Machine Learning

CSCI 5622 Fall 2020

Prof. Claire Monteleoni

Today

- [Finish] Nearest neighbor
- Intro to Linear Classification

With credit to T. Jaakkola and S. Dasgupta

Beyond Simple Nearest Neighbor

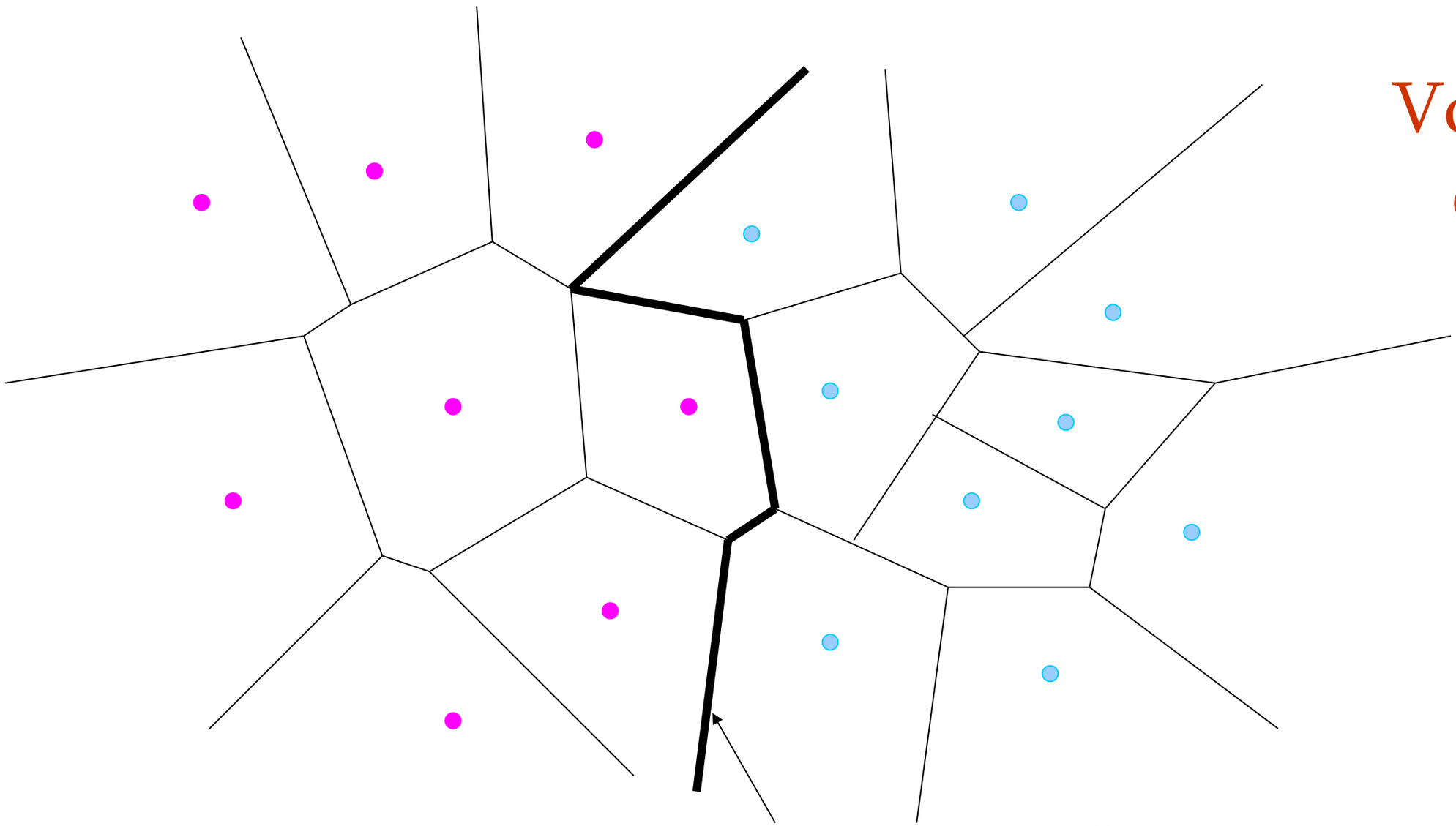
Learning problem:

Design algorithms to perform nearest neighbor classification:

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

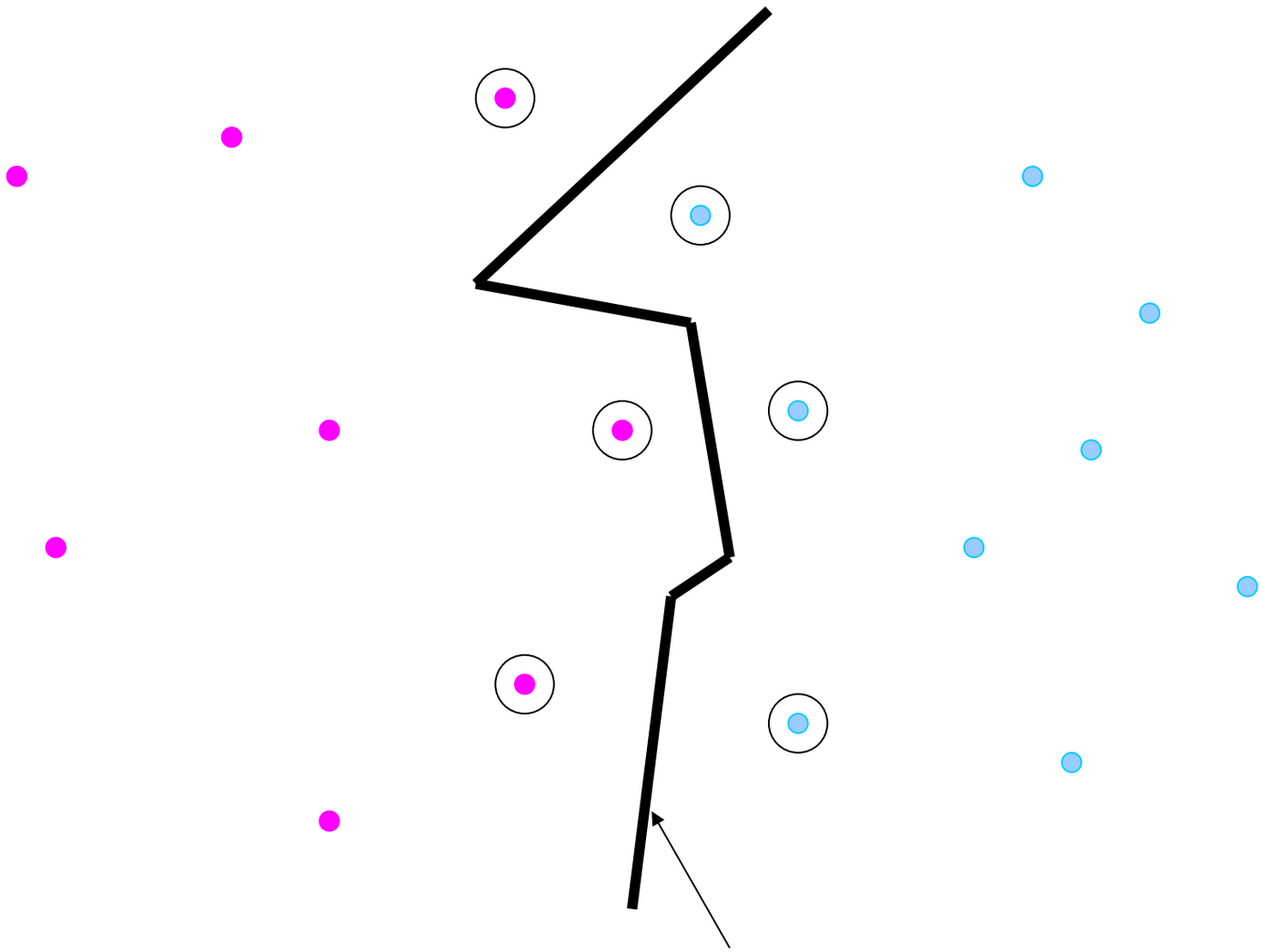
Prototype selection

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

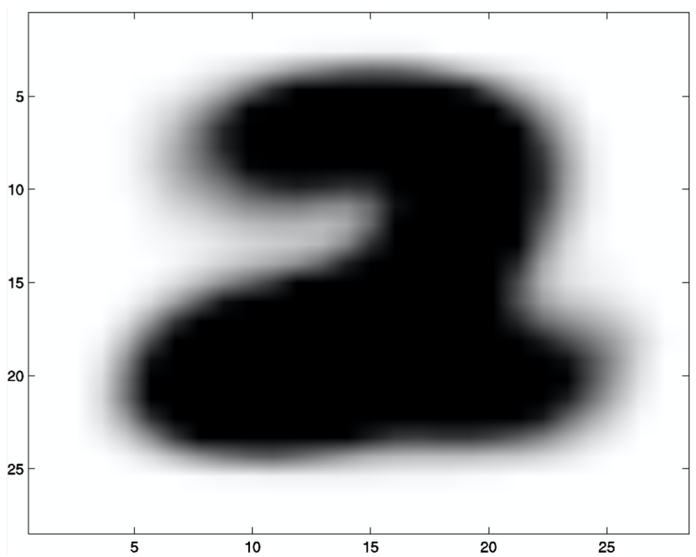
# prototypes	% error rate
20000	6.0
10000	7.4
5000	8.5
2000	10.7
1000	14.3
500	17.8
250	20.8
100	32.3
50	43.0

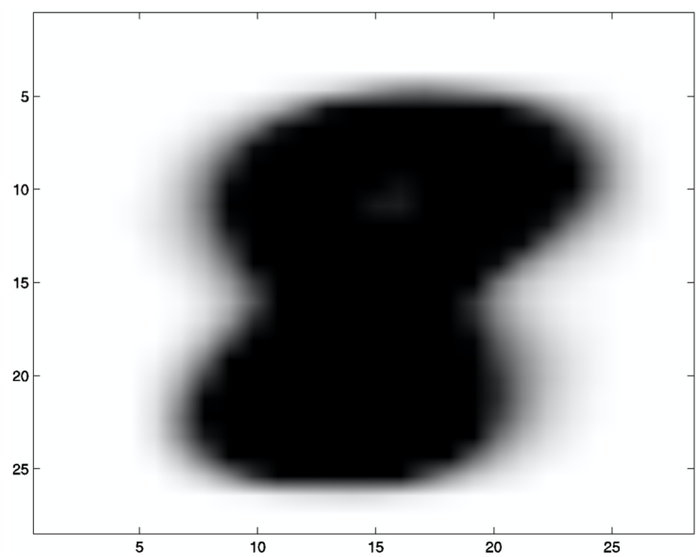
How to pick prototypes?

They do not have to be actual data points!

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

Examples:





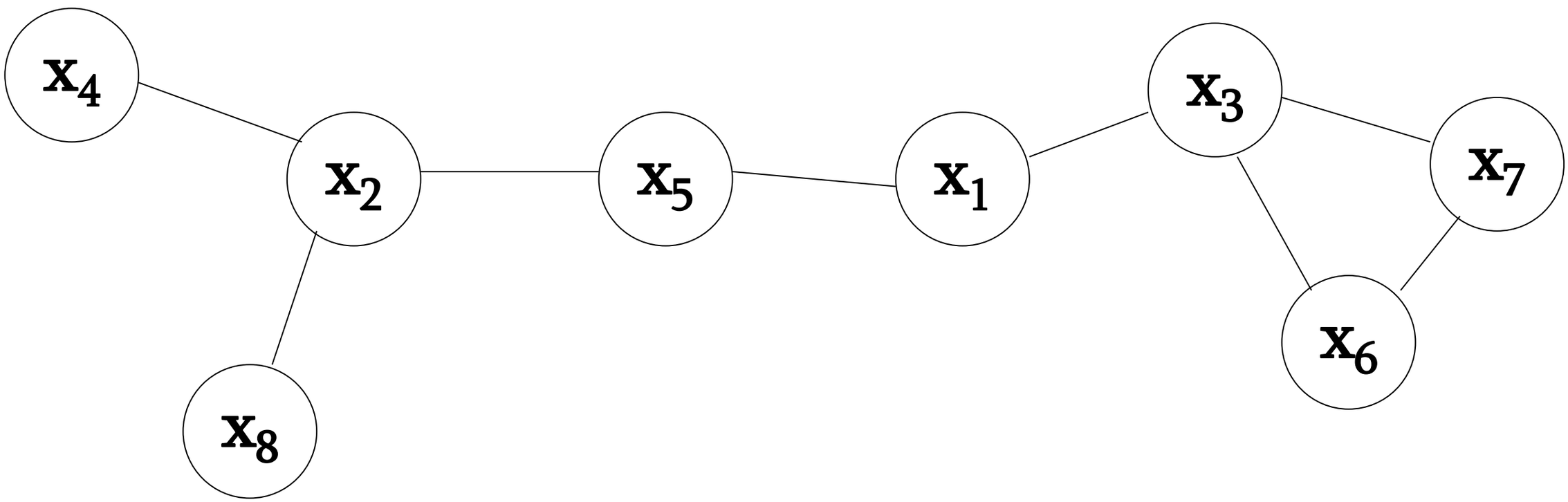
Error = 23%. Number of prototypes =

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]

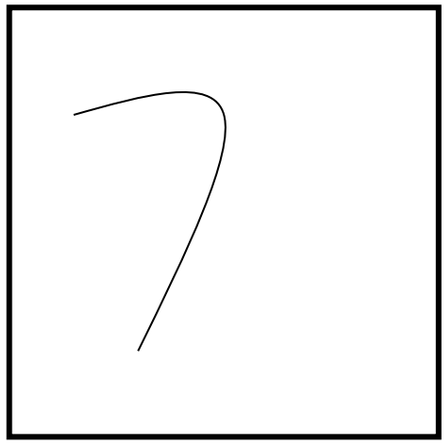
Measure of simplicity:

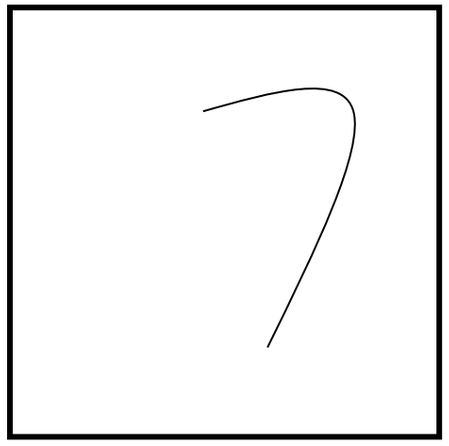
"Vapnik-Chervonenkis dimension" (later in course)

Postscript: representation I

The data lie in 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?)

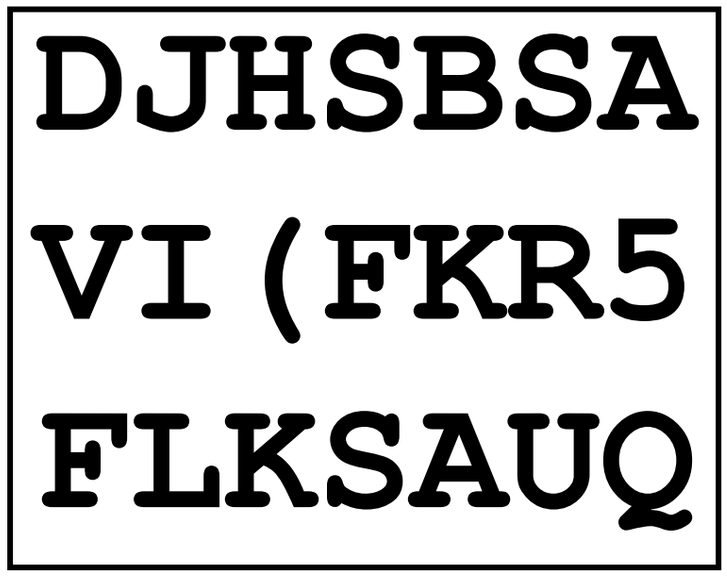
Is there a better distance measure for this application?

Postscript: representation II

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

Often data looks like this (figuratively speaking):

digit 5:

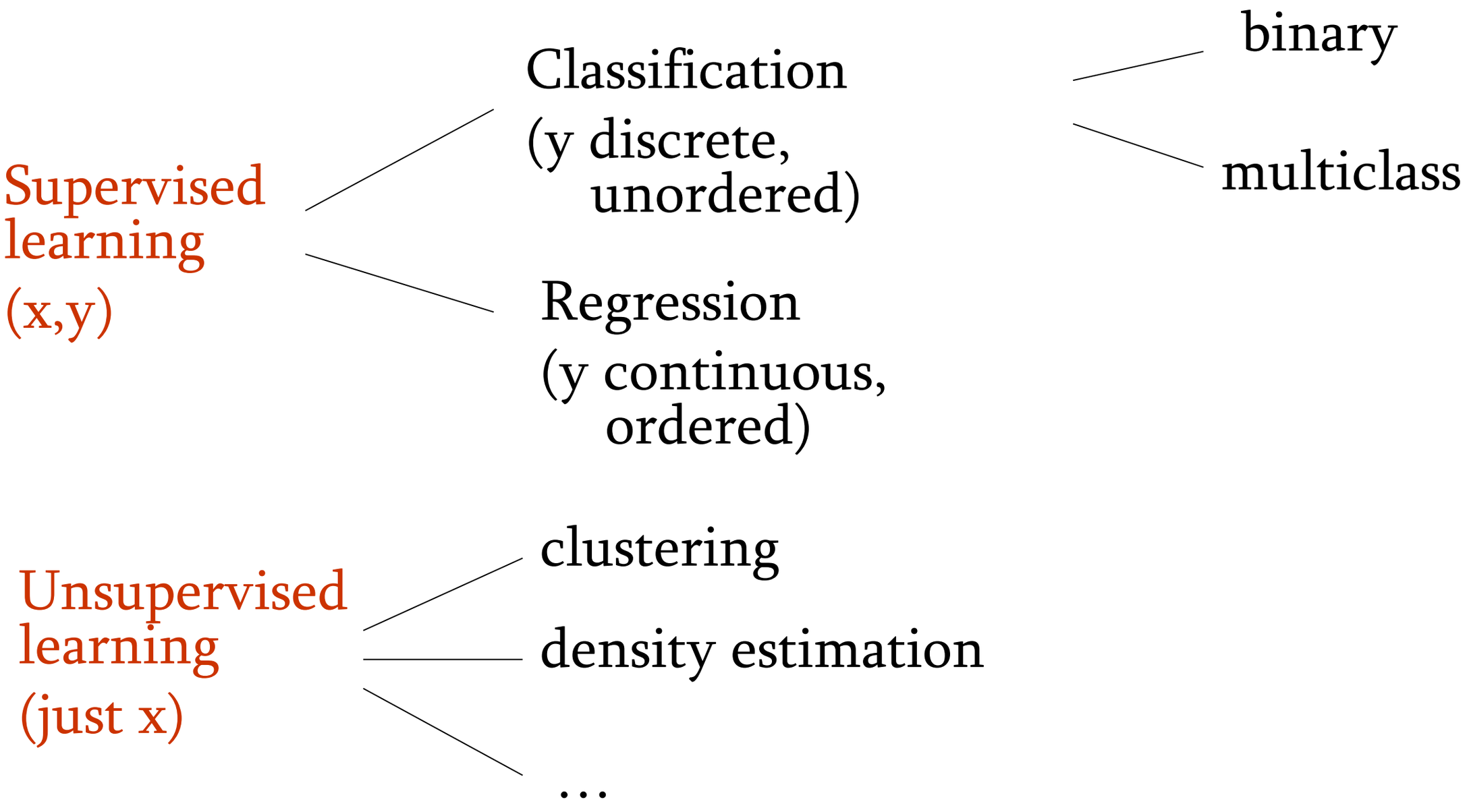


Information buried in a sea of irrelevant features.

Danger for nearest neighbor: match based on irrelevant features.

Feature selection is very important.

Learning tasks

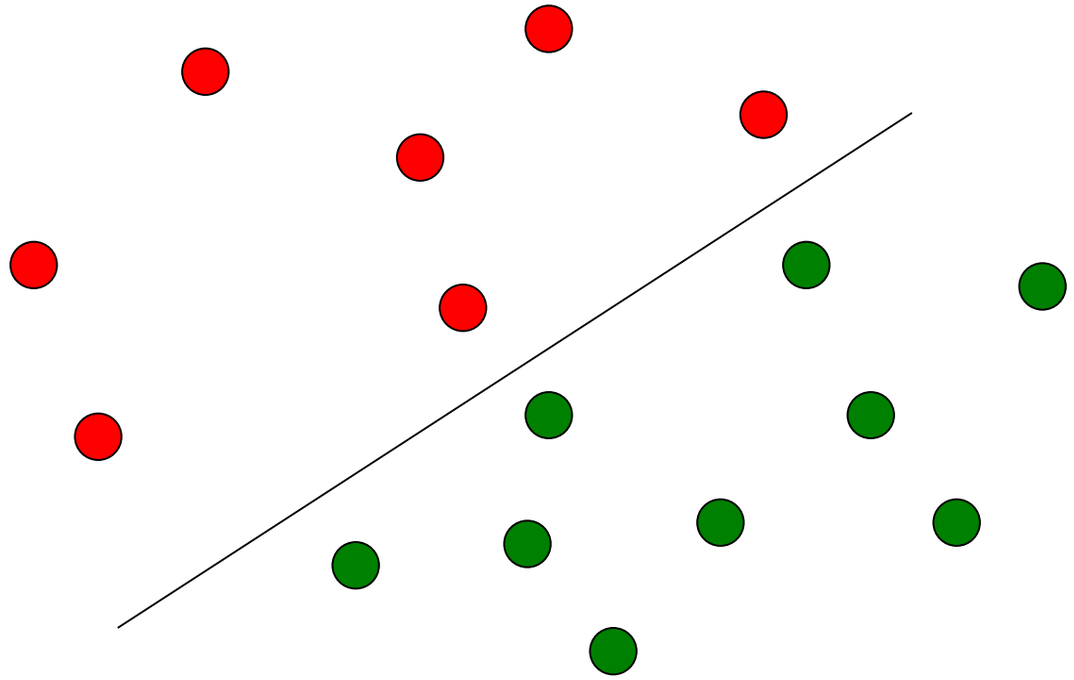


All these tasks have a nearest-neighbor solution.

