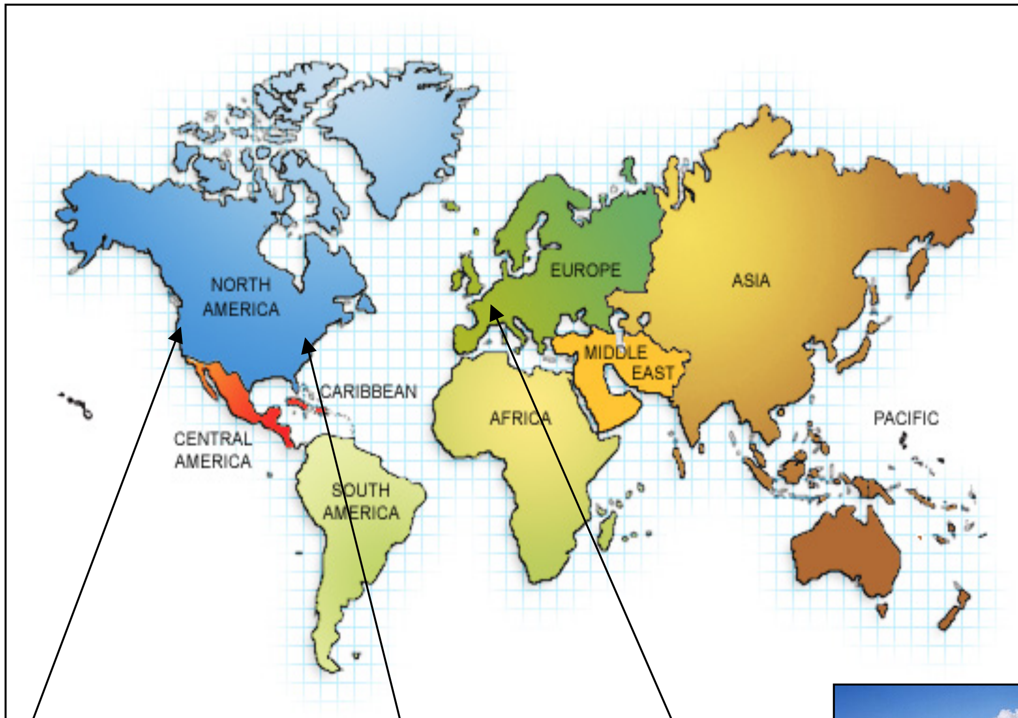
UCB - CS189
Introduction to Machine Learning
Fall 2015
Lecture 2: Linear Classifiers

