6.036/6.862: Introduction to Machine Learning

Lecture: starts Tuesdays 9:35am (Boston time zone)

Course website: introml.odl.mit.edu

Who's talking? Prof. Tamara Broderick

Questions? discourse.odl.mit.edu ("Lecture 2" category)

Materials: Will all be available at course website

Last Time

I. Machine learning setup
II. Linear classifiers
III. Learning algorithms

Today's Plan

I. Perceptron algorithm
II. Harder and easier linear classification
III. Perceptron theorem

Recall: Classifiers

Recall: Classifiers

- A linear classifier:

$h(x; \theta, \theta_0)$

Recall: Classifiers

- A linear classifier:

$$h(x; \theta, \theta_0) = \text{sign}(\theta^\top x + \theta_0)$$

Recall: Classifiers

- A linear classifier:

$$h(x;\theta,\theta_0) = \mathrm{sign}(\theta^\top x + \theta_0)$$

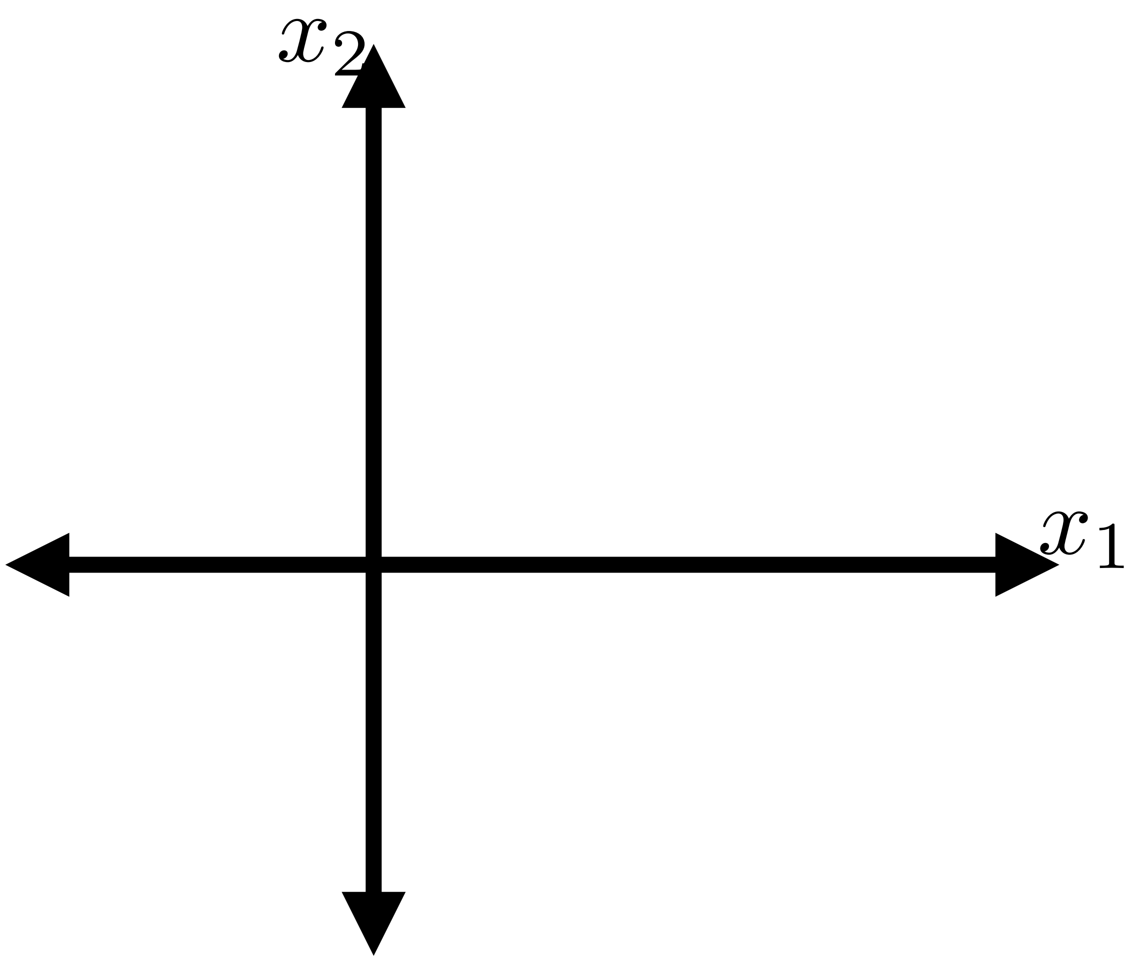
$$= \begin{cases} +1 \text{ if } \theta^\top x + \theta_0 > 0 \\ -1 \text{ if } \theta^\top x + \theta_0 \leq 0 \end{cases}$$

Recall: Classifiers

- A linear classifier:

$$h(x;\theta,\theta_0) = \text{sign}(\theta^\top x + \theta_0)$$

$$= \begin{cases} +1 \text{ if } \theta^\top x + \theta_0 > 0 \\ -1 \text{ if } \theta^\top x + \theta_0 \leq 0 \end{cases}$$

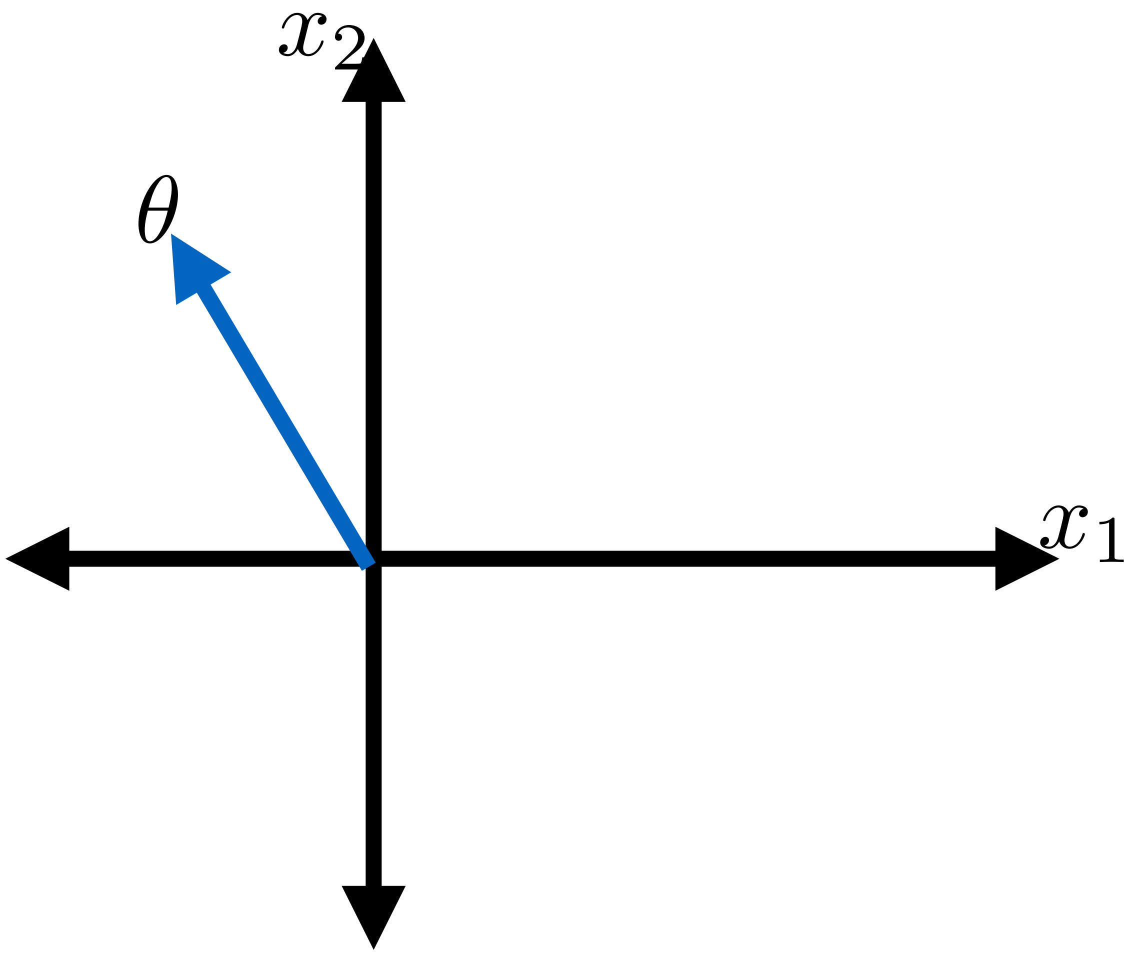


Recall: Classifiers

- A linear classifier:

$$h(x;\theta,\theta_0) = \text{sign}(\theta^\top x + \theta_0)$$

$$= \begin{cases} +1 \text{ if } \theta^\top x + \theta_0 > 0 \\ -1 \text{ if } \theta^\top x + \theta_0 \leq 0 \end{cases}$$

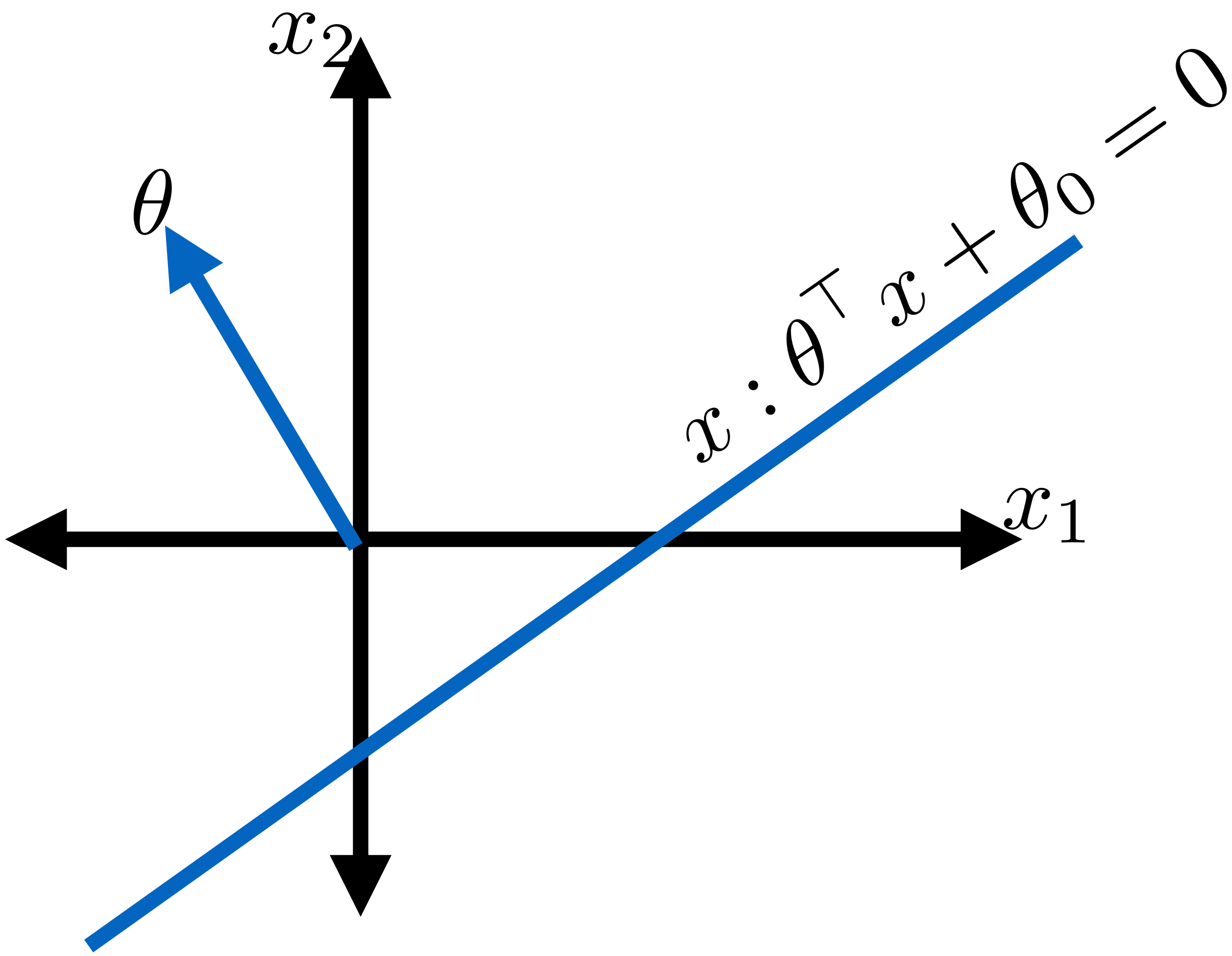


Recall: Classifiers

- A linear classifier:

$$h(x;\theta,\theta_0) = \text{sign}(\theta^\top x + \theta_0)$$

$$= \begin{cases} +1 \text{ if } \theta^\top x + \theta_0 > 0 \\ -1 \text{ if } \theta^\top x + \theta_0 \leq 0 \end{cases}$$

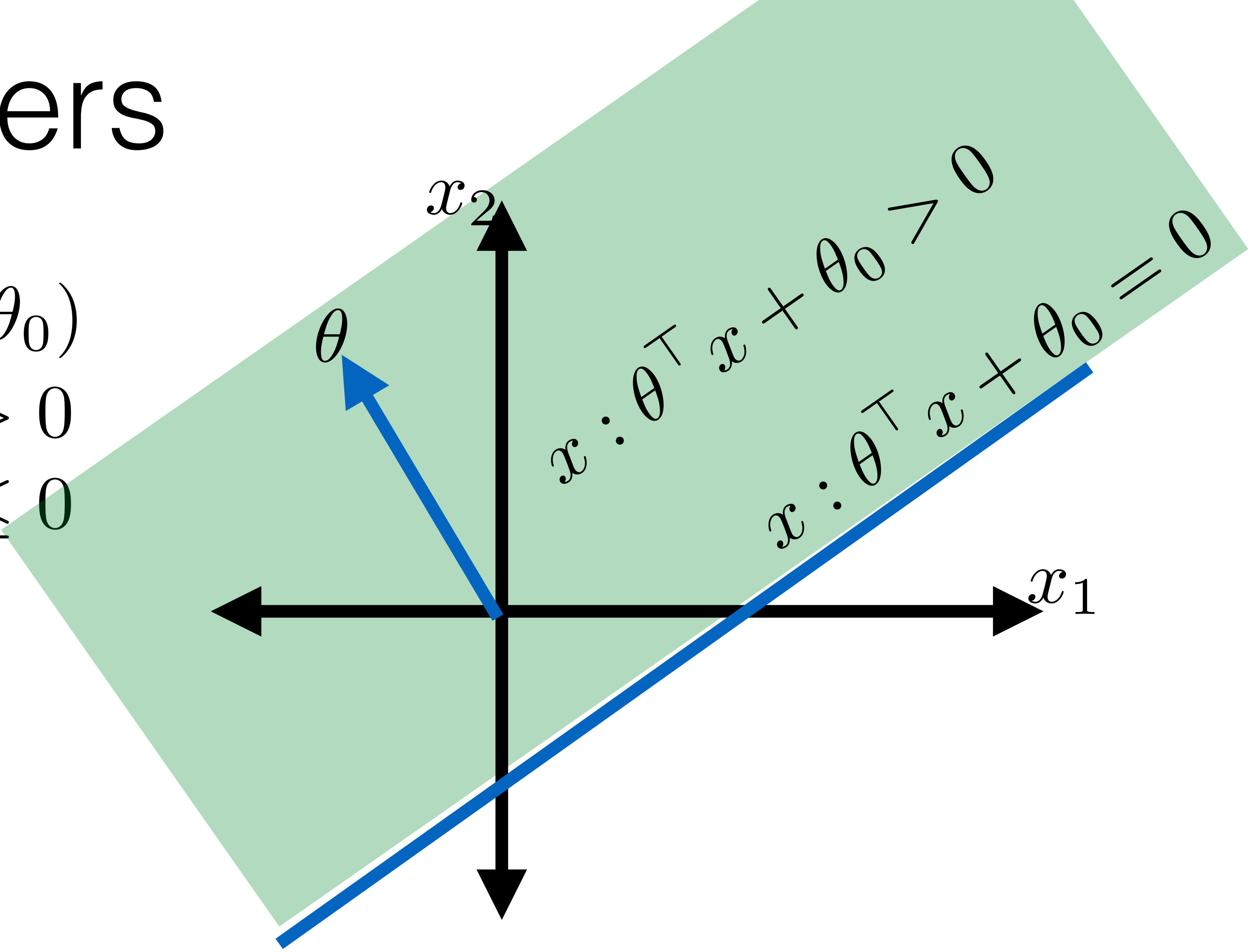


Recall: Classifiers

- A linear classifier:

$$h(x;\theta,\theta_0) = \text{sign}(\theta^\top x + \theta_0)$$

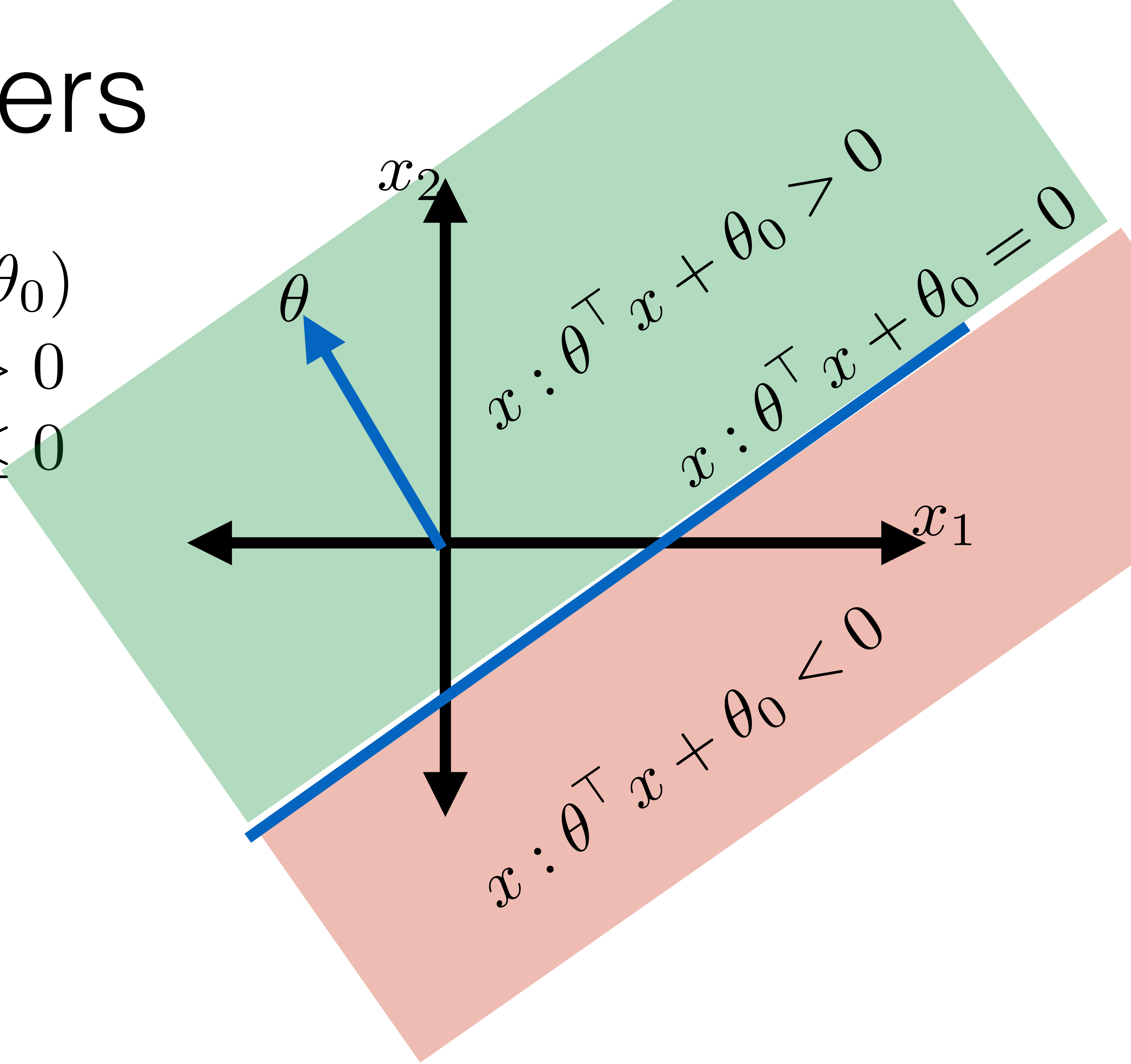
$$= \begin{cases} +1 \text{ if } \theta^\top x + \theta_0 > 0 \\ -1 \text{ if } \theta^\top x + \theta_0 \leq 0 \end{cases}$$



Recall: Classifiers

- A linear classifier:

$$h(x;\theta,\theta_0) = \text{sign}(\theta^\top x + \theta_0)$$
$$= \begin{cases} +1 \text{ if } \theta^\top x + \theta_0 > 0 \\ -1 \text{ if } \theta^\top x + \theta_0 \leq 0 \end{cases}$$

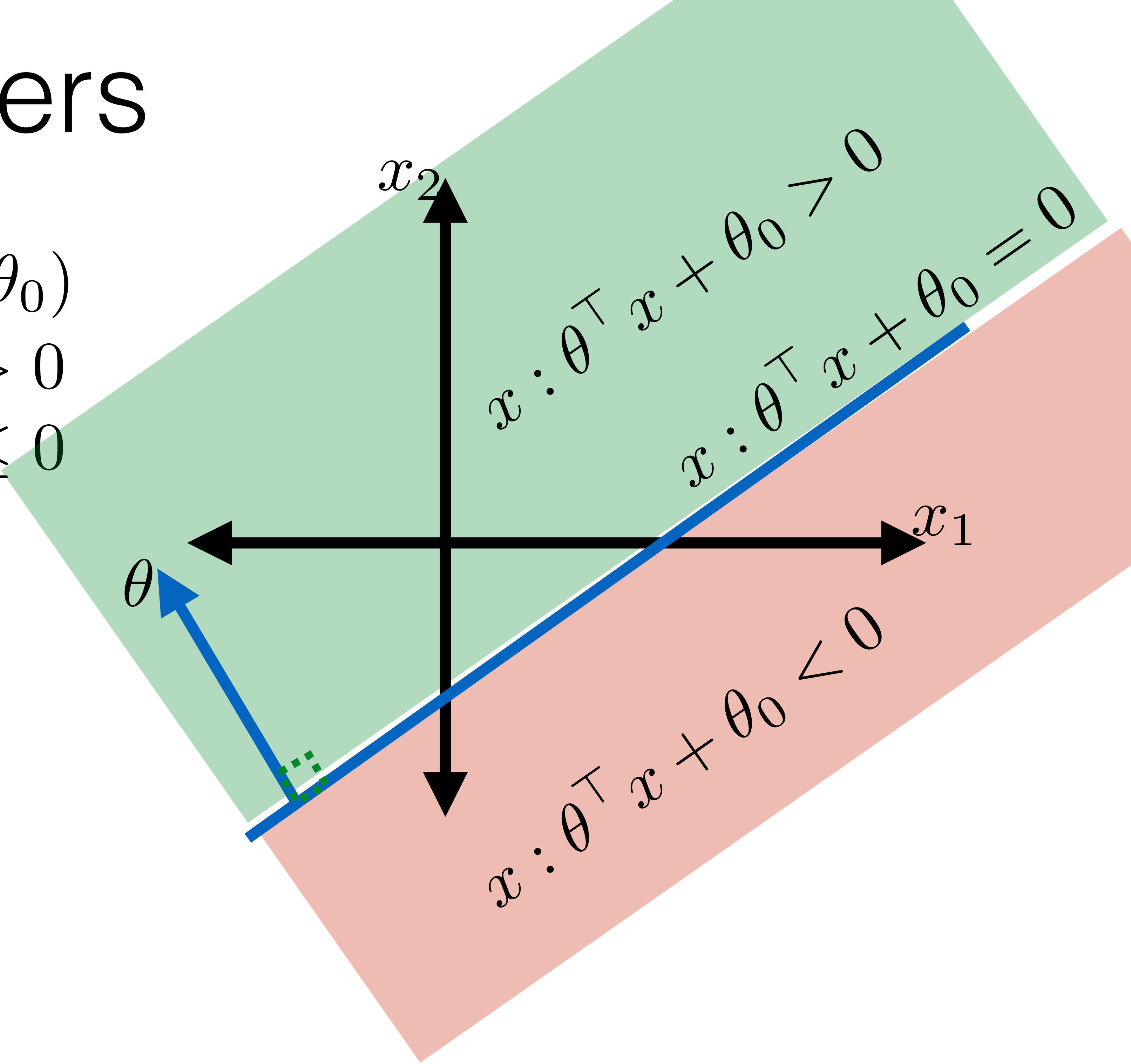


Recall: Classifiers

- A linear classifier:

$$h(x;\theta,\theta_0) = \text{sign}(\theta^\top x + \theta_0)$$

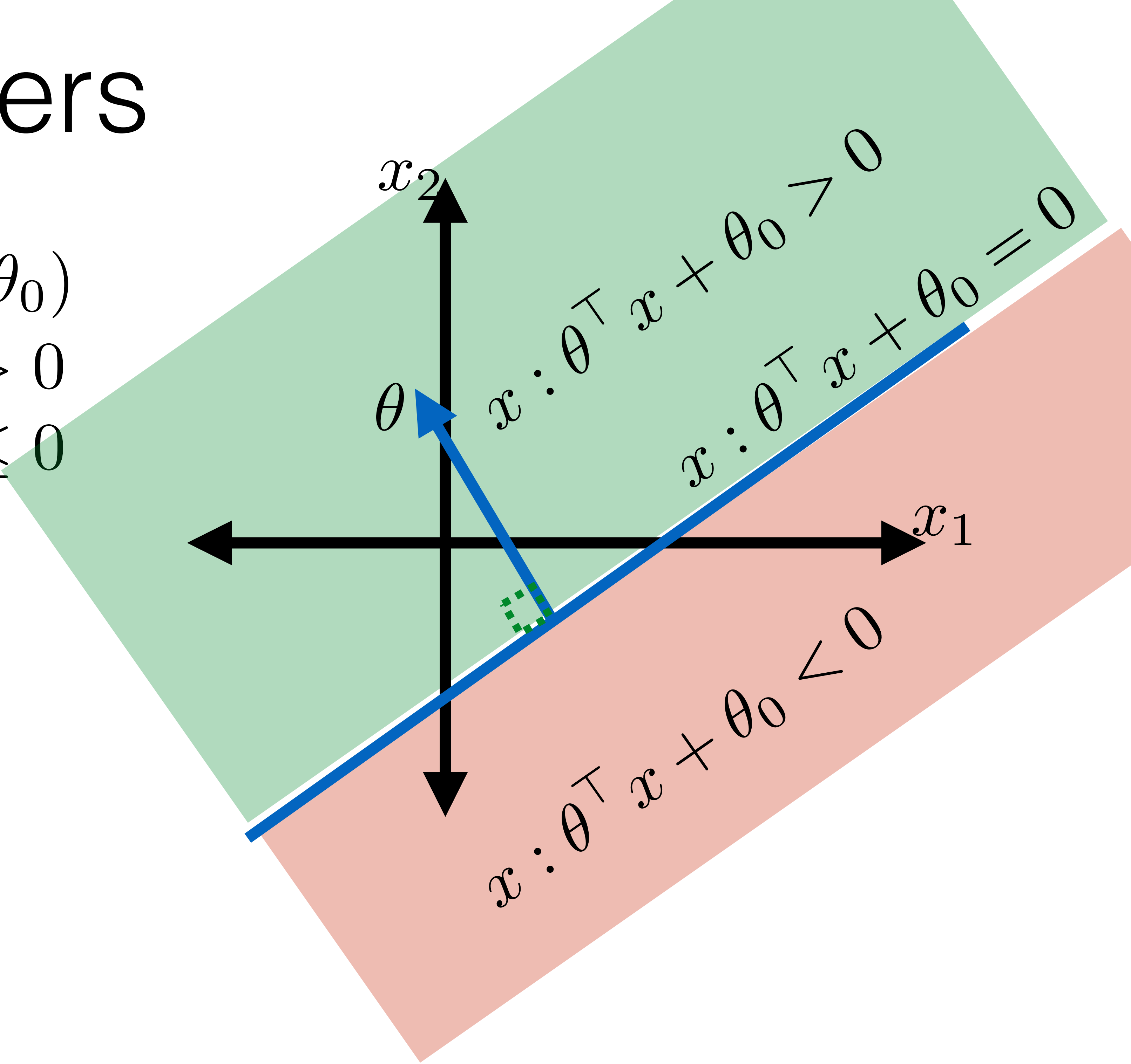
$$= \begin{cases} +1 \text{ if } \theta^\top x + \theta_0 > 0 \\ -1 \text{ if } \theta^\top x + \theta_0 \leq 0 \end{cases}$$



Recall: Classifiers

- A linear classifier:

$$h(x;\theta,\theta_0) = \text{sign}(\theta^\top x + \theta_0)$$
$$= \begin{cases} +1 \text{ if } \theta^\top x + \theta_0 > 0 \\ -1 \text{ if } \theta^\top x + \theta_0 \leq 0 \end{cases}$$

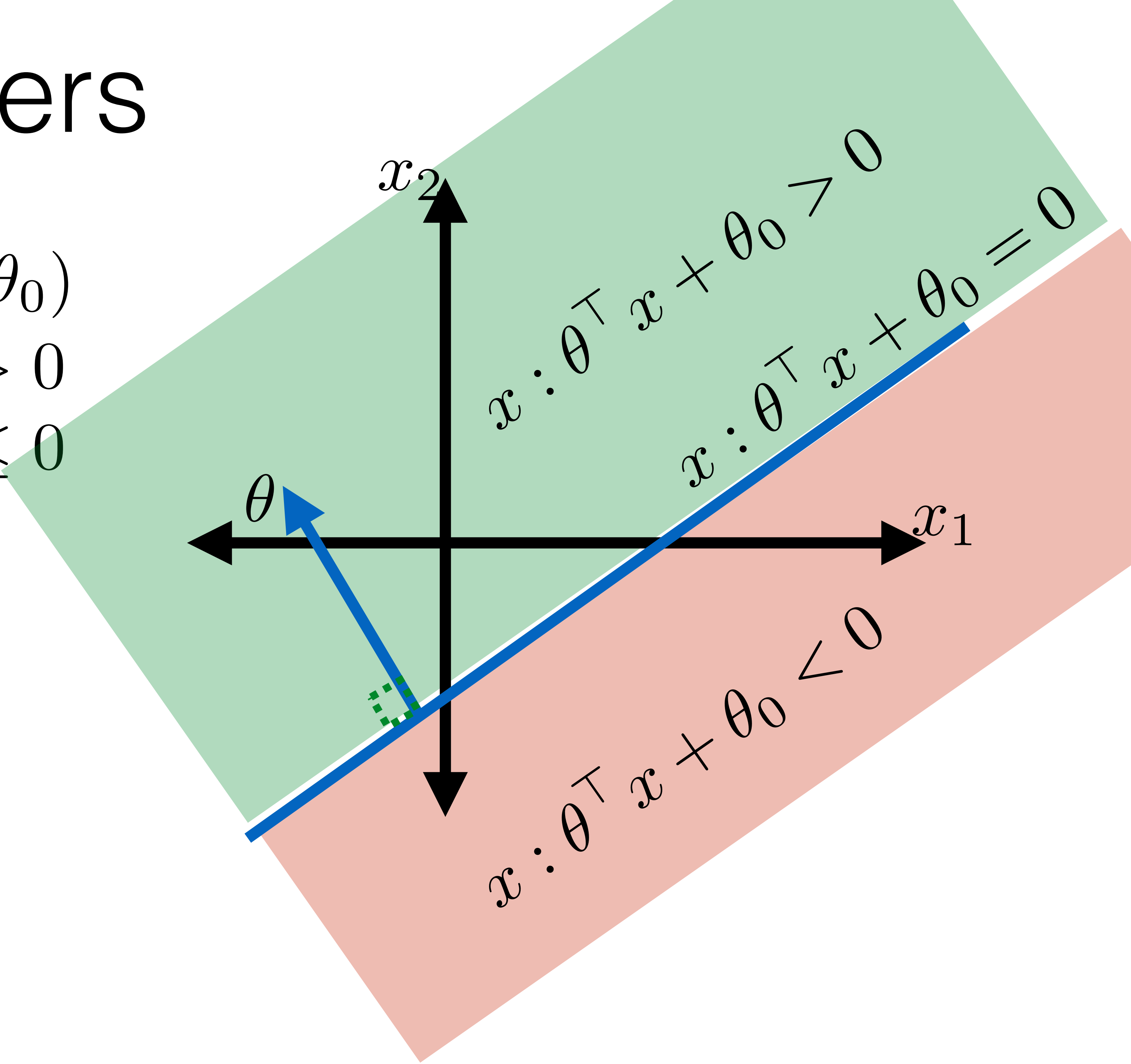


Recall: Classifiers

- A linear classifier:

$$h(x; \theta, \theta_0) = \text{sign}(\theta^\top x + \theta_0)$$

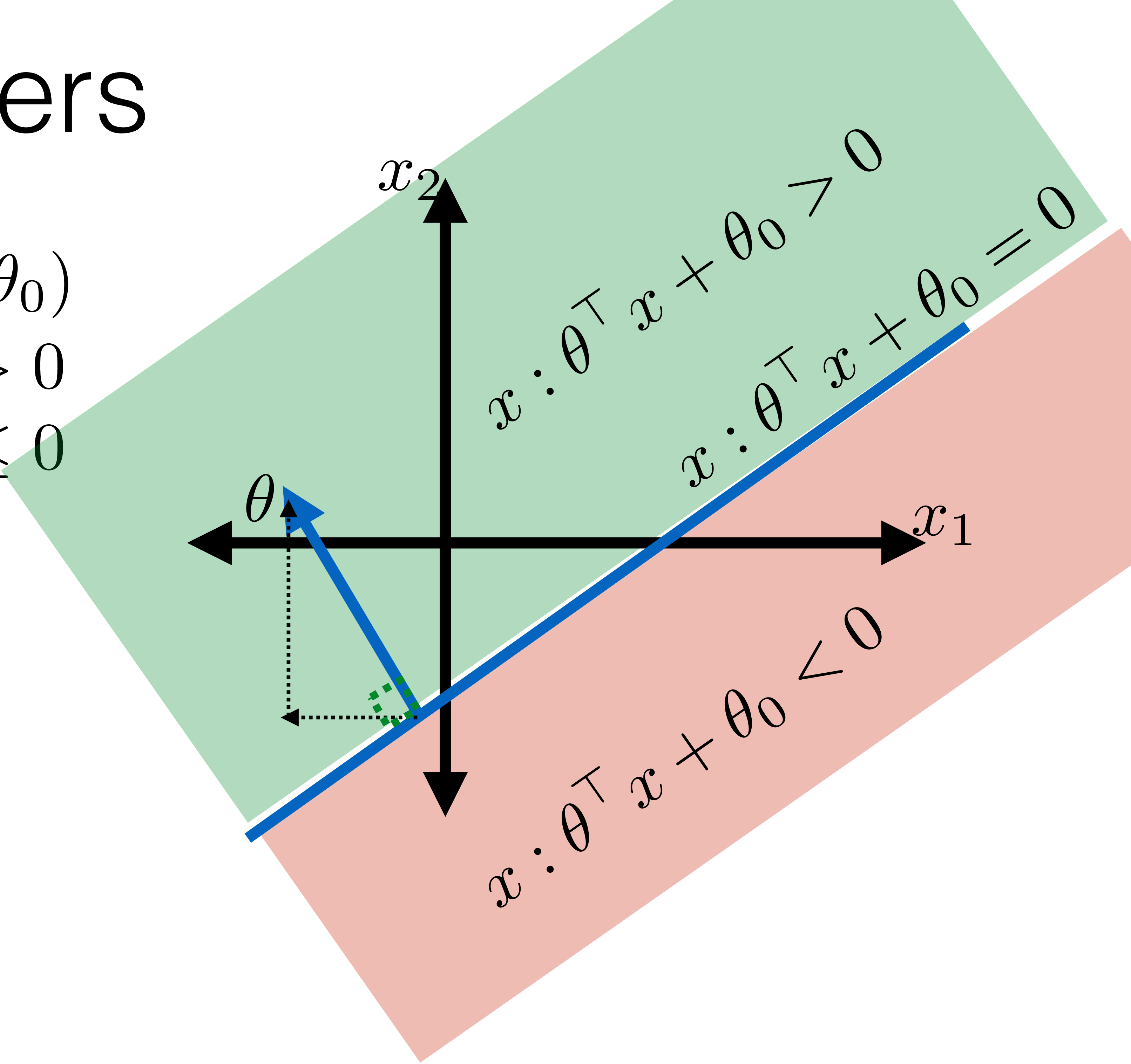
$$= \begin{cases} +1 \text{ if } \theta^\top x + \theta_0 > 0 \\ -1 \text{ if } \theta^\top x + \theta_0 \leq 0 \end{cases}$$



Recall: Classifiers

- A linear classifier:

$$h(x; \theta, \theta_0) = \text{sign}(\theta^\top x + \theta_0)$$
$$= \begin{cases} +1 \text{ if } \theta^\top x + \theta_0 > 0 \\ -1 \text{ if } \theta^\top x + \theta_0 \leq 0 \end{cases}$$

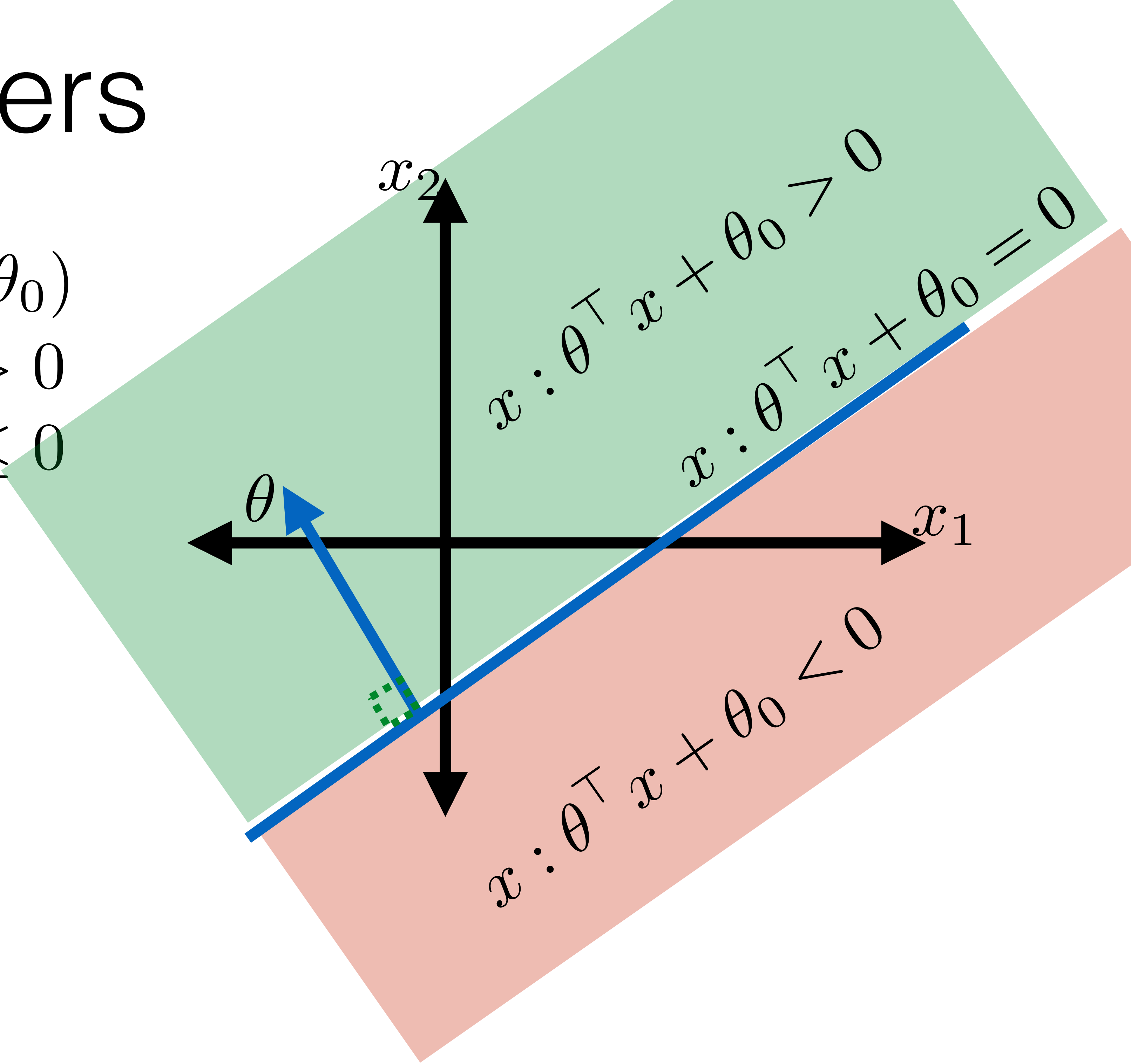


Recall: Classifiers

- A linear classifier:

$$h(x;\theta,\theta_0) = \text{sign}(\theta^\top x + \theta_0)$$

$$= \begin{cases} +1 \text{ if } \theta^\top x + \theta_0 > 0 \\ -1 \text{ if } \theta^\top x + \theta_0 \leq 0 \end{cases}$$

