data”?

there are at least two problems with pixels:

fragility

dimensionality

most recognition systems consist of two basic parts*

extract features (low-dimensional)

build a classifier based on features

** Not unique to face recognition*

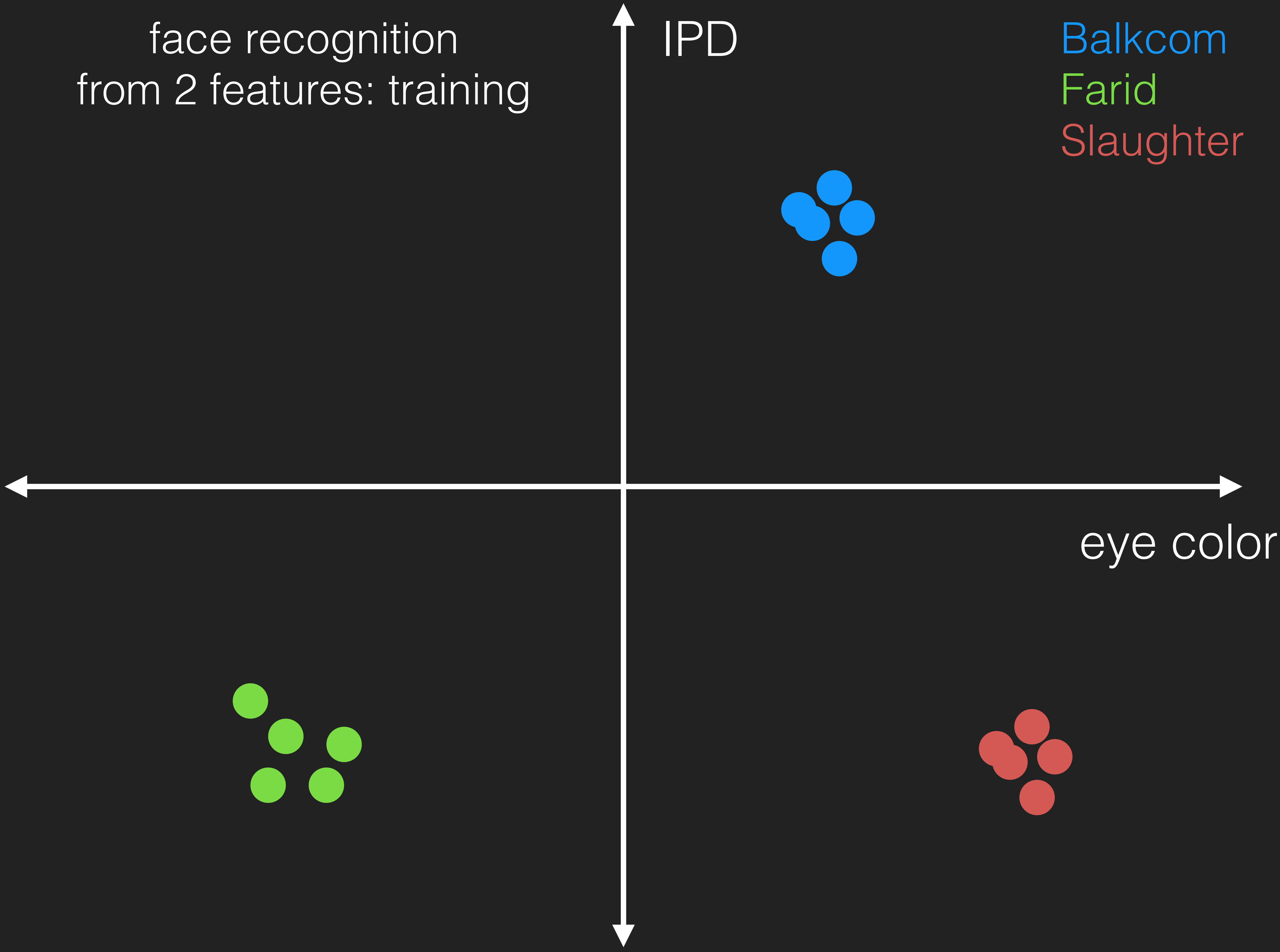
most recognition systems consist of two basic parts*

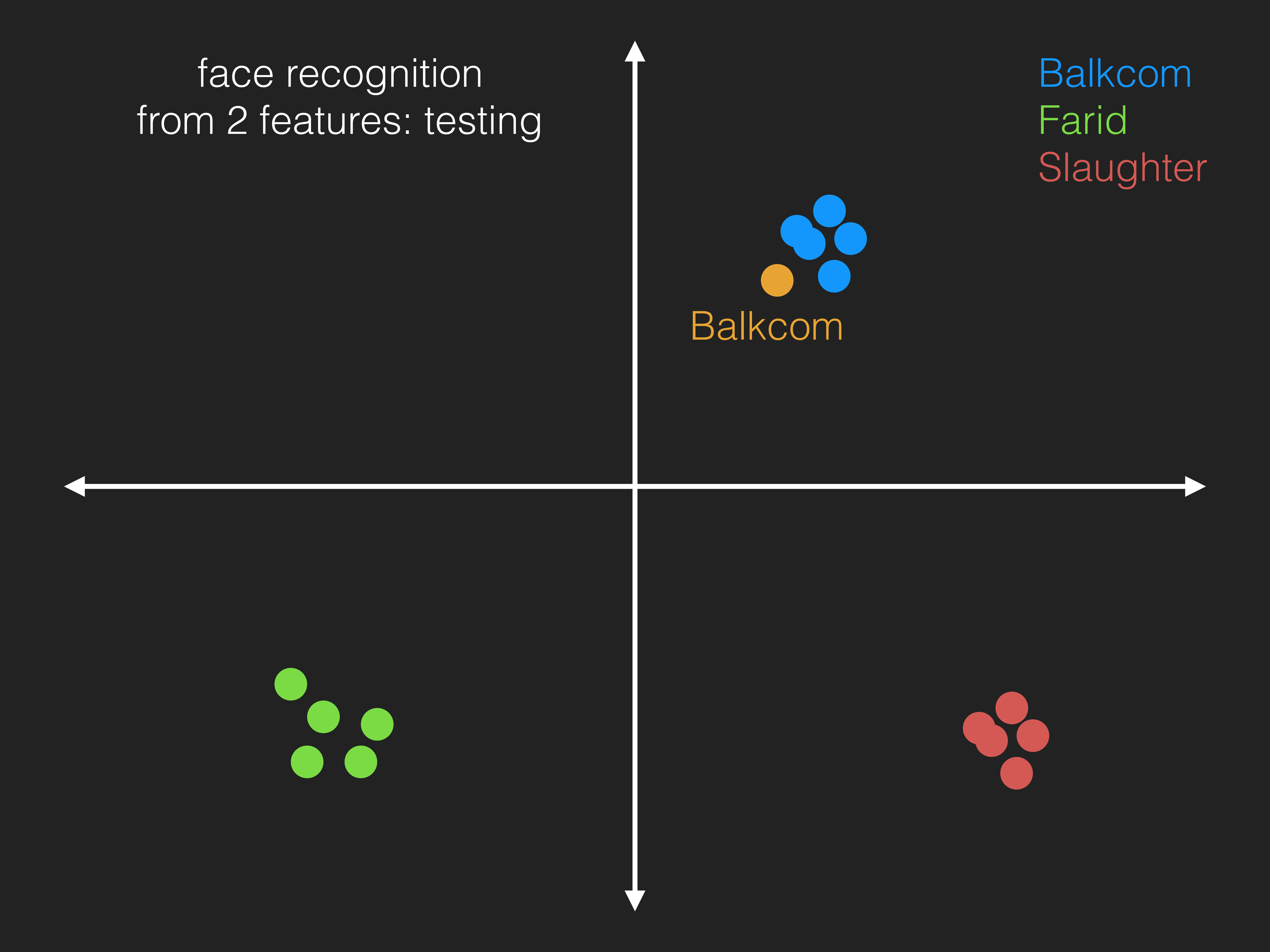
extract features (low-dimensional)

build a classifier based on features

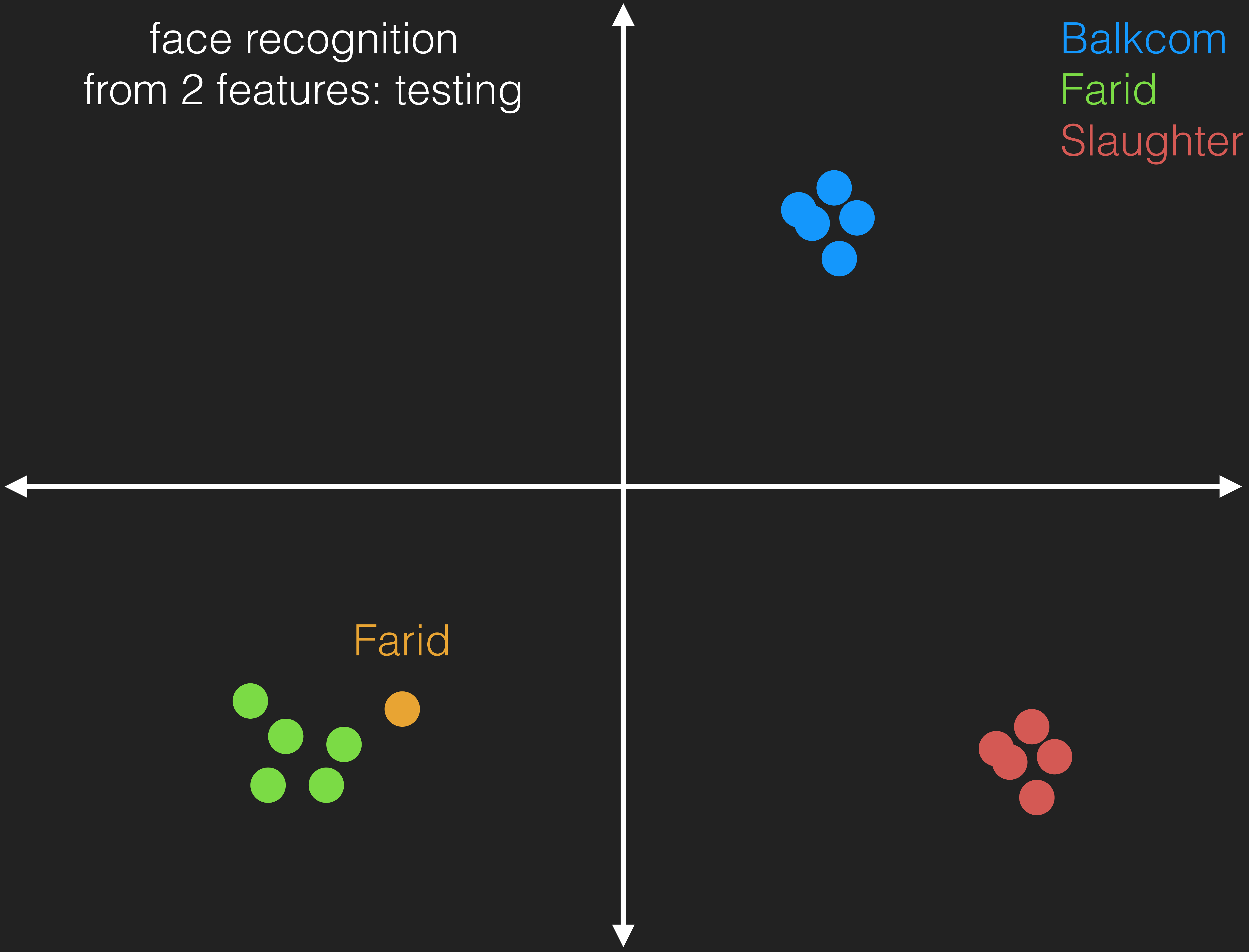
** Deep neural networks combine these into one step*

face recognition
from 2 features: training

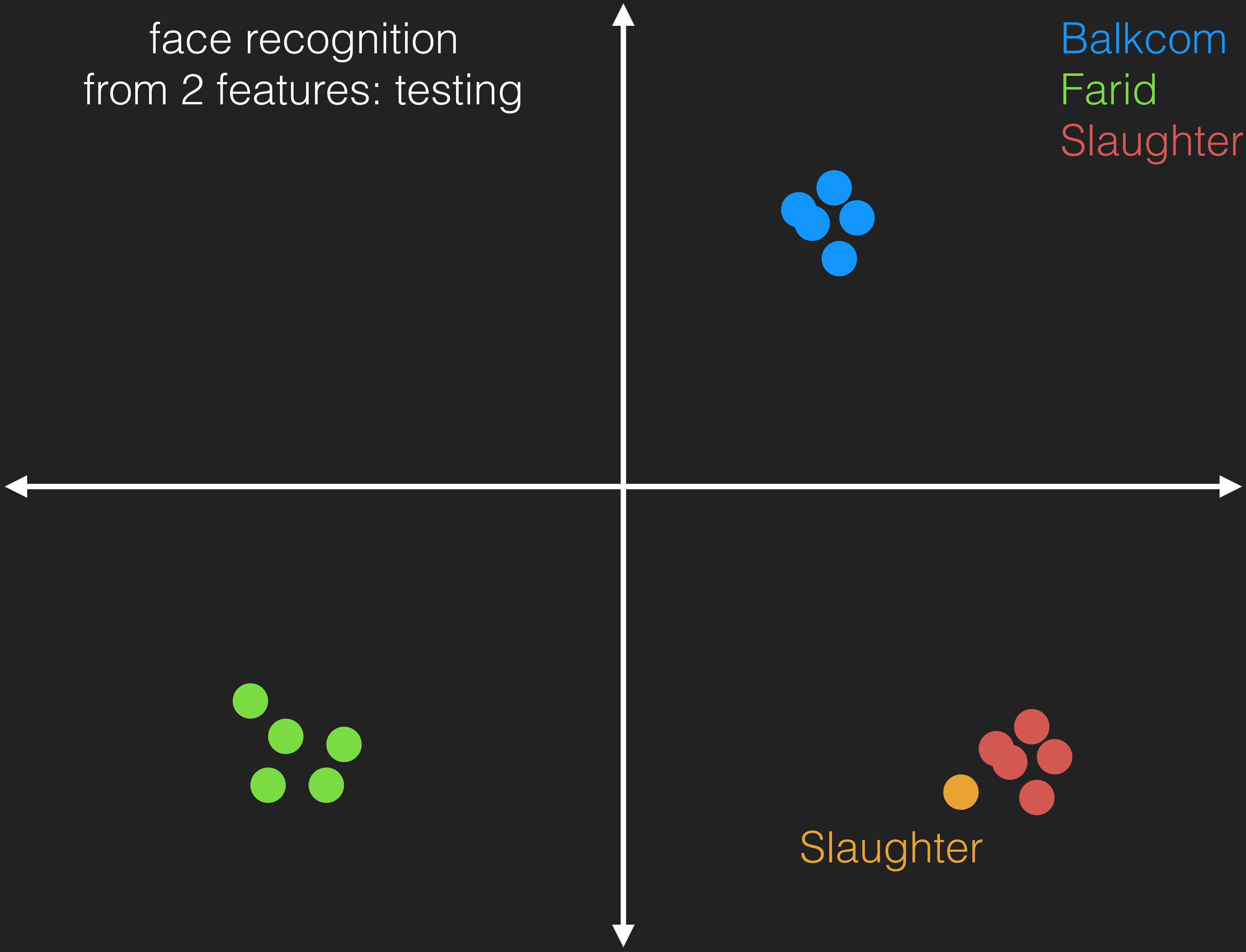



face recognition
from 2 features: testing
Balkcom
Farid
Slaughter
Balkcom

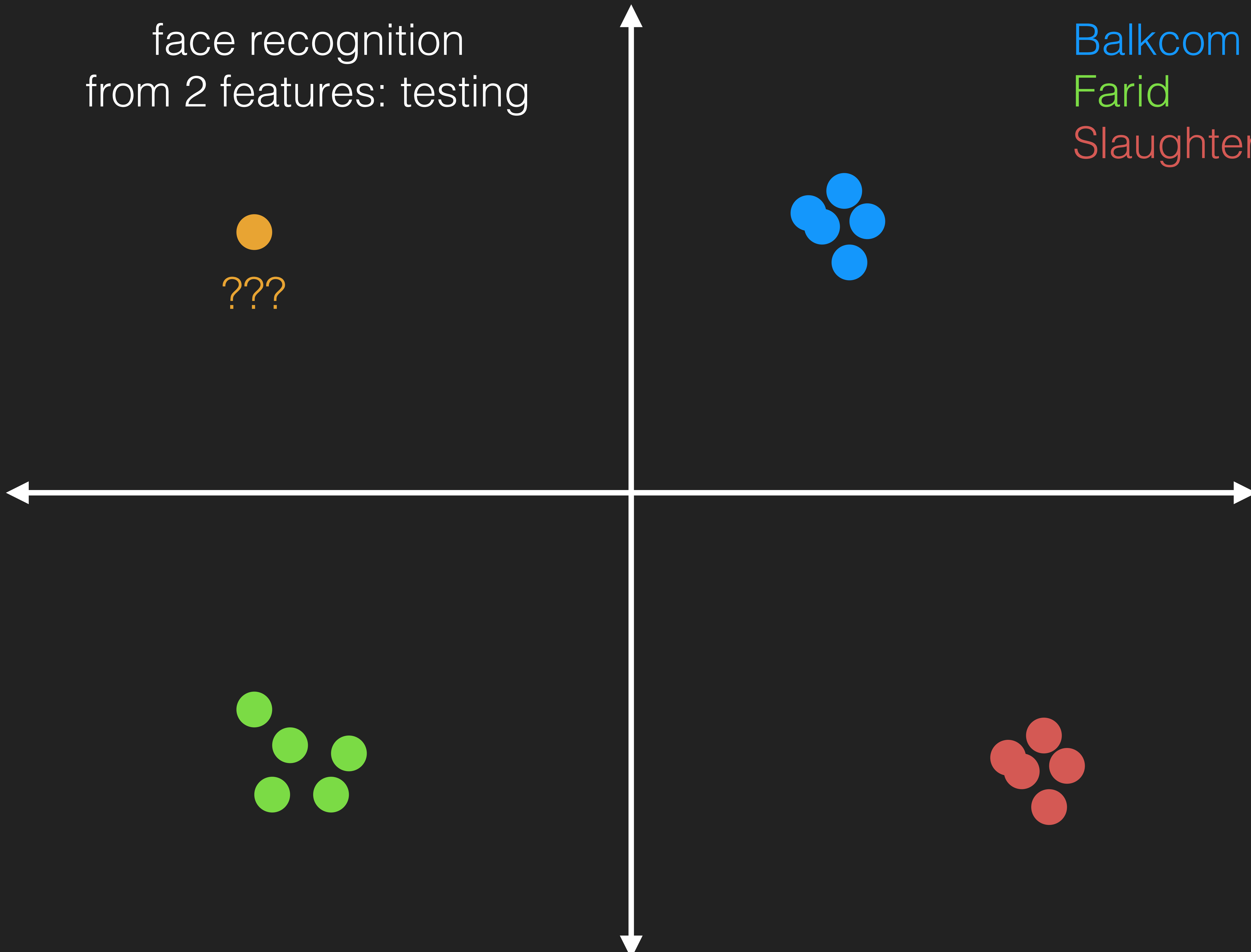
face recognition
from 2 features: testing



face recognition
from 2 features: testing



face recognition
from 2 features: testing



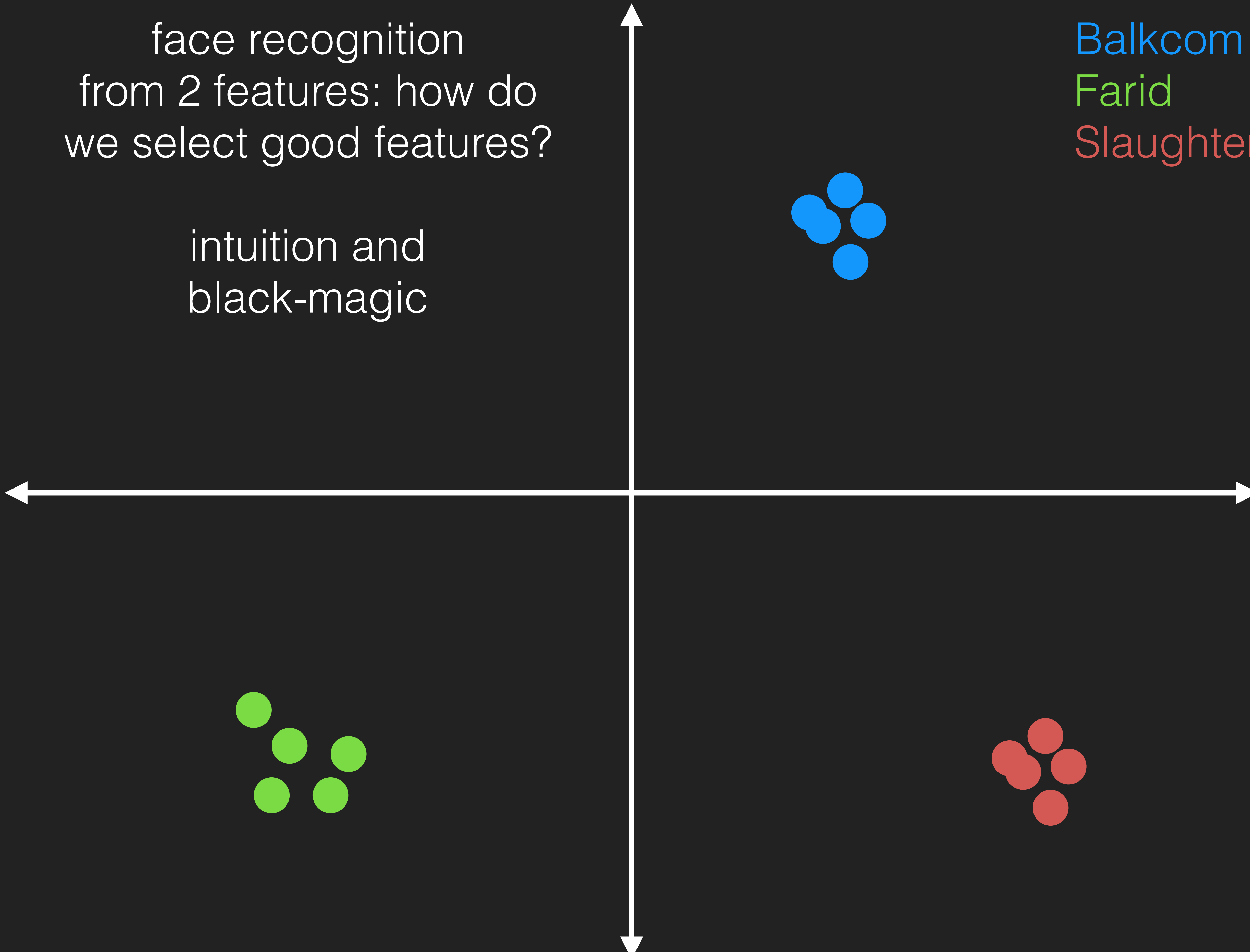
face recognition
from 2 features: what are
good features?

face recognition
from 2 features: what are
good features?

