Recall: Classifiers

- A linear classifier:

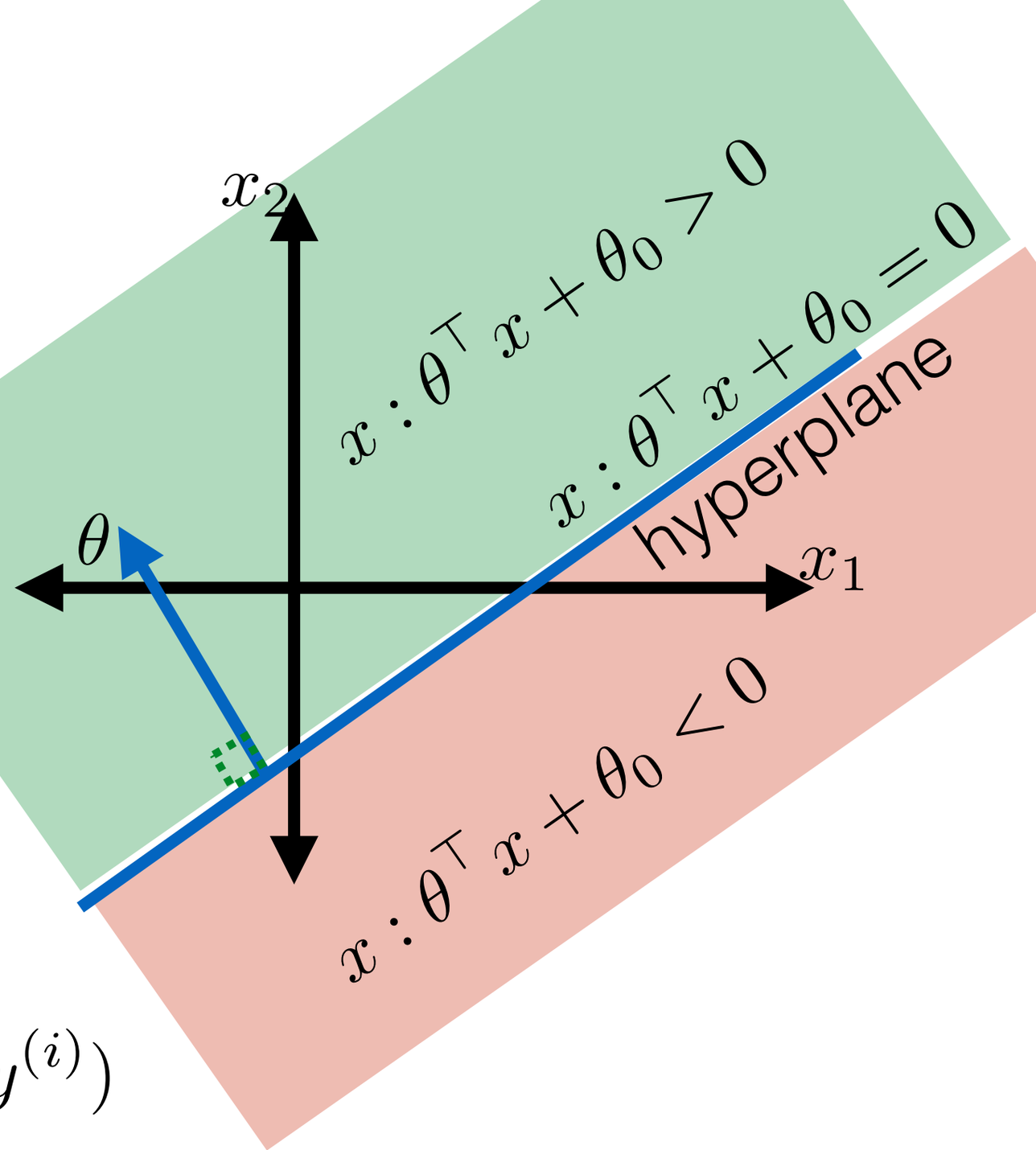
$$h(x;\theta,\theta_0) = \text{sign}(\theta^\top x + \theta_0)$$
$$= \begin{cases} +1 \text{ if } \theta^\top x + \theta_0 > 0 \\ -1 \text{ if } \theta^\top x + \theta_0 \leq 0 \end{cases}$$

- 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( D_n ; k )
    Set j* = argmin_{j∈{1,...,k}} E_n(h^(j))
    Return h^(j*)
```

[demo]

Perceptron Algorithm

Perceptron($\mathcal{D}_n$; τ)