Isabelle Guyon

ChaLearn

Isabelle Guyon

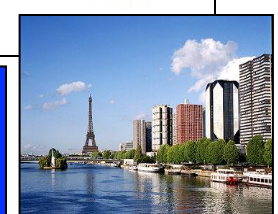


CHALEARN

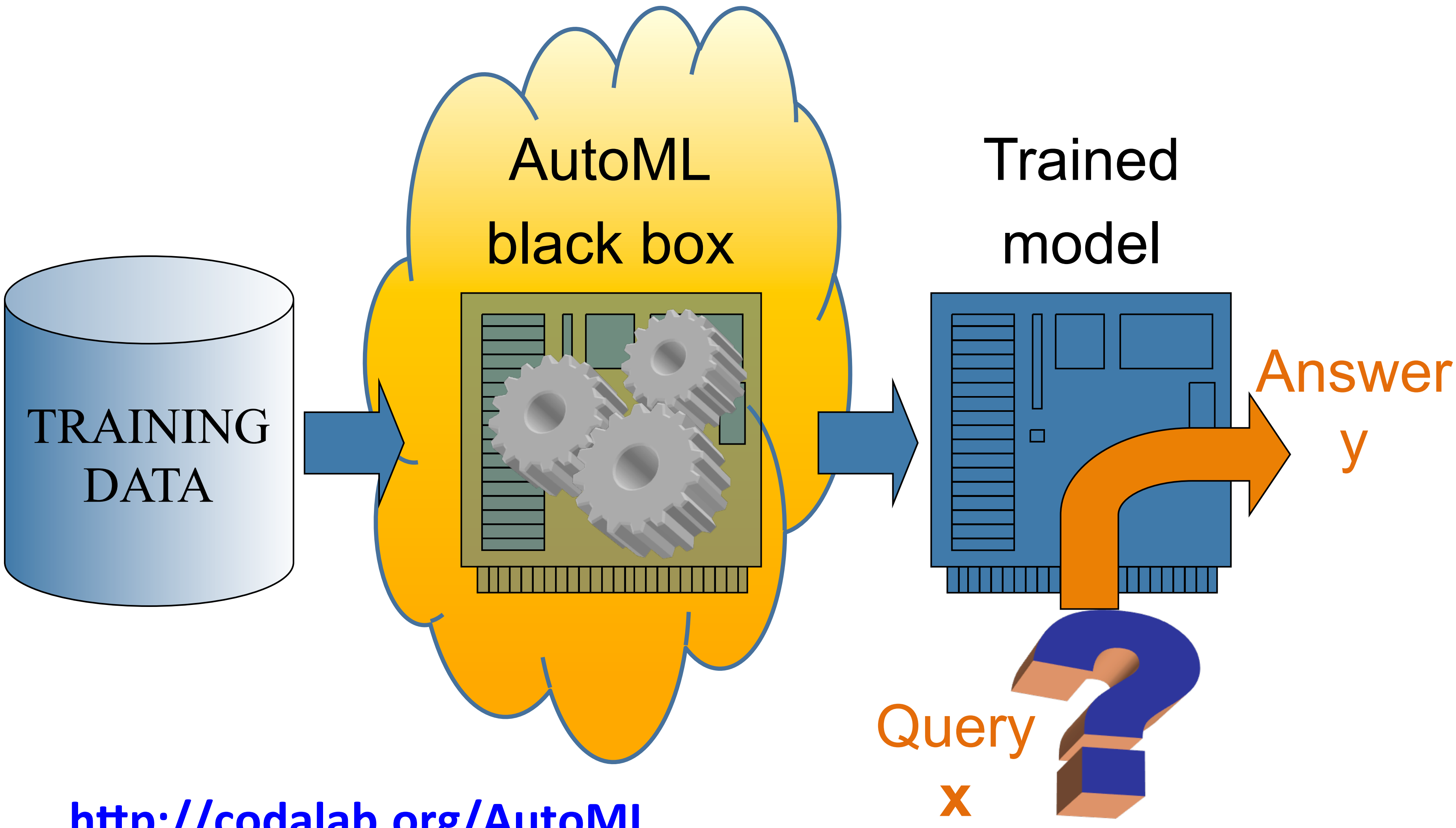
CLOPINET

AT&T Bell Laboratories

ESPCI ParisTech

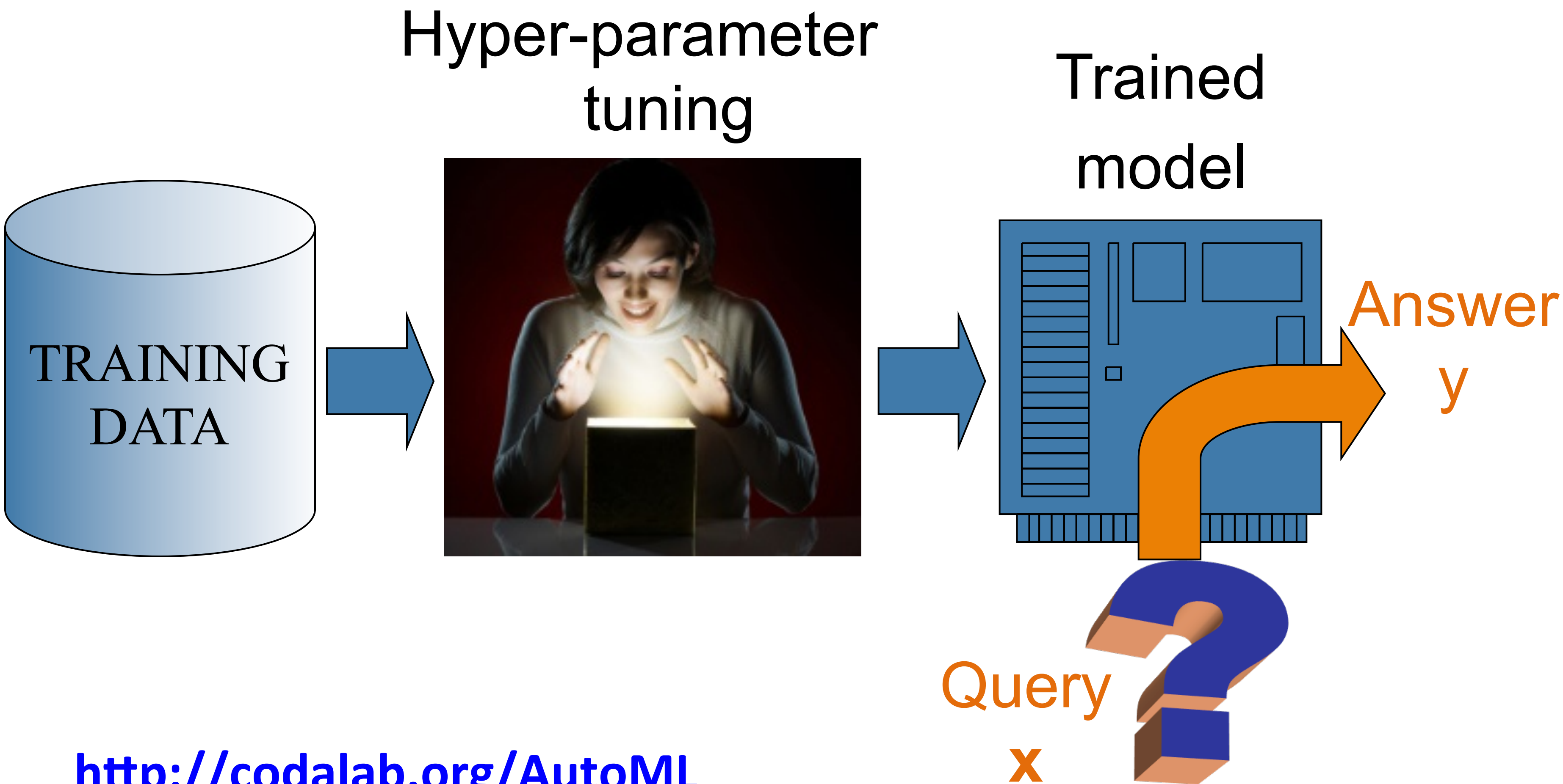


The DREAM



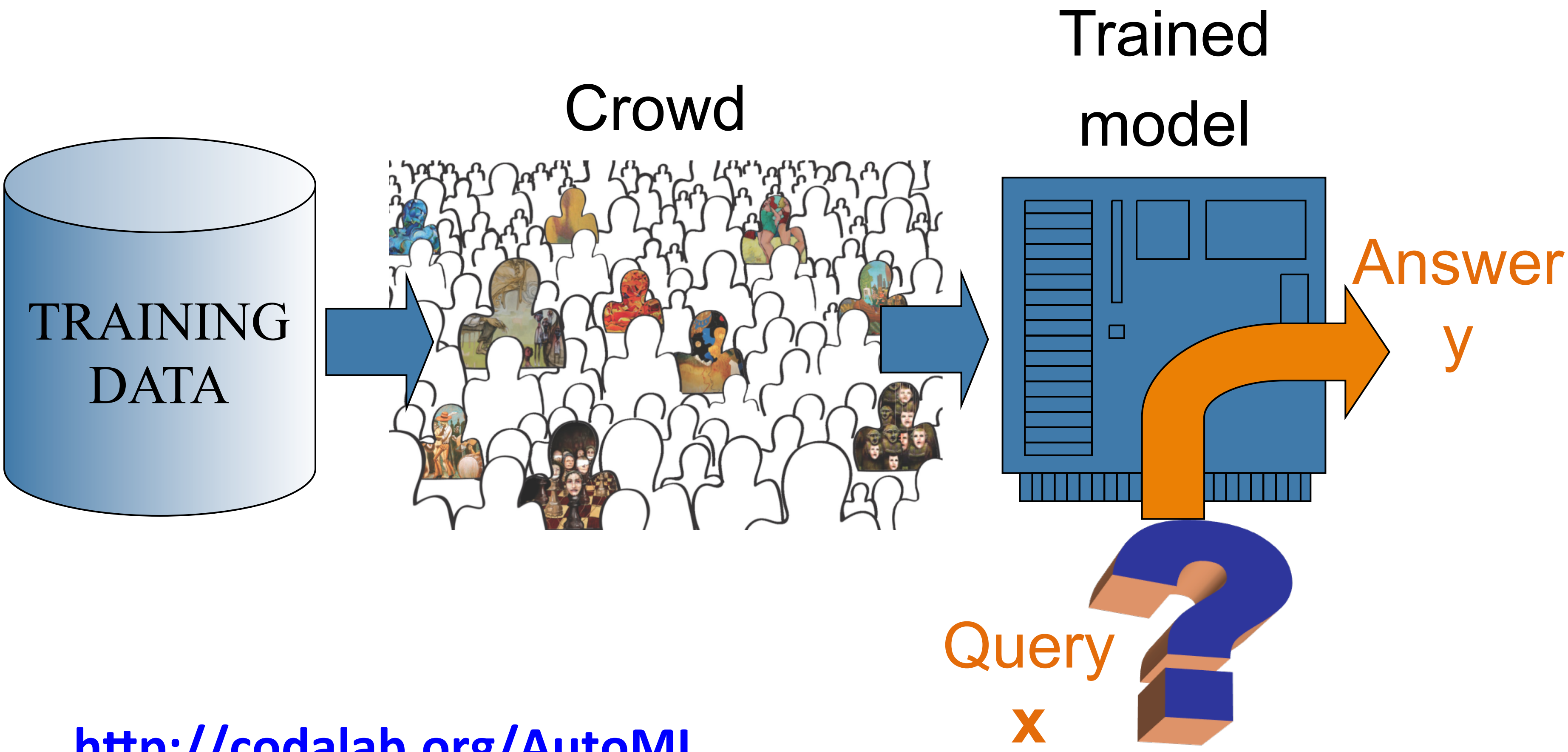
http://codalab.org/AutoML

The REALITY



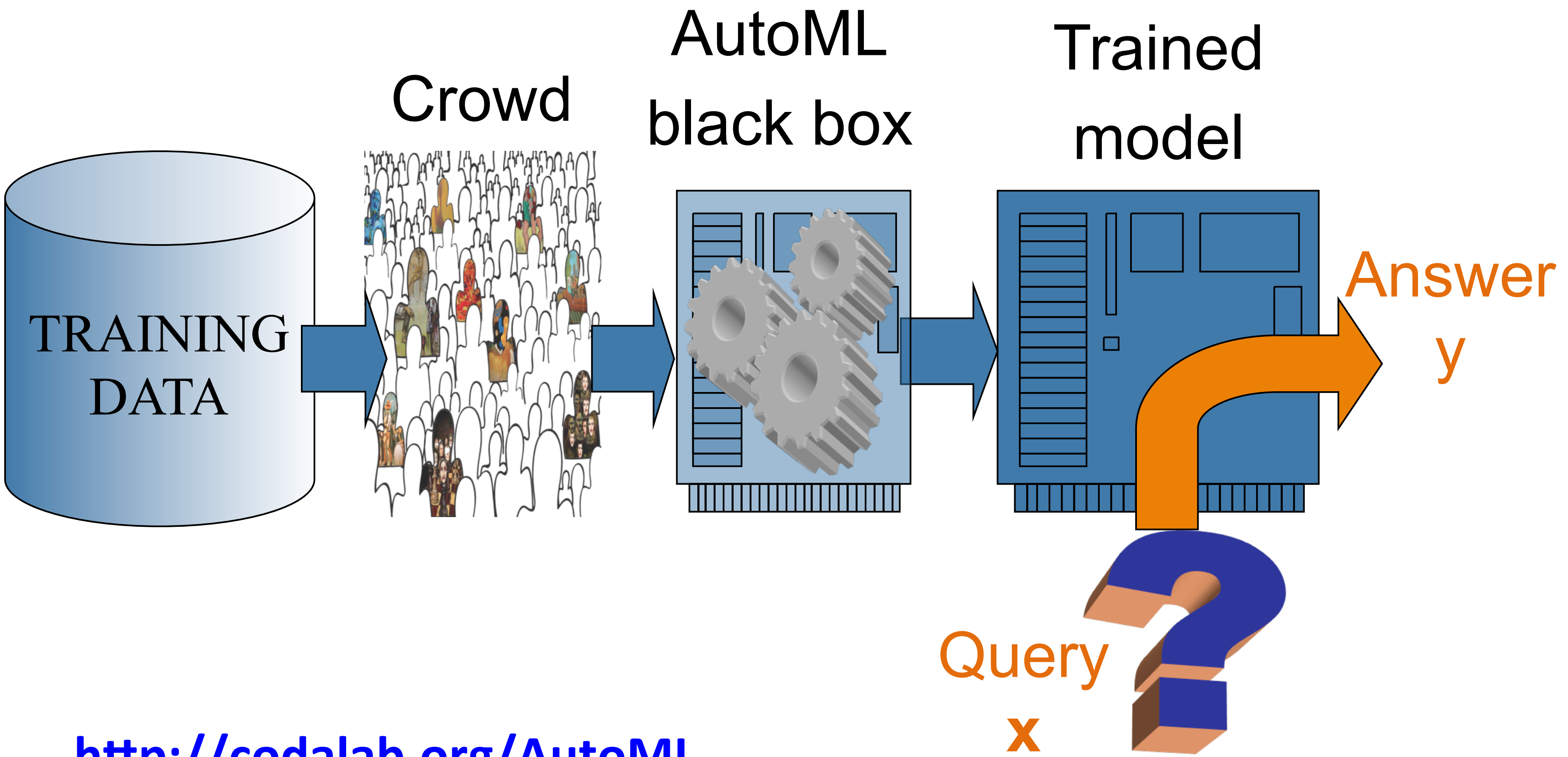
http://codalab.org/AutoML

ChaLearn ML challenges



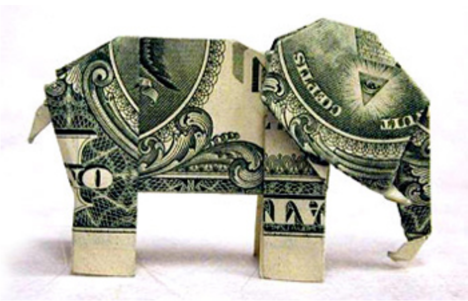
http://codalab.org/AutoML

AutoML challenge



http://codalab.org/AutoML

Why would the crowd do that?



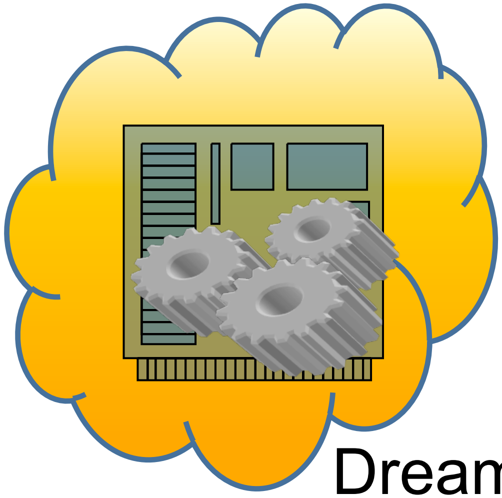
$30000

Fame

Learning

Fun

Workshop

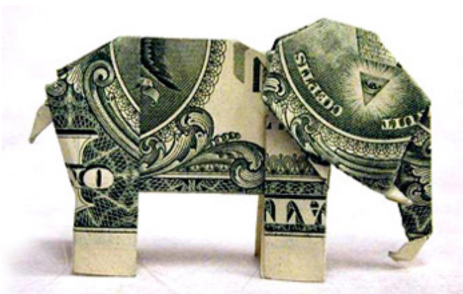


Dream

http://codalab.org/AutoML

Come to my office hours…

Wed 2:30-4:30 Soda 329



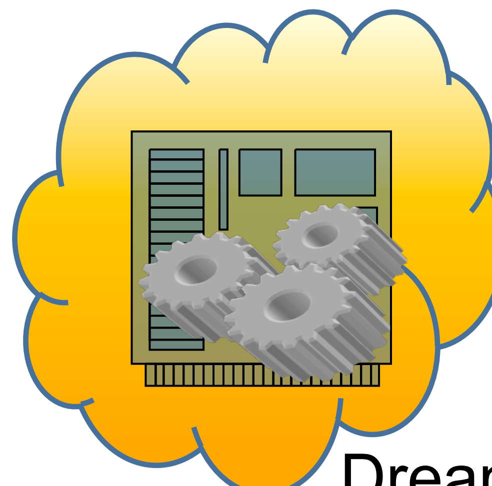
$30000

Fame

Learning

Fun

Workshop



Dream

http://codalab.org/AutoML

Come to my office hours…

Wed 2:30-4:30 Soda 329

Last time

Parametric vs. Non-Parametric

- Linear Classifier (Parametric)
  - small number of parameters, e.g. **w** and b
  - very fast at test time: **w.x** + b
  - after training is done, can throw away the data!
- Nearest-Neighbor (Non-Parametric)
  - Number of parameters grows with size of dataset
  - Very slow at test time -- O(N)
  - No training
- Which one works better?

Come to my office hours…
Wed 2:30-4:30 Soda 329

Today

Parametric ≡ Non-Parametric

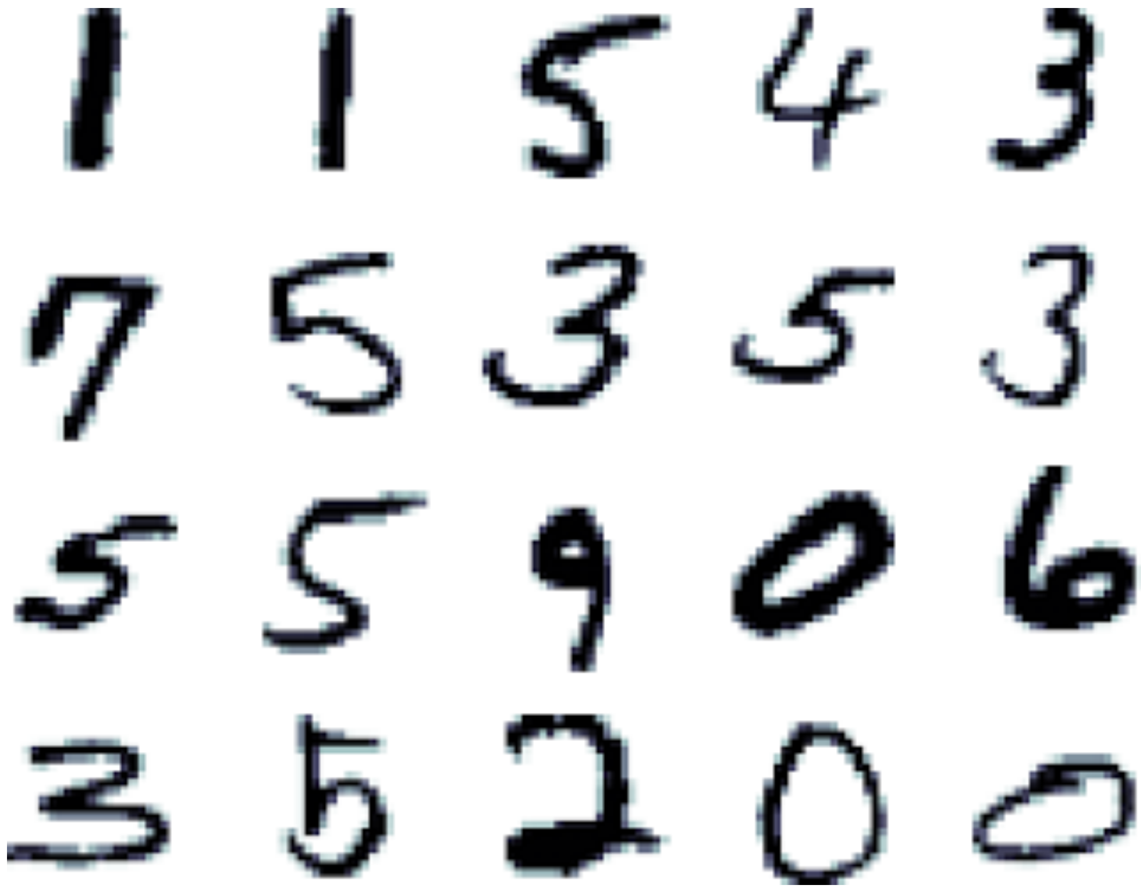
DUAL representations
via the
“kernel trick”

Math prerequisites

- Vector
- Dot product (scalar product)
- Euclidean norm
- Normalizing a vector
- Cosine between two vectors
- Standardizing a vector
- Correlation between two vectors
- Linearity
- Hyperplane

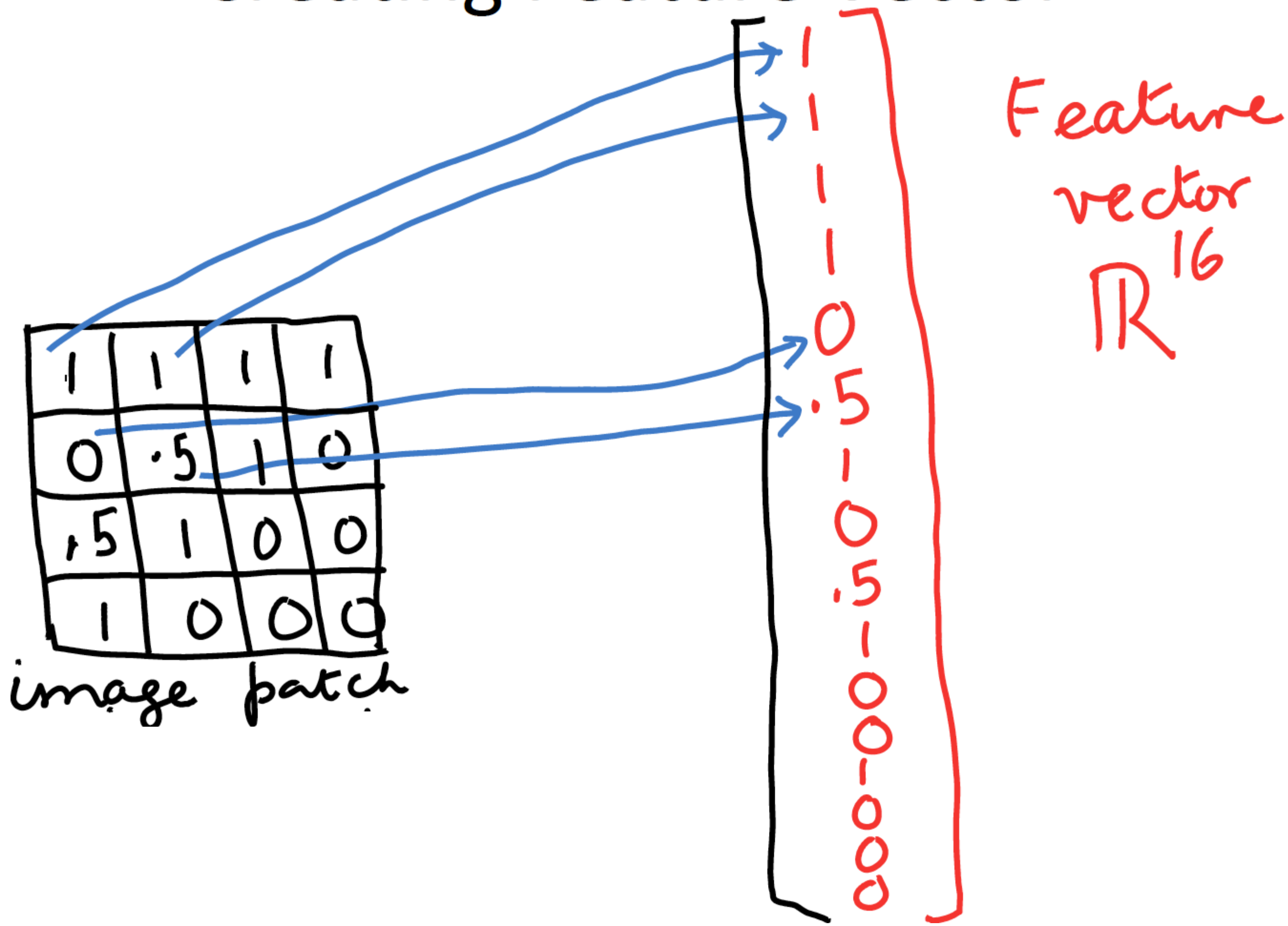
MNIST data

Classification Problems (Homework)



Reminder:

Creating Feature Vector



Reminder:

Creating Feature Vector

1	1	1	1
0	.5	1	0
.5	1	0	0
1	0	0	0

image patch

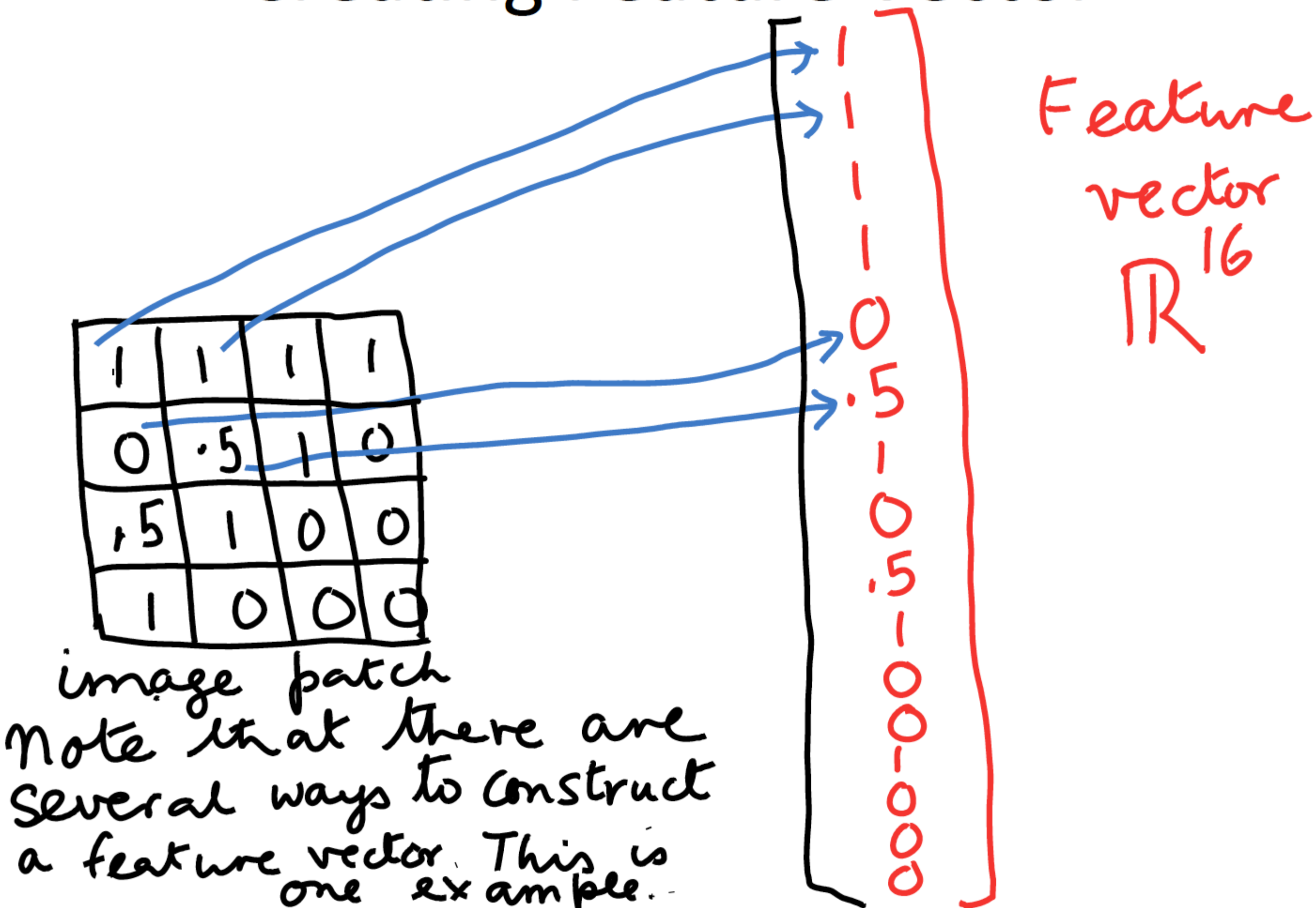
$$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ .5 \\ 1 \\ 0 \\ 0 \\ .5 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Feature vector $\mathbb{R}^{16}$

Note that there are several ways to construct a feature vector. This is one example.

Reminder:

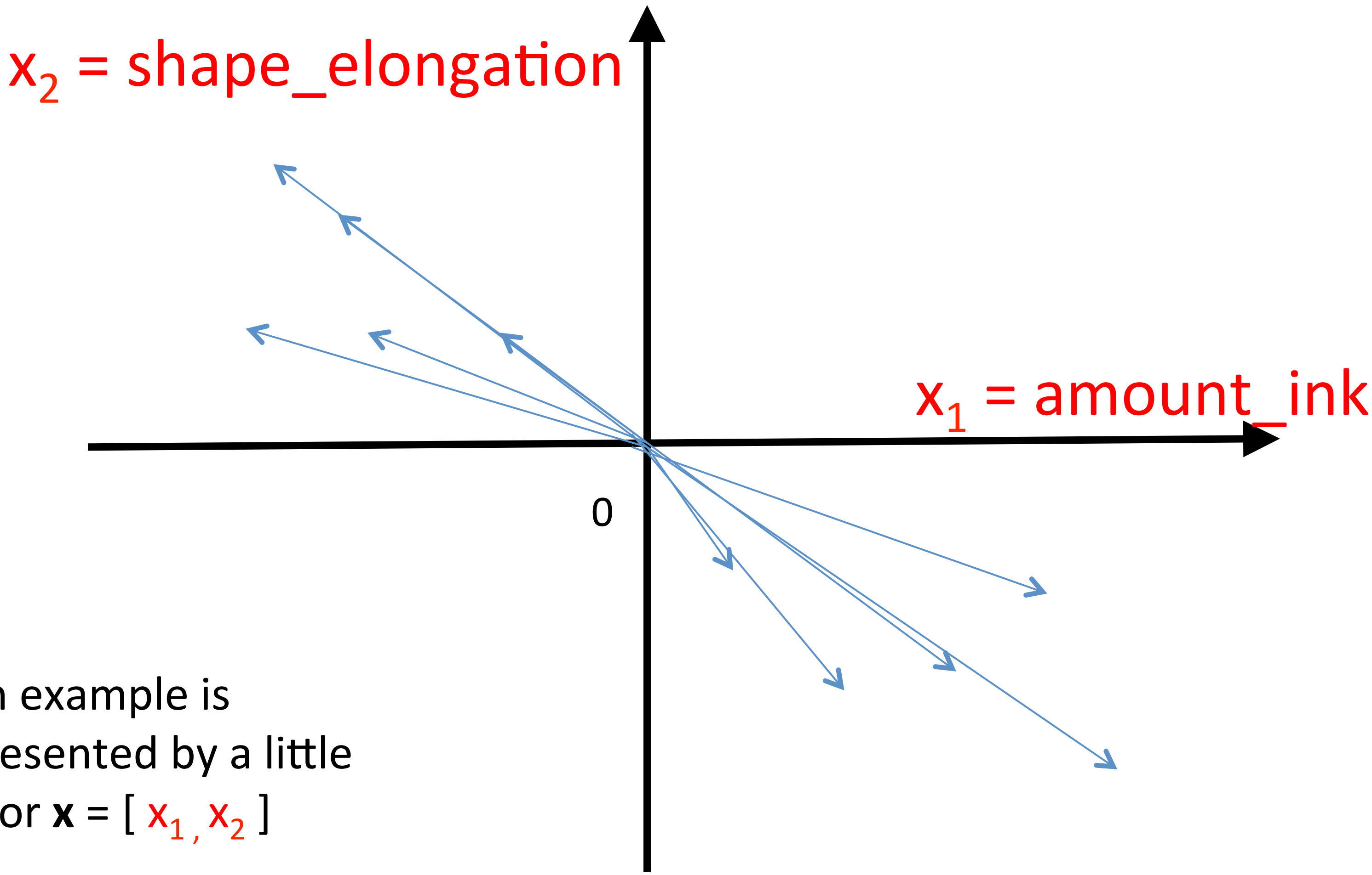
Creating Feature Vector



Another example, extract features: **x** = [amount_ink, shape_elongation]

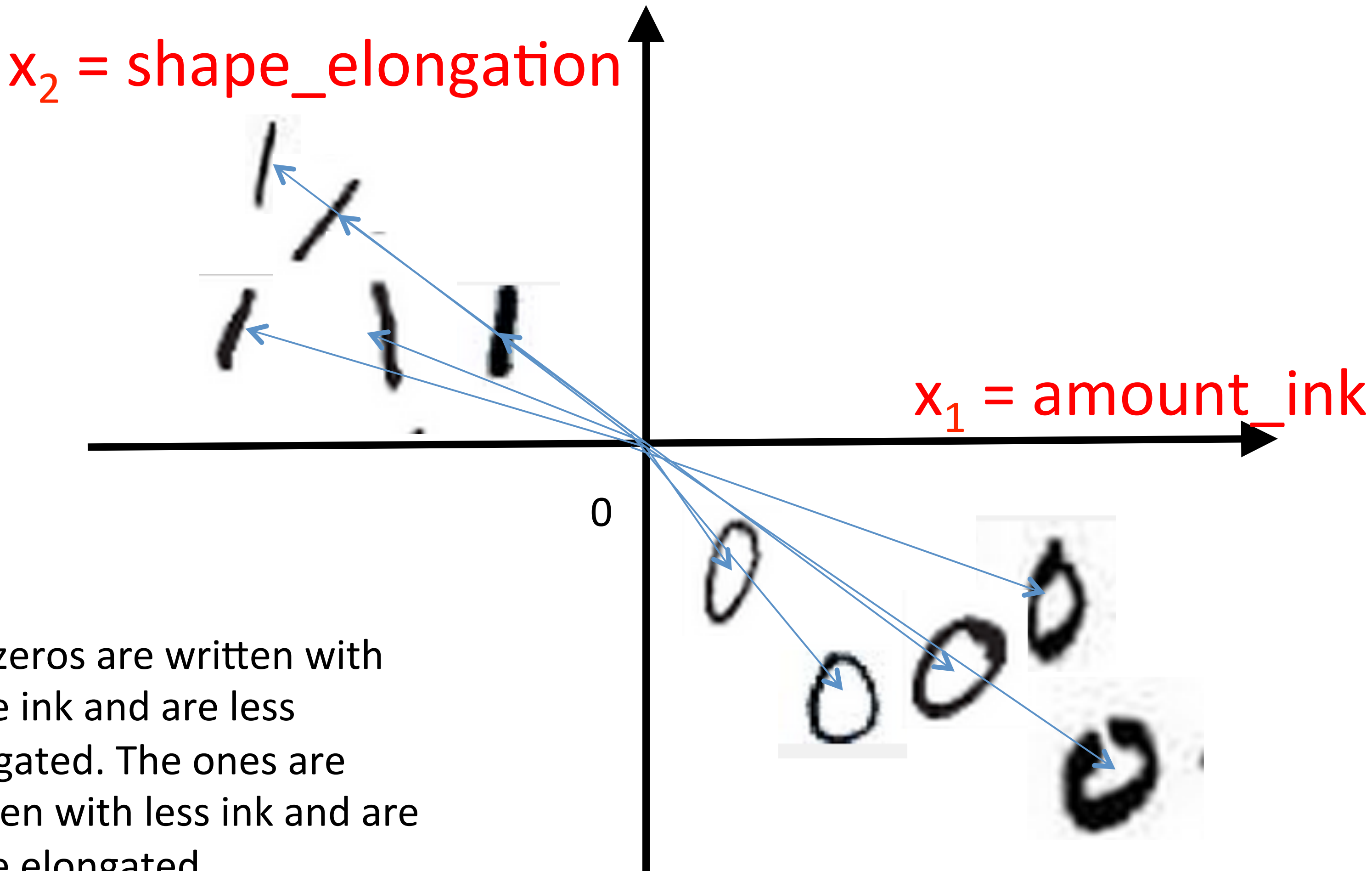
Separate “0” and “1”

in 2 dimensions



Each example is represented by a little vector **x** = [x_1 , x_2]

Separate “0” and “1”
in 2 dimensions



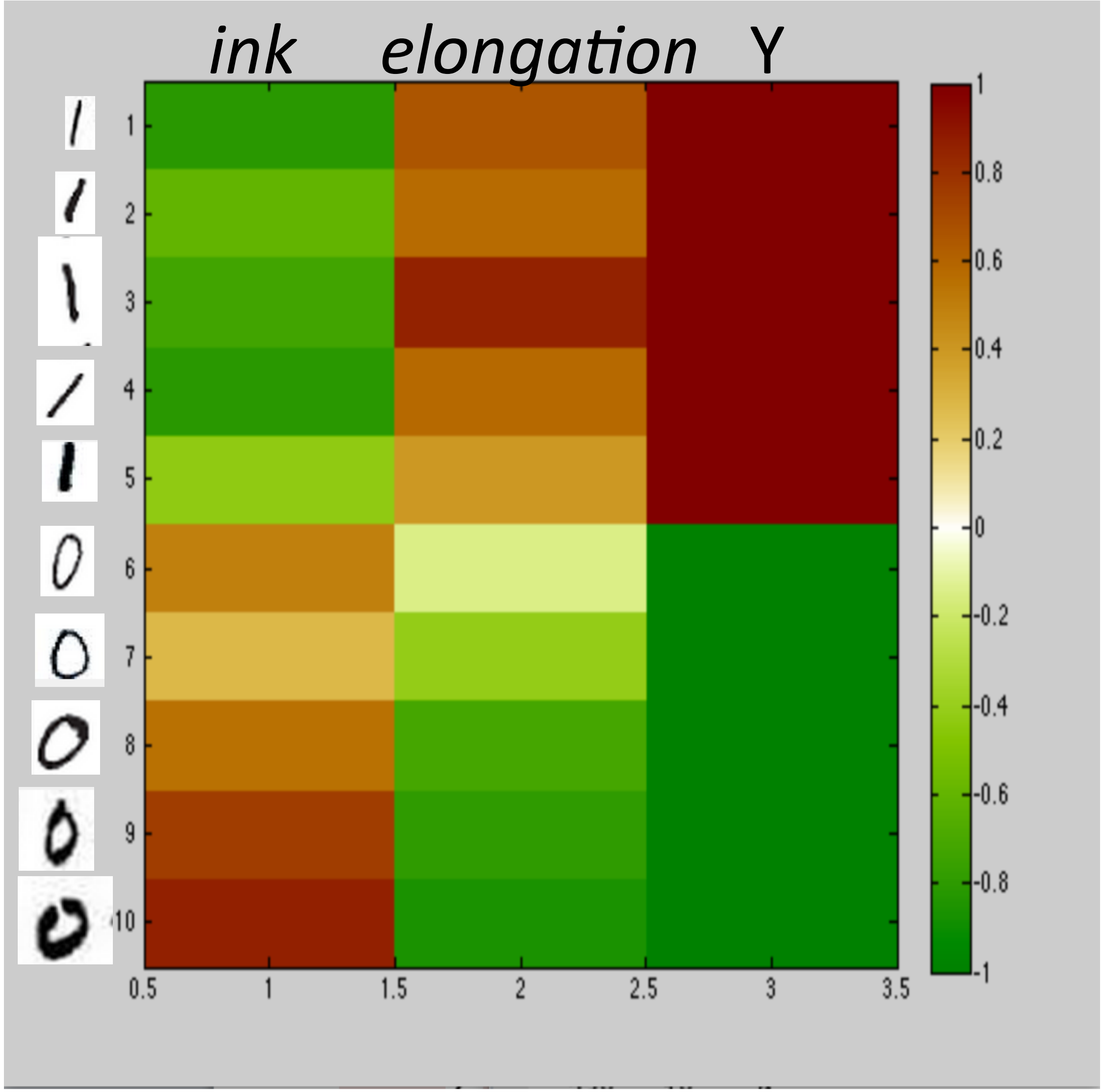
The zeros are written with more ink and are less elongated. The ones are written with less ink and are more elongated.