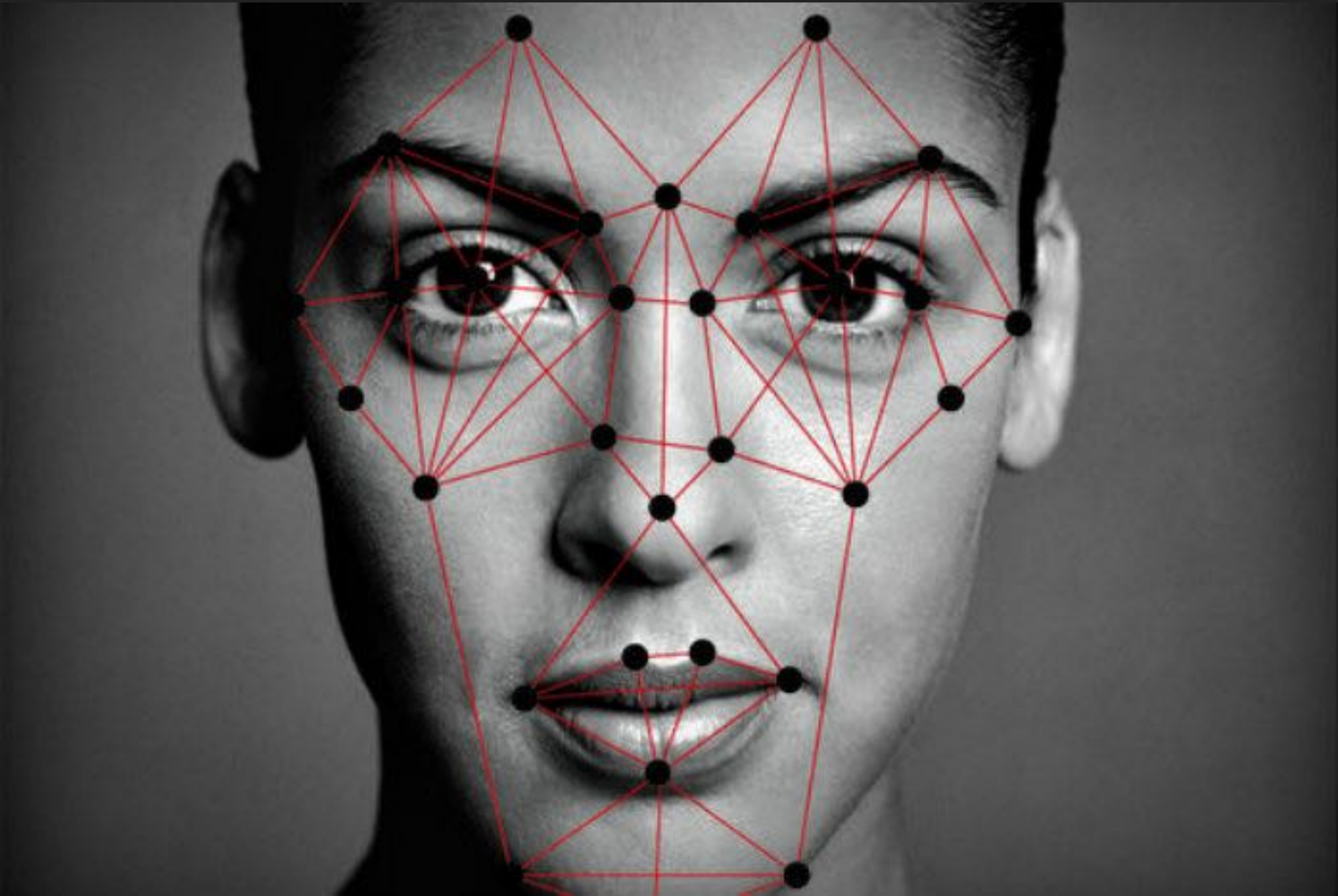
low-dimensional and
discriminating

face recognition from 2 features: how do we select good features?

intuition and black-magic



Facial Features (Geometric)



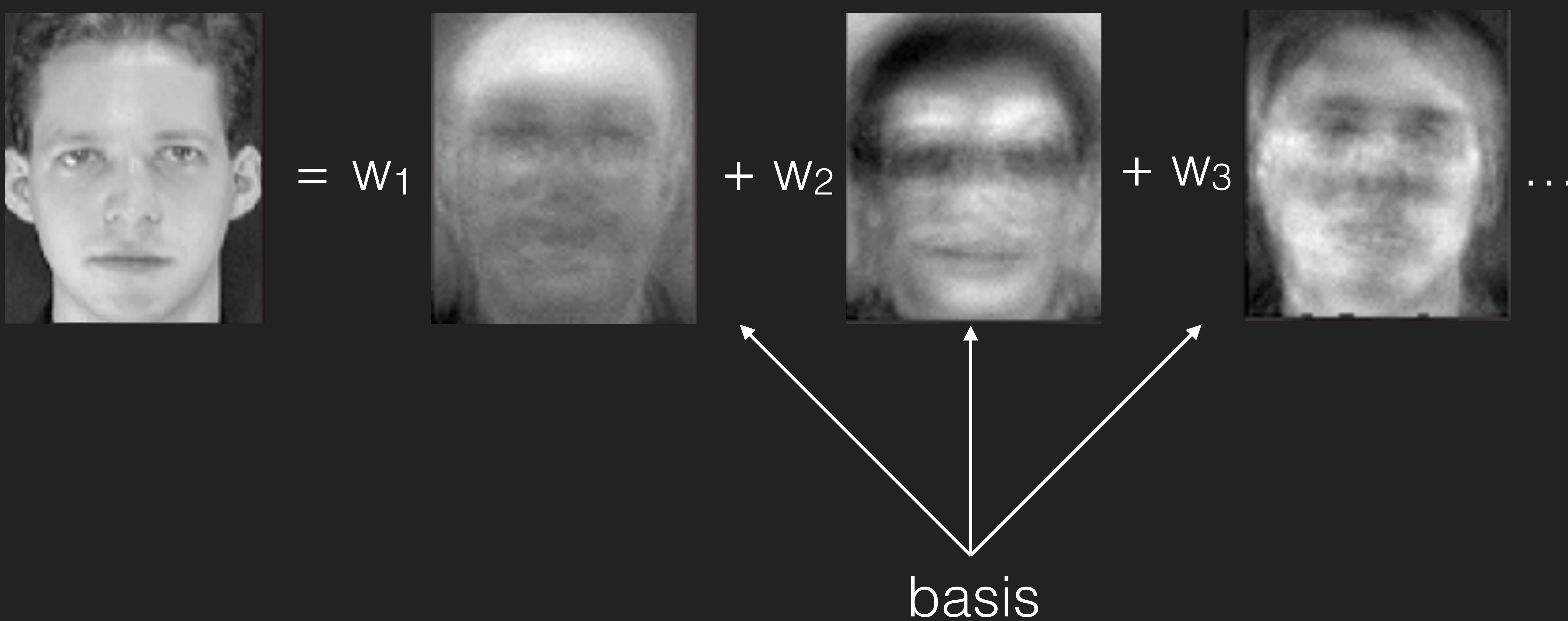
Facial Features (Principal Components Analysis (PCA))*

* aka Multi-Dimensional Scaling (MDS) or eigenfaces

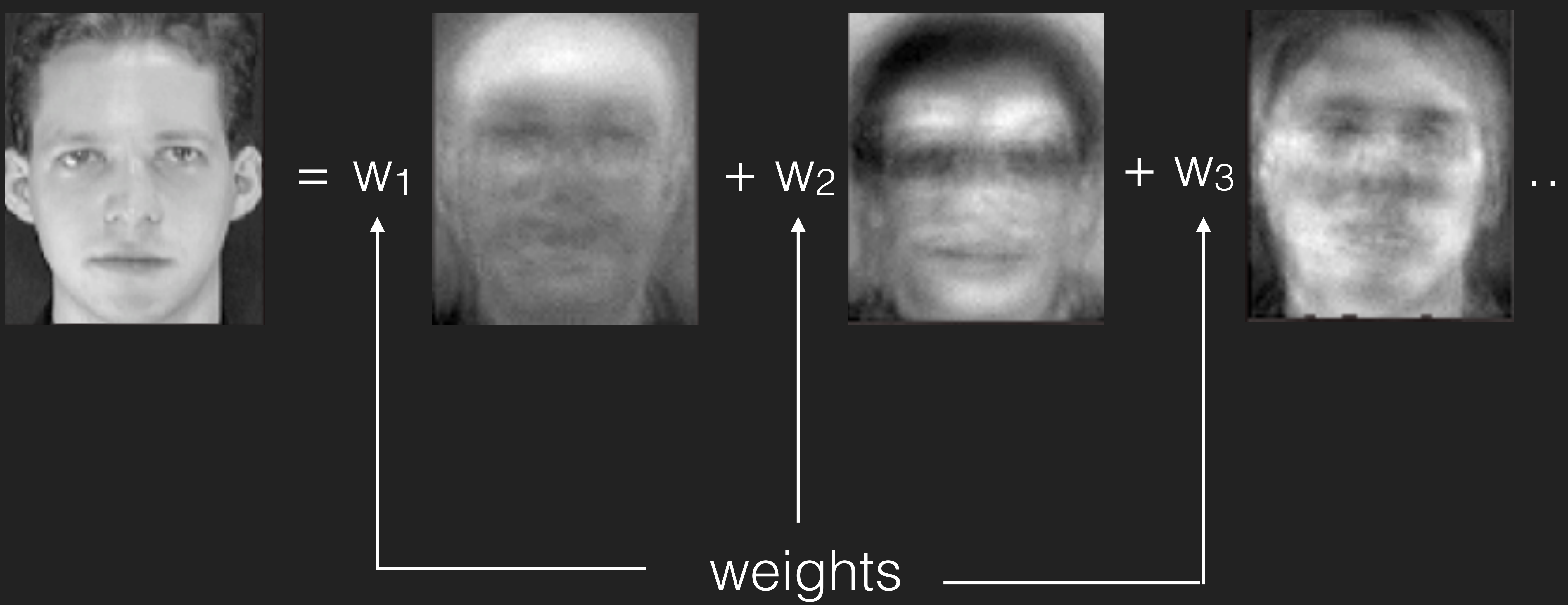
Facial Features (PCA)

 $= w_1$  $+ w_2$  $+ w_3$  $\ldots$

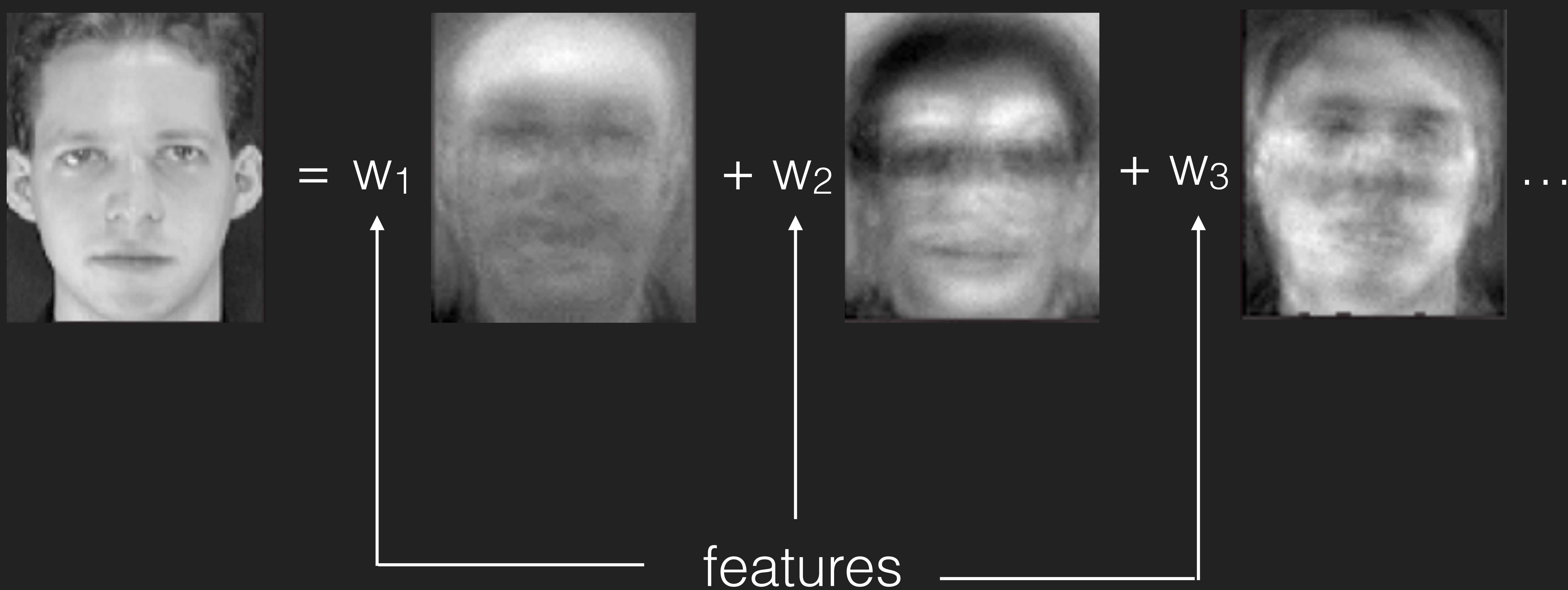
Facial Features (PCA)



Facial Features (PCA)



Facial Features (PCA)

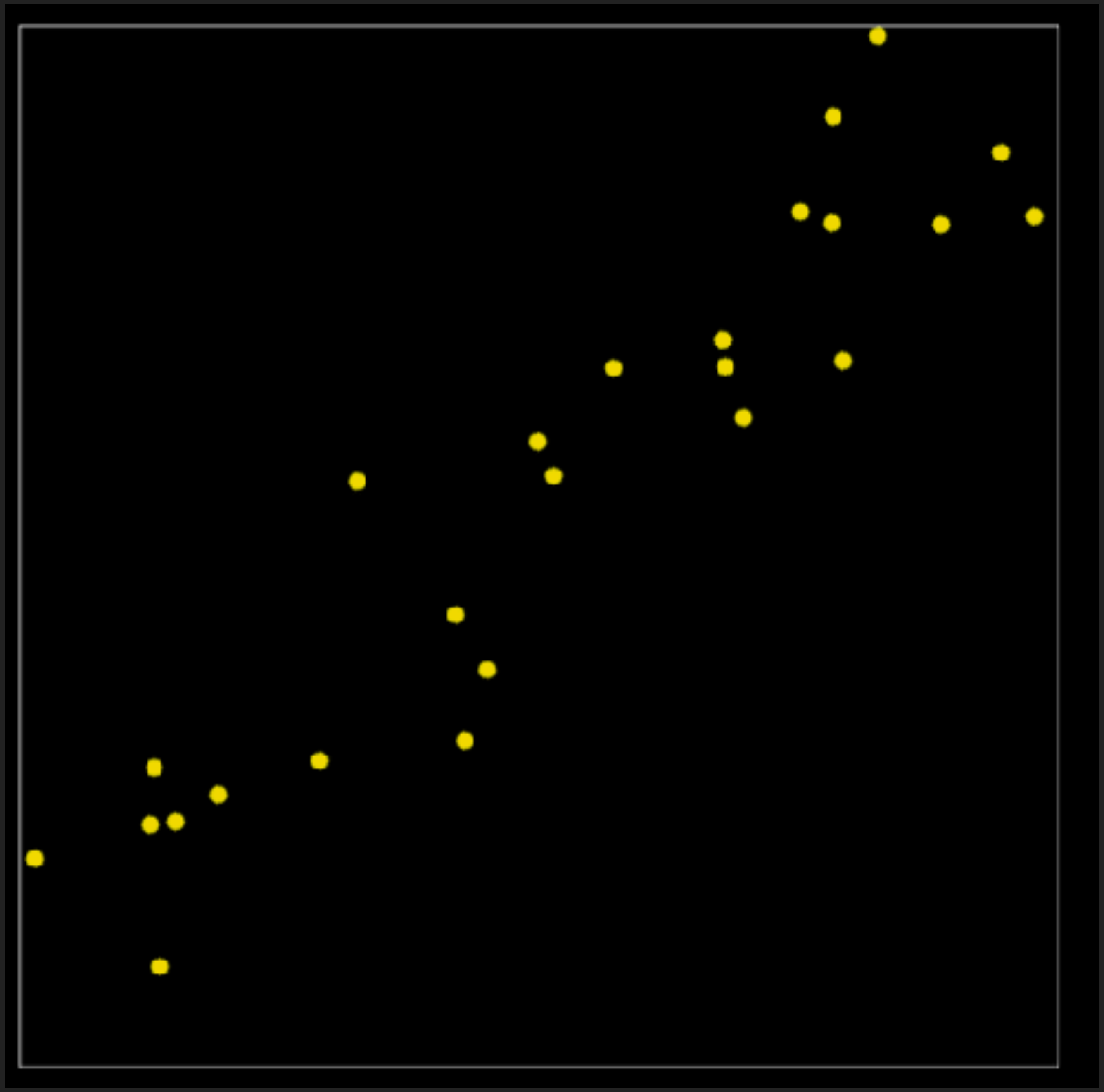


Facial Features (PCA)

 $= w_1$  $+ w_2$  $+ w_3$  $\ldots$

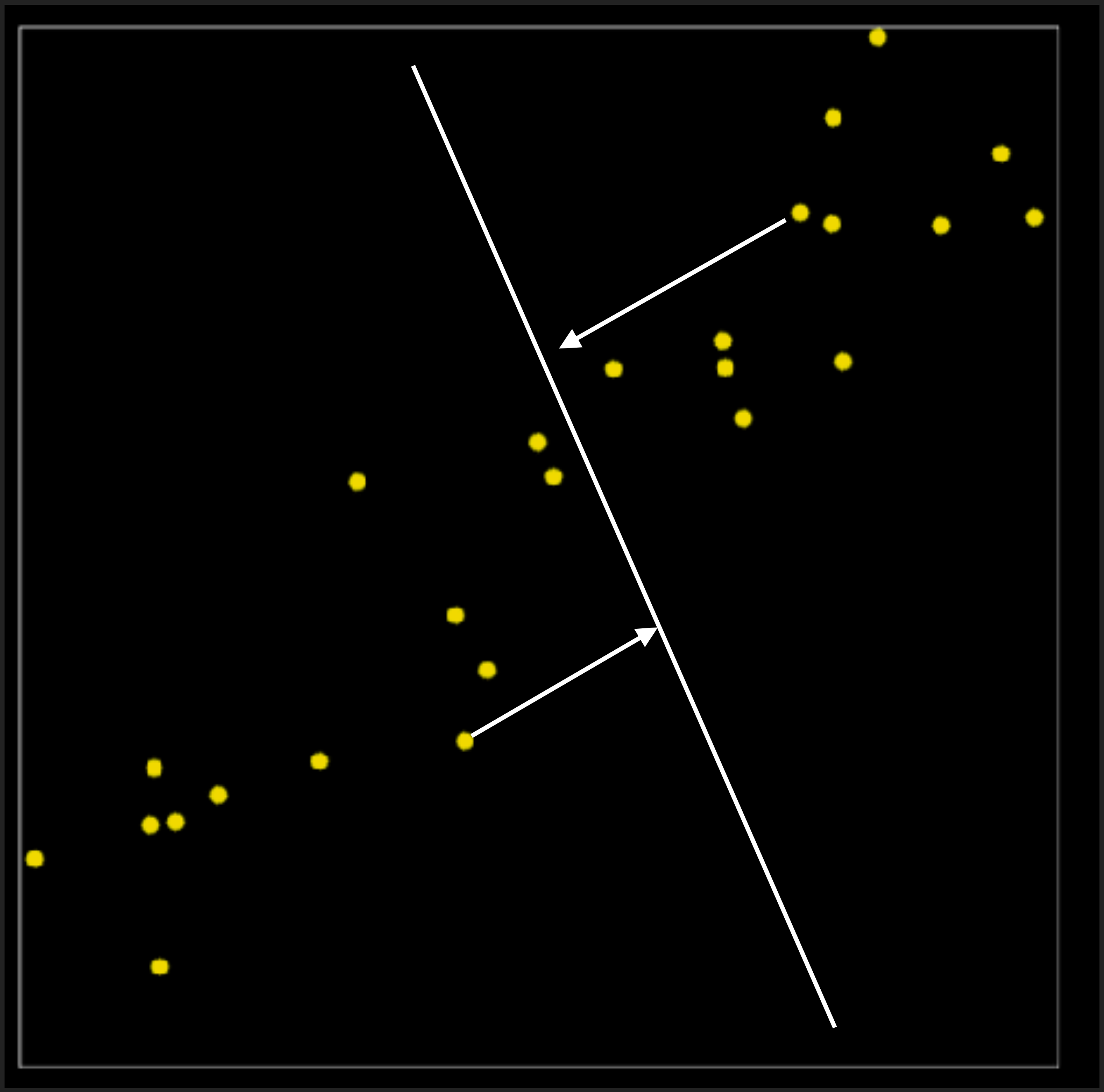
PCA automatically generates a basis from training data

Facial Features (PCA)



original (2-D)

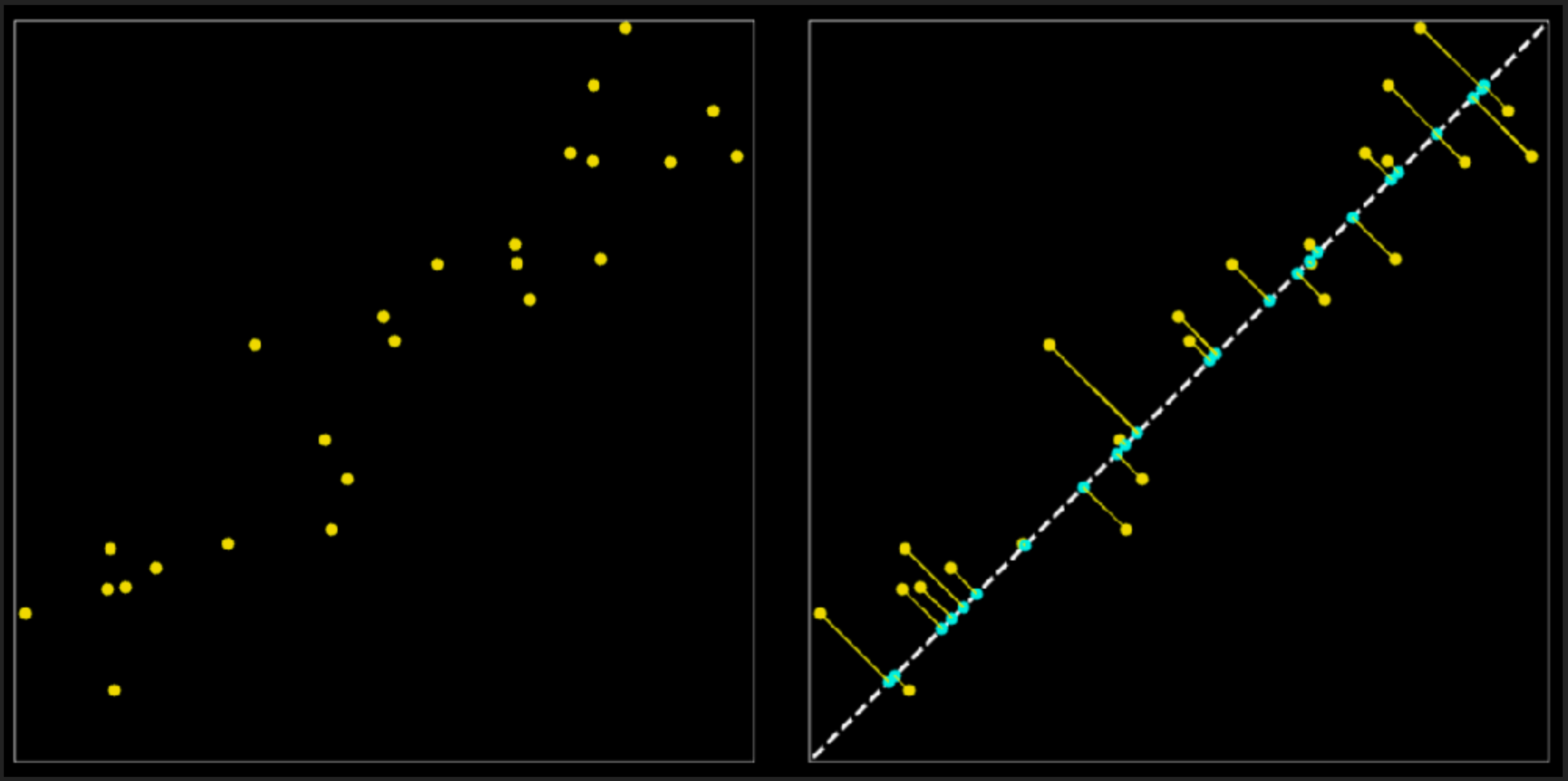
Facial Features (PCA)



original (2-D)

projection reduces dimensionality

Facial Features (PCA)



original (2-D)

PCA (1-D) - project onto axis of maximal variance

Facial Features (PCA)

An N x N pixel image is a point in a N^2 dimensional space

Facial Features (PCA)

An N x N pixel image is a point in a N^2 dimensional space

Given a large set of training images construct a PCA basis (typically of dimensionality ~20)

Facial Features (PCA)

An N x N pixel image is a point in a N^2 dimensional space

Given a large set of training images construct a PCA basis (typically of dimensionality ~20)

Project all training images onto new basis

Facial Features (PCA)

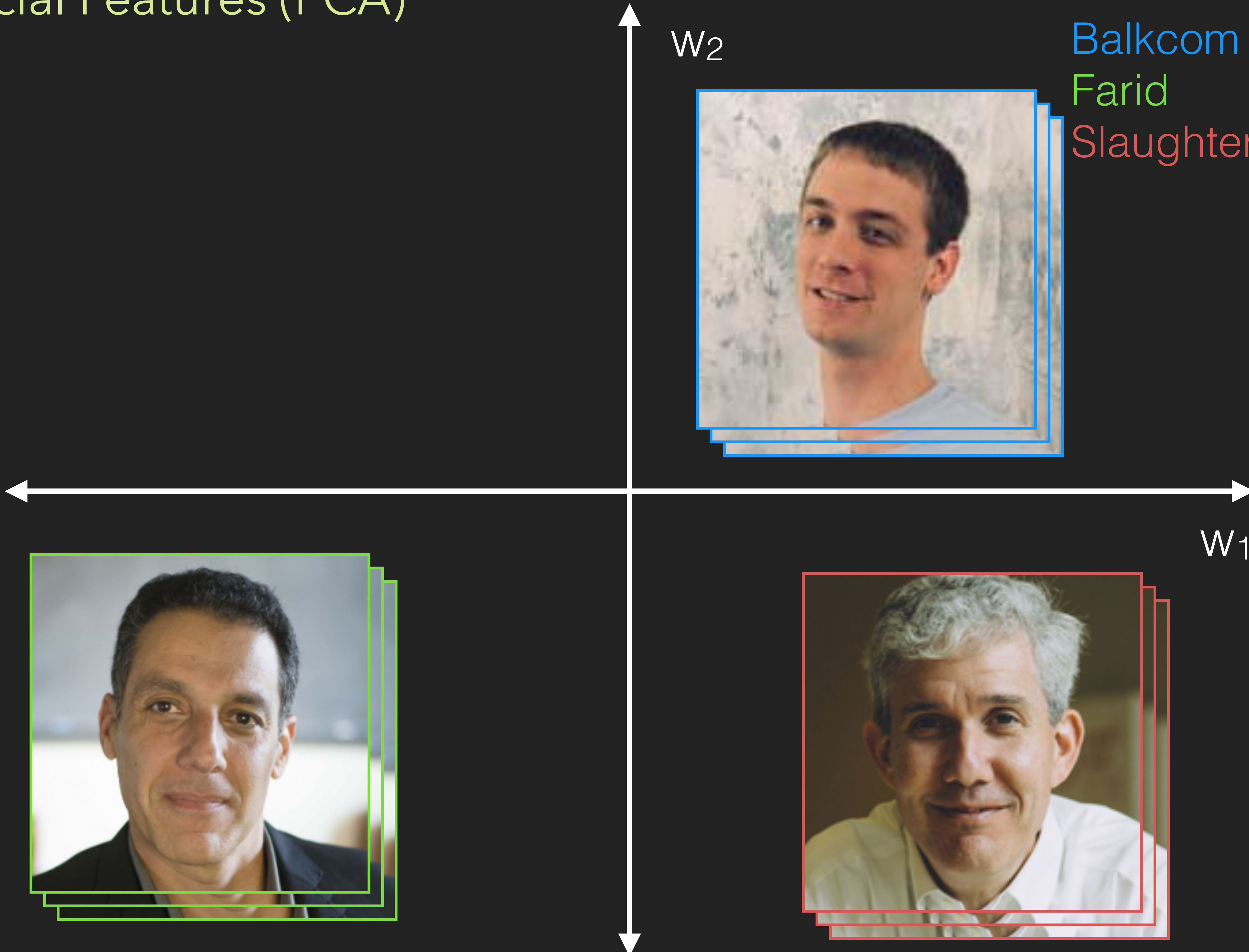
An N x N pixel image is a point in a N^2 dimensional space