  Initialize $\theta = [0 \ 0 \ \dots \ 0]^\top$

  Initialize $\theta_0 = 0$

  for t = 1 to τ

    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 θ, θ_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:- (slide - 3-72)

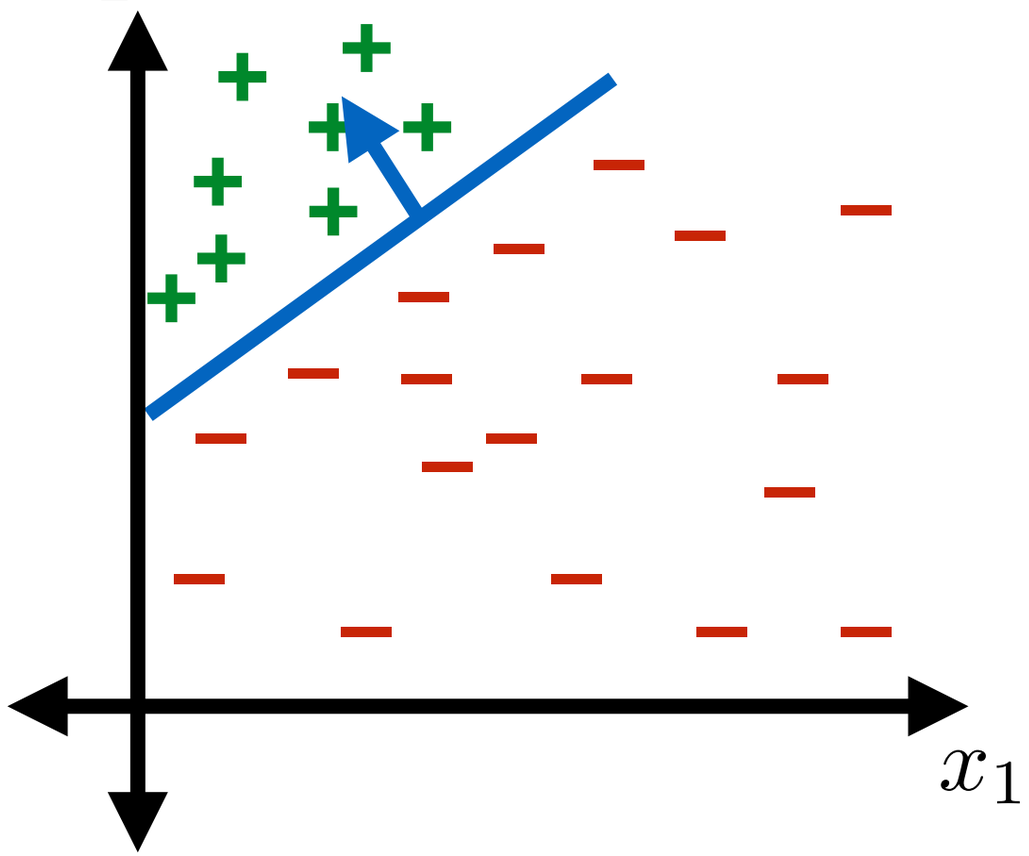
- The percaptron algorithm is a course correction type of algorithm i.e. it updates it's classifier every time the predicted & the actual label for a data point don't match.