Separate “0” and “1”

matrix representation

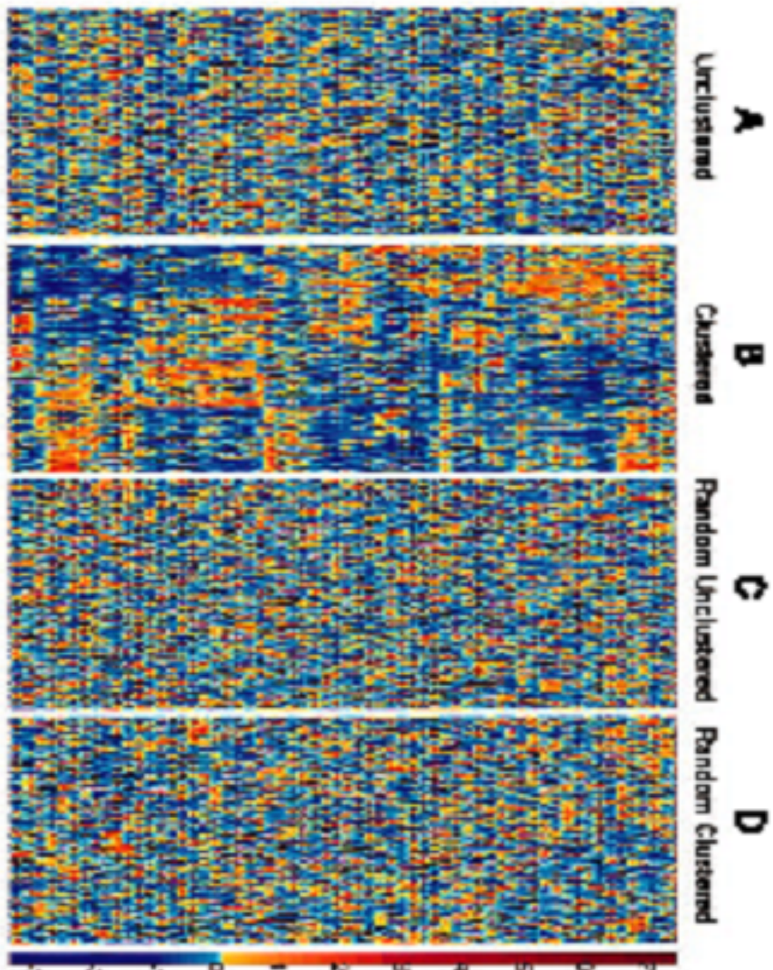
	ink	elongation	
X = [	-0.8147	0.6576	**y**= [1
	-0.6058	0.5706	1
	-0.7270	0.8572	1
	-0.8134	0.5854	1
	-0.4324	0.4003	1
	0.4975	-0.1419	-1
	0.2785	-0.4218	-1
	0.5469	-0.7157	-1
	0.7575	-0.7922	-1
	0.8649	-0.8595]	-1]

Heat map



ink
elongation
Y
1
2
3
4
5
6
7
8
9
10
0.5
1
1.5
2
2.5
3
3.5
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1

Learning problem



Colon cancer, Alon et al 1999

Data matrix: X

N lines = patterns (data points, examples): samples, patients, documents, images, …

d columns = features: (attributes, input variables): genes, proteins, words, pixels, …

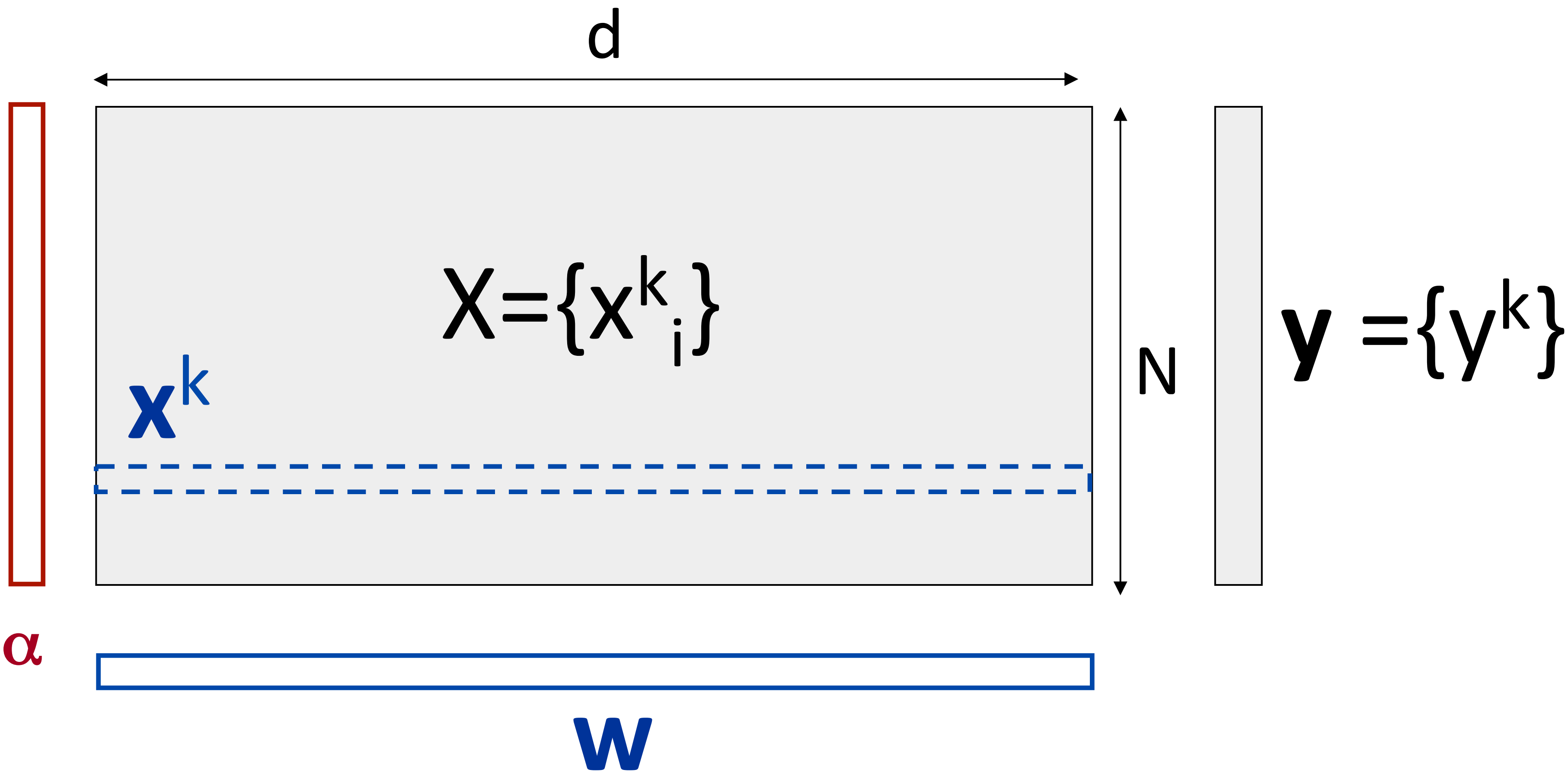
Unsupervised learning

Is there structure in data?

Supervised learning

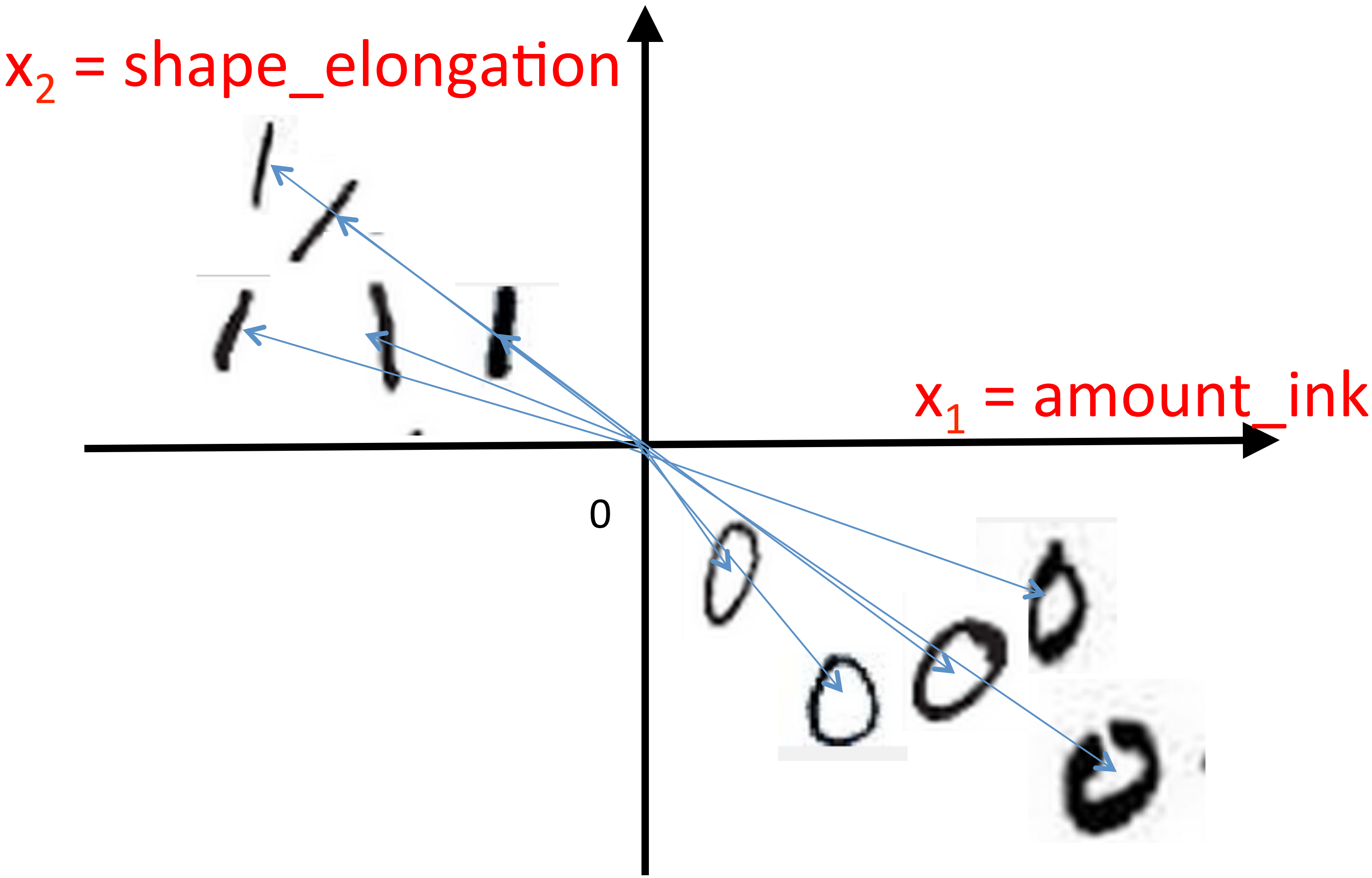
Predict an outcome **y**.

Conventions



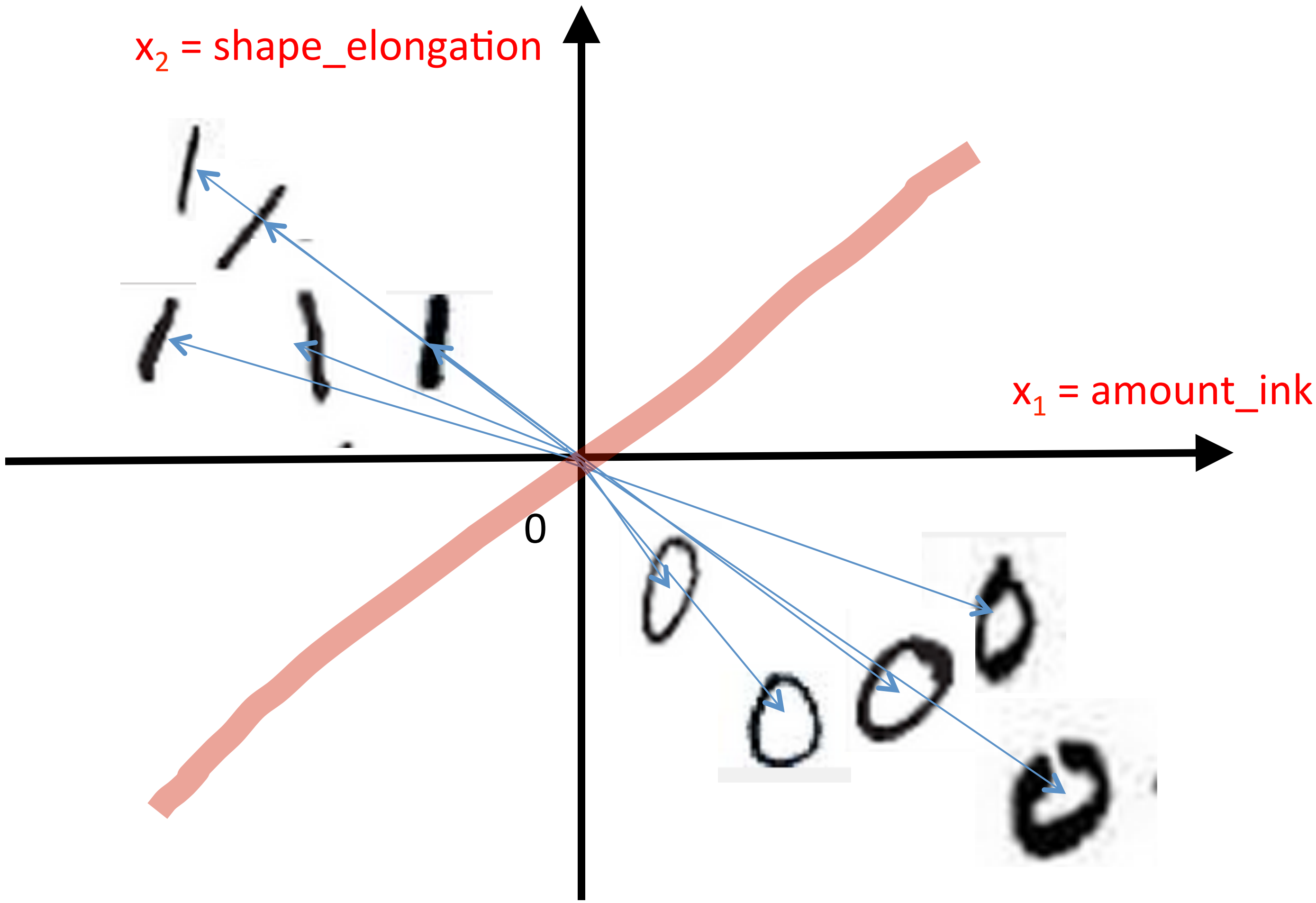
Separate “0” and “1”

How to build your first linear classifier?