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)

Classification

- Learning from examples

+ l

- l

+ l

- l

training set

Classification

- Learning from examples

+1

-1

+1

-1

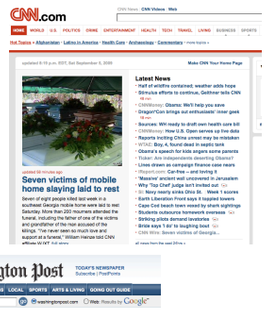
The training set of labeled examples specifies the learning task only implicitly

training set

Classification

- Learning from examples

+1



-1

?

+1

-1

training set

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

Classification

- Learning from examples

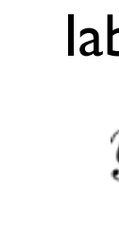
+1

-1

+1

-1

training set



predicted label

new website

$$y = \hat{f}\left(\right)$$

classifier = mapping from websites to labels

“Examples”

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

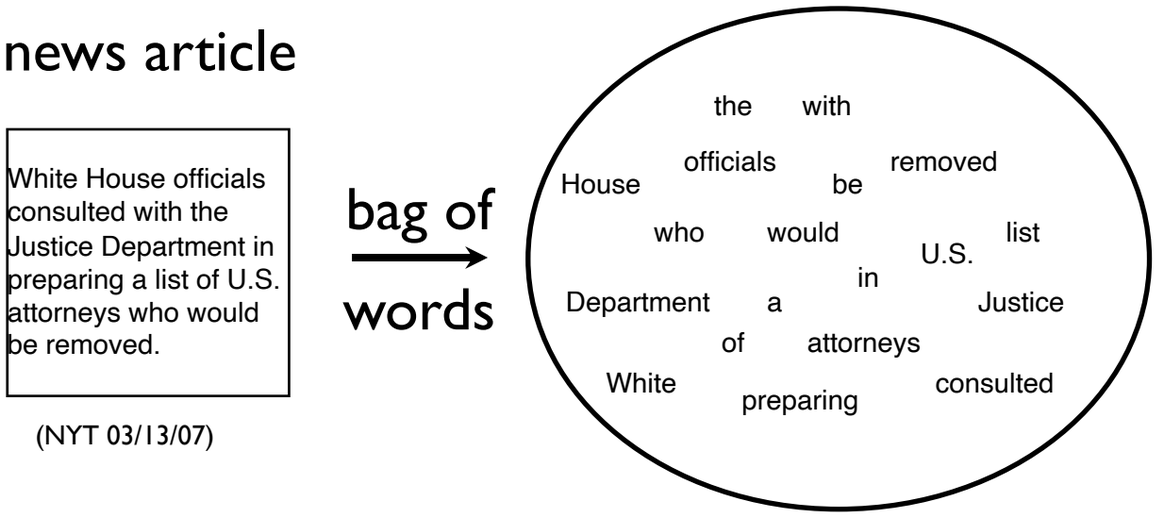
news article

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

(NYT 03/13/07)

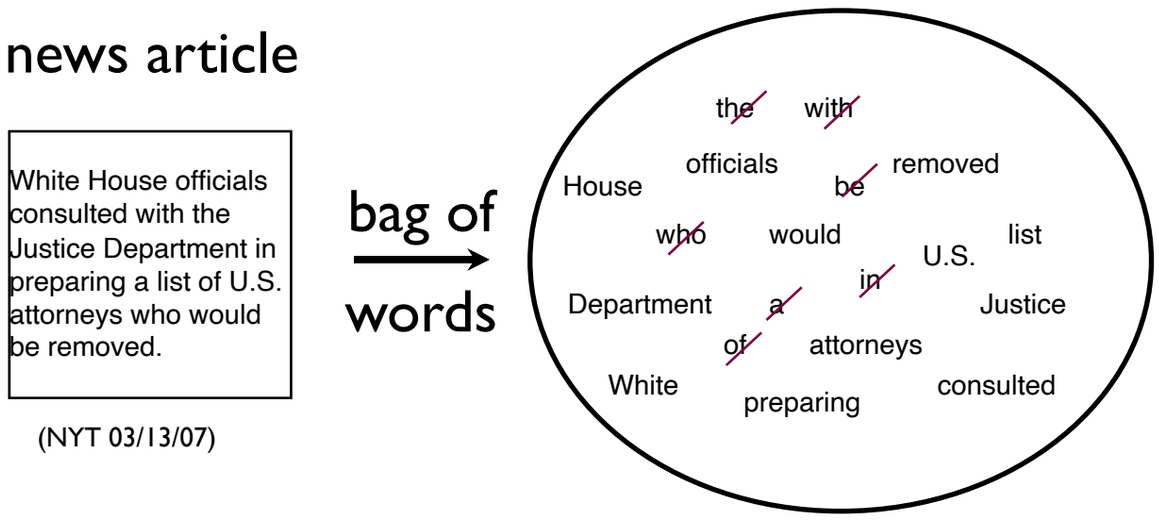
"Examples"

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



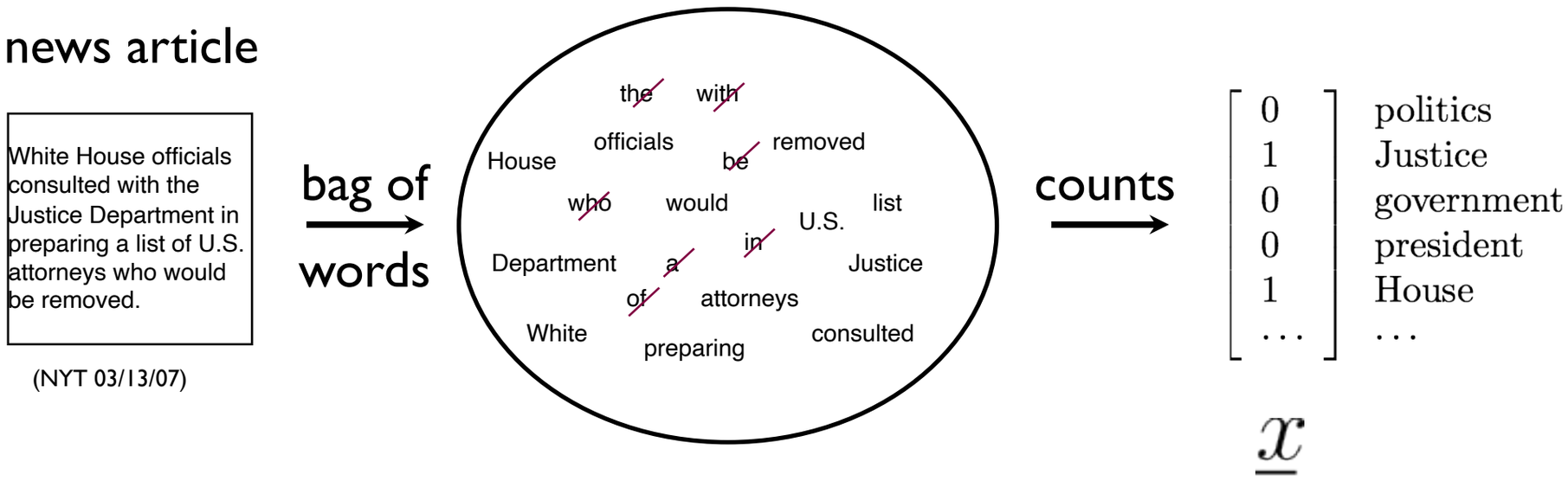
"Examples"

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



"Examples"

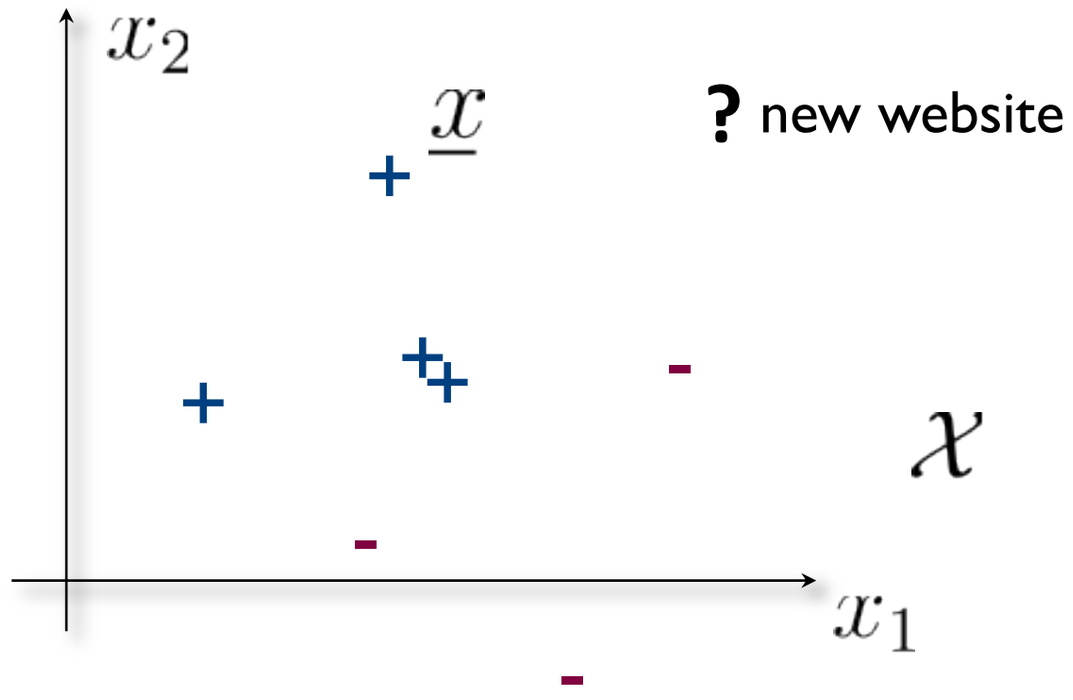
- We will have to first represent the examples (websites) in a manner that can be easily mapped to labels



a vector whose coordinates (features) specify how many times (or whether) particular words appeared in the article

The learning task

- The training set is now a set of labeled points

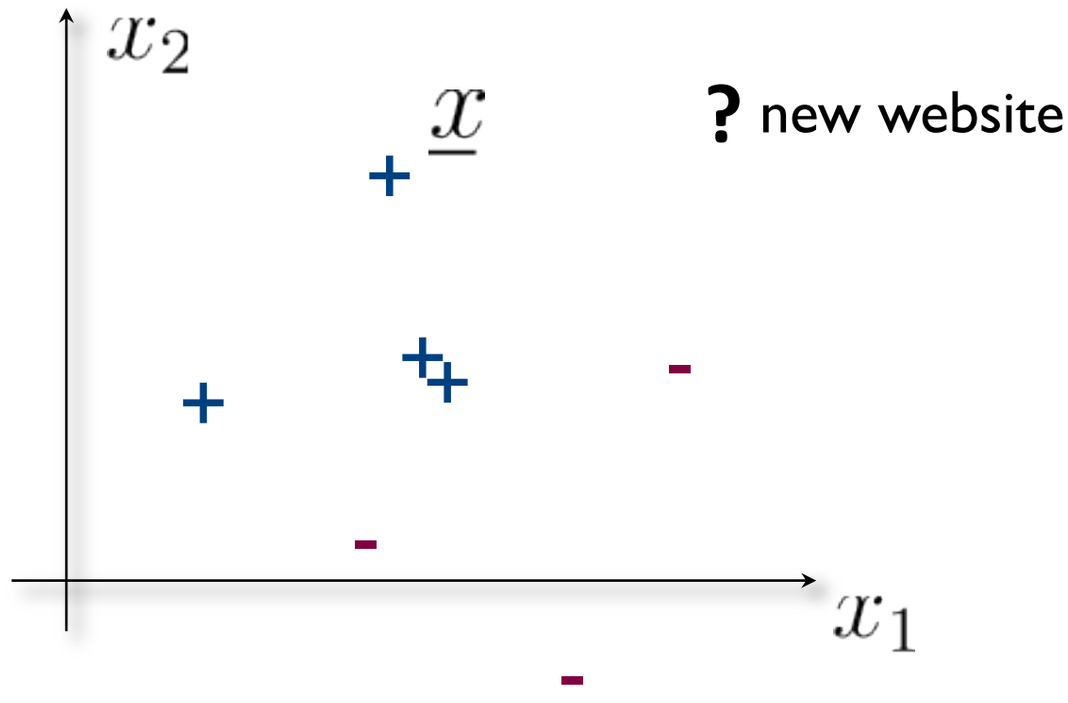


Our goal is to find a "good" classifier $f : \mathcal{X} \rightarrow \{-1, 1\}$ based on the training set $D = \{(\underline{x}_i, y_i)_{i=1,\ldots,n}\}$ so that $f(\underline{x})$ correctly labels any new websites $\underline{x}$

The learning task

- The training set is now a set of labeled points

Part 1:
Model selection
what type of classifiers should we consider?

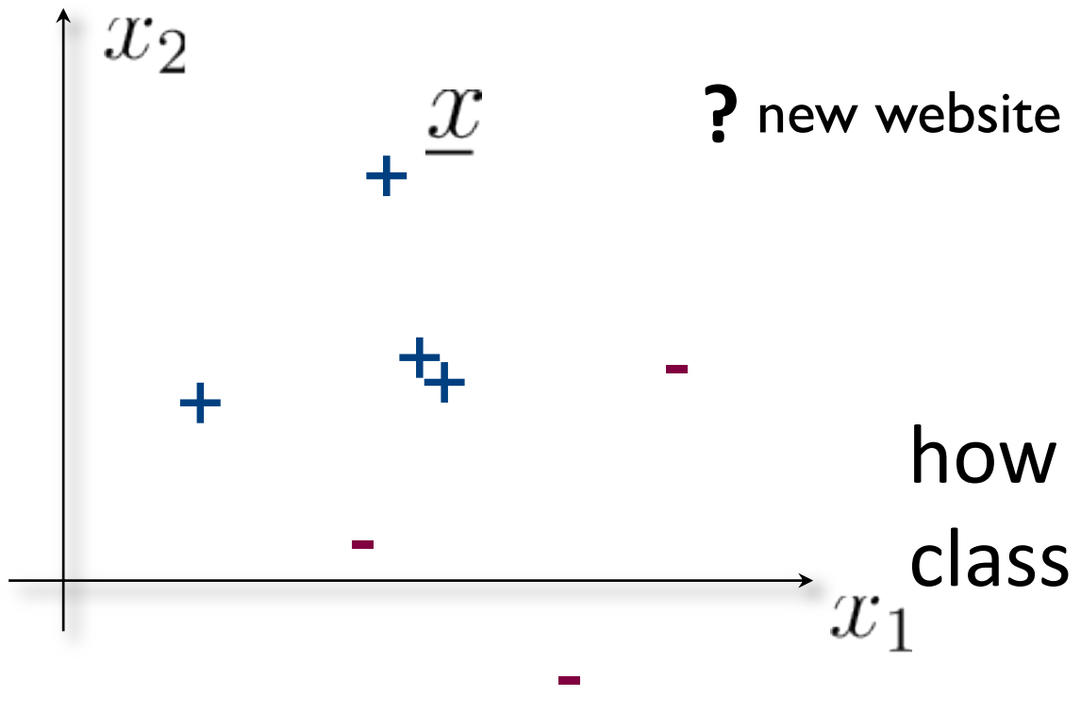


Our goal is to find a "good" classifier $f : \mathcal{X} \rightarrow \{-1, 1\}$
based on the training set $D = \{(\underline{x}_i, y_i)_{i=1,\ldots,n}\}$
so that $f(\underline{x})$ correctly labels any new websites $\underline{x}$

The learning task

- The training set is now a set of labeled points

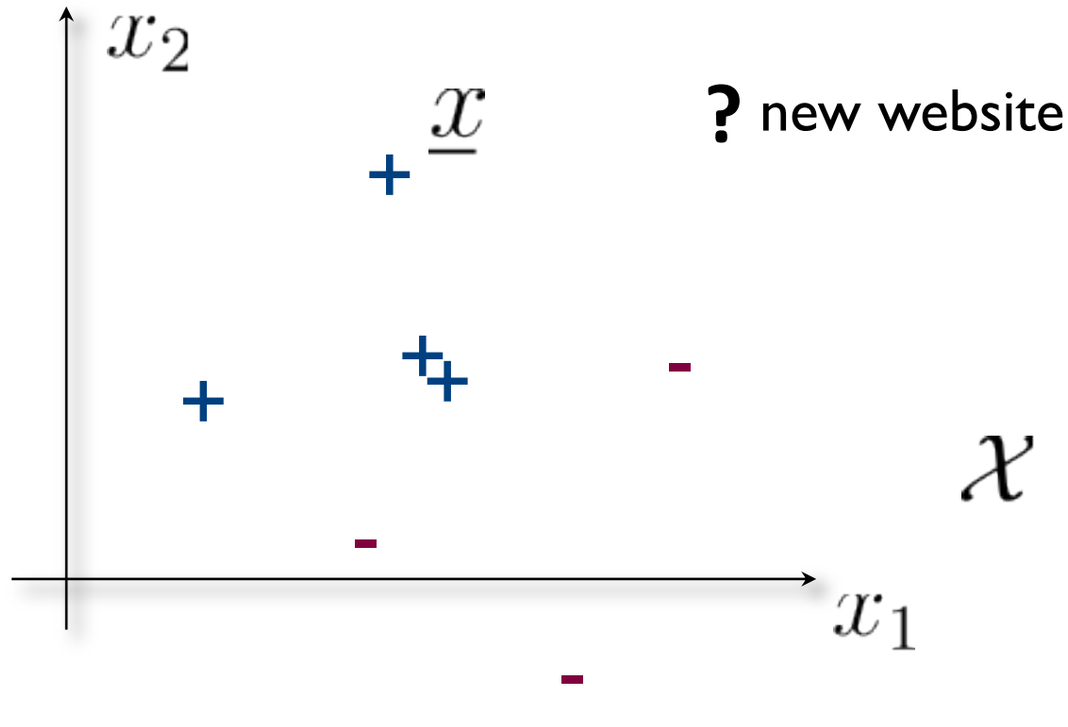
Part 1:
Model selection
what type of classifiers should we consider?



Part 2:
Estimation
how to select the best classifier of that type?

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

Part 1: allowing few classifiers?