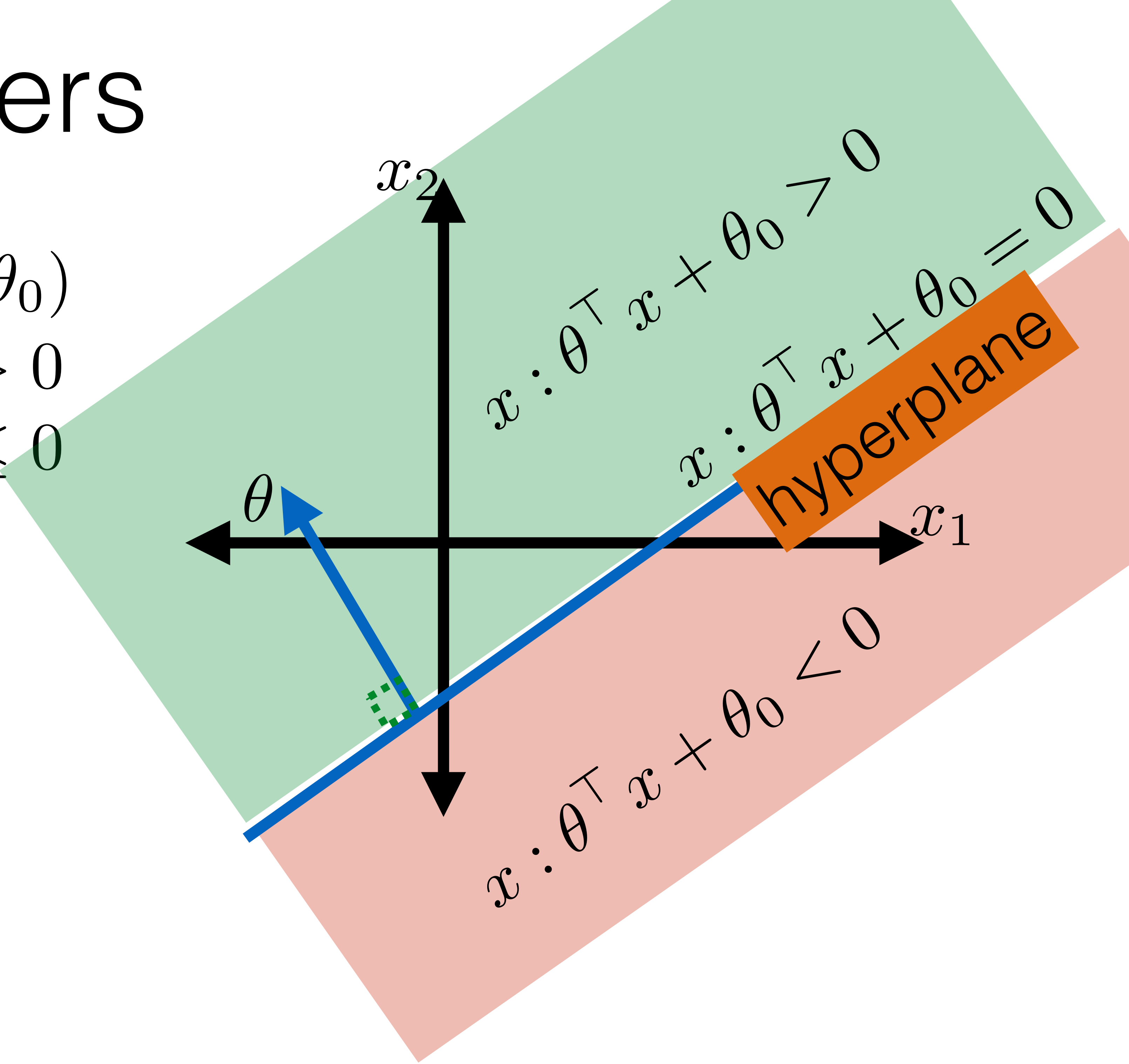


Recall: Classifiers

- A linear classifier:

$$h(x;\theta,\theta_0) = \text{sign}(\theta^\top x + \theta_0)$$

$$= \begin{cases} +1 \text{ if } \theta^\top x + \theta_0 > 0 \\ -1 \text{ if } \theta^\top x + \theta_0 \leq 0 \end{cases}$$

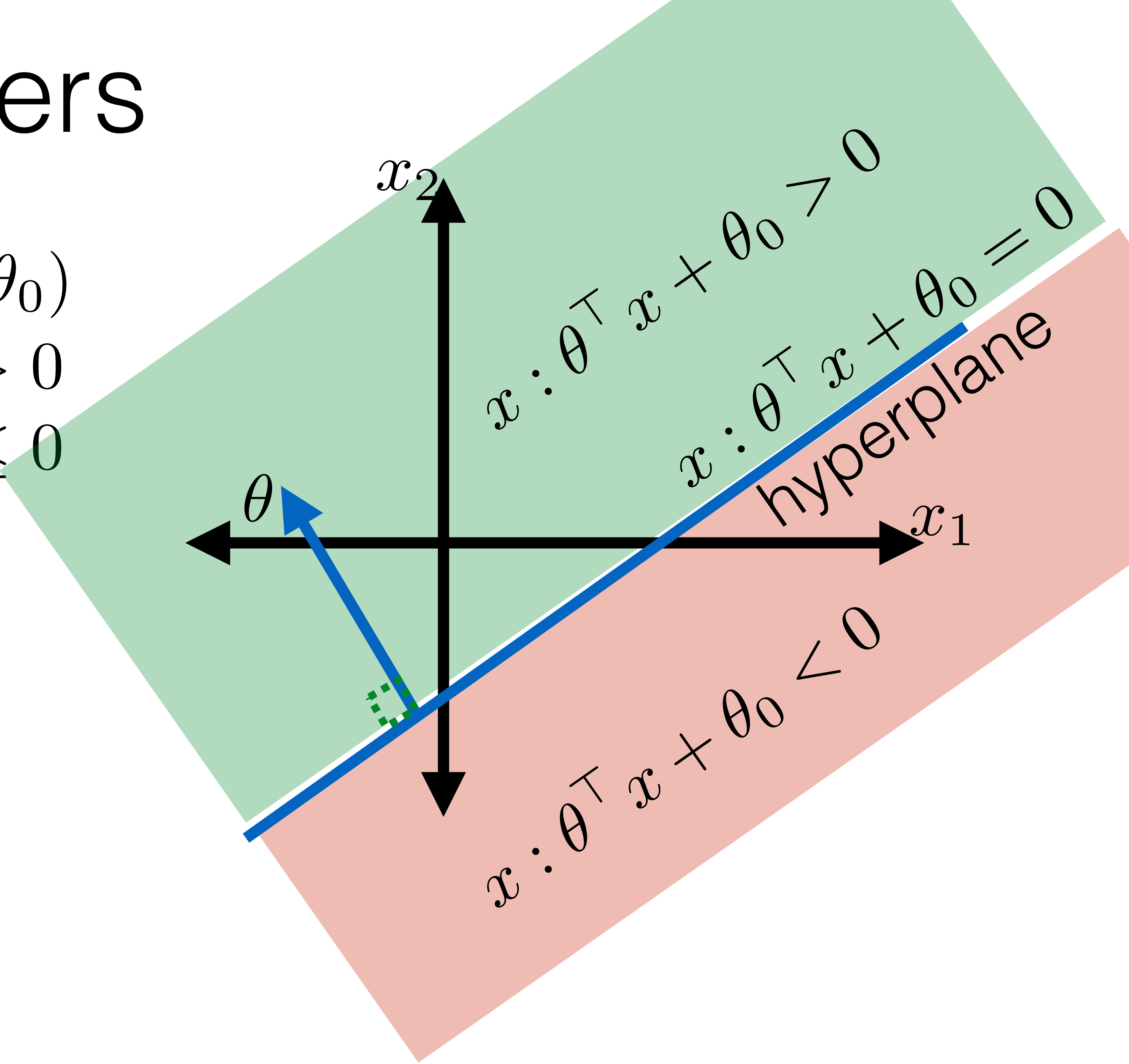


Recall: Classifiers

- A linear classifier:

$$h(x;\theta,\theta_0) = \text{sign}(\theta^\top x + \theta_0)$$

$$= \begin{cases} +1 \text{ if } \theta^\top x + \theta_0 > 0 \\ -1 \text{ if } \theta^\top x + \theta_0 \leq 0 \end{cases}$$



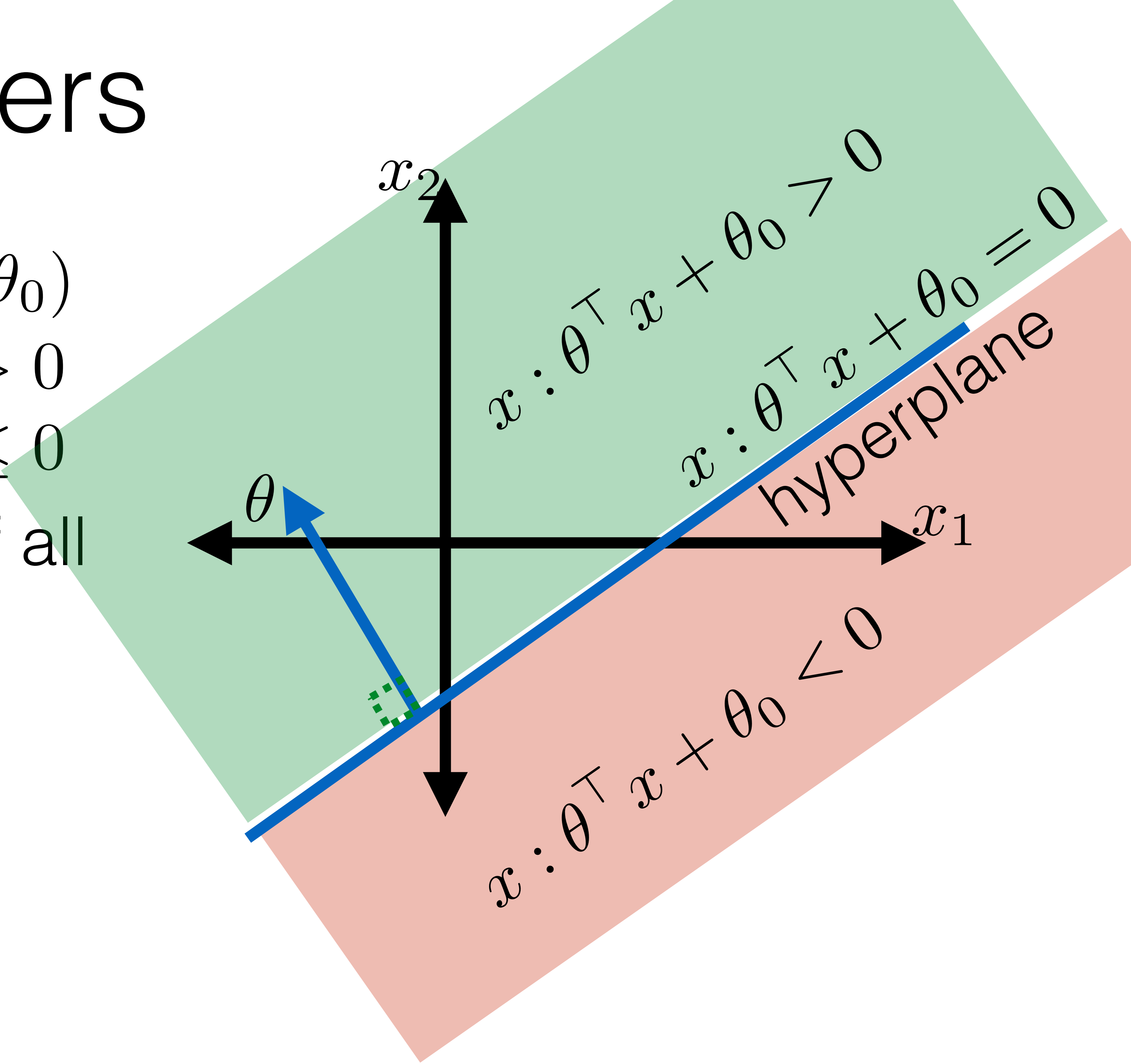
Recall: Classifiers

- A linear classifier:

$$h(x;\theta,\theta_0) = \text{sign}(\theta^\top x + \theta_0)$$

$$= \begin{cases} +1 \text{ if } \theta^\top x + \theta_0 > 0 \\ -1 \text{ if } \theta^\top x + \theta_0 \leq 0 \end{cases}$$

- Hypothesis class $\mathcal{H}$ of all linear classifiers



Recall: Classifiers

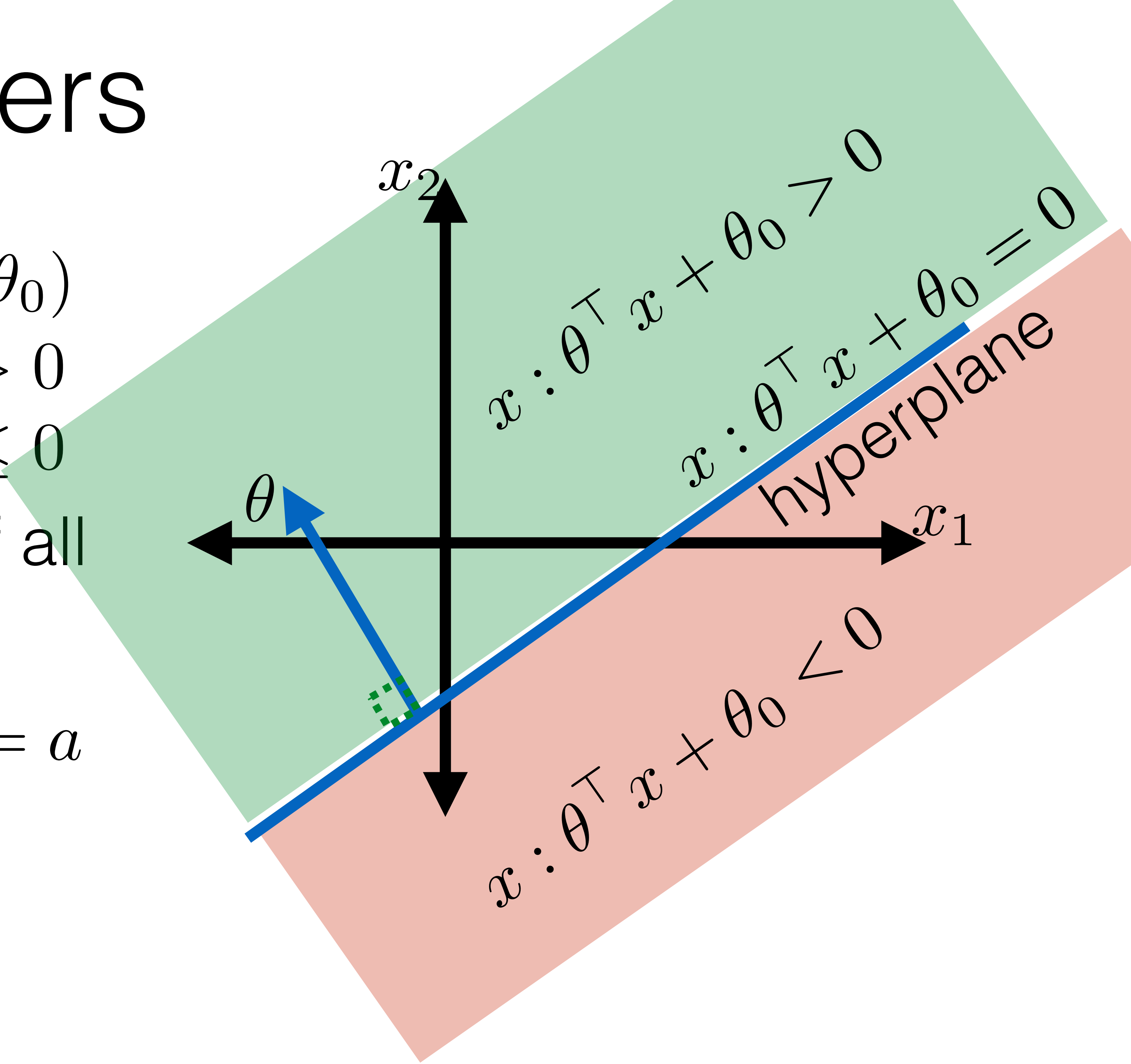
- A linear classifier:

$$h(x;\theta,\theta_0) = \text{sign}(\theta^\top x + \theta_0)$$

$$= \begin{cases} +1 \text{ if } \theta^\top x + \theta_0 > 0 \\ -1 \text{ if } \theta^\top x + \theta_0 \leq 0 \end{cases}$$

- Hypothesis class $\mathcal{H}$ of all linear classifiers
- 0-1 Loss

$$L(g,a) = \begin{cases} 0 \text{ if } g = a \\ 1 \text{ else} \end{cases}$$



Recall: Classifiers

- A linear classifier:

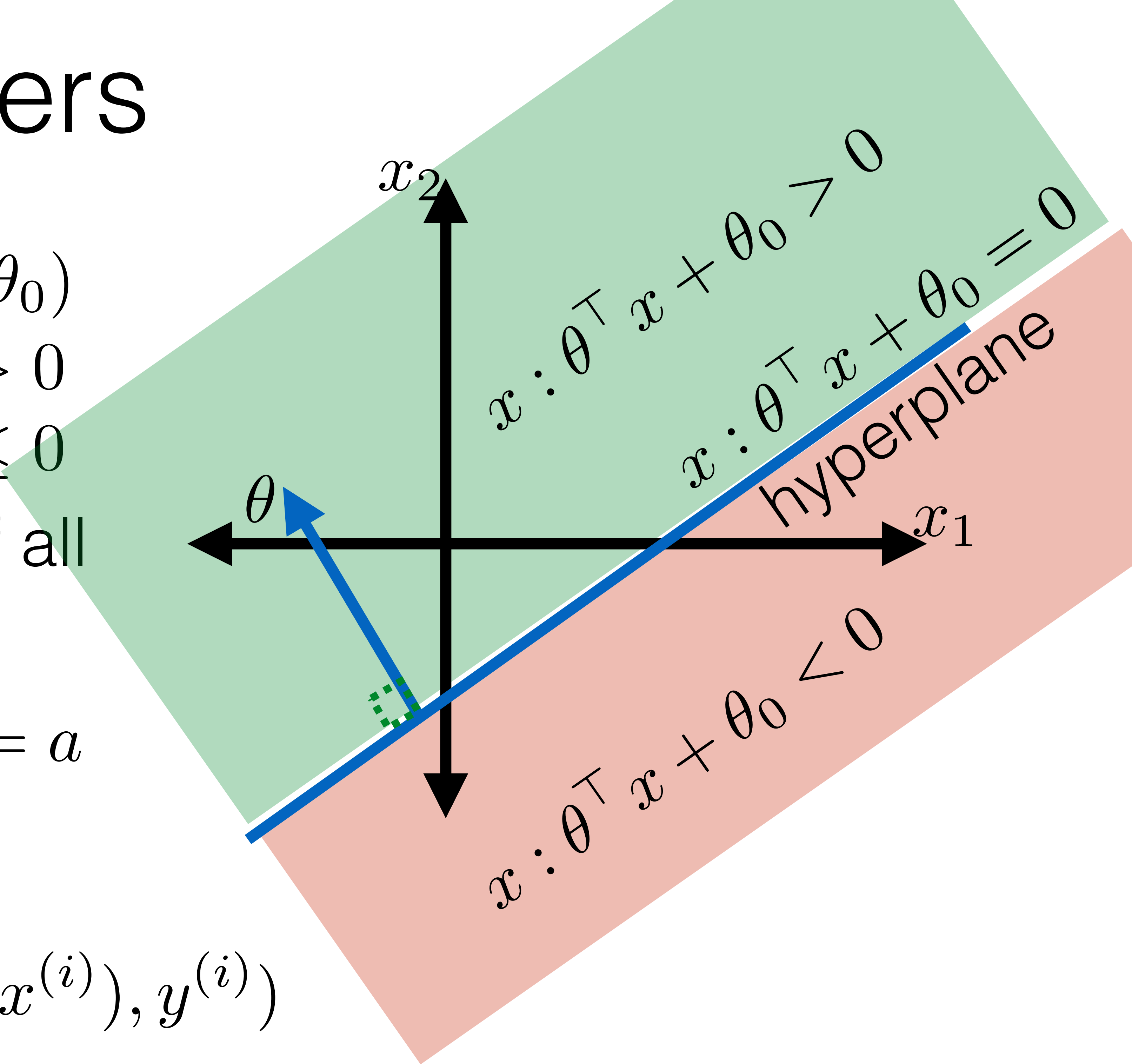
$$h(x;\theta,\theta_0) = \text{sign}(\theta^\top x + \theta_0)$$
$$= \begin{cases} +1 \text{ if } \theta^\top x + \theta_0 > 0 \\ -1 \text{ if } \theta^\top x + \theta_0 \le 0 \end{cases}$$

- Hypothesis class $\mathcal{H}$ of all linear classifiers
- 0-1 Loss

$$L(g,a) = \begin{cases} 0 \text{ if } g = a \\ 1 \text{ else} \end{cases}$$

- Training error

$$\mathcal{E}_n(h) = \frac{1}{n}\sum_{i=1}^{n} L(h(x^{(i)}), y^{(i)})$$



Recall: Classifiers

- A linear classifier:

$$h(x;\theta,\theta_0) = \text{sign}(\theta^\top x + \theta_0)$$
$$= \begin{cases} +1 \text{ if } \theta^\top x + \theta_0 > 0 \\ -1 \text{ if } \theta^\top x + \theta_0 \leq 0 \end{cases}$$

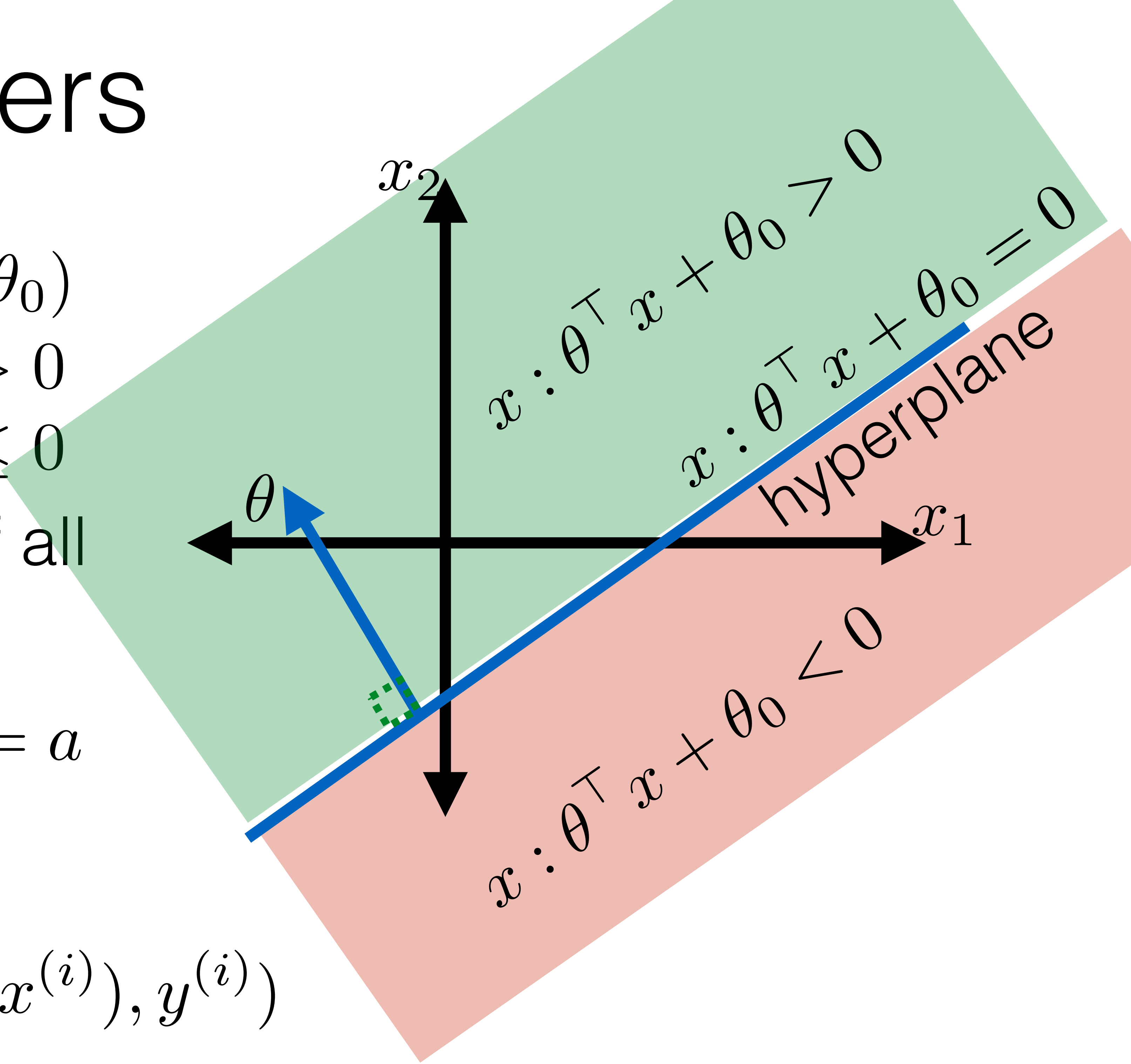
- Hypothesis class $\mathcal{H}$ of all linear classifiers
- 0-1 Loss

$$L(g,a) = \begin{cases} 0 \text{ if } g = a \\ 1 \text{ else} \end{cases}$$

- Training error

$$\mathcal{E}_n(h) = \frac{1}{n}\sum_{i=1}^{n} L(h(x^{(i)}), y^{(i)})$$

- Example learning algorithm (given hypotheses $h^{(j)}$)

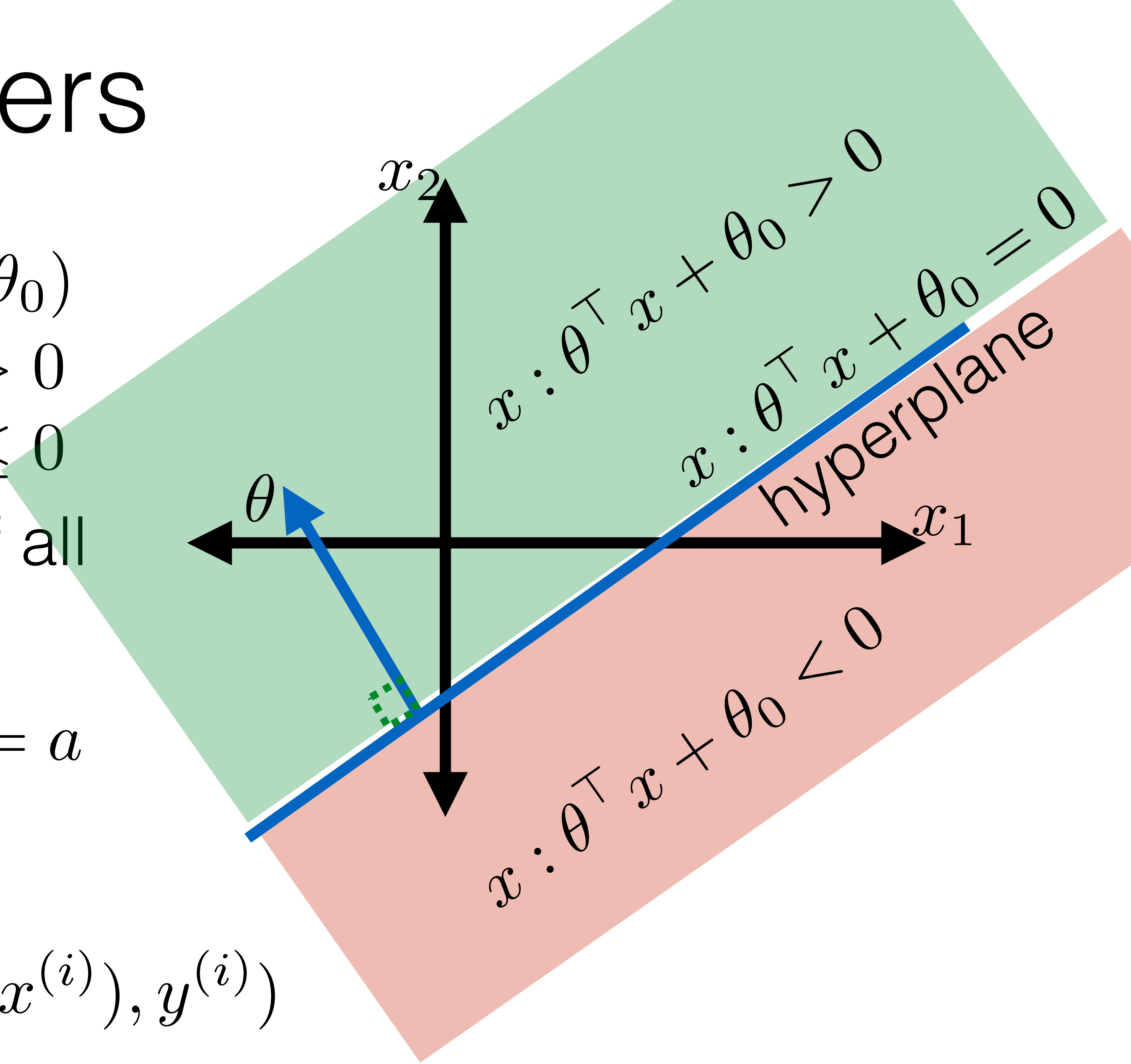


Recall: Classifiers

- A linear classifier:

$$h(x;\theta,\theta_0) = \text{sign}(\theta^\top x + \theta_0)$$

$$= \begin{cases} +1 \text{ if } \theta^\top x + \theta_0 > 0 \\ -1 \text{ if } \theta^\top x + \theta_0 \leq 0 \end{cases}$$



- Hypothesis class $\mathcal{H}$ of all linear classifiers
- 0-1 Loss

$$L(g,a) = \begin{cases} 0 \text{ if } g = a \\ 1 \text{ else} \end{cases}$$

- Training error

$$\mathcal{E}_n(h) = \frac{1}{n}\sum_{i=1}^{n} L(h(x^{(i)}), y^{(i)})$$

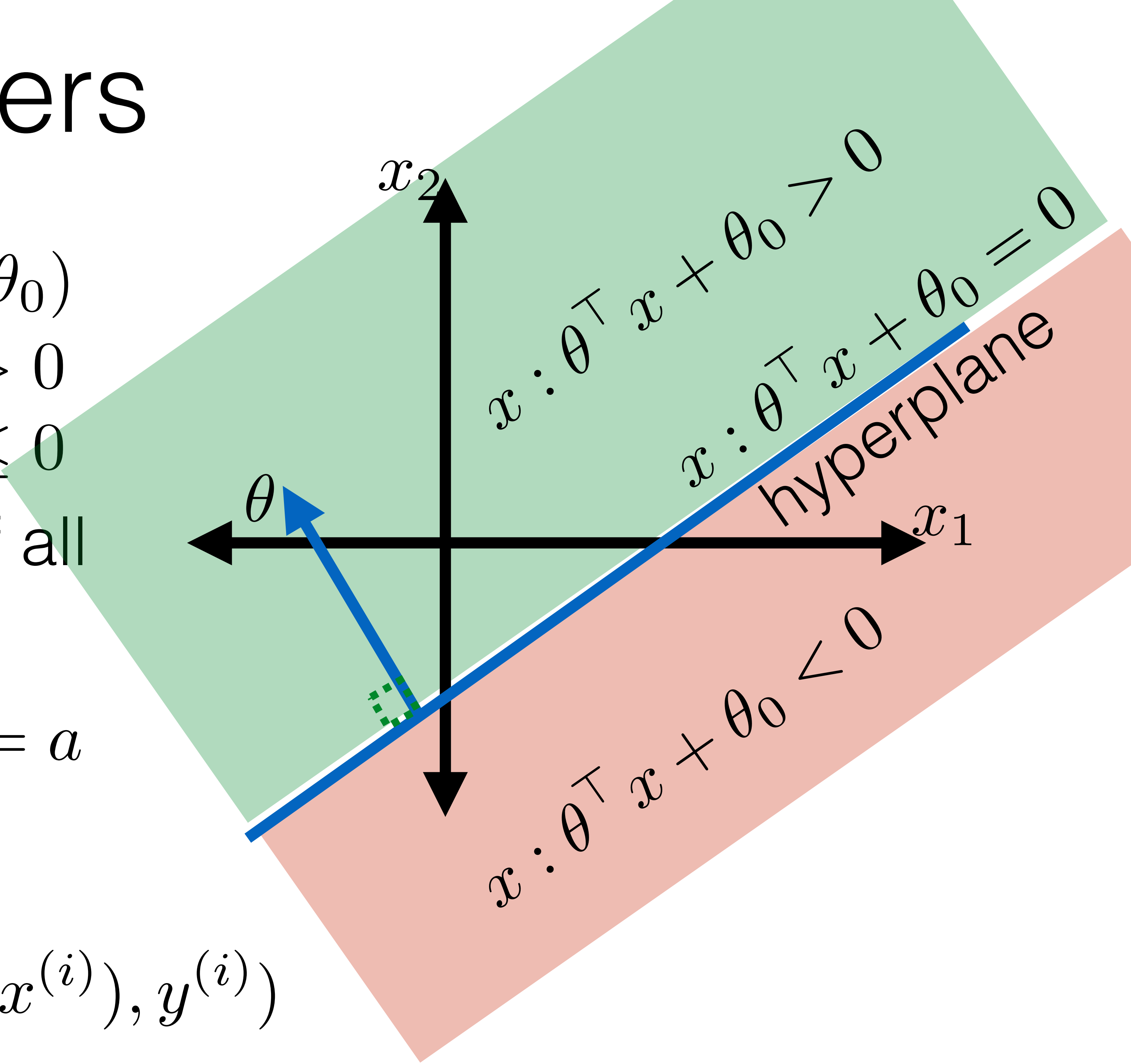
- Example learning algorithm (given hypotheses $h^{(j)}$)

```
Ex_learning_alg( $\mathcal{D}_n$ ; $k$ )
    Set $j^* = \text{argmin}_{j \in \{1,\ldots,k\}} \mathcal{E}_n(h^{(j)})$
    Return $h^{(j^*)}$
```

Recall: Classifiers

- A linear classifier:

$$h(x;\theta,\theta_0) = \text{sign}(\theta^\top x + \theta_0)$$
$$= \begin{cases} +1 \text{ if } \theta^\top x + \theta_0 > 0 \\ -1 \text{ if } \theta^\top x + \theta_0 \leq 0 \end{cases}$$



- Hypothesis class $\mathcal{H}$ of all linear classifiers
- 0-1 Loss

$$L(g,a) = \begin{cases} 0 \text{ if } g = a \\ 1 \text{ else} \end{cases}$$

- Training error

$$\mathcal{E}_n(h) = \frac{1}{n}\sum_{i=1}^{n} L(h(x^{(i)}), y^{(i)})$$

- Example learning algorithm (given hypotheses $h^{(j)}$)

```
Ex_learning_alg( $\mathcal{D}_n$ ; $k$ )
    Set $j^* = \text{argmin}_{j\in\{1,\ldots,k\}}\mathcal{E}_n(h^{(j)})$
    Return $h^{(j^*)}$
```

[demo]

Perceptron Algorithm

Perceptron Algorithm

```
Perceptron
```

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
```

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize $\theta = [0 \;\; 0 \;\; \ldots \;\; 0]^\top$
  Initialize $\theta_0 = 0$
```

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize  $\theta = [0 \;\; 0 \;\; \ldots \;\; 0]^\top$     [How many 0s?]
  Initialize  $\theta_0 = 0$
```

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
    Initialize  $\theta = [0\ \ 0\ \ \ldots\ \ 0]^\top$     [How many 0s?]
    Initialize  $\theta_0 = 0$
    for t = 1 to $\tau$
```

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize  $\theta = [0 \;\; 0 \;\; \ldots \;\; 0]^\top$     [How many 0s?]
  Initialize  $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
```

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize  $\theta = [0 \;\; 0 \;\; \ldots \;\; 0]^\top$     [How many 0s?]
  Initialize  $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
```

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize  $\theta = [0\ \ 0\ \ \ldots\ \ 0]^\top$     [How many 0s?]
  Initialize  $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
```

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize  $\theta = [0 \;\; 0 \;\; \ldots \;\; 0]^\top$     [How many 0s?]
  Initialize  $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
```

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize  $\theta = [0 \ \ 0 \ \ \ldots \ \ 0]^\top$     [How many 0s?]
  Initialize  $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
```

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize  $\theta = [0\ \ 0\ \ \dots\ \ 0]^\top$    [How many 0s?]
  Initialize  $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
```

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize  $\theta = [0 \;\; 0 \;\; \ldots \;\; 0]^\top$     [How many 0s?]
  Initialize  $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
```

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize  $\theta = [0 \;\; 0 \;\; \ldots \;\; 0]^\top$     [How many 0s?]
  Initialize  $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
```

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize  $\theta = [0 \;\; 0 \;\; \ldots \;\; 0]^\top$
  Initialize  $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
```

[How many 0s?]

[i.e. True if either:

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize $\theta = [0 \;\; 0 \;\; \ldots \;\; 0]^\top$
  Initialize $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
```

[How many 0s?]

[i.e. True if either:

A. point is not on the line & prediction is wrong

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize  $\theta = [0 \;\; 0 \;\; \ldots \;\; 0]^\top$
  Initialize  $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
```

[How many 0s?]

[i.e. True if either:

A. point is not on the line & prediction is wrong
B. point is on the line

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize  $\theta = [0 \;\; 0 \;\; \ldots \;\; 0]^\top$
  Initialize  $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
```

[How many 0s?]

[i.e. True if either:

A. point is not on the line & prediction is wrong
B. point is on the line

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize $\theta = [0 \;\; 0 \;\; \ldots \;\; 0]^\top$
  Initialize $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
```

[How many 0s?]

[i.e. True if either:

A. point is not on the line & prediction is wrong
B. point is on the line
C. initial step]

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize $\theta = [0 \;\; 0 \;\; \ldots \;\; 0]^\top$
  Initialize $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
```

[How many 0s?]

[i.e. True if either:

A. point is not on the line & prediction is wrong
B. point is on the line
C. initial step]

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize $\theta = [0 \ \ 0 \ \ \ldots \ \ 0]^\top$
  Initialize $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
        Set $\theta = \theta + y^{(i)} x^{(i)}$
        Set $\theta_0 = \theta_0 + y^{(i)}$
```

[How many 0s?]

[i.e. True if either:

A. point is not on the line & prediction is wrong
B. point is on the line
C. initial step]

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize $\theta = [0\ \ 0\ \ \ldots\ \ 0]^\top$
  Initialize $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
        Set $\theta = \theta + y^{(i)} x^{(i)}$
        Set $\theta_0 = \theta_0 + y^{(i)}$
        changed = True
```

[How many 0s?]

[i.e. True if either:

A. point is not on the line & prediction is wrong
B. point is on the line
C. initial step]

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize $\theta = [0 \;\; 0 \;\; \ldots \;\; 0]^\top$
  Initialize $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
        Set $\theta = \theta + y^{(i)} x^{(i)}$
        Set $\theta_0 = \theta_0 + y^{(i)}$
        changed = True
    if not changed
      break
```

[How many 0s?]

[i.e. True if either:
A. point is not on the line
& prediction is wrong
B. point is on the line
C. initial step]

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize $\theta = [0 \;\; 0 \;\; \ldots \;\; 0]^\top$
  Initialize $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
          Set $\theta = \theta + y^{(i)} x^{(i)}$
          Set $\theta_0 = \theta_0 + y^{(i)}$
          changed = True
    if not changed
      break
  Return $\theta, \theta_0$
```

[How many 0s?]

[i.e. True if either:

A. point is not on the line & prediction is wrong
B. point is on the line
C. initial step]

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize $\theta = [0 \ 0 \ \ldots \ 0]^\top$
  Initialize $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
        Set $\theta = \theta + y^{(i)} x^{(i)}$
        Set $\theta_0 = \theta_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return $\theta, \theta_0$
```

[How many 0s?]

[i.e. True if either:

A. point is not on the line & prediction is wrong
B. point is on the line
C. initial step]

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize  $\theta = [0 \;\; 0 \;\; \ldots \;\; 0]^\top$
  Initialize  $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
        Set  $\theta = \theta + y^{(i)}x^{(i)}$
        Set  $\theta_0 = \theta_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return  $\theta, \theta_0$
```

[How many 0s?]