Given a large set of <u>training</u> images construct a PCA basis (typically of dimensionality ~20)

Project all <u>training</u> images onto new basis

Classify a <u>testing</u> image by projecting and finding nearest neighbor

Facial Features (PCA)



Facial Features (PCA)

Challenges:

invariance to head pose, lighting, facial features, accessories

discrimination on a large-scale (7+ billion humans)

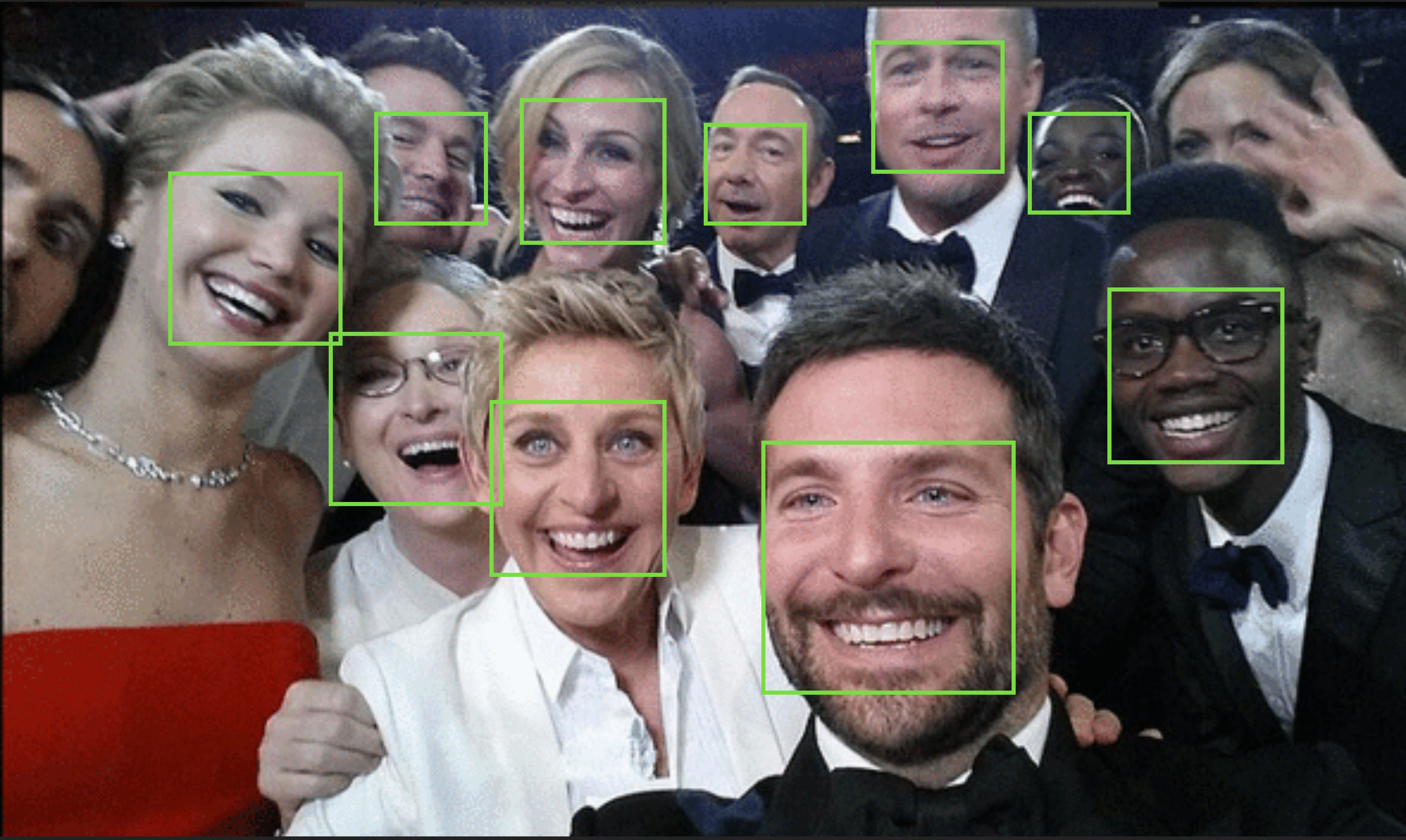
rebroadcast attacks

Facial Features (PCA)

Implementation:

openCV

Finding Faces in Images (Haar Cascades)



Finding Faces in Images (Haar Cascades)

Finding Faces in Images (Haar Cascades)

Finding Faces in Images (Haar Cascades)

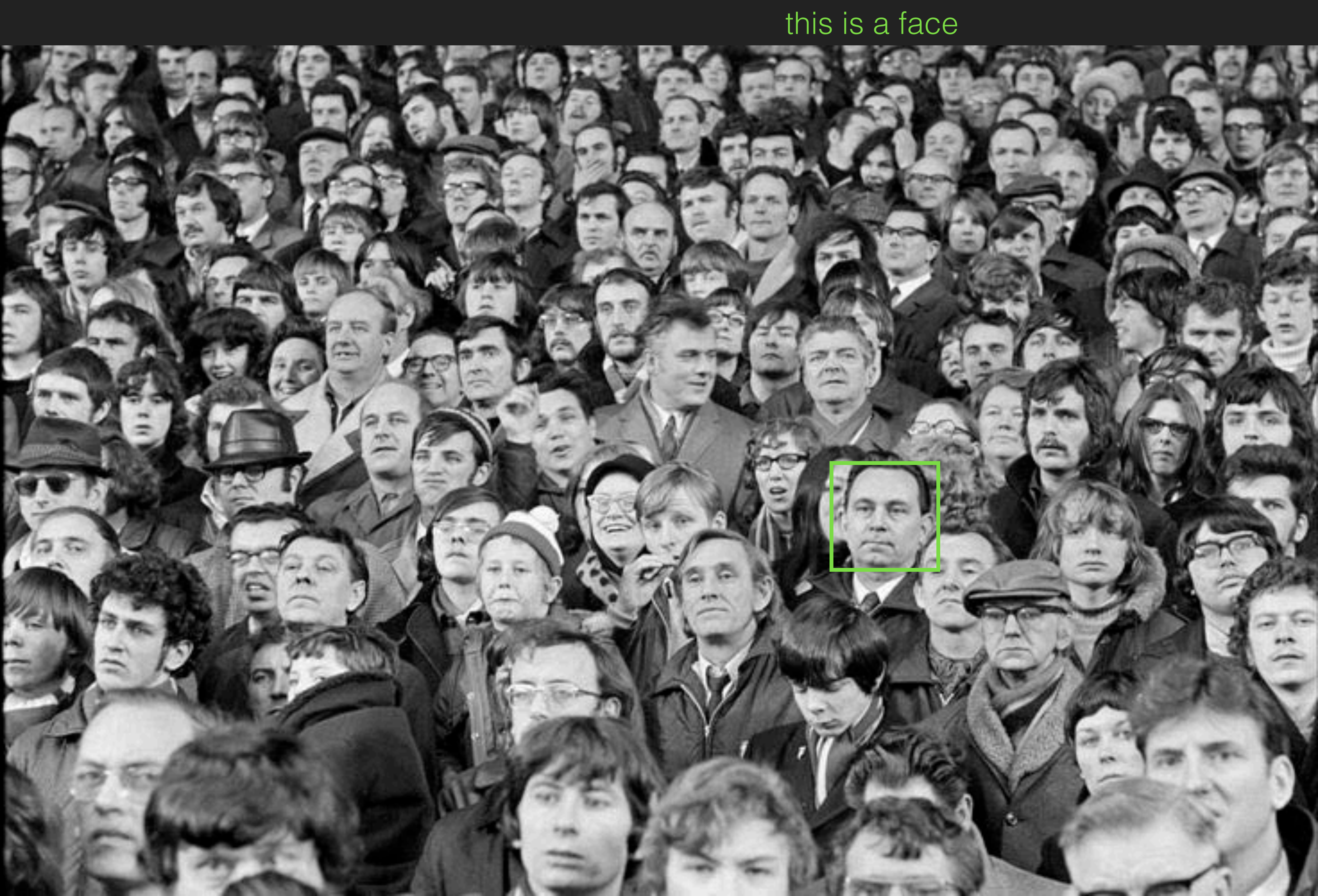
Finding Faces in Images (Haar Cascades)

1	1
1	1
-1	-1
-1	-1

70	244	129	38	236	89
173	86	178	65	89	211
167	149	227	214	50	149
41	57	244	64	64	140
30	191	139	207	157	233
127	65	35	62	120	72

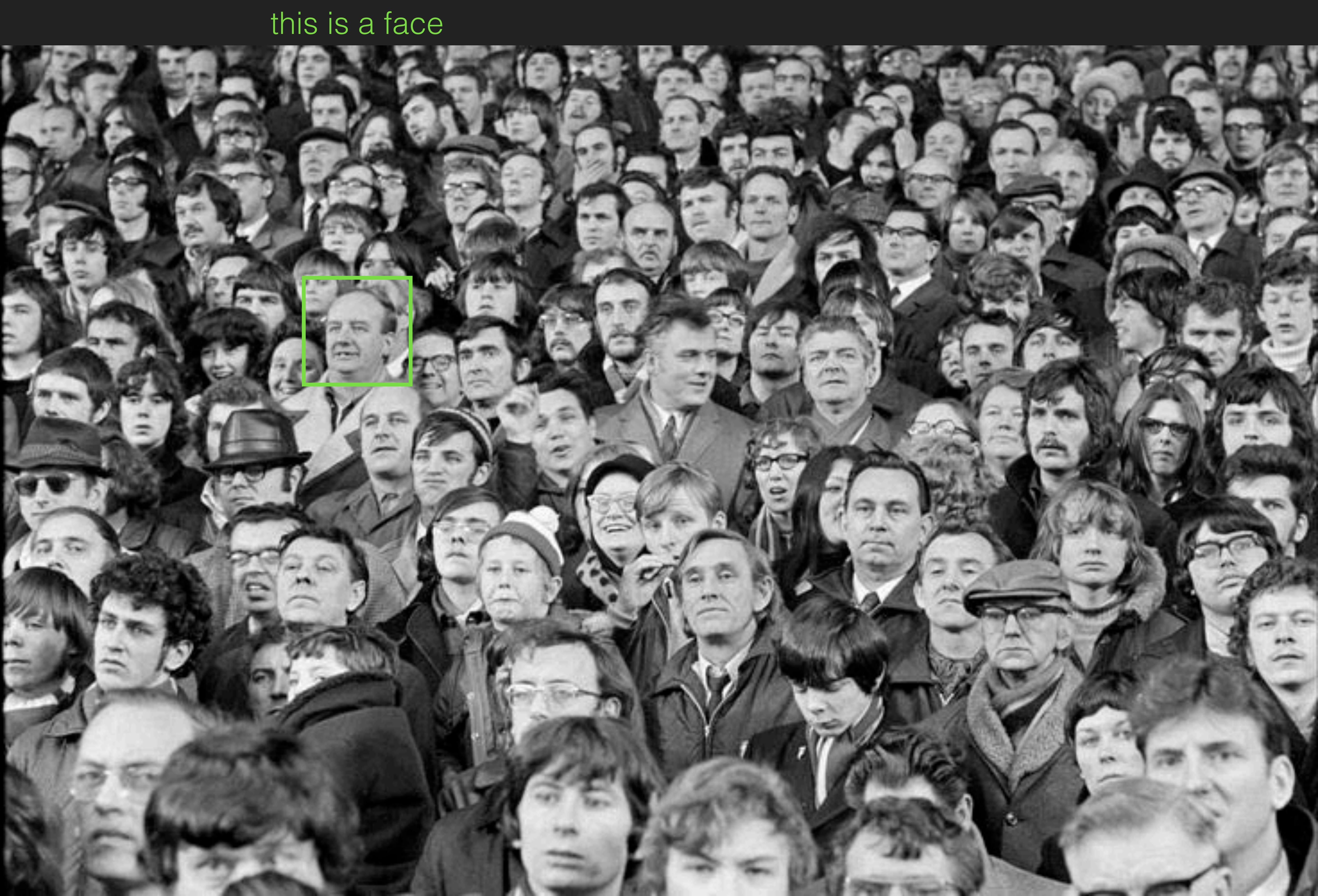
Finding Faces in Images (Haar Cascades)

Finding Faces in Images (Haar Cascades)



Finding Faces in Images (Haar Cascades)

Finding Faces in Images (Haar Cascades)



Finding Faces in Images (Haar Cascades)

Finding Faces in Images (Haar Cascades)

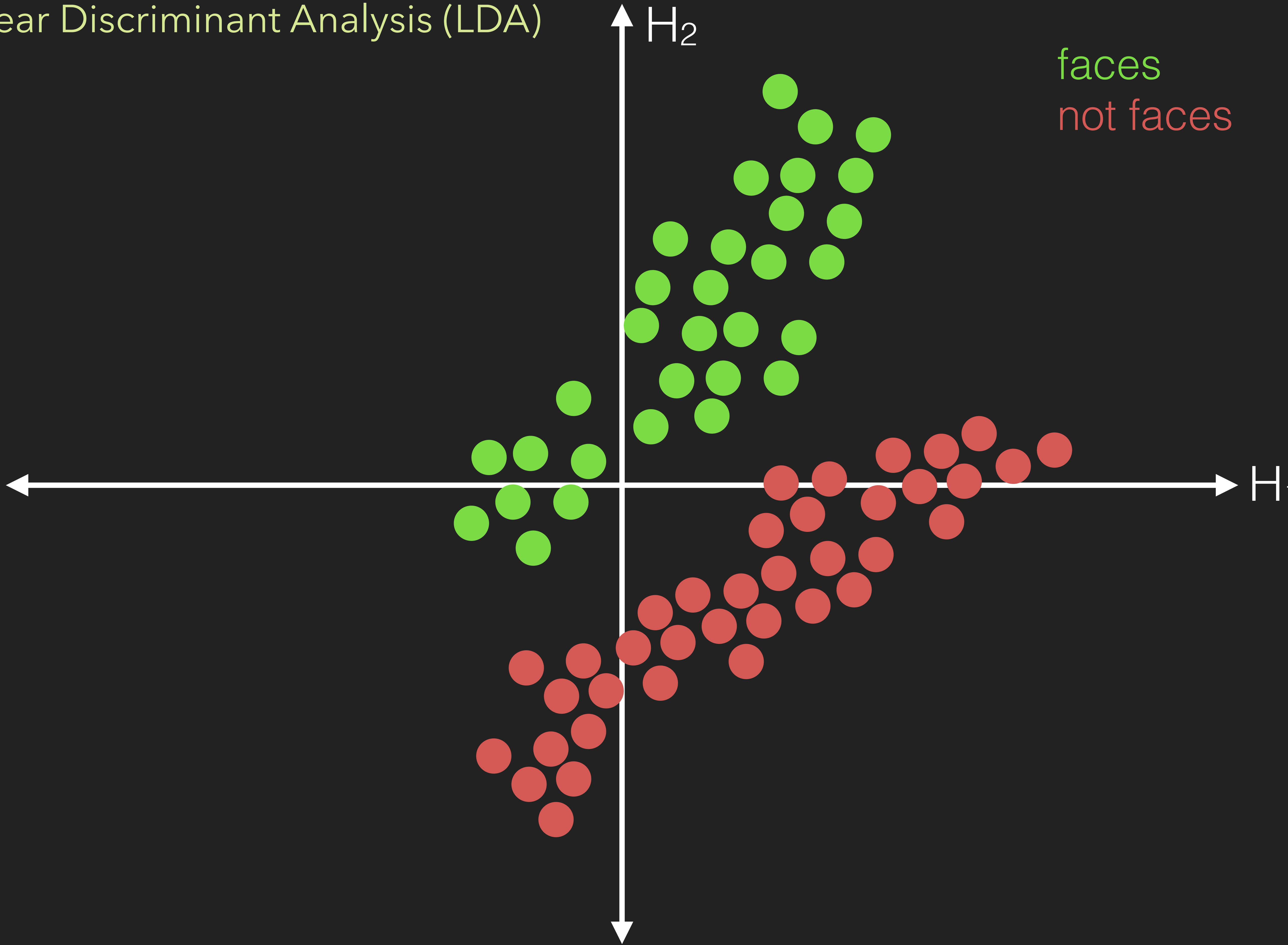
Finding Faces in Images (Haar Cascades)

build a large database of faces
extract features (Haar response)

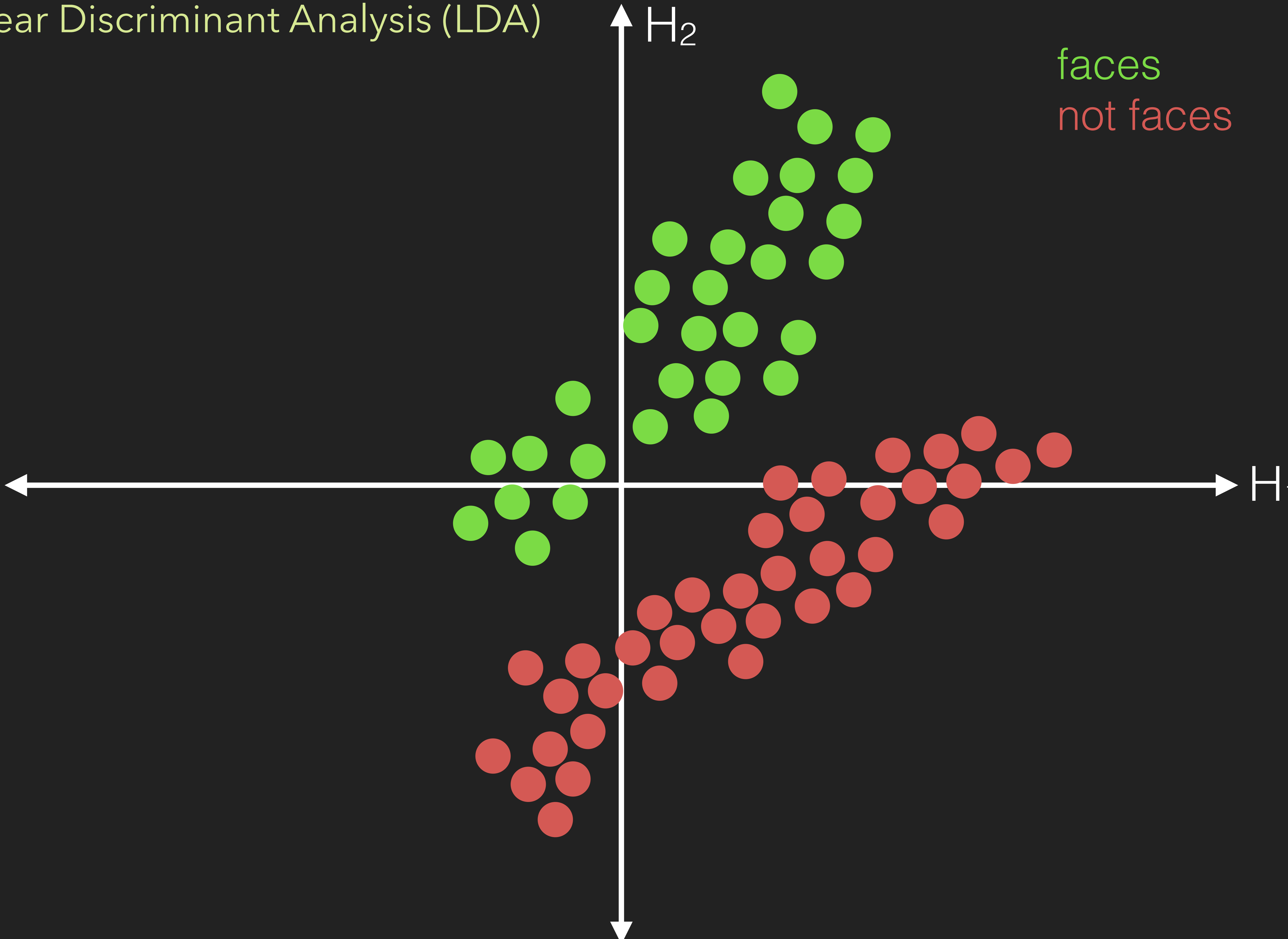
build a large database of non-faces
extract features (Haar response)

train a classifier

Linear Discriminant Analysis (LDA)



Linear Discriminant Analysis (LDA)



Project into low-dimensional space to maximize discriminability

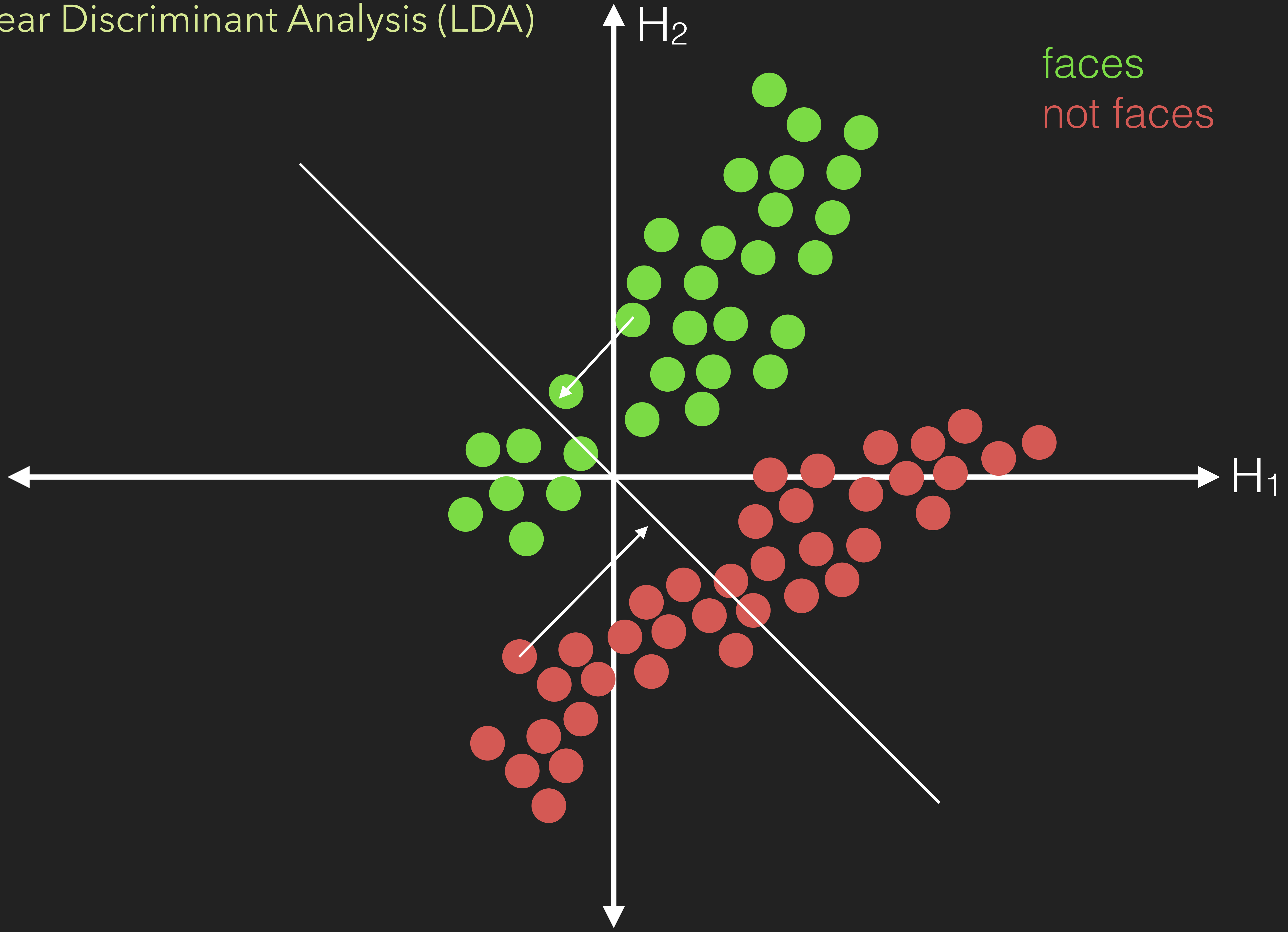
Linear Discriminant Analysis (LDA)