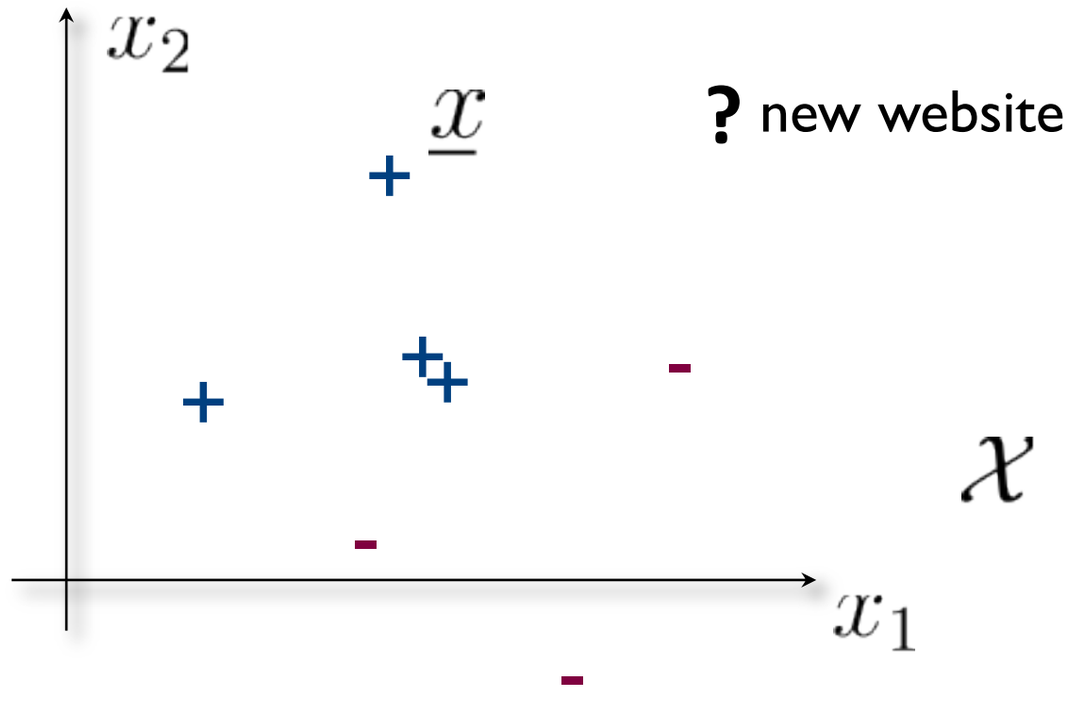


What about classifiers such as:

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

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

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

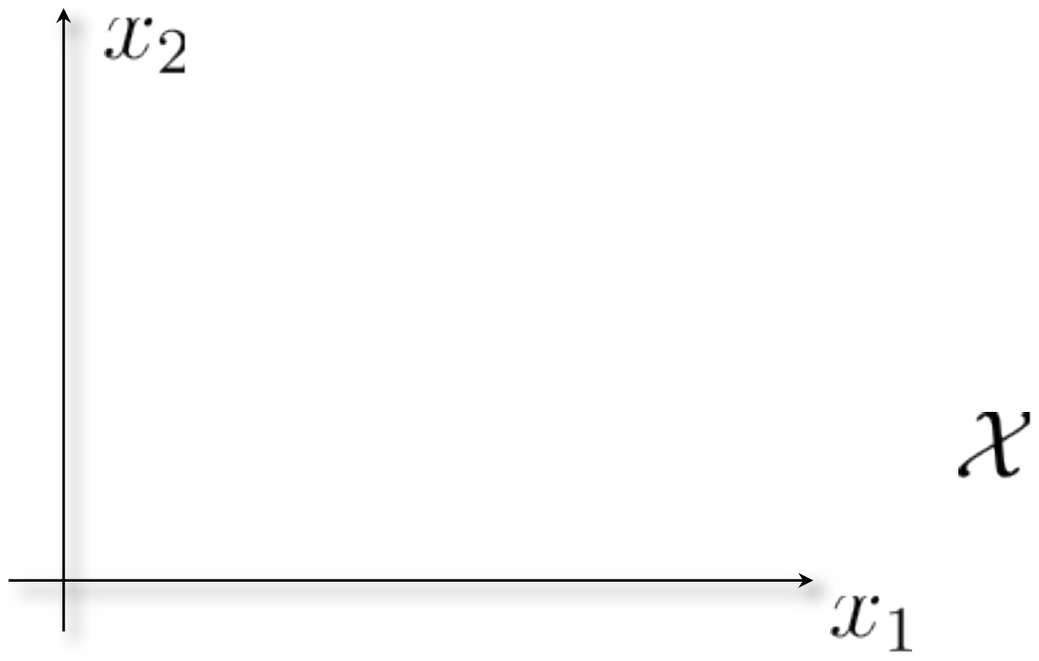
But it doesn't "generalize" (it doesn't classify new points well)

Linear classifiers

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

$$f(\underline{x};\underline{\theta}) = \text{sign}\left(\underline{\theta}\cdot\underline{x}\right) = \text{sign}\left(\theta_1 x_1 + \ldots + \theta_d x_d\right)$$

$$= \begin{cases} +1, & \text{if } \underline{\theta}\cdot\underline{x} > 0 \\ -1, & \text{if } \underline{\theta}\cdot\underline{x} \leq 0 \end{cases}$$

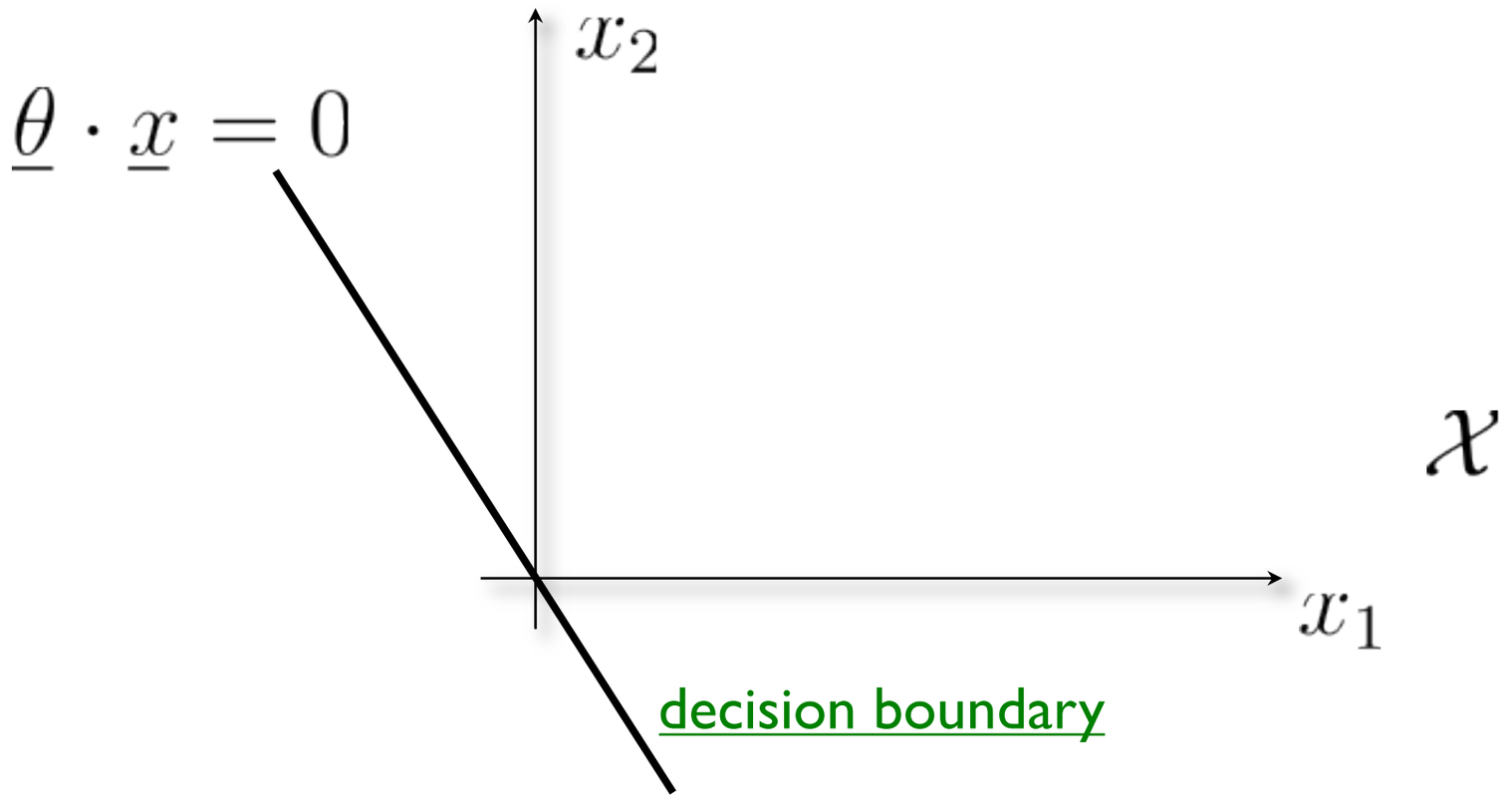


Linear classifiers

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

$$f(\underline{x};\underline{\theta}) = \text{sign}\left(\underline{\theta}\cdot\underline{x}\right) = \text{sign}\left(\theta_1 x_1 + \ldots + \theta_d x_d\right)$$

$$= \begin{cases} +1, & \text{if } \underline{\theta}\cdot\underline{x} > 0 \\ -1, & \text{if } \underline{\theta}\cdot\underline{x} \leq 0 \end{cases}$$

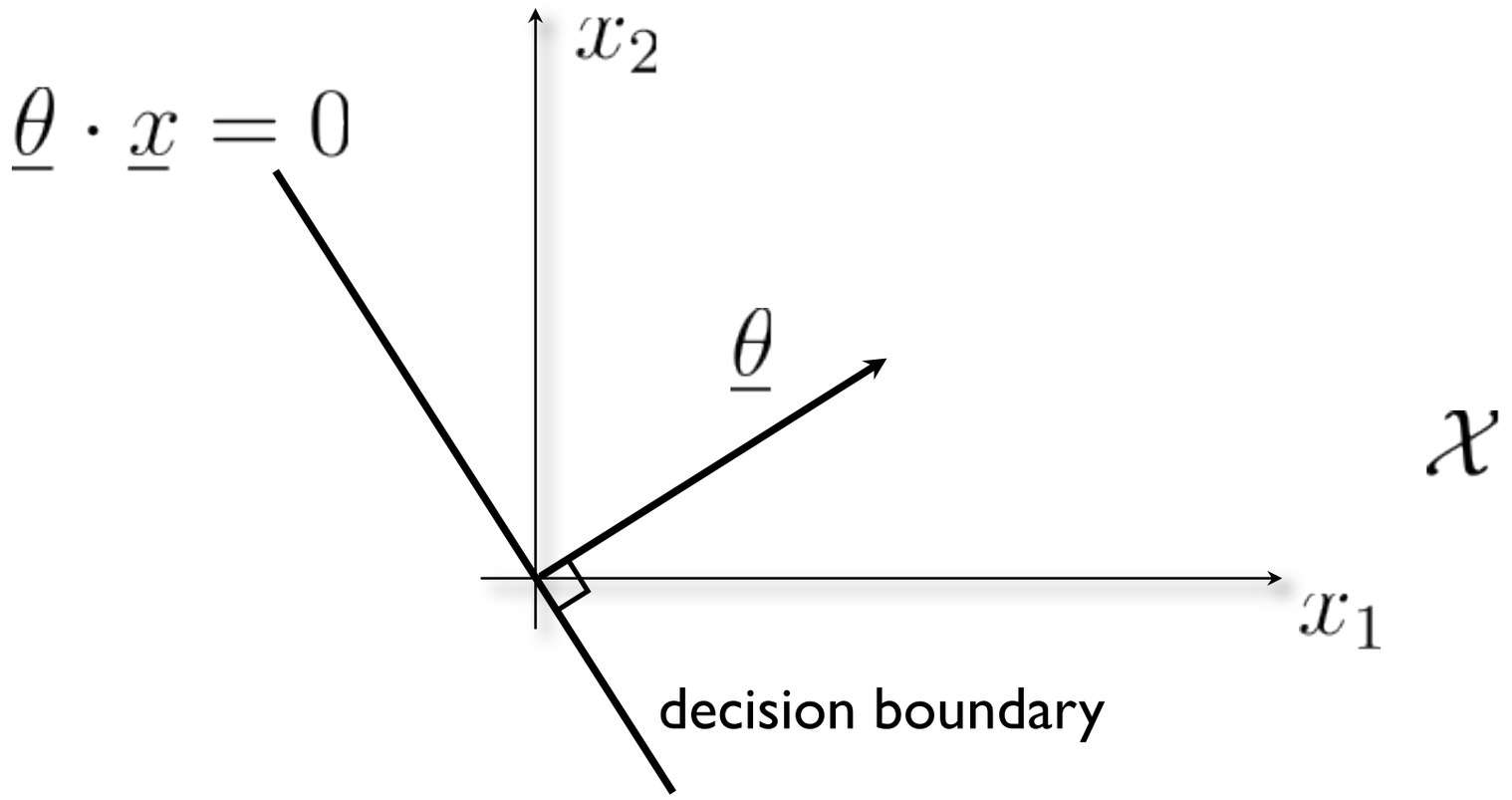


Linear classifiers

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

$$f(\underline{x};\underline{\theta}) = \text{sign}\left(\underline{\theta}\cdot\underline{x}\right) = \text{sign}\left(\theta_1 x_1 + \ldots + \theta_d x_d\right)$$

$$= \begin{cases} +1, & \text{if } \underline{\theta}\cdot\underline{x} > 0 \\ -1, & \text{if } \underline{\theta}\cdot\underline{x} \leq 0 \end{cases}$$

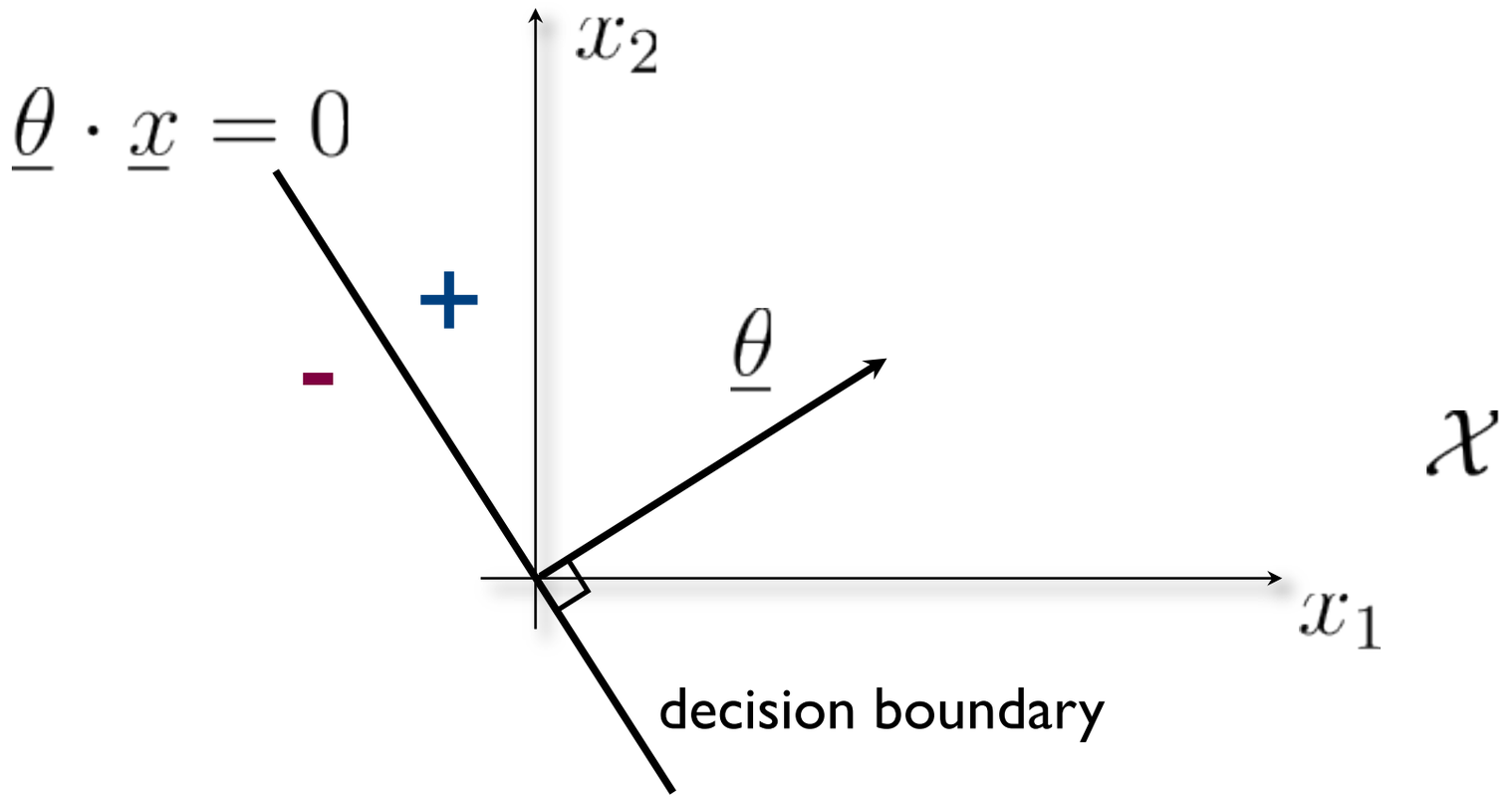


Linear classifiers

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

$$f(\underline{x};\underline{\theta}) = \text{sign}\left(\underline{\theta}\cdot\underline{x}\right) = \text{sign}\left(\theta_1 x_1 + \ldots + \theta_d x_d\right)$$

$$= \begin{cases} +1, & \text{if } \underline{\theta}\cdot\underline{x} > 0 \\ -1, & \text{if } \underline{\theta}\cdot\underline{x} \leq 0 \end{cases}$$



(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.

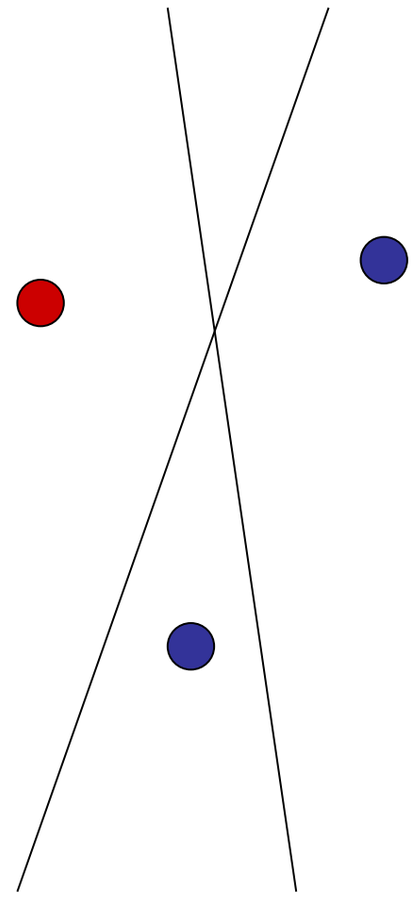
and the stored pattern. According to this hypothesis, if one understood the code or "wiring diagram" of the nervous system, one should, in principle, be able to discover exactly what an organism remembers by reconstructing the original sensory patterns from the "memory traces" which they have left, much 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.

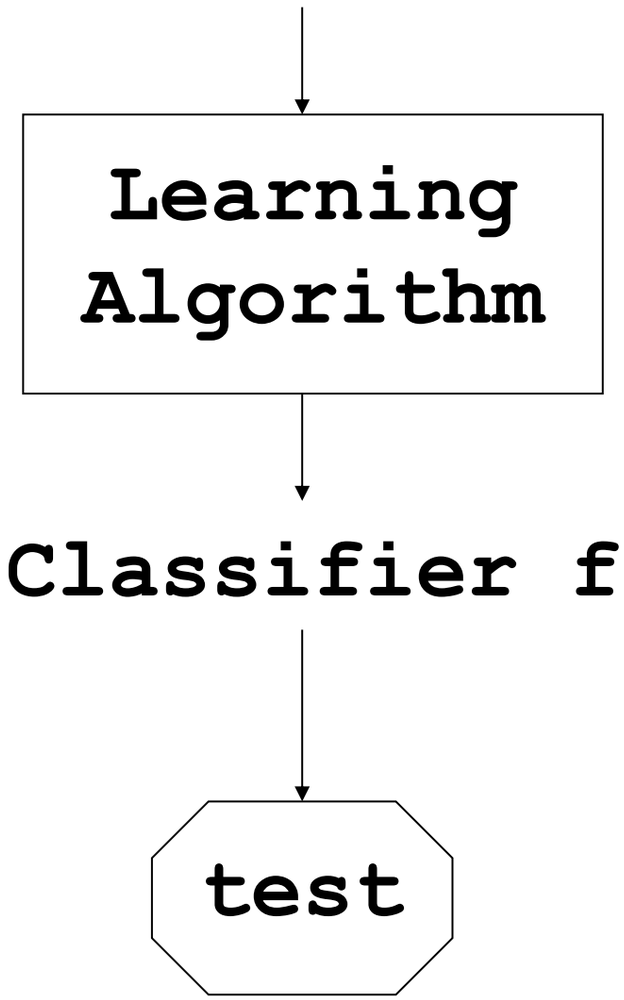
Can also cycle through a batch of data this way.

