

Machine Learning

CSCI 5622 Fall 2020

Prof. Claire Monteleoni

Today

- Collaborative filtering and nearest neighbor
- [If time] Intro to Linear Classification

With credit to T. Jaakkola and S. Dasgupta

Collaborative Filtering and Nearest Neighbor



Jade Yoga Elite Yoga Mat

by [Jade Yoga](#)

(1 customer review) | Like (0)



Customers Who Bought This Item Also Bought

Page 1 of 4



Aurorae Organic Yoga Mat Wash-Cleans, Restores, Refreshes... by Aurorae

(27)

\$11.95



Jade Harmony Professional 3 / 16-Inch Yoga Mat by Jade Yoga

(162)

\$62.95 - \$105.00



Jade Fusion 74-inch Yoga Mat by Jade Yoga

(36)

\$116.78 - \$169.95

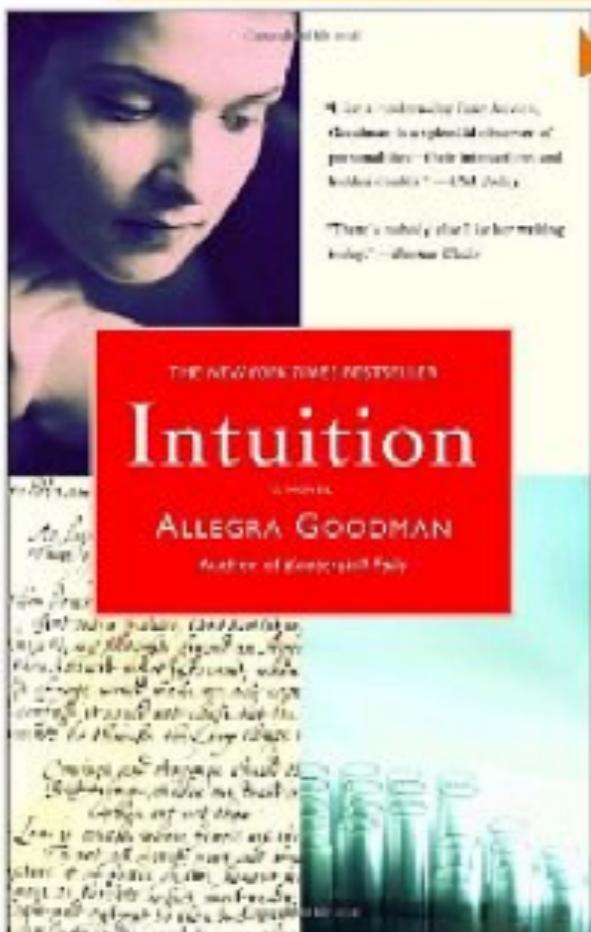


Yoga Mat Bag Extra Large Easy Open Zipper 100% C... by Bean Products

(44)

\$18.95 - \$21.95

[Click to LOOK INSIDE!](#)



Intuition [Paperback]

[Allegra Goodman](#) (Author)



[\(69 customer reviews\)](#)

[Like](#) (4)

This intimate portrait of life in a research institute in Cambridge, Massachusetts, revolves around a scientific mystery....

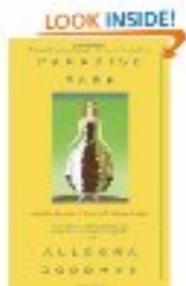
Customers Who Bought This Item Also Bought



[Kaaterskill Falls](#) by
Allegra Goodman

(62)

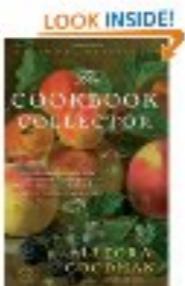
\$10.40



[Paradise Park](#) by Allegra
Goodman

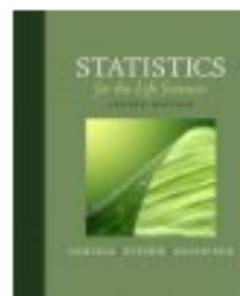
(36)

\$11.12



[The Cookbook
Collector: A Novel](#) by
Allegra Goodman

(135)
\$10.20



[Statistics for the Life
Sciences \(4th Edition\)](#) by
Myra L. Samuels

(20)
\$113.99

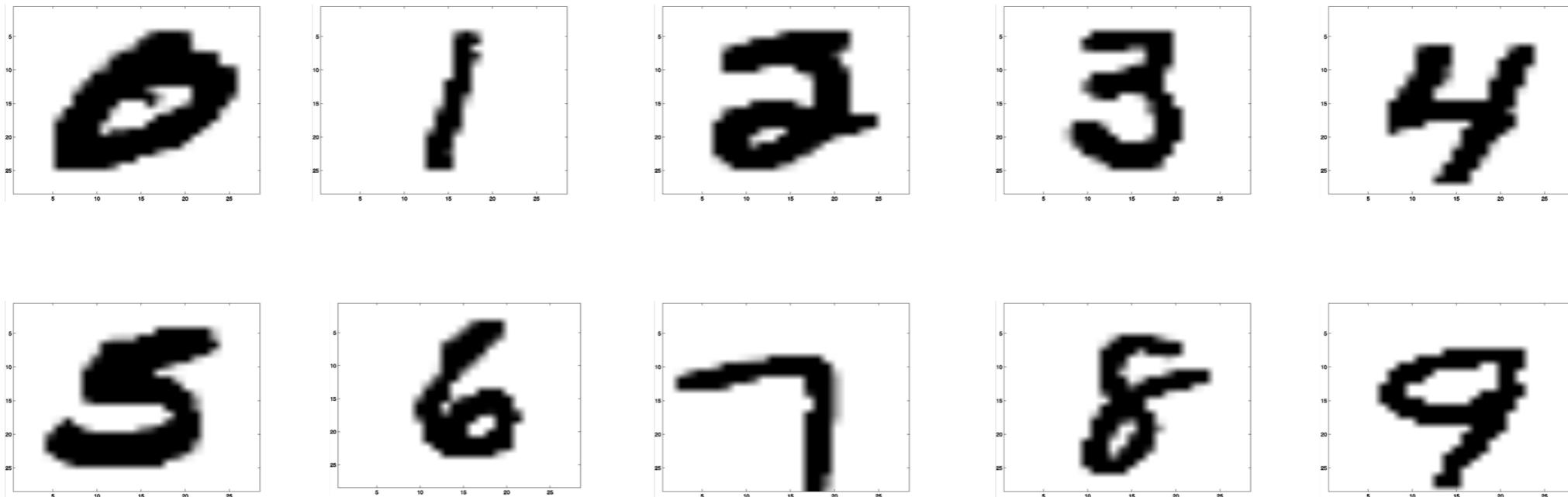
Page 1 of 25

Nearest Neighbor Classification

[With much credit to S. Dasgupta]

A first example

Database of 20,000 images of handwritten digits, each
labeled by a human



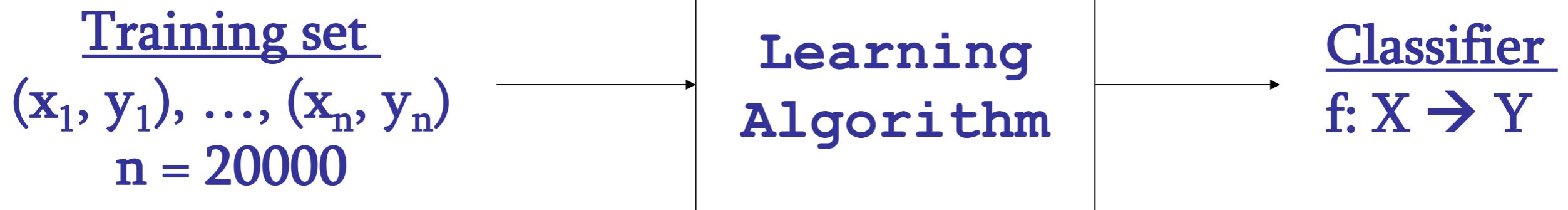
[28 x 28 greyscale; pixel values 0-255; labels 0-9]

Use these to learn a classifier which will label digit-images automatically...

The learning problem

Input space $X = \{0, 1, \dots, 255\}^{784}$

Output space $Y = \{0, 1, \dots, 9\}$



To measure how good f is: use a test set.

We have a test set with 100 instances of each digit.

A possible strategy

Input space $X = \{0, 1, \dots, 255\}^{784}$

Output space $Y = \{0, 1, \dots, 9\}$

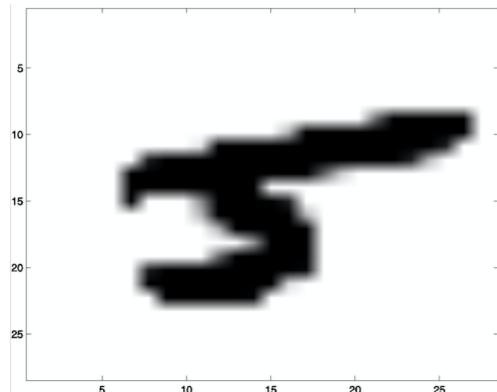
Treat each image as a point in 784-dimensional Euclidean space

To classify a new (unlabeled) image:
find its nearest neighbor in the database (training set)
and return that label

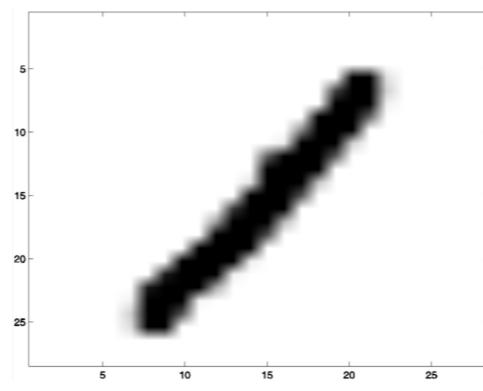
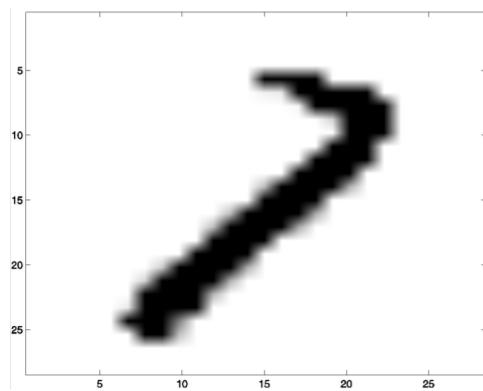
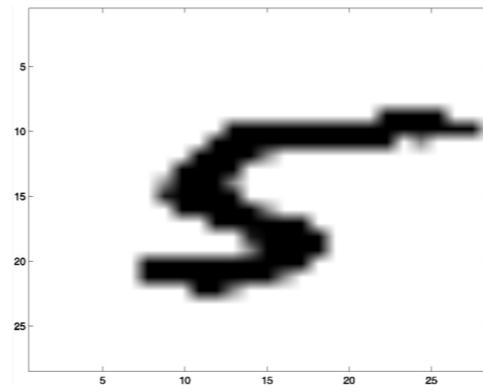
$f = \text{search engine} + \text{entire training set}$

Nearest neighbor

Image to label



Nearest neighbor



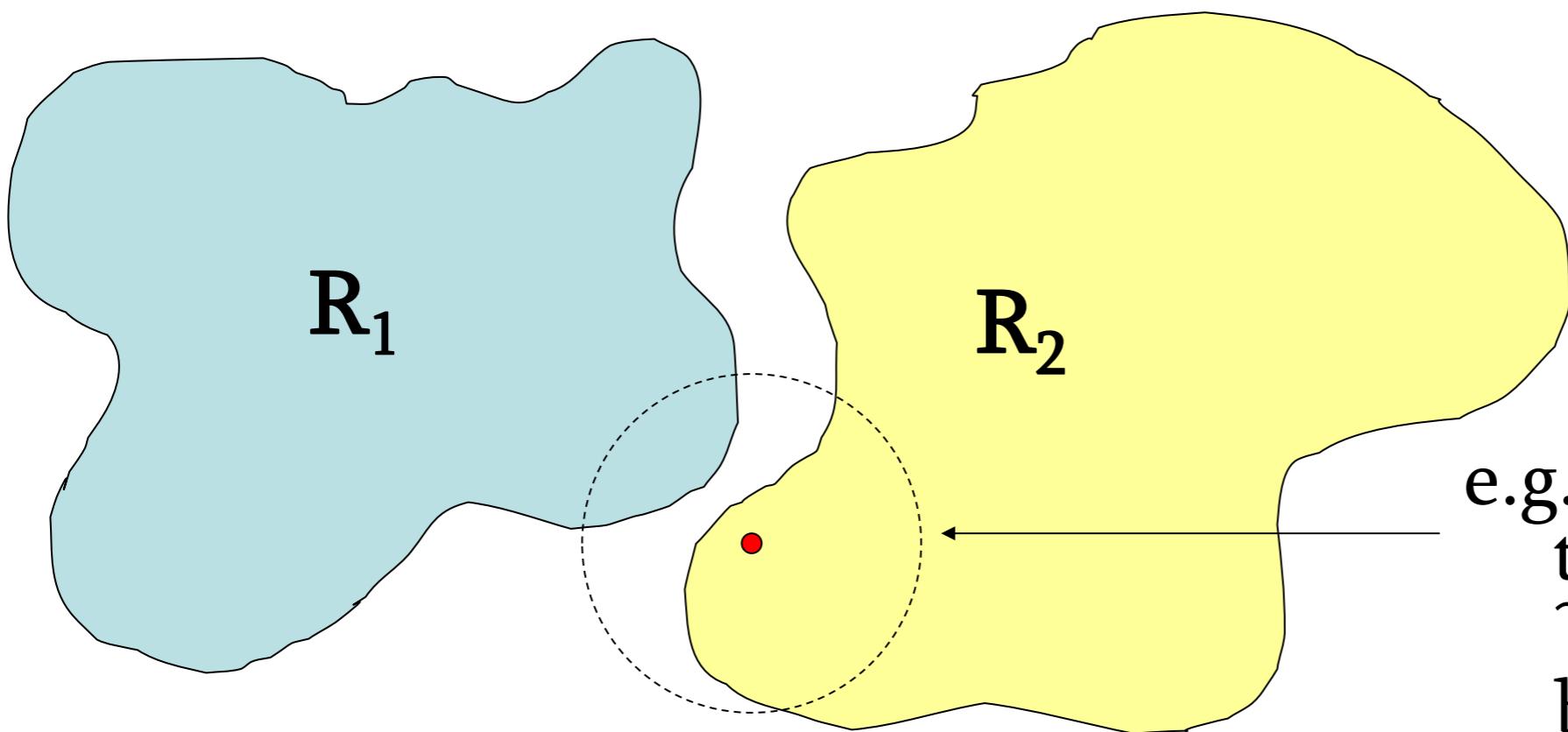
Overall:
error rate = 6%
(on test set)

Question: what is
the error rate for
random guessing?

What does it get wrong?

Who knows... but here's a hypothesis:

Each digit corresponds to some connected region of \mathbb{R}^{784} . Some of the regions come close to each other; problems occur at these boundaries.



e.g. a random point in
this ball has only
~70% chance of
being in R_2

“Boost” the probability of success

Analogy: suppose a (biased) coin has

$$\Pr(\text{heads}) = 0.70$$

Flip it 11 times and return the majority vote:

$$\Pr(\text{heads}) = 0.92$$

Therefore: to classify x , find its k nearest neighbors (in the training set) and return their majority vote

[Large deviation theory: the foundation of machine learning...]

Time complexity

To classify a new (unlabeled) image:

find its nearest neighbor in the database (training set)
and return that label

Given n training points in R^d , what is the time complexity
to label a new image, q ?

For each image, x , in the database: $= O(n)$

Compute the Euclidean distance between x and q .

Return the x_{\min} that minimizes this distance.

$$d(q, x) = \sqrt{\sum_{i=1}^d (q_i - x_i)^2} = O(d)$$

$$= O(nd)$$

(Simple) Nearest neighbor: pros and cons

Pros

Simple

Flexible

Excellent performance on a wide range of tasks

Cons

Algorithmic:

- Time consuming: with n training points in \mathbb{R}^d , time to label a new point is $O(nd)$.

Statistical:

- This is just memorization, not learning!
- No insight is gained about the problem.
- Would prefer a compact classifier.

Beyond Simple Nearest Neighbor

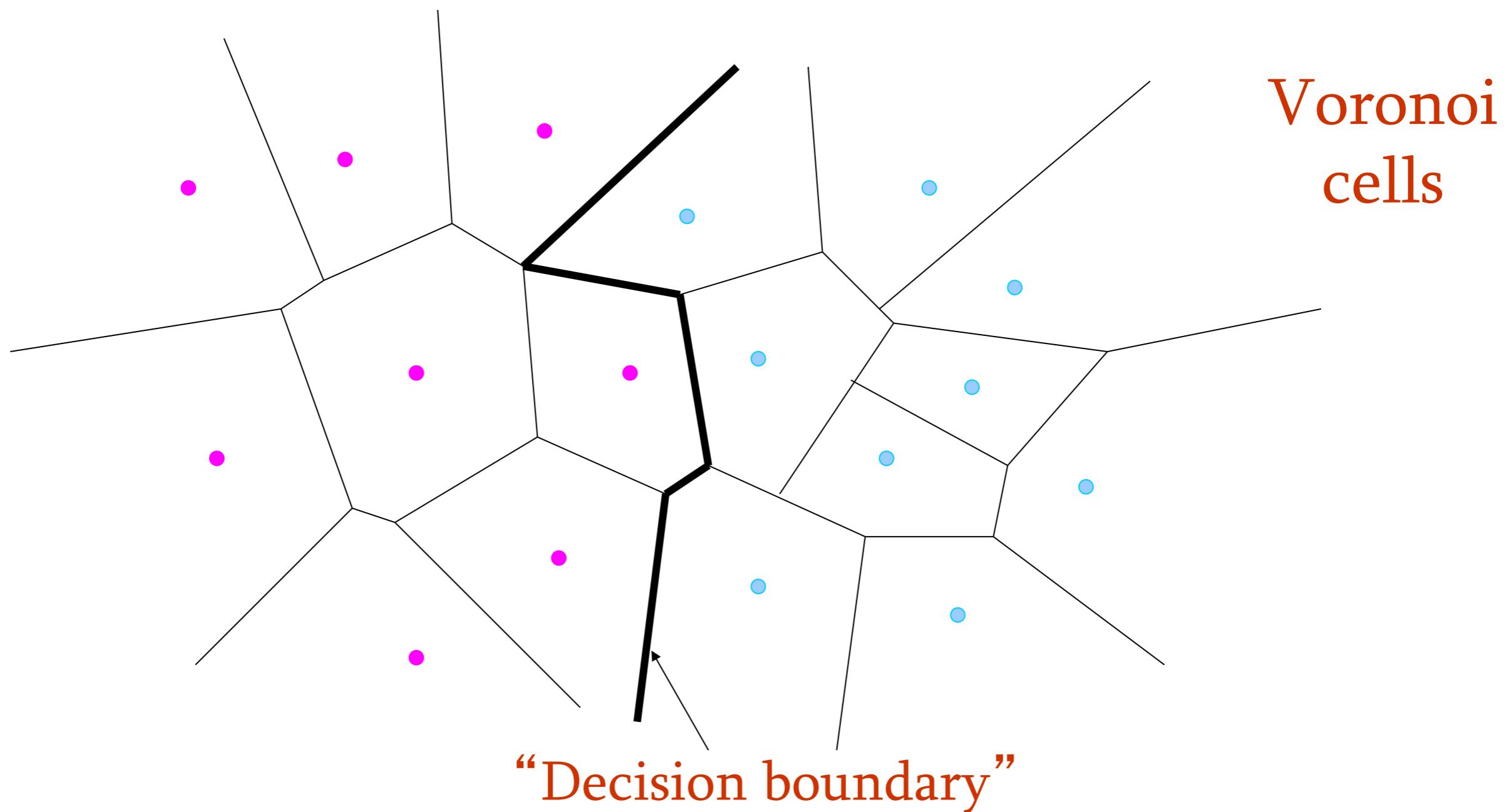
Learning problem:

Design algorithms to perform nearest neighbor classification:

- with low error
- and low memory: do not store all the training examples!

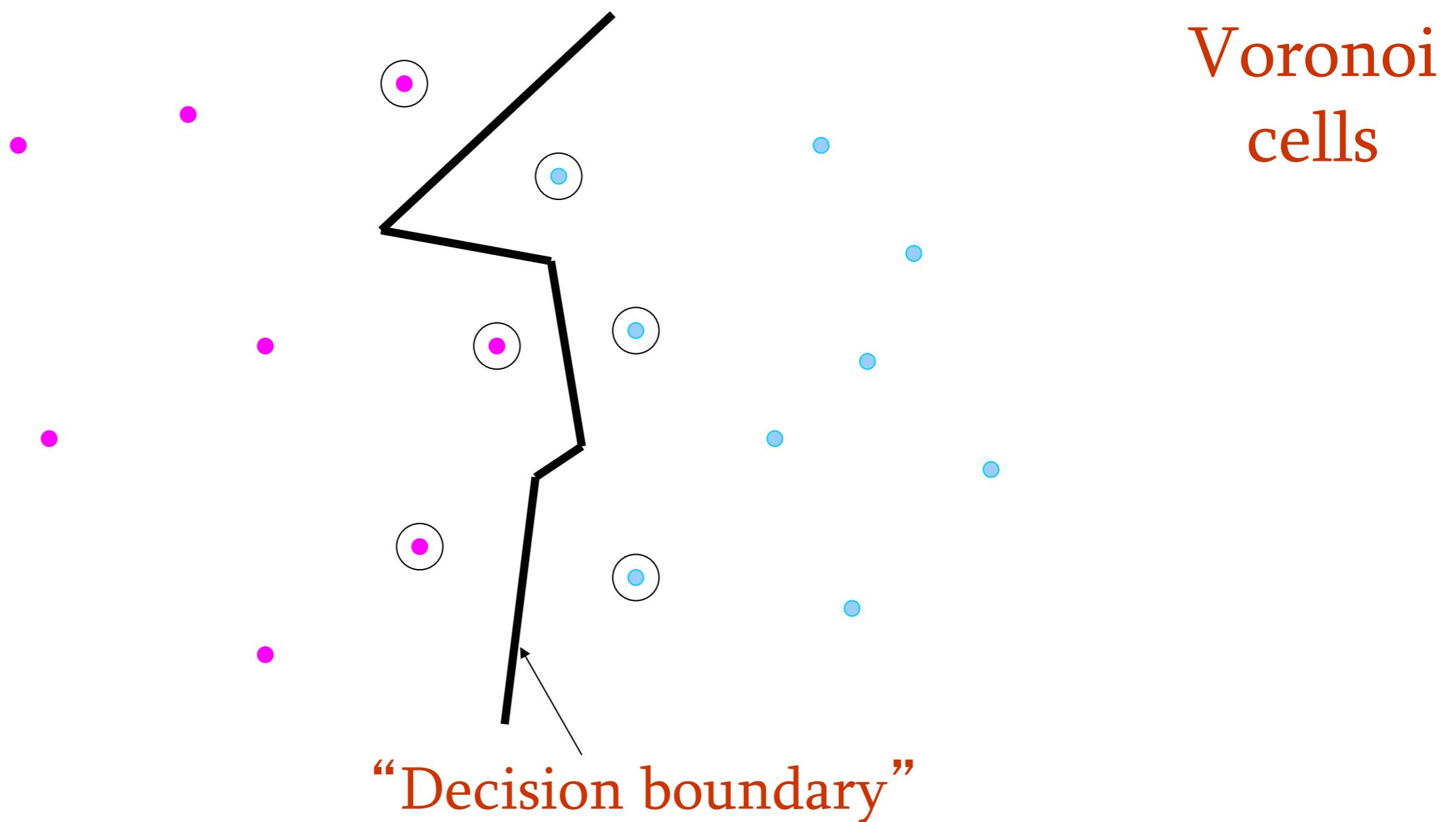
Prototype selection

A possible fix: instead of the entire training set, just keep a “representative sample”



Prototype selection

A possible fix: instead of the entire training set, just keep a “representative sample”



How to pick prototypes?

