Lecture 6: Linear Model & Perceptron

Fall 2022

Kai-Wei Chang

CS @ UCLA

kw+cm146@kwchang.net

The instructor gratefully acknowledges Dan Roth, Vivek Srikuar, Sriram Sankararaman, Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Announcement

- Quiz 1 will be due Today!!

- Hw1 update (please see the pinned message at Piazza)

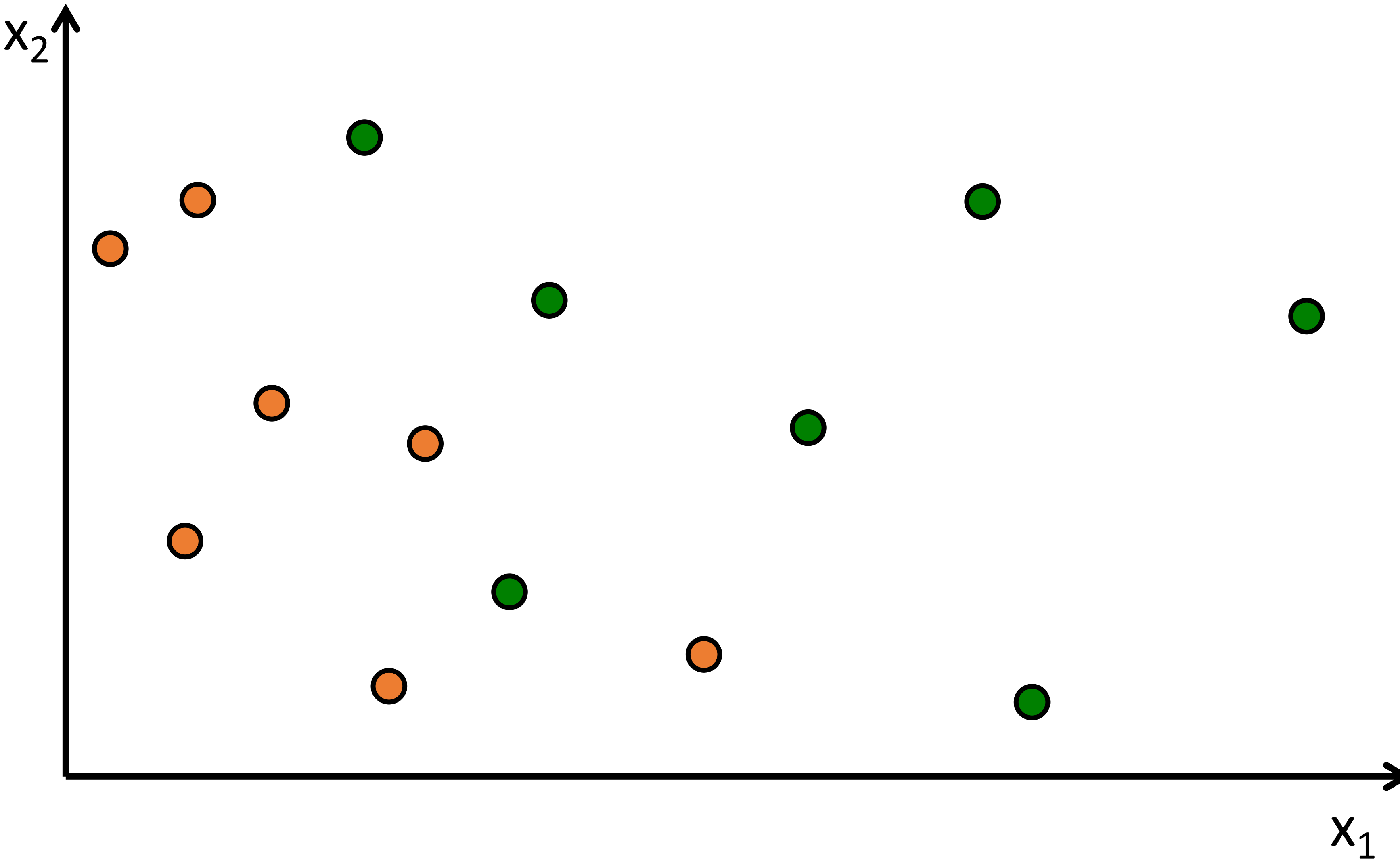
```python

# Shuffle the data for cross-validation
import random
idx = list(range(n))
random.shuffle(idx)
X = np.take(X, idx, axis=0)
y = np.take(y, idx, axis=0)
```

nks, Wenda Fu, for identifying these issues.

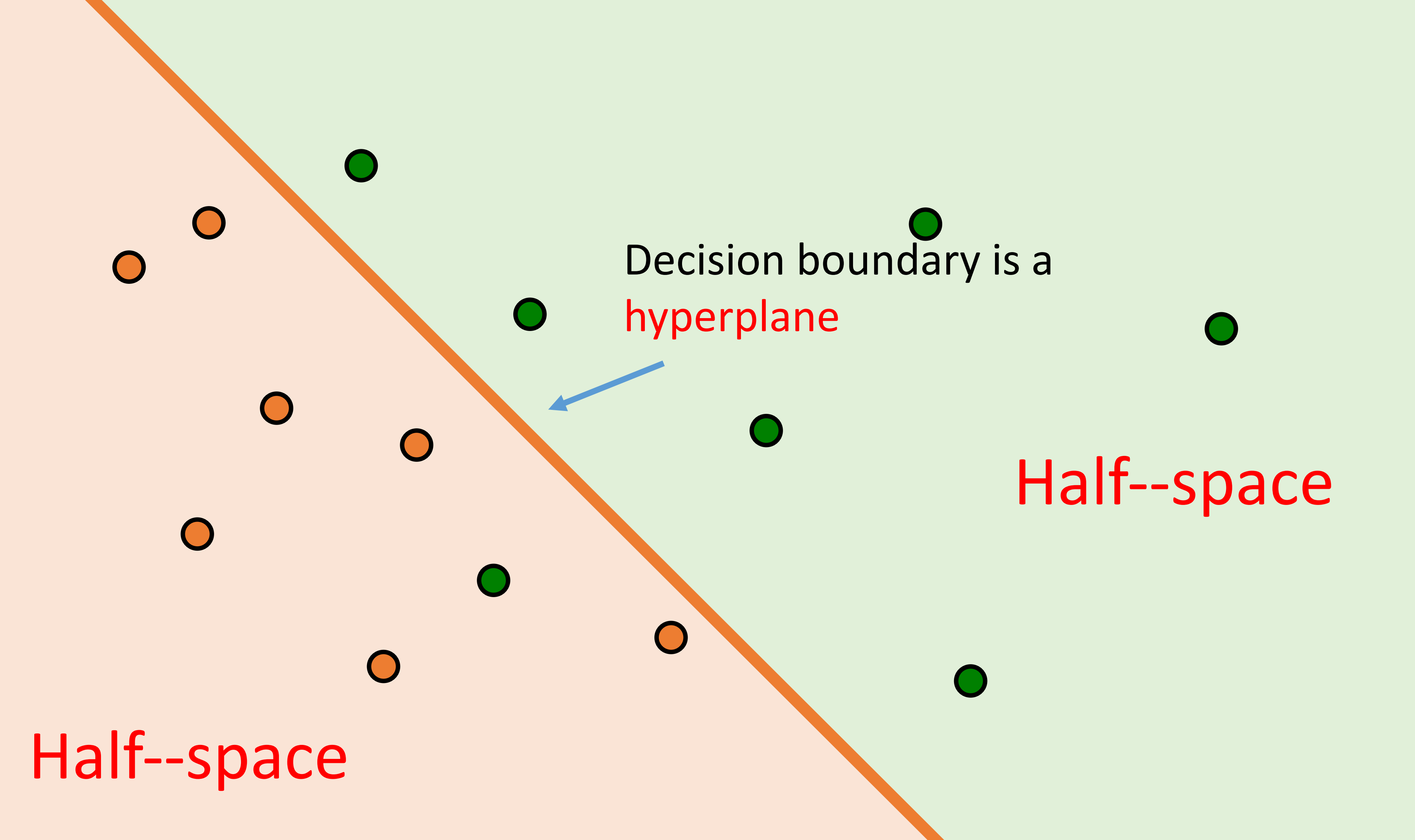
What you will Learn Today

- Linear model
  - Basic linear algebra & linear classifier
  - Trick to remove bias term b in $w^T x + b = 0$
- Perceptron Algorithm
  - Perceptron Update
  - Why it works
  - Convergence theorem – mistake bound