[i.e. True if either:

A. point is not on the line & prediction is wrong

B. point is on the line

C. initial step]

What does an update do?

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize  $\theta = [0 \;\; 0 \;\; \ldots \;\; 0]^\top$
  Initialize  $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
        Set  $\theta = \theta + y^{(i)} x^{(i)}$
        Set  $\theta_0 = \theta_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return  $\theta, \theta_0$
```

[How many 0s?]

[i.e. True if either:

A. point is not on the line & prediction is wrong
B. point is on the line
C. initial step]

What does an update do?

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize  $\theta = [0 \;\; 0 \;\; \ldots \;\; 0]^\top$
  Initialize  $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
        Set  $\theta = \theta + y^{(i)} x^{(i)}$
        Set  $\theta_0 = \theta_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return  $\theta, \theta_0$
```

[How many 0s?]

[i.e. True if either:

A. point is not on the line & prediction is wrong
B. point is on the line
C. initial step]

What does an update do?

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize $\theta = [0 \;\; 0 \;\; \ldots \;\; 0]^\top$
  Initialize $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
        Set $\theta = \theta + y^{(i)}x^{(i)}$
        Set $\theta_0 = \theta_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return $\theta, \theta_0$
```

[How many 0s?]

[i.e. True if either:

A. point is not on the line & prediction is wrong
B. point is on the line
C. initial step]

What does an update do?

$$y^{(i)}\left(\boxed{\theta_{\text{updated}}^\top} x^{(i)} + \boxed{\theta_{0,\text{updated}}}\right)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize $\theta = [0 \;\; 0 \;\; \ldots \;\; 0]^\top$
  Initialize $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \le 0$
        Set $\theta = \theta + y^{(i)} x^{(i)}$
        Set $\theta_0 = \theta_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return $\theta, \theta_0$
```

[How many 0s?]

[i.e. True if either:
A. point is not on the line
   & prediction is wrong
B. point is on the line
C. initial step]

What does an update do?

$$y^{(i)} \left((\theta + y^{(i)} x^{(i)})^\top x^{(i)} + (\theta_0 + y^{(i)}) \right)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize  $\theta = [0 \;\; 0 \;\; \ldots \;\; 0]^\top$
  Initialize  $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
        Set  $\theta = \theta + y^{(i)} x^{(i)}$
        Set  $\theta_0 = \theta_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return  $\theta, \theta_0$
```

[How many 0s?]

[i.e. True if either:

A. point is not on the line & prediction is wrong
B. point is on the line
C. initial step]

What does an update do?

$$y^{(i)}\left((\theta + y^{(i)}x^{(i)})^\top x^{(i)} + (\theta_0 + y^{(i)})\right)$$

$$= y^{(i)}(\theta^\top x^{(i)} + \theta_0) + (y^{(i)})^2(x^{(i)\top} x^{(i)} + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize  $\theta = [0 \ \ 0 \ \ \ldots \ \ 0]^\top$
  Initialize  $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
        Set  $\theta = \theta + y^{(i)} x^{(i)}$
        Set  $\theta_0 = \theta_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return  $\theta, \theta_0$
```

[How many 0s?]

[i.e. True if either:

A. point is not on the line
& prediction is wrong

B. point is on the line

C. initial step]

What does an update do?

$$y^{(i)}\left(\left(\theta + y^{(i)}x^{(i)}\right)^\top x^{(i)} + \left(\theta_0 + y^{(i)}\right)\right)$$

$$= y^{(i)}(\theta^\top x^{(i)} + \theta_0) + (y^{(i)})^2(x^{(i)\top} x^{(i)} + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize  $\theta = [0 \ \ 0 \ \ \ldots \ \ 0]^\top$
  Initialize  $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
        Set  $\theta = \theta + y^{(i)} x^{(i)}$
        Set  $\theta_0 = \theta_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return  $\theta, \theta_0$
```

[How many 0s?]

[i.e. True if either:

A. point is not on the line & prediction is wrong

B. point is on the line

C. initial step]

What does an update do?

$$y^{(i)}\left((\theta + y^{(i)}x^{(i)})^\top x^{(i)} + (\theta_0 + y^{(i)})\right)$$

$$= y^{(i)}(\theta^\top x^{(i)} + \theta_0) + (y^{(i)})^2(x^{(i)\top} x^{(i)} + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize  $\theta = [0 \ 0 \ \ldots \ 0]^\top$
  Initialize  $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
        Set  $\theta = \theta + y^{(i)} x^{(i)}$
        Set  $\theta_0 = \theta_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return  $\theta, \theta_0$
```

[How many 0s?]

[i.e. True if either:

A. point is not on the line & prediction is wrong

B. point is on the line

C. initial step]

What does an update do?

$$y^{(i)}\left(\left(\theta + y^{(i)} x^{(i)}\right)^\top x^{(i)} + \left(\theta_0 + y^{(i)}\right)\right)$$

$$= y^{(i)}(\theta^\top x^{(i)} + \theta_0) + (y^{(i)})^2 (x^{(i)\top} x^{(i)} + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize  $\theta = [0 \;\; 0 \;\; \ldots \;\; 0]^\top$
  Initialize  $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
        Set  $\theta = \theta + y^{(i)} x^{(i)}$
        Set  $\theta_0 = \theta_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return  $\theta, \theta_0$
```

[How many 0s?]

[i.e. True if either:

A. point is not on the line & prediction is wrong
B. point is on the line
C. initial step]

What does an update do?

$$y^{(i)}\left(\left(\theta + y^{(i)} x^{(i)}\right)^\top x^{(i)} + \left(\theta_0 + y^{(i)}\right)\right)$$

$$= y^{(i)}(\theta^\top x^{(i)} + \theta_0) + (y^{(i)})^2 (x^{(i)\top} x^{(i)} + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize  $\theta = [0 \;\; 0 \;\; \ldots \;\; 0]^\top$
  Initialize  $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
        Set  $\theta = \theta + y^{(i)} x^{(i)}$
        Set  $\theta_0 = \theta_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return  $\theta, \theta_0$
```

[How many 0s?]

[i.e. True if either:

A. point is not on the line & prediction is wrong
B. point is on the line
C. initial step]

What does an update do?

$$y^{(i)}\left(\left(\theta + y^{(i)}x^{(i)}\right)^\top x^{(i)} + \left(\theta_0 + y^{(i)}\right)\right)$$

$$= y^{(i)}(\theta^\top x^{(i)} + \theta_0) + (y^{(i)})^2(x^{(i)\top} x^{(i)} + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize  $\theta = [0 \;\; 0 \;\; \ldots \;\; 0]^\top$
  Initialize  $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
        Set  $\theta = \theta + y^{(i)} x^{(i)}$
        Set  $\theta_0 = \theta_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return  $\theta, \theta_0$
```

[How many 0s?]

[i.e. True if either:

A. point is not on the line & prediction is wrong
B. point is on the line
C. initial step]

What does an update do?

$$y^{(i)}\left(\left(\theta + y^{(i)}x^{(i)}\right)^\top x^{(i)} + \left(\theta_0 + y^{(i)}\right)\right)$$

$$= y^{(i)}(\theta^\top x^{(i)} + \theta_0) + (y^{(i)})^2(x^{(i)\top} x^{(i)} + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize $\theta = [0 \ 0 \ \ldots \ 0]^\top$
  Initialize $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
        Set $\theta = \theta + y^{(i)}x^{(i)}$
        Set $\theta_0 = \theta_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return $\theta, \theta_0$
```

[How many 0s?]

[i.e. True if either:

A. point is not on the line & prediction is wrong
B. point is on the line
C. initial step]

What does an update do?

$$y^{(i)}\left((\theta + y^{(i)}x^{(i)})^\top x^{(i)} + (\theta_0 + y^{(i)})\right)$$
$$= y^{(i)}(\theta^\top x^{(i)} + \theta_0) + (y^{(i)})^2(x^{(i)\top}x^{(i)} + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize $\theta = [0 \;\; 0 \;\; \ldots \;\; 0]^\top$
  Initialize $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
        Set $\theta = \theta + y^{(i)} x^{(i)}$
        Set $\theta_0 = \theta_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return $\theta, \theta_0$
```

[How many 0s?]

[i.e. True if either:

A. point is not on the line & prediction is wrong

B. point is on the line

C. initial step]

What does an update do?

$$y^{(i)}\left(\left(\theta + y^{(i)}x^{(i)}\right)^\top x^{(i)} + \left(\theta_0 + y^{(i)}\right)\right)$$

$$= y^{(i)}(\theta^\top x^{(i)} + \theta_0) + \left(y^{(i)}\right)^2 \left(x^{(i)\top} x^{(i)} + 1\right)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize $\theta = [0 \;\; 0 \;\; \ldots \;\; 0]^\top$
  Initialize $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
        Set $\theta = \theta + y^{(i)} x^{(i)}$
        Set $\theta_0 = \theta_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return $\theta, \theta_0$
```

[How many 0s?]

[i.e. True if either:

A. point is not on the line & prediction is wrong
B. point is on the line
C. initial step]

What does an update do?

$$y^{(i)}\left((\theta + y^{(i)}x^{(i)})^\top x^{(i)} + (\theta_0 + y^{(i)})\right)$$

$$= y^{(i)}(\theta^\top x^{(i)} + \theta_0) + (y^{(i)})^2(x^{(i)\top} x^{(i)} + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize  $\theta = [0 \;\; 0 \;\; \ldots \;\; 0]^\top$
  Initialize  $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
        Set  $\theta = \theta + y^{(i)}x^{(i)}$
        Set  $\theta_0 = \theta_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return  $\theta, \theta_0$
```

[How many 0s?]

[i.e. True if either:

A. point is not on the line & prediction is wrong
B. point is on the line
C. initial step]

What does an update do?

$$y^{(i)}\left((\theta + y^{(i)}x^{(i)})^\top x^{(i)} + (\theta_0 + y^{(i)})\right)$$

$$= y^{(i)}(\theta^\top x^{(i)} + \theta_0) + (y^{(i)})^2(x^{(i)\top}x^{(i)} + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize  $\theta = [0 \ \ 0 \ \ \ldots \ \ 0]^\top$
  Initialize  $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
        Set  $\theta = \theta + y^{(i)} x^{(i)}$
        Set  $\theta_0 = \theta_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return  $\theta, \theta_0$
```

[How many 0s?]

[i.e. True if either:

A. point is not on the line & prediction is wrong
B. point is on the line
C. initial step]

What does an update do?

$$y^{(i)}\left((\theta + y^{(i)}x^{(i)})^\top x^{(i)} + (\theta_0 + y^{(i)})\right)$$

$$= y^{(i)}(\theta^\top x^{(i)} + \theta_0) + (y^{(i)})^2(x^{(i)\top}x^{(i)} + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize  $\theta = [0 \;\; 0 \;\; \ldots \;\; 0]^\top$
  Initialize  $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
        Set  $\theta = \theta + y^{(i)} x^{(i)}$
        Set  $\theta_0 = \theta_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return  $\theta, \theta_0$
```

[How many 0s?]

[i.e. True if either:

A. point is not on the line & prediction is wrong
B. point is on the line
C. initial step]

What does an update do?

$$y^{(i)}\left((\theta + y^{(i)}x^{(i)})^\top x^{(i)} + (\theta_0 + y^{(i)})\right)$$

$$= y^{(i)}(\theta^\top x^{(i)} + \theta_0) + (y^{(i)})^2(x^{(i)\top}x^{(i)} + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize $\theta = [0 \;\; 0 \;\; \ldots \;\; 0]^\top$
  Initialize $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
        Set $\theta = \theta + y^{(i)} x^{(i)}$
        Set $\theta_0 = \theta_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return $\theta, \theta_0$
```

[How many 0s?]

[i.e. True if either:

A. point is not on the line & prediction is wrong
B. point is on the line
C. initial step]

What does an update do?

$$y^{(i)}\left((\theta + y^{(i)}x^{(i)})^\top x^{(i)} + (\theta_0 + y^{(i)})\right)$$

$$= y^{(i)}(\theta^\top x^{(i)} + \theta_0) + (y^{(i)})^2(x^{(i)\top} x^{(i)} + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize  $\theta = [0 \;\; 0 \;\; \dots \;\; 0]^\top$
  Initialize  $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
        Set  $\theta = \theta + y^{(i)} x^{(i)}$
        Set  $\theta_0 = \theta_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return  $\theta, \theta_0$
```

[How many 0s?]

[i.e. True if either:

A. point is not on the line & prediction is wrong

B. point is on the line

C. initial step]

What does an update do?

$$y^{(i)} \left((\theta + y^{(i)} x^{(i)})^\top x^{(i)} + (\theta_0 + y^{(i)}) \right)$$

$$= y^{(i)}(\theta^\top x^{(i)} + \theta_0) + (y^{(i)})^2 (x^{(i)\top} x^{(i)} + 1)$$

$$= y^{(i)}(\theta^\top x^{(i)} + \theta_0) + (\|x^{(i)}\|^2 + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize  $\theta = [0 \;\; 0 \;\; \ldots \;\; 0]^\top$
  Initialize  $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
        Set  $\theta = \theta + y^{(i)} x^{(i)}$
        Set  $\theta_0 = \theta_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return  $\theta, \theta_0$
```

[How many 0s?]

[i.e. True if either:

A. point is not on the line & prediction is wrong
B. point is on the line
C. initial step]

What does an update do?

$$y^{(i)}\left((\theta + y^{(i)}x^{(i)})^\top x^{(i)} + (\theta_0 + y^{(i)})\right)$$

$$= y^{(i)}(\theta^\top x^{(i)} + \theta_0) + (y^{(i)})^2(x^{(i)\top}x^{(i)} + 1)$$

$$= y^{(i)}(\theta^\top x^{(i)} + \theta_0) + (\|x^{(i)}\|^2 + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize  $\theta = [0 \;\; 0 \;\; \dots \;\; 0]^\top$
  Initialize  $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
        Set  $\theta = \theta + y^{(i)}x^{(i)}$
        Set  $\theta_0 = \theta_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return  $\theta, \theta_0$
```

[How many 0s?]

[i.e. True if either:

A. point is not on the line & prediction is wrong
B. point is on the line
C. initial step]

What does an update do?

$$y^{(i)}\left((\theta + y^{(i)}x^{(i)})^\top x^{(i)} + (\theta_0 + y^{(i)})\right)$$

$$= y^{(i)}(\theta^\top x^{(i)} + \theta_0) + (y^{(i)})^2(x^{(i)\top}x^{(i)} + 1)$$

$$= y^{(i)}(\theta^\top x^{(i)} + \theta_0) + (\|x^{(i)}\|^2 + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize $\theta = [0 \;\; 0 \;\; \ldots \;\; 0]^\top$
  Initialize $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
        Set $\theta = \theta + y^{(i)} x^{(i)}$
        Set $\theta_0 = \theta_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return $\theta, \theta_0$
```

[How many 0s?]

[i.e. True if either:

A. point is not on the line & prediction is wrong
B. point is on the line
C. initial step]

What does an update do?

$$y^{(i)} \left((\theta + y^{(i)} x^{(i)})^\top x^{(i)} + (\theta_0 + y^{(i)}) \right)$$

$$= y^{(i)}(\theta^\top x^{(i)} + \theta_0) + (y^{(i)})^2 (x^{(i)\top} x^{(i)} + 1)$$

$$= y^{(i)}(\theta^\top x^{(i)} + \theta_0) + (\|x^{(i)}\|^2 + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize  $\theta = [0 \;\; 0 \;\; \dots \;\; 0]^\top$
  Initialize  $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \le 0$
        Set  $\theta = \theta + y^{(i)}x^{(i)}$
        Set  $\theta_0 = \theta_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return  $\theta, \theta_0$
```

[How many 0s?]

[i.e. True if either:

A. point is not on the line
   & prediction is wrong
B. point is on the line
C. initial step]

What does an update do?

$$y^{(i)}\left((\theta + y^{(i)}x^{(i)})^\top x^{(i)} + (\theta_0 + y^{(i)})\right)$$

$$= y^{(i)}(\theta^\top x^{(i)} + \theta_0) + (y^{(i)})^2(x^{(i)\top}x^{(i)} + 1)$$

$$= y^{(i)}(\theta^\top x^{(i)} + \theta_0) + (\|x^{(i)}\|^2 + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $\tau$ )
  Initialize $\theta = [0 \ 0 \ \ldots \ 0]^\top$
  Initialize $\theta_0 = 0$
  for t = 1 to $\tau$
    changed = False
    for i = 1 to n
      if $y^{(i)}(\theta^\top x^{(i)} + \theta_0) \leq 0$
          Set $\theta = \theta + y^{(i)}x^{(i)}$
          Set $\theta_0 = \theta_0 + y^{(i)}$
          changed = True
    if not changed
      break
  Return $\theta, \theta_0$
```

[demo]

[How many 0s?]

[i.e. True if either:

A. point is not on the line & prediction is wrong

B. point is on the line

C. initial step]

What does an update do?

$$y^{(i)}\left((\theta + y^{(i)}x^{(i)})^\top x^{(i)} + (\theta_0 + y^{(i)})\right)$$

$$= y^{(i)}(\theta^\top x^{(i)} + \theta_0) + (y^{(i)})^2(x^{(i)\top}x^{(i)} + 1)$$

$$= y^{(i)}(\theta^\top x^{(i)} + \theta_0) + (\|x^{(i)}\|^2 + 1)$$

Let’s Talk About Classifier Quality

Let’s Talk About Classifier Quality

- *Definition*: A training set $\mathcal{D}_n$ is **linearly separable** if there exist θ, θ_0 such that, for every point index $i \in \{1, \ldots, n\}$, we have

$$y^{(i)}(\theta^\top x^{(i)} + \theta_0) > 0$$

Let’s Talk About Classifier Quality

- *Definition*: A training set $\mathcal{D}_n$ is **linearly separable** if there exist θ, θ_0 such that, for every point index $i \in \{1, \ldots, n\}$, we have

$$y^{(i)}(\theta^\top x^{(i)} + \theta_0) > 0$$

Let’s Talk About Classifier Quality

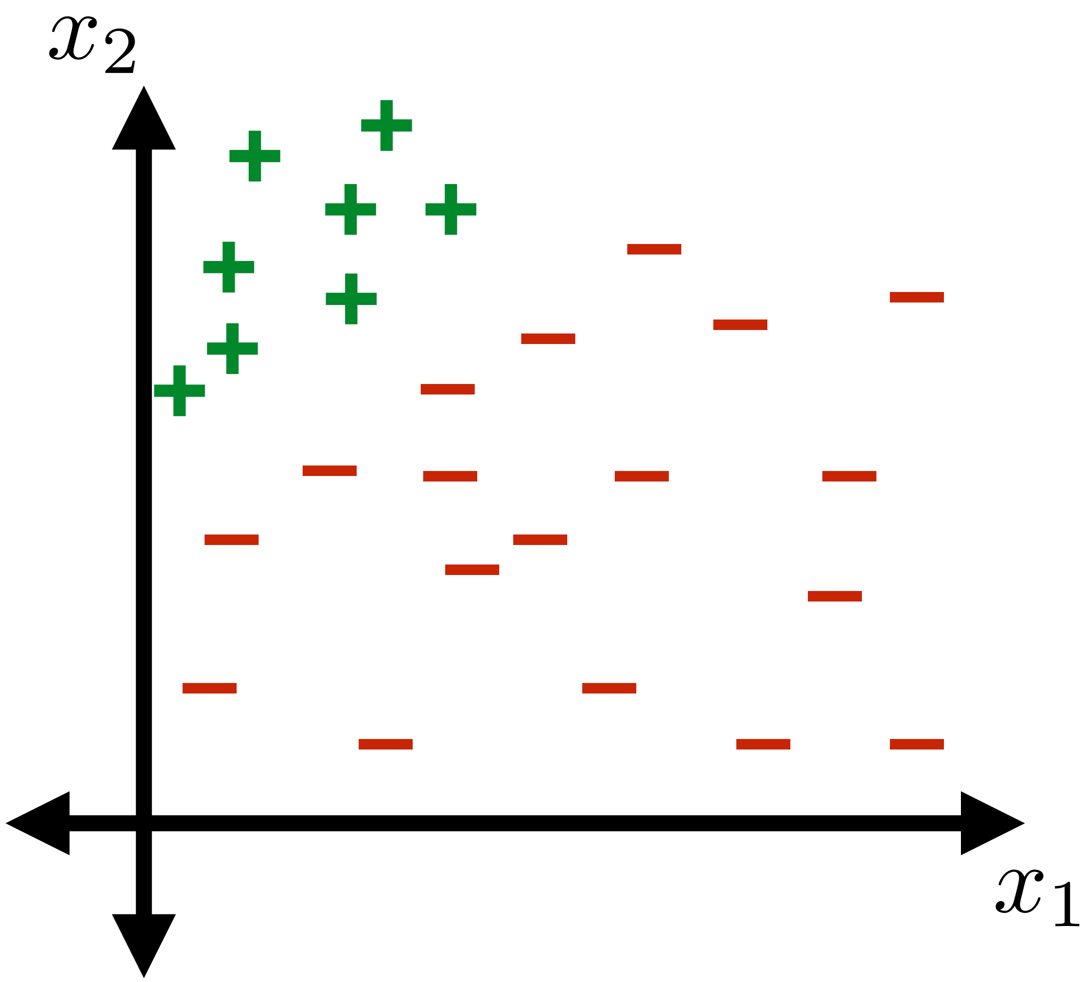
- *Definition*: A training set $\mathcal{D}_n$ is **linearly separable** if there exist θ, θ_0 such that, for every point index $i \in \{1, \ldots, n\}$, we have

$$y^{(i)}(\theta^\top x^{(i)} + \theta_0) > 0$$

Let’s Talk About Classifier Quality

- *Definition*: A training set $\mathcal{D}_n$ is **linearly separable** if there exist θ, θ_0 such that, for every point index $i \in \{1, \ldots, n\}$, we have

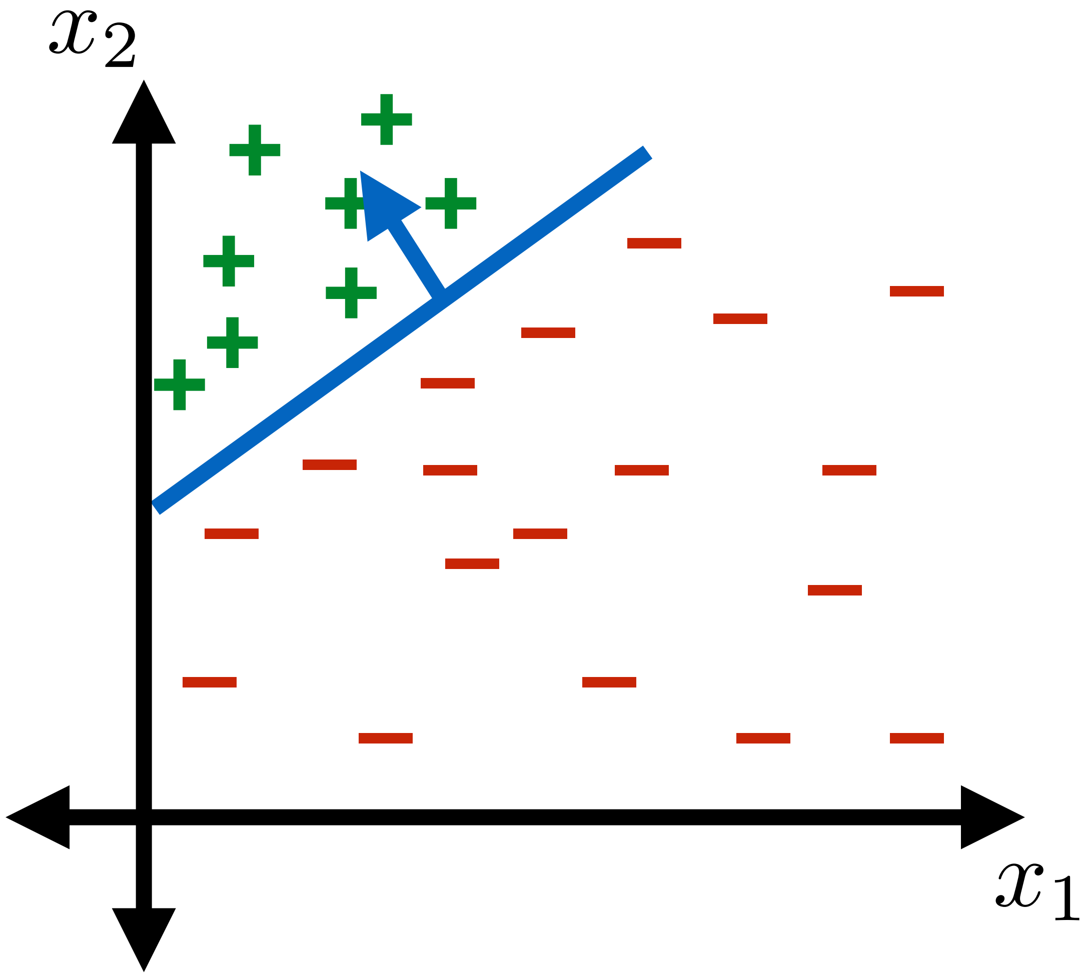
$$y^{(i)}(\theta^\top x^{(i)} + \theta_0) > 0$$



Let’s Talk About Classifier Quality

- *Definition*: A training set $\mathcal{D}_n$ is **linearly separable** if there exist θ, θ_0 such that, for every point index $i \in \{1, \ldots, n\}$, we have

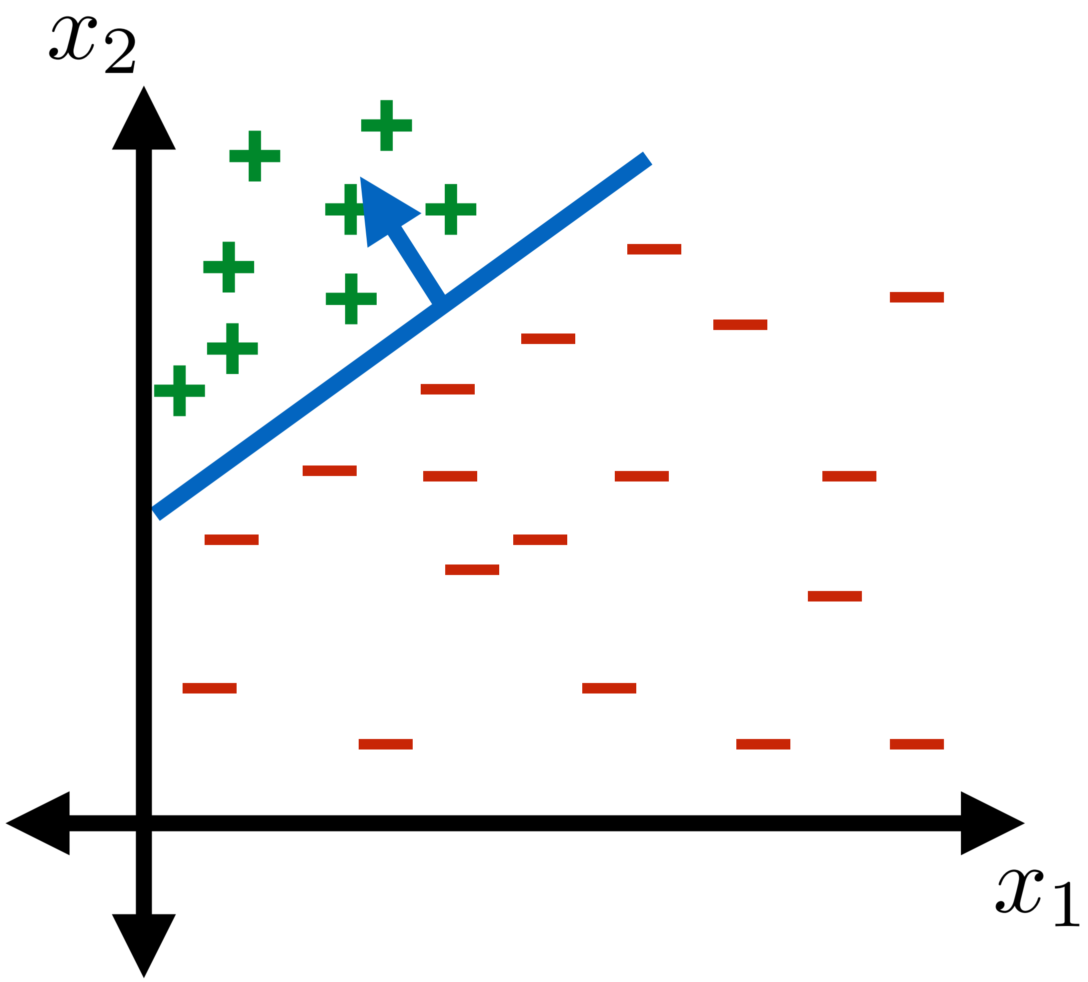
$$y^{(i)}(\theta^\top x^{(i)} + \theta_0) > 0$$

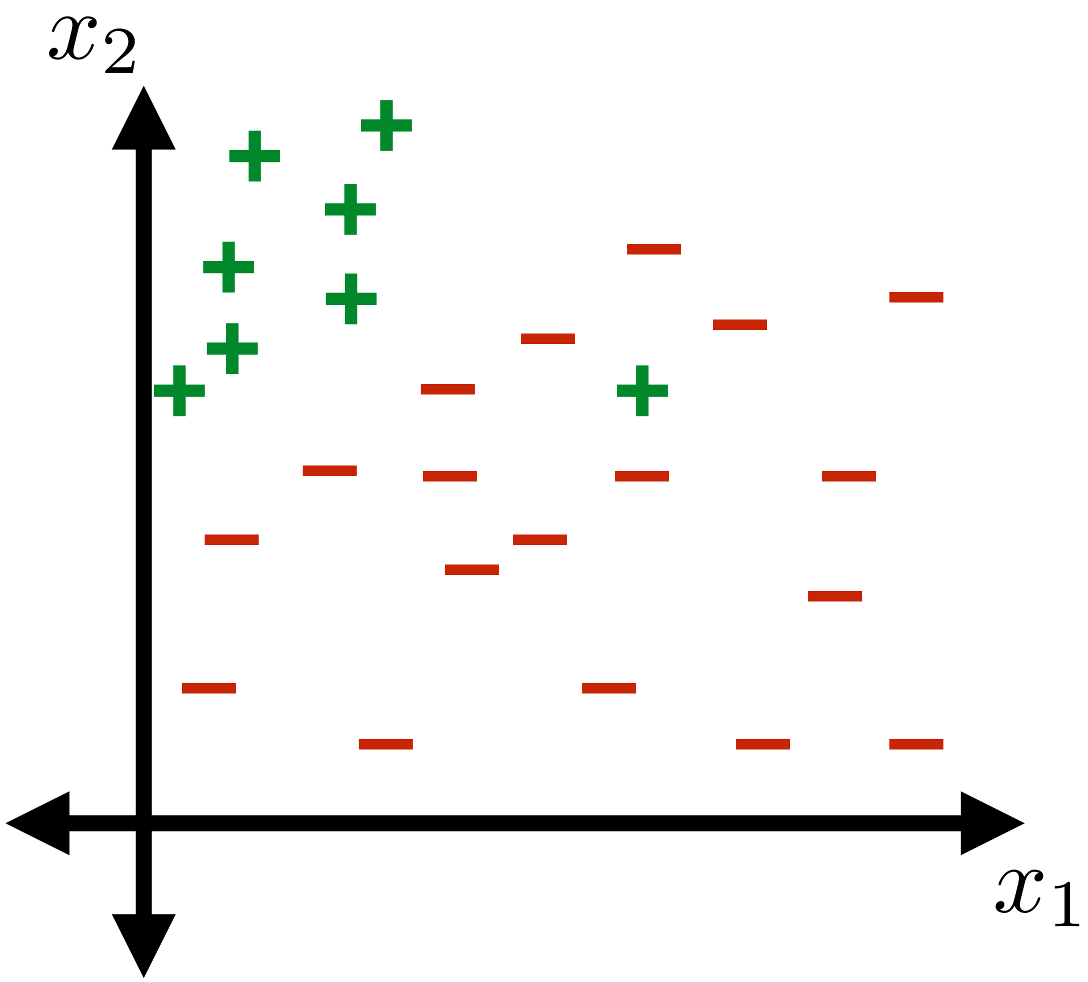


Let’s Talk About Classifier Quality

- *Definition*: A training set $\mathcal{D}_n$ is **linearly separable** if there exist θ, θ_0 such that, for every point index $i \in \{1, \ldots, n\}$, we have

$$y^{(i)}(\theta^\top x^{(i)} + \theta_0) > 0$$

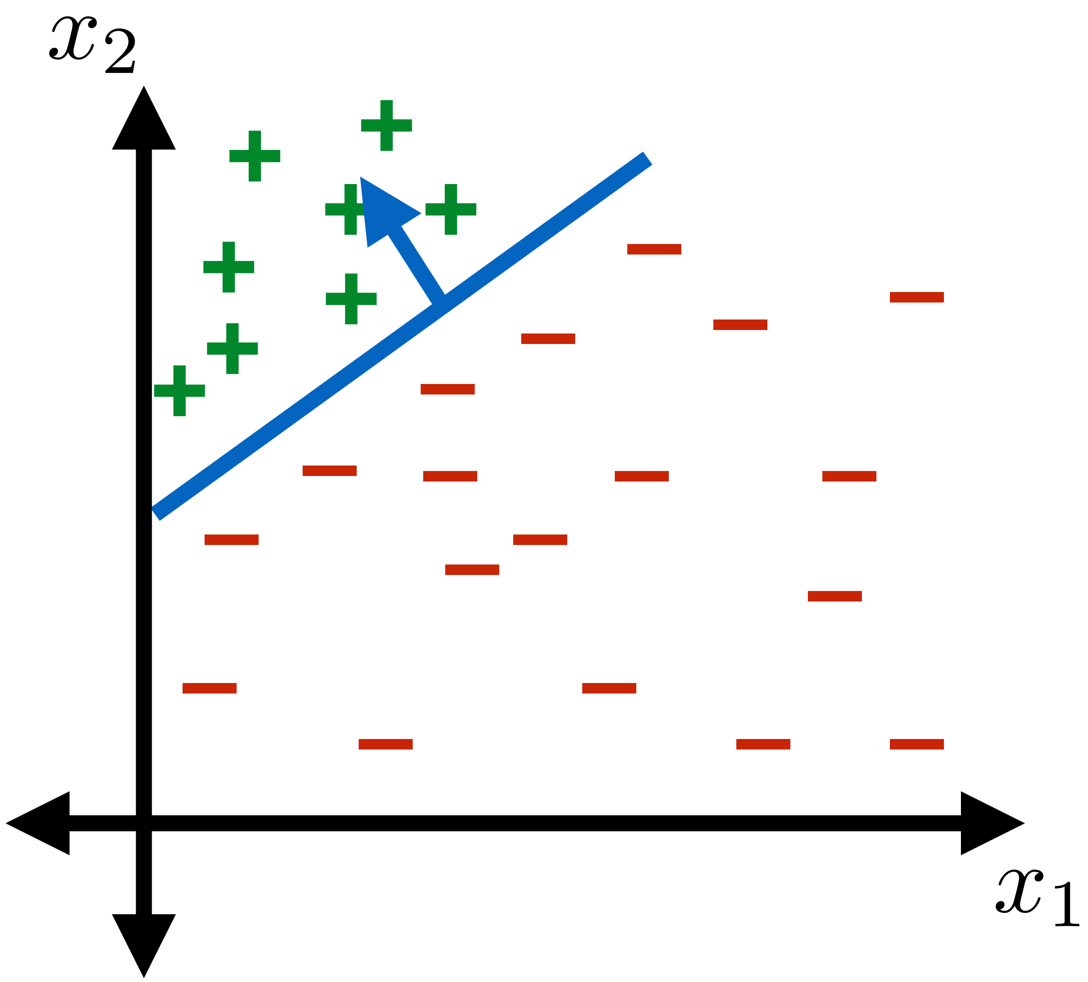


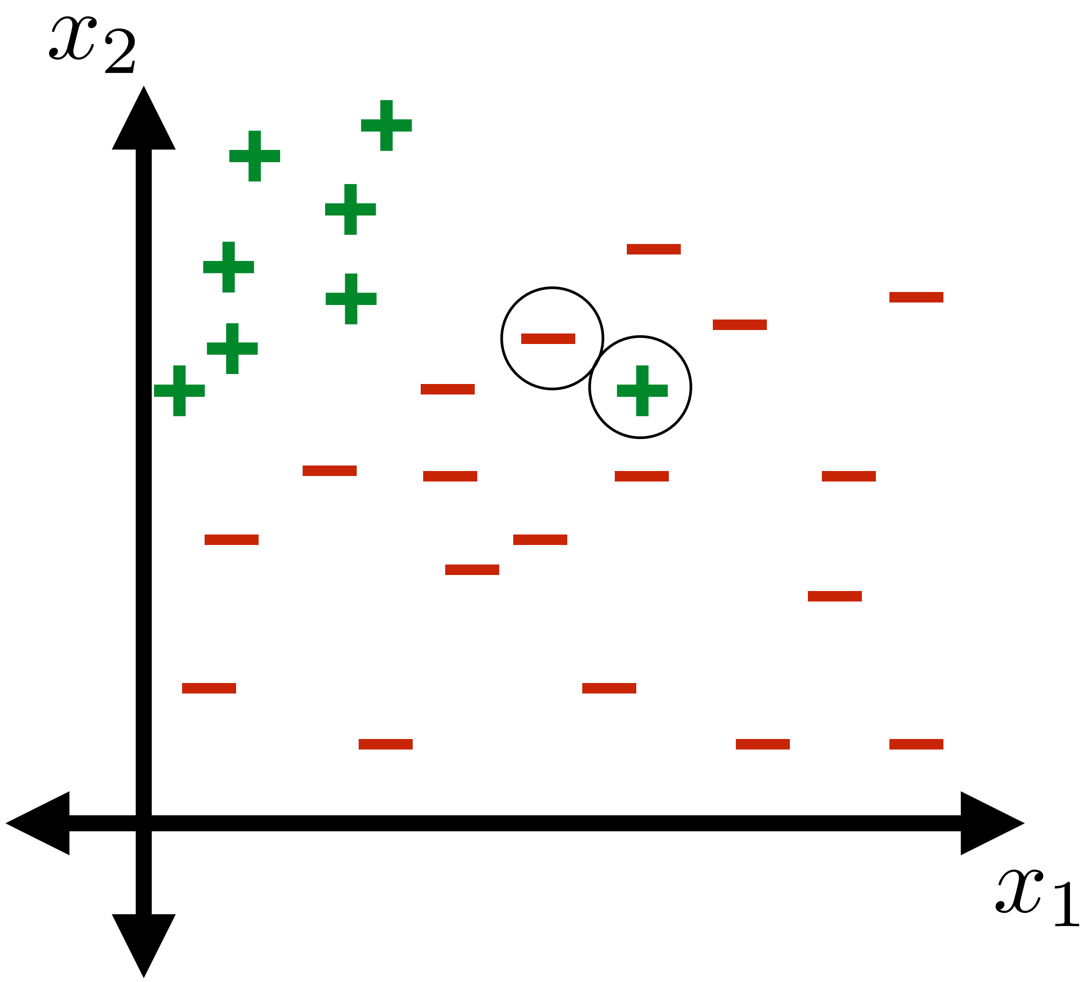


Let's Talk About Classifier Quality

- *Definition*: A training set $\mathcal{D}_n$ is **linearly separable** if there exist θ, θ_0 such that, for every point index $i \in \{1, \ldots, n\}$, we have

$$y^{(i)}(\theta^\top x^{(i)} + \theta_0) > 0$$

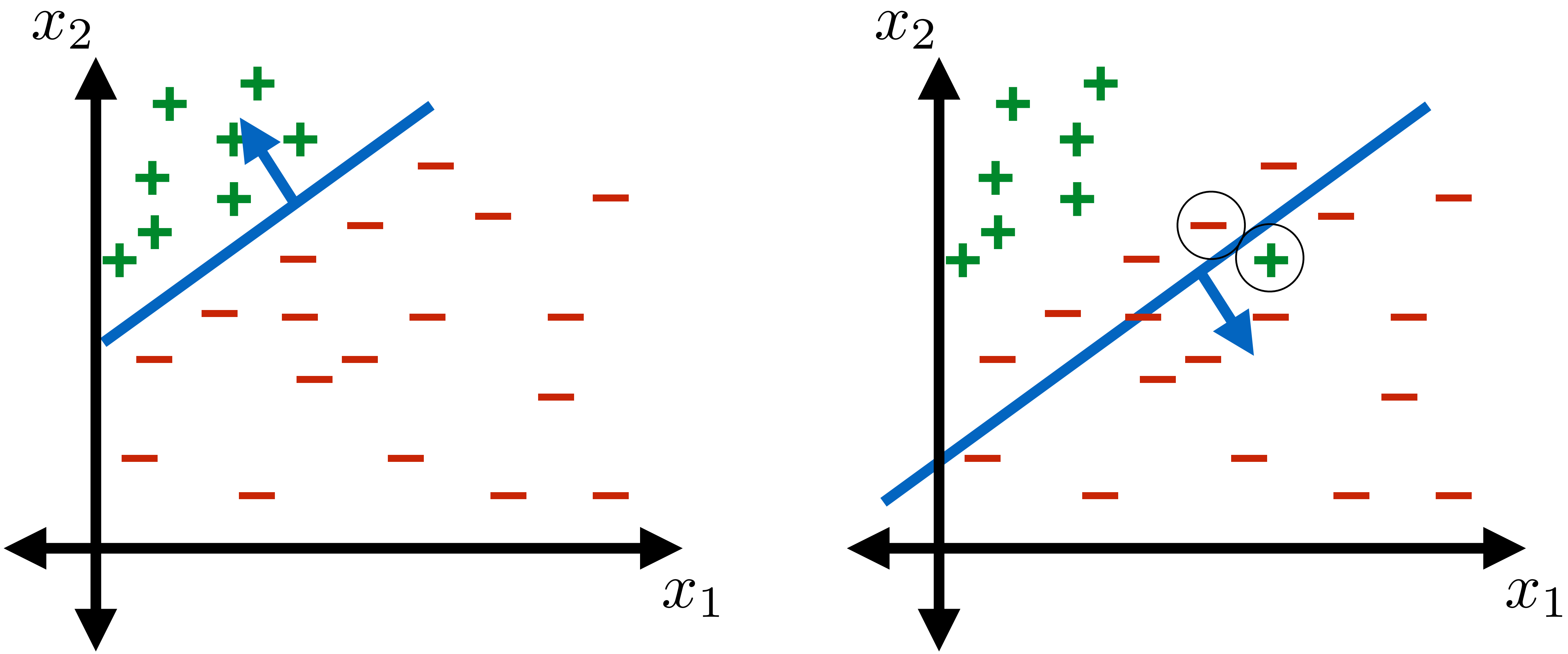




Let’s Talk About Classifier Quality

- *Definition*: A training set $\mathcal{D}_n$ is **linearly separable** if there exist θ, θ_0 such that, for every point index $i \in \{1, \ldots, n\}$, we have

$$y^{(i)}(\theta^\top x^{(i)} + \theta_0) > 0$$

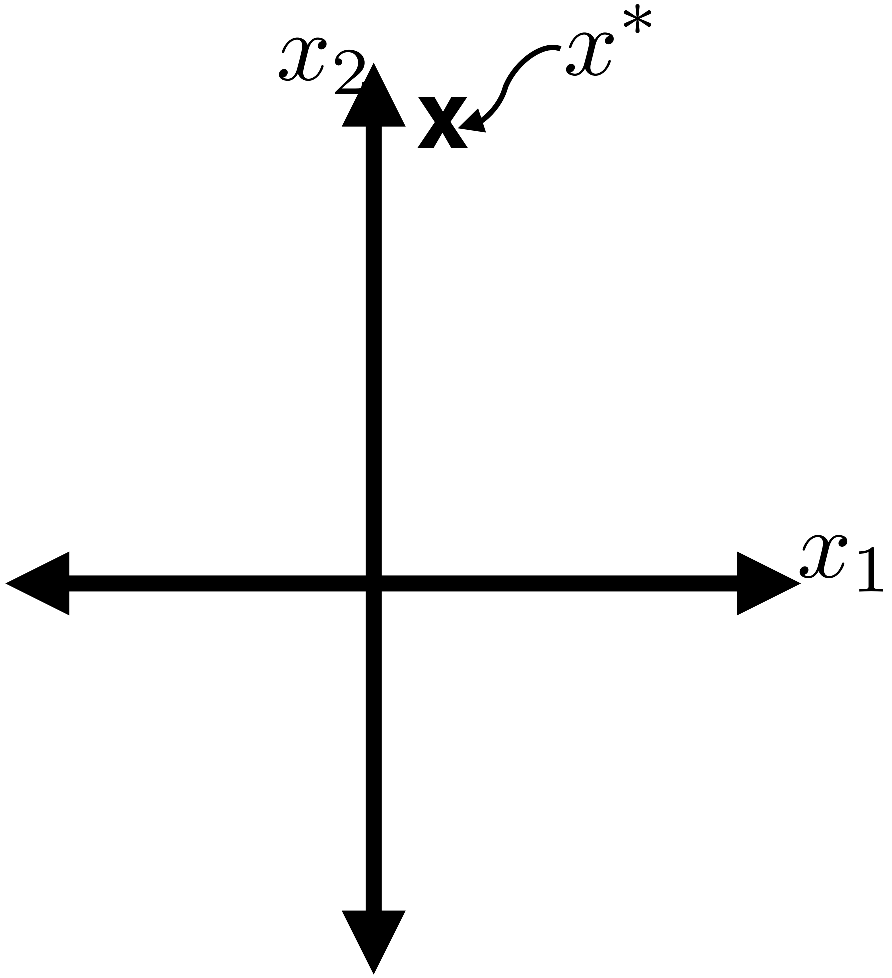


Let’s Talk About Classifier Quality

Math facts!