Idea 1: sample at random from training data

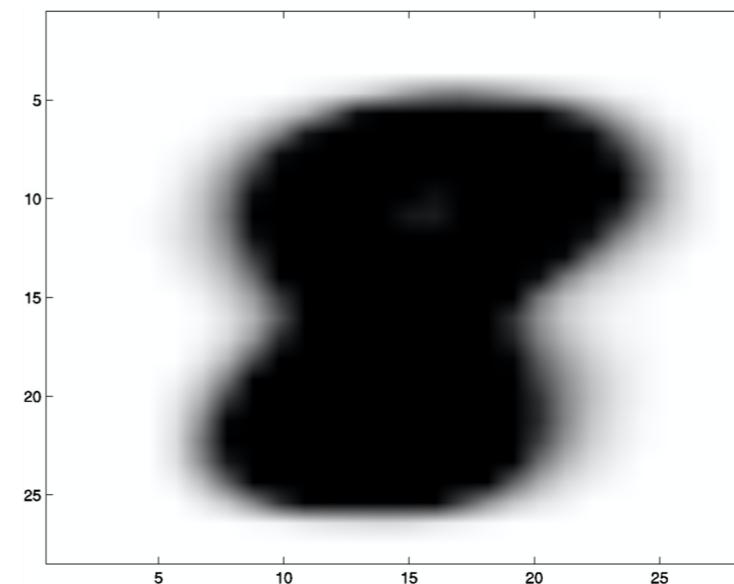
# prototypes	% error rate
20000	6.0
10000	7.4
5000	8.5
2000	10.7
1000	14.3
500	17.8
250	20.8
100	32.3
50	43.0

How to pick prototypes?

They do not have to be actual data points!

Idea 2: one prototype per class: **mean** of training points

Examples:



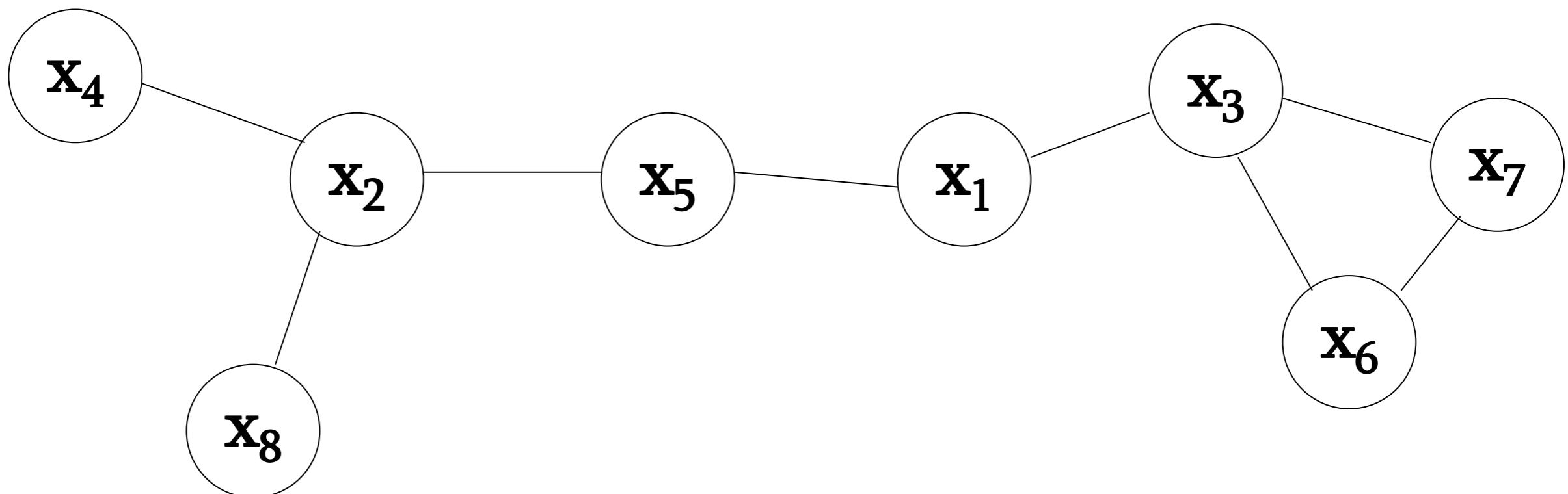
Error = 23%. Number of prototypes =

Picking prototypes

Other methods? e.g. Gain geometric insight from the
neighborhood graph

Node = training point

Edges = connect close neighbors (small distance)



This is an area of machine learning research.

Generalization

Suppose a set of prototypes performs well on the training data.

==> Will it do well on future test data?

What kinds of classifiers generalize well?

Answer: the simplest (most compact) classifiers generalize the best. [cf. Occam's razor]

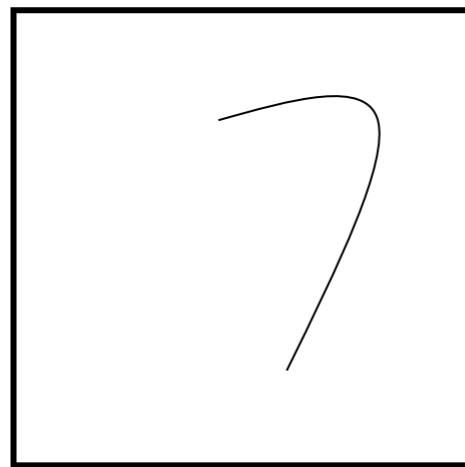
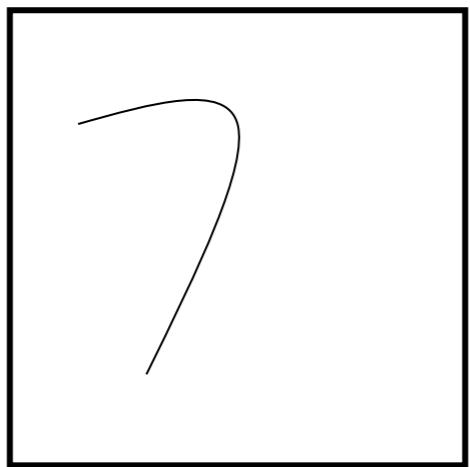
Measure of simplicity:

“Vapnik-Chervonenkis dimension” (later in course)

Postscript: representation I

The data lie in \mathbb{R}^{784} , but our particular choice of Euclidean distance was pretty arbitrary.

Also suboptimal, eg.:



... are pretty far apart! (when computing Euclidean distance between vectors of pixels... Can you see why?)

Is there a better distance measure for this application?

Postscript: representation II

Our 784 features: pixels are not very relevant to this task!

Often data looks like this (figuratively speaking):

digit 5:

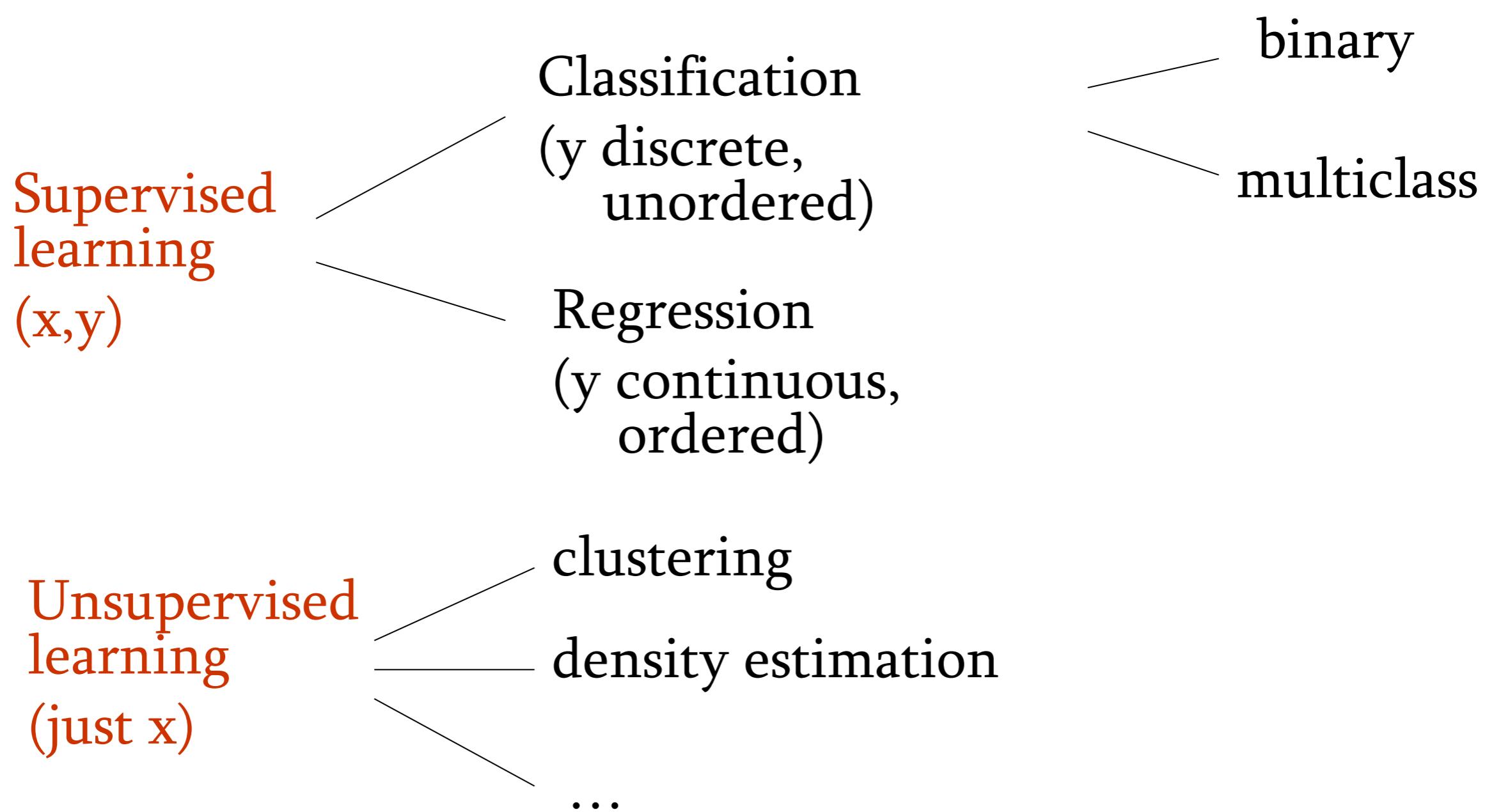
DJHSBSA
VI (FKR5
FLKSAUQ

Information buried in a sea of irrelevant features.

Danger for nearest neighbor: match based on
irrelevant features.

Feature selection is very important.

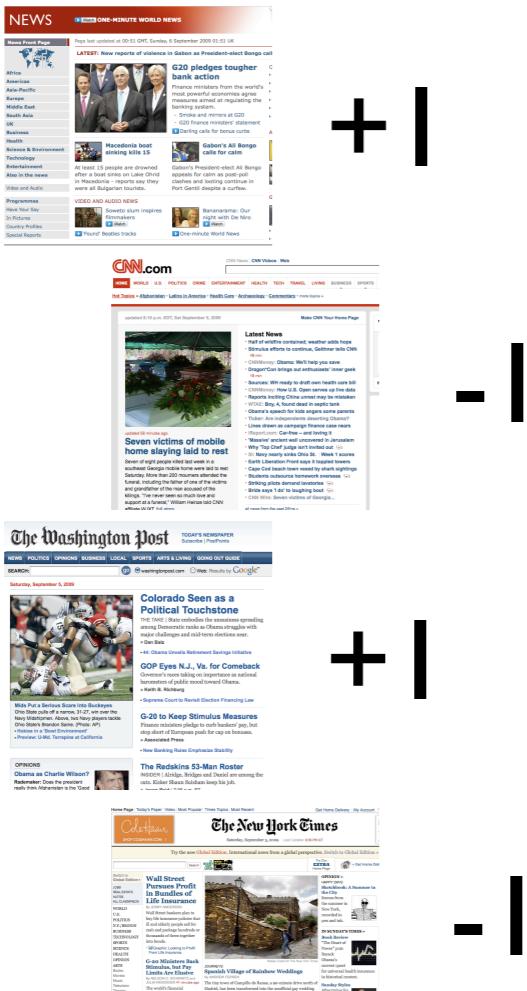
Postscript: learning tasks



All these tasks have a nearest-neighbor solution.

Classification

- Learning from examples



training set

Classification

- Learning from examples



The training set of labeled examples specifies the learning task only implicitly

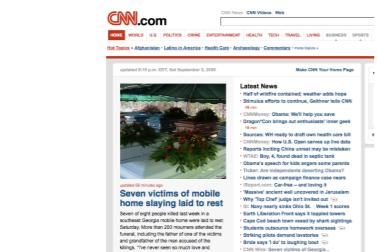
training set

Classification

- Learning from examples



+



-

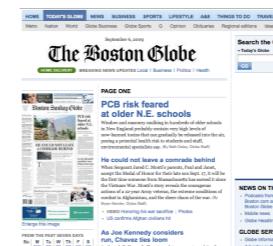


+



-

training set



?

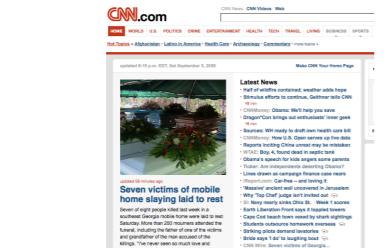
Our goal is to accurately label new websites that were not part of the training set

Classification

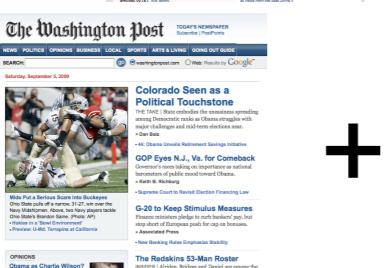
- Learning from examples



+



-



+



-

predicted
label

$$y = \hat{f}(\quad)$$



new website

classifier = mapping
from websites to labels

training set

“Examples”

- We will have to first represent the examples (websites) in a manner that can be easily mapped to labels

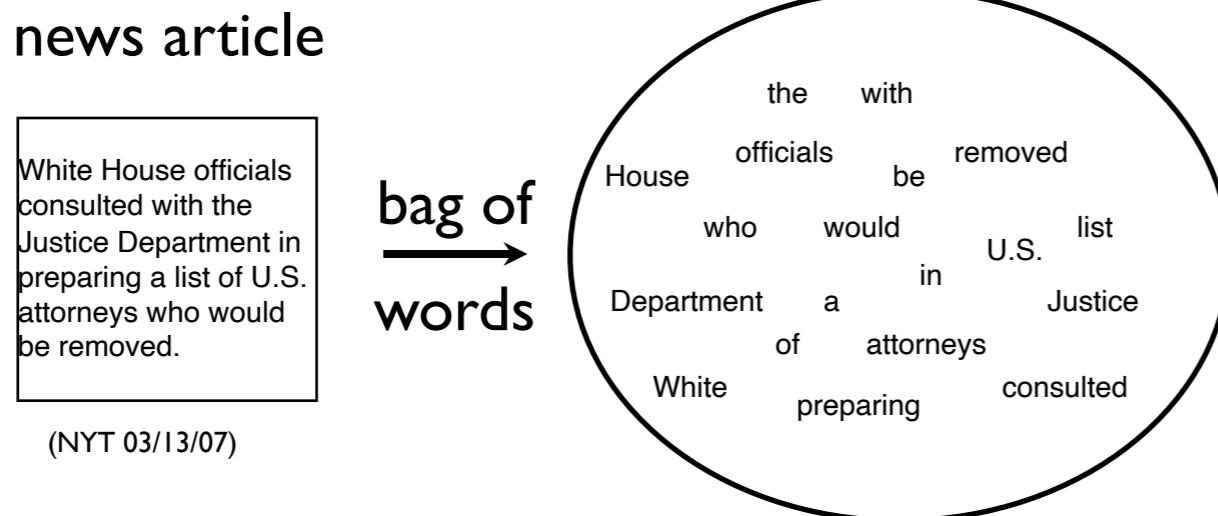
news article

White House officials consulted with the Justice Department in preparing a list of U.S. attorneys who would be removed.

(NYT 03/13/07)

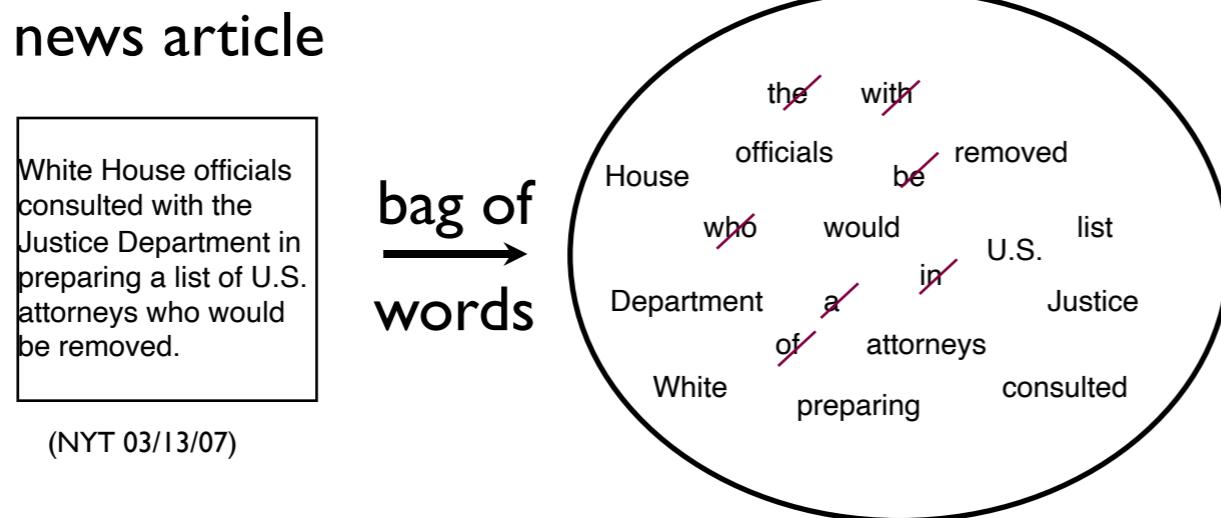
“Examples”

- We will have to first represent the examples (websites) in a manner that can be easily mapped to labels



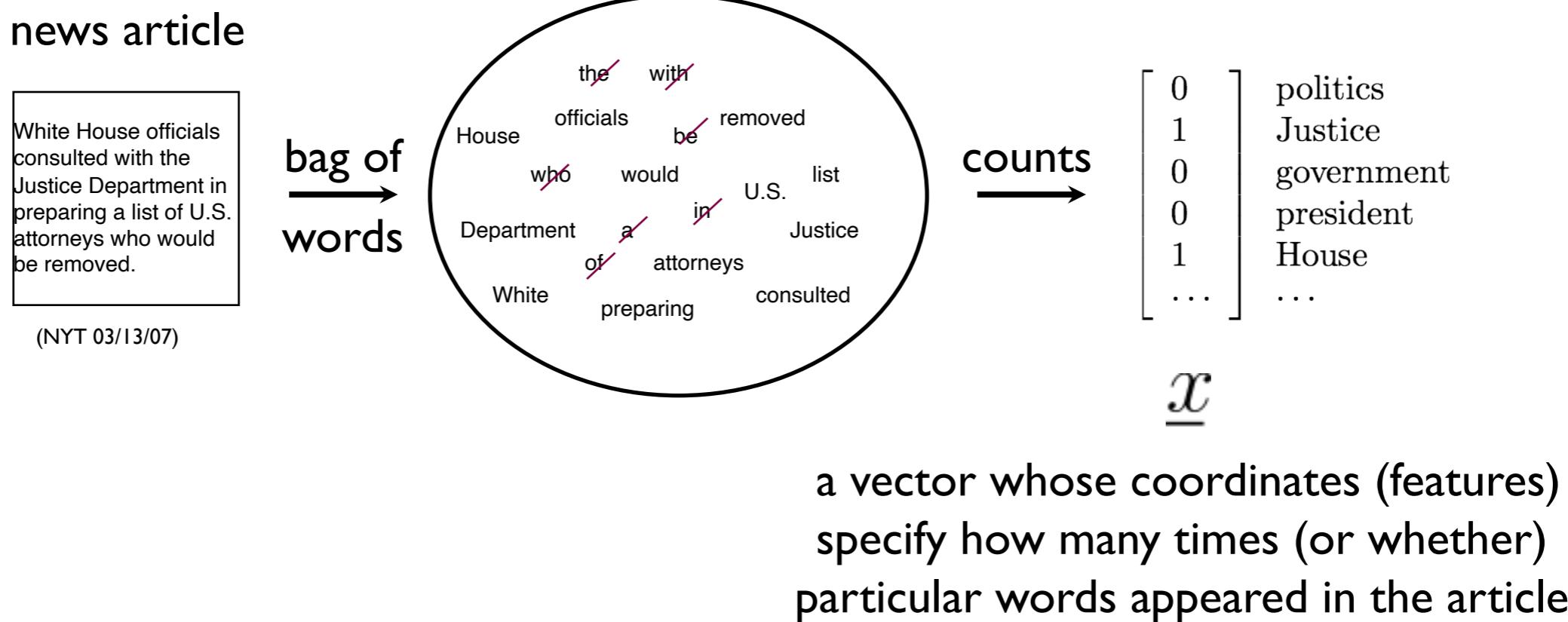
“Examples”

- We will have to first represent the examples (websites) in a manner that can be easily mapped to labels



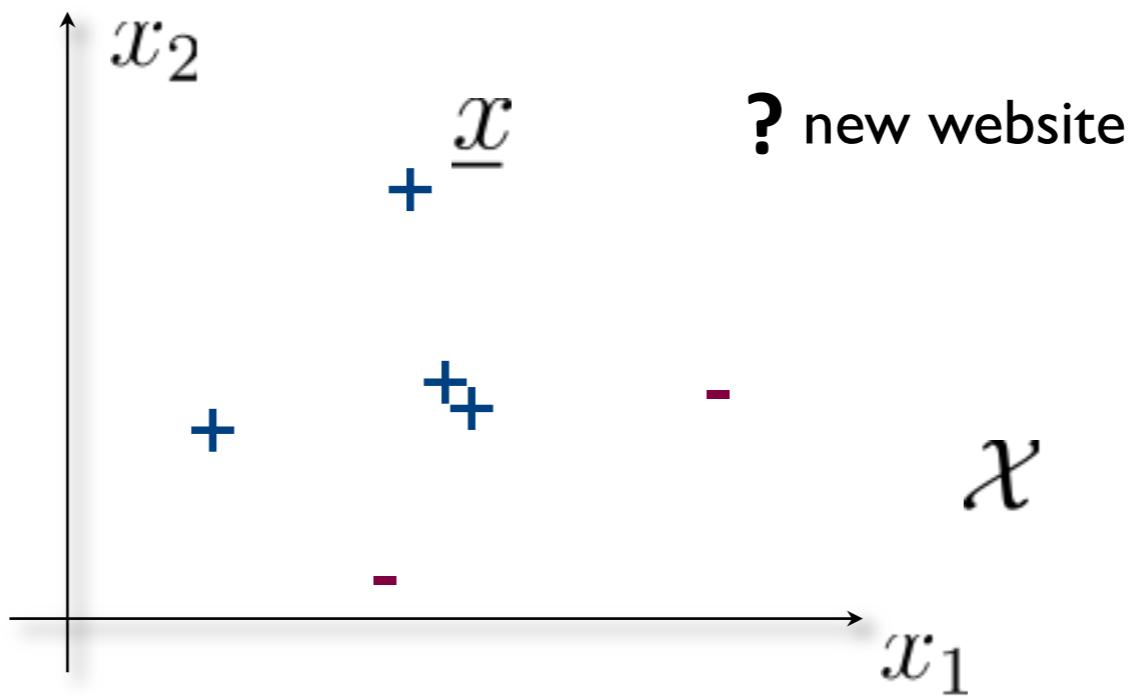
“Examples”

- We will have to first represent the examples (websites) in a manner that can be easily mapped to labels



The learning task

- The training set is now a set of labeled points



Our goal is to find a “good” classifier $f : \mathcal{X} \rightarrow \{-1, 1\}$
based on the training set $D = \{(\underline{x}_i, y_i)_{i=1,\dots,n}\}$
so that $f(\underline{x})$ correctly labels any new websites \underline{x}

