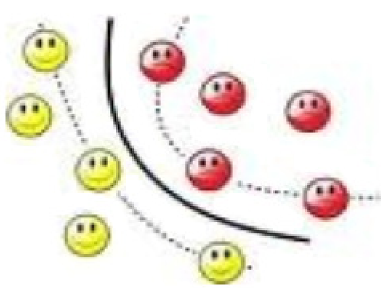
CIS 606 Machine Learning
Spring 2013

Wei Lee Woon and Zeyar Aung

Masdar INSTITUTE



Lecture 8:

Machine Learning for Classification

Machine Learning for Classification

(Today's topics)

- Linear classification

- Perceptron algorithm

- Convergence

Machine learning

- Learning to predict from labeled examples


+1


-1


+1


-1

training set

Machine learning

- Learning to predict from labeled examples

+1

-1

+1

-1

training set

The training set of labeled examples specifies the learning task only implicitly

Machine learning

- Learning to predict from labeled examples

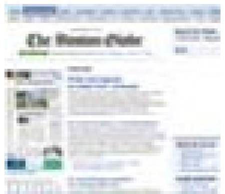
+1

-1

+1

-1

training set



?

Our goal is to accurately label new websites that were not part of the training set

test set

Machine learning

- Learning to predict from labeled examples

+1

-1

+1

-1

training set

predicted label

new website

$$y = \hat{f}\Big(\ \Big)$$

classifier = mapping from websites to labels

applied to new examples in the test set

“Examples”

- We will have to first represent the examples (websites) in a manner that can be easily mapped to labels

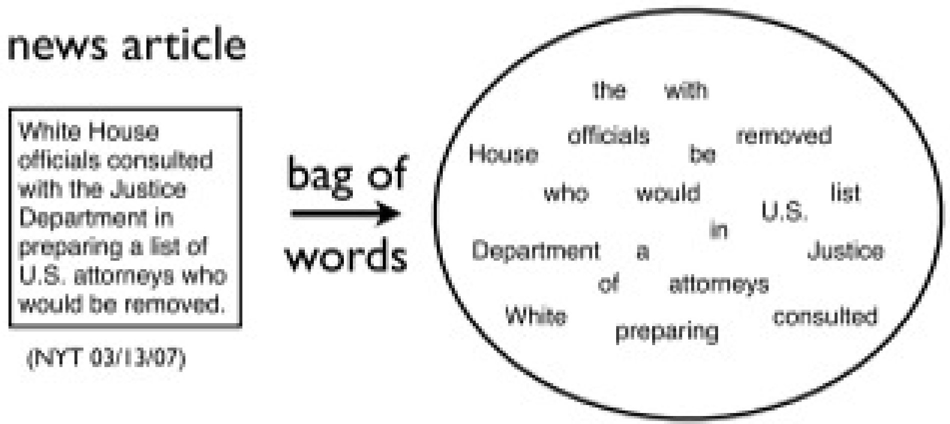
news article

White House officials consulted with the Justice Department in preparing a list of U.S. attorneys who would be removed.

(NYT 03/13/07)

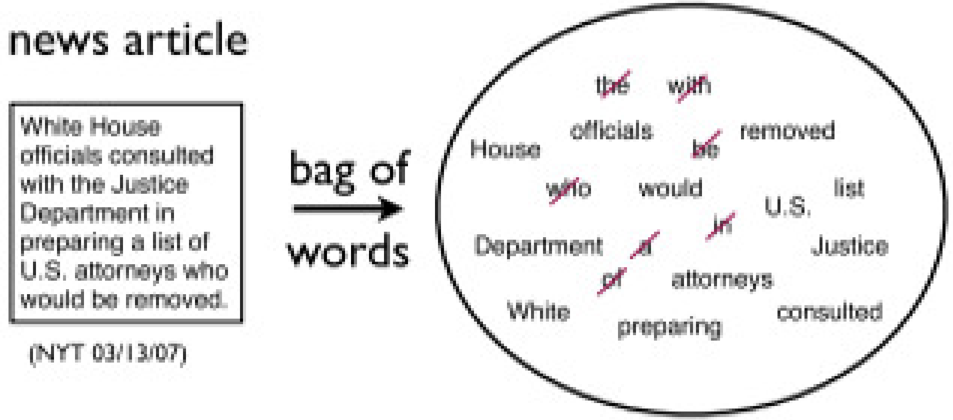
"Examples"

- We will have to first represent the examples (websites) in a manner that can be easily mapped to labels



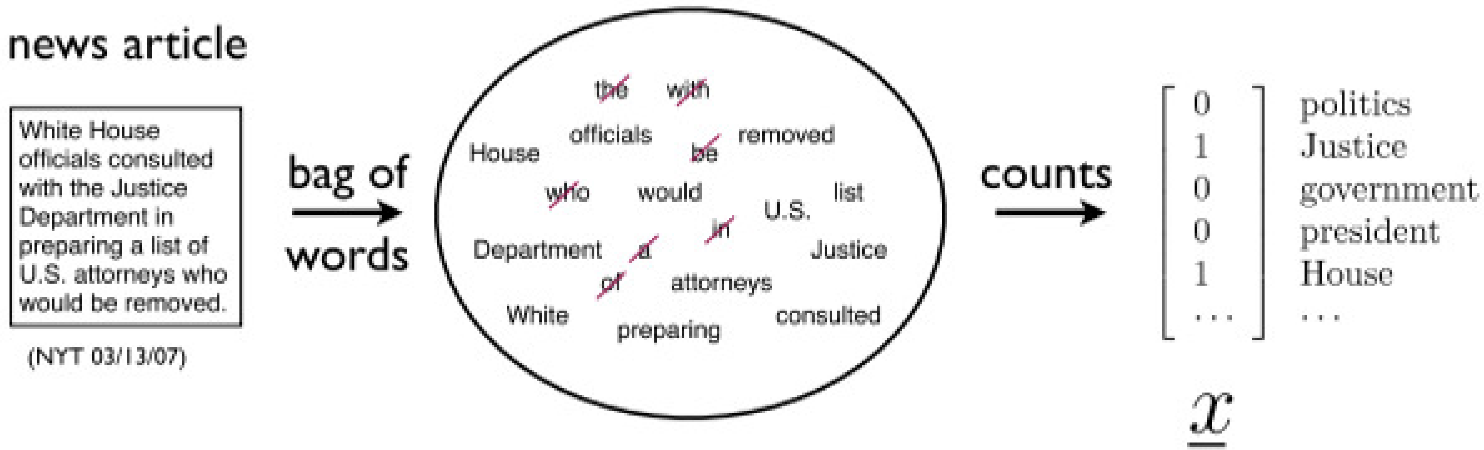
“Examples”

- We will have to first represent the examples (websites) in a manner that can be easily mapped to labels



“Examples”

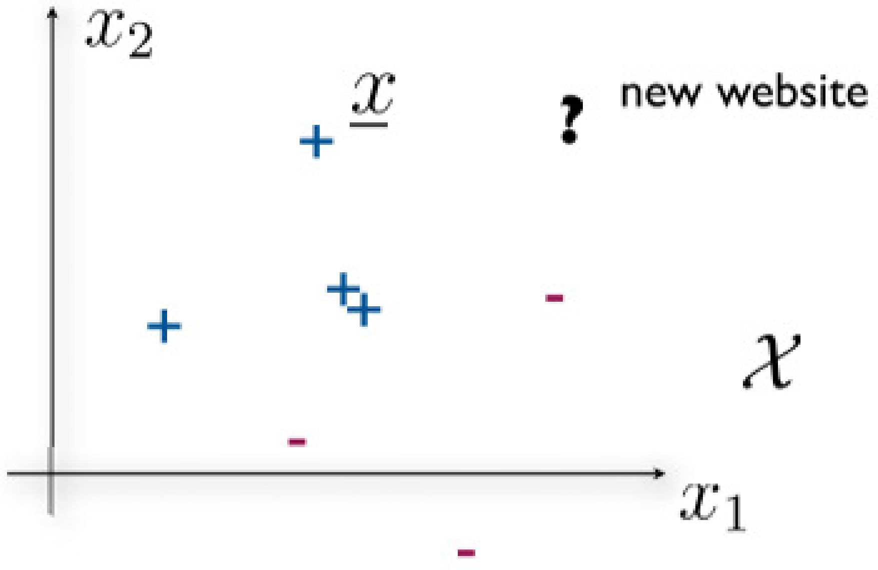
- We will have to first represent the examples (websites) in a manner that can be easily mapped to labels



a vector whose coordinates (features) specify how many times (or whether) particular words appeared in the article

The learning task

- The training set is now a set of labeled points

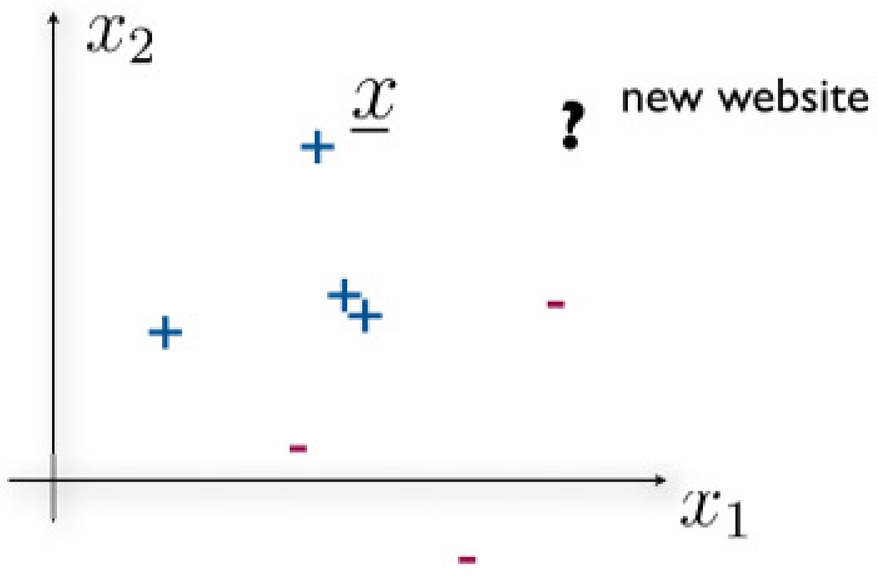


- Our goal is to find a "good" classifier $f : \mathcal{X} \rightarrow \{-1, 1\}$ based on the training set $D = \{(\underline{x}_i, y_i)_{i=1,\ldots,n}\}$ so that $f(\underline{x})$ correctly labels any new websites $\underline{x}$

The learning task

- The training set is now a set of labeled points

Part 1:
Model selection
what type (or set) of classifiers should we consider?