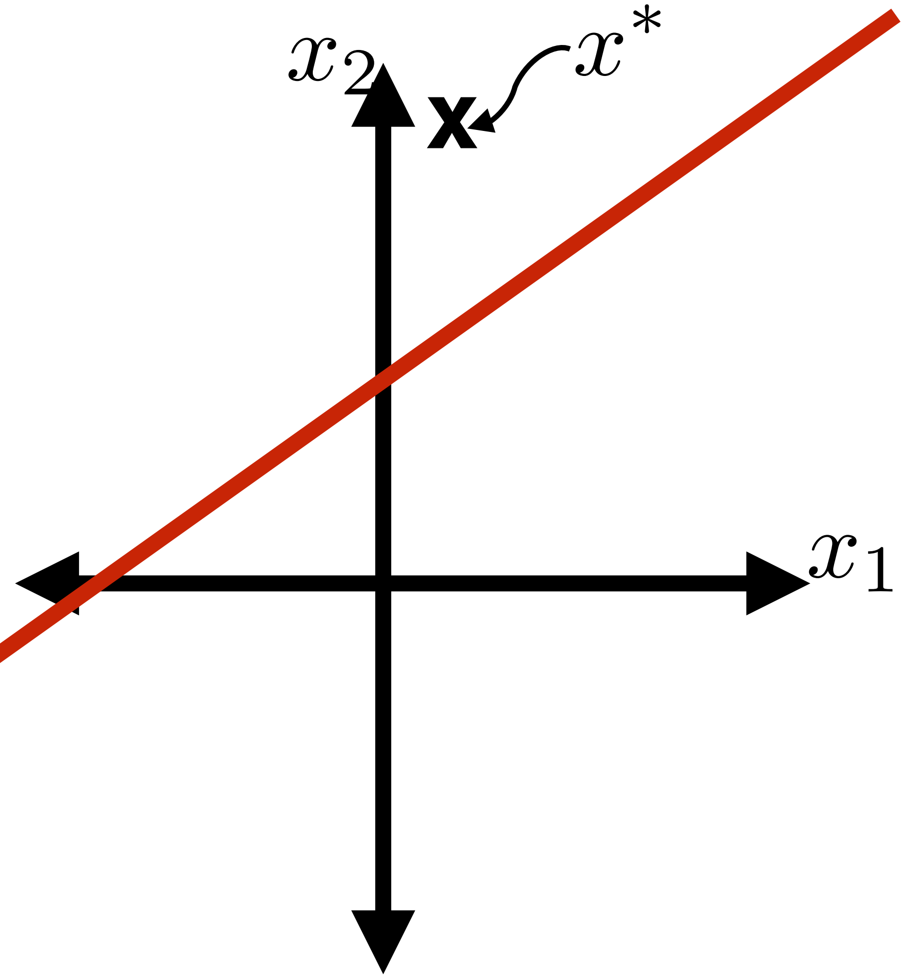
Let’s Talk About Classifier Quality

Math facts!



Let’s Talk About Classifier Quality

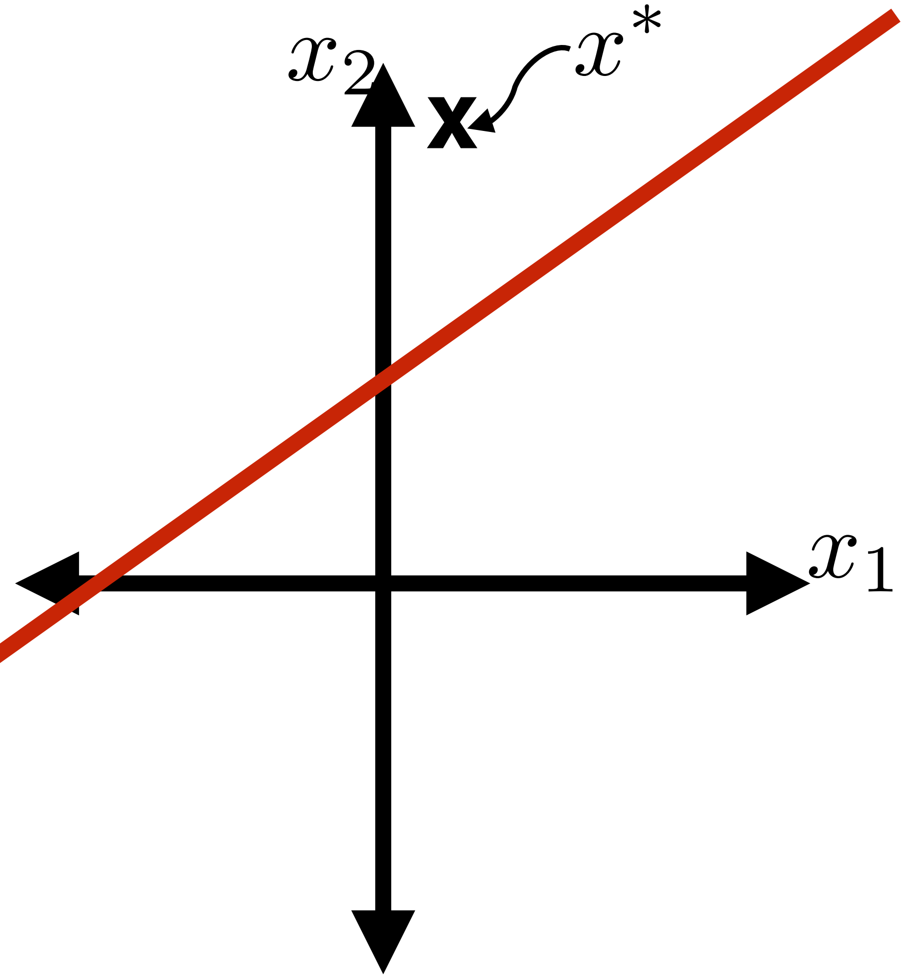
Math facts!



Let’s Talk About Classifier Quality

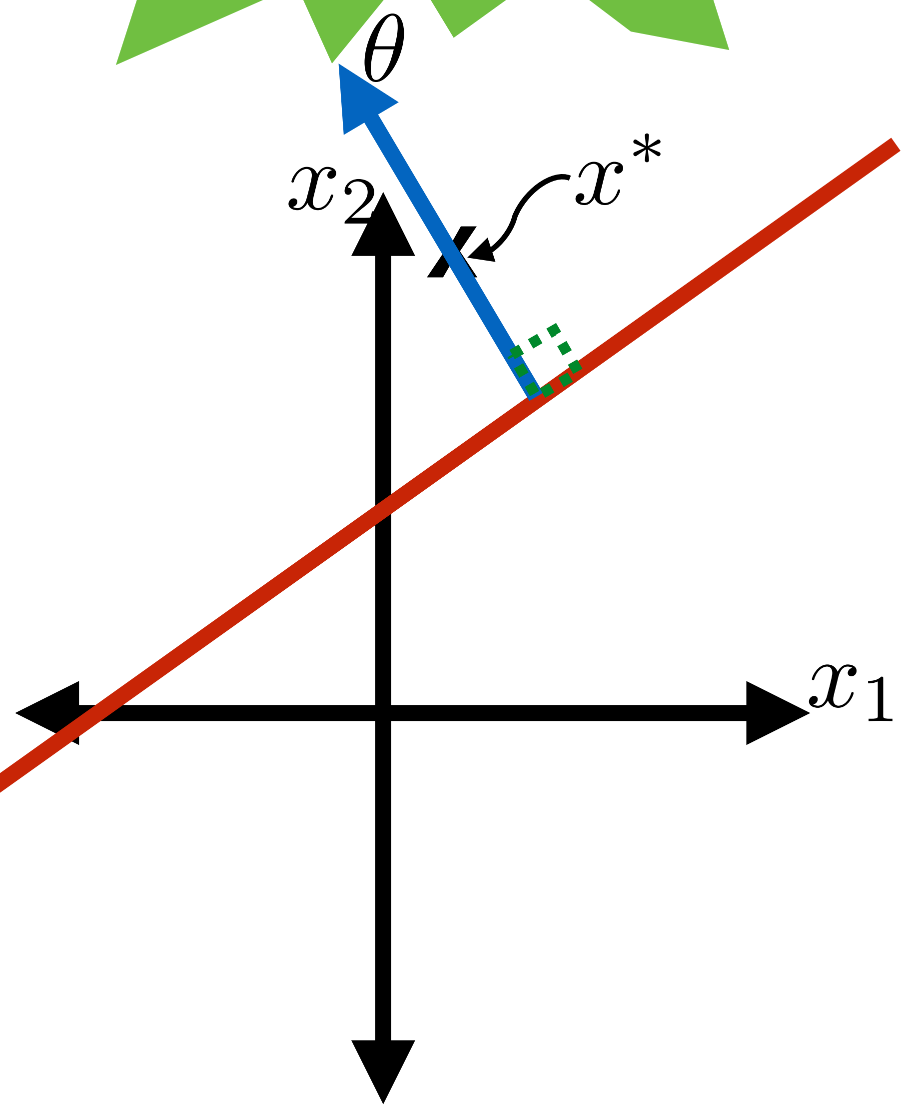
Math facts!

- The signed distance from a hyperplane defined by θ, θ_0 to a point x^* is:



Let’s Talk About Classifier Quality

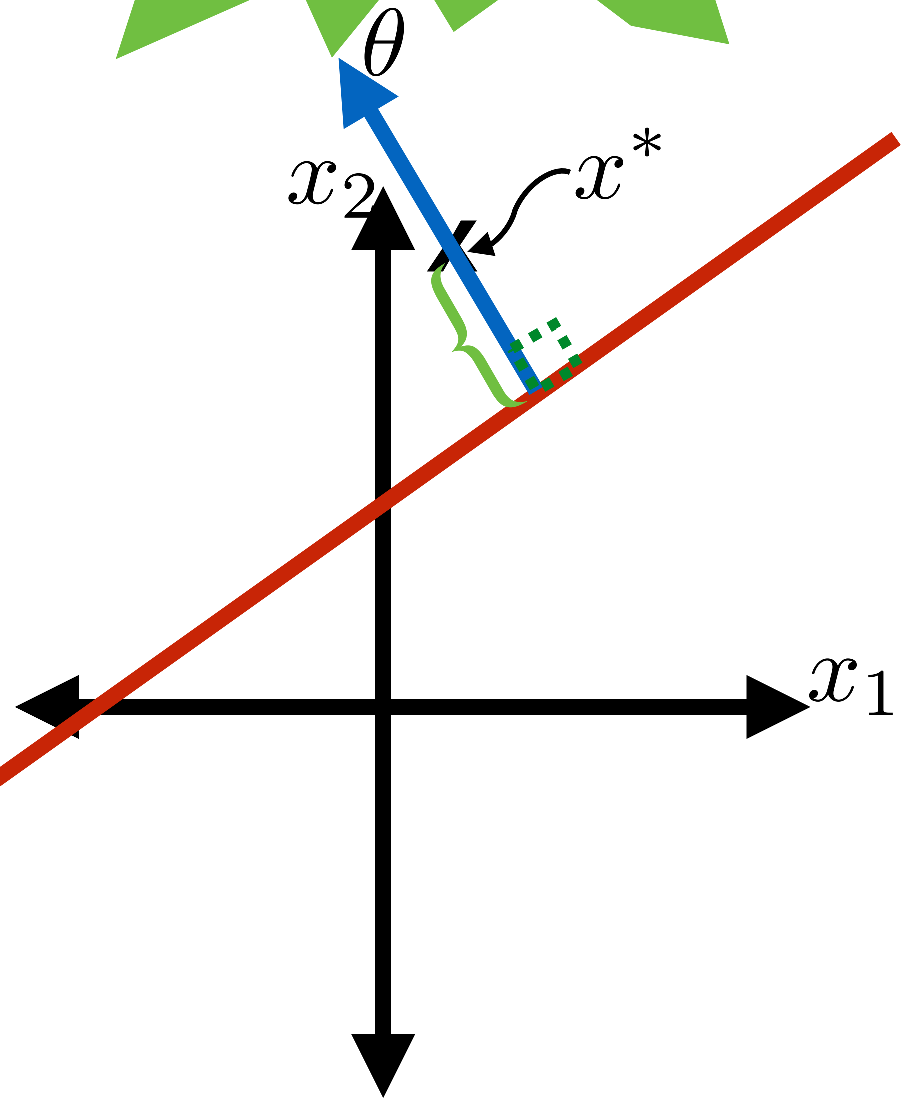
Math facts!



- The signed distance from a hyperplane defined by θ, θ_0 to a point x^* is:

Let’s Talk About Classifier Quality

Math facts!

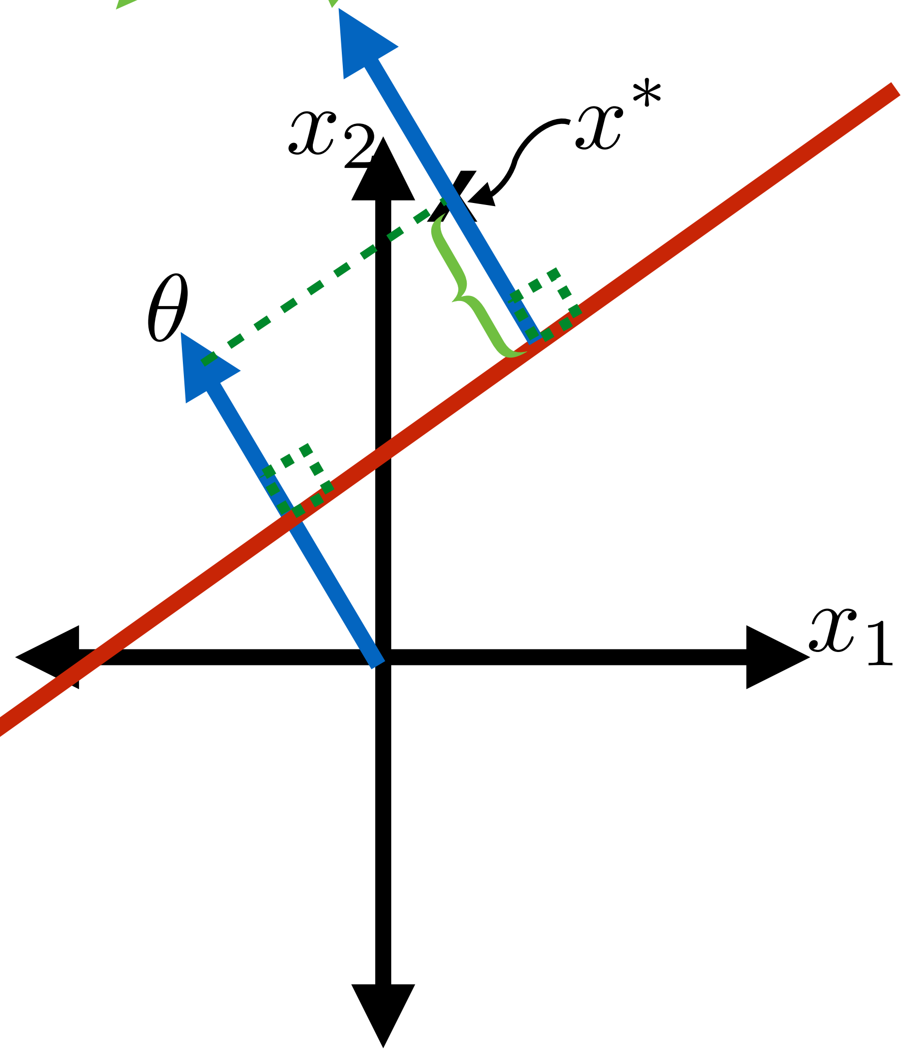


- The signed distance from a hyperplane defined by θ, θ_0 to a point x^* is:

Let's Talk About Classifier Quality

Math facts!

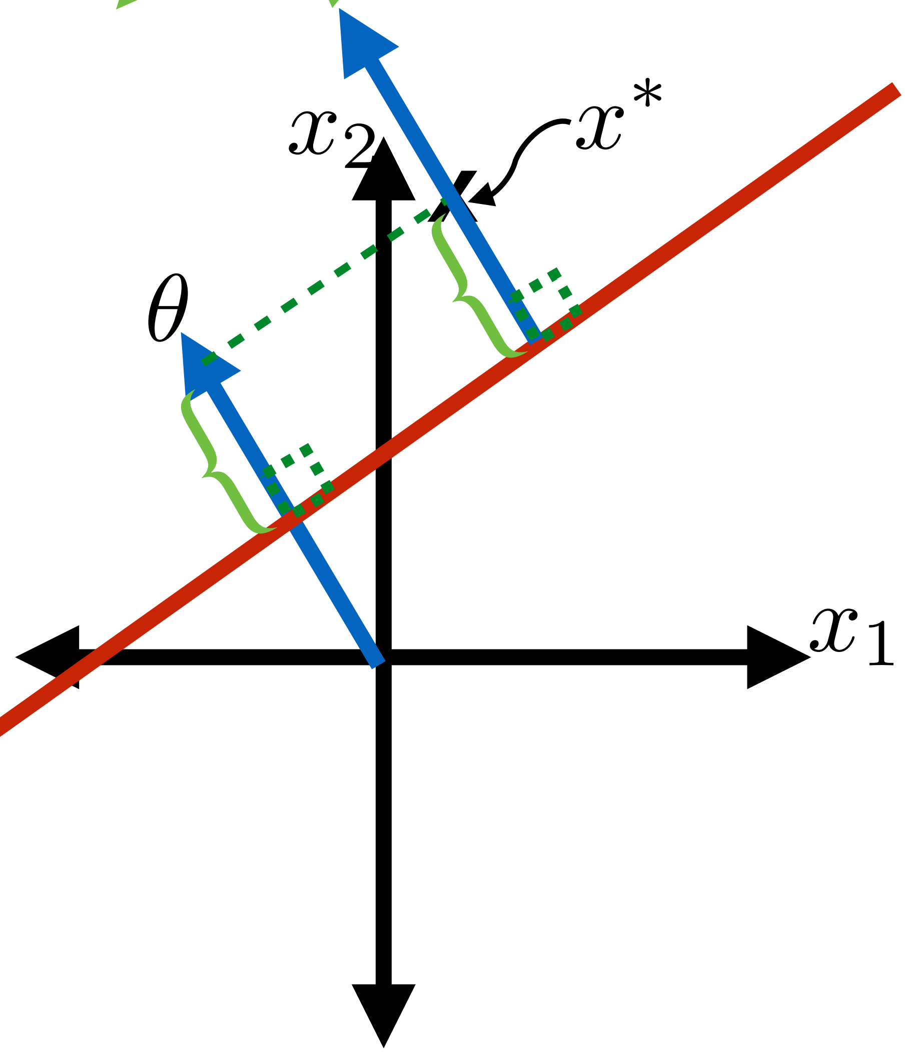
- The signed distance from a hyperplane defined by θ, θ_0 to a point x^* is:



Let’s Talk About Classifier Quality

Math facts!

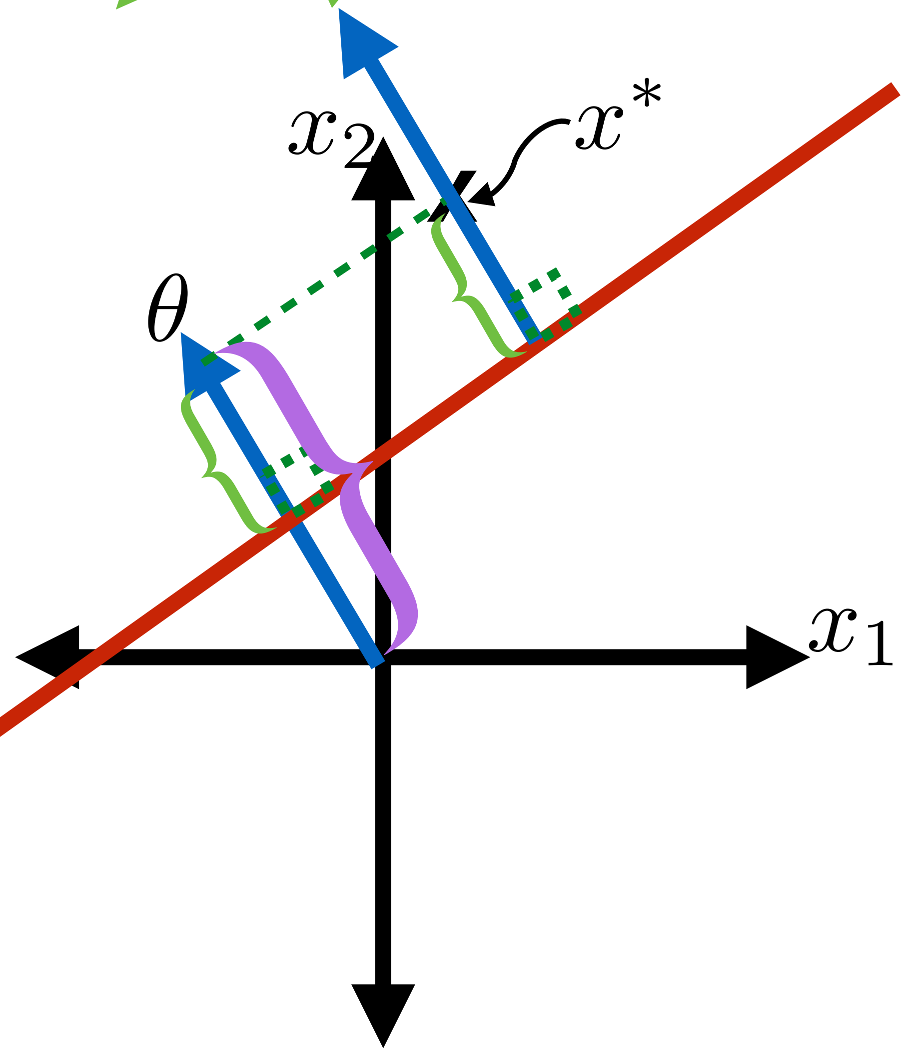
- The signed distance from a hyperplane defined by θ, θ_0 to a point x^* is:



Let's Talk About Classifier Quality

Math facts!

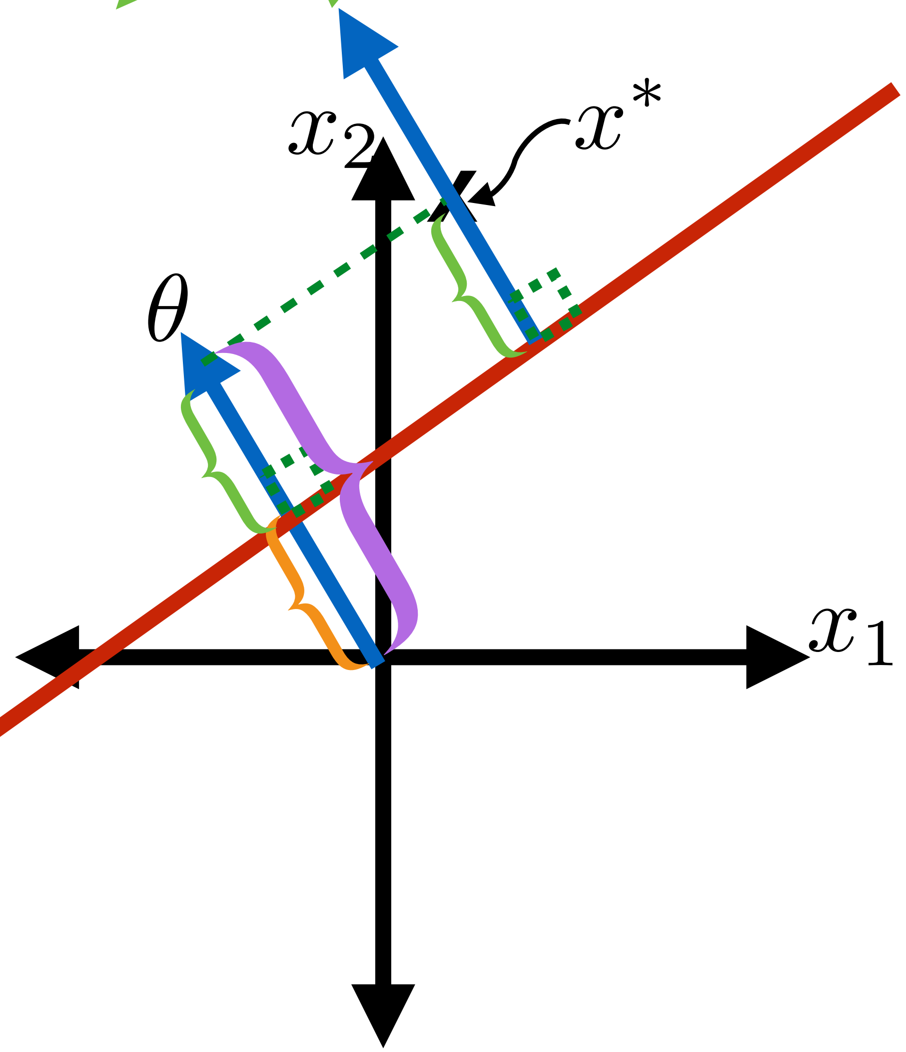
- The signed distance from a hyperplane defined by θ, θ_0 to a point x^* is:



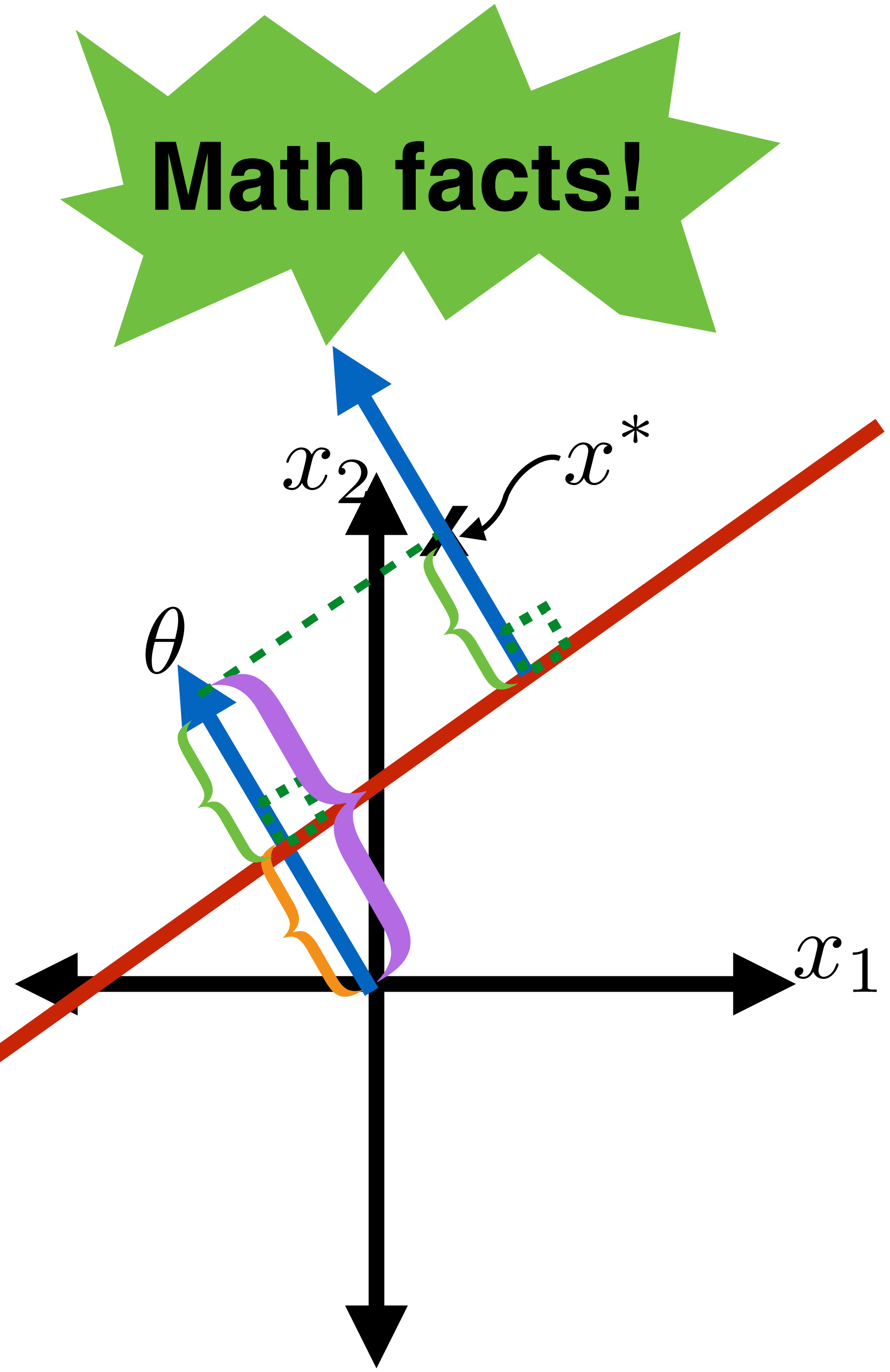
Let’s Talk About Classifier Quality

Math facts!

- The signed distance from a hyperplane defined by θ, θ_0 to a point x^* is:



Let’s Talk About Classifier Quality

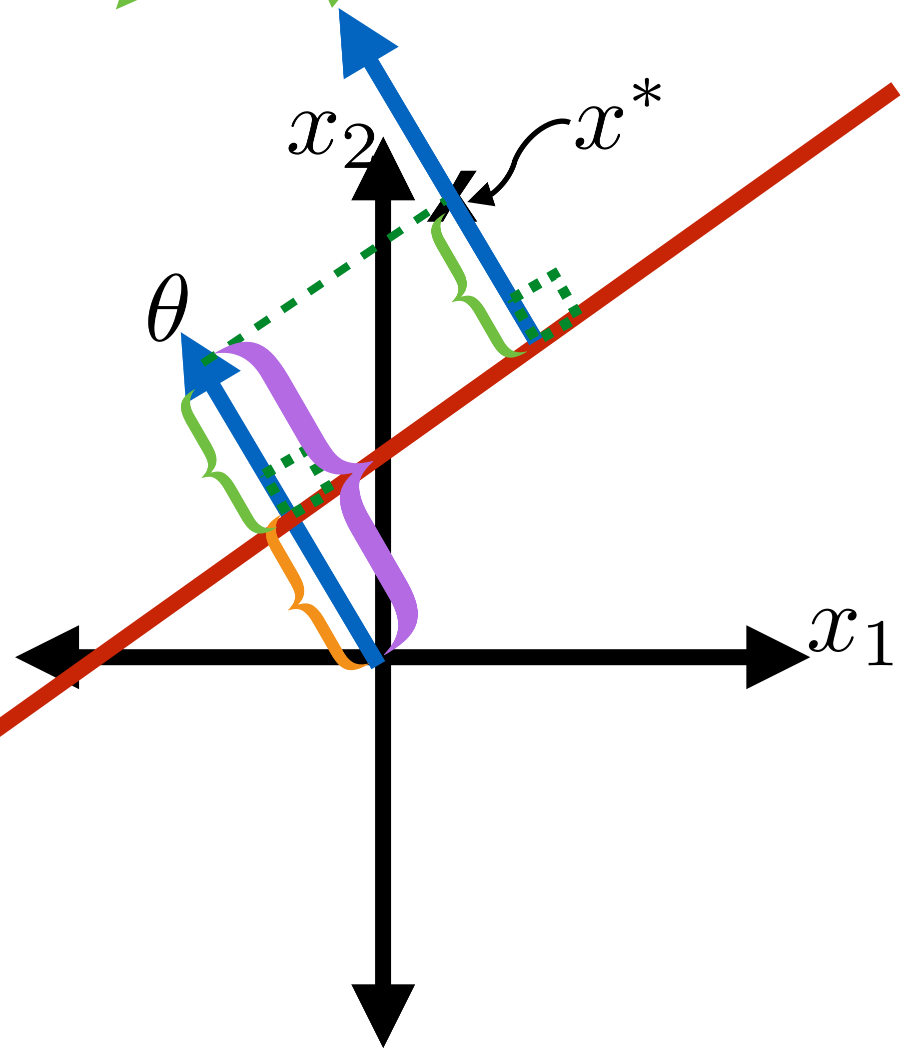


- The signed distance from a hyperplane defined by θ, θ_0 to a point x^* is:

  = projection of x^* on θ

Let’s Talk About Classifier Quality

Math facts!



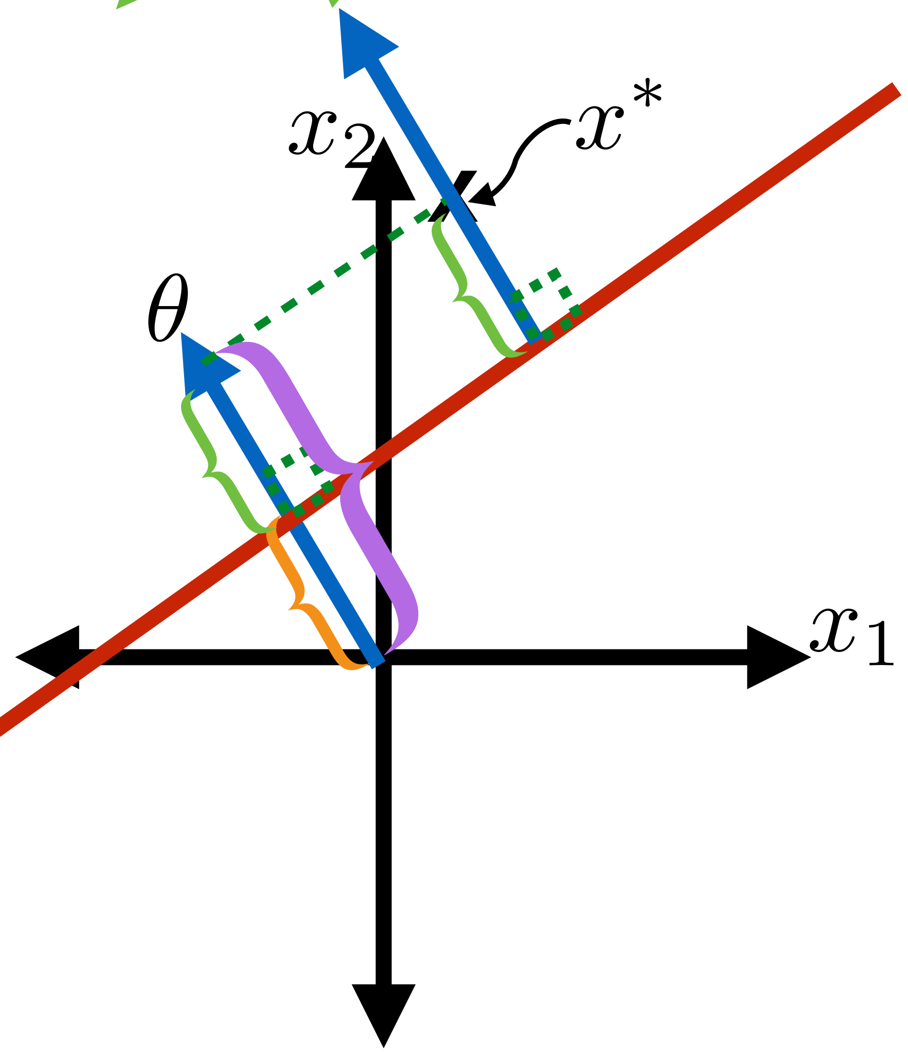
- The signed distance from a hyperplane defined by θ, θ_0 to a point x^* is:

= projection of x^* on θ

– signed distance of line to origin

Let’s Talk About Classifier Quality

Math facts!