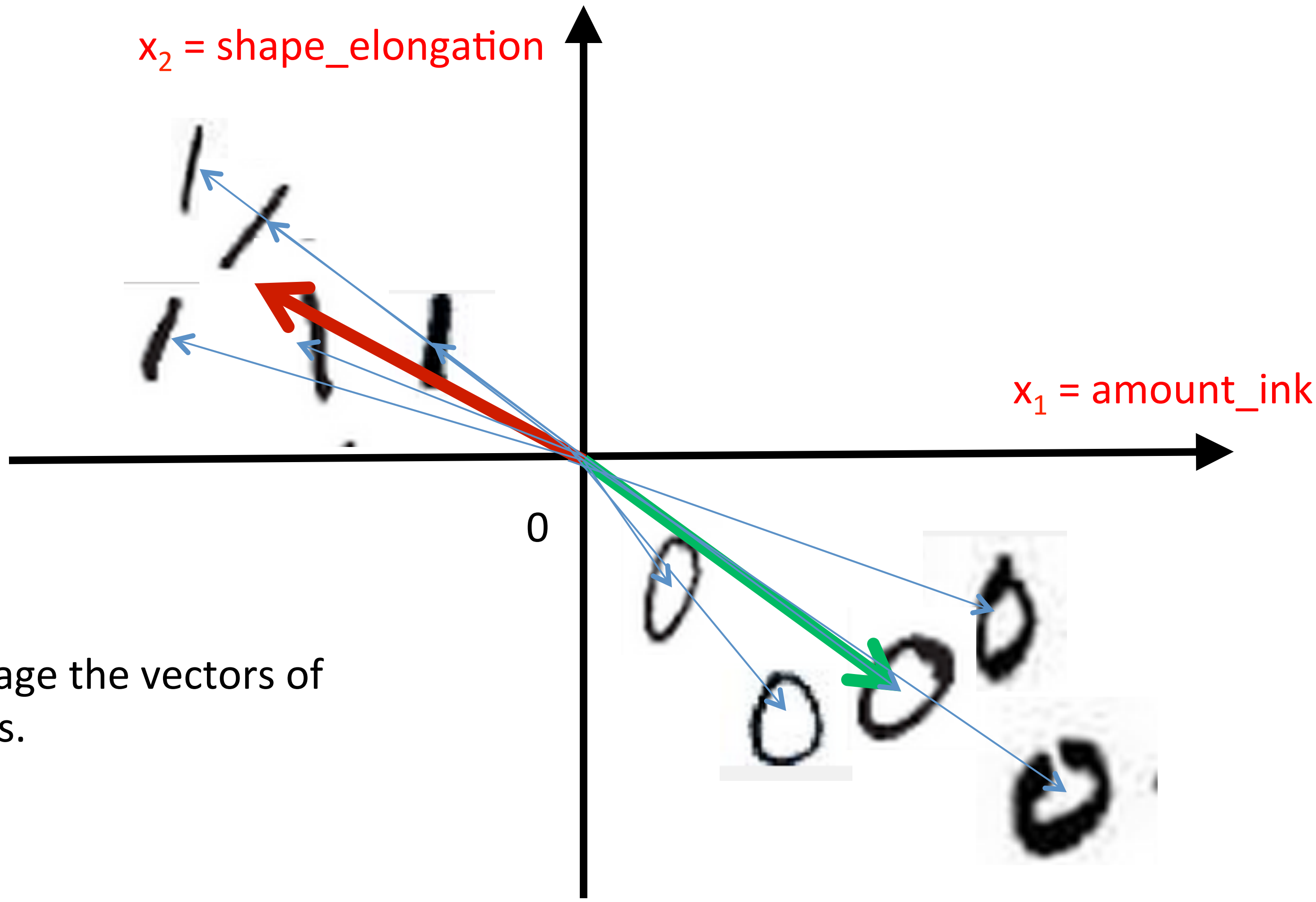
Separate “0” and “1”

How to build your first linear classifier?



Separate "0" and "1"

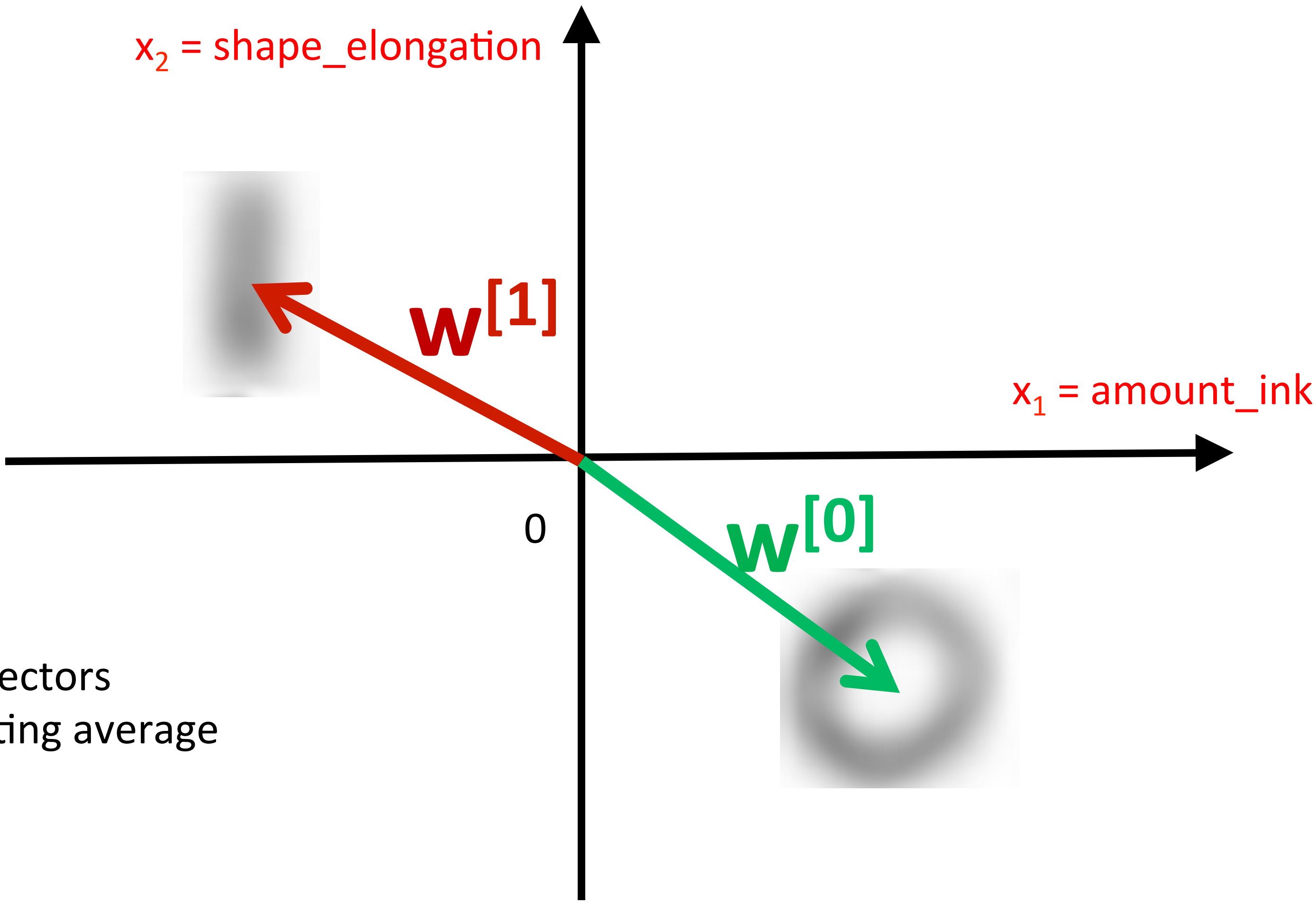
1) Find the class "centroids"



Just average the vectors of each class.

Separate "0" and "1"

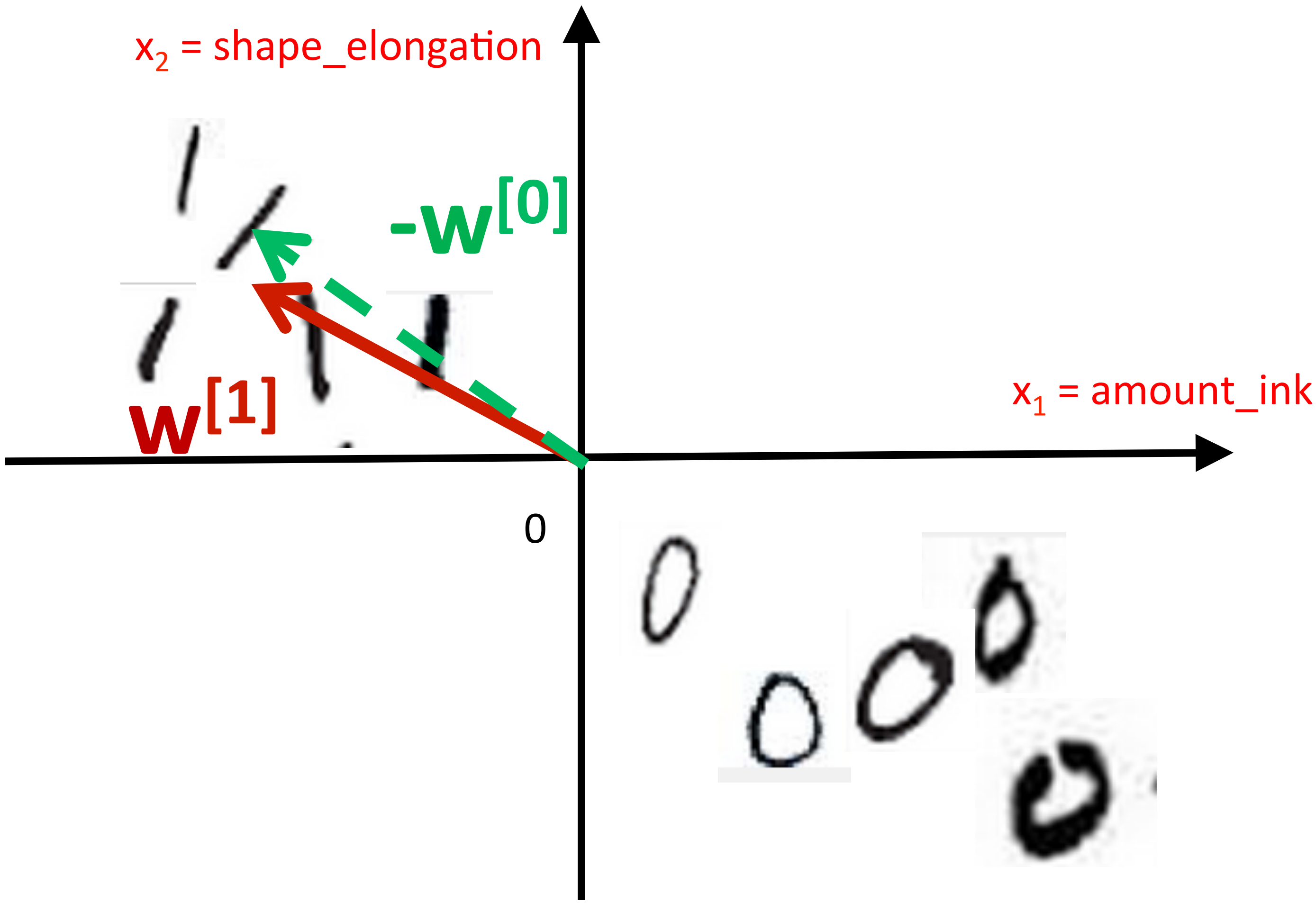
1) Find the class "centroids"



You get vectors representing average shapes.

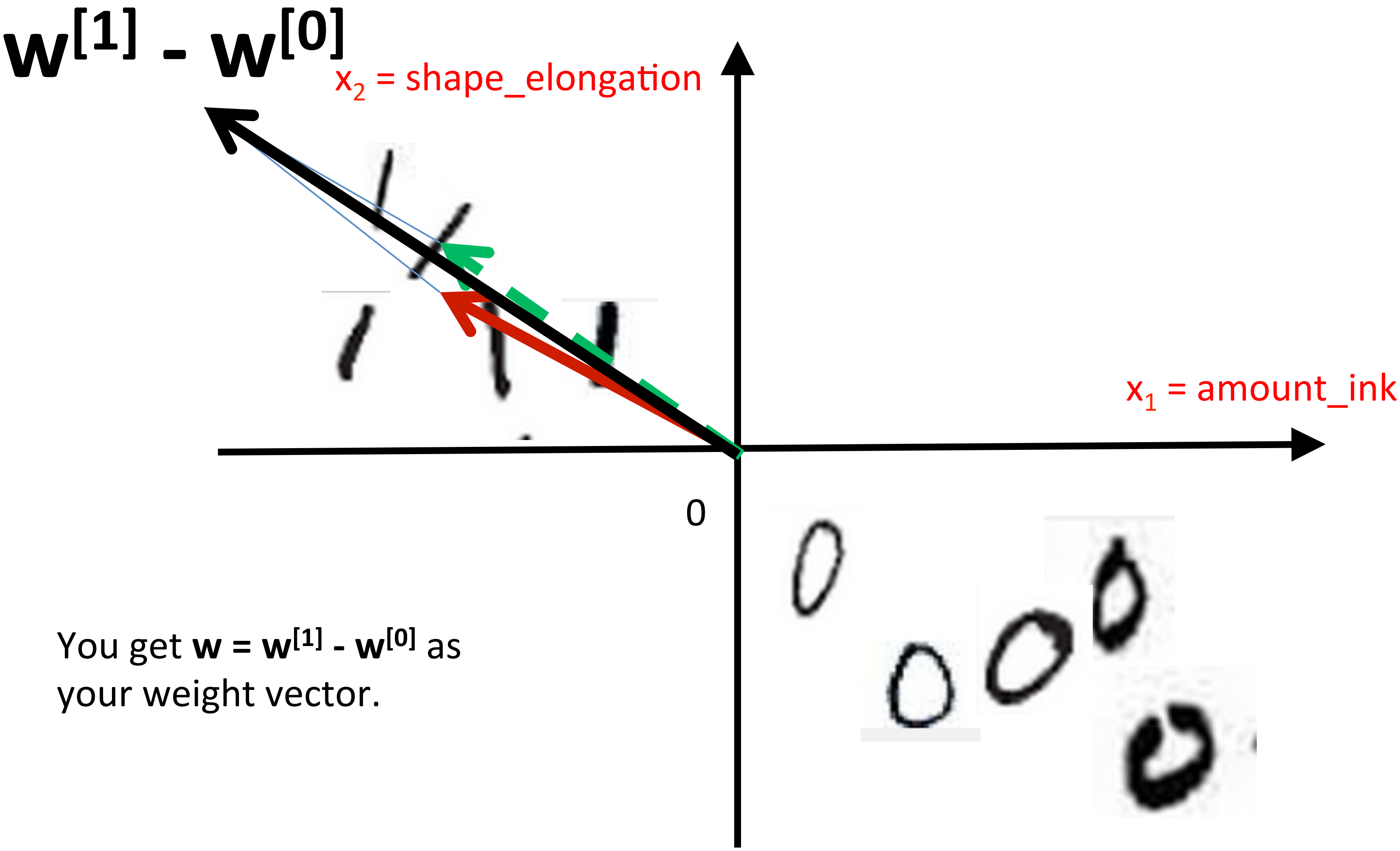
Separate "0" and "1"

