Lecture 9: Linear Models

COMP 343, Spring 2022

Victoria Manfredi

Acknowledgements: These slides are based primarily on those created by Vivek Srikumar (Utah) and Dan Roth (Penn), and lectures by Balaraman Ravindran (IIT Madras)

Today's Topics

Homework 4 out

- Due Thursday, March 3 by 5p

Checkpoint

- The bigger picture

Linear models

- Overview
- What functions do linear classifiers express?

vumanfredi@wesleyan.edu

Homework 4 discussion

Decision tree questions?

Python questions?

Scikit questions?

Cross-validation questions?

Checkpoint

THE BIGGER PICTURE

Checkpoint: the bigger picture

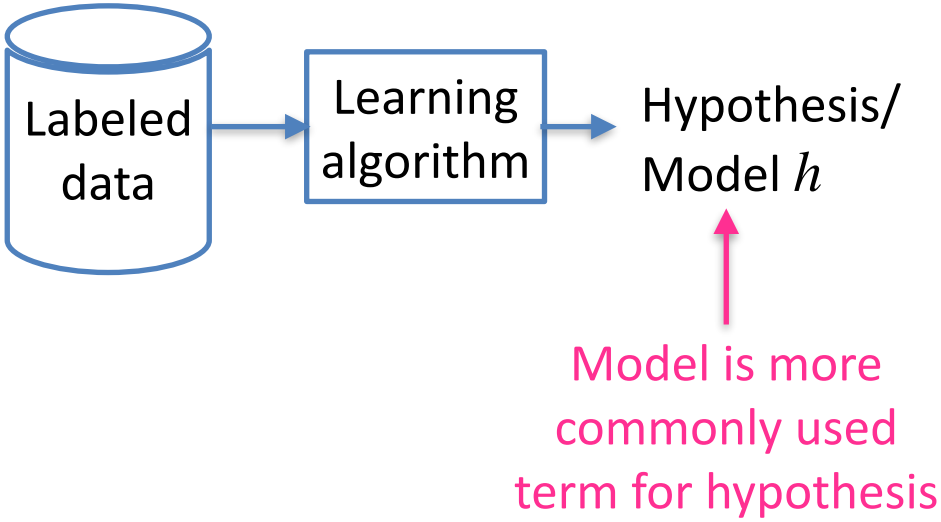
Supervised learning: instances and hypotheses

Inputs to functions

Functions

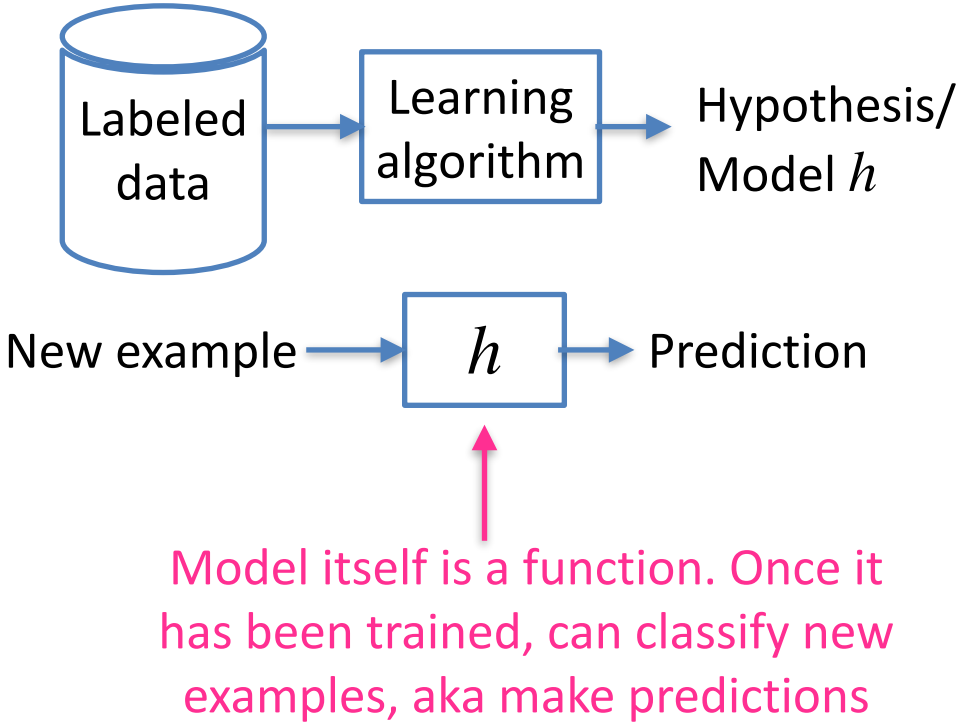
Checkpoint: the bigger picture

Supervised learning: instances and hypotheses



Checkpoint: the bigger picture

Supervised learning: instances and hypotheses

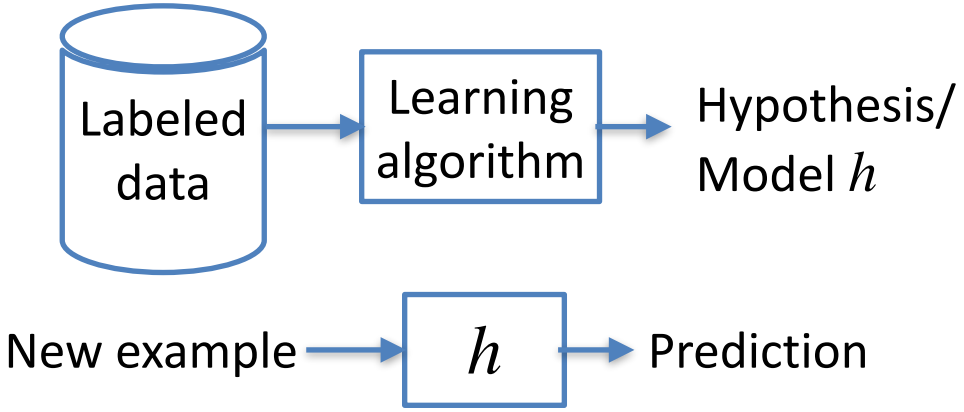


Checkpoint: the bigger picture

Supervised learning: instances and hypotheses

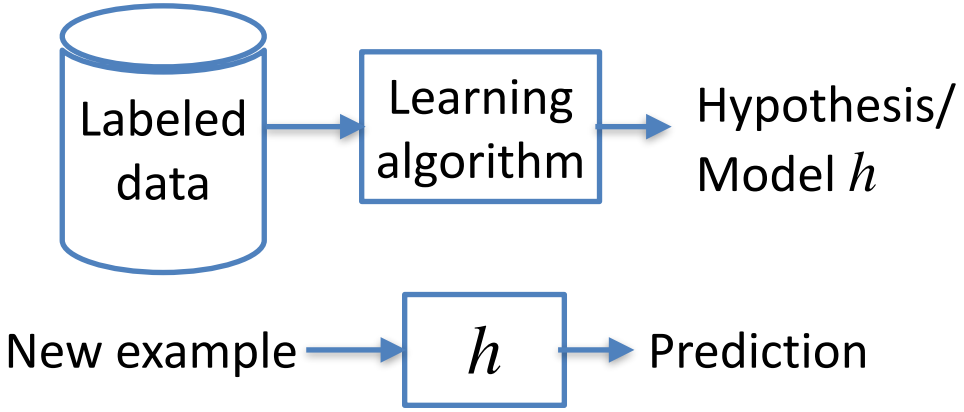
Specific learners

- Decision trees



Checkpoint: the bigger picture

Supervised learning: instances and hypotheses

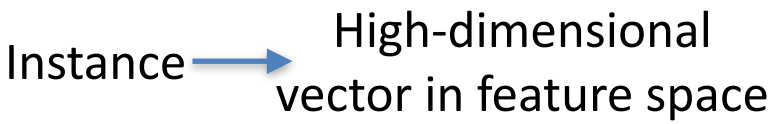


Specific learners

- Decision trees

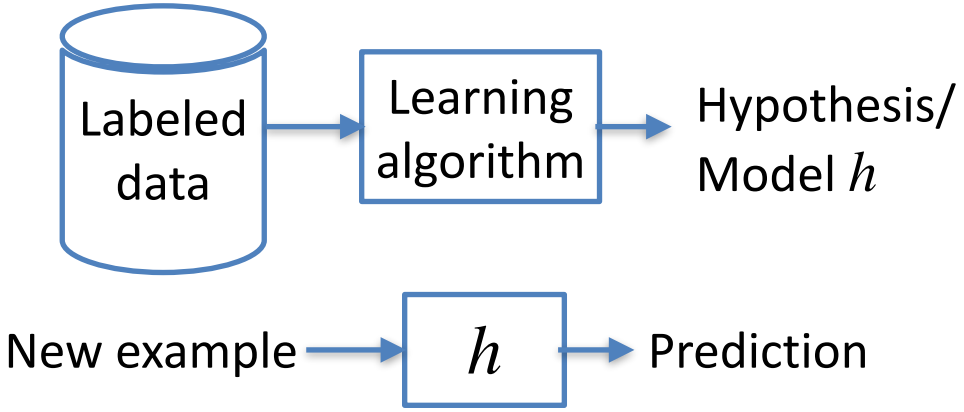
General ML ideas

- Features as high dimensional vectors



Checkpoint: the bigger picture

Supervised learning: instances and hypotheses



Specific learners

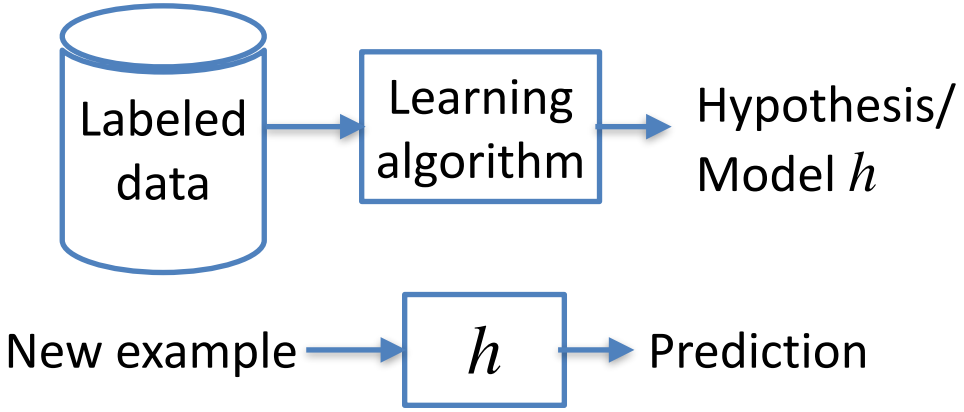
- Decision trees

General ML ideas

- Features as high dimensional vectors
- Overfitting

Checkpoint: the bigger picture

Supervised learning: instances and hypotheses



Specific learners

- Decision trees

General ML ideas → More general than just decision trees

- Features as high dimensional vectors
- Overfitting

Linear models

OVERVIEW

Where are we?

What are linear models? → A hypothesis class

- Why linear classifiers (and regressors)?
- Geometry of linear classifiers
- A notational simplification
- How do you learn a linear classifier?

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 7 outputs

We *cannot* know what the rest are without seeing them

- Think of an adversary filing in the labels every time you make a guess at a 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 d
  fu

We hav

We *cannot* know what the rest are without seeing them

- Think of an adversary filing in the labels every time you make a guess at a function

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

How could we possibly learn anything?

Solution: restrict the search space

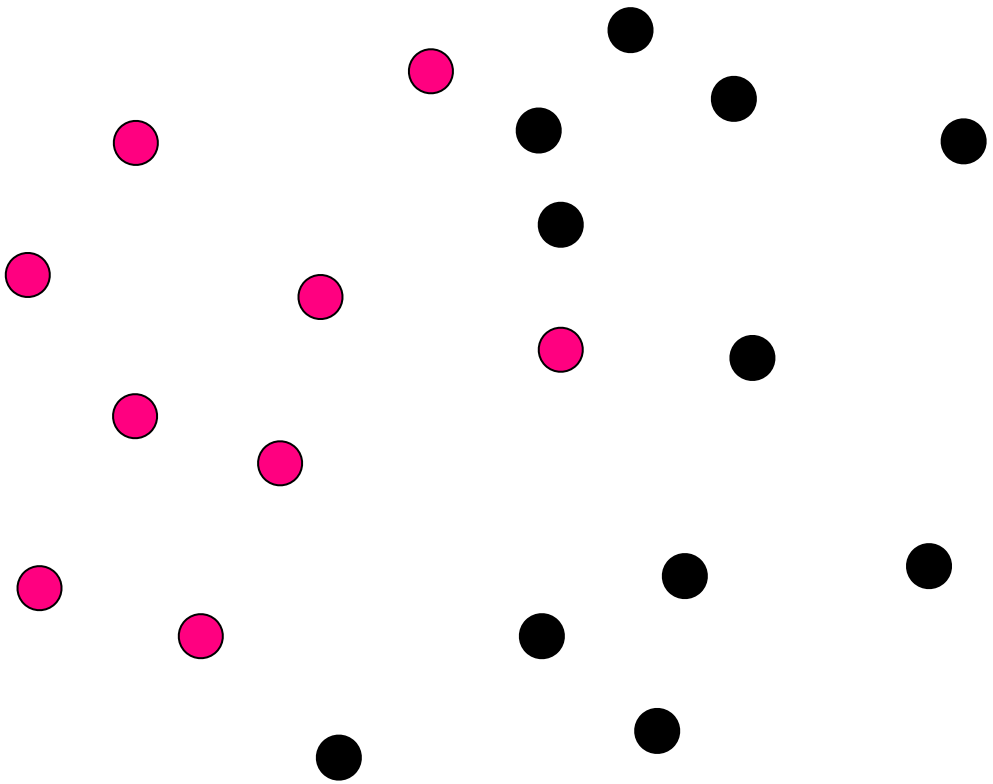
A hypothesis space is the set of possible functions we consider

We were looking at the space of all Boolean functions. Instead we choose a hypothesis space that is smaller than the space of all functions

For example:

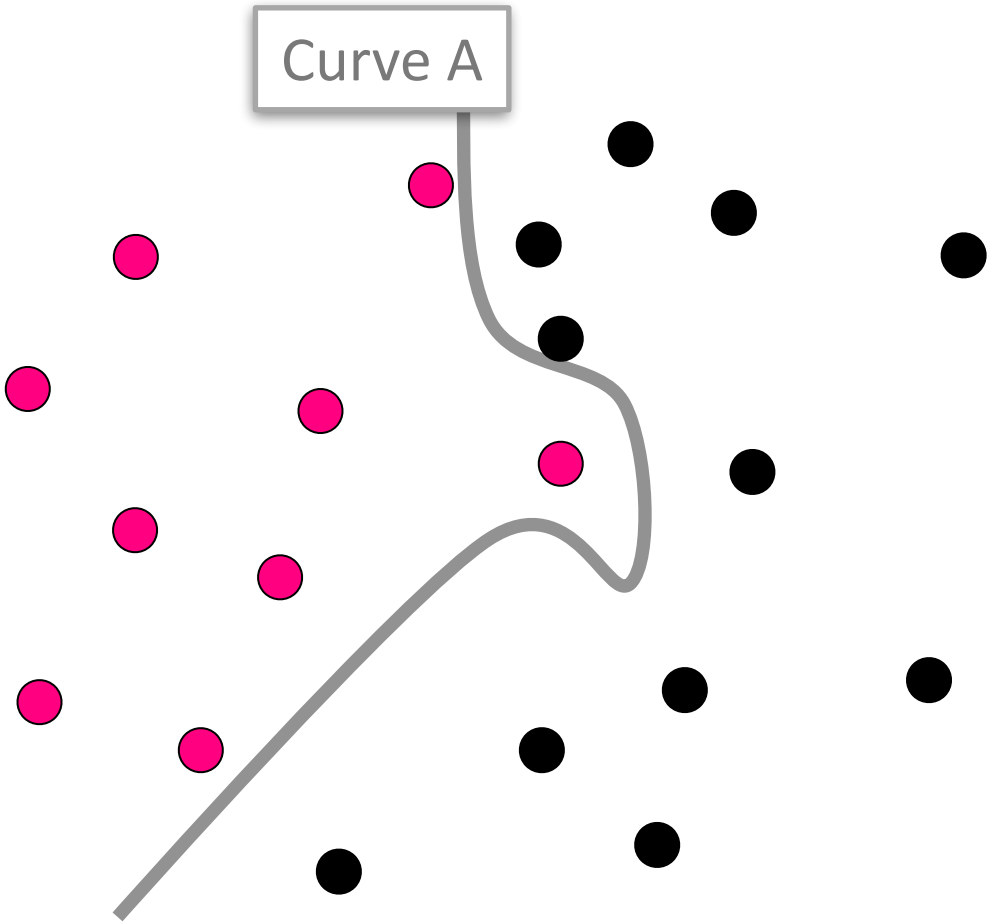
- Only simple conjunctions with 4 variables, there are 16 conjunctions without negations
- Simple disjunction
- m-of-n rules: Fix a set of n variables. At least m of them must be true
- Linear functions
- …

Which is the better classifier?