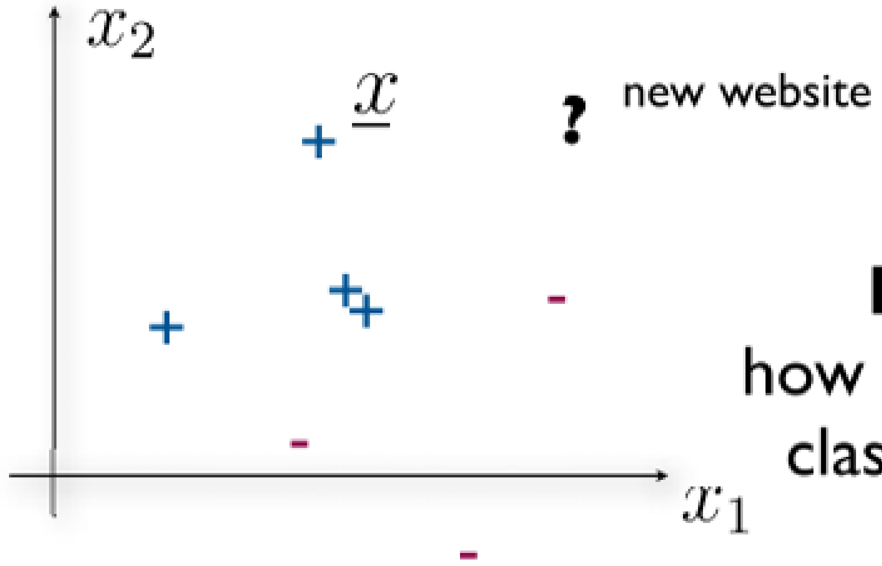


- Our goal is to find a "good" classifier $f : \mathcal{X} \rightarrow \{-1, 1\}$ based on the training set $D = \{(\underline{x}_i, y_i)_{i=1,\ldots,n}\}$ so that $f(\underline{x})$ correctly labels any new websites $\underline{x}$

The learning task

- The training set is now a set of labeled points

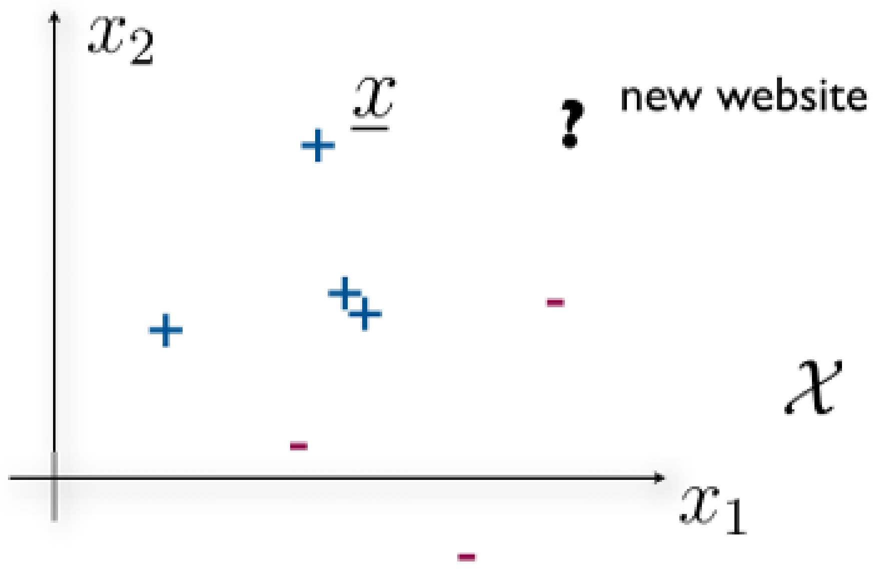
Part 1: Model selection
what type (or set) of classifiers should we consider?



Part 2: Estimation
how to select the best classifier in the set?

- Our goal is to find a "good" classifier $f : \mathcal{X} \rightarrow \{-1, 1\}$ based on the training set $D = \{(\underline{x}_i, y_i)_{i=1,\ldots,n}\}$ so that $f(\underline{x})$ correctly labels any new websites $\underline{x}$

Part 1: allowing all classifiers?

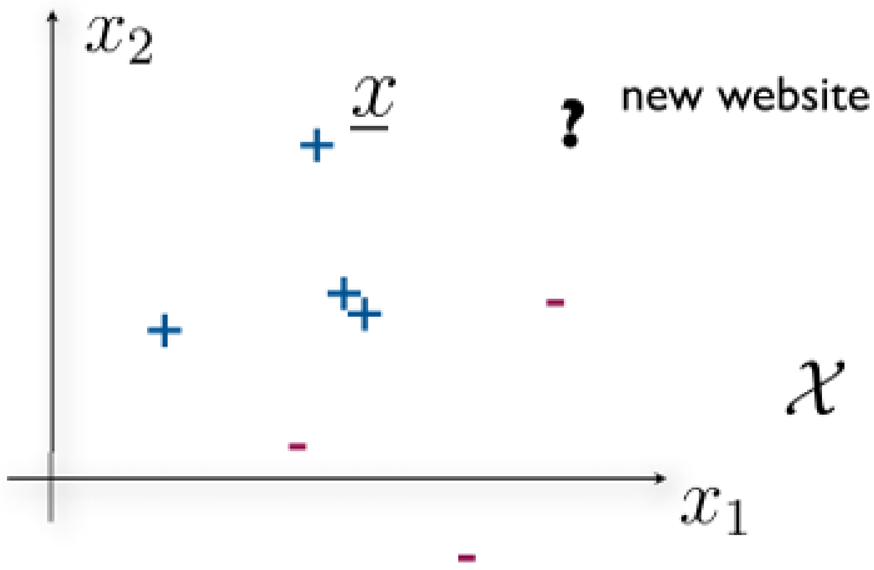


- We can easily construct a "silly classifier" that perfectly classifiers any distinct set of training points

$$f(\underline{x}) = \begin{cases} y_i, & \text{if } \underline{x} = \underline{x}_i \text{ for some } i \\ -1, & \text{otherwise} \end{cases}$$

- But it doesn't "generalize" (it doesn't classify new points very well)

Part 1: allowing few classifiers?



- We could instead consider very few alternatives such as

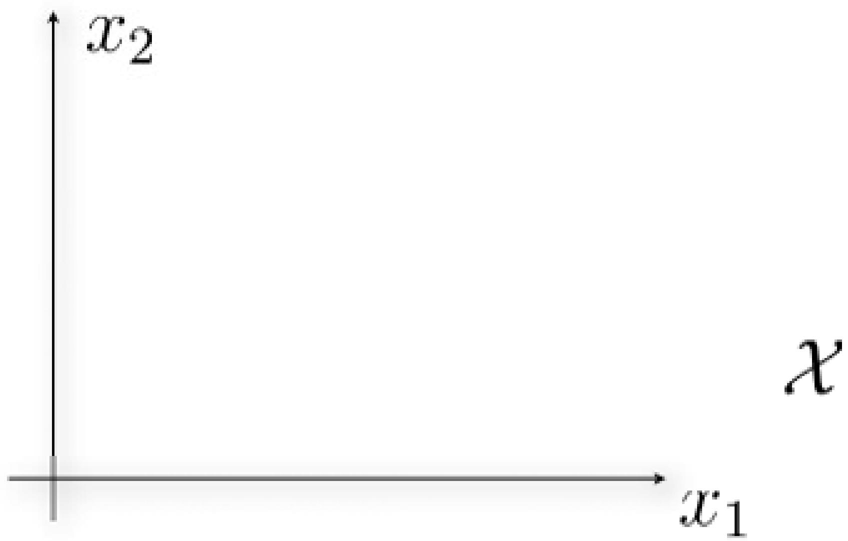
$$\begin{aligned} f(\underline{x}) &= 1, \quad \text{for all} \quad \underline{x} \in \mathcal{X}, \quad \text{or} \\ f(\underline{x}) &= -1, \quad \text{for all} \quad \underline{x} \in \mathcal{X}, \end{aligned}$$

- But neither one classifies even training points very well

Part 1: linear classifiers

- A linear classifier (through origin) with parameters $\underline{\theta}$ divides the space into positive and negative halves

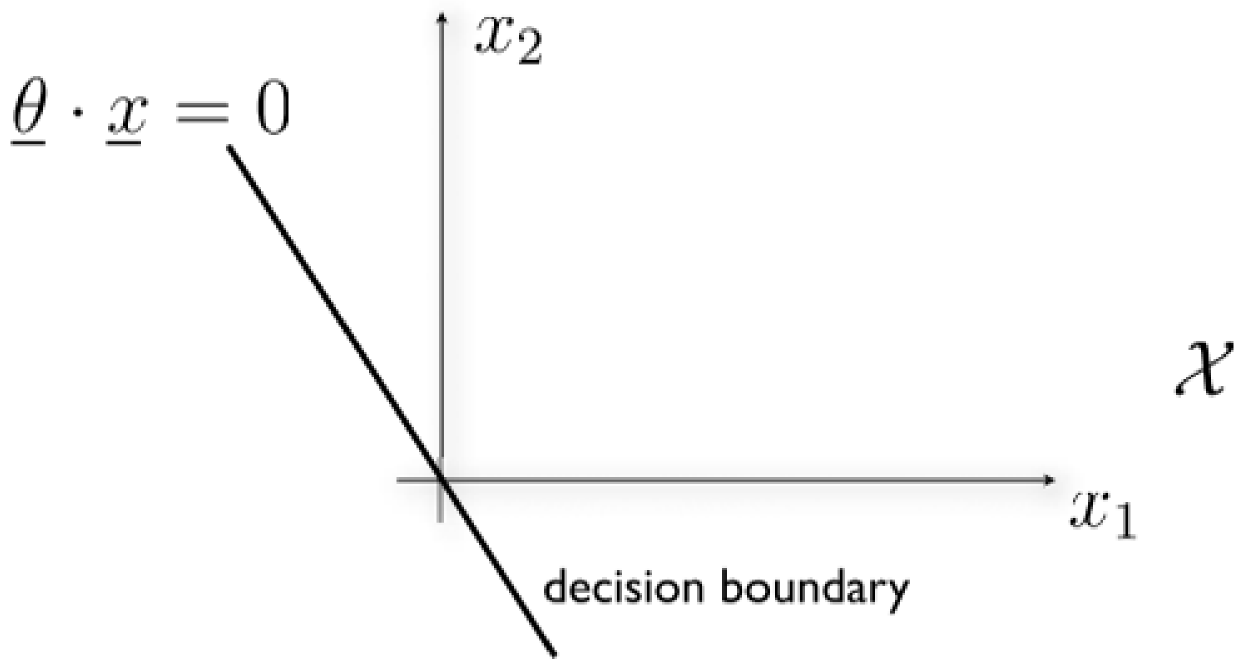
$$
\begin{aligned}
f(\underline{x};\underline{\theta}) &= \operatorname{sign}\big(\,\underline{\theta}\cdot\underline{x}\,\big) = \operatorname{sign}\big(\,\theta_1 x_1 + \ldots + \theta_d x_d\,\big) \\
&= \begin{cases} +1, & \text{if } \underline{\theta}\cdot\underline{x} > 0 \\ -1, & \text{if } \underline{\theta}\cdot\underline{x} \leq 0 \end{cases}
\end{aligned}
$$