Learn a linear classifier:

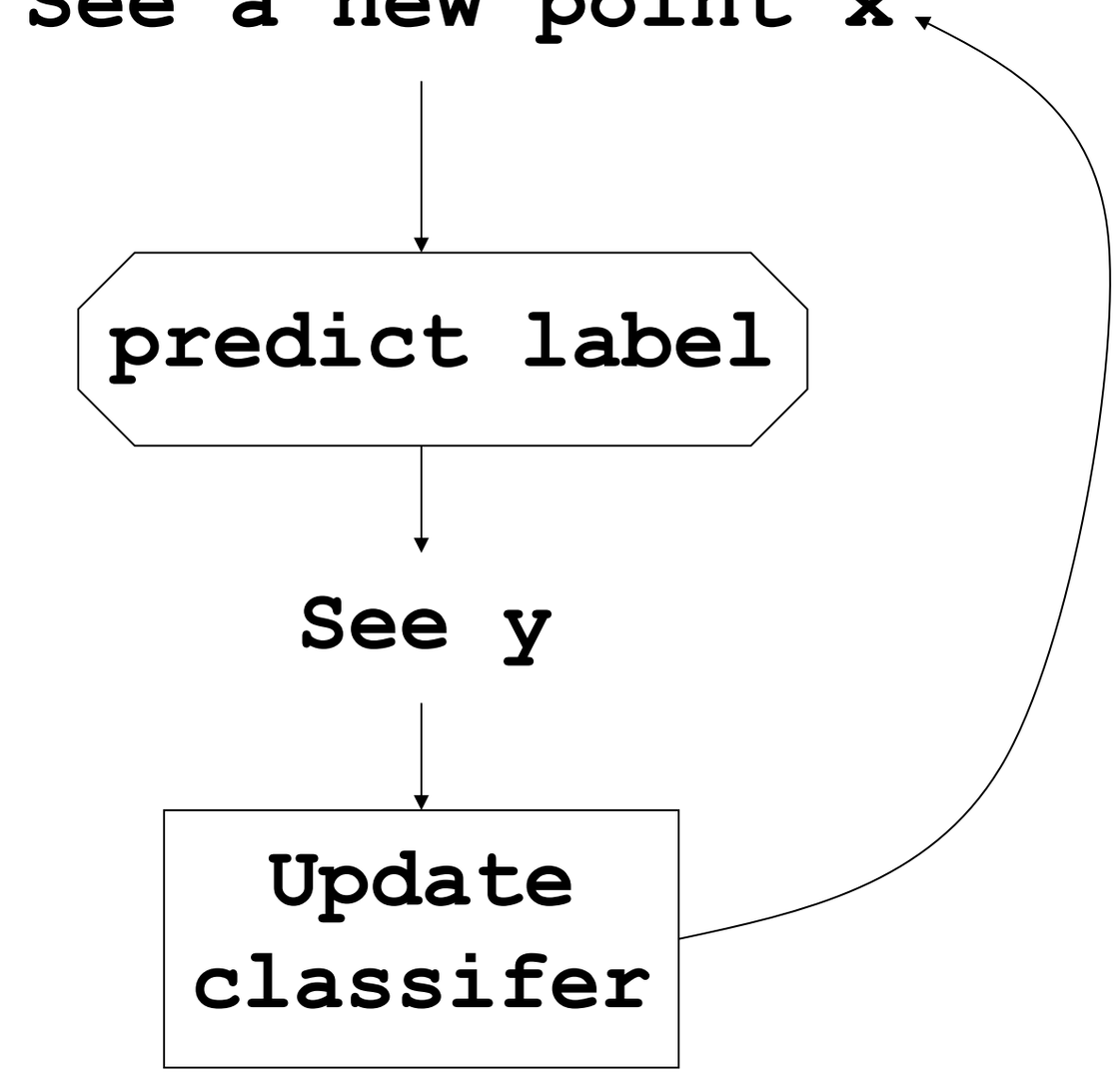


Batch vs. Online Learning

Batch learning



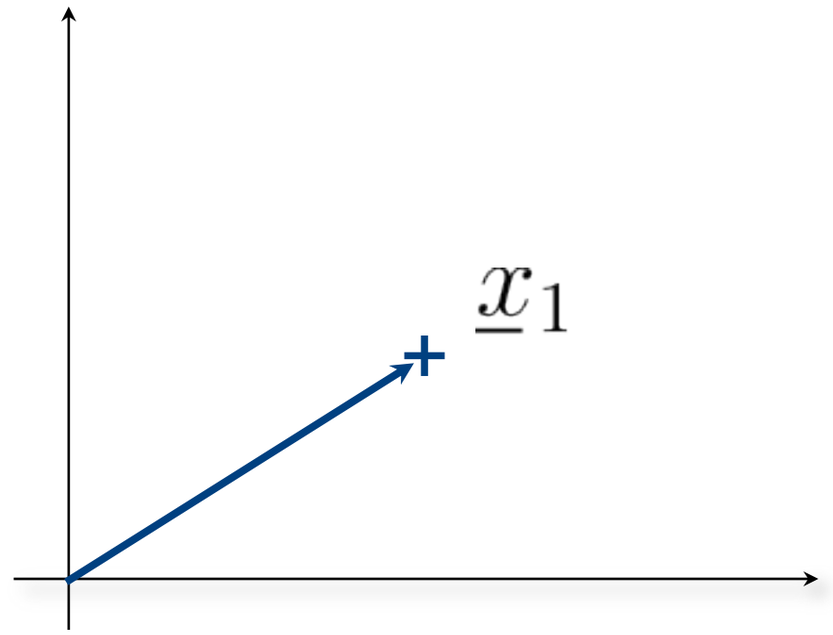
Online learning



Perceptron algorithm

- Iterative updates based on mistakes

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

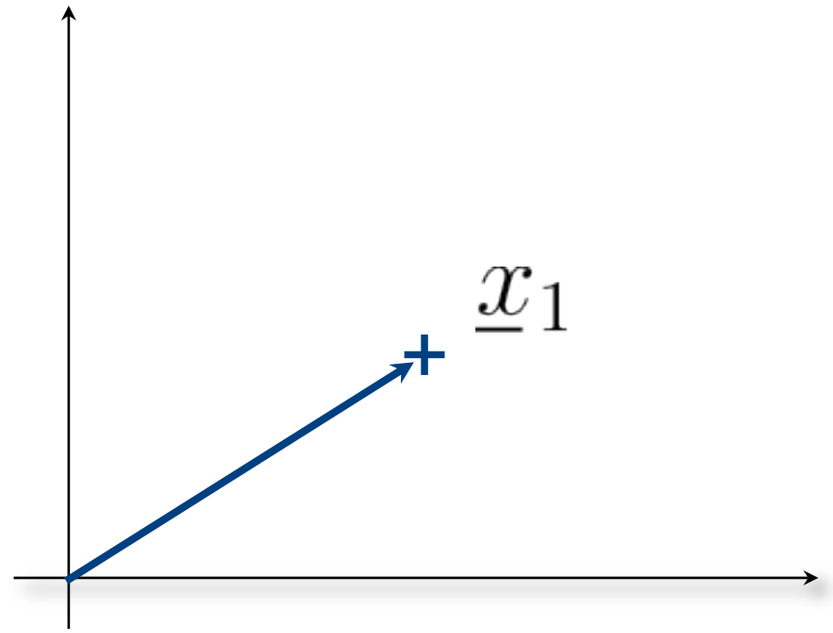


Perceptron algorithm

- Iterative updates based on mistakes

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

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

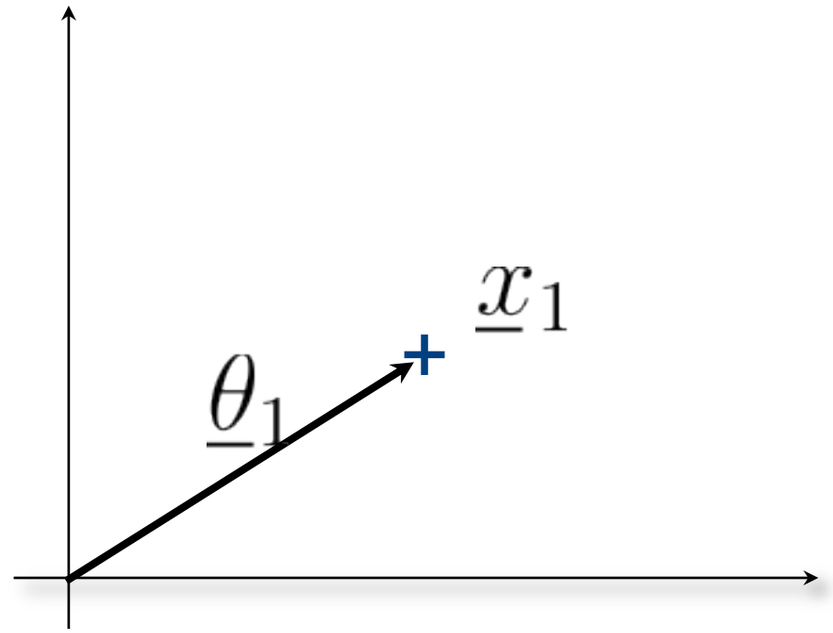


Perceptron algorithm

- Iterative updates based on mistakes

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

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

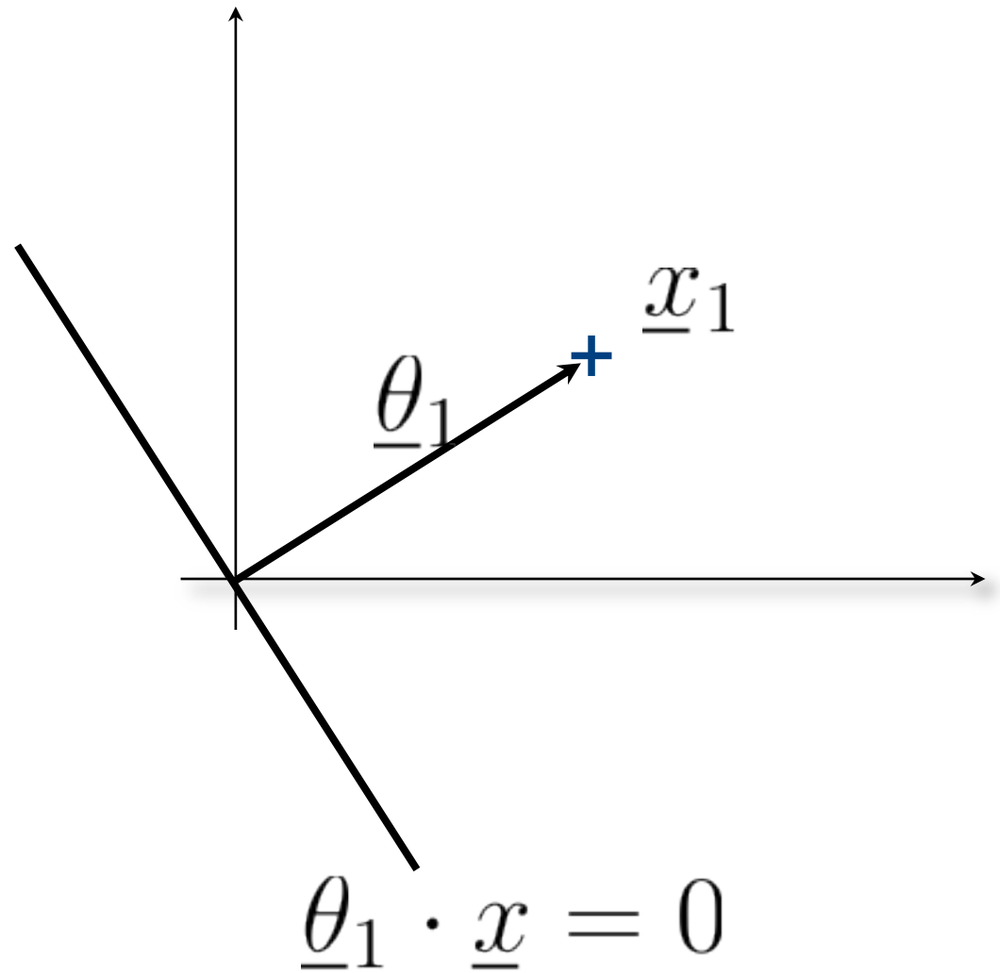


Perceptron algorithm

- Iterative updates based on mistakes

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

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

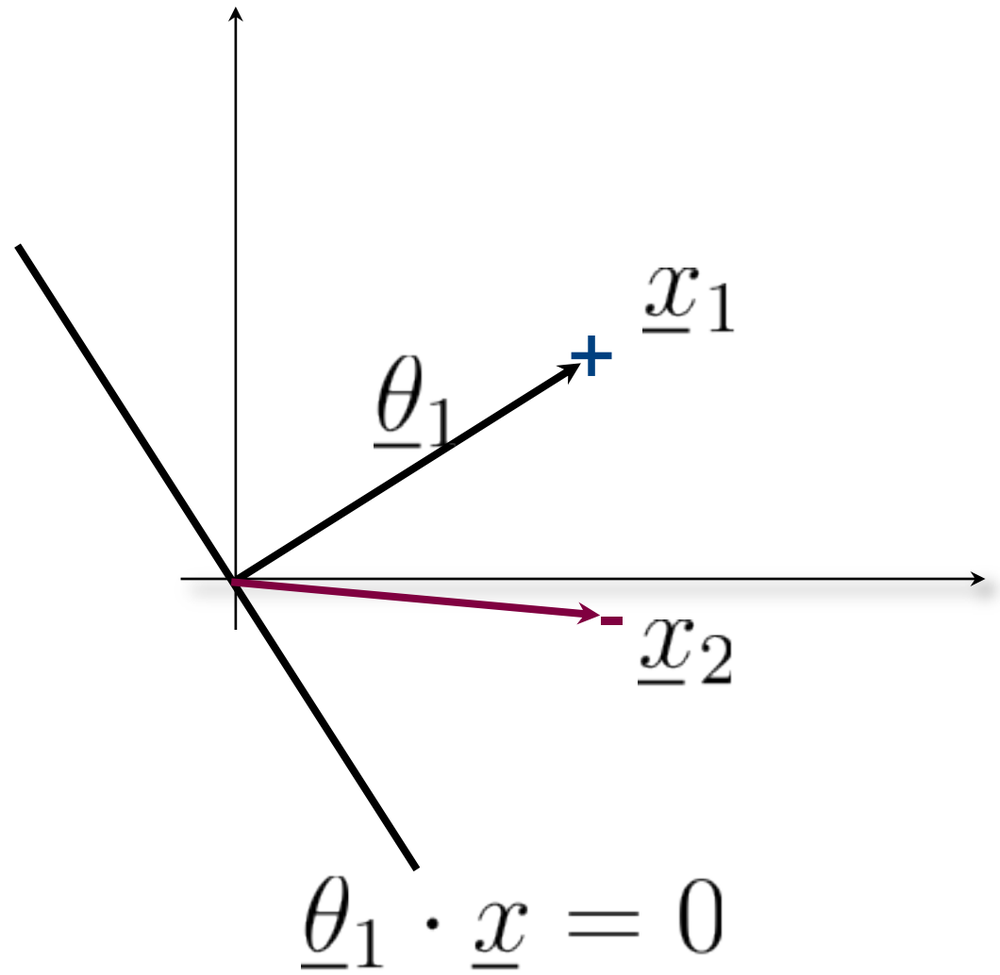


Perceptron algorithm

- Iterative updates based on mistakes

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

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



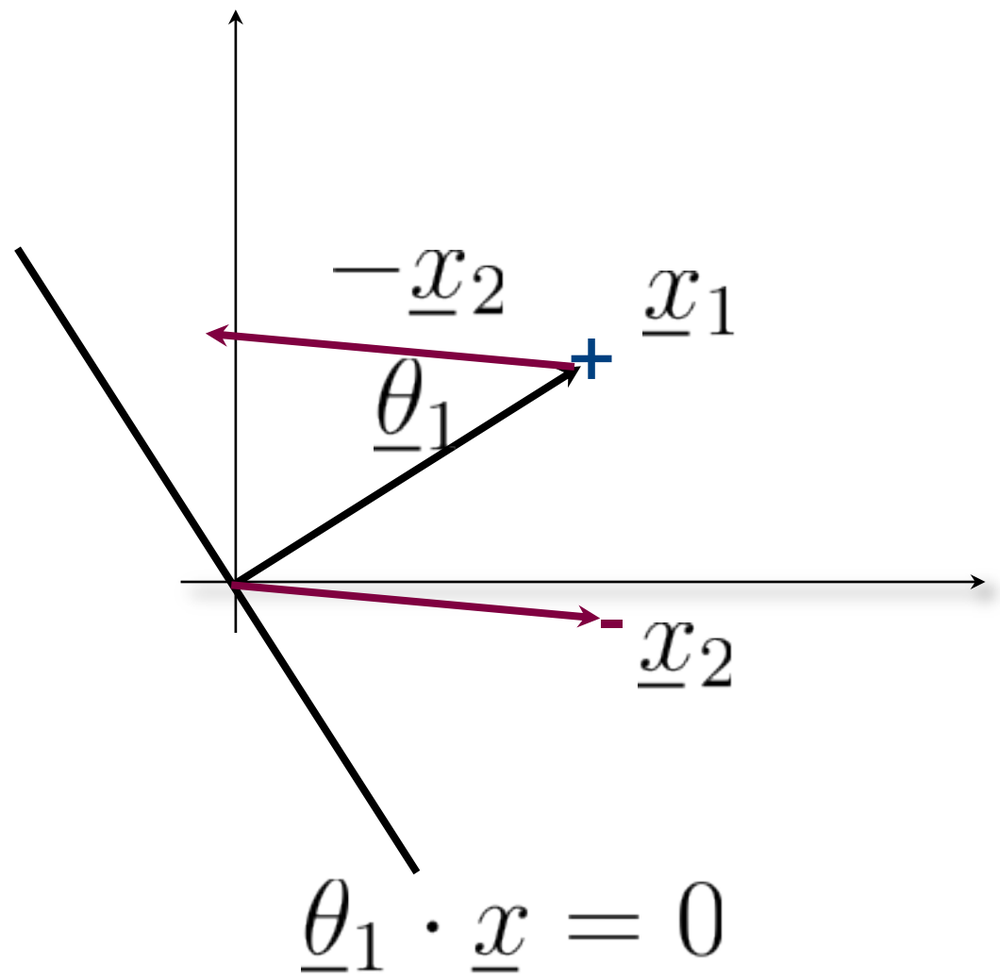
Perceptron algorithm

- Iterative updates based on mistakes

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

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

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



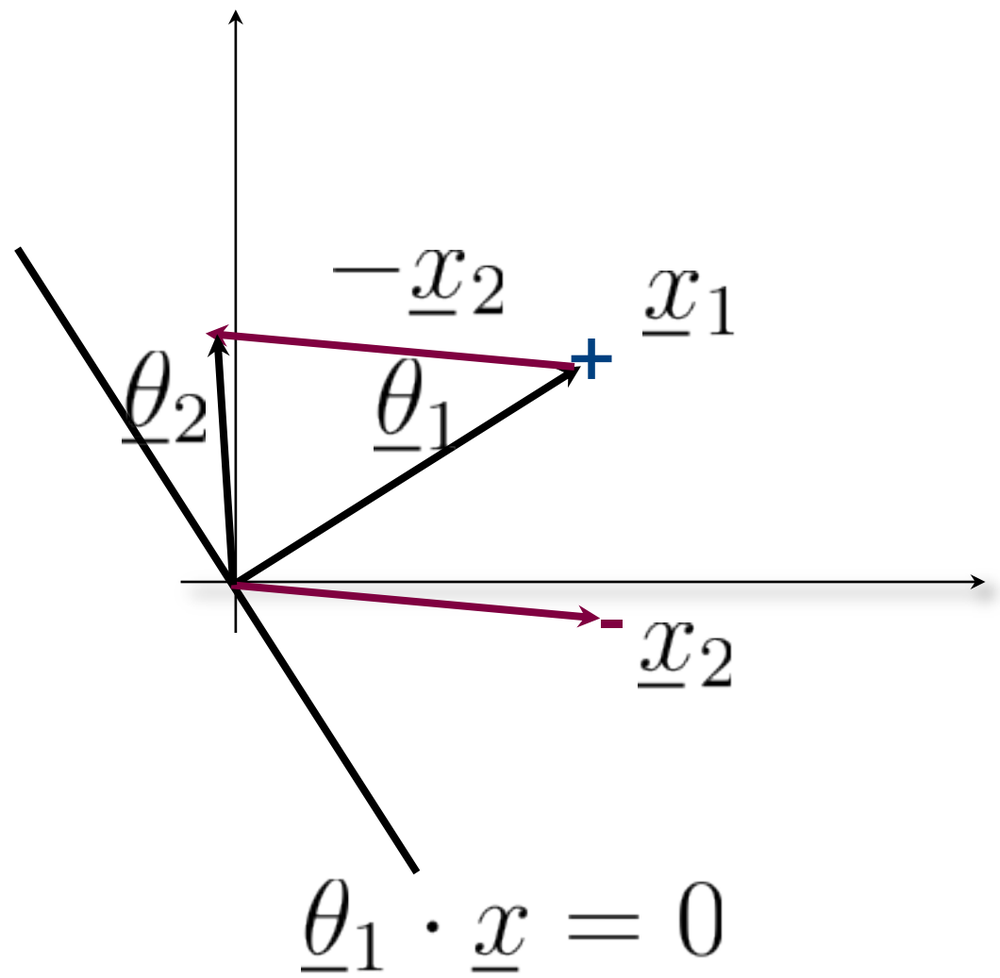
Perceptron algorithm

- Iterative updates based on mistakes