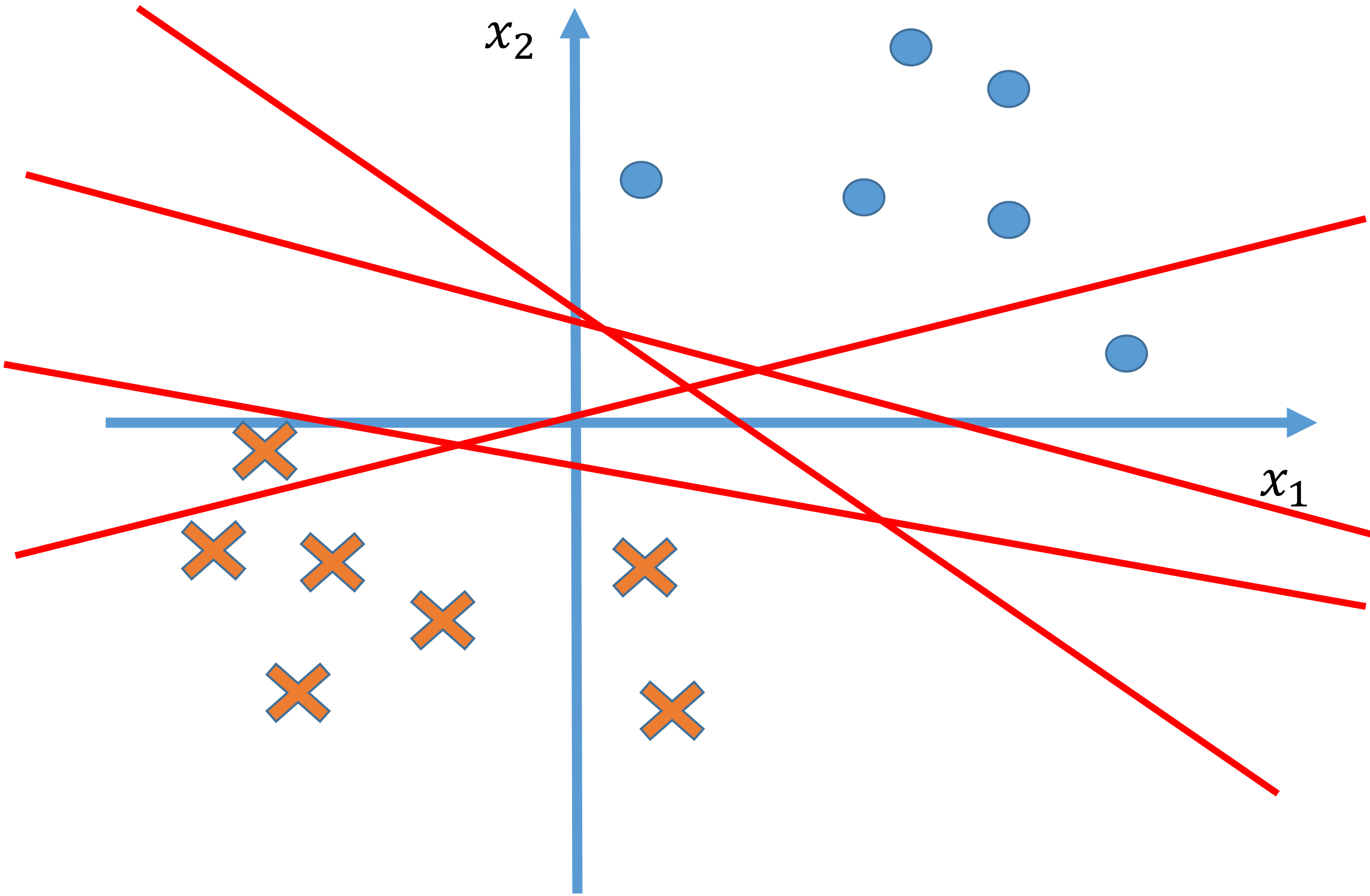
Training data



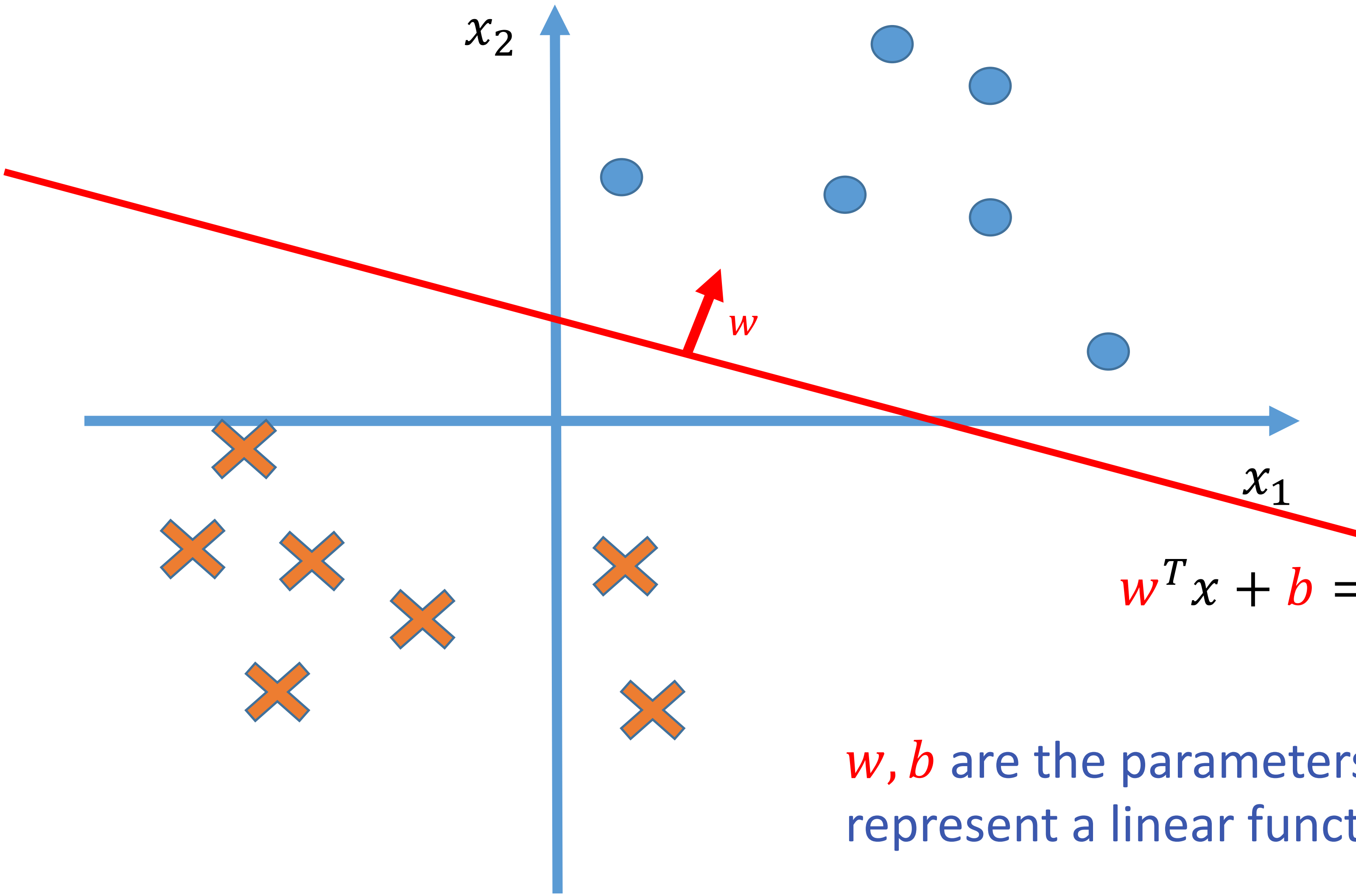
Hyperplane Separates the Space



Hypothesis space: linear models



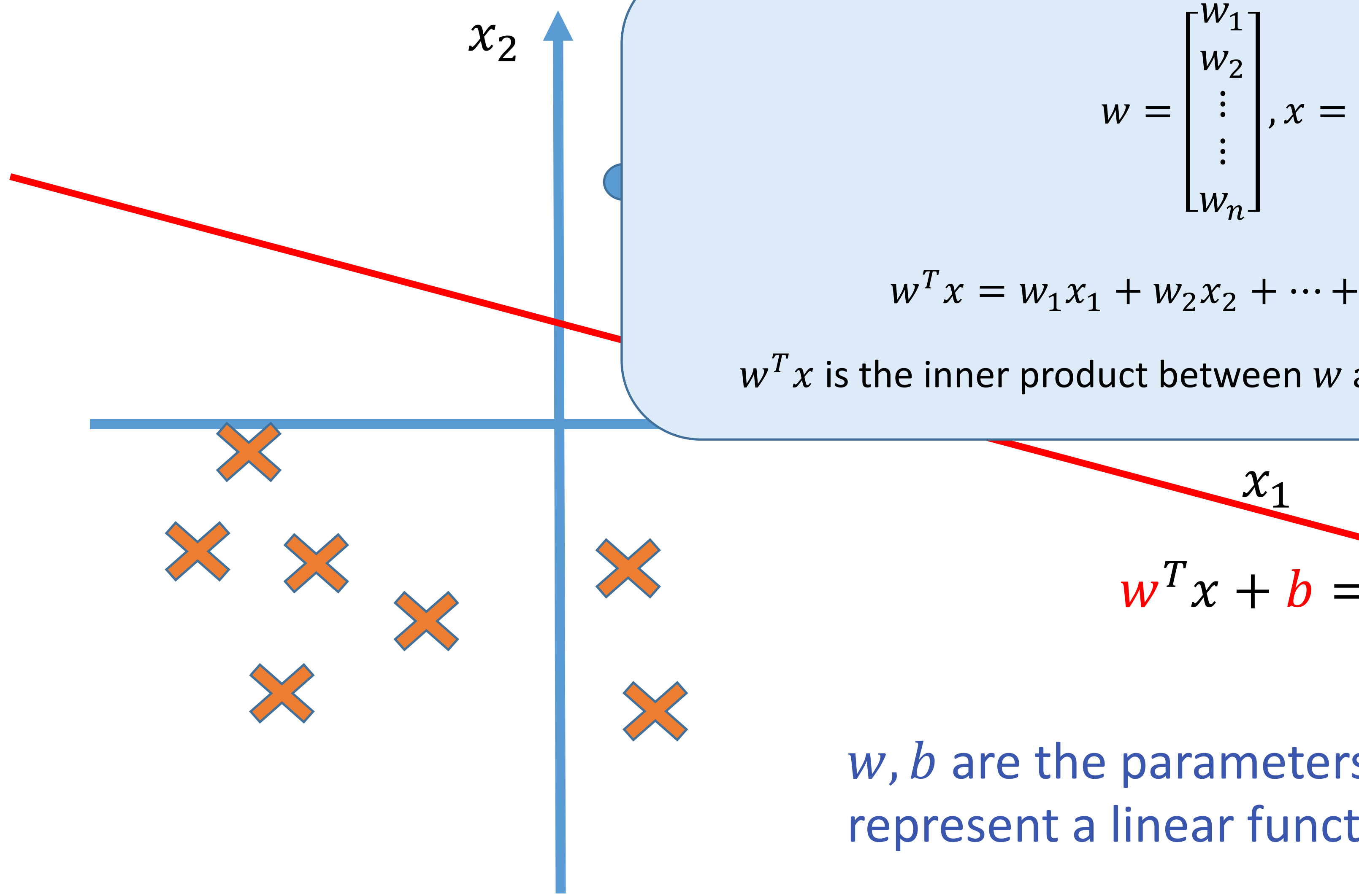
Hypothesis space: linear model



$w^T x + b = 0$

w, b are the parameters to represent a linear function

Hypothesis space: linear model