- The signed distance from a hyperplane defined by θ, θ_0 to a point x^* is:

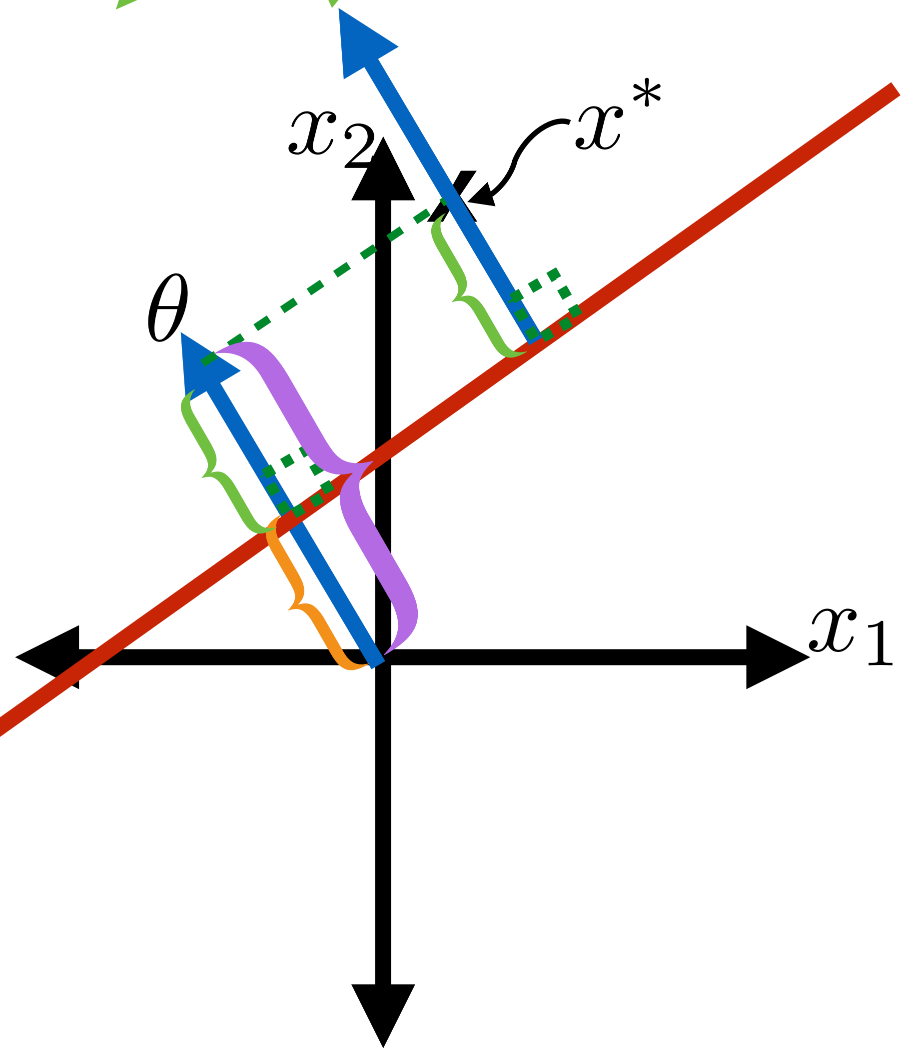
= projection of x^* on θ

– signed distance of line to origin

$$= \frac{\theta^\top x^*}{||\theta||}$$

Let’s Talk About Classifier Quality

Math facts!



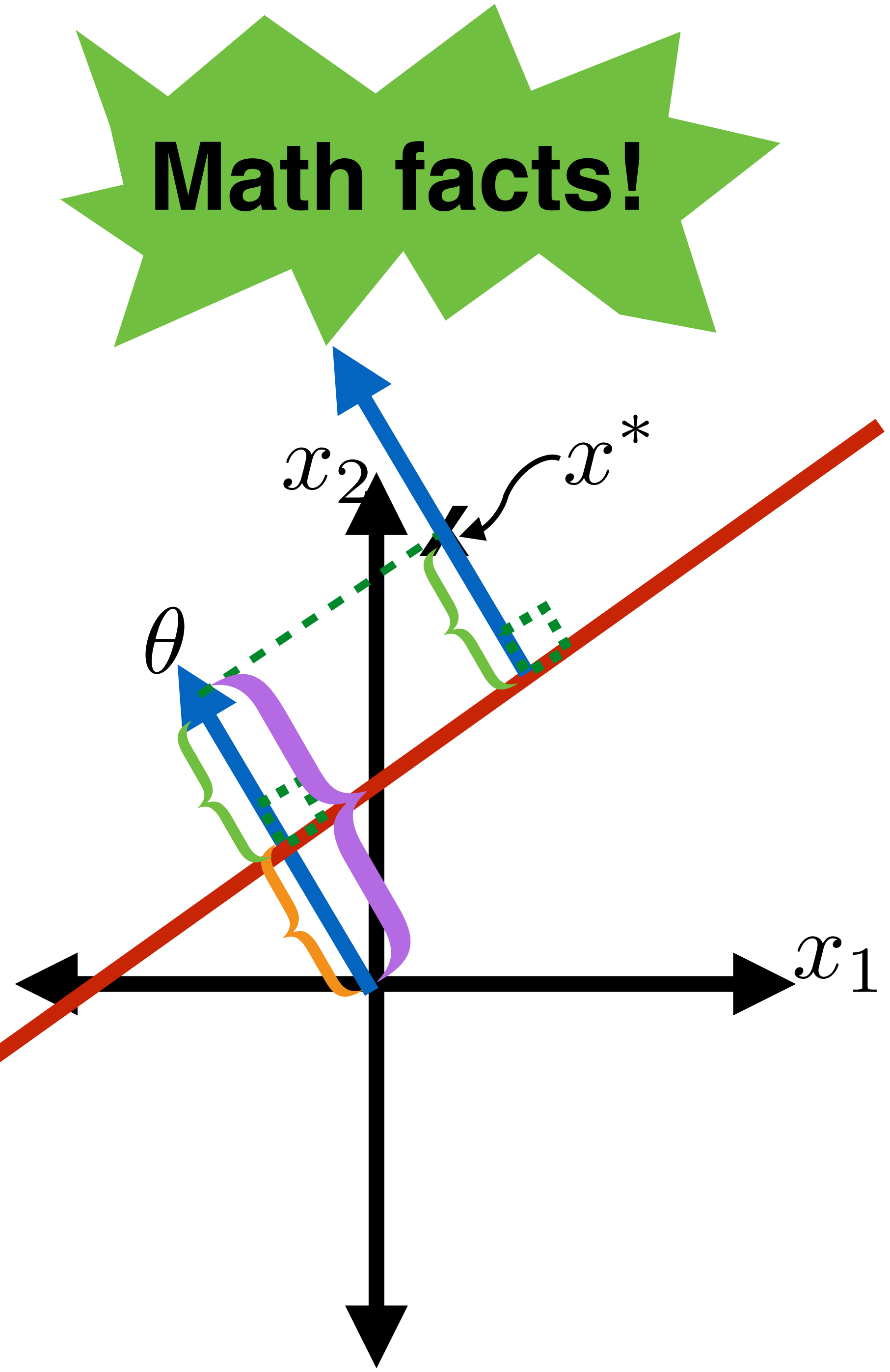
- The signed distance from a hyperplane defined by θ, θ_0 to a point x^* is:

= projection of x^* on θ

– signed distance of line to origin

$$= \frac{\theta^\top x^*}{\|\theta\|} - \frac{-\theta_0}{\|\theta\|}$$

Let’s Talk About Classifier Quality



- The signed distance from a hyperplane defined by θ, θ_0 to a point x^* is:

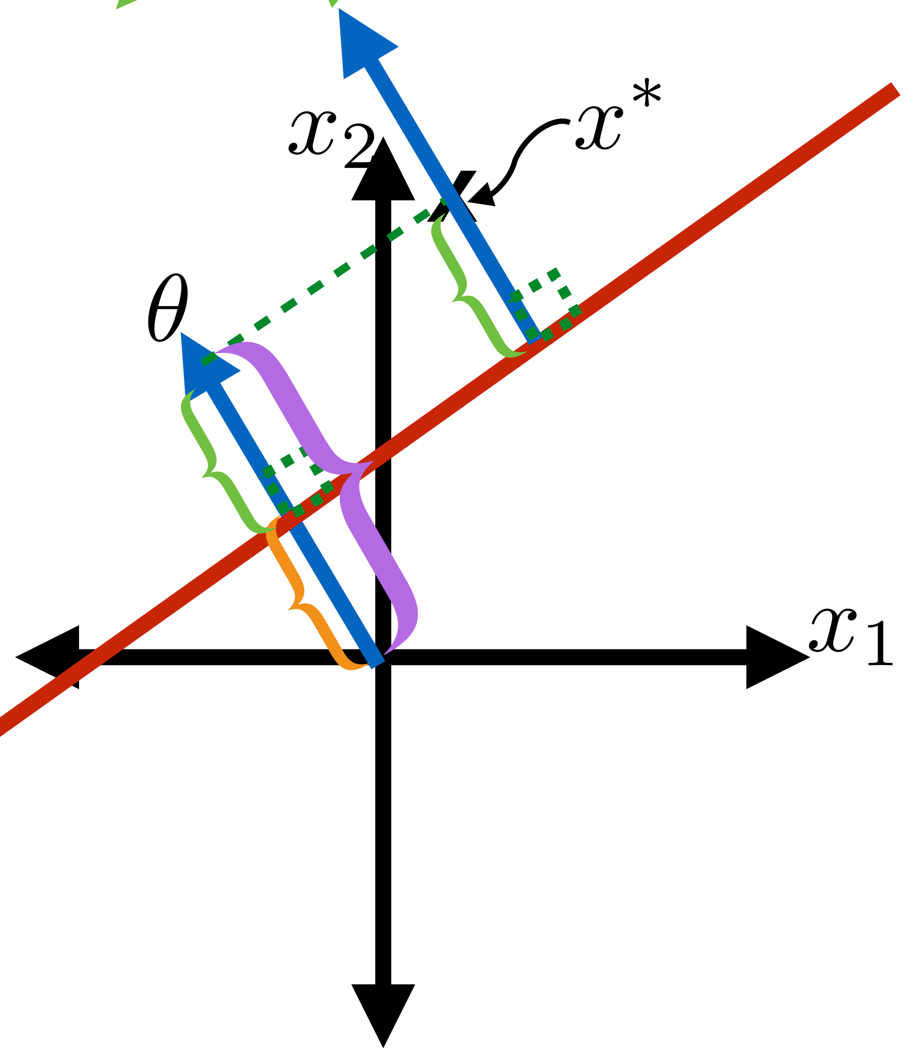
= projection of x^* on θ

– signed distance of line to origin

$$= \frac{\theta^\top x^*}{\|\theta\|} - \frac{-\theta_0}{\|\theta\|} = \frac{\theta^\top x^* + \theta_0}{\|\theta\|}$$

Let’s Talk About Classifier Quality

Math facts!



- The signed distance from a hyperplane defined by θ, θ_0 to a point x^* is:

  = projection of x^* on θ

  – signed distance of line to origin

$$= \frac{\theta^\top x^*}{||\theta||} - \frac{-\theta_0}{||\theta||} = \frac{\theta^\top x^* + \theta_0}{||\theta||}$$

- *Definition*: The **margin of the labelled point** (x^*, y^*) with respect to the hyperplane defined by θ, θ_0 is:

Let's Talk About Classifier Quality

Math facts!

- The signed distance from a hyperplane defined by θ, θ_0 to a point x^* is:

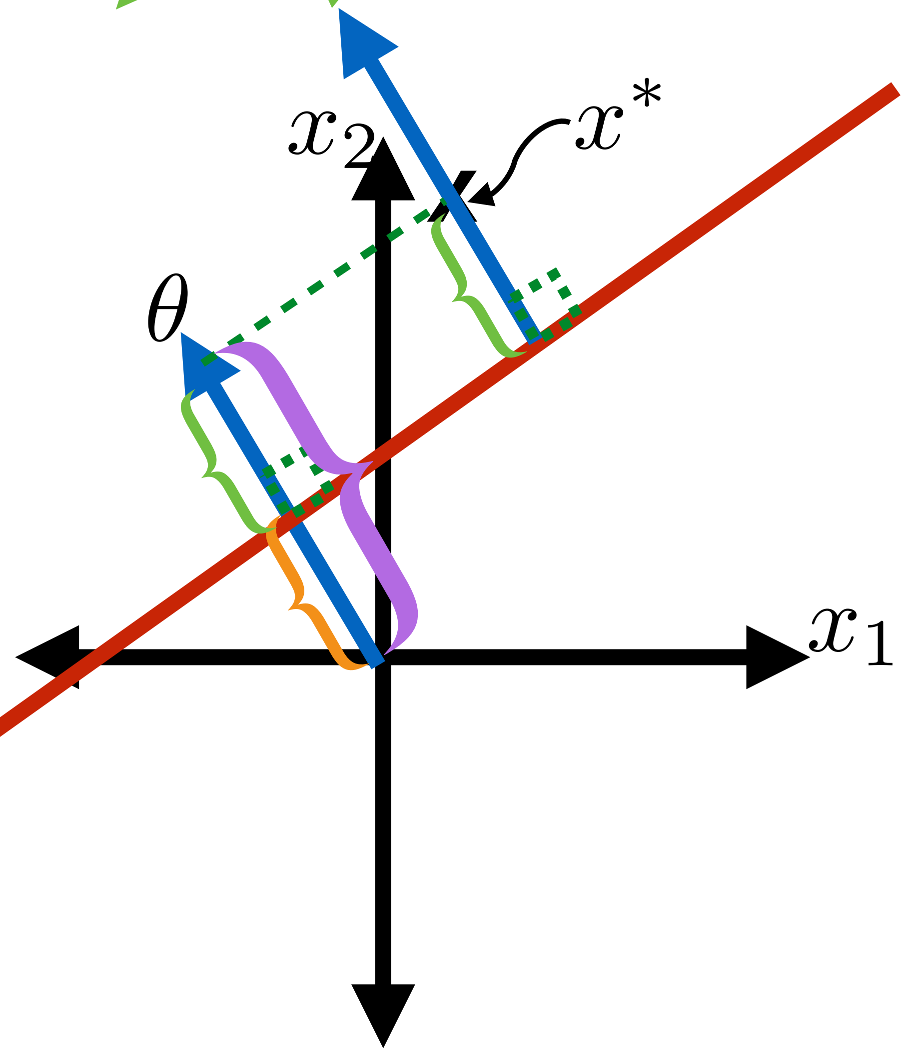
  = projection of x^* on θ

  – signed distance of line to origin

$$= \frac{\theta^\top x^*}{\|\theta\|} - \frac{-\theta_0}{\|\theta\|} = \frac{\theta^\top x^* + \theta_0}{\|\theta\|}$$

- *Definition*: The **margin of the labelled point** (x^*, y^*) with respect to the hyperplane defined by θ, θ_0 is:

$$y^* \left(\frac{\theta^\top x^* + \theta_0}{\|\theta\|} \right)$$



Let's Talk About Classifier Quality

Math facts!

- The signed distance from a hyperplane defined by θ, θ_0 to a point x^* is:

= projection of x^* on θ

– signed distance of line to origin

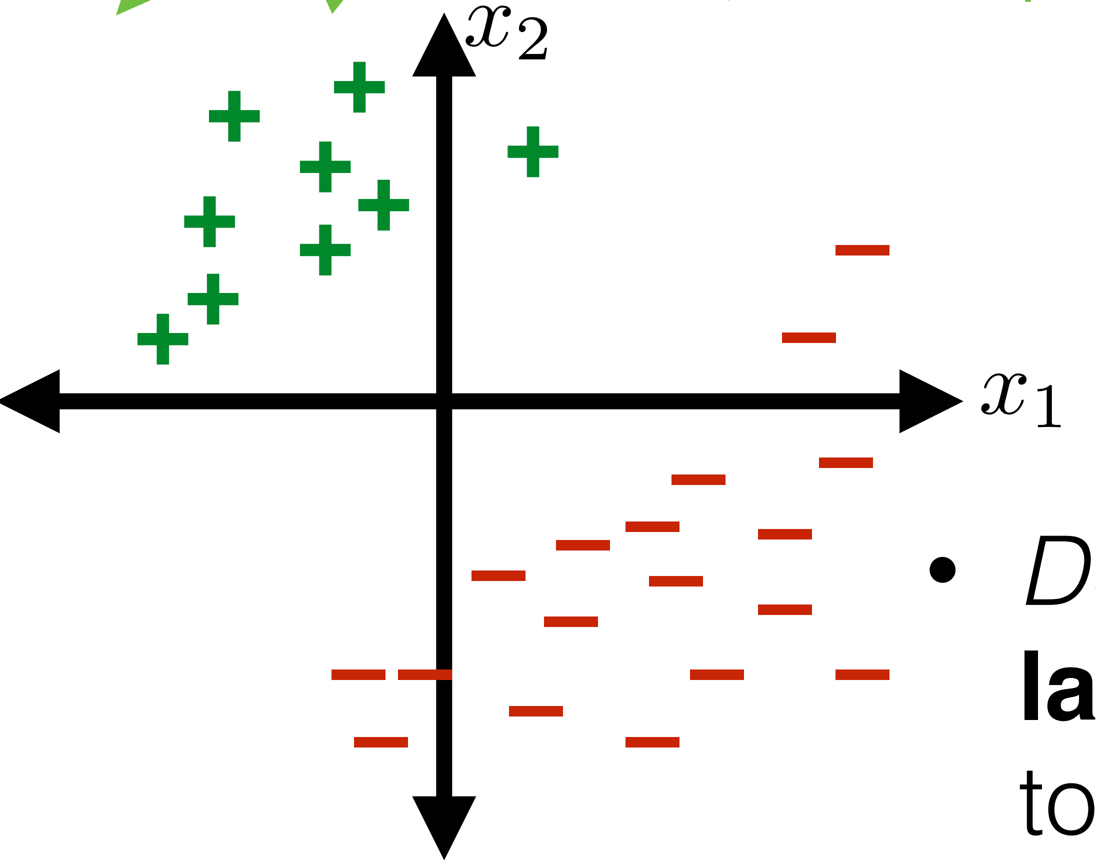
$$= \frac{\theta^\top x^*}{\|\theta\|} - \frac{-\theta_0}{\|\theta\|} = \frac{\theta^\top x^* + \theta_0}{\|\theta\|}$$

- *Definition*: The **margin of the labelled point** (x^*, y^*) with respect to the hyperplane defined by θ, θ_0 is:

$$y^* \left(\frac{\theta^\top x^* + \theta_0}{\|\theta\|} \right)$$

Let’s Talk About Classifier Quality

Math facts!



- The signed distance from a hyperplane defined by θ, θ_0 to a point x^* is:

= projection of x^* on θ
– signed distance of line to origin

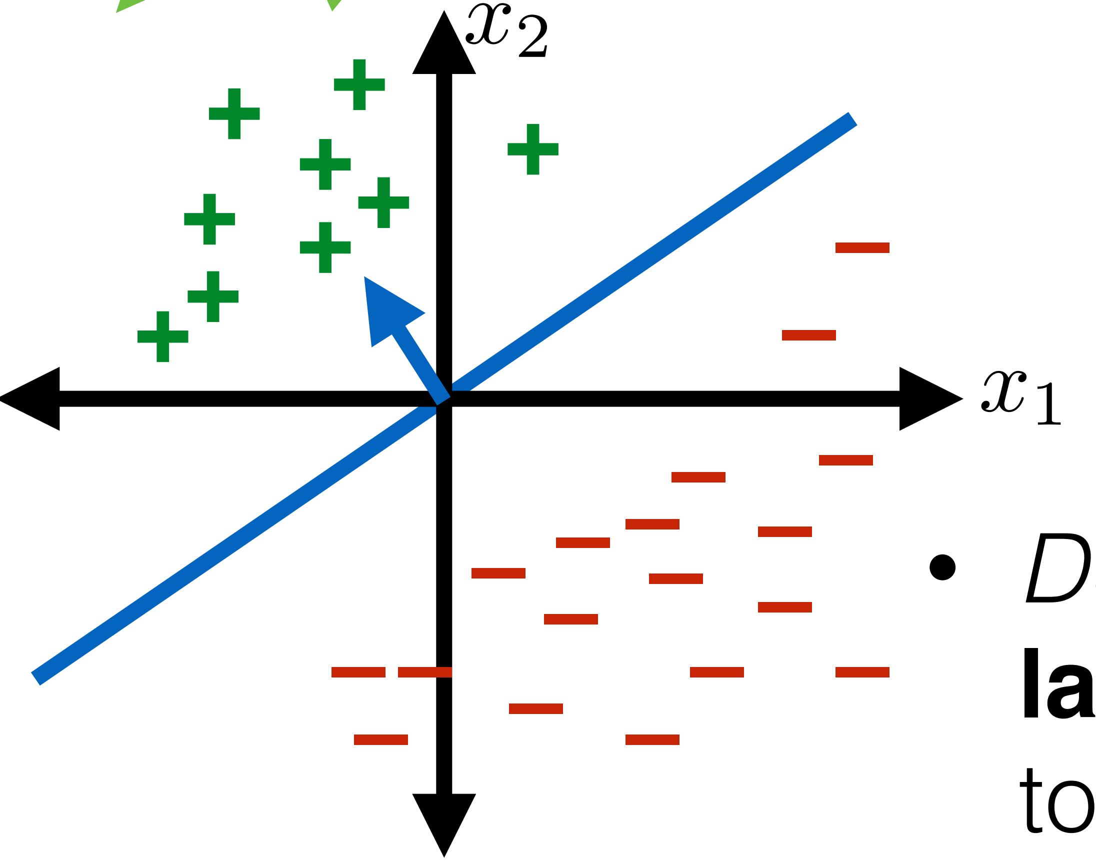
$$= \frac{\theta^\top x^*}{\|\theta\|} - \frac{-\theta_0}{\|\theta\|} = \frac{\theta^\top x^* + \theta_0}{\|\theta\|}$$

- *Definition*: The **margin of the labelled point** (x^*, y^*) with respect to the hyperplane defined by θ, θ_0 is:

$$y^* \left(\frac{\theta^\top x^* + \theta_0}{\|\theta\|} \right)$$

Let's Talk About Classifier Quality

Math facts!



- The signed distance from a hyperplane defined by θ, θ_0 to a point x^* is:

= projection of x^* on θ

– signed distance of line to origin

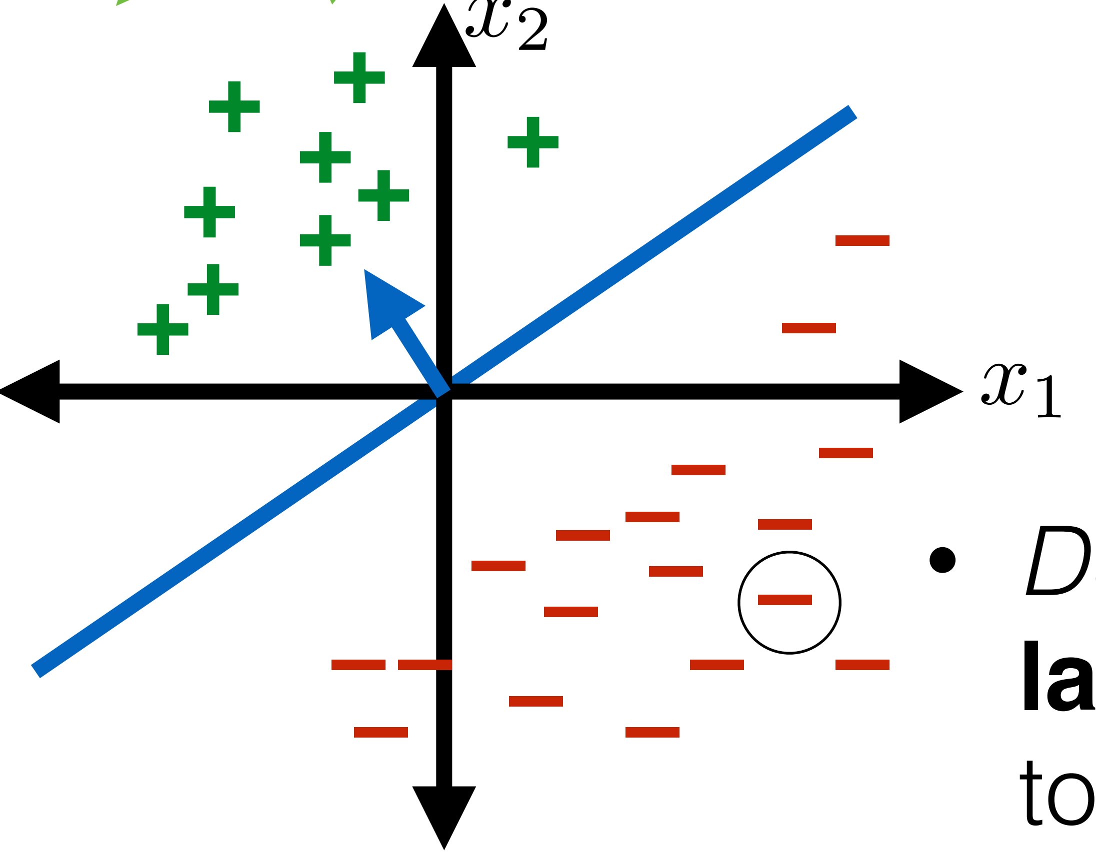
$$= \frac{\theta^\top x^*}{\|\theta\|} - \frac{-\theta_0}{\|\theta\|} = \frac{\theta^\top x^* + \theta_0}{\|\theta\|}$$

- *Definition*: The **margin of the labelled point** (x^*, y^*) with respect to the hyperplane defined by θ, θ_0 is:

$$y^* \left(\frac{\theta^\top x^* + \theta_0}{\|\theta\|} \right)$$

Let’s Talk About Classifier Quality

Math facts!



- The signed distance from a hyperplane defined by θ, θ_0 to a point x^* is:

  = projection of x^* on θ

  – signed distance of line to origin

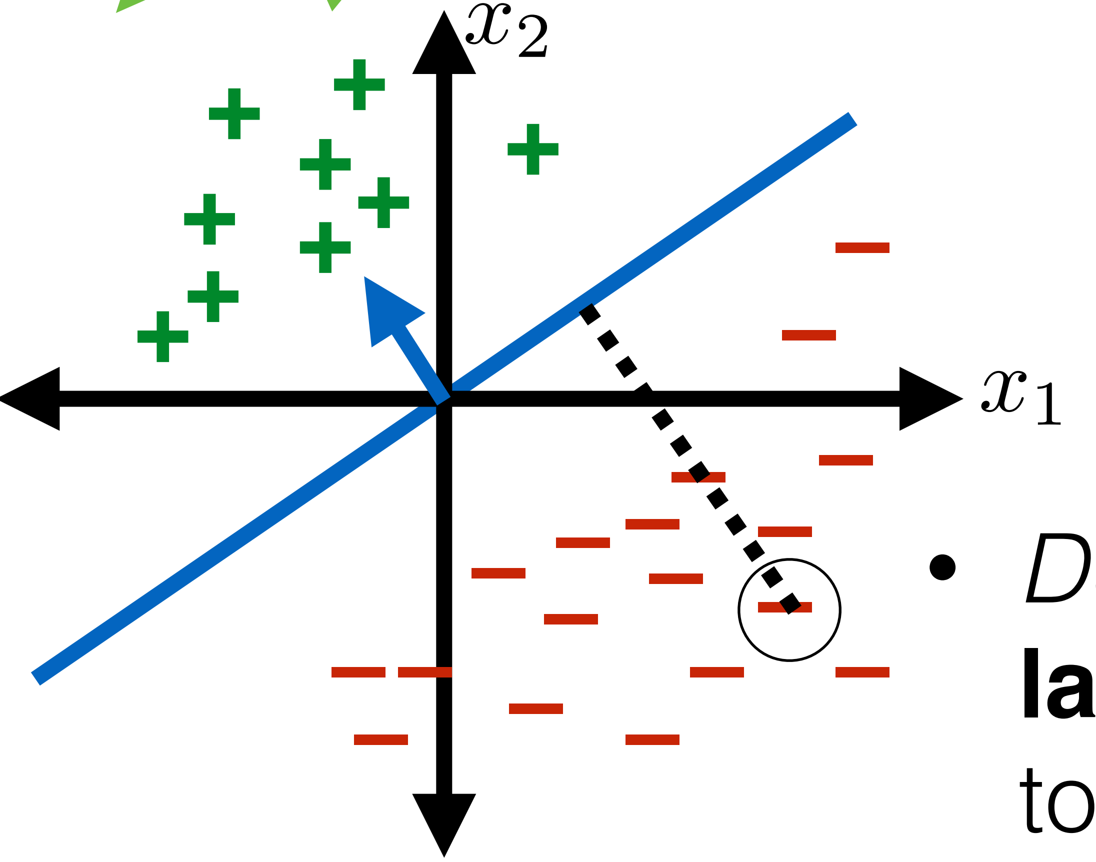
$$= \frac{\theta^\top x^*}{\|\theta\|} - \frac{-\theta_0}{\|\theta\|} = \frac{\theta^\top x^* + \theta_0}{\|\theta\|}$$

- *Definition*: The **margin of the labelled point** (x^*, y^*) with respect to the hyperplane defined by θ, θ_0 is:

$$y^* \left(\frac{\theta^\top x^* + \theta_0}{\|\theta\|} \right)$$

Let’s Talk About Classifier Quality

Math facts!



- The signed distance from a hyperplane defined by θ, θ_0 to a point x^* is:

  = projection of x^* on θ

  – signed distance of line to origin

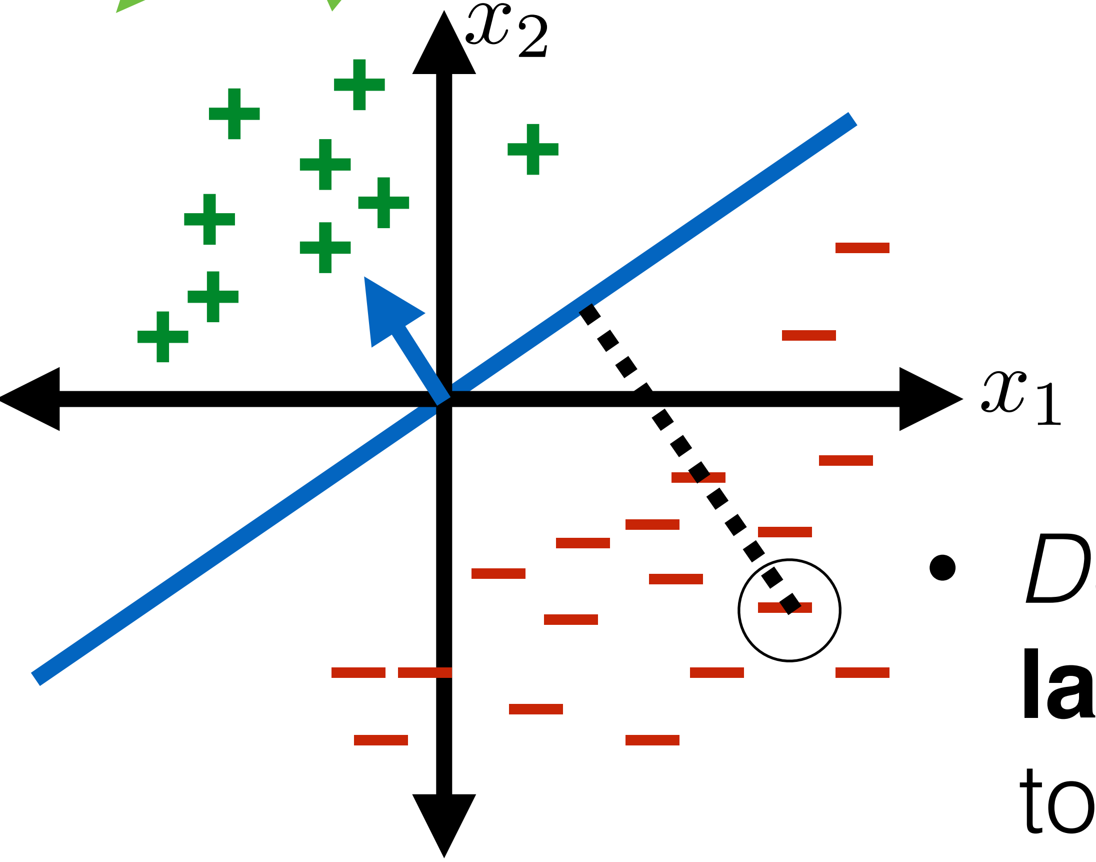
$$= \frac{\theta^\top x^*}{\|\theta\|} - \frac{-\theta_0}{\|\theta\|} = \frac{\theta^\top x^* + \theta_0}{\|\theta\|}$$

- *Definition*: The **margin of the labelled point** (x^*, y^*) with respect to the hyperplane defined by θ, θ_0 is:

$$y^* \left(\frac{\theta^\top x^* + \theta_0}{\|\theta\|} \right)$$

Let's Talk About Classifier Quality

Math facts!



- The signed distance from a hyperplane defined by θ, θ_0 to a point x^* is:

= projection of x^* on θ

– signed distance of line to origin

$$= \frac{\theta^\top x^*}{\|\theta\|} - \frac{-\theta_0}{\|\theta\|} = \frac{\theta^\top x^* + \theta_0}{\|\theta\|}$$

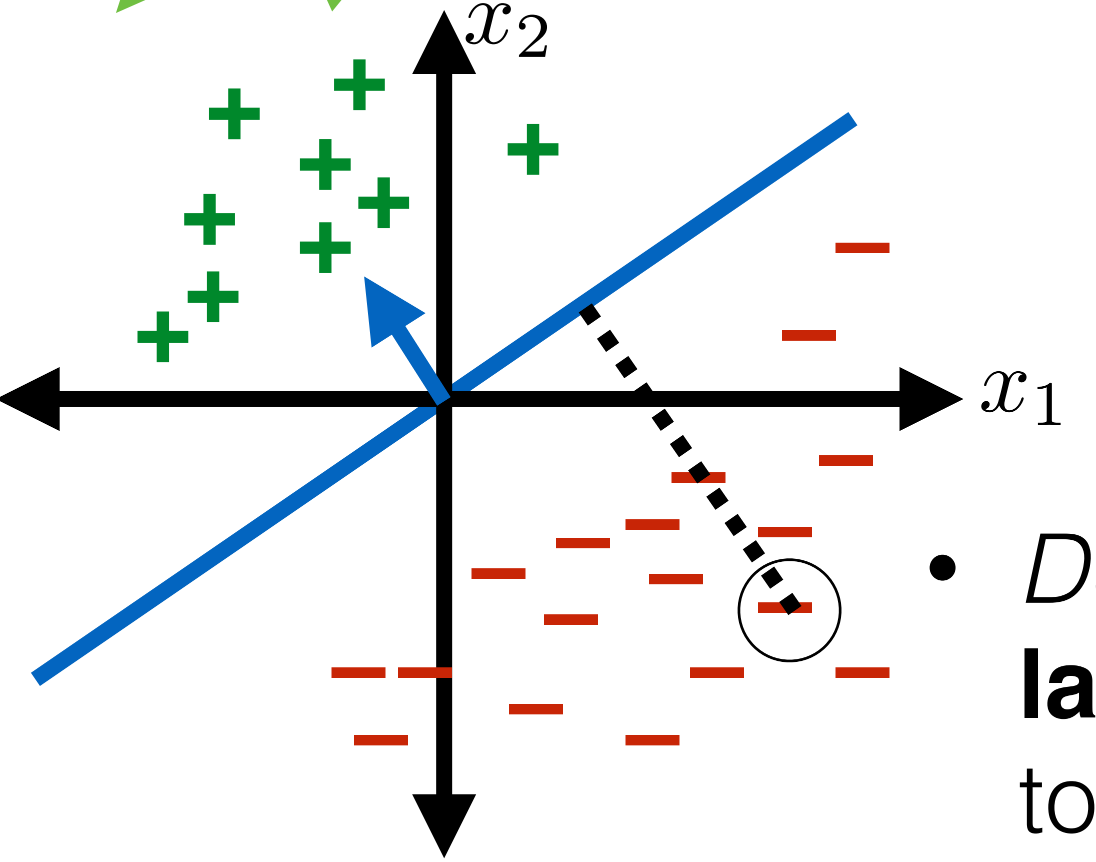
- *Definition*: The **margin of the labelled point** (x^*, y^*) with respect to the hyperplane defined by θ, θ_0 is:

$$y^* \left(\frac{\theta^\top x^* + \theta_0}{\|\theta\|} \right)$$

- *Definition*: The **margin of the training set** $\mathcal{D}_n$ with respect to the hyperplane defined by θ, θ_0 is:

Let's Talk About Classifier Quality

Math facts!



- The signed distance from a hyperplane defined by θ, θ_0 to a point x^* is:

  = projection of x^* on θ

  – signed distance of line to origin

$$= \frac{\theta^\top x^*}{\|\theta\|} - \frac{-\theta_0}{\|\theta\|} \quad = \frac{\theta^\top x^* + \theta_0}{\|\theta\|}$$

- *Definition*: The **margin of the labelled point** (x^*, y^*) with respect to the hyperplane defined by θ, θ_0 is:

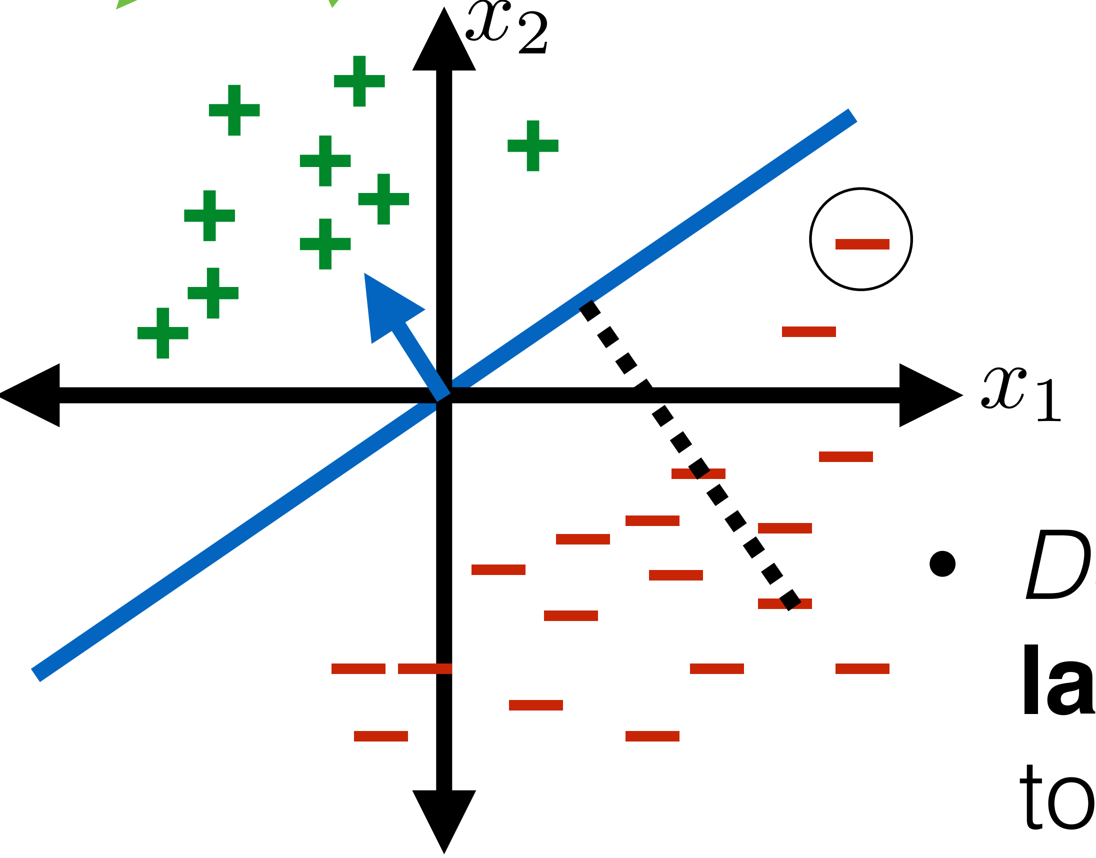
$$y^* \left(\frac{\theta^\top x^* + \theta_0}{\|\theta\|} \right)$$

- *Definition*: The **margin of the training set** $\mathcal{D}_n$ with respect to the hyperplane defined by θ, θ_0 is:

$$\min_{i \in \{1,\ldots,n\}} y^{(i)} \left(\frac{\theta^\top x^{(i)} + \theta_0}{\|\theta\|} \right)$$

Let's Talk About Classifier Quality

Math facts!



