Linear Models

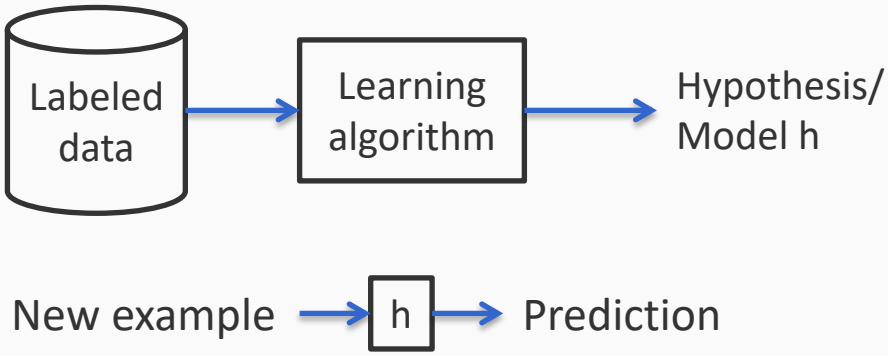
Machine Learning

Checkpoint: The bigger picture

- Supervised learning: instances, concepts, and hypotheses

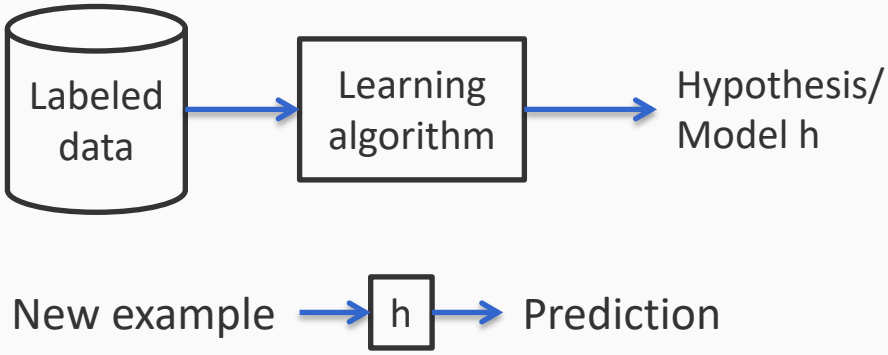
Checkpoint: The bigger picture

- Supervised learning: instances, concepts, and hypotheses



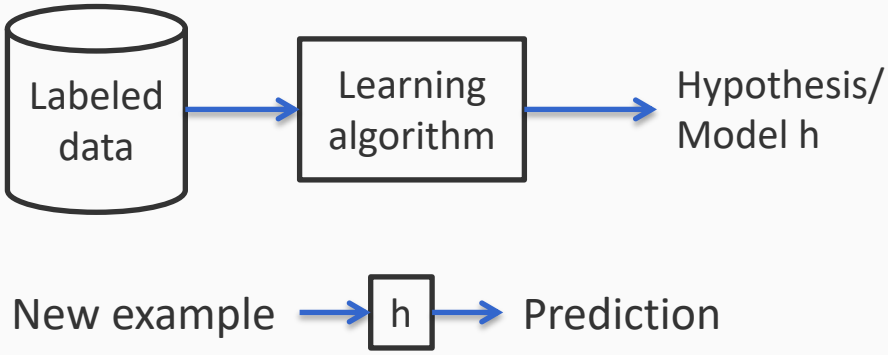
Checkpoint: The bigger picture

- Supervised learning: instances, concepts, and hypotheses
- Specific learners
  - Decision trees



Checkpoint: The bigger picture

- Supervised learning: instances, concepts, and hypotheses
- Specific learners
  - Decision trees



- General ML ideas
  - Features as high dimensional vectors
  - Overfitting

Questions?

Lecture outline

- Linear models

- What functions do linear classifiers express?

Where are we?

- Linear models
  - Introduction: Why linear classifiers and regressors?
  - Geometry of linear classifiers
  - A notational simplification
  - Learning linear classifiers: The lay of the land

- What functions do linear classifiers express?

Is learning possible at all?

- There are $2^{16} = 65536$ possible Boolean functions over 4 inputs
  - Why? There are 16 possible outputs. Each way to fill these 16 slots is a different function, giving 2^{16} functions.
- We have seen only 7 outputs
- We *cannot* know what the rest are without seeing them
  - Think of an adversary filling in the labels every time you make a guess at the function

x_1	x_2	x_3	x_4	y
0	0	0	0	?
0	0	0	1	?
0	0	1	0	**0** ←
0	0	1	1	**1** ←
0	1	0	0	**0** ←
0	1	0	1	**0** ←
0	1	1	0	**0** ←
0	1	1	1	?
1	0	0	0	?
1	0	0	1	**1** ←
1	0	1	0	?
1	0	1	1	?
1	1	0	0	**0** ←
1	1	0	1	?
1	1	1	0	?
1	1	1	1	?

Is learning possible at all?

- There are $2^{16} = 65536$ possible Boolean functions over 4 inputs
  - Why? There are 16 possible outputs. Each way to fill these 16 slots is a different function, giving 2^{16} functions.
- We have seen only 7 outputs
- We *cannot* know what the rest are without seeing them
  - Think of an adversary filling in the labels every time you make a guess at the function

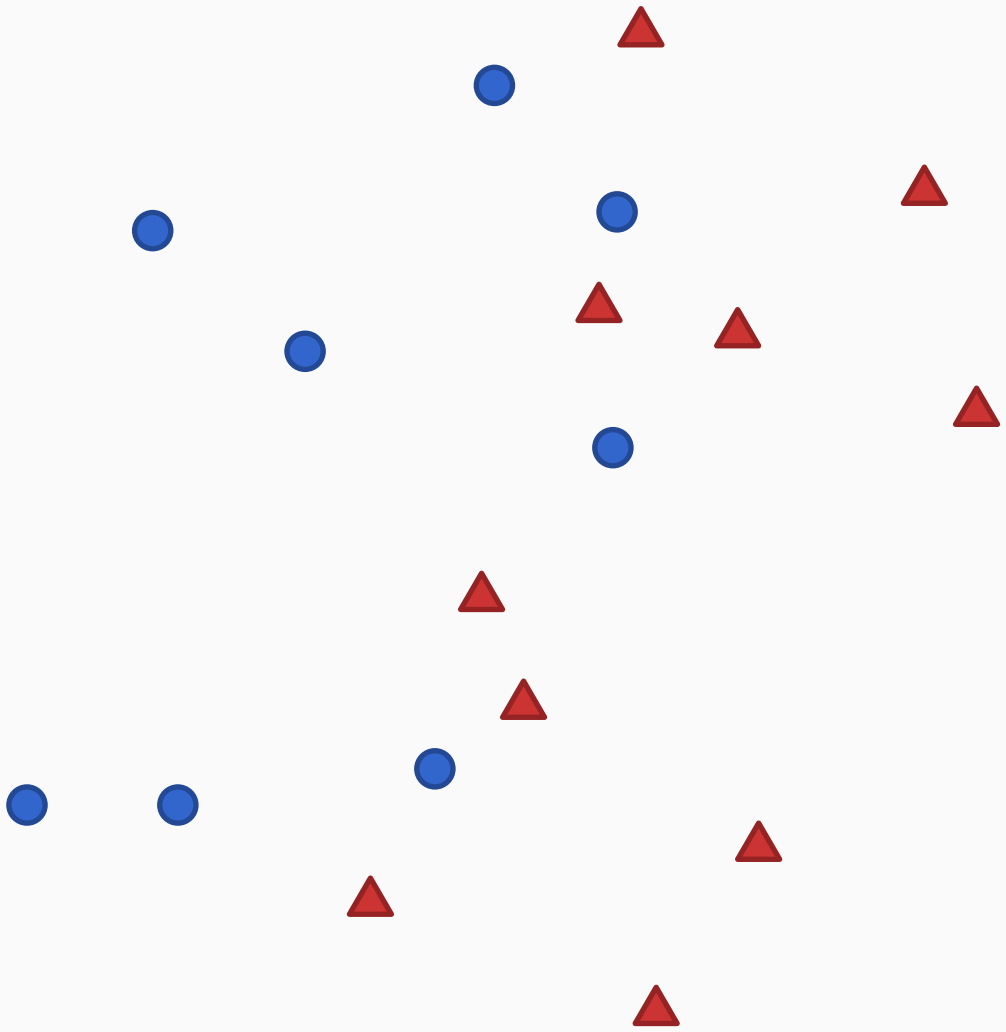
How could we possibly learn anything?

x_1	x_2	x_3	x_4	y	
0	0	0	0	?	
0	0	0	1	?	
0	0	1	0	**0**	←
0	0	1	1	**1**	←
0	1	0	0	**0**	←
			1	**0**	←
			0	**0**	←
0	1	1	1	?	
1	0	0	0	?	
1	0	0	1	**1**	←
1	0	1	0	?	
1	0	1	1	?	
1	1	0	0	**0**	←
1	1	0	1	?	
1	1	1	0	?	
1	1	1	1	?	