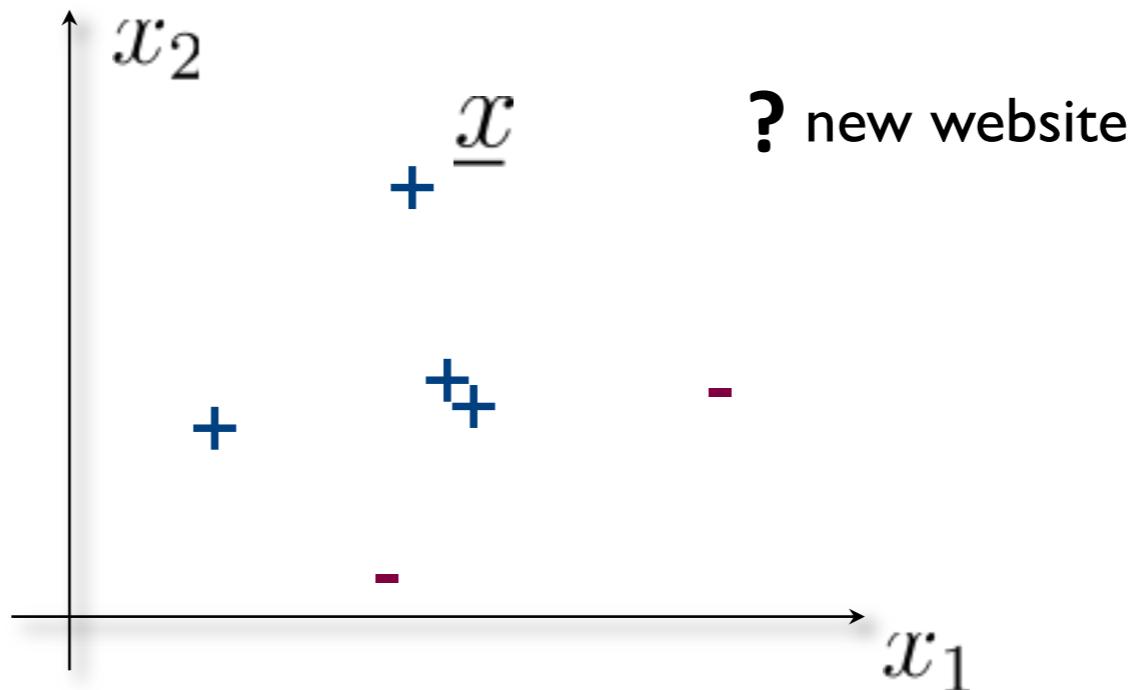
The learning task

- The training set is now a set of labeled points

Part 1:

Model selection

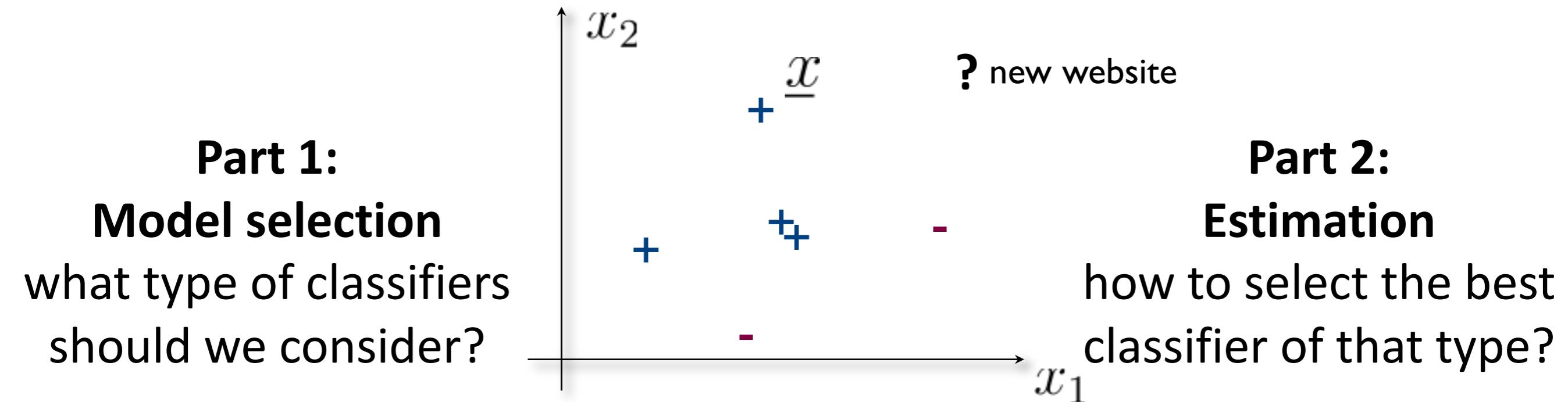
what type of classifiers
should we consider?



Our goal is to find a “good” classifier $f : \mathcal{X} \rightarrow \{-1, 1\}$
based on the training set $D = \{(\underline{x}_i, y_i)_{i=1,\dots,n}\}$
so that $f(\underline{x})$ correctly labels any new websites \underline{x}

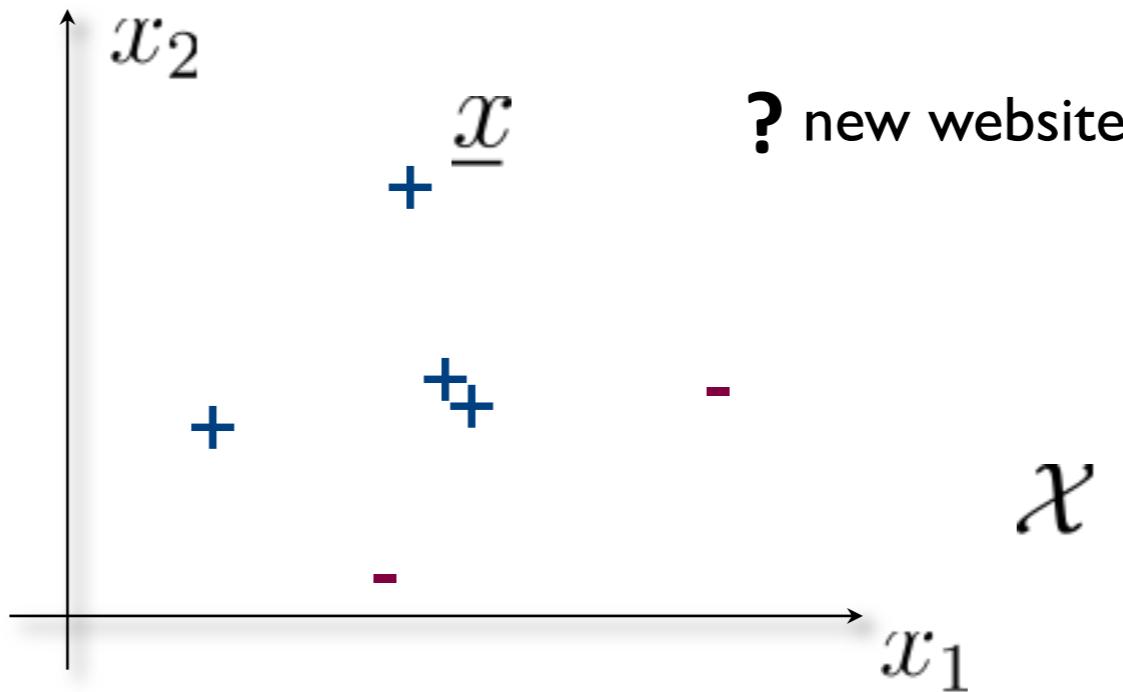
The learning task

- The training set is now a set of labeled points



Our goal is to find a “good” classifier $f : \mathcal{X} \rightarrow \{-1, 1\}$
based on the training set $D = \{(\underline{x}_i, y_i)_{i=1,\dots,n}\}$
so that $f(\underline{x})$ correctly labels any new websites \underline{x}

Part 1: allowing few classifiers?



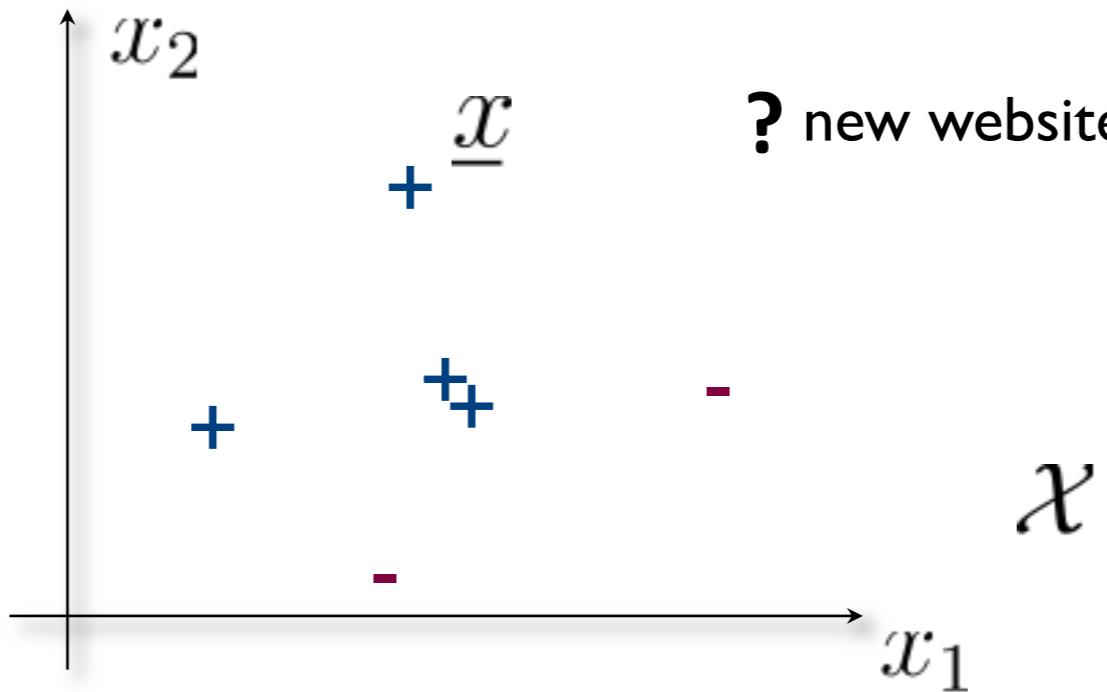
What about classifiers such as:

$$f(\underline{x}) = 1, \text{ for all } \underline{x} \in \mathcal{X}, \text{ or}$$

$$f(\underline{x}) = -1, \text{ for all } \underline{x} \in \mathcal{X},$$

But neither one classifies even the training points well!

Part 1: allowing all classifiers?



We can easily construct a “silly classifier” that perfectly classifies any distinct set of training points

$$f(\underline{x}) = \begin{cases} y_i, & \text{if } \underline{x} = \underline{x}_i \text{ for some } i \\ -1, & \text{otherwise} \end{cases}$$

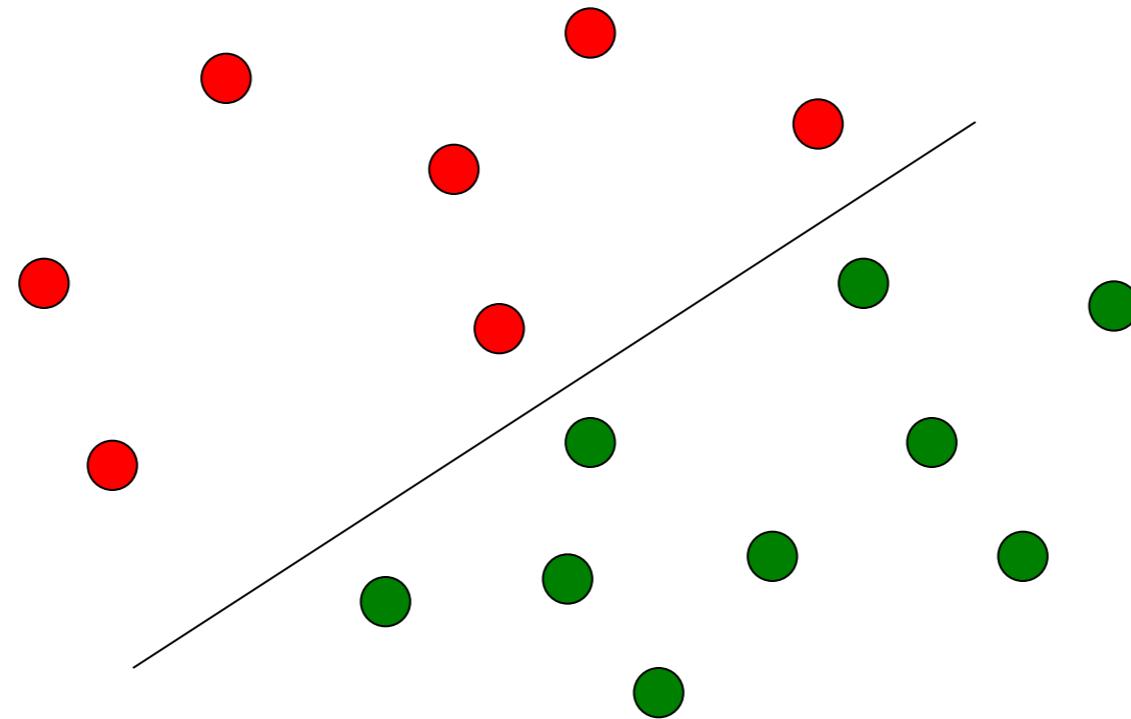
But it doesn’t “generalize” (it doesn’t classify new points well)

Linear classification

Given labeled data points, find a linear separator.

Describes the data

Generalizes well



Line in 2-d
Plane in 3-d
Hyperplane in n -d

Linear separator vs. nearest neighbor

Linear separators

parametric model

fixed number of parameters to learn from the data

Nearest neighbor

nonparametric

prediction on test point x depends only on training data *near* x , not on the rest of the training data

Advantages of linear separators:

compact

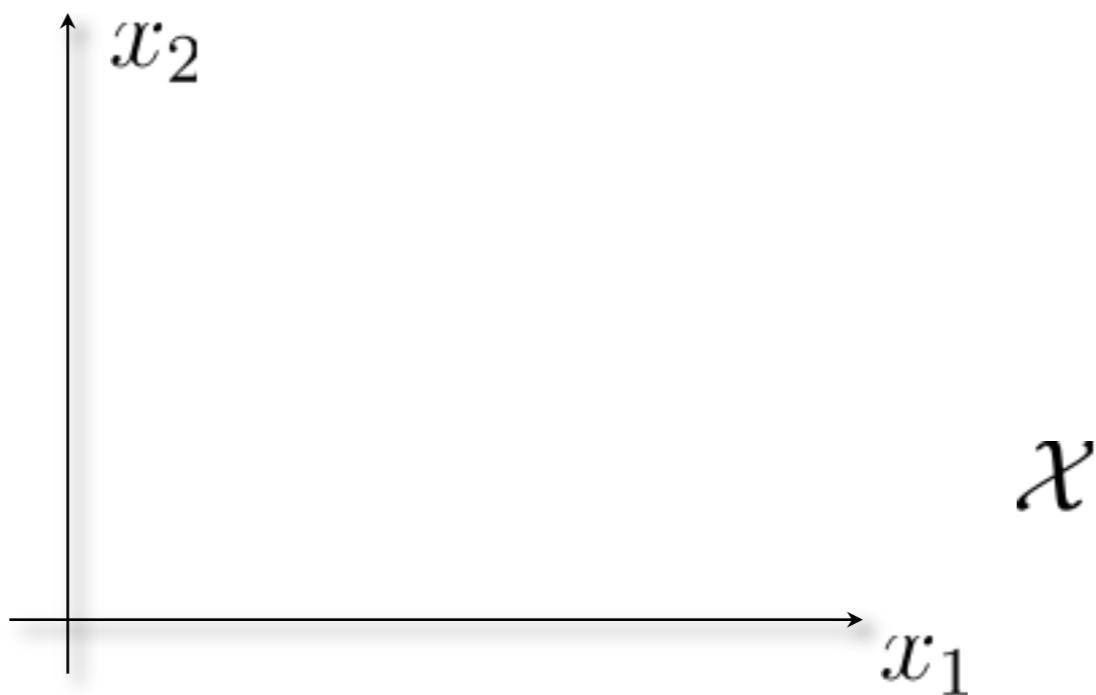
fast convergence (of error, w.r.t. number of examples)

potentially meaningful (interpretability)

Linear classifiers

- A linear classifier (through origin) with parameters $\underline{\theta}$ divides the space into positive and negative halves

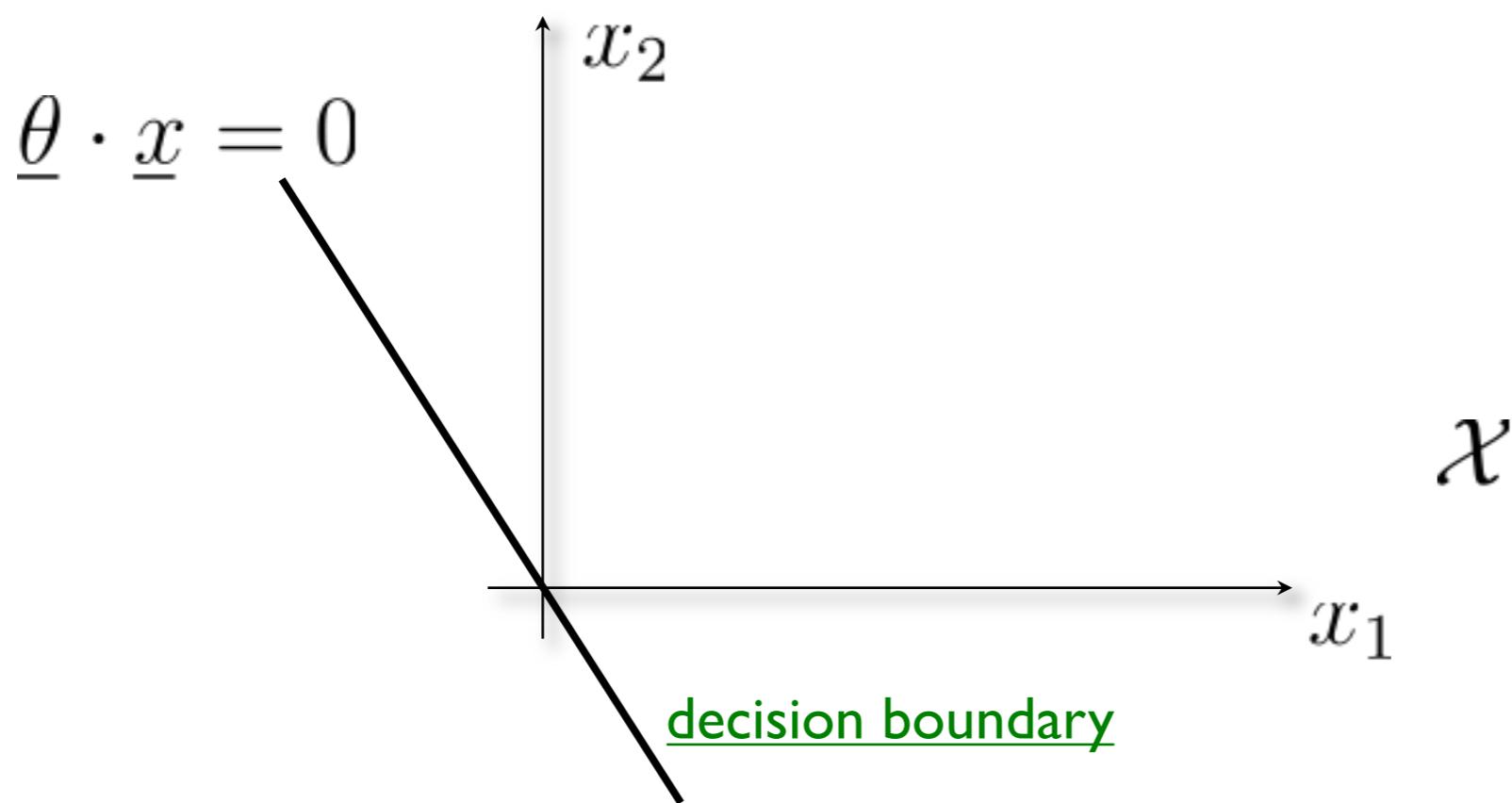
$$\begin{aligned} f(\underline{x}; \underline{\theta}) &= \text{sign}(\underline{\theta} \cdot \underline{x}) = \text{sign}(\theta_1 x_1 + \dots + \theta_d x_d) \\ &= \begin{cases} +1, & \text{if } \underline{\theta} \cdot \underline{x} > 0 \\ -1, & \text{if } \underline{\theta} \cdot \underline{x} \leq 0 \end{cases} \end{aligned}$$



Linear classifiers

- A linear classifier (through origin) with parameters $\underline{\theta}$ divides the space into positive and negative halves

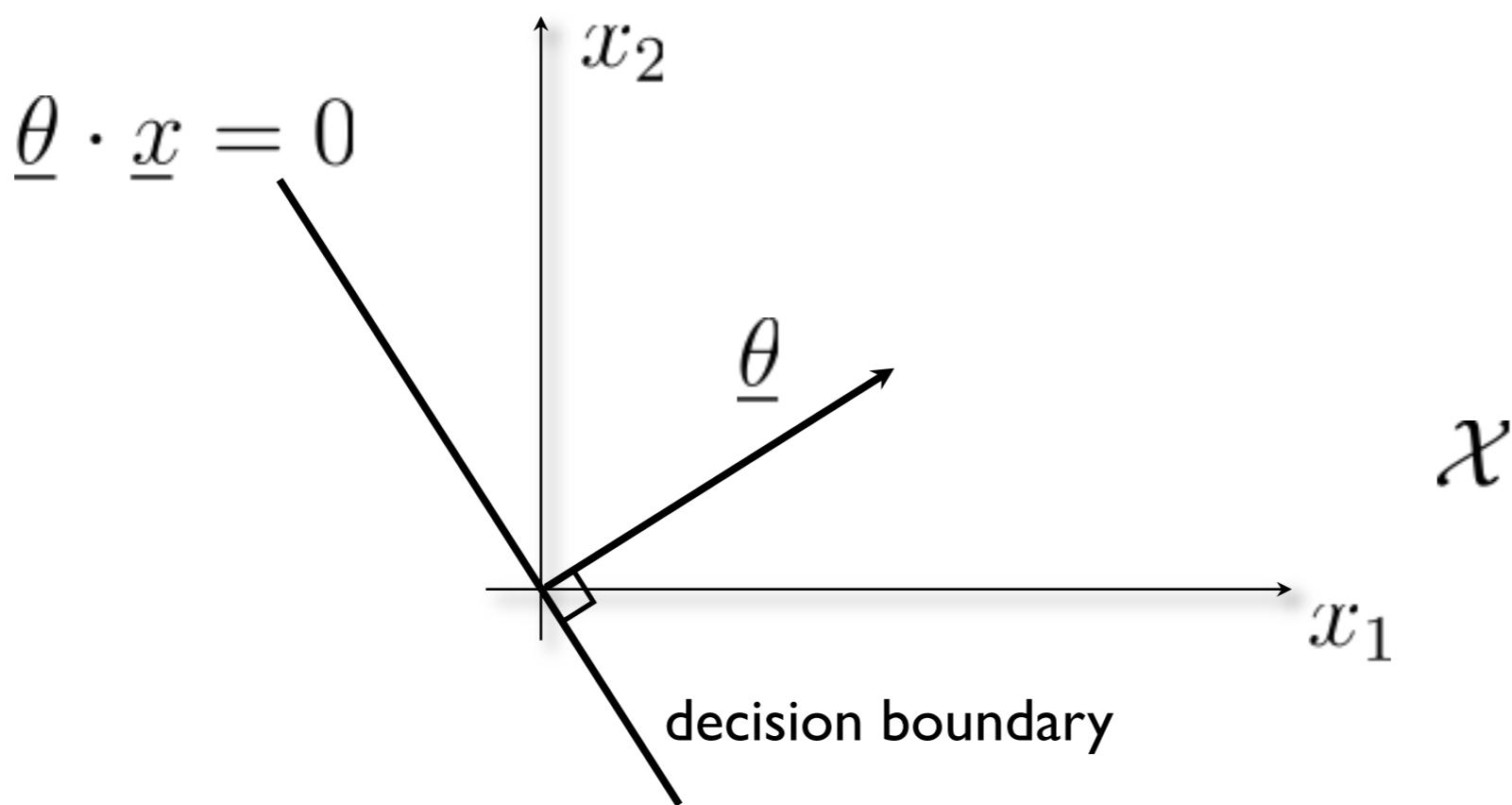
$$\begin{aligned} f(\underline{x}; \underline{\theta}) &= \text{sign}(\underline{\theta} \cdot \underline{x}) = \text{sign}(\theta_1 x_1 + \dots + \theta_d x_d) \\ &= \begin{cases} +1, & \text{if } \underline{\theta} \cdot \underline{x} > 0 \\ -1, & \text{if } \underline{\theta} \cdot \underline{x} \leq 0 \end{cases} \end{aligned}$$



Linear classifiers

- A linear classifier (through origin) with parameters $\underline{\theta}$ divides the space into positive and negative halves

