MIT · 6.036 | Introduction to Machine Learning (2020)

MIT 6.036(2020)· 课程资料包 @ShowMeAI

视频	课件	笔记	代码
中英双语字幕	一键打包下载	官方笔记翻译	作业项目解析

视频 · B 站 [扫码或点击链接]
https://www.bilibili.com/video/BV1y44y187wN

课件 & 代码 · 博客 [扫码或点击链接]
http://blog.showmeai.tech/mit-6.036

机器学习 循环神经 神经网络 感知器 特征构建 聚类 马尔可夫决策过程 网络 随机森林 决策树 逻辑回归 卷积神经网络 状态机

Awesome AI Courses Notes Cheatsheets 是 ShowMeAI 资料库的分支系列，覆盖最具知名度的 **TOP50+** 门 AI 课程，旨在为读者和学习者提供一整套高品质中文学习笔记和速查表。

点击课程名称，跳转至课程**资料包**页面，**一键下载**课程全部资料！

机器学习	深度学习	自然语言处理	计算机视觉
Stanford · CS229	Stanford · CS230	Stanford · CS224n	Stanford · CS231n
# Awesome AI Courses Notes Cheatsheets· 持续更新中			
知识图谱	**图机器学习**	**深度强化学习**	**自动驾驶**
Stanford · CS520	Stanford · CS224W	UCBerkeley · CS285	MIT · 6.S094

微信公众号

资料下载方式 2：扫码点击底部菜单栏

称为 **AI 内容创作者？** 回复 [添砖加瓦]

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:

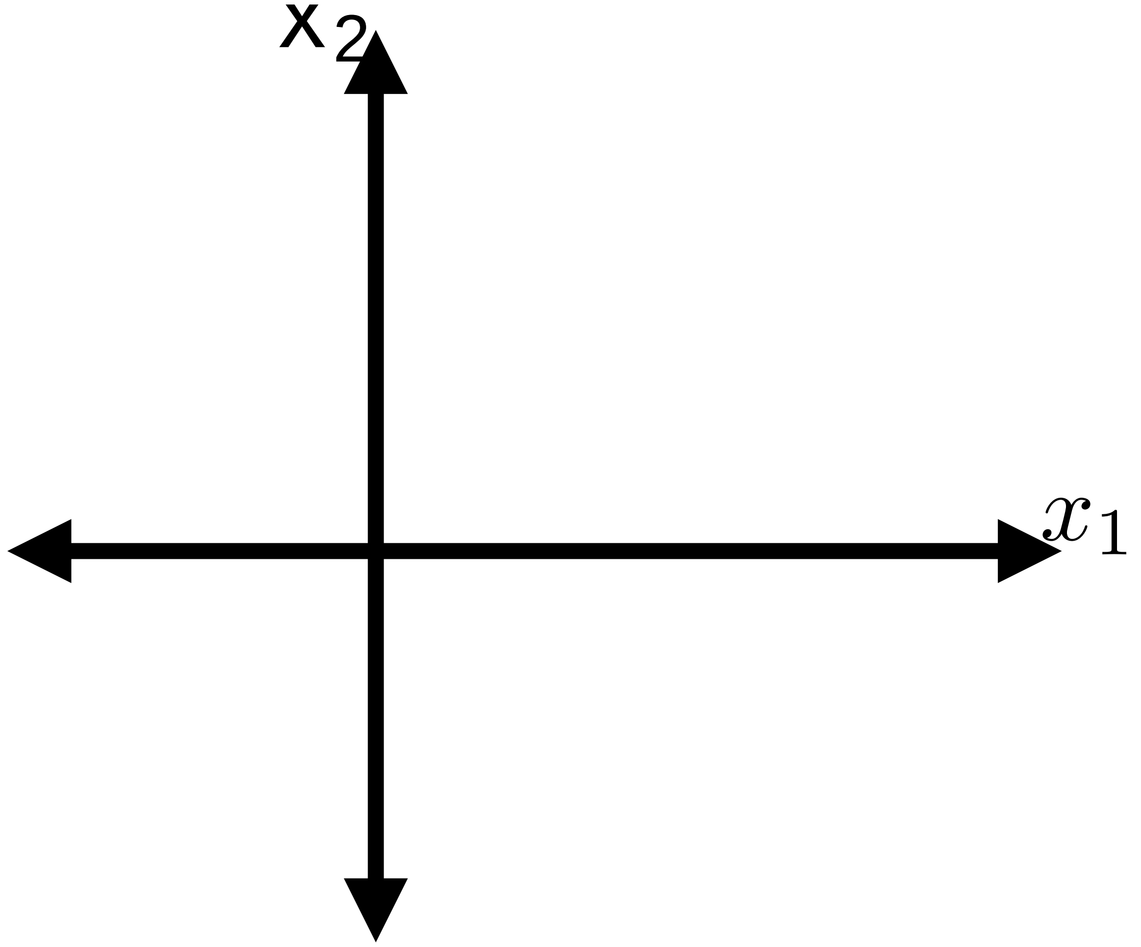
$$h(x; \theta, \theta_0) = \text{sign}(\theta^\top x + \theta_0)$$

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

Recall: Classifiers

- A linear classifier:

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

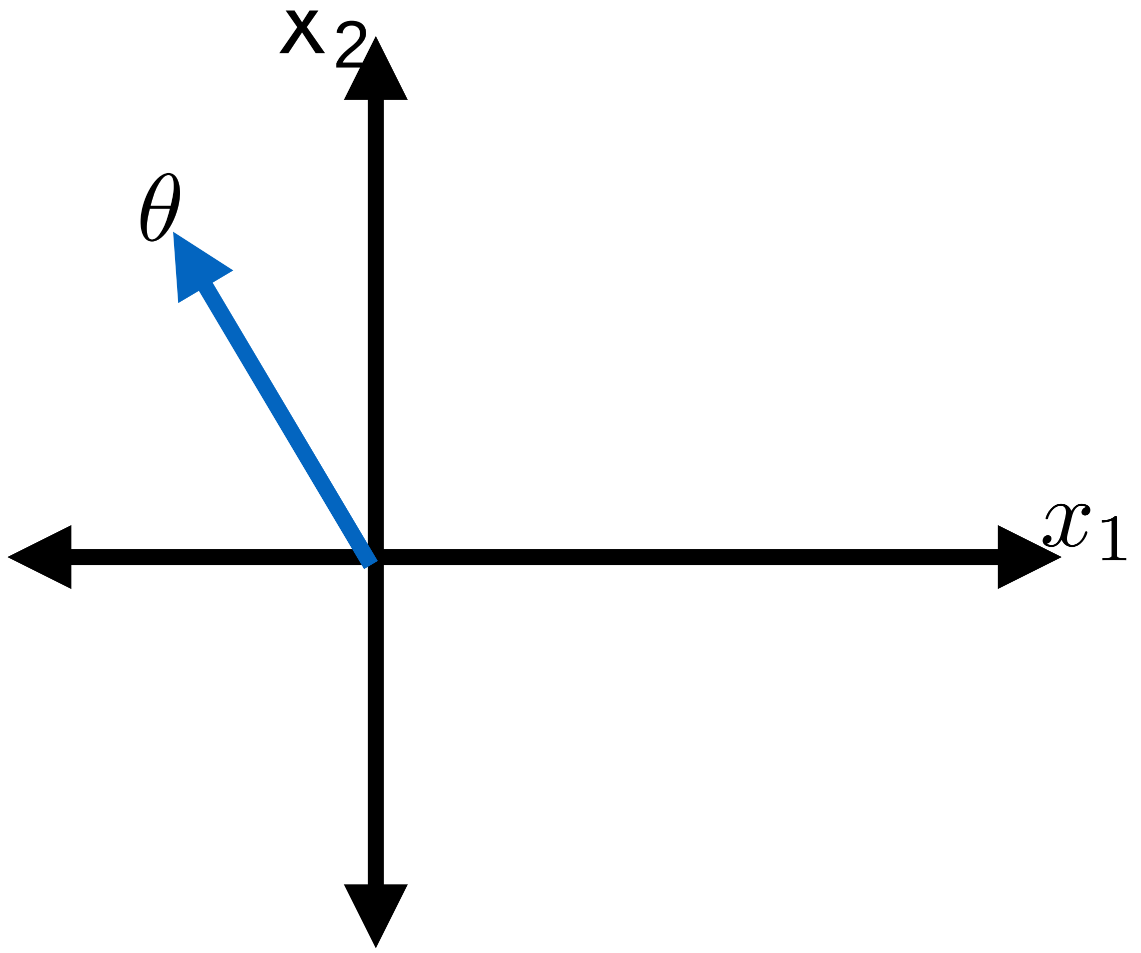


Recall: Classifiers

- A linear classifier:

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

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

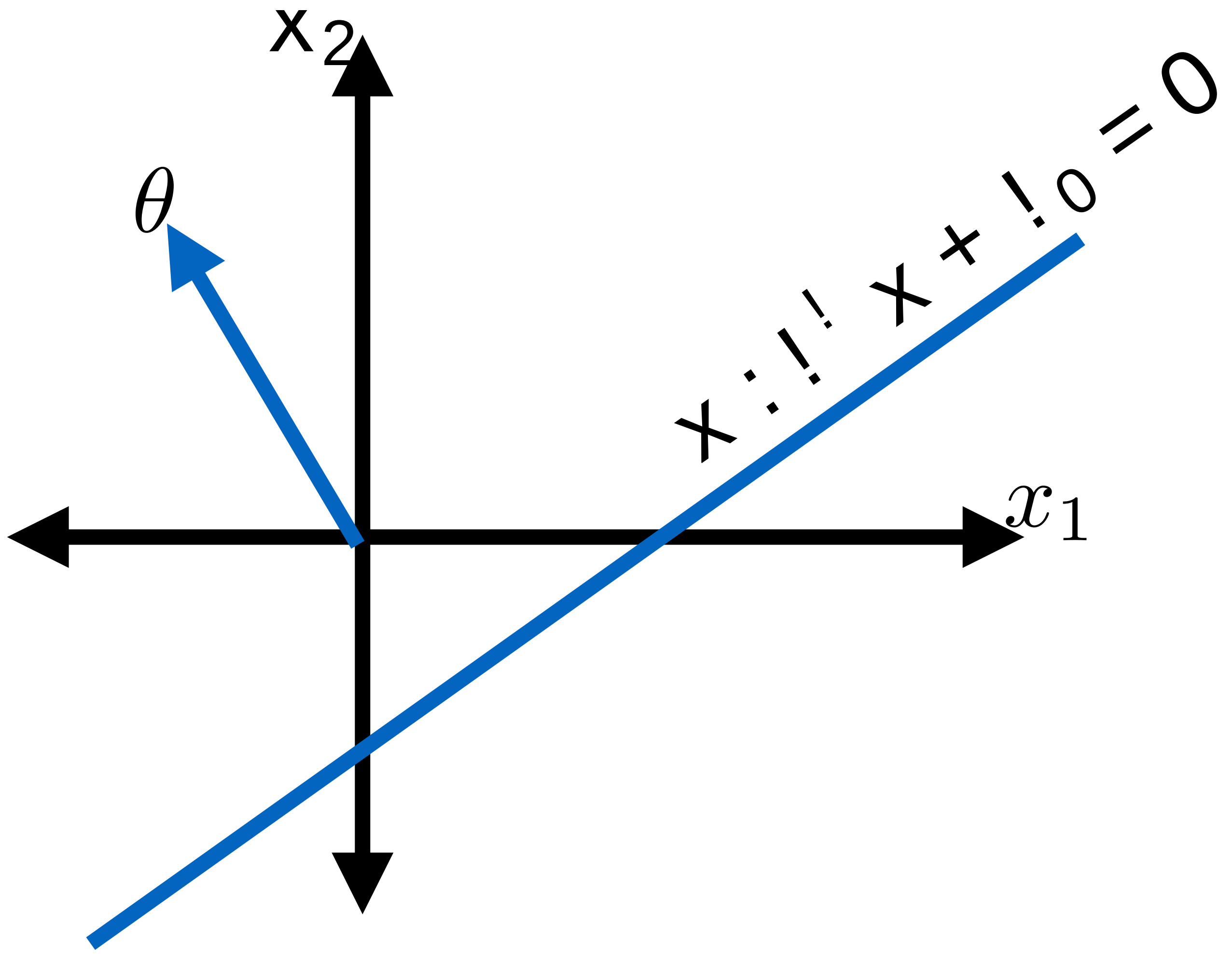


Recall: Classifiers

- A linear classifier:

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

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

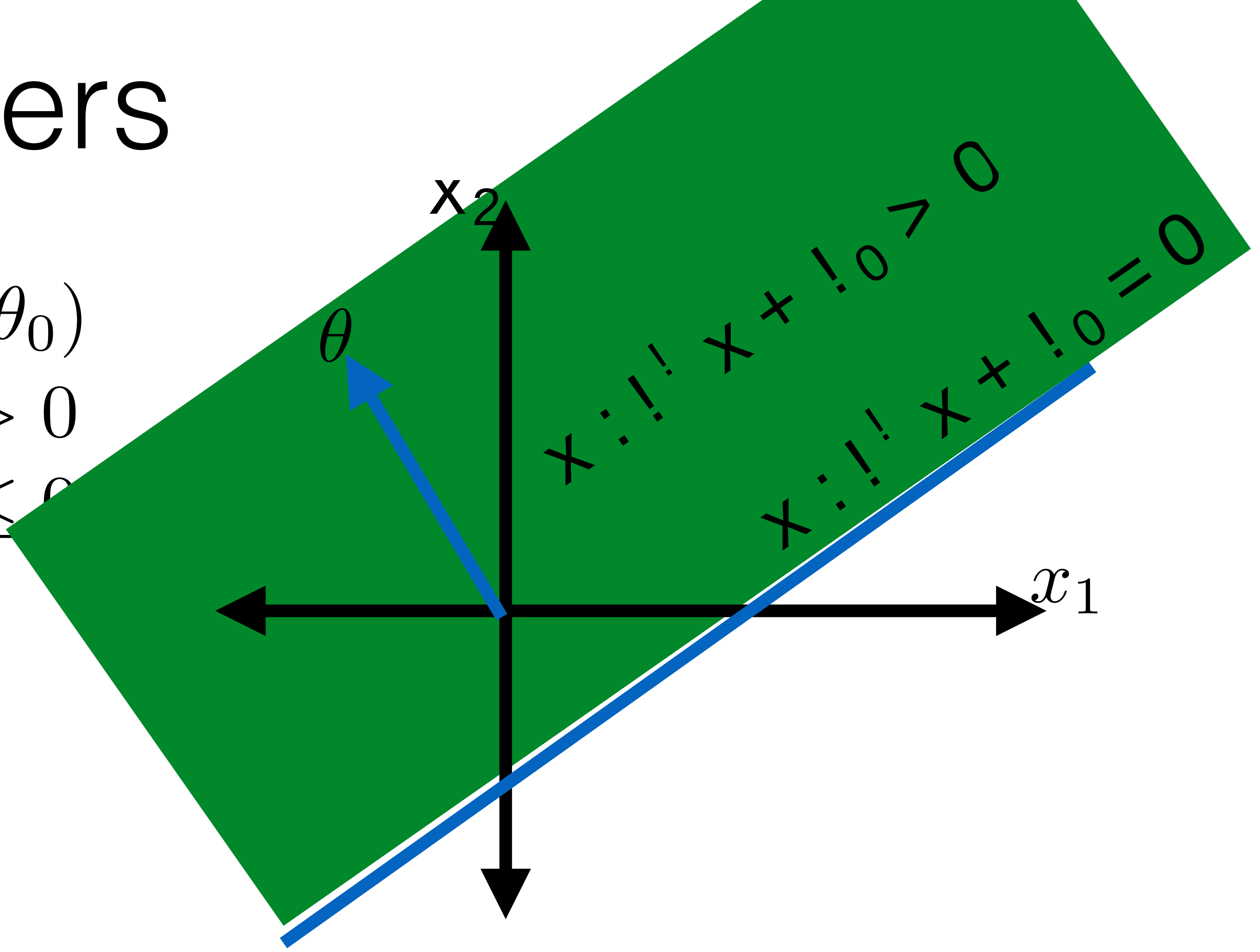


Recall: Classifiers

- A linear classifier:

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

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

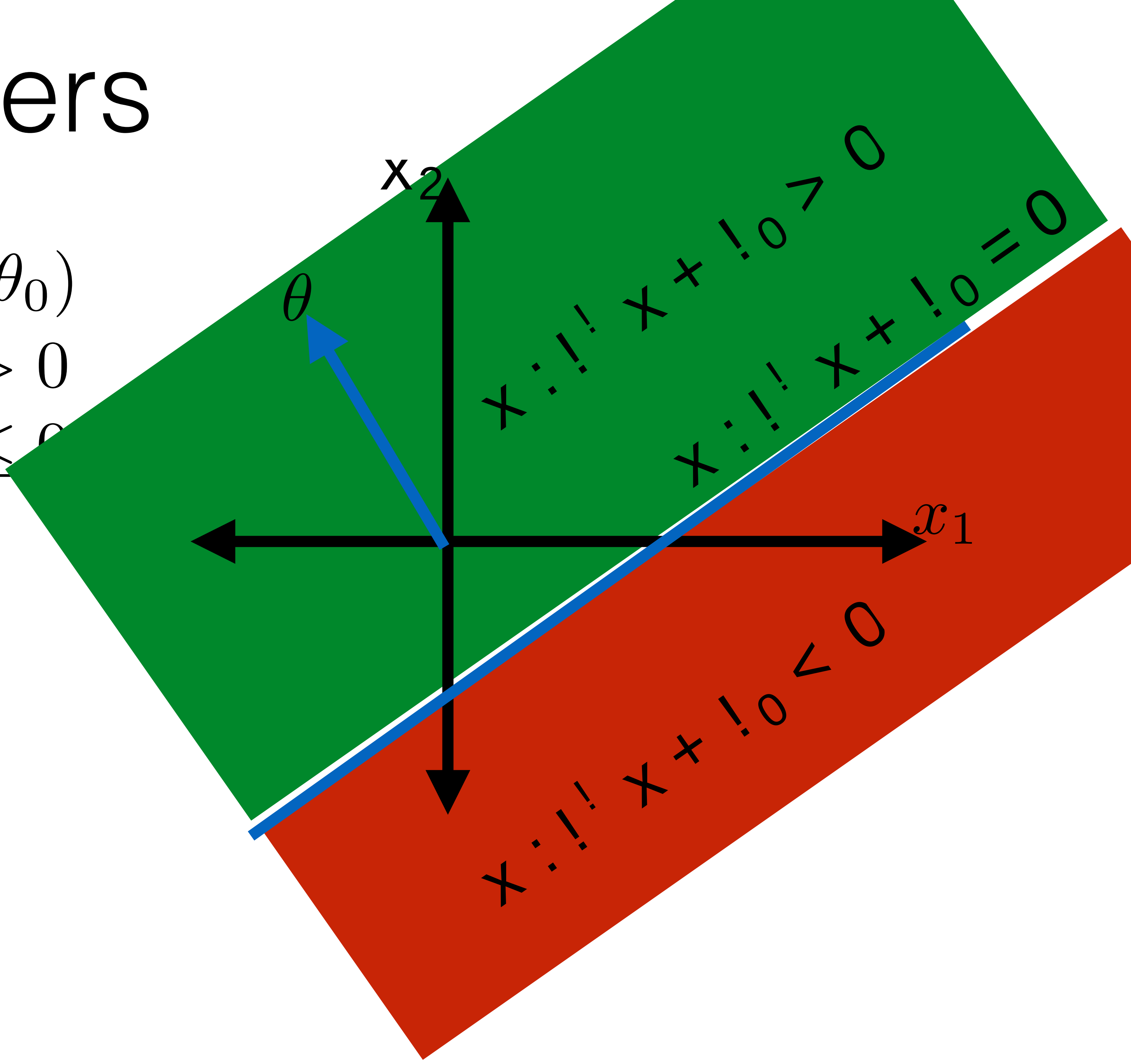


Recall: Classifiers

- A linear classifier:

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

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

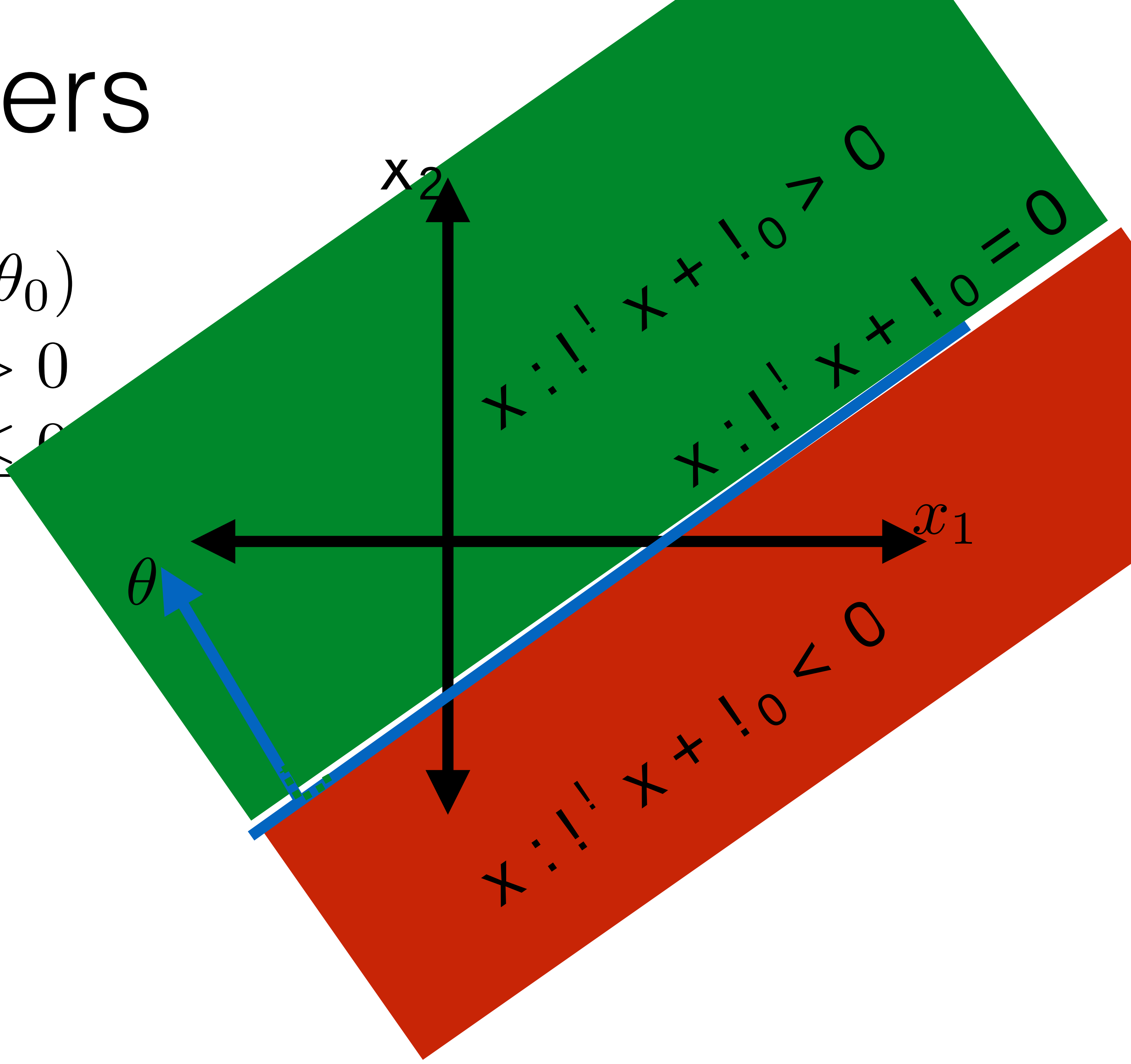


Recall: Classifiers

- A linear classifier:

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

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

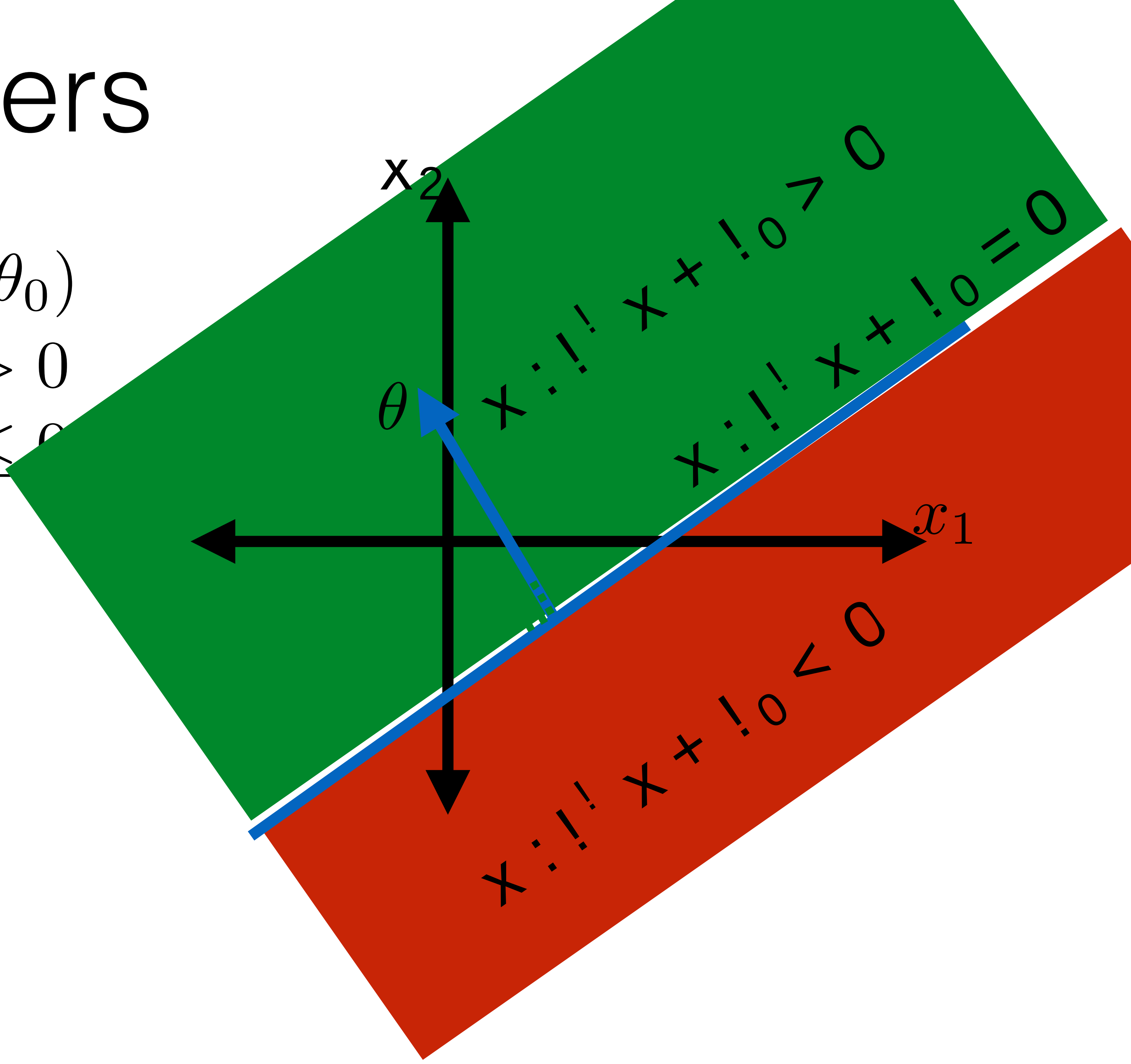


Recall: Classifiers

- A linear classifier:

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

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

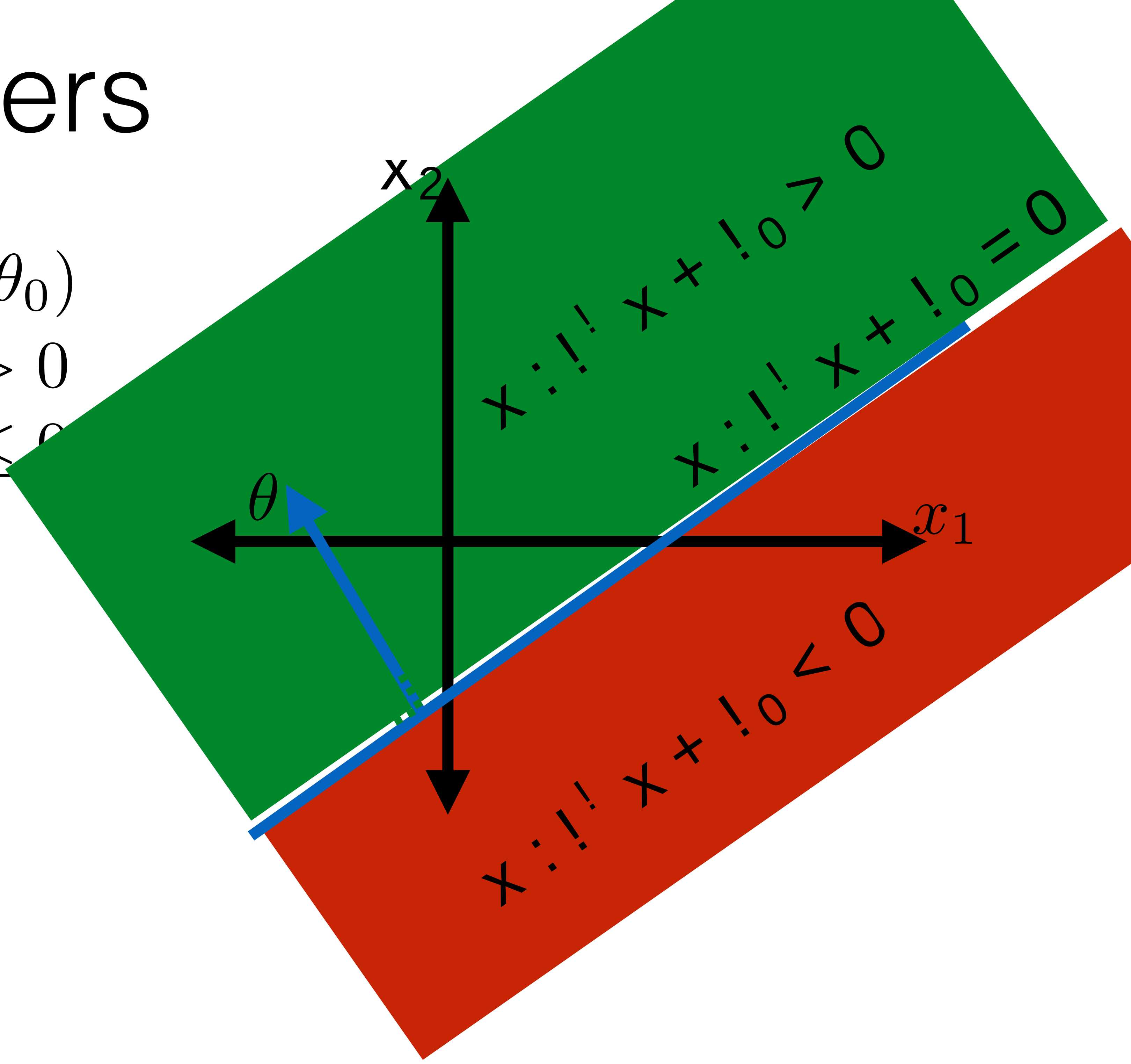


Recall: Classifiers

- A linear classifier:

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

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

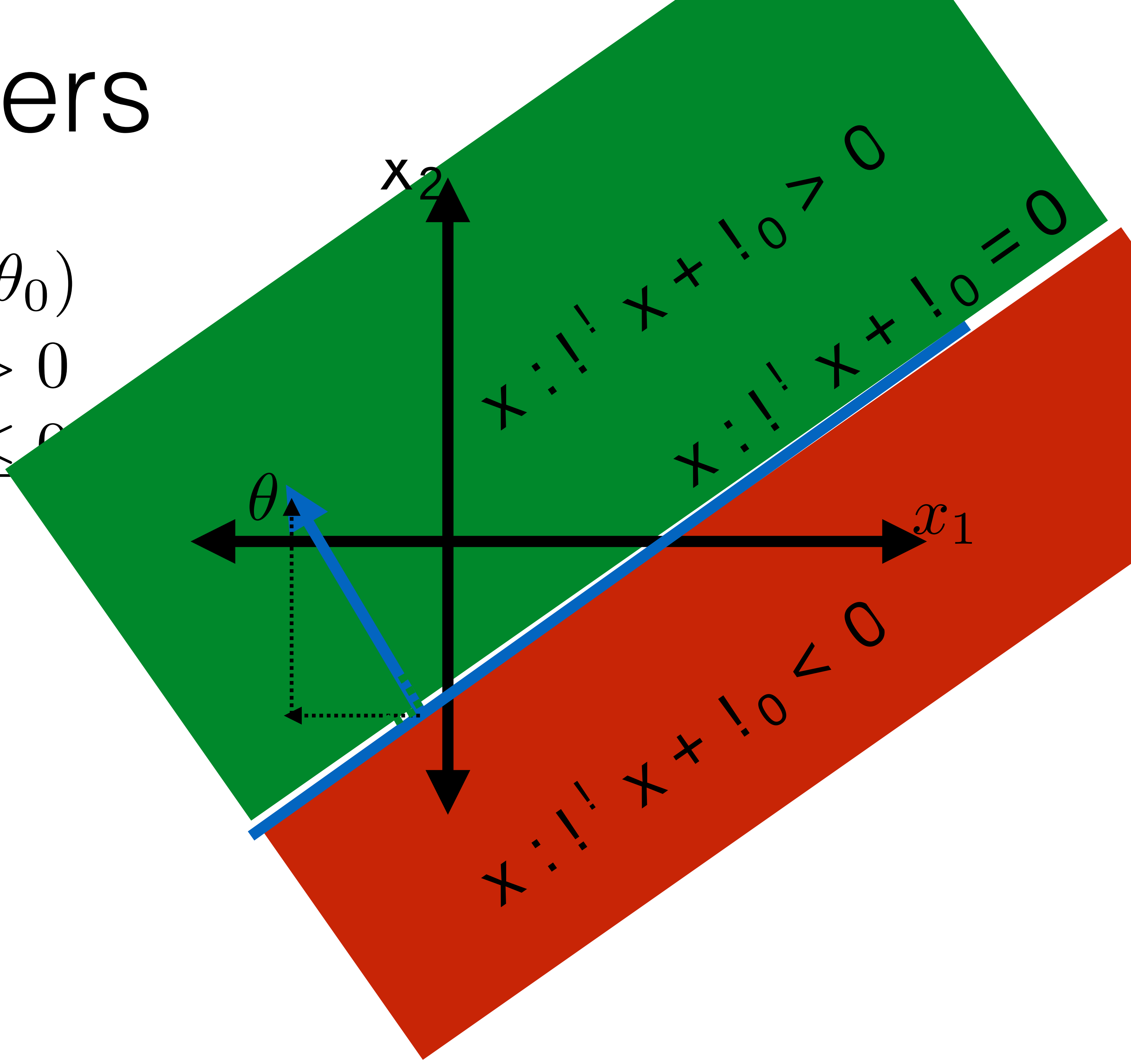


Recall: Classifiers

- A linear classifier:

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

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

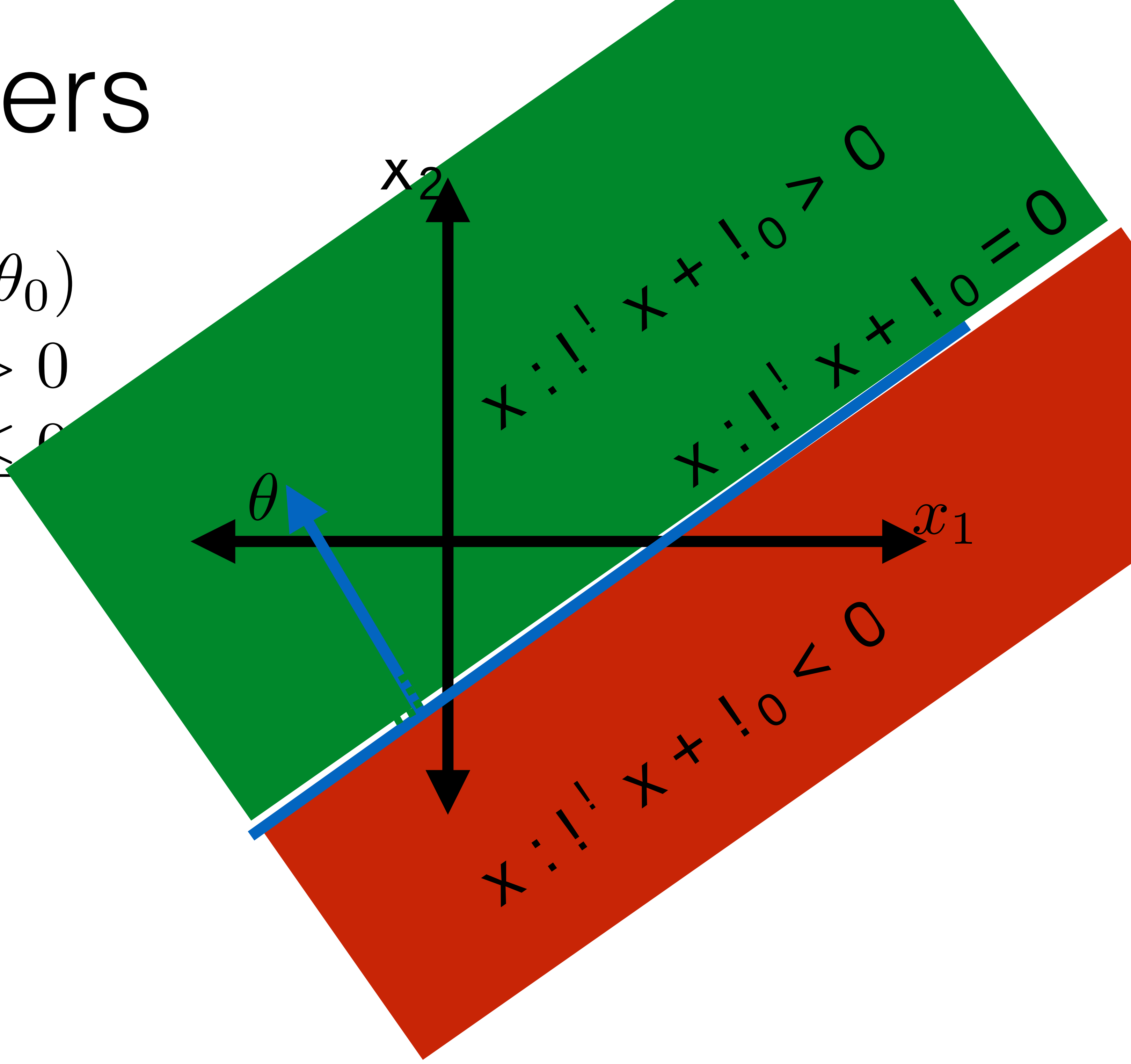


Recall: Classifiers

- A linear classifier:

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

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

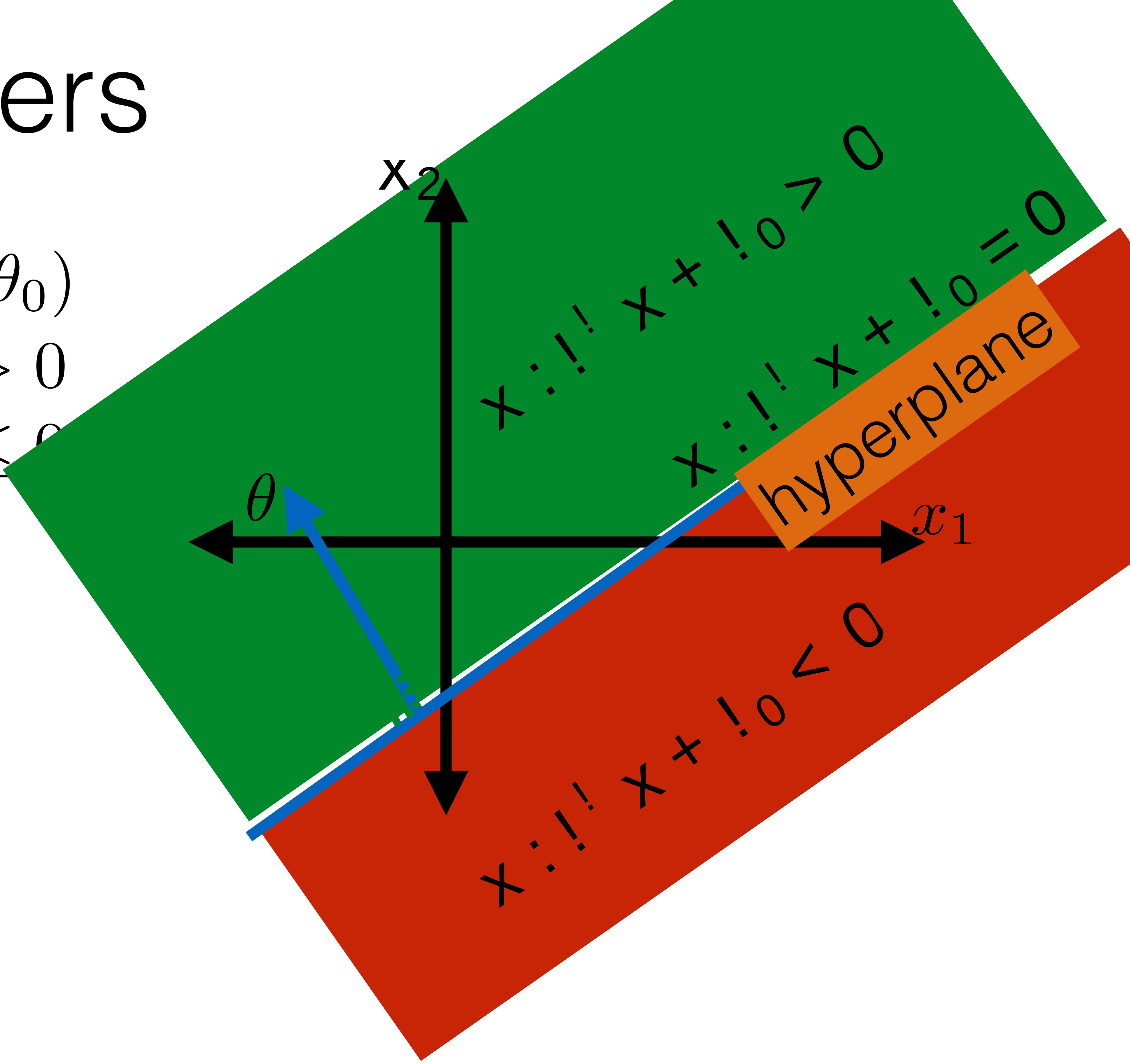


Recall: Classifiers

- A linear classifier:

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

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

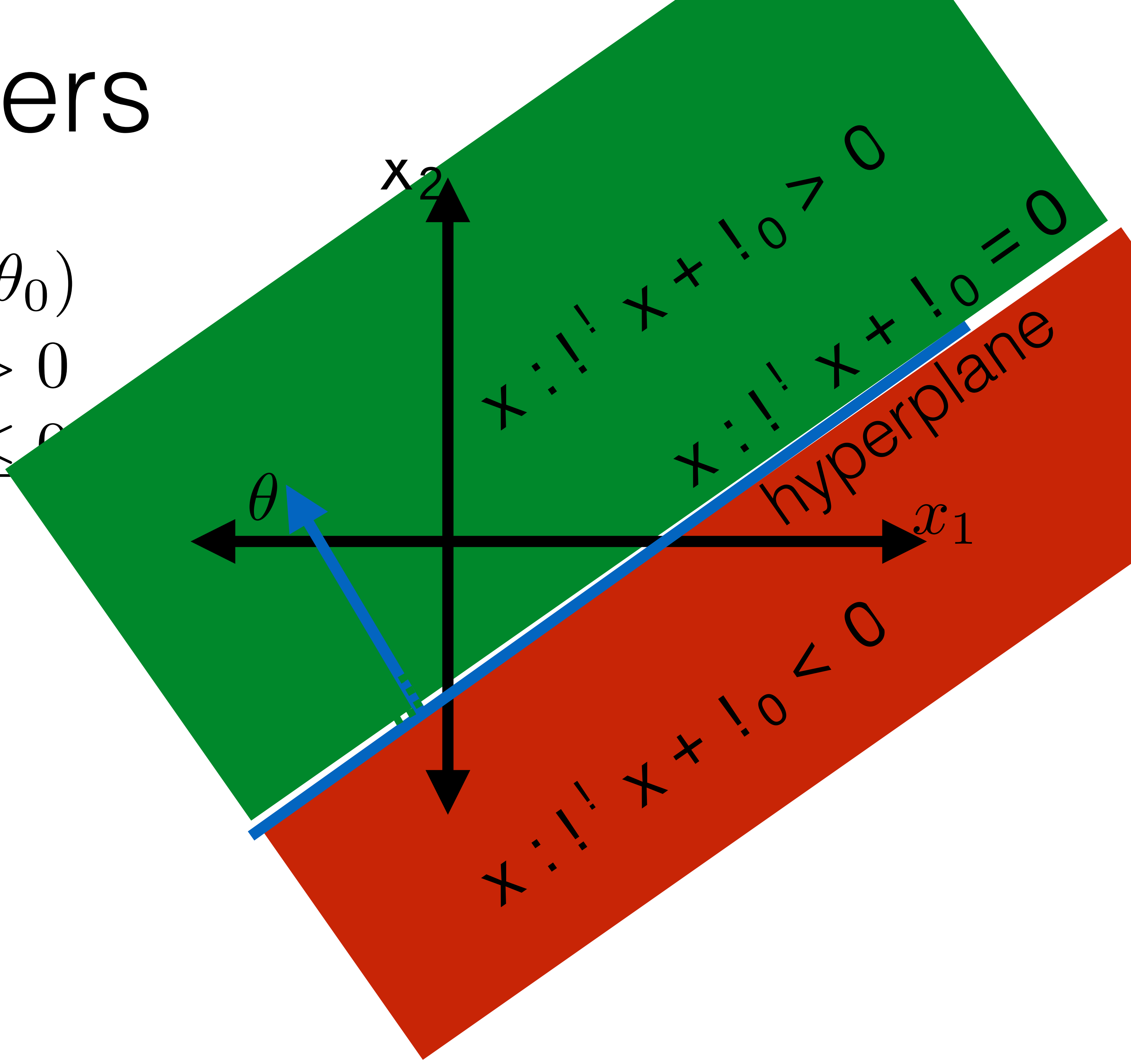


Recall: Classifiers

- A linear classifier:

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

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

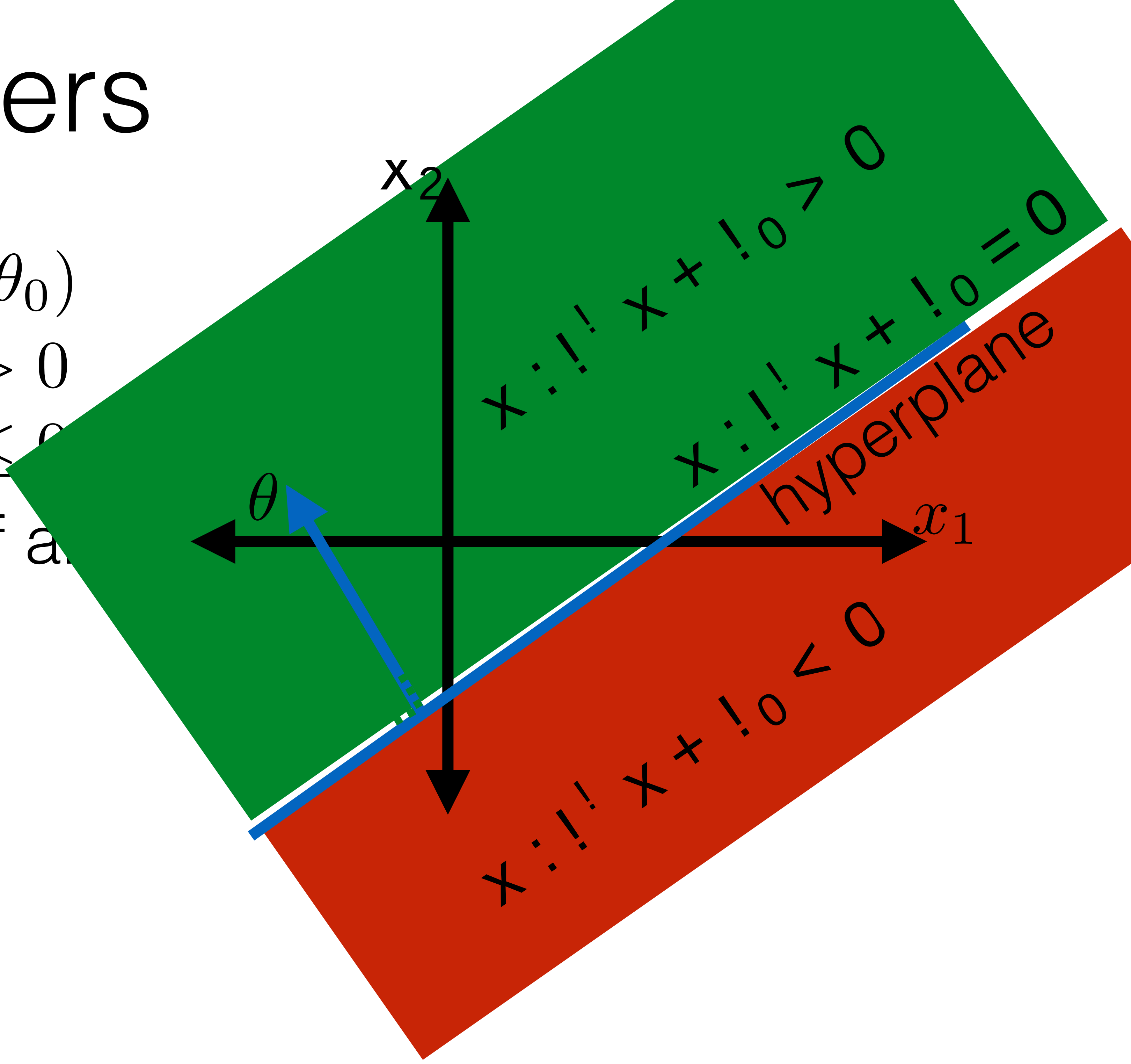


Recall: Classifiers

- A linear classifier:

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

- Hypothesis class **H** of a
  linear classifiers



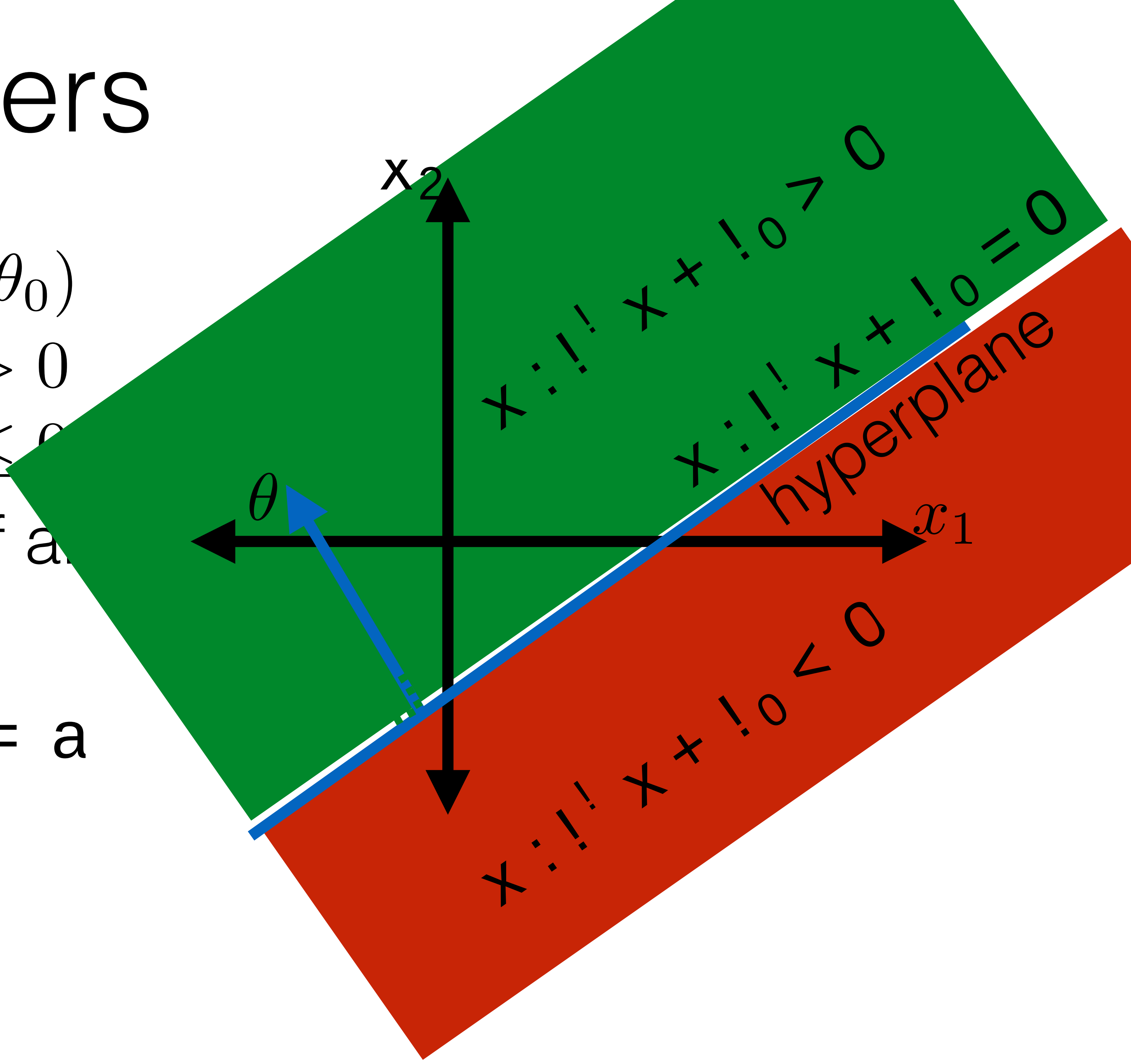
Recall: Classifiers

- A linear classifier:

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

- Hypothesis class H of a linear classifiers
- 0-1 Loss

L(g, a) = 0 if g = a
1 else



Recall: Classifiers

- A linear classifier:

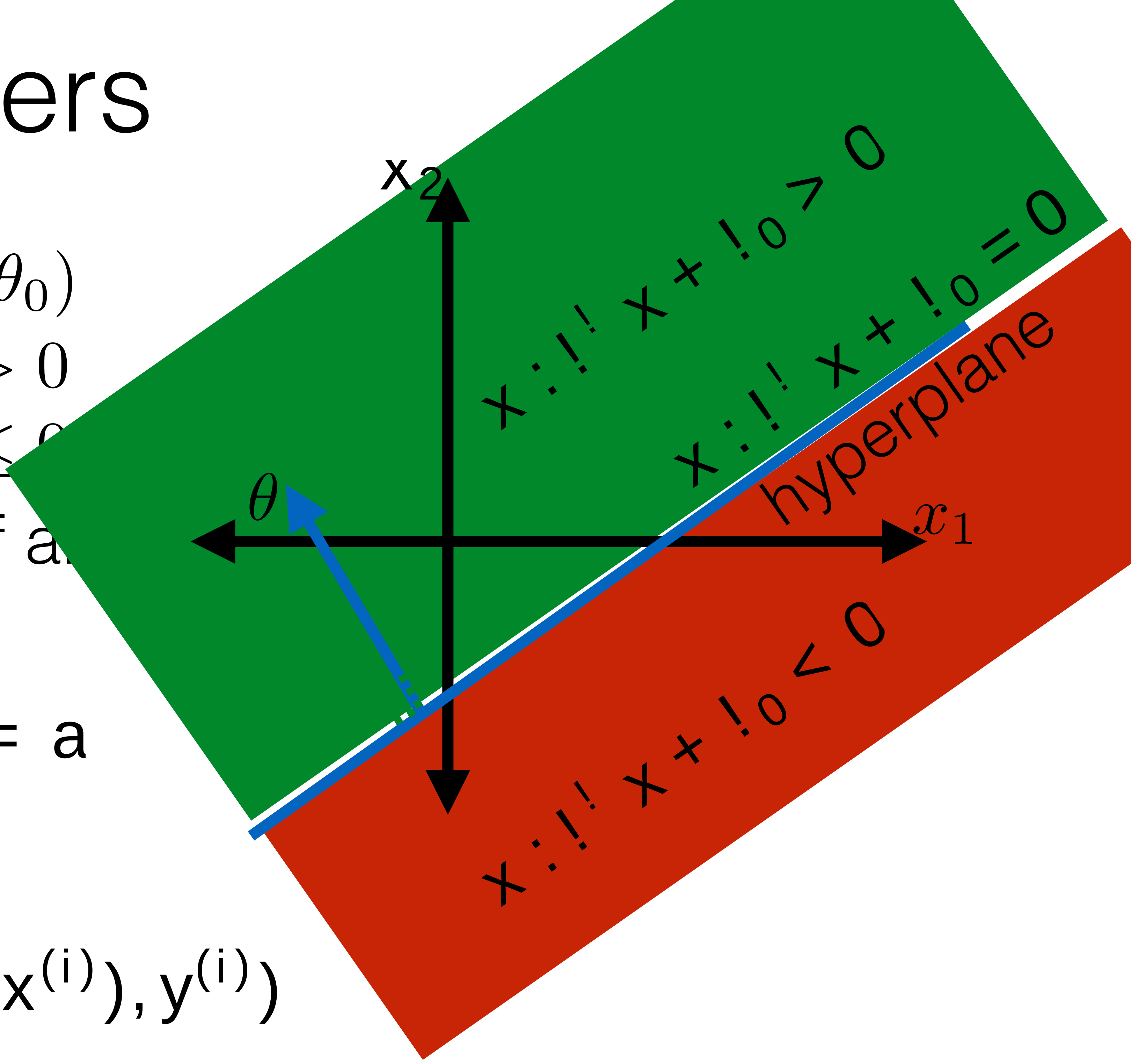
$$h(x;\theta,\theta_0) = \text{sign}(\theta^\top x + \theta_0)$$
$$= \begin{cases} +1 \text{ if } \theta^\top x + \theta_0 > 0 \\ -1 \text{ if } \theta^\top x + \theta_0 \leq 0 \end{cases}$$

- Hypothesis class H of a
  linear classifiers
- 0-1 Loss

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

- Training error

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



Recall: Classifiers

- A linear classifier:

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

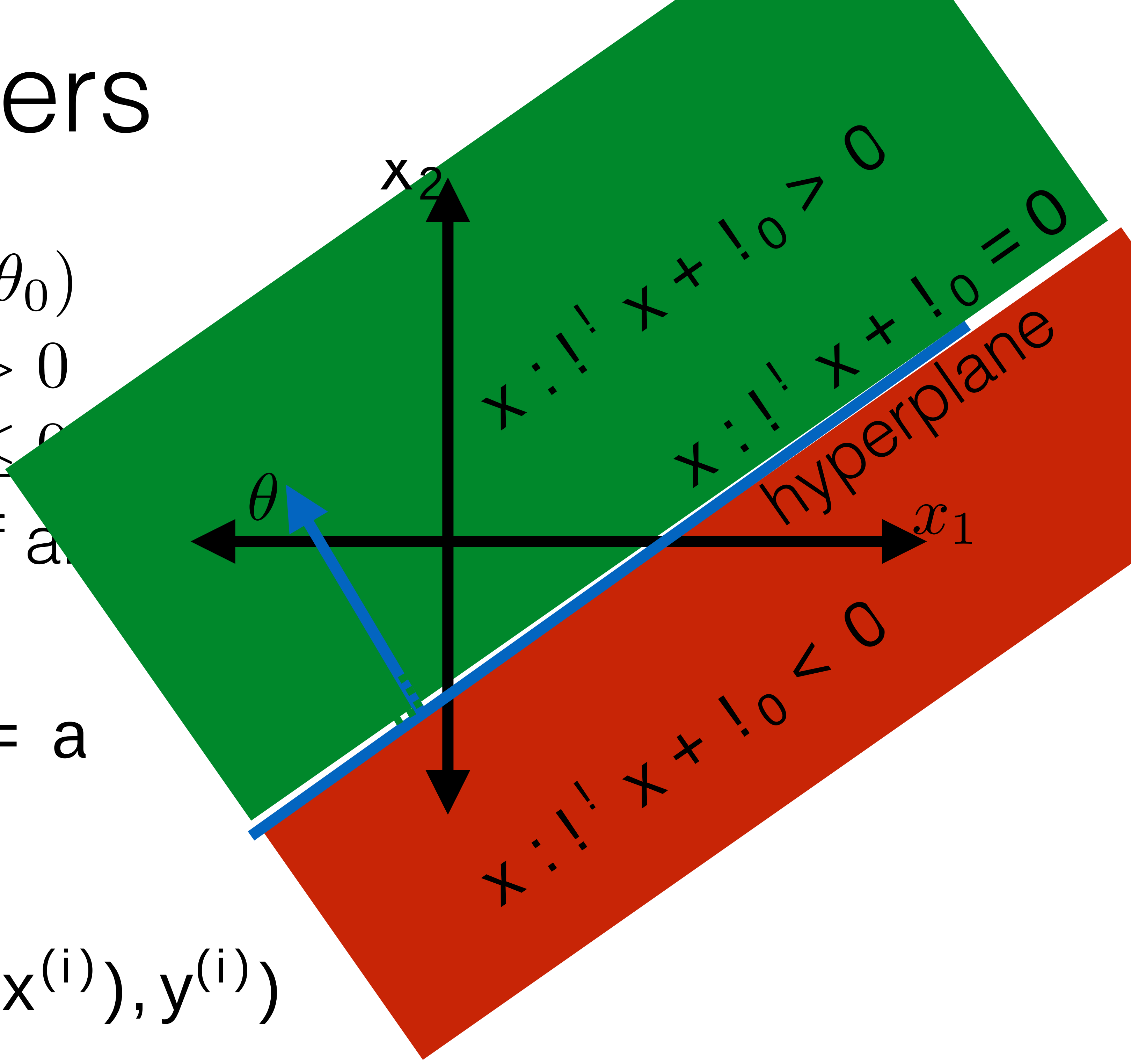
- Hypothesis class H of a
  linear classifiers
- 0-1 Loss

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

- Training error

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

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



Recall: Classifiers

- A linear classifier:

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

- Hypothesis class H of a
  linear classifiers
- 0-1 Loss

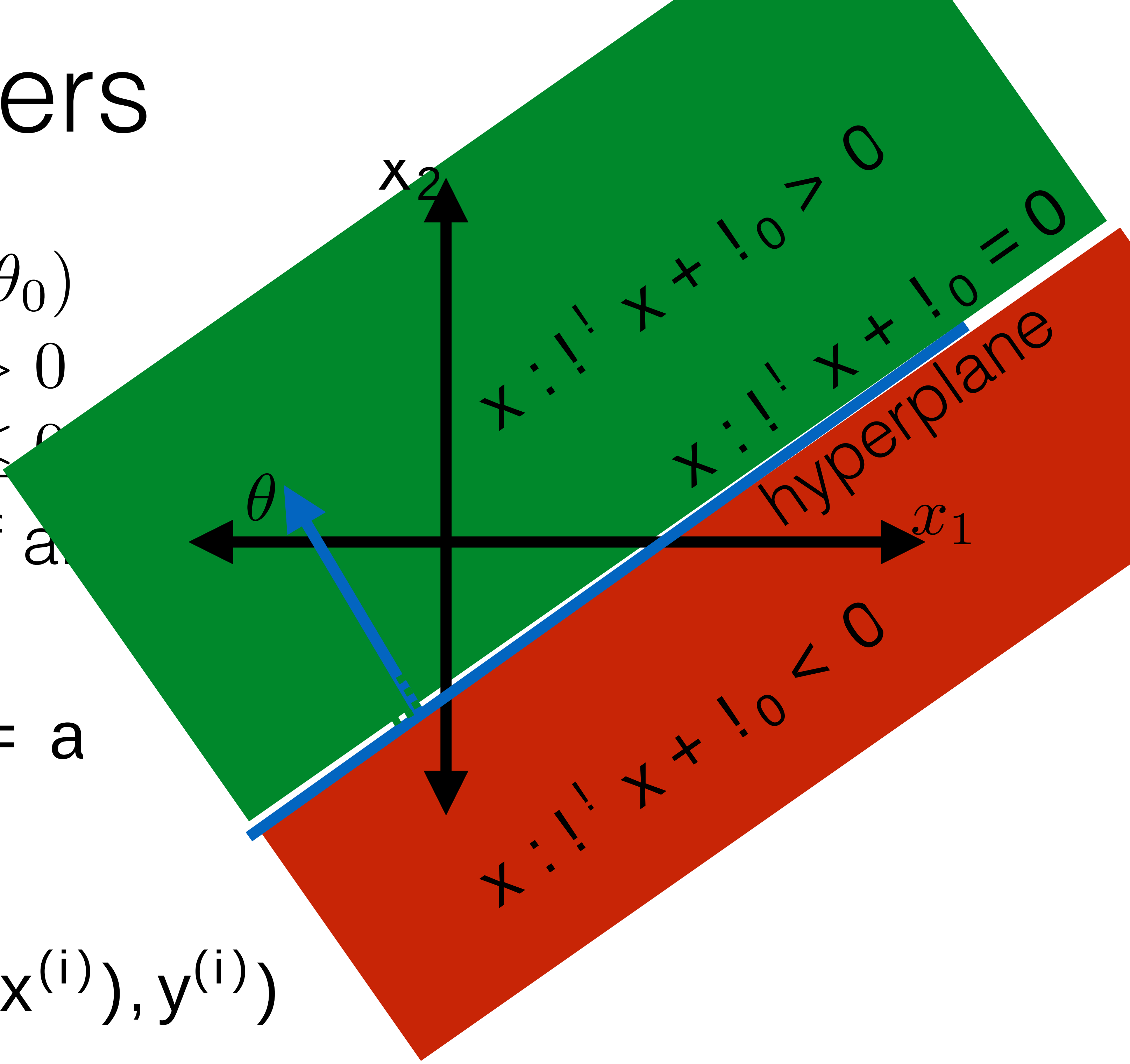
$$L(g, a) = \begin{cases} 0 \text{ if } g = a \\ 1 \text{ else} \end{cases}$$

- Training error

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

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

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



Recall: Classifiers

- A linear classifier:

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

- Hypothesis class H of a
  linear classifiers
- 0-1 Loss

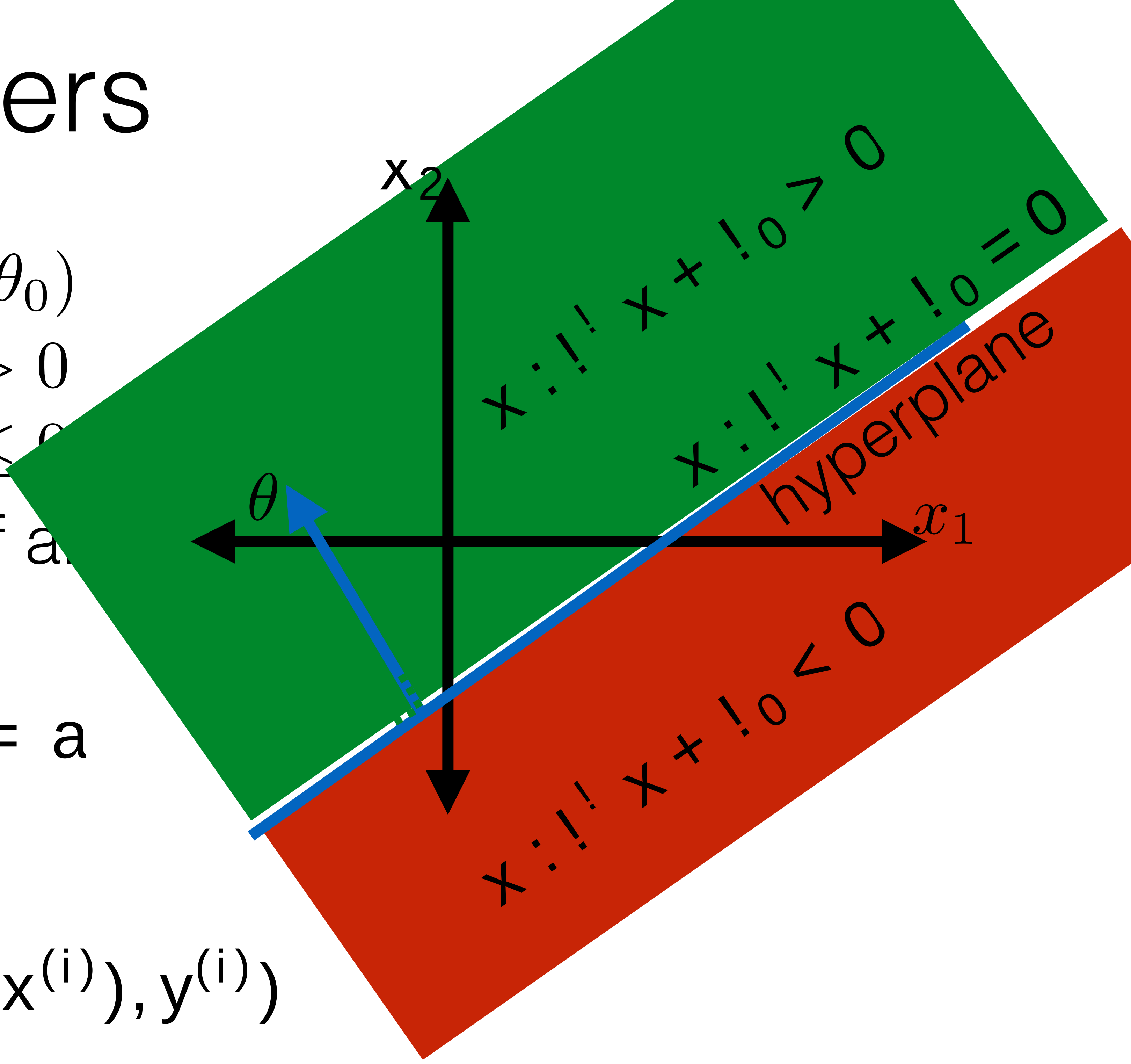
$$L(g, a) = \begin{cases} 0 \text{ if } g = a \\ 1 \text{ else} \end{cases}$$

- Training error

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

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

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



[demo]

Perceptron Algorithm

Perceptron Algorithm

```
Perceptron
```

Perceptron Algorithm

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

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; ! )
    Initialize ! = [0  0  …  0]$^{!}$
    Initialize $!_0$ = 0
```

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; ! )
  Initialize  ! = [0  0  …  0]$^!$        [How many 0s?]
  Initialize  $!_0$ = 0
```

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; ! )
  Initialize ! = [0  0  …  0]$^!$        [How many 0s?]
  Initialize $!_0$ = 0
  **for** t = 1 to !
```

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; ! )
  Initialize ! = [0  0  …  0]$^!$        [How many 0s?]
  Initialize $!_0$ = 0
  for t = 1 to !
    changed = False
```

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; ! )
  Initialize ! = [0  0  …  0]$^!$        [How many 0s?]
  Initialize $!_0$ = 0
  for t = 1 to !
    changed = False
    for i = 1 to n
```

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; ! )
  Initialize ! = [0 0 … 0]$^!$      [How many 0s?]
  Initialize $!_0$ = 0
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0)$ ! 0
```

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; ! )
  Initialize ! = [0  0  …  0]$^!$        [How many 0s?]
  Initialize $!_0$ = 0
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0) ! 0$
```

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; ! )
  Initialize ! = $[0 \; 0 \; \ldots \; 0]^!$      [How many 0s?]
  Initialize $!_0 = 0$
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0) ! 0$
```

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; ! )
  Initialize ! = [0  0  …  0]$^!$        [How many 0s?]
  Initialize $!_0$ = 0
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0) !$ 0
```

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; ! )
  Initialize ! = [0  0  …  0]$^!$        [How many 0s?]
  Initialize $!_0$ = 0
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0) ! \; 0$
```

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; ! )
  Initialize ! = [0  0  …  0]!      [How many 0s?]
  Initialize $!_0$ = 0
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^{!} x^{(i)} + !_0) !\ 0$
```

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; ! )
  Initialize ! = [0  0  …  0]$^!$
  Initialize $!_0$ = 0
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0)$ ! 0
```

[How many 0s?]

[i.e. True if either:

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; ! )
  Initialize  ! = [0  0  …  0]!
  Initialize  !$_0$ = 0
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}$(!$^!$ $x^{(i)}$ + !$_0$) !  0
```

[How many 0s?]

[i.e. True if either:

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

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; ! )
  Initialize ! = [0 0 … 0]$^!$
  Initialize $!_0$ = 0
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0)$ ! 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( $\mathcal{D}_n$ ; ! )
  Initialize ! = [0 0 … 0]$^!$
  Initialize !$_0$ = 0
  for t = 1 to !
    changed = False
    for i = 1 to n
      if y$^{(i)}$(!$^!$ x$^{(i)}$ + !$_0$) ! 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( $\mathcal{D}_n$ ; ! )
  Initialize ! = [0 0 … 0]$^!$
  Initialize $!_0$ = 0
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0) !$ 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( $\mathcal{D}_n$ ; ! )
  Initialize $! = [0\ 0\ \ldots\ 0]^!$
  Initialize $!_0 = 0$
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0) !\ 0$
```

[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( $\mathcal{D}_n$ ; ! )
  Initialize ! = [0 0 … 0]$^!$
  Initialize $!_0 = 0$
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0) !\ 0$
        Set $! = ! + y^{(i)}x^{(i)}$
        Set $!_0 = !_0 + y^{(i)}$
```

[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( $\mathcal{D}_n$ ; ! )
  Initialize $! = [0\ 0\ \ldots\ 0]^!$
  Initialize $!_0 = 0$
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0) !\ 0$
        Set $! = ! + y^{(i)}x^{(i)}$
        Set $!_0 = !_0 + y^{(i)}$
        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( $\mathcal{D}_n$ ; ! )
  Initialize $! = [0 \ 0 \ \ldots \ 0]^!$
  Initialize $!_0 = 0$
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0) ! \ 0$
        Set $! = ! + y^{(i)}x^{(i)}$
        Set $!_0 = !_0 + y^{(i)}$
        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( $\mathcal{D}_n$ ; ! )
  Initialize $! = [0\ 0\ \ldots\ 0]^!$
  Initialize $!_0 = 0$
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0) !\ 0$
        Set $! = ! + y^{(i)} x^{(i)}$
        Set $!_0 = !_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return $!, !_0$
```

[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( $\mathcal{D}_n$ ; ! )
  Initialize $! = [0 \ 0 \ \ldots \ 0]^!$
  Initialize $!_0 = 0$
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0) ! \ 0$
        Set $! = ! + y^{(i)} x^{(i)}$
        Set $!_0 = !_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return $!, !_0$
```

[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( $\mathcal{D}_n$ ; ! )
  Initialize $! = [0\ 0\ \ldots\ 0]^!$
  Initialize $!_0 = 0$
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0) ! 0$
        Set $! = ! + y^{(i)} x^{(i)}$
        Set $!_0 = !_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return $!, !_0$
```

[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( $\mathcal{D}_n$ ; ! )
  Initialize ! = [0  0  …  0]$^!$
  Initialize $!_0 = 0$
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0) !\ 0$
        Set $! = ! + y^{(i)}x^{(i)}$
        Set $!_0 = !_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return $!, !_0$
```

[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( $\mathcal{D}_n$ ; ! )
  Initialize  $! = [0\ \ 0\ \ \ldots\ \ 0]^{!}$
  Initialize  $!_0 = 0$
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^{!}\ x^{(i)} + !_0)\ !\ \ 0$
        Set $! = ! + y^{(i)}x^{(i)}$
        Set $!_0 = !_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return $!, !_0$
```

[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( $\mathcal{D}_n$ ; ! )
  Initialize $! = [0\ 0\ \ldots\ 0]^!$
  Initialize $!_0 = 0$
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0) ! 0$
        Set $! = ! + y^{(i)} x^{(i)}$
        Set $!_0 = !_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return $!, !_0$
```

[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?

$y^{(i)}\ .\ !^!_{\text{updated}}\ x^{(i)} + (\ !_{0,\text{updated}}$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; ! )
  Initialize $! = [0 \ 0 \ \ldots \ 0]^!$
  Initialize $!_0 = 0$
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0)$ ! 0
        Set $! = ! + y^{(i)}x^{(i)}$
        Set $!_0 = !_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return $!, !_0$
```

[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?

$$y^{(i)} \cdot (! + y^{(i)}x^{(i)})^! x^{(i)} + (!_0 + y^{(i)})$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; ! )
  Initialize $! = [0\ \ 0\ \ \ldots\ \ 0]^!$
  Initialize $!_0 = 0$
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0)\ !\ 0$
        Set $! = ! + y^{(i)}x^{(i)}$
        Set $!_0 = !_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return $!, !_0$
```

[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?

$$y^{(i)}\ \ (! + y^{(i)}x^{(i)})^! x^{(i)} + (!_0 + y^{(i)})$$

$$= y^{(i)}(!^! x^{(i)} + !_0) + (y^{(i)})^2(x^{(i)!} x^{(i)} + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; ! )
  Initialize $! = [0\ \ 0\ \ \ldots\ \ 0]^!$
  Initialize $!_0 = 0$
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0)\ !\ \ 0$
        Set $! = ! + y^{(i)}x^{(i)}$
        Set $!_0 = !_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return $!, !_0$
```

[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?

$$y^{(i)} \cdot (! + y^{(i)}x^{(i)})^! \ x^{(i)} + (!_0 + y^{(i)})$$

$$= y^{(i)}(!^! \ x^{(i)} + !_0) + (y^{(i)})^2(x^{(i)!} \ x^{(i)} + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; ! )
  Initialize $! = [0\ 0\ \ldots\ 0]^!$
  Initialize $!_0 = 0$
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0)\ !\ 0$
        Set $! = ! + y^{(i)}x^{(i)}$
        Set $!_0 = !_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return $!, !_0$
```

[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?

$$y^{(i)} \cdot (! + y^{(i)}x^{(i)})^! x^{(i)} + (!_0 + y^{(i)})$$

$$= y^{(i)}(!^! x^{(i)} + !_0) + (y^{(i)})^2(x^{(i)!} x^{(i)} + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; ! )
  Initialize $! = [0 \;\; 0 \;\; \ldots \;\; 0]^!$
  Initialize $!_0 = 0$
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0) ! \; 0$
        Set $! = ! + y^{(i)}x^{(i)}$
        Set $!_0 = !_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return $!, !_0$
```

[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?