Part 1: linear classifiers

- A linear classifier (through origin) with parameters $\underline{\theta}$ divides the space into positive and negative halves

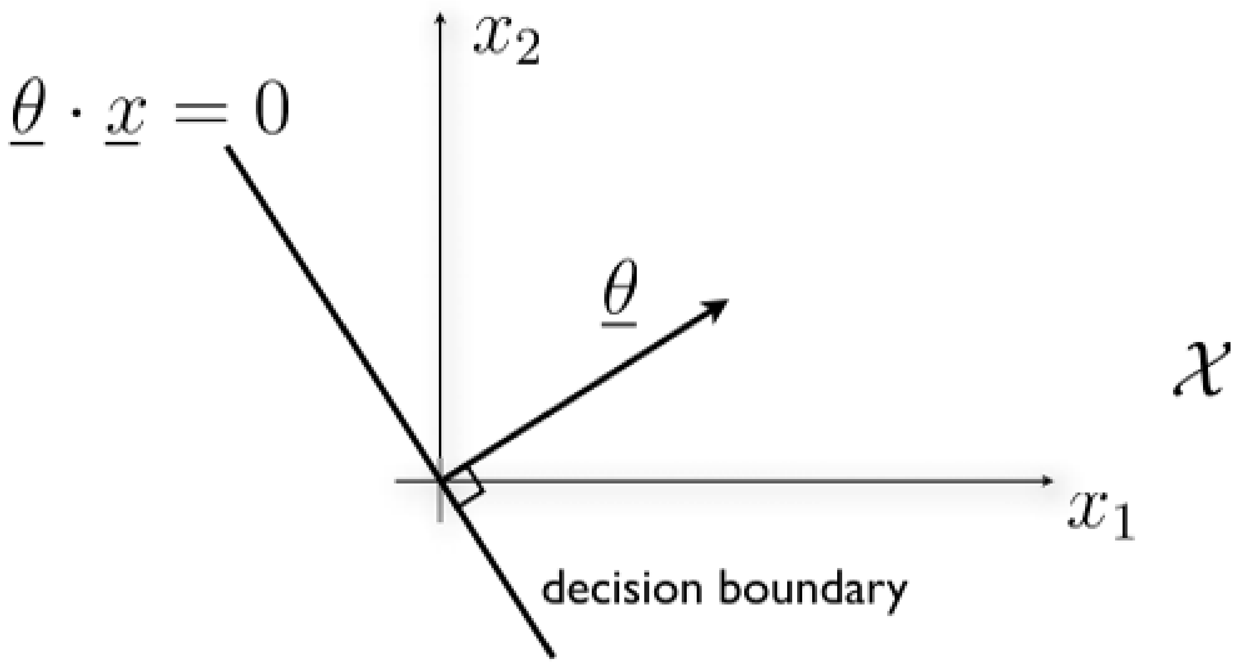
$$
\begin{aligned}
f(\underline{x};\underline{\theta}) &= \text{sign}\big(\,\underline{\theta}\cdot\underline{x}\,\big) = \text{sign}\big(\,\theta_1 x_1 + \ldots + \theta_d x_d\,\big) \\
&= \begin{cases} +1, & \text{if } \underline{\theta}\cdot\underline{x} > 0 \\ -1, & \text{if } \underline{\theta}\cdot\underline{x} \le 0 \end{cases}
\end{aligned}
$$



Part 1: linear classifiers

- A linear classifier (through origin) with parameters $\underline{\theta}$ divides the space into positive and negative halves

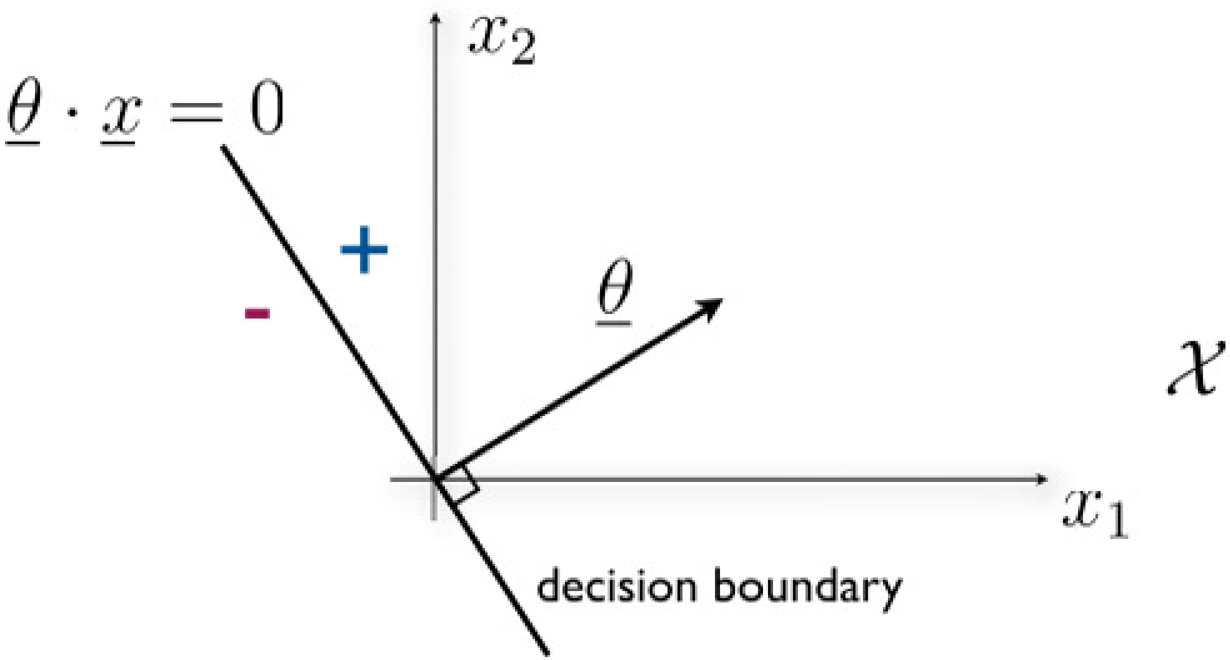
$$
\begin{aligned}
f(\underline{x};\underline{\theta}) &= \operatorname{sign}(\,\underline{\theta}\cdot\underline{x}\,) = \operatorname{sign}(\,\theta_1 x_1 + \ldots + \theta_d x_d\,) \\
&= \begin{cases} +1, & \text{if } \underline{\theta}\cdot\underline{x} > 0 \\ -1, & \text{if } \underline{\theta}\cdot\underline{x} \le 0 \end{cases}
\end{aligned}
$$



Part 1: linear classifiers

- A linear classifier (through origin) with parameters $\underline{\theta}$ divides the space into positive and negative halves

$$\begin{aligned} f(\underline{x};\underline{\theta}) &= \operatorname{sign}(\,\underline{\theta}\cdot\underline{x}\,) = \operatorname{sign}(\,\theta_1 x_1 + \ldots + \theta_d x_d\,) \\ &= \begin{cases} +1, & \text{if } \underline{\theta}\cdot\underline{x} > 0 \\ -1, & \text{if } \underline{\theta}\cdot\underline{x} \leq 0 \end{cases} \end{aligned}$$



Part 2: estimation

- We can use the training error as a surrogate criterion for finding the "best" linear classifier through origin

$$\hat{R}_n(\underline{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \mathrm{Loss}\big(\, y_i, f(\underline{x}_i;\underline{\theta})\,\big)$$

$$\text{where}\ \ \mathrm{Loss}(\, y, y'\,) = \begin{cases} 1, \ \text{if } y \neq y' \\ 0, \ \text{o.w.} \end{cases}$$

- Other choices are possible (and often preferable)

Perceptron algorithm

- The perceptron algorithm considers each training point in turn, adjusting the parameters to correct any mistakes

$$\text{Initialize: } \underline{\theta} = 0$$

$$\text{Repeat until convergence:}$$

$$\text{for } t = 1, \ldots, n$$

$$\text{if } y_t(\underline{\theta} \cdot \underline{x}_t) \leq 0 \text{ (mistake)}$$

$$\underline{\theta} \leftarrow \underline{\theta} + y_t \underline{x}_t$$

- The algorithm will converge (no further mistakes) if the training points are *linearly separable through origin*; otherwise it won't converge

Perceptron algorithm: motivation

- If we make a mistake on the tth training point, then

$$\mathrm{y}_t(\underline{\theta} \cdot \underline{x}_t) \leq 0$$

- After the update, we have

$$\underline{\theta}' = \underline{\theta} + y_t \underline{x}_t$$

Perceptron algorithm: motivation

- If we make a mistake on the t^th training point, then

$$\mathrm{y}_t(\underline{\theta} \cdot \underline{x}_t) \leq 0$$

- After the update, we have

$$\underline{\theta}' = \underline{\theta} + y_t \underline{x}_t$$

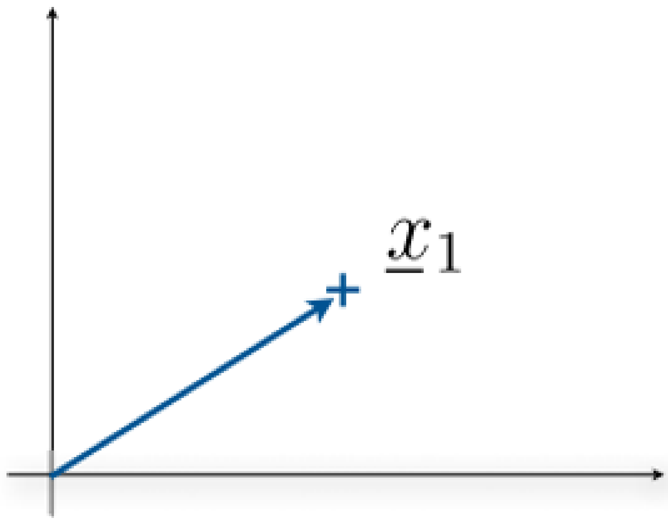
$$\begin{aligned} y_t(\underline{\theta}' \cdot \underline{x}_t) &= y_t([\underline{\theta} + y_t \underline{x}_t] \cdot \underline{x}_t) \\ &= y_t(\underline{\theta} \cdot \underline{x}_t + y_t \underline{x}_t \cdot \underline{x}_t) \\ &= y_t(\underline{\theta} \cdot \underline{x}_t) + y_t^2 \underline{x}_t \cdot \underline{x}_t \\ &= y_t(\underline{\theta} \cdot \underline{x}_t) + \|\underline{x}_t\|^2 \end{aligned}$$