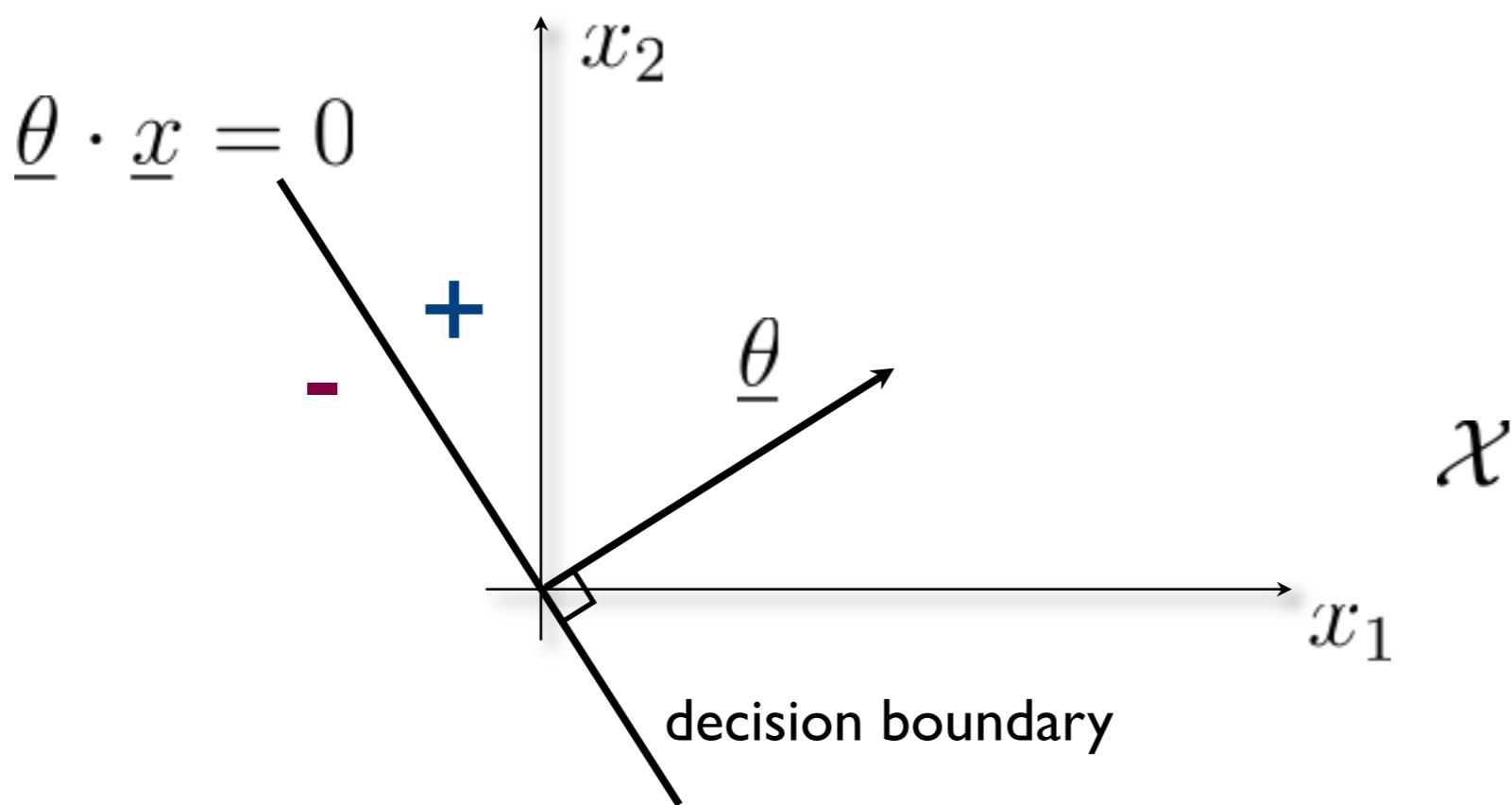
$$\begin{aligned} f(\underline{x}; \underline{\theta}) &= \text{sign}(\underline{\theta} \cdot \underline{x}) = \text{sign}(\theta_1 x_1 + \dots + \theta_d x_d) \\ &= \begin{cases} +1, & \text{if } \underline{\theta} \cdot \underline{x} > 0 \\ -1, & \text{if } \underline{\theta} \cdot \underline{x} \leq 0 \end{cases} \end{aligned}$$



Linear classifiers

- A linear classifier (through origin) with parameters $\underline{\theta}$ divides the space into positive and negative halves

$$\begin{aligned} f(\underline{x}; \underline{\theta}) &= \text{sign}(\underline{\theta} \cdot \underline{x}) = \text{sign}(\theta_1 x_1 + \dots + \theta_d x_d) \\ &= \begin{cases} +1, & \text{if } \underline{\theta} \cdot \underline{x} > 0 \\ -1, & \text{if } \underline{\theta} \cdot \underline{x} \leq 0 \end{cases} \end{aligned}$$



(1958)
F. Rosenblatt

The perceptron: a probabilistic model
for information storage and organization in the brain
Psychological Review 65: 386-408

If we are eventually to understand the capability of higher organisms for perceptual recognition, generalization, recall, and thinking, we must first have answers to three fundamental questions:

1. How is information about the physical world sensed, or detected, by the biological system?
2. In what form is information stored, or remembered?
3. How does information contained in storage, or in memory, influence recognition and behavior?

The first of these questions is in the province of sensory physiology, and is the only one for which appreciable understanding has been achieved. This article will be concerned primarily with the second and third questions, which are still subject to a vast amount of speculation, and where the few relevant facts currently supplied by neurophysiology have not yet been integrated into an acceptable theory.

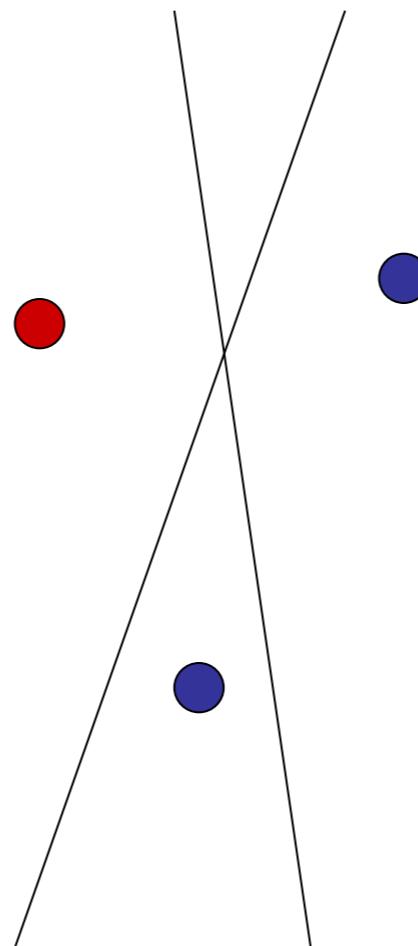
and the stored pattern. According to this hypothesis, if one understood the code or "wiring diagram" of the nervous system, one should, in principle, be able to discover exactly what an organism remembers by reconstructing the original sensory patterns from the "memory traces" which they have left, much as we might develop a photographic negative, or translate the pattern of electrical charges in the "memory" of a digital computer. This hypothesis is appealing in its simplicity and ready intelligibility, and a large family of theoretical brain models has been developed around the idea of a coded, representational memory (2, 3, 9, 14). The alternative approach, which stems from the tradition of British empiricism, hazards the guess that the images of stimuli may never really be recorded at all, and that the central nervous system simply acts as an intricate switching network, where retention takes the form of new connections, or pathways, between centers of activity. In many of the more recent developments of

Learning from one data point at a time

Instead of receiving a **batch** of data ahead of time,
Learner receives **one data point** at a time.

Can also cycle through a batch of data this way.

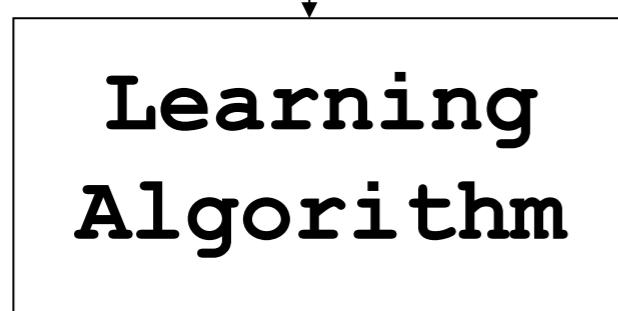
Learn a linear classifier:



Batch vs. Online Learning

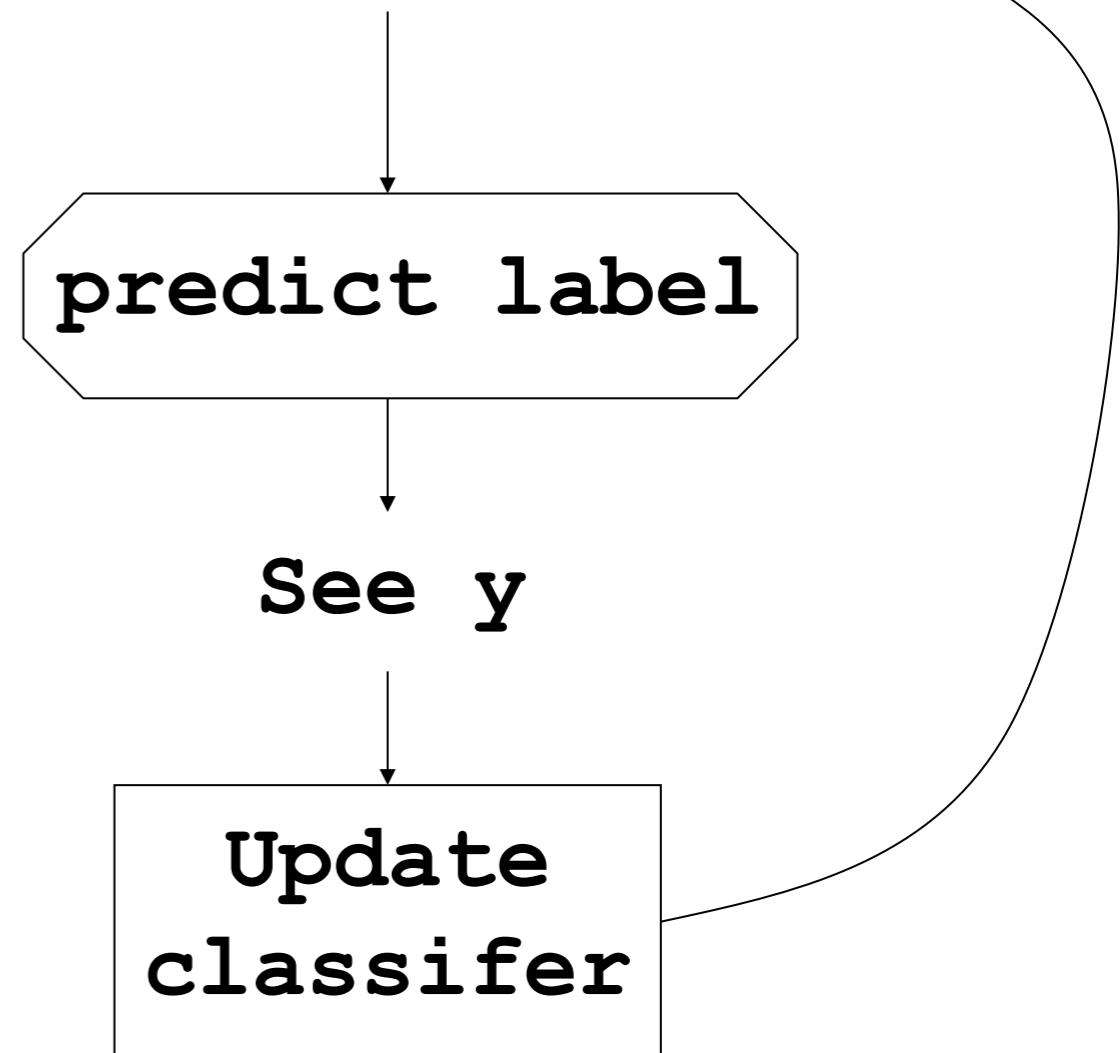
Batch learning

Training data



Online learning

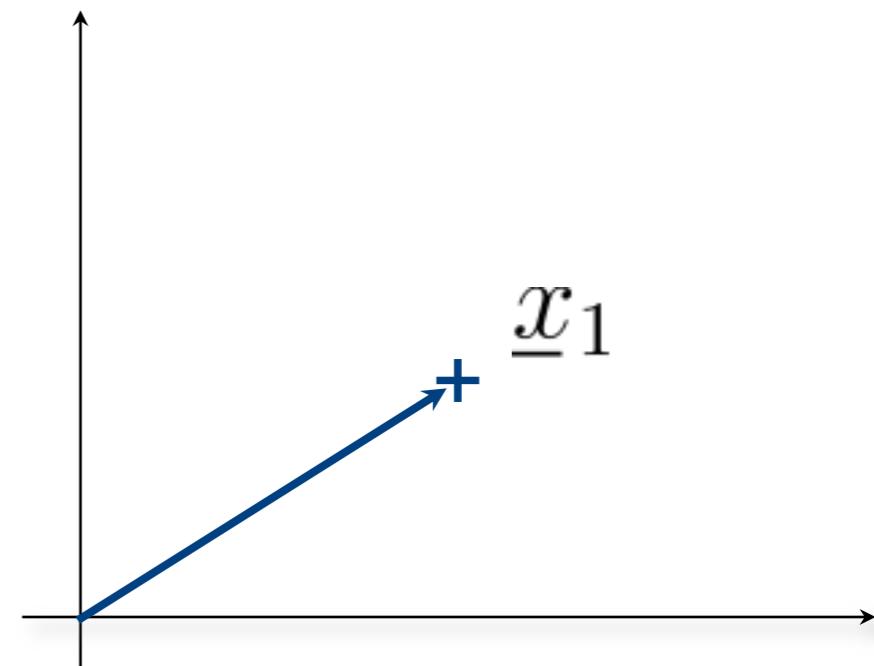
See a new point $x_$



Perceptron algorithm

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$

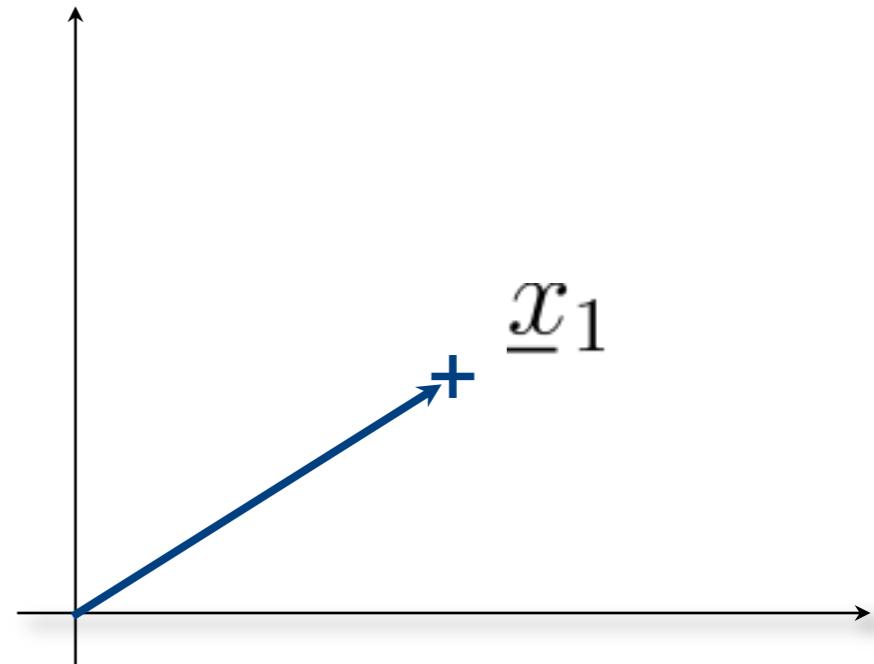


Perceptron algorithm

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$

$$\underline{\theta}_1 = \underline{\theta}_0 + 1 \underline{x}_1$$

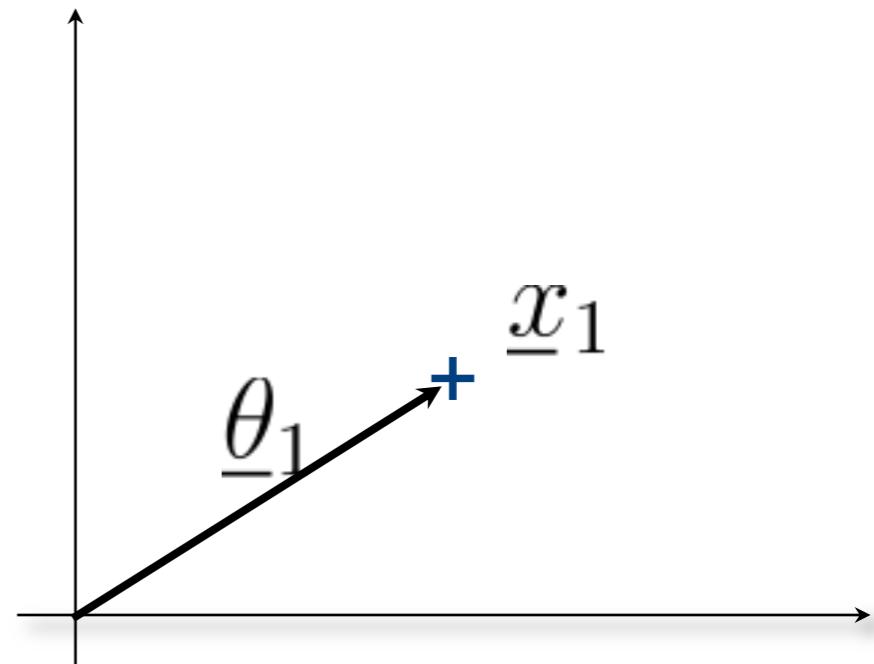


Perceptron algorithm

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$

$$\underline{\theta}_1 = \underline{\theta}_0 + 1 \underline{x}_1$$

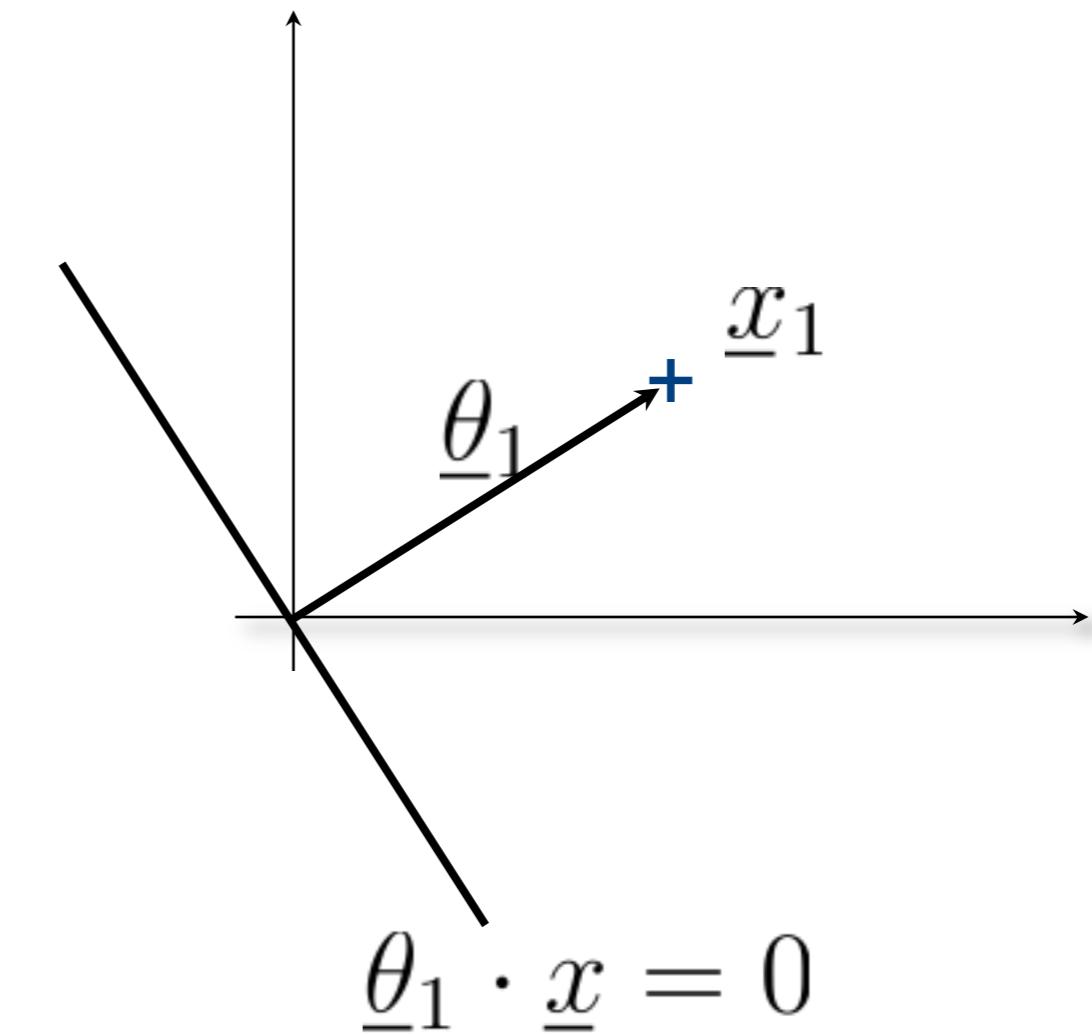


Perceptron algorithm

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$

$$\underline{\theta}_1 = \underline{\theta}_0 + 1 \underline{x}_1$$

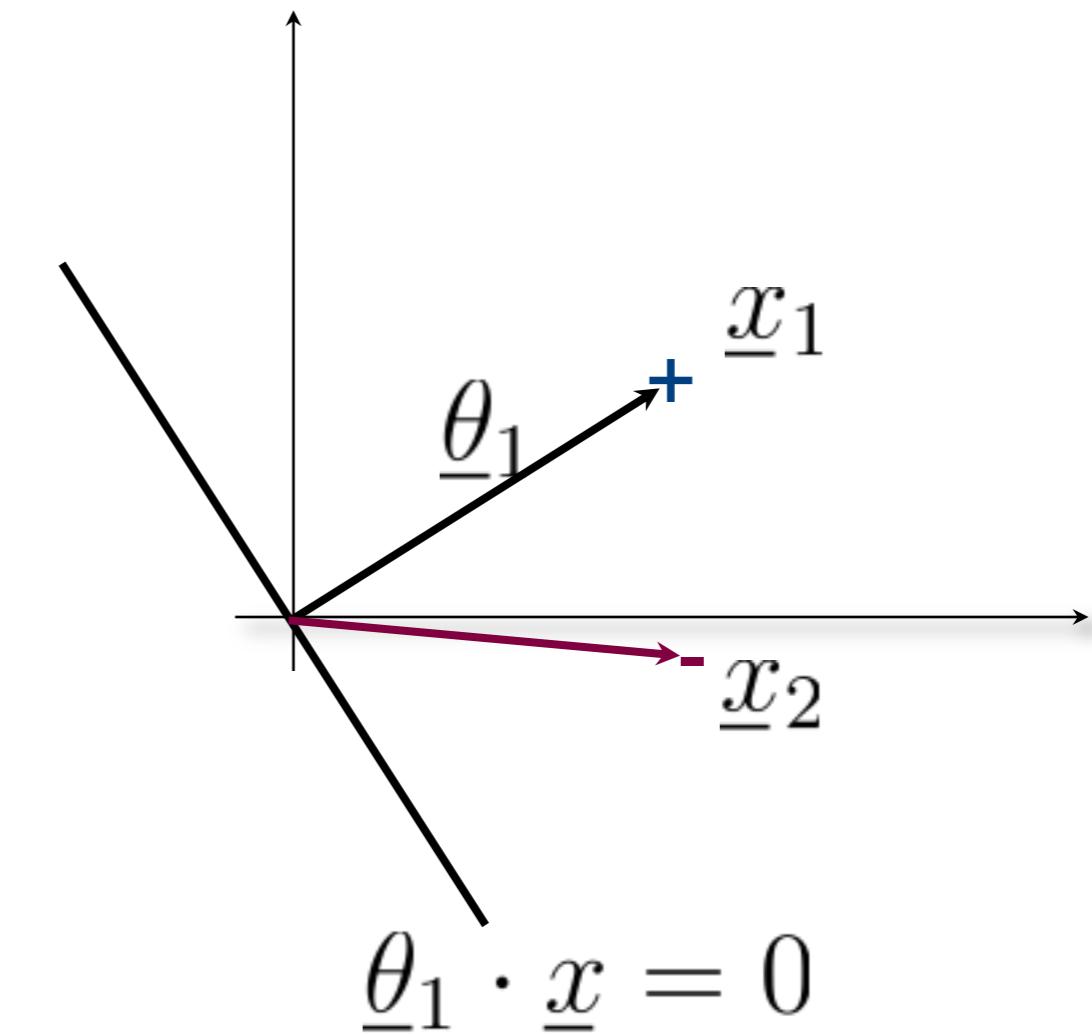


Perceptron algorithm

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$

$$\underline{\theta}_1 = \underline{\theta}_0 + 1 \underline{x}_1$$



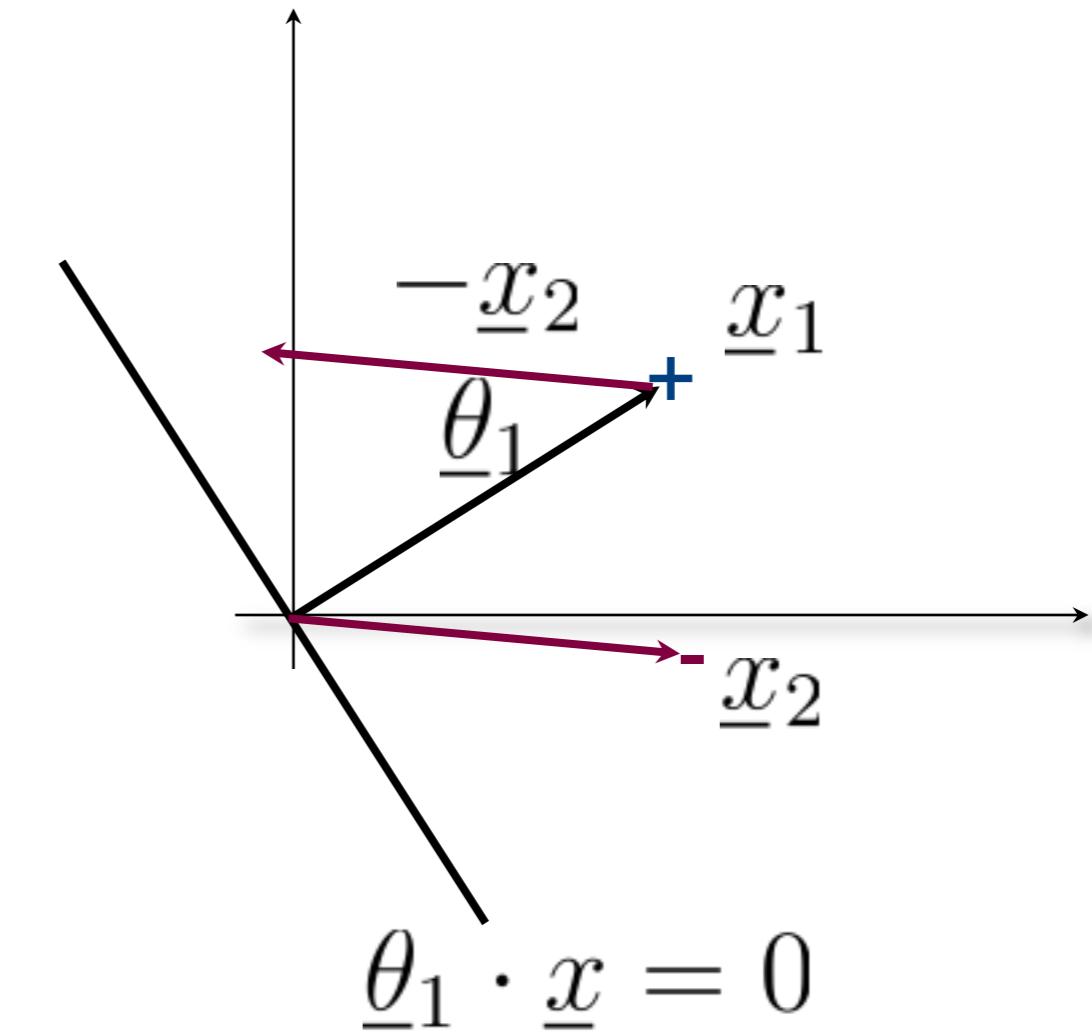
Perceptron algorithm

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$

$$\underline{\theta}_1 = \underline{\theta}_0 + 1 \underline{x}_1$$

$$\underline{\theta}_2 = \underline{\theta}_1 + (-1) \underline{x}_2$$



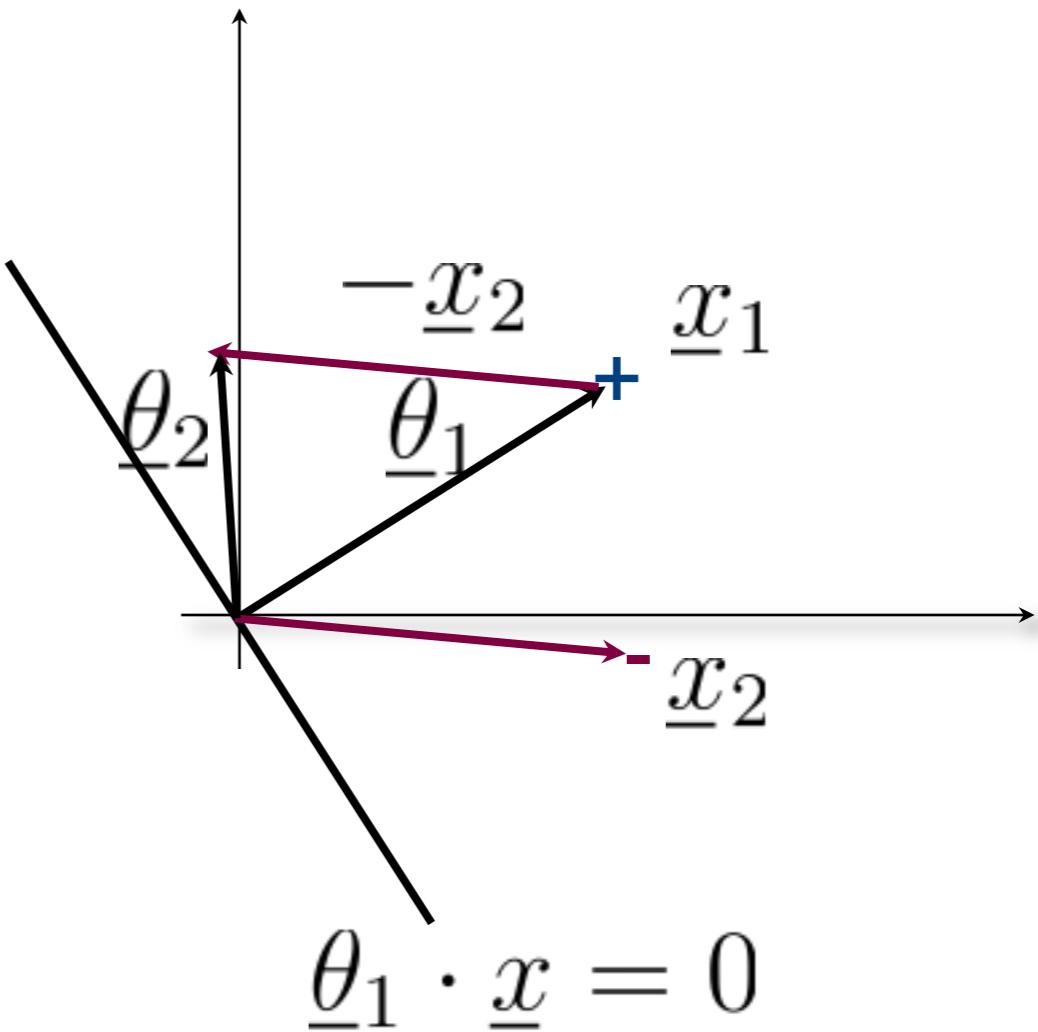
Perceptron algorithm

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$

$$\underline{\theta}_1 = \underline{\theta}_0 + 1 \underline{x}_1$$

$$\underline{\theta}_2 = \underline{\theta}_1 + (-1) \underline{x}_2$$



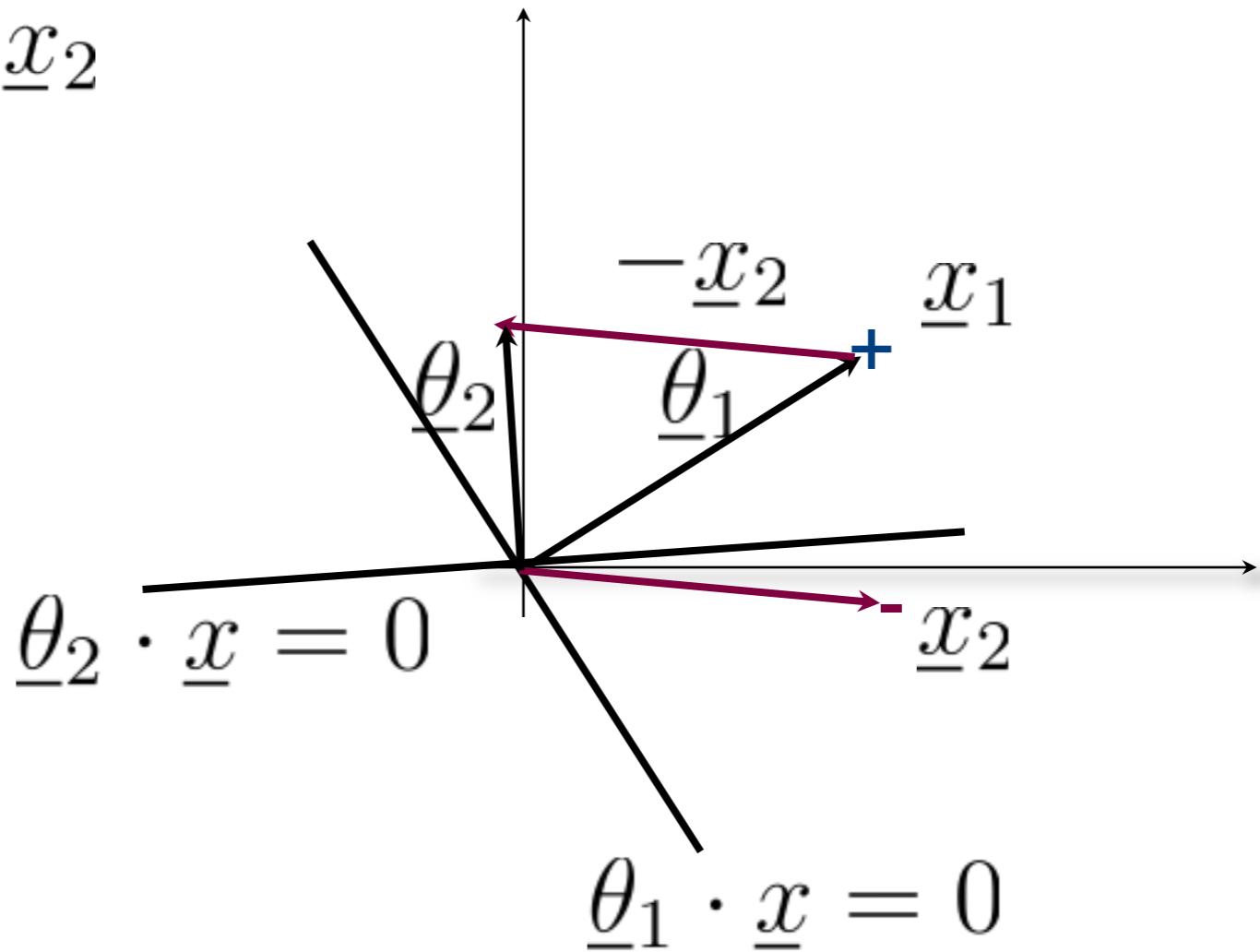
Perceptron algorithm

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$

$$\underline{\theta}_1 = \underline{\theta}_0 + 1 \underline{x}_1$$

$$\underline{\theta}_2 = \underline{\theta}_1 + (-1) \underline{x}_2$$



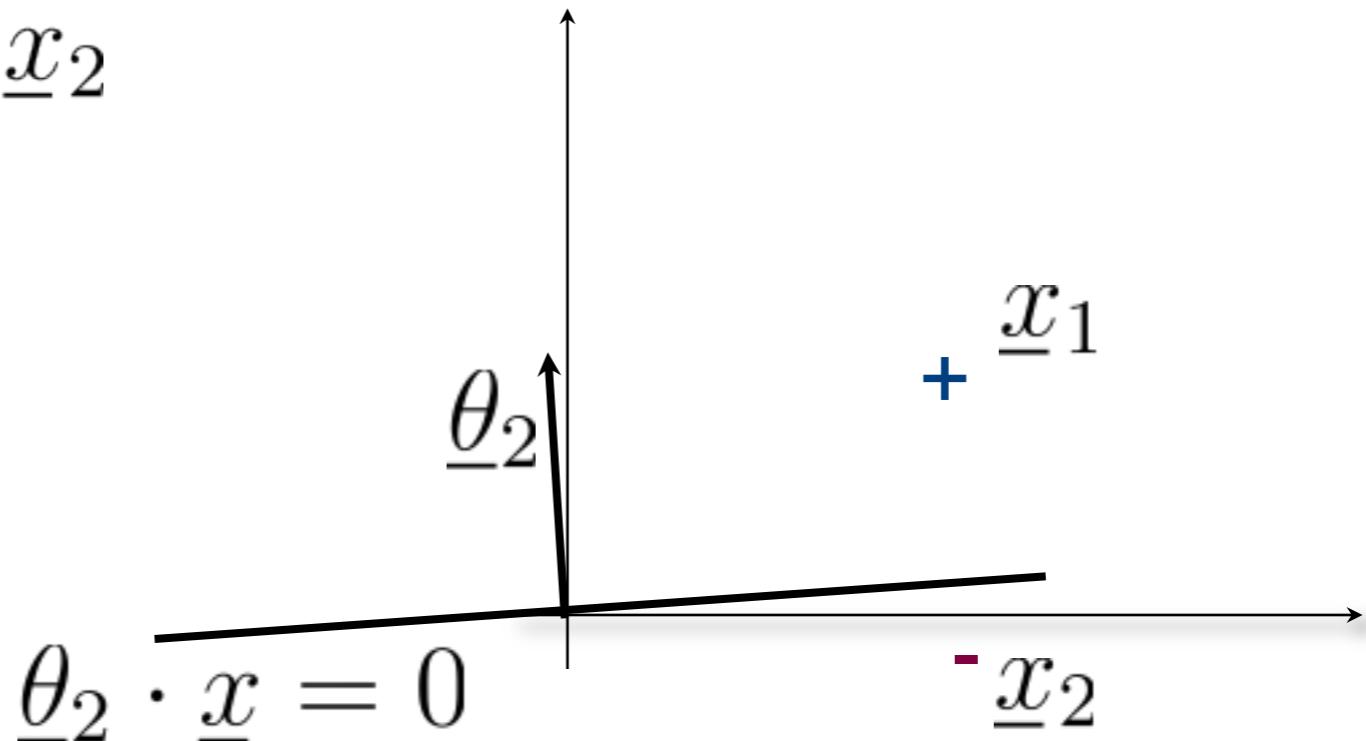
Perceptron algorithm

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$

$$\underline{\theta}_1 = \underline{\theta}_0 + 1 \underline{x}_1$$

$$\underline{\theta}_2 = \underline{\theta}_1 + (-1) \underline{x}_2$$

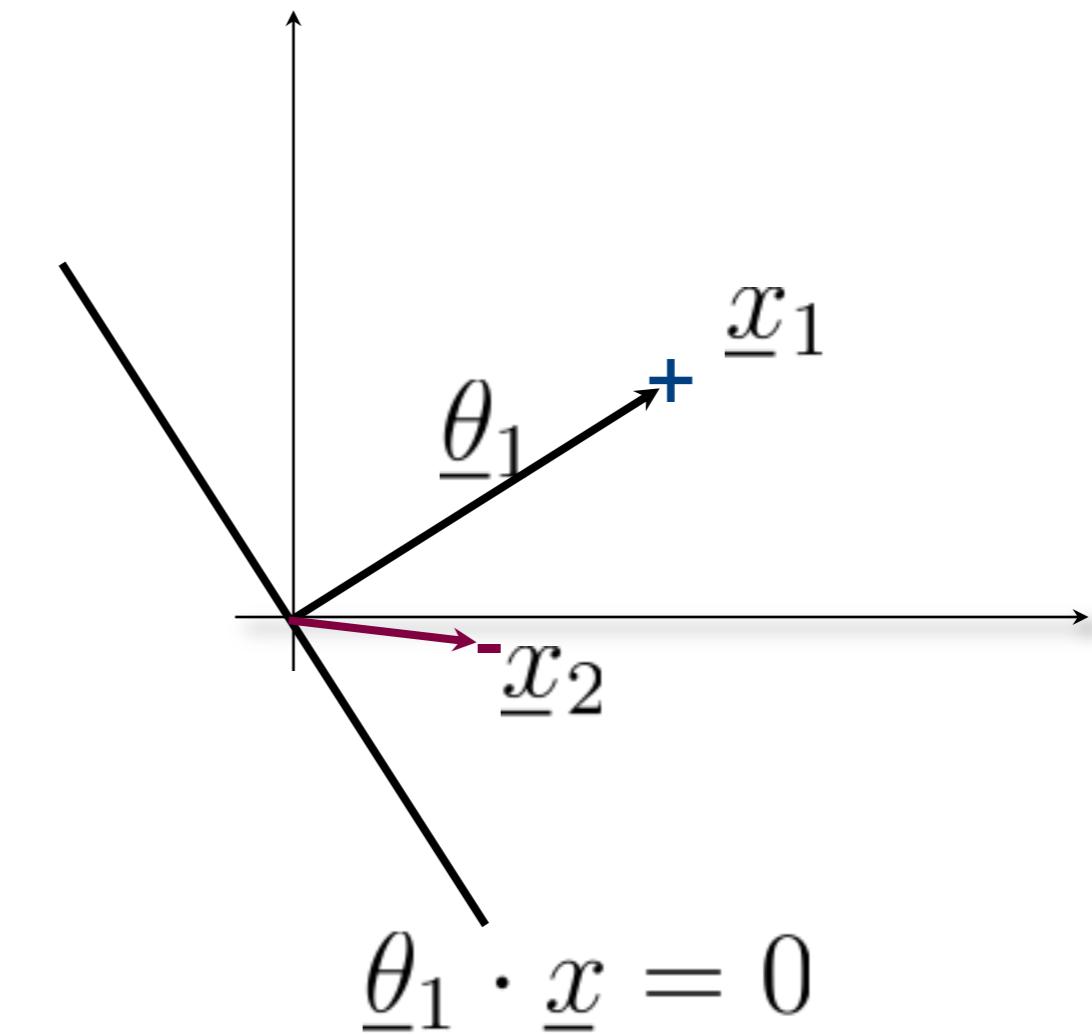


Perceptron algorithm (take 2)

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$

$$\underline{\theta}_1 = \underline{\theta}_0 + 1 \underline{x}_1$$



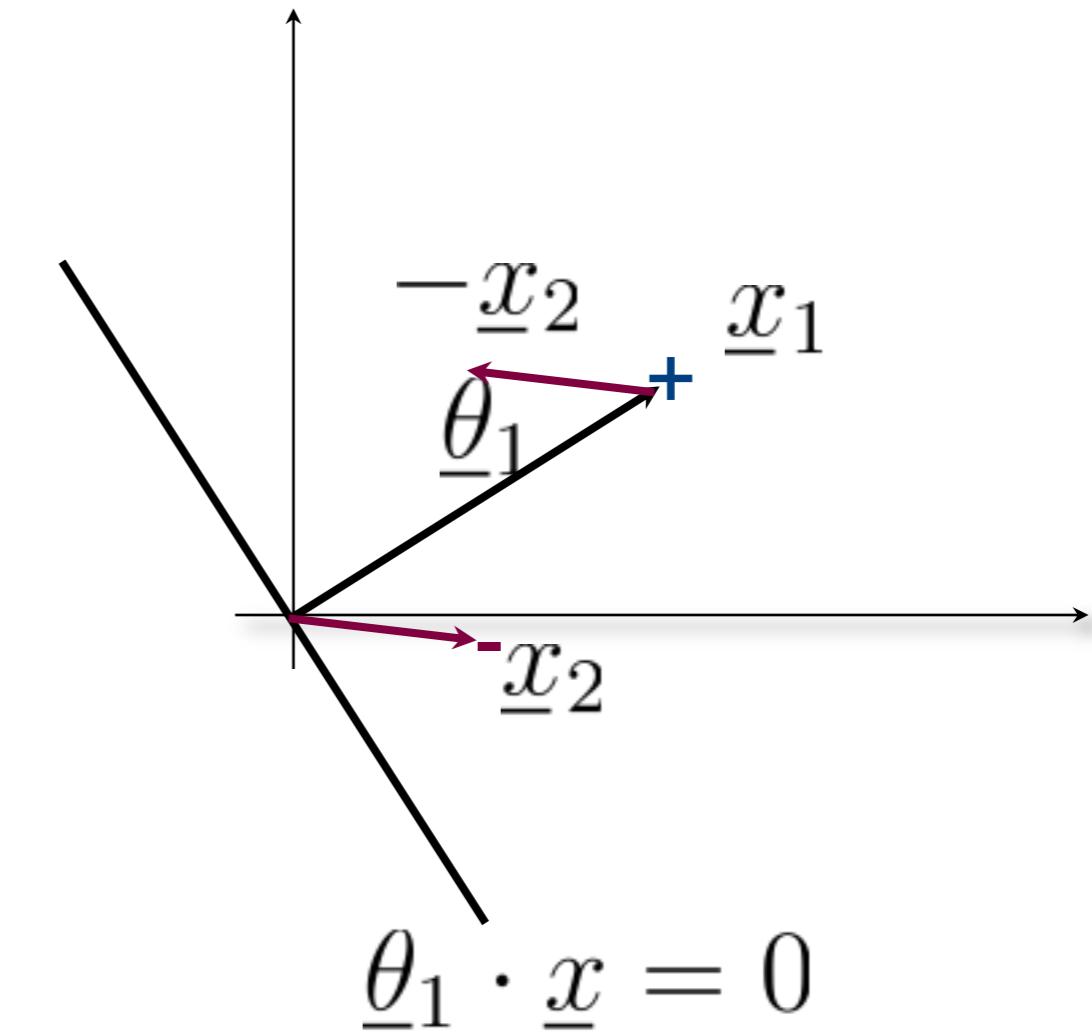
Perceptron algorithm (take 2)

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$

$$\underline{\theta}_1 = \underline{\theta}_0 + 1 \underline{x}_1$$

$$\underline{\theta}_2 = \underline{\theta}_1 + (-1) \underline{x}_2$$



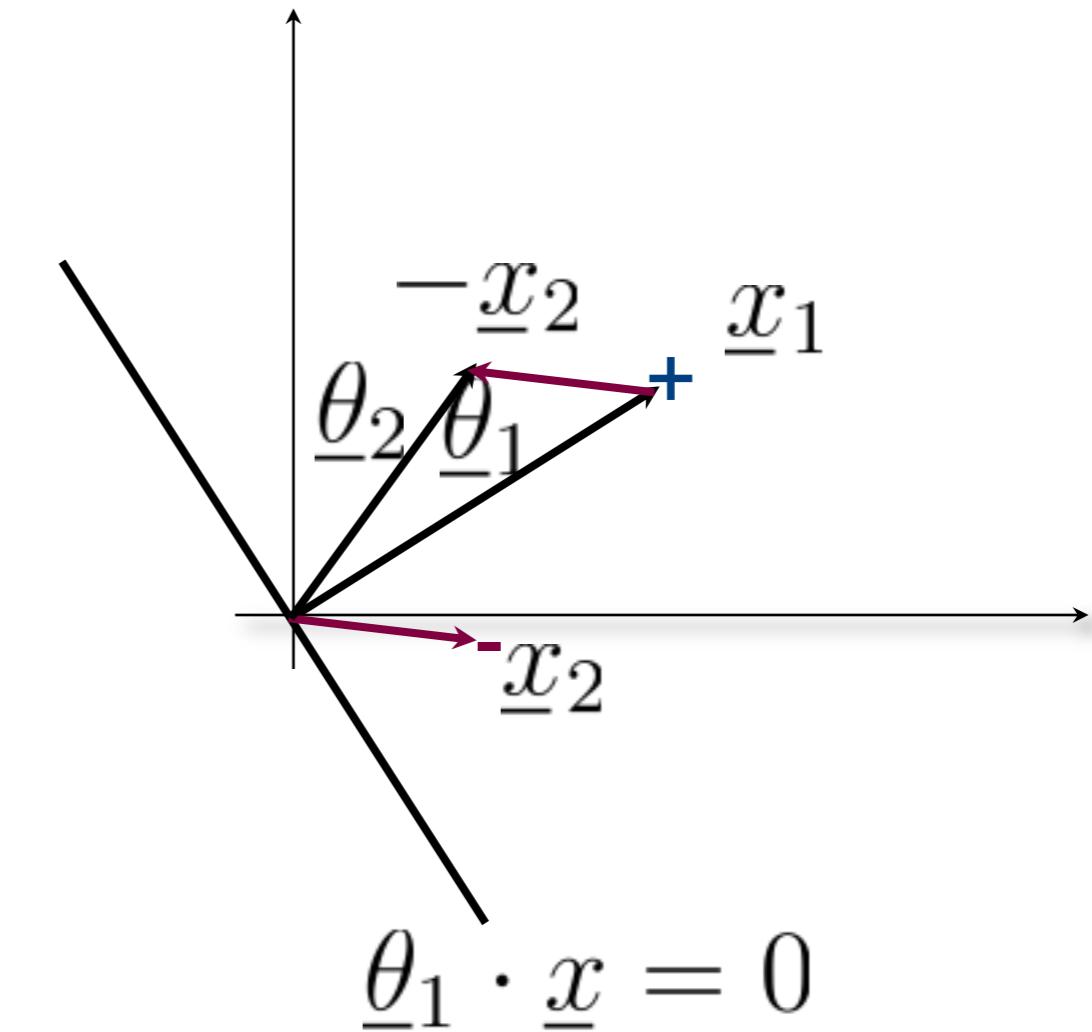
Perceptron algorithm (take 2)