Project into low-dimensional space to maximize discriminability

Linear Discriminant Analysis (LDA)

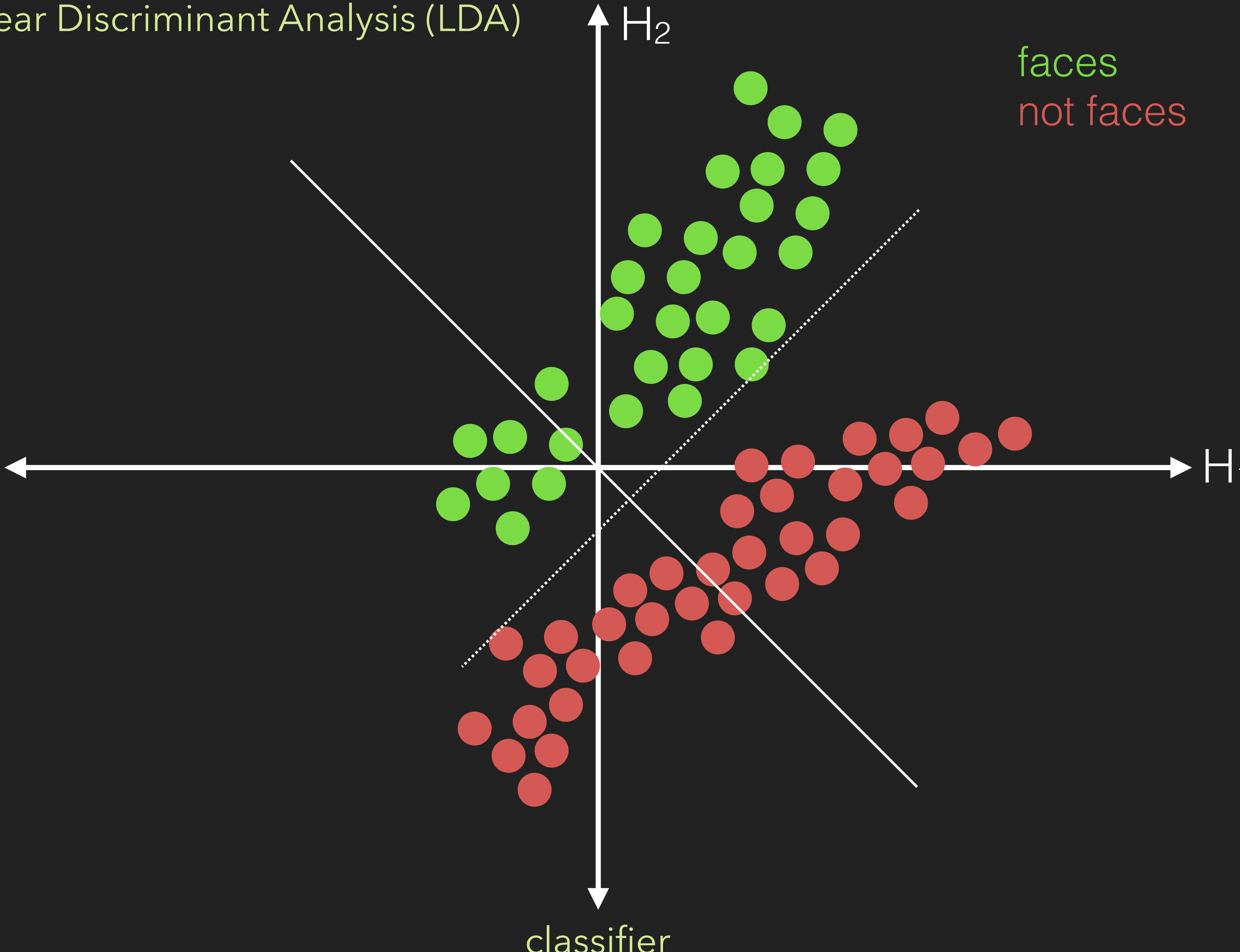
Project into low-dimensional space to maximize discriminability

Linear Discriminant Analysis (LDA)

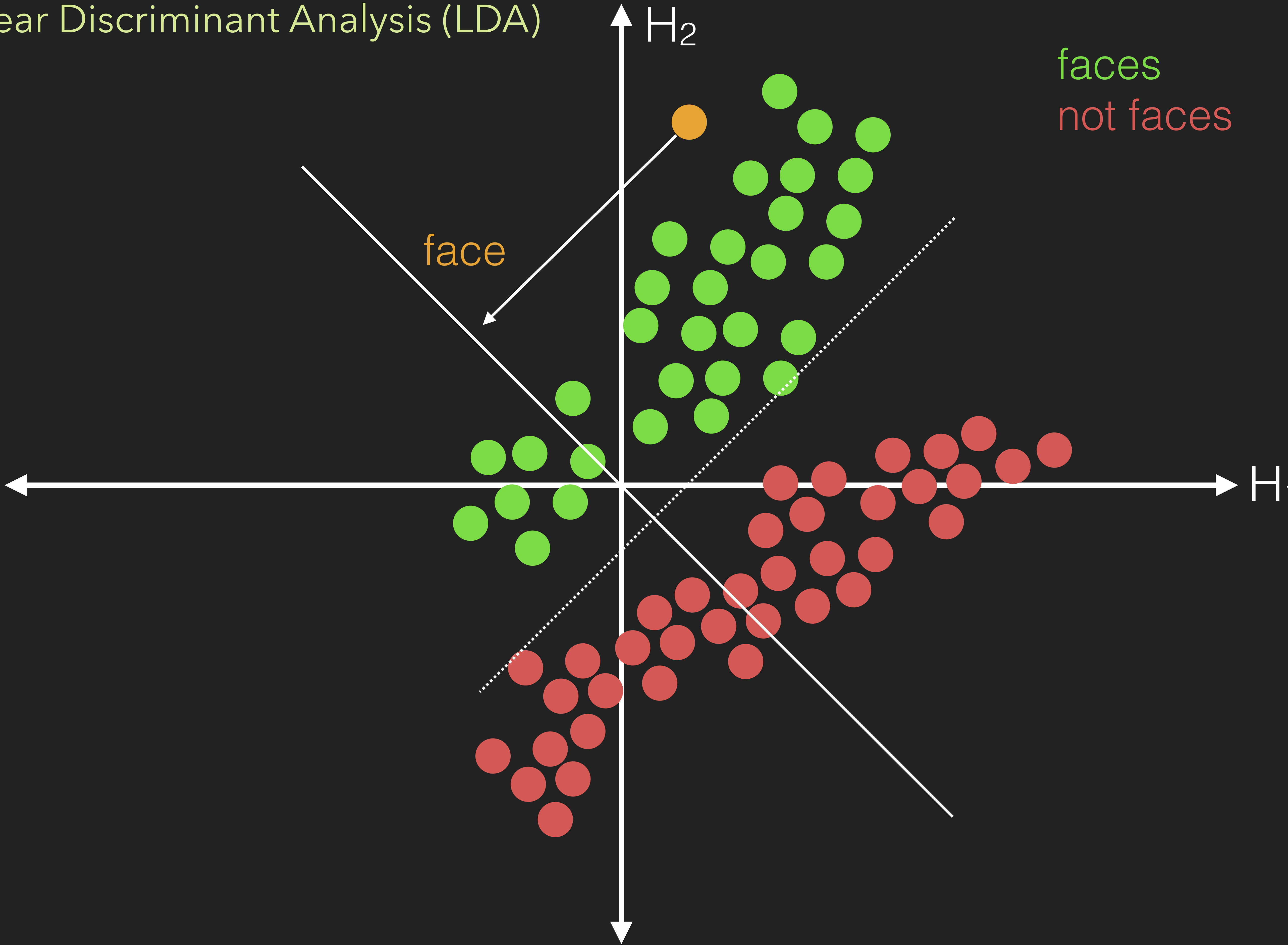


objective: minimize within-class variance & maximize across-class variance

Linear Discriminant Analysis (LDA)

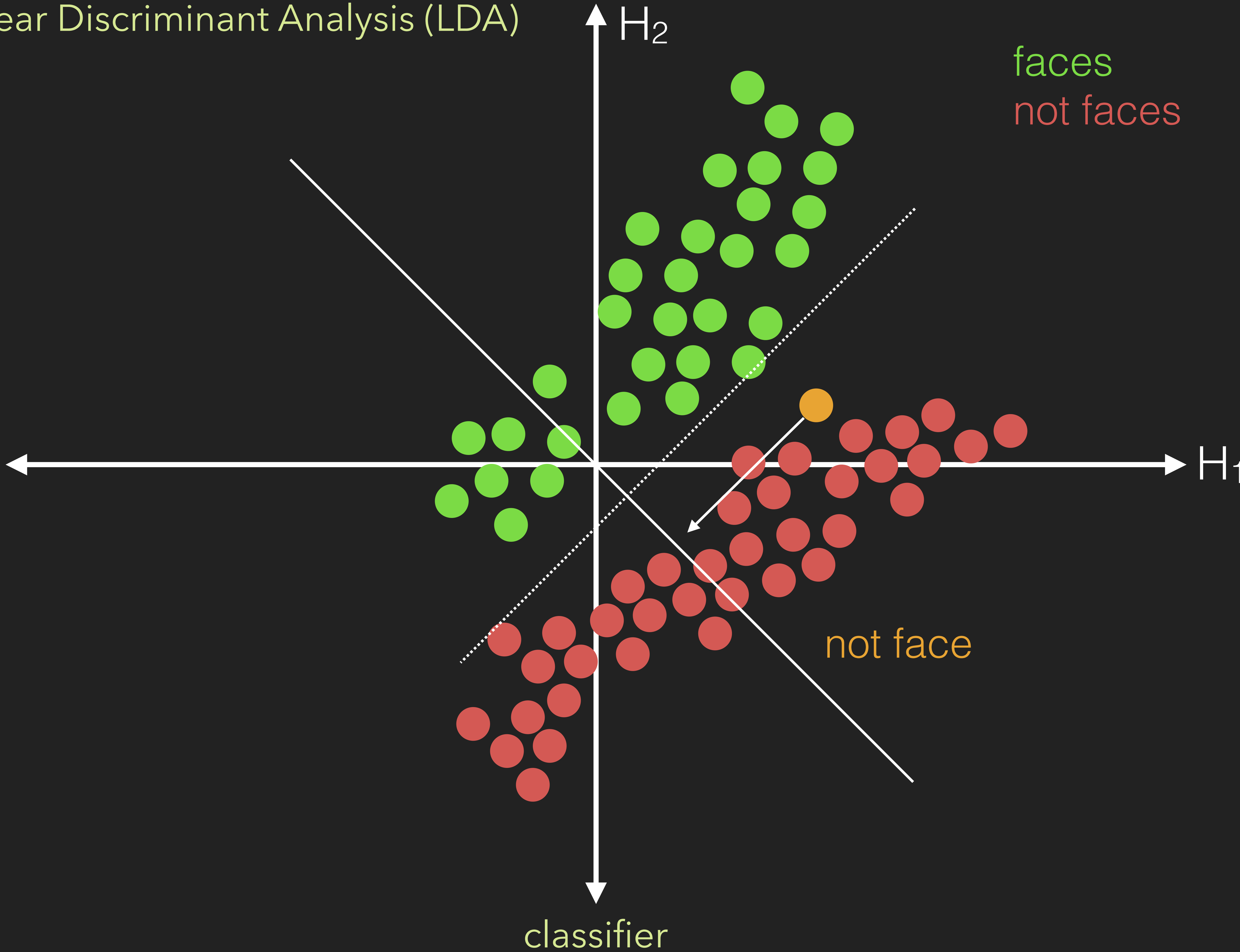


Linear Discriminant Analysis (LDA)

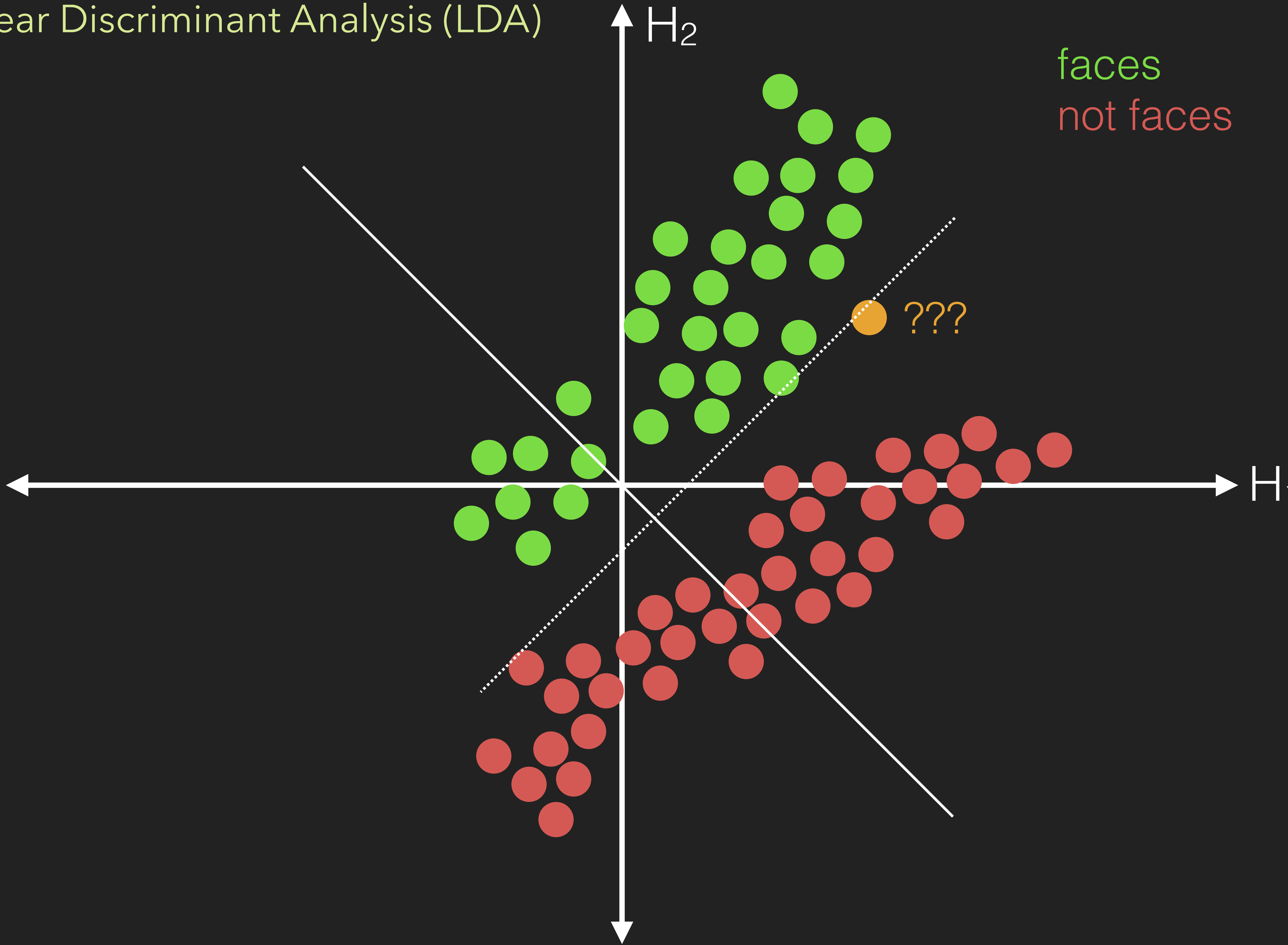


classifier

Linear Discriminant Analysis (LDA)

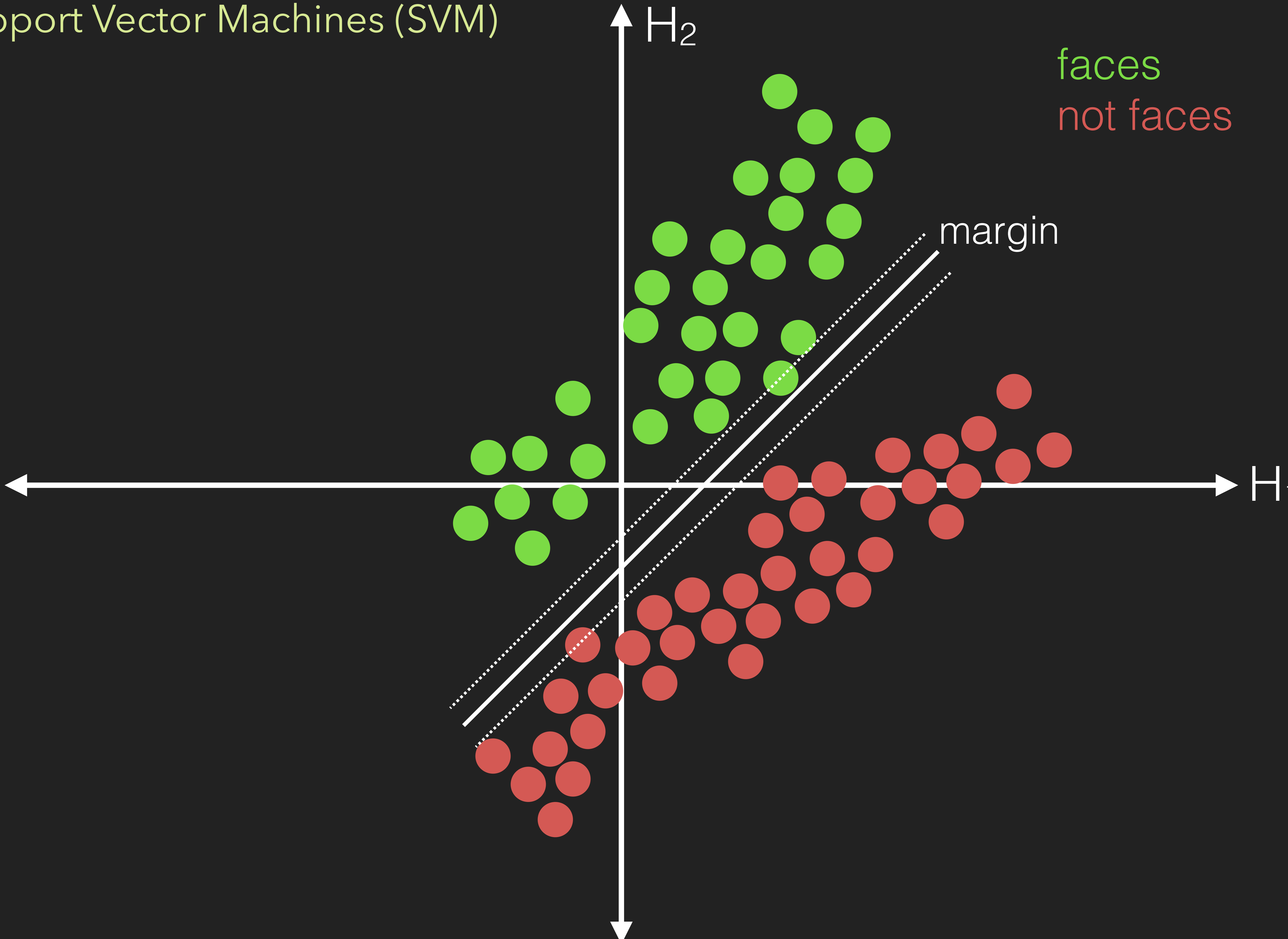


Linear Discriminant Analysis (LDA)



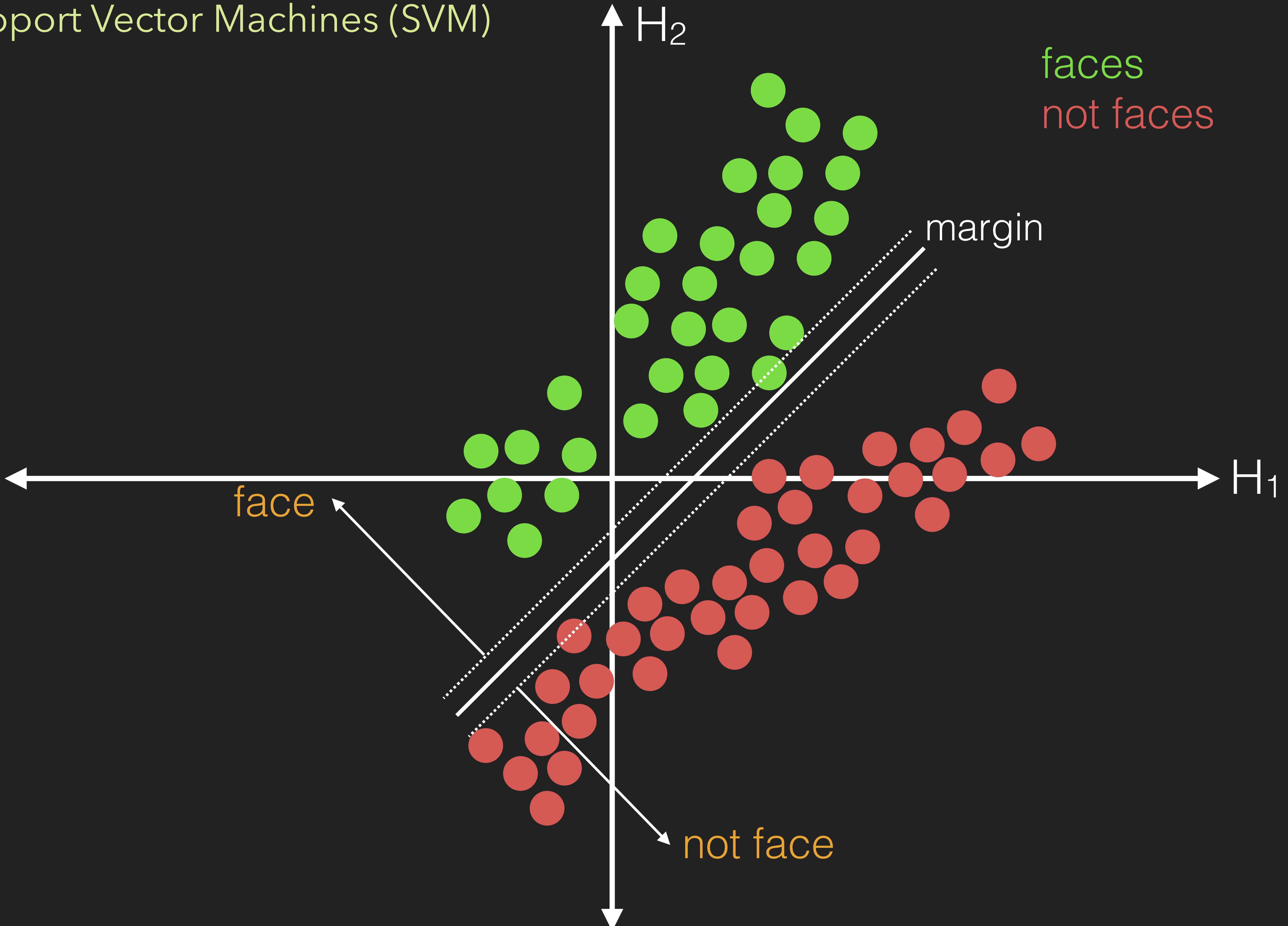
classifier

Support Vector Machines (SVM)