$$y^{(i)} \cdot \left((! + y^{(i)}x^{(i)})^! \, x^{(i)} + (!_0 + y^{(i)})\right)$$

$$= y^{(i)}(!^! \, x^{(i)} + !_0) + (y^{(i)})^2(x^{(i)!} \, x^{(i)} + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; ! )
  Initialize $! = [0\ \ 0\ \ \ldots\ \ 0]^!$
  Initialize $!_0 = 0$
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0)\ !\ \ 0$
        Set $! = ! + y^{(i)}x^{(i)}$
        Set $!_0 = !_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return $!, !_0$
```

[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?

$$y^{(i)}\ \left(\left(! + y^{(i)}x^{(i)}\right)^! x^{(i)} + \left(!_0 + y^{(i)}\right)\right)$$

$$= y^{(i)}(!^! x^{(i)} + !_0) + \left(\left(y^{(i)}\right)^2\left(x^{(i)!}\ x^{(i)}\right) + 1\right)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; ! )
  Initialize $! = [0 \ 0 \ \ldots \ 0]^!$
  Initialize $!_0 = 0$
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0) ! \ 0$
        Set $! = ! + y^{(i)}x^{(i)}$
        Set $!_0 = !_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return $!, !_0$
```

[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?

$$y^{(i)} \cdot (! + y^{(i)}x^{(i)})^! \ x^{(i)} + (!_0 + y^{(i)})$$

$$= y^{(i)}(!^! \ x^{(i)} + !_0) + (y^{(i)})^2(x^{(i)!} \ x^{(i)} + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; ! )
  Initialize $! = [0\ \ 0\ \ \ldots\ \ 0]^{!}$
  Initialize $!_0 = 0$
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^{!}\ x^{(i)} + !_0)\ !\ \ 0$
        Set $! = ! + y^{(i)}x^{(i)}$
        Set $!_0 = !_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return $!, !_0$
```

[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?

$$y^{(i)}\ \left((! + y^{(i)}x^{(i)})^{!}\ x^{(i)} + (!_0 + y^{(i)})\right)$$

$$= y^{(i)}(!^{!}\ x^{(i)} + !_0) + (y^{(i)})^2(x^{(i)!}\ x^{(i)} + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; ! )
  Initialize  $! = [0 \;\; 0 \;\; \ldots \;\; 0]^!$
  Initialize  $!_0 = 0$
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0) ! \;\; 0$
        Set $! = ! + y^{(i)} x^{(i)}$
        Set $!_0 = !_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return $!, !_0$
```

[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?

$$y^{(i)} \cdot \left((! + y^{(i)} x^{(i)})^! \, x^{(i)} + (!_0 + y^{(i)}) \right)$$

$$= y^{(i)}(!^! \, x^{(i)} + !_0) + (y^{(i)})^2 (x^{(i)!} \, x^{(i)} + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; ! )
  Initialize $! = [0\ 0\ \ldots\ 0]^!$
  Initialize $!_0 = 0$
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0) !\ 0$
        Set $! = ! + y^{(i)}x^{(i)}$
        Set $!_0 = !_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return $!, !_0$
```

[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?

$$y^{(i)} \cdot \left((! + y^{(i)}x^{(i)})^! x^{(i)} + (!_0 + y^{(i)})\right)$$

$$= y^{(i)}(!^! x^{(i)} + !_0) + (y^{(i)})^2(x^{(i)!} x^{(i)} + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; ! )
  Initialize $! = [0 \ 0 \ \ldots \ 0]^!$
  Initialize $!_0 = 0$
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0) ! \ 0$
        Set $! = ! + y^{(i)}x^{(i)}$
        Set $!_0 = !_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return $!, !_0$
```

[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?

$$y^{(i)} \cdot (! + y^{(i)}x^{(i)})^! \, x^{(i)} + (!_0 + y^{(i)})$$

$$= y^{(i)}(!^! \, x^{(i)} + !_0) + (y^{(i)})^2(x^{(i)!} \, x^{(i)} + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; ! )
  Initialize $! = [0 \ 0 \ \ldots \ 0]^!$
  Initialize $!_0 = 0$
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0) ! \ 0$
        Set $! = ! + y^{(i)}x^{(i)}$
        Set $!_0 = !_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return $!, !_0$
```

[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?

$$y^{(i)} \ (! + y^{(i)}x^{(i)})^! \ x^{(i)} + (!_0 + y^{(i)})$$

$$= y^{(i)}(!^! \ x^{(i)} + !_0) + (y^{(i)})^2(x^{(i)!} \ x^{(i)} + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; ! )
  Initialize $! = [0\ 0\ \ldots\ 0]^!$
  Initialize $!_0 = 0$
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0)\ !\ 0$
        Set $! = ! + y^{(i)}x^{(i)}$
        Set $!_0 = !_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return $!, !_0$
```

[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?

$$y^{(i)}\ (! + y^{(i)}x^{(i)})^!\ x^{(i)} + (!_0 + y^{(i)})$$

$$= y^{(i)}(!^!\ x^{(i)} + !_0) + (y^{(i)})^2(x^{(i)!}\ x^{(i)} + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; ! )
  Initialize $! = [0 \;\; 0 \;\; \ldots \;\; 0]^!$
  Initialize $!_0 = 0$
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0) \; ! \; 0$
        Set $! = ! + y^{(i)}x^{(i)}$
        Set $!_0 = !_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return $!, !_0$
```

[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?

$$y^{(i)} \; (! + y^{(i)}x^{(i)})^! \, x^{(i)} + (!_0 + y^{(i)})$$

$$= y^{(i)}(!^! \, x^{(i)} + !_0) + (y^{(i)})^2(x^{(i)!} \, x^{(i)} + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; ! )
  Initialize $! = [0\ 0\ \ldots\ 0]^!$
  Initialize $!_0 = 0$
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0)\ !\ 0$
        Set $! = ! + y^{(i)}x^{(i)}$
        Set $!_0 = !_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return $!, !_0$
```

[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?

$$y^{(i)}\ \left((! + y^{(i)}x^{(i)})^! x^{(i)} + (!_0 + y^{(i)})\right)$$

$$= y^{(i)}(!^! x^{(i)} + !_0) + (y^{(i)})^2(x^{(i)!} x^{(i)} + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; ! )
  Initialize $! = [0 \;\; 0 \;\; \ldots \;\; 0]^!$
  Initialize $!_0 = 0$
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! \, x^{(i)} + !_0) ! \;\; 0$
        Set $! = ! + y^{(i)}x^{(i)}$
        Set $!_0 = !_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return $!, !_0$
```

[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?

$$y^{(i)} \;\; (! + y^{(i)}x^{(i)})^! \; x^{(i)} + (!_0 + y^{(i)})$$

$$= y^{(i)}(!^! \; x^{(i)} + !_0) + (y^{(i)})^2(x^{(i)!} \; x^{(i)} + 1)$$

$$= y^{(i)}(!^! \; x^{(i)} + !_0) + (\,! x^{(i)} !^{\,2} + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; ! )
  Initialize ! $= [0\ \ 0\ \ \ldots\ \ 0]^!$
  Initialize $!_0 = 0$
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0)\ !\ \ 0$
        Set $! = ! + y^{(i)}x^{(i)}$
        Set $!_0 = !_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return $!, !_0$
```

[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?

$$y^{(i)}\ \ (! + y^{(i)}x^{(i)})^!\ x^{(i)} + (!_0 + y^{(i)})$$

$$= y^{(i)}(!^!\ x^{(i)} + !_0) + (y^{(i)})^2(x^{(i)!}\ x^{(i)} + 1)$$

$$= y^{(i)}(!^!\ x^{(i)} + !_0) + (\,!\,x^{(i)}!^{\,2} + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; ! )
  Initialize  $! = [0 \;\; 0 \;\; \ldots \;\; 0]^!$
  Initialize  $!_0 = 0$
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0) ! \;\; 0$
        Set $! = ! + y^{(i)}x^{(i)}$
        Set $!_0 = !_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return $!, !_0$
```

[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?

$$y^{(i)} \;\; (! + y^{(i)}x^{(i)})^! \; x^{(i)} + (!_0 + y^{(i)})$$

$$= y^{(i)}(!^! \; x^{(i)} + !_0) + (y^{(i)})^2(x^{(i)!} \; x^{(i)} + 1)$$

$$= y^{(i)}(!^! \; x^{(i)} + !_0) + (!x^{(i)}!^2 + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; $!$ )
  Initialize $! = [0 \ 0 \ \ldots \ 0]^!$
  Initialize $!_0 = 0$
  for t = 1 to $!$
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0) ! 0$
        Set $! = ! + y^{(i)}x^{(i)}$
        Set $!_0 = !_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return $!, !_0$
```

[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?

$$y^{(i)} \left((! + y^{(i)}x^{(i)})^! x^{(i)} + (!_0 + y^{(i)}) \right)$$

$$= y^{(i)}(!^! x^{(i)} + !_0) + (y^{(i)})^2(x^{(i)!} x^{(i)} + 1)$$

$$= y^{(i)}(!^! x^{(i)} + !_0) + (!x^{(i)}!^2 + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; ! )
  Initialize  $! = [0\ \ 0\ \ \ldots\ \ 0]^!$
  Initialize  $!_0 = 0$
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0) !\ \ 0$
        Set $! = ! + y^{(i)}x^{(i)}$
        Set $!_0 = !_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return $!, !_0$
```

[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?

$$y^{(i)}\ \ (! + y^{(i)}x^{(i)})^!\ x^{(i)} + (!_0 + y^{(i)})$$

$$= y^{(i)}(!^!\ x^{(i)} + !_0) + (y^{(i)})^2(x^{(i)!}\ x^{(i)} + 1)$$

$$= y^{(i)}(!^!\ x^{(i)} + !_0) + (\,!\,x^{(i)}!^{\,2} + 1)$$

Perceptron Algorithm