$$w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ \vdots \\ w_n \end{bmatrix}, x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{bmatrix}$$

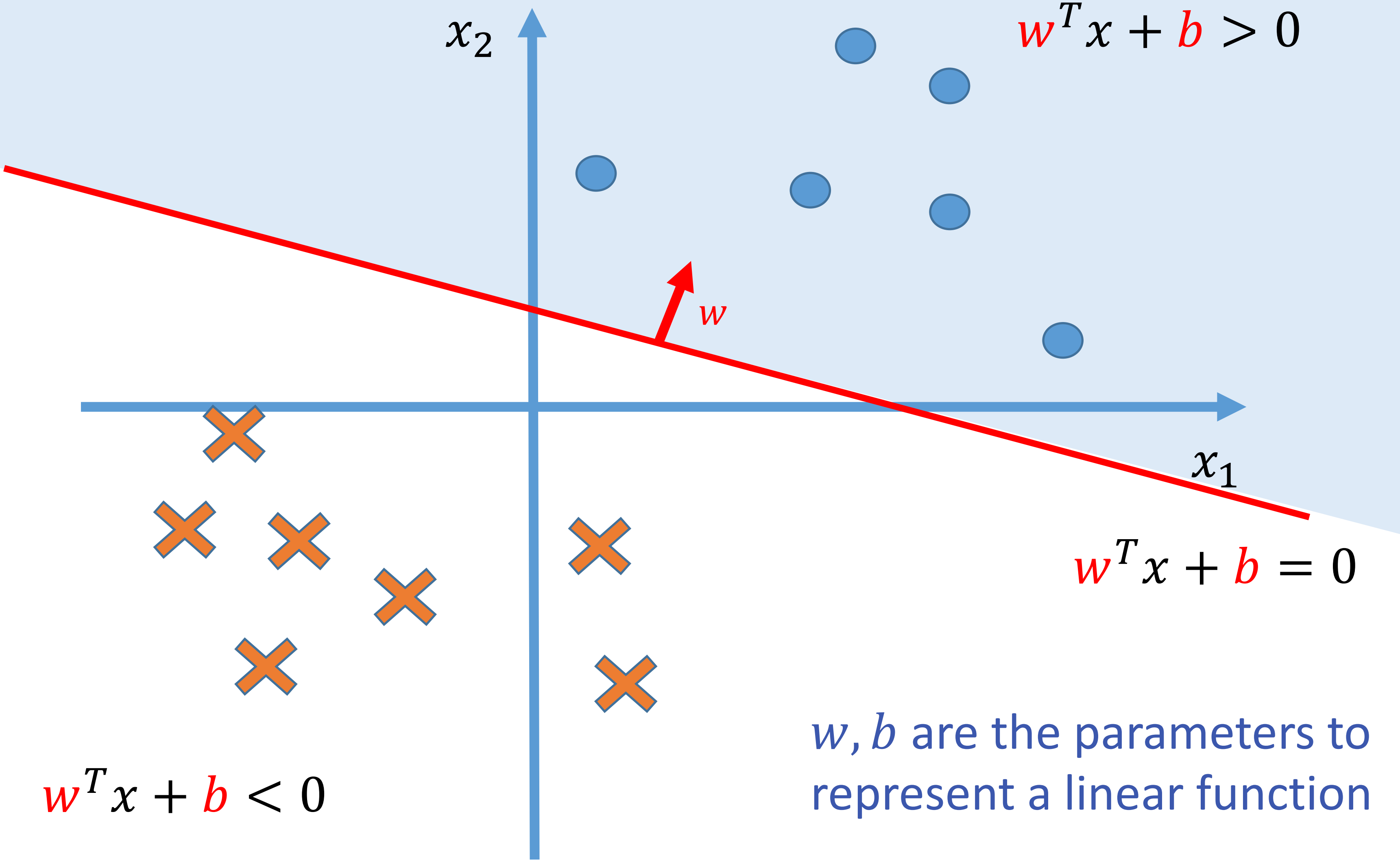
$$w^T x = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

$w^T x$ is the inner product between w and x

$$w^T x + b = 0$$

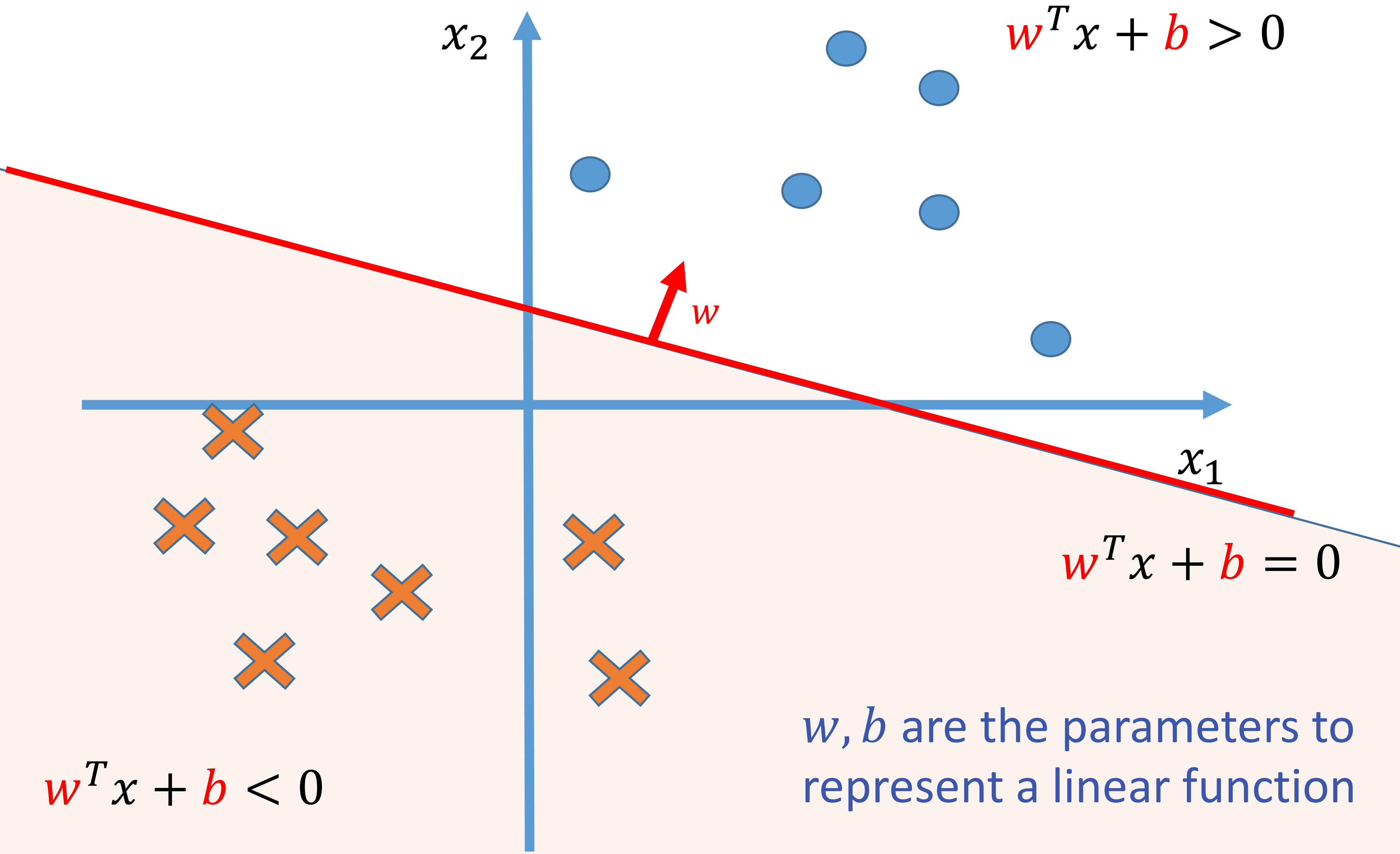
w, b are the parameters to represent a linear function