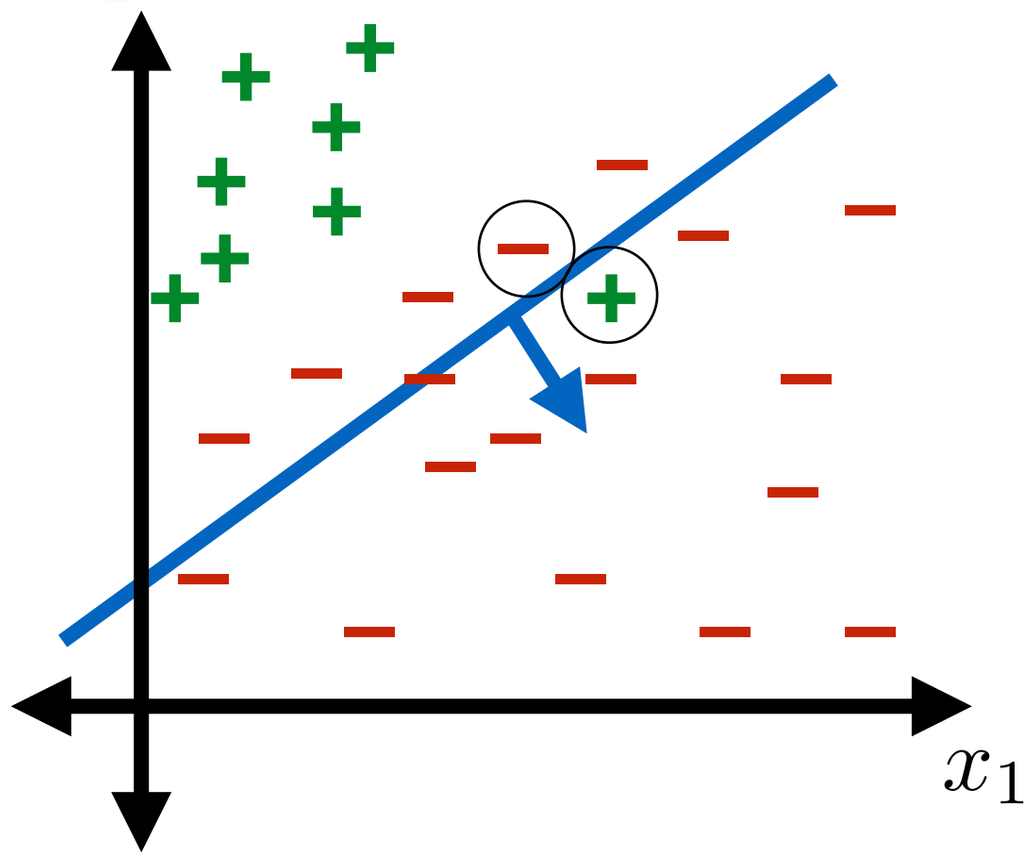
- For a dataset of D_n with n data points, we first define how many iterations of this algorithm we want to perform. It's a matter of choice & this number is 'T'
- We first initialize $\theta = [0\ 0\ 0 \dots 0]^T$ & $\theta_0 = 0$
- Then, we run a loop 'T' times. This is the outer loop & 'T' is arbitary.
- Inside the loop, we have maintain a boolean variable which is set to False by default. It checks for updation of classifier.
- Then inside the inner loop (which runs over n data points) we check whether the signs of predicted & actual labels are same.
- If they mismatch, their product is negative (i.e. ≤ 0). Now, we make an update to θ & θ_0 such that this product may come out positive. And we update the bool variable to True.
- In the if statement, if the bool value was False (i.e. the loop ran over all data points without updation) that means we have found the perfect classifier & no need to search further. Hence, we break the loop & return θ & θ_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