```
Perceptron( $\mathcal{D}_n$ ; ! )
  Initialize $! = [0 \;\; 0 \;\; \ldots \;\; 0]^!$
  Initialize $!_0 = 0$
  for t = 1 to !
    changed = False
    for i = 1 to n
      if $y^{(i)}(!^! x^{(i)} + !_0) ! 0$
        Set $! = ! + y^{(i)}x^{(i)}$
        Set $!_0 = !_0 + y^{(i)}$
        changed = True
    if not changed
      break
  Return $!, !_0$
```

[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?

$$y^{(i)} \quad (! + y^{(i)}x^{(i)})^! \, x^{(i)} + (!_0 + y^{(i)})$$

$$= y^{(i)}(!^! \, x^{(i)} + !_0) + (y^{(i)})^2(x^{(i)!} \, x^{(i)} + 1)$$

$$= y^{(i)}(!^! \, x^{(i)} + !_0) + (\,!x^{(i)}!^2 + 1)$$

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 $!, !_0$ such that, for every point index $i \; ! \; \{1, \ldots, n\}$, we have

$$y^{(i)}(!^{!} x^{(i)} + !_0) > 0$$

Let's Talk About Classifier Quality

- *Definition*: A training set $\mathcal{D}_n$ is **linearly separable** if there exist $!, !_0$ such that, for every point index $i \; ! \; \{1, \ldots, n\}$, we have

$$y^{(i)}(!^! x^{(i)} + !_0) > 0$$

Let's Talk About Classifier Quality

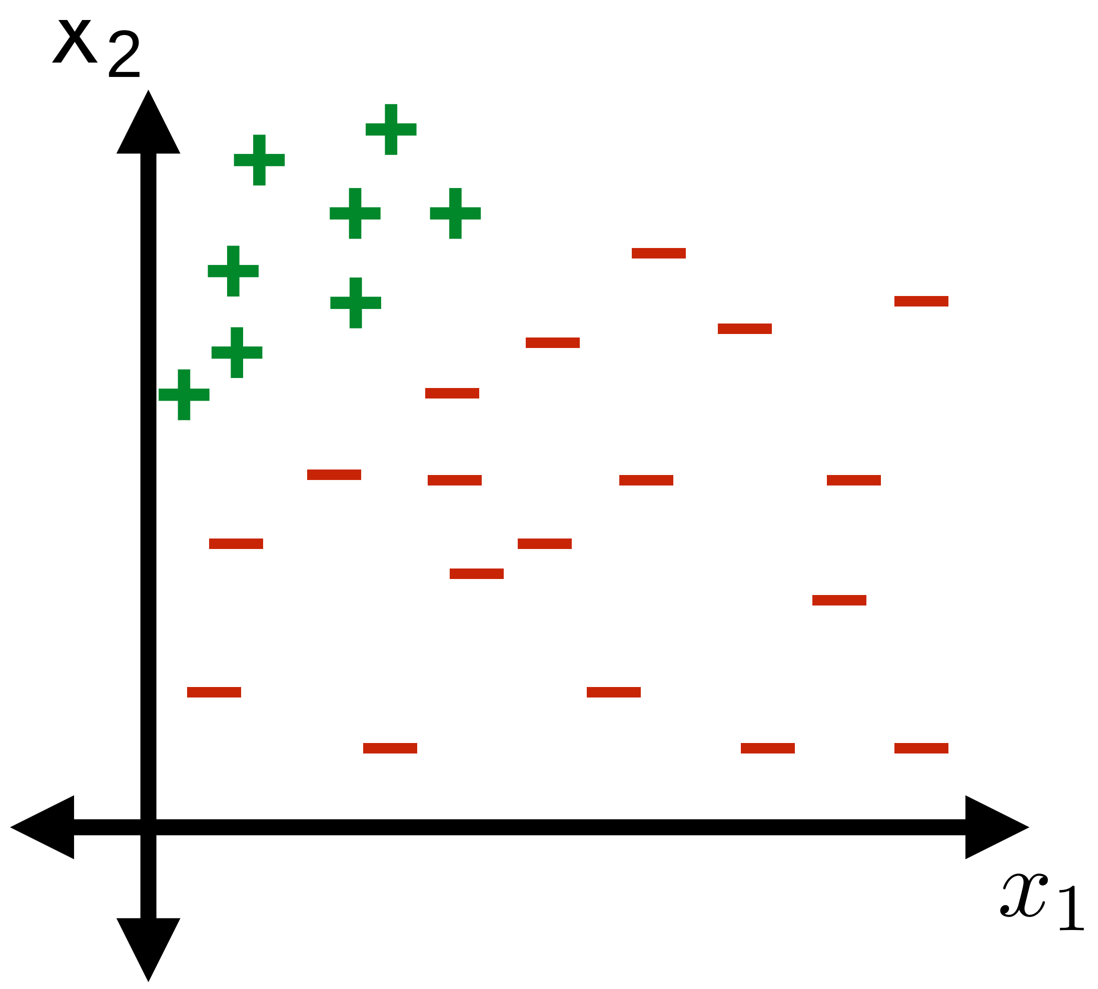
- *Definition*: A training set $\mathcal{D}_n$ is **linearly separable** if there exist $!, !_0$ such that, for every point index $i\ !\ \{1, \ldots, n\}$, we have

$$y^{(i)}(!^{!}\ x^{(i)} + !_0) > 0$$

Let’s Talk About Classifier Quality

- *Definition*: A training set $\mathcal{D}_n$ is **linearly separable** if there exist $!, !_0$ such that, for every point index $i \, ! \, \{1, \ldots, n\}$, we have

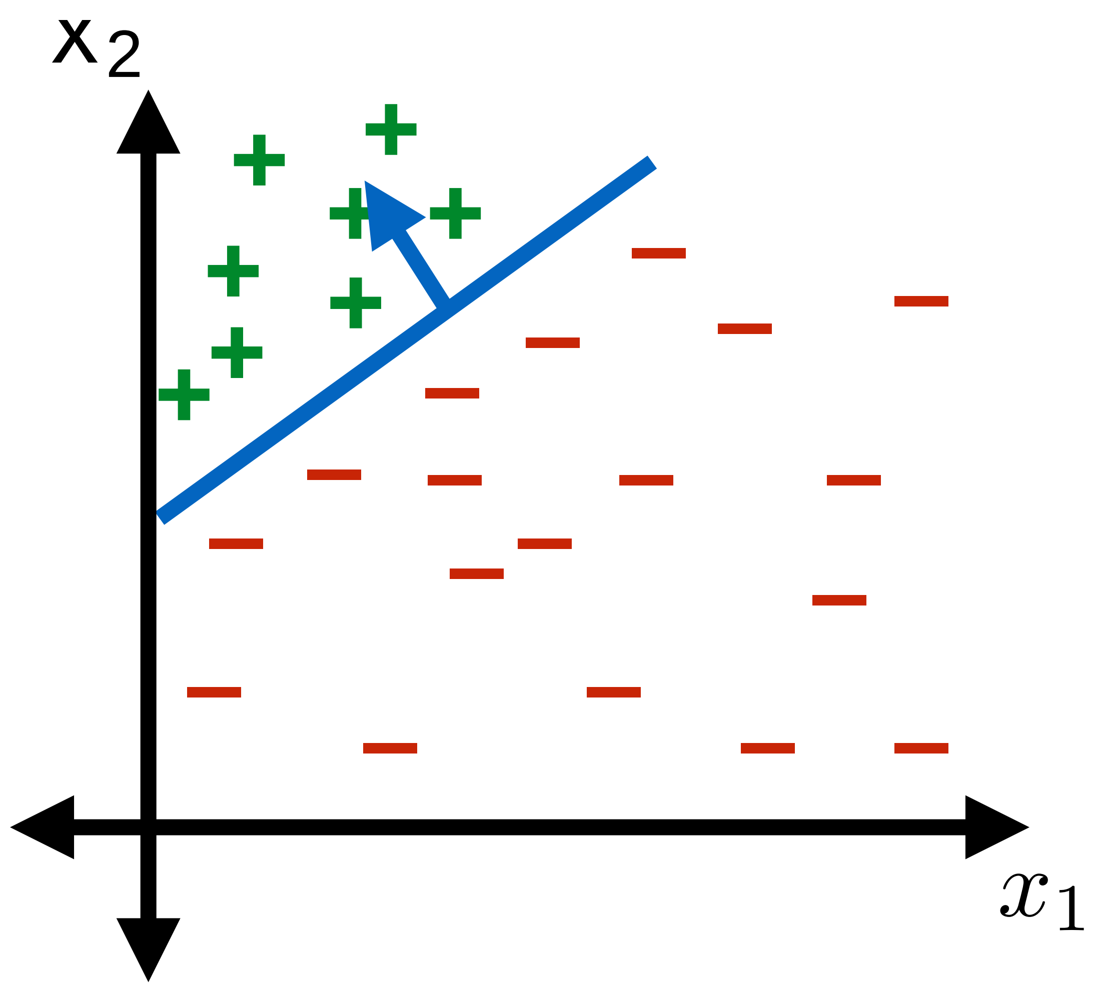
$$y^{(i)}(!^{!} x^{(i)} + !_0) > 0$$



Let’s Talk About Classifier Quality

- *Definition*: A training set $\mathcal{D}_n$ is **linearly separable** if there exist $!, !_0$ such that, for every point index $i \; ! \; \{1, \ldots, n\}$, we have

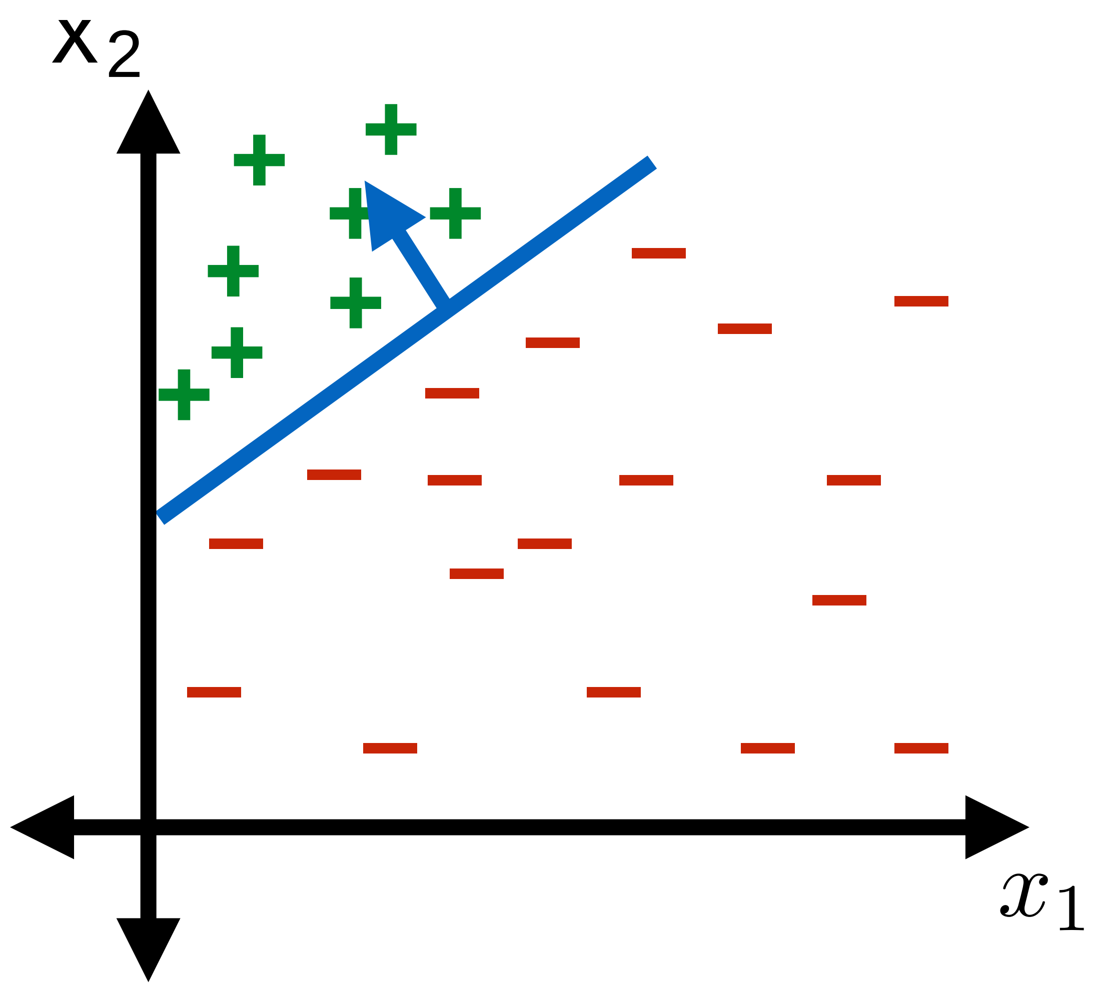
$$y^{(i)}(!^{!} x^{(i)} + !_0) > 0$$

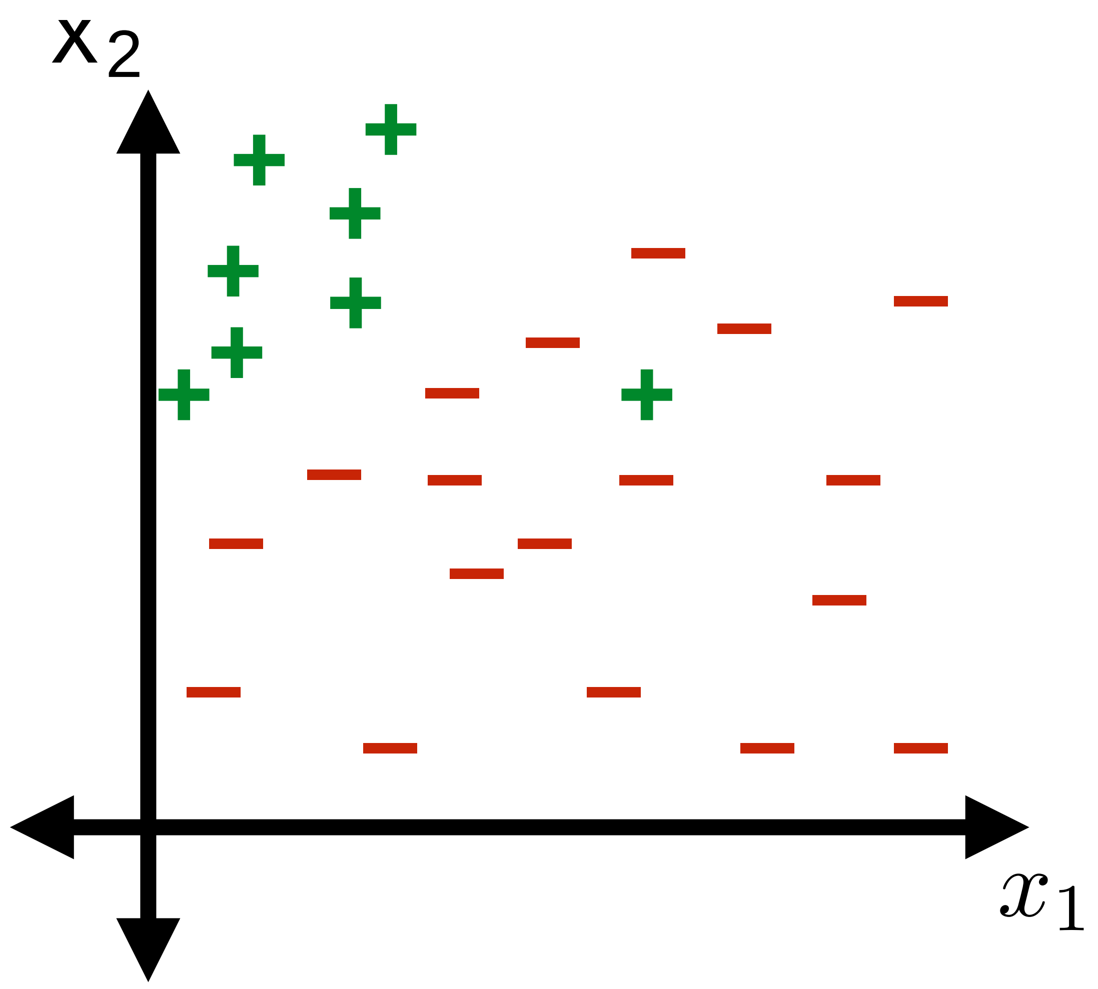


Let’s Talk About Classifier Quality

- *Definition*: A training set $\mathcal{D}_n$ is **linearly separable** if there exist $!, !_0$ such that, for every point index $i \; ! \; \{1, \ldots, n\}$, we have

$$y^{(i)}(!^{!}\, x^{(i)} + !_0) > 0$$

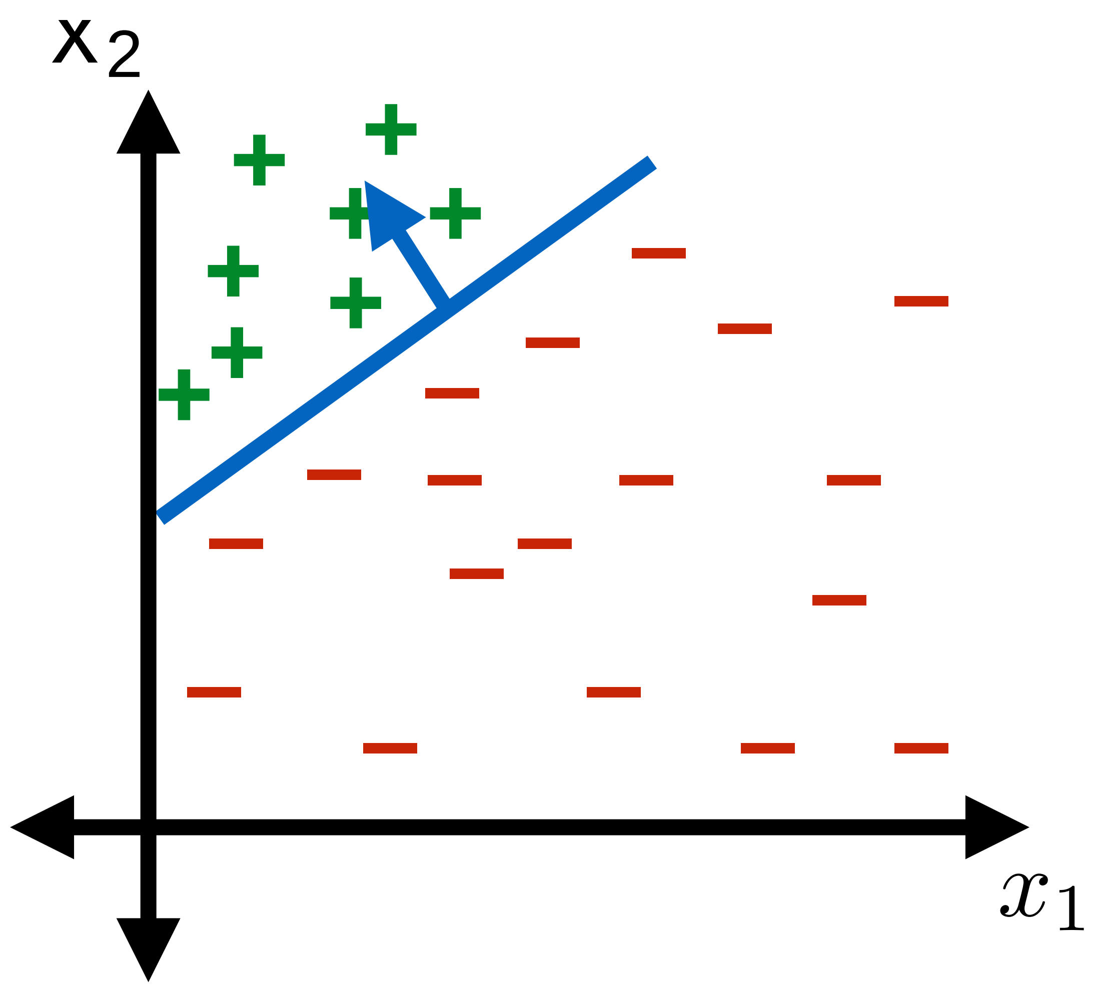


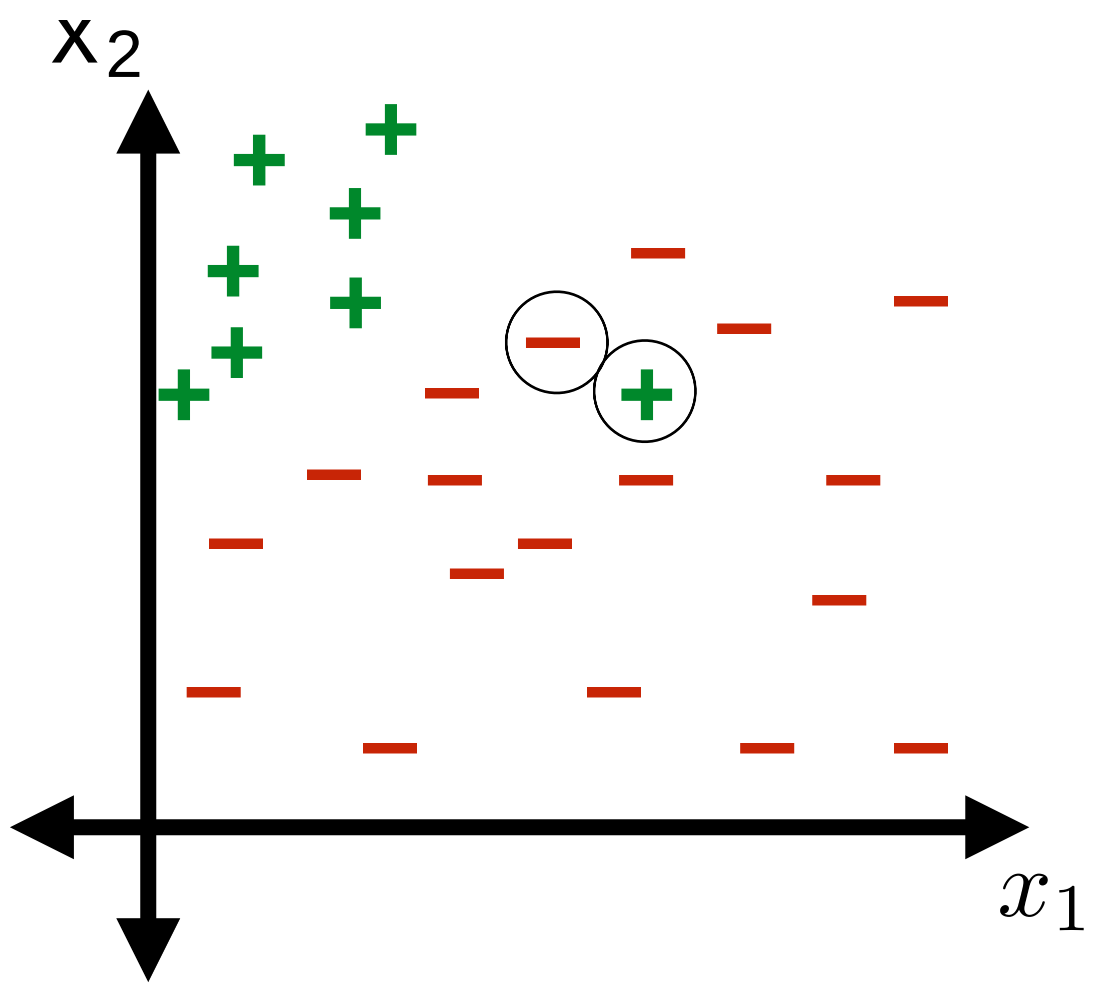


Let's Talk About Classifier Quality

- *Definition*: A training set $\mathcal{D}_n$ is **linearly separable** if there exist $!, !_0$ such that, for every point index $i \; ! \; \{1, \ldots, n\}$, we have

$$y^{(i)}(!^{!} x^{(i)} + !_0) > 0$$

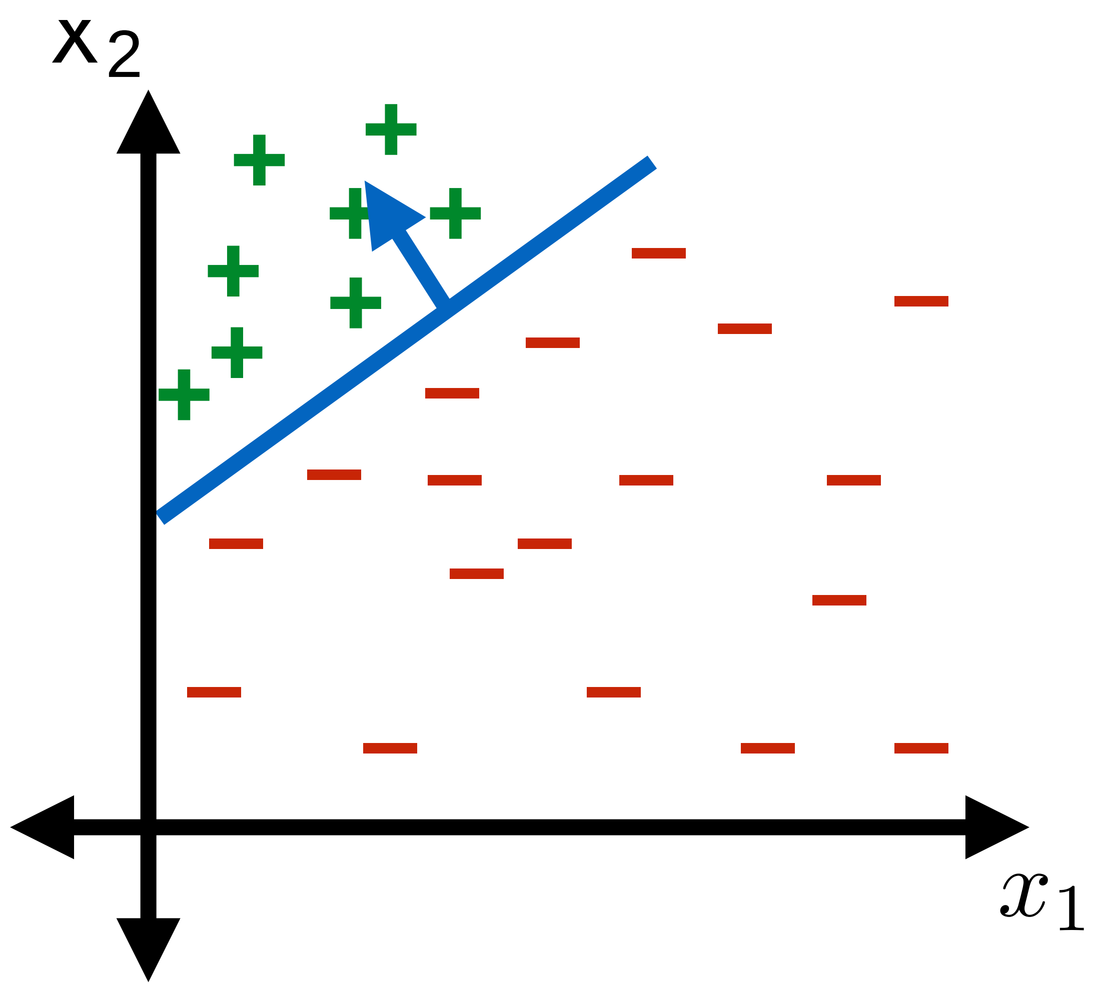


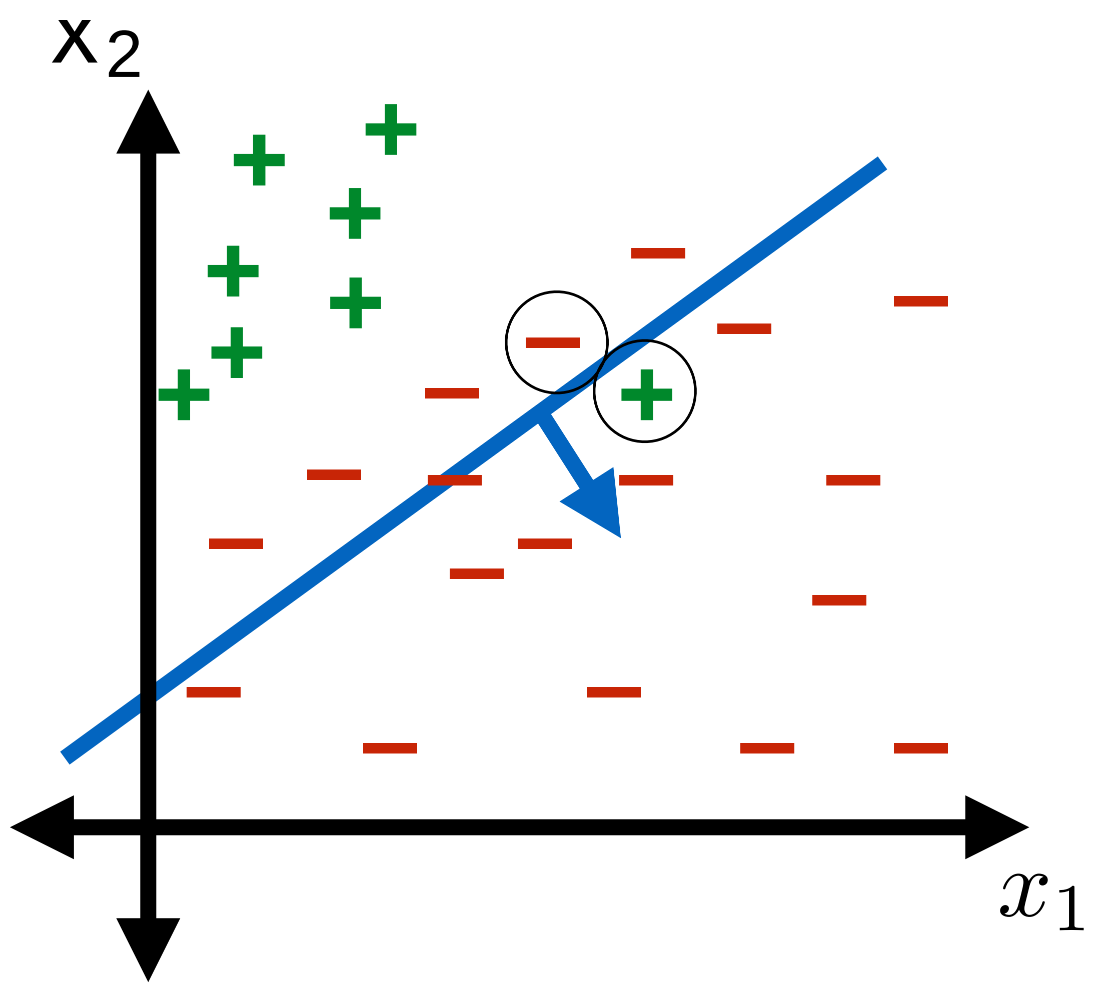


Let’s Talk About Classifier Quality

- *Definition*: A training set $\mathcal{D}_n$ is **linearly separable** if there exist $!, !_0$ such that, for every point index $i \; ! \; \{1, \ldots, n\}$, we have

$$y^{(i)}(!^{!} x^{(i)} + !_0) > 0$$



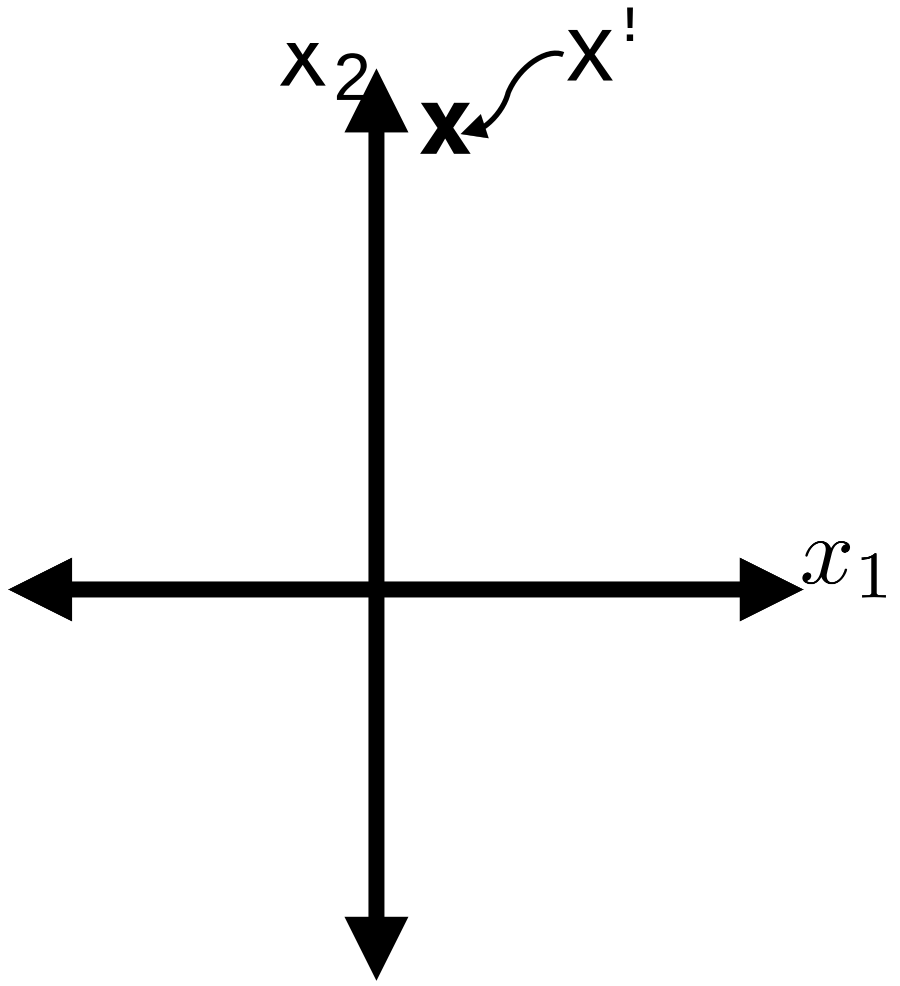


Let’s Talk About Classifier Quality

Math facts!

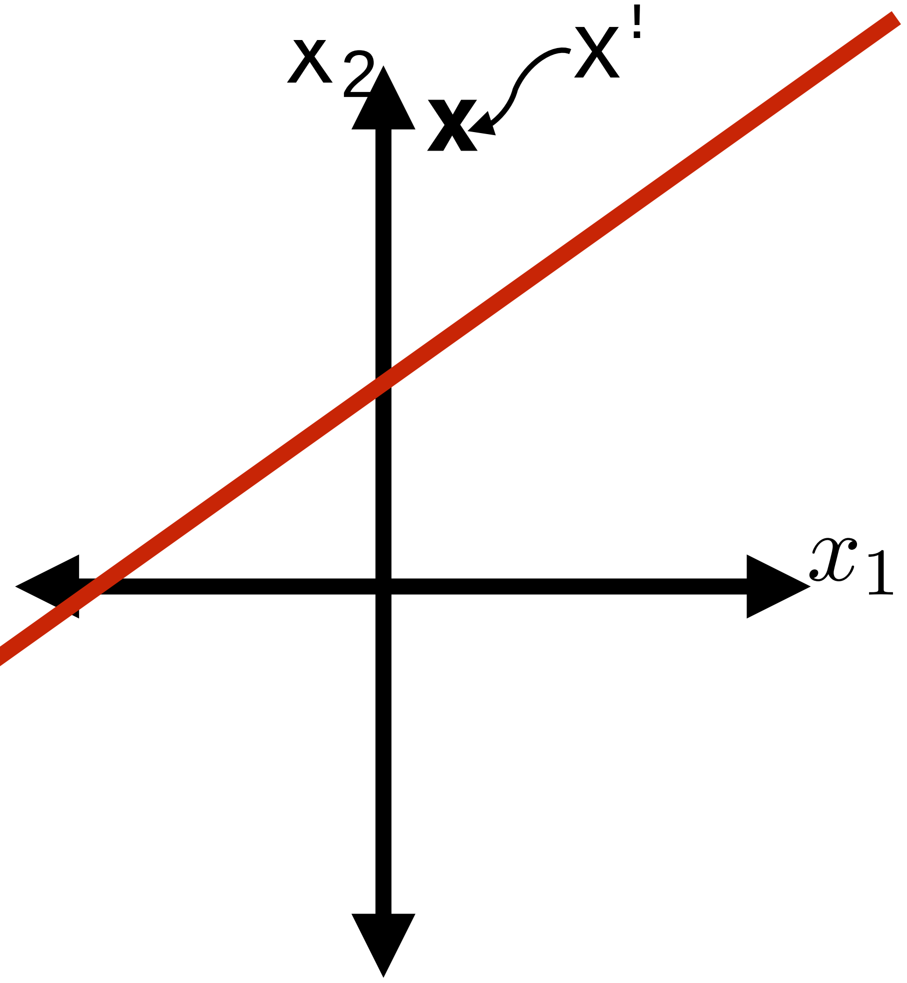
Let's Talk About Classifier Quality

Math facts!



Let’s Talk About Classifier Quality

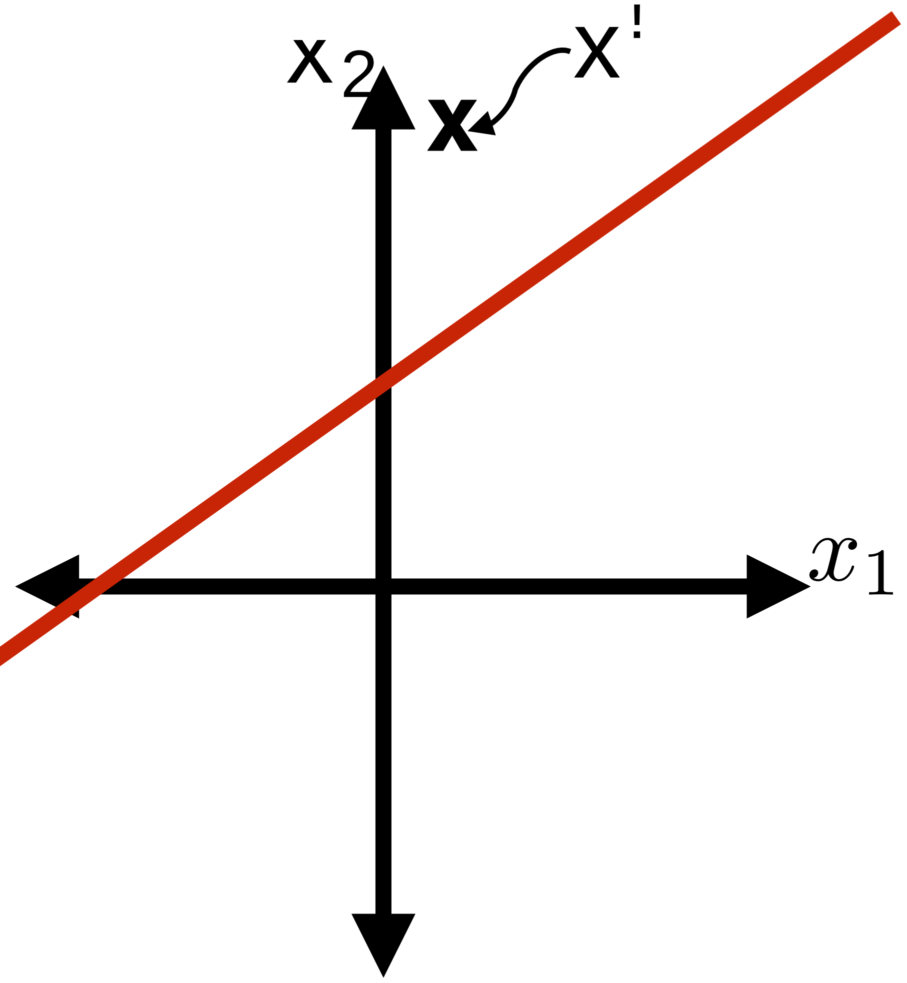
Math facts!



Let’s Talk About Classifier Quality

Math facts!

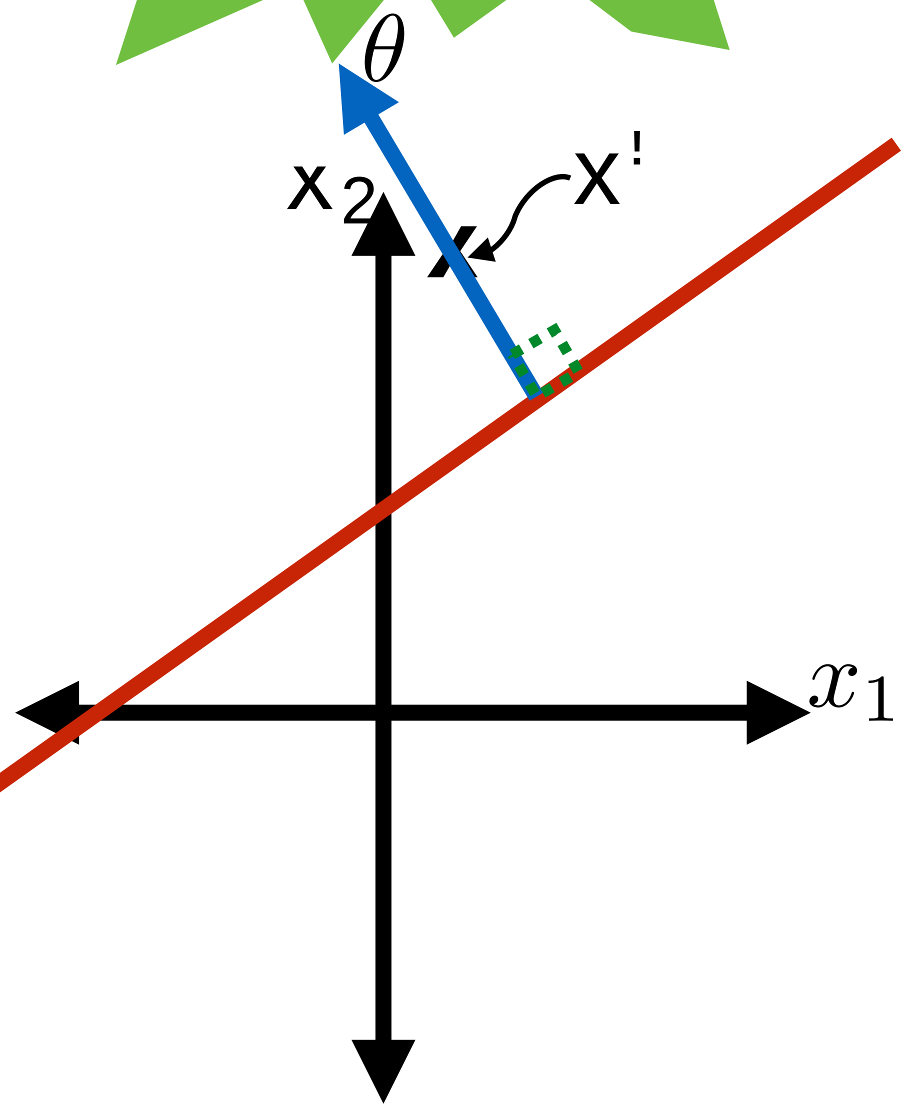
- The signed distance from a hyperplane defined by $!, !_0$ to a point $x^!$ is:



Let’s Talk About Classifier Quality

Math facts!

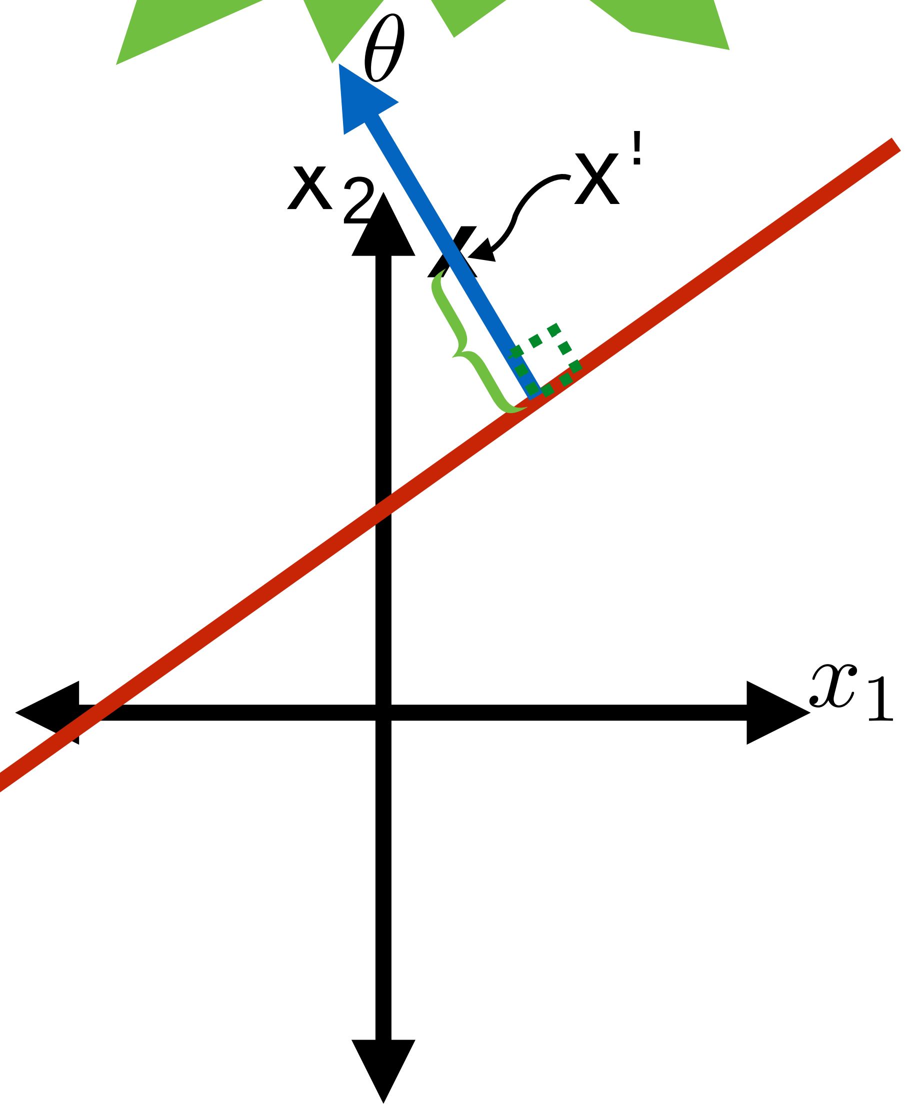
- The signed distance from a hyperplane defined by $!, !_0$ to a point $x^!$ is:



Let’s Talk About Classifier Quality

Math facts!

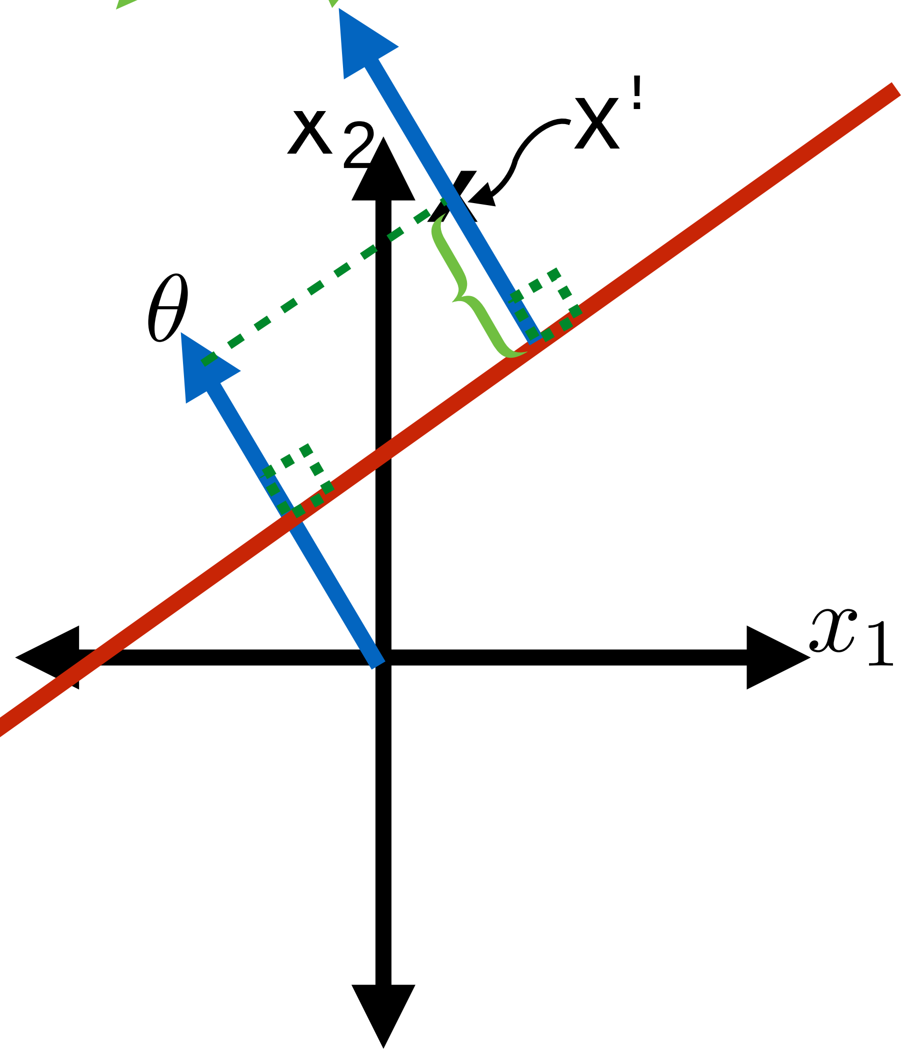
- The signed distance from a hyperplane defined by $!,!_0$ to a point $x^!$ is:



Let’s Talk About Classifier Quality

Math facts!

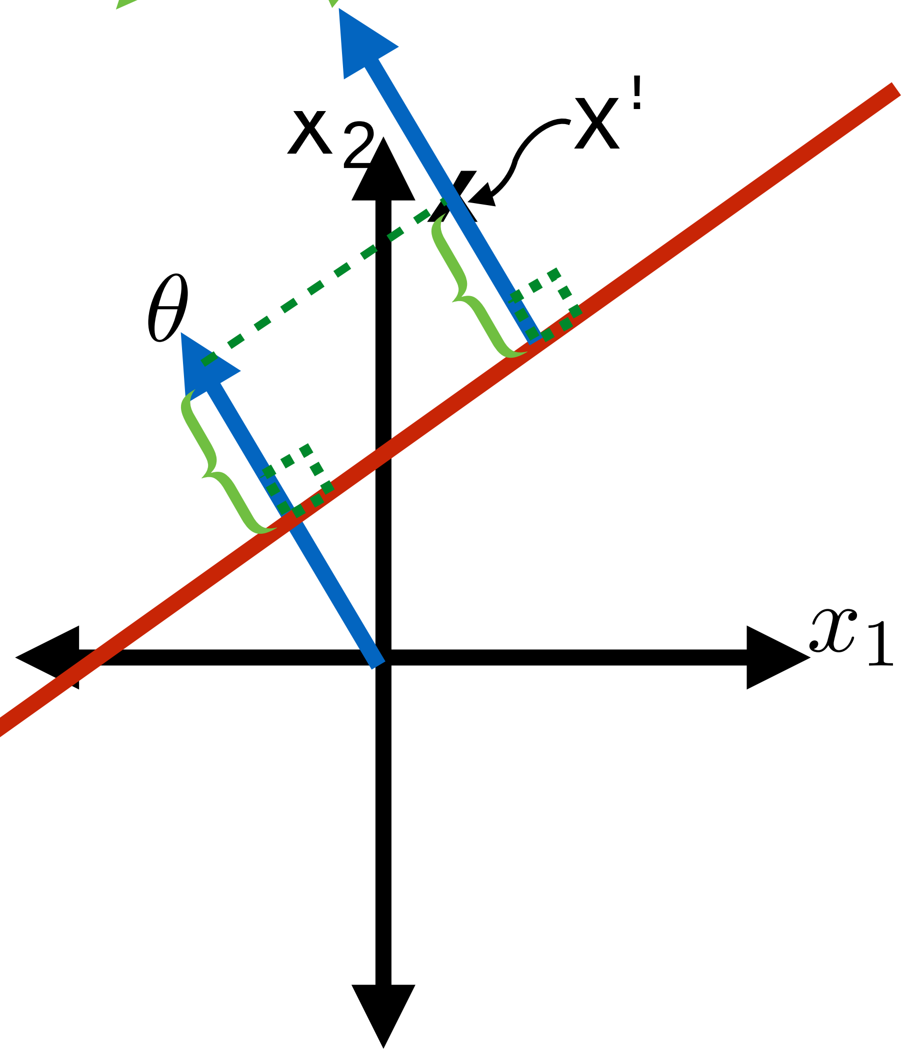
- The signed distance from a hyperplane defined by $!,!_0$ to a point $x^!$ is:



Let’s Talk About Classifier Quality

Math facts!

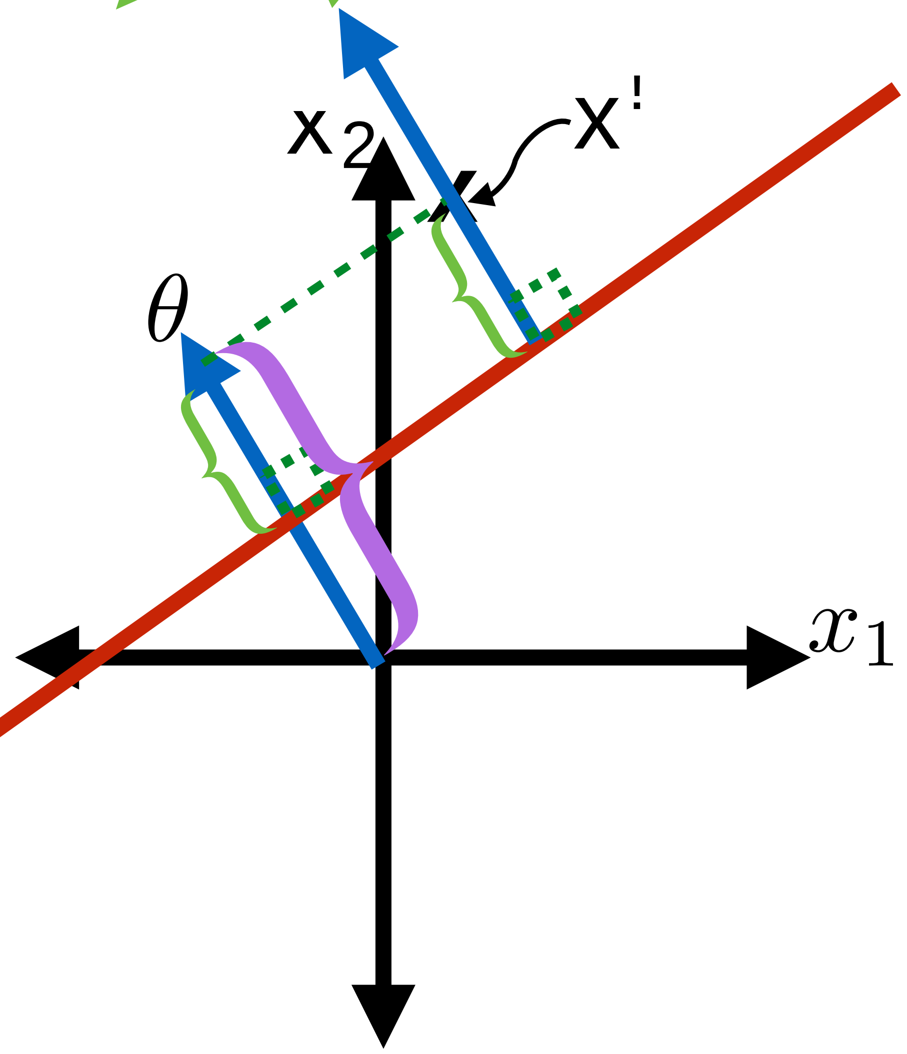
- The signed distance from a hyperplane defined by $!, !_0$ to a point $x^!$ is:



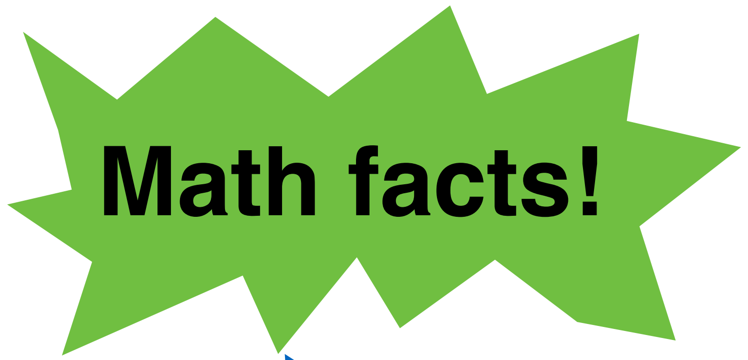
Let’s Talk About Classifier Quality

Math facts!

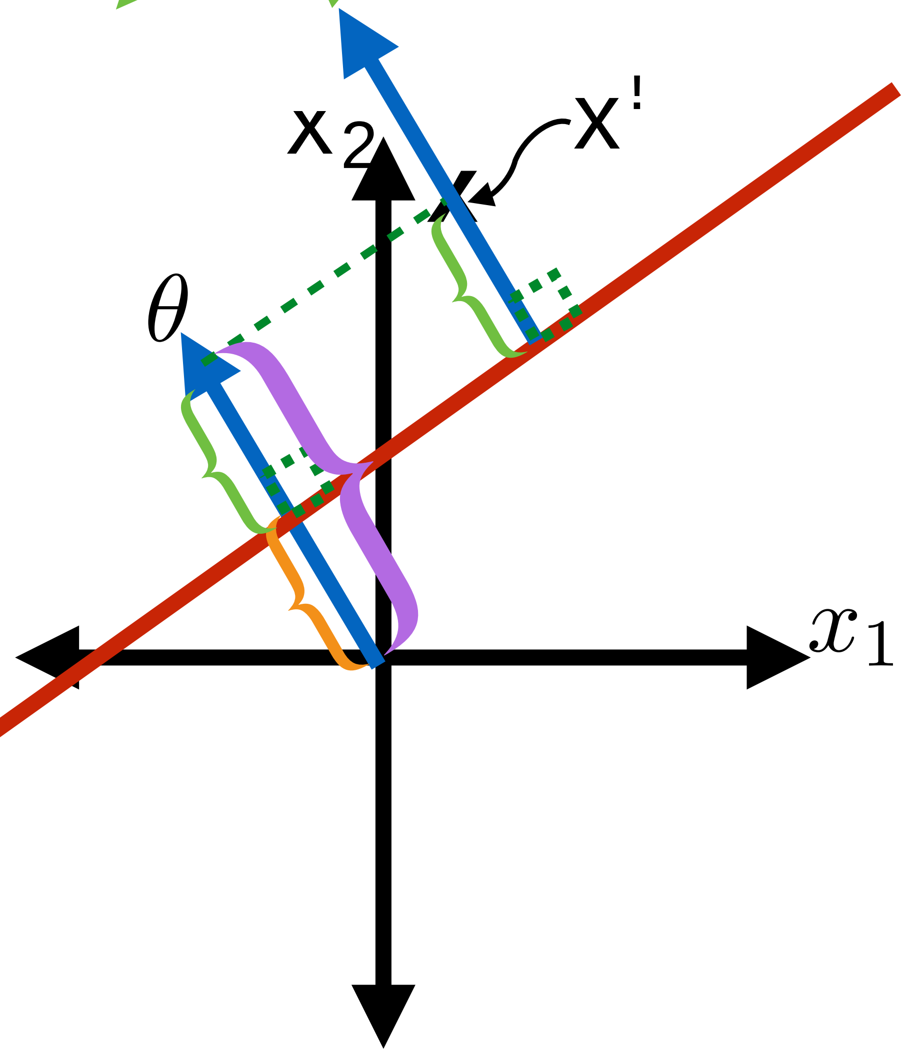
- The signed distance from a hyperplane defined by $!, !_0$ to a point $x^!$ is:



Let's Talk About Classifier Quality



- The signed distance from a hyperplane defined by !,!$_0$ to a point x$^!$ is:

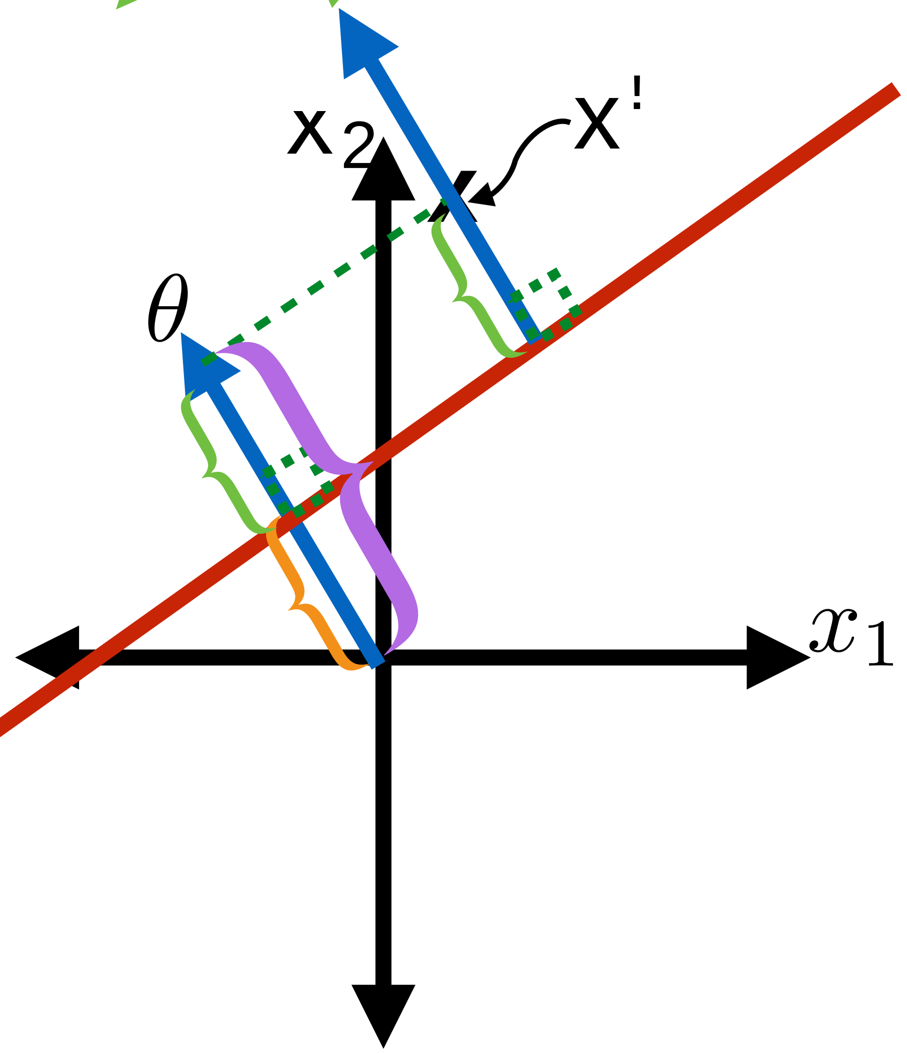


Let’s Talk About Classifier Quality

Math facts!

- The signed distance from a hyperplane defined by !,!$_0$ to a point x$^!$ is:

  = projection of x$^!$ on θ



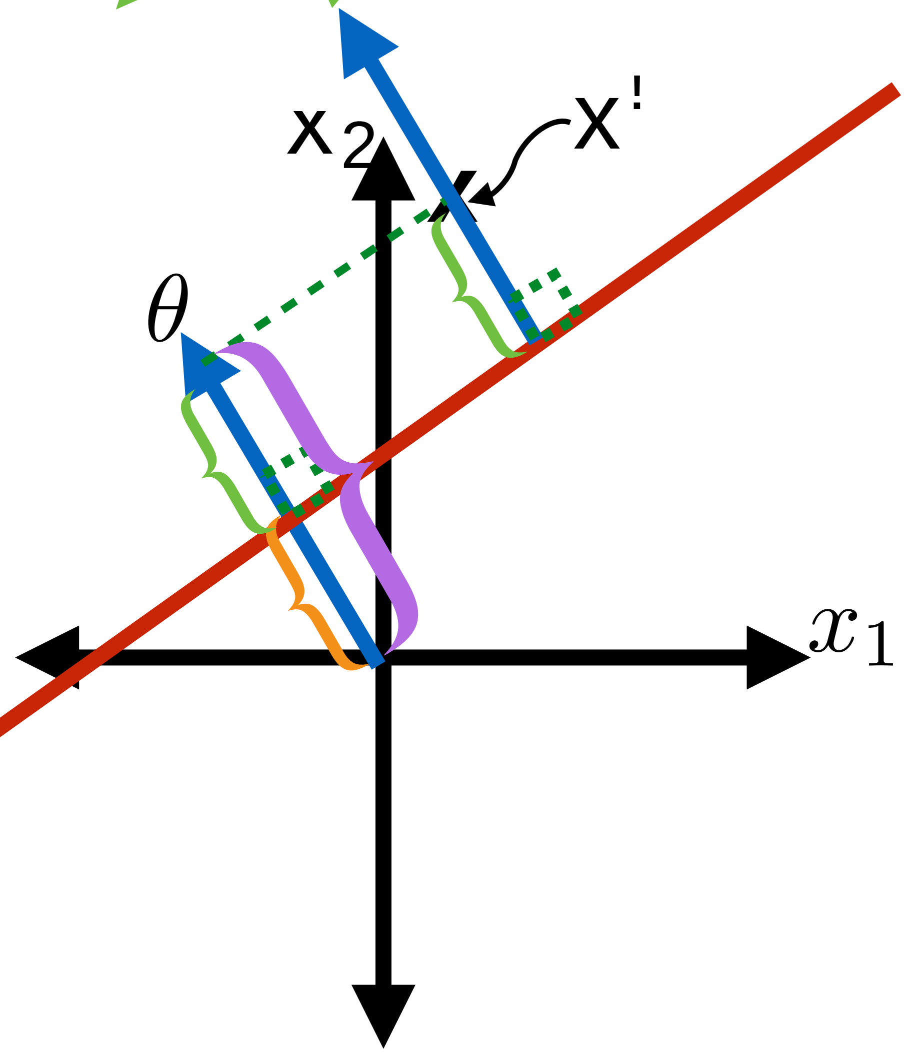
Let’s Talk About Classifier Quality

Math facts!

- The signed distance from a hyperplane defined by $!, !_0$ to a point $x^!$ is:

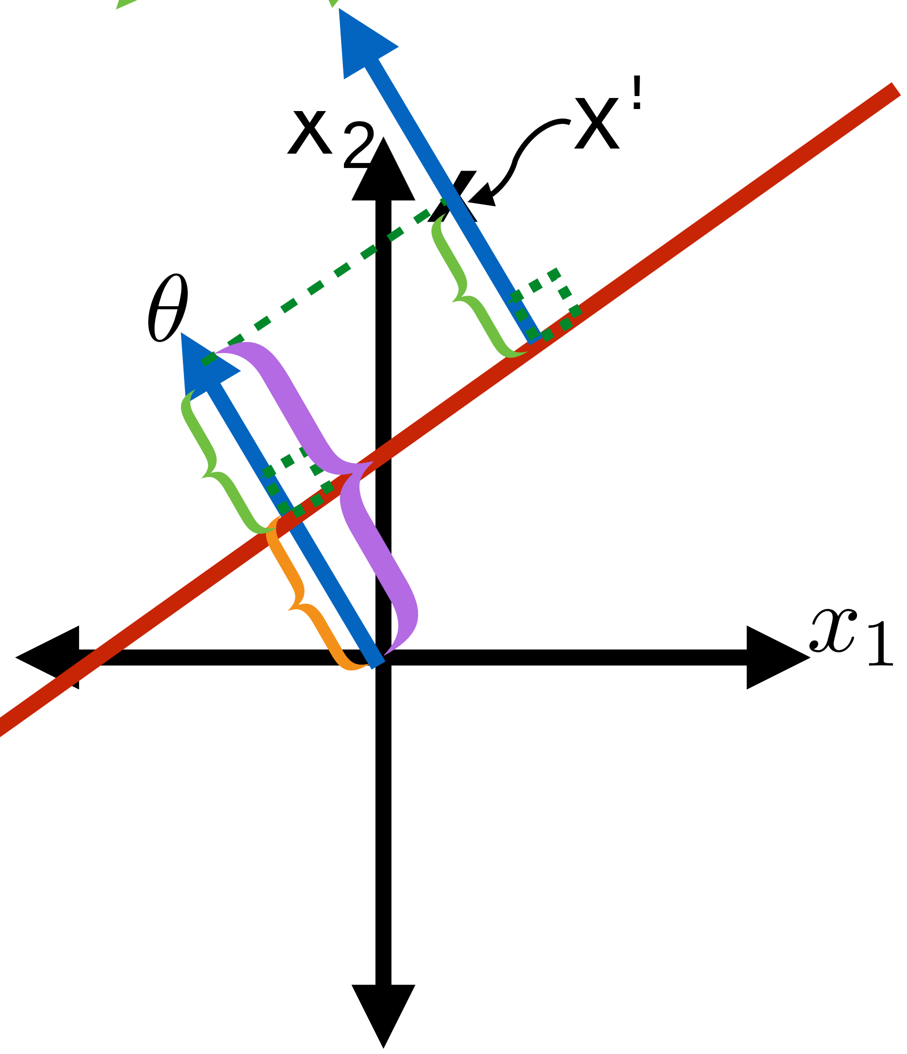
  = projection of $x^!$ on θ

  – signed distance of line to origin



Let's Talk About Classifier Quality

Math facts!



- The signed distance from a hyperplane defined by $!, !_0$ to a point $x^{!}$ is:

  = projection of $x^{!}$ on θ

  – signed distance of line to origin

$$= \frac{!^{!}\,x^{"}}{!\,!\,!}$$

Let's Talk About Classifier Quality