Suppose this is our training set: we have to separate pink circles from black circles

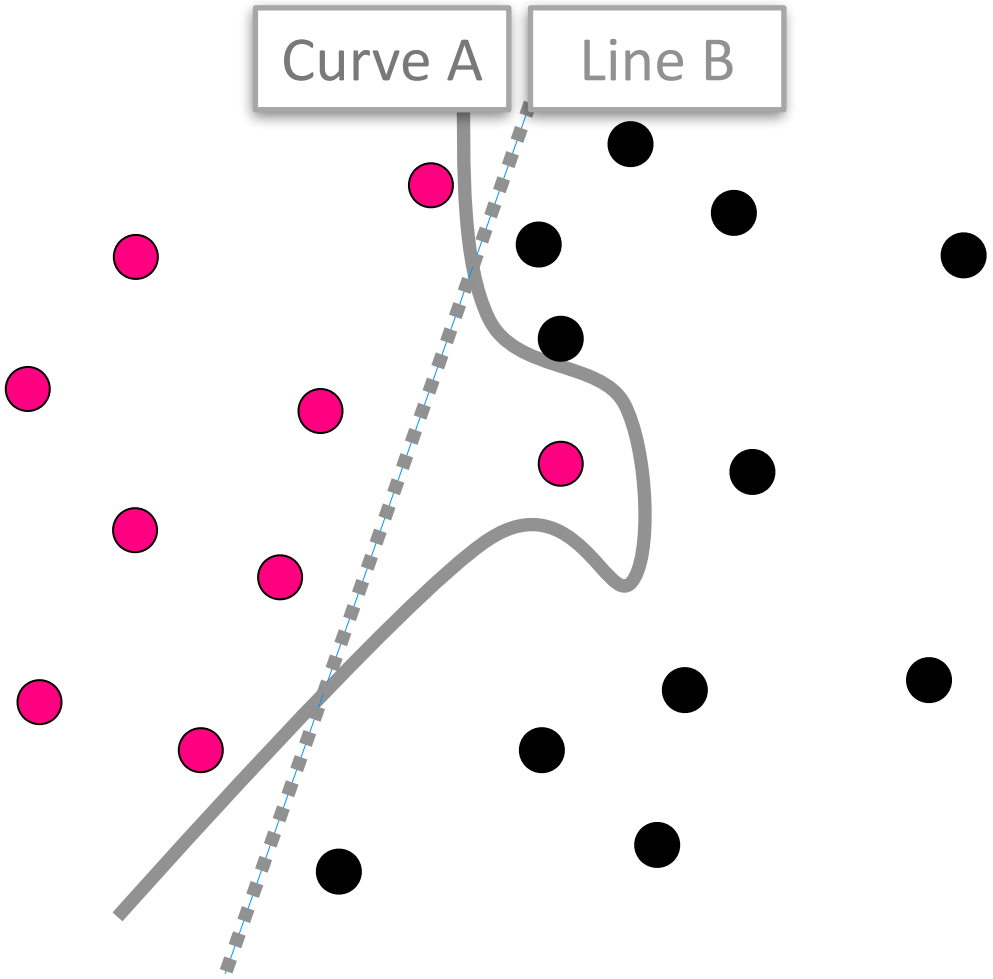
Which is the better classifier?



Suppose this is our training set: we have to separate pink circles from black circles

Curve A perfectly separates pink from black

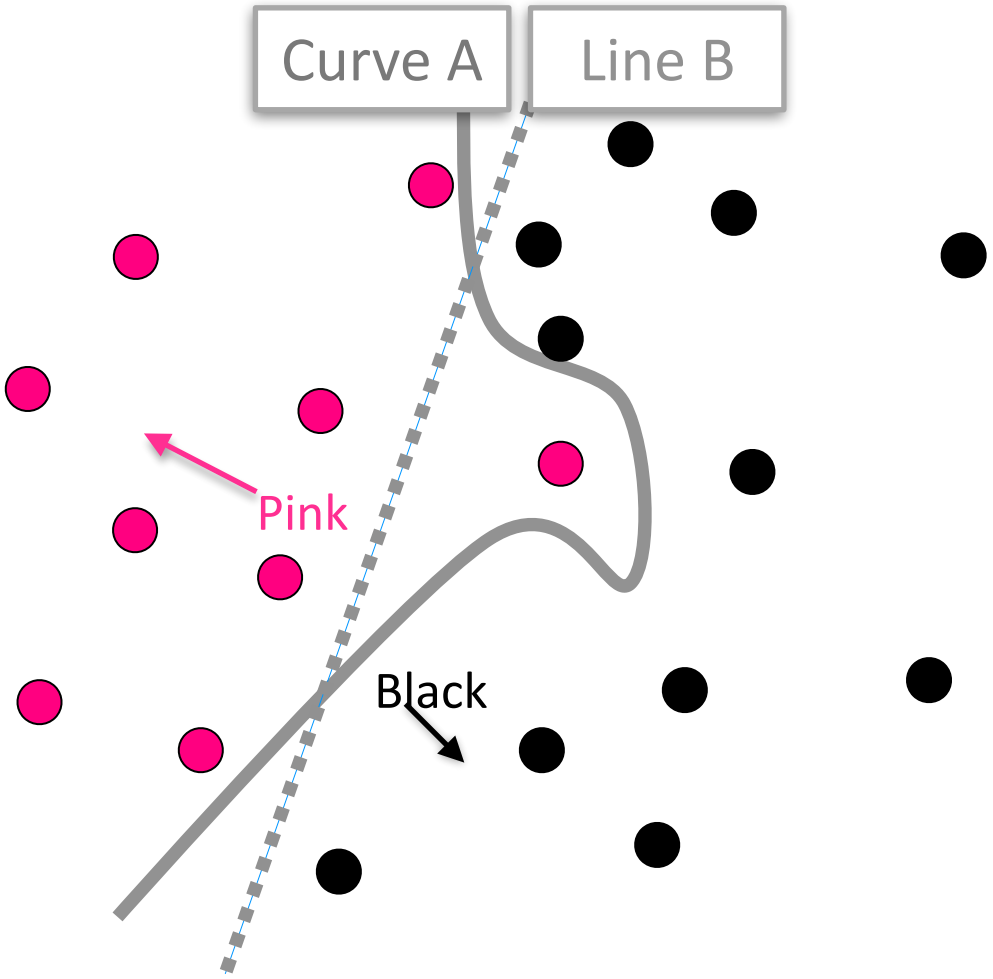
Which is the better classifier?



Suppose this is our training set: we have to separate pink circles from black circles

Curve A perfectly separates pink from black … Line B separates less well

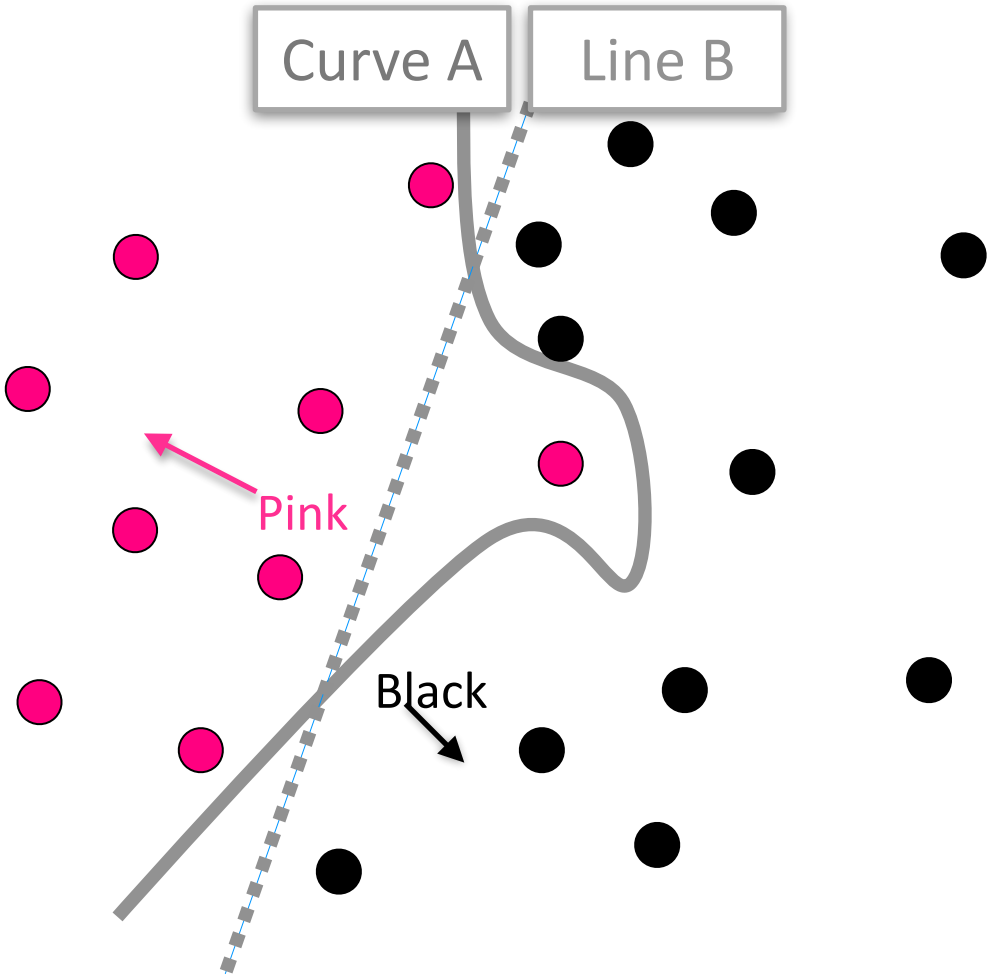
Which is the better classifier?



Suppose this is our training set: we have to separate pink circles from black circles

Curve A and Line B form decision boundaries

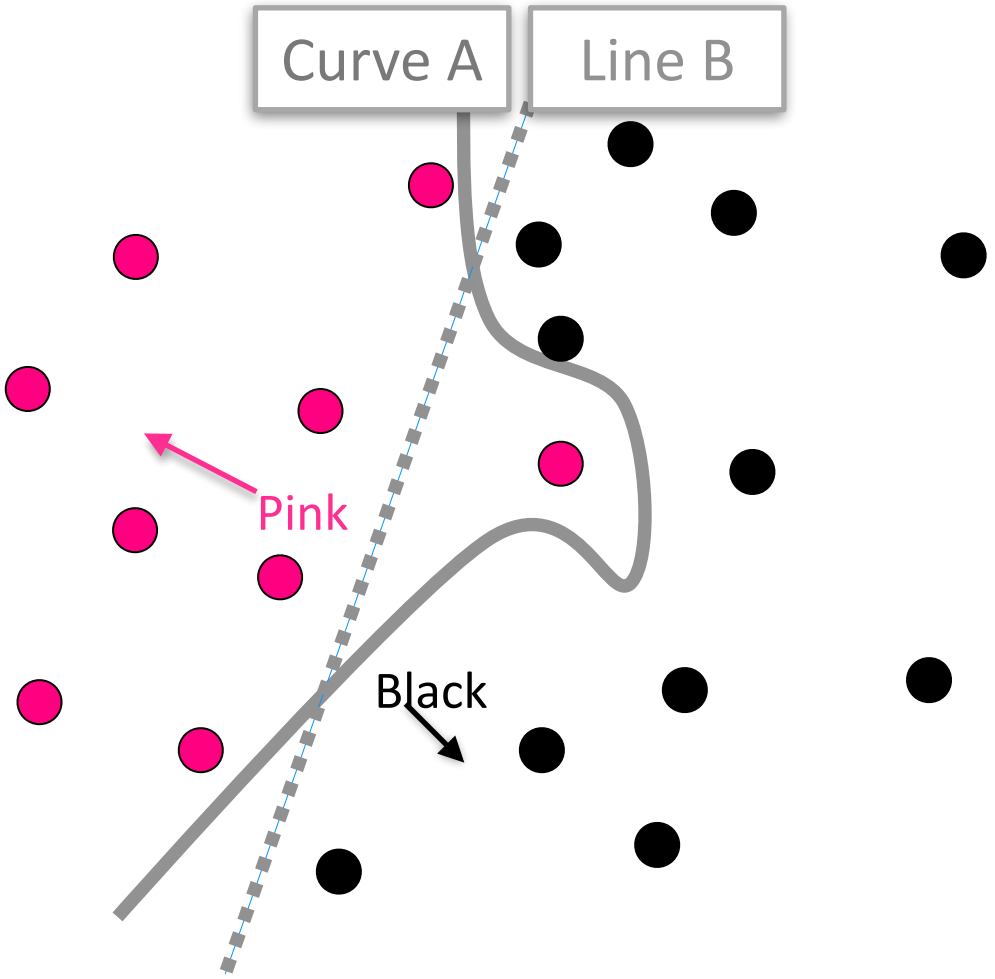
Which is the better classifier?



Suppose this is our training set: we have to separate pink circles from black circles

Which is is better? How do we define better?

Which is the better classifier?



Suppose this is our training set: we have to separate pink circles from black circles

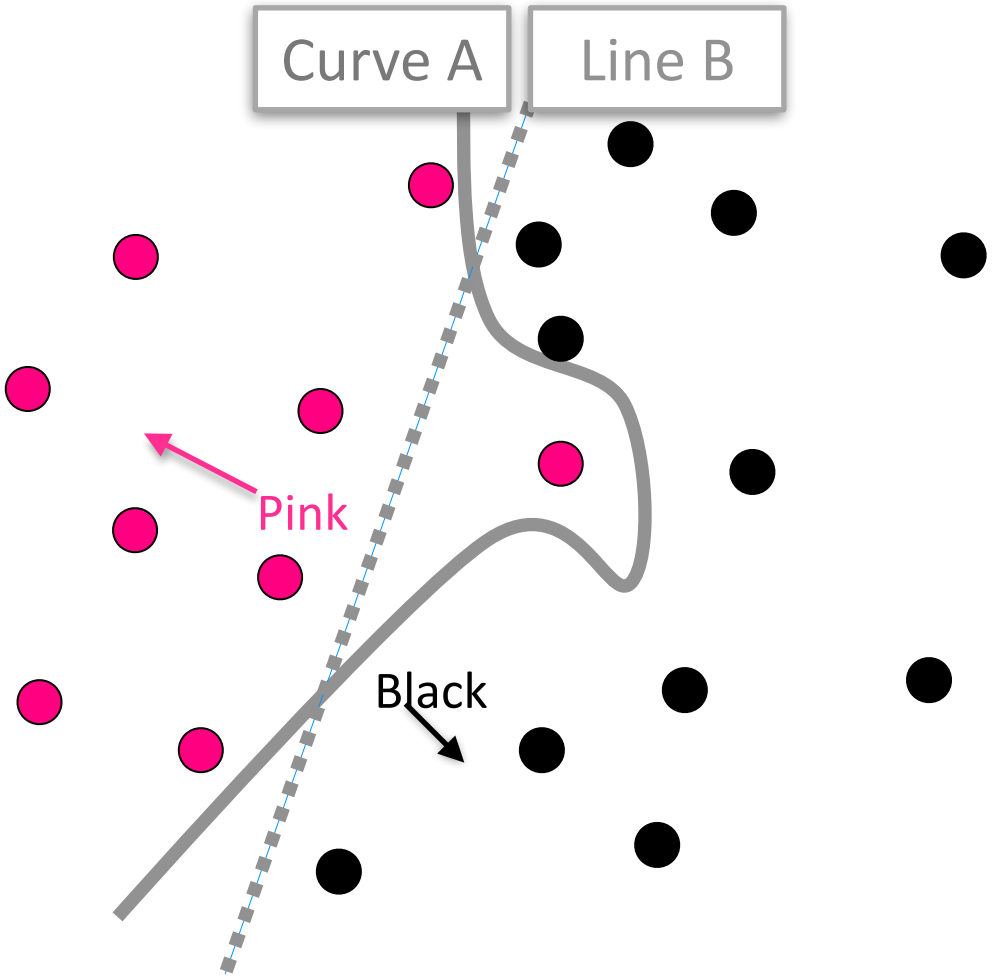
Think about overfitting

Which curve runs the risk of overfitting?

Simplicity versus accuracy

Which is is better? How do we define better?

Which is the better classifier?



Suppose this is our training set: we have to separate pink circles from black circles

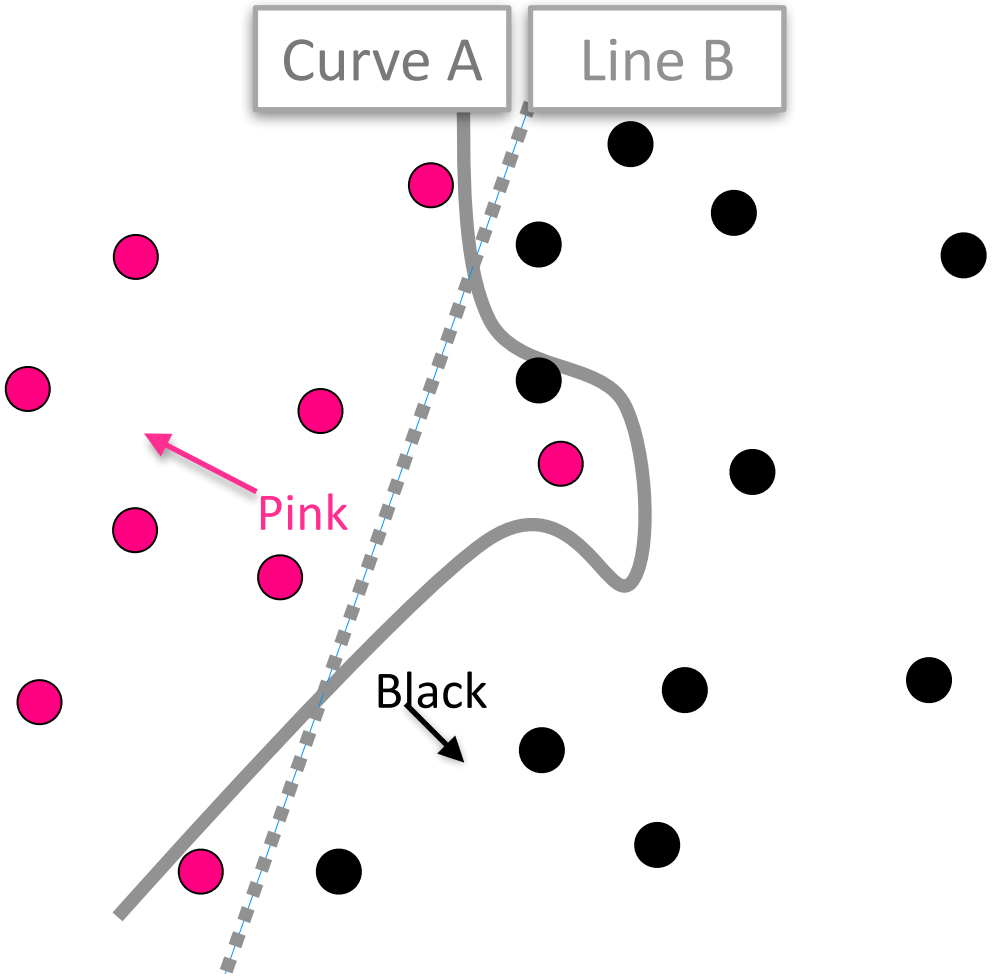
Think about overfitting

Which curve runs the risk of overfitting?

Simplicity versus accuracy

Which is is better? How do we define better?
In terms of generalization

Which is the better classifier?



Suppose this is our training set: we have to separate pink circles from black circles

Think about overfitting

Which curve runs the risk of overfitting?