2) Subtract $\mathbf{w}^{[0]}$ from $\mathbf{w}^{[1]}$



Separate "0" and "1"

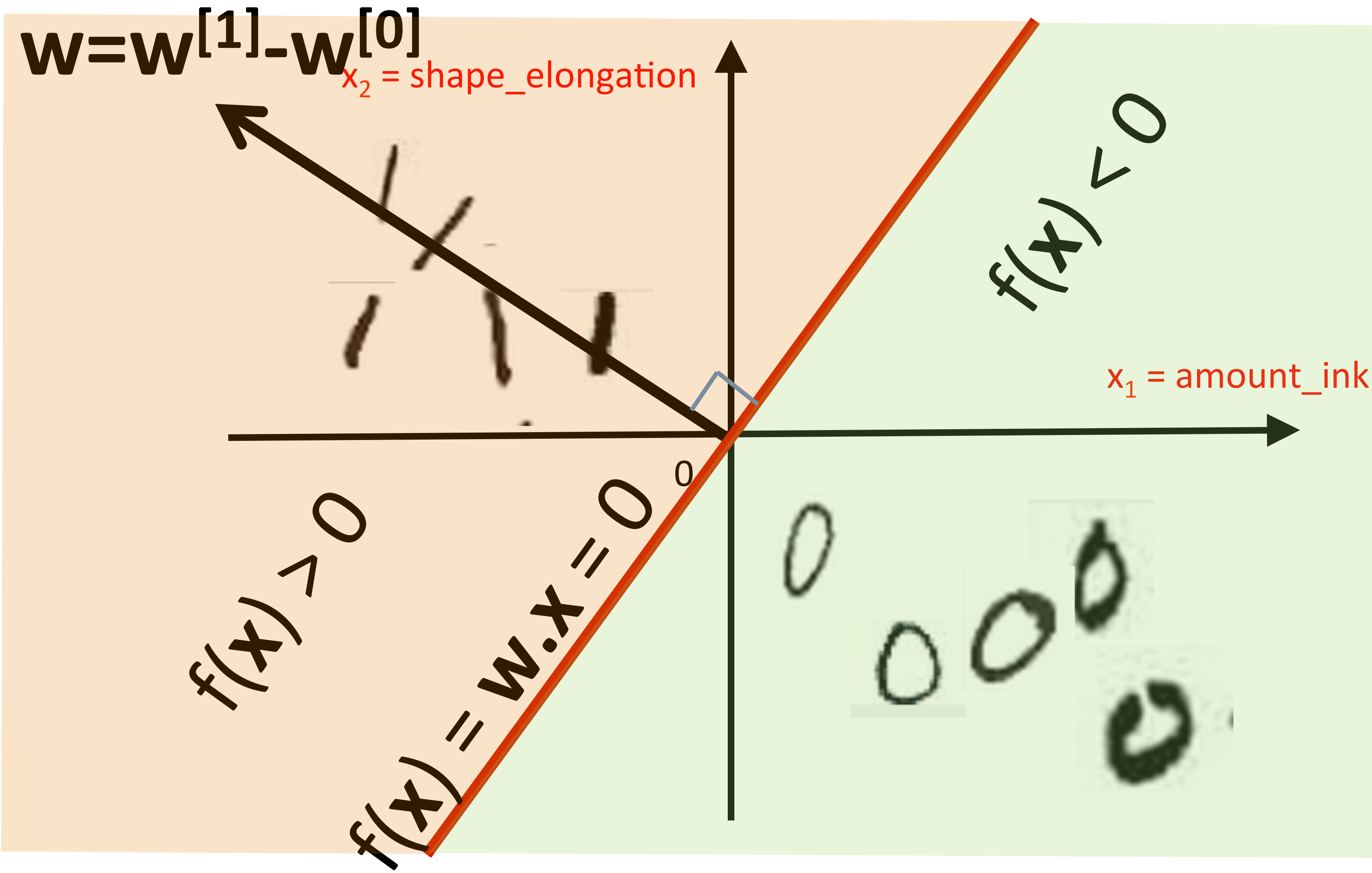
2) Subtract $\mathbf{w}^{[0]}$ from $\mathbf{w}^{[1]}$



You get $\mathbf{w} = \mathbf{w}^{[1]} - \mathbf{w}^{[0]}$ as your weight vector.

Separate "0" and "1"

3) Get the separaring "hyperplane"



Separate “0” and “1”

4) ALL the “math”

Input vector $\mathbf{x} = [x_1, x_2]$

x_1 = amount_ink

x_2 = shape_elongation

Target value $y = \pm 1$

+1 = “one”; -1 = “zero”

Training examples:

$\{ (\mathbf{x}^1, y^1), (\mathbf{x}^2, y^2), \ldots , (\mathbf{x}^N, y^N) \}$

Class centroids:

$\mathbf{w}^{[0]} \sim \sum_{\{y^k==-1\}} \mathbf{x}^k$ (~ means “proportional”, omitting to divide by class cardinality)

$\mathbf{w}^{[1]} \sim \sum_{\{y^k==+1\}} \mathbf{x}^k$

Weight vector:

$\mathbf{w} = \mathbf{w}^{[1]} - \mathbf{w}^{[0]} \sim \sum_k y^k \mathbf{x}^k$ (for “balanced” classes)

Separate "0" and "1"

4) ALL the "math" (continued)

Decision function:

$f(\mathbf{x}) = \mathbf{w} \,.\, \mathbf{x}$

$f(\mathbf{x}) > 0$, decide that this is a "one"

$f(\mathbf{x}) < 0$, decide that this is a "zero"

Dot product:

$\mathbf{w} \,.\, \mathbf{x} = w_1 x_1 + w_2 x_2$ This is a weighted sum.

Equivalent "centroid" method:

$f(\mathbf{x}) = \mathbf{w}^{[1]} \,.\, \mathbf{x} - \mathbf{w}^{[0]} \,.\, \mathbf{x}$ This is because $\mathbf{w} = \mathbf{w}^{[1]} - \mathbf{w}^{[0]}$.

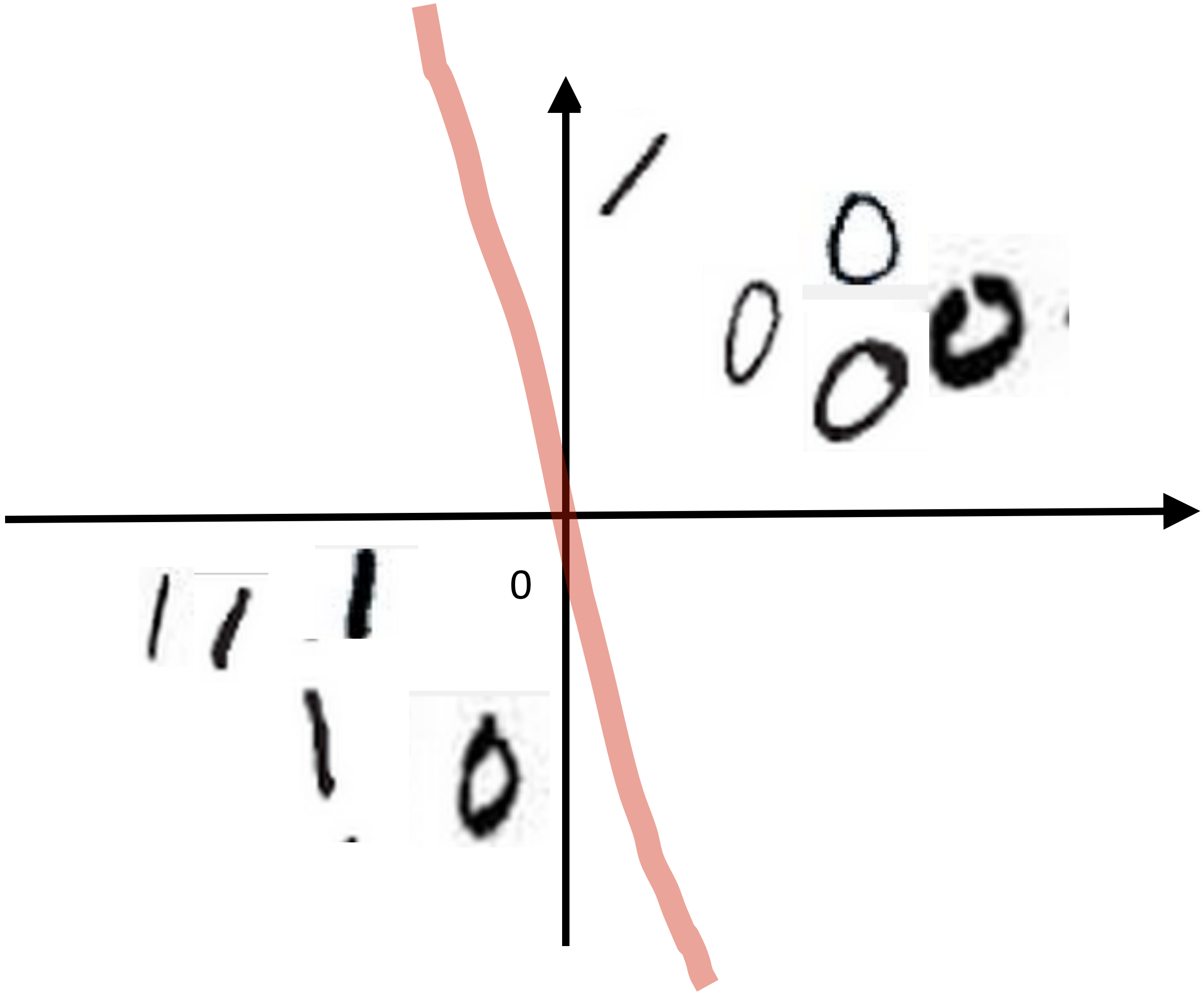
Decide "one" if $\mathbf{w}^{[1]} \,.\, \mathbf{x} > \mathbf{w}^{[0]} \,.\, \mathbf{x}$ and "zero" otherwise

A dot product is a similarity measure.

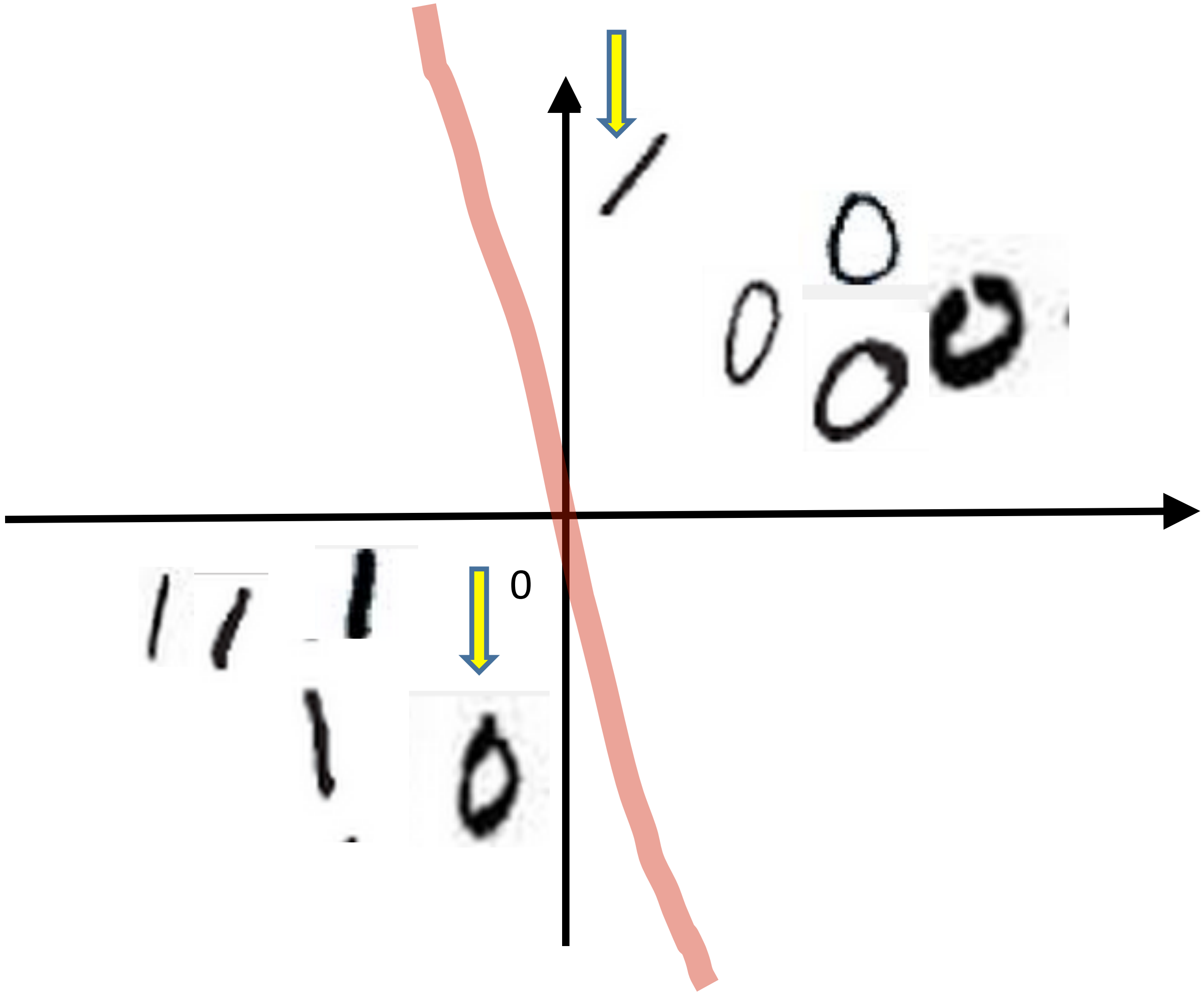
Equivalent "kernel" method: This is because $\mathbf{w} = \sum_k y^k \mathbf{x}^k$

$f(\mathbf{x}) = \sum_k y^k \mathbf{x}^k \,.\, \mathbf{x} = \sum_k \alpha^k\, k(\mathbf{x}^k, \mathbf{x})$ (in the case of identical number of examples for the two classes)