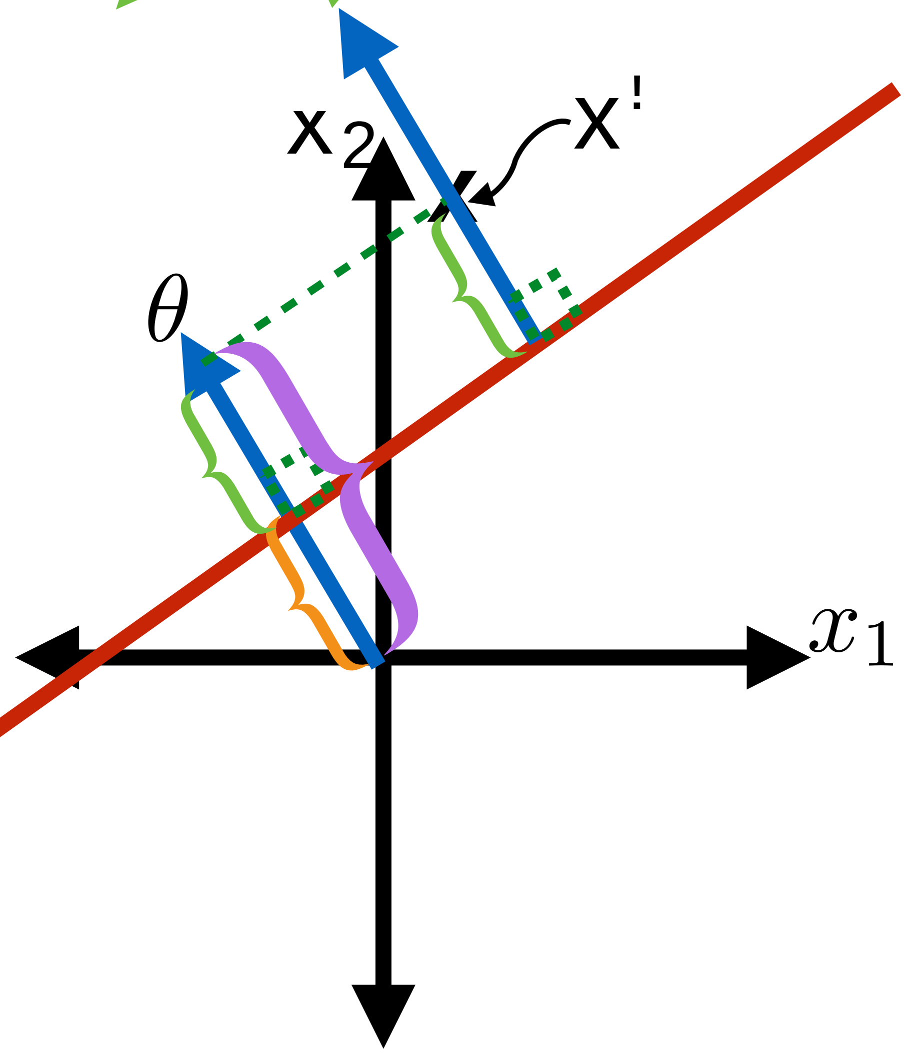
Math facts!

- The signed distance from a hyperplane defined by $!, !_0$ to a point $x^!$ is:

  = projection of $x^!$ on θ

  – signed distance of line to origin

$$= \frac{!^! x^"}{!!!} " \frac{" !_0}{!!!}$$



Let's Talk About Classifier Quality

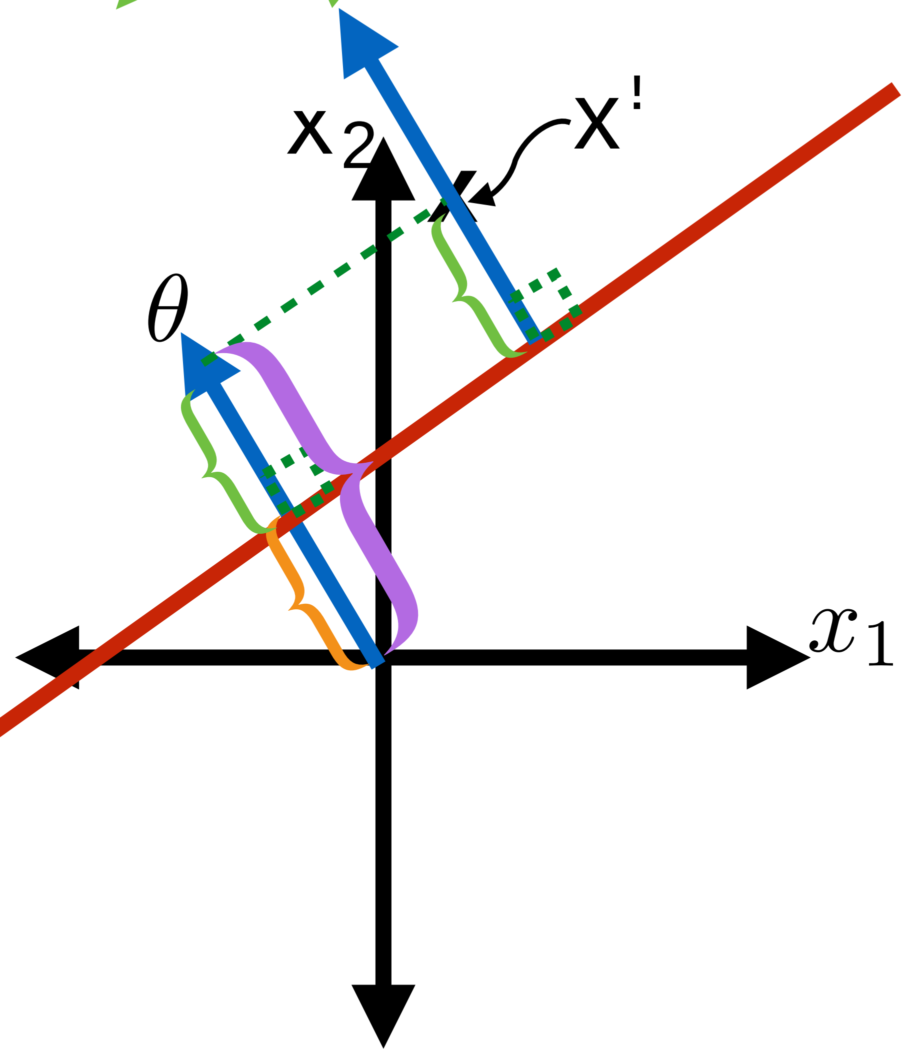
Math facts!

- The signed distance from a hyperplane defined by $!, !_0$ to a point $x^!$ is:

  = projection of $x^!$ on θ

  – signed distance of line to origin

$$= \frac{!^! x^"}{!\,!\,!} \; " \; \frac{" \; !_0}{!\,!\,!} = \frac{!^! x^" + !_0}{!\,!\,!}$$



Let’s Talk About Classifier Quality

Math facts!

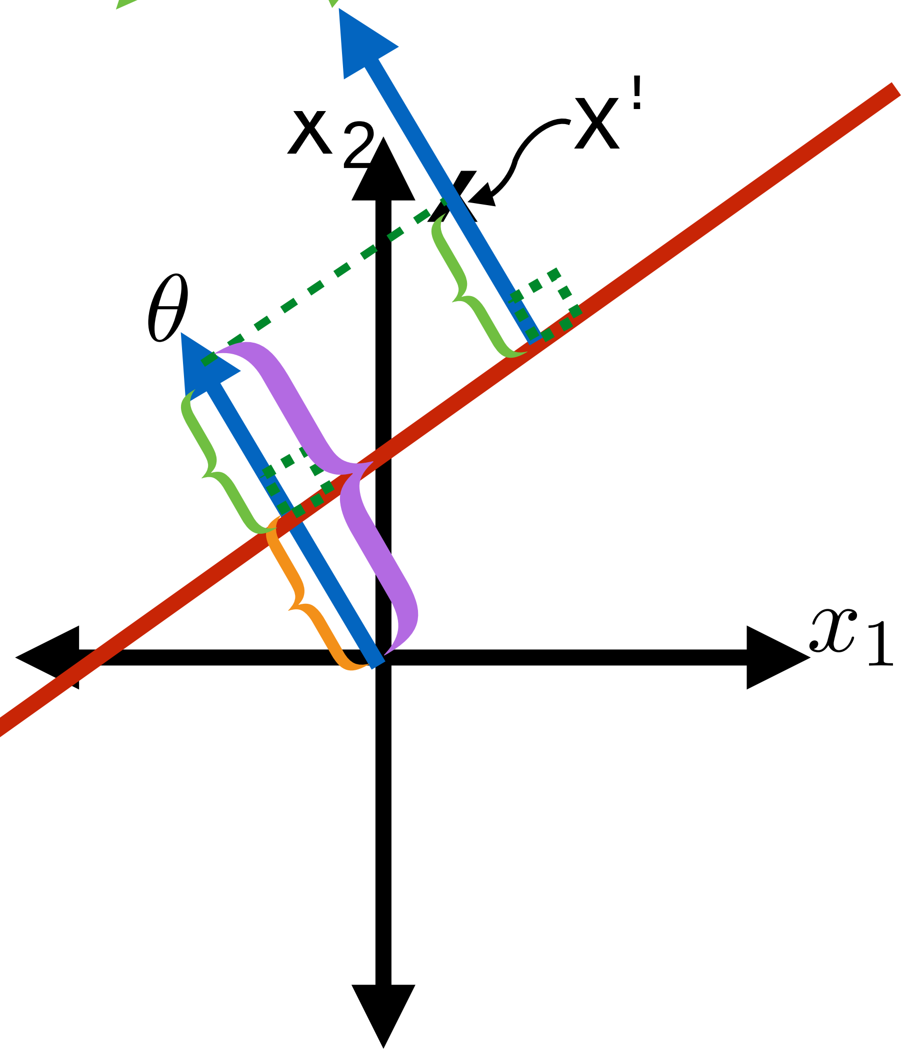
- The signed distance from a hyperplane defined by $!, !_0$ to a point $x^!$ is:

  = projection of $x^!$ on θ

  – signed distance of line to origin

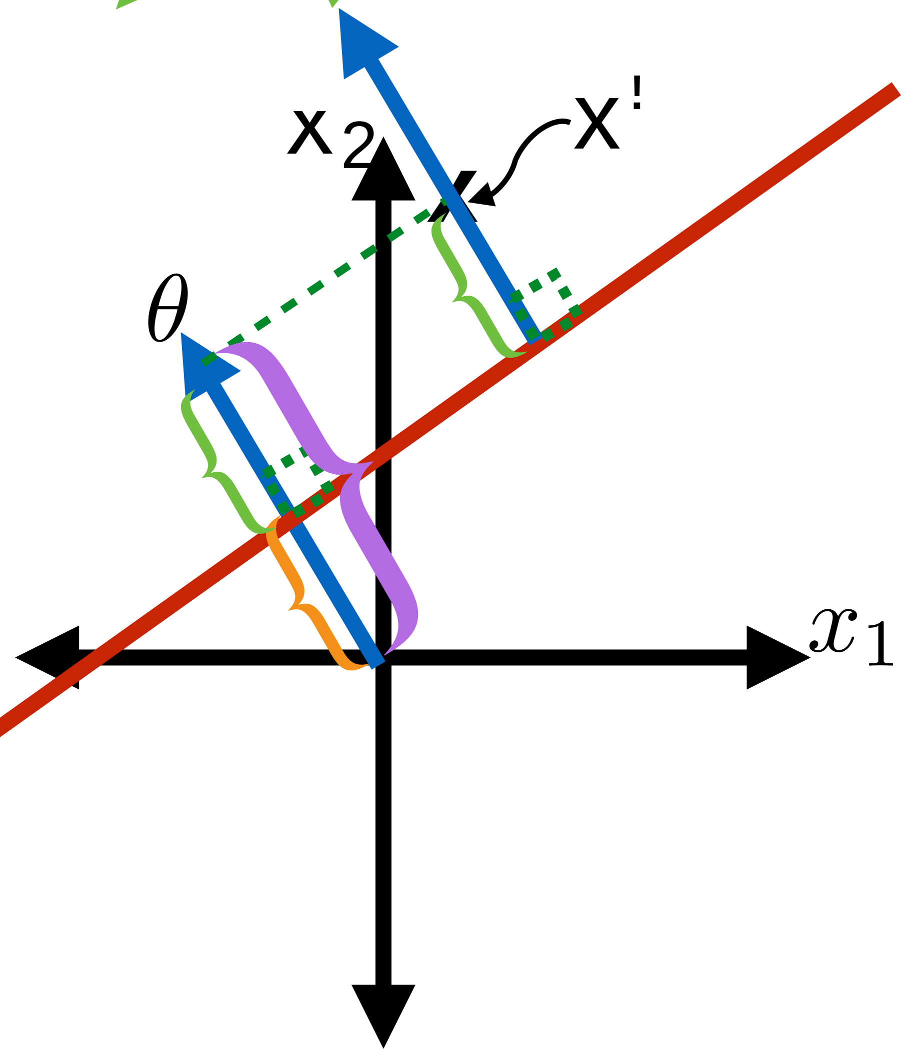
$$= \frac{!^! x^"}{!!!} " \frac{" !_0}{!!!} = \frac{!^! x^" + !_0}{!!!}$$

- *Definition*: The **margin of the labelled point** $(x^!, y^!)$ with respect to the hyperplane defined by $!, !_0$ is:



Let’s Talk About Classifier Quality

Math facts!



- The signed distance from a hyperplane defined by $!, !_0$ to a point $x^!$ is:

  = projection of $x^!$ on θ

  – signed distance of line to origin

$$= \frac{!^! x^"}{!!!} " \frac{" !_0}{!!!} = \frac{!^! x^" + !_0}{!!!}$$

- *Definition*: The **margin of the labelled point** $(x^!, y^!)$ with respect to the hyperplane defined by $!, !_0$ is:

$$y^! \;\dot{}\; \frac{!^" x^! + !_0}{!!!}$$

Let's Talk About Classifier Quality

Math facts!

- The signed distance from a hyperplane defined by $!, !_0$ to a point $x^!$ is:

  = projection of $x^!$ on θ

  – signed distance of line to origin

$$= \frac{!^! x^"}{!!!} " \frac{" !_0}{!!!} = \frac{!^! x^" + !_0}{!!!}$$

- *Definition*: The **margin of the labelled point** $(x^!, y^!)$ with respect to the hyperplane defined by $!, !_0$ is:

$$y^! \cdot \frac{!^" x^! + !_0}{!!!}$$

Let’s Talk About Classifier Quality

Math facts!

- The signed distance from a hyperplane defined by $!, !_0$ to a point $x^!$ is:

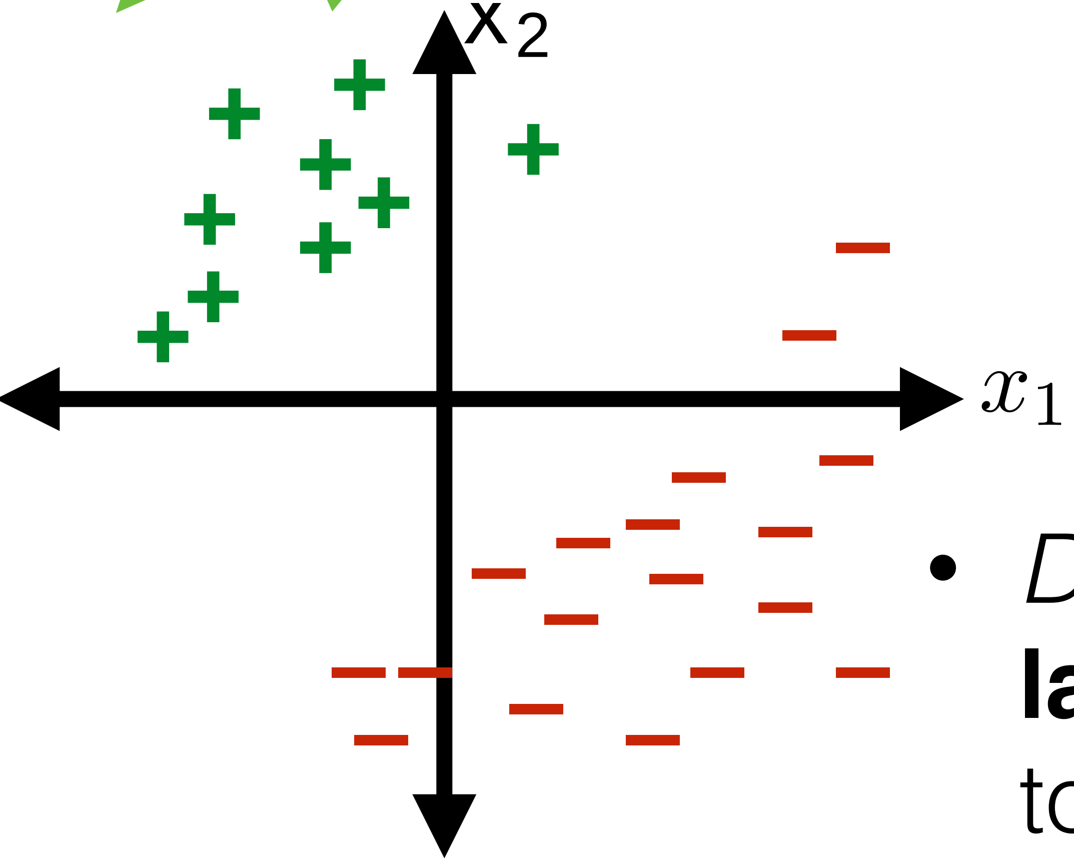
  = projection of $x^!$ on θ

  – signed distance of line to origin

$$= \frac{!^! x^"}{!!!} " \frac{" !_0}{!!!} = \frac{!^! x^" + !_0}{!!!}$$

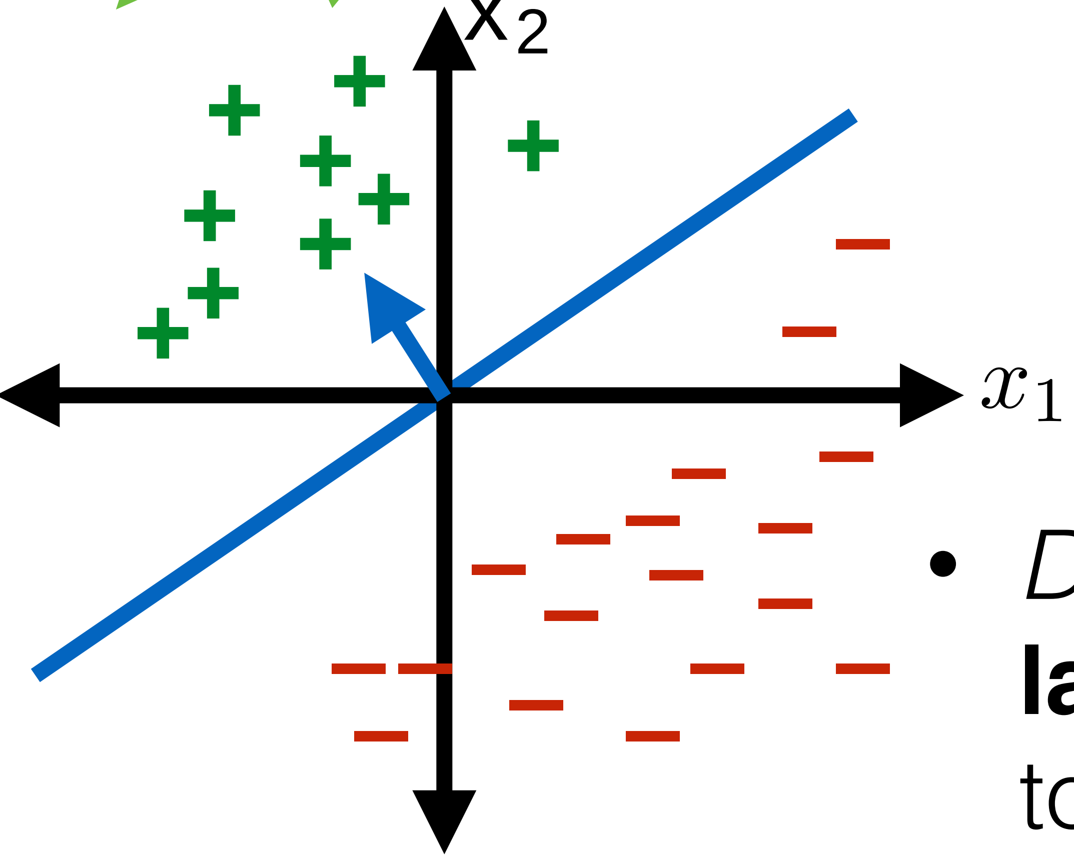
- *Definition*: The **margin of the labelled point** $(x^!, y^!)$ with respect to the hyperplane defined by $!, !_0$ is:

$$y^! \cdot \frac{!^" x^! + !_0}{!!!}$$



Let’s Talk About Classifier Quality

Math facts!



- The signed distance from a hyperplane defined by $!, !_0$ to a point $x^{!}$ is:

  = projection of $x^{!}$ on θ

  – signed distance of line to origin

$$= \frac{!^{!}\, x^{"}}{!\,!\,!} \;"\; \frac{"\;!_0}{!\,!\,!} = \frac{!^{!}\, x^{"} + !_0}{!\,!\,!}$$

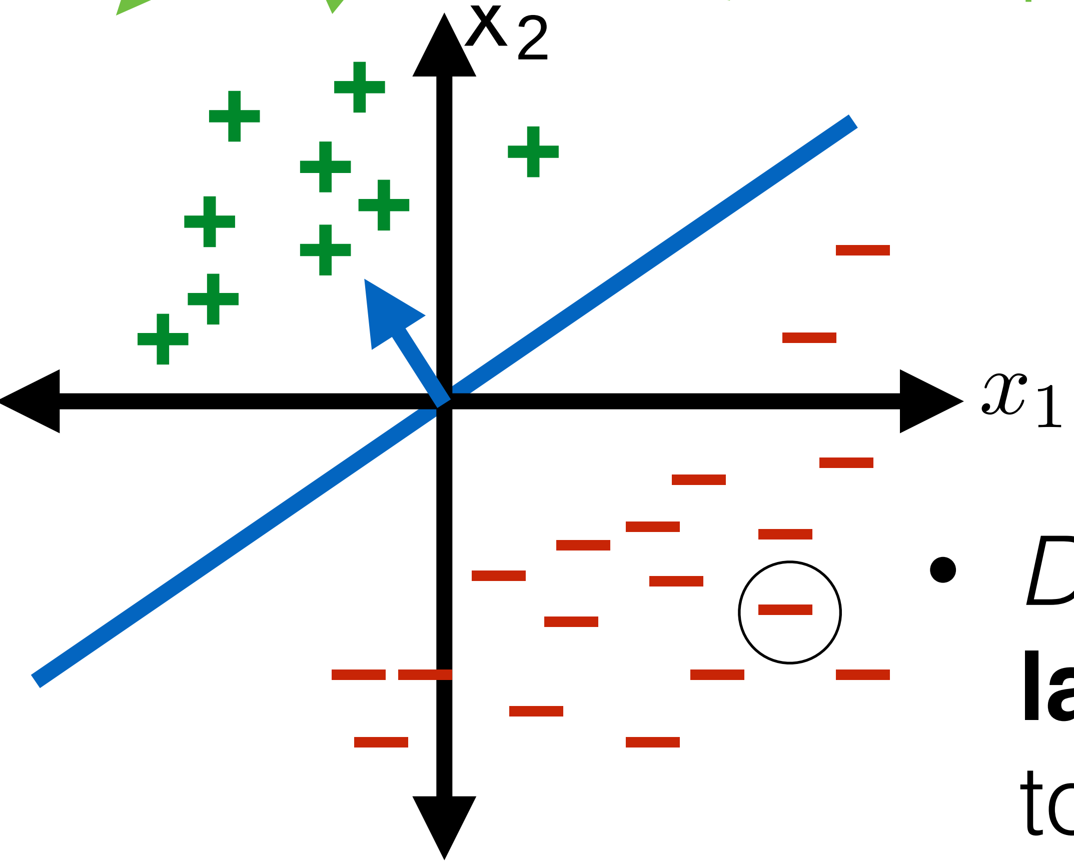
- *Definition*: The **margin of the labelled point** $(x^{!}, y^{!})$ with respect to the hyperplane defined by $!, !_0$ is:

$$y^{!} \cdot \frac{!^{"}\, x^{!} + !_0}{!\,!\,!}$$

5

Let's Talk About Classifier Quality

Math facts!



- The signed distance from a hyperplane defined by $!, !_0$ to a point $x^{!}$ is:

  = projection of $x^{!}$ on θ

  – signed distance of line to origin

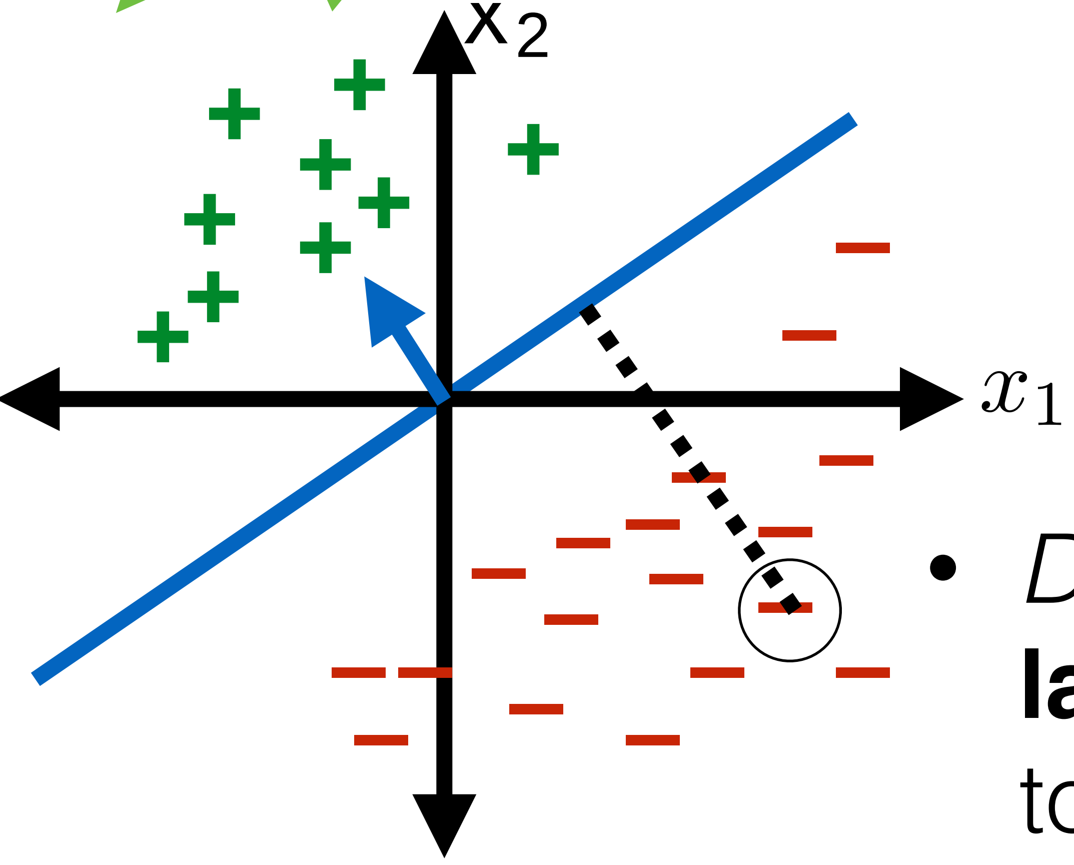
$$= \frac{!^{!}\, x^{"}}{!\,!\,!} \; " \; \frac{"\;!_0}{!\,!\,!} \; = \; \frac{!^{!}\, x^{"} + !_0}{!\,!\,!}$$

- *Definition*: The **margin of the labelled point** $(x^{!}, y^{!})$ with respect to the hyperplane defined by $!, !_0$ is:

$$y^{!} \; \dot{} \; \frac{!^{"}\, x^{!} + !_0}{!\,!\,!}$$

Let's Talk About Classifier Quality

Math facts!



- The signed distance from a hyperplane defined by $!,!_0$ to a point $x^!$ is:

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

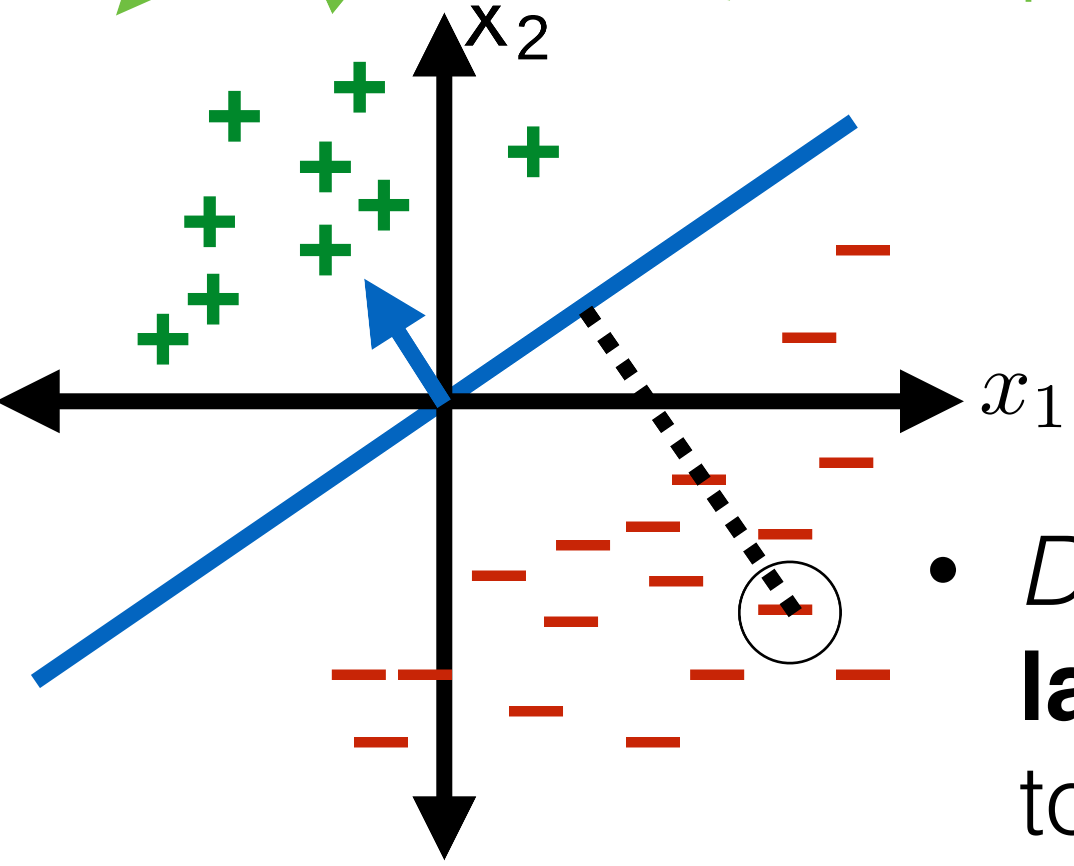
$$= \frac{!^! x^"}{!!!} \; " \; \frac{" \; !_0}{!!!} = \frac{!^! x^" + !_0}{!!!}$$

- *Definition*: The **margin of the labelled point** $(x^!, y^!)$ with respect to the hyperplane defined by $!,!_0$ is:

$$y^! \; \dot{} \; \frac{!^" x^! + !_0}{!!!}$$

Let’s Talk About Classifier Quality

Math facts!



- The signed distance from a hyperplane defined by ! , !$_0$ to a point x$^!$ is:

  = projection of x$^!$ on θ

  – signed distance of line to origin

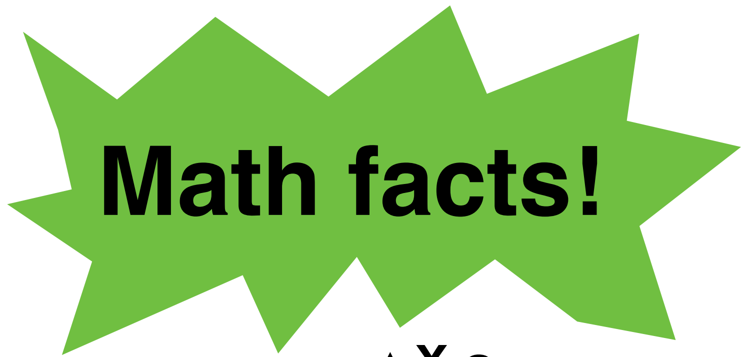
  $$= \frac{!^{!}\,x^{"}}{!\,!\,!} \;"\; \frac{"\;!_0}{!\,!\,!} \;=\; \frac{!^{!}\,x^{"} + !_0}{!\,!\,!}$$

- *Definition*: The **margin of the labelled point** (x$^!$, y$^!$) with respect to the hyperplane defined by ! , !$_0$ is:

  $$y^{!} \;\dot{}\; \frac{!^{"}\,x^{!} + !_0}{!\,!\,!}$$

- *Definition*: The **margin of the training set** D$_n$ with respect to the hyperplane defined by ! , !$_0$ is:

Let's Talk About Classifier Quality

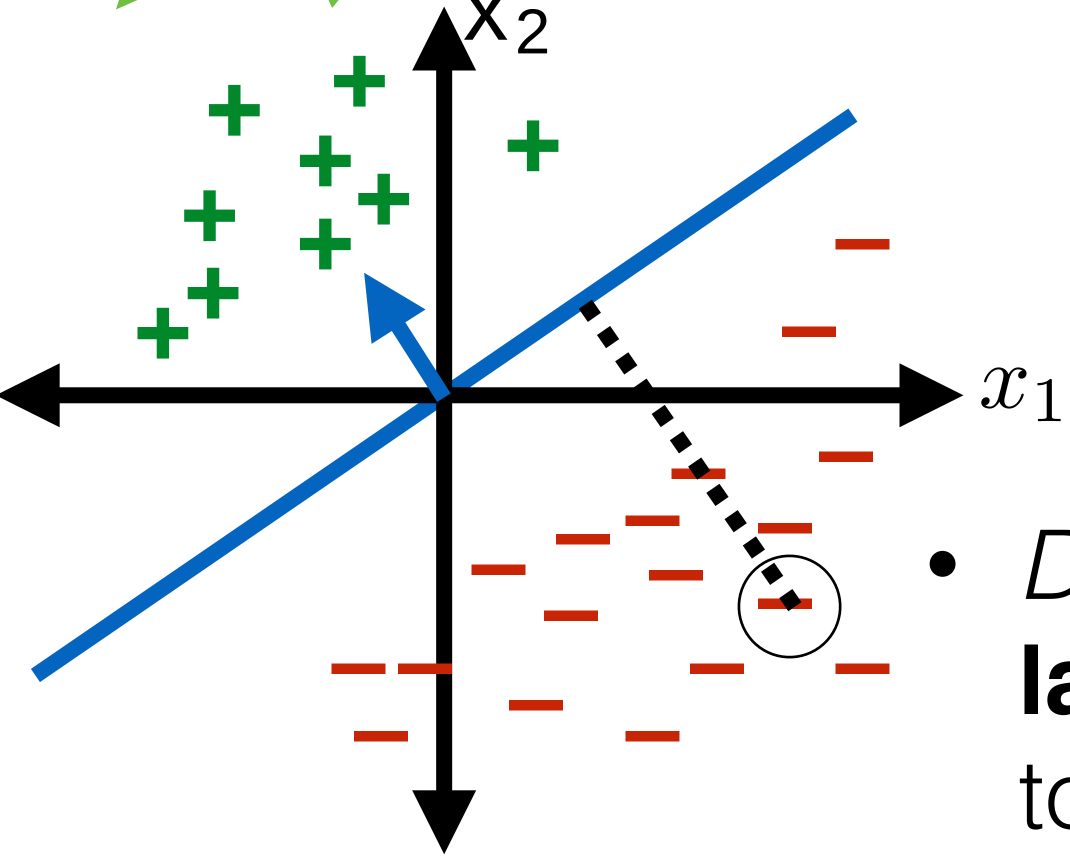


- The signed distance from a hyperplane defined by $! , !_0$ to a point $x^!$ is:

  $=$ projection of $x^!$ on θ

  $-$ signed distance of line to origin

  $$= \frac{!^! x^"}{!!!} " \frac{" !_0}{!!!} = \frac{!^! x^" + !_0}{!!!}$$



- *Definition*: The **margin of the labelled point** $(x^! , y^!)$ with respect to the hyperplane defined by $! , !_0$ is:

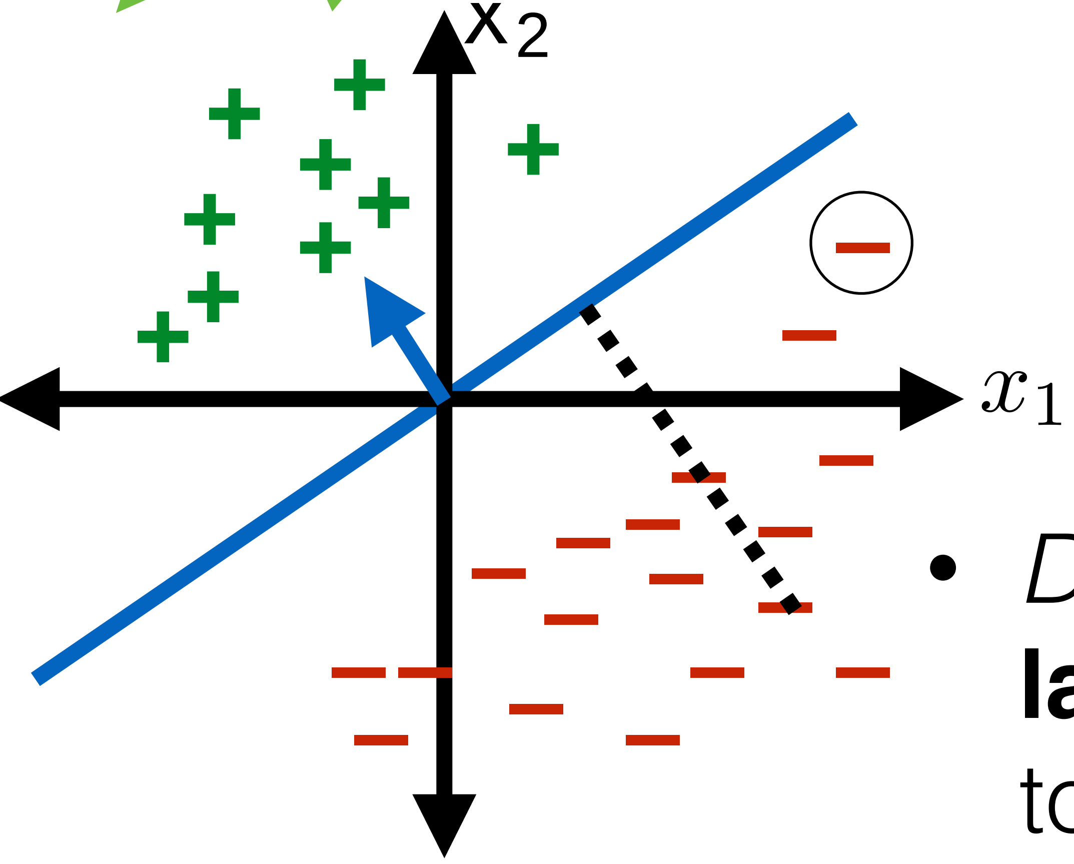
  $$y^! \cdot \frac{!^" x^! + !_0}{!!!}$$

- *Definition*: The **margin of the training set** D_n with respect to the hyperplane defined by $! , !_0$ is:

  $$\min_{i! \{1,\ldots,n\}} y^{(i)} \cdot \frac{!^" x^{(i)} + !_0}{!!!}$$

Let's Talk About Classifier Quality

Math facts!



- The signed distance from a hyperplane defined by $!, !_0$ to a point $x^!$ is:

  = projection of $x^!$ on θ

  – signed distance of line to origin

$$= \frac{!^! x^"}{!!!} " \frac{" !_0}{!!!} = \frac{!^! x^" + !_0}{!!!}$$

- *Definition*: The **margin of the labelled point** $(x^!, y^!)$ with respect to the hyperplane defined by $!, !_0$ is:

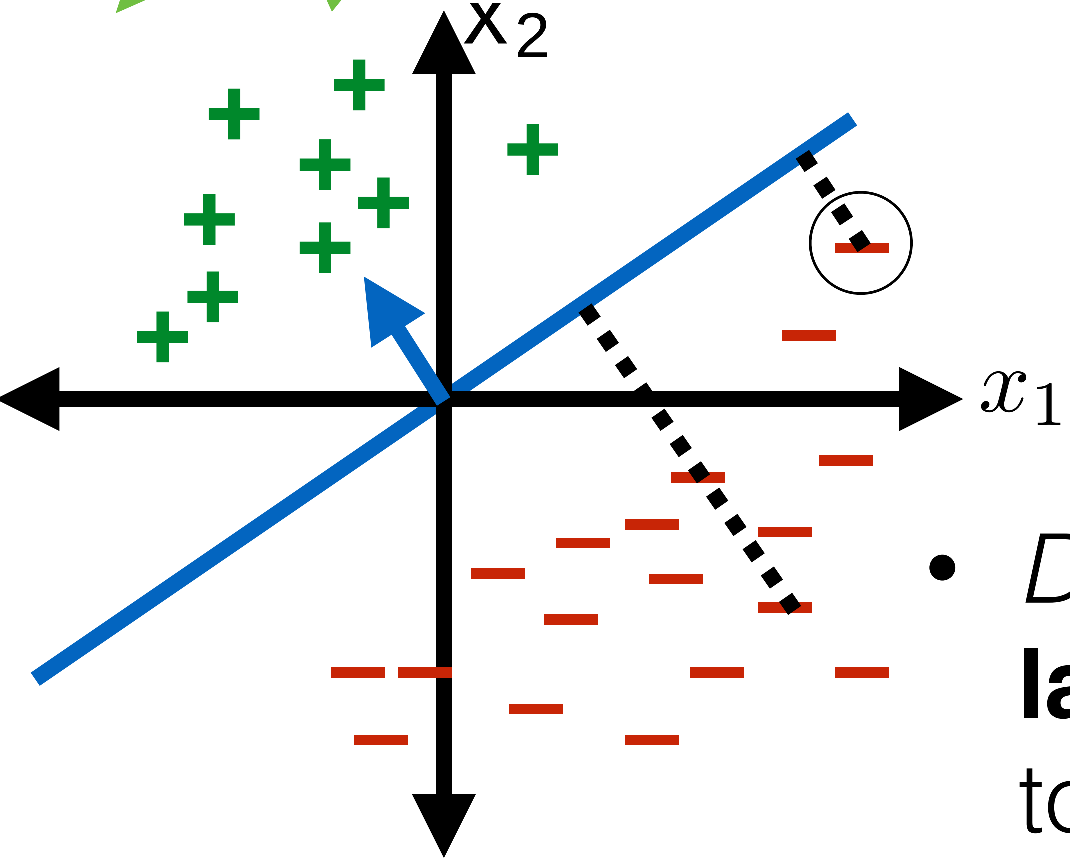
$$y^! \cdot \frac{!^" x^! + !_0}{!!!}$$

- *Definition*: The **margin of the training set** D_n with respect to the hyperplane defined by $!, !_0$ is:

$$\min_{i! \{1,\dots,n\}} y^{(i)} \cdot \frac{!^" x^{(i)} + !_0}{!!!}$$

Let’s Talk About Classifier Quality

Math facts!



- The signed distance from a hyperplane defined by ! , !$_0$ to a point x$^!$ is:

  = projection of x$^!$ on θ

  – signed distance of line to origin

  $$= \frac{!^! \, x^"}{!\,!\,!} " \frac{" \, !_0}{!\,!\,!} = \frac{!^! \, x^" + !_0}{!\,!\,!}$$

- *Definition*: The **margin of the labelled point** (x$^!$, y$^!$) with respect to the hyperplane defined by ! , !$_0$ is:

  $$y^! \;\dot{}\; \frac{!^" \, x^! + !_0}{!\,!\,!}$$

- *Definition*: The **margin of the training set** D$_n$ with respect to the hyperplane defined by ! , !$_0$ is:

  $$\min_{i!\{1,\ldots,n\}} y^{(i)} \;\dot{}\; \frac{!^" \, x^{(i)} + !_0}{!\,!\,!}$$

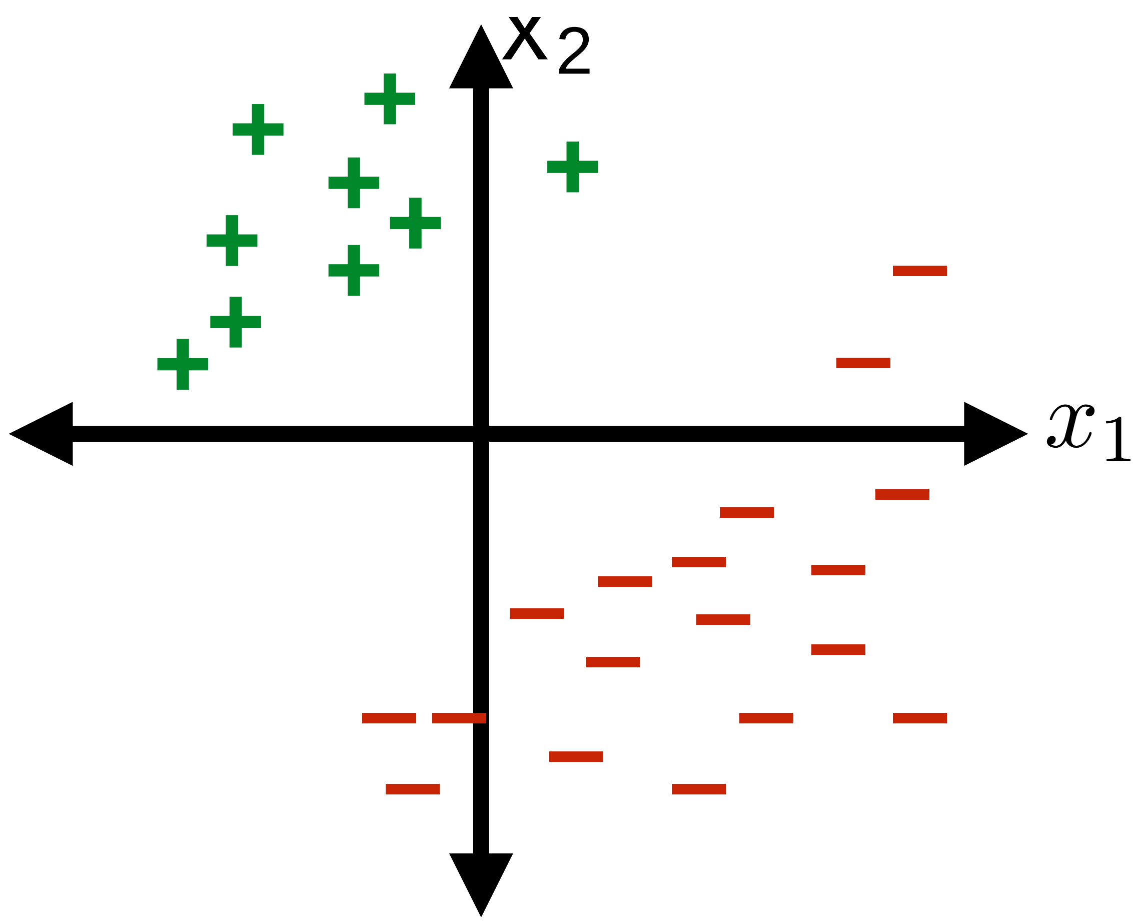
Theorem: Perceptron Performance

Theorem: Perceptron Performance

- **Assumptions**:

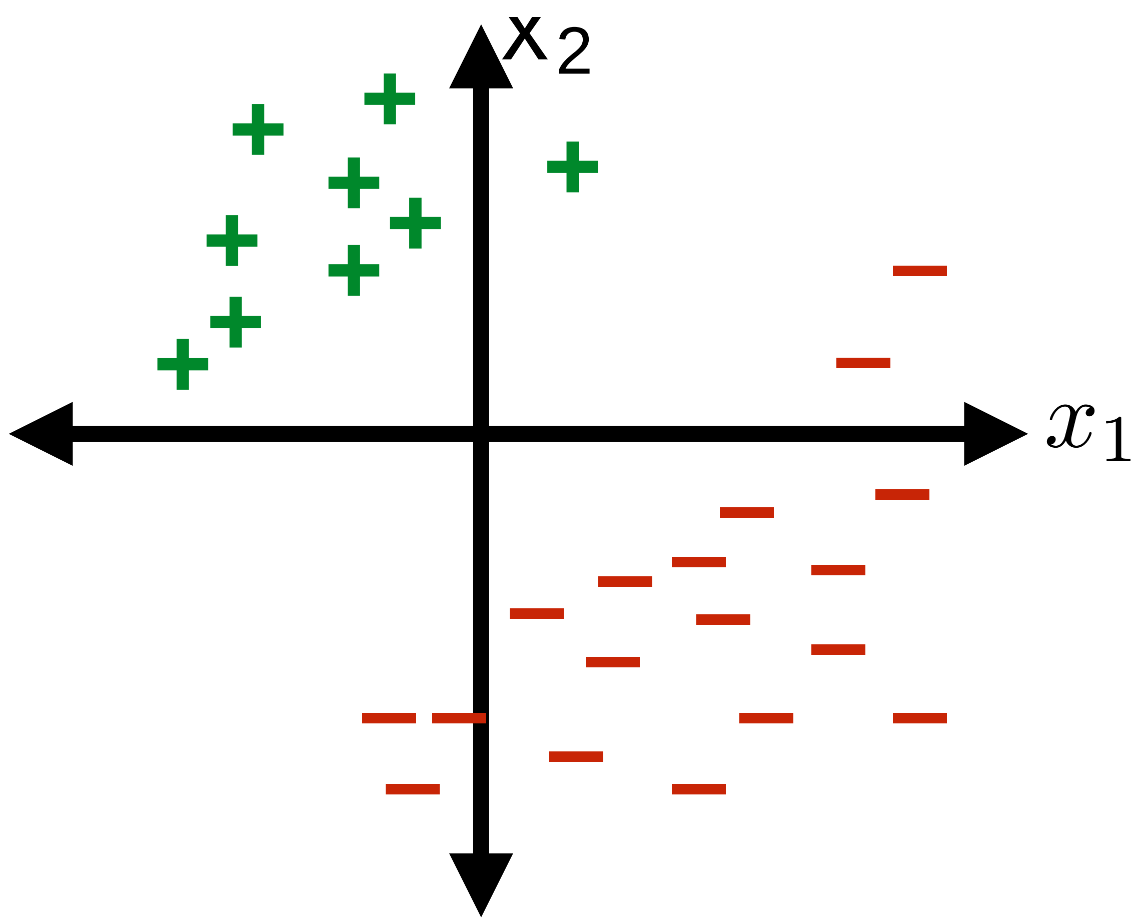
Theorem: Perceptron Performance

- **Assumptions**:



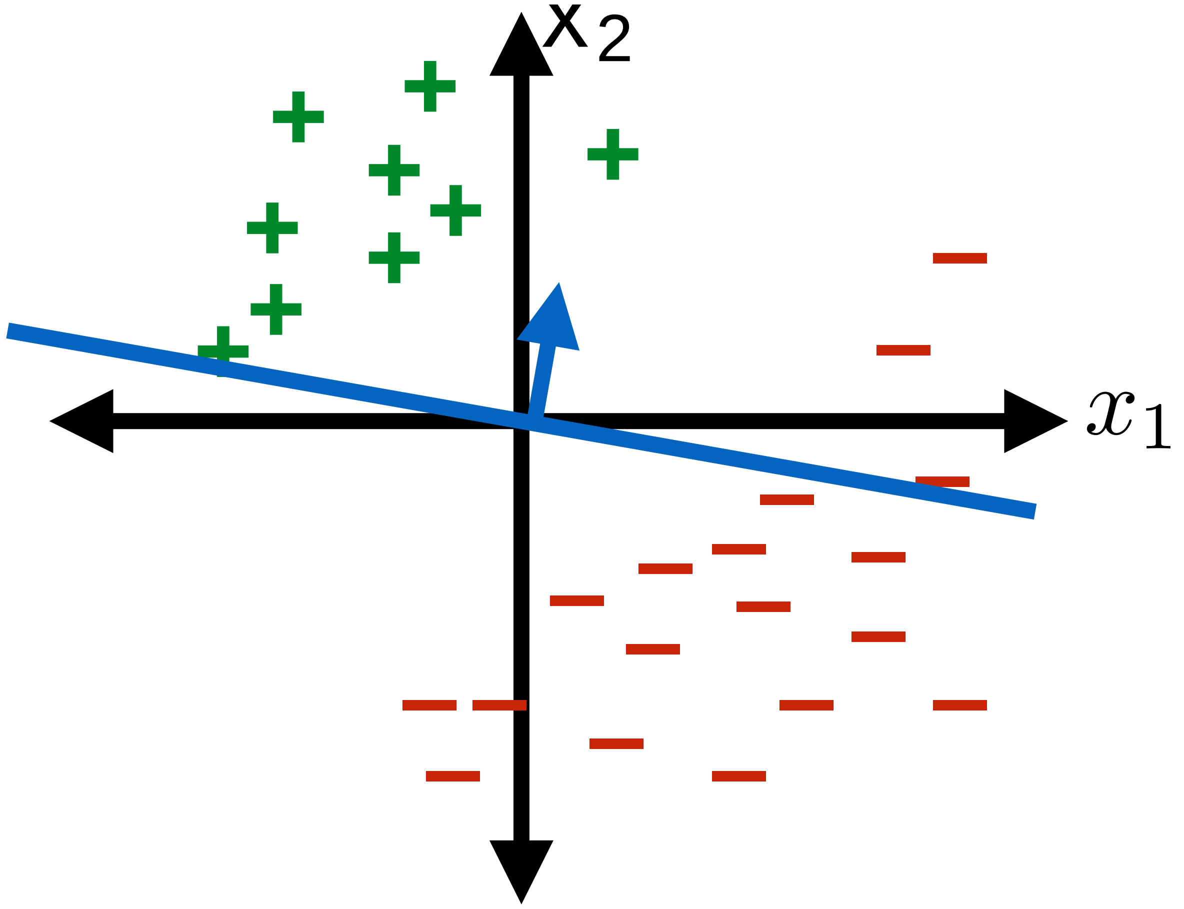
Theorem: Perceptron Performance

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



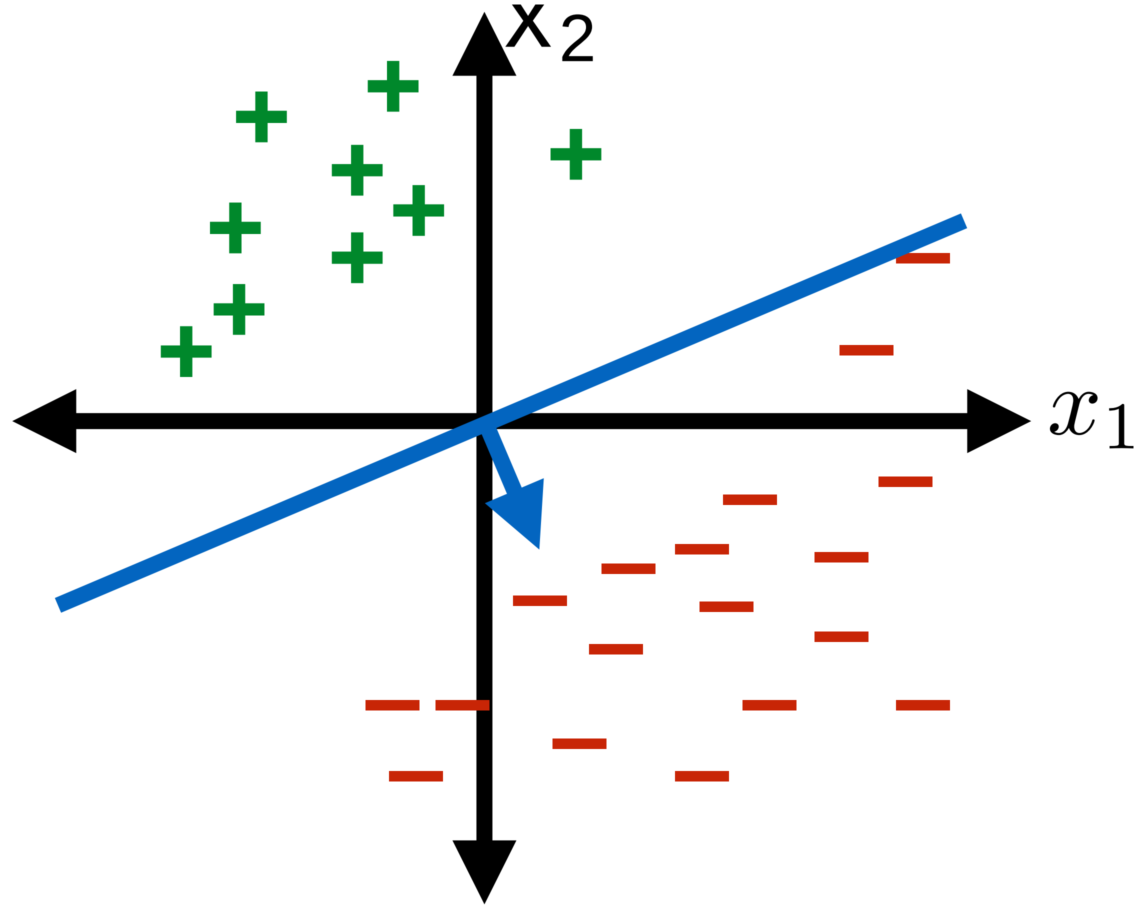
Theorem: Perceptron Performance

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



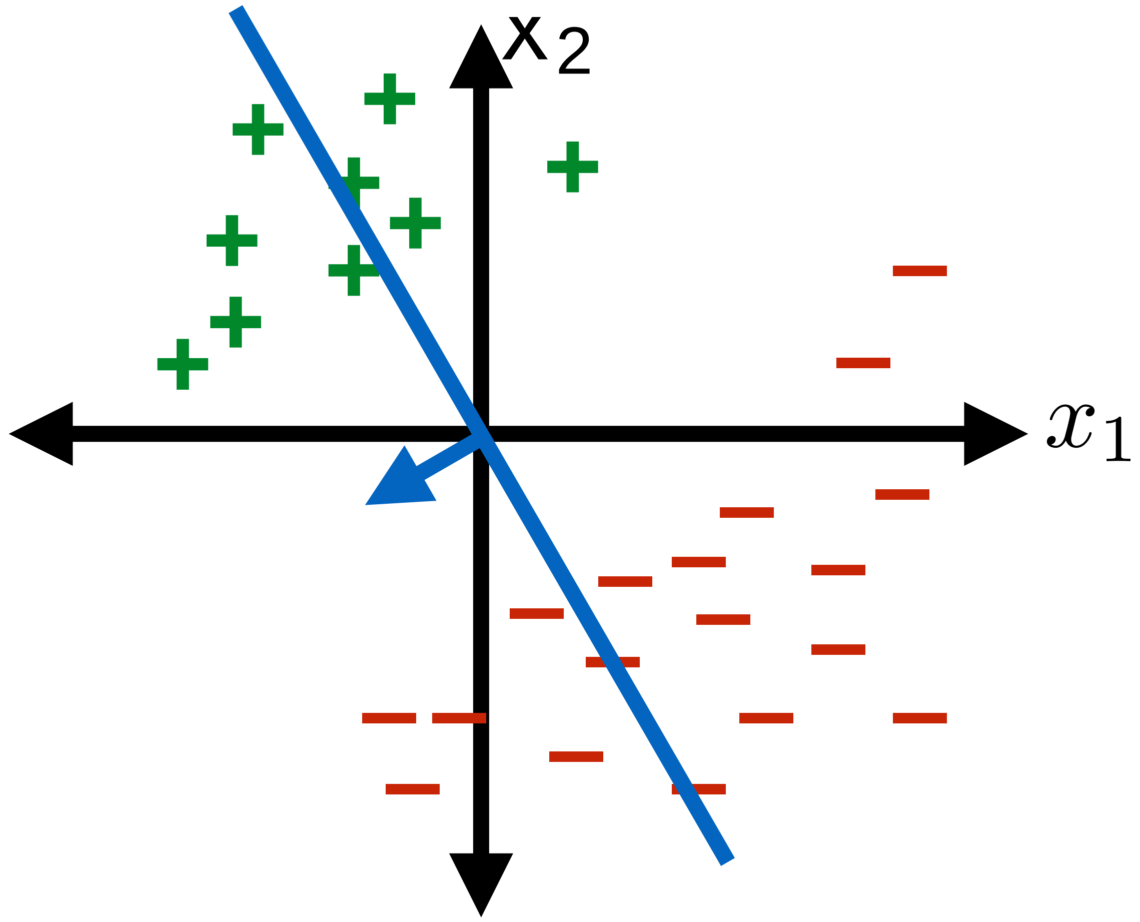
Theorem: Perceptron Performance

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



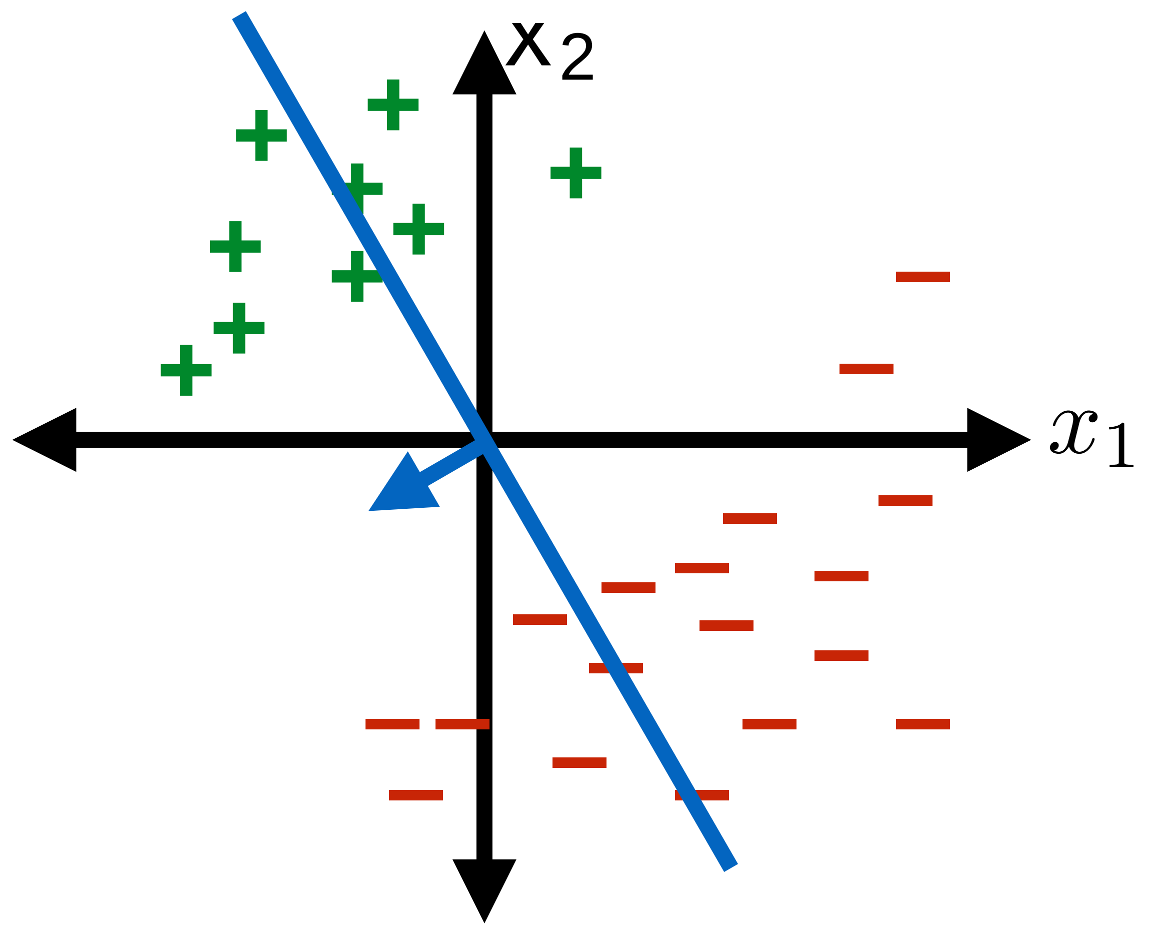
Theorem: Perceptron Performance

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



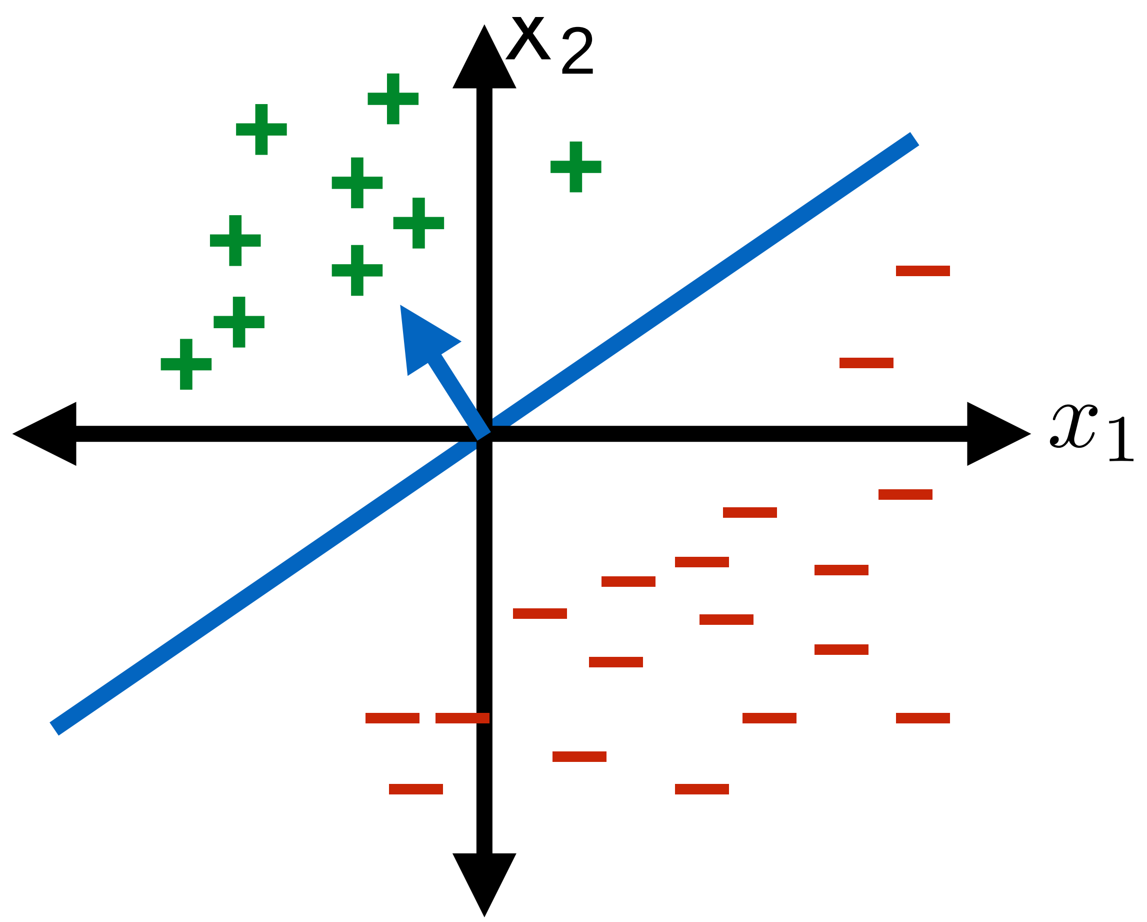
Theorem: Perceptron Performance

- **Assumptions**:
  A. Our hypothesis class = classifiers with separating hyperplanes that pass through the origin (i.e. $\theta_0 = 0$)
  B. There exist $!^{!}$ and $!$ such that $! > 0$ and, for every $i ! \{1, . . . , n\}$, we have $y^{(i)} \frac{! ^{!"} \; x^{(i)}}{!!!} > "$



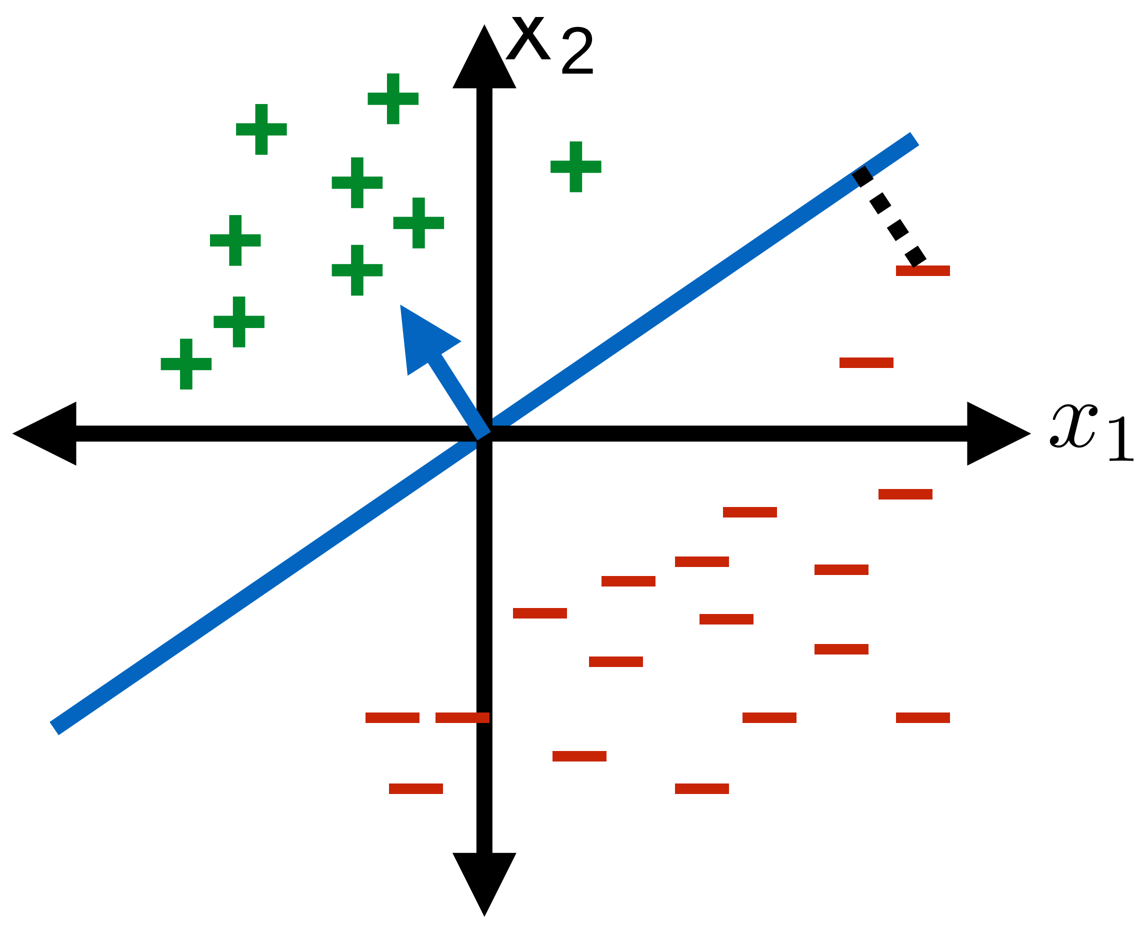
Theorem: Perceptron Performance

- **Assumptions**:
  A. Our hypothesis class = classifiers with separating hyperplanes that pass through the origin (i.e. $\theta_0 = 0$)
  B. There exist $!^{!}$ and $!$ such that $! > 0$ and, for every $i \; ! \; \{1, \ldots, n\}$, we have $y^{(i)} \cdot \frac{!^{!"} \; x^{(i)}}{!!!} > "$



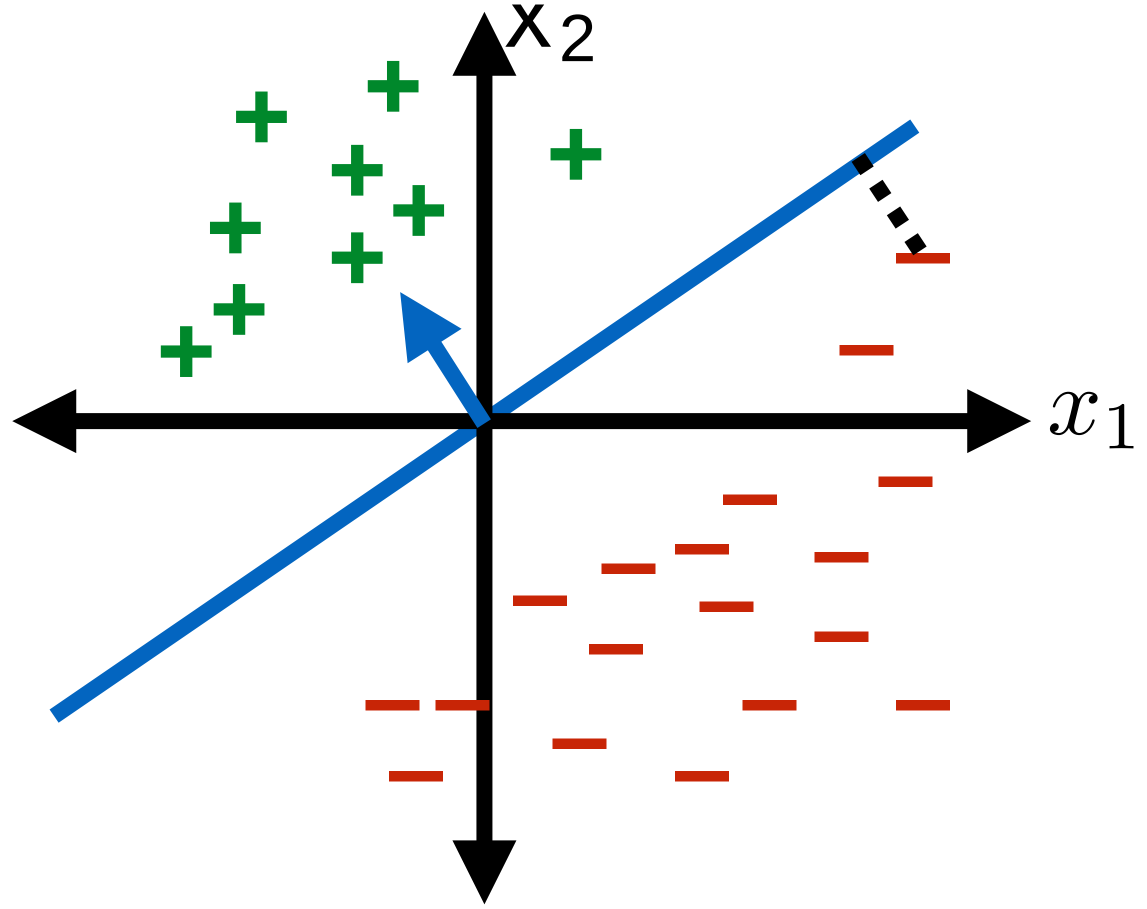
Theorem: Perceptron Performance

- **Assumptions**:
  A. Our hypothesis class = classifiers with separating hyperplanes that pass through the origin (i.e. $\theta_0 = 0$)
  B. There exist ! ! and ! such that ! > 0 and, for every i ! {1, . . . , n}, we have $y^{(i)} \frac{! !" \; x^{(i)}}{!!!} > "$



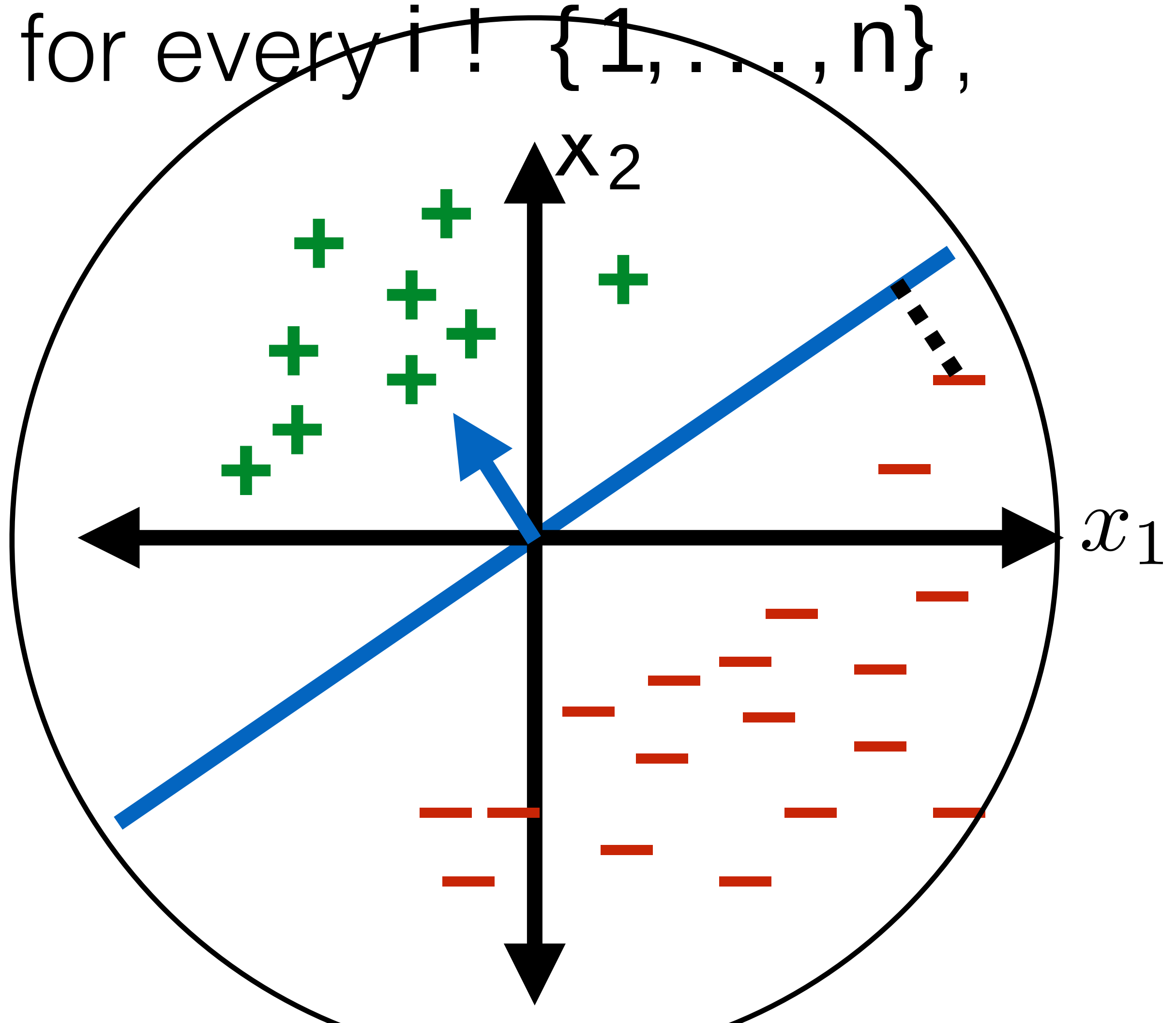
Theorem: Perceptron Performance

- **Assumptions**:
  A. Our hypothesis class = classifiers with separating hyperplanes that pass through the origin (i.e. $\theta_0 = 0$)
  B. There exist ! ' and ! such that ! > 0 and, for every i ! {1, . . . , n}, we have $y^{(i)} \cdot \frac{! !" \; x^{(i)}}{!!!} > "$
  C. There exists R such that, for every i ! {1, . . . , n}, we have $! x^{(i)} ! " \; R$



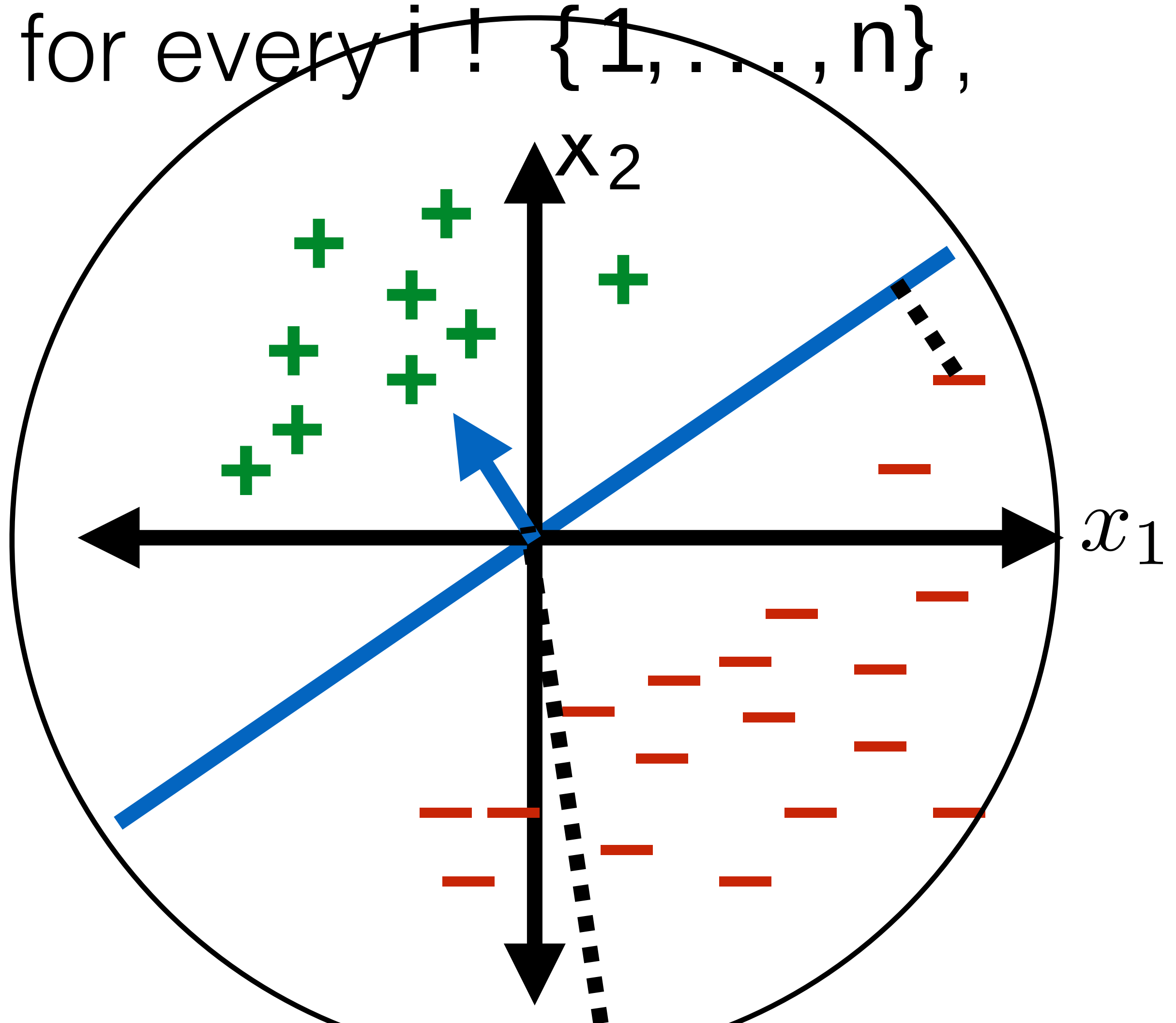
Theorem: Perceptron Performance

- **Assumptions**:
  A. Our hypothesis class = classifiers with separating hyperplanes that pass through the origin (i.e. $\theta_0 = 0$)
  B. There exist ! ' and ! such that ! > 0 and, for every i ! {1, . . . , n}, we have $y^{(i)} \cdot \frac{! !" \; x^{(i)}}{!!!} > "$
  C. There exists R such that, for every i ! {1, . . . , n}, we have ! x$^{(i)}$! " R



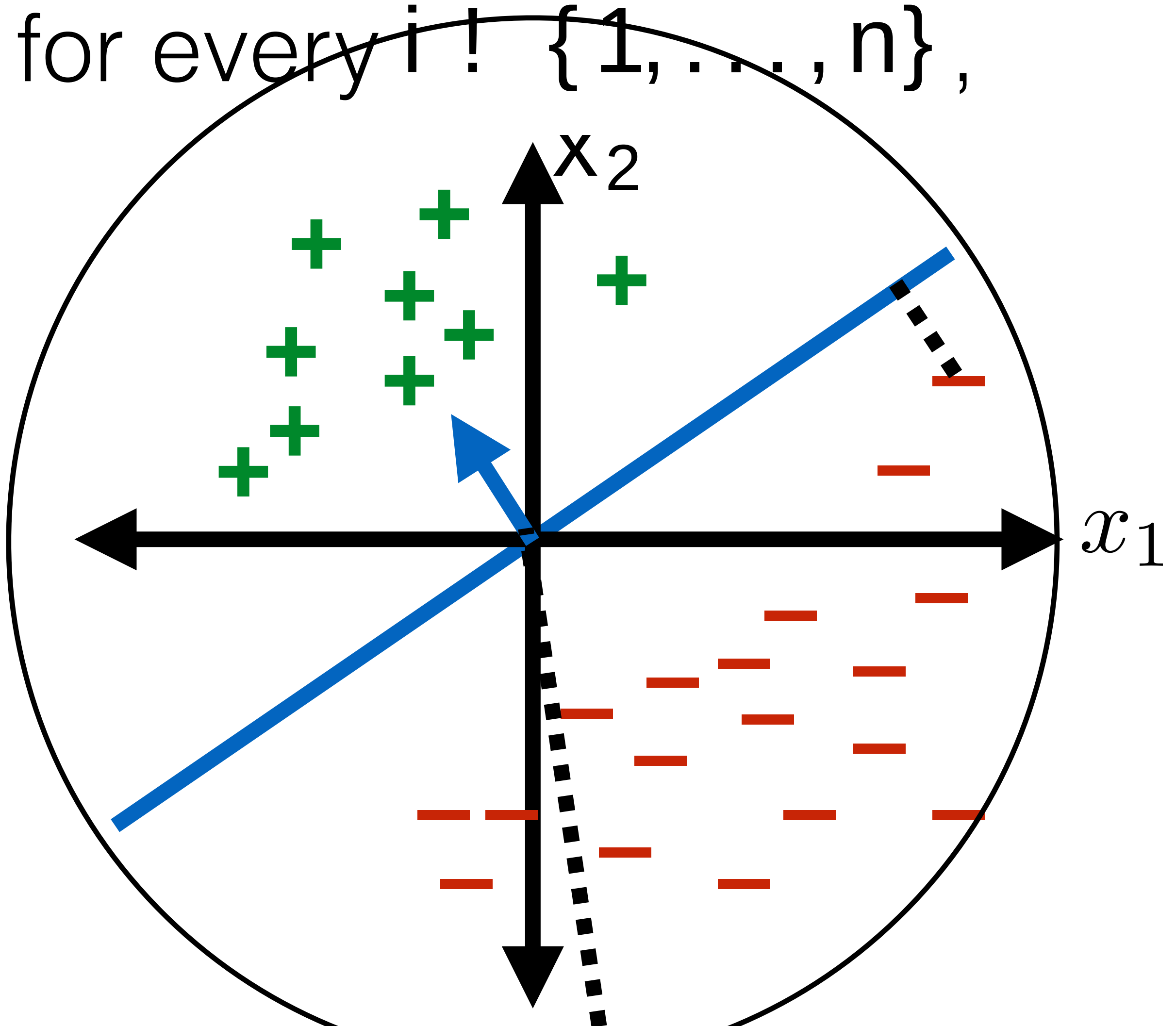
Theorem: Perceptron Performance

- **Assumptions**:
  A. Our hypothesis class = classifiers with separating hyperplanes that pass through the origin (i.e. $\theta_0 = 0$)
  B. There exist ! ! and ! such that ! > 0 and, for every i ! {1, . . . , n}, we have $y^{(i)} \cdot \frac{! !" \; x^{(i)}}{!!!} > "$
  C. There exists R such that, for every i ! {1, . . . , n}, we have ! x$^{(i)}$! " 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 ! $^{!}$ and ! such that ! > 0 and, for every i ! {1, . . . , n}, we have $y^{(i)} \dot{} \frac{! ^{!"} \; x^{(i)}}{!!!} > "$
  C. There exists R such that, for every i ! {1, . . . , n}, we have ! x$^{(i)}$! " R