Solution: Restrict the search space

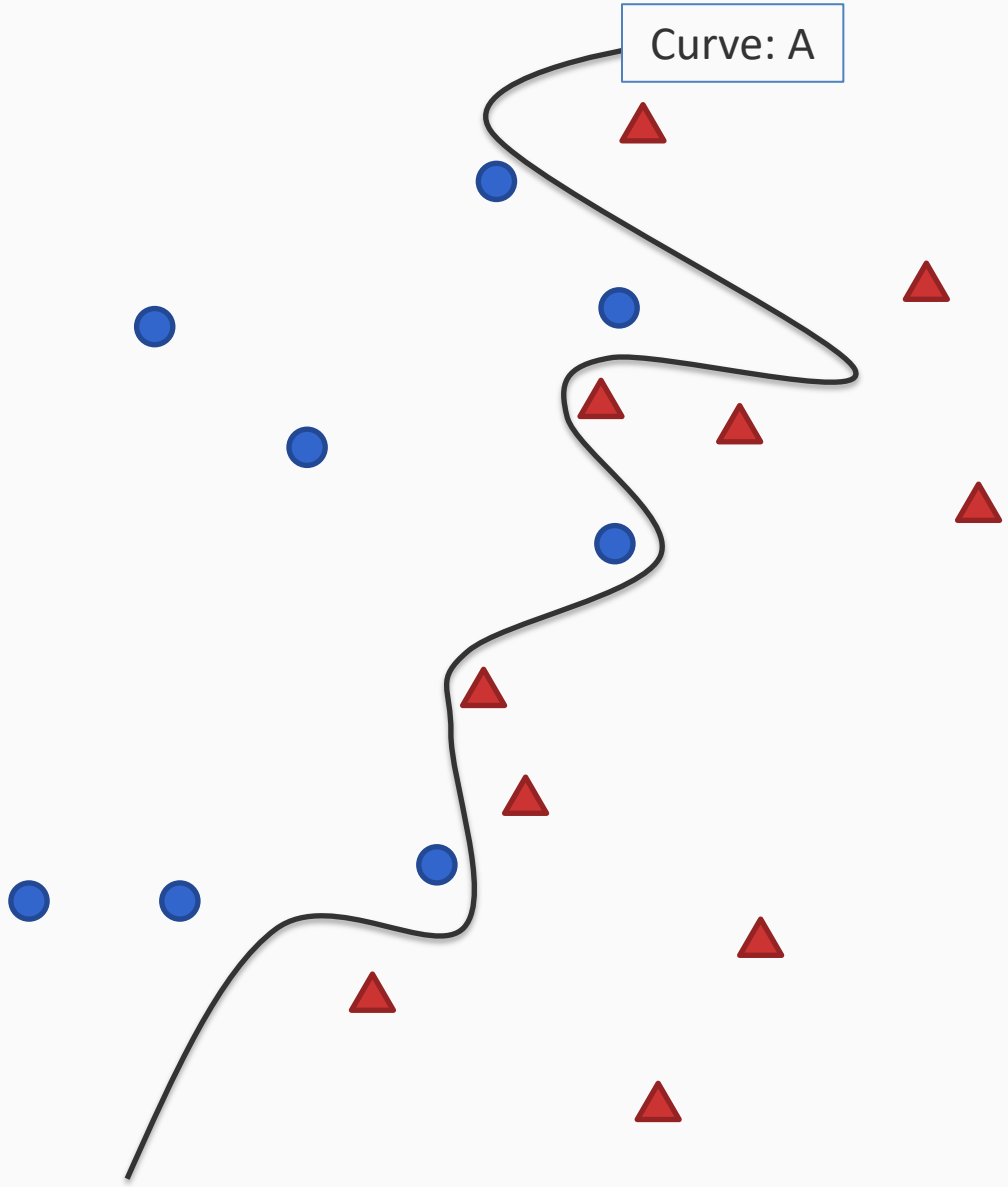
- A *hypothesis space* is the set of possible functions we consider
  - We were looking at the space of all Boolean functions
  - Instead choose a hypothesis space that is smaller than the space of all functions
    - Only simple conjunctions (with four variables, there are only 16 conjunctions without negations)
    - Simple disjunctions
    - m-of-n rules: Fix a set of n variables. At least m of them must be true
    - Linear functions

Which is the better classifier?



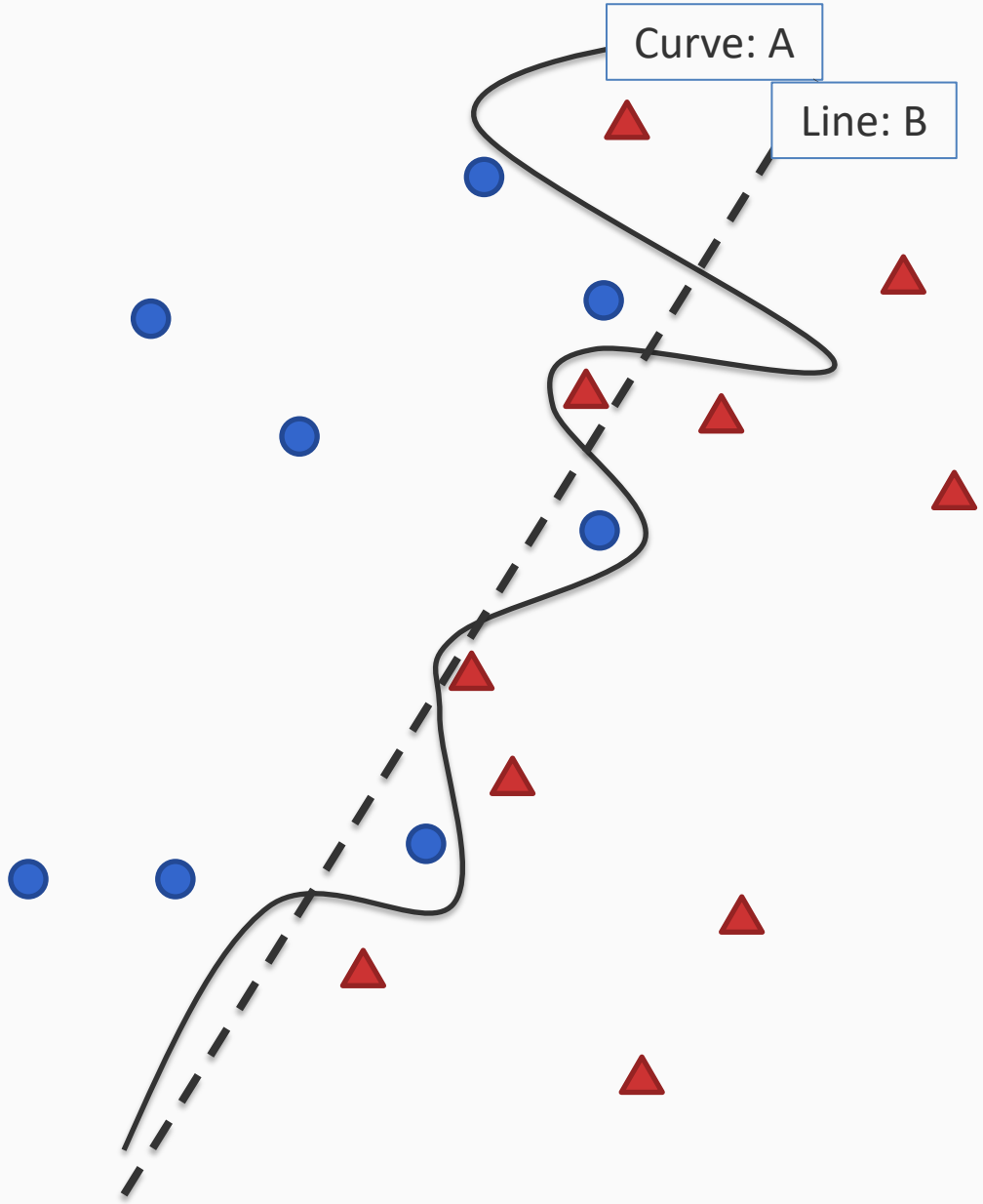
Suppose this our training set and we have to separate the blue circles from the red triangles

Which is the better classifier?