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



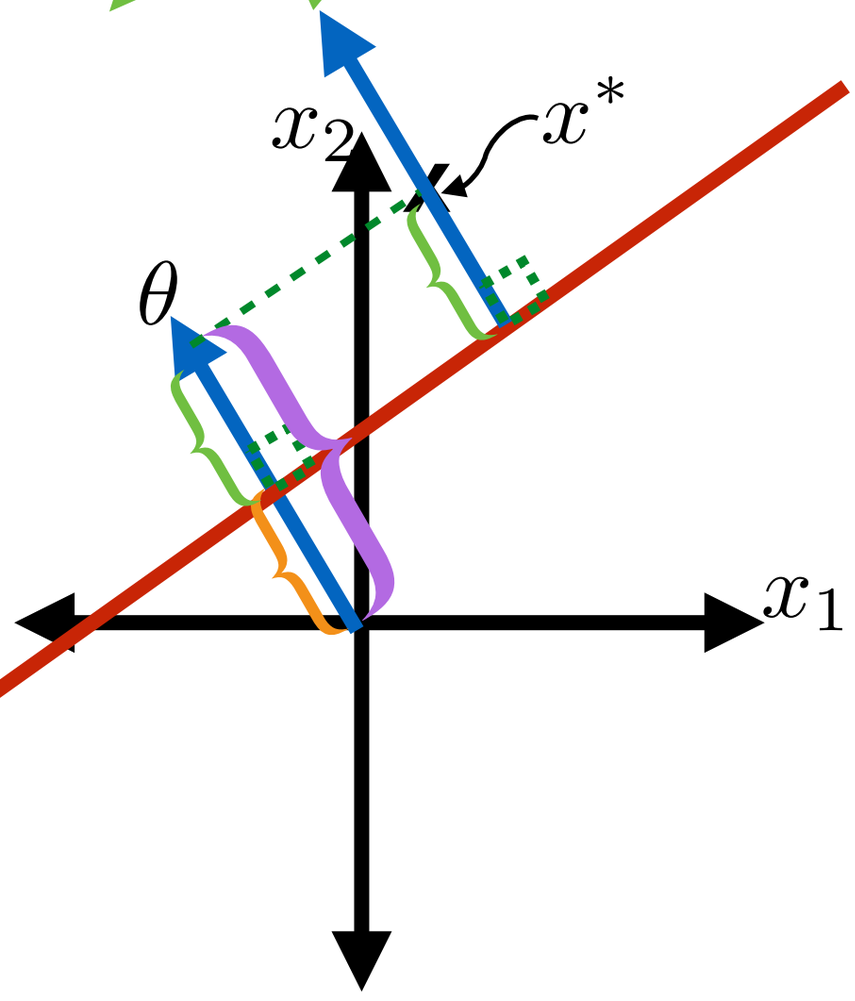


✱ Classifier Quality :- (slide - 4 -)

- By definition a good classifier classifies every single point in the dataset D_n successfully.

Let's Talk About Classifier Quality

Math facts!



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

$$= \text{projection of } x^* \text{ on } \theta$$
$$- \text{signed distance of line to origin}$$
$$= \frac{\theta^\top x^*}{\|\theta\|} - \frac{-\theta_0}{\|\theta\|} = \frac{\theta^\top x^* + \theta_0}{\|\theta\|}$$

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

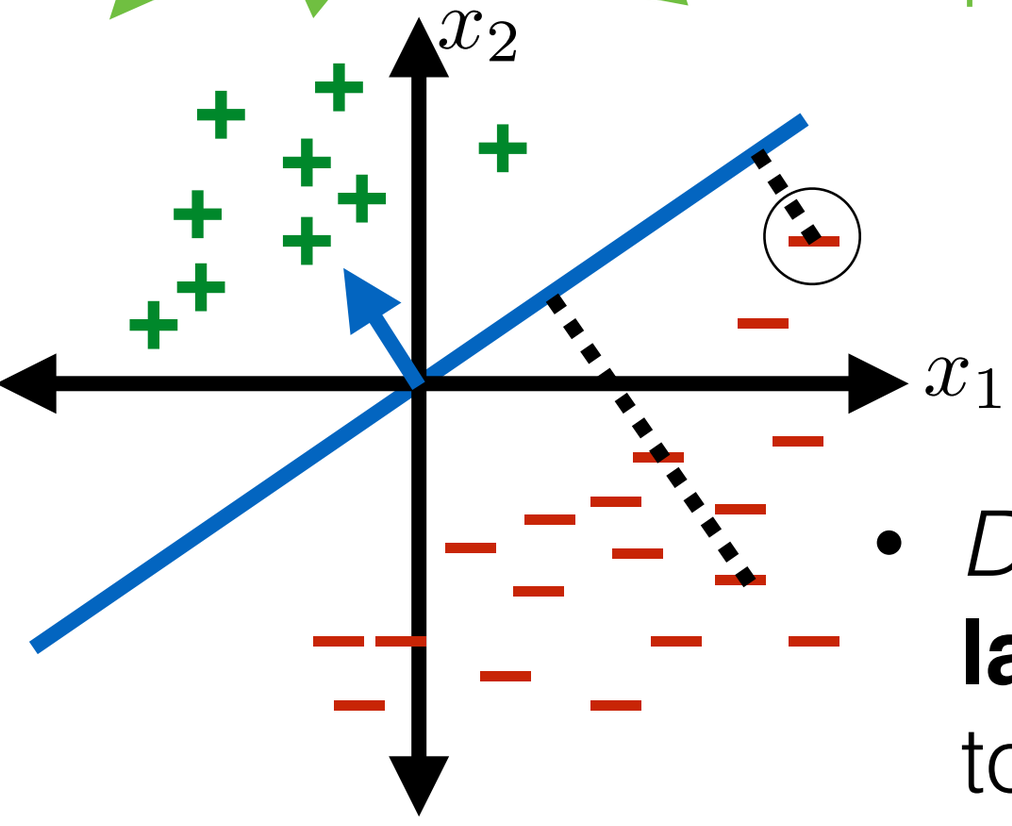
$$y^* \left(\frac{\theta^\top x^* + \theta_0}{\|\theta\|} \right)$$

★ Maths facts - Margin of point (slide - 5 -)

- The margin is basically the signed distance of a point from the classifier line / plane defined by θ, θ_0.