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$

$$\underline{\theta}_1 = \underline{\theta}_0 + 1 \underline{x}_1$$

$$\underline{\theta}_2 = \underline{\theta}_1 + (-1) \underline{x}_2$$



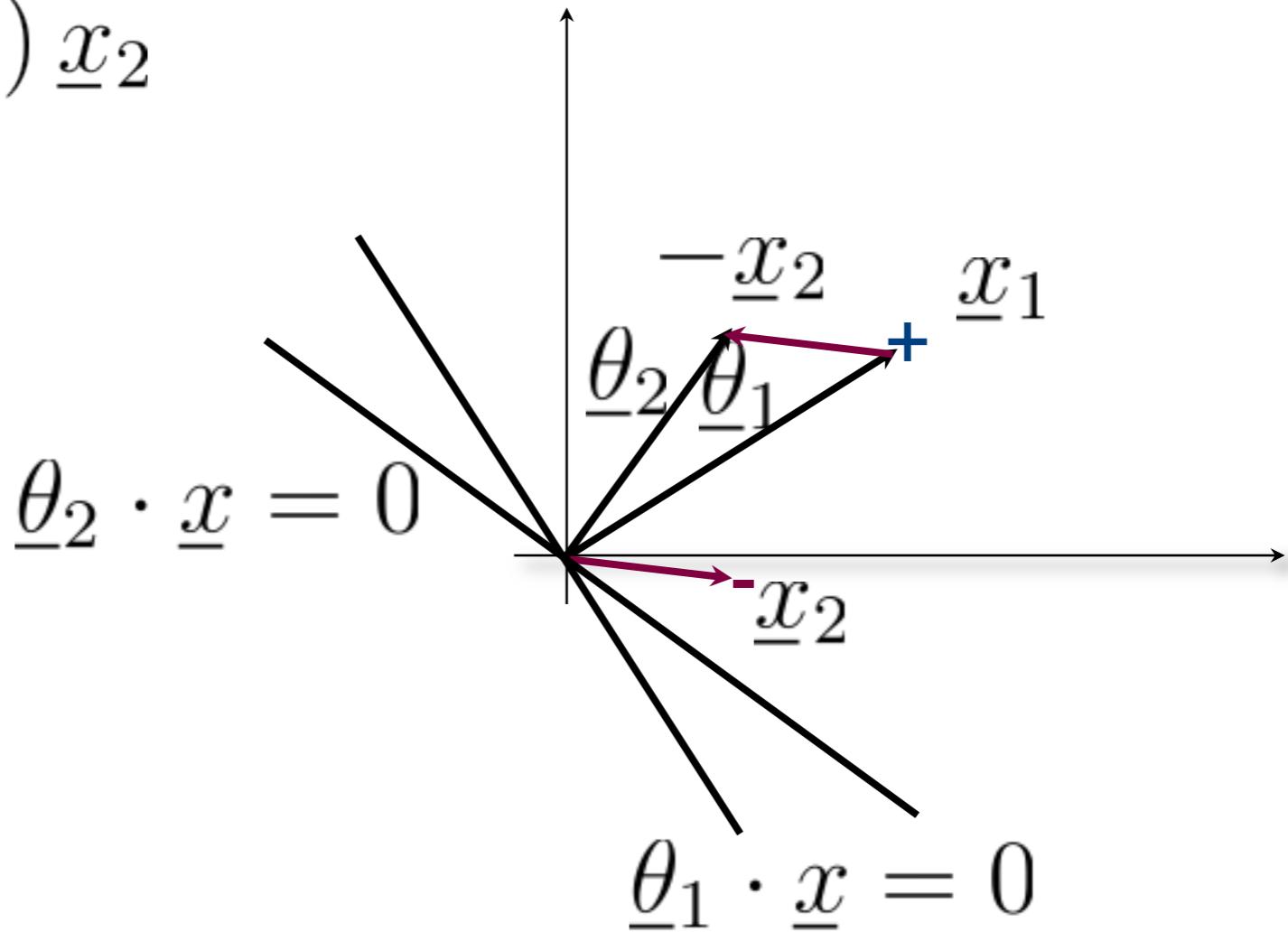
Perceptron algorithm (take 2)

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$

$$\underline{\theta}_1 = \underline{\theta}_0 + 1 \underline{x}_1$$

$$\underline{\theta}_2 = \underline{\theta}_1 + (-1) \underline{x}_2$$



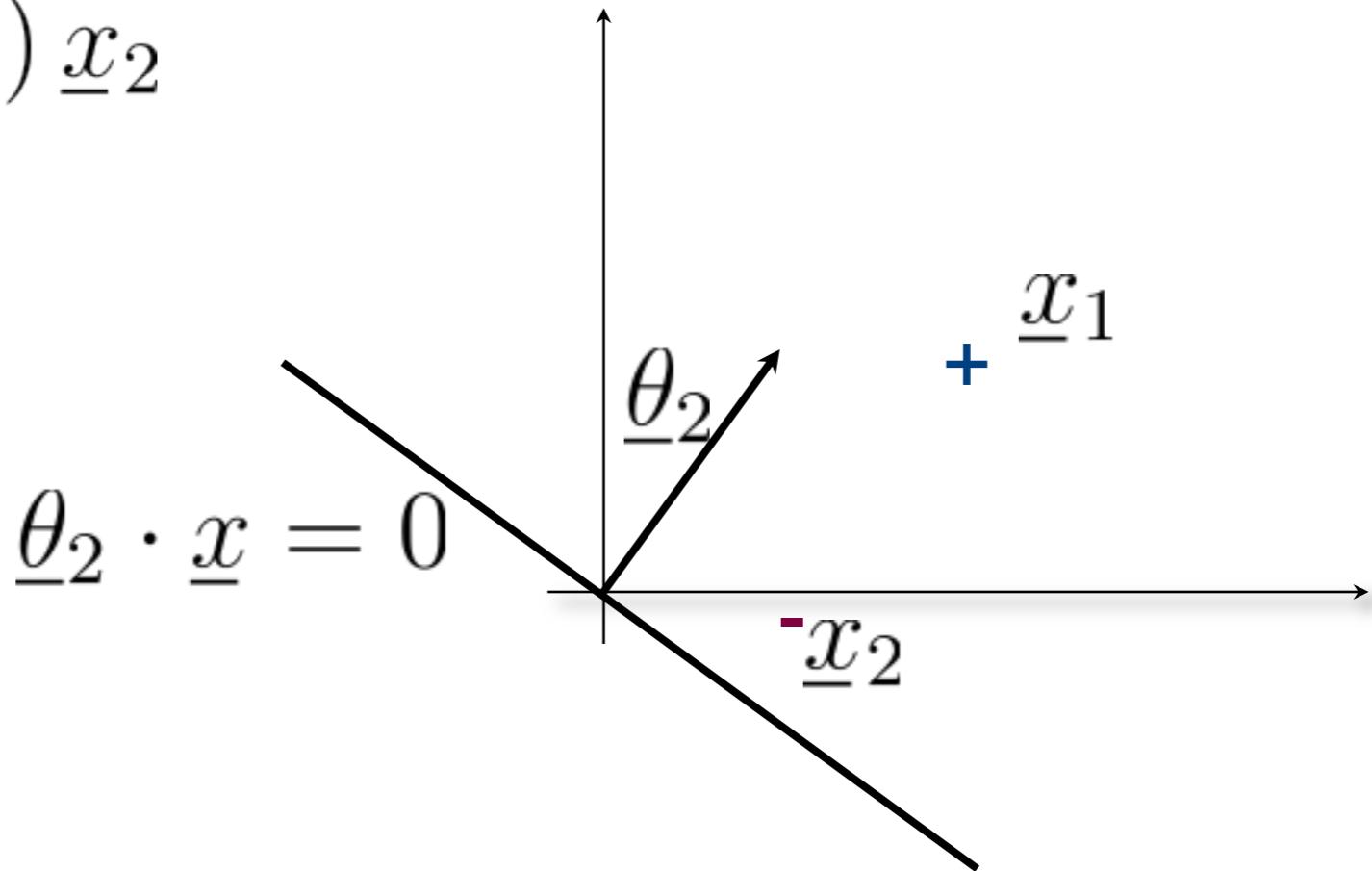
Perceptron algorithm (take 2)

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$

$$\underline{\theta}_1 = \underline{\theta}_0 + 1 \underline{x}_1$$

$$\underline{\theta}_2 = \underline{\theta}_1 + (-1) \underline{x}_2$$

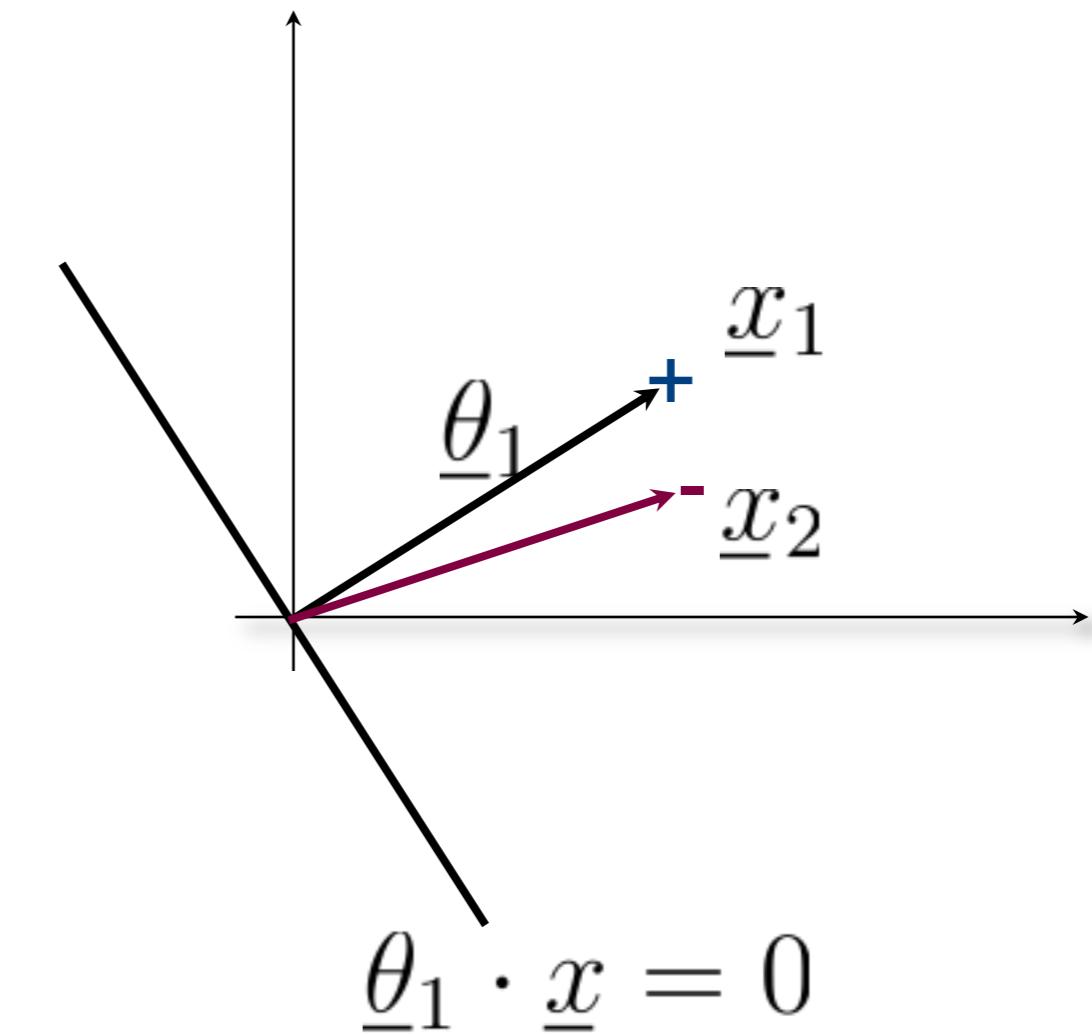


Perceptron algorithm (take 3)

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$

$$\underline{\theta}_1 = \underline{\theta}_0 + 1 \underline{x}_1$$



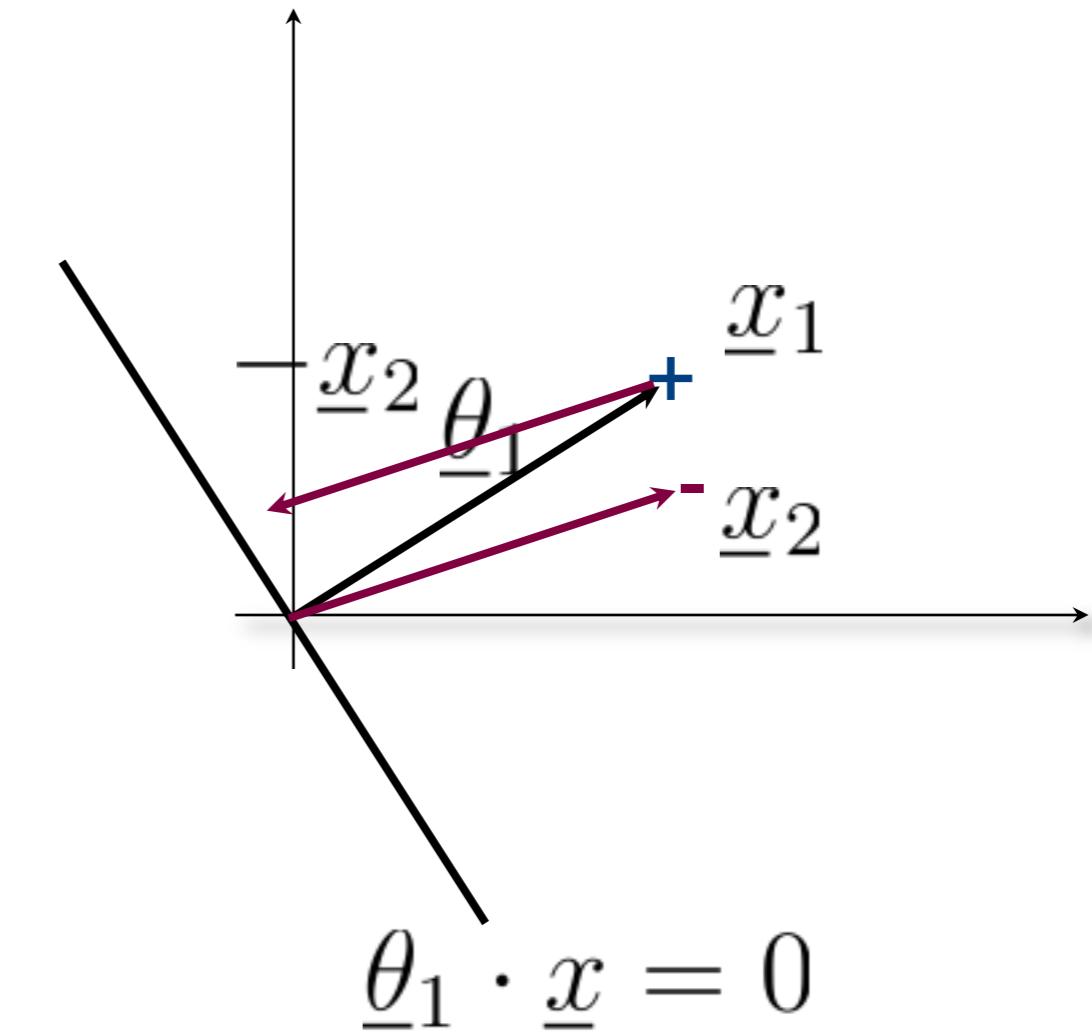
Perceptron algorithm (take 3)

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$

$$\underline{\theta}_1 = \underline{\theta}_0 + 1 \underline{x}_1$$

$$\underline{\theta}_2 = \underline{\theta}_1 + (-1) \underline{x}_2$$



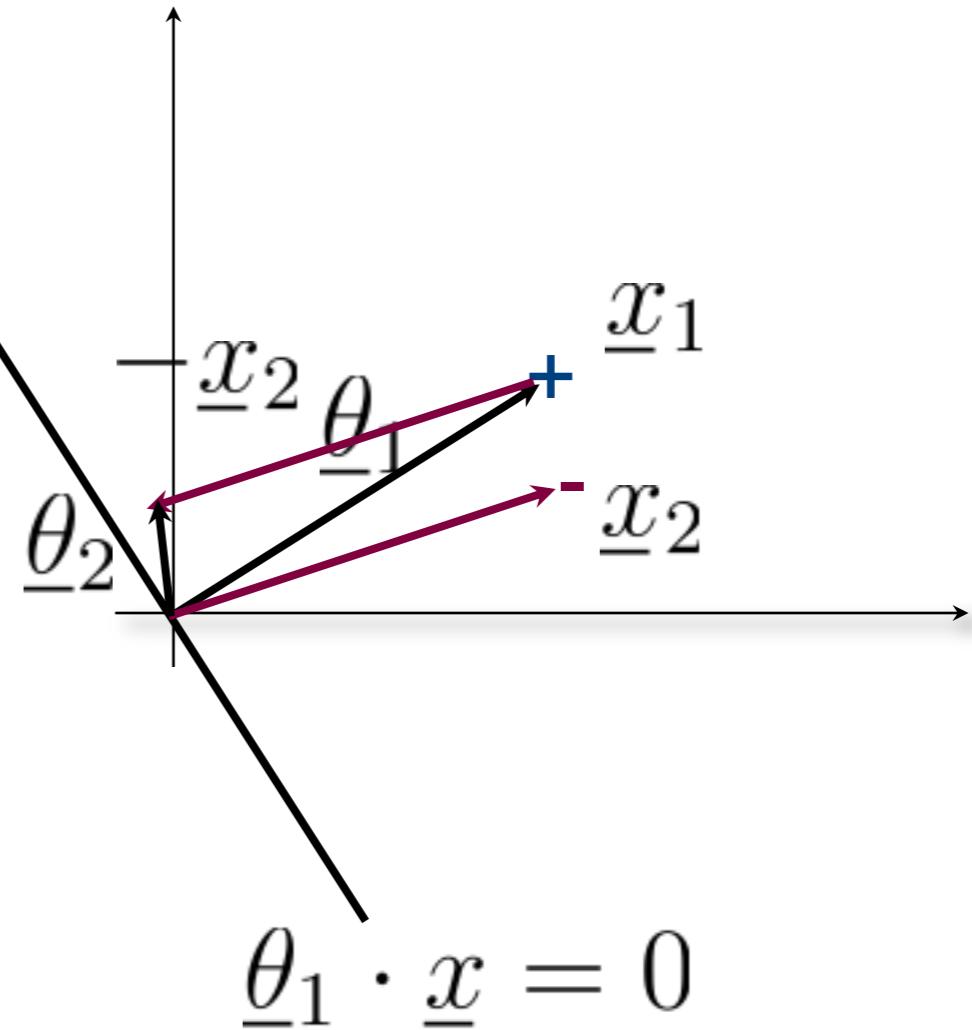
Perceptron algorithm (take 3)

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$

$$\underline{\theta}_1 = \underline{\theta}_0 + 1 \underline{x}_1$$

$$\underline{\theta}_2 = \underline{\theta}_1 + (-1) \underline{x}_2$$



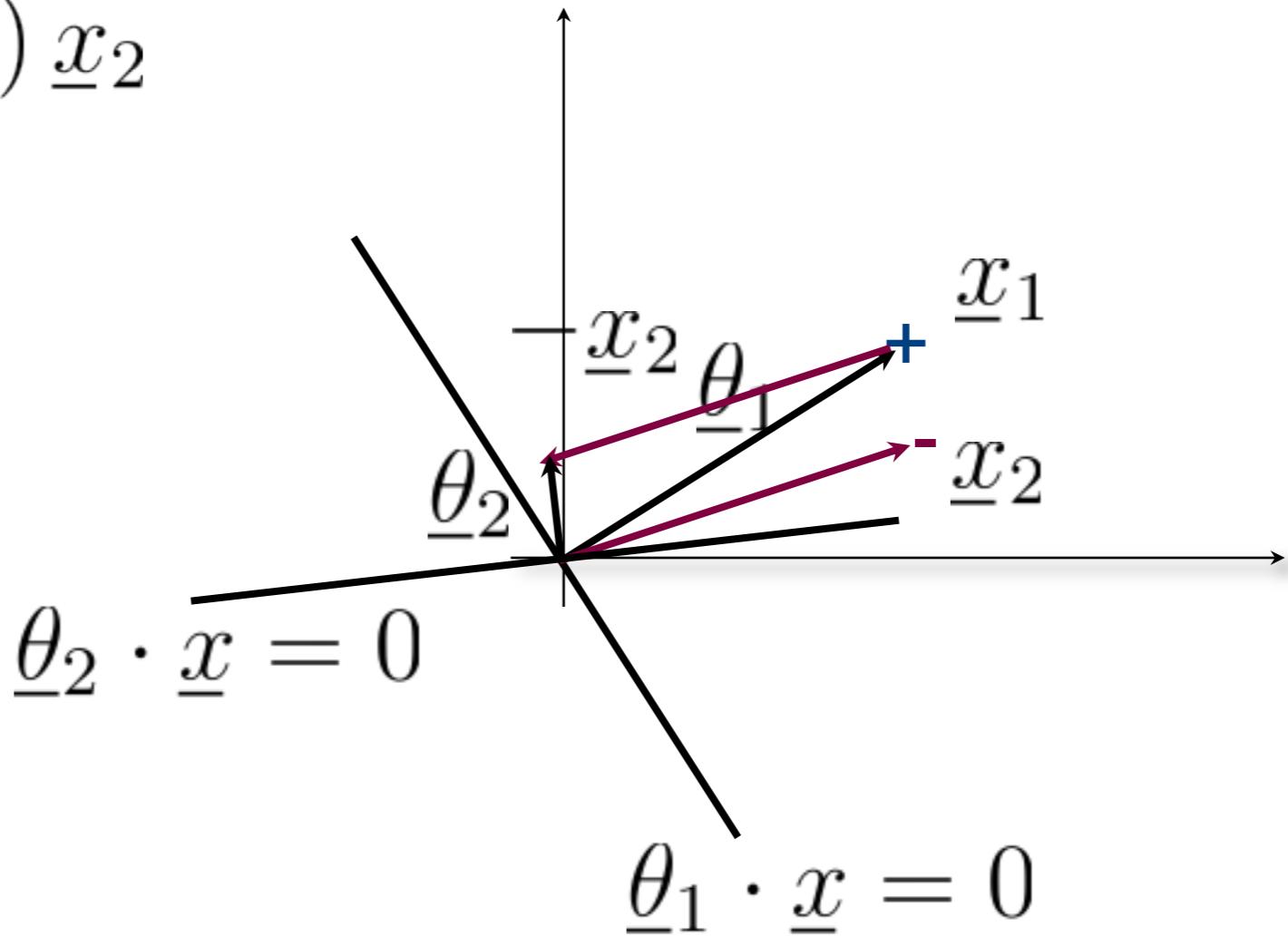
Perceptron algorithm (take 3)