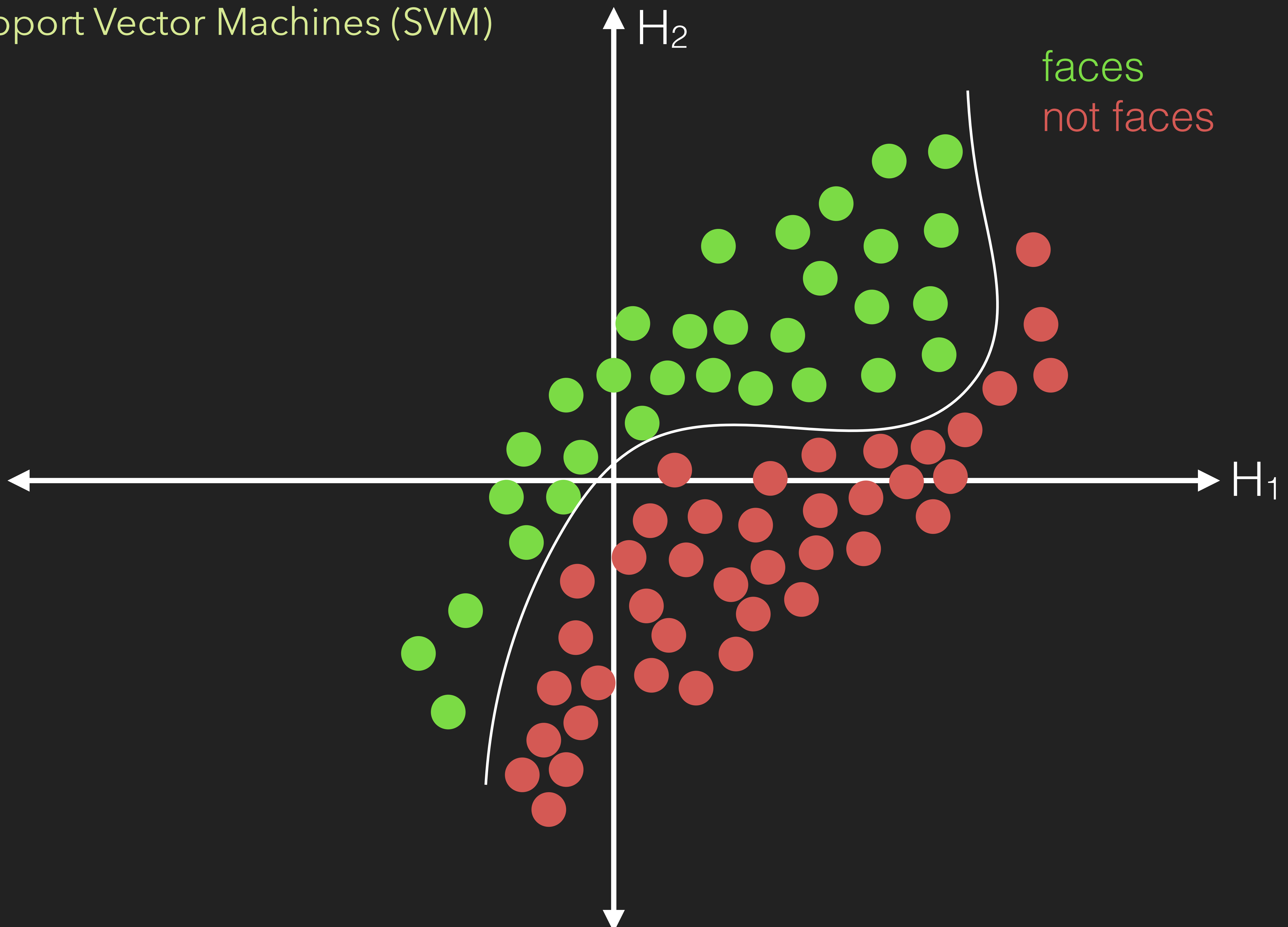
objective: maximize margin

Support Vector Machines (SVM)



objective: maximize margin

Support Vector Machines (SVM)



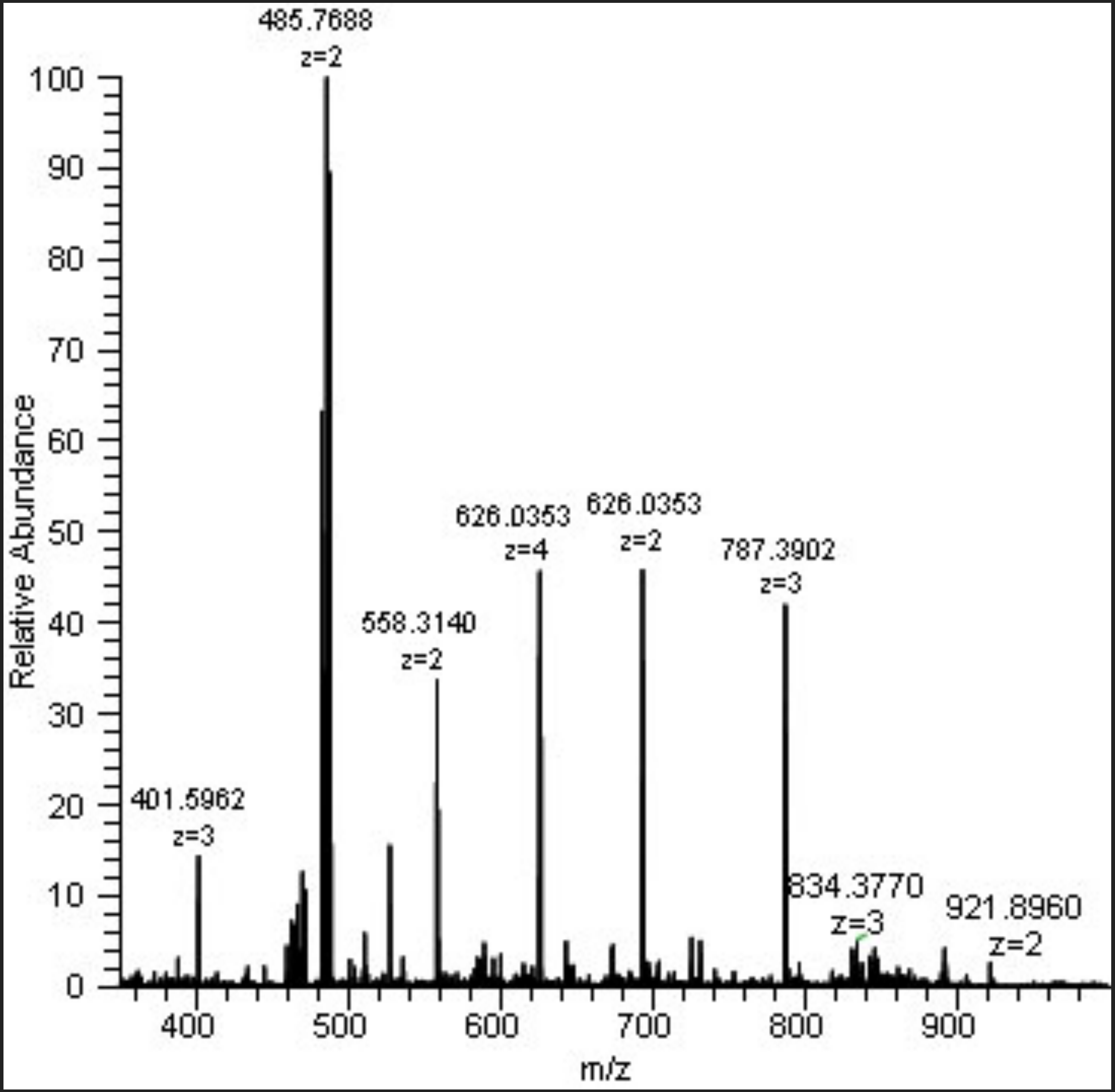
linear vs. non-linear SVM

Finding Faces in Images (Haar Cascades, LDA, SVM)

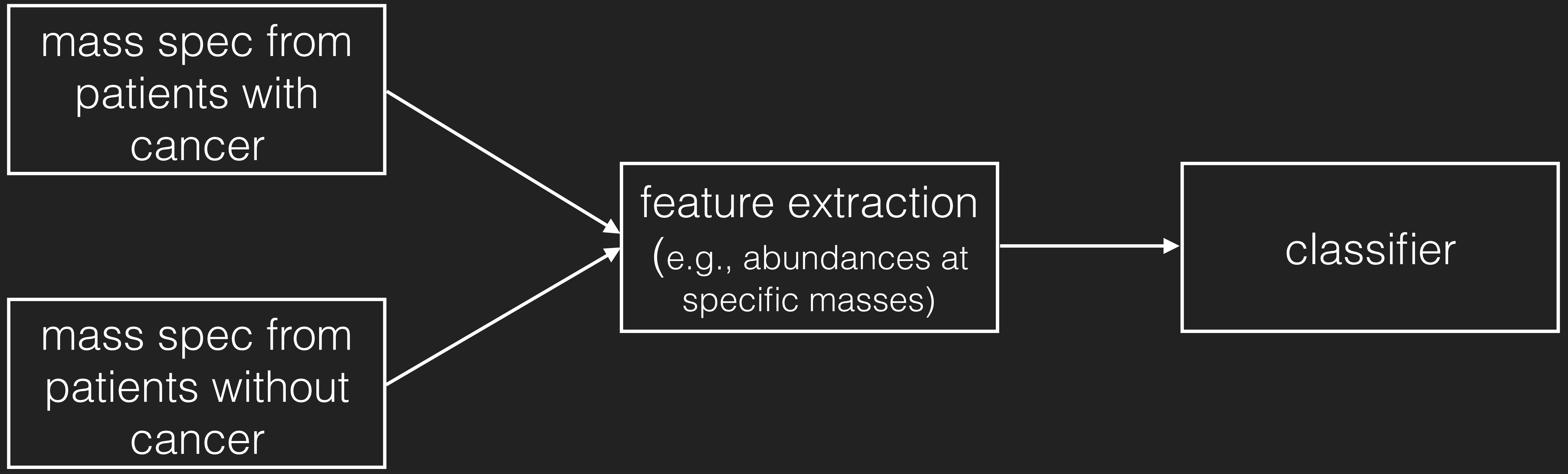
Implementation:

openCV

Cancer Diagnosis

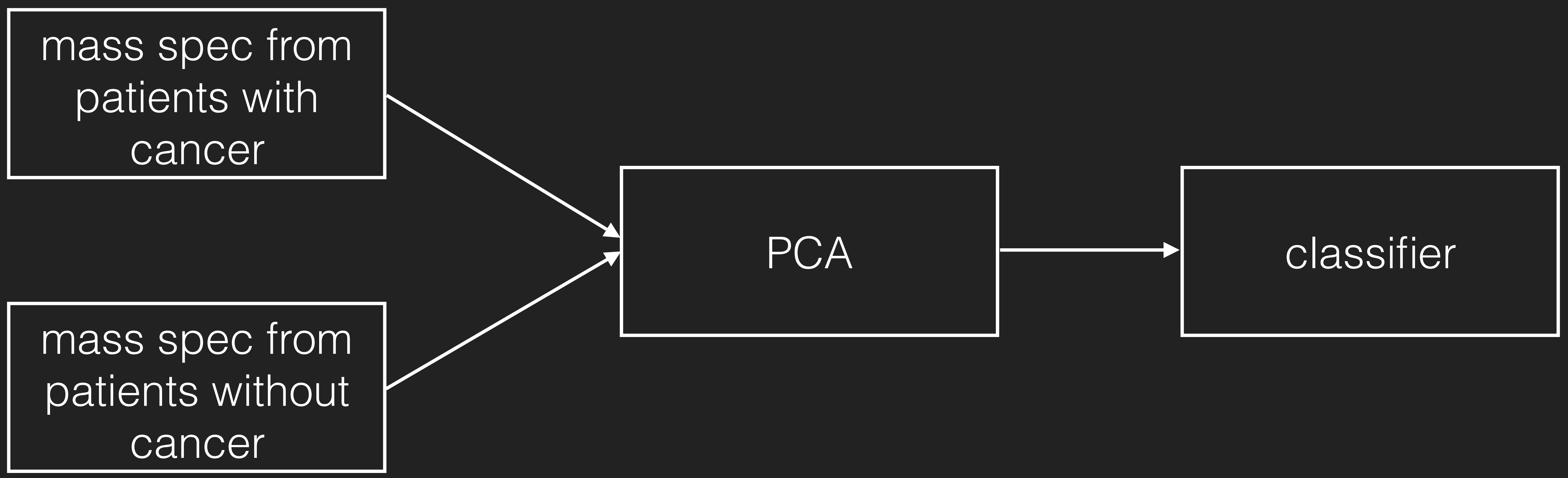


Cancer Diagnosis



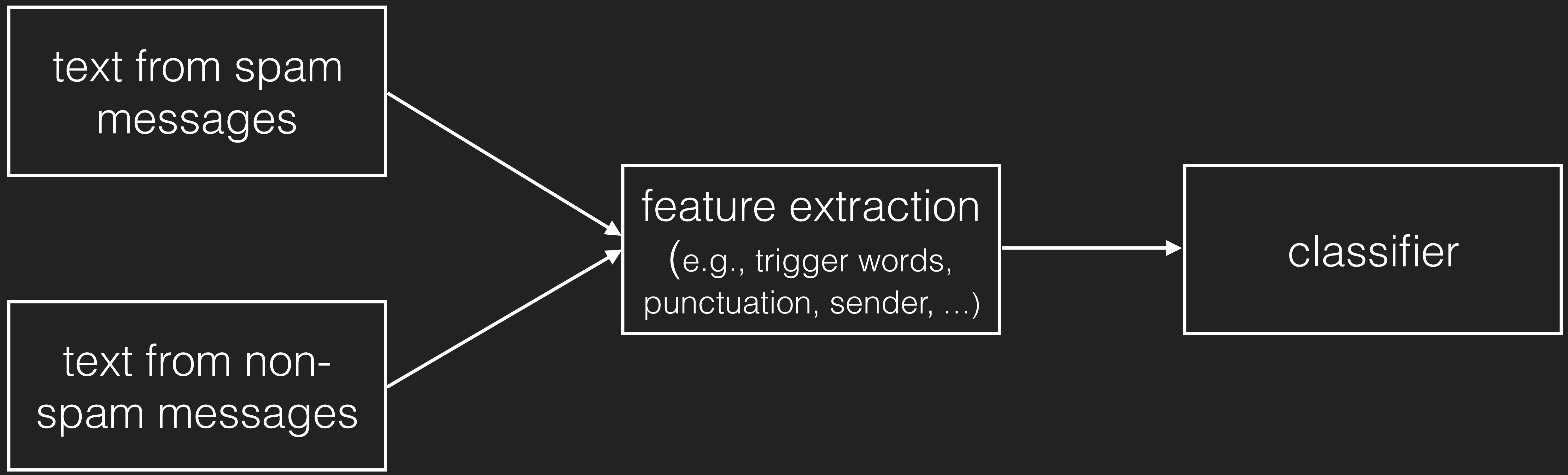
hypothesis-driven

Cancer Diagnosis

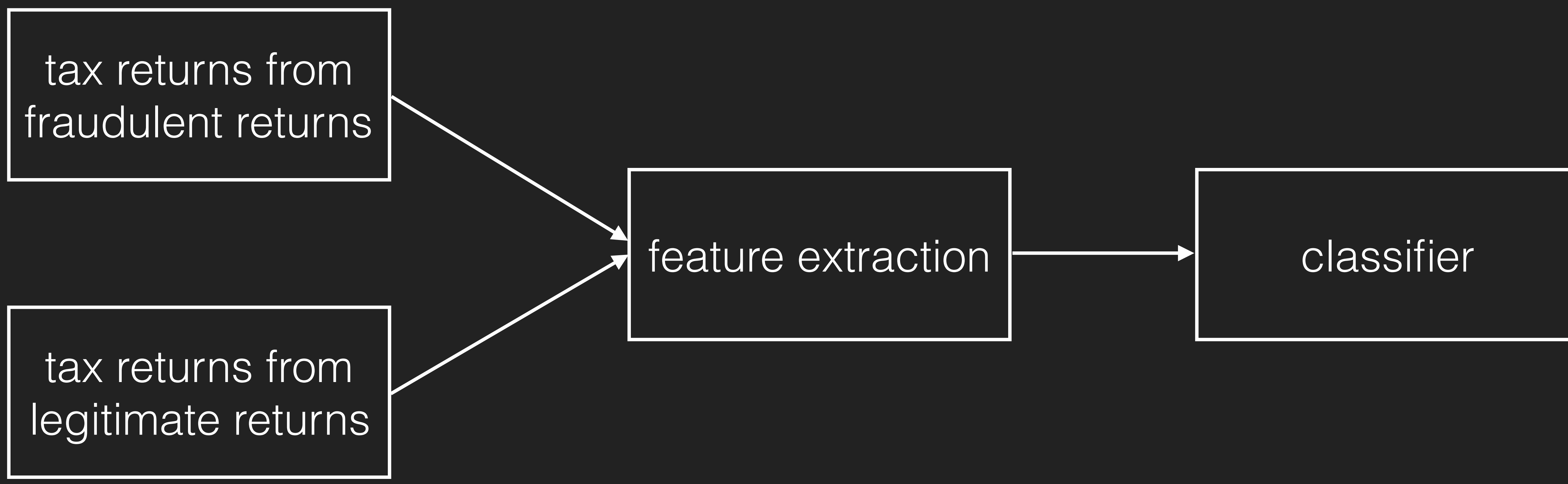


discovery-driven

Spam

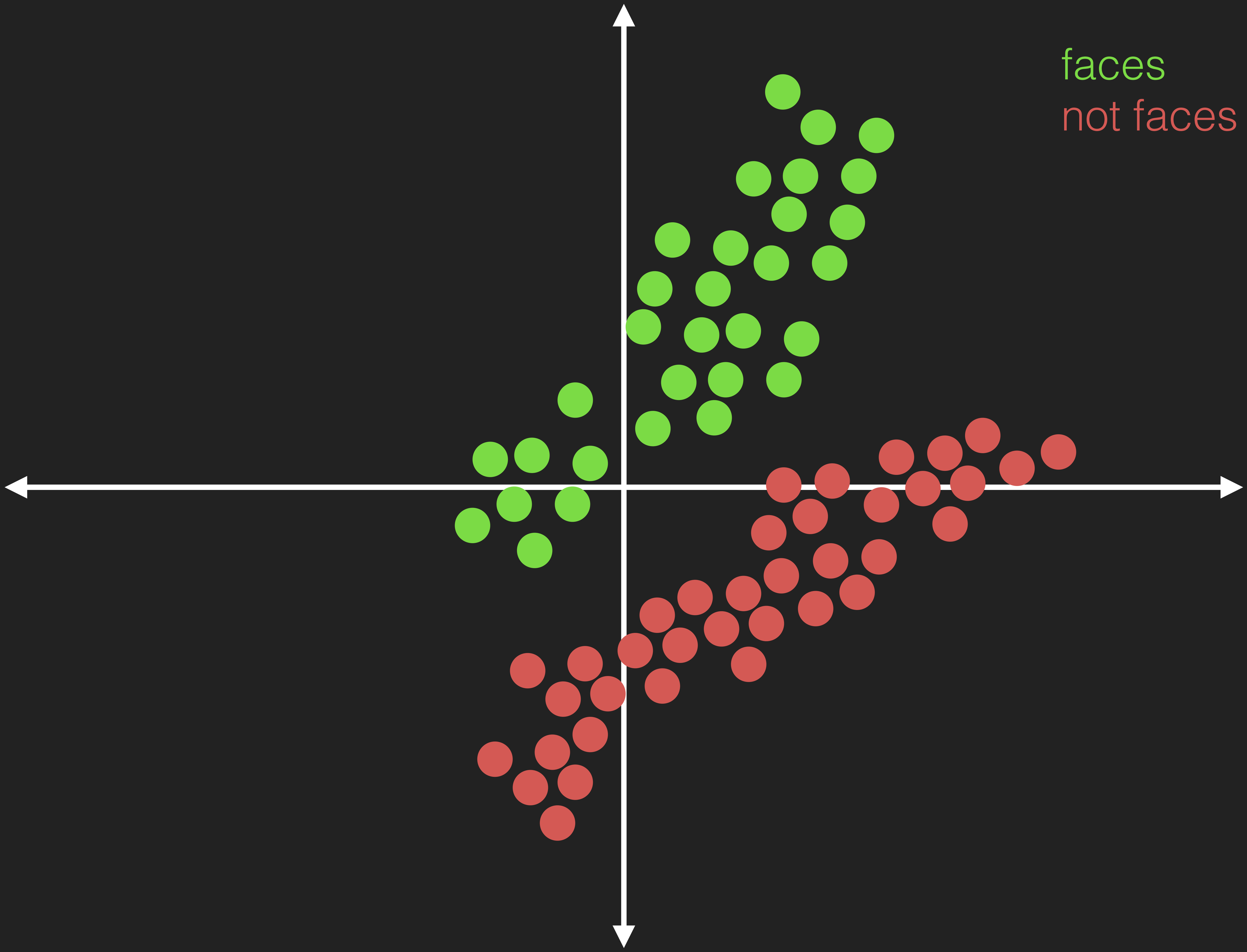

text from spam messages
text from non-spam messages
feature extraction
(e.g., trigger words, punctuation, sender, …)
classifier

Financial Fraud

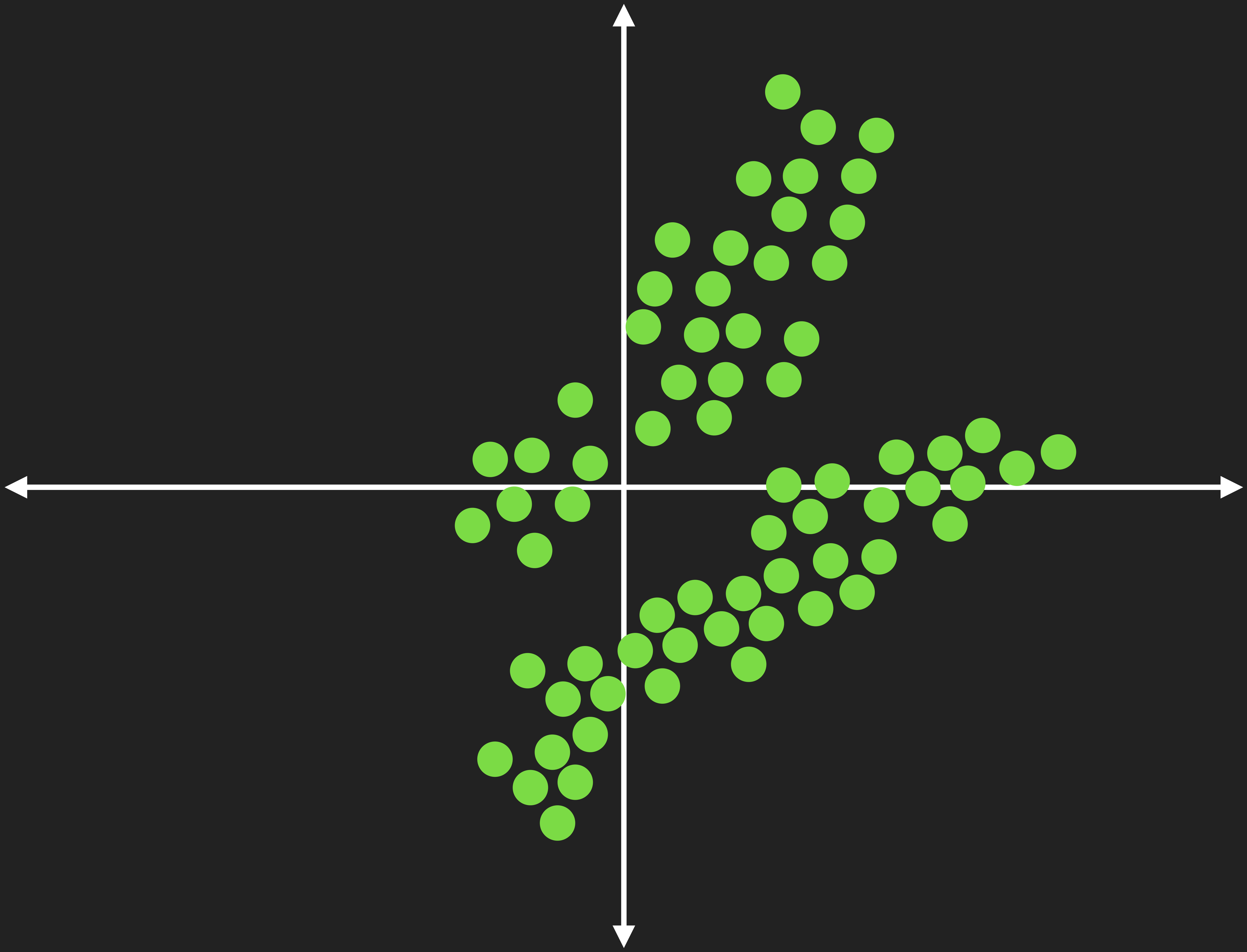


An artificial network consists of a pool of simple processing units which communicate by sending signals to each other over a large number of weighted connections.