Hypothesis space: linear model



w, b are the parameters to represent a linear function

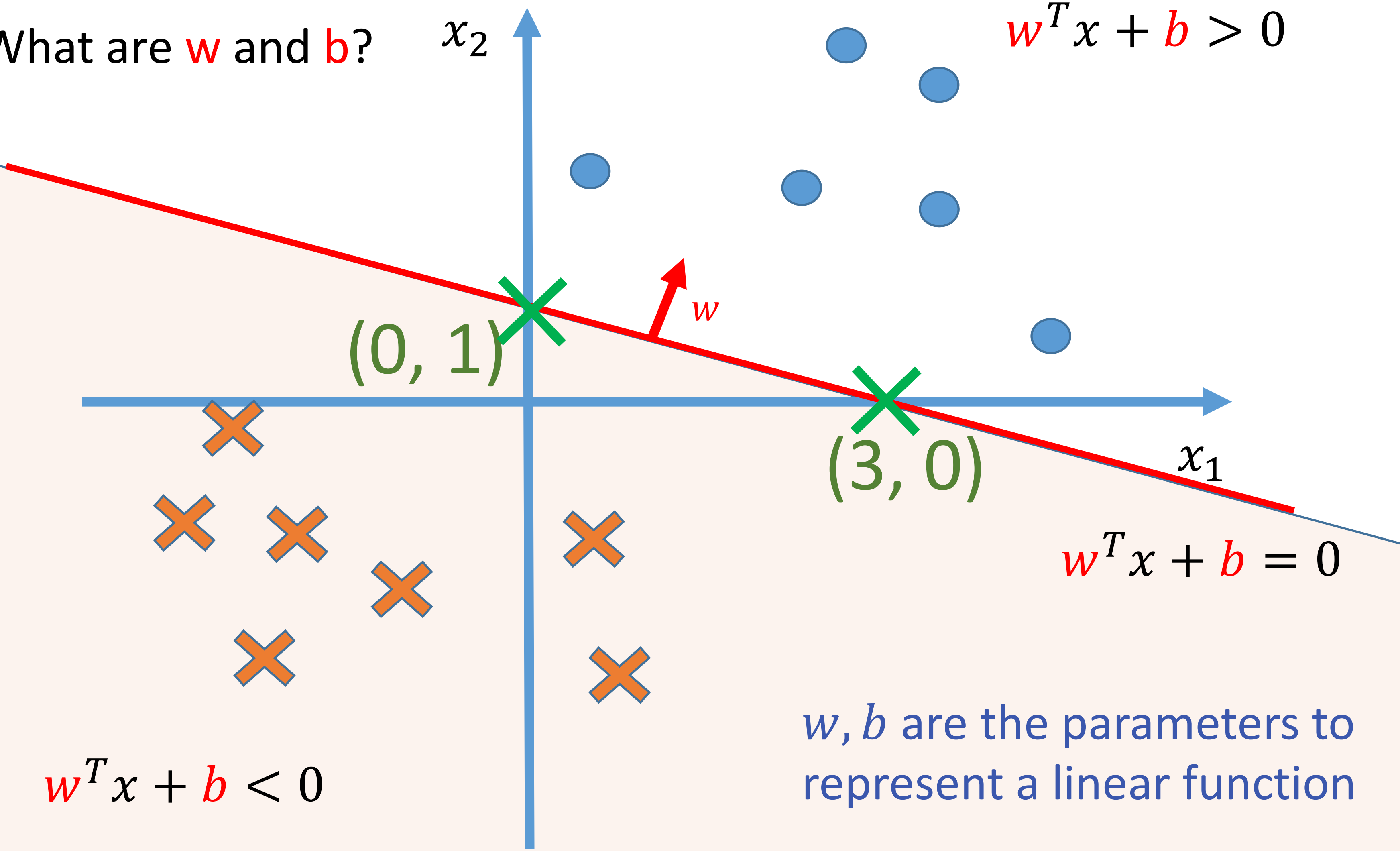
Hypothesis space: linear model



w, b are the parameters to represent a linear function

Example

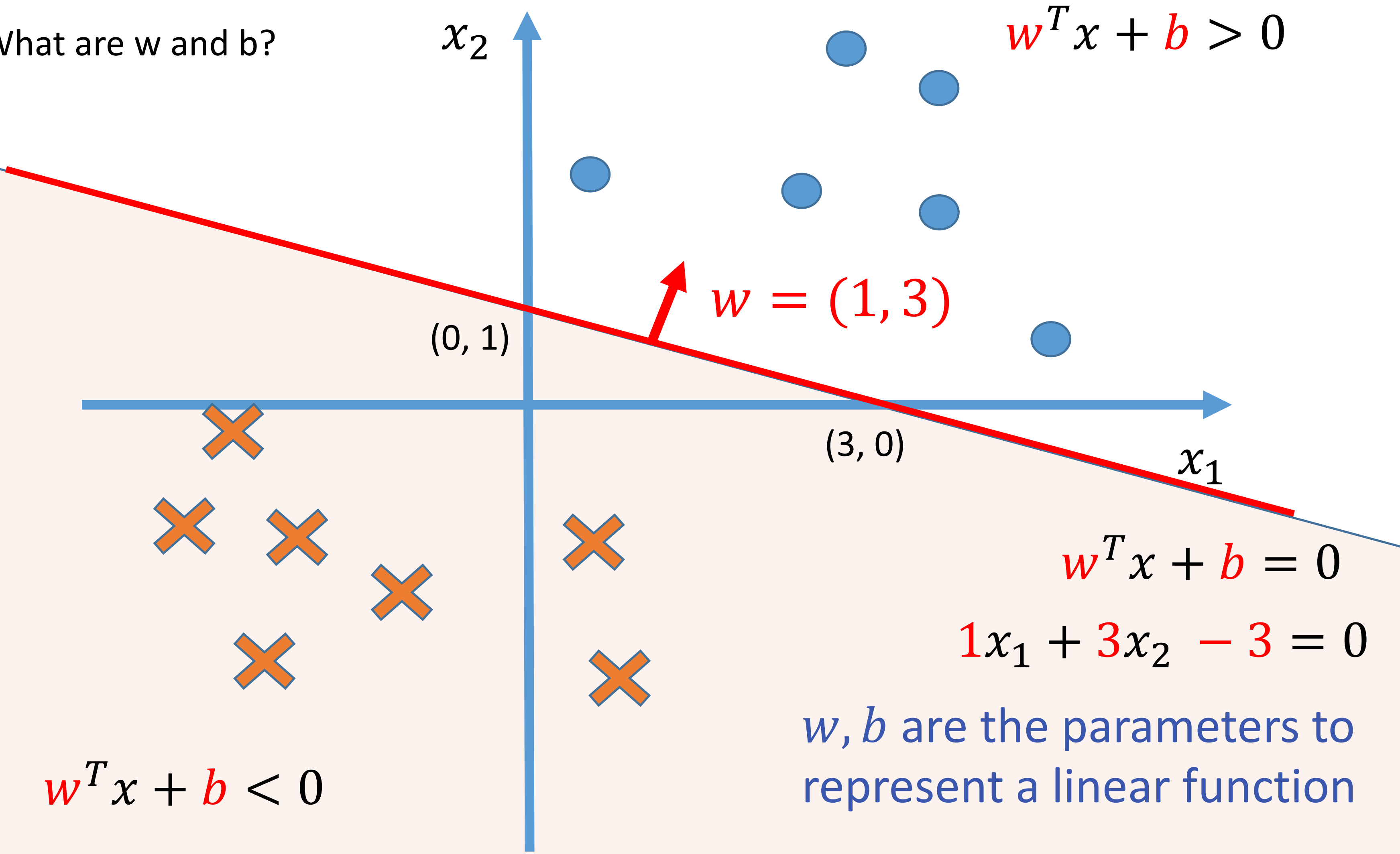
What are w and b?



w, b are the parameters to represent a linear function

Example

What are w and b?