- It also tells whether the point is a misclassification or not.
- It sort of tells us, how far away ~~on average~~ is the data from the classifier plane / line.

Let's Talk About Classifier Quality

Math facts!



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

$$= \text{projection of } x^* \text{ on } \theta$$
$$- \text{signed distance of line to origin}$$
$$= \frac{\theta^\top x^*}{\|\theta\|} - \frac{-\theta_0}{\|\theta\|} = \frac{\theta^\top x^* + \theta_0}{\|\theta\|}$$

- *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:

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

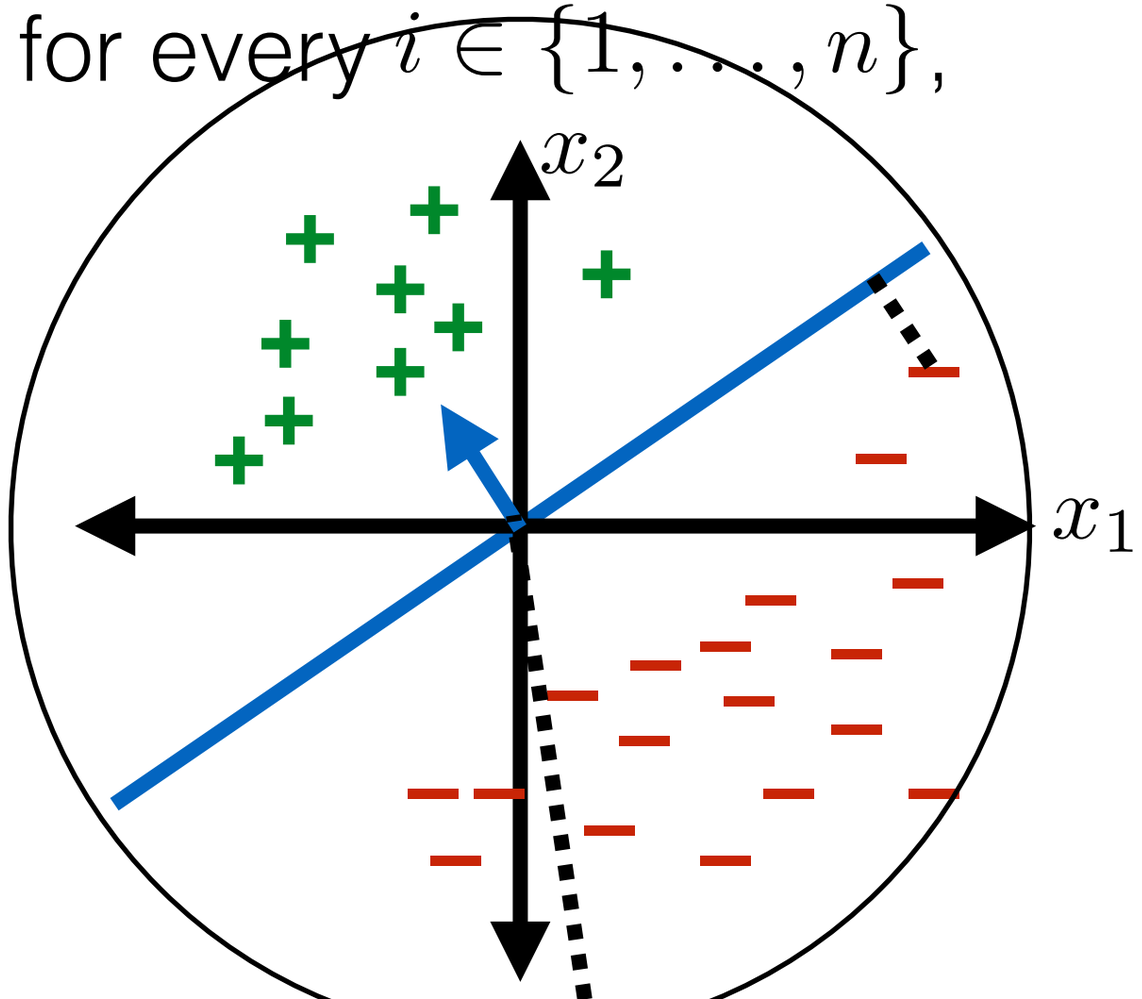
★ Classifier quality 2 & Margin of dataset (Slide - 5 -)

- Margin of the dataset is the minimum of all margins of individual points in the dataset. i.e. even if there's 1 misclassification, margin of dataset will be negative.