more tomorrow…

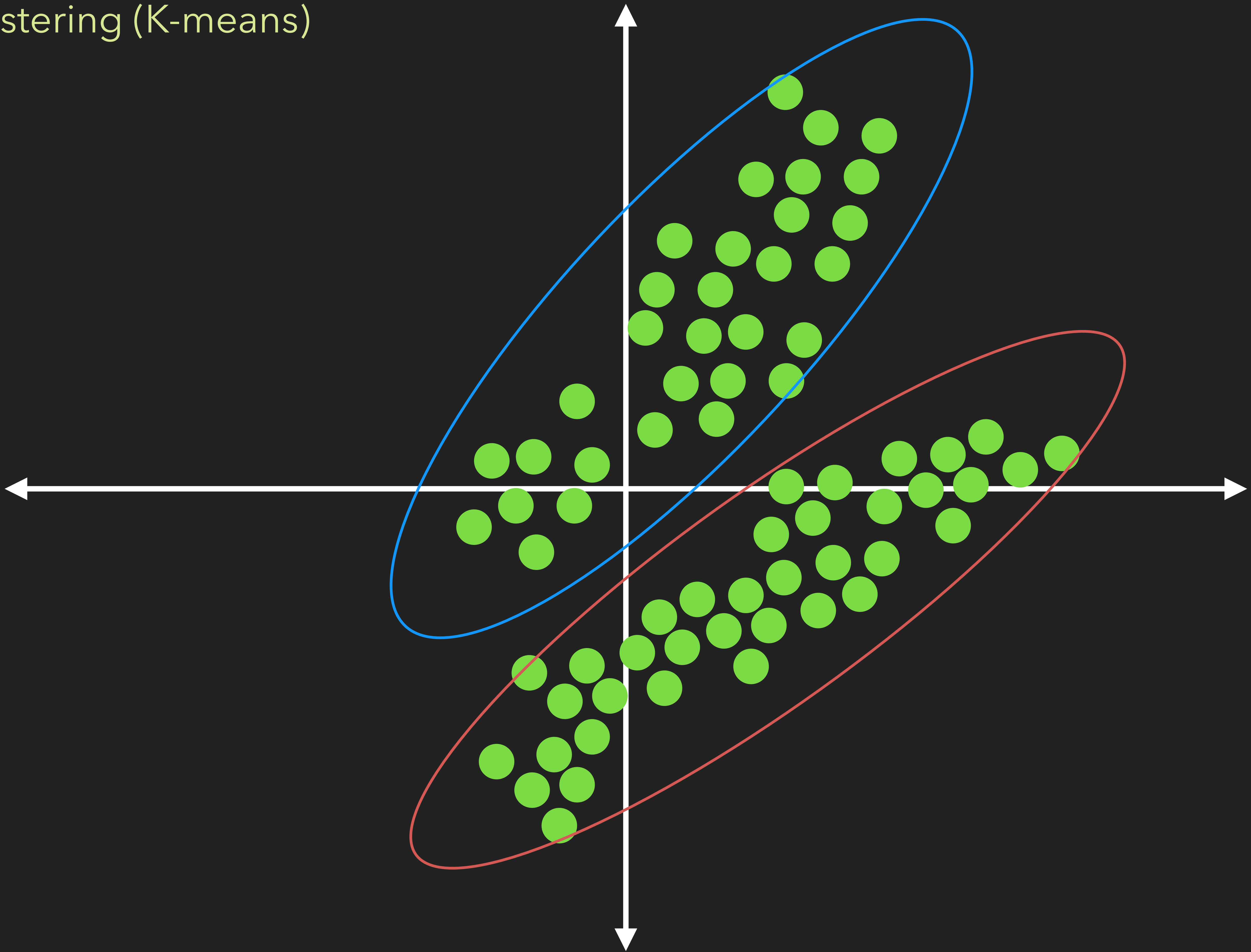


Supervised Learning



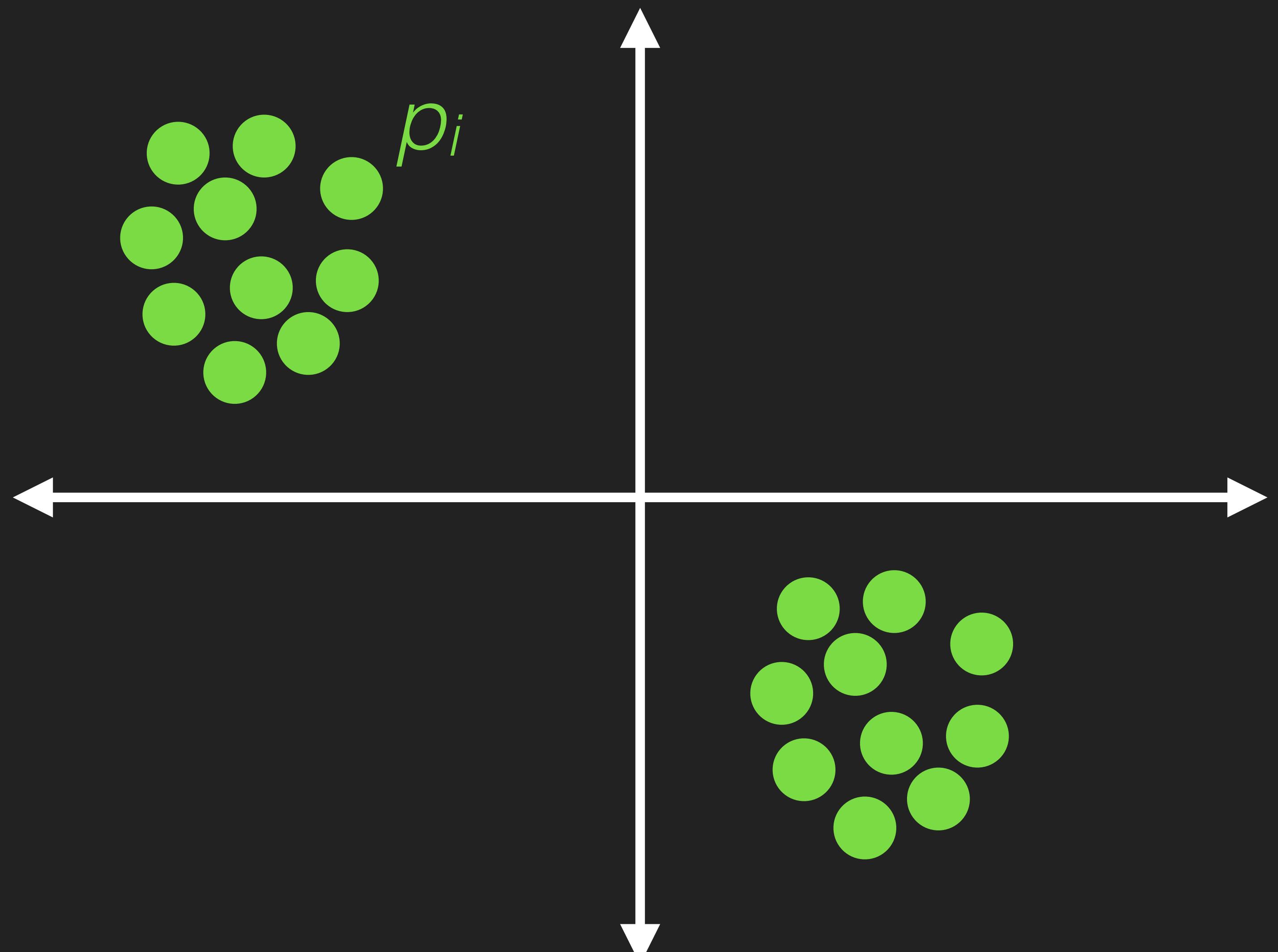
Unsupervised Learning

Clustering (K-means)



Unsupervised Learning

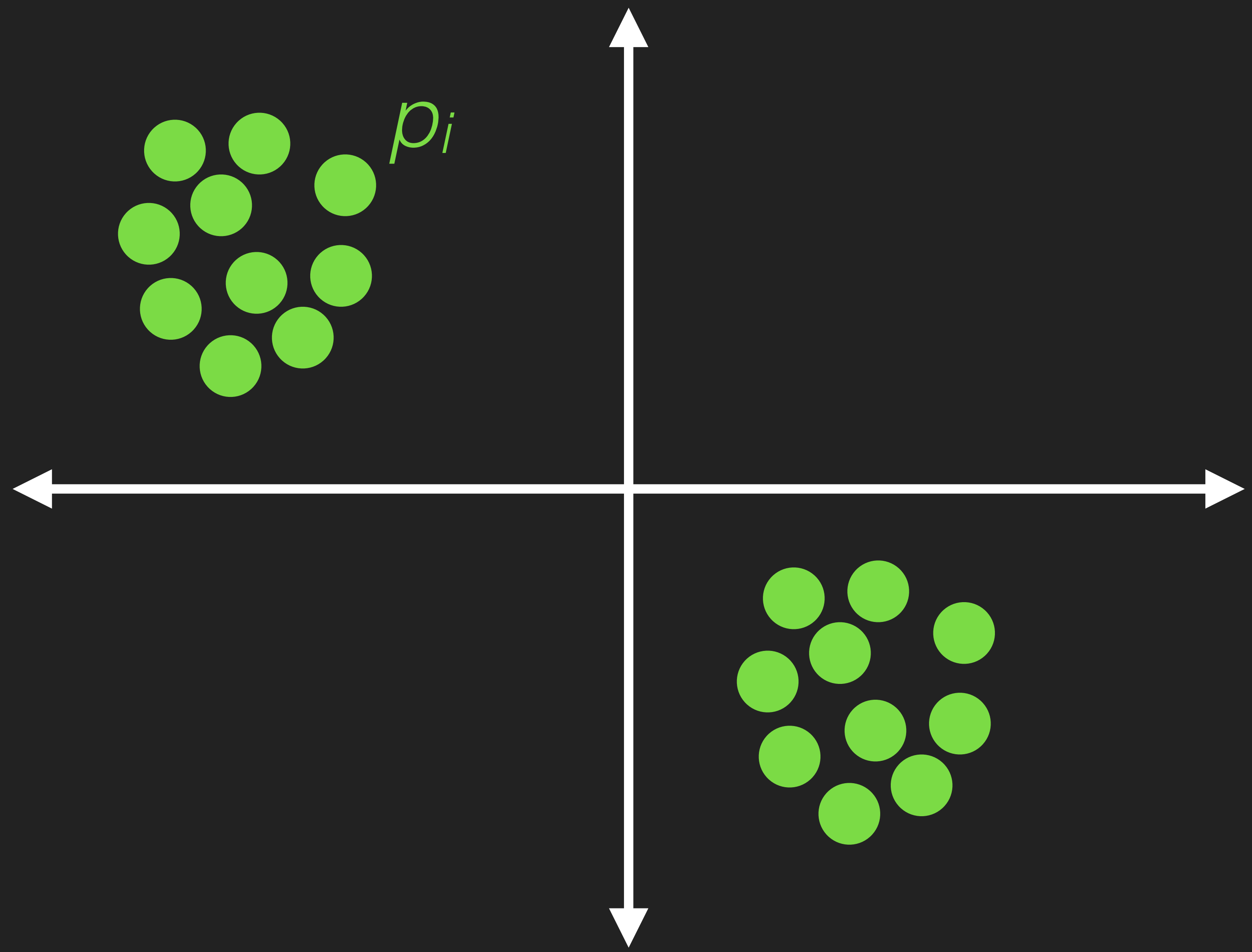
denote your data as p_i where $i = 1,2,\ldots,m$



Clustering (K-means)

denote your data as p_i where $i = 1,2,\ldots,m$

denote n as the number of clusters ($n=2$ in this example)

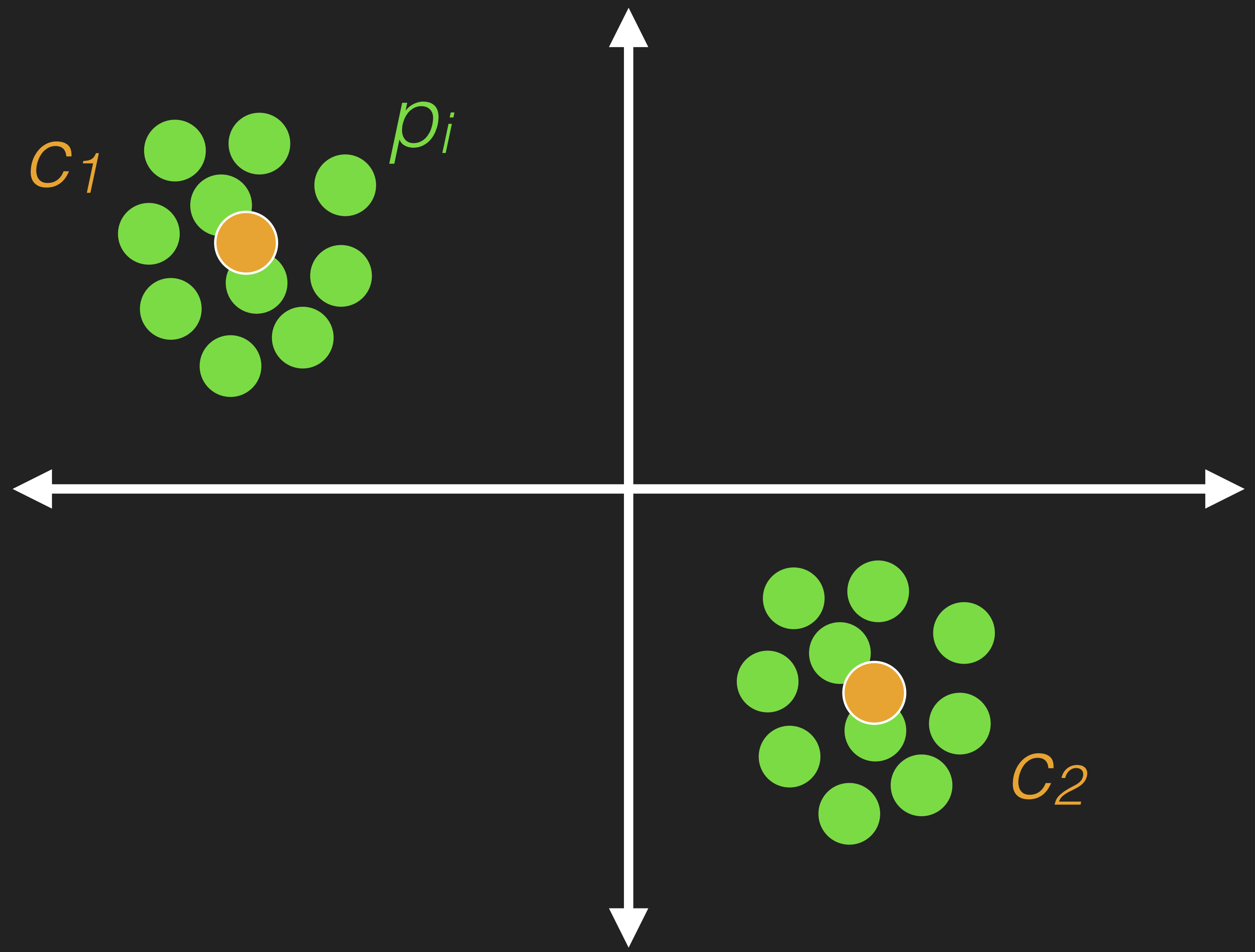


Clustering (K-means)

denote your data as p_i where $i = 1,2,\ldots,m$

denote n as the number of clusters ($n=2$ in this example)

we seek n cluster centers c_i

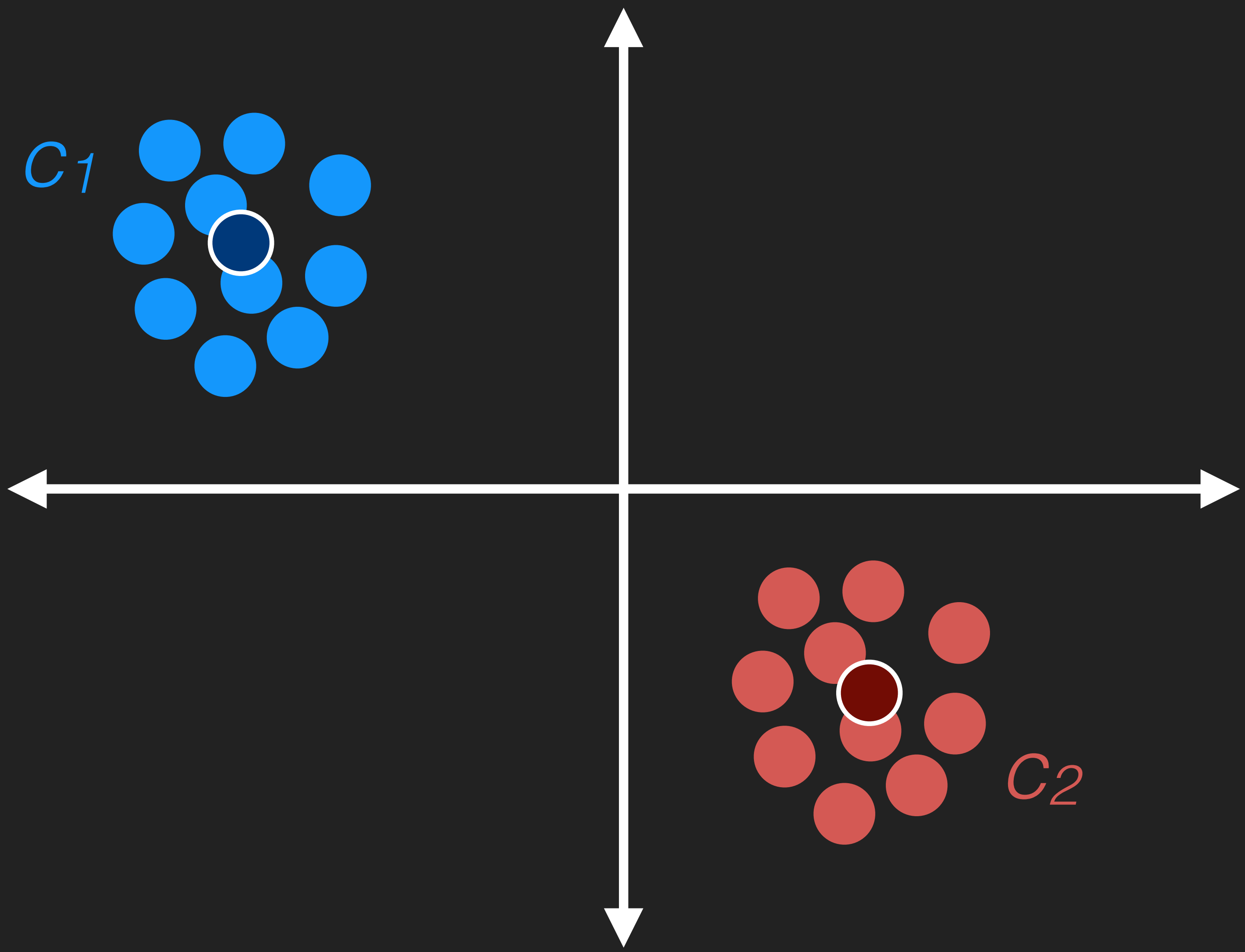


Clustering (K-means)

denote your data as p_i where $i = 1,2,\ldots,m$

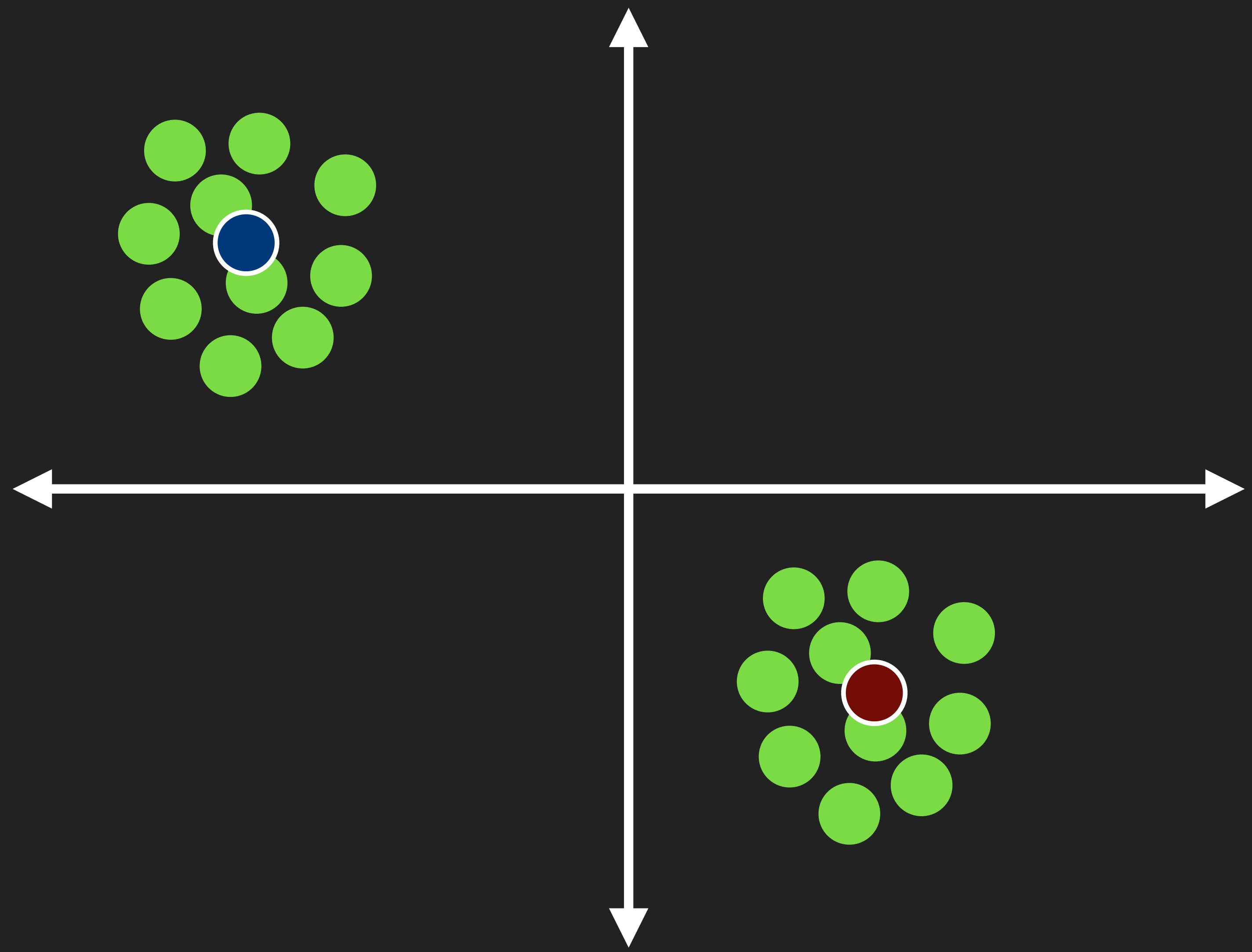
denote n as the number of clusters ($n=2$ in this example)

we seek n cluster centers c_i and an assignment of each p_i to a cluster

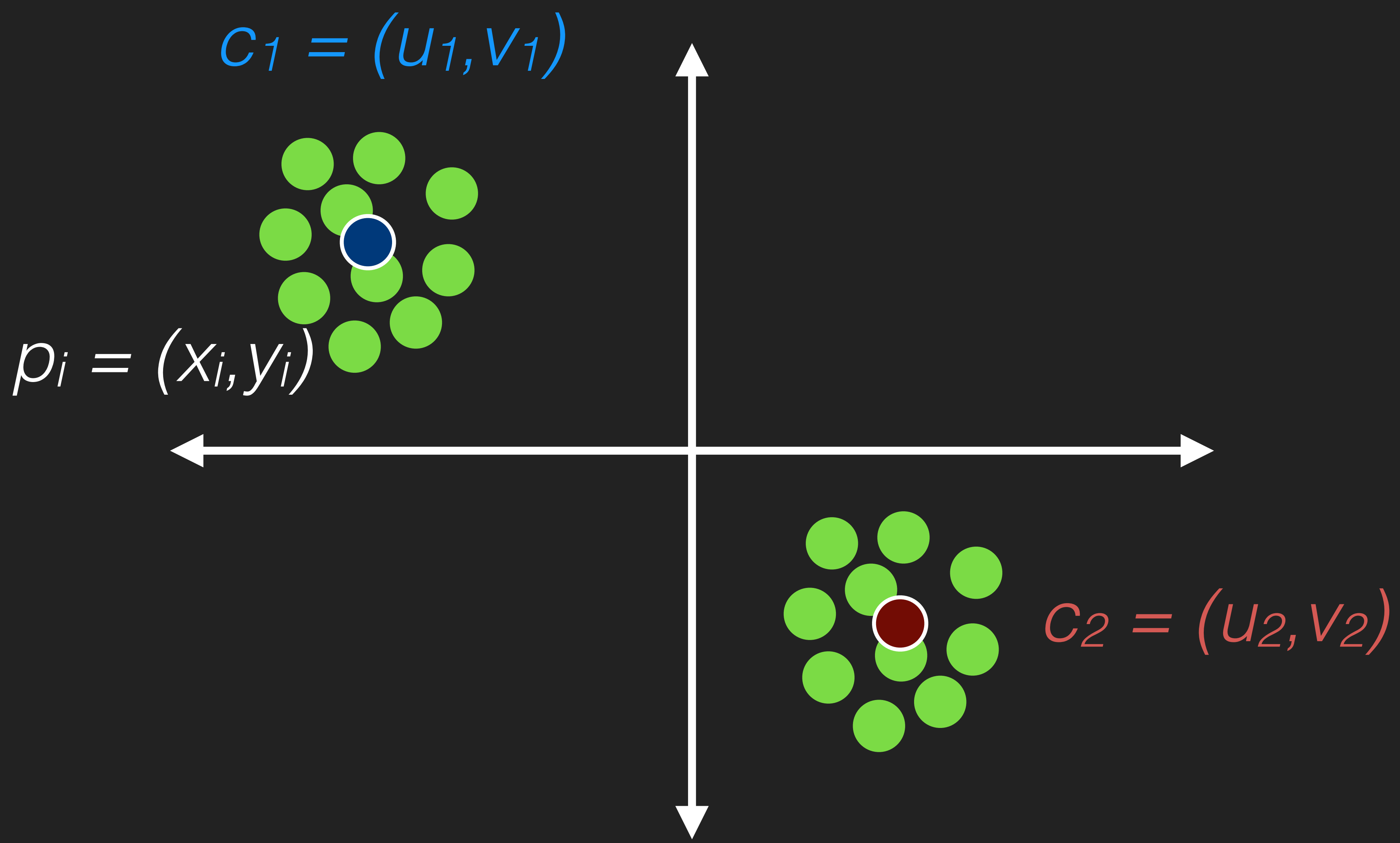


Clustering (K-means)