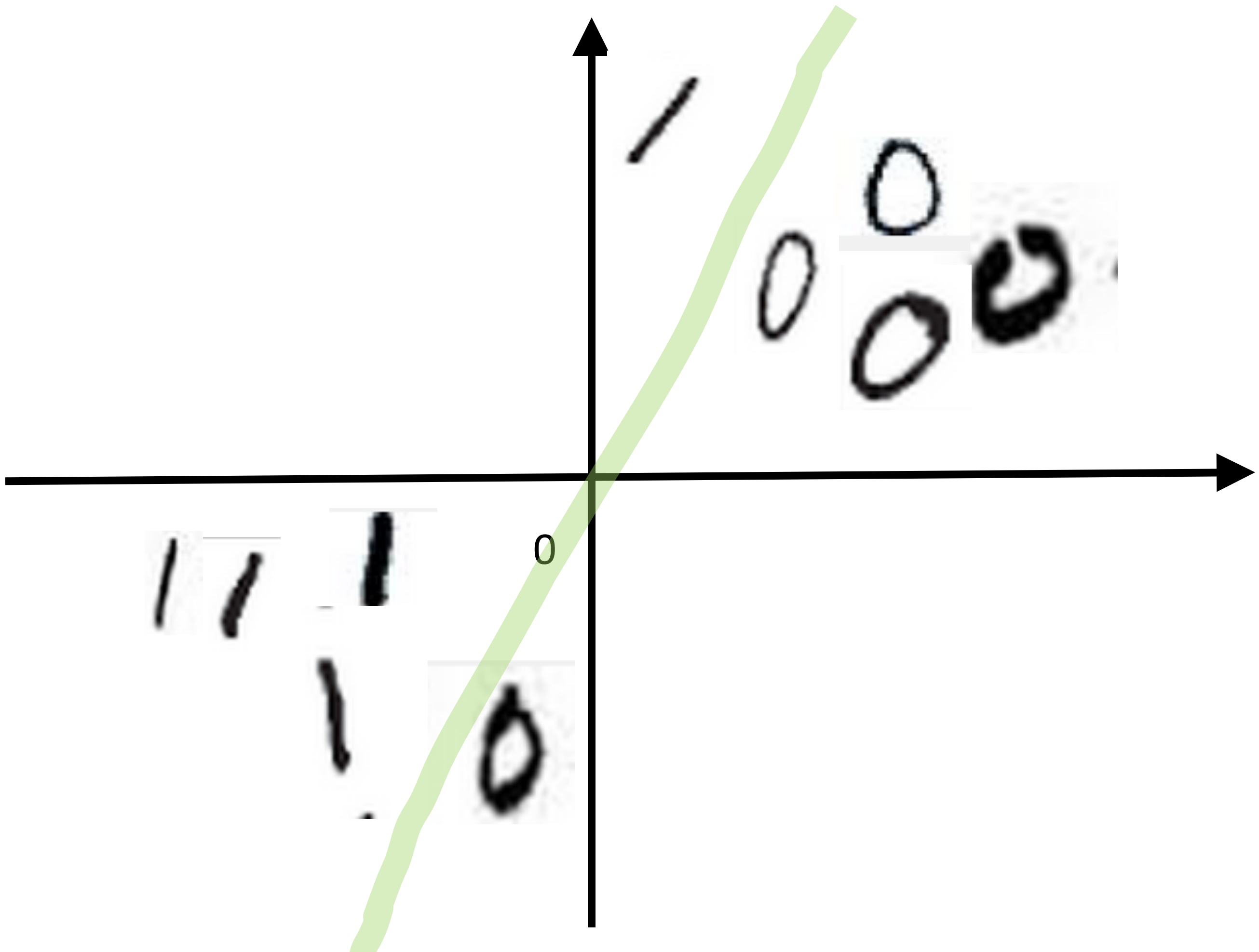
Centroid methods don't always work…



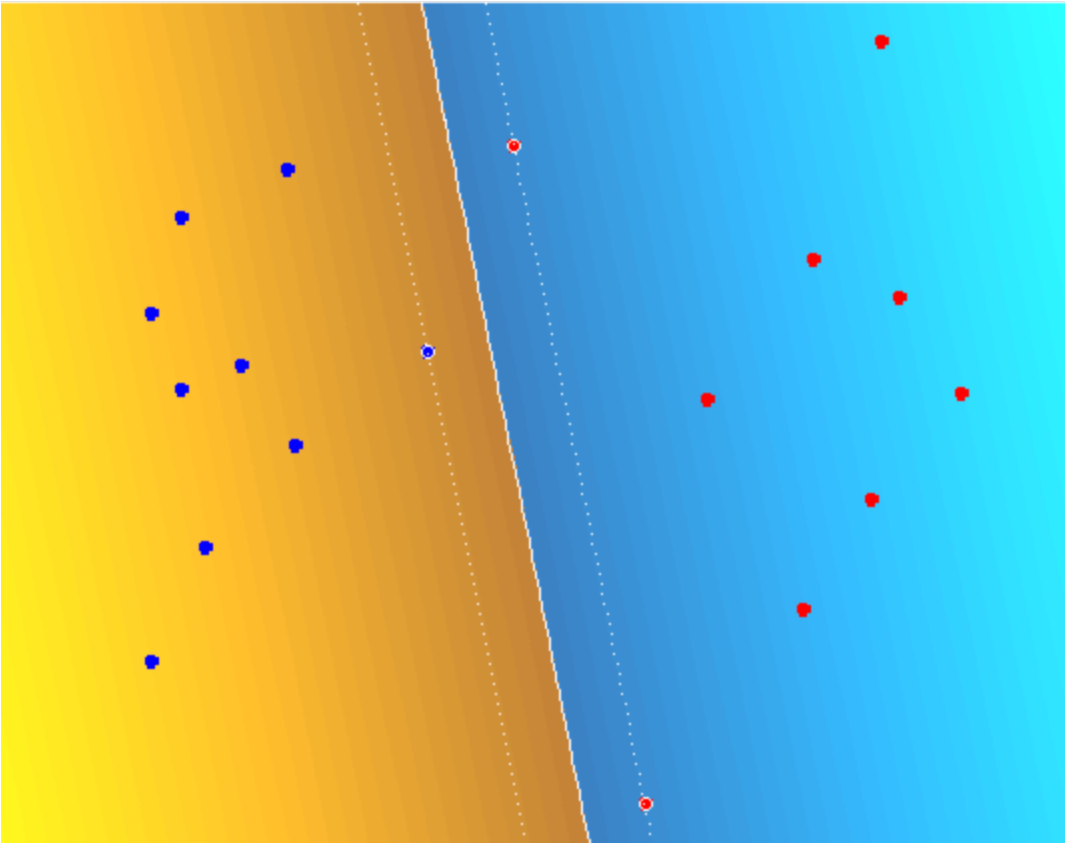
Centroid methods don't always work...



Centroid methods don't always work…



Demo of SVM
(downloadable from bCourses)



Linear Models

"Regular" linear models, linear in their inputs and parameters

$$f(\mathbf{x}) = \mathbf{w} \bullet \mathbf{x} + b = \sum_{i=1:d} w_i x_i + b$$

Models *linear in their parameters, NOT in their inputs.*

$$f(\mathbf{x}) = \mathbf{w} \bullet \Phi(\mathbf{x}) + b = \sum_i w_i \phi_i(\mathbf{x}) + b \quad \textit{(Perceptron)}$$

$$f(\mathbf{x}) = \sum_{k=1:N} \alpha_k k(\mathbf{x}^k, \mathbf{x}) + b \quad \textit{(Kernel method)}$$

Linear Models

"Regular" linear models, <u>linear in their inputs and parameters</u>

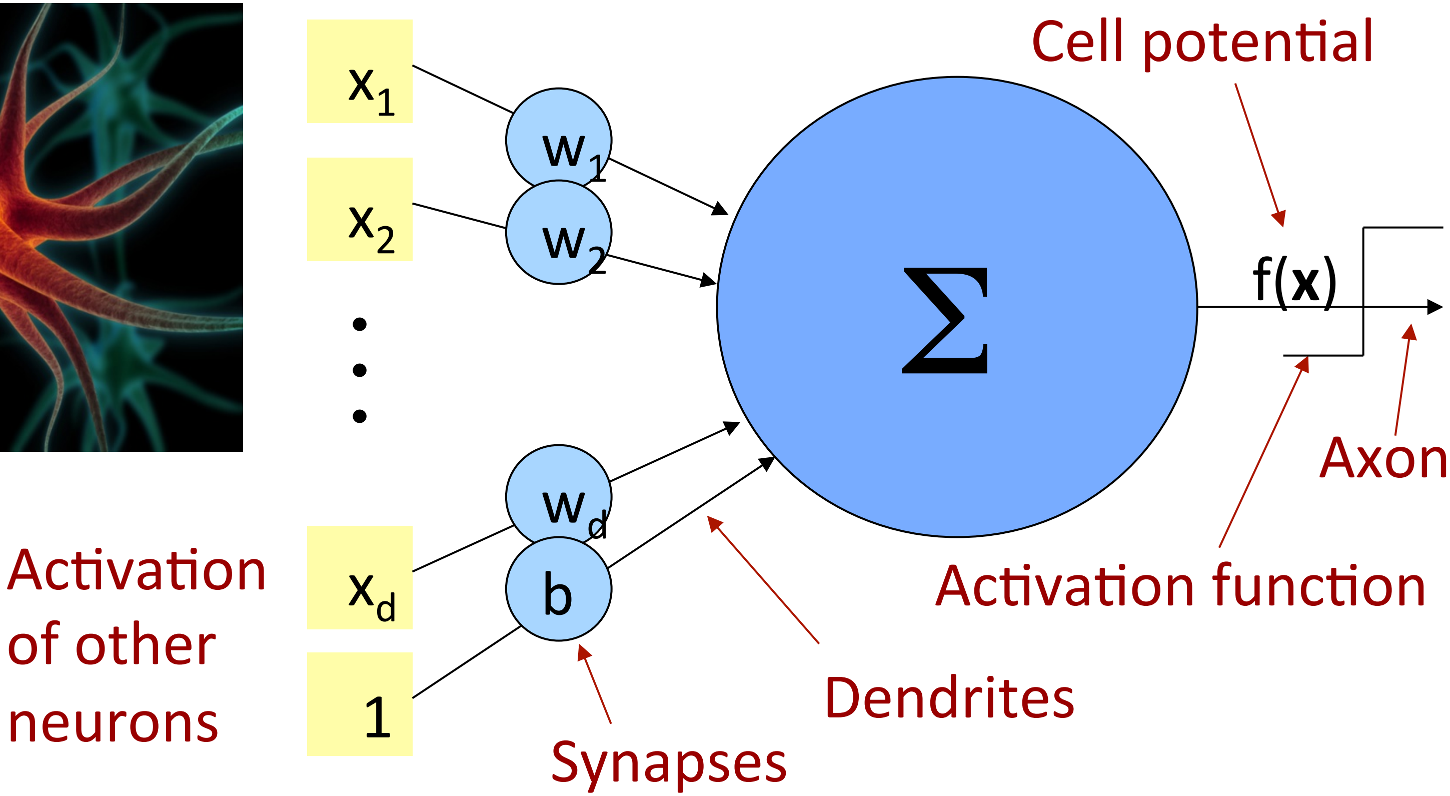
$$f(\mathbf{x}) = \mathbf{w} \bullet \mathbf{x} + b = \sum_{i=1:d} w_i x_i + b$$

Models *<u>linear in their parameters,</u> NOT in their inputs.*

$$f(\mathbf{x}) = \mathbf{w} \bullet \Phi(\mathbf{x}) + b = \sum_i w_i \phi_i(\mathbf{x}) + b \quad \textit{(Perceptron)}$$

$$f(\mathbf{x}) = \sum_{k=1:N} \alpha_k \, k(\mathbf{x}^k, \mathbf{x}) + b \quad \textit{(Kernel method)}$$