- The signed distance from a hyperplane defined by θ, θ_0 to a point x^* is:

  = projection of x^* on θ

  – signed distance of line to origin

$$= \frac{\theta^\top x^*}{\|\theta\|} - \frac{-\theta_0}{\|\theta\|} = \frac{\theta^\top x^* + \theta_0}{\|\theta\|}$$

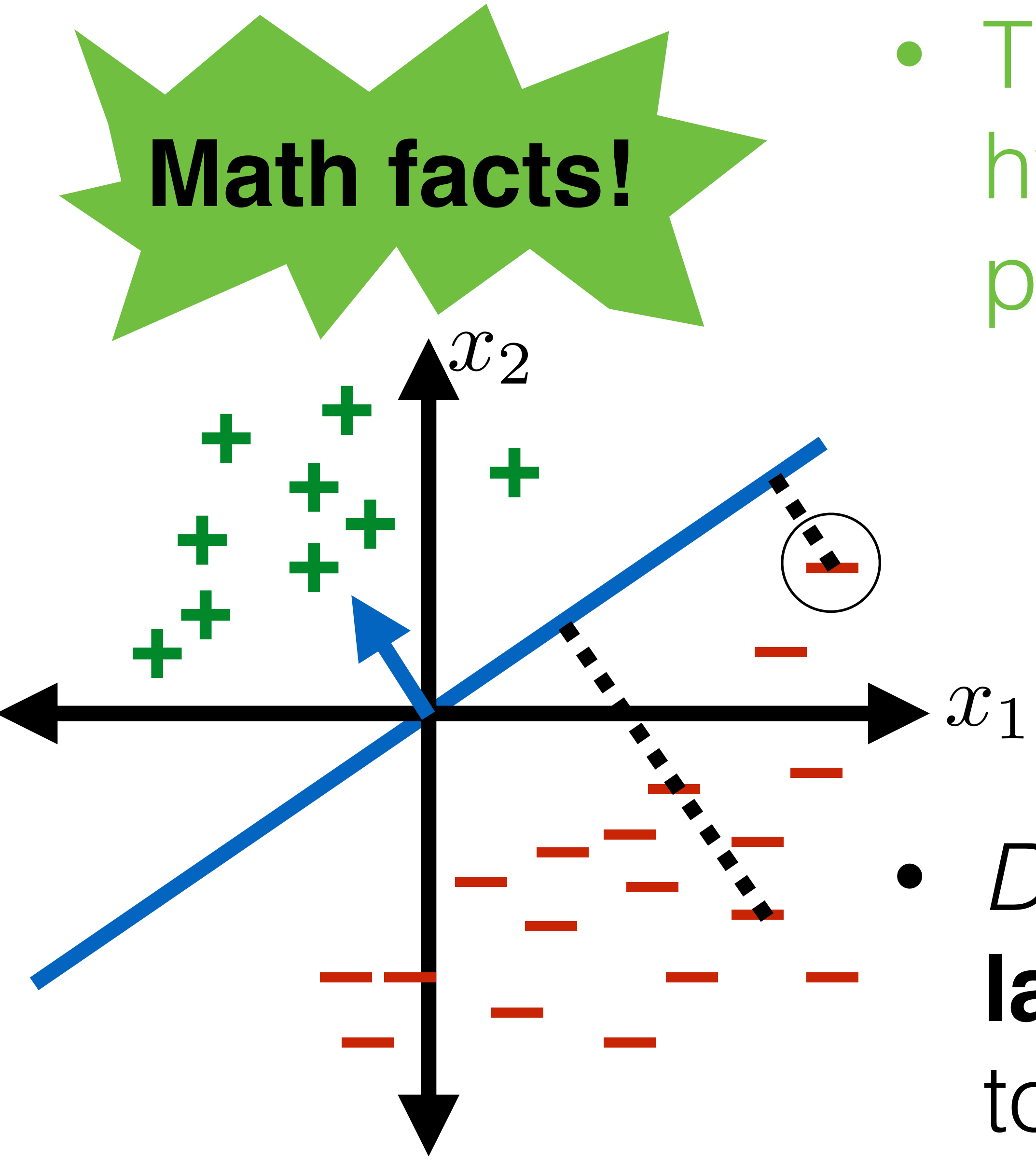
- *Definition*: The **margin of the labelled point** (x^*, y^*) with respect to the hyperplane defined by θ, θ_0 is:

$$y^* \left(\frac{\theta^\top x^* + \theta_0}{\|\theta\|} \right)$$

- *Definition*: The **margin of the training set** $\mathcal{D}_n$ with respect to the hyperplane defined by θ, θ_0 is:

$$\min_{i \in \{1,\dots,n\}} y^{(i)} \left(\frac{\theta^\top x^{(i)} + \theta_0}{\|\theta\|} \right)$$

Let's Talk About Classifier Quality



- The signed distance from a hyperplane defined by θ, θ_0 to a point x^* is:

= projection of x^* on θ

– signed distance of line to origin

$$= \frac{\theta^\top x^*}{\|\theta\|} - \frac{-\theta_0}{\|\theta\|} = \frac{\theta^\top x^* + \theta_0}{\|\theta\|}$$

- *Definition*: The **margin of the labelled point** (x^*, y^*) with respect to the hyperplane defined by θ, θ_0 is:

$$y^* \left(\frac{\theta^\top x^* + \theta_0}{\|\theta\|} \right)$$

- *Definition*: The **margin of the training set** $\mathcal{D}_n$ with respect to the hyperplane defined by θ, θ_0 is:

$$\min_{i \in \{1,\ldots,n\}} y^{(i)} \left(\frac{\theta^\top x^{(i)} + \theta_0}{\|\theta\|} \right)$$

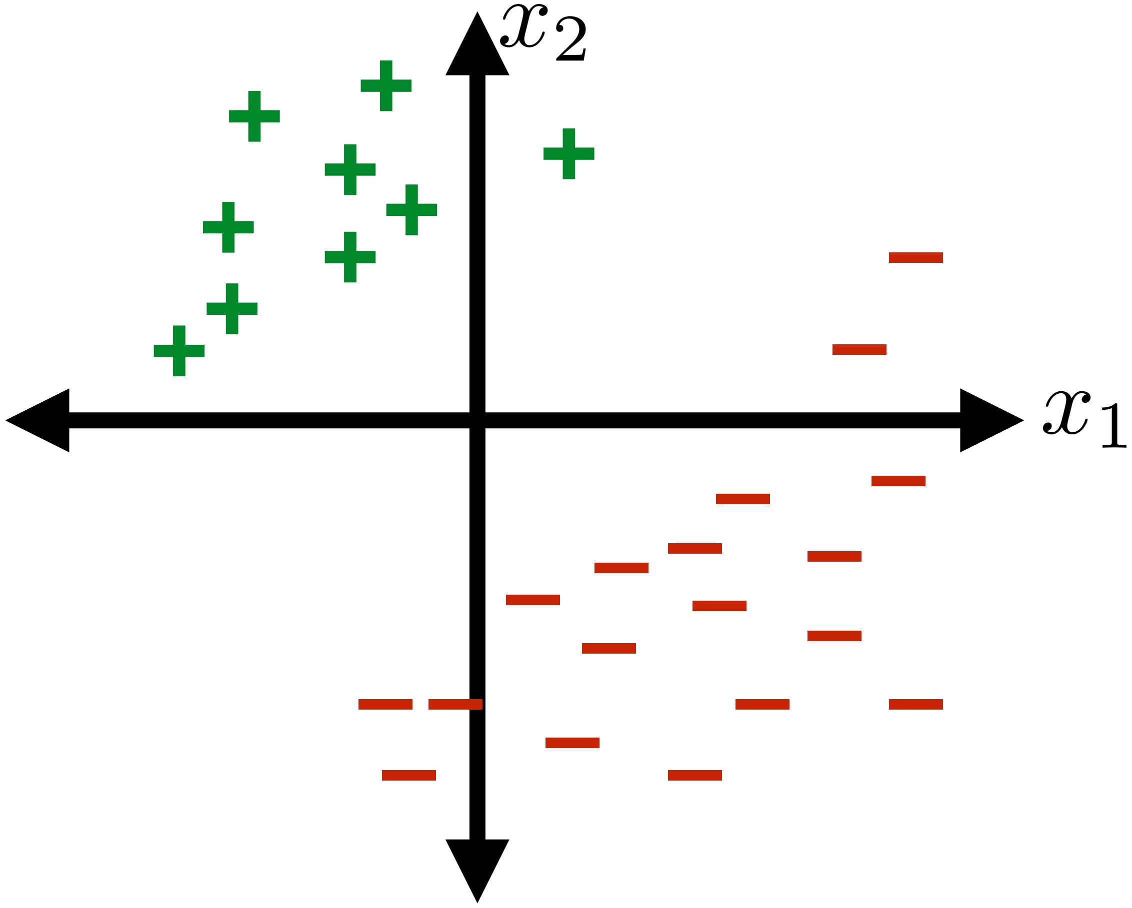
Theorem: Perceptron Performance

Theorem: Perceptron Performance

- **Assumptions**:

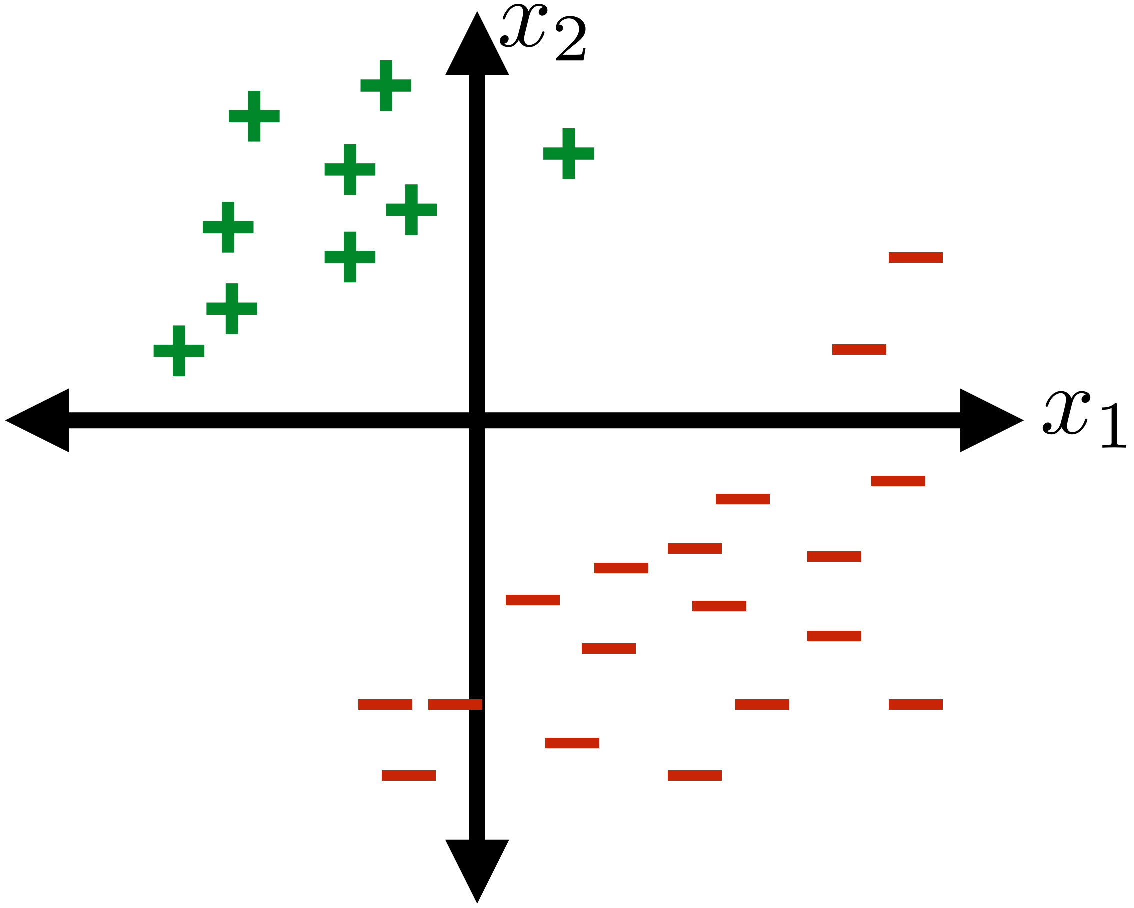
Theorem: Perceptron Performance

- **Assumptions**:



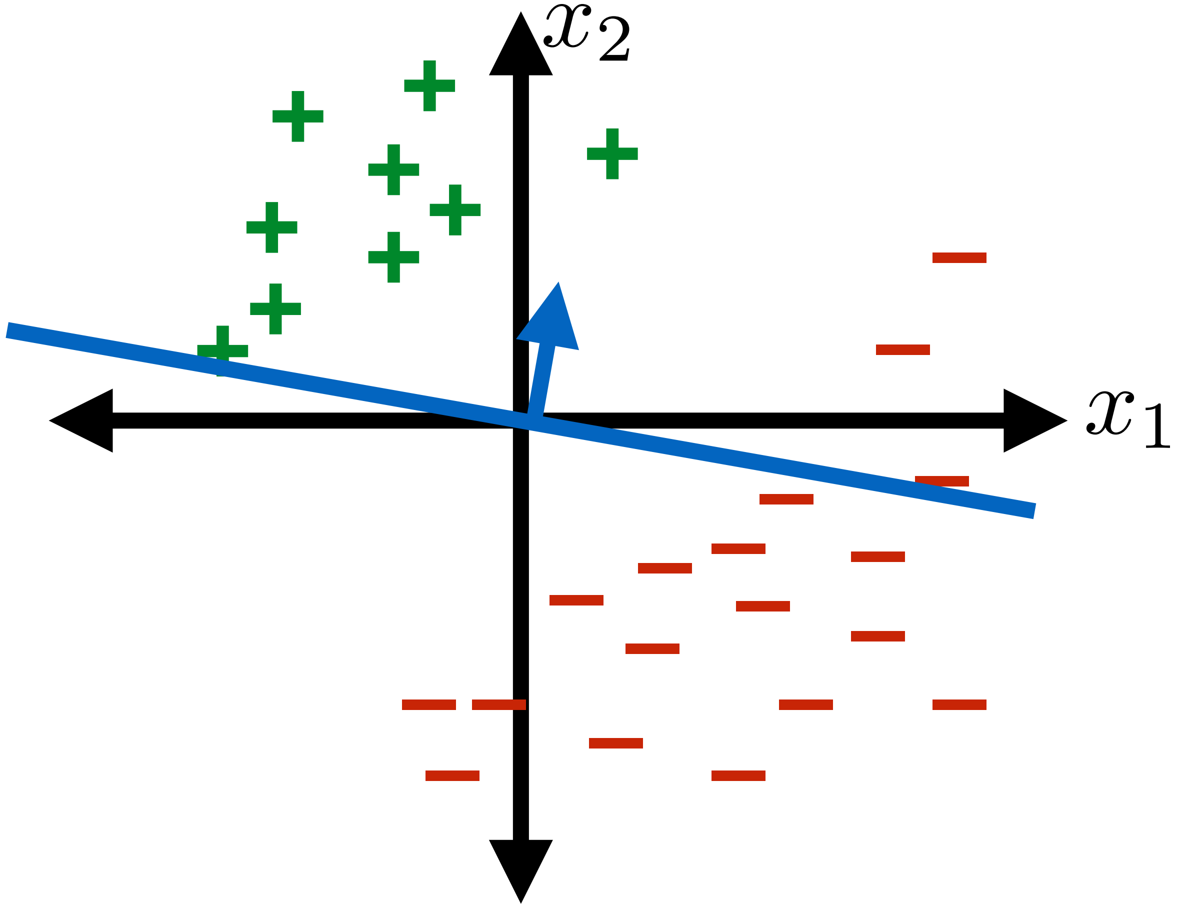
Theorem: Perceptron Performance

- **Assumptions**:
  A. Our hypothesis class = classifiers with separating hyperplanes that pass through the origin (i.e. $\theta_0 = 0$)



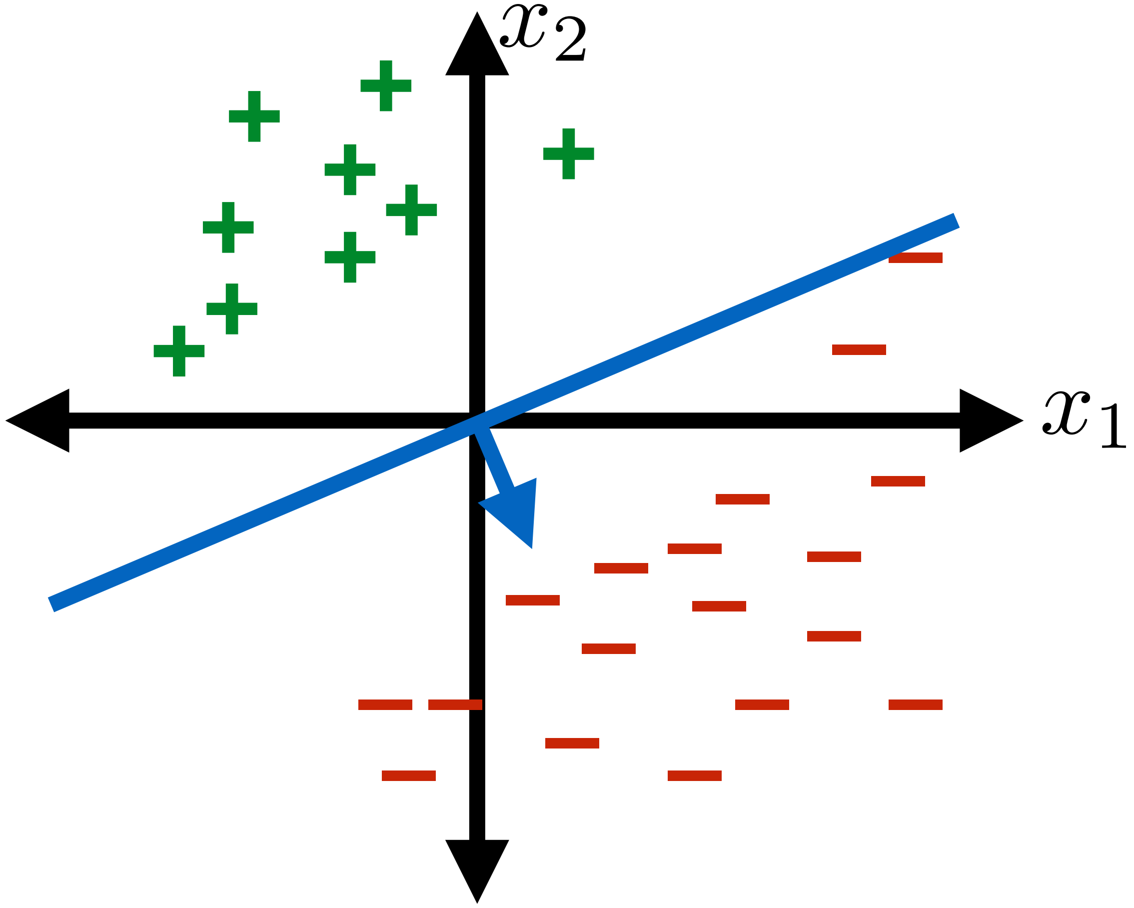
Theorem: Perceptron Performance

- **Assumptions**:
  A. Our hypothesis class = classifiers with separating hyperplanes that pass through the origin (i.e. $\theta_0 = 0$)



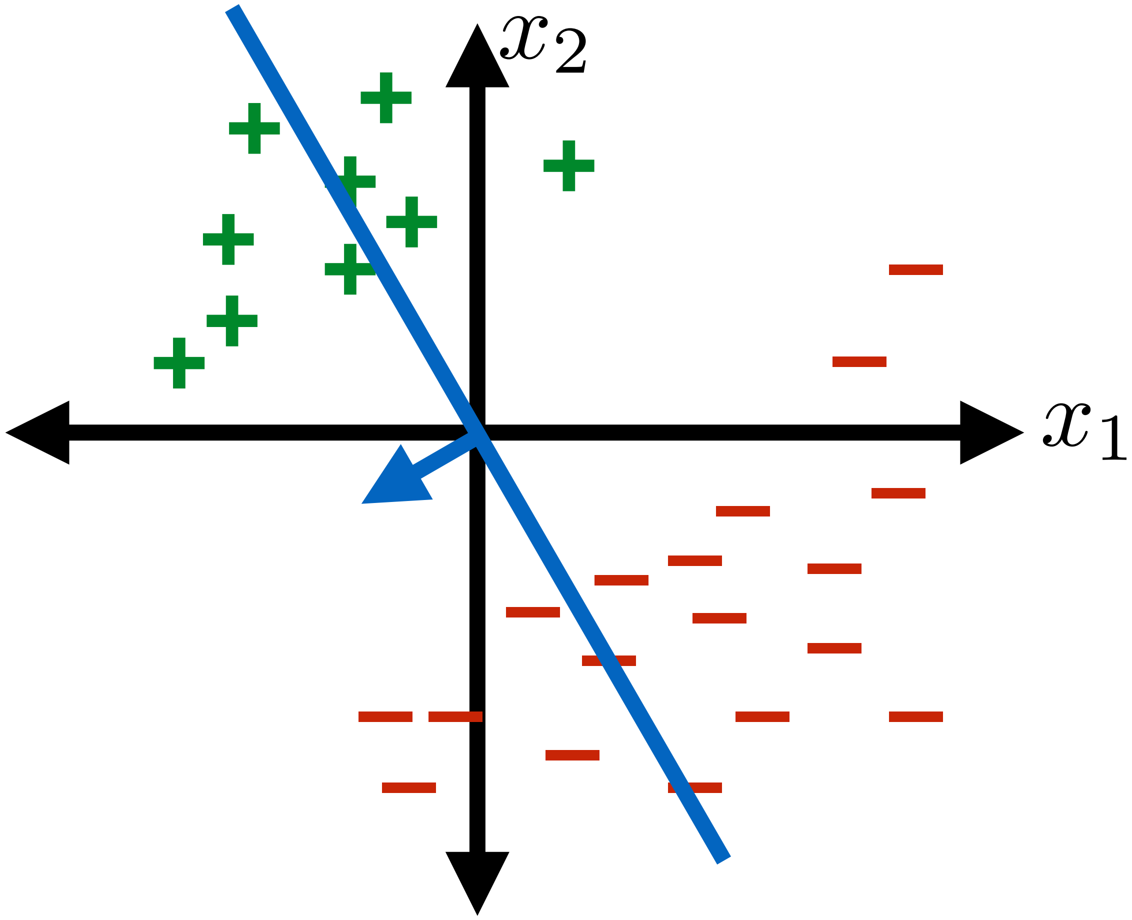
Theorem: Perceptron Performance

- **Assumptions**:
  A. Our hypothesis class = classifiers with separating hyperplanes that pass through the origin (i.e. $\theta_0 = 0$)



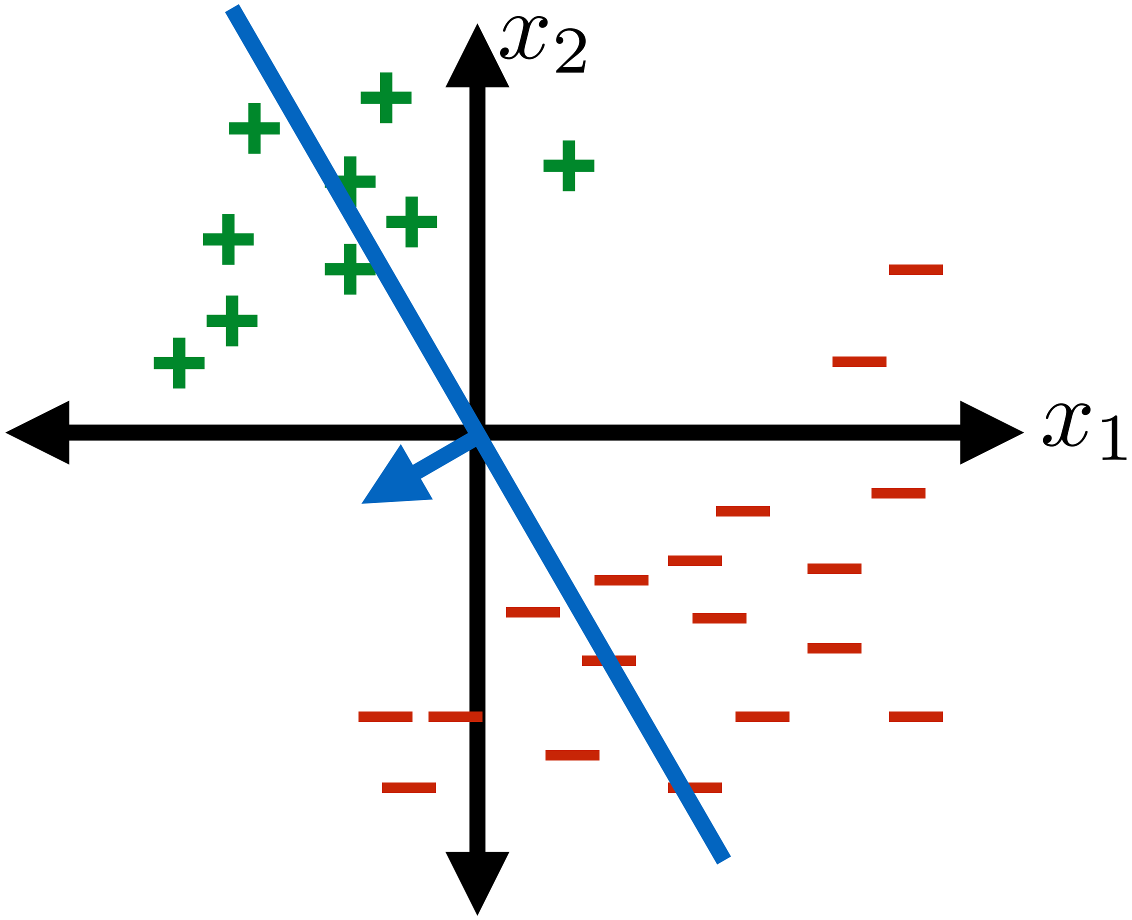
Theorem: Perceptron Performance

- **Assumptions**:
  A. Our hypothesis class = classifiers with separating hyperplanes that pass through the origin (i.e. $\theta_0 = 0$)



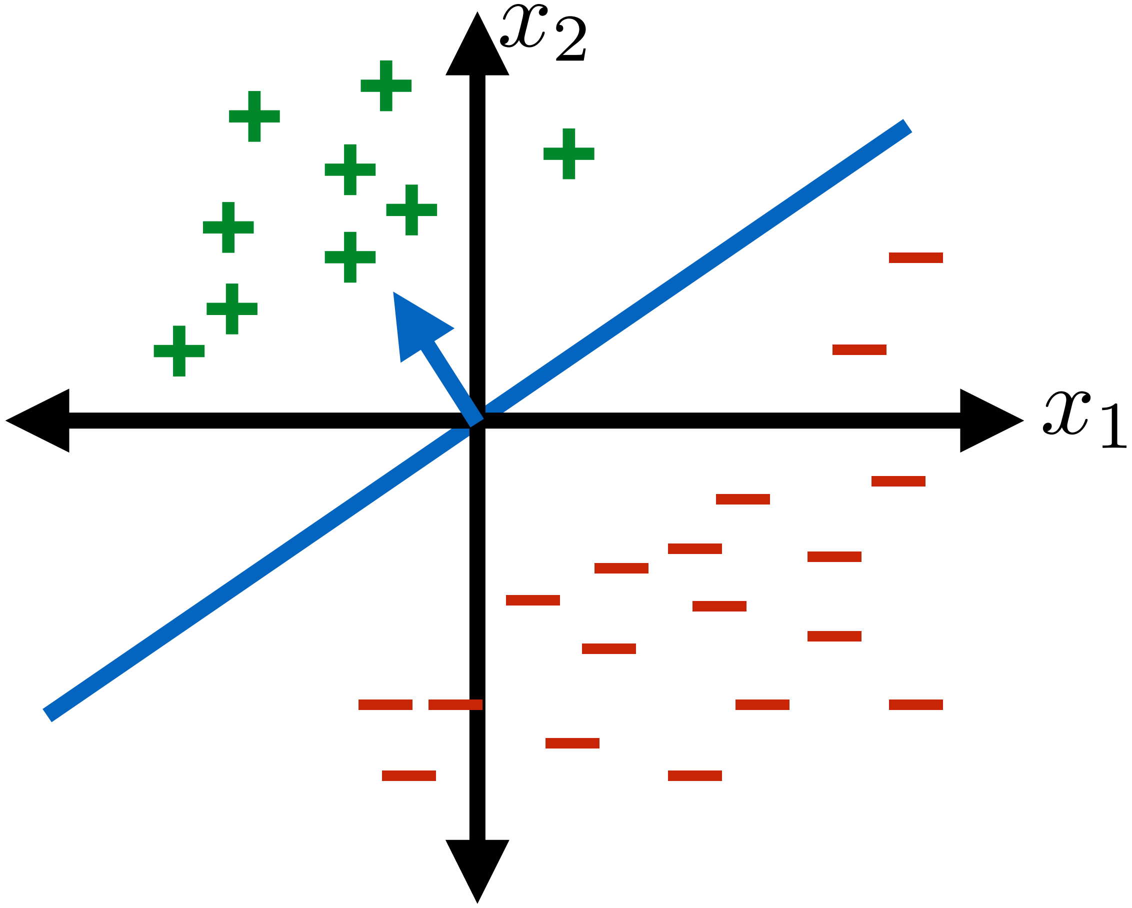
Theorem: Perceptron Performance

- **Assumptions**:
  A. Our hypothesis class = classifiers with separating hyperplanes that pass through the origin (i.e. $\theta_0 = 0$)
  B. There exist θ^* and γ such that $\gamma > 0$ and, for every $i \in \{1, \ldots, n\}$, we have $y^{(i)} \left(\frac{\theta^{*\top} x^{(i)}}{\|\theta\|} \right) > \gamma$



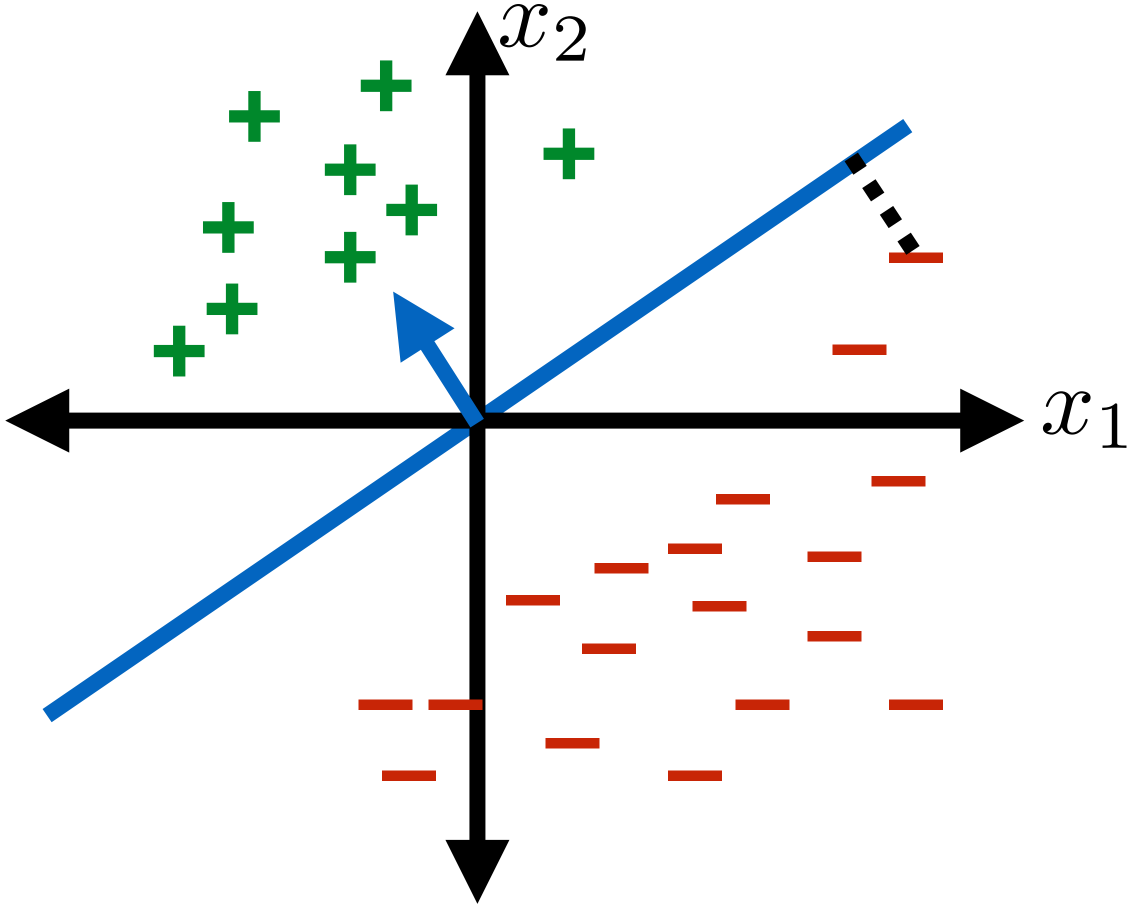
Theorem: Perceptron Performance

- **Assumptions**:
  A. Our hypothesis class = classifiers with separating hyperplanes that pass through the origin (i.e. $\theta_0 = 0$)
  B. There exist θ^* and γ such that $\gamma > 0$ and, for every $i \in \{1, \ldots, n\}$, we have $y^{(i)} \left(\frac{\theta^{*\top} x^{(i)}}{\|\theta\|} \right) > \gamma$



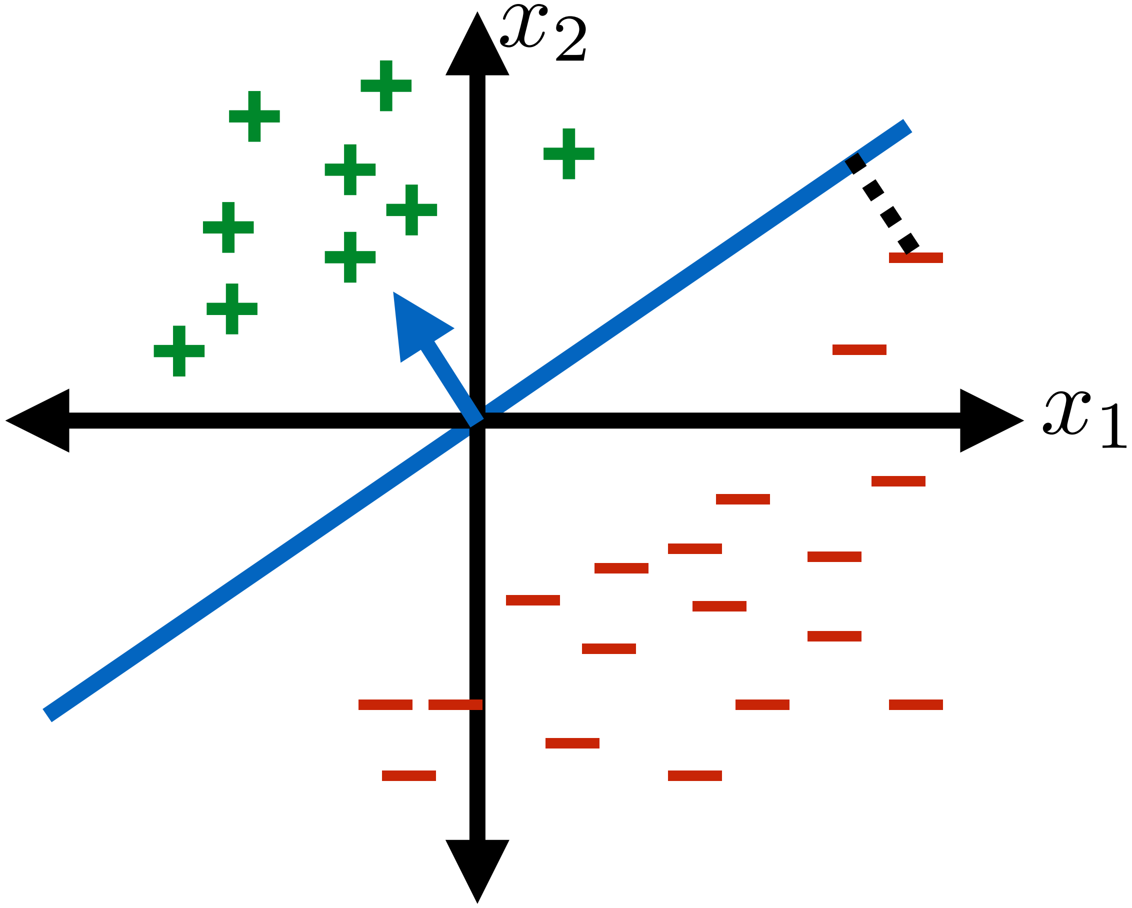
Theorem: Perceptron Performance

- **Assumptions**:
  A. Our hypothesis class = classifiers with separating hyperplanes that pass through the origin (i.e. $\theta_0 = 0$)
  B. There exist θ^* and γ such that $\gamma > 0$ and, for every $i \in \{1, \ldots, n\}$, we have $y^{(i)} \left(\frac{\theta^{*\top} x^{(i)}}{\|\theta\|} \right) > \gamma$



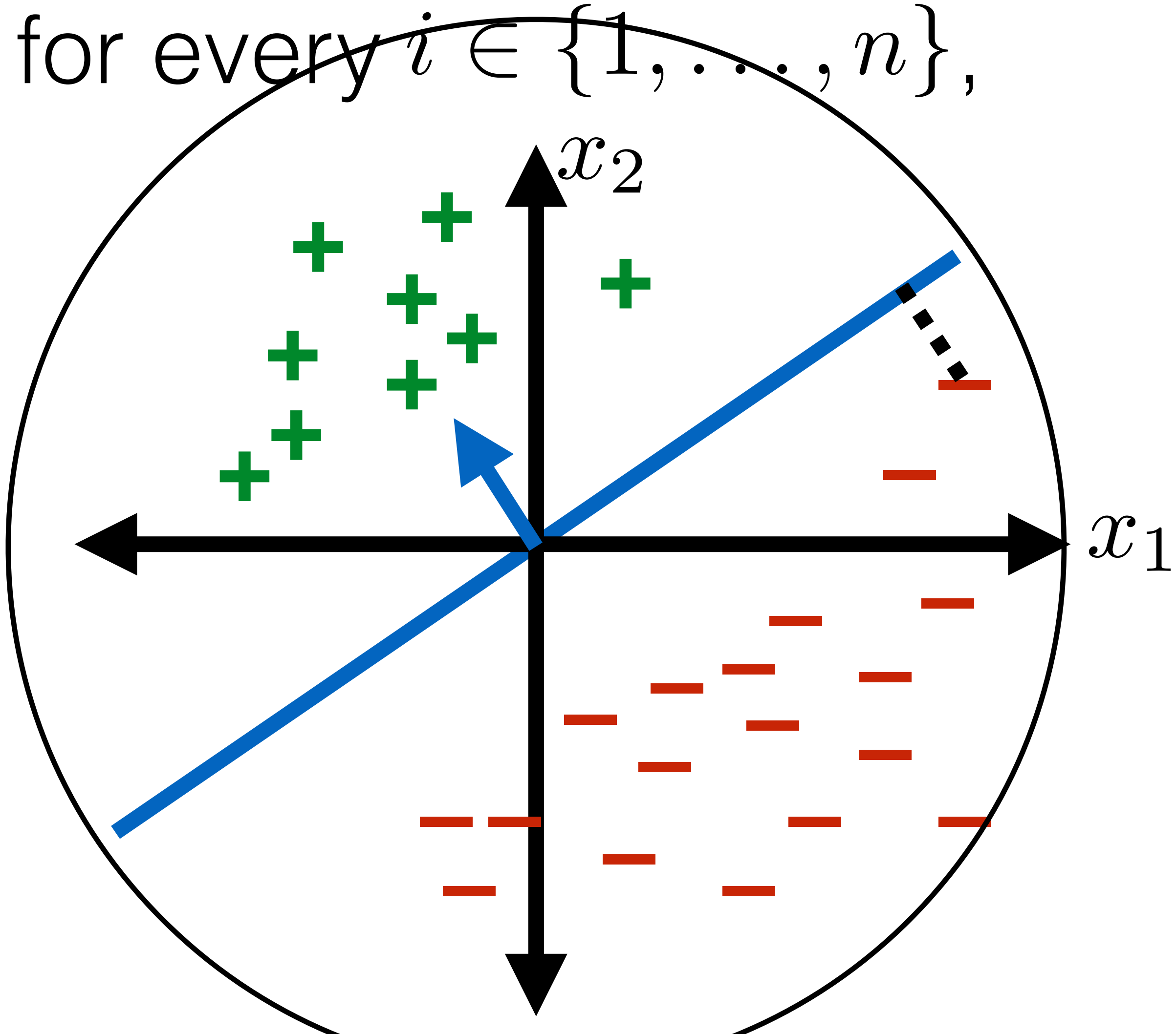
Theorem: Perceptron Performance

- **Assumptions**:
  A. Our hypothesis class = classifiers with separating hyperplanes that pass through the origin (i.e. $\theta_0 = 0$)
  B. There exist θ^* and γ such that $\gamma > 0$ and, for every $i \in \{1, \ldots, n\}$, we have $y^{(i)} \left(\frac{\theta^{*\top} x^{(i)}}{\|\theta\|} \right) > \gamma$
  C. There exists R such that, for every $i \in \{1, \ldots, n\}$, we have $\|x^{(i)}\| \leq R$