Simplicity versus accuracy

If noise in data and some points move, could end up on wrong side of curve. Curve may be fitting noise

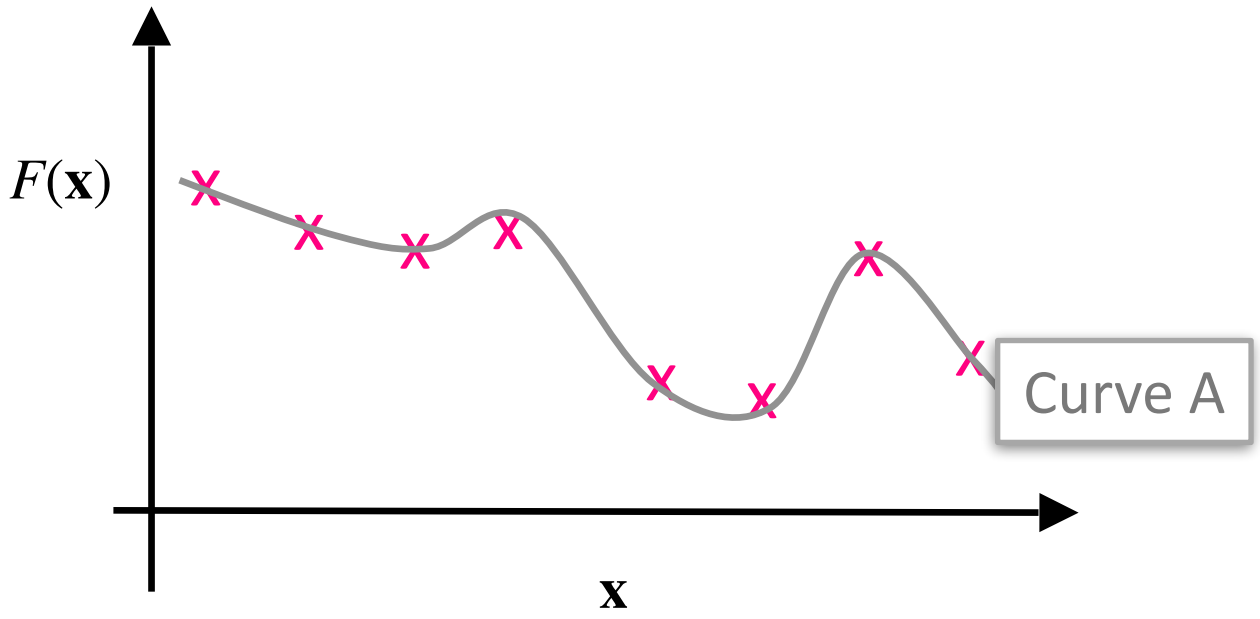
Linear classification vs. regression

Linear classification is about predicting a discrete class label

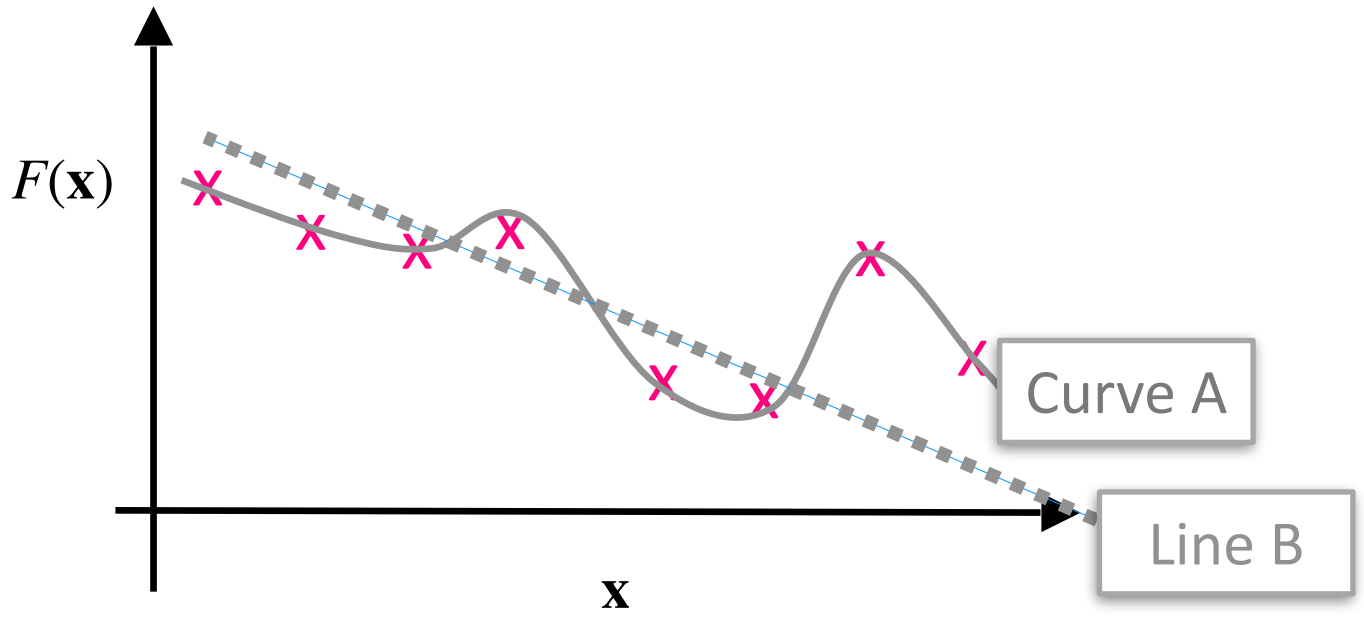
- +1 or -1
- SPAM or NOT-SPAM
- Or more than two categories

Linear regression is about predicting real valued outputs

Similar argument for regression

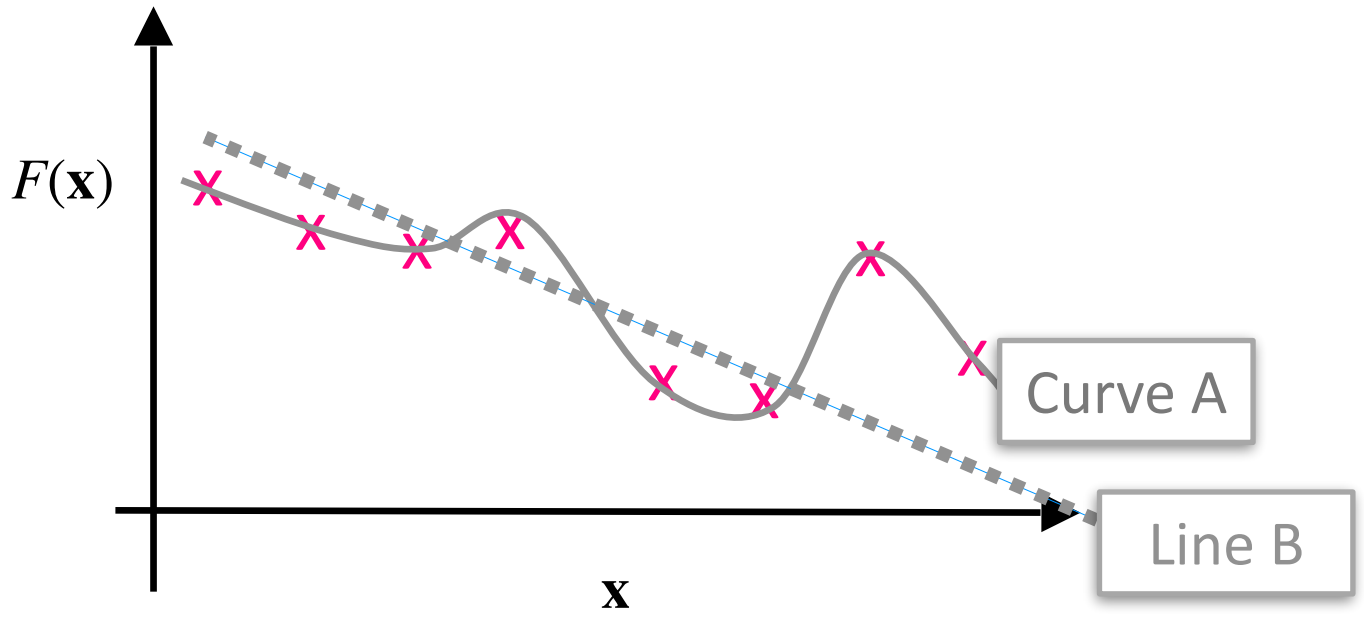


Similar argument for regression



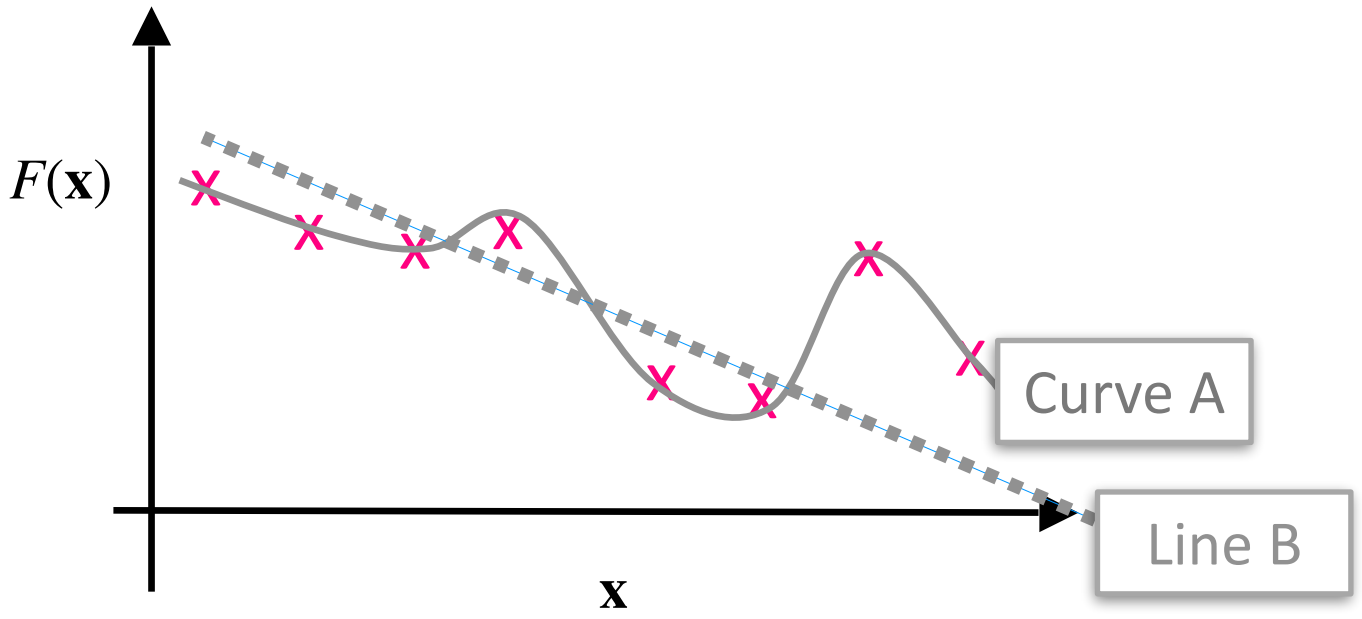
Linear regression tends to make smaller errors on new points than polynomial regression with arbitrary polynomials ($x^{17} + 100x^{68} + x^3$)

Similar argument for regression



In general, picking extremely expressive functions tends to overfit. One of the simplest functions we can pick are linear functions

Similar argument for regression



Linear classifiers and regressors: one of most studied class of functions

Why? Simplicity

Linear Classifiers

OVERVIEW

Linear classifiers

Learn a linear function that separates instances of different classes

Linear classifiers

Learn a linear function that separates instances of different classes

A linear function divides the coordinate space into two parts

- Every point is either on one side of the line (or plane or hyperplane) or the other (unless it is exactly on the line and need to break ties)

Linear classifiers

Learn a linear function that separates instances of different classes

A linear function divides the coordinate space into two parts

- Every point is either on one side of the line (or plane or hyperplane) or the other (unless it is exactly on the line and need to break ties)

This means it can only separate two classes

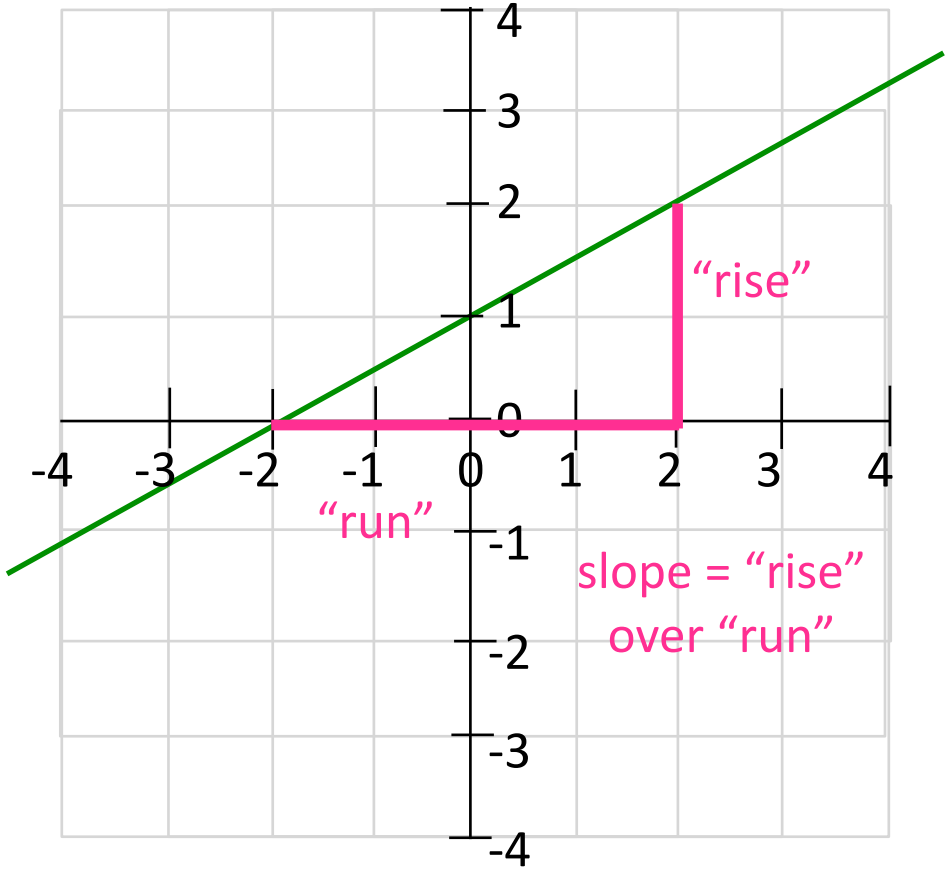
- Classification with two classes is called **binary classification**
- Conventionally, one class is called the positive class and the other is the negative class

Slope-intercept form of a line

$$f(x) = mx + b$$

slope intercept

$y = 1/2x + 1$



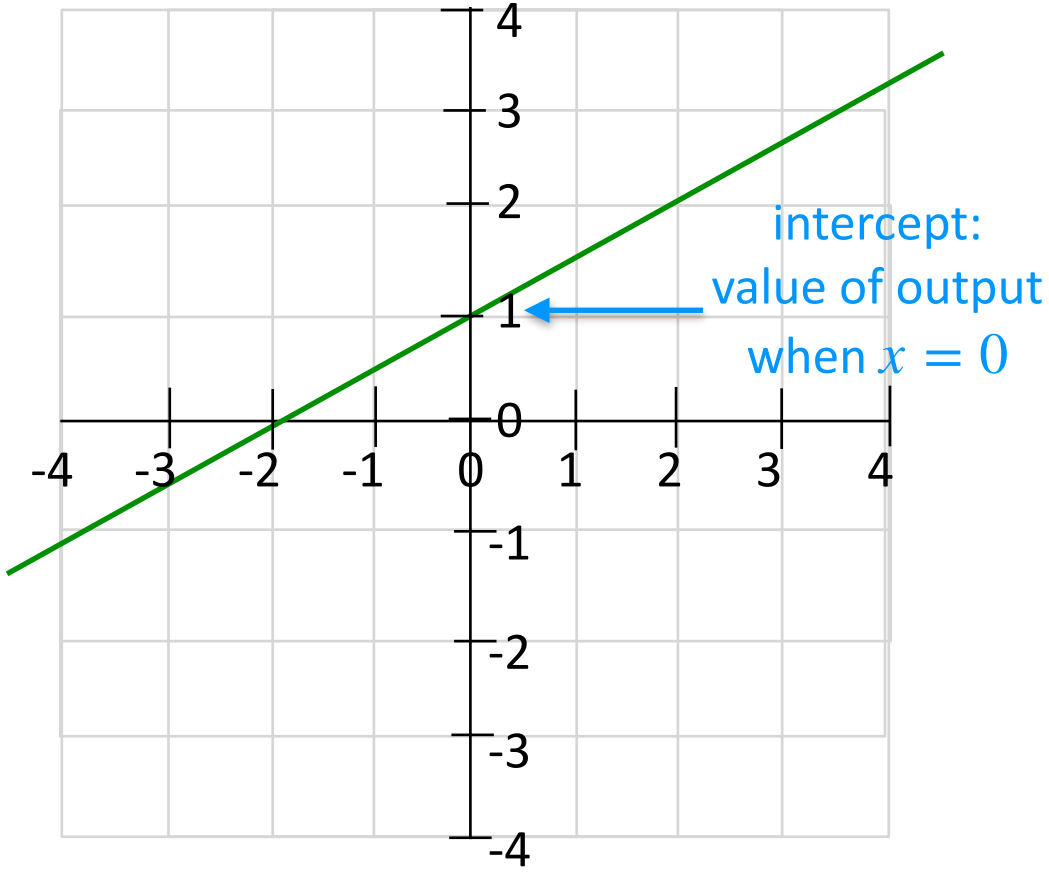
This equation uses x as a parameter. When $x = 0$ and $y = b$, the point $(0,b)$ is the intersection of the line with the y axis

Slope-intercept form of a line

$$f(x) = mx + b$$

slope intercept

$y = 1/2x + 1$



This equation uses x as a parameter. When $x = 0$ and $y = b$, the point $(0,b)$ is the intersection of the line with the y axis

General equation for a line

Views line as a geometric object rather than graph of function

- Generalizes to 3+ dimensions unlike slope-intercept form

$$ax + by = c$$

a, b, and c are real numbers

a and b are not both zero

General equation for a line

Views line as a geometric object rather than graph of function

- Generalizes to 3+ dimensions unlike slope-intercept form

$$ax + by = c$$

a, b, and c are real numbers

a and b are not both zero

Can convert to slope-intercept form by solving for y

$$y = \frac{-ax}{b} + \frac{c}{b}$$

What is a linear classifier?

Instance space X is d-dimensional: an instance $\mathbf{x} \in X$ is d-dimensional vector,

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

What is a linear classifier?