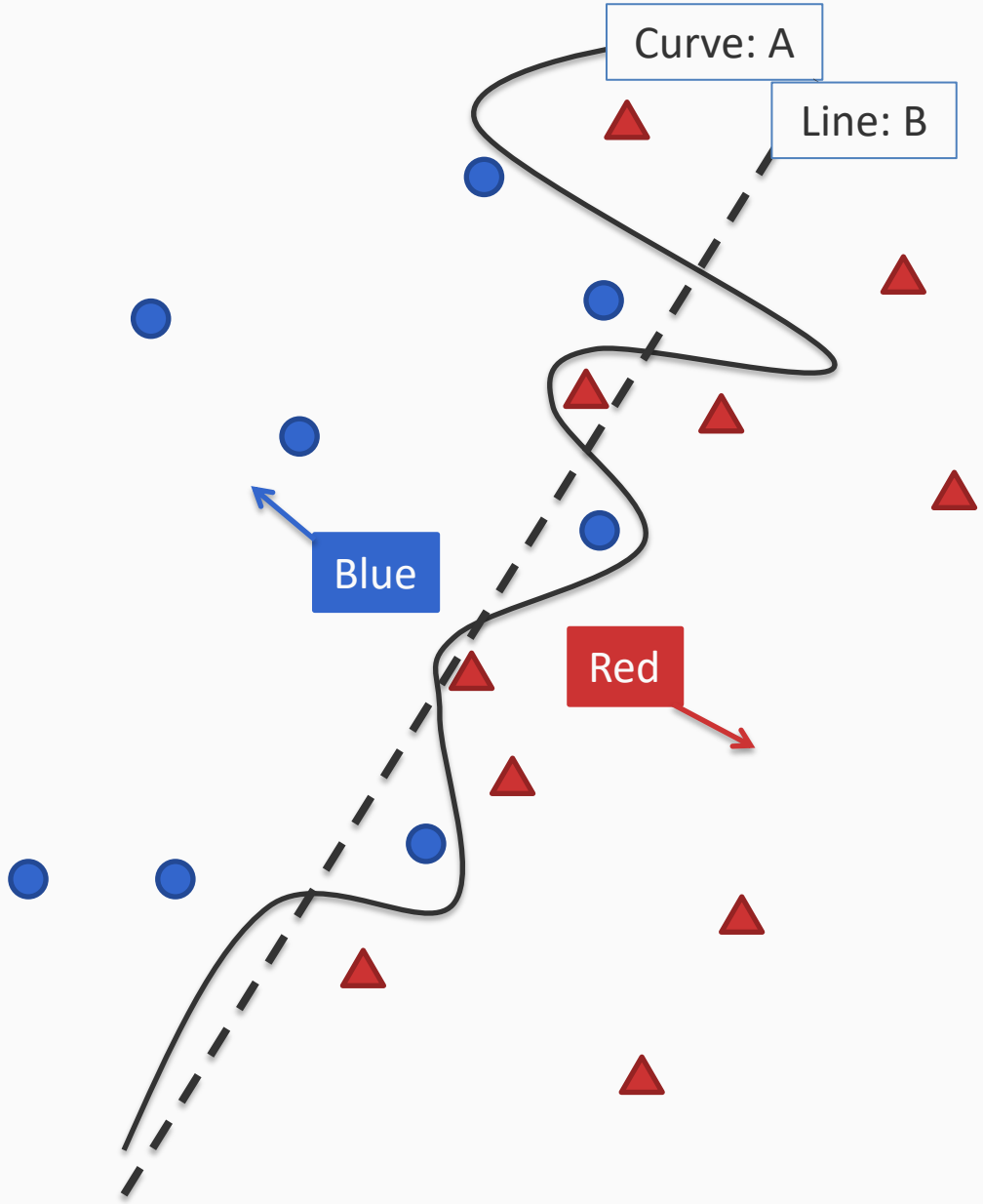
Suppose this our training set and we have to separate the blue circles from the red triangles

Which is the better classifier?



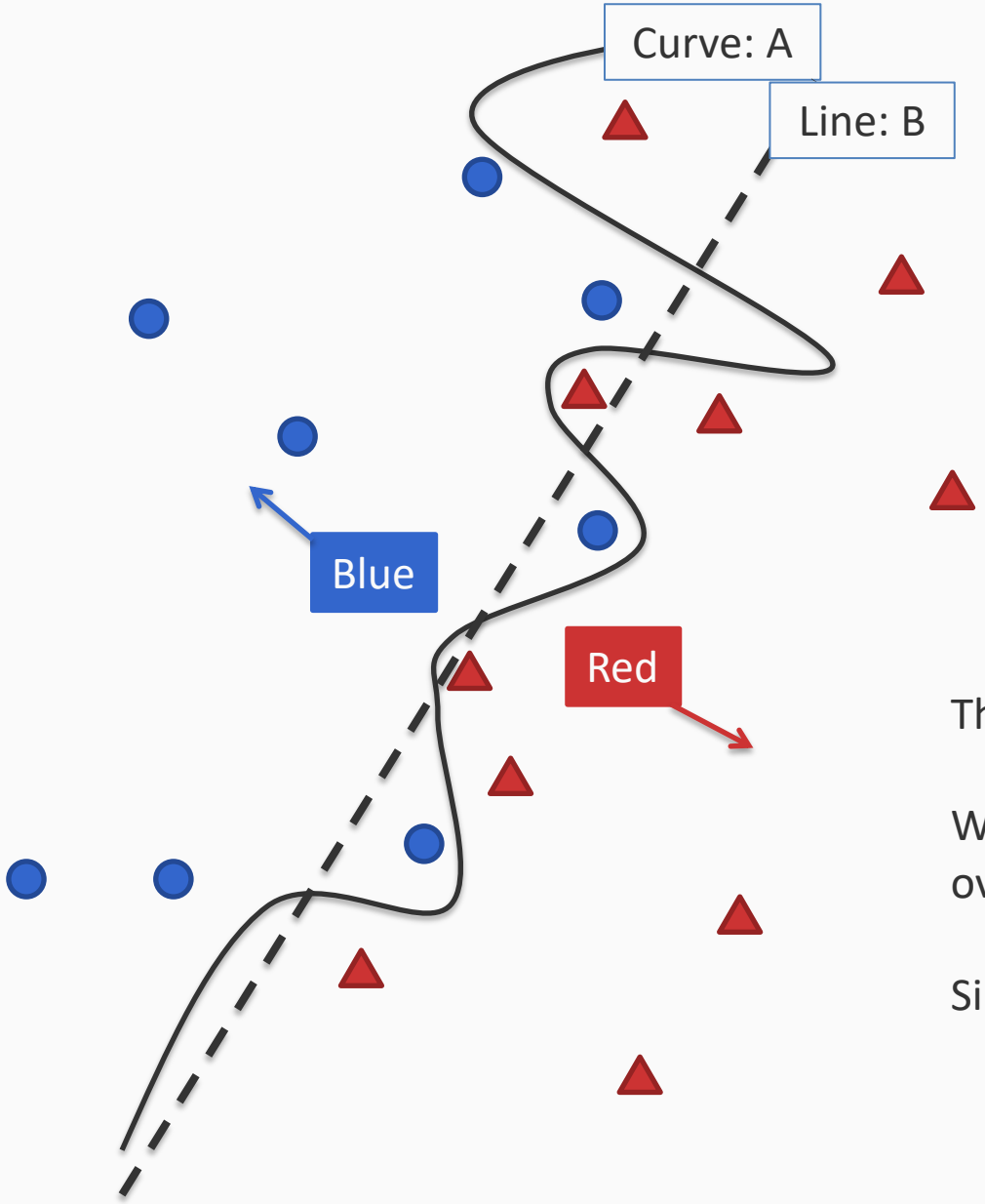
Suppose this our training set and we have to separate the blue circles from the red triangles

Which is the better classifier?



Suppose this our training set and we have to separate the blue circles from the red triangles

Which is the better classifier?



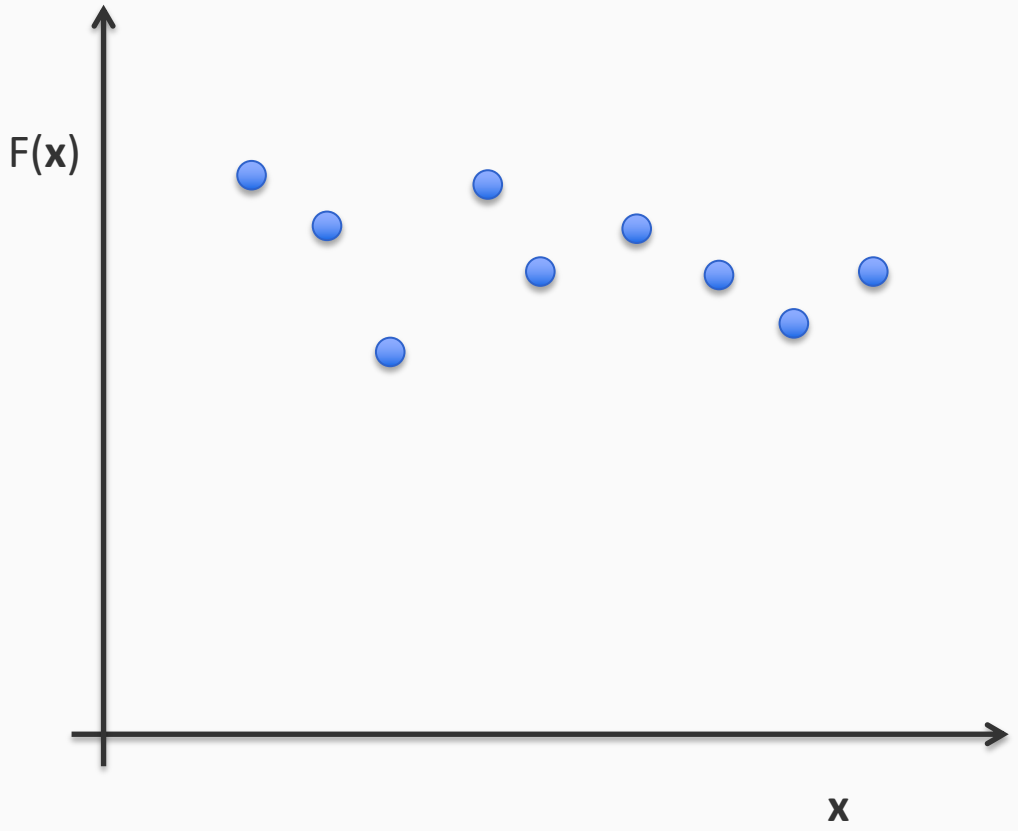
Suppose this our training set and we have to separate the blue circles from the red triangles

Think about overfitting

Which curve runs the risk of overfitting?