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$

$$\underline{\theta}_1 = \underline{\theta}_0 + 1 \underline{x}_1$$

$$\underline{\theta}_2 = \underline{\theta}_1 + (-1) \underline{x}_2$$



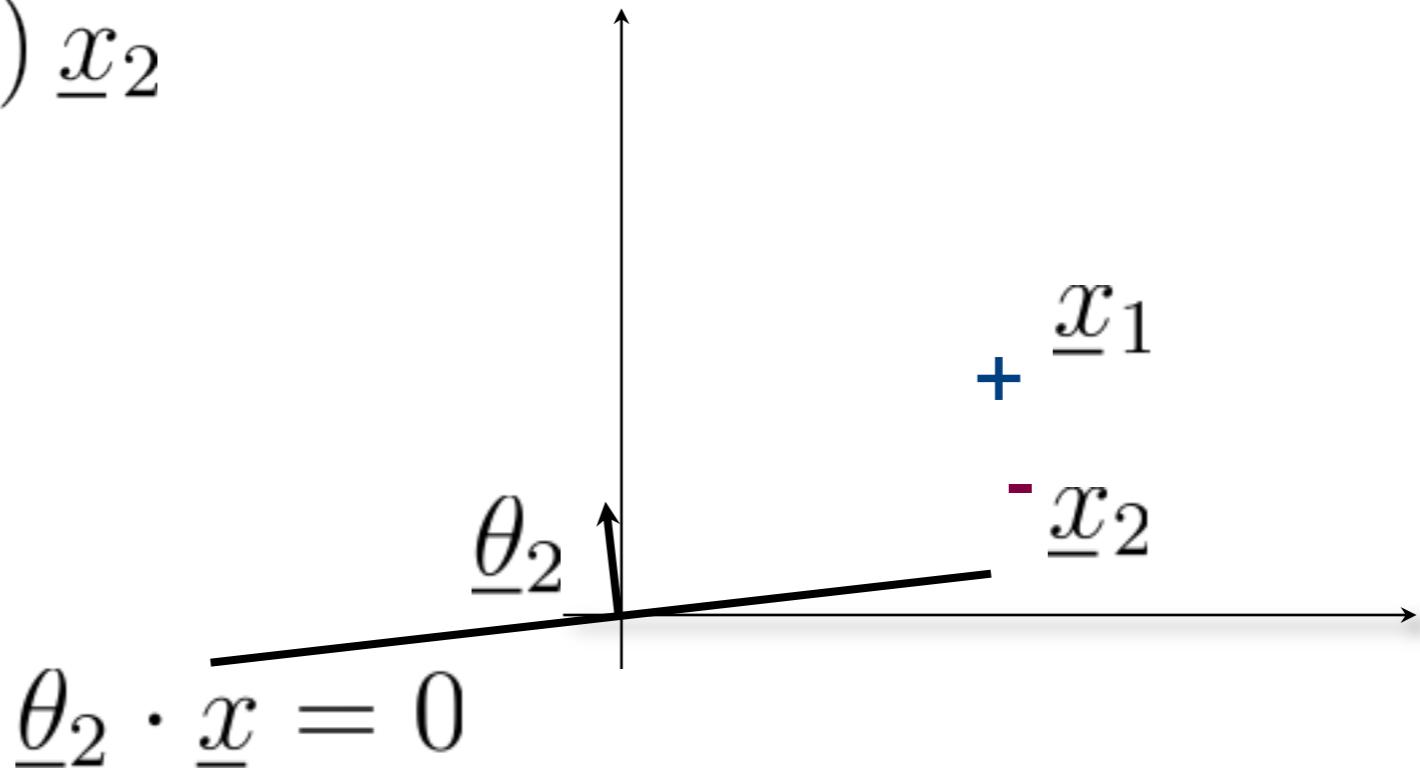
Perceptron algorithm (take 3)

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$

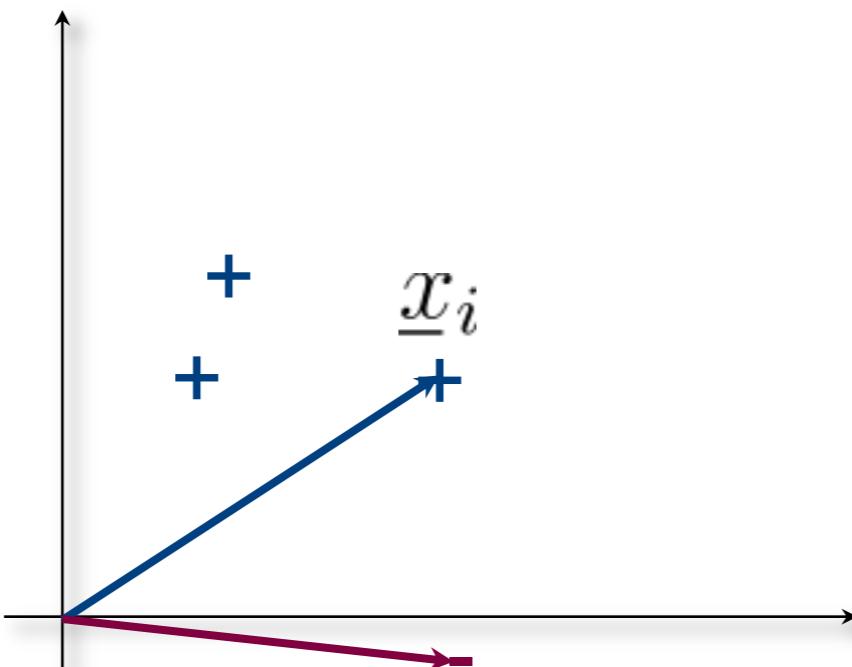
$$\underline{\theta}_1 = \underline{\theta}_0 + 1 \underline{x}_1$$

$$\underline{\theta}_2 = \underline{\theta}_1 + (-1) \underline{x}_2$$



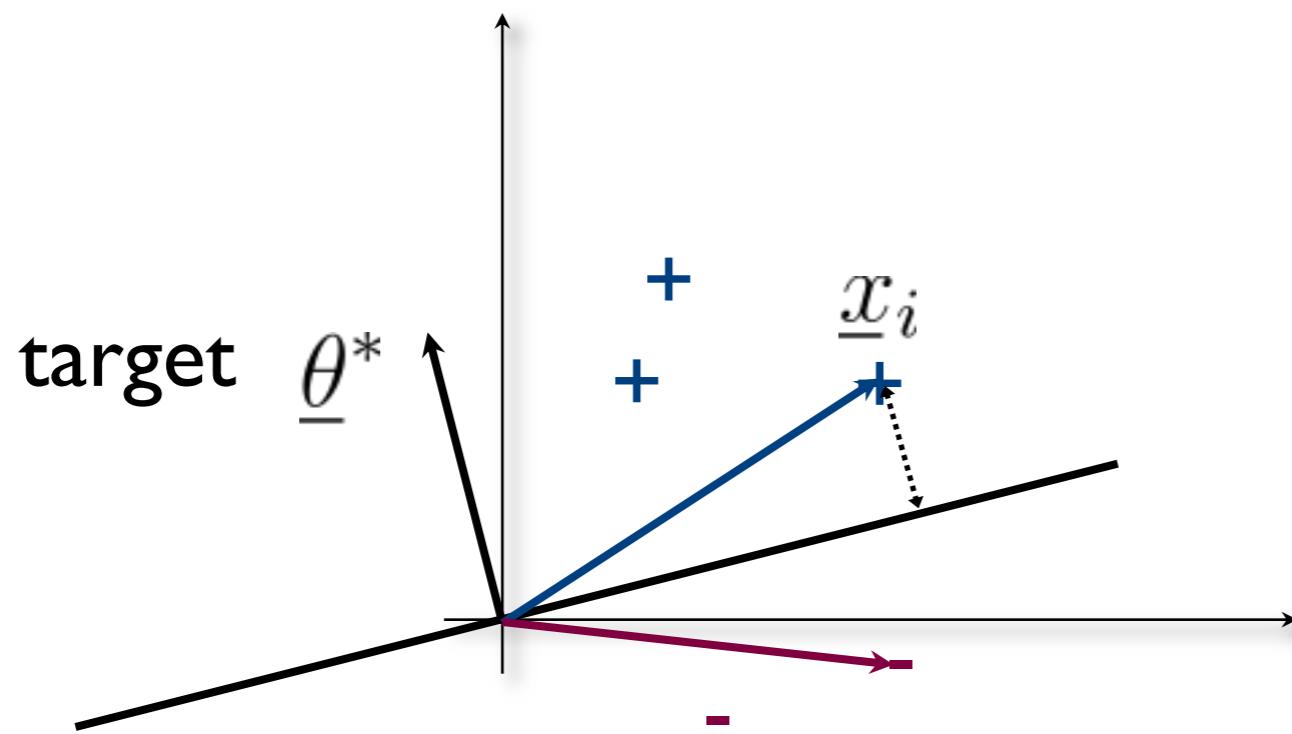
“Margin”

- We can get a handle on convergence by assuming that there exists a target classifier θ^* with good properties
- One such property is margin, i.e., how close the separating boundary is to the points



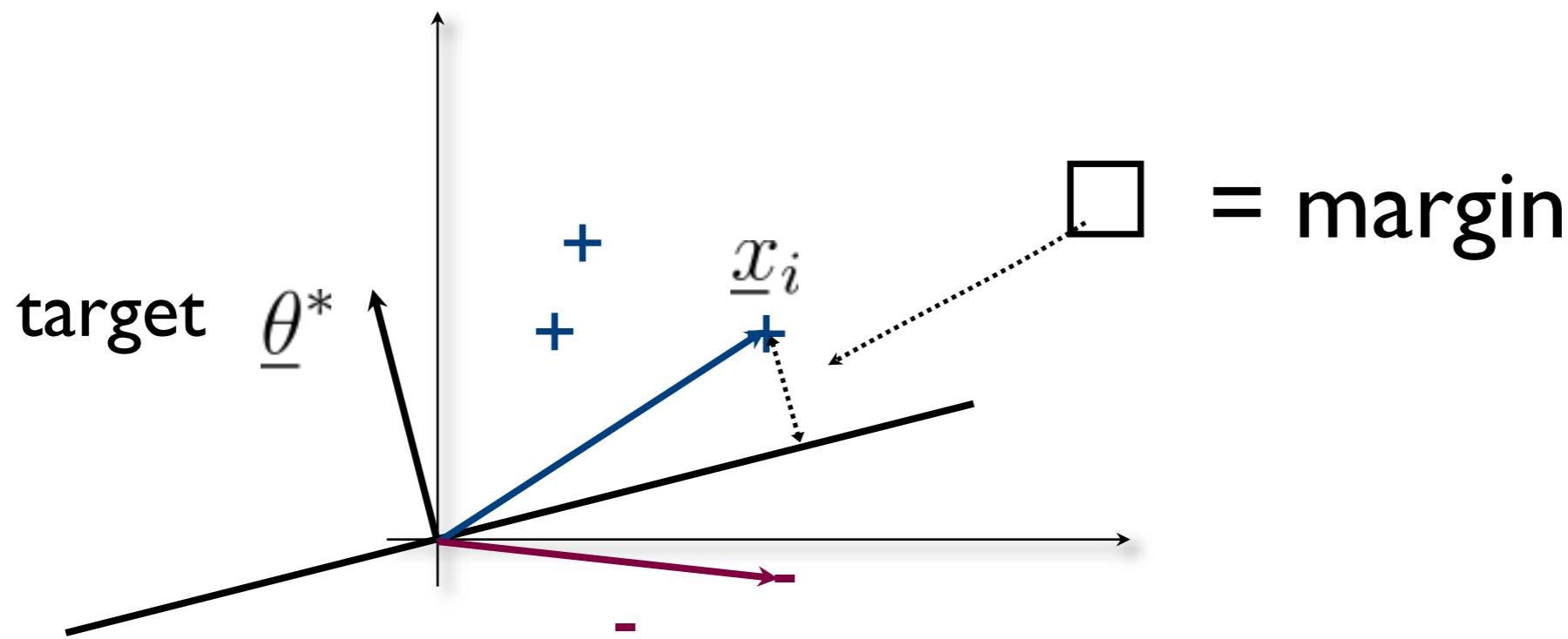
“Margin”

- We can get a handle on convergence by assuming that there exists a **target classifier** $\underline{\theta}^*$ with good properties
- One such property is margin, i.e., how close the separating boundary is to the points



“Margin”

- We can get a handle on convergence by assuming that there exists a **target classifier** $\underline{\theta}^*$ with good properties
- One such property is margin, i.e., how close the separating boundary is to the points



Perceptron

Perceptron: If $y_t(v_t \cdot x_t - \tau) < 0$ Filtering rule
 $v_{t+1} = v_t + \eta y_t x_t$ Update step

NOTE: Additive updates. $X=R^d$. Algorithm credited to [Rosenblatt '58].

We will now assume $\eta=1, \tau=0$.

Perceptron: If $y_t(v_t \cdot X_t) < 0$ Filtering rule
 $v_{t+1} = v_t + y_t x_t$ Update step

Note: here we use the notation $v = \theta$, $u = \theta^*$

Problem framework

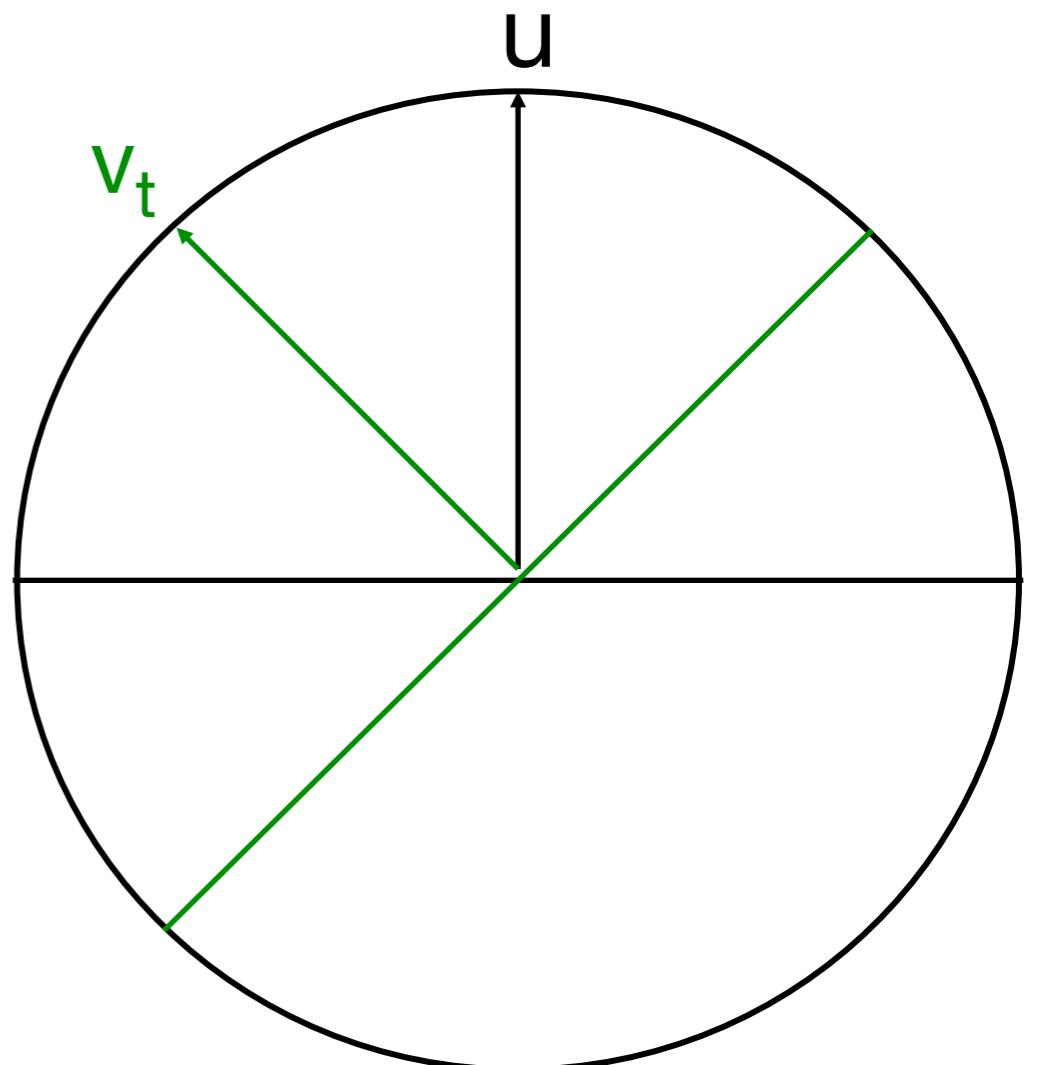
$$x_t \in \mathbb{R}^d, \|x\| \leq R, y_t \in \{-1, +1\}$$

Separability: there exists some perfect classifier, u , such that:

$$u : y_t(u \cdot x_t) > 0 \quad \forall t, \|u\| = 1$$

Assume u is through origin.
→ Always possible, by increasing dimension by 1.

Current hypothesis: v_t
(Called θ on previous slides)



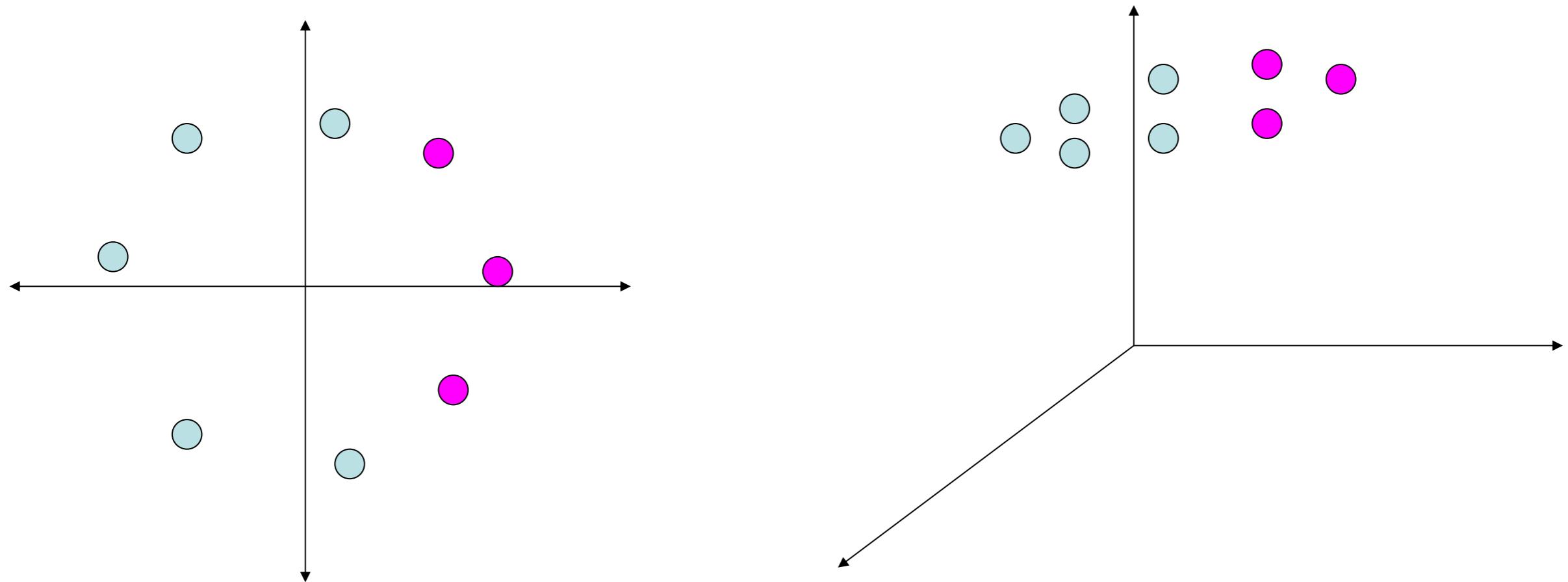
Preprocessing step

Points (x, y) , where input space $X = \mathbb{R}^d$, label space $Y = \{+1, -1\}$

Add an extra feature to x , and set it to 1:

$$x^0 = (x, 1) \text{ in } \mathbb{R}^{d+1}$$

Then: points (x, y) linearly separable \Leftrightarrow points (x^0, y) linearly separable by a hyperplane through the origin

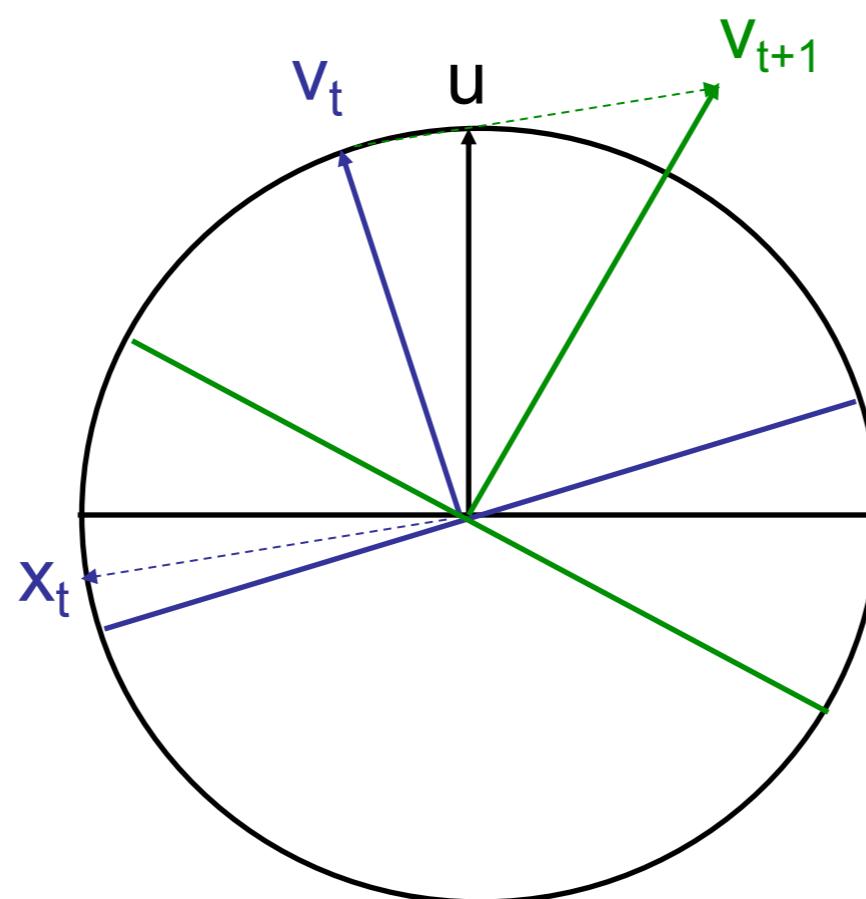


Perceptron

Perceptron: If $y_t(v_t \cdot x_t) < 0$ Filtering rule
 $v_{t+1} = v_t + y_t x_t$ Update step

NOTE: Additive updates. Algorithm credited to [Rosenblatt '58].

Example: data is separable by u :



Perceptron

Analyses of standard Perceptron:

Linearly separable (through origin) data, uniform distribution:

- $\tilde{O}(d/\varepsilon^2)$ mistakes (to reach error ε) [Baum '89].
- $\Omega(1/\varepsilon^2)$ mistakes [Dasgupta, Kalai & Monteleoni, '05].
- $\Omega(1/\varepsilon^2)$ labels for active learning [Dasgupta, Kalai & Monteleoni, '05].

Margin assumption: no distribution assumption except
linearly separable (through origin), with margin γ

$$y_t(u \cdot x_t) \geq \gamma \text{ for all } t.$$

- $O(1/\gamma^2)$ mistakes to reach zero error [Novikoff '62].

Perceptron analysis with margin

Margin assumption: no distribution assumption except separable (through origin), AND: $y_t(u \cdot x_t) \geq \gamma$ for all t .

- $O(1/\gamma^2)$ mistakes to reach zero error [Novikoff '62].

Proof: Let $\|u\| = 1$. Let (x, y) be a mistake, i.e. $y(v_t \cdot x) < 0$,
 $\|x\| \leq R$.

Lemma 1: $u \cdot v_{t+1} \geq u \cdot v_t + \gamma$.

Proof: $u \cdot v_{t+1} = u \cdot (v_t + y x) = u \cdot v_t + y(u \cdot x) \geq u \cdot v_t + \gamma$
(by definition of margin, γ).

Lemma 2: $\|v_{t+1}\|^2 \leq \|v_t\|^2 + R^2$

Proof: $\|v_{t+1}\|^2 = \|v_t + y x\|^2 = \|v_t\|^2 + 2y(v_t \cdot x) + \|x\|^2$
 $\leq \|v_t\|^2 + 2y(v_t \cdot x) + R^2$
 $< \|v_t\|^2 + R^2$

because v_t makes a mistake on (x, y) , i.e. $2y(v_t \cdot x) < 0$.

Perceptron analysis with margin

Proof continued:

Let $\|u\| = 1$. Let (x, y) be a mistake, i.e. $y(v_t \cdot x) < 0$, $\|x\| \leq R$.

Lemma 1: $u \cdot v_{t+1} \geq u \cdot v_t + \gamma$.