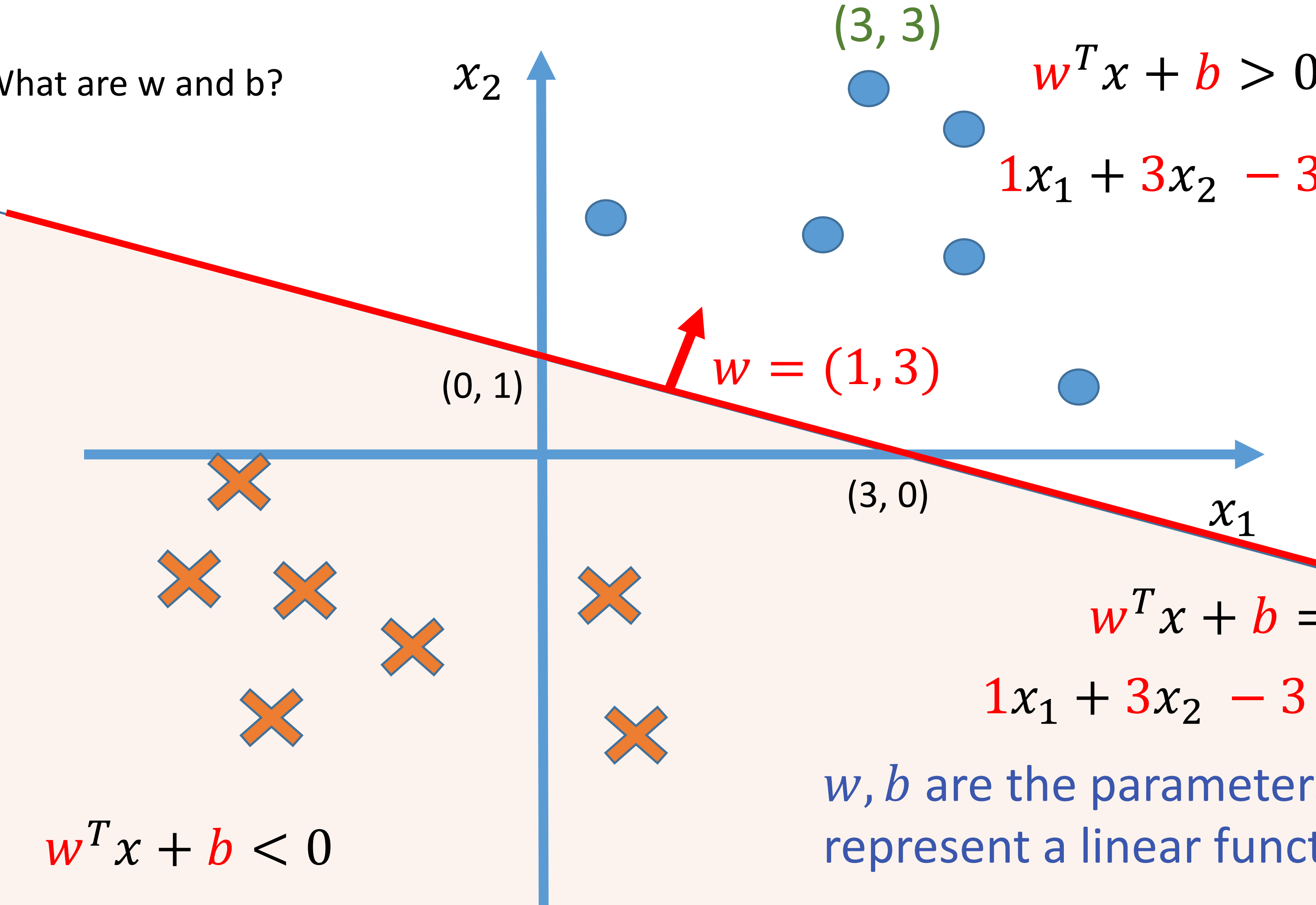
$$w^T x + b = 0$$

$$1x_1 + 3x_2 - 3 = 0$$

w, b are the parameters to represent a linear function

Example

What are w and b?



$$w^T x + b > 0$$
$$1x_1 + 3x_2 - 3 > 0$$

$$w^T x + b = 0$$
$$1x_1 + 3x_2 - 3 = 0$$

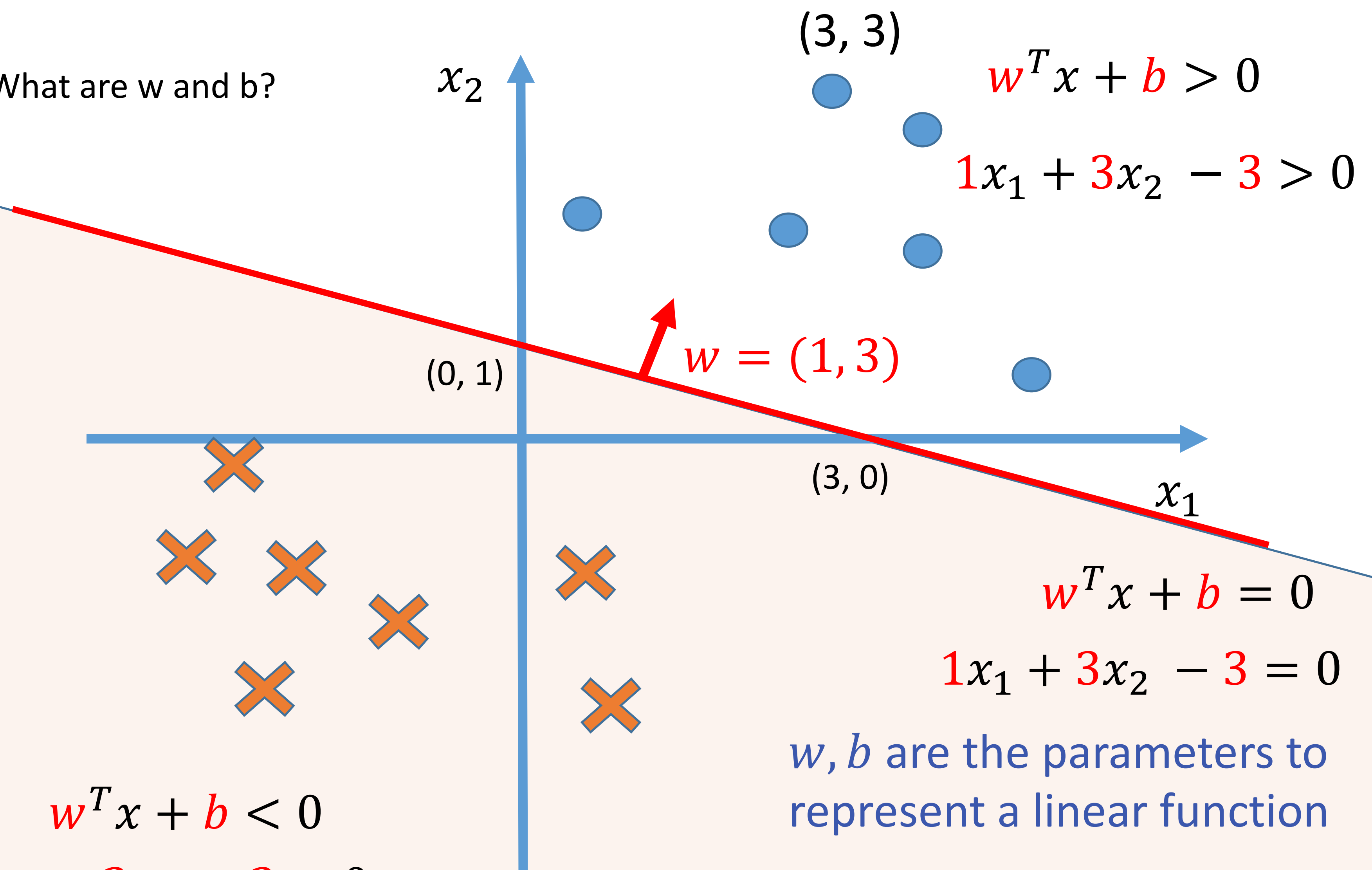
w, b are the parameters to represent a linear function

$$w^T x + b < 0$$
$$1x_1 + 3x_2 - 3 < 0$$

Example

What are w and b?