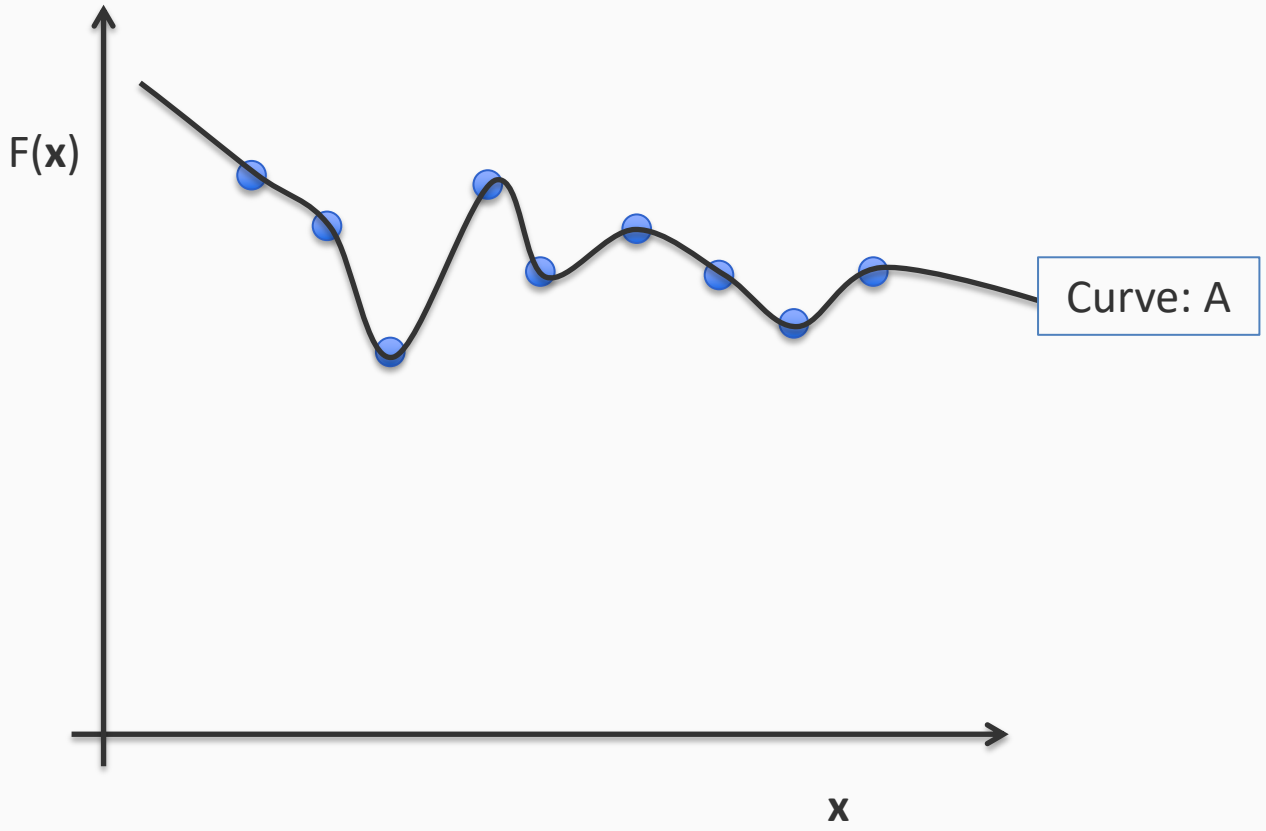
Simplicity versus Accuracy

Similar argument for regression



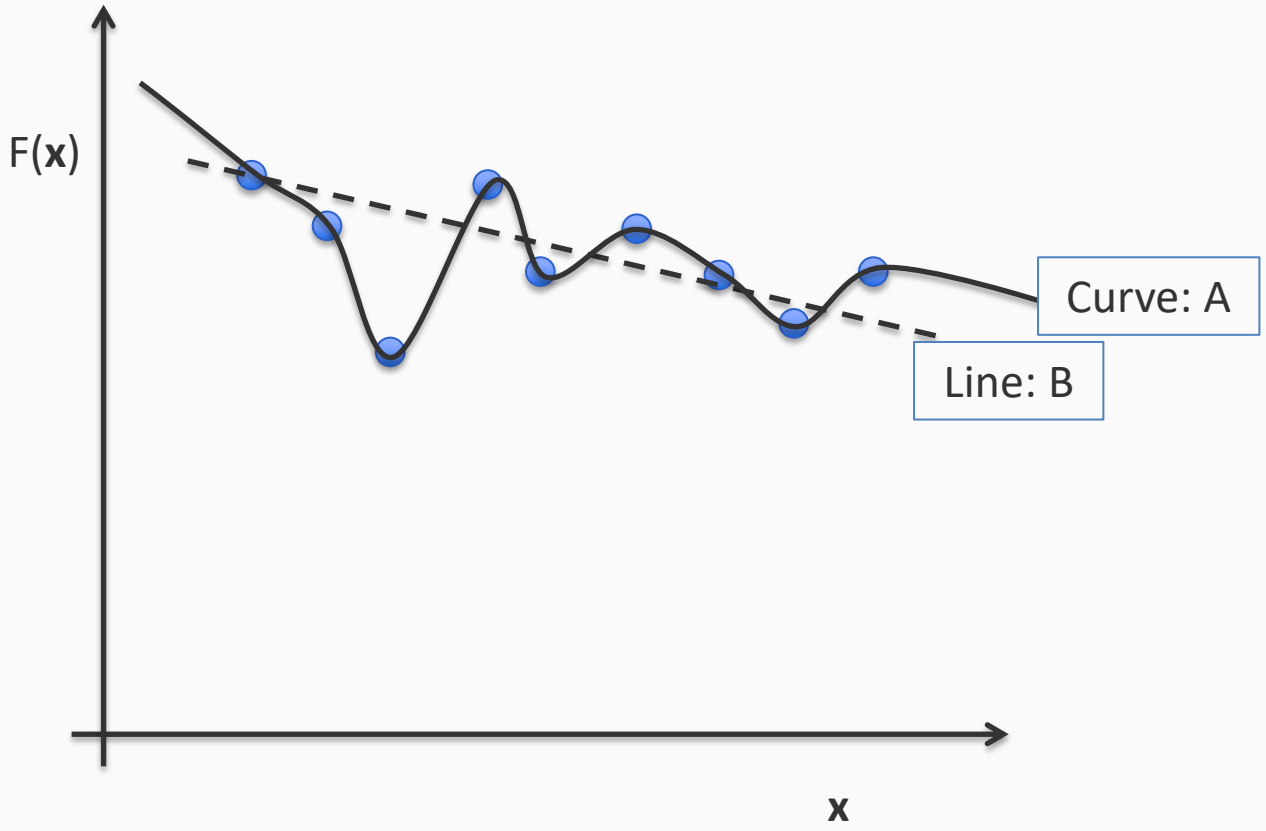
Linear regression might make smaller errors on new points

Similar argument for regression



Linear regression might make smaller errors on new points

Similar argument for regression



Linear regression might make smaller errors on new points

Recall: Regression vs. Classification

- Linear regression is about predicting real valued outputs

- Linear classification is about predicting a discrete class label
  - `+1` or `-1`
  - `SPAM` or `NOT-SPAM`
  - Or more than two categories

Linear classifiers: An example

Suppose we want to determine whether a robot arm is defective or not using two measurements:

1. The maximum distance the arm can reach d
2. The maximum angle it can rotate a

Linear classifiers: An example

Suppose we want to determine whether a robot arm is defective or not using two measurements:

1. The maximum distance the arm can reach d
2. The maximum angle it can rotate a

Suppose we use a linear decision rule that predicts

$$\text{defective if } 2d + 0.01a \geq 7$$

Linear classifiers: An example

Suppose we want to determine whether a robot arm is defective or not using two measurements:

1. The maximum distance the arm can reach d
2. The maximum angle it can rotate a

Suppose we use a linear decision rule that predicts

$$\text{defective if } 2d + 0.01a \geq 7$$

We can apply this rule if we have the two measurements

For example: for a certain arm, if d = 3 and a = 200, then

$$2d + 0.01a = 8 \geq 7$$

The arm would be labeled as not defective

Linear classifiers: An example

Suppose we want to determine whether a robot arm is defective or not using two measurements:

1. The maximum distance the arm can reach d
2. The maximum angle it can rotate a

Suppose we use a linear decision rule that predicts

$$\text{defective if } 2d + 0.01a \geq 7$$

This rule is an example of a linear classifier

Features are weighted and added up,
the sum is checked against a threshold

Linear Classifiers

Inputs are d dimensional feature vectors, denoted by $\mathbf{x}$

Output is a label $y \in \{-1, 1\}$

Linear Classifiers

Inputs are d dimensional feature vectors, denoted by $\mathbf{x}$

Output is a label $y \in \{-1, 1\}$

Linear Threshold Units classify an example $\mathbf{x}$ using parameters $\mathbf{w}$ (a d dimensional vector) and b (a real number) according the following classification rule

$$\text{Output} = \text{sign}\left(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b\right) = \text{sign}\left(\sum_i w_i x_i + b\right)$$

Linear Classifiers

Inputs are d dimensional feature vectors, denoted by $\mathbf{x}$

Output is a label $y \in \{-1, 1\}$

Linear Threshold Units classify an example $\mathbf{x}$ using parameters $\mathbf{w}$ (a d dimensional vector) and b (a real number) according the following classification rule