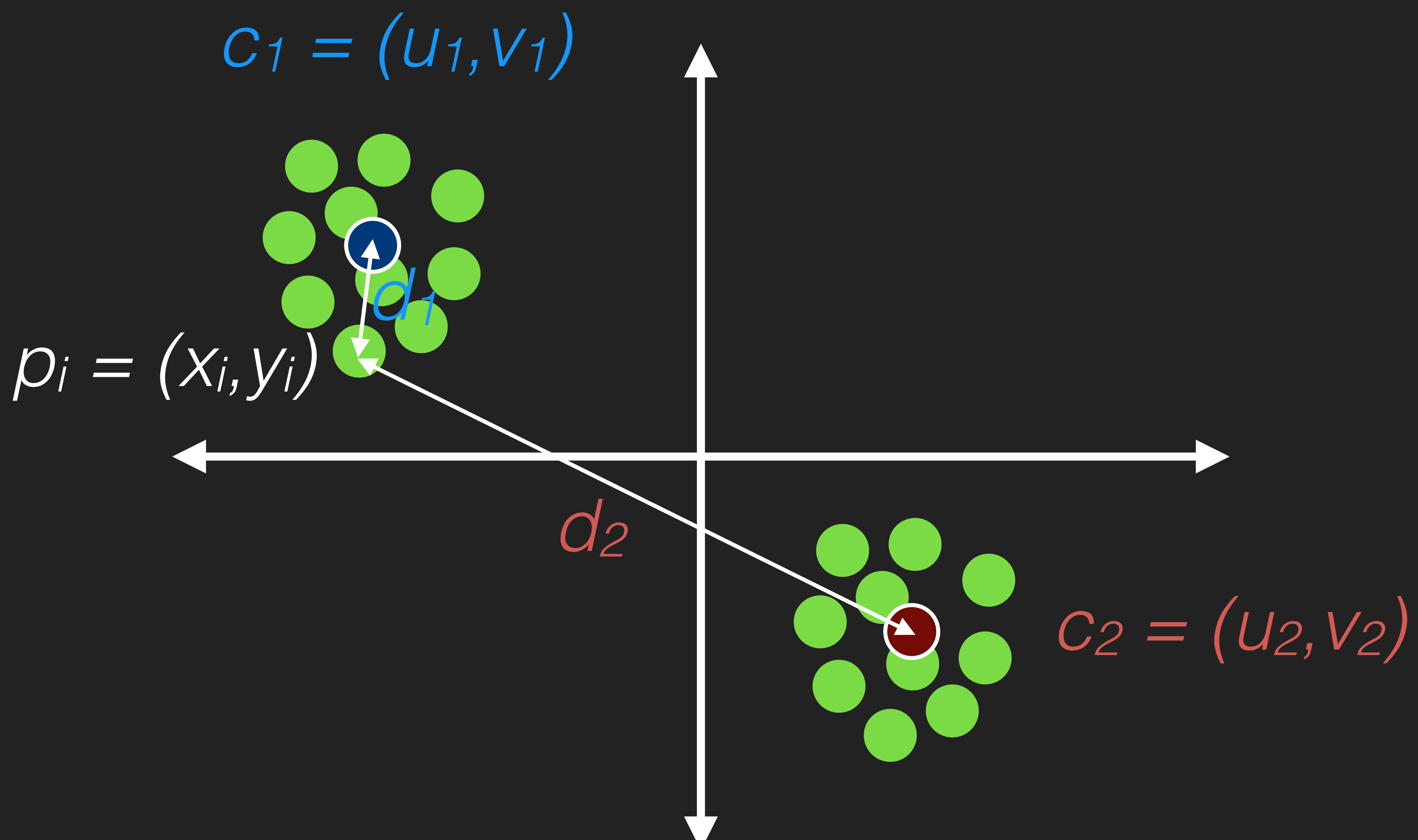
if we know c_i then assignment of each p_i to a cluster is easy



if we know c_i then assignment of each p_i to a cluster is easy



if we know c_i then assignment of each p_i to a cluster is easy

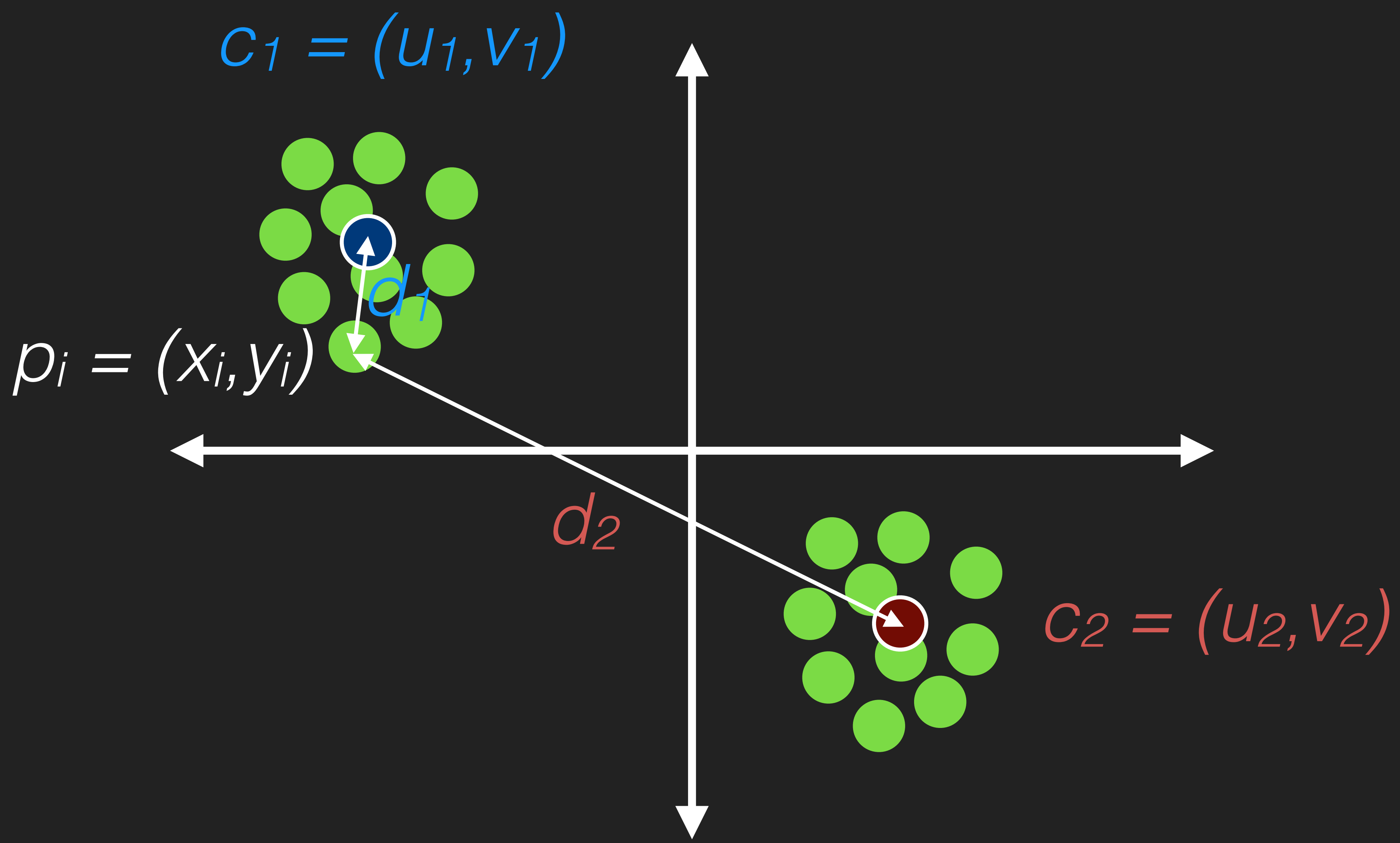


Clustering (K-means)

if we know c_i then assignment of each p_i to a cluster is easy

$d_1 = sqrt((x_i-u_1)^2 + (y_i-v_1)^2)$

$d_2 = sqrt((x_i-u_2)^2 + (y_i-v_2)^2)$

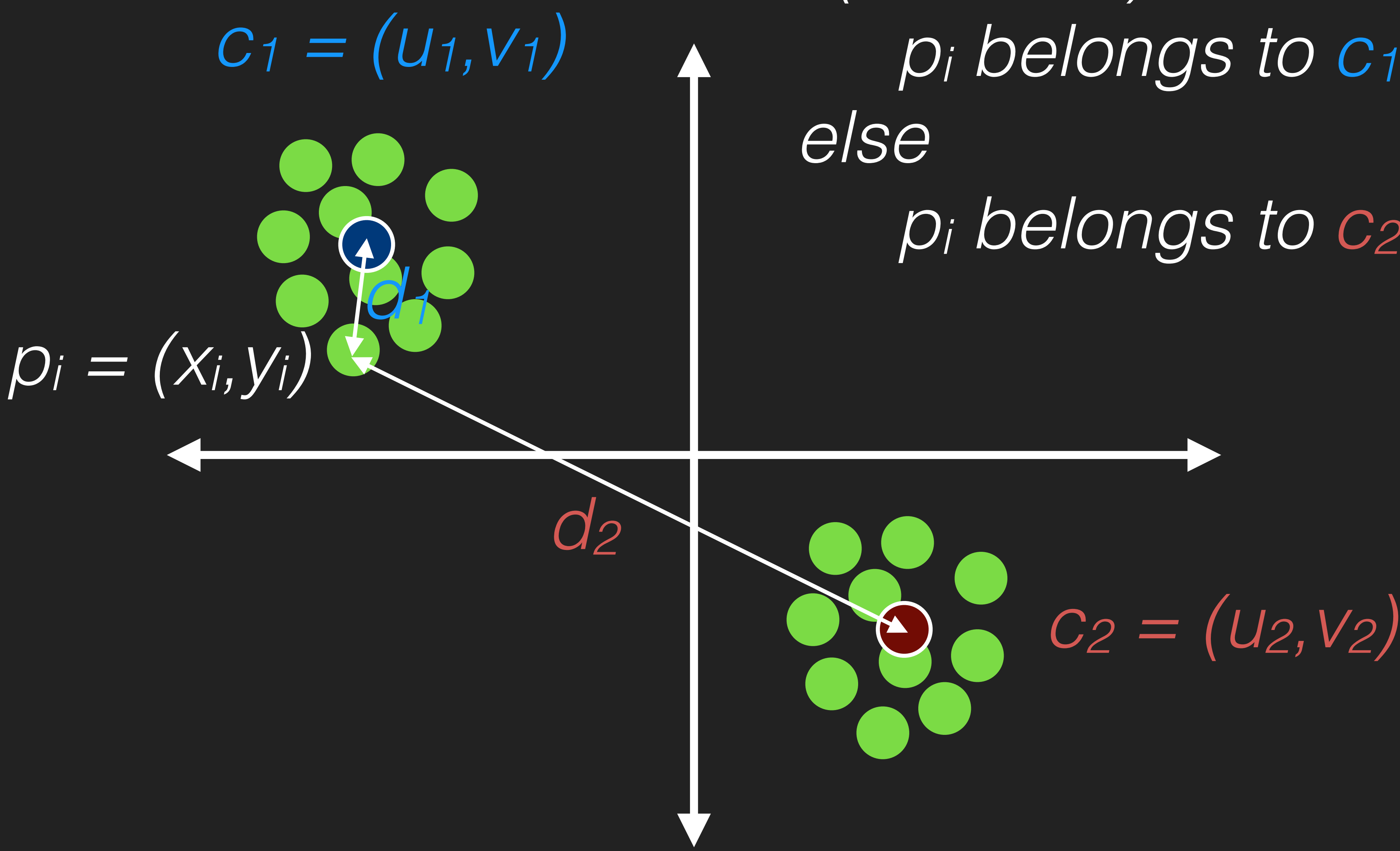


Clustering (K-means)

if we know c_i then assignment of each p_i to a cluster is easy

$$d_1 = sqrt((x_i-u_1)^2 + (y_i-v_1)^2)$$
$$d_2 = sqrt((x_i-u_2)^2 + (y_i-v_2)^2)$$

if($d_1 < d_2$)
    p_i *belongs to* c_1
else
    p_i *belongs to* c_2

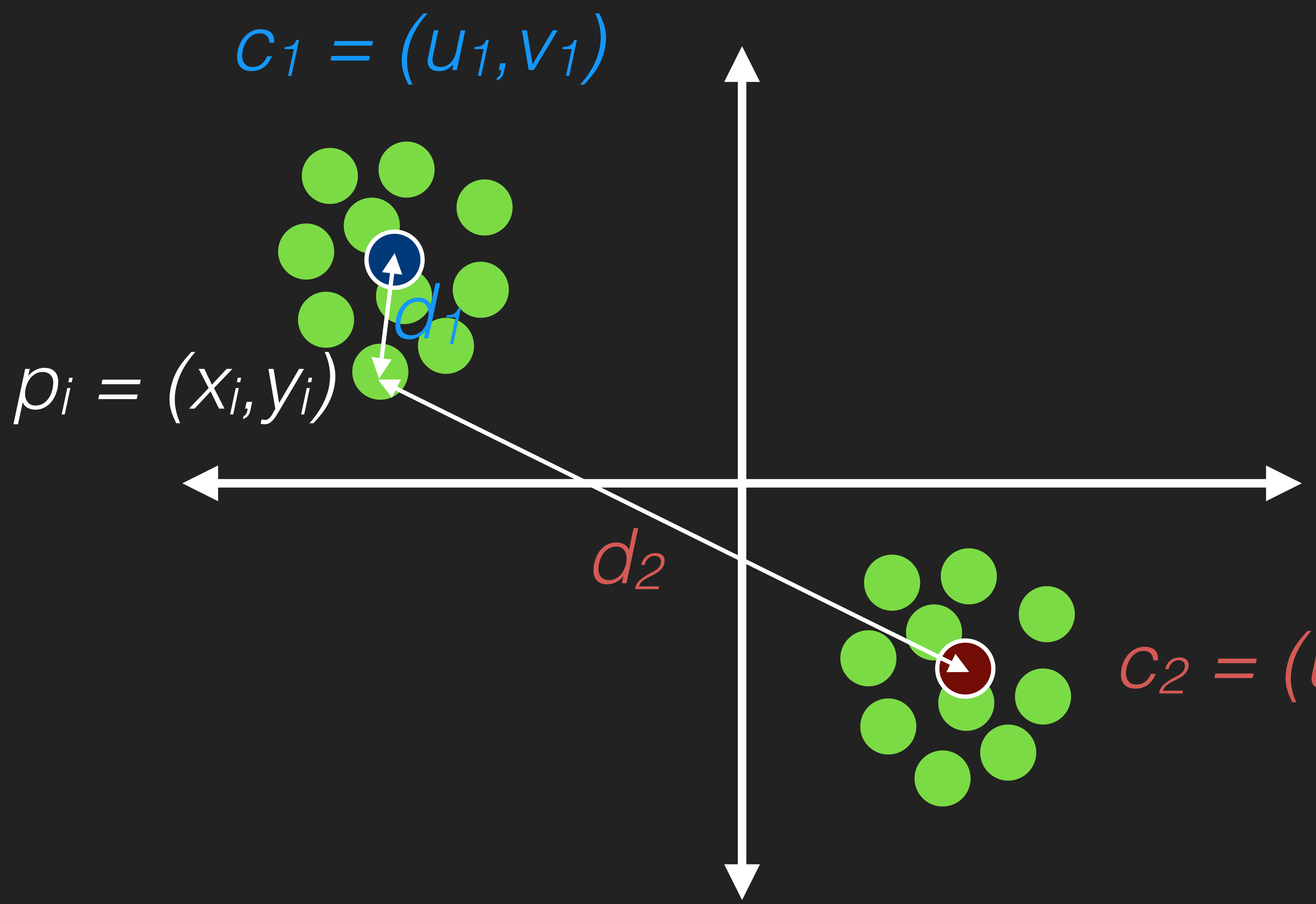


Clustering (K-means)

if we know c_i then assignment of each p_i to a cluster is easy

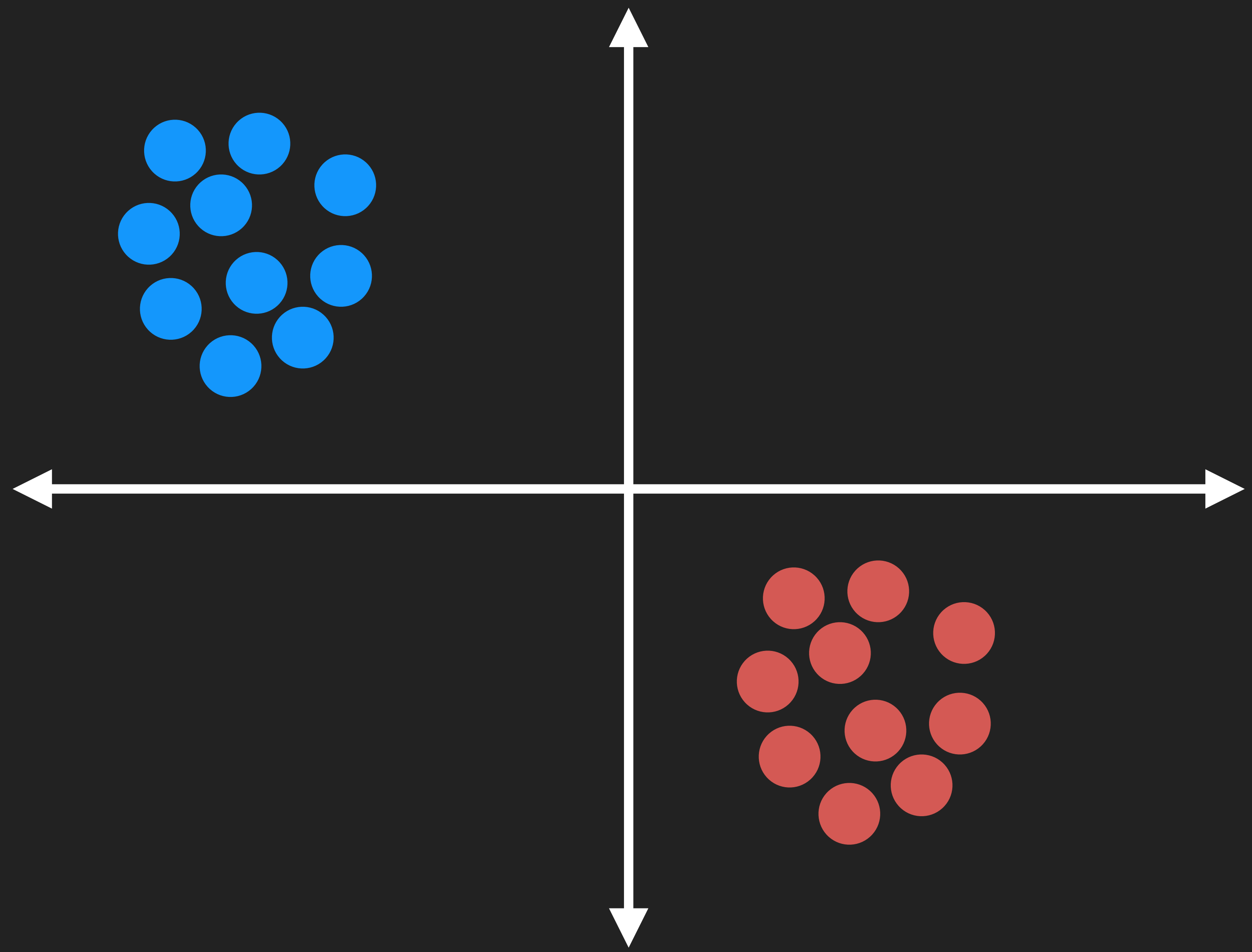
$d_1 = sqrt((x_i-u_1)^2 + (y_i-v_1)^2)$
$d_2 = sqrt((x_i-u_2)^2 + (y_i-v_2)^2)$

3-D, 4-D, …?

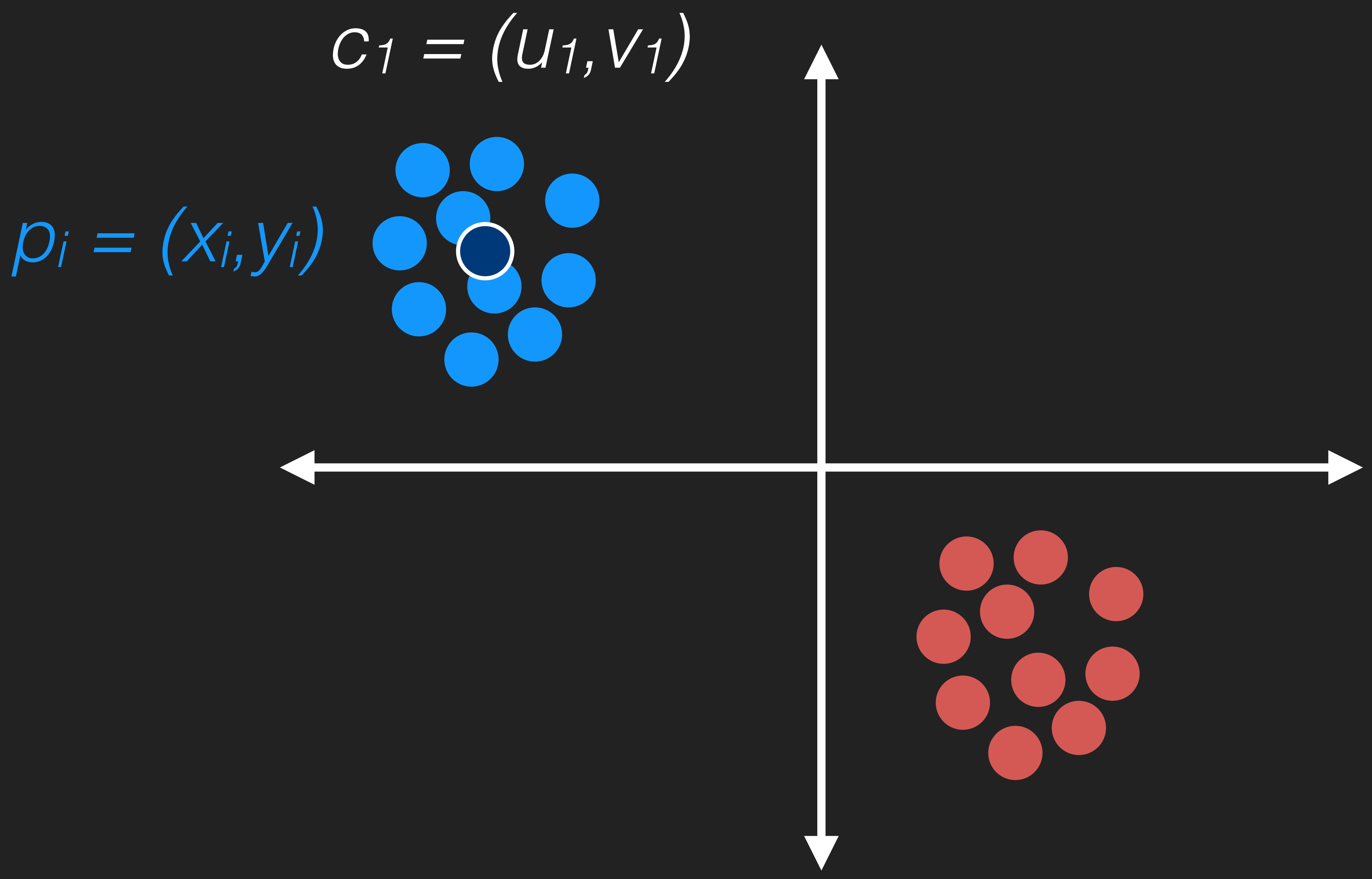


Clustering (K-means)

if we know assignment of each p_i then estimation of c_i is easy



if we know assignment of each p_i then estimation of c_i is easy

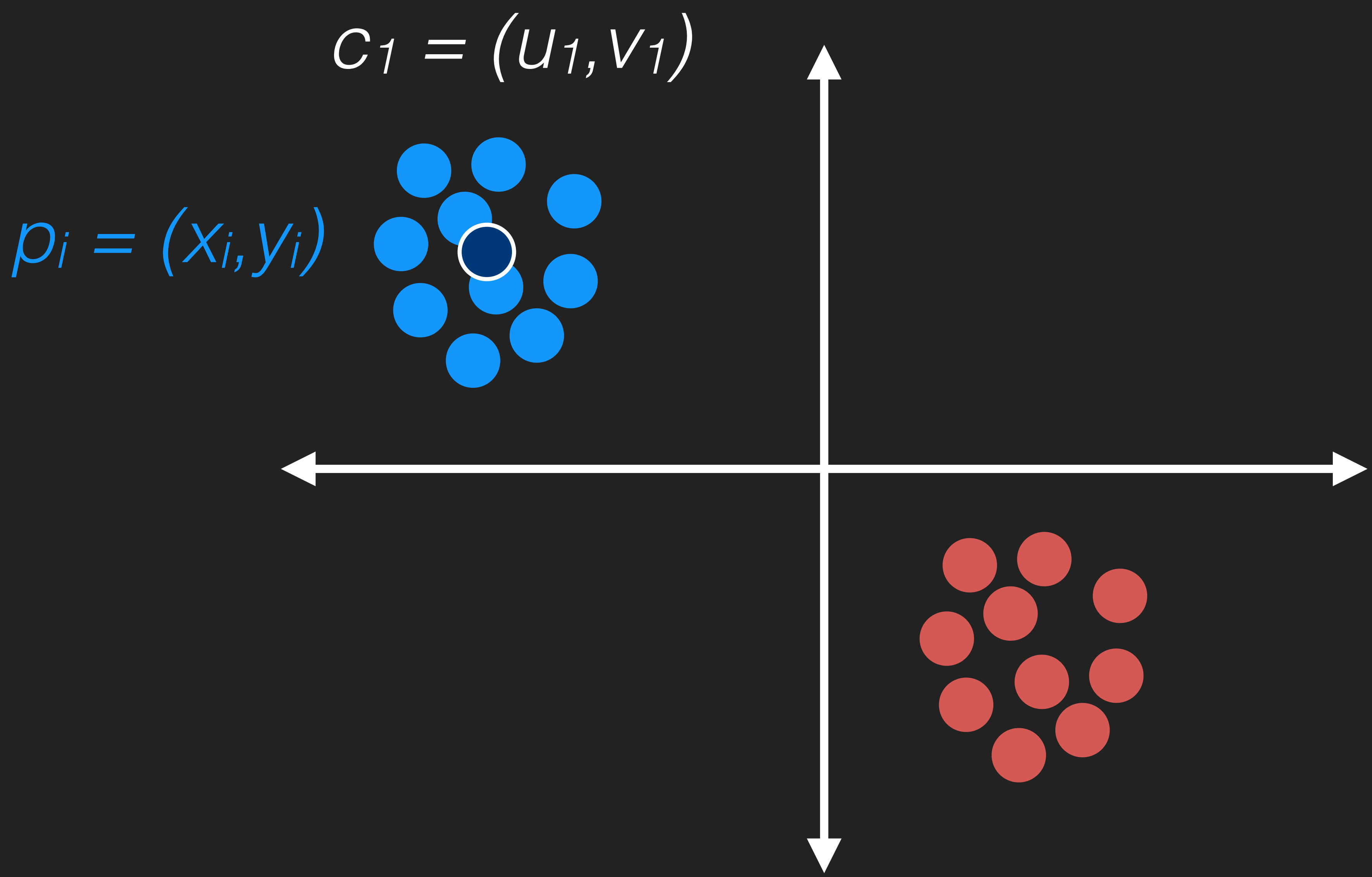


if we know assignment of each p_i then estimation of c_i is easy

center of mass:

$u_1 = 1/n\,(x_1 + x_2 + \dots + x_n)$

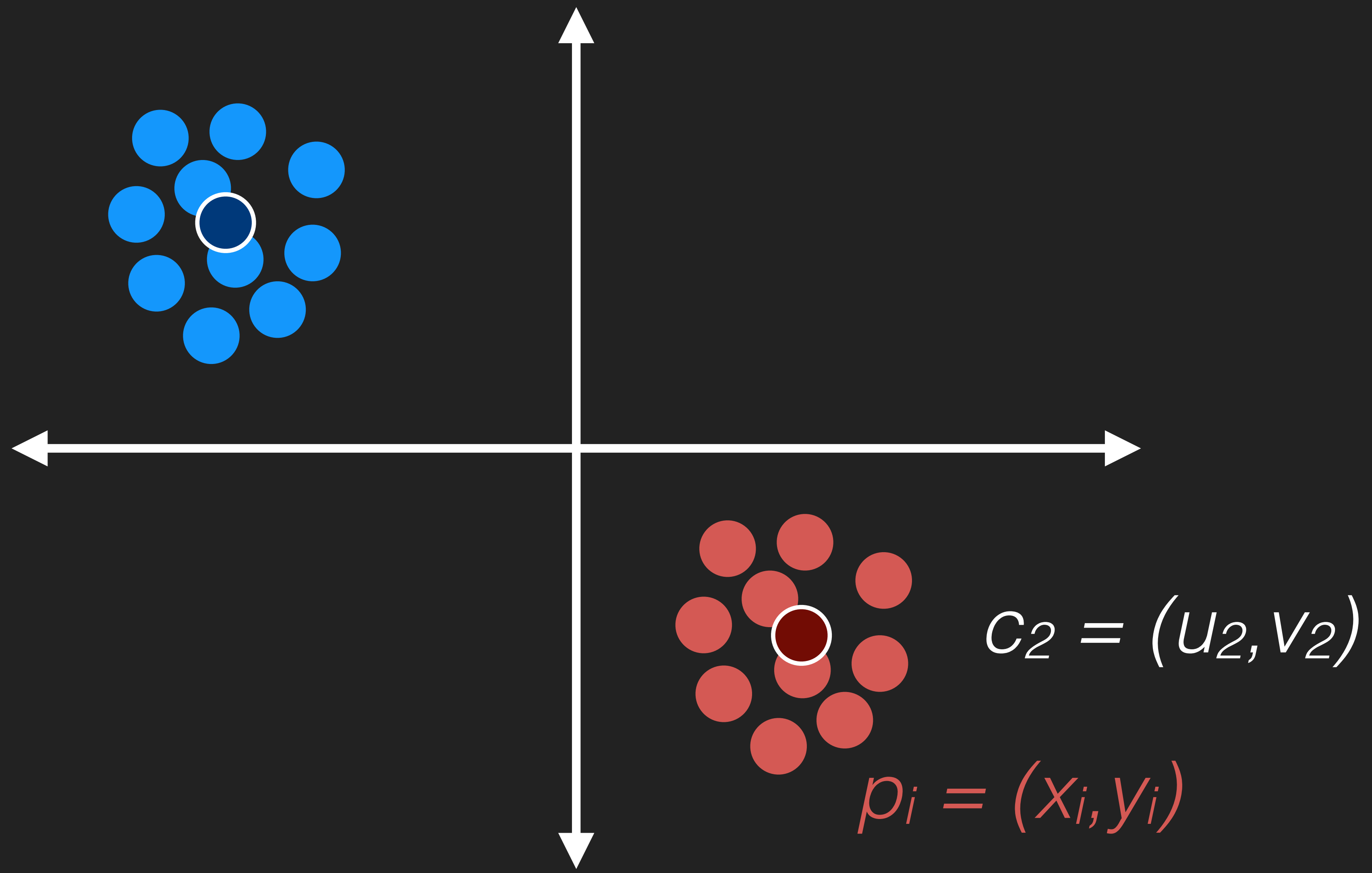
$v_1 = 1/n\,(y_1 + y_2 + \dots + y_n)$



if we know assignment of each p_i then estimation of c_i is easy

center of mass:

$$u_2 = 1/m\,(x_1 + x_2 + \dots + x_m)$$
$$v_2 = 1/m\,(y_1 + y_2 + \dots + y_m)$$



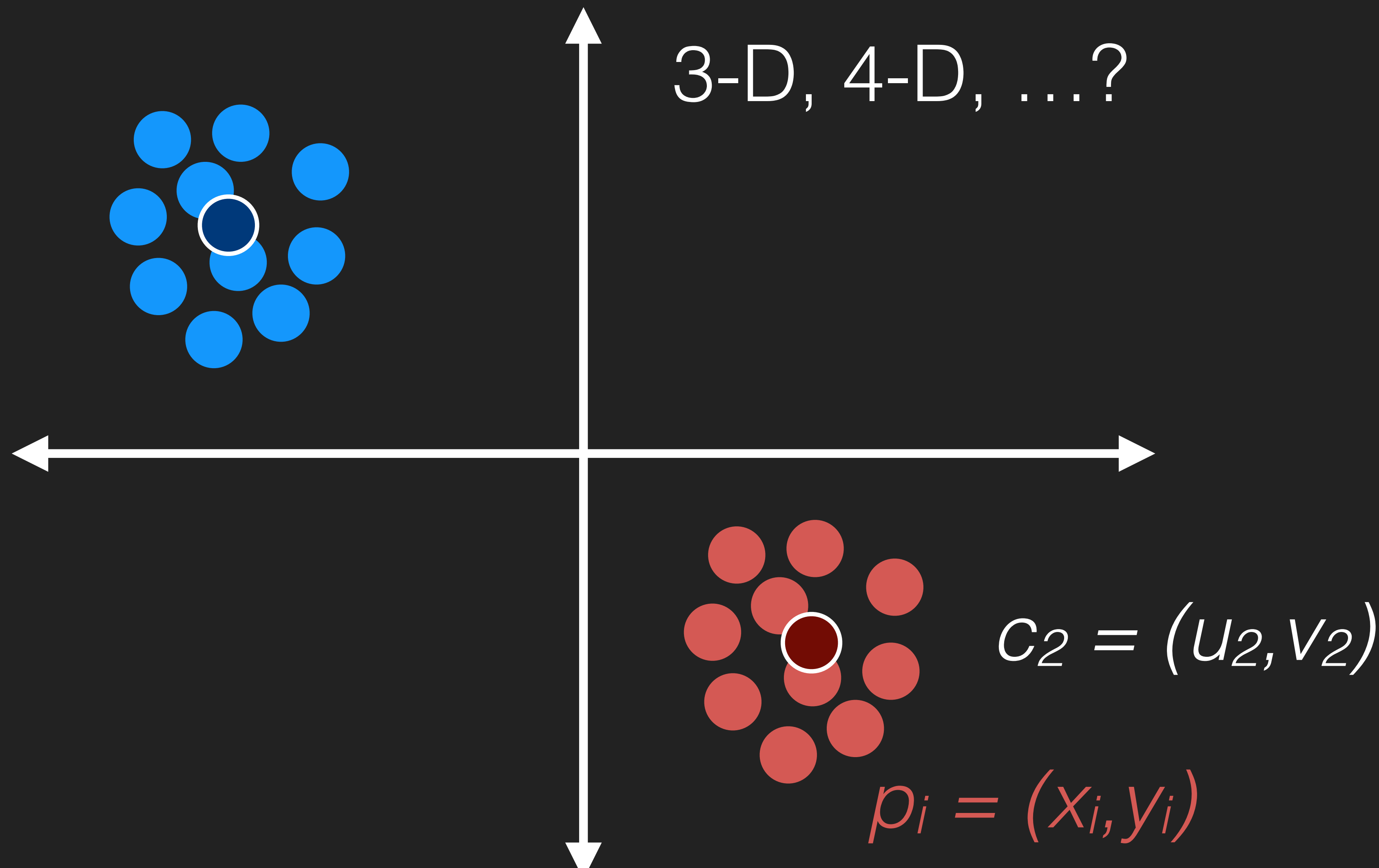
Clustering (K-means)

if we know assignment of each p_i then estimation of c_i is easy

center of mass:

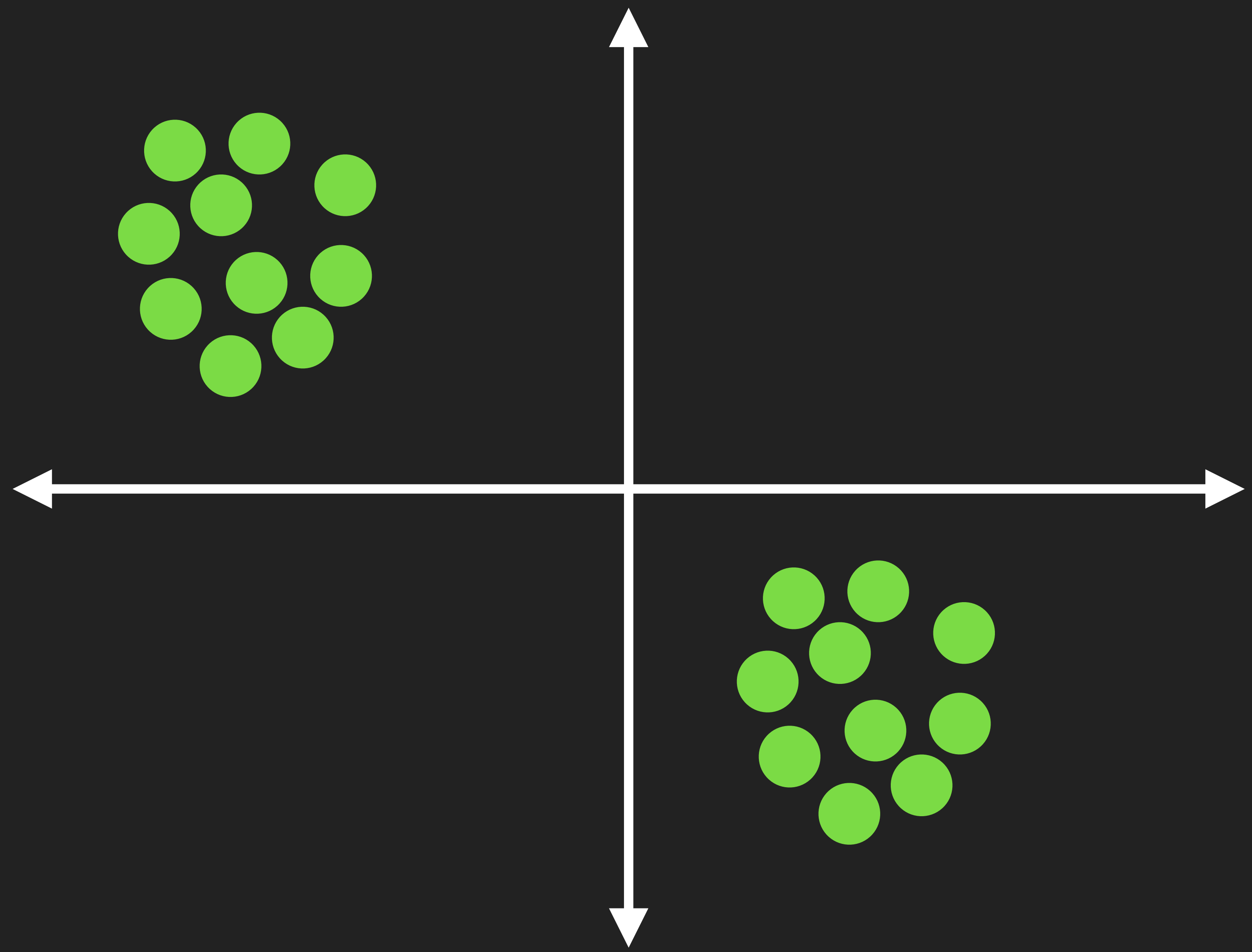
$u_2 = 1/m\ (x_1 + x_2 + \ldots. + x_m)$

$v_2 = 1/m\ (y_1 + y_2 + \ldots. + y_m)$

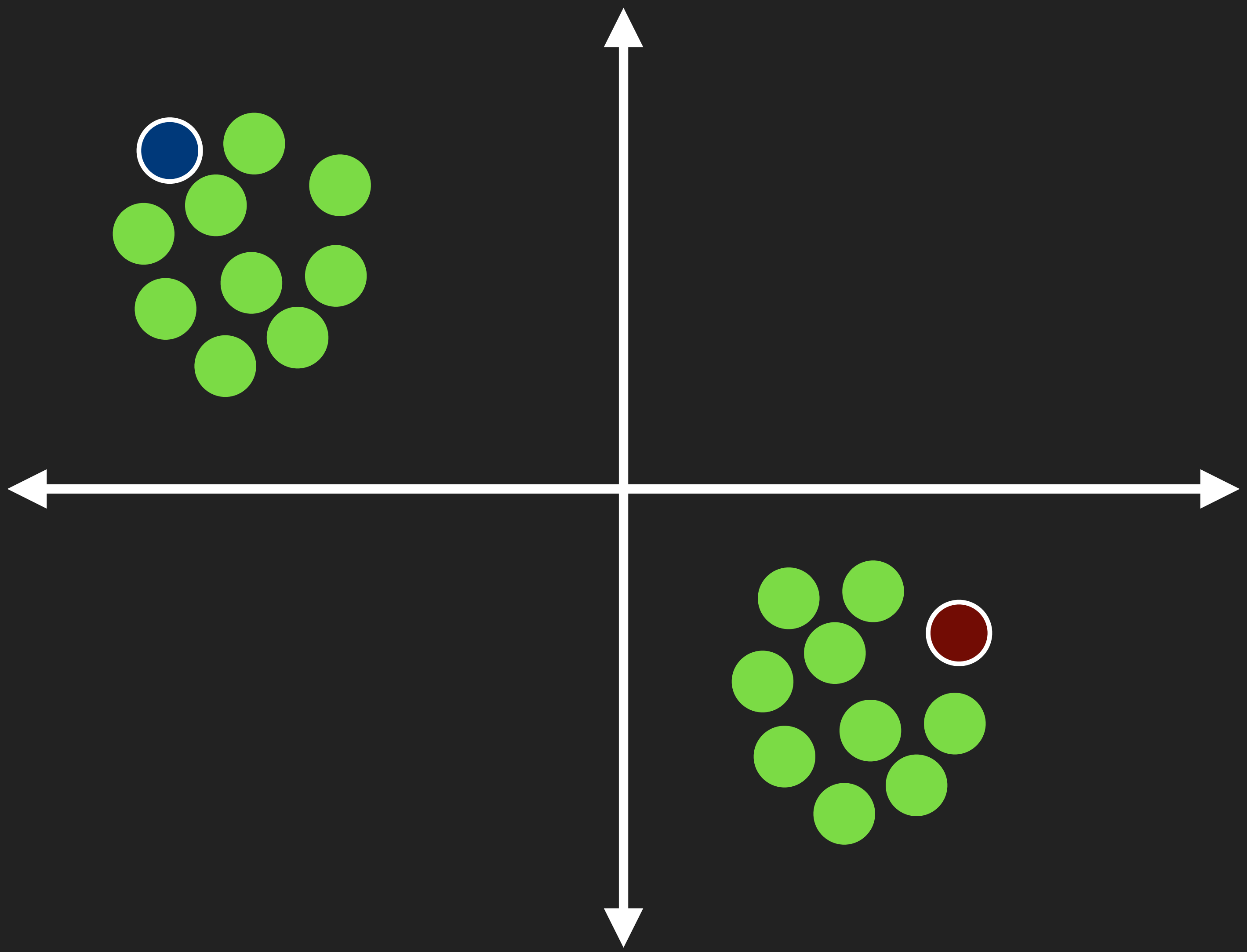


Clustering (K-means)

but we don’t know cluster center or cluster assignment - we have a chicken and egg problem.

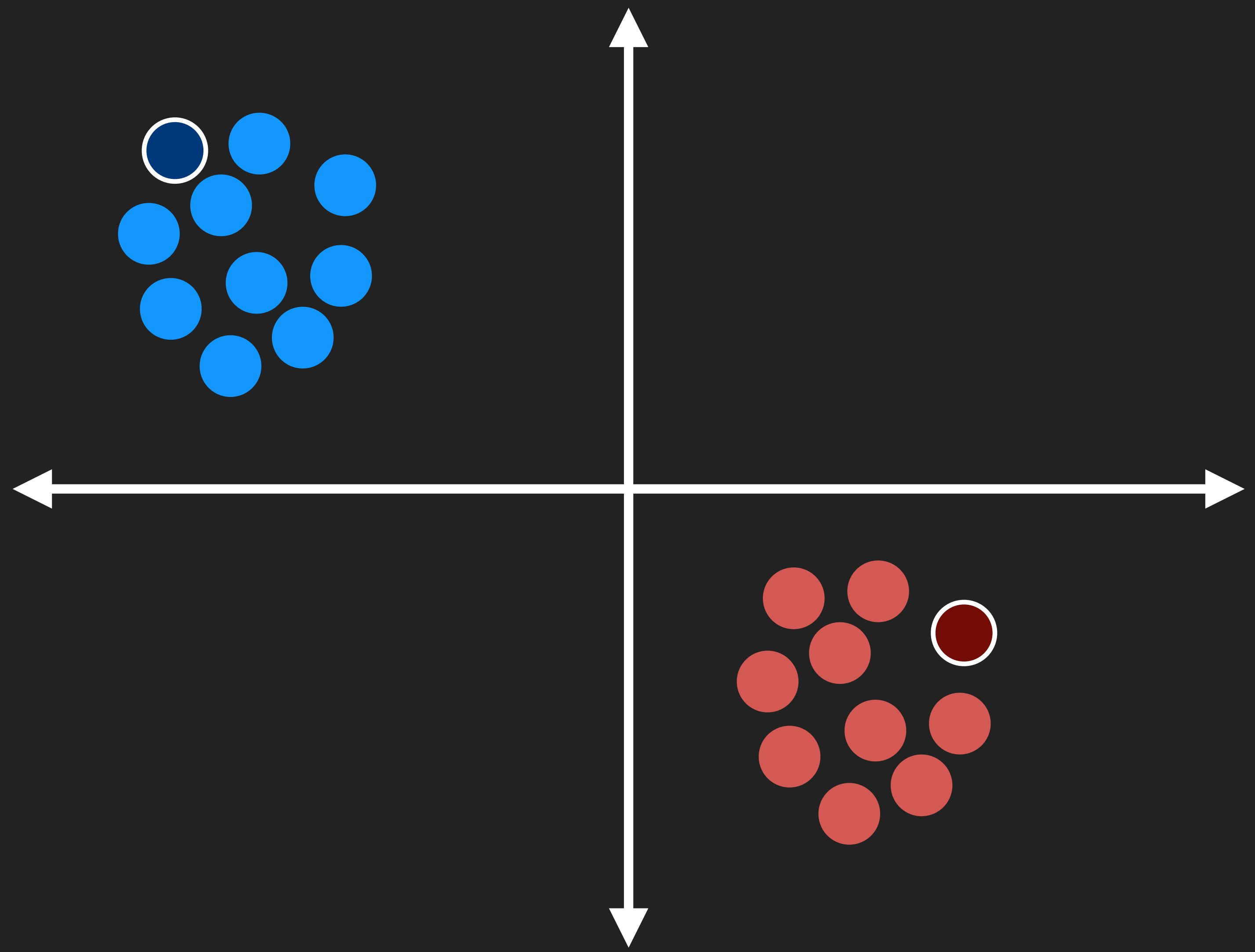


1. initialize clusters (*c_1 and c_2*) randomly

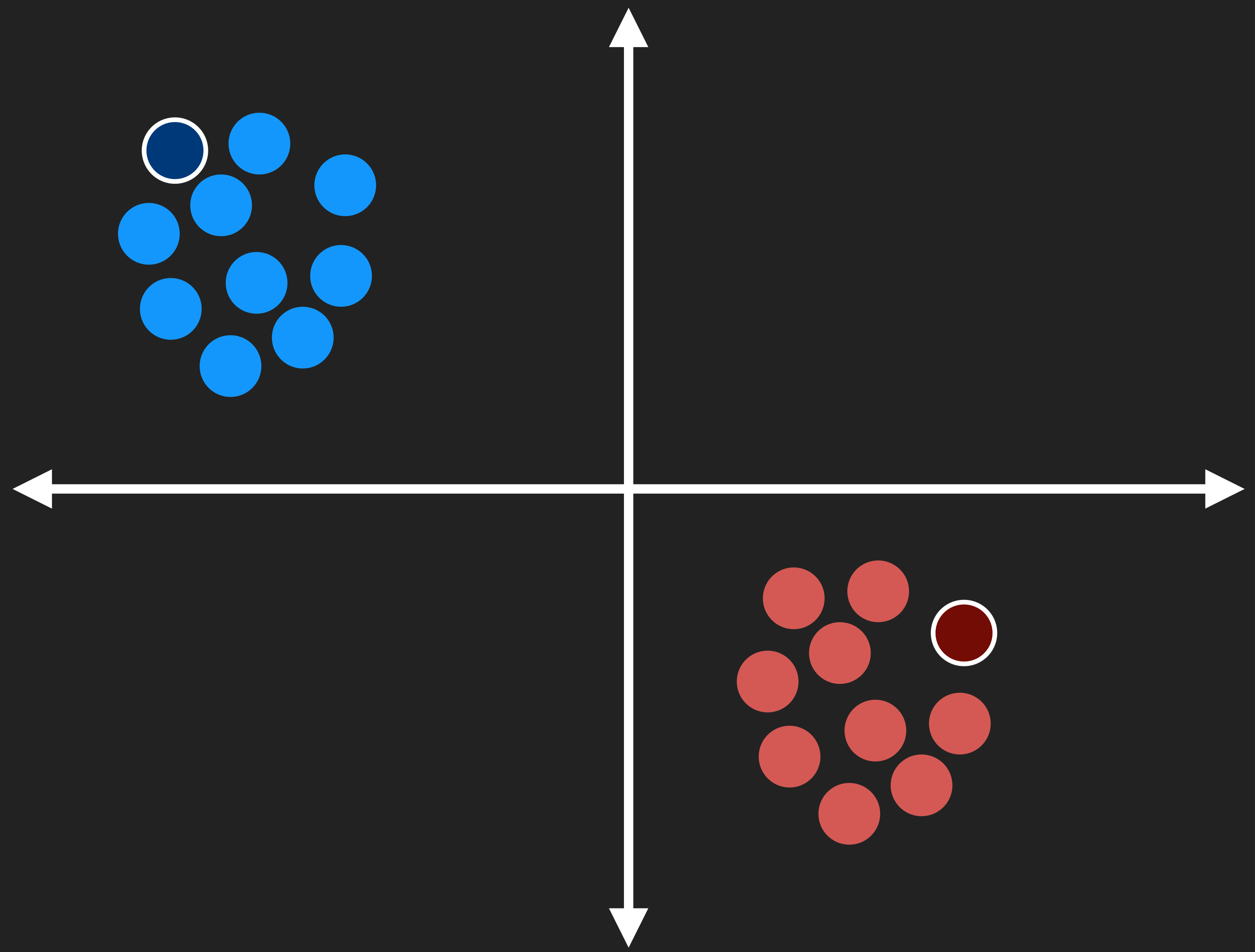


Clustering (K-means)

1. initialize clusters (c_1 *and* c_2) randomly
2. assign each point (p_i) to the closest cluster



1. initialize clusters (*c_1 and c_2*) randomly
2. assign each point (p_i) to the closest cluster
3. re-estimate cluster centers
4. repeat until assignment doesn't change



[demo]

Clustering (K-means): assumptions