- **Conclusion**: Then the perceptron algorithm will make at most (R/ !)^2 updates to ! . 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

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

$$x : !^{!}\ x + !_{0} = 0$$

Why classifiers through the origin?

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

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

    $$x : !^{!}\ x + !_0 = 0$$

  - Classifier without offset

Why classifiers through the origin?

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

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

$$\mathrm{x} : !^{!}\ \mathrm{x} + !_0 = 0$$

  - Classifier without offset

$$\mathrm{x}_{\mathrm{new}}\ !\ \mathrm{R}^{\mathrm{d+1}}, !_{\mathrm{new}}\ !\ \mathrm{R}^{\mathrm{d+1}}$$

$$\mathrm{x}_{\mathrm{new}} = [\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_\mathrm{d}, 1]^{!}\ , !_{\mathrm{new}} = [!_1, !_2, \ldots, !_\mathrm{d}, !_0]^{!}$$

Why classifiers through the origin?

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

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

$$x : !^{!}\ x + !_0 = 0$$

  - Classifier without offset

$$x_{\text{new}}\ !\ R^{d+1}, !_{\text{new}}\ !\ R^{d+1}$$

$$x_{\text{new}} = [x_1, x_2, \ldots, x_d, 1]^{!}, !_{\text{new}} = [!_1, !_2, \ldots, !_d, !_0]^{!}$$

$$x_{\text{new},1:d} : !^{!}_{\text{new}} x_{\text{new}} = 0$$

Why classifiers through the origin?

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

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

    $$x : !^{!} x + !_0 = 0$$

  - Classifier without offset

    $$x_{new} \; ! \; R^{d+1}, !_{new} \; ! \; R^{d+1}$$

    $$x_{new} = [x_1, x_2, \ldots, x_d, 1]^{!}, !_{new} = [!_1, !_2, \ldots, !_d, !_0]^{!}$$

    $$x_{new,1:d} : !_{new}^{!} x_{new} = 0$$

Why classifiers through the origin?

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

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

$$x : !^{!}\ x + !_0 \overset{<}{=} 0$$

  - Classifier without offset

$$x_{new}\ !\ \ R^{d+1}, !_{new}\ !\ \ R^{d+1}$$

$$x_{new} = [x_1, x_2, \ldots, x_d, 1]^{!}\ , !_{new} = [!_1, !_2, \ldots, !_d, !_0]^{!}$$

$$x_{new,1:d} : !^{!}_{new} x_{new} \overset{<}{=} 0$$

Why classifiers through the origin?

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

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

$$x : !^{!}\ x + !_0 \underset{\boxed{>}}{\overset{<}{=}} 0$$

  - Classifier without offset

$$x_{new}\ !\ R^{d+1}, !_{new}\ !\ R^{d+1}$$

$$x_{new} = [x_1, x_2, \ldots, x_d, 1]^{!}\ , !_{new} = [!_1, !_2, \ldots, !_d, !_0]^{!}$$

$$x_{new,1:d} : !^{!}_{new} x_{new} \underset{\boxed{>}}{\overset{<}{=}} 0$$

Why classifiers through the origin?

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

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

$$\mathrm{x} : !^{!}\ \mathrm{x} + !_0 \overset{<}{\underset{>}{=}} 0$$

  - Classifier without offset

$$\mathrm{x}_{\mathrm{new}}\ !\ \ \mathrm{R}^{\mathrm{d+1}}\ , !_{\mathrm{new}}\ !\ \ \mathrm{R}^{\mathrm{d+1}}$$

$$\mathrm{x}_{\mathrm{new}} = [\,\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_{\mathrm{d}}, 1]^{!}\ , !_{\mathrm{new}} = [\,!_1, !_2, \ldots, !_{\mathrm{d}}, !_0]^{!}$$

$$\mathrm{x}_{\mathrm{new},1:\mathrm{d}} : !^{!}_{\mathrm{new}} \mathrm{x}_{\mathrm{new}} \overset{<}{\underset{>}{=}} 0$$

Why classifiers through the origin?

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

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

    $$\mathrm{x} : !^{!}\ \mathrm{x} + !_0 \overset{<}{\underset{>}{=}} 0$$

  - Classifier without offset

    $$\mathrm{x}_{\mathrm{new}}\ !\ \ \mathrm{R}^{\mathrm{d+1}}, !_{\mathrm{new}}\ !\ \ \mathrm{R}^{\mathrm{d+1}}$$

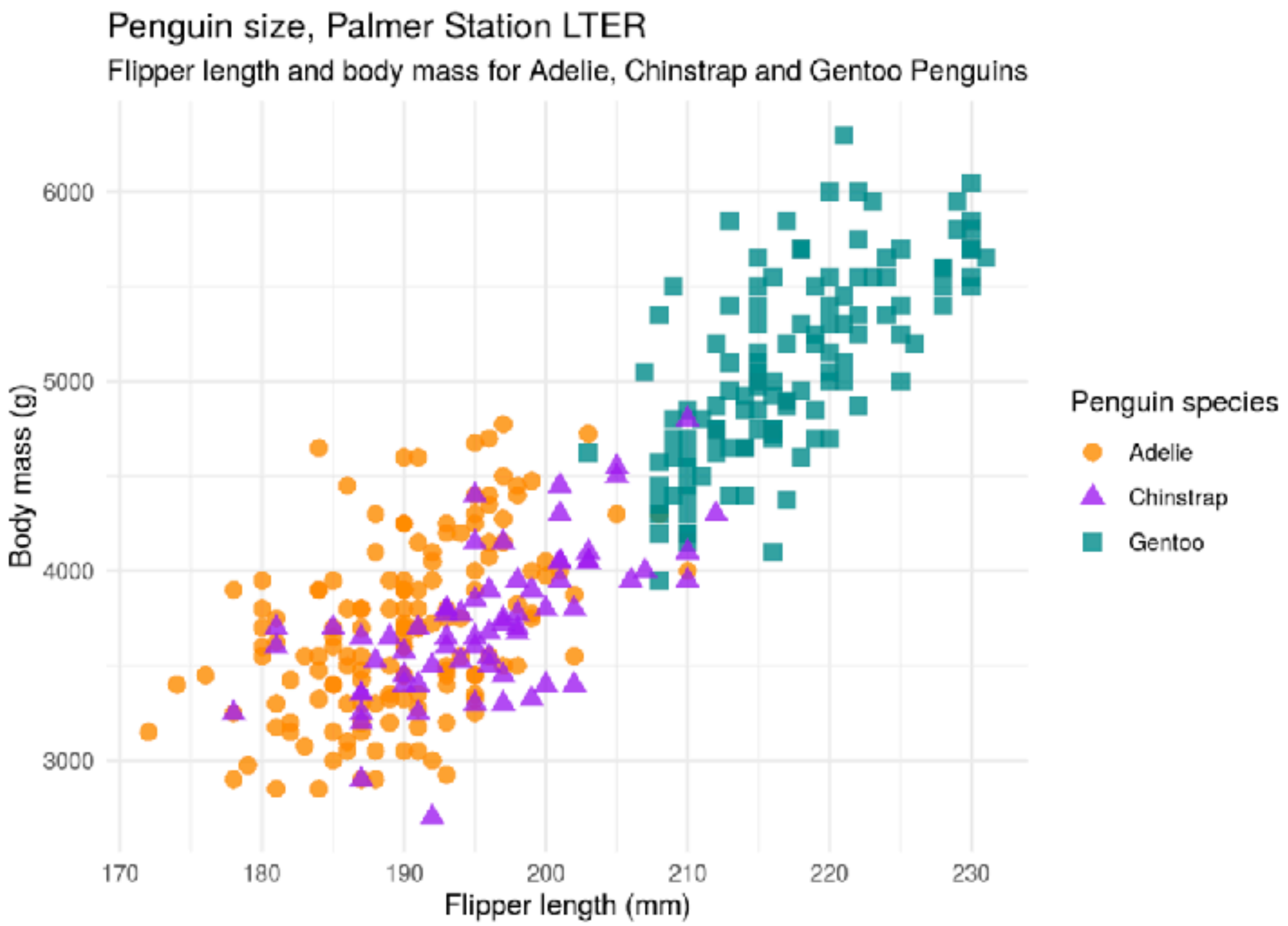
    $$\mathrm{x}_{\mathrm{new}} = [\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_\mathrm{d}, 1]^{!}\ , !_{\mathrm{new}} = [!_1, !_2, \ldots, !_\mathrm{d}, !_0]^{!}$$

    $$\mathrm{x}_{\mathrm{new,1:d}} : !^{!}_{\mathrm{new}} \mathrm{x}_{\mathrm{new}} \overset{<}{\underset{>}{=}} 0$$

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

Problem: data not linearly separable

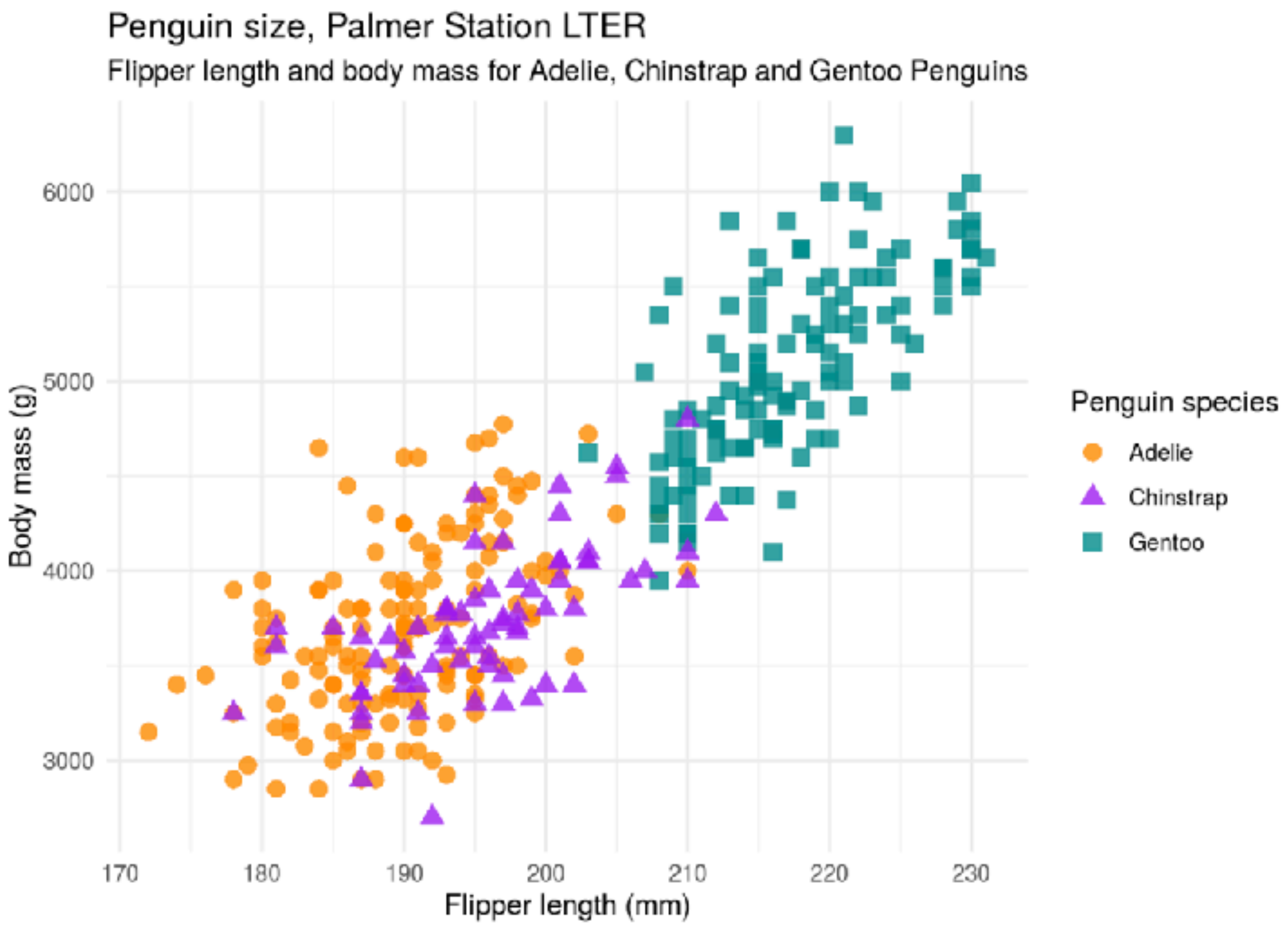
- Typical real data sets aren't linearly separable



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

Problem: data not linearly separable

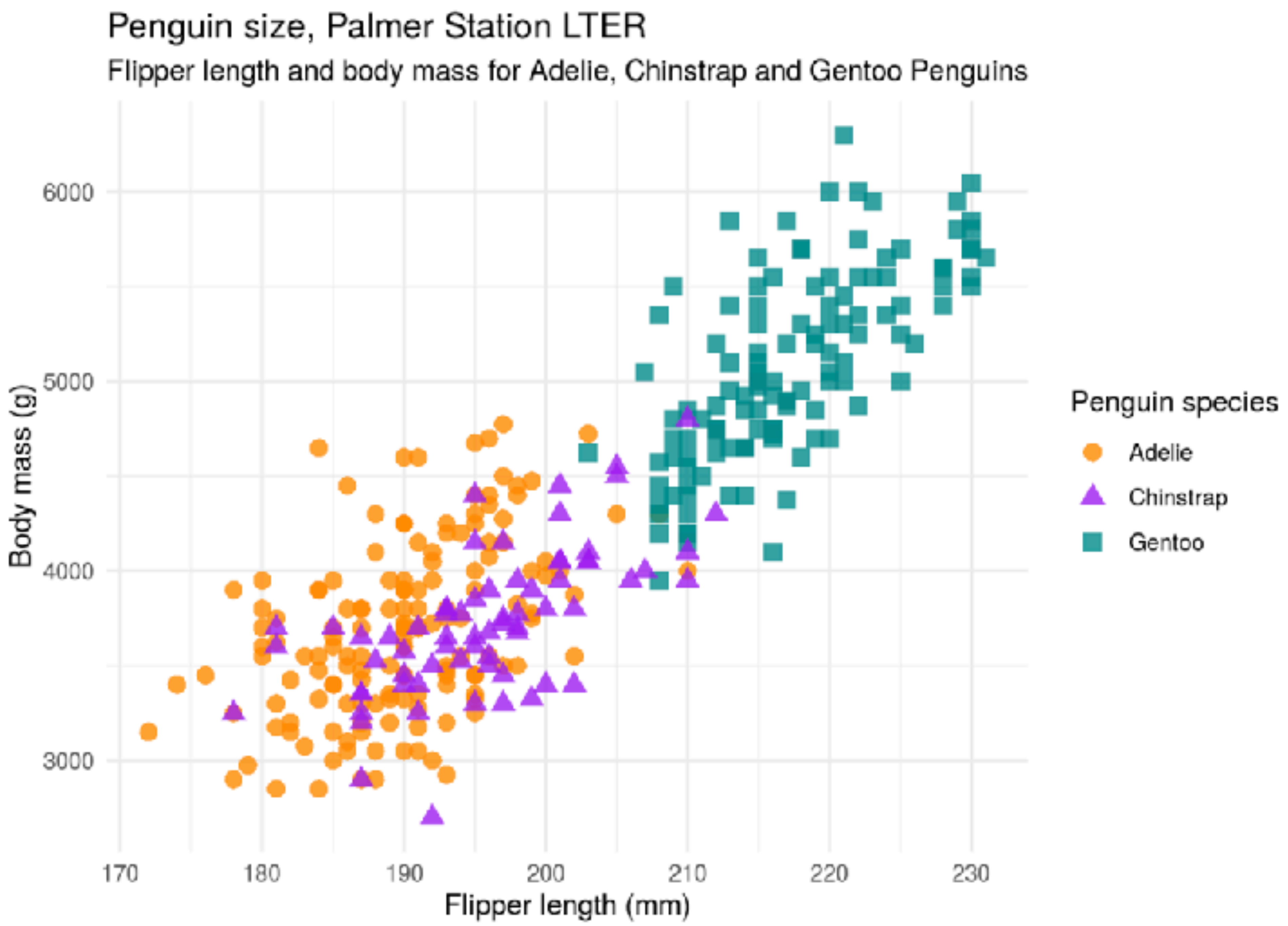
- Typical real data sets aren't linearly separable [demo]



[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]

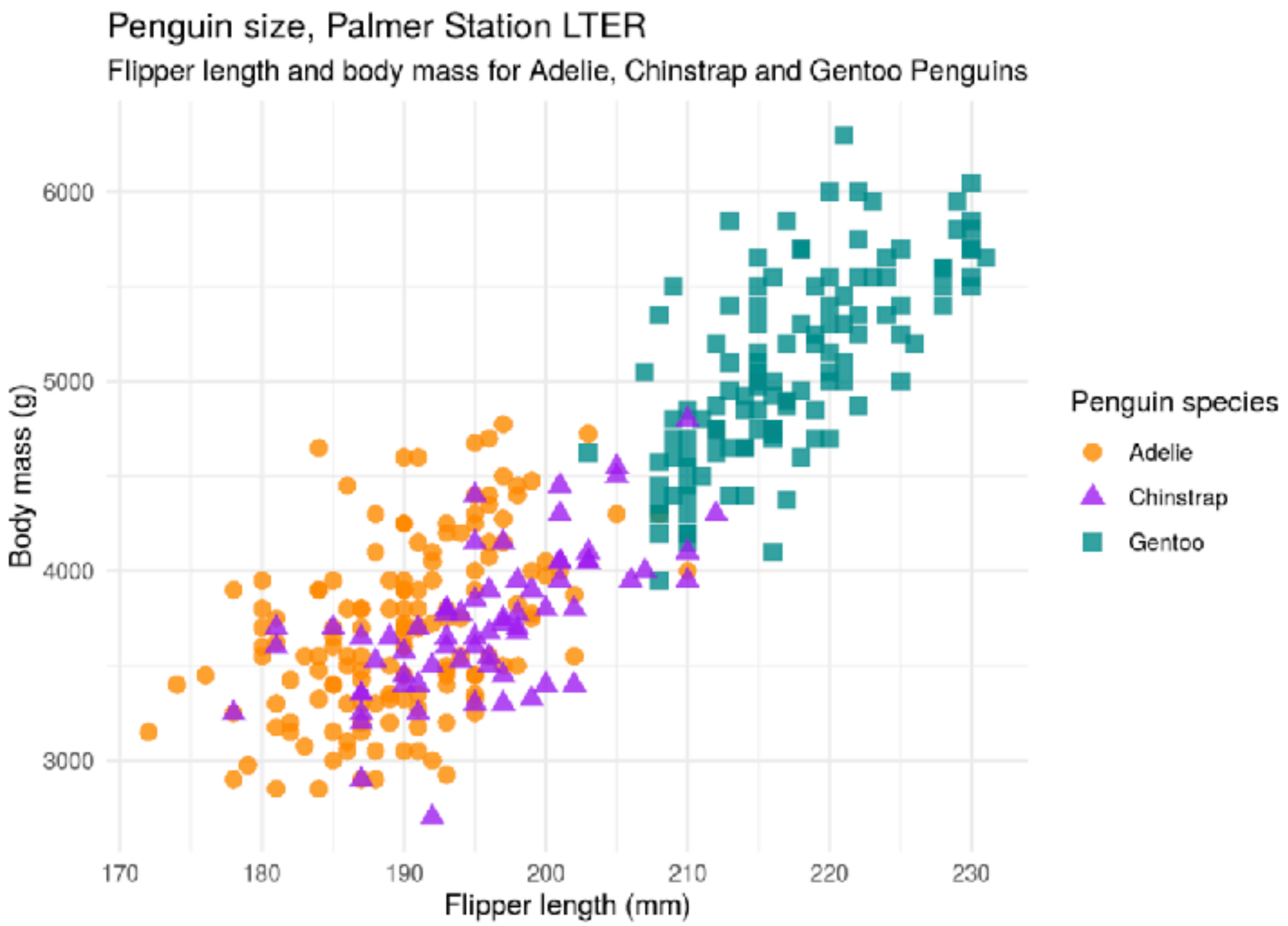


- What can we do?

[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

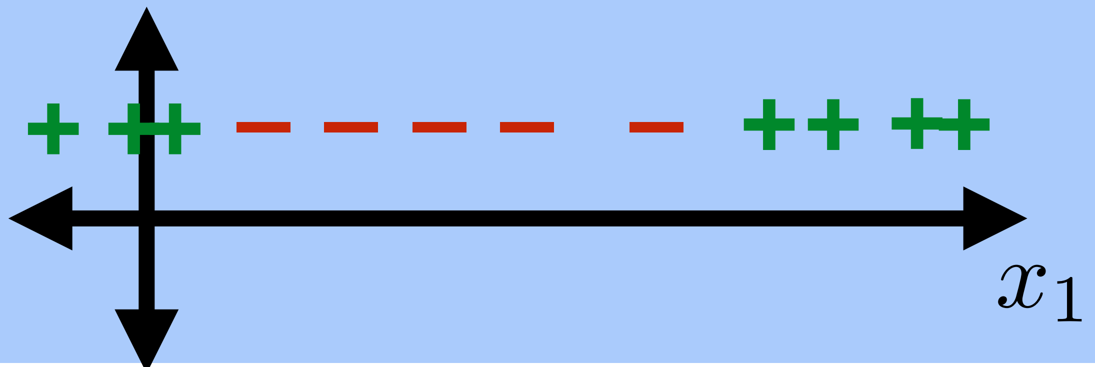
- **Binary/two-class classification**

Machine Learning Tasks

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

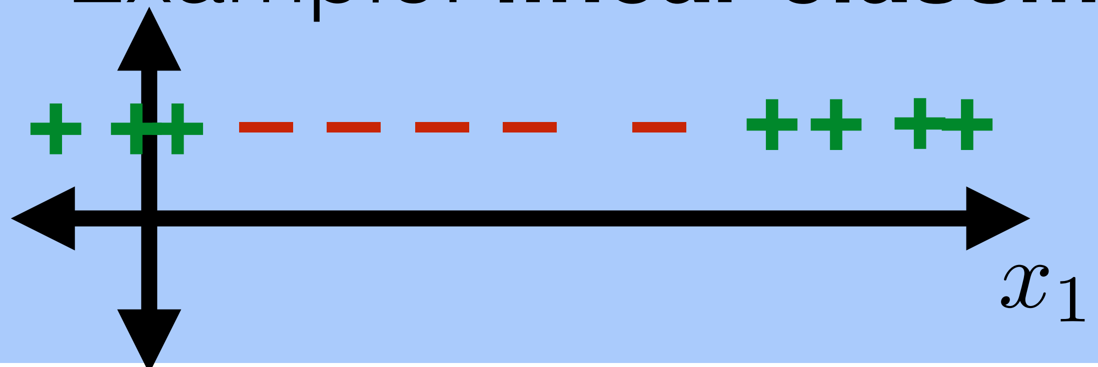
Machine Learning Tasks

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



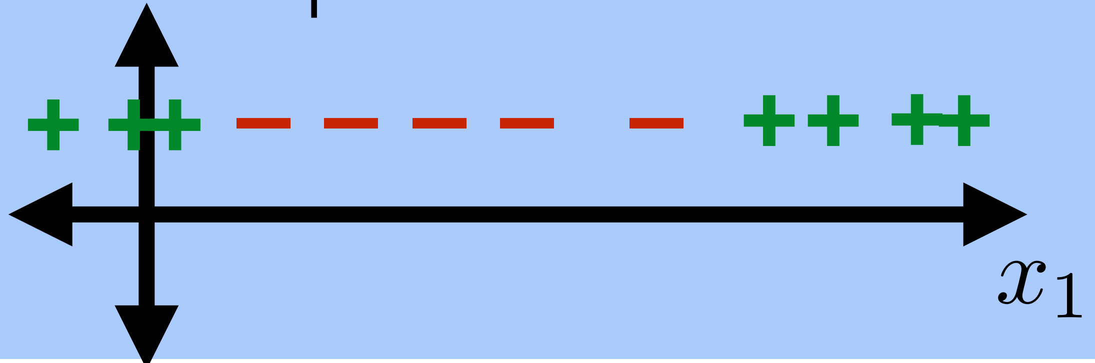
Machine Learning Tasks

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



Machine Learning Tasks

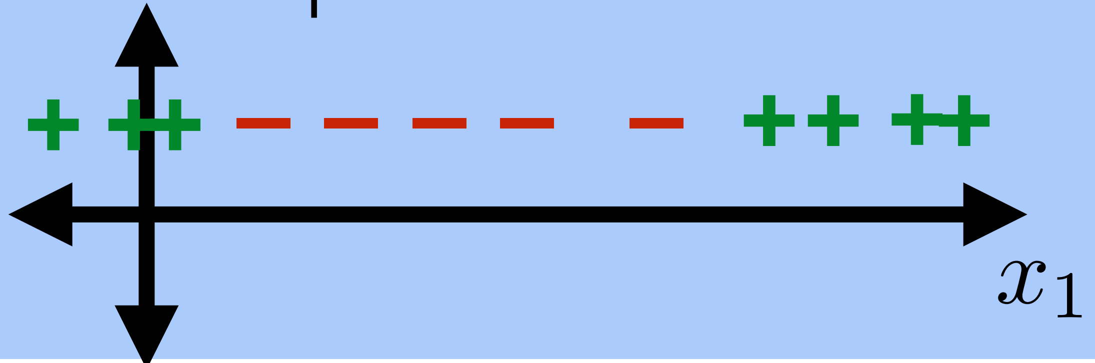
- **Binary/two-class classification:** Learn a mapping: $\mathbb{R}^d \rightarrow \{-1, +1\}$