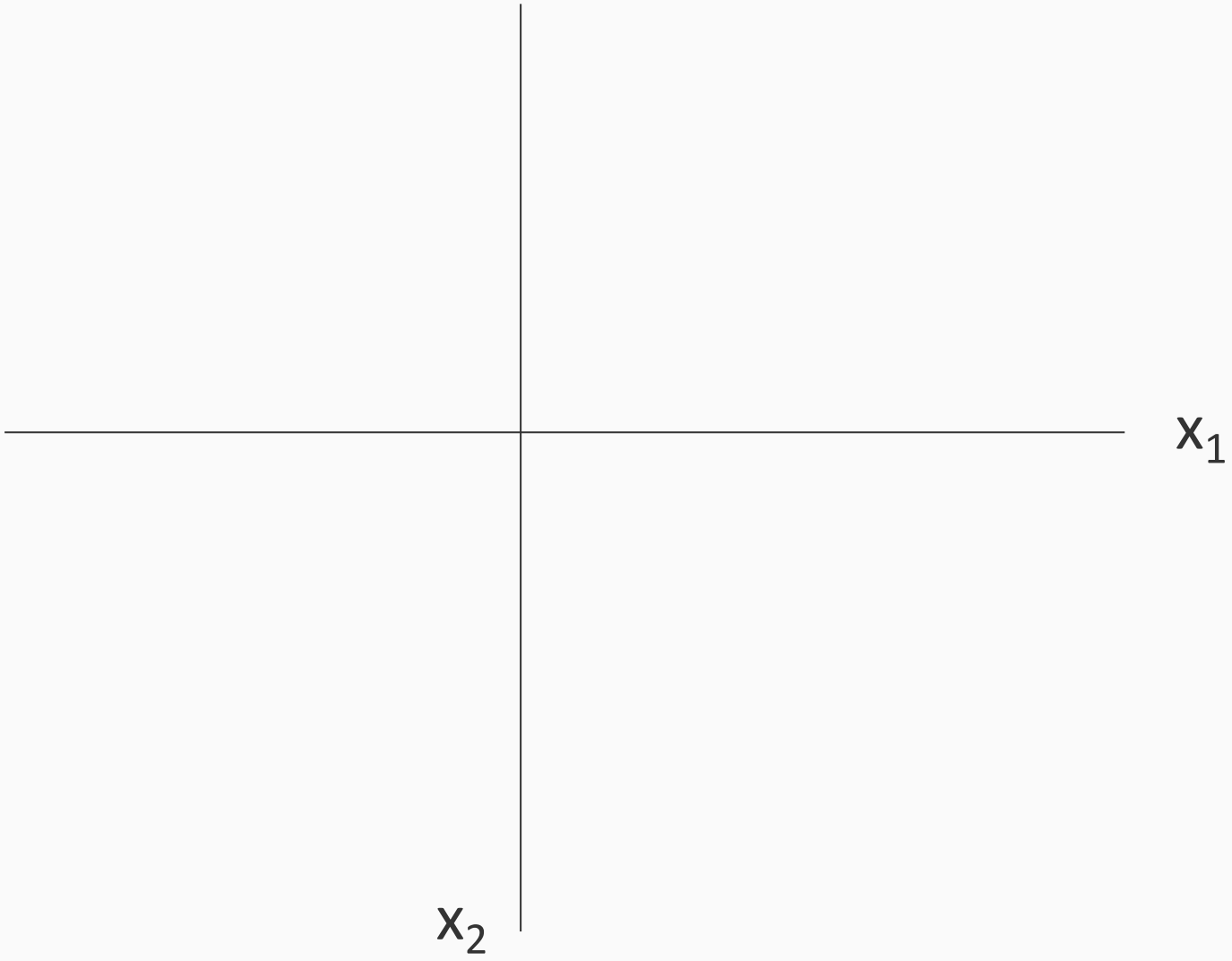
$$\text{Output} = \text{sign}\left(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b\right) = \text{sign}\left(\sum_i w_i x_i + b\right)$$

$$\text{if } \mathbf{w}^{\mathrm{T}}\mathbf{x} + b \geq 0 \text{ then predict } y = +1$$
$$\text{if } \mathbf{w}^{\mathrm{T}}\mathbf{x} + b < 0 \text{ then predict } y = -1$$