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

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

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



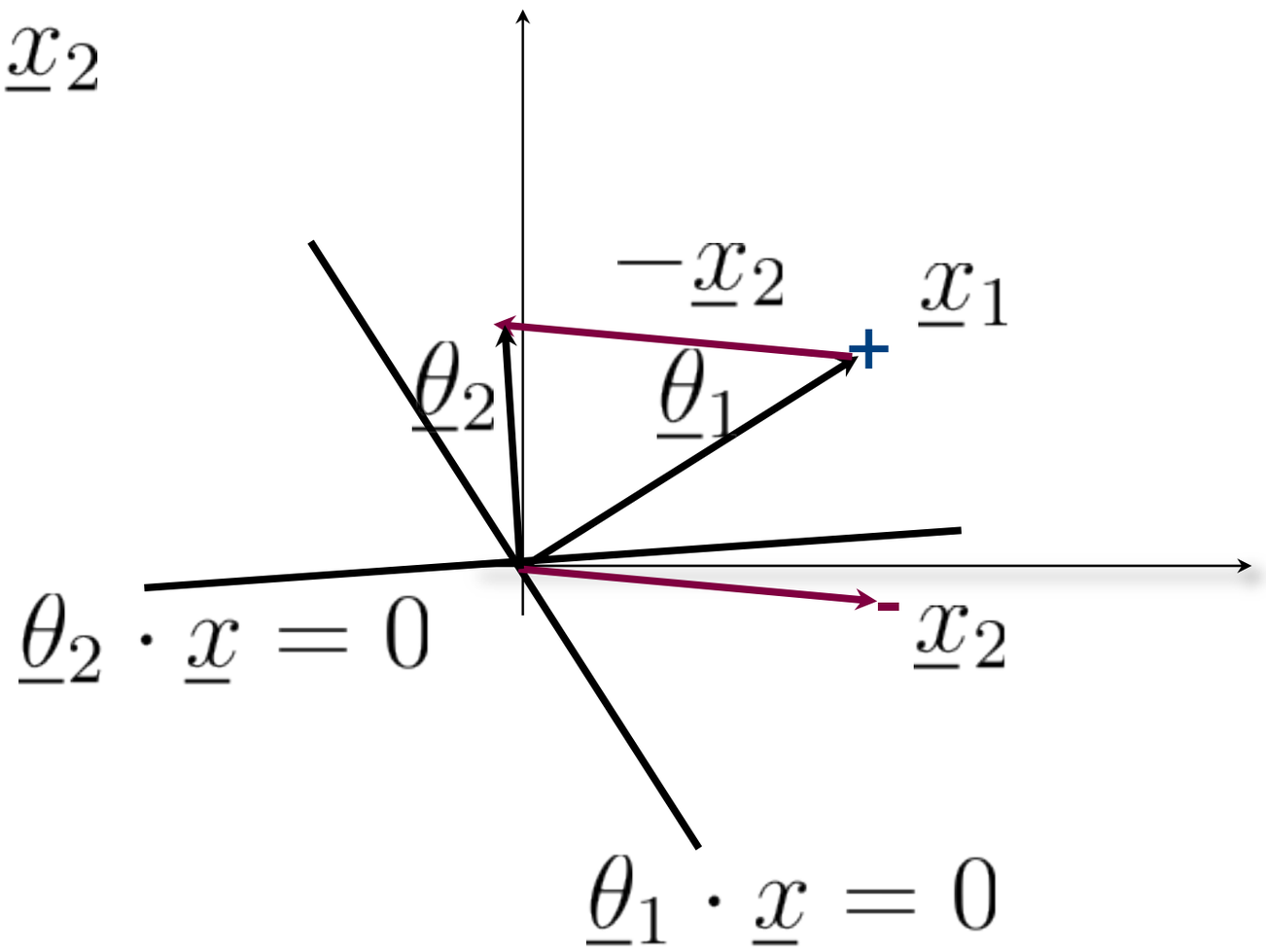
Perceptron algorithm

- Iterative updates based on mistakes

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

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

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



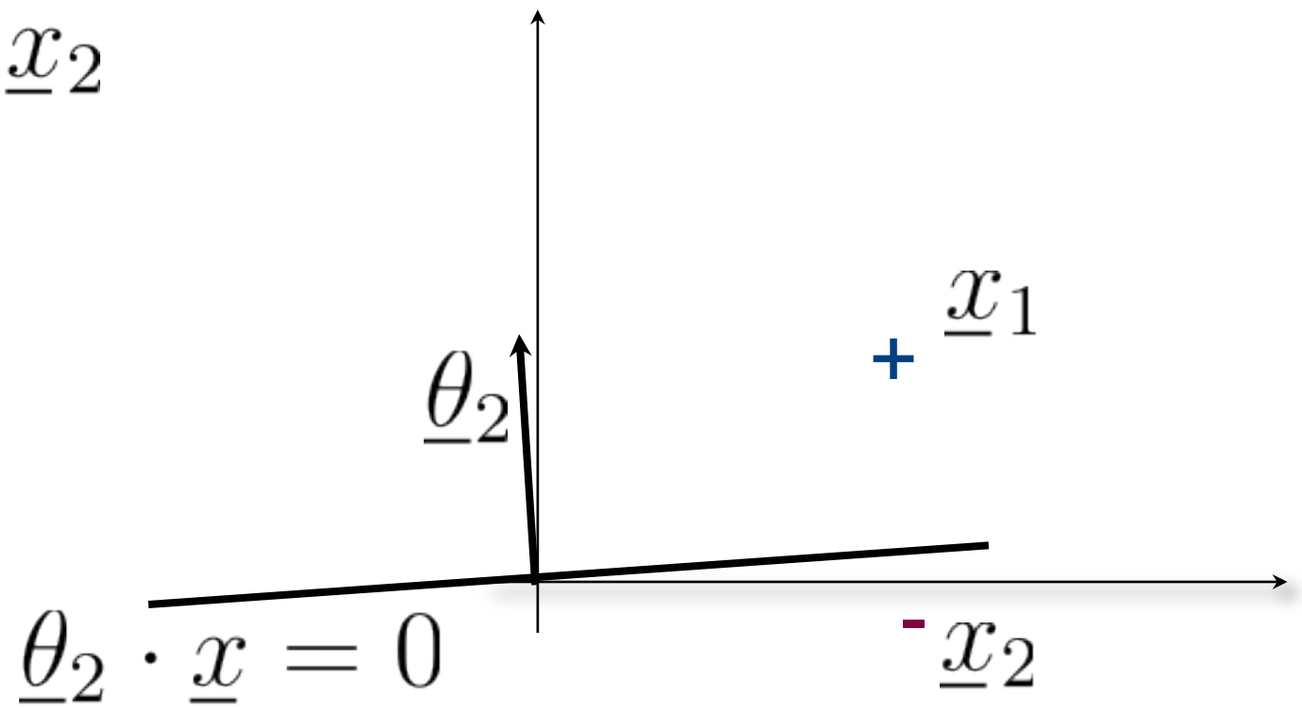
Perceptron algorithm

- Iterative updates based on mistakes

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

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

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

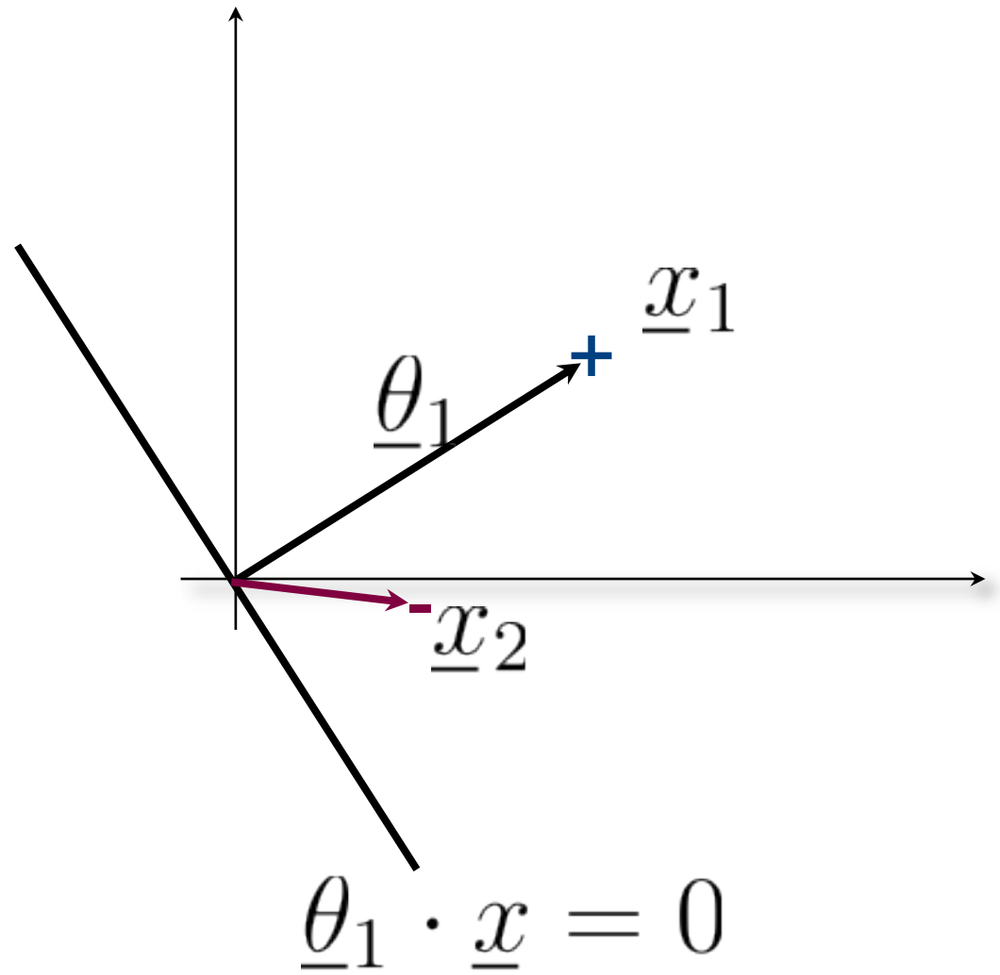


Perceptron algorithm (take 2)

- Iterative updates based on mistakes

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

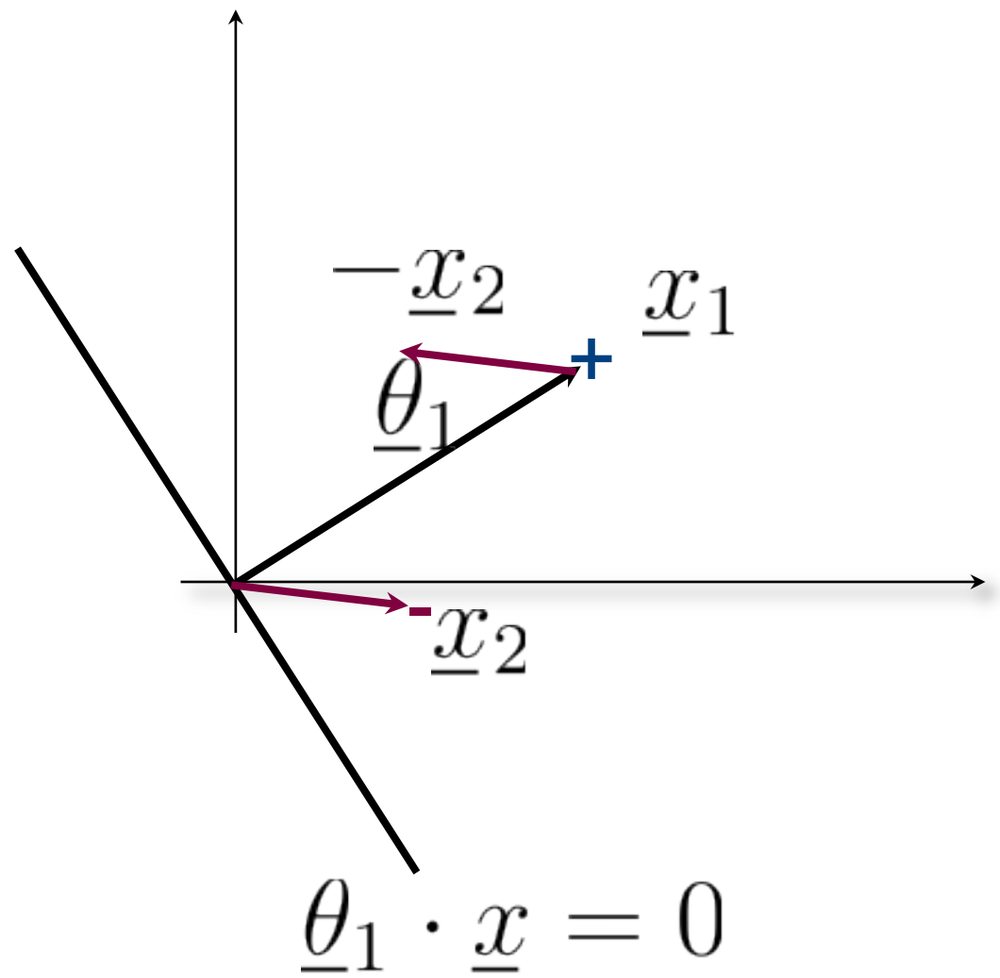
$$\underline{\theta}_1 = \underline{\theta}_0 + 1\,\underline{x}_1$$



Perceptron algorithm (take 2)

- Iterative updates based on mistakes

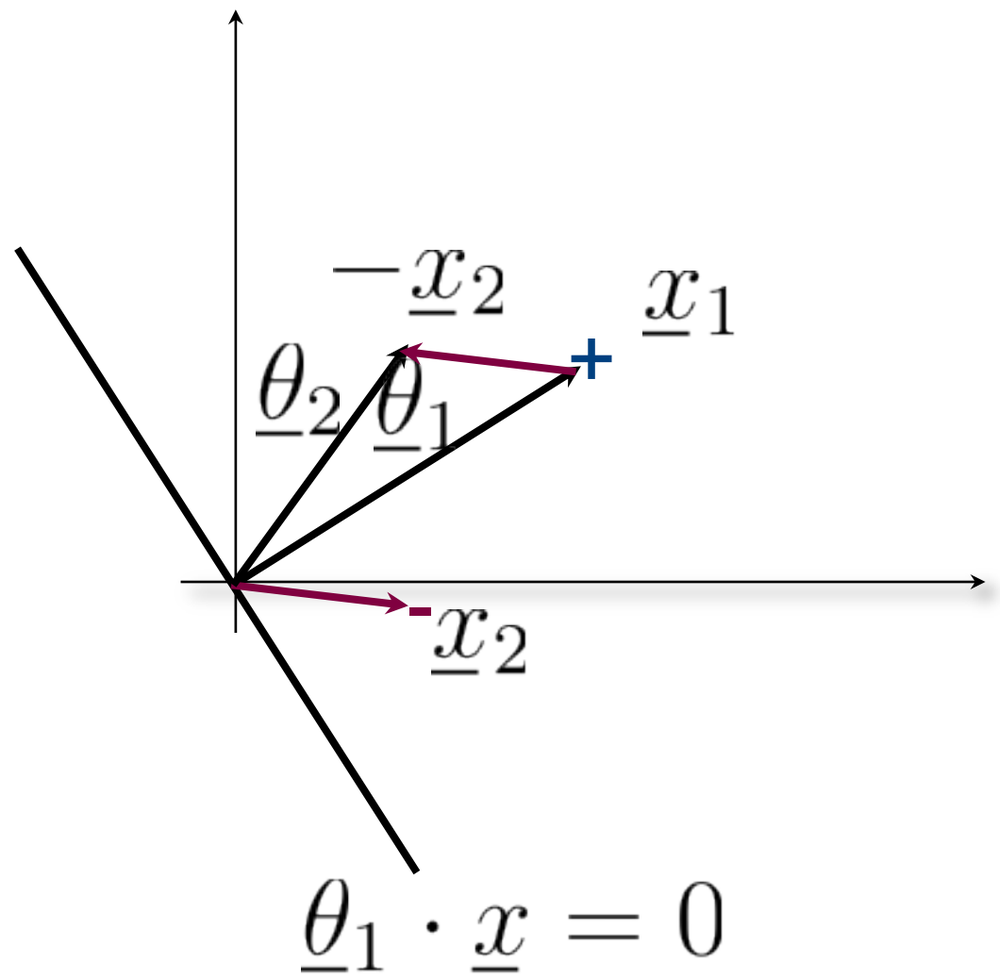
$$\underline{\theta}_0 = 0$$
$$\underline{\theta}_1 = \underline{\theta}_0 + 1\,\underline{x}_1$$
$$\underline{\theta}_2 = \underline{\theta}_1 + (-1)\,\underline{x}_2$$



Perceptron algorithm (take 2)

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$
$$\underline{\theta}_1 = \underline{\theta}_0 + 1\,\underline{x}_1$$
$$\underline{\theta}_2 = \underline{\theta}_1 + (-1)\,\underline{x}_2$$



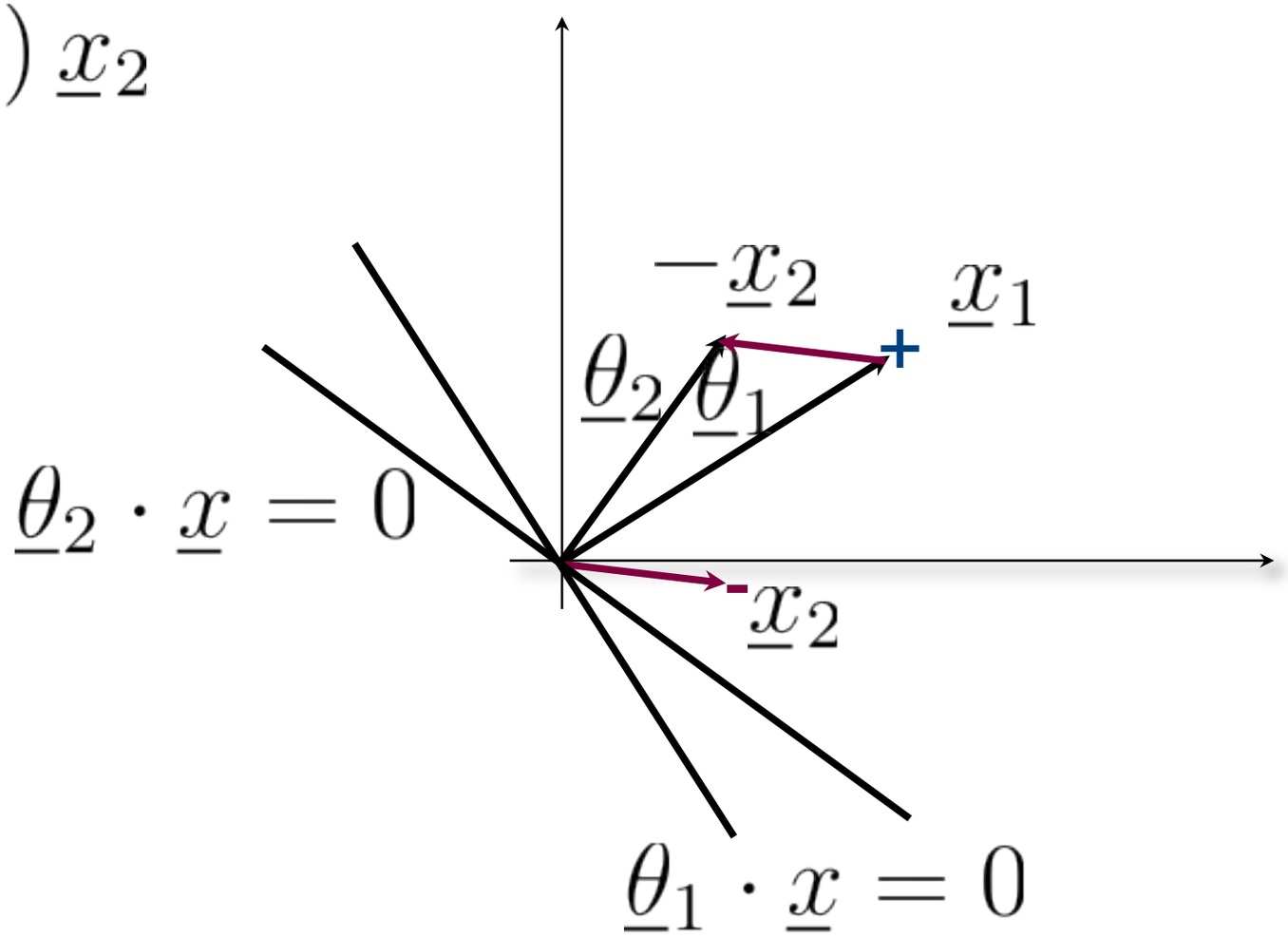
Perceptron algorithm (take 2)