Instance space X is d-dimensional: an instance $\mathbf{x} \in X$ is d-dimensional vector,

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

What is a linear classifier?

Instance space X is d-dimensional: an instance $\mathbf{x} \in X$ is d-dimensional vector,

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

Goal is to divide this d-dimensional space given by the inputs into two parts

What is a linear classifier?

Instance space X is d-dimensional: an instance $\mathbf{x} \in X$ is d-dimensional vector,

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 to the following classification rule

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

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

Dot product

Dot product is defined as:

$$\mathbf{w}^T\mathbf{x} \text{ or } \mathbf{w} \cdot \mathbf{x} \text{ where } \mathbf{w}^T\mathbf{x} = \sum_{i=1}^{k} w_i x_i$$

Measures amount one vector goes in the direction of the other

Ex:

$$\mathbf{w} = < 5.13,\ 1.08,\ -0.03,\ 7.29 >$$

$$\mathbf{x} = < x_1, x_2, x_3, x_4 >$$

$$\mathbf{w}^T\mathbf{x} = 5.13x_1 + 1.08x_2 - 0.03x_3 + 7.29x_4$$

If dot product of two vectors is zero: means the two vectors are orthogonal (90° angle)

What is a linear classifier?

Instance space X is d-dimensional: an instance $\mathbf{x} \in X$ is d-dimensional vector,

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 to the following classification rule

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

$$\text{if } \mathbf{w}^T\mathbf{x} + b \geq 0 \Rightarrow y = +1$$

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

Given input produce one of 2 possible labels

b is called the ***bias*** term

What is a linear classifier?

Instance space X is d-dimensional: an instance $\mathbf{x} \in X$ is d-dimensional vector,

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 to the following classification rule

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

$$\text{if } \mathbf{w}^T\mathbf{x} + b \geq 0 \Rightarrow y = +1$$

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

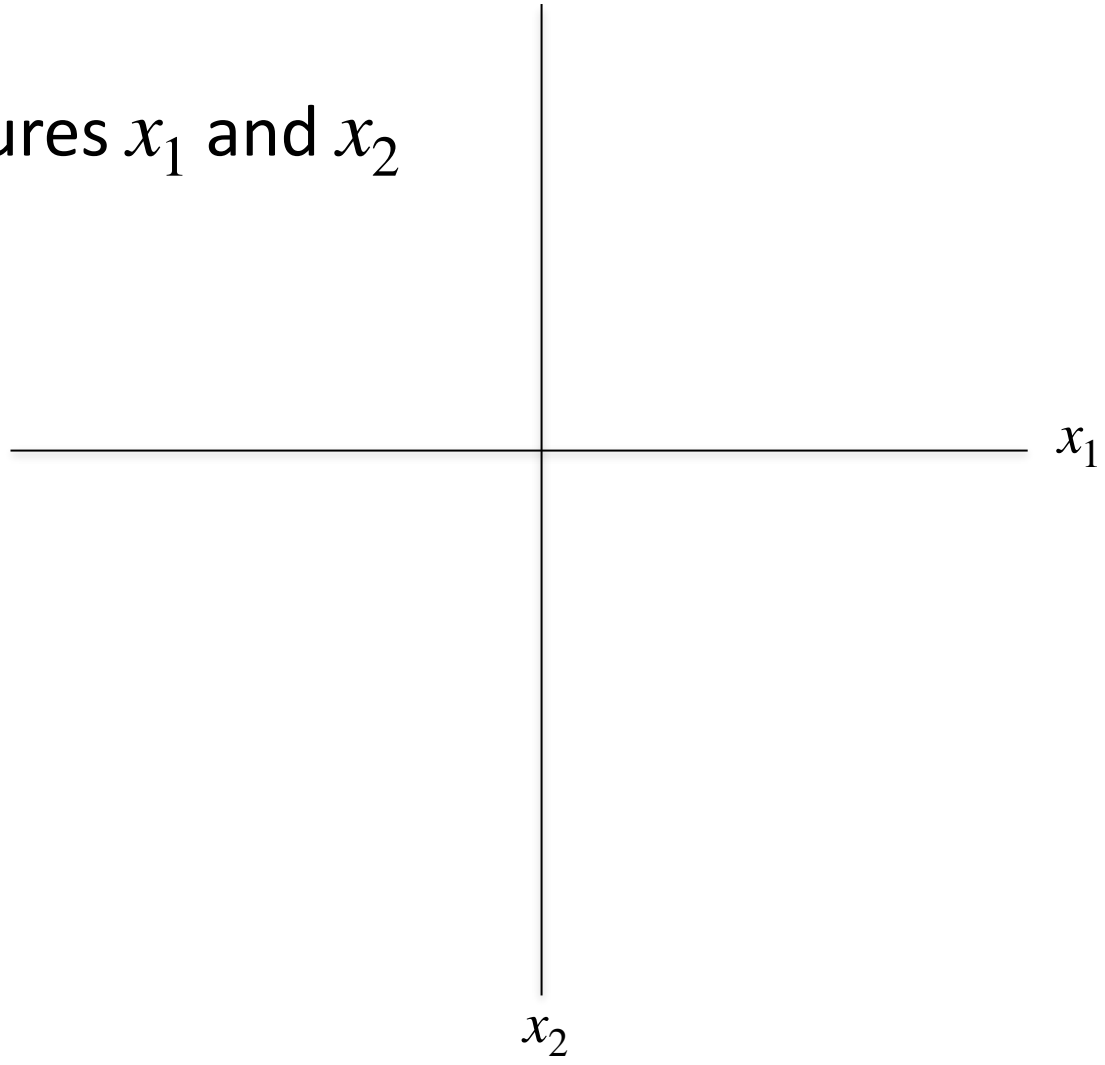
$\mathbf{w}^T\mathbf{x} + b = 0$ gives the decision boundary: a $d-1$ dimensional hyperplane in the d-dimensional instance

If 2-dimensions, only need a line to separate

The geometry of a linear classifier

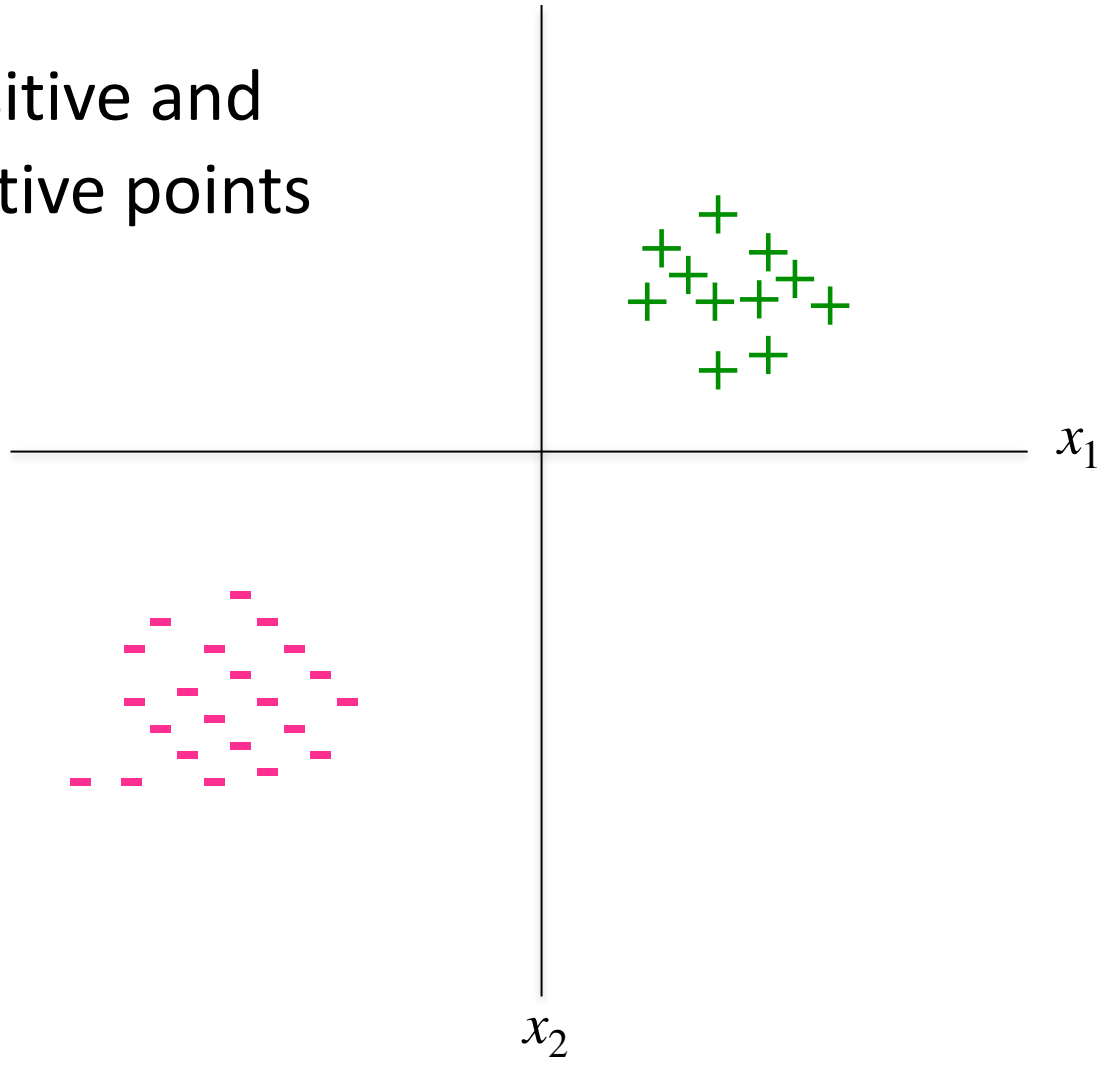
An illustration in two dimensions

Two features x_1 and x_2



The geometry of a linear classifier

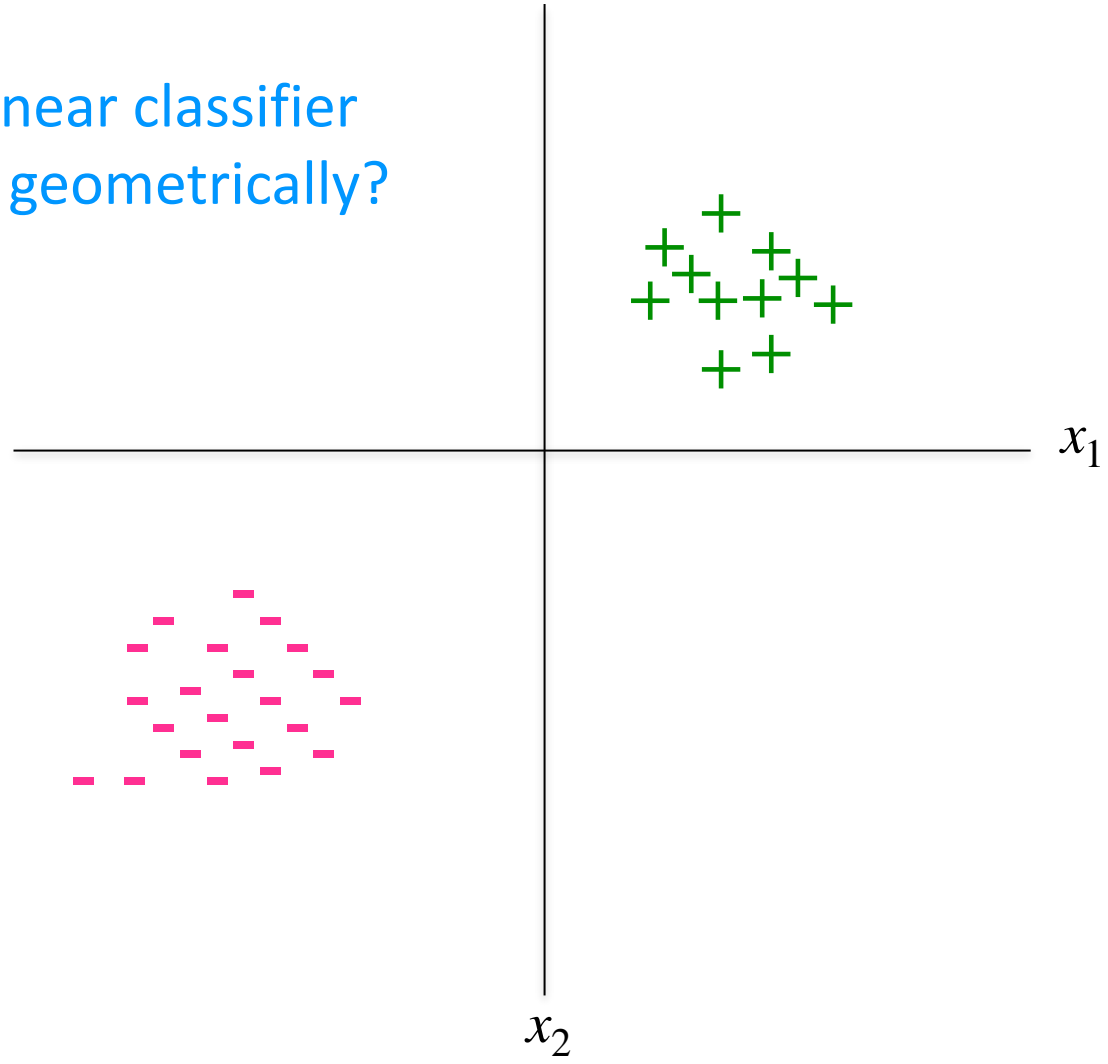
Positive and
negative points



The geometry of a linear classifier

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

What is linear classifier describing geometrically?

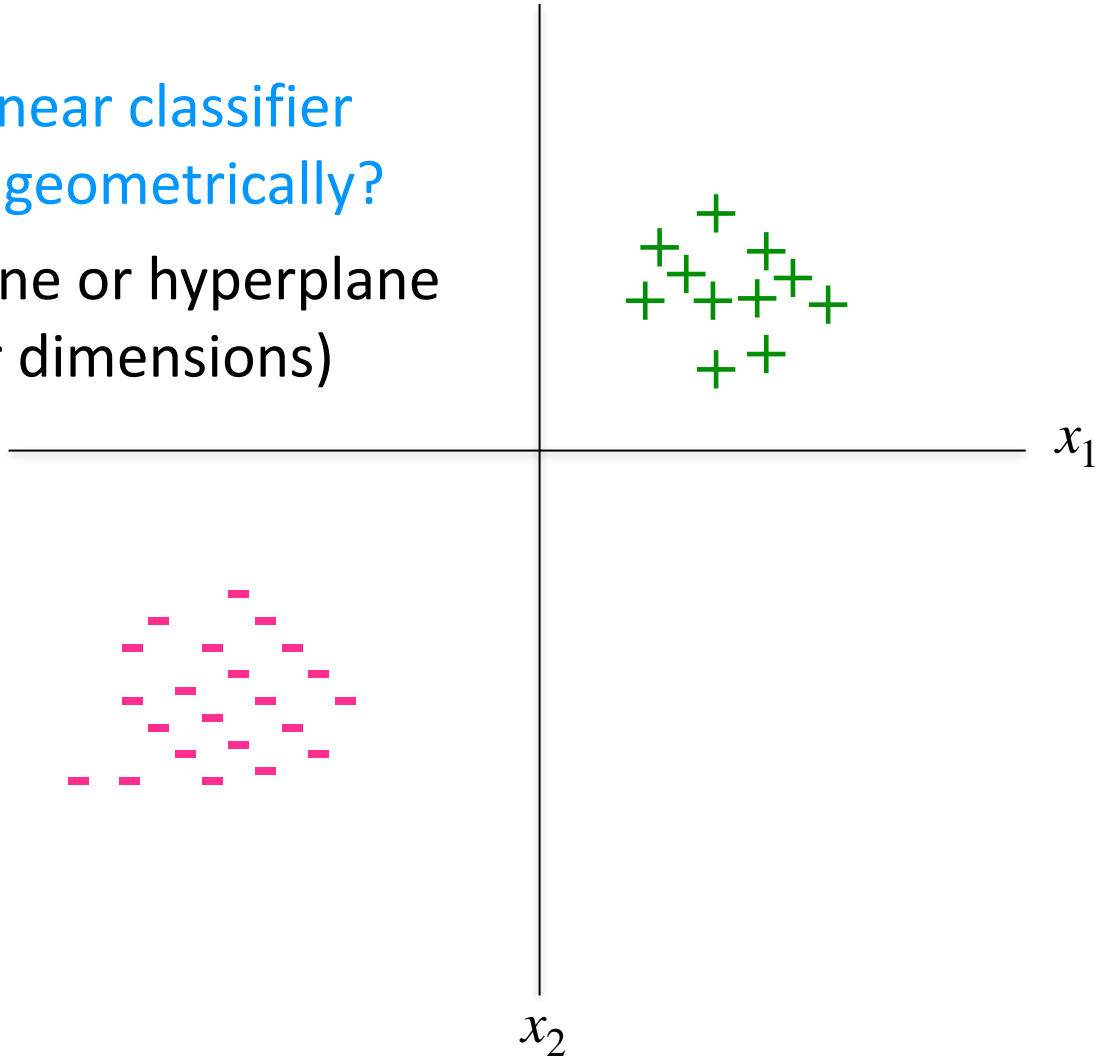


The geometry of a linear classifier

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

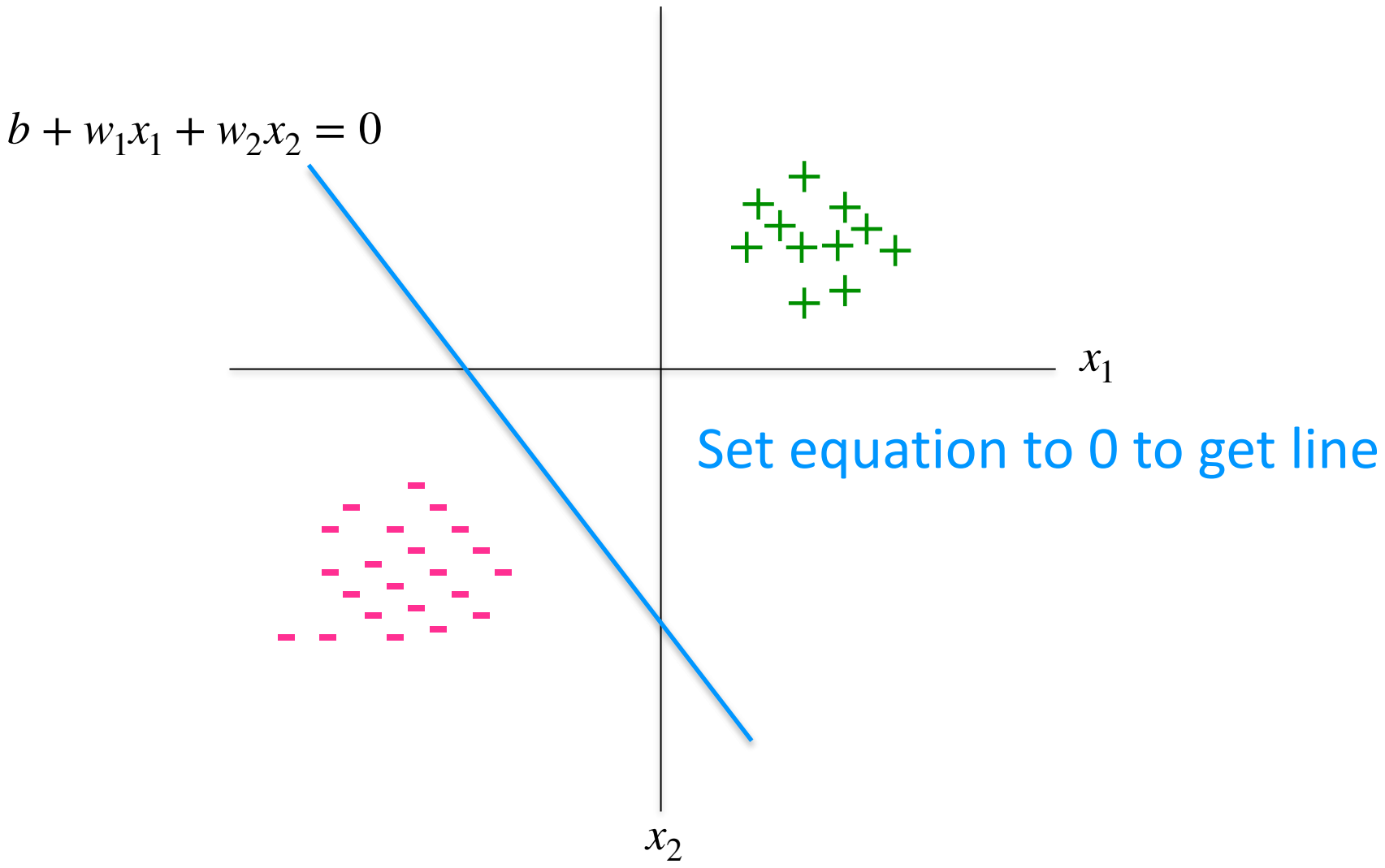
What is linear classifier describing geometrically?