b is called the bias term

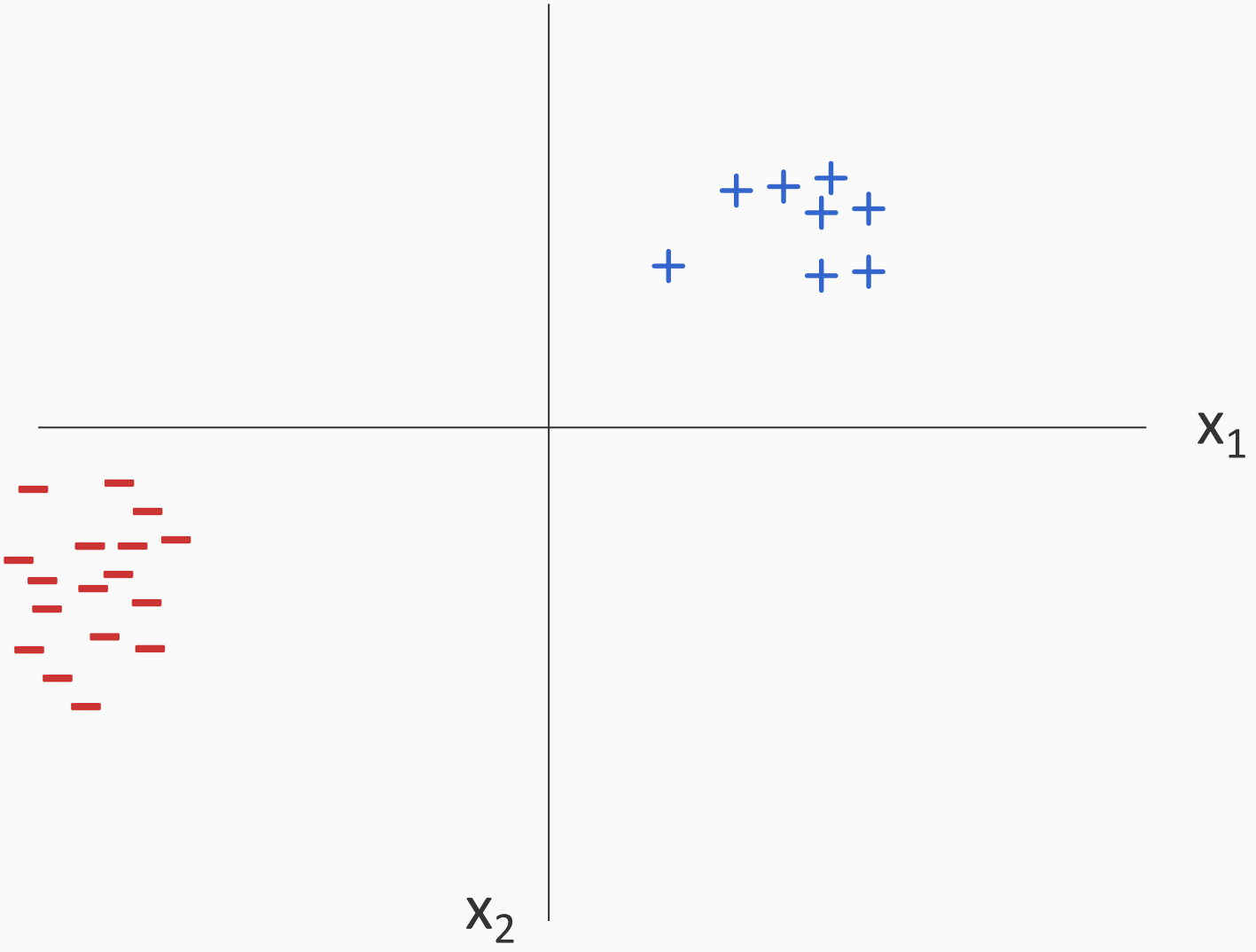
The geometry of a linear classifier

An illustration in two dimensions



The geometry of a linear classifier

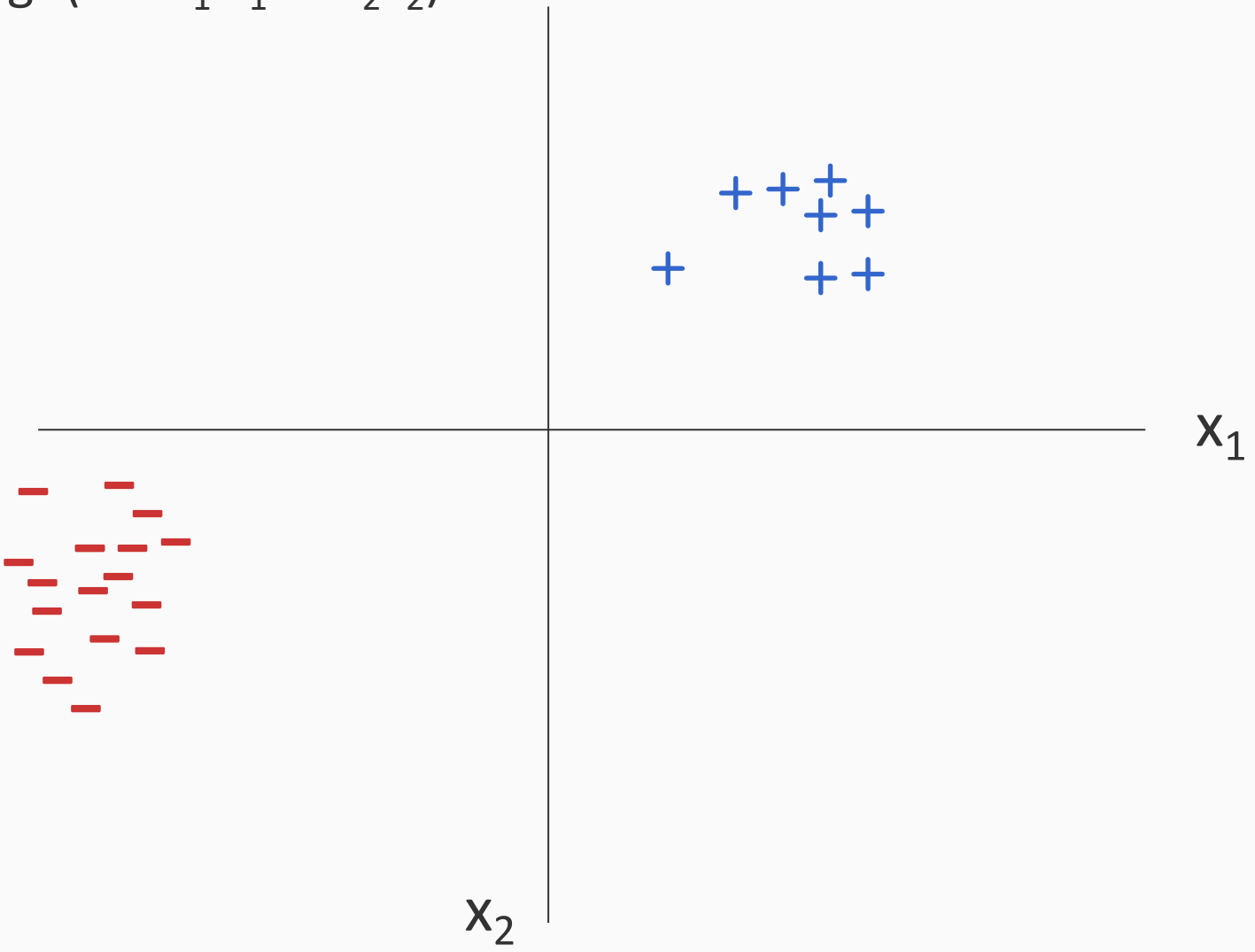
An illustration in two dimensions



The geometry of a linear classifier

An illustration in two dimensions

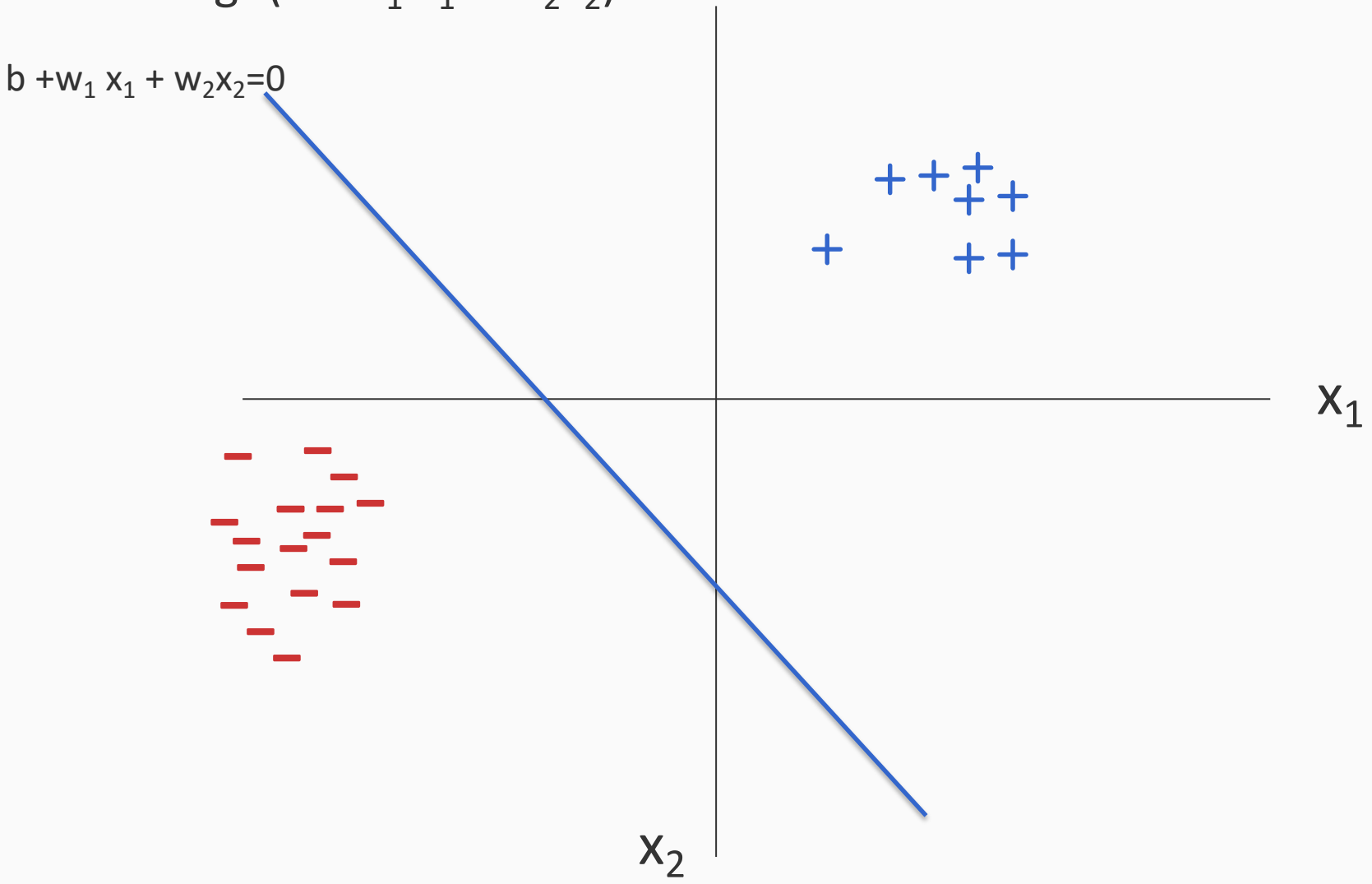
$\text{sgn}(b + w_1 x_1 + w_2 x_2)$



The geometry of a linear classifier

An illustration in two dimensions

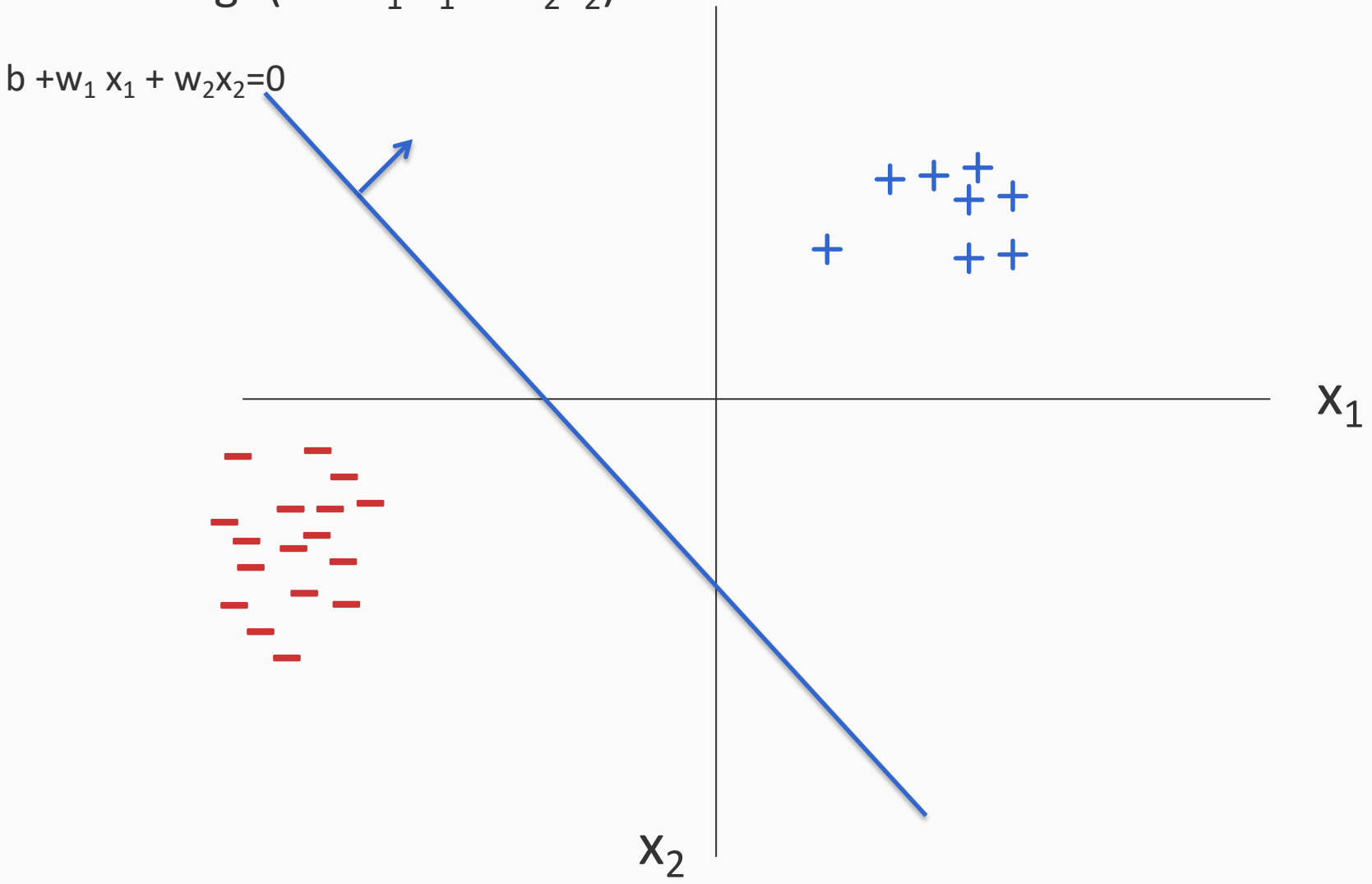
$\text{sgn}(b + w_1 x_1 + w_2 x_2)$



The geometry of a linear classifier

An illustration in two dimensions

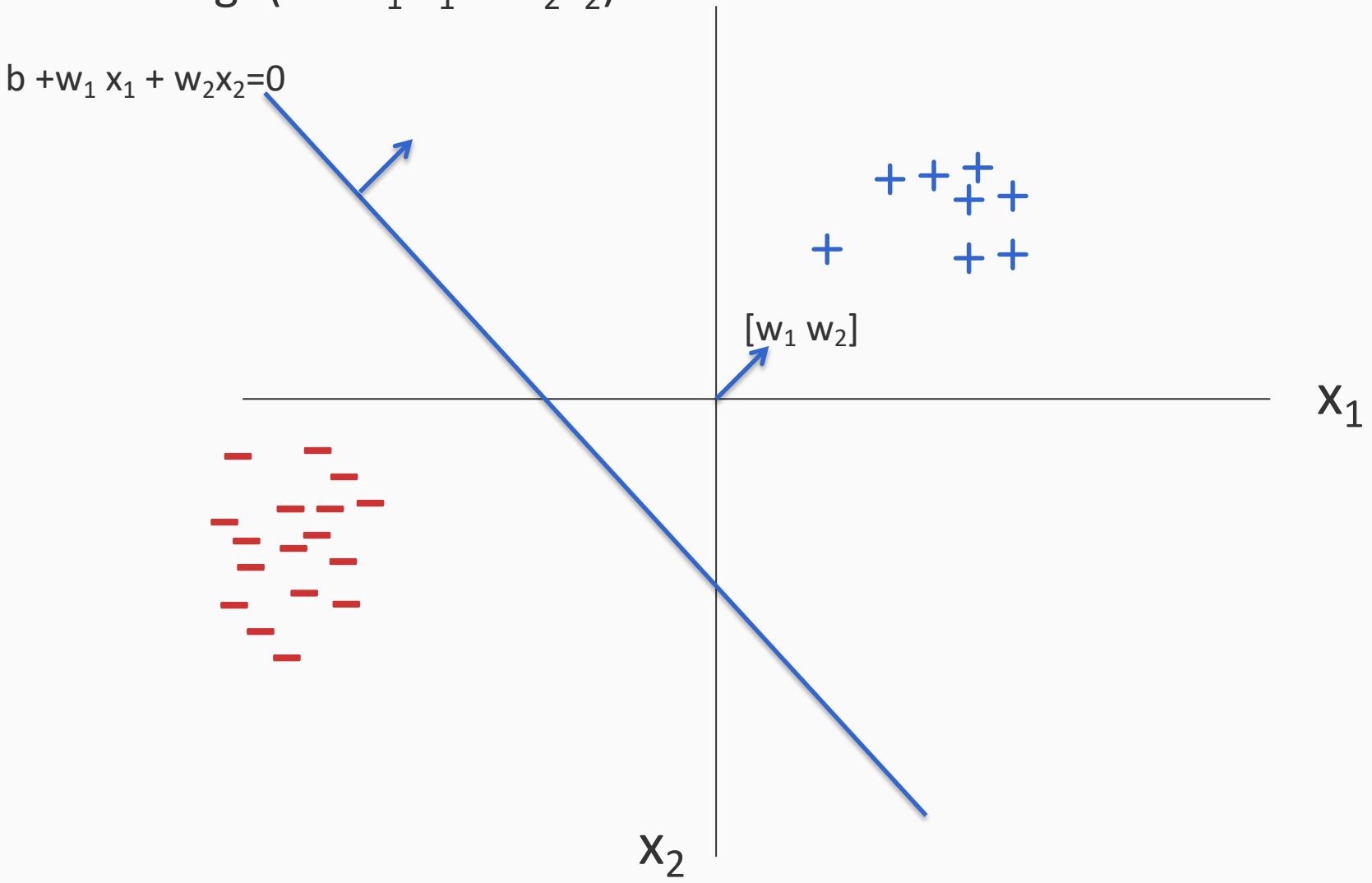
$\text{sgn}(b + w_1 x_1 + w_2 x_2)$



The geometry of a linear classifier

An illustration in two dimensions

$\text{sgn}(b + w_1 x_1 + w_2 x_2)$

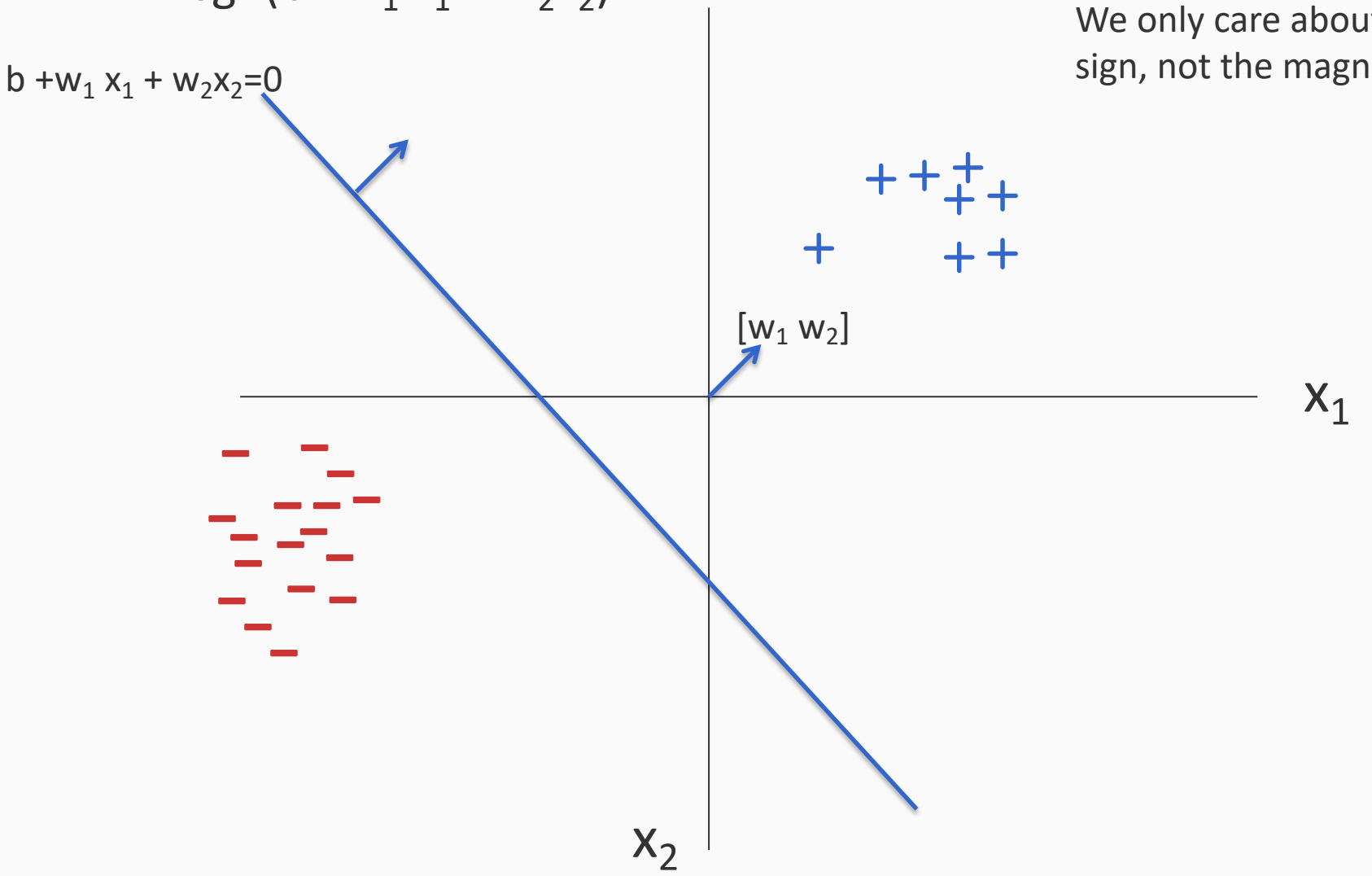


The geometry of a linear classifier

An illustration in two dimensions

$\text{sgn}(b + w_1 x_1 + w_2 x_2)$

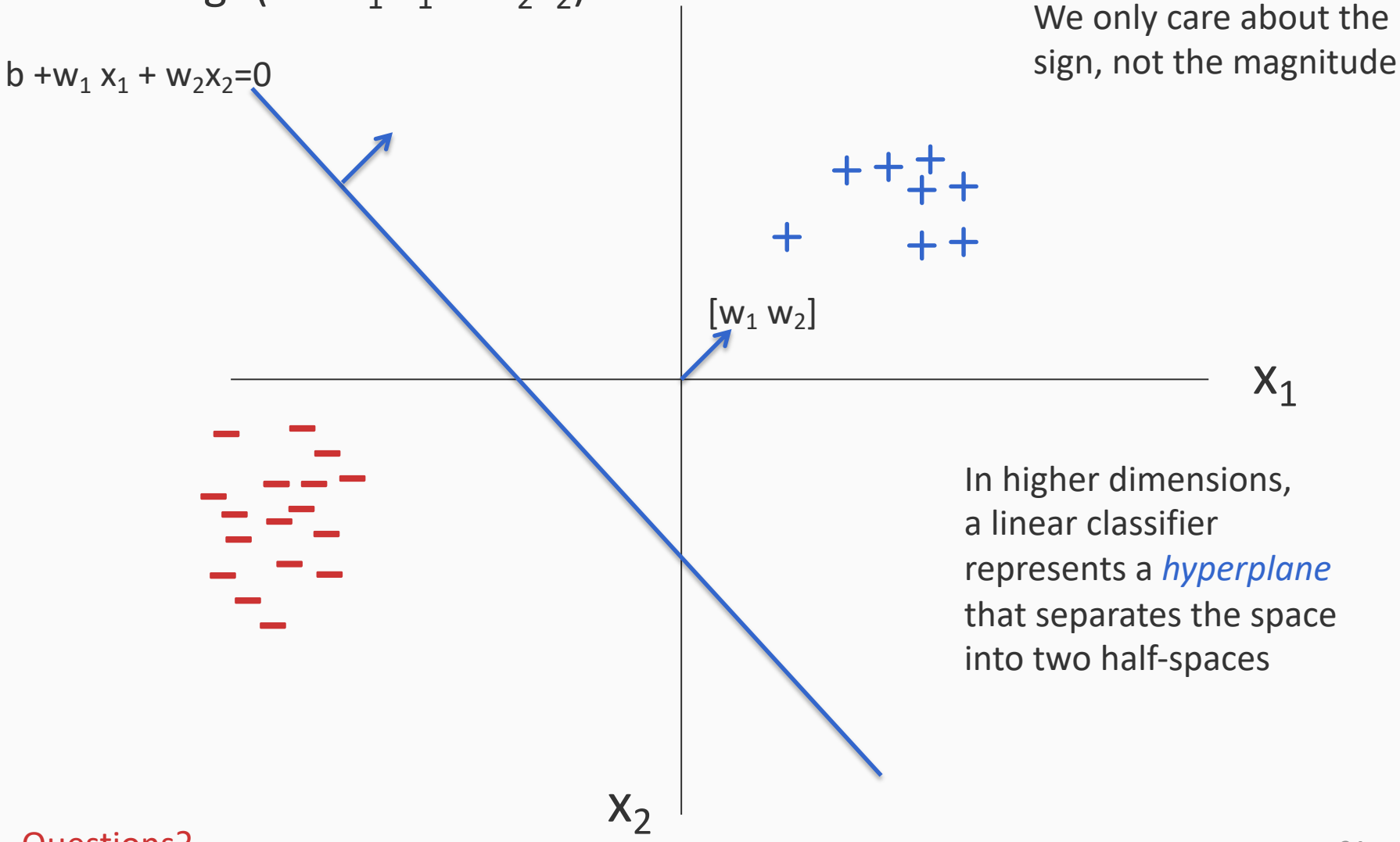
We only care about the sign, not the magnitude



The geometry of a linear classifier

An illustration in two dimensions

$\text{sgn}(b + w_1 x_1 + w_2 x_2)$



We only care about the sign, not the magnitude

In higher dimensions, a linear classifier represents a *hyperplane* that separates the space into two half-spaces

Questions?

Simplifying notation

We can stop writing b at each step using notational sugar:

The prediction function is $\text{sgn}(\mathbf{w}^T\mathbf{x} + b) = \text{sgn}(\sum_i w_i x_i + b)$

Simplifying notation

We can stop writing b at each step using notational sugar:

The prediction function is $\text{sgn}(\mathbf{w}^T\mathbf{x} + b) = \text{sgn}(\sum_i w_i x_i + b)$

Rewrite $\mathbf{x}$ as $\begin{bmatrix}\mathbf{x}\\1\end{bmatrix}$. Call this $\mathbf{x}'$