- Iterative updates based on mistakes

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

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

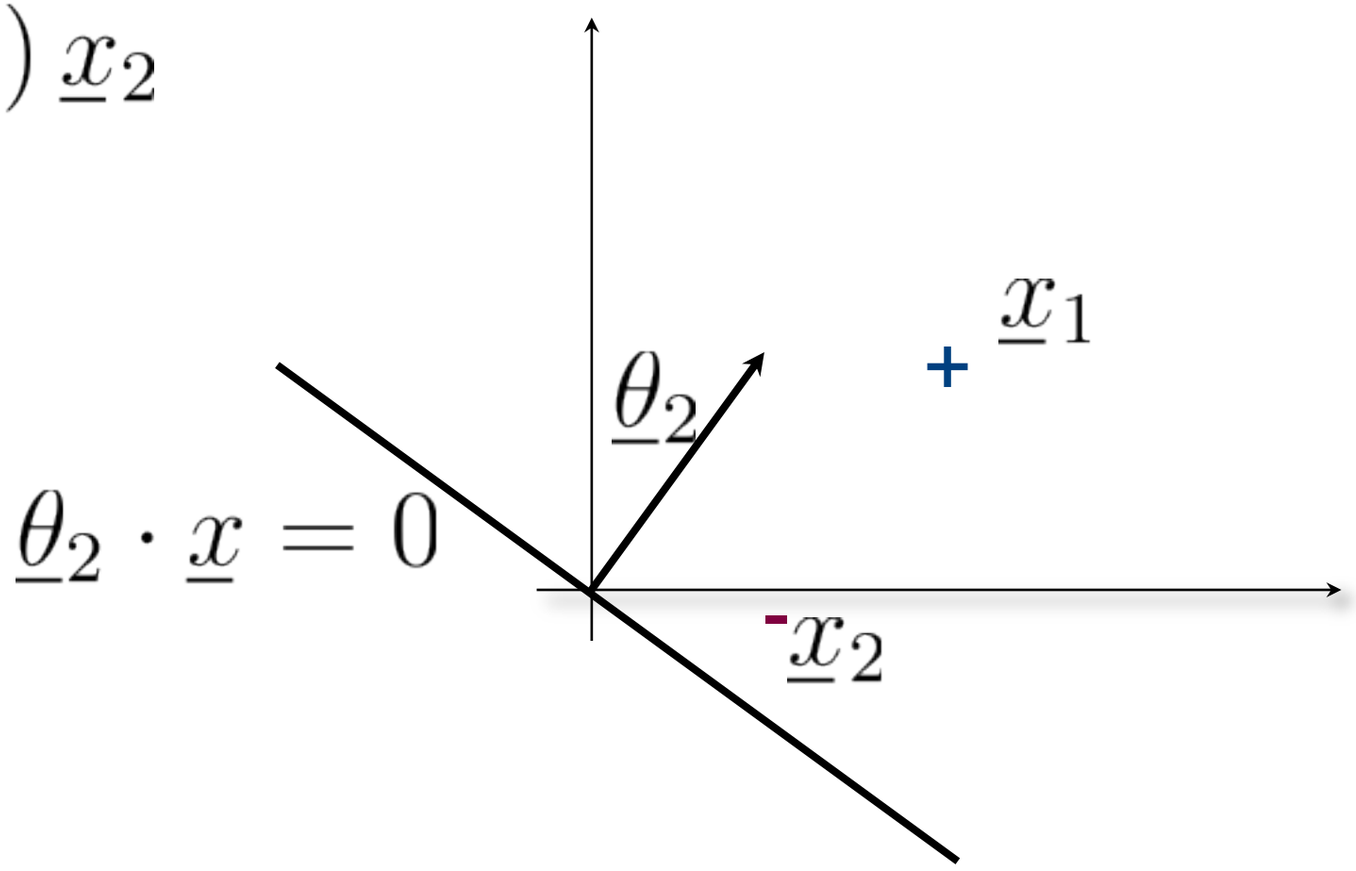
$$\underline{\theta}_2 = \underline{\theta}_1 + (-1)\,\underline{x}_2$$



Perceptron algorithm (take 2)

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$
$$\underline{\theta}_1 = \underline{\theta}_0 + 1\,\underline{x}_1$$
$$\underline{\theta}_2 = \underline{\theta}_1 + (-1)\,\underline{x}_2$$

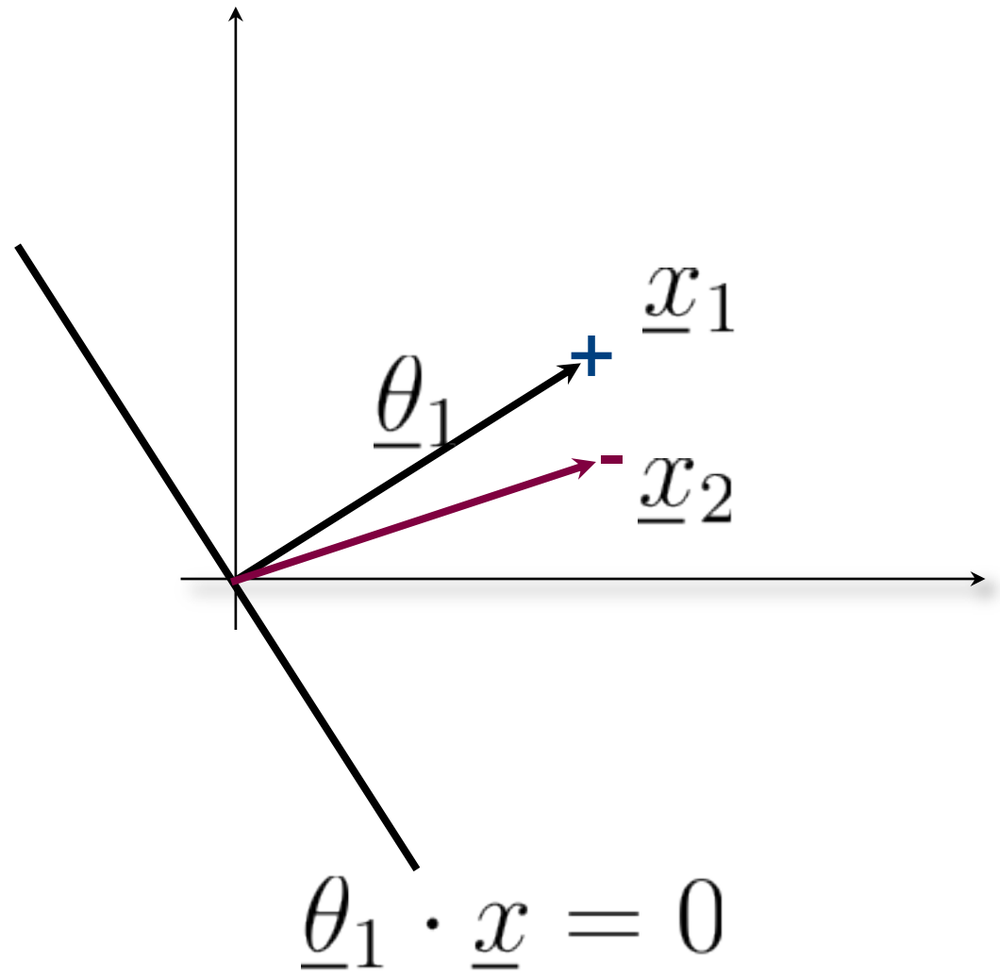


Perceptron algorithm (take 3)

- Iterative updates based on mistakes

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

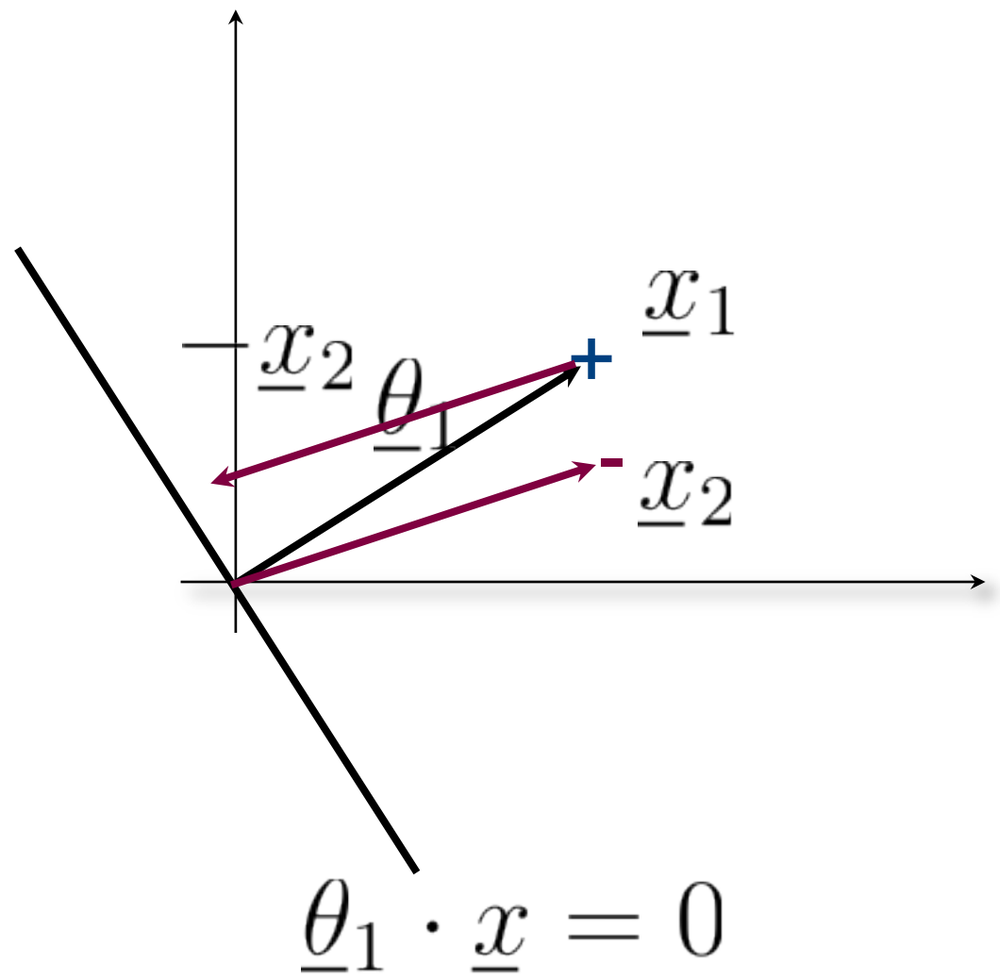
$$\underline{\theta}_1 = \underline{\theta}_0 + 1\,\underline{x}_1$$



Perceptron algorithm (take 3)

- Iterative updates based on mistakes

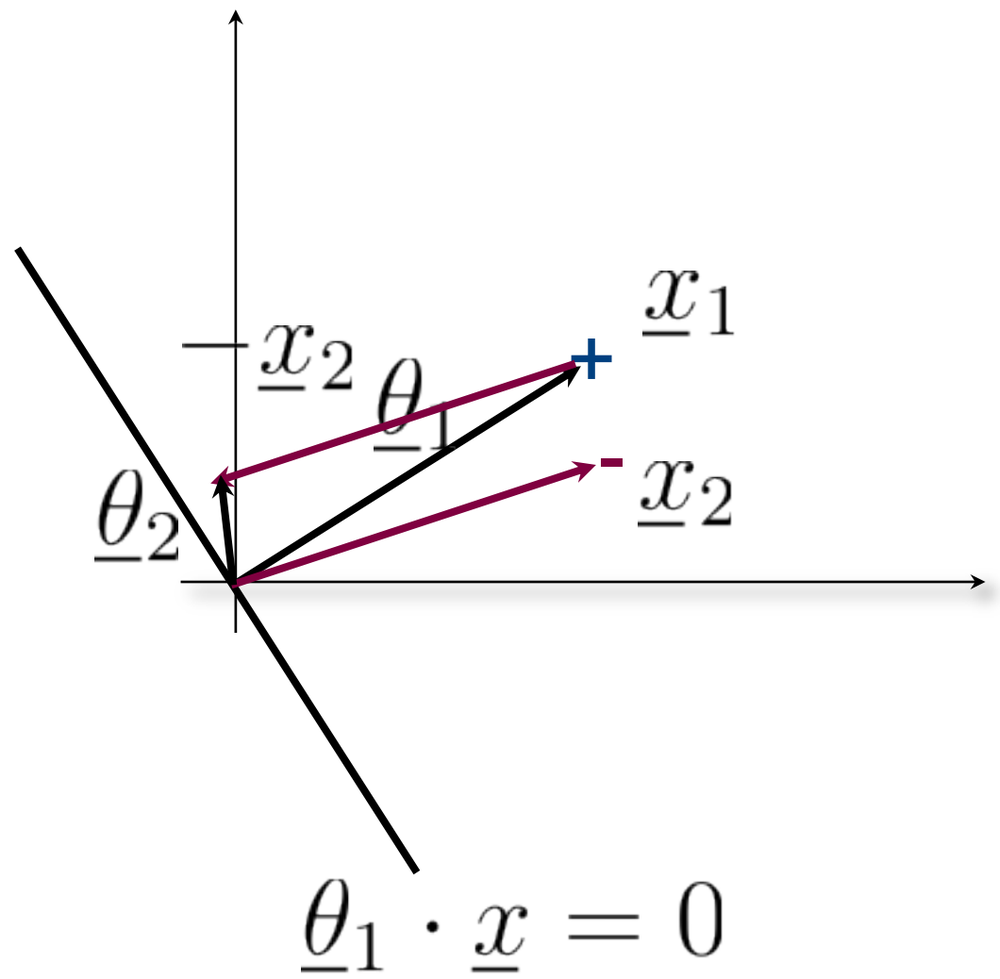
$$\underline{\theta}_0 = 0$$
$$\underline{\theta}_1 = \underline{\theta}_0 + 1\,\underline{x}_1$$
$$\underline{\theta}_2 = \underline{\theta}_1 + (-1)\,\underline{x}_2$$



Perceptron algorithm (take 3)

- Iterative updates based on mistakes

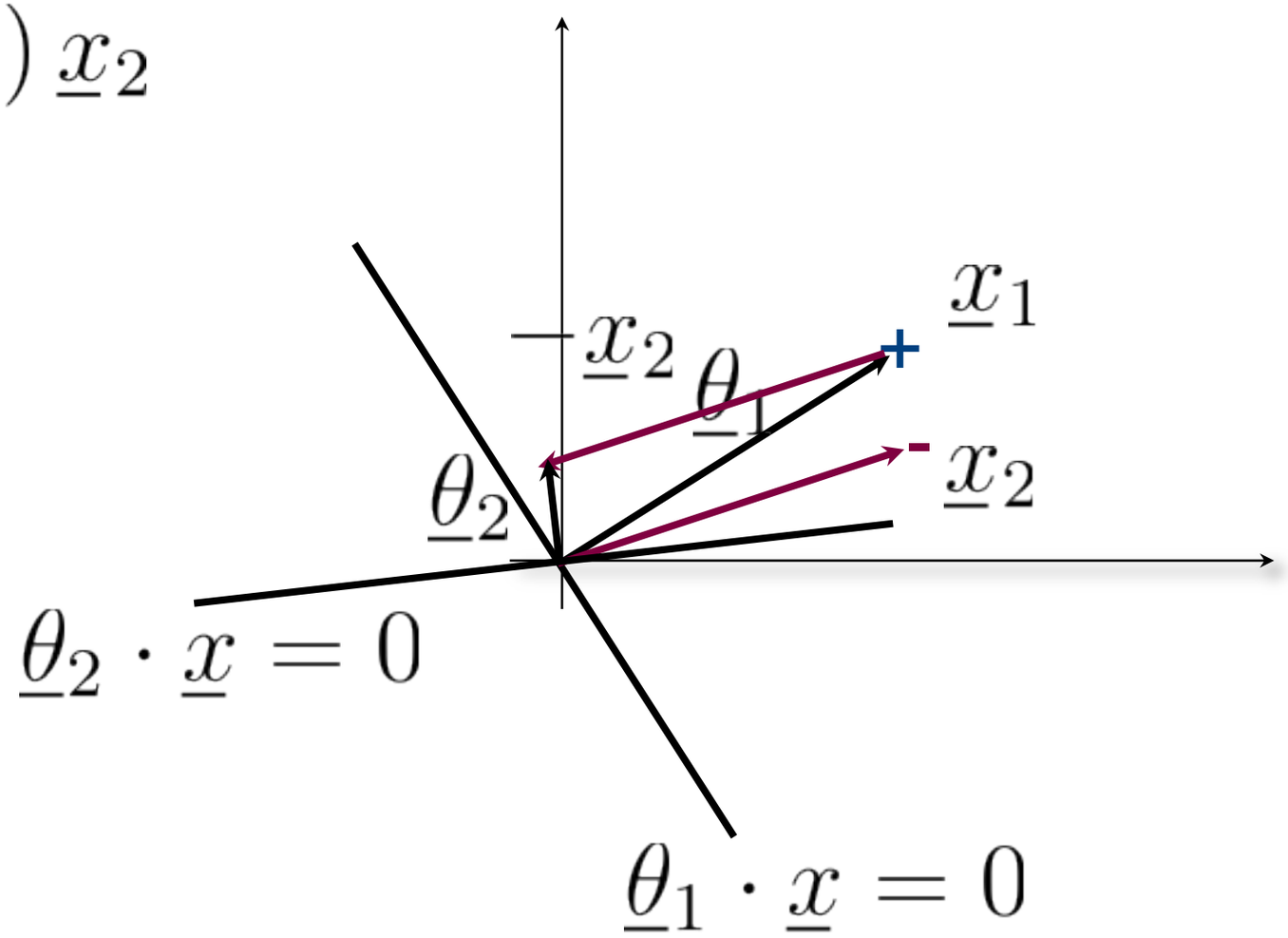
$$\underline{\theta}_0 = 0$$
$$\underline{\theta}_1 = \underline{\theta}_0 + 1\,\underline{x}_1$$
$$\underline{\theta}_2 = \underline{\theta}_1 + (-1)\,\underline{x}_2$$



Perceptron algorithm (take 3)

- Iterative updates based on mistakes

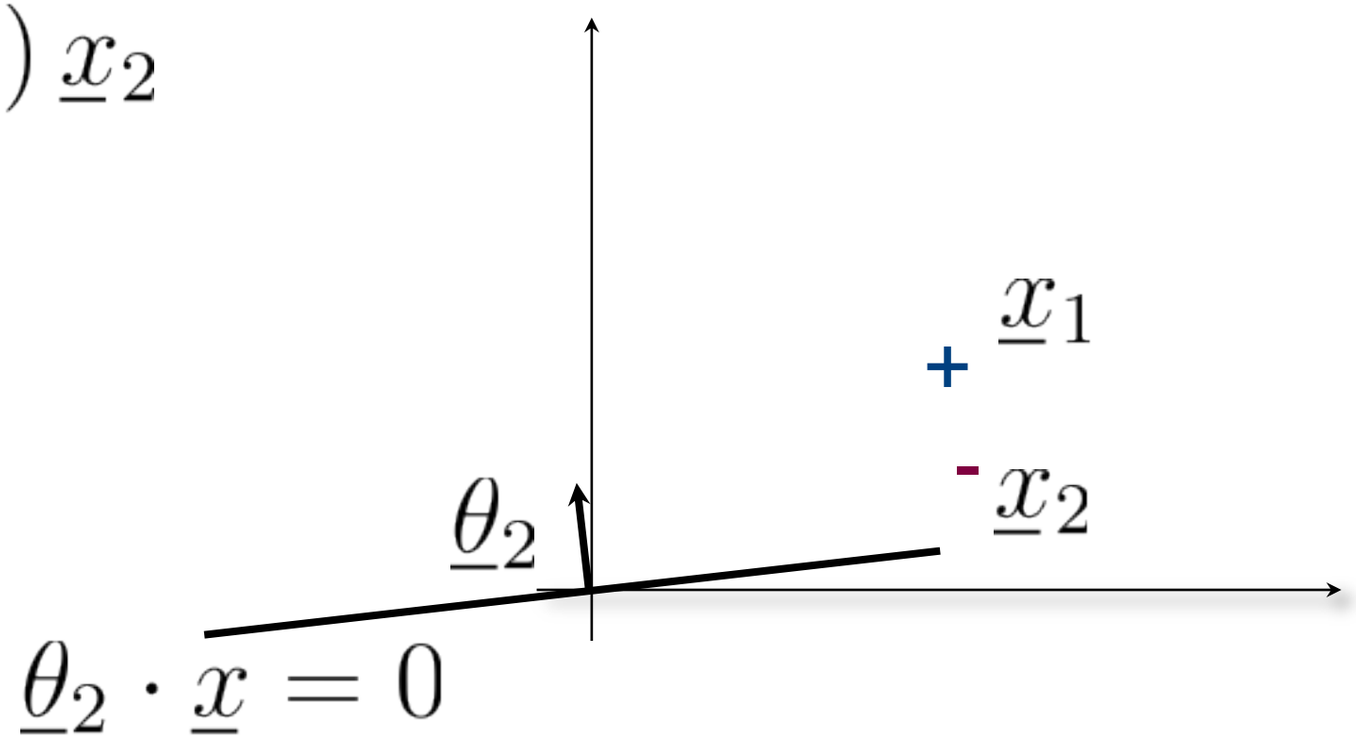
$$\underline{\theta}_0 = 0$$
$$\underline{\theta}_1 = \underline{\theta}_0 + 1\,\underline{x}_1$$
$$\underline{\theta}_2 = \underline{\theta}_1 + (-1)\,\underline{x}_2$$



Perceptron algorithm (take 3)

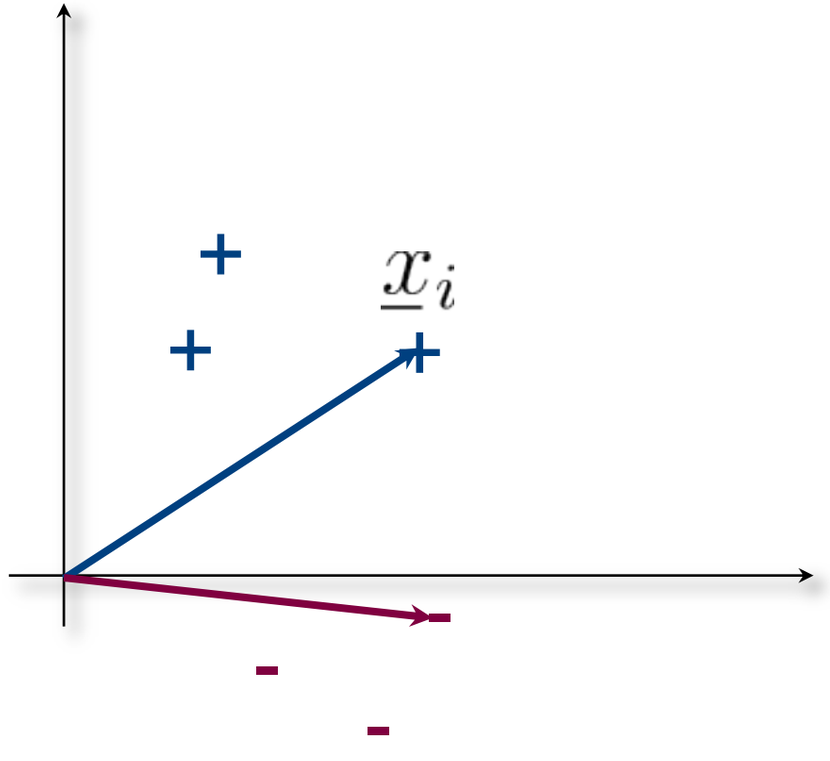
- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$
$$\underline{\theta}_1 = \underline{\theta}_0 + 1\,\underline{x}_1$$
$$\underline{\theta}_2 = \underline{\theta}_1 + (-1)\,\underline{x}_2$$