A line (or plane or hyperplane in higher dimensions)



The geometry of a linear classifier

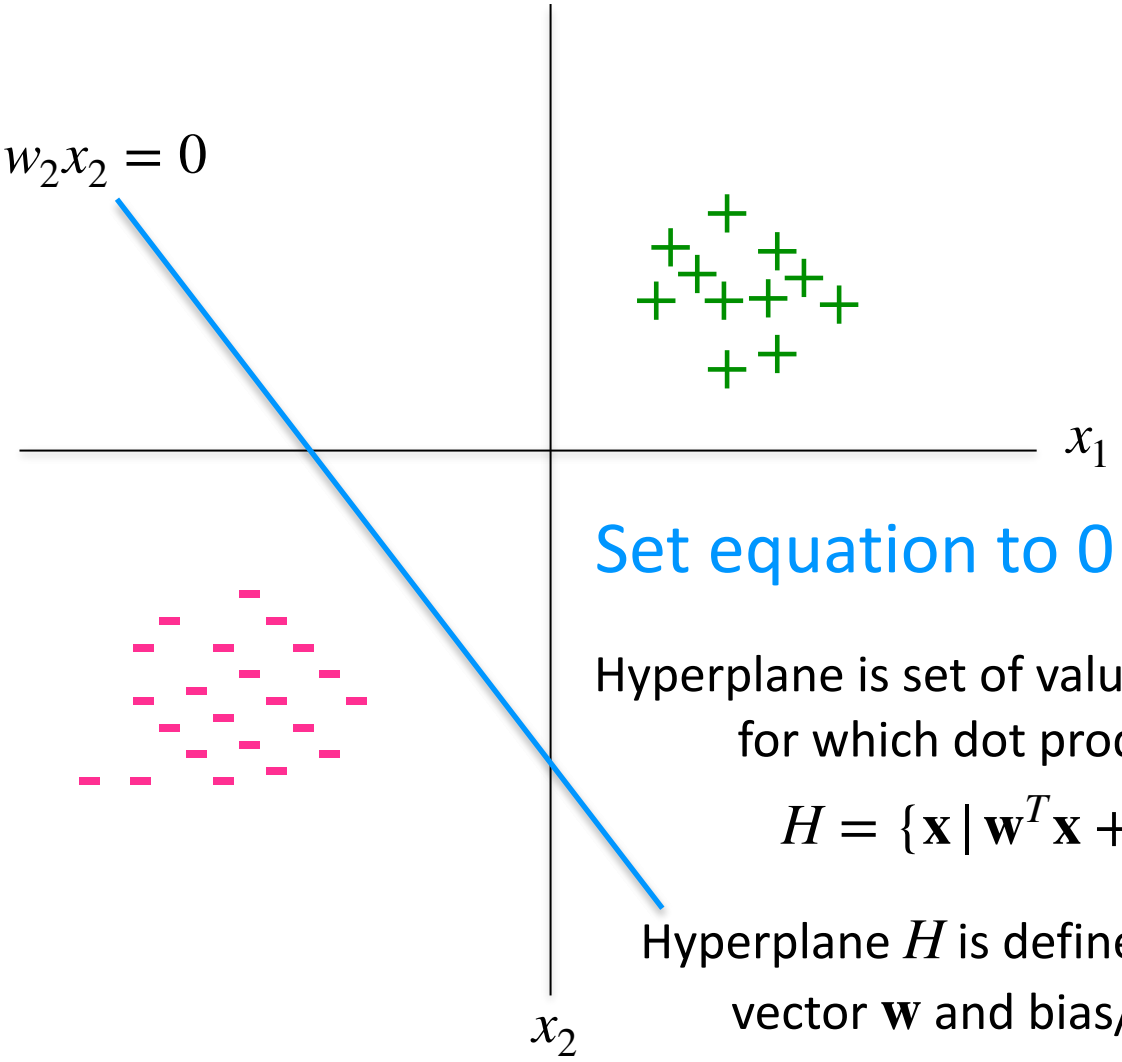
$$\text{sgn}(b + w_1 x_1 + w_2 x_2)$$



The geometry of a linear classifier

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

$$b + w_1 x_1 + w_2 x_2 = 0$$



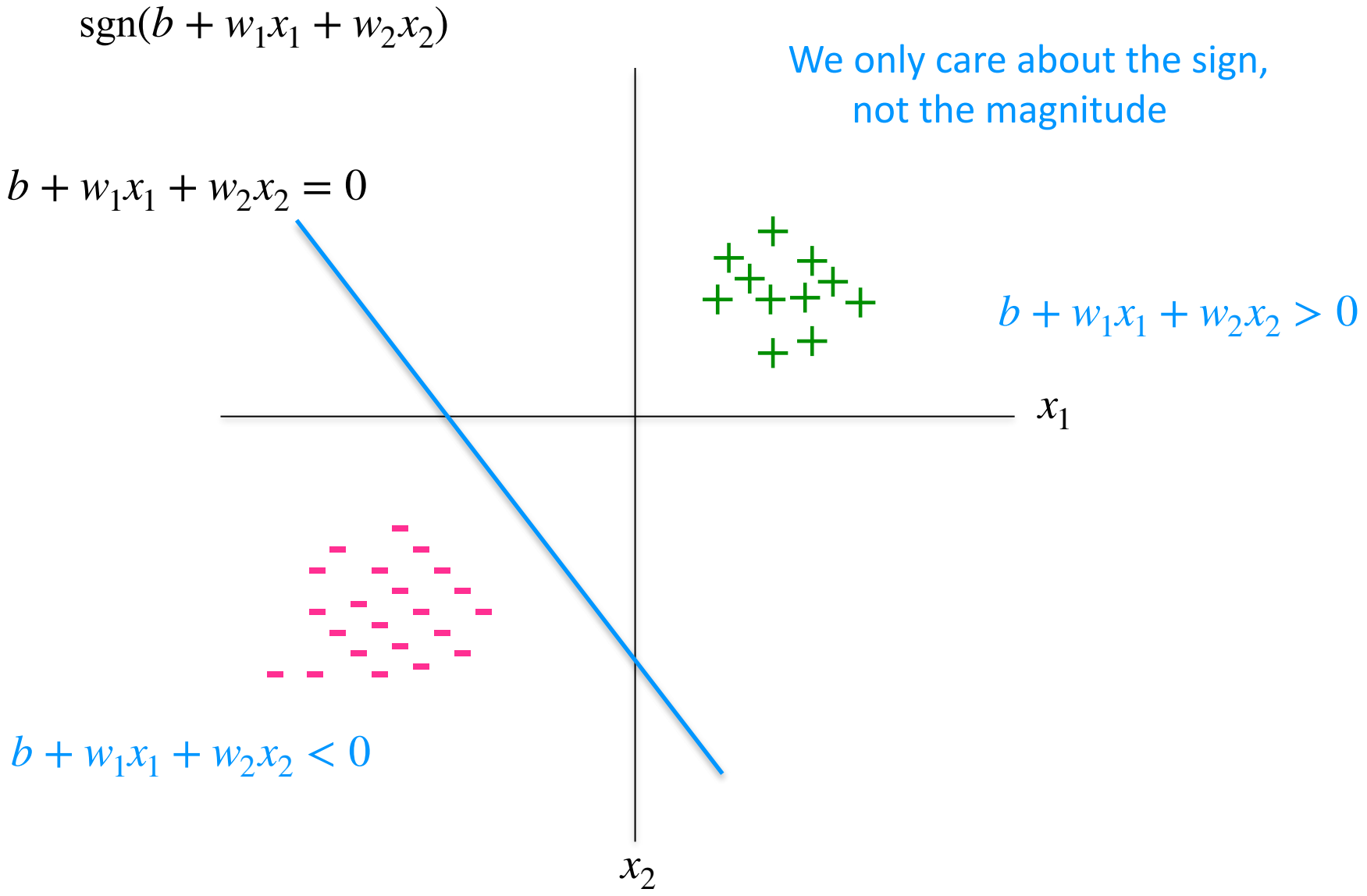
Set equation to 0 to get line

Hyperplane is set of values of x_1 and x_2 for which dot product is 0

$$H = \{\mathbf{x} \mid \mathbf{w}^T \mathbf{x} + b = 0\}$$

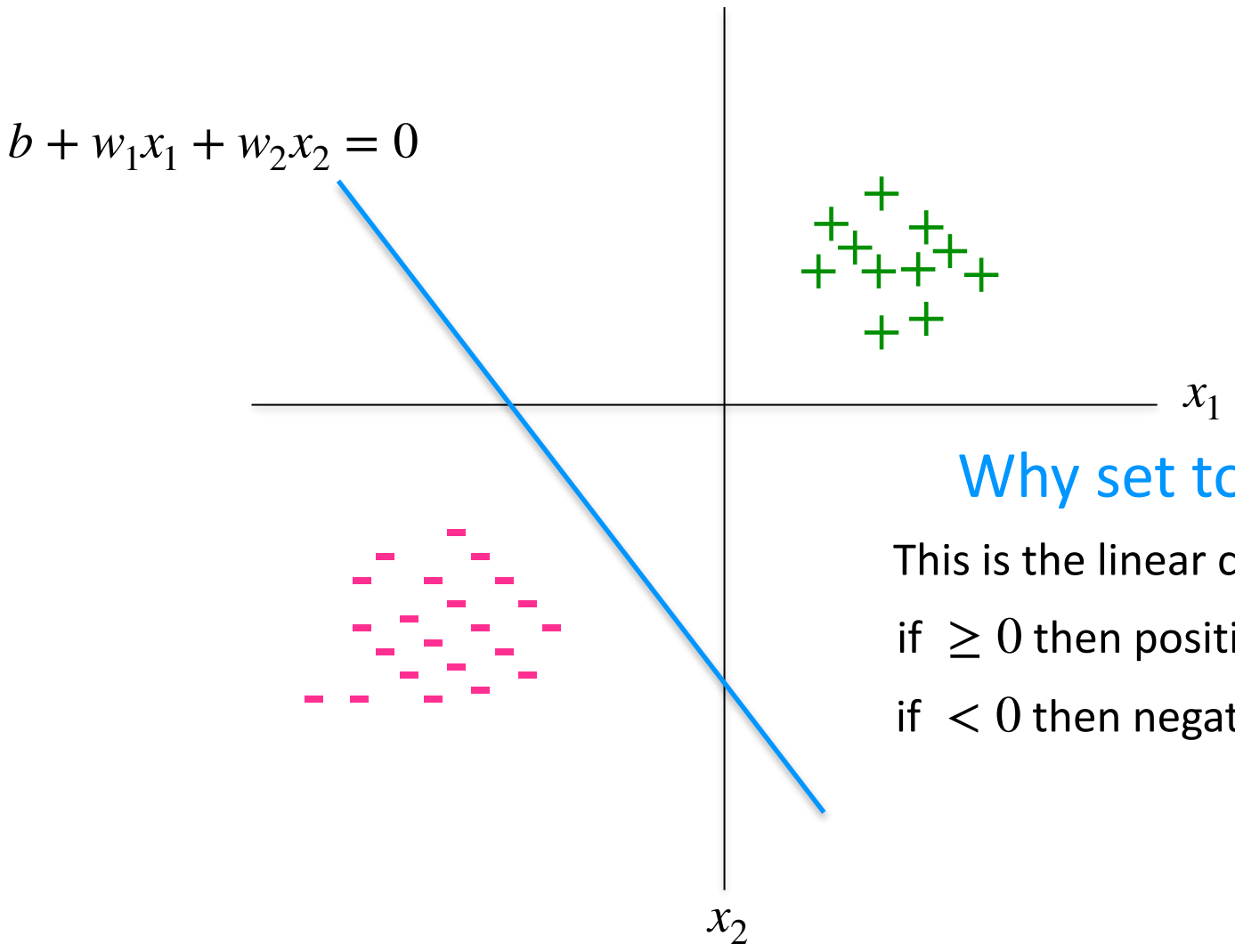
Hyperplane H is defined by weight vector $\mathbf{w}$ and bias/offset b

The geometry of a linear classifier



The geometry of a linear classifier

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



Why set to 0?

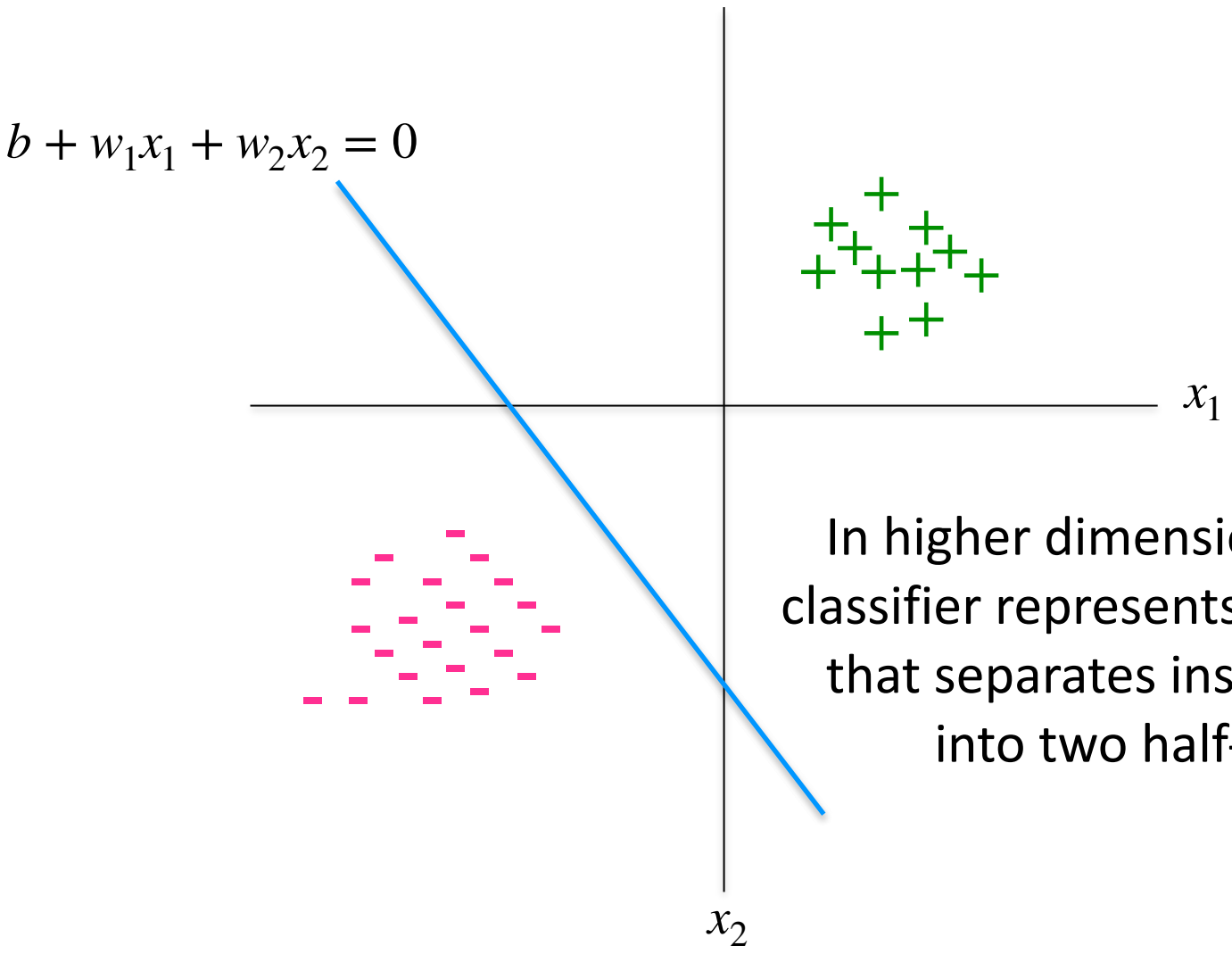
This is the linear classifier.