Rewrite $\mathbf{w}$ as $\begin{bmatrix}\mathbf{w}\\b\end{bmatrix}$. Call this $\mathbf{w}'$

Simplifying notation

We can stop writing b at each step using notational sugar:

The prediction function is $\text{sgn}(\mathbf{w}^T\mathbf{x} + b) = \text{sgn}(\sum_i w_i x_i + b)$

Rewrite $\mathbf{x}$ as $\begin{bmatrix}\mathbf{x}\\1\end{bmatrix}$. Call this $\mathbf{x}'$

Rewrite $\mathbf{w}$ as $\begin{bmatrix}\mathbf{w}\\b\end{bmatrix}$. Call this $\mathbf{w}'$

Note that $\mathbf{w}^{\mathrm{T}}\mathbf{x} + b$ is the same as $\mathbf{w}'^{\boldsymbol{T}}\mathbf{x}'$

Simplifying notation

We can stop writing b at each step using notational sugar:

The prediction function is $\text{sgn}(\mathbf{w}^T\mathbf{x} + b) = \text{sgn}(\sum_i w_i x_i + b)$

Rewrite $\mathbf{x}$ as $\begin{bmatrix}\mathbf{x}\\1\end{bmatrix}$. Call this $\mathbf{x}'$

Rewrite $\mathbf{w}$ as $\begin{bmatrix}\mathbf{w}\\b\end{bmatrix}$. Call this $\mathbf{w}'$

Note that $\mathbf{w}^{\mathrm{T}}\mathbf{x} + b$ is the same as $\mathbf{w}'^{\boldsymbol{T}}\mathbf{x}'$

The prediction function is now $\text{sgn}(\mathbf{w}'^{\boldsymbol{T}}\mathbf{x}')$

Increases dimensionality by one

Equivalent to adding a feature that is a constant: always 1

Simplifying notation

We can stop writing b at each step using notational sugar:

The prediction function is $\text{sgn}(\mathbf{w}^T\mathbf{x} + b) = \text{sgn}(\sum_i w_i x_i + b)$

Rewrite $\mathbf{x}$ as $\begin{bmatrix}\mathbf{x}\\1\end{bmatrix}$. Call this $\mathbf{x}'$

Rewrite $\mathbf{w}$ as $\begin{bmatrix}\mathbf{w}\\b\end{bmatrix}$. Call this $\mathbf{w}'$

Note that $\mathbf{w}^{\mathrm{T}}\mathbf{x} + b$ is the same as $\mathbf{w}'^{\boldsymbol{T}}\mathbf{x}'$

The prediction function is now $\text{sgn}(\mathbf{w}'^{\boldsymbol{T}}\mathbf{x}')$