Artificial Neurons

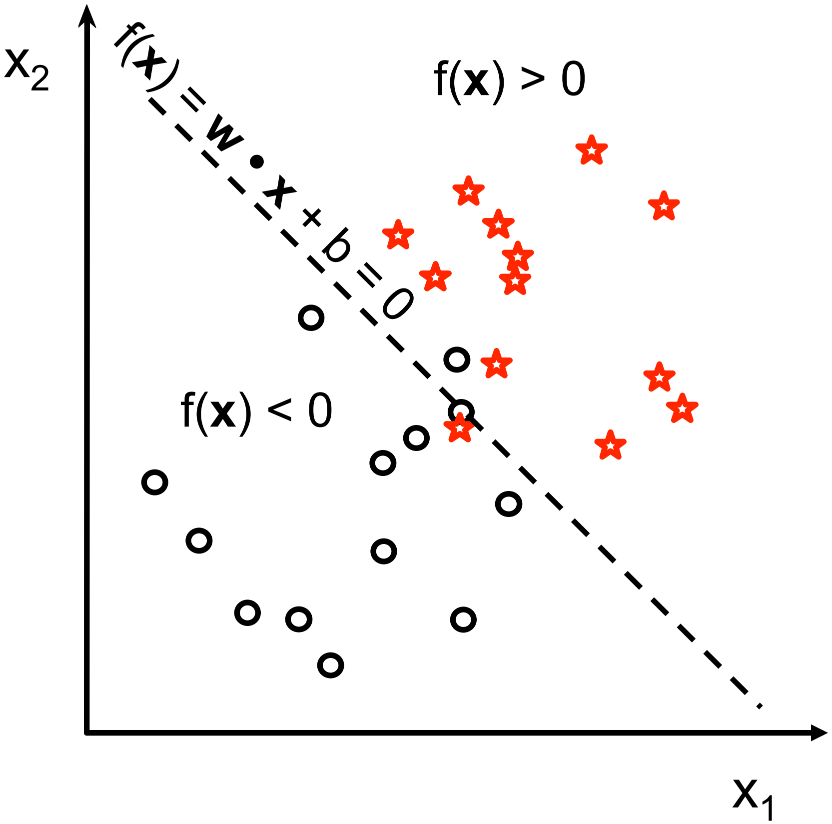


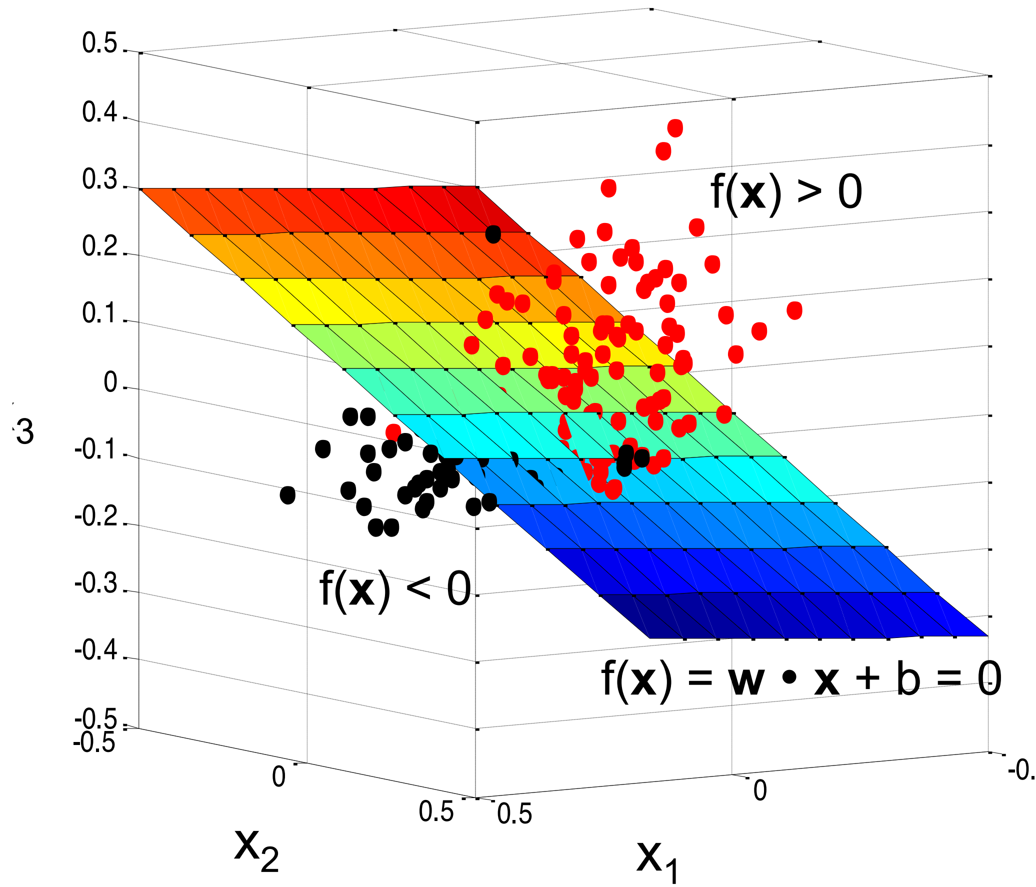
McCulloch and Pitts, 1943

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$$

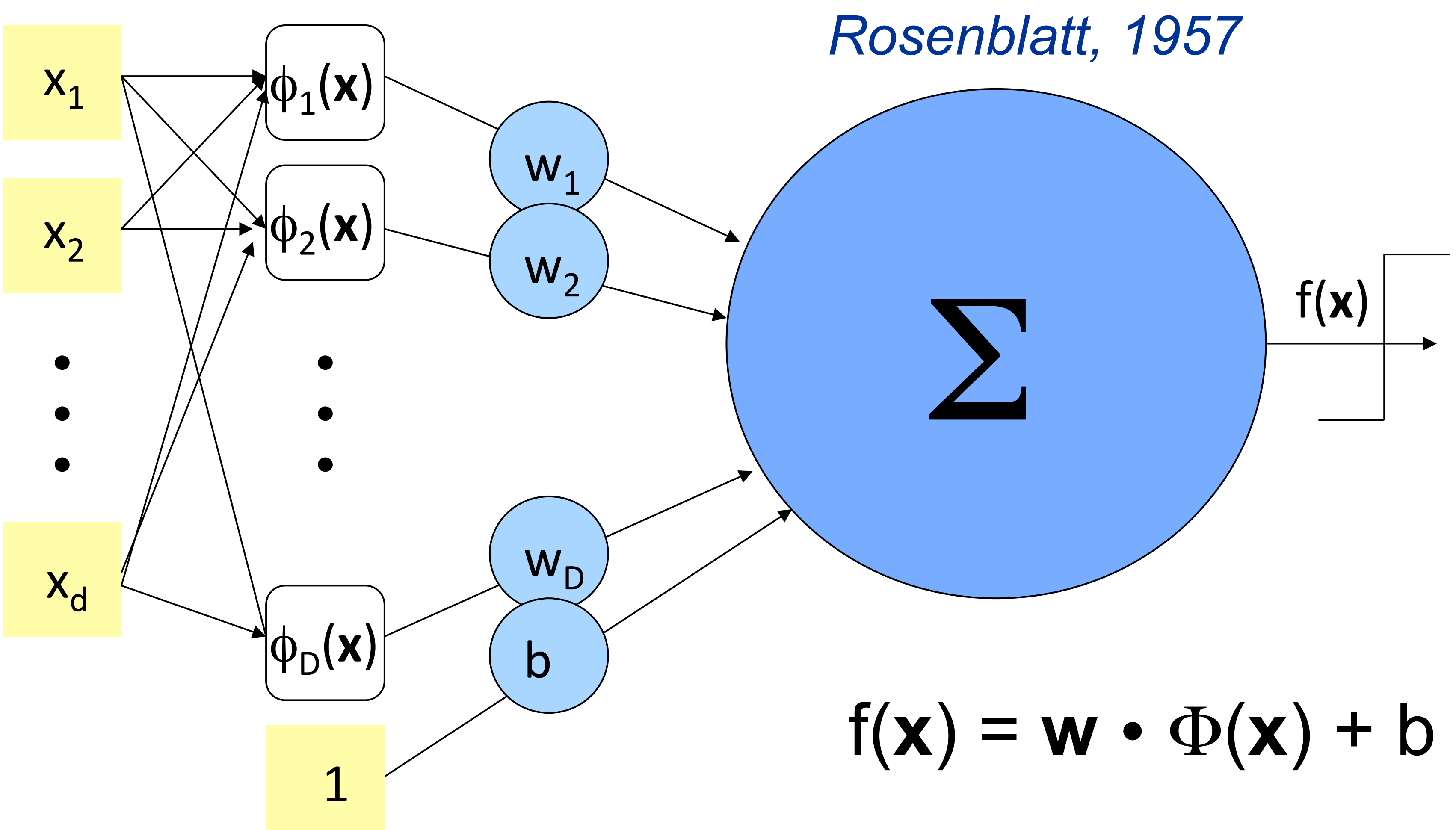
Linear decision boundary

hyperplane



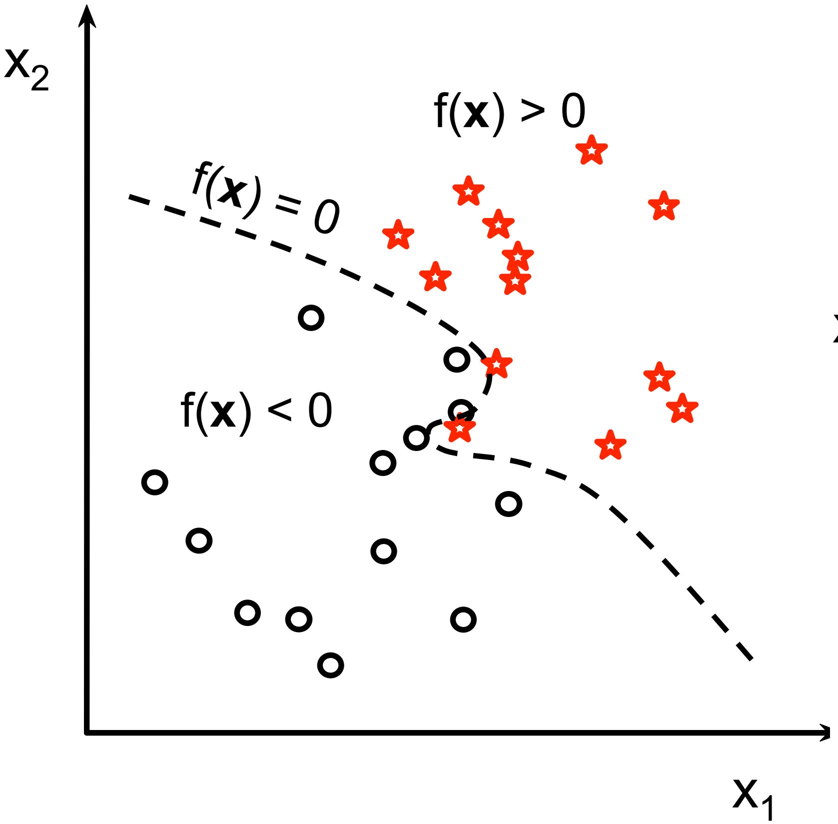


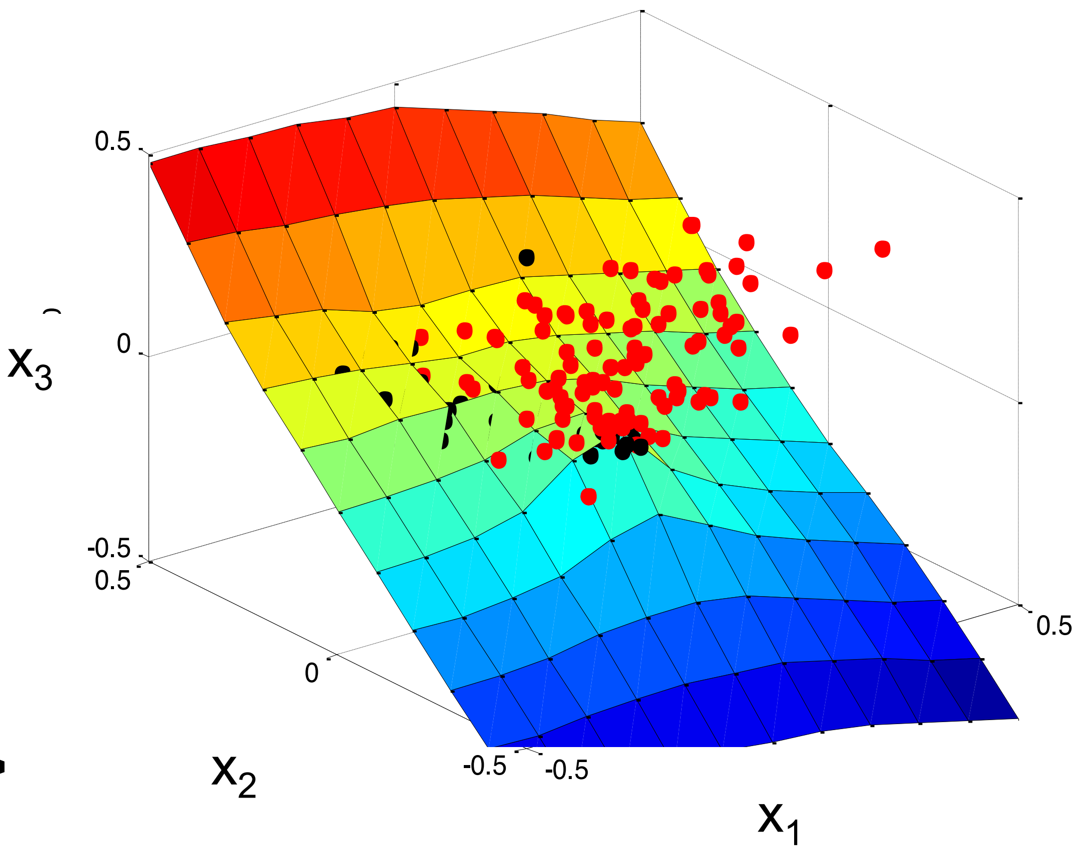
Perceptron



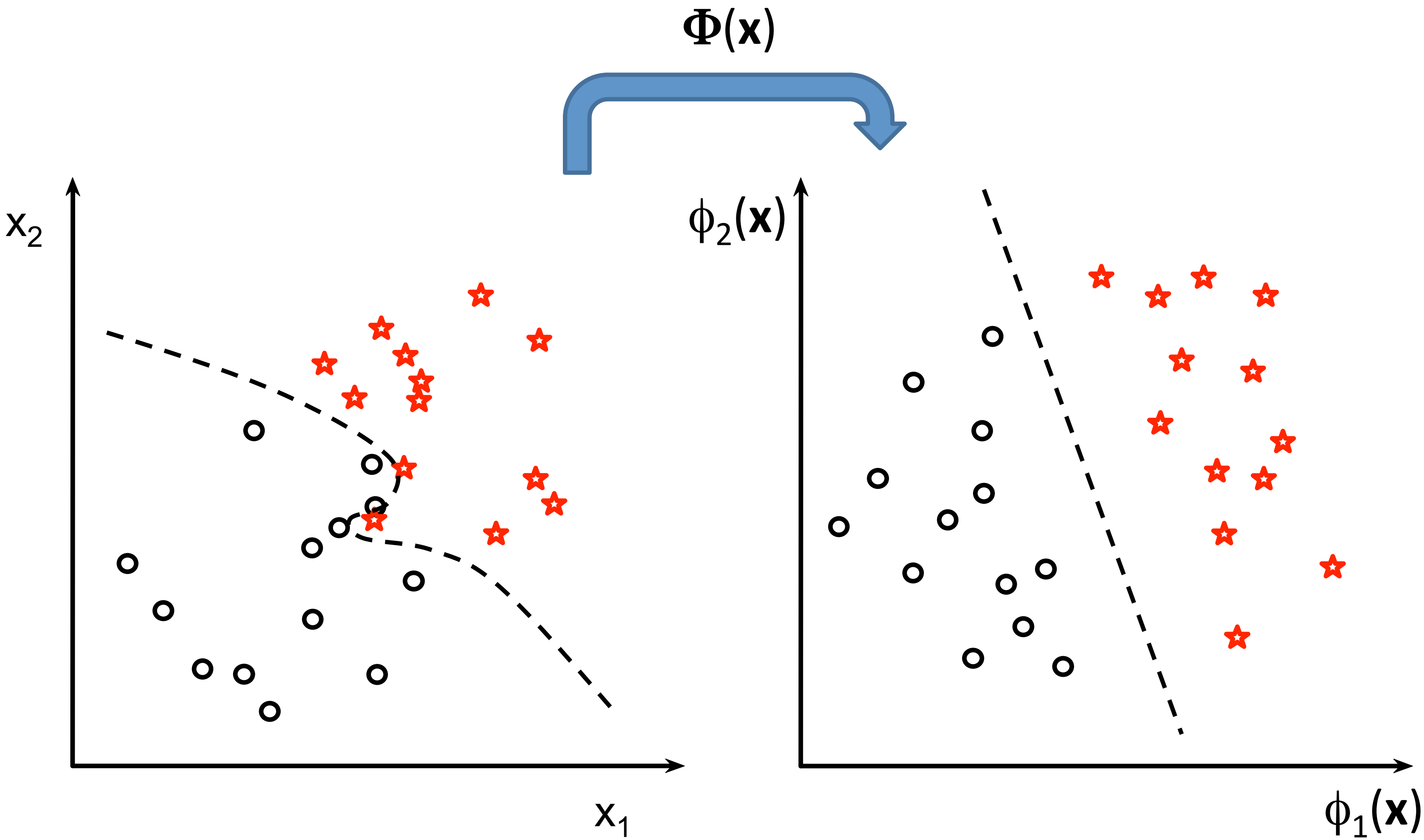
$$f(\mathbf{x}) = \mathbf{w} \cdot \Phi(\mathbf{x}) + b$$

Non-linear decision boundary





Linear decision boundary in Φ-space



Summary

- We represent patterns as vectors **x** in a space of d dimensions.
- A "discriminant function" f(**x**) is a function such that f(**x**) > 0 for one class and f(**x**) < 0 for the other. f(**x**)=0 is the equation of the decision boundary.
- Given a weight vector **w**, f(**x**)=**w.x** is a linear discriminant function. The corresponding decision boundary **w.x**=0 is a hyperplane (a subspace of dimension (d-1)).
- Feature transforms **x** -> Φ(**x**) permit to built non-linear decision boundaries, while using discriminant linear in **w** (NOT in x).

Come to my office hours…
Wed 2:30-4:30 Soda 329

Next time

- With linear threshold units (“neurons”) we can build:
  - Linear discriminant
  - Kernel methods
  - Neural networks

DUAL

PARAMETRIC

NON PARAMETRIC

- The architectural hyper-parameters may include:
  - The choice of basis functions ϕ (features)
  - The kernel
  - The number of hidden units.

Control the COMPLEXITY

- “Complex” models are prone to overfitting.