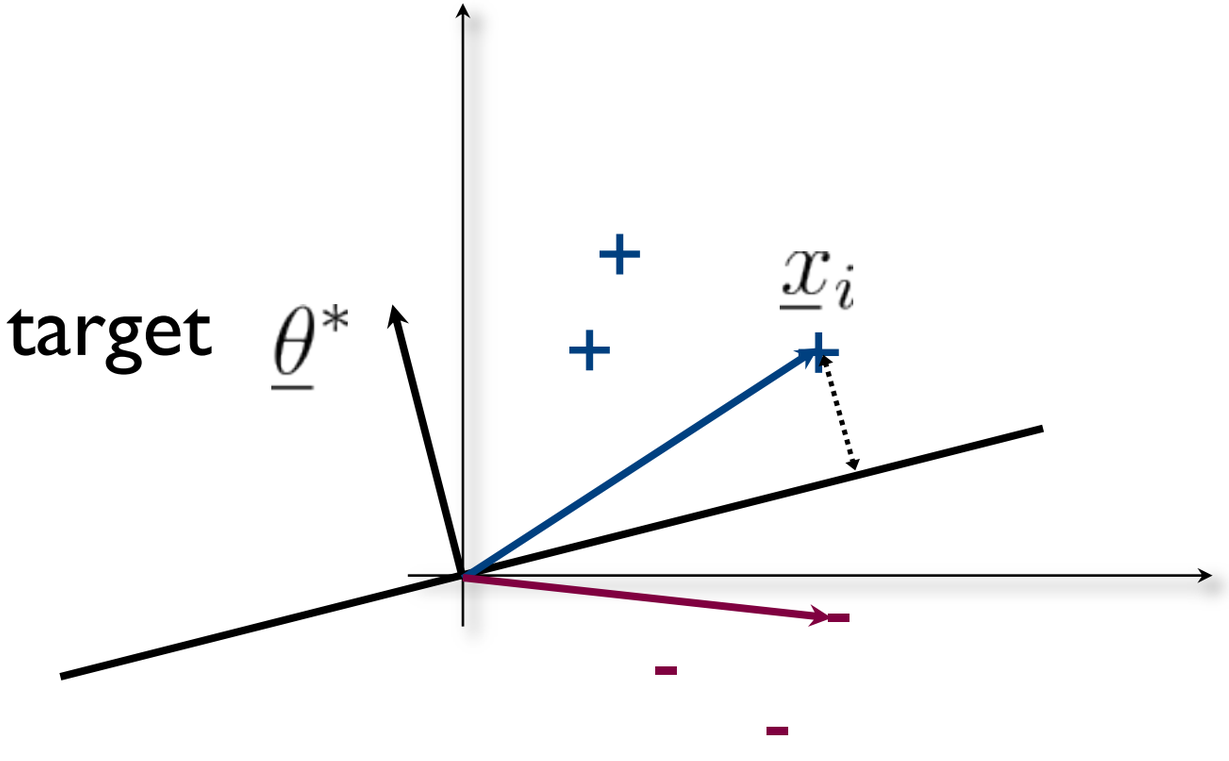
“Margin”

- 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



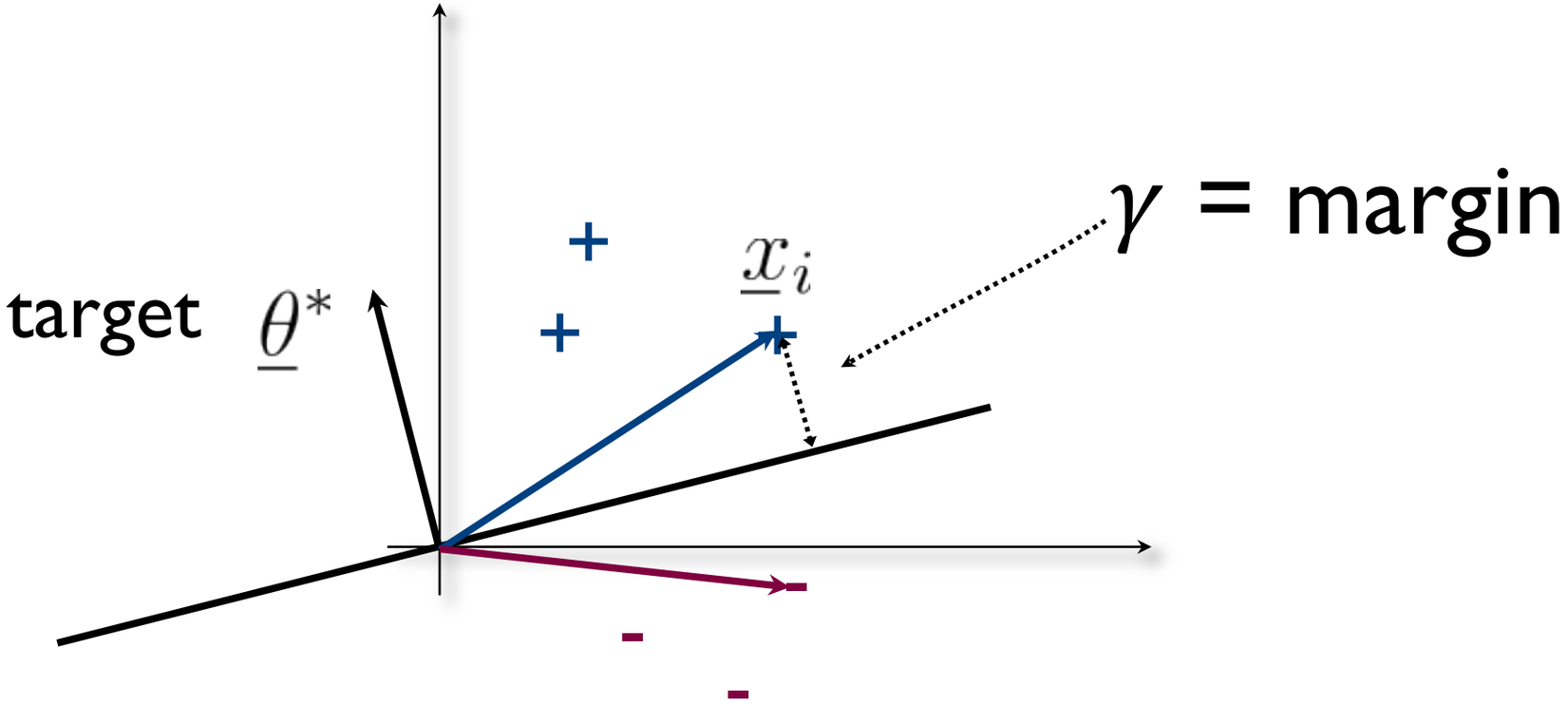
“Margin”

- We can get a handle on convergence by assuming that there exists a target classifier 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 θ^* 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:

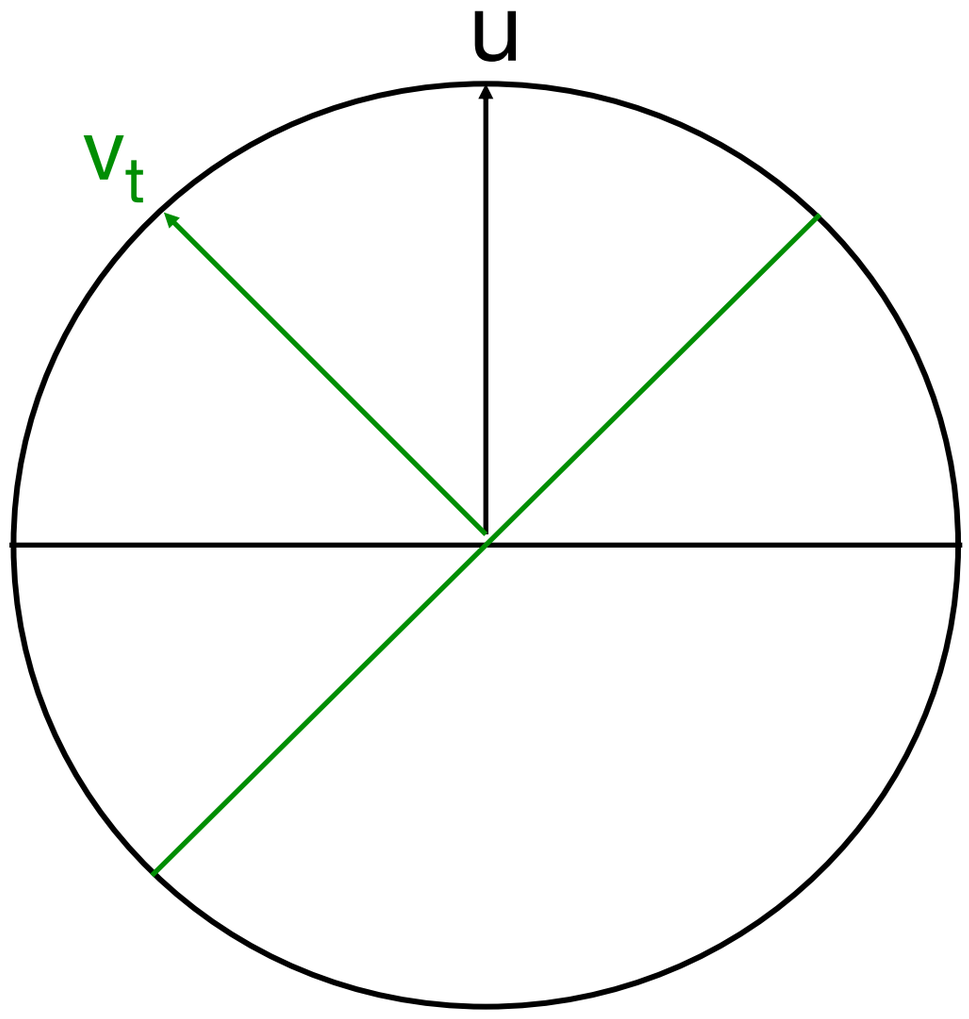
$$u : y_t(u \cdot x_t) > 0 \quad \forall t, \quad \|u\| = 1$$

Assume u is through origin.

→ Always possible, by increasing dimension by 1.

Current hypothesis: v_t

(Called θ on previous slides)



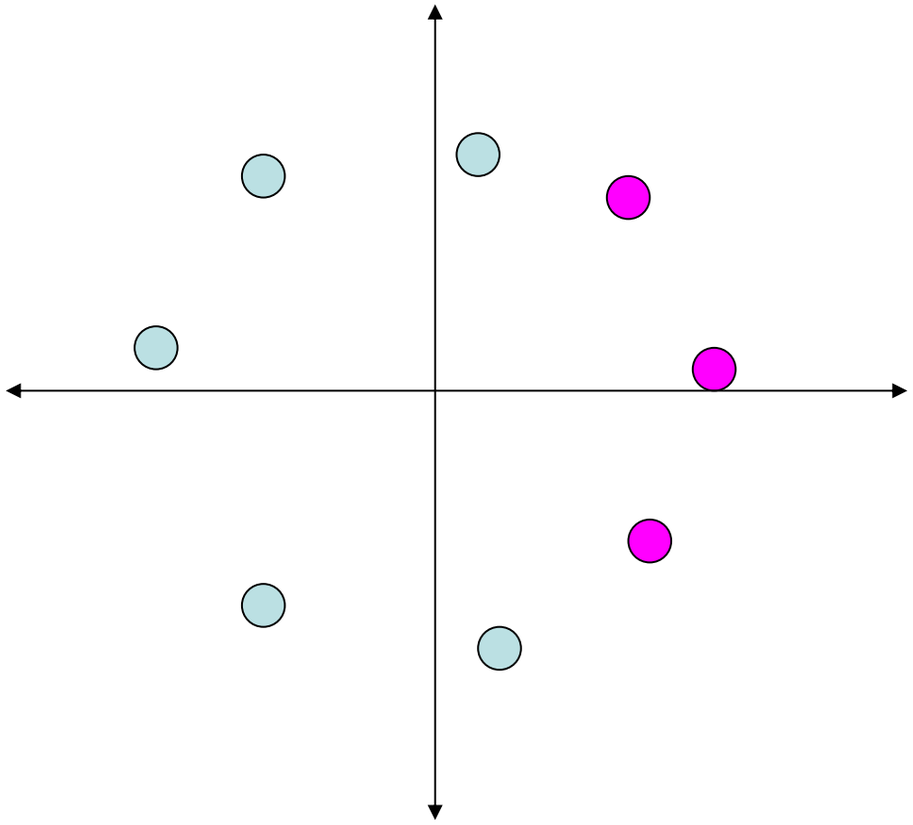
Preprocessing step

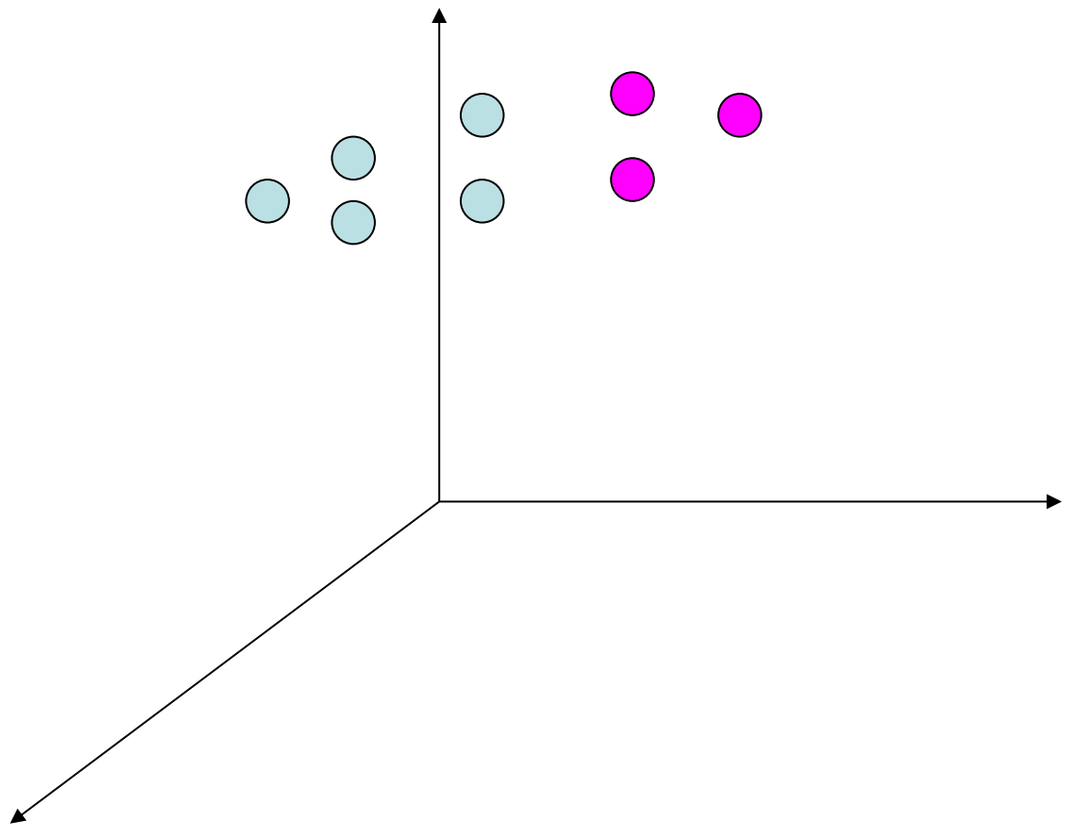
Points (x,y), where input space $X = R^d$, label space $Y = \{+1,-1\}$

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

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

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





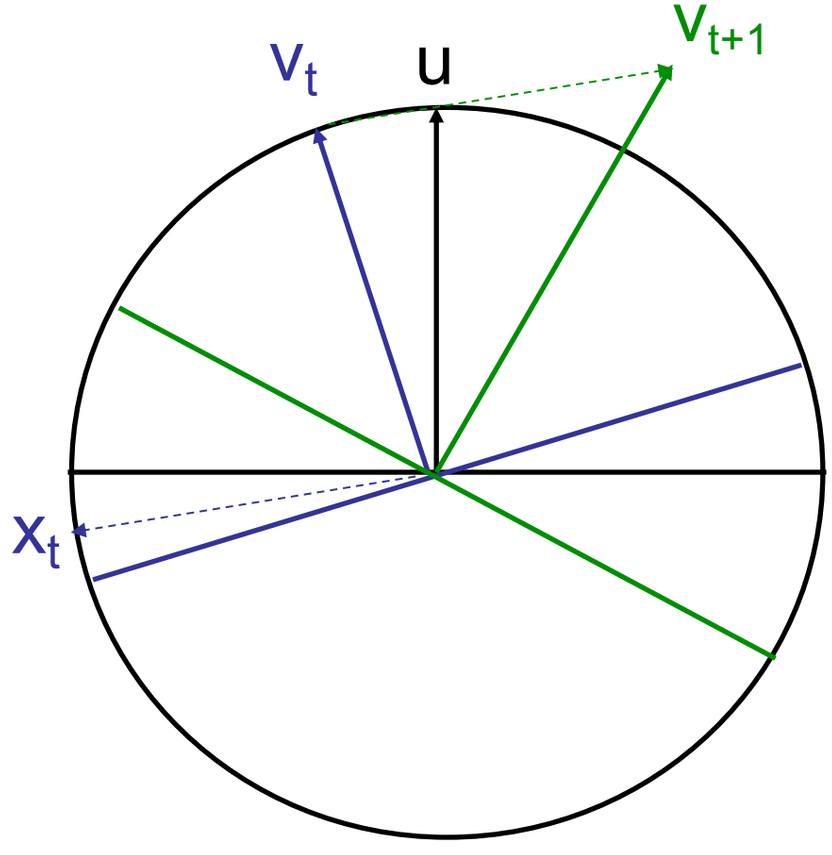
Perceptron

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. $y\,(v_t \cdot x) < 0$, $\|x\| \le R$.

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

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

Finally, after M mistakes:

a. $u \cdot v_{M+1} \ge M\,\gamma$, by Lemma 1 (expanding the recurrence).

b. $\|v_{M+1}\|^2 \le M\,R^2$, by Lemma 2. So $\|v_{M+1}\| \le M^{1/2}\,R$.