- So that $y_t(\underline{\theta}' \cdot \underline{x}_t)$ increases based on the update

Perceptron algorithm

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$

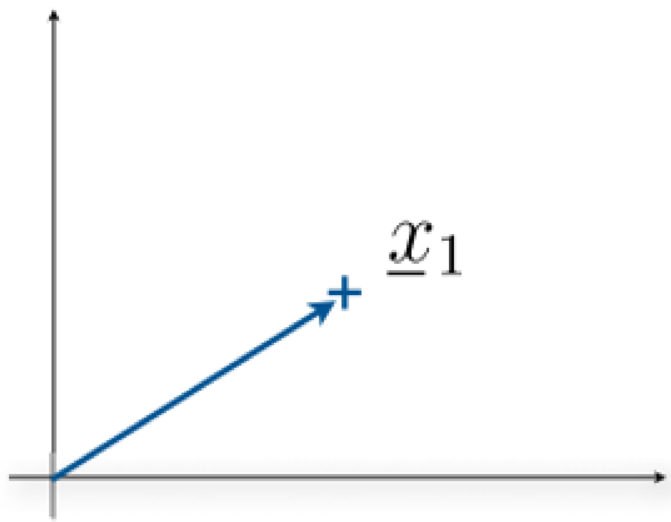


Perceptron algorithm

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$

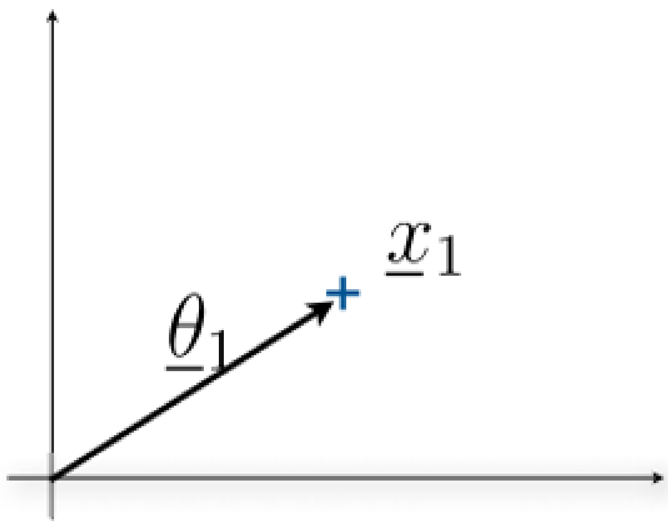
$$\underline{\theta}_1 = \underline{\theta}_0 + 1\,\underline{x}_1$$



Perceptron algorithm

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$
$$\underline{\theta}_1 = \underline{\theta}_0 + 1\,\underline{x}_1$$

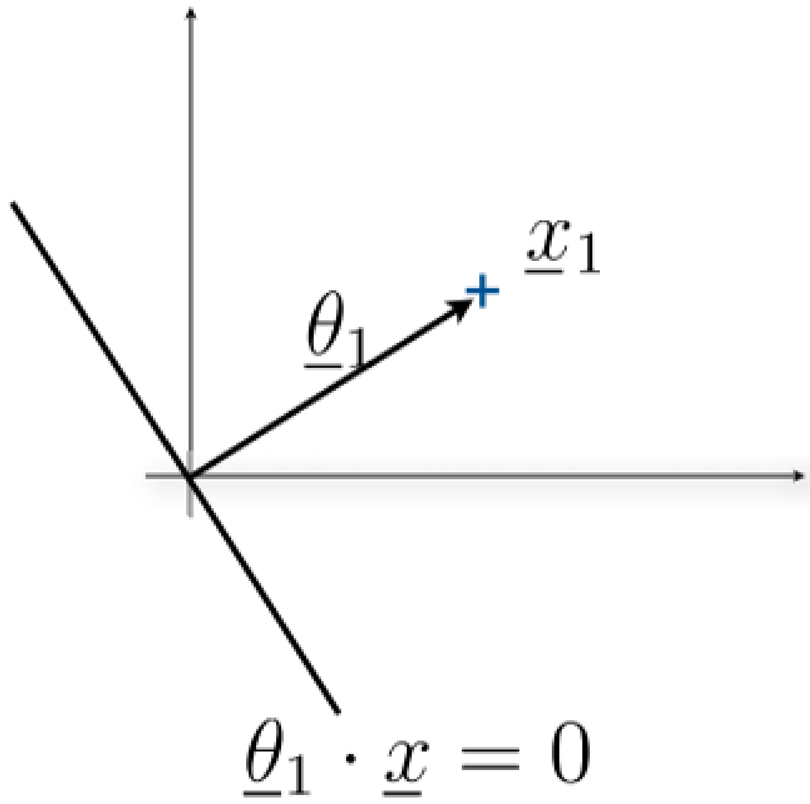


Perceptron algorithm

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$

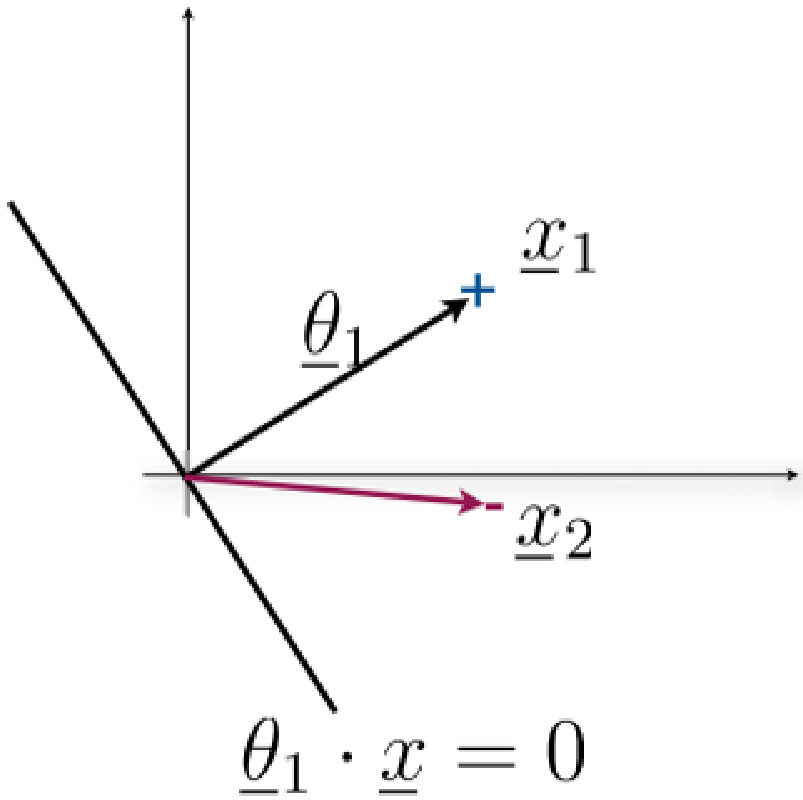
$$\underline{\theta}_1 = \underline{\theta}_0 + 1\,\underline{x}_1$$



Perceptron algorithm

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$
$$\underline{\theta}_1 = \underline{\theta}_0 + 1\,\underline{x}_1$$