$3 + 9 - 3 = 9 > 0$



$w^T x + b > 0$

$1x_1 + 3x_2 - 3 > 0$

$w^T x + b = 0$

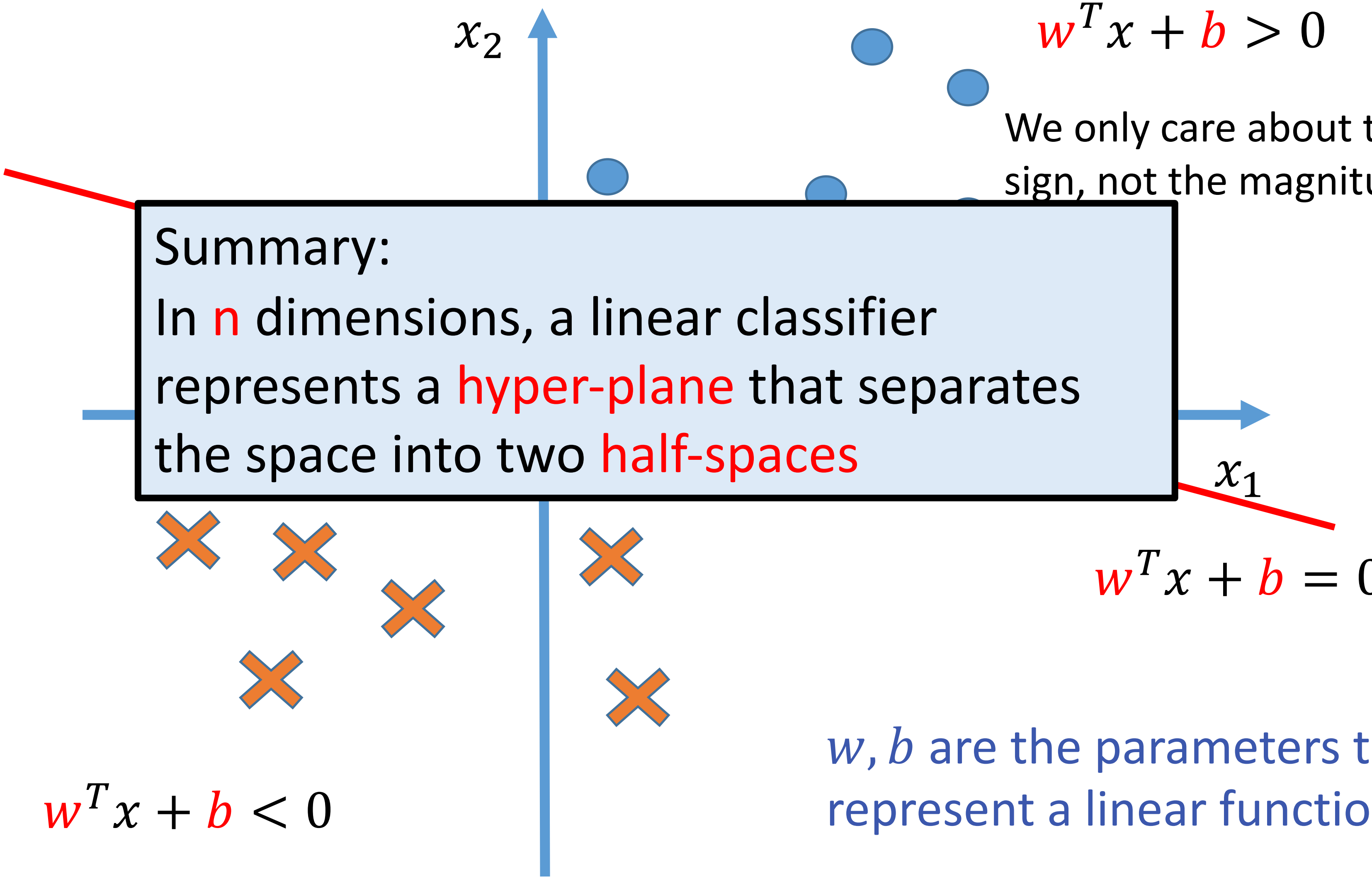
$1x_1 + 3x_2 - 3 = 0$

w, b are the parameters to represent a linear function

$w^T x + b < 0$

$1x_1 + 3x_2 - 3 < 0$

Hypothesis space: linear model



We only care about the sign, not the magnitude

Summary:
In n dimensions, a linear classifier represents a hyper-plane that separates the space into two half-spaces

w, b are the parameters to represent a linear function

Linear Models for Binary Classification

- Given training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}, x \in R^d, y \in \{-1, +1\}$, we use them to learn a hypothesis function $h \in H$

$$H = \{\, h \mid h(\boldsymbol{x}) = \mathrm{sgn}(w^T x + b) \,\}$$

such that $y = h(x)$

$$\mathrm{sgn}(z) = \begin{cases} 1 \text{ if } z \geq 0 \\ -1 \text{ otherwise} \end{cases}$$

Note: when z=0 we can either assign sgn(z) = 1 or -1

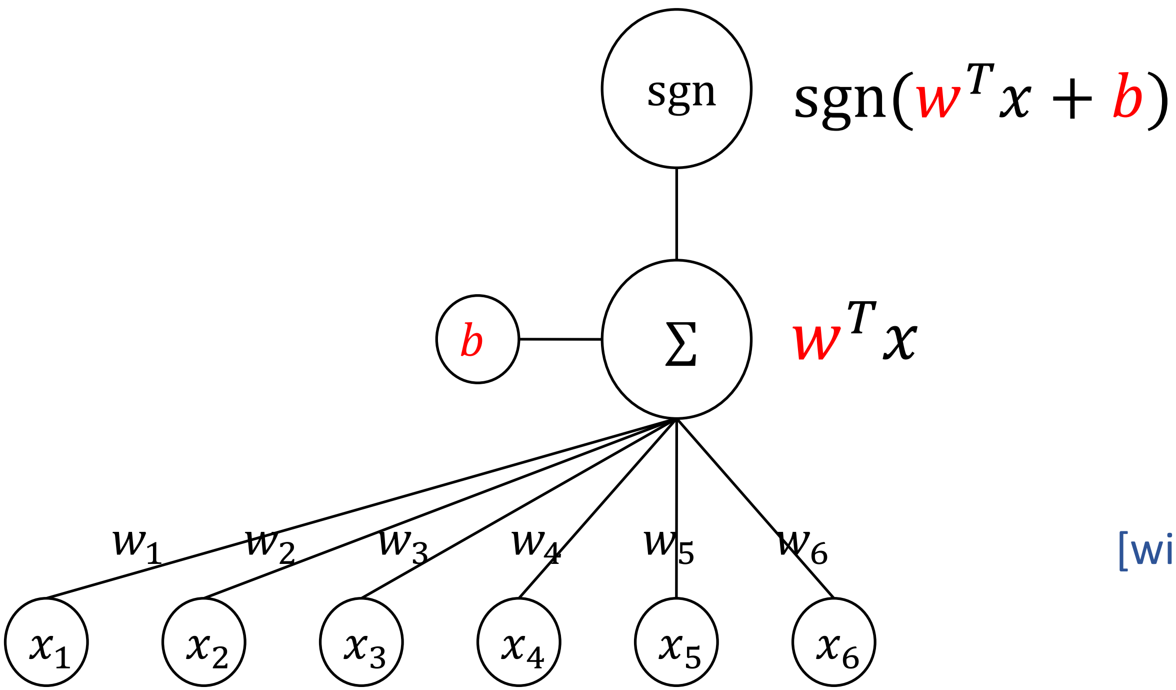
model parameters (learnable parameters)

Learn = train = find the best parameters w, b

Footnote: For some algorithms it is mathematically easier to represent False as -1, and at other times, as 0. For the Perceptron algorithm, treat -1 as false and +1 as true.