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

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

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

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

$M\,\gamma \le M^{1/2}\,R$

$M^{1/2} \le R / \gamma$, and $M \le (R / \gamma)^2$ □

Fisher's IRIS data

Four features

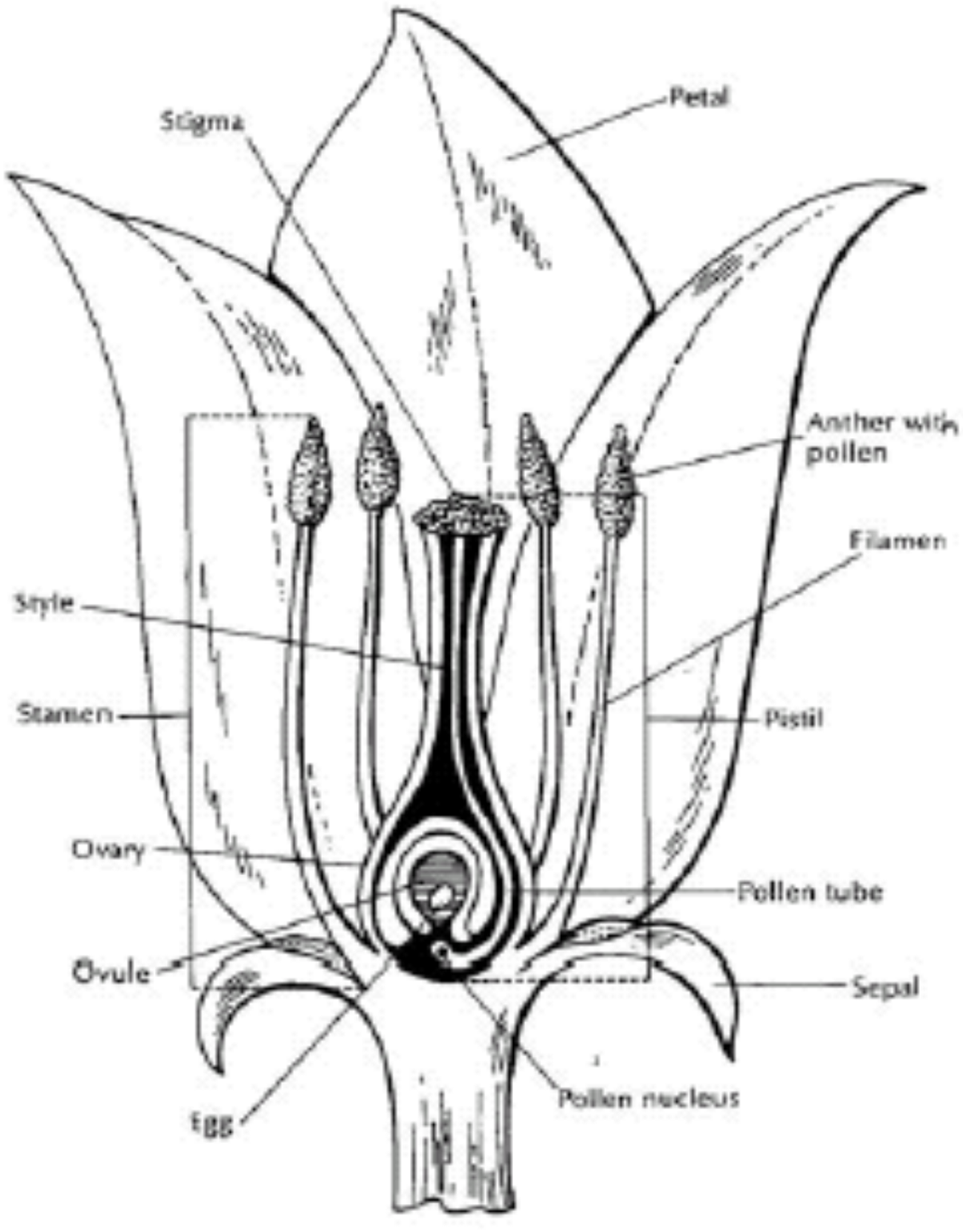
- sepal length
- sepal width
- petal length
- petal width

Three classes (species of iris)

- setosa
- versicolor
- virginica

50 instances of each

Parts of a Flower

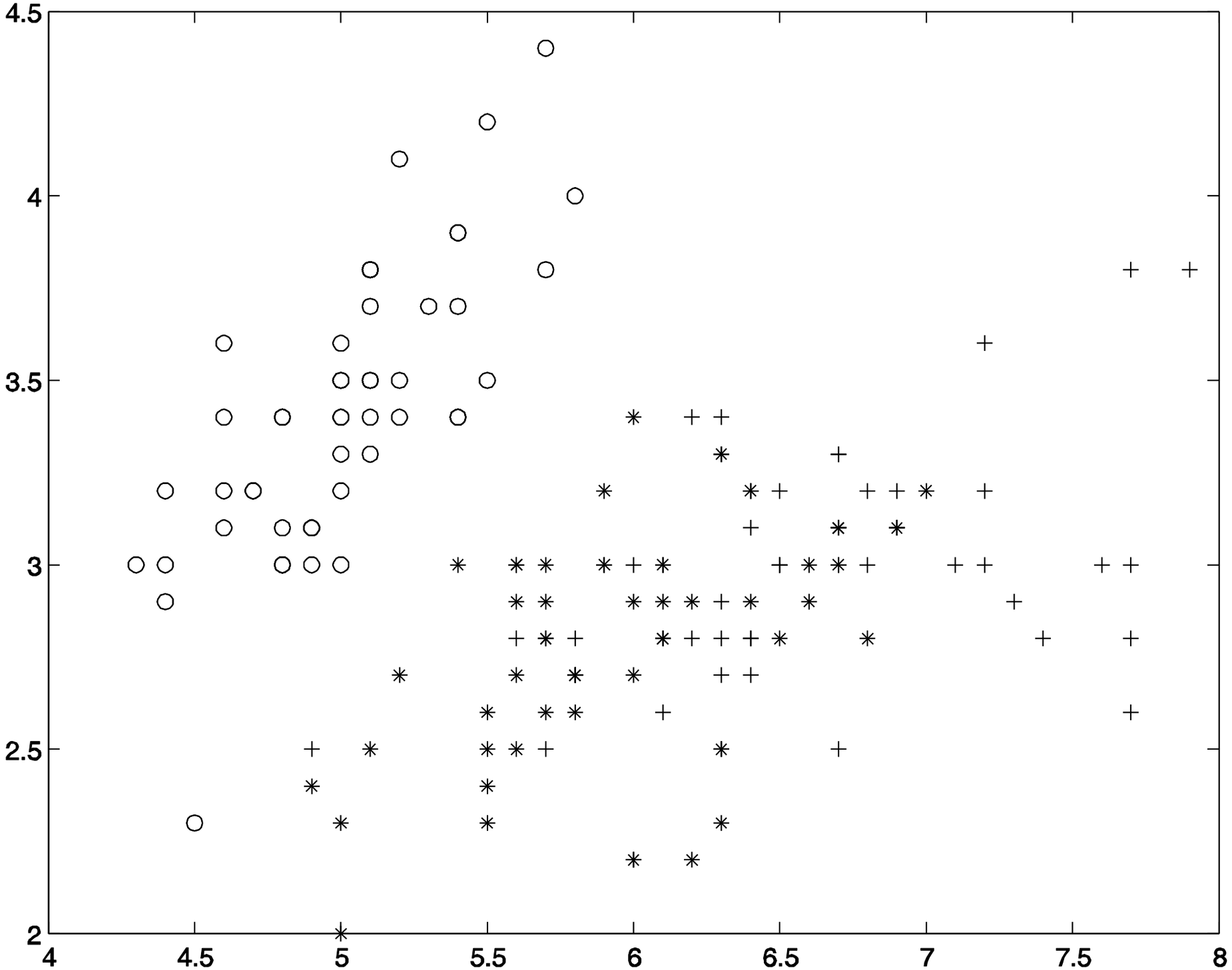


THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS

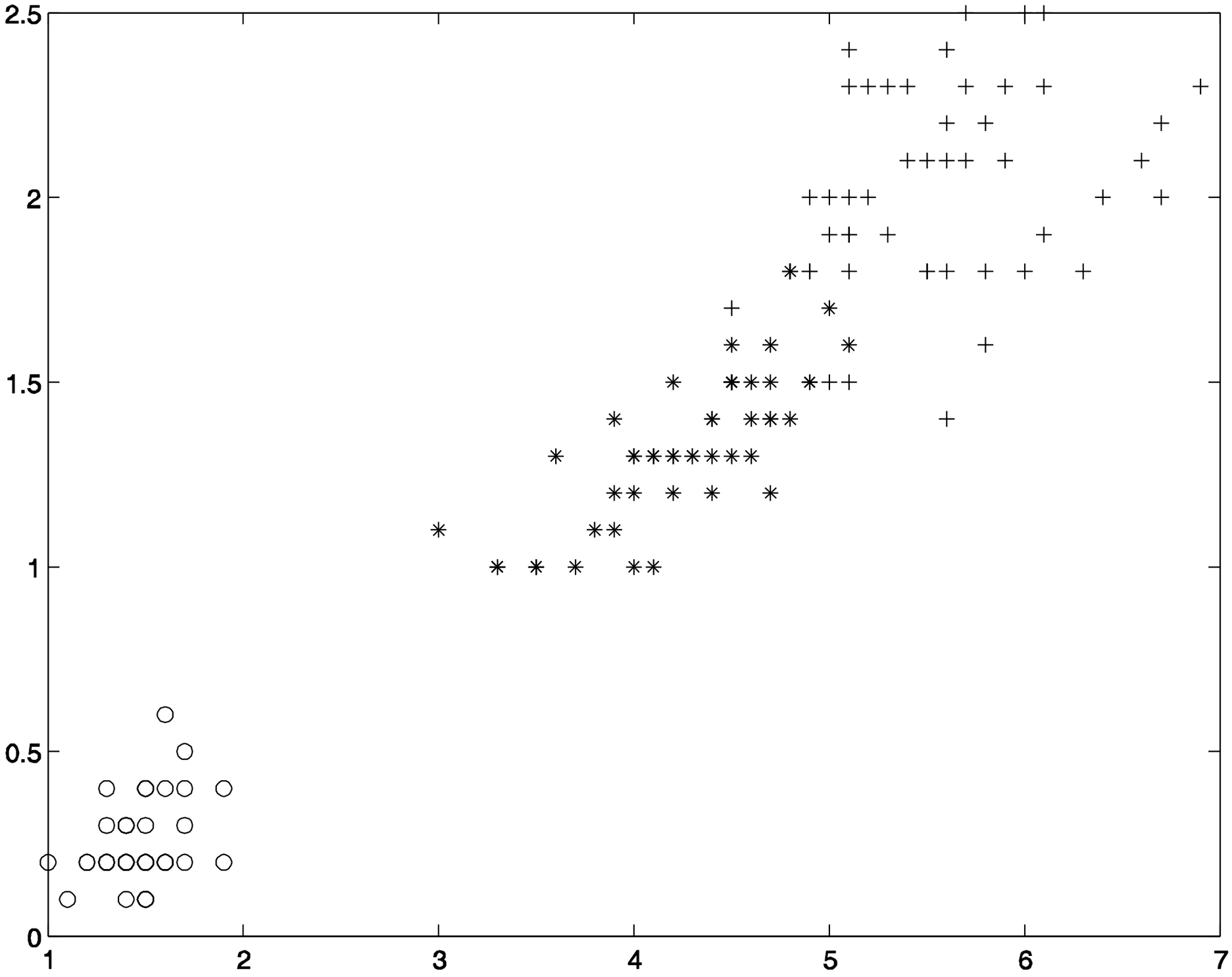
By R. A. FISHER, Sc.D., F.R.S.

I. Discriminant functions

When two or more populations have been measured in several characters, $x_1, \ldots, x_p$, 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.

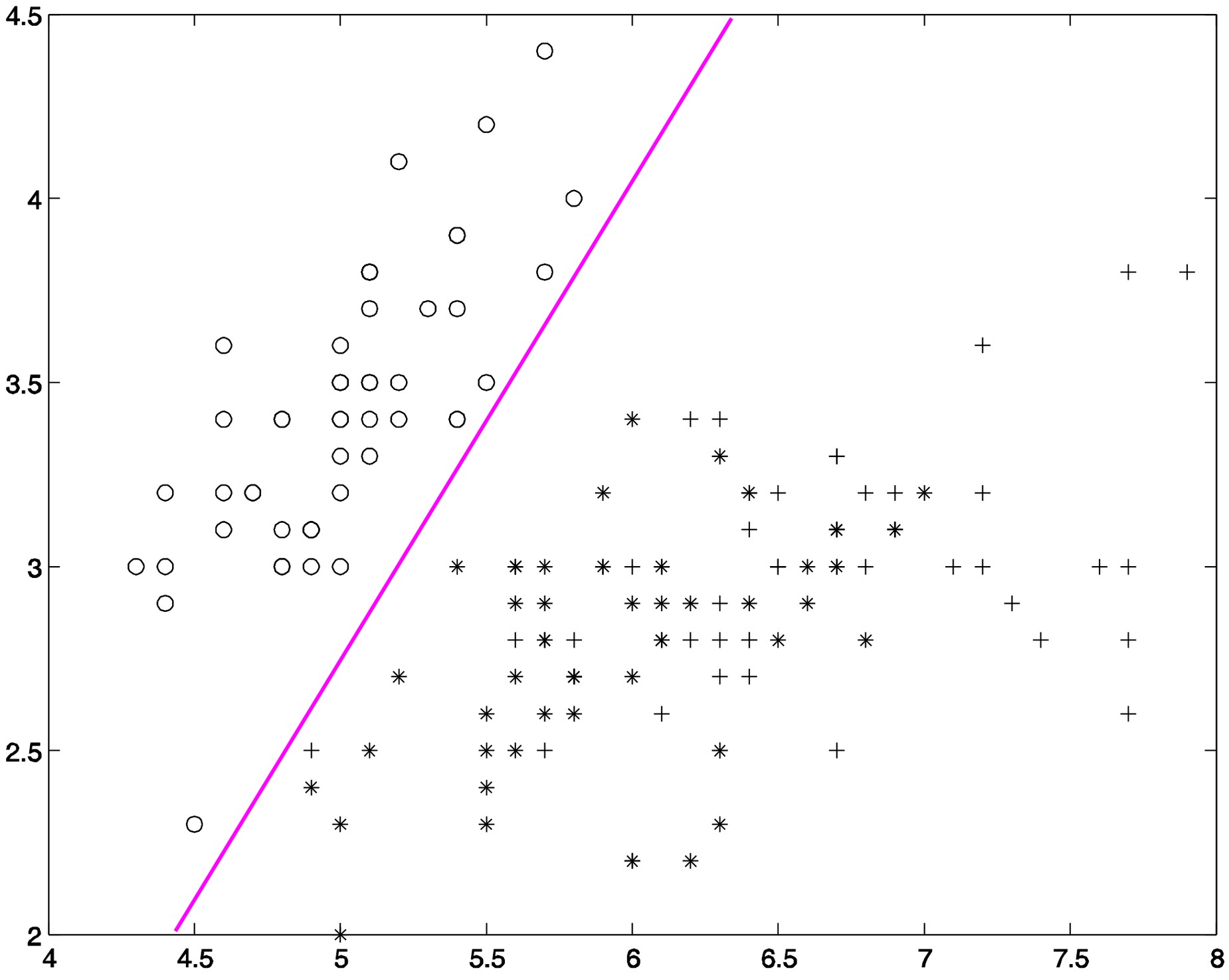


Features 1 and 2 (sepal width/length)



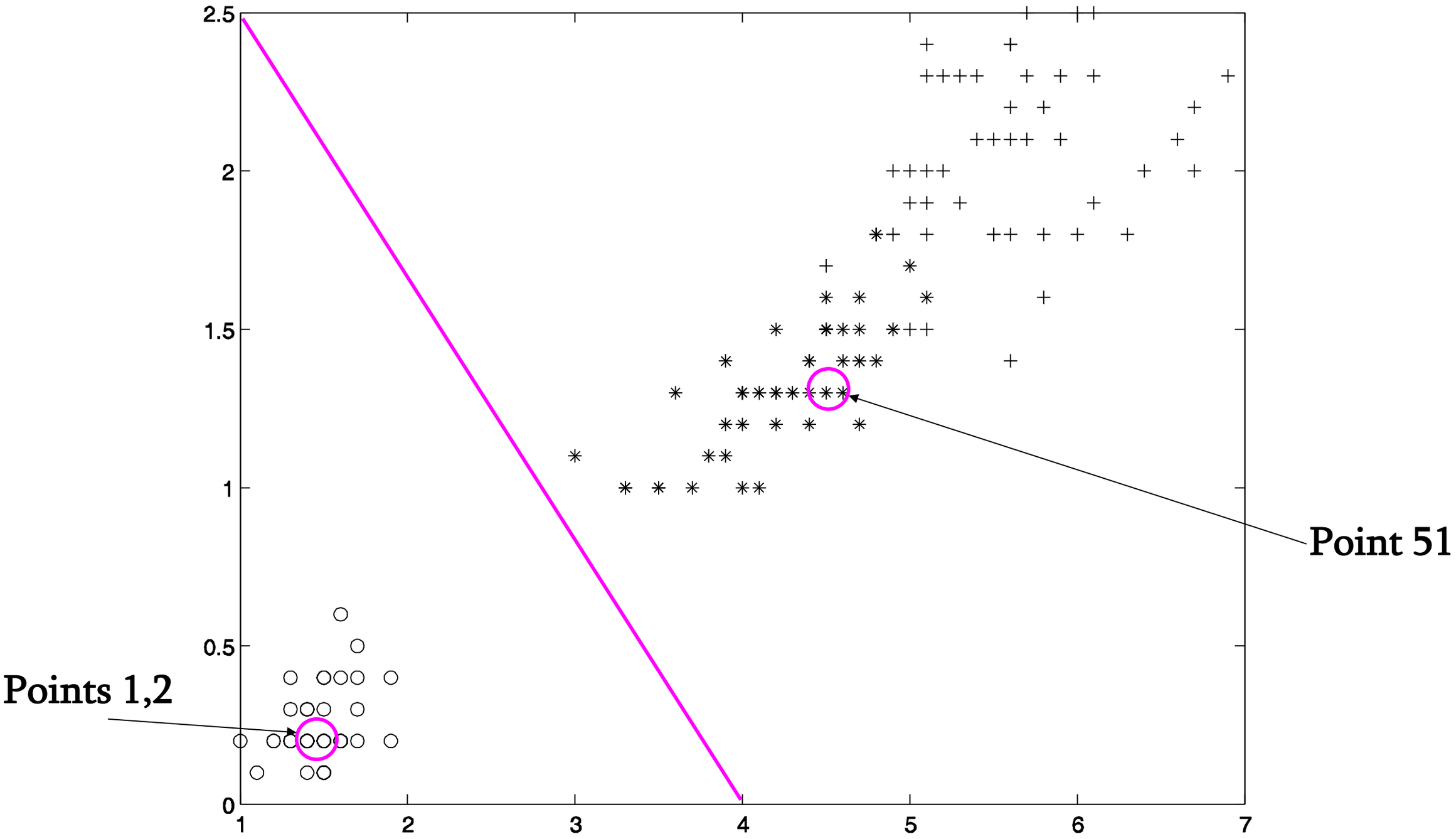
Features 3 and 4 (petal width/length)

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



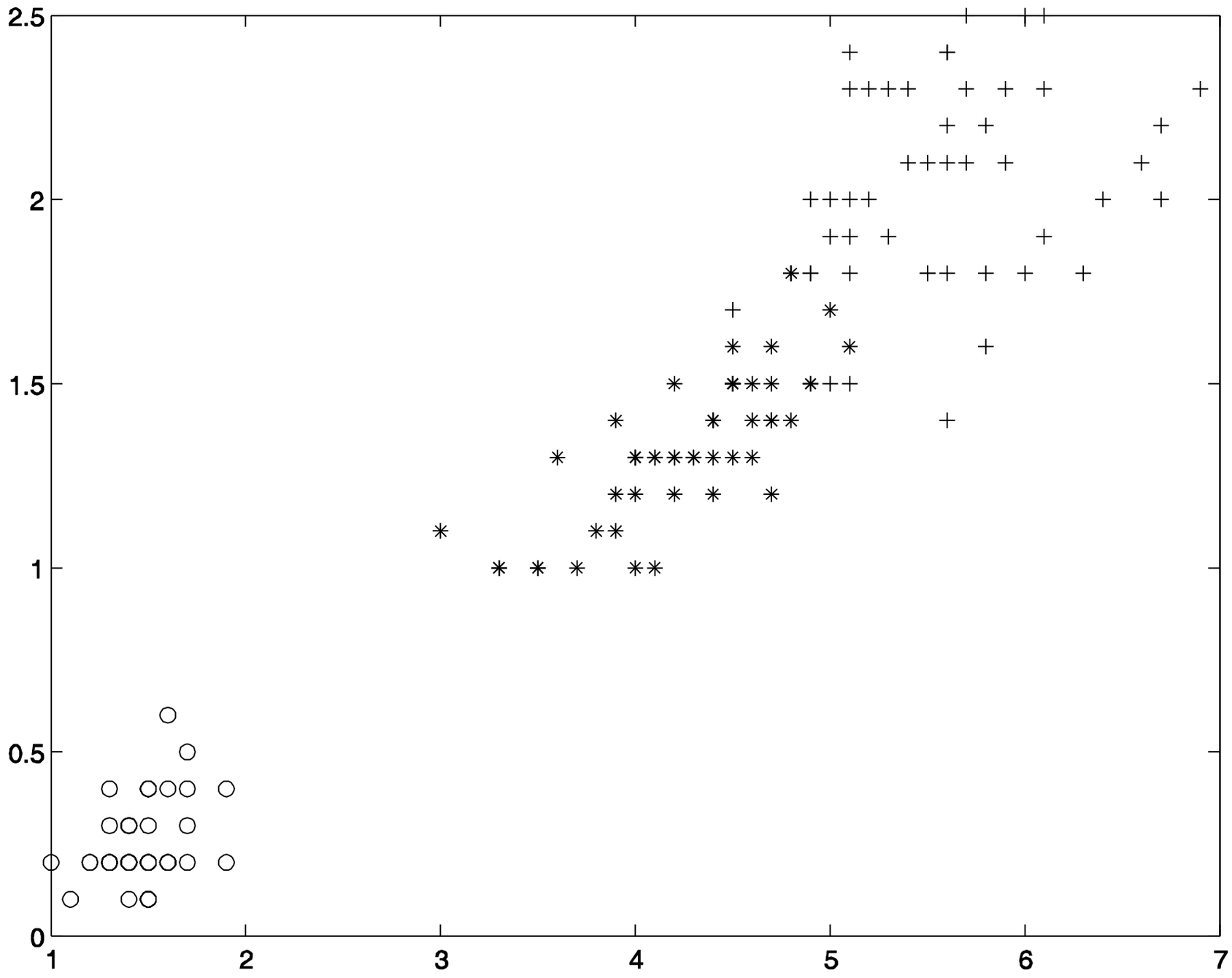
1500 updates (different permutation: 900)

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



Iteration 1 [1,51] Iteration 2 [1,2] Iteration 3 [ ]

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



What if the data is not linearly separable?