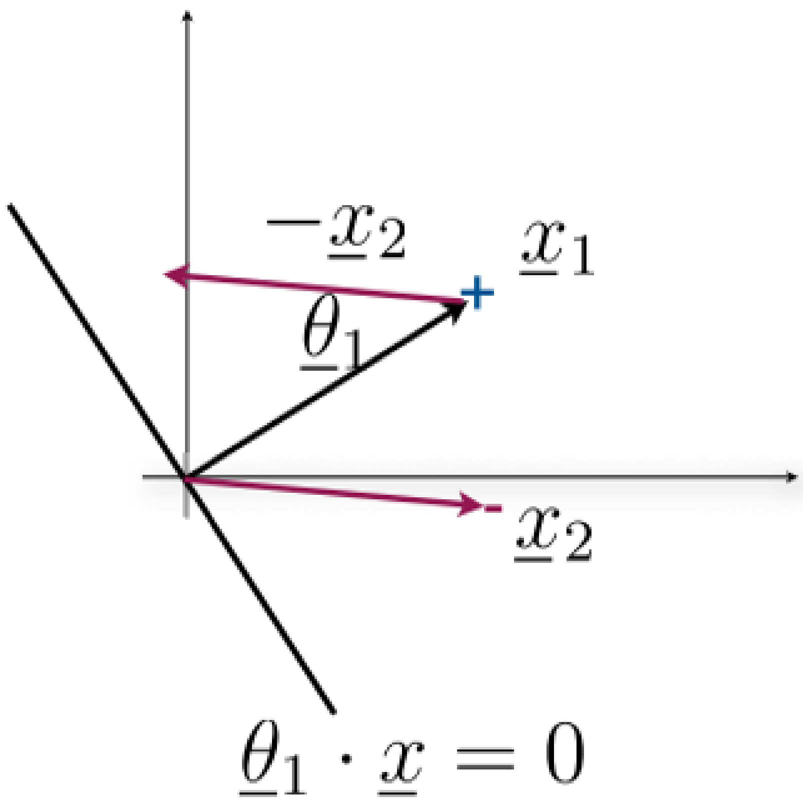
Perceptron algorithm

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$

$$\underline{\theta}_1 = \underline{\theta}_0 + 1\, \underline{x}_1$$

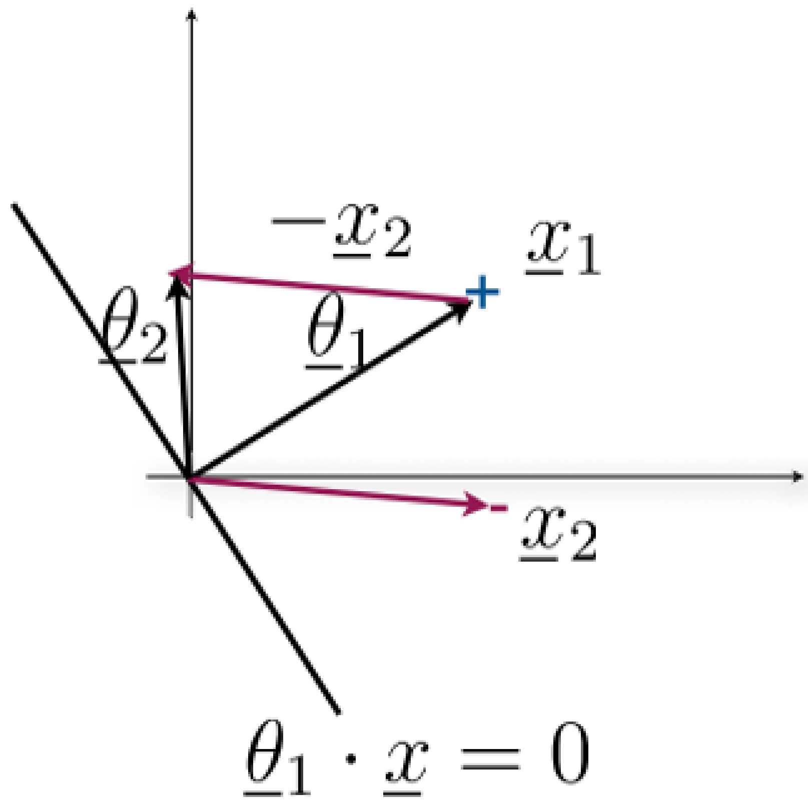
$$\underline{\theta}_2 = \underline{\theta}_1 + (-1)\, \underline{x}_2$$



Perceptron algorithm

- Iterative updates based on mistakes

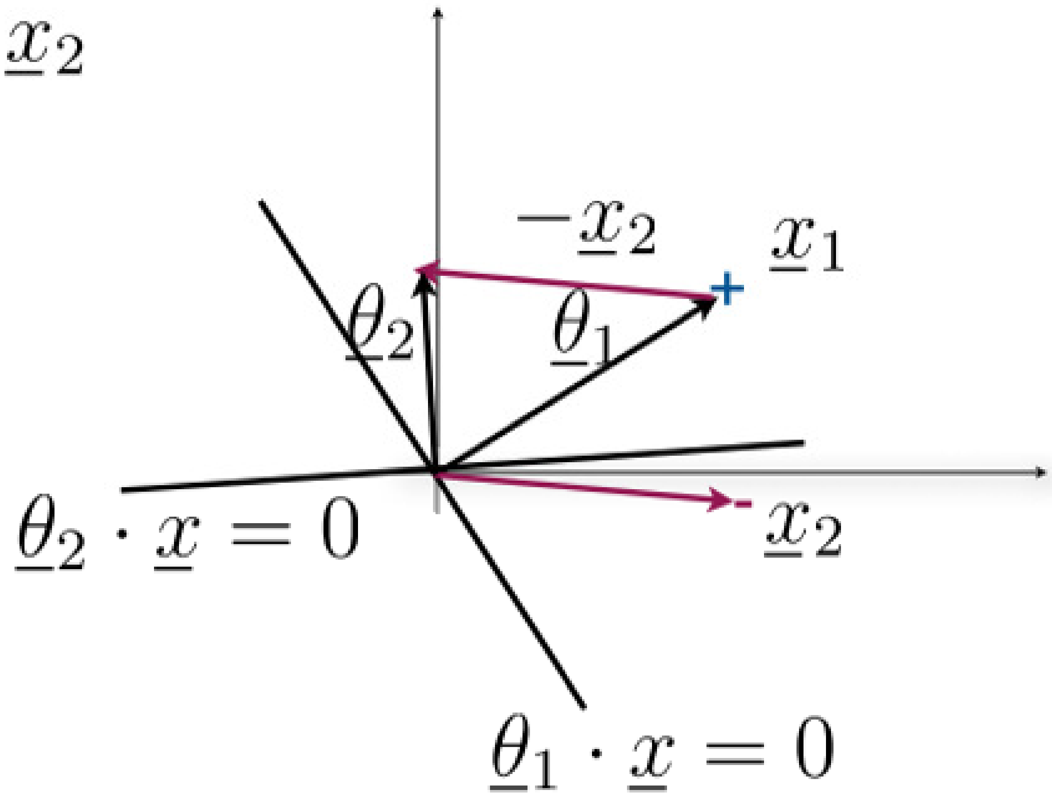
$$\underline{\theta}_0 = 0$$
$$\underline{\theta}_1 = \underline{\theta}_0 + 1\,\underline{x}_1$$
$$\underline{\theta}_2 = \underline{\theta}_1 + (-1)\,\underline{x}_2$$



Perceptron algorithm

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$
$$\underline{\theta}_1 = \underline{\theta}_0 + 1\,\underline{x}_1$$
$$\underline{\theta}_2 = \underline{\theta}_1 + (-1)\,\underline{x}_2$$



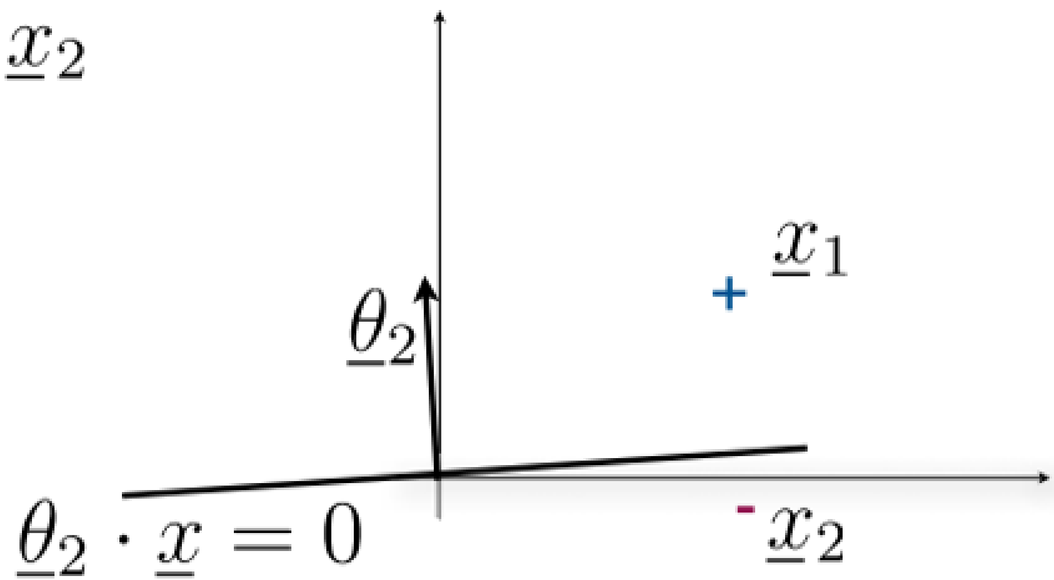
Perceptron algorithm

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$

$$\underline{\theta}_1 = \underline{\theta}_0 + 1\,\underline{x}_1$$

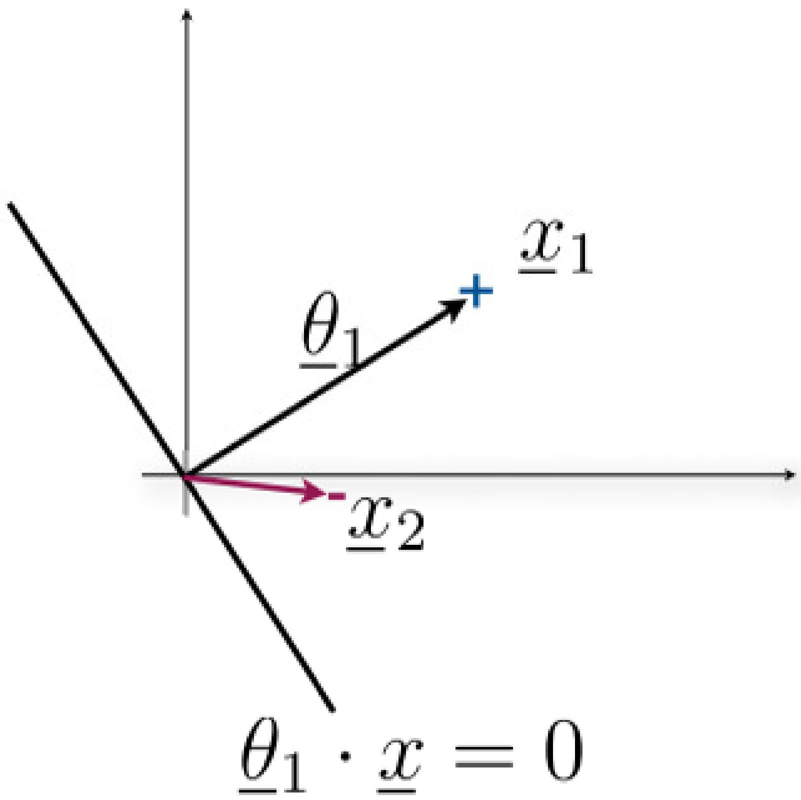
$$\underline{\theta}_2 = \underline{\theta}_1 + (-1)\,\underline{x}_2$$



Perceptron algorithm (take 2)

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$
$$\underline{\theta}_1 = \underline{\theta}_0 + 1\,\underline{x}_1$$



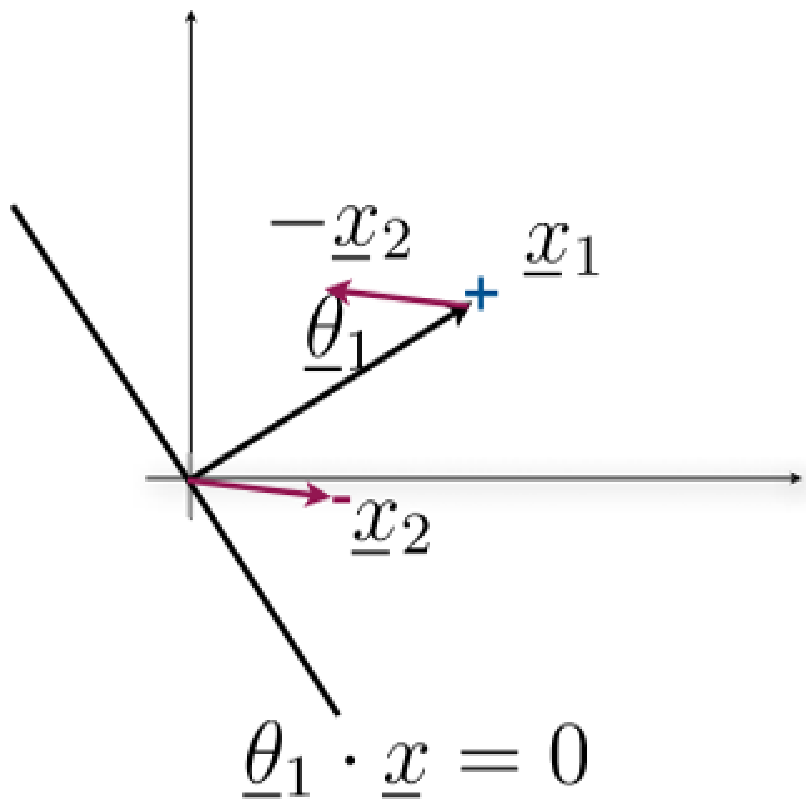
Perceptron algorithm (take 2)