if ≥ 0 then positive label

if < 0 then negative label

The geometry of a linear classifier

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



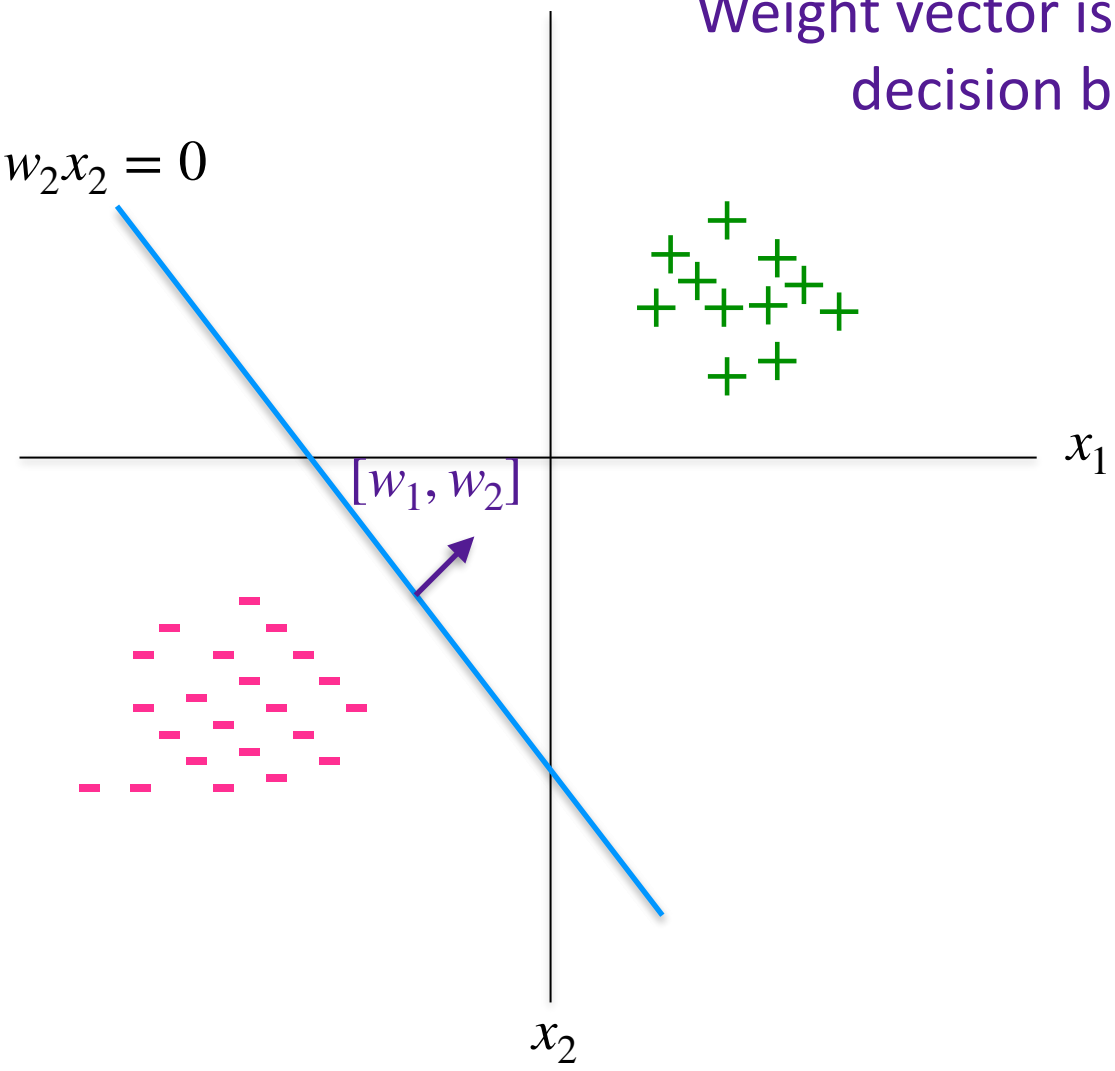
In higher dimensions, a linear classifier represents a *hyperplane* that separates instance space into two half-spaces

The geometry of a linear classifier

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

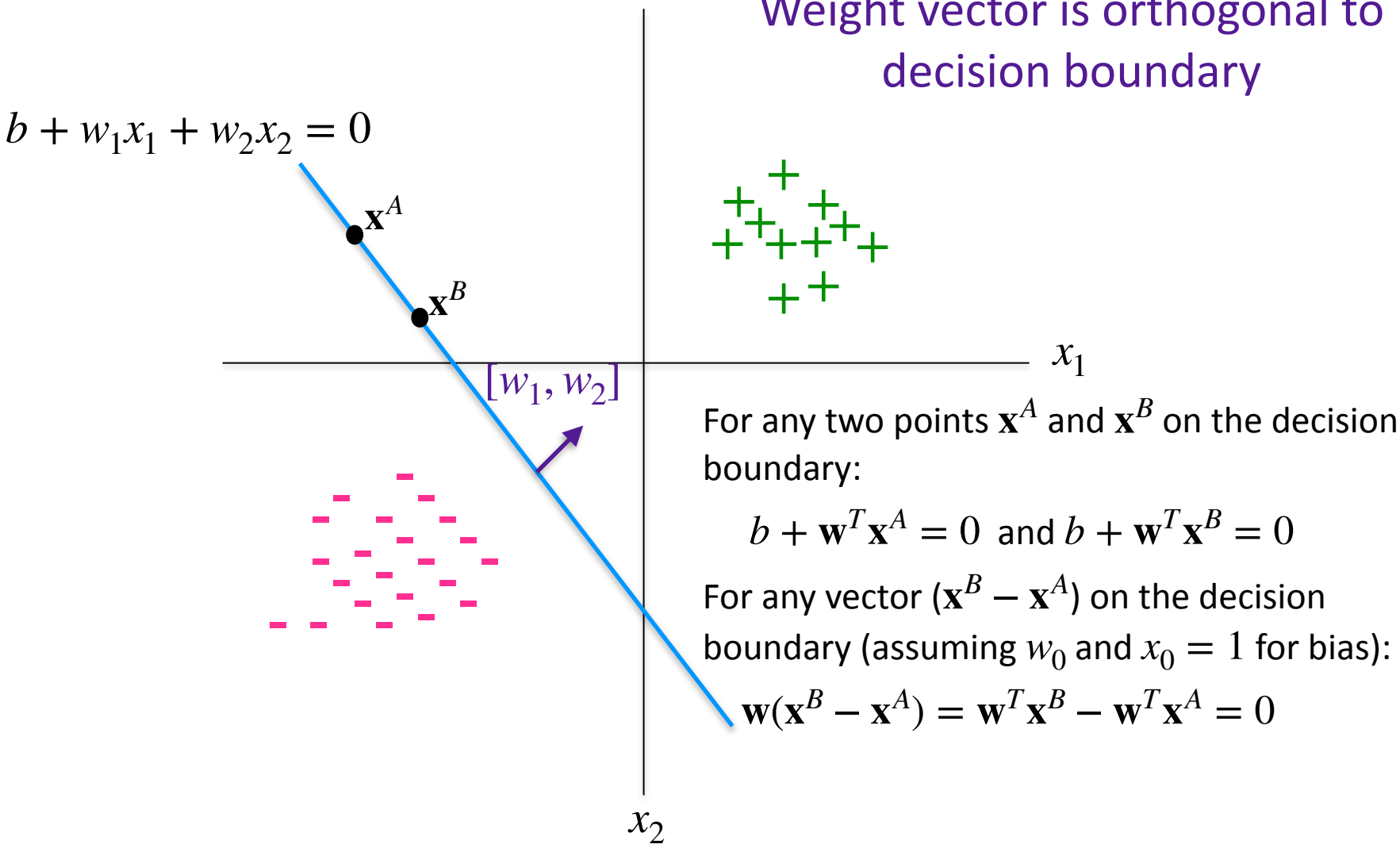
Weight vector is orthogonal to decision boundary

$b + w_1 x_1 + w_2 x_2 = 0$



The geometry of a linear classifier

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



Weight vector is orthogonal to decision boundary

For any two points $\mathbf{x}^A$ and $\mathbf{x}^B$ on the decision boundary:

$$b + \mathbf{w}^T\mathbf{x}^A = 0 \text{ and } b + \mathbf{w}^T\mathbf{x}^B = 0$$

For any vector $(\mathbf{x}^B - \mathbf{x}^A)$ on the decision boundary (assuming w_0 and $x_0 = 1$ for bias):

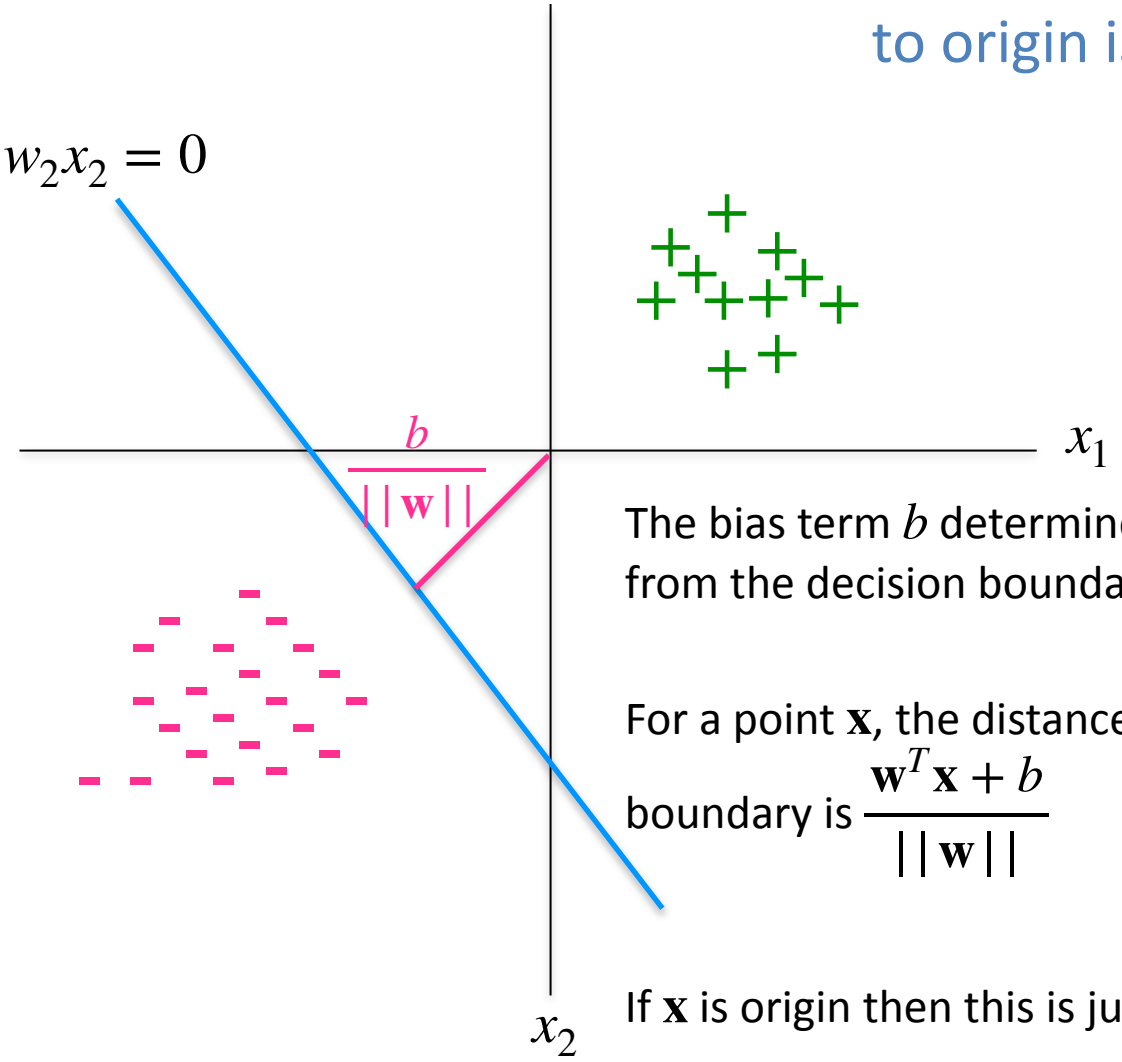
$$\mathbf{w}(\mathbf{x}^B - \mathbf{x}^A) = \mathbf{w}^T\mathbf{x}^B - \mathbf{w}^T\mathbf{x}^A = 0$$

If dot product of two vectors is zero: means the two vectors are orthogonal (90° angle)

The geometry of a linear classifier

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

$$b + w_1 x_1 + w_2 x_2 = 0$$



Distance of line or hyperplane to origin is b

The bias term b determines distance from the decision boundary to the origin

For a point $\mathbf{x}$, the distance to the decision boundary is $\frac{\mathbf{w}^T\mathbf{x} + b}{||\mathbf{w}||}$

If $\mathbf{x}$ is origin then this is just $\frac{b}{||\mathbf{w}||}$

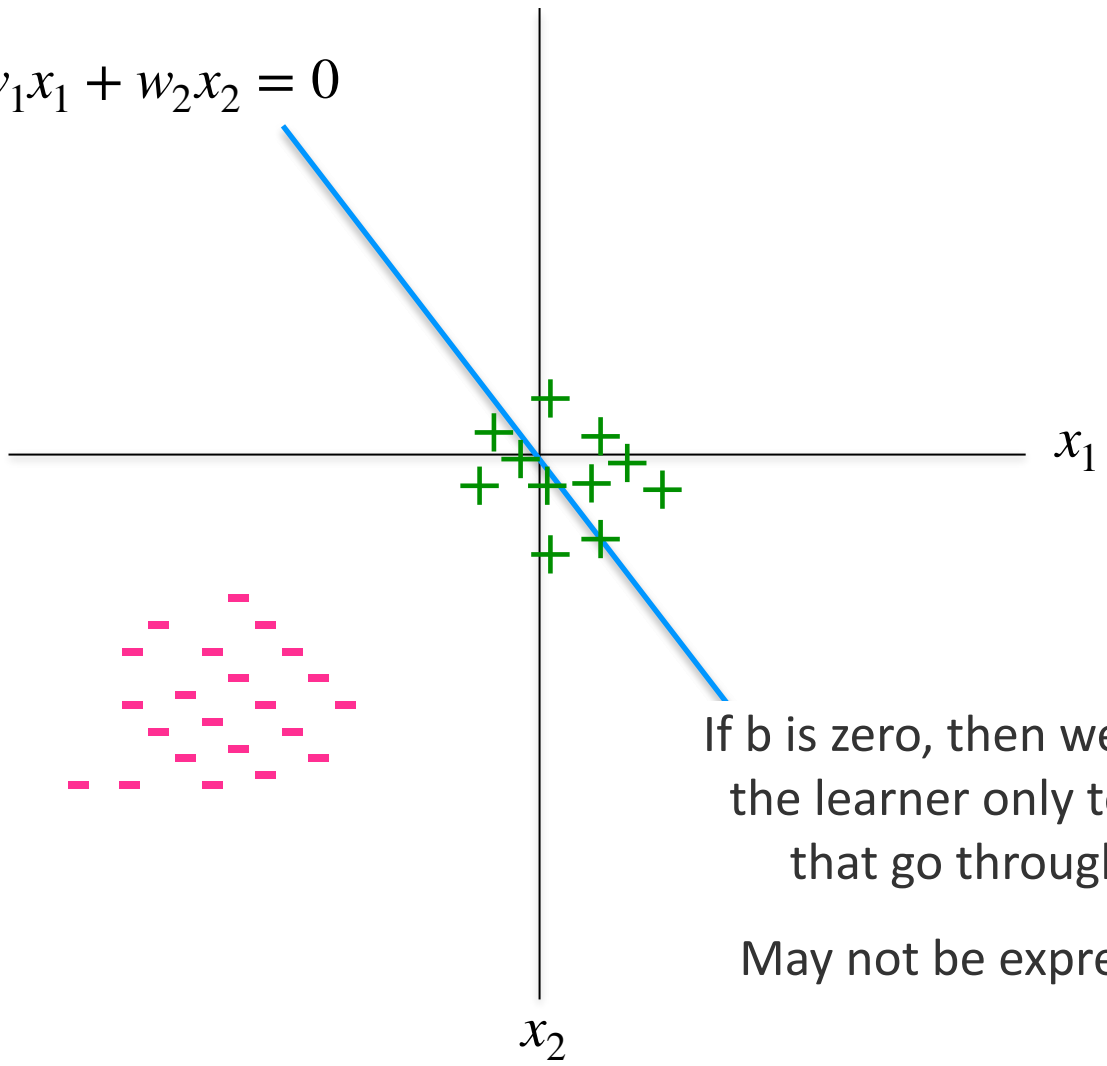
Vector length

Vector length, aka l_2 norm

$$||\mathbf{x}|| = \sqrt{\sum_i x_i^2}$$

Why is the bias term needed?

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



If b is zero, then we are restricting the learner only to hyperplanes that go through the origin

May not be expressive enough

Simplifying notation

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

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

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

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

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

Increases dimensionality by one

Equivalent to adding a feature that is 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 $\mathbf{w}^T\mathbf{x} + b$ is the same as $\mathbf{w}'^T\mathbf{x}'$

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

Increases dimensionality by one

Equivalent to adding a feature that is constant: always 1

In the increased dimensional space, the vector 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 $\mathbf{w}^T\mathbf{x} + b$ is the same as $\mathbf{w}'^T\mathbf{x}'$

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

Increases dimensionality by one

Equivalent to adding a feature that is constant: always 1

In the increased dimensional space, the vector 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.