- If all margins of points are positive (perfect classifier) then the margin of dataset will tell the minimum distance between classifier plane & data points.

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.



★ Theorem: Perceptron performance:- (Slide - 6 -)

- Assumption B basically states that the margin of the dataset must be greater than γ which is an arbitrary value. i.e. There must be sufficient distance between data points & classifier line.
- Assumption C describes distance R such that no point in the dataset has magnitude greater than R. All datapoints line IN the circle defined by radius R at origin.

Why classifiers through the origin?

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

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

$$x : \theta^\top x + \theta_0 \begin{matrix} < \\ = \\ > \end{matrix} 0$$

  - Classifier without offset

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

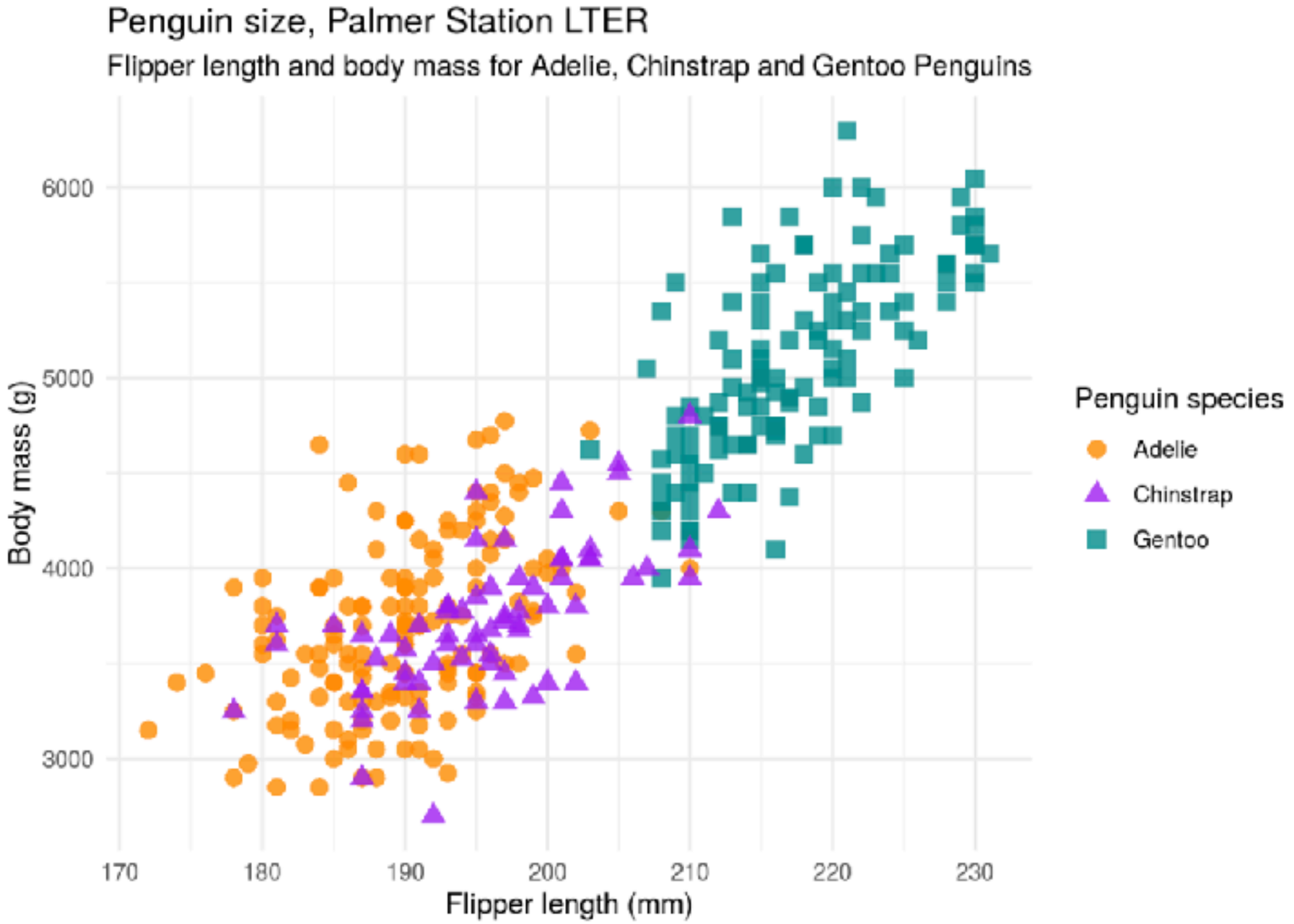
$$x_{\text{new}} = [x_1, x_2, \ldots, x_d, 1]^\top, \theta_{\text{new}} = [\theta_1, \theta_2, \ldots, \theta_d, \theta_0]^\top$$

$$x_{\text{new},1:d} : \theta_{\text{new}}^\top x_{\text{new}} \begin{matrix} < \\ = \\ > \end{matrix} 0$$

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

Problem: data not linearly separable

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

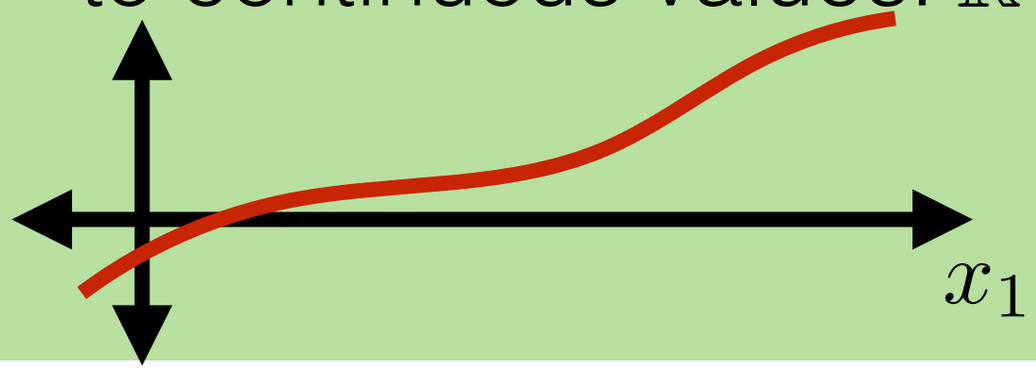


- What can we do? See upcoming lectures!

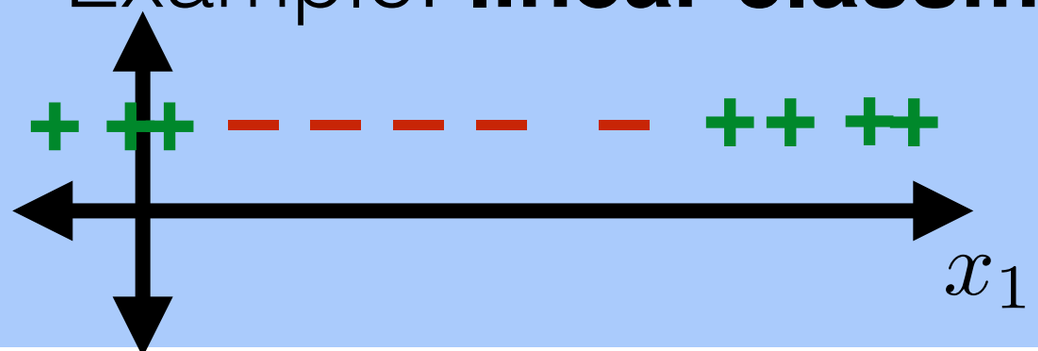
[https://towardsdatascience.com/penguins-dataset-overview-iris-alternative-9453bb8c8d95, https://github.com/allisonhorst/palmerpenguins]

Machine Learning Tasks

- **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