Increases dimensionality by one

Equivalent to adding a feature that is a constant: always 1

In the increased dimensional space, the vector $\mathbf{w}'$ goes through the origin

Simplifying notation

We can stop writing b at each step using notational sugar:

The prediction function is $\text{sgn}(\mathbf{w}^T\mathbf{x} + b) = \text{sgn}(\sum_i w_i x_i + b)$

Rewrite $\mathbf{x}$ as $\begin{bmatrix}\mathbf{x}\\1\end{bmatrix}$. Call this $\mathbf{x}'$ Rewrite $\mathbf{w}$ as $\begin{bmatrix}\mathbf{w}\\b\end{bmatrix}$. Call this $\mathbf{w}'$

Note that $\mathbf{w}^{\mathrm{T}}\mathbf{x} + b$ is the same as $\mathbf{w}'^{\boldsymbol{T}}\mathbf{x}'$

The prediction function is now $\text{sgn}(\mathbf{w}'^{\boldsymbol{T}}\mathbf{x}')$

Increases dimensionality by one

Equivalent to adding a feature that is a constant: always 1

In the increased dimensional space, the vector $\mathbf{w}'$ goes through the origin

We sometimes hide the bias b, and instead fold the bias term into the weights by adding an extra constant feature. But remember that it is there.

Questions?

Coming up (next several weeks): Linear *classification*

- **Perceptron**: Error driven learning, updates the hypothesis if there is an error
- **Support Vector Machines**: Define a different cost function that includes an error term *and* a term that targets future performance
- **Naïve Bayes classifier**: A simple linear classifier with a probabilistic interpretation
- **Logistic regression**: Another probabilistic linear classifier, bears similarity to support vector machines

In all cases, the prediction will be done with the same rule:

$$\mathbf{w}^{\mathrm{T}}\mathbf{x} + b \geq 0 \Rightarrow y = +1$$
$$\mathbf{w}^{\mathrm{T}}\mathbf{x} + b < 0 \Rightarrow y = -1$$