Many standard learning algorithms are linear classifiers

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

Naive Bayes classifier: a simple linear classifier with a probabilistic interpretation

Logistic regression: another probabilistic linear classifier, bears similarity to linear regression and support vector machines

Many standard learning algorithms are linear classifiers

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

Naive Bayes classifier: a simple linear classifier with a probabilistic interpretation

Logistic regression: another probabilistic linear classifier, bears similarity to linear regression and support vector machines

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

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

Linear Classifiers

EXAMPLE

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 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 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 b 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 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

Example

Suppose we want to predict whether a web user will click on an ad for a refrigerator

Four features:

- Recently searched “refrigerator repair”
- Recently searched “refrigerator reviews”
- Recently bought a refrigerator
- Has clicked on any ad in the recent past

These are all **binary features** (values can be either 0 or 1)

Example

Suppose these are the weights

- Recently searched “refrigerator repair”: 2.0
- Recently searched “refrigerator reviews”: 8.0
- Recently bought a refrigerator: -15.0
- Has clicked on any ad in the recent past: 5.0
- b: -9.0

Prediction function

$$f(\mathbf{x}) = \begin{cases} 1 & \mathbf{w}^T\mathbf{x} + b \geq 0 \\ -1 & \mathbf{w}^T\mathbf{x} + b < 0 \end{cases}$$

Example

Suppose these are the weights

- Recently searched “refrigerator repair”: 2.0
- Recently searched “refrigerator reviews”: 8.0
- Recently bought a refrigerator: -15.0
- Has clicked on any ad in the recent past: 5.0
- b: -9.0

$$\mathbf{w}^T\mathbf{x} + b$$
$$= 2 \cdot 0 + 8 \cdot 1 - 15 \cdot 0 + 5 \cdot 0 + -9$$
$$= 8 - 9 = -1$$

Prediction: No

Example

Suppose these are the weights

- Recently searched “refrigerator repair”: 2.0
- Recently searched “refrigerator reviews”: 8.0
- Recently bought a refrigerator: -15.0
- Has clicked on any ad in the recent past: 5.0
- b: -9.0

$$\mathbf{w}^T\mathbf{x} + b$$
$$= 2 \cdot 1 + 8 \cdot 1 - 15 \cdot 0 + 5 \cdot 0 + -9$$
$$= 2 + 8 - 9 = 1$$

Prediction: Yes

Example

Suppose these are the weights

- Recently searched “refrigerator repair”: 2.0
- Recently searched “refrigerator reviews”: 8.0
- Recently bought a refrigerator: -15.0
- Has clicked on any ad in the recent past: 5.0
- b: -9.0

$$\mathbf{w}^T\mathbf{x} + b$$
$$= 2 \cdot 0 + 8 \cdot 1 - 15 \cdot 0 + 5 \cdot 1 + -9$$
$$= 8 + 5 - 9 = 1$$

Prediction: Yes

Example

Suppose these are the weights

- Recently searched “refrigerator repair”: 2.0
- Recently searched “refrigerator reviews”: 8.0
- Recently bought a refrigerator: -15.0
- Has clicked on any ad in the recent past: 5.0
- b: -9.0

$$\mathbf{w}^T\mathbf{x} + b$$
$$= 2 \cdot 0 + 8 \cdot 1 - 15 \cdot 1 + 5 \cdot 1 + -9$$
$$= 8 - 15 + 5 - 9 = -11$$

Prediction: No

If someone bought a refrigerator recently, they probably aren’t interested in shopping for another one anything soon

Linear Classifiers

EXPRESSIVENESS

Why use linear classifiers?

Simple and expressive

For any new hypothesis space, think about which kinds of functions does this hypothesis space include

Recall: Decision trees: can express any Boolean function

Which Boolean functions can linear classifiers represent?

Linear classifiers are an expressive hypothesis class

If you have only Boolean features, what Boolean functions are can be captured by linear classifiers?

Function is linearly separable if linear classier that captures it

Many Boolean functions are linearly separable

- Not all though
- In comparison, decision trees can represent any Boolean function

Conjunctions and disjunctions

$y = x_1 \wedge x_2 \wedge x_3$ is equivalent to "$y = 1$ whenever $x_1 + x_2 + x_3 \geq 3$"

x_1	x_2	x_3	y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Conjunctions are linearly separable

Conjunctions and disjunctions

$y = x_1 \wedge x_2 \wedge x_3$ is equivalent to "$y = 1$ whenever $x_1 + x_2 + x_3 \geq 3$"

$sgn[x_1 + x_2 + x_3 - 3]$

What is vector $\mathbf{w}$?

x_1	x_2	x_3	y	$x_1 + x_2 + x_3 = 3$	sign
0	0	0	0	-3	0
0	0	1	0	-2	0
0	1	0	0	-2	0
0	1	1	0	-1	0
1	0	0	0	-2	0
1	0	1	0	-1	0
1	1	0	0	-1	0
1	1	1	1	0	1

Enumerate all possible cases and compute the function

The columns y and sign are the same, so the two functions are the same

Conjunctions and disjunctions

$y = x_1 \wedge x_2 \wedge x_3$ is equivalent to "$y = 1$ whenever $x_1 + x_2 + x_3 \geq 3$"

x_1	x_2	x_3	y	$x_1 + x_2 + x_3 = 3$	sign
0	0	0	0	-3	0
0	0	1	0	-2	0
0	1	0	0	-2	0
0	1	1	0	-1	0
1	0	0	0	-2	0
1	0	1	0	-1	0
1	1	0	0	-1	0
1	1	1	1	0	1

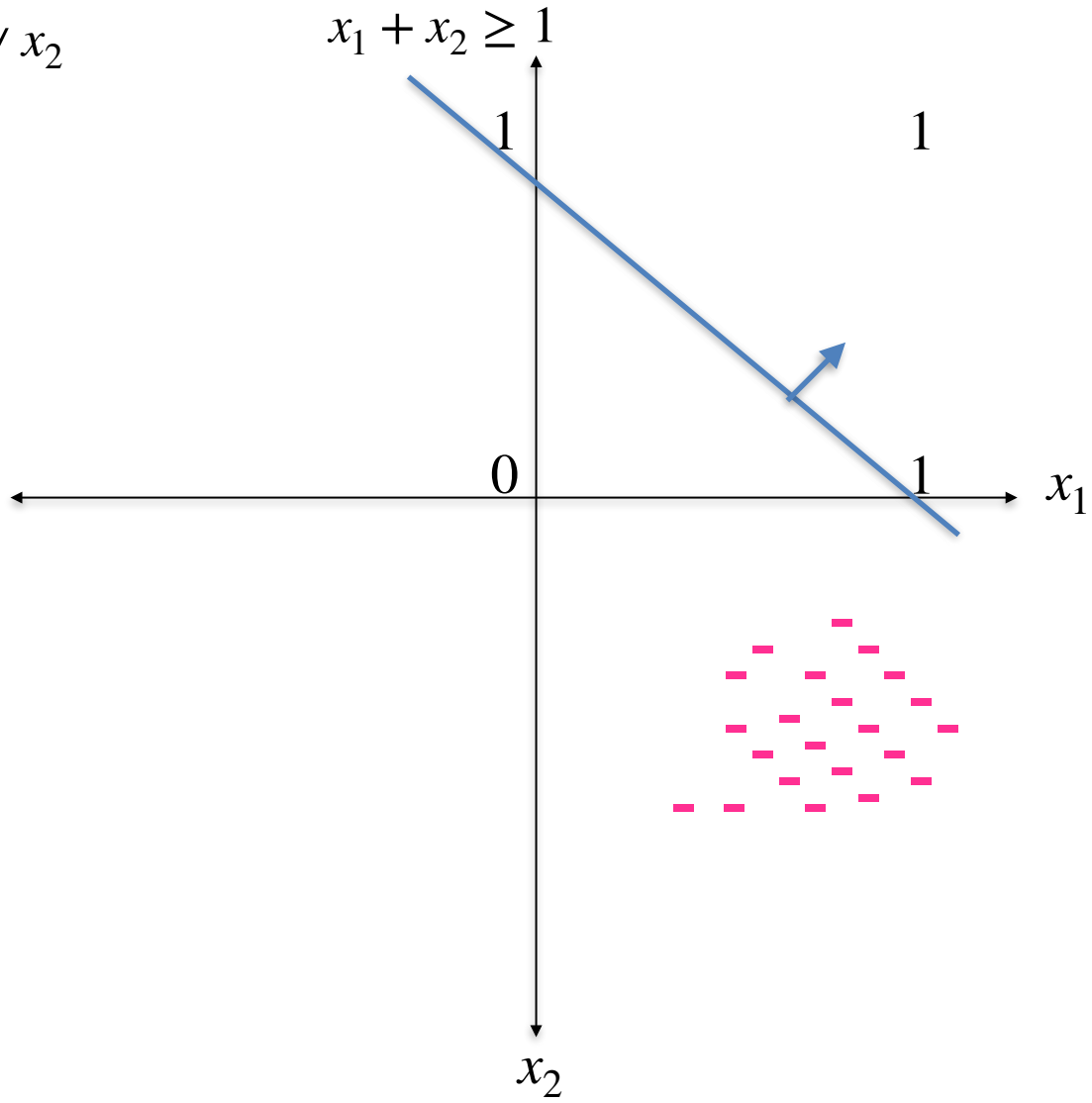
Negations are okay too. In general, use $1 - x$ in the linear threshold unit if x is negated

$y = x_1 \wedge x_2 \wedge \neg x_3$ corresponds to

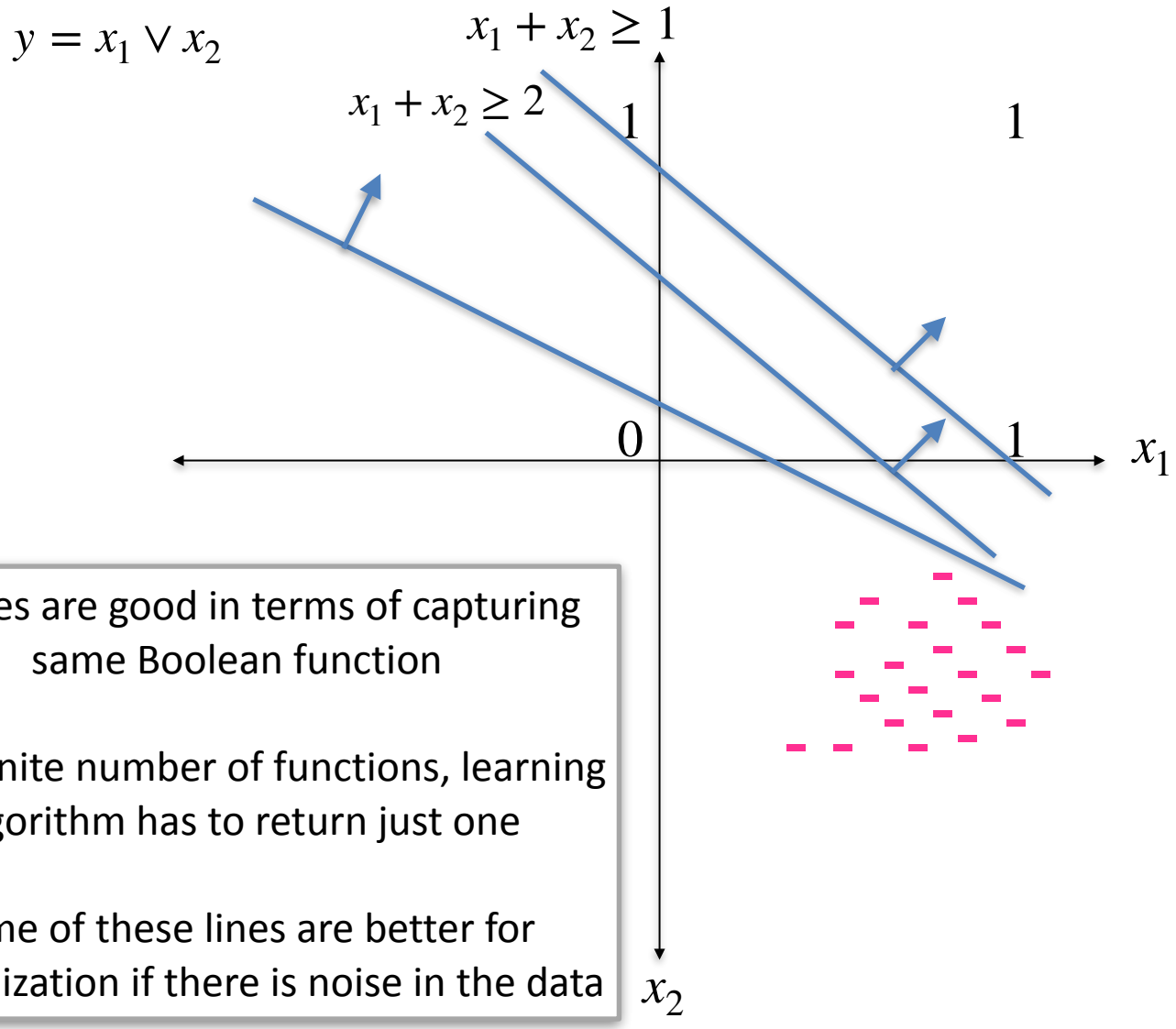
$x_1 + x_2 + (1 - x_3) \geq 3$

How do you know which line is best?

$y = x_1 \vee x_2$



How do you know which line is best?



All lines are good in terms of capturing same Boolean function

But infinite number of functions, learning algorithm has to return just one

Some of these lines are better for generalization if there is noise in the data

m-of-n functions

m-of-n rules

- There is a fixed set of n variables
- $y =$ true if and only if at least m of them are true
- All other variables are ignored

Suppose there are five Boolean variables: x_1, x_2, x_3, x_4, x_5

What is a threshold unit that is equivalent to the classification rule "at least 2 of $\{x_1, x_2, x_3\}$" should be true?

m-of-n functions

m-of-n rules

- There is a fixed set of n variables
- $y =$ true if and only if at least m of them are true
- All other variables are ignored

Suppose there are five Boolean variables: x_1, x_2, x_3, x_4, x_5

What is a threshold unit that is equivalent to the classification rule "at least 2 of $\{x_1, x_2, x_3\}$" should be true?

$$x_1 + x_2 + x_3 \geq 2$$

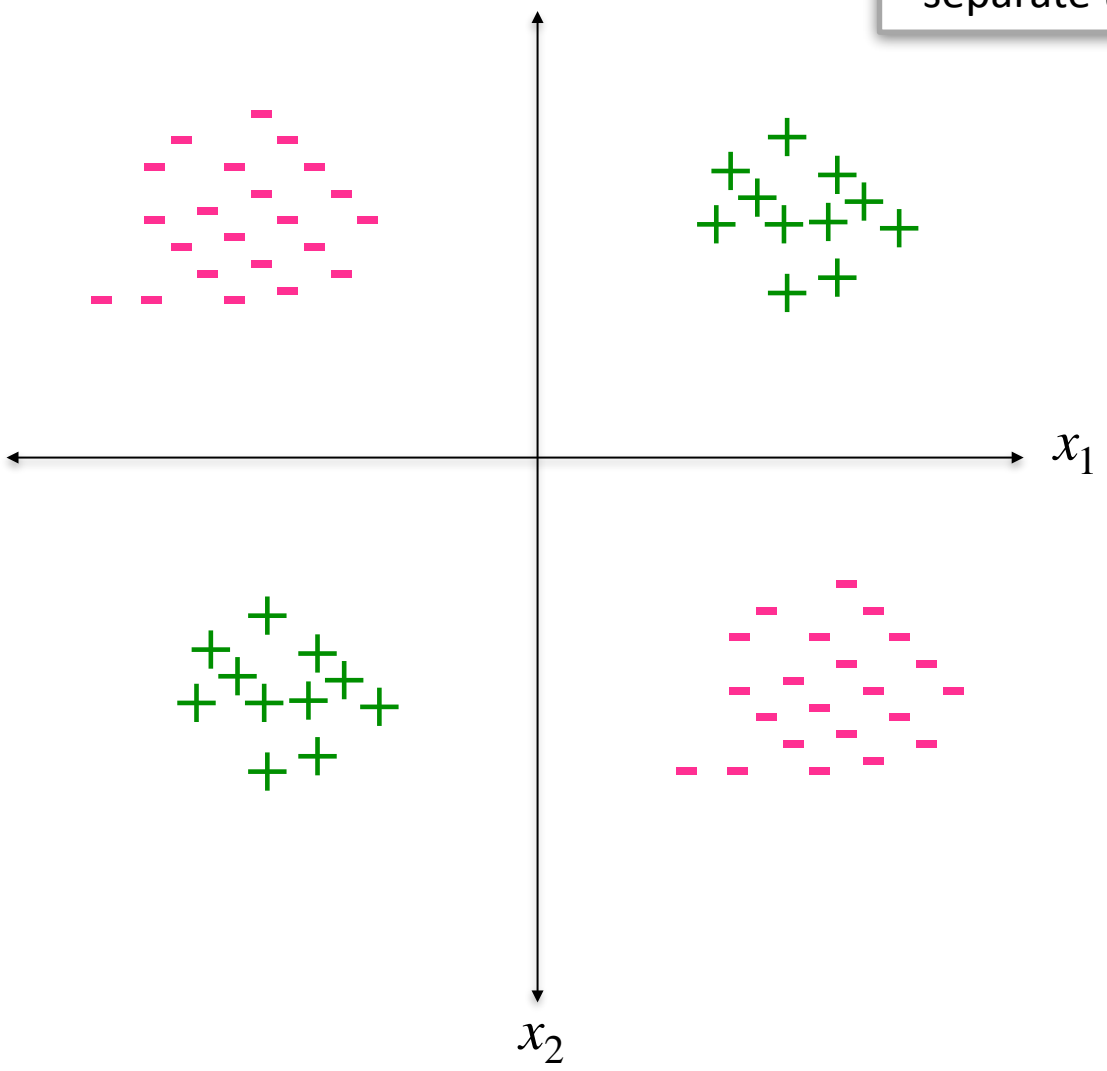
$$b = -2 \qquad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$$

Not all functions are linearly separable

(The parity or XOR function)

Can't draw a line to separate the two classes

x_1	x_2	f
-1	-1	+1
+1	-1	-1
-1	+1	-1
+1	+1	+1



Not all functions are linearly separable

XOR is not linear

- $y = x$ XOR y
- $y = (x \wedge \neg y) \vee (\neg x \wedge y)$
- Parity cannot be represented as a linear classifiers
  - $f(\mathbf{x}) = 1$ if the number of 1s is odd

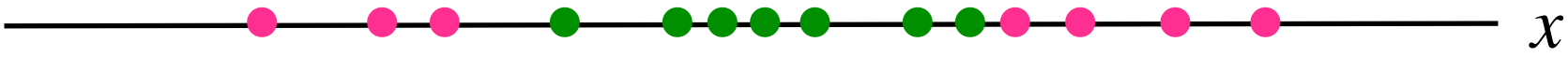
Many non-trivial Boolean functions

- Example: $y = (x_1 \wedge x_2) \vee (x_3 \wedge \neg x_4)$
- The function is not linear in the four variables

Even these functions can be made linear

These points are not separable in 1-dimension by a line

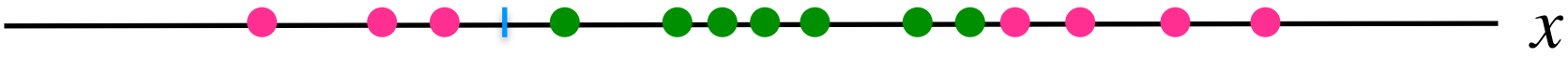
What is a one-dimensional line, by the way?