Linear Classifiers

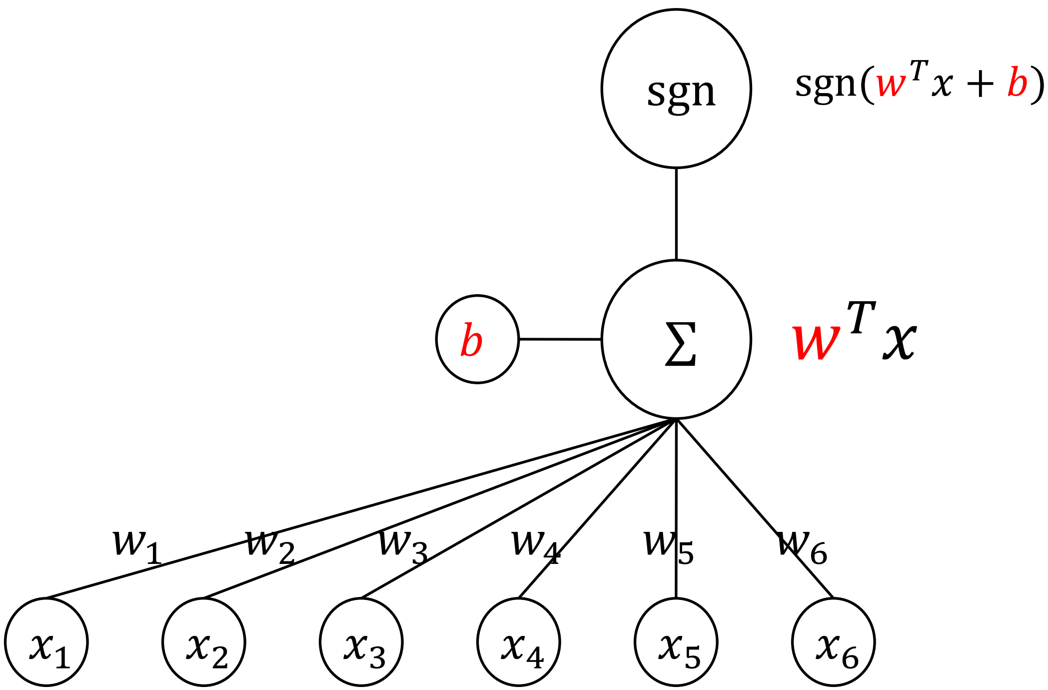
- *Linear classifiers* classify an example **x** using the following classification rule



$$[\text{wind=weak}] = \begin{cases} 1 & \text{if wind} = \text{weak} \\ 0 & \text{otherwise} \end{cases}$$

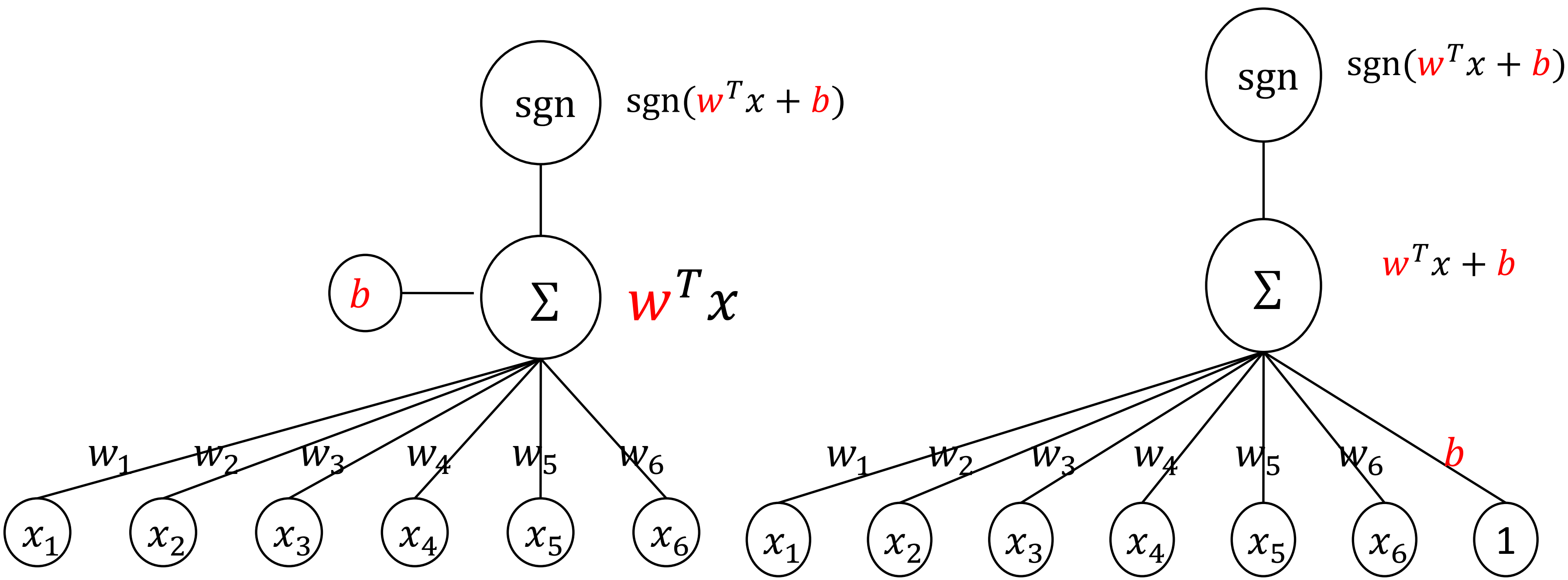
E.g., 3 * [wind=weak] + 6 * [outlook=sunny] – 2* [temperature=high] + … - 2 > 0

Linear Classifiers



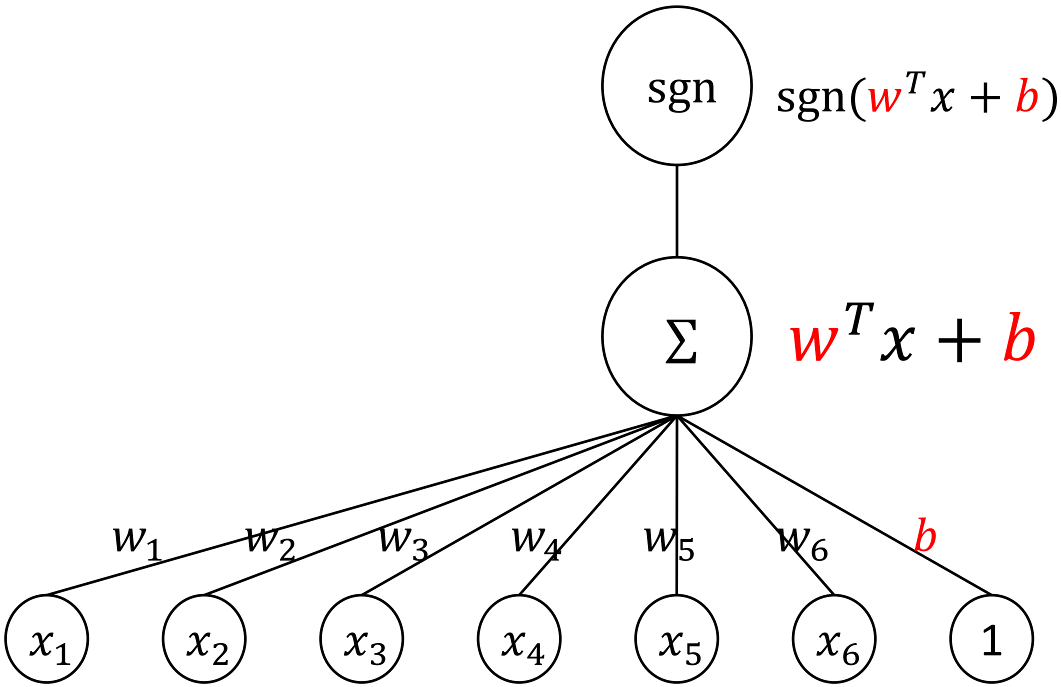
E.g., 3 * [wind=weak] + 6 * [outlook=sunny] – 2* [temperature=high] + … - 2 > 0

Linear Classifiers



E.g., 3 * [wind=weak] + 6 * [outlook=sunny] – 2* [temperature=high] + … - 2 > 0

A simple trick to remove the bias term b

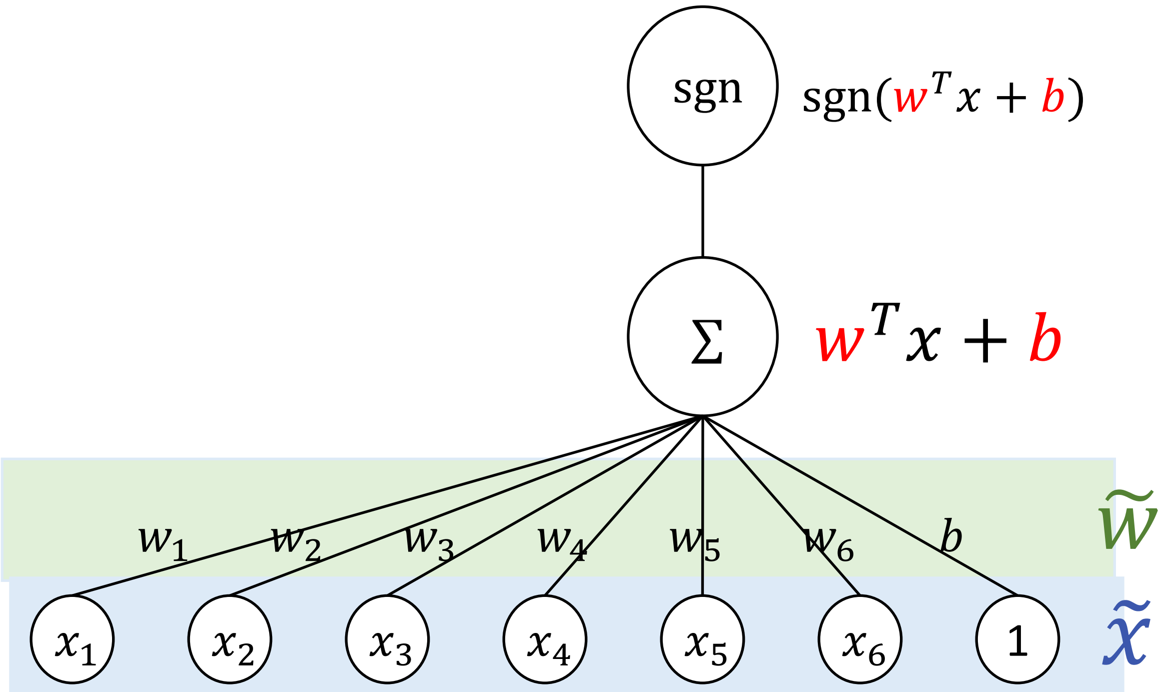


$$
\begin{aligned}
& w^T x + b \\
&= [w^T \; b] \cdot \begin{bmatrix} x \\ 1 \end{bmatrix} \\
&= \tilde{w} \cdot \tilde{x}
\end{aligned}
$$