Lemma 2: $\|v_{t+1}\|^2 \leq \|v_t\|^2 + R^2$.

Finally, after M mistakes:

- $u \cdot v_{M+1} \geq M \gamma$, by Lemma 1 (expanding the recurrence).
- $\|v_{M+1}\|^2 \leq M R^2$, by Lemma 2. So $\|v_{M+1}\| \leq M^{1/2} R$.

Since u is a unit vector, $u \cdot v_t \leq \|v_t\|$ by Cauchy-Schwartz: $|u \cdot v| \leq \|u\| \|v\|$

So, $u \cdot v_{M+1} \leq \|v_{M+1}\|$.

Using a. and b. for LHS and RHS respectively,

$$M \gamma \leq u \cdot v_{M+1} \leq \|v_{M+1}\| \leq M^{1/2} R$$

$$M \gamma \leq M^{1/2} R$$

$$M^{1/2} \leq R / \gamma, \text{ and } M \leq (R / \gamma)^2 \quad \square$$

Fisher's IRIS data

Four features

sepal length

sepal width

petal length

petal width

Three classes (species of iris)

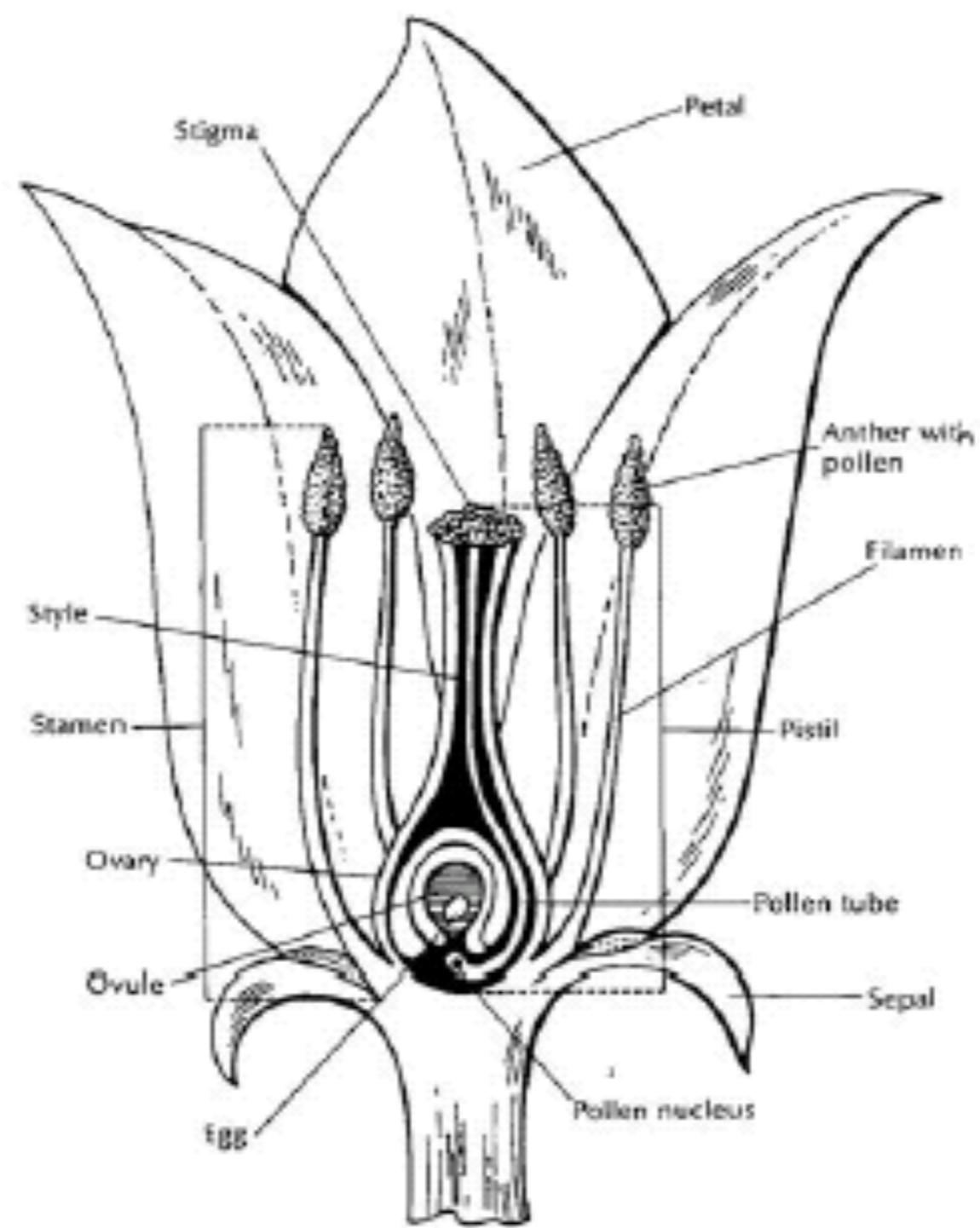
setosa

versicolor

virginica

50 instances of each

Parts of a Flower

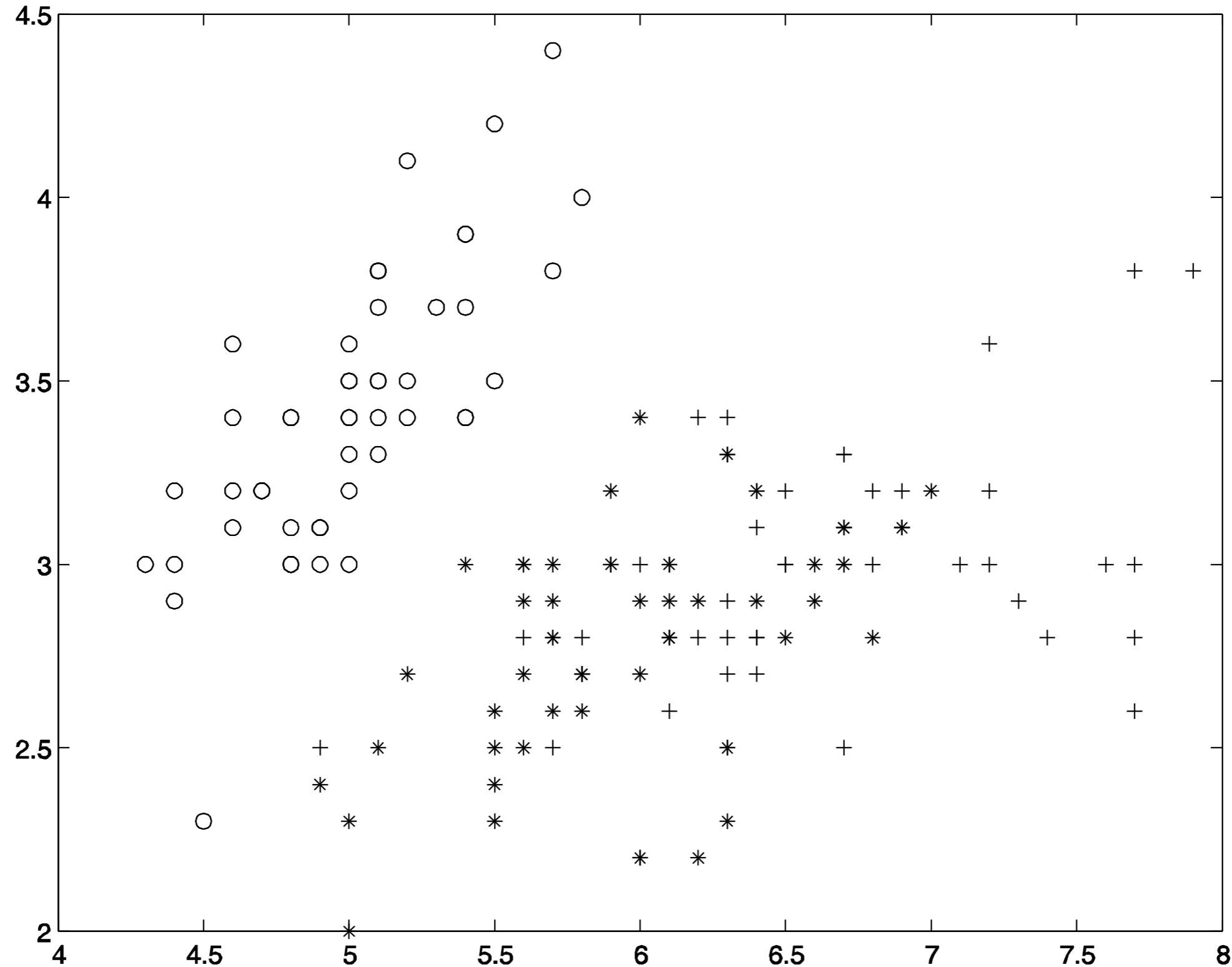


THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS

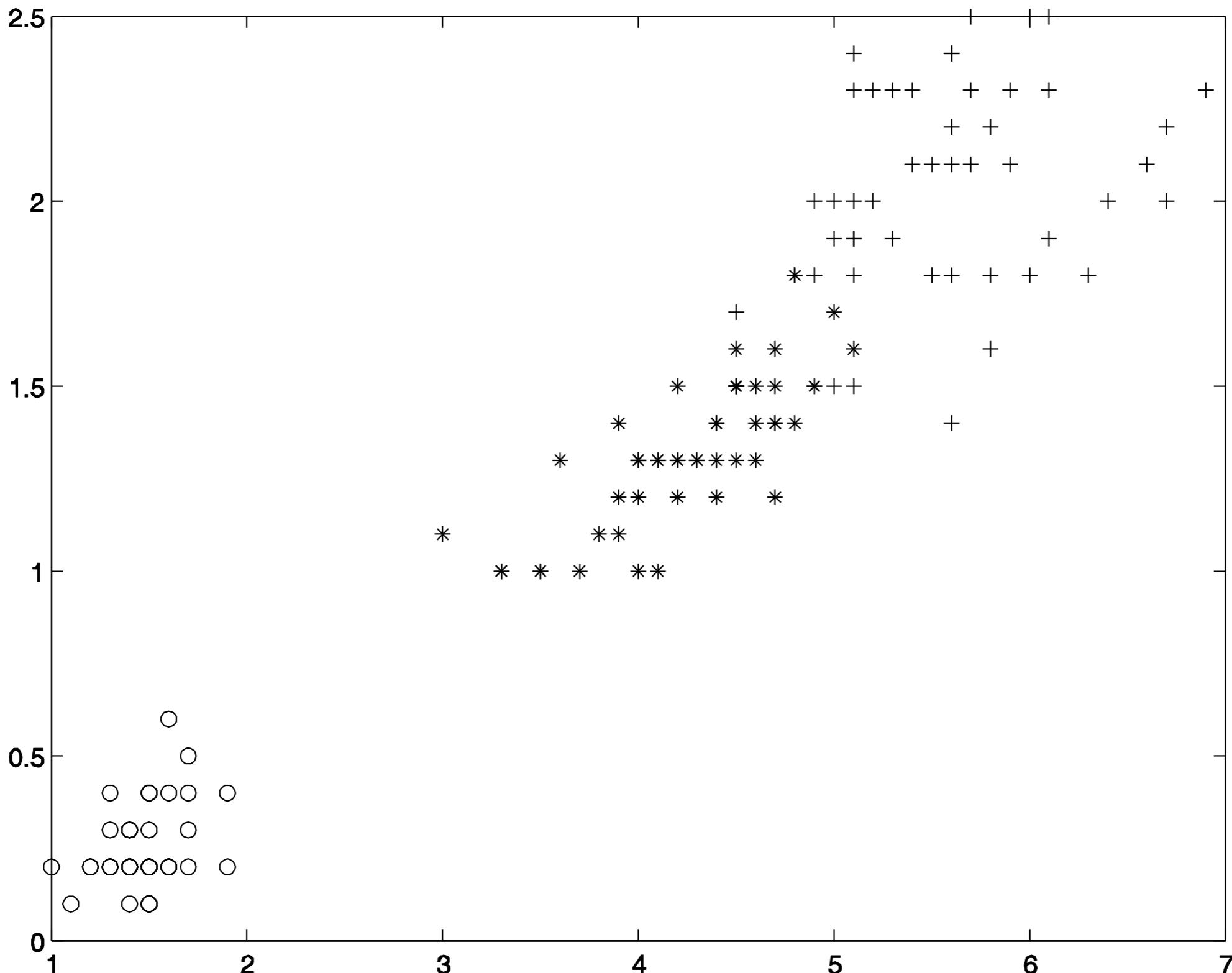
BY R. A. FISHER, Sc.D., F.R.S.

I. DISCRIMINANT FUNCTIONS

WHEN two or more populations have been measured in several characters, x_1, \dots, x_s , special interest attaches to certain linear functions of the measurements by which the populations are best discriminated. At the author's suggestion use has already been made of this fact in craniometry (*a*) by Mr E. S. Martin, who has applied the principle to the sex differences in measurements of the mandible, and (*b*) by Miss Mildred Barnard, who showed how to obtain from a series of dated series the particular compound of cranial measurements showing most distinctly a progressive or secular trend. In the present paper the application of the same principle will be illustrated on a taxonomic problem; some questions connected with the precision of the processes employed will also be discussed.

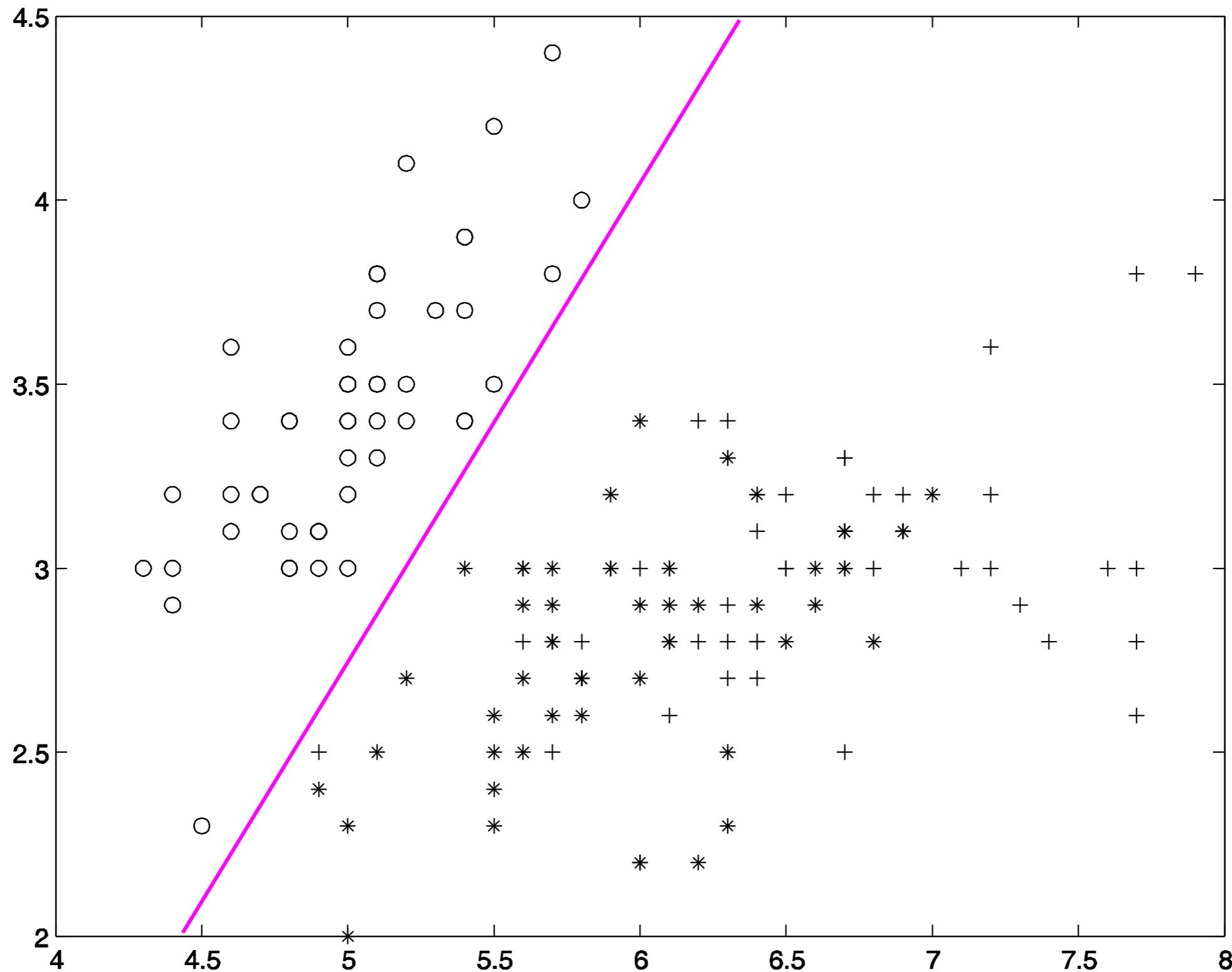


Features 1 and 2 (sepal width/length)



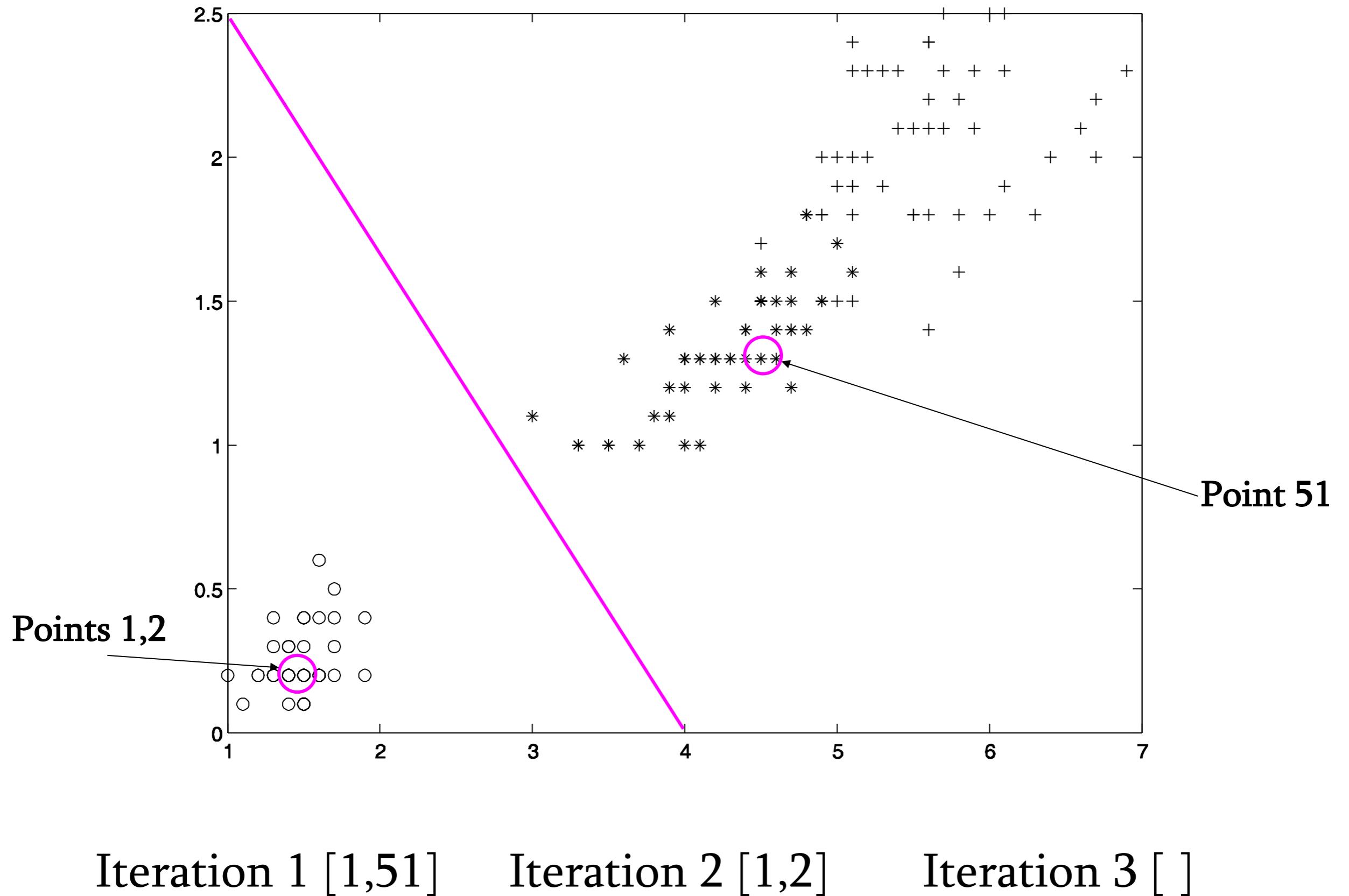
Features 3 and 4 (petal width/length)

Features 1 and 2; goal: separate setosa from other two

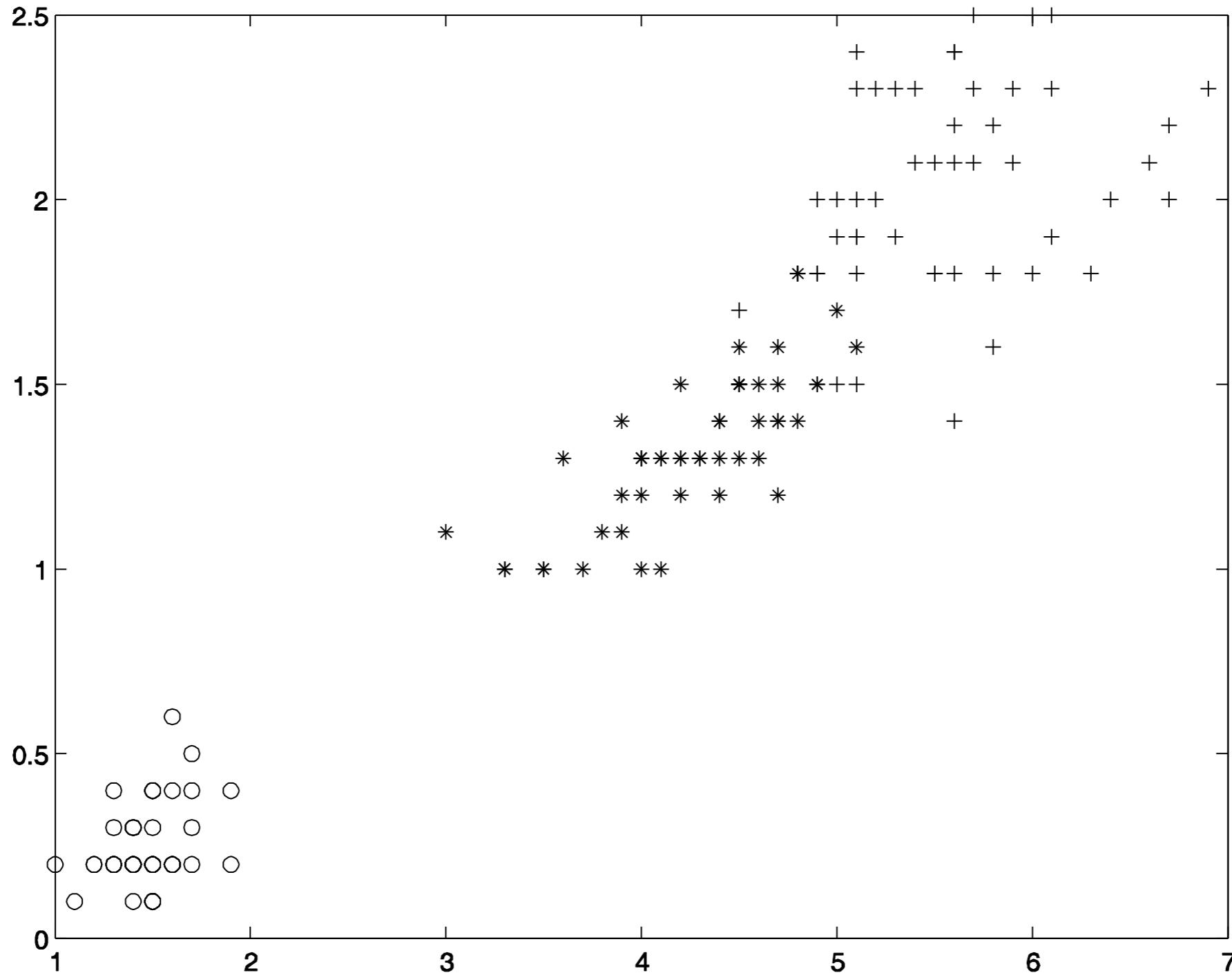


1500 updates (different permutation: 900)

Features 3 and 4; goal: separate setosa from other two



Features 3 and 4; goal: separate versicolor from other two



What if the data is not linearly separable?