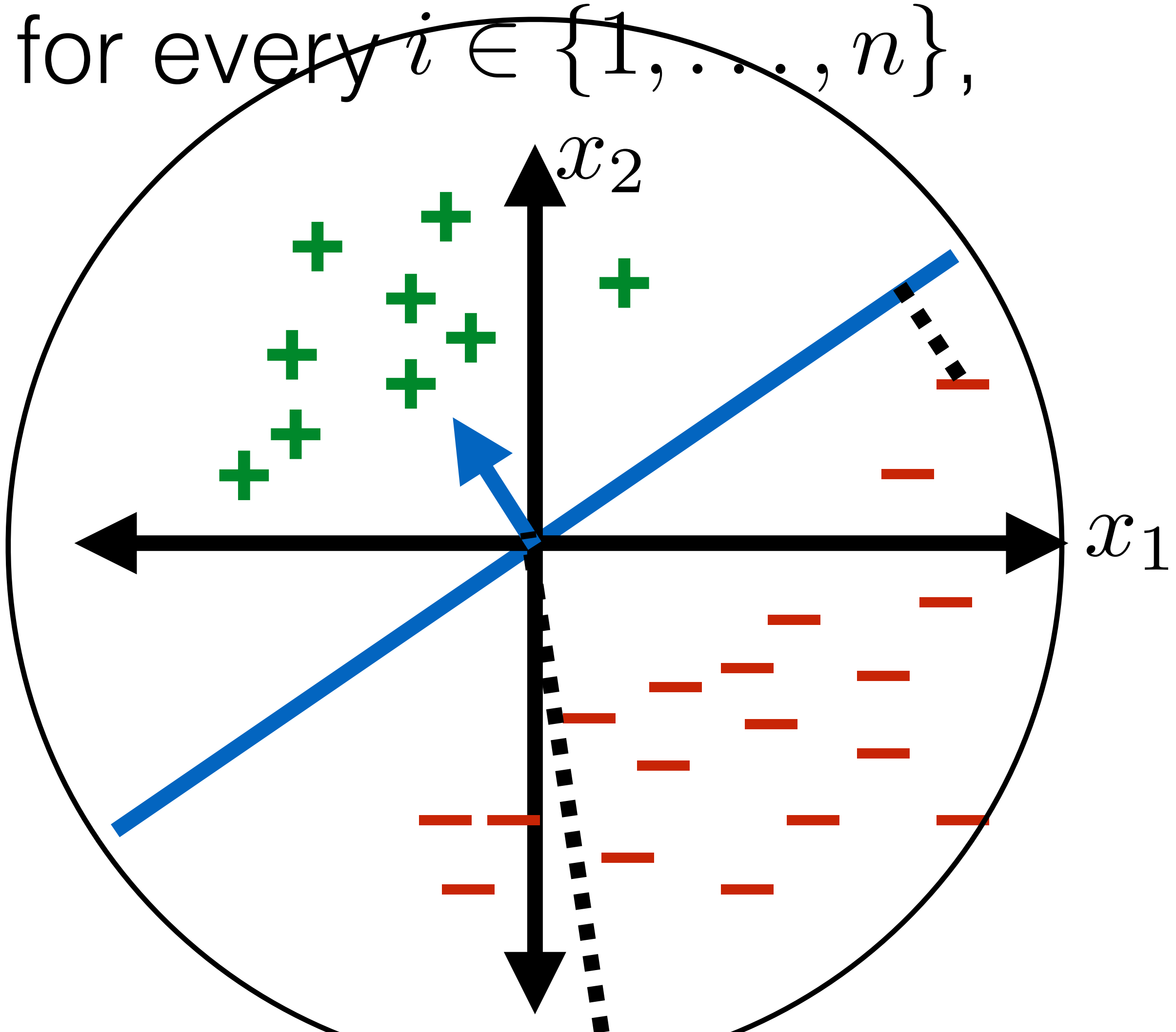
Theorem: Perceptron Performance

- **Assumptions**:
  A. Our hypothesis class = classifiers with separating hyperplanes that pass through the origin (i.e. $\theta_0 = 0$)
  B. There exist θ^* and γ such that $\gamma > 0$ and, for every $i \in \{1, \ldots, n\}$, we have $y^{(i)} \left(\frac{\theta^{*\top} x^{(i)}}{\|\theta\|} \right) > \gamma$
  C. There exists R such that, for every $i \in \{1, \ldots, n\}$, we have $\|x^{(i)}\| \leq R$



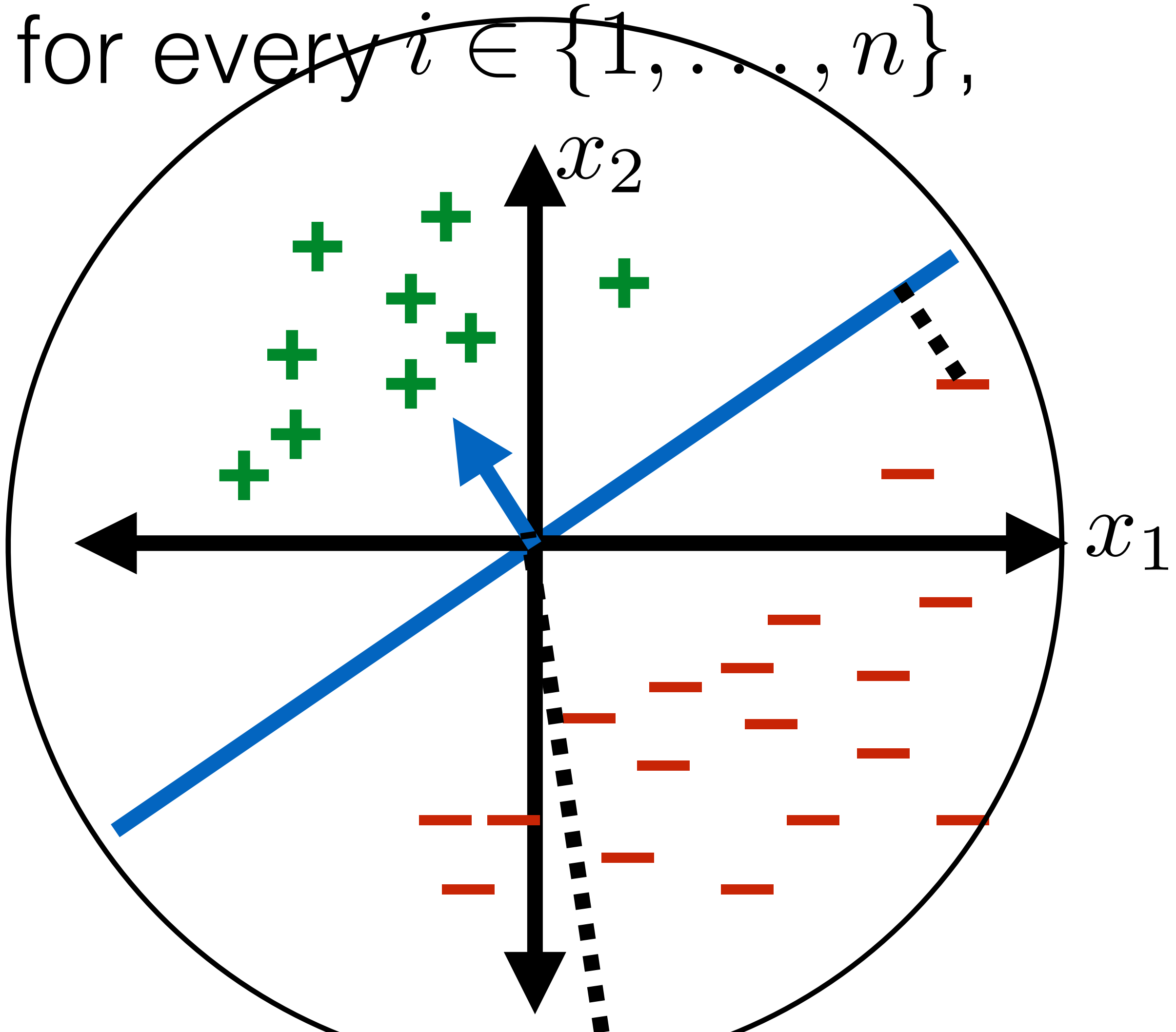
Theorem: Perceptron Performance

- **Assumptions**:
  A. Our hypothesis class = classifiers with separating hyperplanes that pass through the origin (i.e. $\theta_0 = 0$)
  B. There exist θ^* and γ such that $\gamma > 0$ and, for every $i \in \{1, \ldots, n\}$, we have $$y^{(i)} \left(\frac{\theta^{*\top} x^{(i)}}{\|\theta\|} \right) > \gamma$$
  C. There exists R such that, for every $i \in \{1, \ldots, n\}$, we have $\|x^{(i)}\| \leq R$



Theorem: Perceptron Performance

- **Assumptions**:
  A. Our hypothesis class = classifiers with separating hyperplanes that pass through the origin (i.e. $\theta_0 = 0$)
  B. There exist θ^* and γ such that $\gamma > 0$ and, for every $i \in \{1, \ldots, n\}$, we have $$y^{(i)} \left(\frac{\theta^{*\top} x^{(i)}}{\|\theta\|} \right) > \gamma$$
  C. There exists R such that, for every $i \in \{1, \ldots, n\}$, we have $\|x^{(i)}\| \leq R$
- **Conclusion**: Then the perceptron algorithm will make at most $(R/\gamma)^2$ updates to θ. Once it goes through a pass of *i* without changes, the training error of its hypothesis will be 0.



Why classifiers through the origin?

Why classifiers through the origin?

- If we’re clever, we don’t lose any flexibility

Why classifiers through the origin?

- If we're clever, we don't lose any flexibility
    - Classifier with offset

Why classifiers through the origin?

- If we're clever, we don't lose any flexibility
  - Classifier with offset

$$x \in \mathbb{R}^d, \theta \in \mathbb{R}^d, \theta_0 \in \mathbb{R}$$

$$x : \theta^\top x + \theta_0 = 0$$

Why classifiers through the origin?

- If we're clever, we don't lose any flexibility
  - Classifier with offset

$$x \in \mathbb{R}^d, \theta \in \mathbb{R}^d, \theta_0 \in \mathbb{R}$$

$$x : \theta^\top x + \theta_0 = 0$$

  - Classifier without offset

Why classifiers through the origin?

- If we're clever, we don't lose any flexibility
  - Classifier with offset

$$x \in \mathbb{R}^d, \theta \in \mathbb{R}^d, \theta_0 \in \mathbb{R}$$

$$x : \theta^\top x + \theta_0 = 0$$

  - Classifier without offset

$$x_{\text{new}} \in \mathbb{R}^{d+1}, \theta_{\text{new}} \in \mathbb{R}^{d+1}$$

$$x_{\text{new}} = [x_1, x_2, \ldots, x_d, 1]^\top, \theta_{\text{new}} = [\theta_1, \theta_2, \ldots, \theta_d, \theta_0]^\top$$

Why classifiers through the origin?

- If we're clever, we don't lose any flexibility
  - Classifier with offset

$$x \in \mathbb{R}^d, \theta \in \mathbb{R}^d, \theta_0 \in \mathbb{R}$$

$$x : \theta^\top x + \theta_0 = 0$$

  - Classifier without offset

$$x_{\text{new}} \in \mathbb{R}^{d+1}, \theta_{\text{new}} \in \mathbb{R}^{d+1}$$

$$x_{\text{new}} = [x_1, x_2, \ldots, x_d, 1]^\top, \theta_{\text{new}} = [\theta_1, \theta_2, \ldots, \theta_d, \theta_0]^\top$$

$$x_{\text{new},1:d} : \theta_{\text{new}}^\top x_{\text{new}} = 0$$

Why classifiers through the origin?

- If we're clever, we don't lose any flexibility
  - Classifier with offset

$$x \in \mathbb{R}^d, \theta \in \mathbb{R}^d, \theta_0 \in \mathbb{R}$$

$$x : \theta^\top x + \theta_0 = 0$$

  - Classifier without offset

$$x_{\text{new}} \in \mathbb{R}^{d+1}, \theta_{\text{new}} \in \mathbb{R}^{d+1}$$

$$x_{\text{new}} = [x_1, x_2, \ldots, x_d, 1]^\top, \theta_{\text{new}} = [\theta_1, \theta_2, \ldots, \theta_d, \theta_0]^\top$$

$$x_{\text{new},1:d} : \theta_{\text{new}}^\top x_{\text{new}} = 0$$

Why classifiers through the origin?

- If we're clever, we don't lose any flexibility
  - Classifier with offset

$$x \in \mathbb{R}^d, \theta \in \mathbb{R}^d, \theta_0 \in \mathbb{R}$$

$$x : \theta^\top x + \theta_0 \overset{<}{=} 0$$

  - Classifier without offset

$$x_{\text{new}} \in \mathbb{R}^{d+1}, \theta_{\text{new}} \in \mathbb{R}^{d+1}$$

$$x_{\text{new}} = [x_1, x_2, \ldots, x_d, 1]^\top, \theta_{\text{new}} = [\theta_1, \theta_2, \ldots, \theta_d, \theta_0]^\top$$

$$x_{\text{new},1:d} : \theta_{\text{new}}^\top x_{\text{new}} \overset{<}{=} 0$$

Why classifiers through the origin?

- If we're clever, we don't lose any flexibility
  - Classifier with offset

$$x \in \mathbb{R}^d, \theta \in \mathbb{R}^d, \theta_0 \in \mathbb{R}$$

$$x : \theta^\top x + \theta_0 \underset{>}{\overset{<}{=}} 0$$

  - Classifier without offset

$$x_{\text{new}} \in \mathbb{R}^{d+1}, \theta_{\text{new}} \in \mathbb{R}^{d+1}$$

$$x_{\text{new}} = [x_1, x_2, \ldots, x_d, 1]^\top, \theta_{\text{new}} = [\theta_1, \theta_2, \ldots, \theta_d, \theta_0]^\top$$

$$x_{\text{new},1:d} : \theta_{\text{new}}^\top x_{\text{new}} \underset{>}{\overset{<}{=}} 0$$

Why classifiers through the origin?

- If we're clever, we don't lose any flexibility
  - Classifier with offset

$$x \in \mathbb{R}^d, \theta \in \mathbb{R}^d, \theta_0 \in \mathbb{R}$$

$$x : \theta^\top x + \theta_0 \overset{<}{\underset{>}{=}} 0$$

  - Classifier without offset

$$x_{\text{new}} \in \mathbb{R}^{d+1}, \theta_{\text{new}} \in \mathbb{R}^{d+1}$$

$$x_{\text{new}} = [x_1, x_2, \ldots, x_d, 1]^\top, \theta_{\text{new}} = [\theta_1, \theta_2, \ldots, \theta_d, \theta_0]^\top$$

$$x_{\text{new},1:d} : \theta_{\text{new}}^\top x_{\text{new}} \overset{<}{\underset{>}{=}} 0$$

Why classifiers through the origin?

- If we're clever, we don't lose any flexibility
  - Classifier with offset

$$x \in \mathbb{R}^d, \theta \in \mathbb{R}^d, \theta_0 \in \mathbb{R}$$

$$x : \theta^\top x + \theta_0 \overset{<}{\underset{>}{=}} 0$$

  - Classifier without offset

$$x_{\text{new}} \in \mathbb{R}^{d+1}, \theta_{\text{new}} \in \mathbb{R}^{d+1}$$

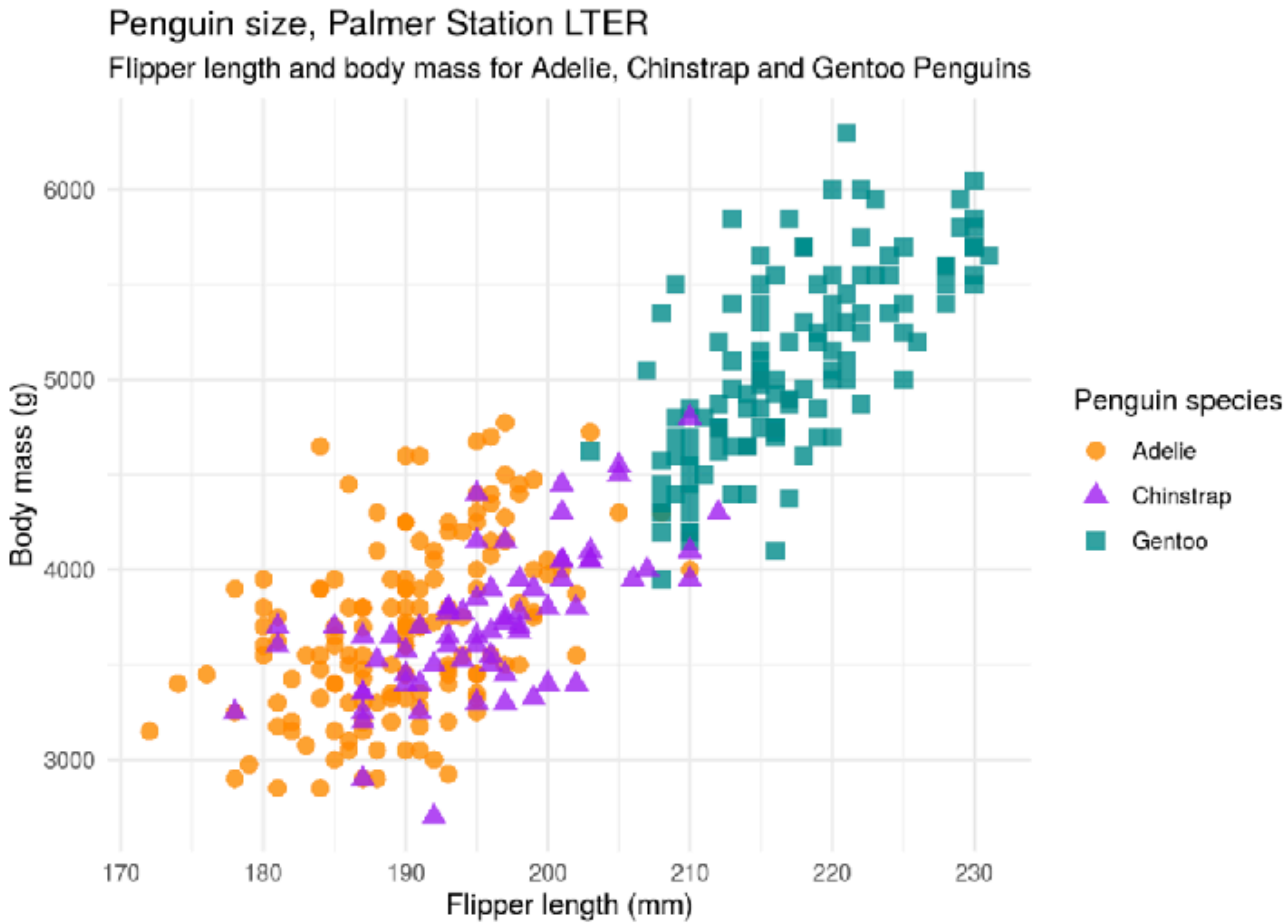
$$x_{\text{new}} = [x_1, x_2, \ldots, x_d, 1]^\top, \theta_{\text{new}} = [\theta_1, \theta_2, \ldots, \theta_d, \theta_0]^\top$$

$$x_{\text{new},1:d} : \theta_{\text{new}}^\top x_{\text{new}} \overset{<}{\underset{>}{=}} 0$$

- Can first convert to "expanded" feature space, then apply theorem

Problem: data not linearly separable

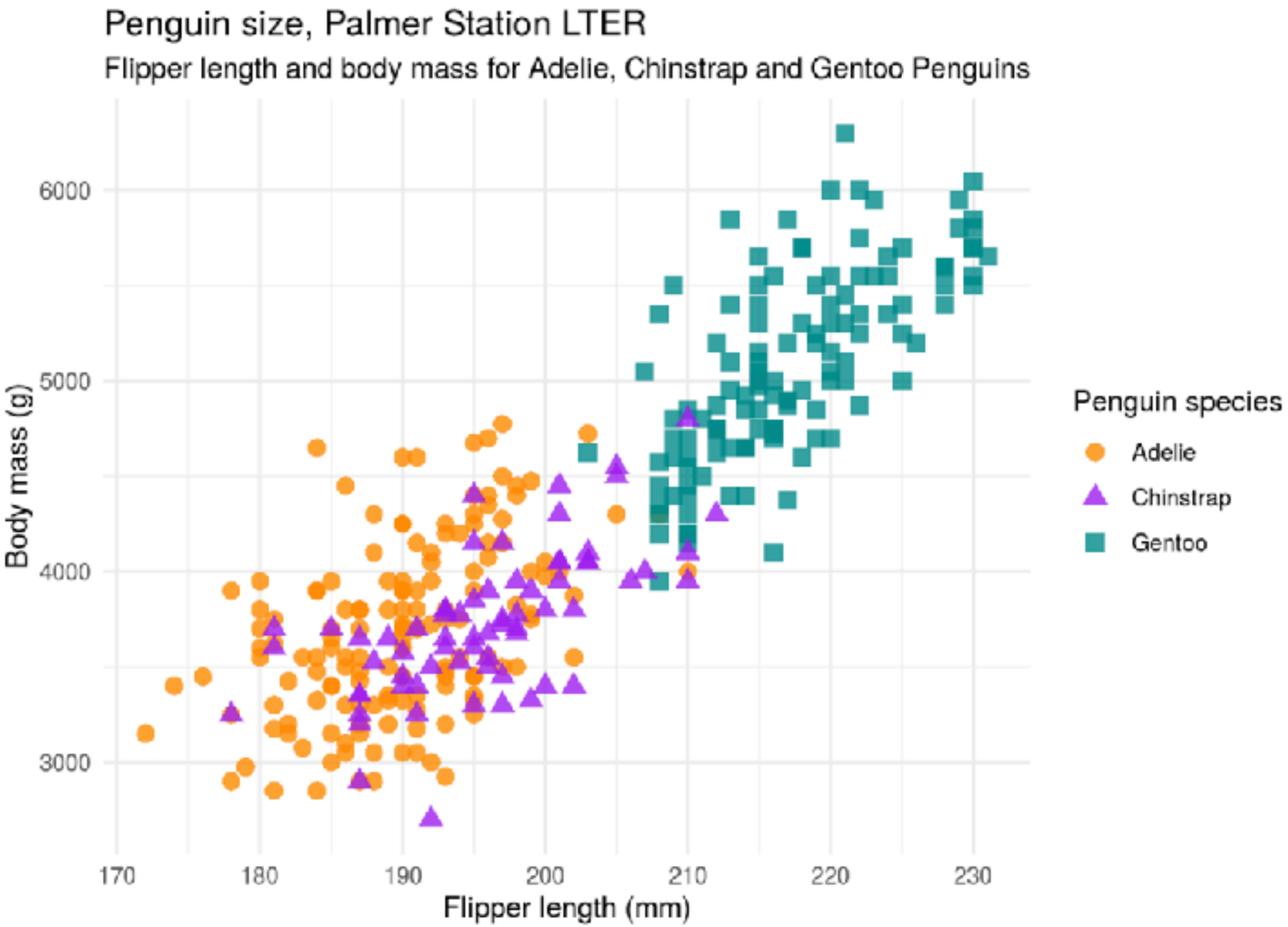
- Typical real data sets aren't linearly separable



[https://towardsdatascience.com/penguins-dataset-overview-iris-alternative-9453bb8c8d95, https://github.com/allisonhorst/palmerpenguins]

Problem: data not linearly separable

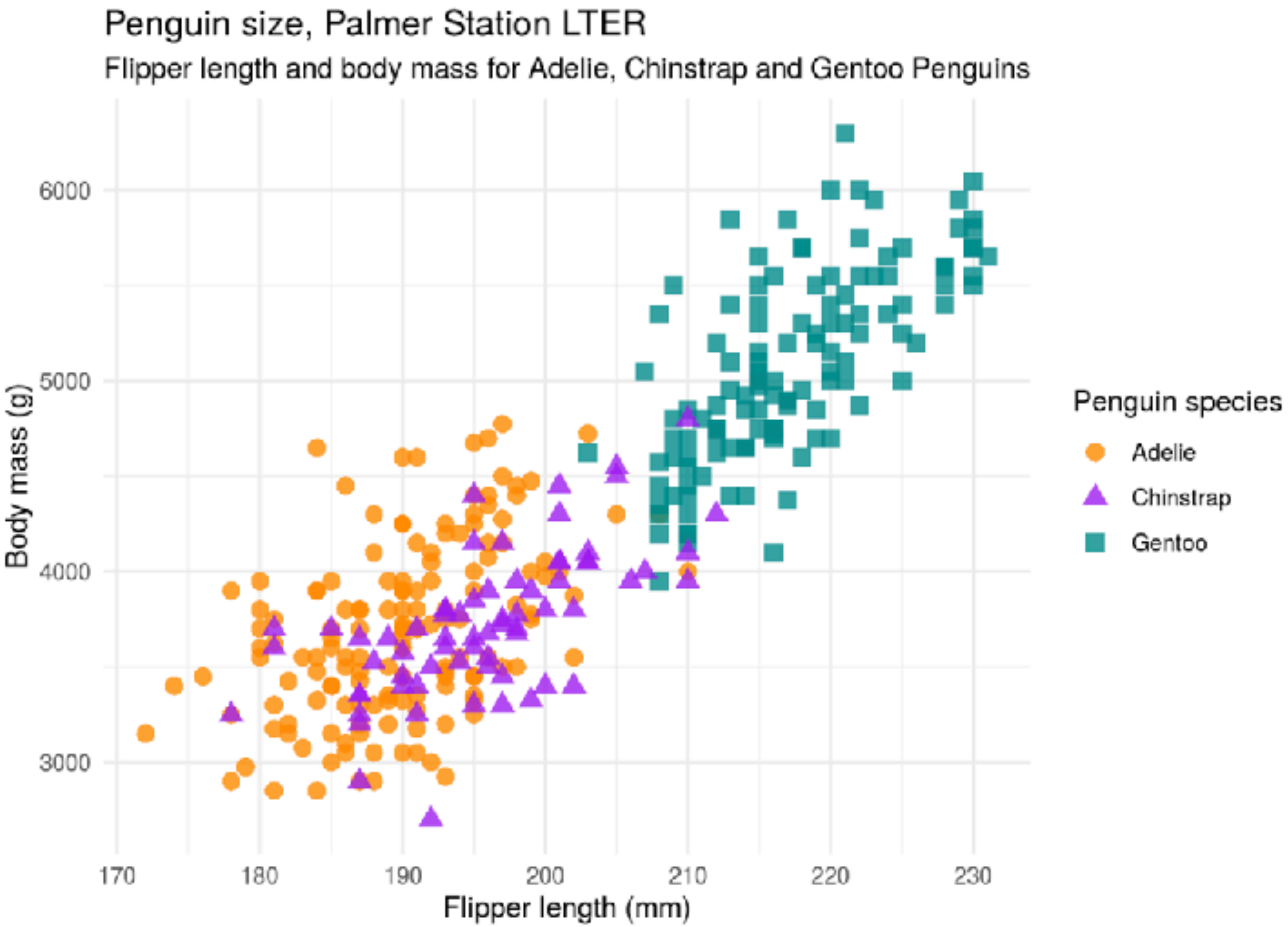
- Typical real data sets aren't linearly separable [demo]



[https://towardsdatascience.com/penguins-dataset-overview-iris-alternative-9453bb8c8d95, https://github.com/allisonhorst/palmerpenguins]

Problem: data not linearly separable

- Typical real data sets aren't linearly separable [demo]

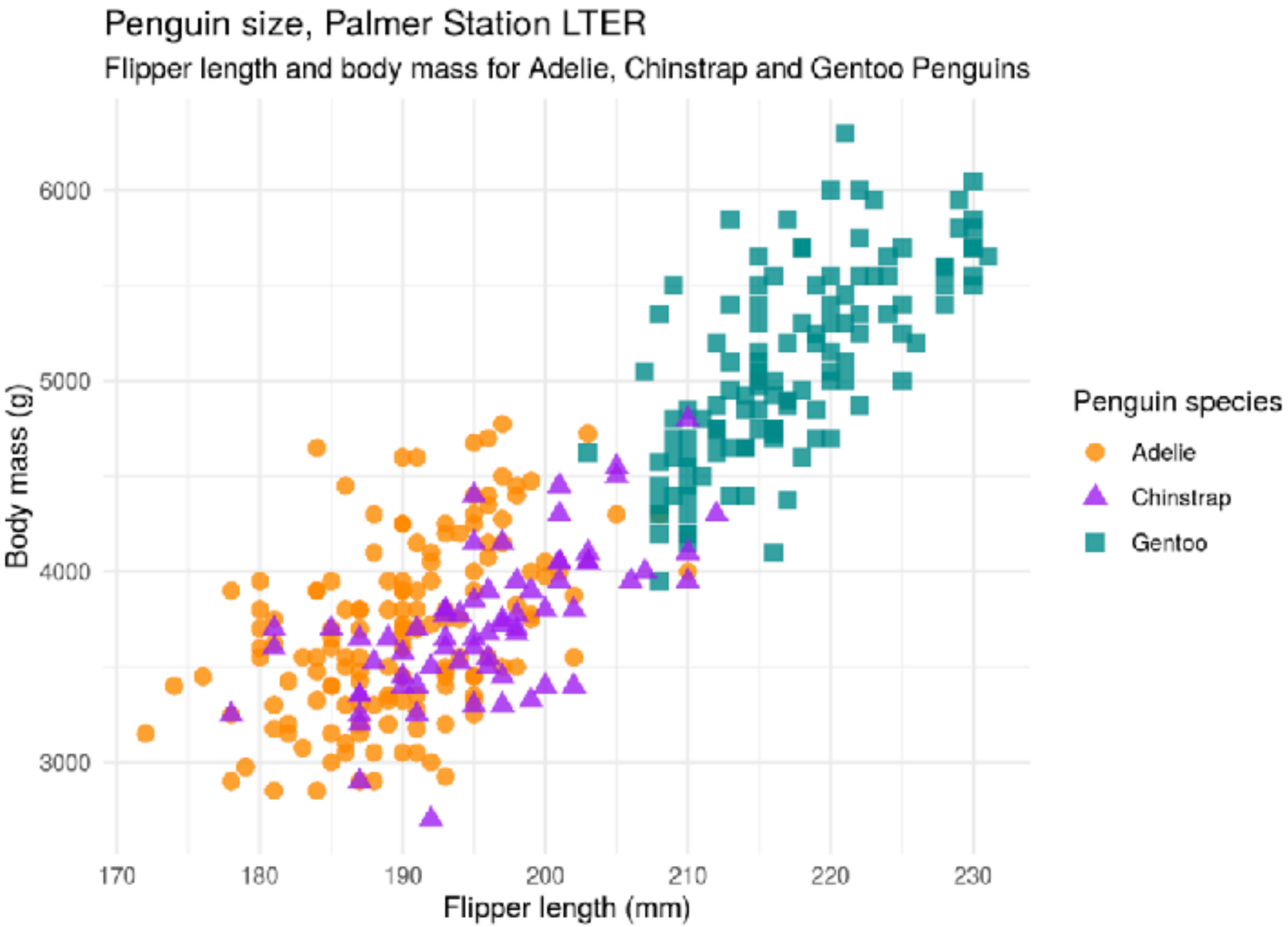


- What can we do?

[https://towardsdatascience.com/penguins-dataset-overview-iris-alternative-9453bb8c8d95, https://github.com/allisonhorst/palmerpenguins]

Problem: data not linearly separable

- Typical real data sets aren't linearly separable [demo]



- What can we do? See upcoming lectures!

[https://towardsdatascience.com/penguins-dataset-overview-iris-alternative-9453bb8c8d95, https://github.com/allisonhorst/palmerpenguins]

Machine Learning Tasks

Machine Learning Tasks

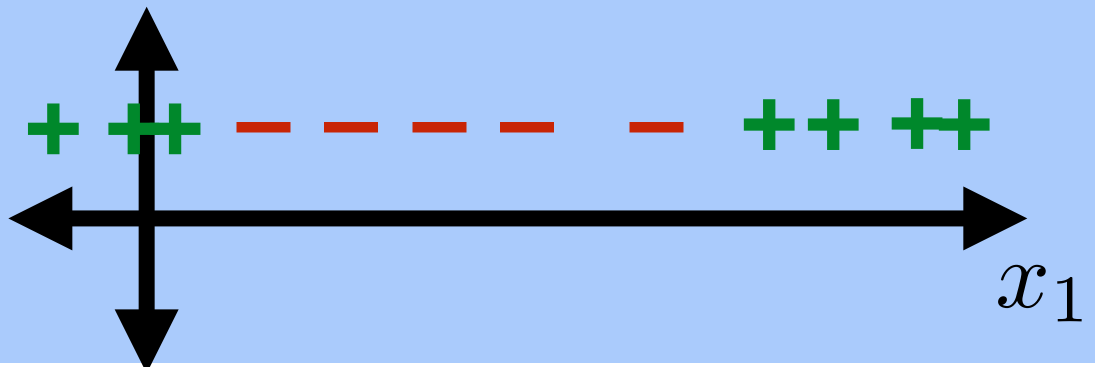
- **Binary/two-class classification**

Machine Learning Tasks

- **Binary/two-class classification:** Learn a mapping: $\mathbb{R}^d \rightarrow \{-1, +1\}$

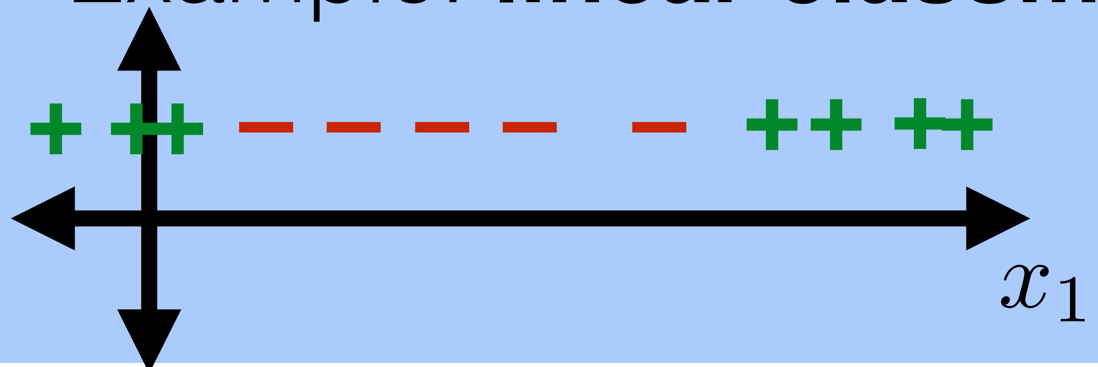
Machine Learning Tasks

- **Binary/two-class classification:** Learn a mapping: $\mathbb{R}^d \rightarrow \{-1, +1\}$



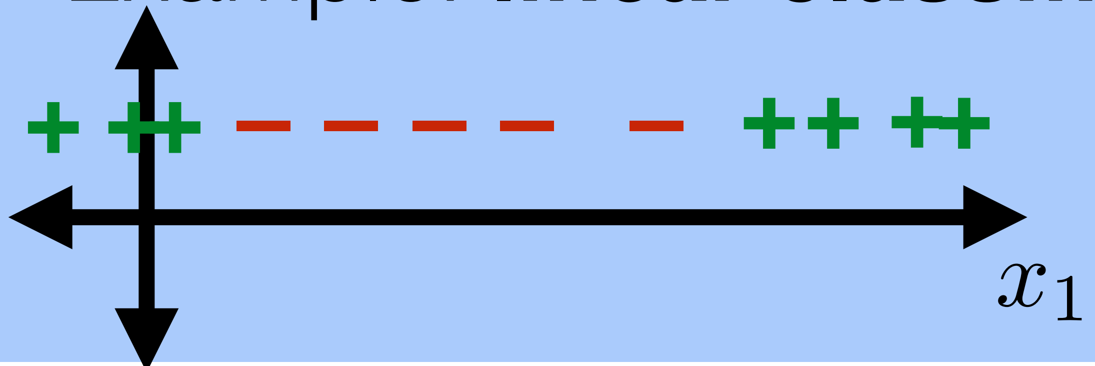
Machine Learning Tasks

- **Binary/two-class classification:** Learn a mapping: $\mathbb{R}^d \rightarrow \{-1, +1\}$
  - Example: **linear classification**



Machine Learning Tasks

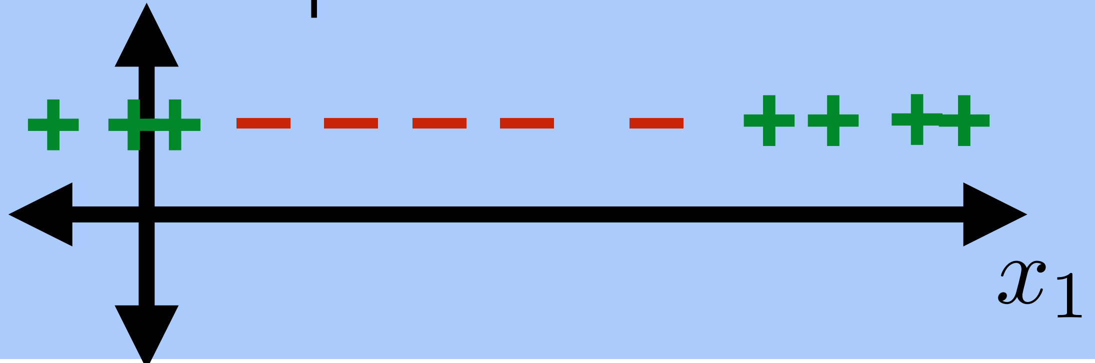
- **Binary/two-class classification:** Learn a mapping: $\mathbb{R}^d \rightarrow \{-1, +1\}$
  - Example: **linear classification**



- **Multi-class classification:**

Machine Learning Tasks

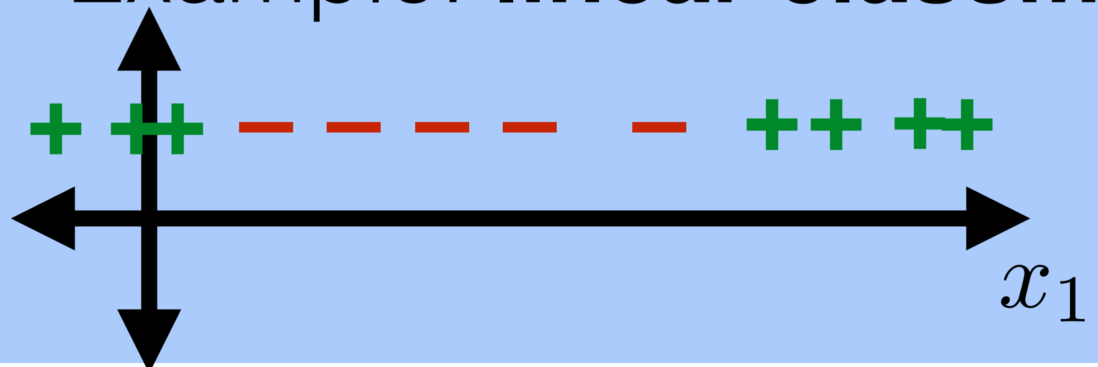
- **Binary/two-class classification:** Learn a mapping: $\mathbb{R}^d \rightarrow \{-1, +1\}$
  - Example: **linear classification**



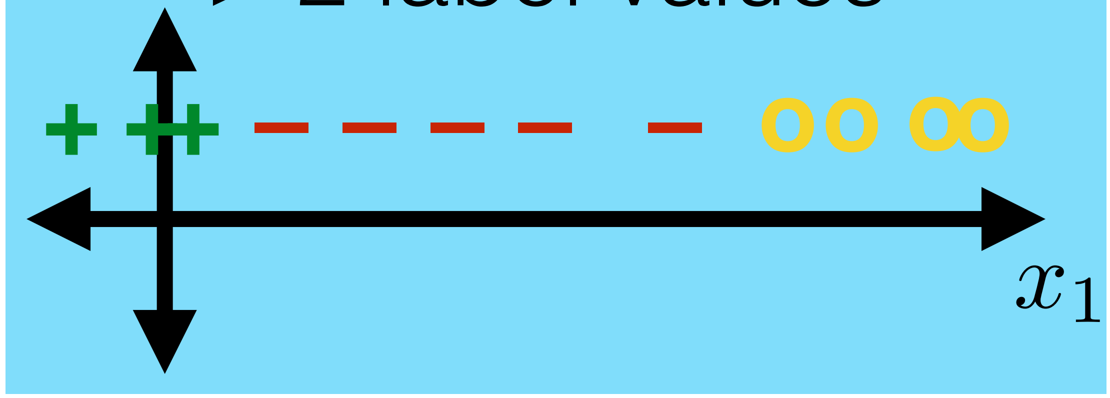
- **Multi-class classification:** > 2 label values

Machine Learning Tasks

- **Binary/two-class classification:** Learn a mapping: $\mathbb{R}^d \rightarrow \{-1, +1\}$
  - Example: **linear classification**