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$

$$\underline{\theta}_1 = \underline{\theta}_0 + 1\,\underline{x}_1$$

$$\underline{\theta}_2 = \underline{\theta}_1 + (-1)\,\underline{x}_2$$



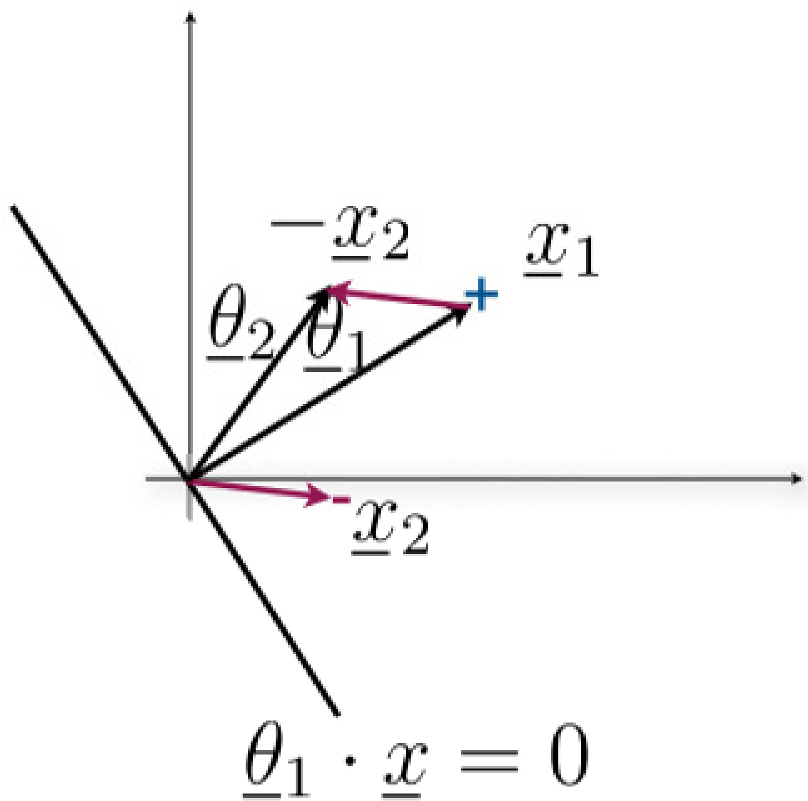
Perceptron algorithm (take 2)

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$

$$\underline{\theta}_1 = \underline{\theta}_0 + 1\,\underline{x}_1$$

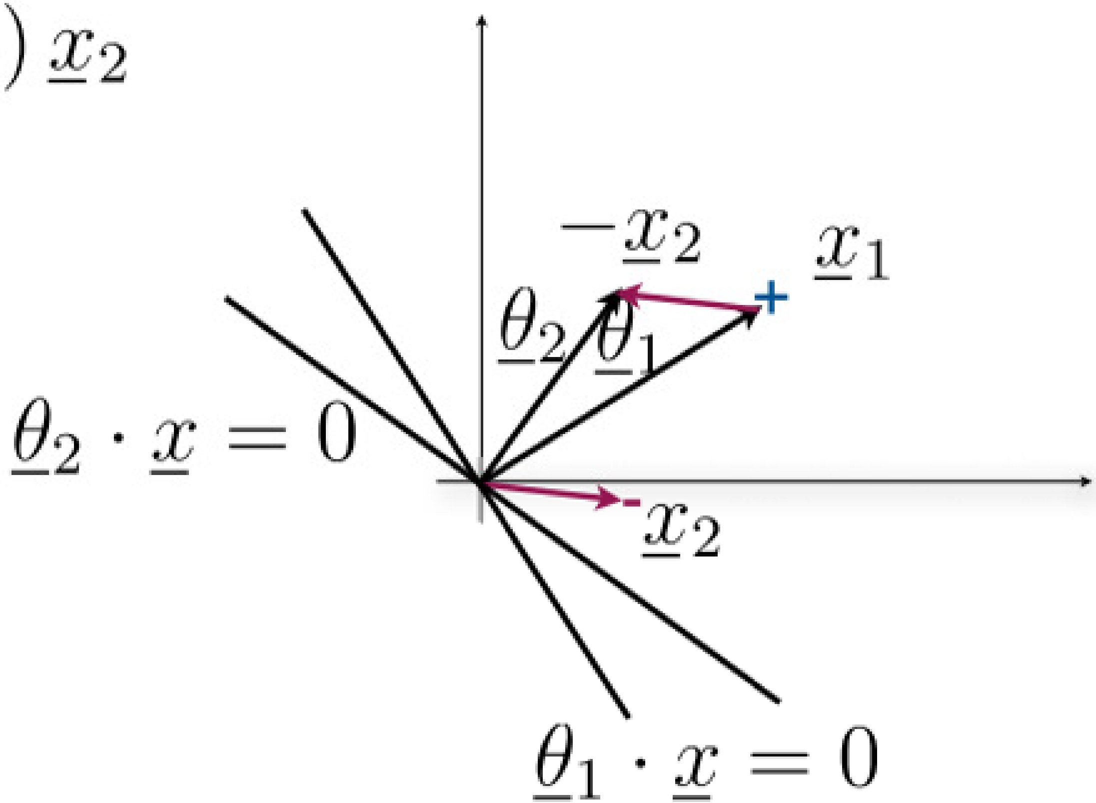
$$\underline{\theta}_2 = \underline{\theta}_1 + (-1)\,\underline{x}_2$$



Perceptron algorithm (take 2)

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$
$$\underline{\theta}_1 = \underline{\theta}_0 + 1\,\underline{x}_1$$
$$\underline{\theta}_2 = \underline{\theta}_1 + (-1)\,\underline{x}_2$$



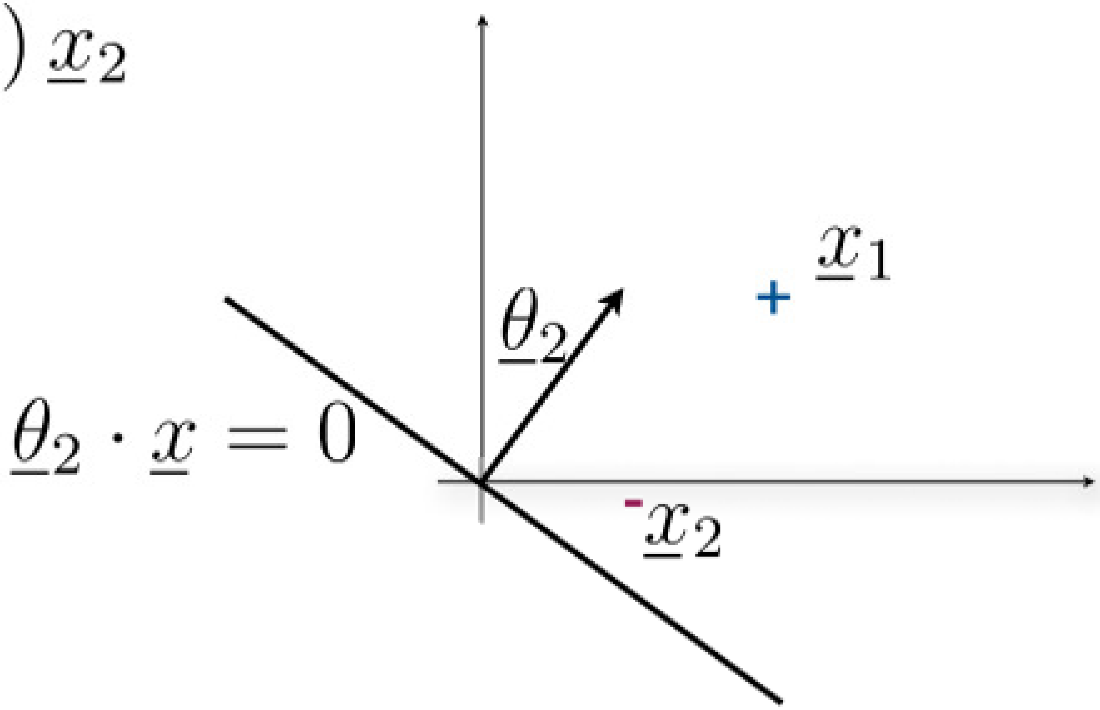
Perceptron algorithm (take 2)

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$

$$\underline{\theta}_1 = \underline{\theta}_0 + 1\,\underline{x}_1$$

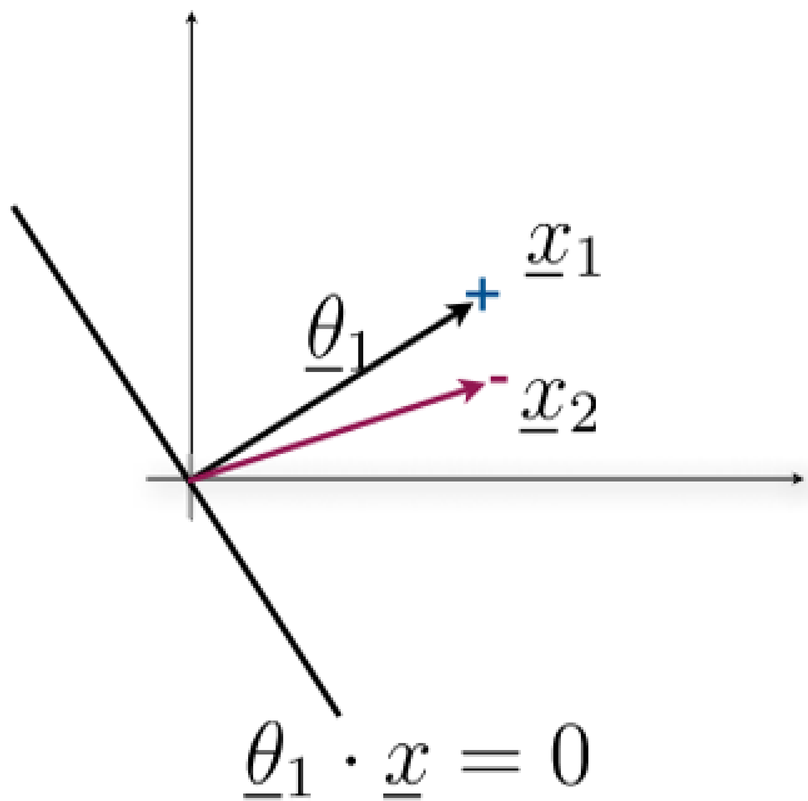
$$\underline{\theta}_2 = \underline{\theta}_1 + (-1)\,\underline{x}_2$$