  - Example: **linear classification**



- **Multi-class classification:**

Machine Learning Tasks

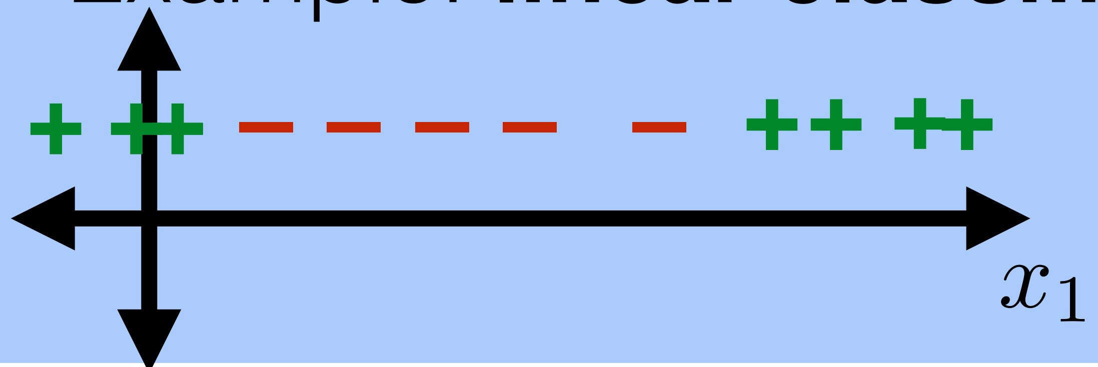
- **Binary/two-class classification:** Learn a mapping: $\mathbb{R}^d \rightarrow \{-1, +1\}$
  - Example: **linear classification**



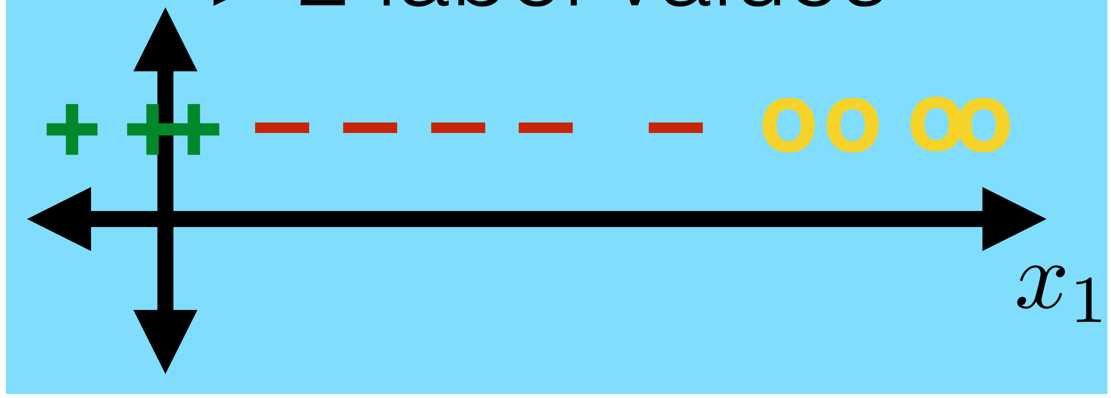
- **Multi-class classification:** > 2 label values

Machine Learning Tasks

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

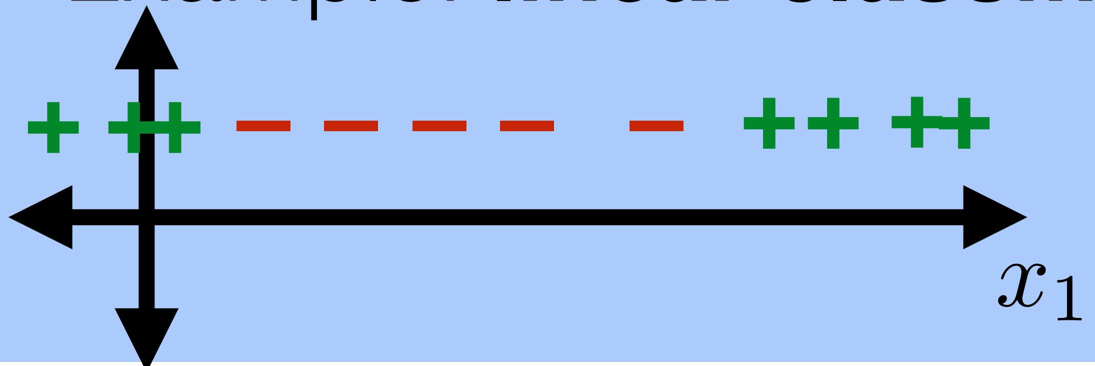


- **Multi-class classification:** > 2 label values

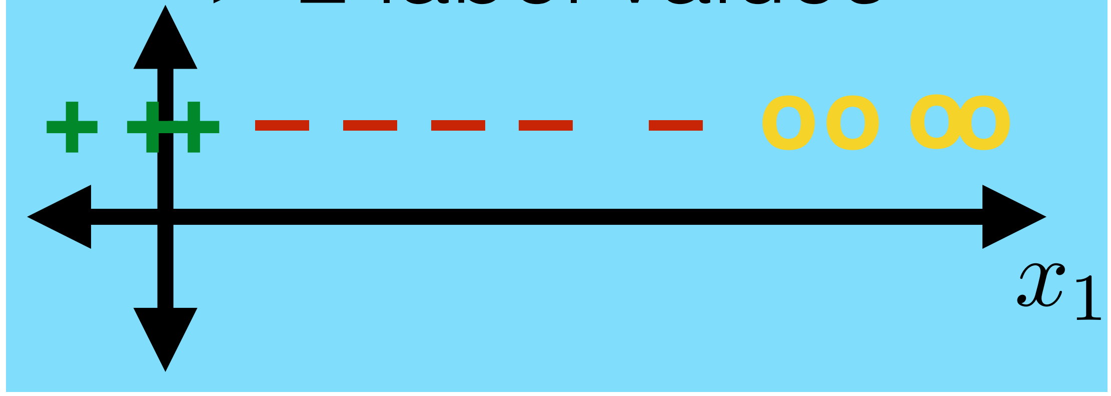


Machine Learning Tasks

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



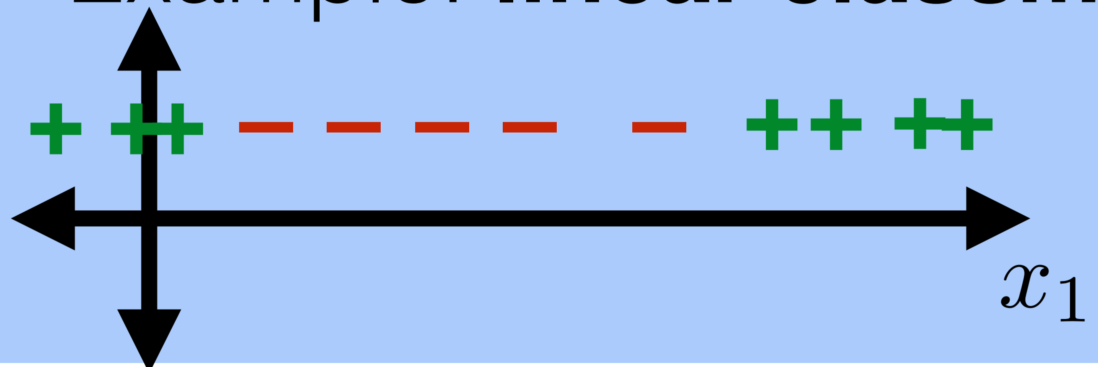
- **Multi-class classification:** > 2 label values



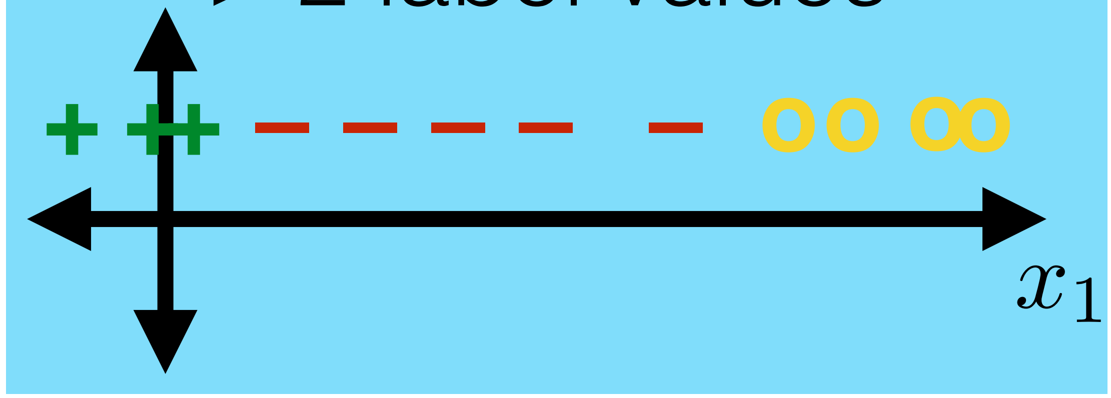
Machine Learning Tasks

- **Classification**

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



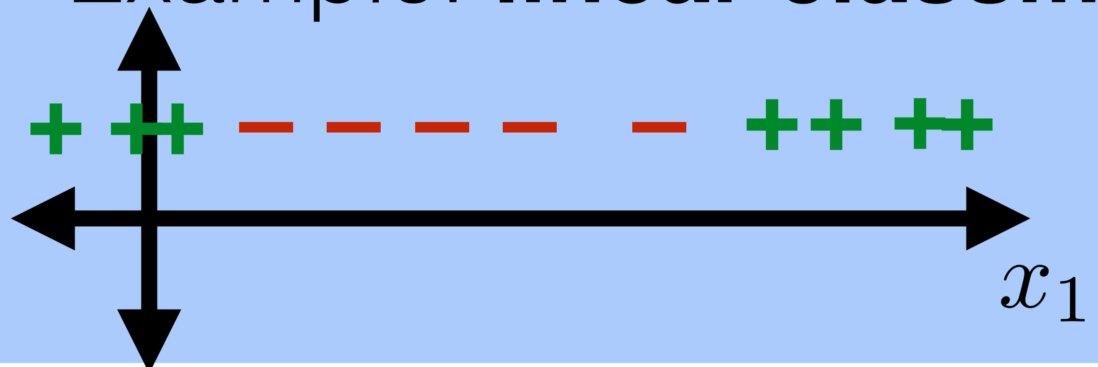
- **Multi-class classification:** > 2 label values



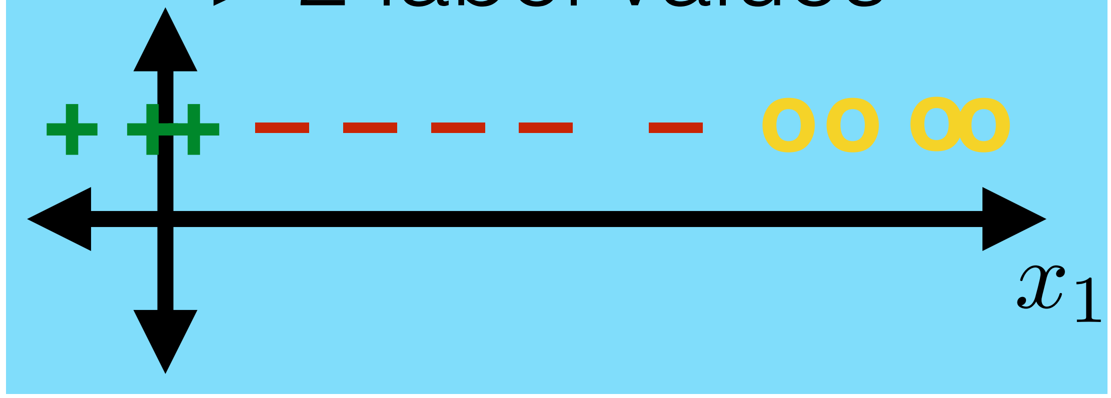
Machine Learning Tasks

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

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



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

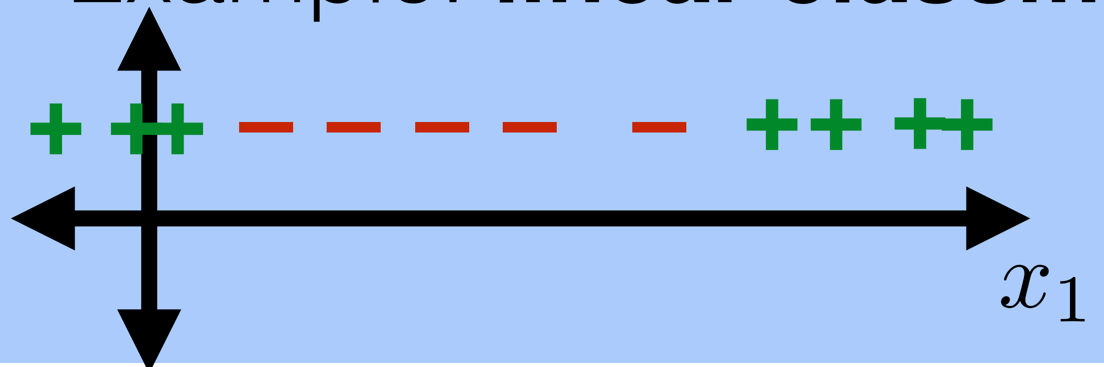


Machine Learning Tasks

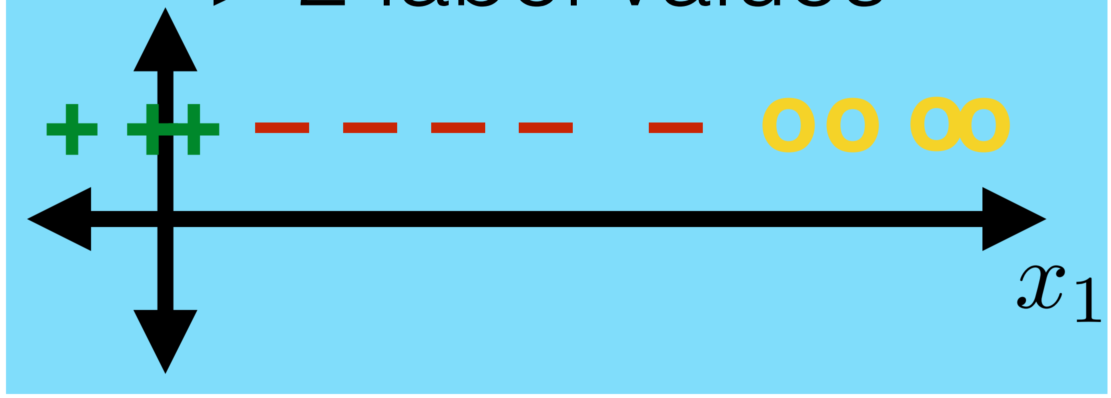
- **Regression**

- **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

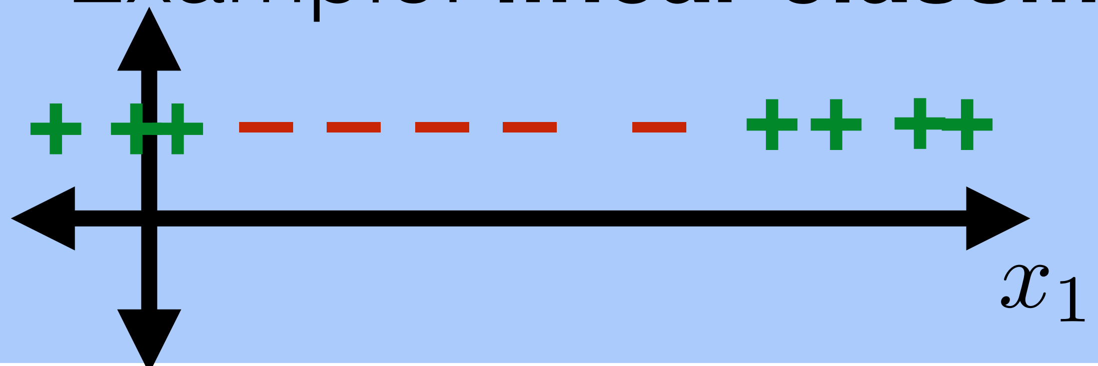


Machine Learning Tasks

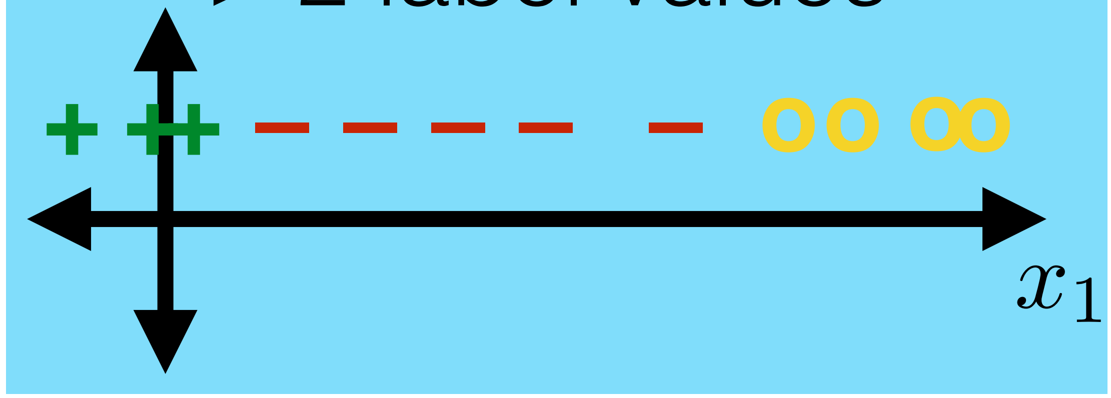
- **Regression:** Learn a mapping to continuous values: R^d ! R^k

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

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

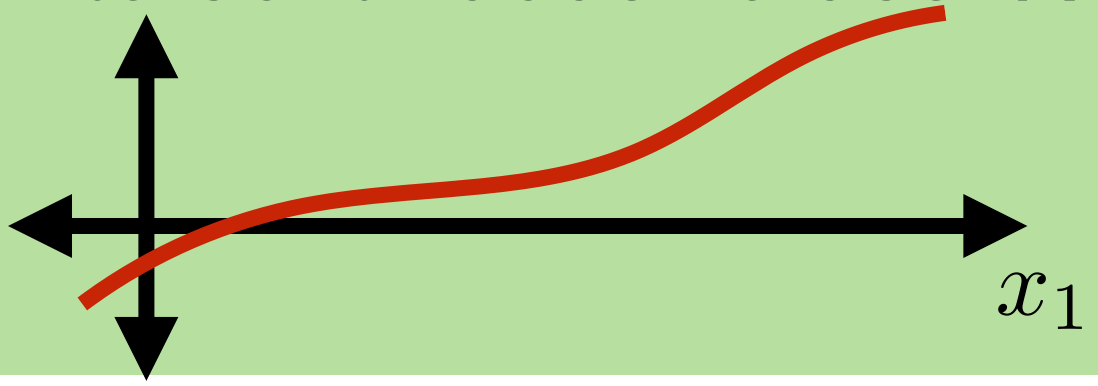


- **Multi-class classification:** > 2 label values



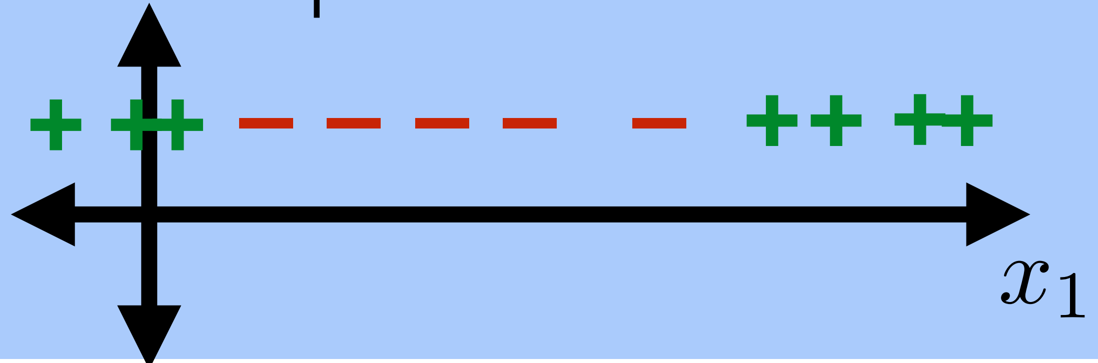
Machine Learning Tasks

- **Regression:** Learn a mapping to continuous values: R^d ! R^k

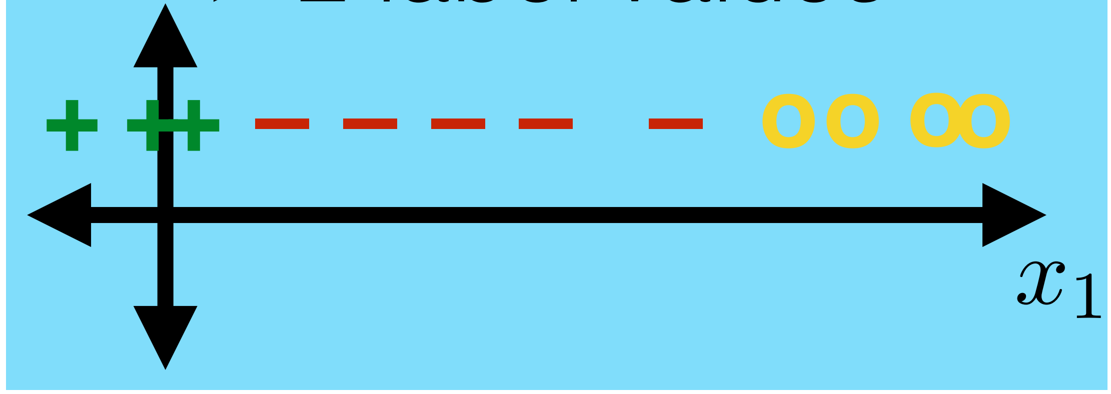


- **Classification:** Learn a mapping to a discrete set

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

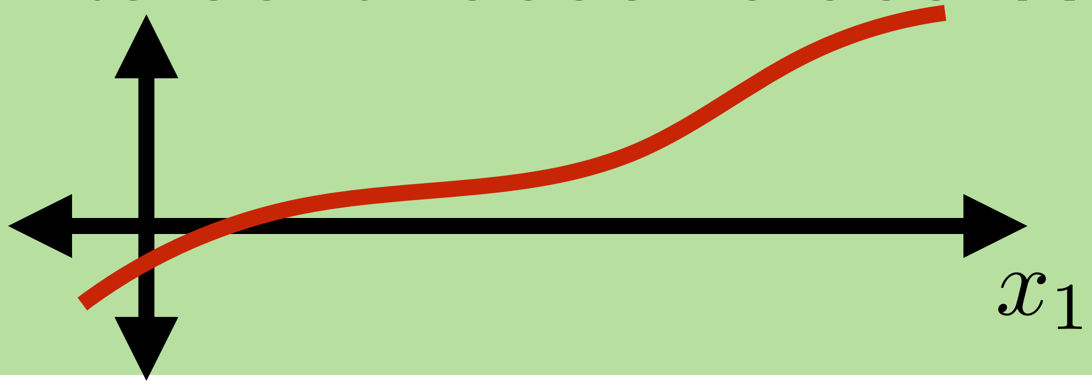


- **Multi-class classification:** > 2 label values



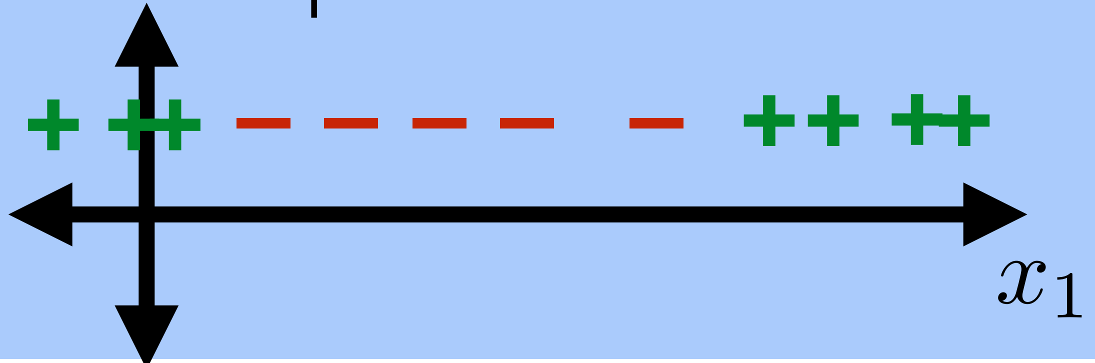
Machine Learning Tasks

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

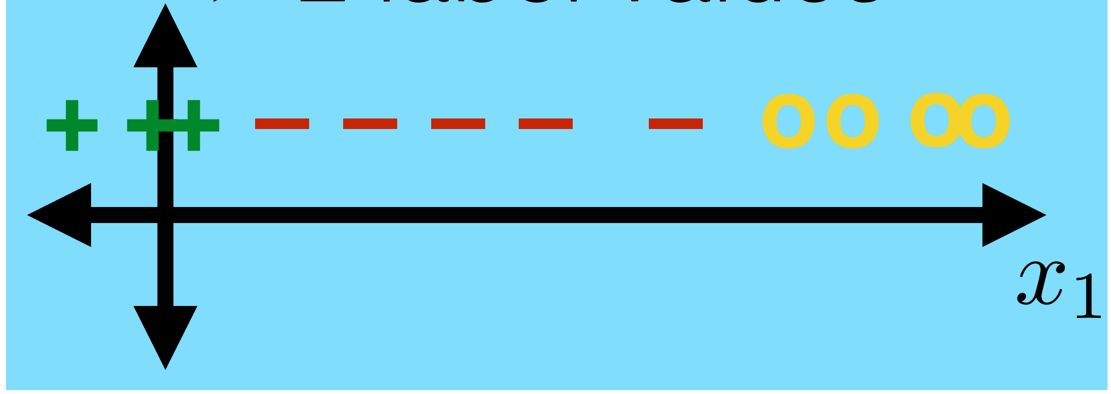


- **Classification:** Learn a mapping to a discrete set

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



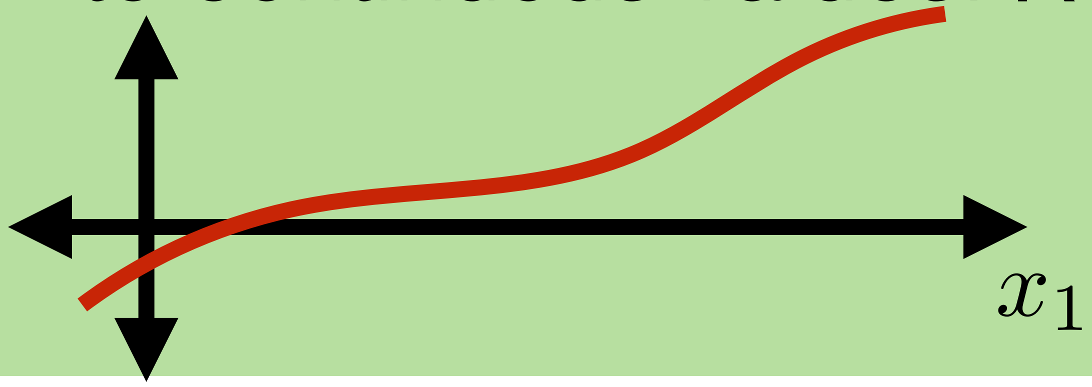
- **Multi-class classification:** > 2 label values



Machine Learning Tasks

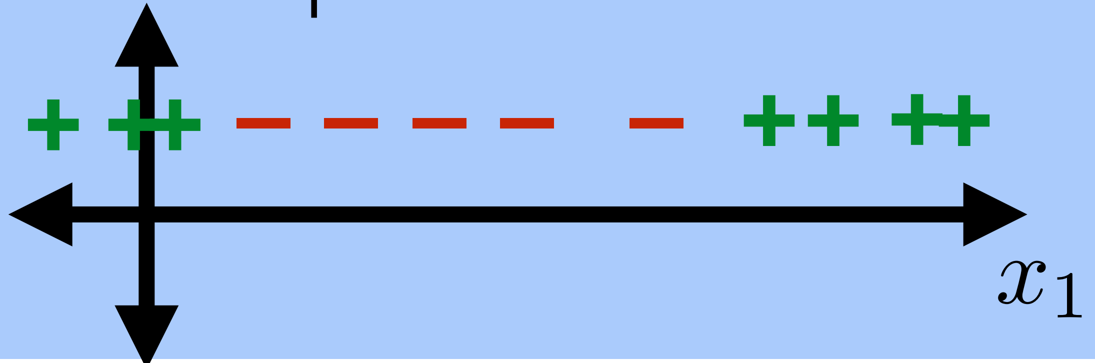
- **Supervised learning**

- **Regression:** Learn a mapping to continuous values: R^d ! R^k

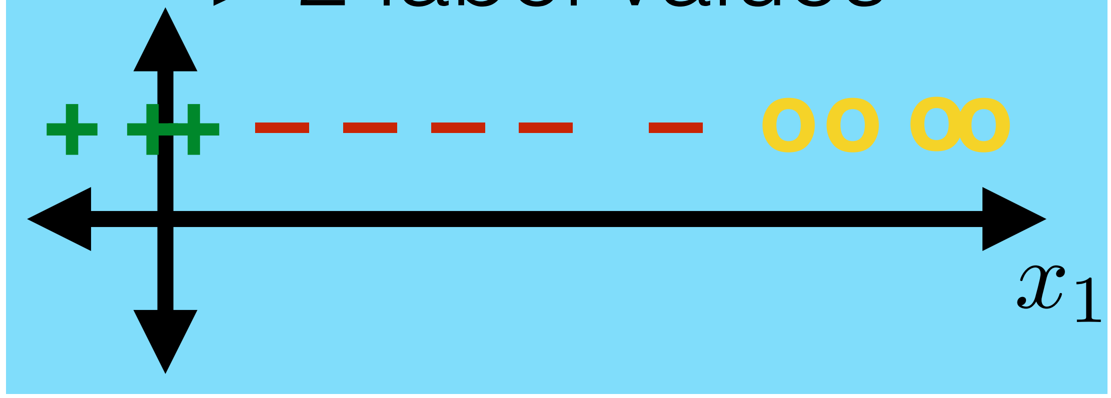


- **Classification:** Learn a mapping to a discrete set

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



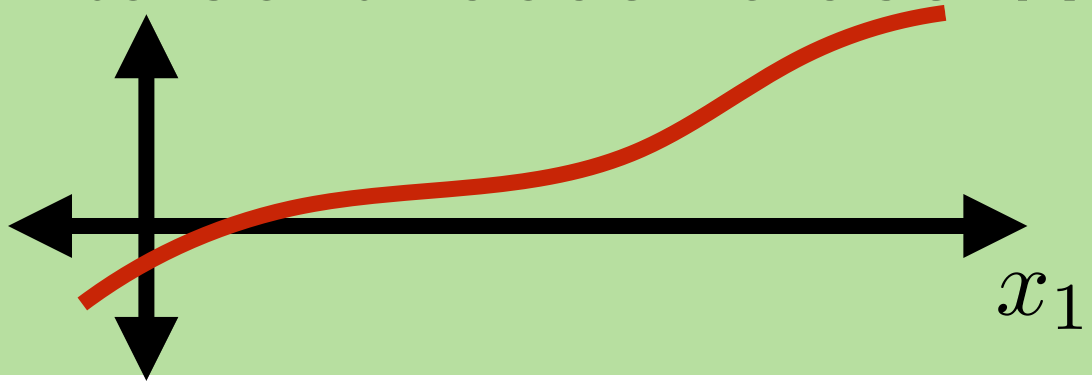
- **Multi-class classification:** > 2 label values



Machine Learning Tasks

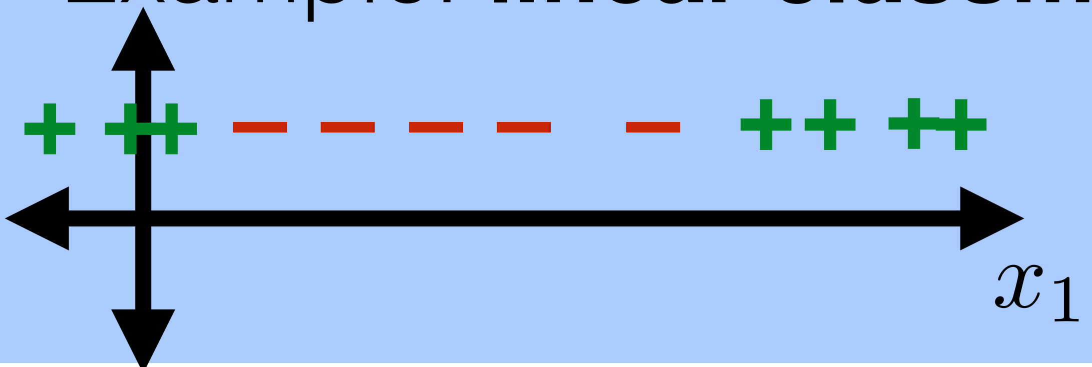
- **Supervised learning:** Learn a mapping from features to labels

- **Regression:** Learn a mapping to continuous values: R^d ! R^k

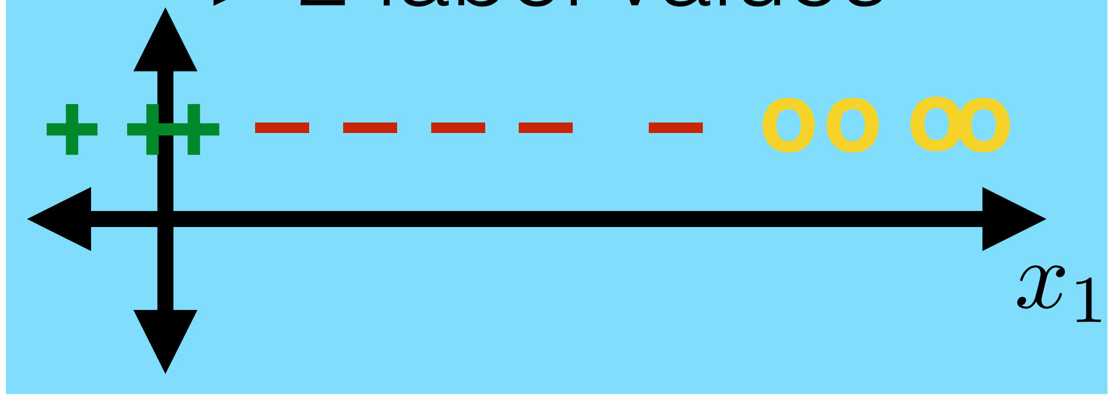


- **Classification:** Learn a mapping to a discrete set

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



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

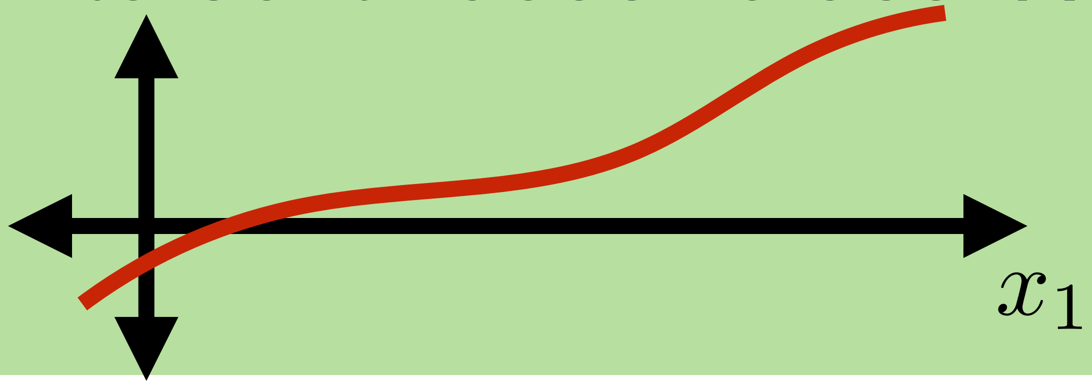


Machine Learning Tasks

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

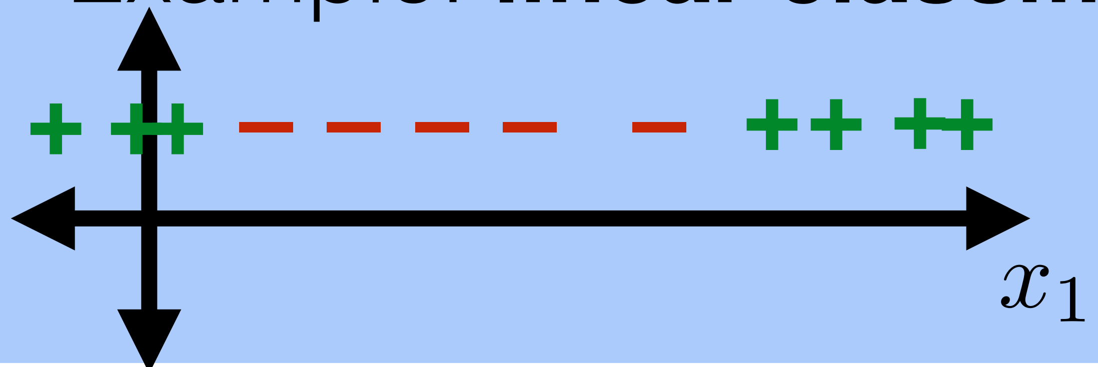
- **Unsupervised learning**

- **Regression:** Learn a mapping to continuous values: R^d ! R^k

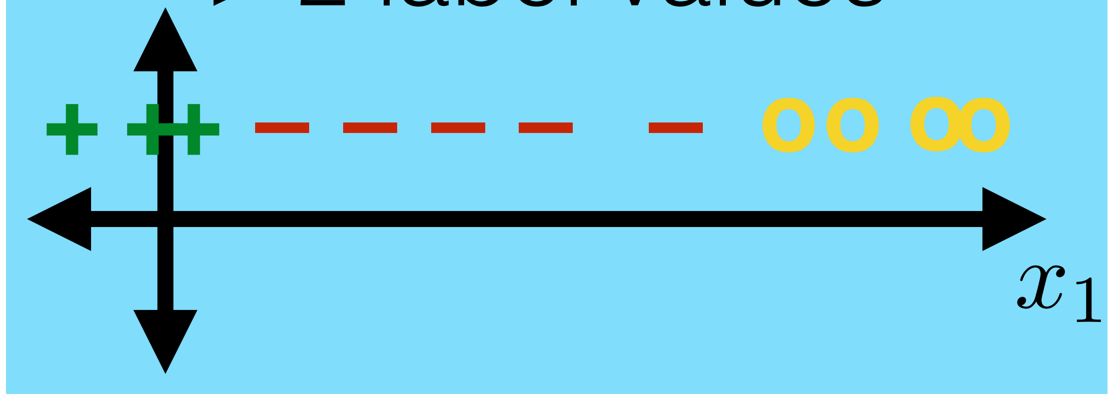


- **Classification:** Learn a mapping to a discrete set

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



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

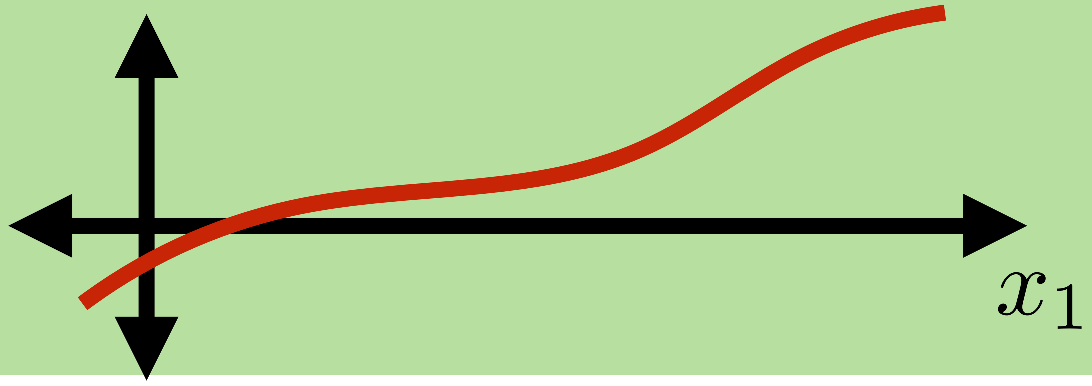


Machine Learning Tasks

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

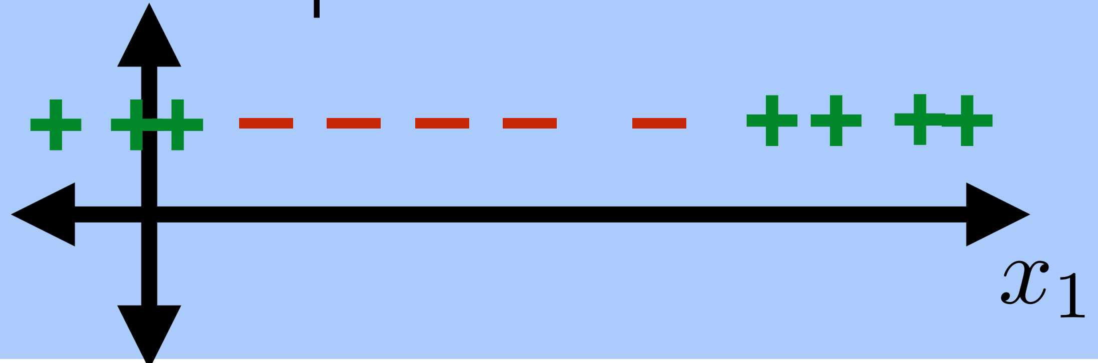
- **Unsupervised learning:** No labels; find patterns

- **Regression:** Learn a mapping to continuous values: R^d ! R^k

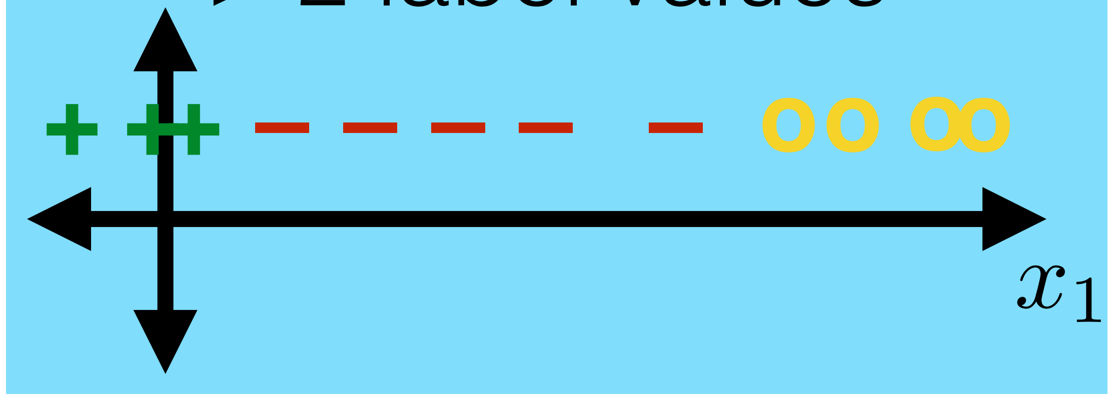


- **Classification:** Learn a mapping to a discrete set

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



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



MIT · 6.036 | Introduction to Machine Learning (2020)

MIT 6.036(2020)· 课程资料包 @ShowMeAI

视频	课件	笔记	代码
中英双语字幕	一键打包下载	官方笔记翻译	作业项目解析

视频 · B 站 [扫码或点击链接]
https://www.bilibili.com/video/BV1y44y187wN

课件 & 代码 · 博客 [扫码或点击链接]
http://blog.showmeai.tech/mit-6.036

机器学习 循环神经 神经网络 感知器 特征构建 聚类 马尔可夫决策过程 网络 随机森林 决策树 逻辑回归 卷积神经网络 状态机

Awesome AI Courses Notes Cheatsheets 是 ShowMeAI 资料库的分支系列，覆盖最具知名度的 TOP50+ 门 AI 课程，旨在为读者和学习者提供一整套高品质中文学习笔记和速查表。

点击课程名称，跳转至课程**资料包**页面，**一键下载**课程全部资料！

机器学习	深度学习	自然语言处理	计算机视觉
Stanford · CS229	Stanford · CS230	Stanford · CS224n	Stanford · CS231n
# Awesome AI Courses Notes Cheatsheets· 持续更新中			
知识图谱	图机器学习	深度强化学习	自动驾驶
Stanford · CS520	Stanford · CS224W	UCBerkeley · CS285	MIT · 6.S094

微信公众号

资料下载方式 2：扫码点击底部菜单栏

称为 **AI 内容创作者?** 回复 [添砖加瓦]