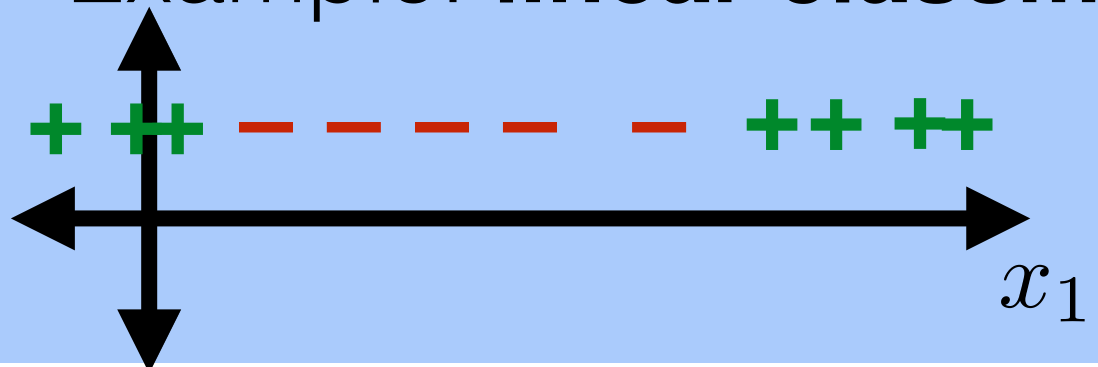


- **Multi-class classification:** > 2 label values

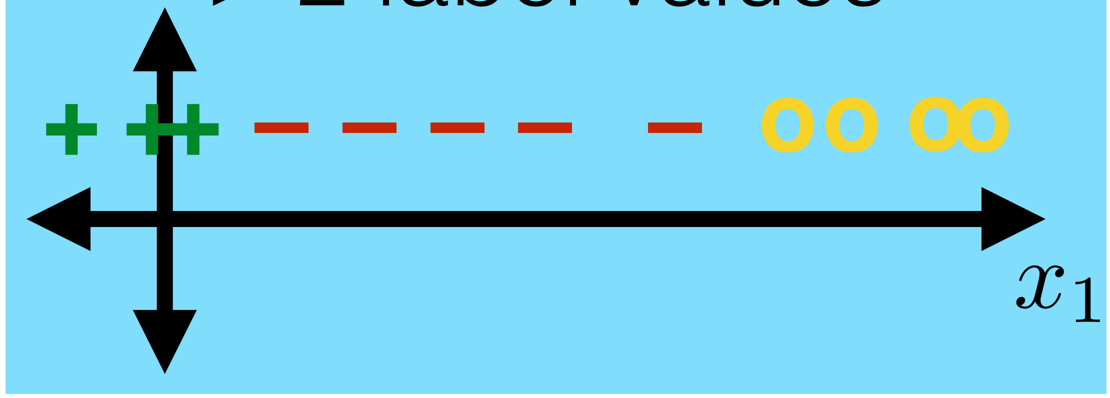


Machine Learning Tasks

- **Binary/two-class classification:** Learn a mapping: $\mathbb{R}^d \rightarrow \{-1, +1\}$
  - Example: **linear classification**



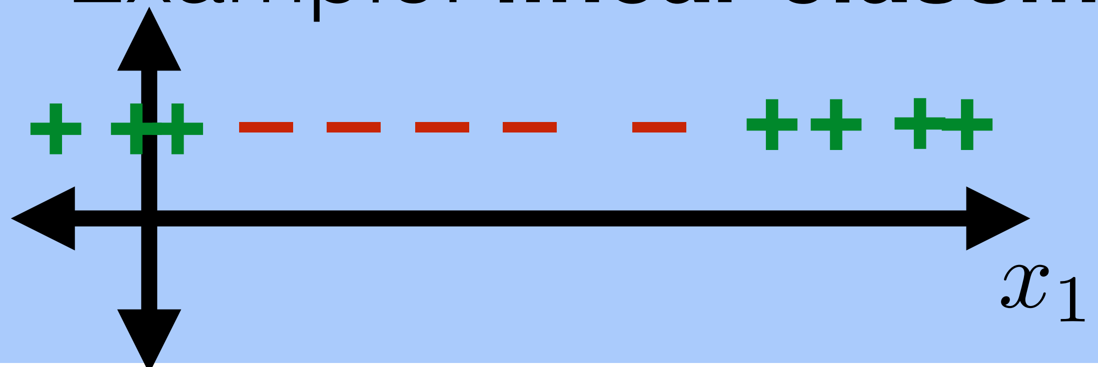
- **Multi-class classification:** > 2 label values



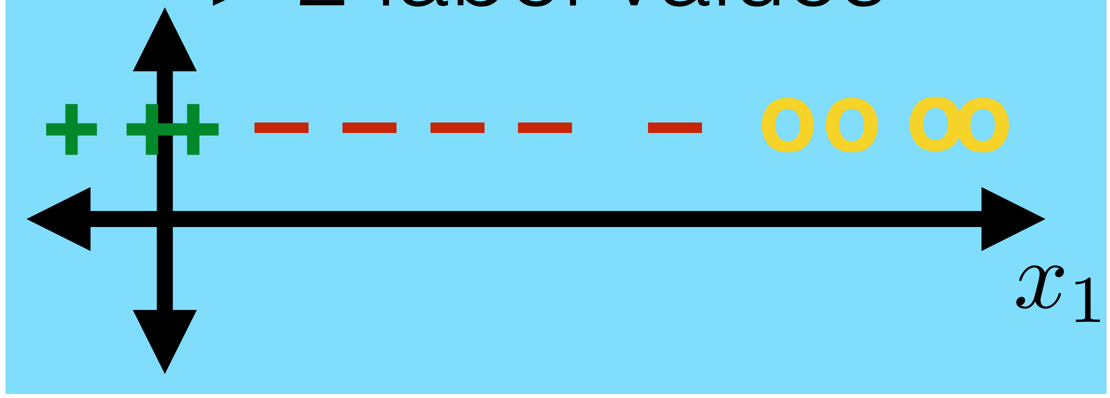
Machine Learning Tasks

- **Classification**

- **Binary/two-class classification:** Learn a mapping: $\mathbb{R}^d \rightarrow \{-1, +1\}$
  - Example: **linear classification**



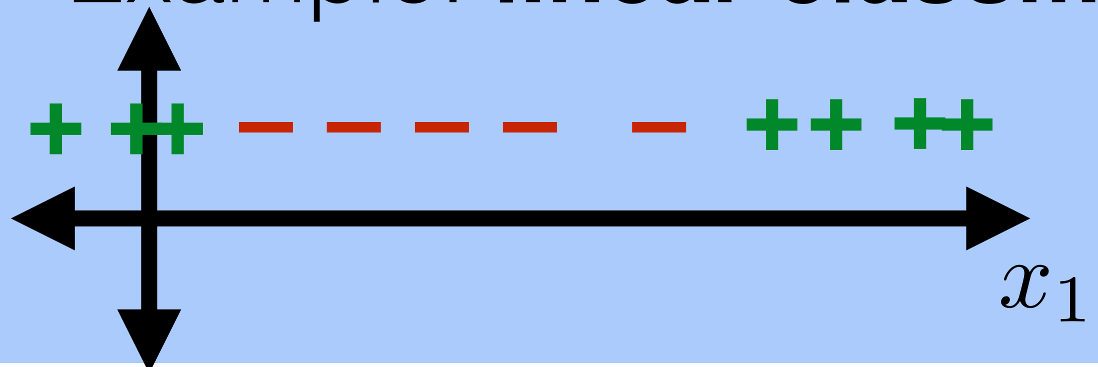
- **Multi-class classification:** > 2 label values



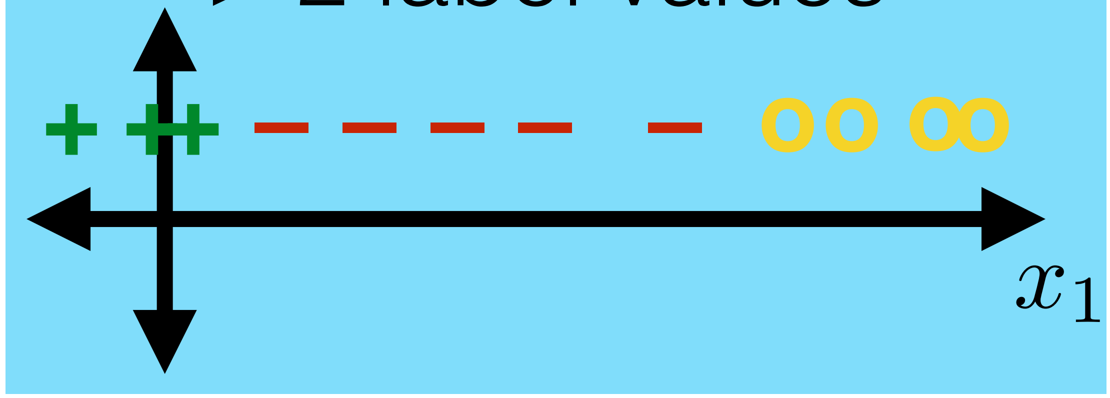
Machine Learning Tasks

- **Classification:** Learn a mapping to a discrete set

- **Binary/two-class classification:** Learn a mapping: $\mathbb{R}^d \rightarrow \{-1,+1\}$
  - Example: **linear classification**



- **Multi-class classification:** > 2 label values

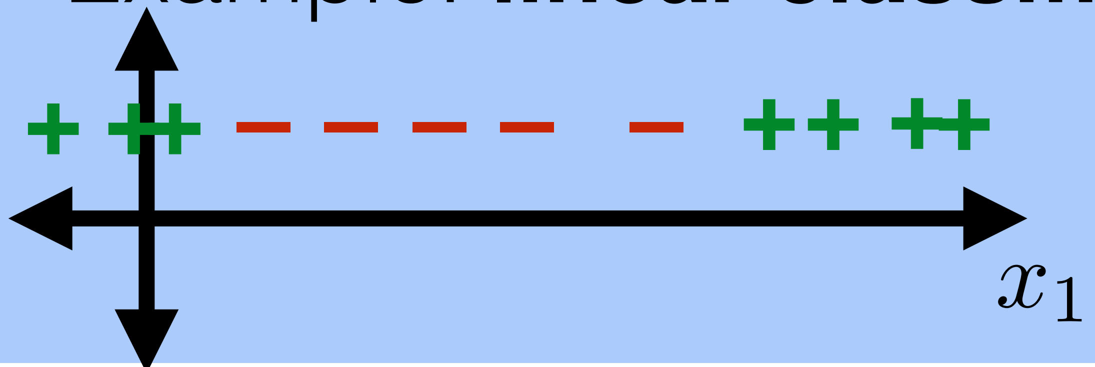


Machine Learning Tasks

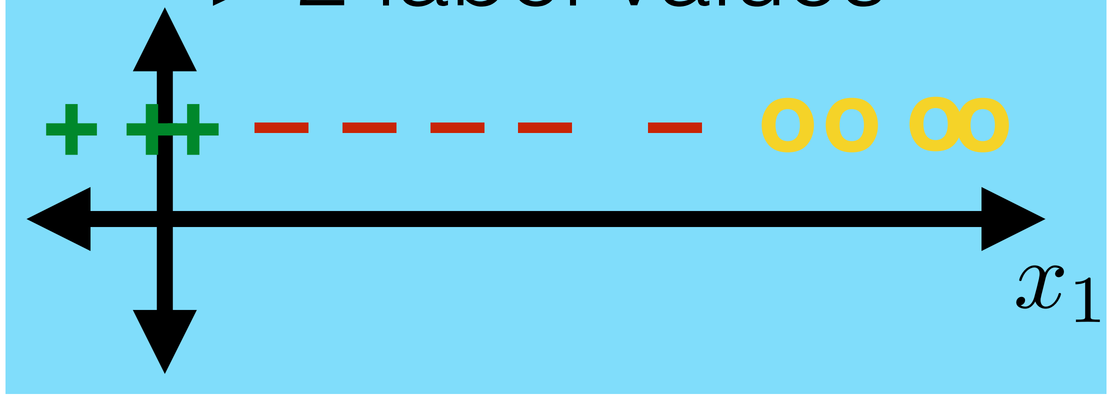
- **Regression**

- **Classification:** Learn a mapping to a discrete set

- **Binary/two-class classification:** Learn a mapping: $\mathbb{R}^d \rightarrow \{-1, +1\}$
  - Example: **linear classification**



- **Multi-class classification:** > 2 label values

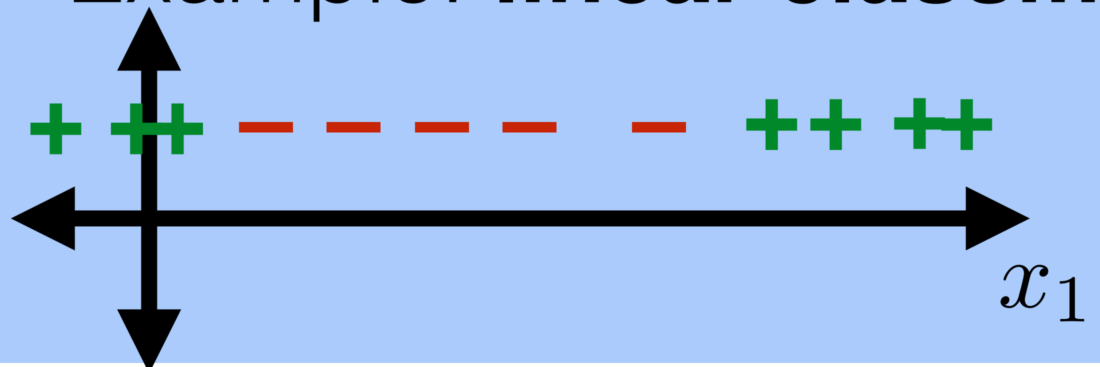


Machine Learning Tasks

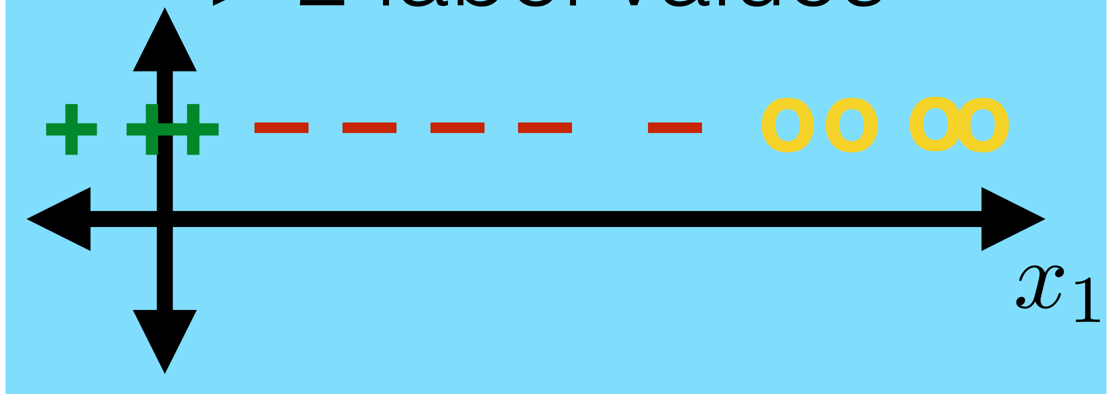
- **Regression:** Learn a mapping to continuous values: $\mathbb{R}^d \rightarrow \mathbb{R}^k$

- **Classification:** Learn a mapping to a discrete set

- **Binary/two-class classification:** Learn a mapping: $\mathbb{R}^d \rightarrow \{-1, +1\}$
  - Example: **linear classification**

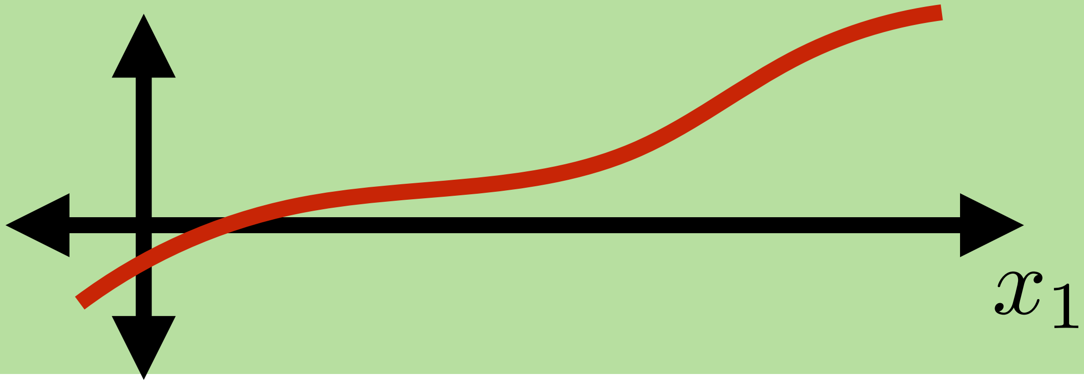


- **Multi-class classification:** > 2 label values



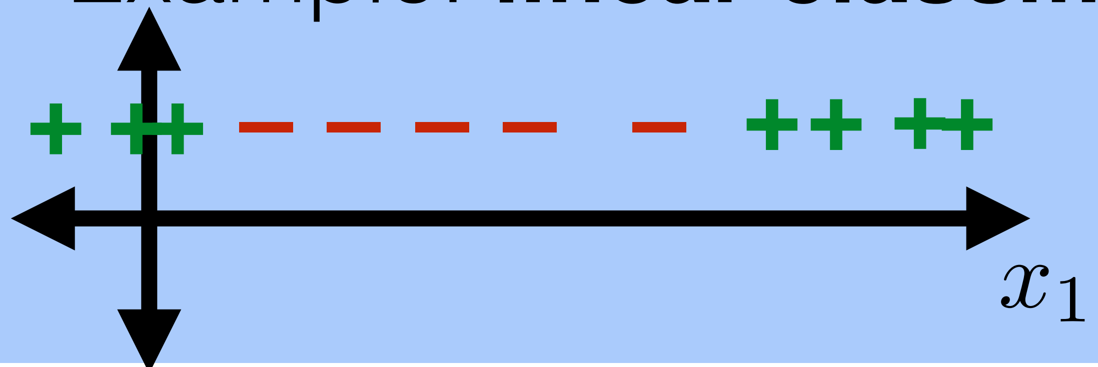
Machine Learning Tasks

- **Regression:** Learn a mapping to continuous values: $\mathbb{R}^d \rightarrow \mathbb{R}^k$

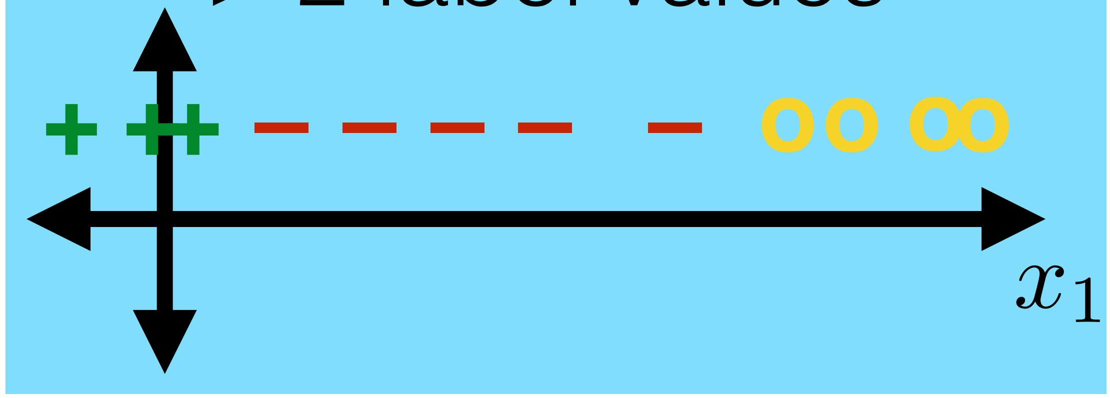


- **Classification:** Learn a mapping to a discrete set

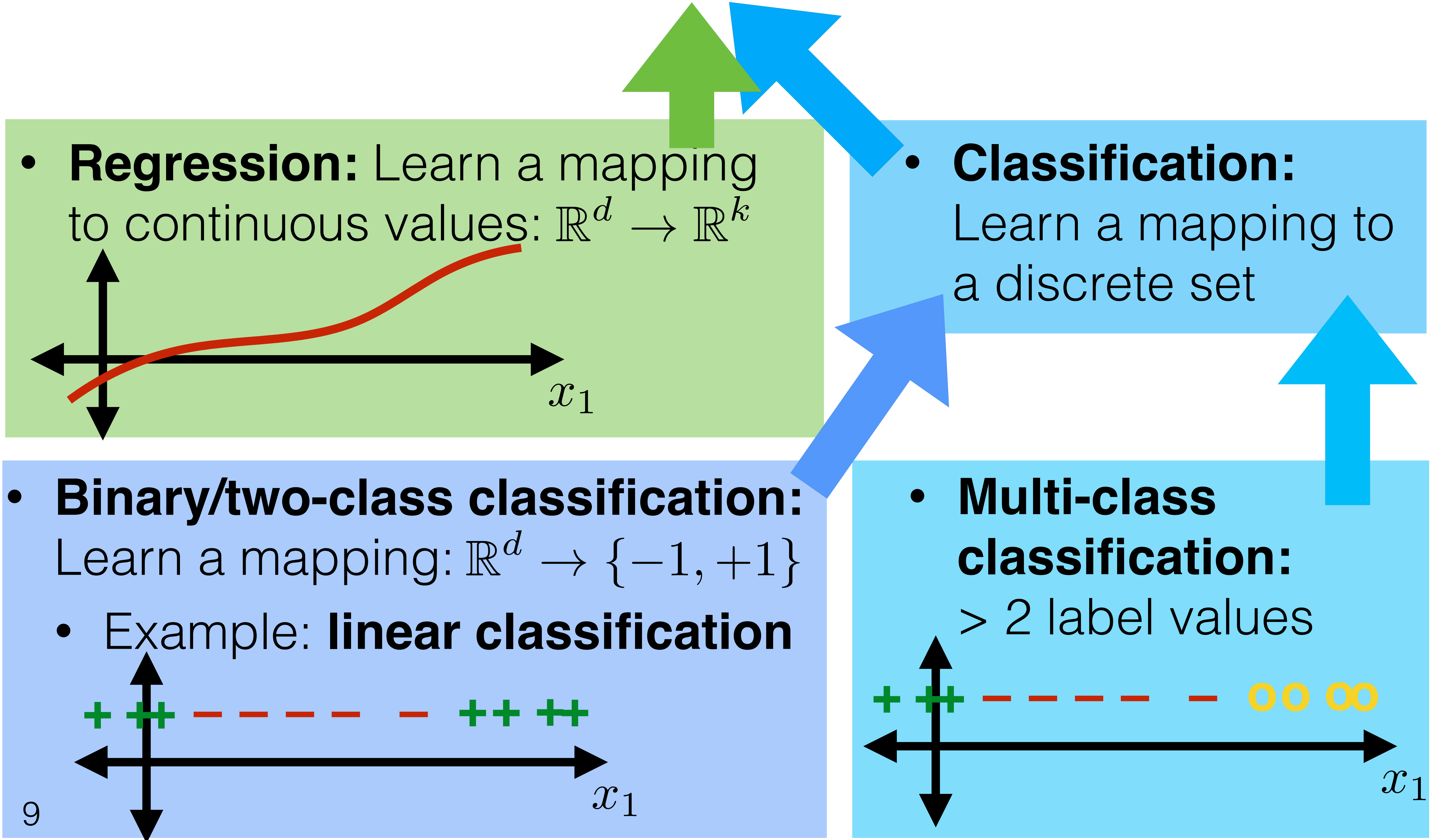
- **Binary/two-class classification:** Learn a mapping: $\mathbb{R}^d \rightarrow \{-1, +1\}$
  - Example: **linear classification**



- **Multi-class classification:** > 2 label values



Machine Learning Tasks



- **Regression:** Learn a mapping to continuous values: $\mathbb{R}^d \rightarrow \mathbb{R}^k$

- **Classification:** Learn a mapping to a discrete set

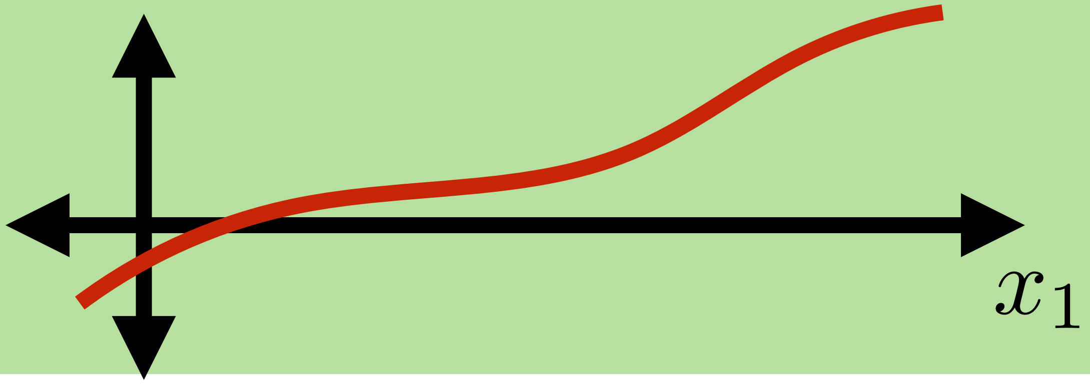
- **Binary/two-class classification:** Learn a mapping: $\mathbb{R}^d \rightarrow \{-1, +1\}$
  - Example: **linear classification**

- **Multi-class classification:** > 2 label values

Machine Learning Tasks

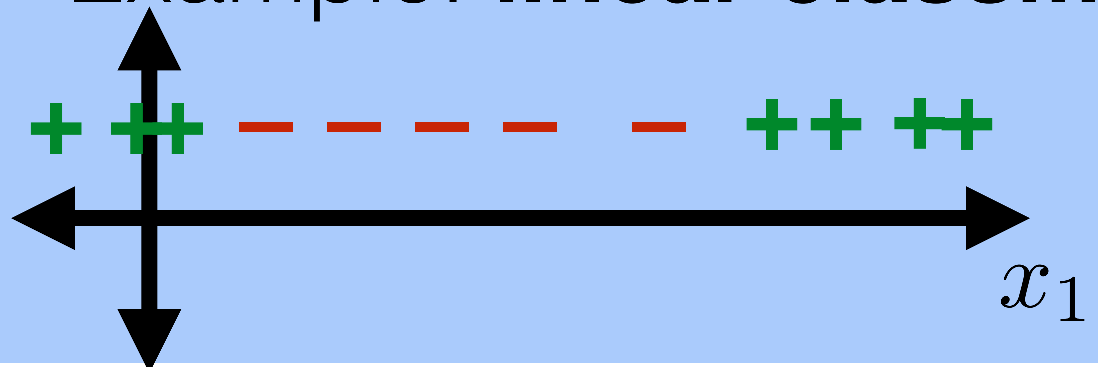
- **Supervised learning**

- **Regression:** Learn a mapping to continuous values: $\mathbb{R}^d \rightarrow \mathbb{R}^k$

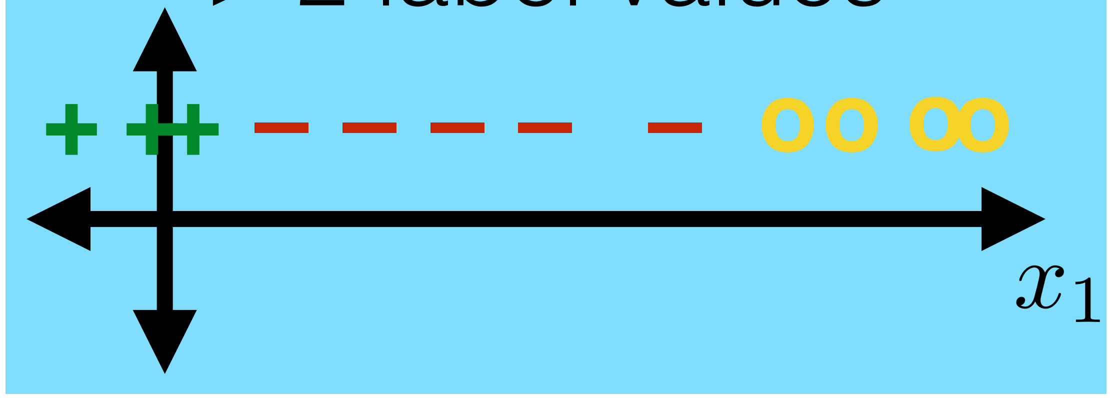


- **Classification:** Learn a mapping to a discrete set

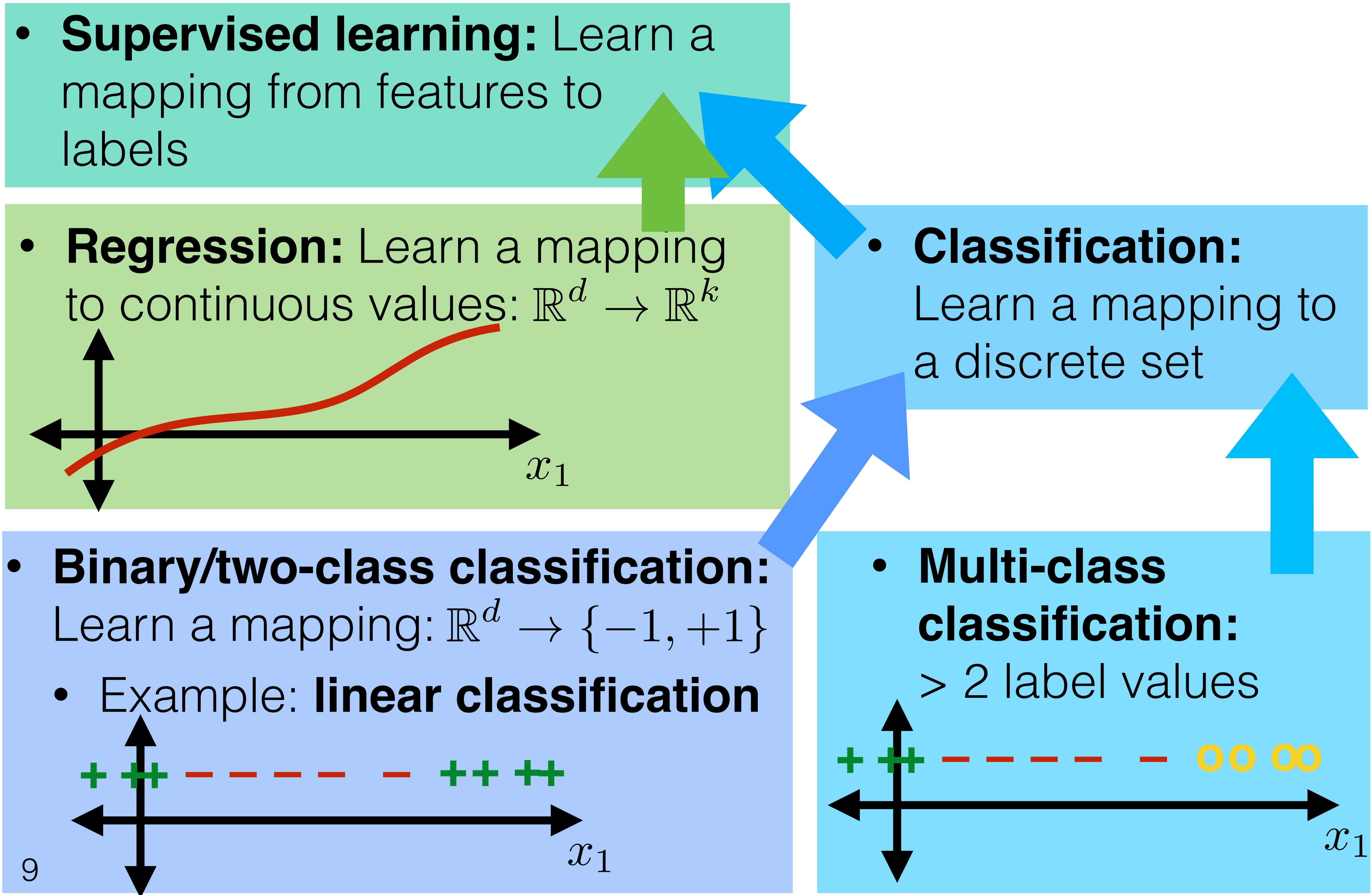
- **Binary/two-class classification:** Learn a mapping: $\mathbb{R}^d \rightarrow \{-1, +1\}$
  - Example: **linear classification**



- **Multi-class classification:** > 2 label values



Machine Learning Tasks



- **Supervised learning:** Learn a mapping from features to labels

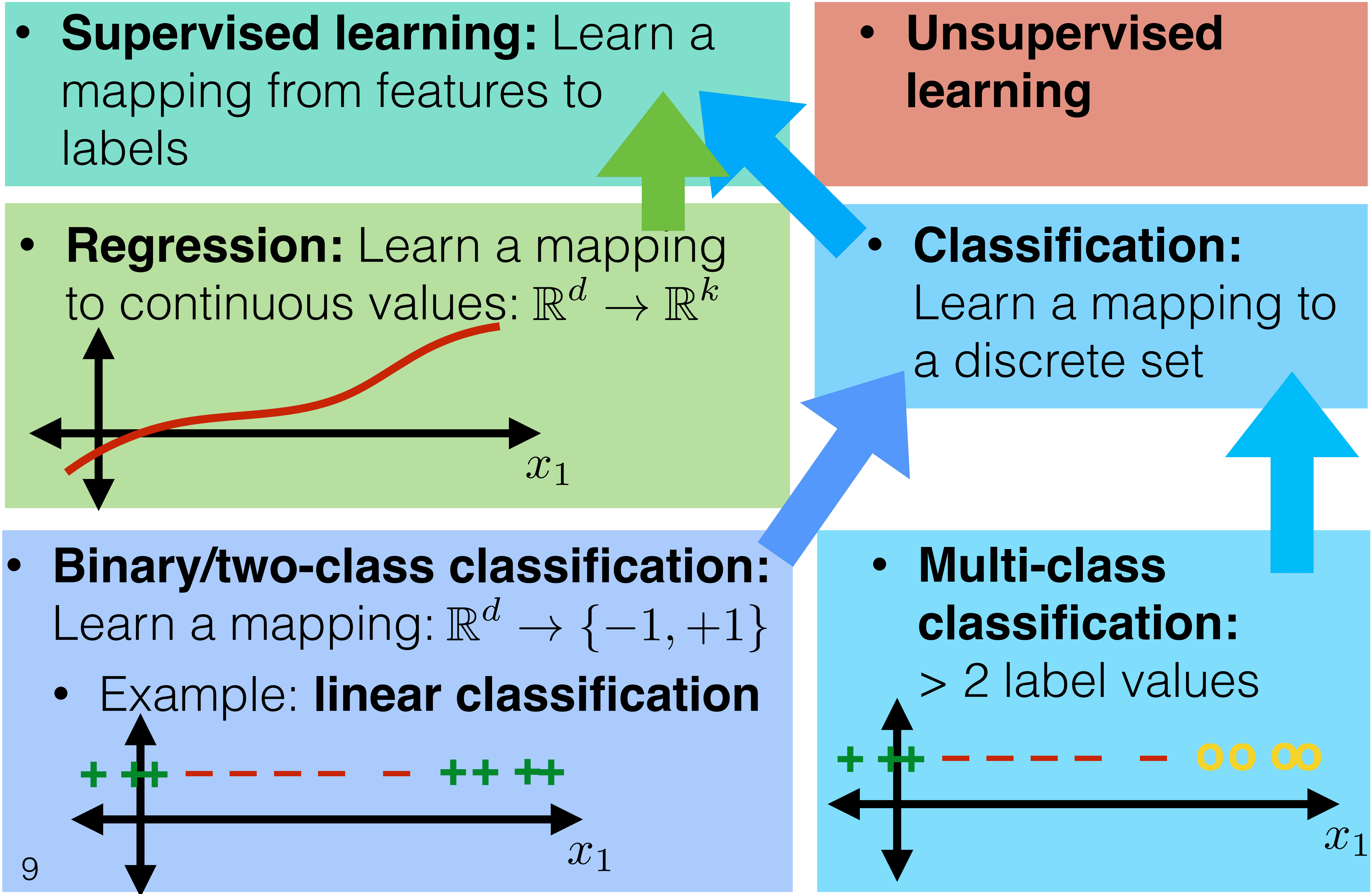
- **Regression:** Learn a mapping to continuous values: $\mathbb{R}^d \rightarrow \mathbb{R}^k$

- **Classification:** Learn a mapping to a discrete set

- **Binary/two-class classification:** Learn a mapping: $\mathbb{R}^d \rightarrow \{-1, +1\}$
  - Example: **linear classification**

- **Multi-class classification:** > 2 label values

Machine Learning Tasks



- **Supervised learning:** Learn a mapping from features to labels

- **Unsupervised learning**

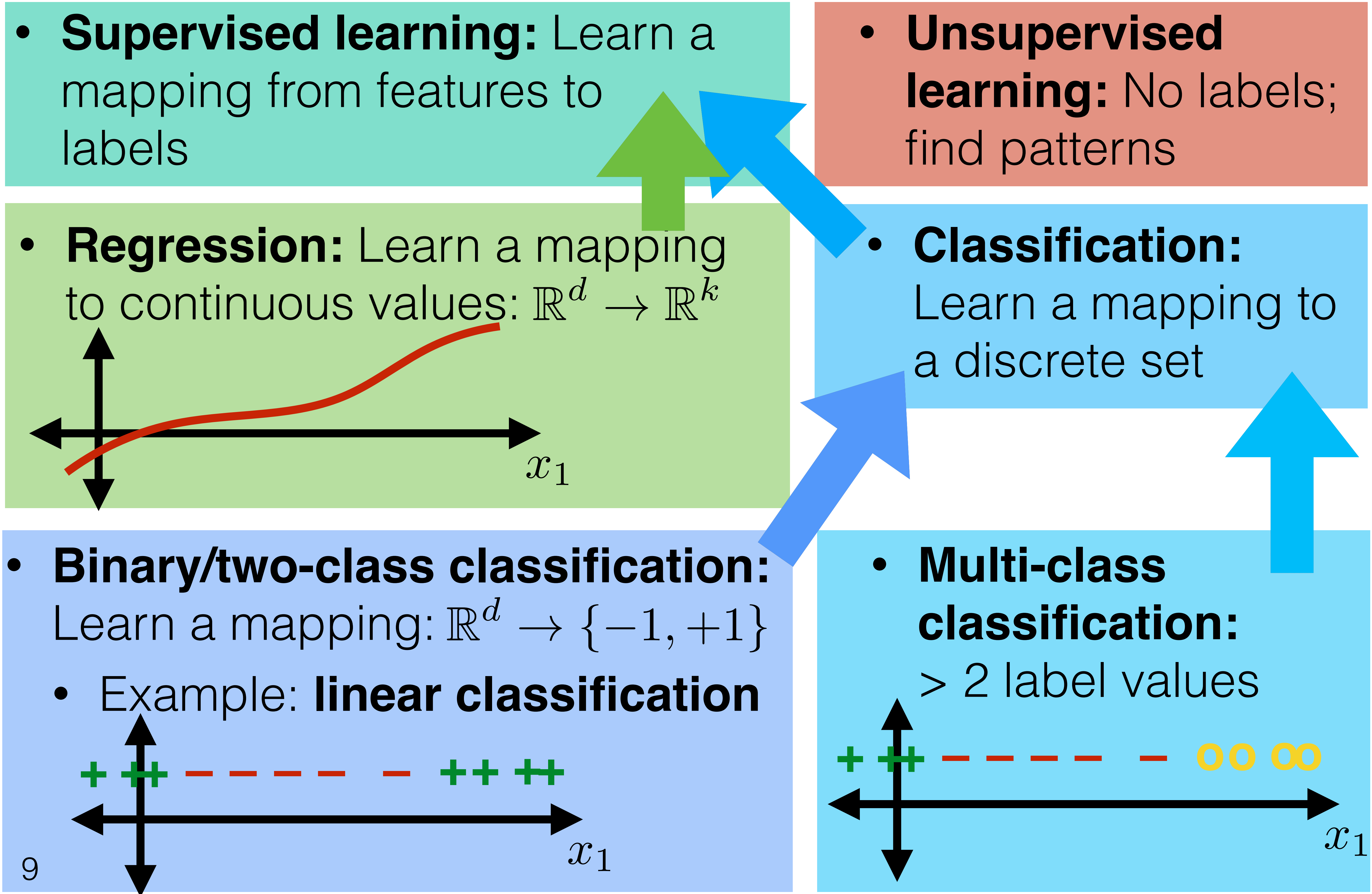
- **Regression:** Learn a mapping to continuous values: $\mathbb{R}^d \rightarrow \mathbb{R}^k$

- **Classification:** Learn a mapping to a discrete set

- **Binary/two-class classification:** Learn a mapping: $\mathbb{R}^d \rightarrow \{-1, +1\}$
  - Example: **linear classification**

- **Multi-class classification:** > 2 label values

Machine Learning Tasks



- **Supervised learning:** Learn a mapping from features to labels

- **Unsupervised learning:** No labels; find patterns

- **Regression:** Learn a mapping to continuous values: $\mathbb{R}^d \rightarrow \mathbb{R}^k$

- **Classification:** Learn a mapping to a discrete set

- **Binary/two-class classification:** Learn a mapping: $\mathbb{R}^d \rightarrow \{-1, +1\}$
  - Example: **linear classification**

- **Multi-class classification:** > 2 label values