Perceptron algorithm (take 3)

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$
$$\underline{\theta}_1 = \underline{\theta}_0 + 1\,\underline{x}_1$$



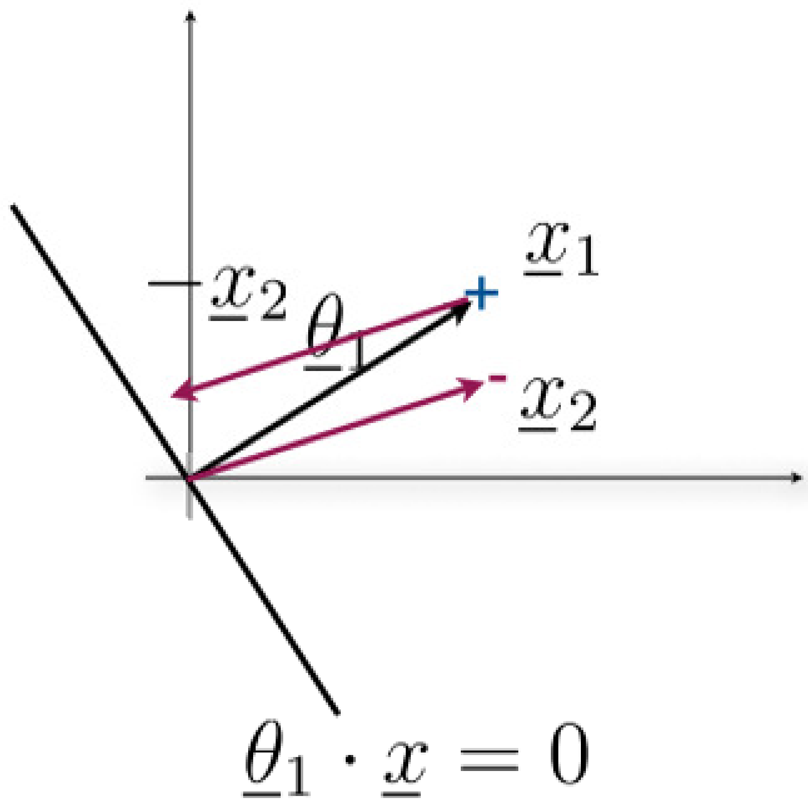
Perceptron algorithm (take 3)

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$

$$\underline{\theta}_1 = \underline{\theta}_0 + 1\,\underline{x}_1$$

$$\underline{\theta}_2 = \underline{\theta}_1 + (-1)\,\underline{x}_2$$



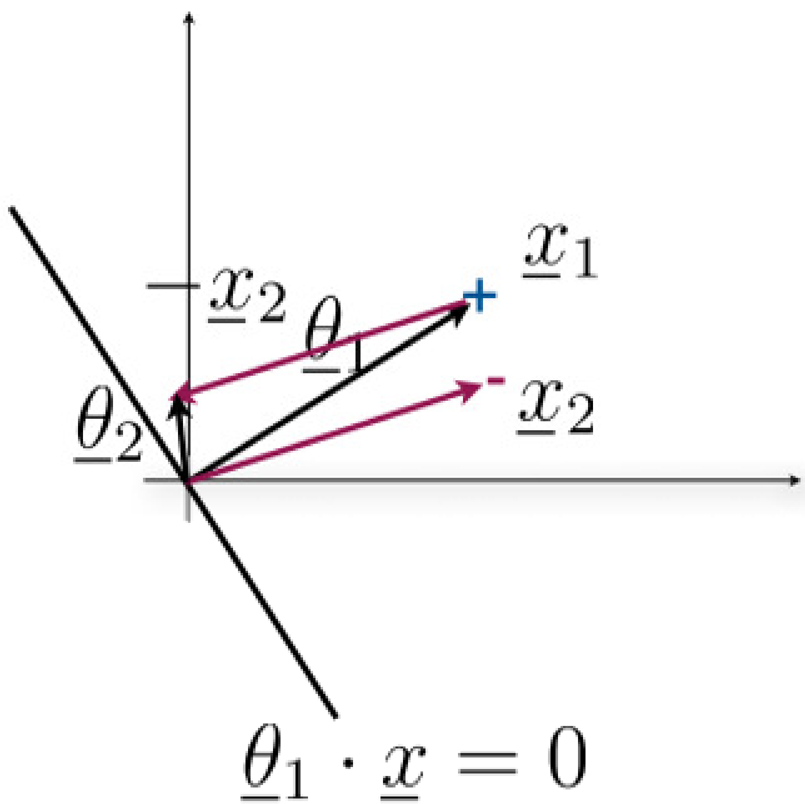
Perceptron algorithm (take 3)

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$

$$\underline{\theta}_1 = \underline{\theta}_0 + 1\,\underline{x}_1$$

$$\underline{\theta}_2 = \underline{\theta}_1 + (-1)\,\underline{x}_2$$



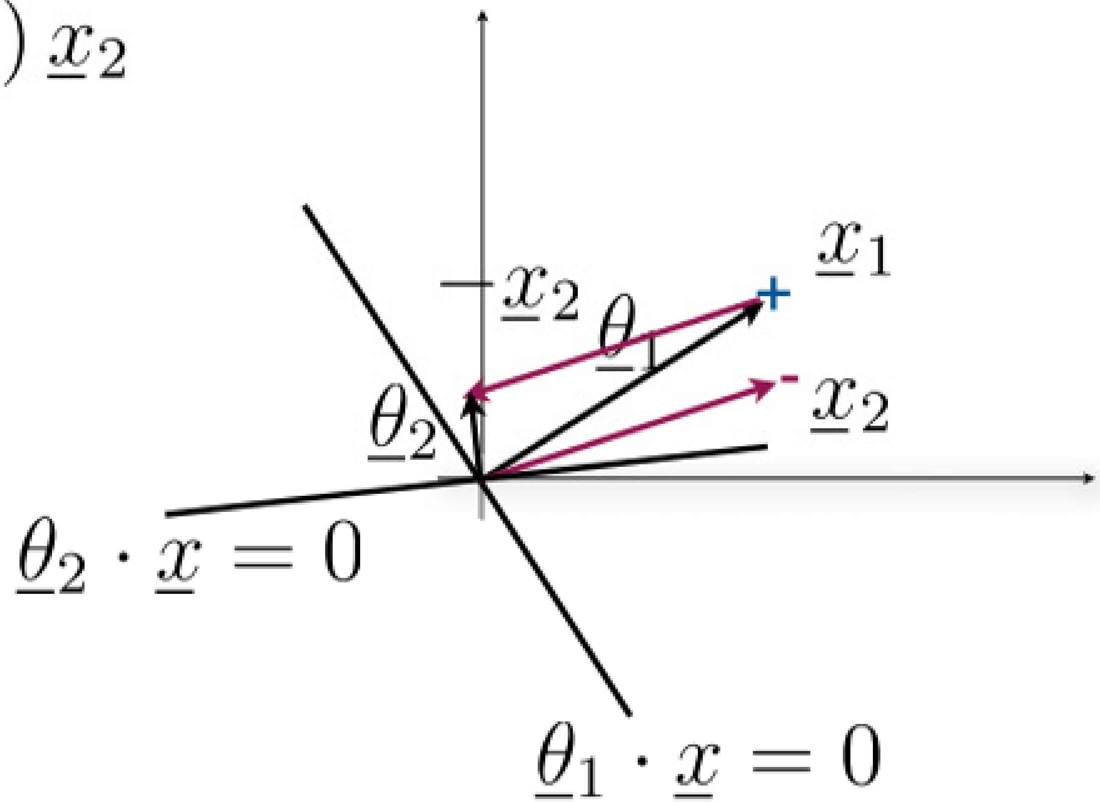
Perceptron algorithm (take 3)

- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$

$$\underline{\theta}_1 = \underline{\theta}_0 + 1\,\underline{x}_1$$

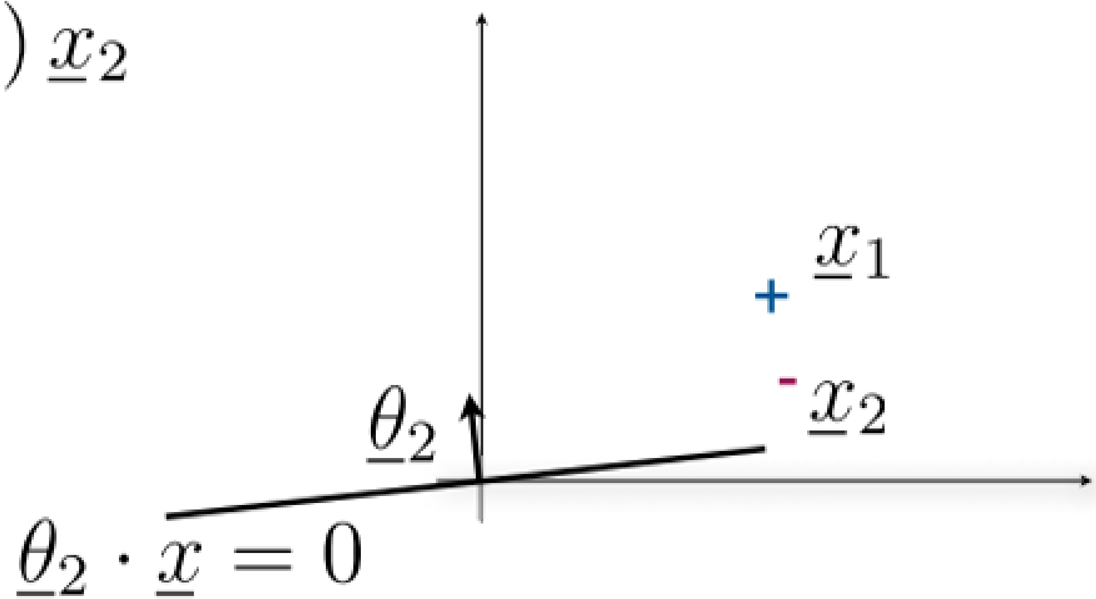
$$\underline{\theta}_2 = \underline{\theta}_1 + (-1)\,\underline{x}_2$$