Even these functions can be made linear

These points are not separable in 1-dimension by a line

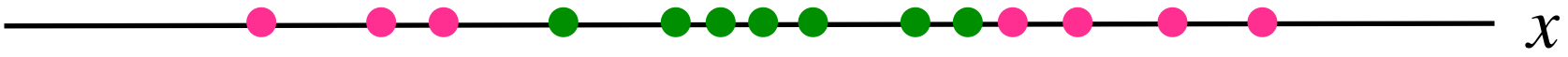
What is a one-dimensional line, by the way? Point



Even these functions can be made linear

These points are not separable in 1-dimension by a line

What is a one-dimensional line, by the way?

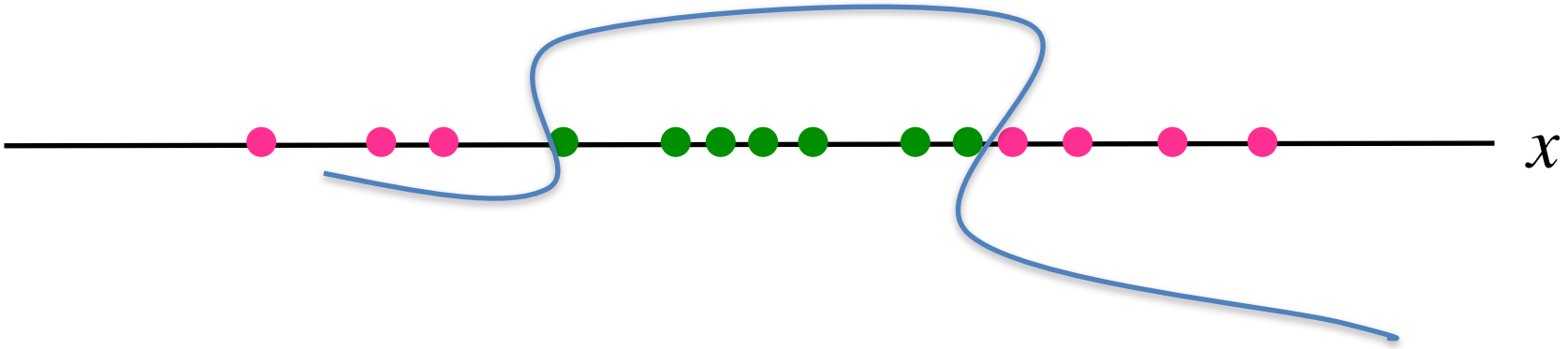


What can we do?

Even these functions can be made linear

These points are not separable in 1-dimension by a line

What is a one-dimensional line, by the way?

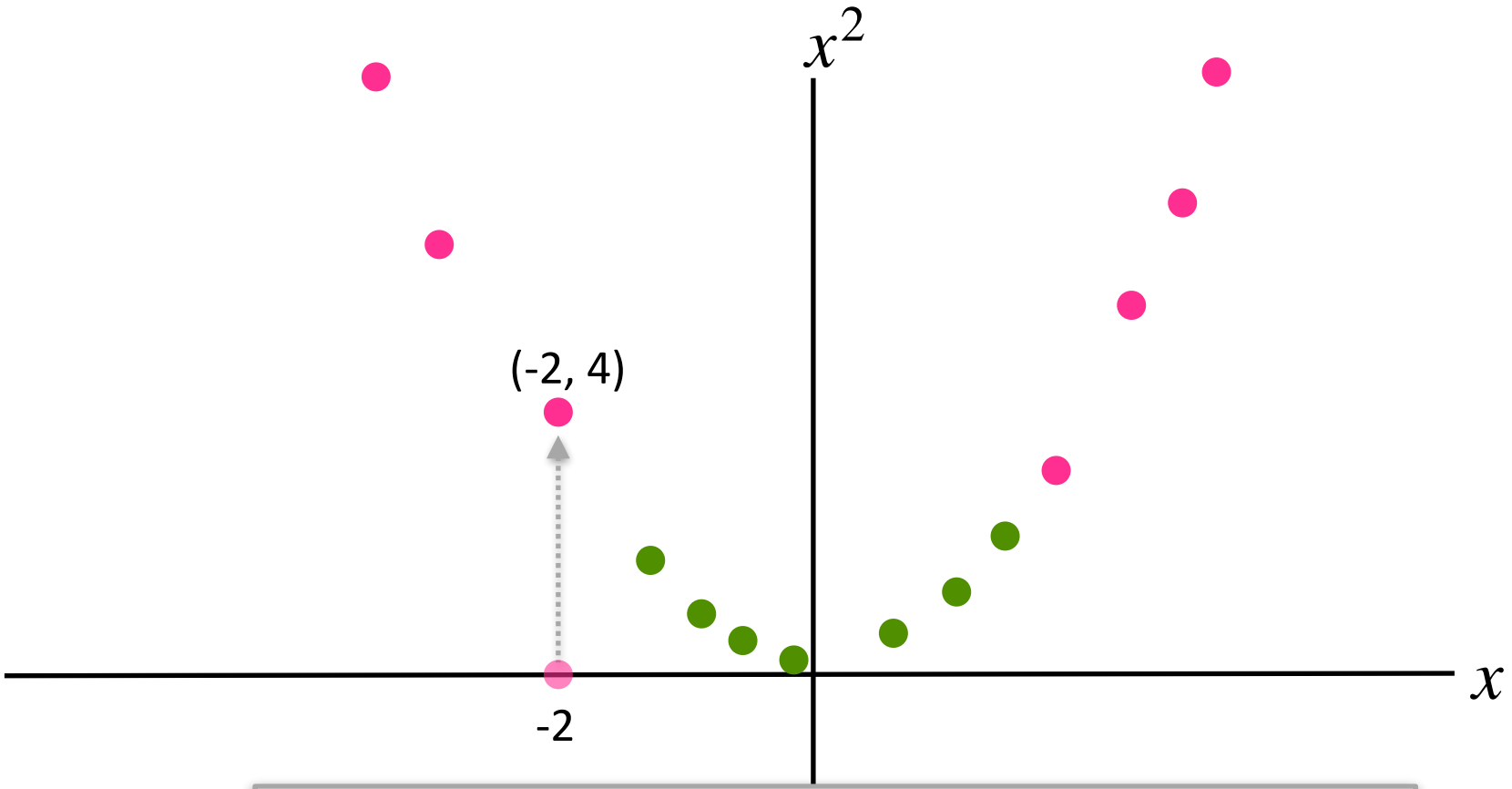


What can we do?

The blown up feature space

The trick: use feature conjunctions

Transform points: represent each point x in 2 dimensions by (x, x^2)

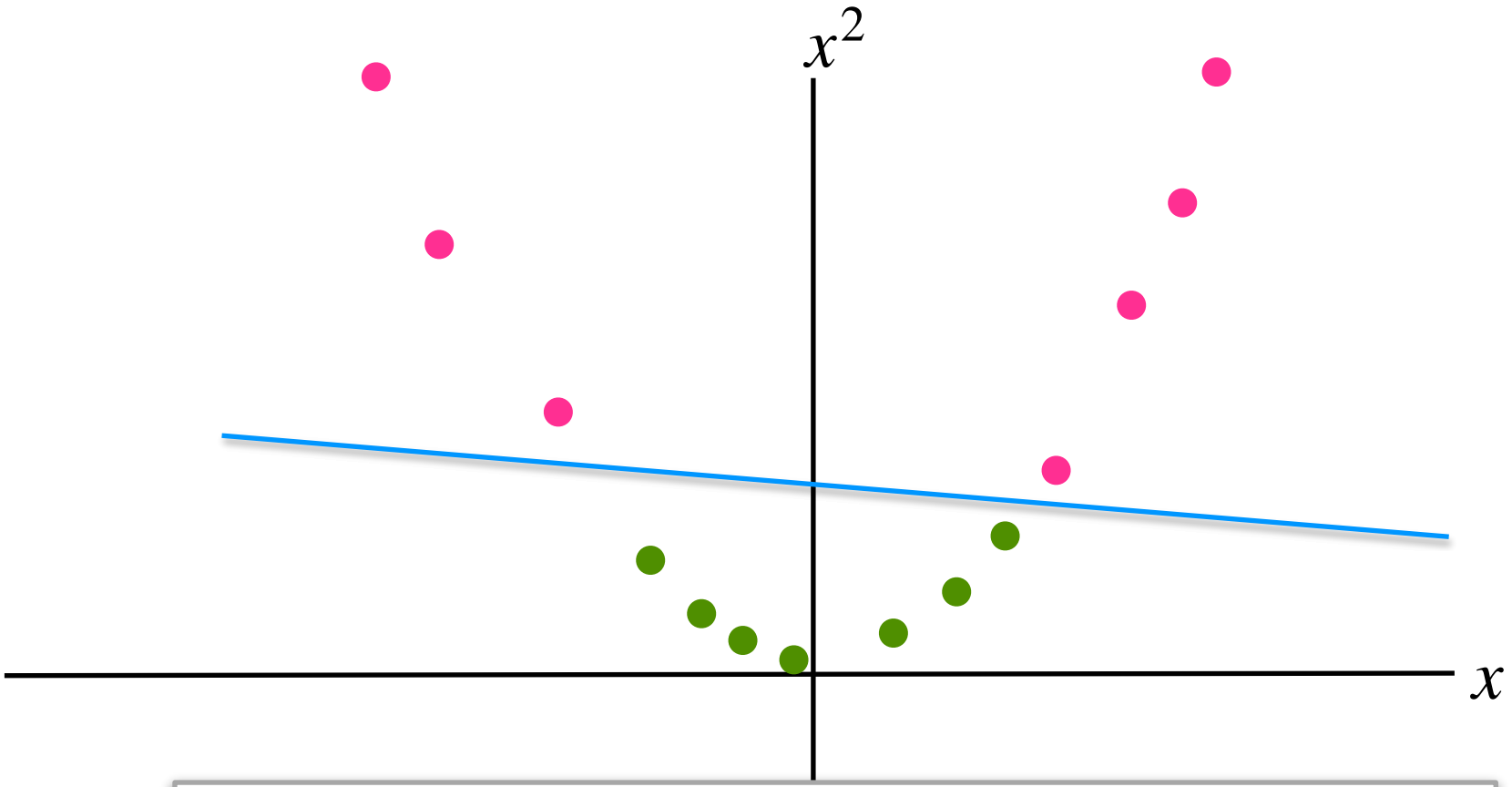


Now the data is linearly separable in this space!

The blown up feature space

The trick: use feature conjunctions

Transform points: represent each point x in 2 dimensions by (x, x^2)

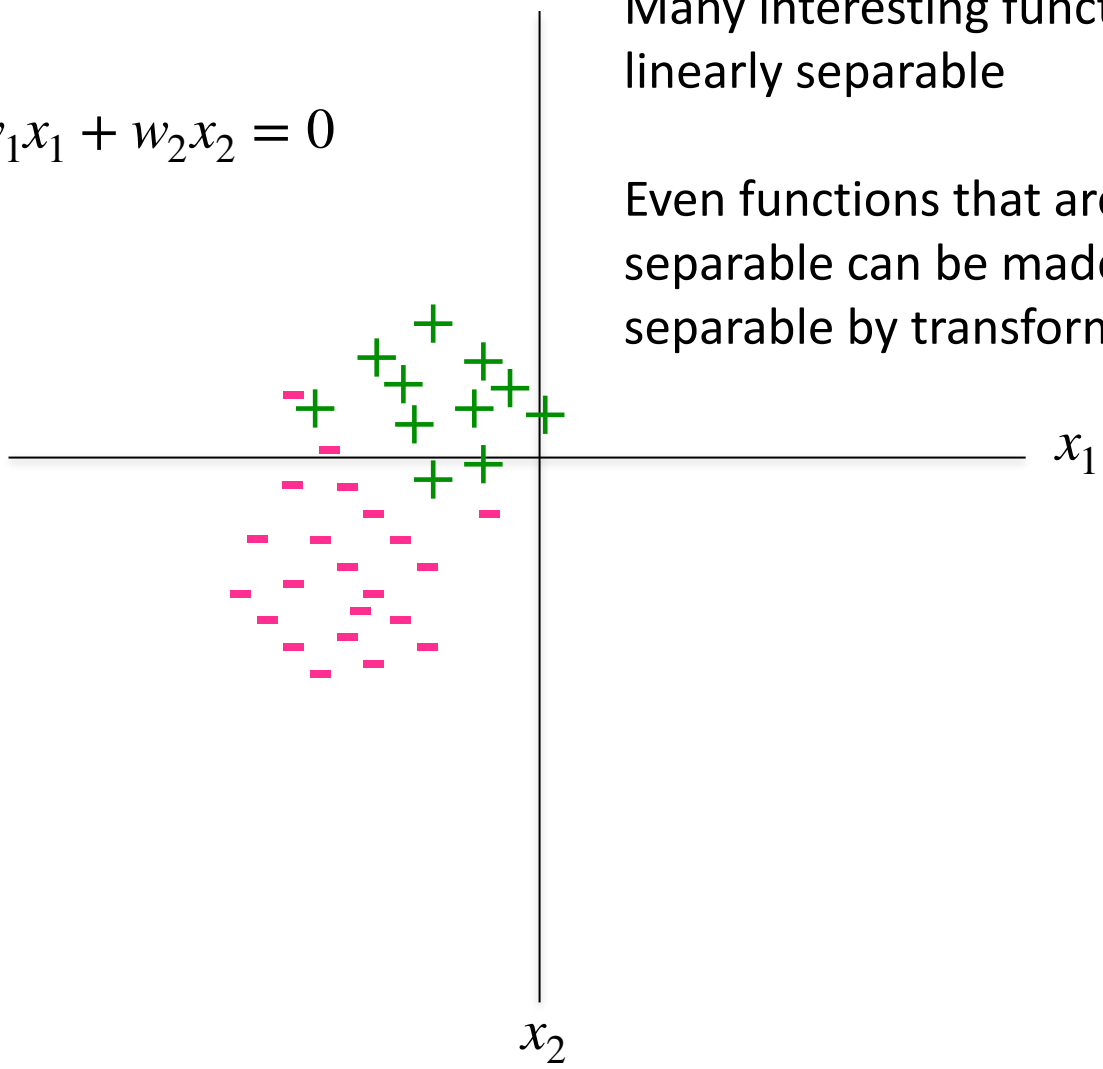


Key issue: representation. What features to use?

Almost linearly separable data

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

$$b + w_1 x_1 + w_2 x_2 = 0$$



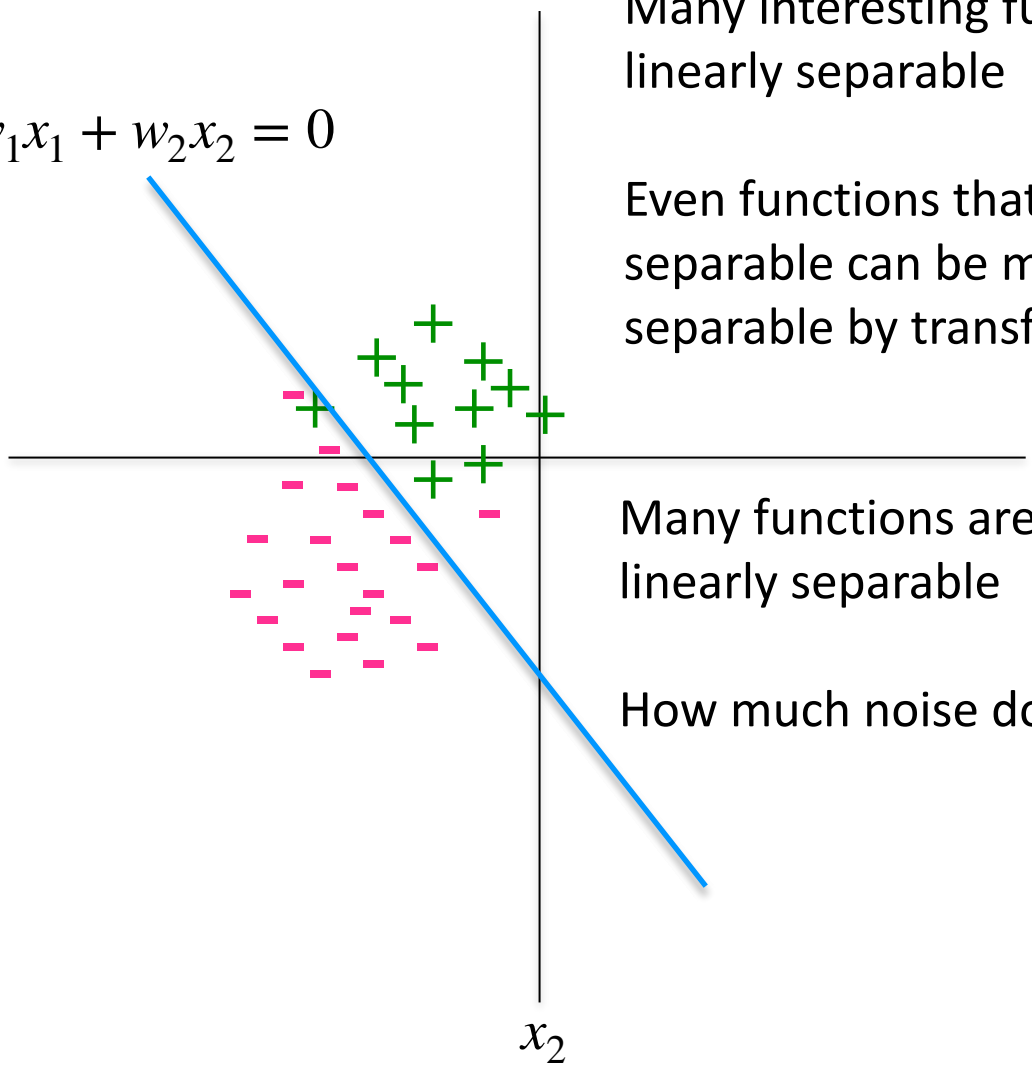
Many interesting functions are linearly separable

Even functions that are not linearly separable can be made linearly separable by transforming the data

Almost linearly separable data

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

$$b + w_1 x_1 + w_2 x_2 = 0$$



Many interesting functions are linearly separable

Even functions that are not linearly separable can be made linearly separable by transforming the data

Many functions are also almost linearly separable

How much noise do we allow for?

Linear classifiers: an expressive hypothesis class

Many functions are linear

Often a good guess for a hypothesis space

Some functions are not linear

- The XOR function
- Non-trivial Boolean functions

But there are ways of making them linear in a higher dimensional feature space