Perceptron algorithm (take 3)

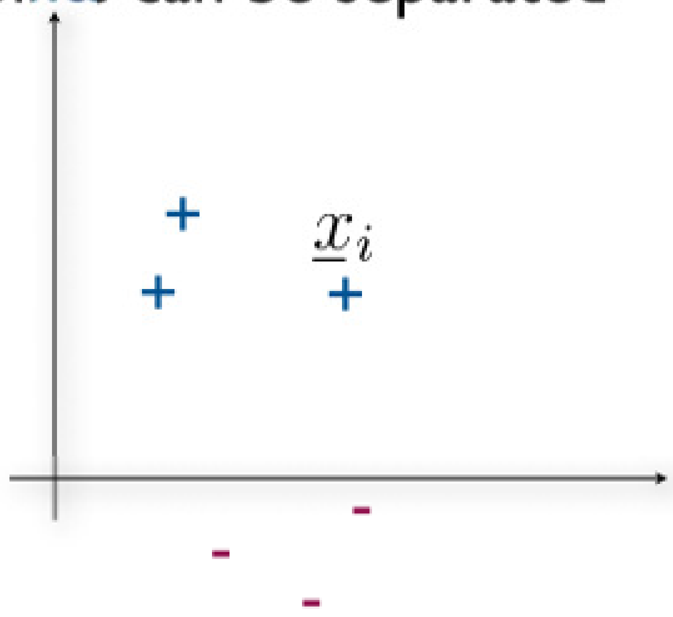
- Iterative updates based on mistakes

$$\underline{\theta}_0 = 0$$
$$\underline{\theta}_1 = \underline{\theta}_0 + 1\,\underline{x}_1$$
$$\underline{\theta}_2 = \underline{\theta}_1 + (-1)\,\underline{x}_2$$



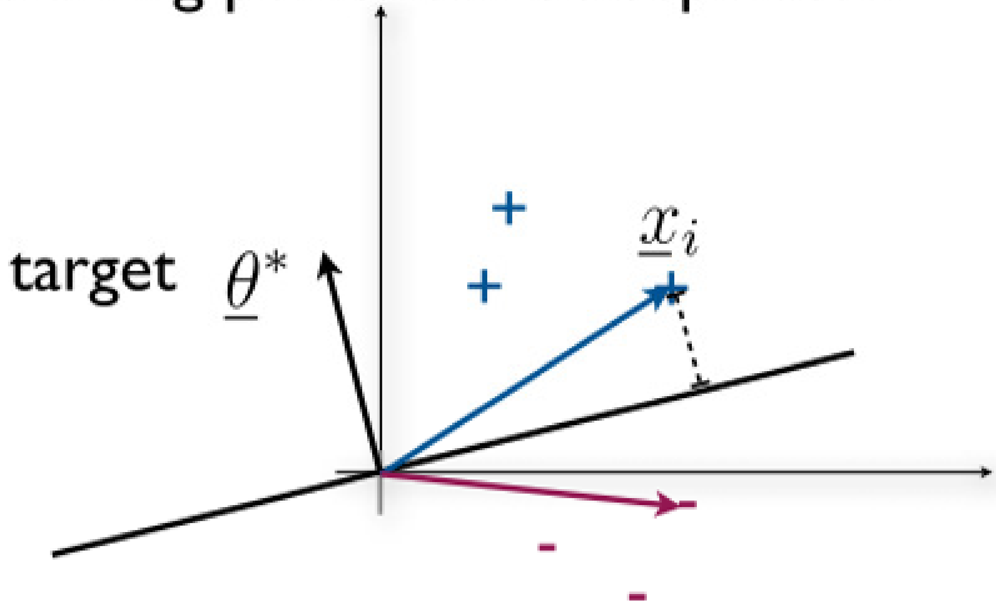
Number of mistakes

- We can bound the number of mistakes that the perceptron algorithm makes on the training set by assuming that there exists a target classifier with specific properties
- One such key property is margin, i.e., how well the training points can be separated



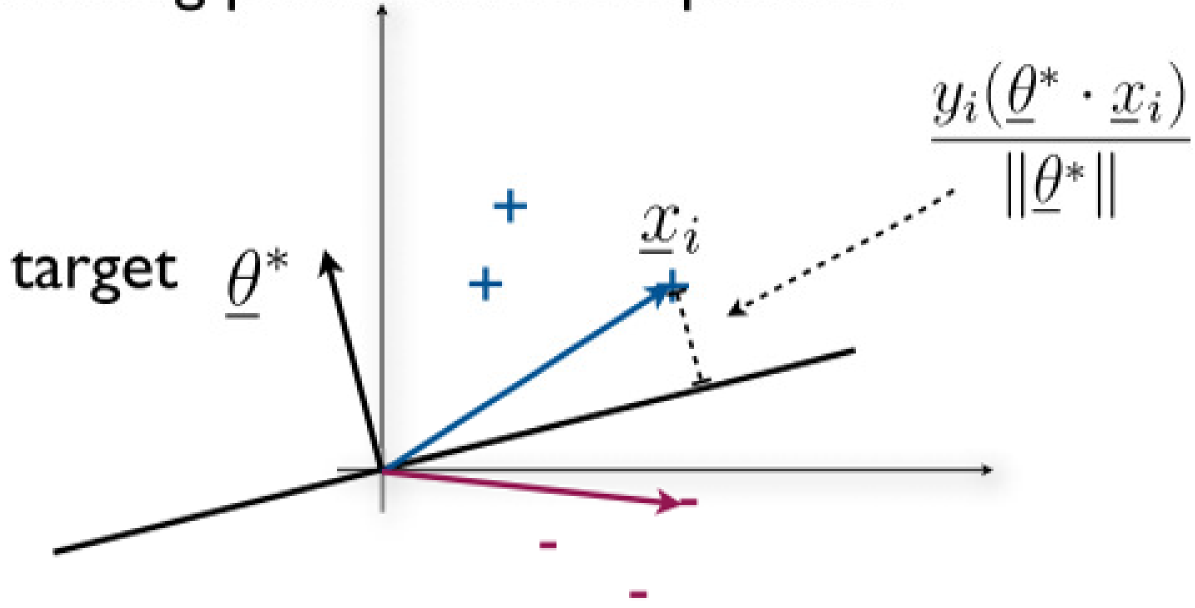
“Margin”

- We can bound the number of mistakes that the perceptron algorithm makes on the training set by assuming that there exists a target classifier with specific properties
- One such key property is margin, i.e., how well the training points can be separated



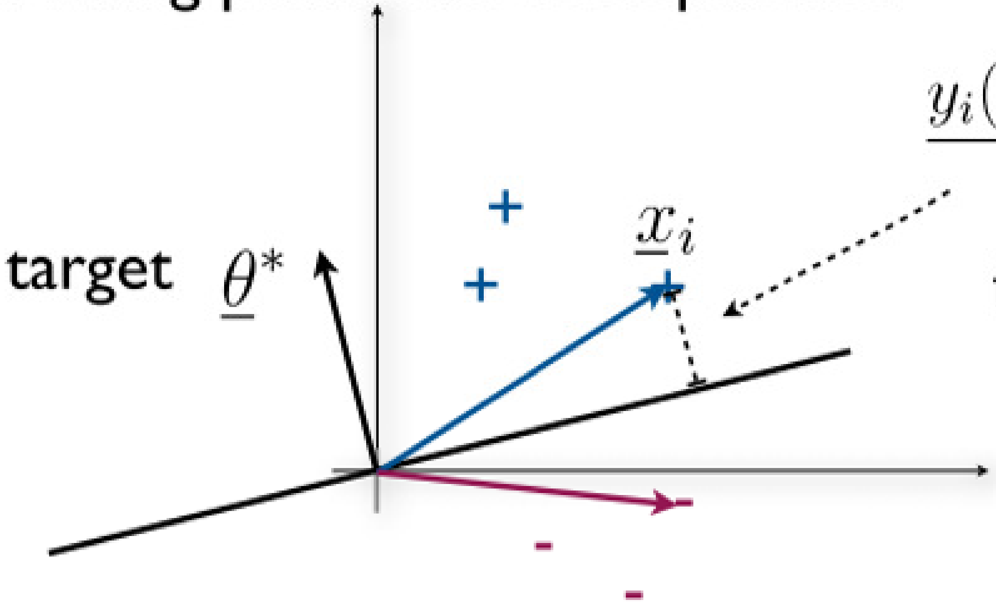
"Margin"

- We can bound the number of mistakes that the perceptron algorithm makes on the training set by assuming that there exists a target classifier with specific properties
- One such key property is margin, i.e., how well the training points can be separated



“Margin”

- We can bound the number of mistakes that the perceptron algorithm makes on the training set by assuming that there exists a target classifier with specific properties
- One such key property is margin, i.e., how well the training points can be separated



$$\frac{y_i(\underline{\theta}^* \cdot \underline{x}_i)}{\|\underline{\theta}^*\|} \geq \gamma_g \quad \text{= margin}$$

for all $i = 1, \ldots, n$

Perceptron convergence theorem

- **Assumption 1**: Suppose there exists $\underline{\theta}^*$ that attains margin γ_g for all the points (in the training set),

$$\frac{y_i(\underline{\theta}^* \cdot \underline{x}_i)}{\|\underline{\theta}^*\|} \geq \gamma_g, \quad i = 1, \ldots, n$$

- **Assumption 2**: all the points are bounded $\|\underline{x}_i\| \leq R$

Perceptron convergence theorem

- **Assumption 1**: Suppose there exists $\underline{\theta}^*$ that attains margin γ_g for all the points (in the training set),

$$\frac{y_i(\underline{\theta}^* \cdot \underline{x}_i)}{\|\underline{\theta}^*\|} \geq \gamma_g, \quad i = 1, \ldots, n$$

- **Assumption 2**: all the points are bounded $\|\underline{x}_i\| \leq R$

- **Theorem** Under the assumptions 1 and 2, the perceptron algorithm makes at most

$$\frac{R^2}{\gamma_g^2}$$

mistakes (on the training set)

- NOTE: the results does not depend on the dimension d of the examples or the number of training points n

The original source of these lecture slides

is the course materials of MIT 6.867

Machine Learning (Fall 2010)

by Prof. Tommi Jaakkola.