$$
\tilde{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \\ b \end{bmatrix}, \tilde{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ 1 \end{bmatrix}
$$

For simplicity, I may write $\tilde{w}$ and $\tilde{x}$ as w and x when there is no confusion

A simple trick to remove the bias term b



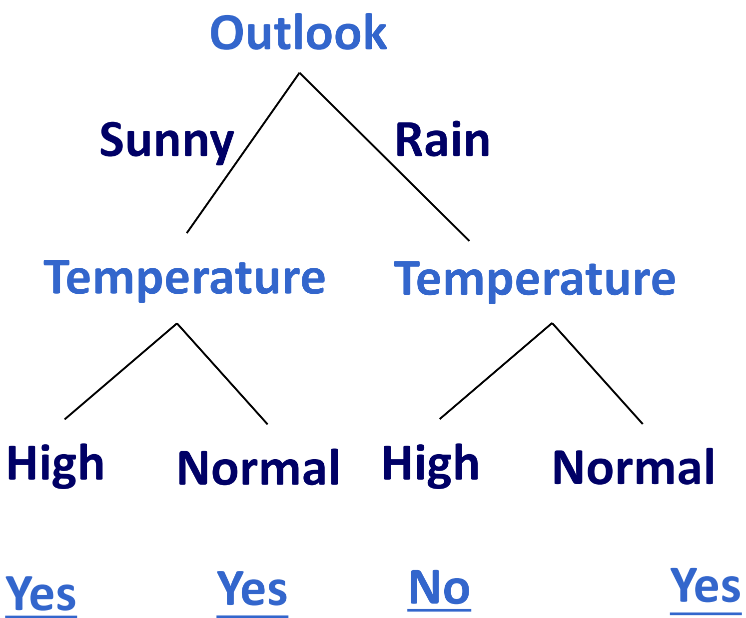
$$w^T x + b$$
$$= [w^T \; b] \cdot \begin{bmatrix} x \\ 1 \end{bmatrix}$$
$$= \tilde{w} \cdot \tilde{x}$$

$$\tilde{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \\ b \end{bmatrix}, \tilde{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ 1 \end{bmatrix}$$

For simplicity, I may write $\tilde{w}$ and $\tilde{x}$ as w and x when there is no confusion

Exercise

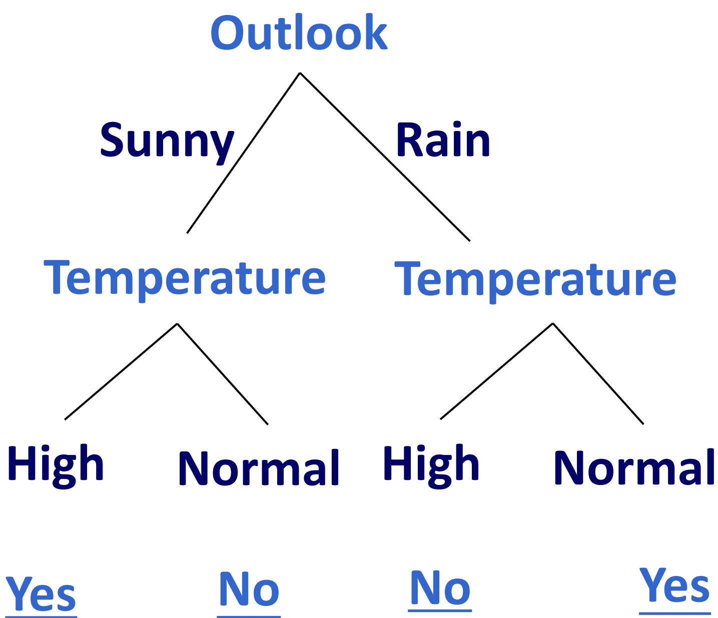
Find a linear model that is equivalent to the decision tree on the left



$$\tilde{w} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ b \end{bmatrix}, \tilde{x} = \begin{bmatrix} [\text{outlook} = \text{sunny}] \\ [\text{outlook} = \text{rain}] \\ [\text{temp} = \text{high}] \\ [\text{temp} = \text{normal}] \\ 1 \end{bmatrix}$$

Exercise

Find a linear model that is equivalent to the decision tree on the left



$$\tilde{w} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ b \end{bmatrix}, \tilde{x} = \begin{bmatrix} [\text{outlook} = \text{sunny}] \\ [\text{outlook} = \text{rain}] \\ [\text{temp} = \text{high}] \\ [\text{temp} = \text{normal}] \\ 1 \end{bmatrix}$$

Learning a Linear Classifier

Learn = train = find the best parameters w, b

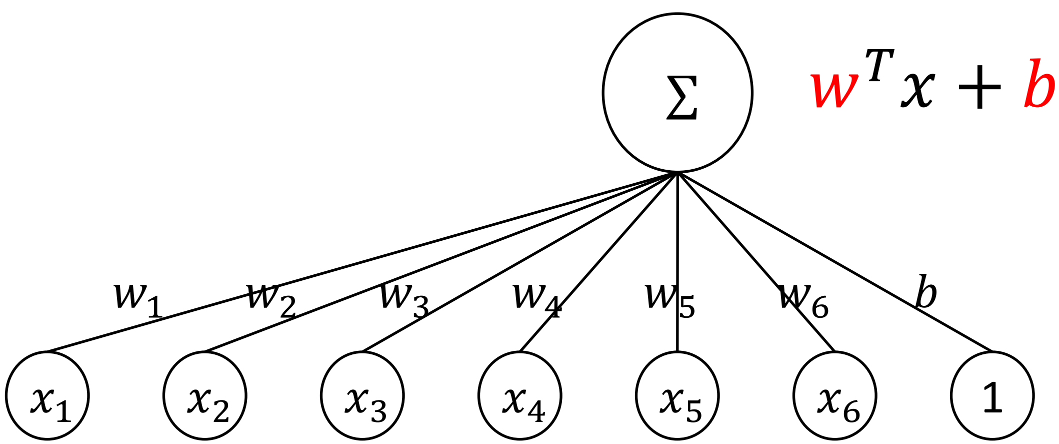
- There are several algorithms/models
  - Perceptron
  - Logistic Regression
  - (Linear) Support Vector Machines
  - Naïve Bayes
  - …
- Different methods define **best** in a different way

Linear Regression

- *Linear regression* maps an example **x** into a real value y
- Given training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}, x \in R^d, y \in R$, we use them to learn a hypothesis function $h \in H$

$$H = \{\, h \mid h(\boldsymbol{x}) = w^T x + b \,\}$$

such that $y = h(x)$



Perceptron

Psychological Review
Vol. 65, No. 6, 1958

THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN[1]

F. ROSENBLATT

Cornell Aeronautical Laboratory

The Hype

NEW NAVY DEVICE LEARNS BY DOING

Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser

WASHINGTON, July 7 (UPI) —The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.

The embryo—the Weather Bureau's $2,000,000 "704" computer—learned to differentiate between right and left after fifty attempts in the Navy's demonstration for newsmen.

The service said it would use this principle to build the first of its Perceptron thinking machines that will be able to read and write. It is expected to be finished in about a year at a cost of $100,000.

The New York Times, July 8 1958

HAVING told you about the giant digital computer known as I.B.M. 704 and how it has been taught to play a fairly creditable game of chess, we'd like to tell you about an even more remarkable machine, the perceptron, which, as its name implies, is capable of what amounts to original thought. The first perceptron has yet to be built,

The New Yorker, December 6, 1958 P. 44

The Hype

NEW NAVY DEVICE LEARNS BY DOING

Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser

WASHINGTON, July 7 (UPI) —The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.

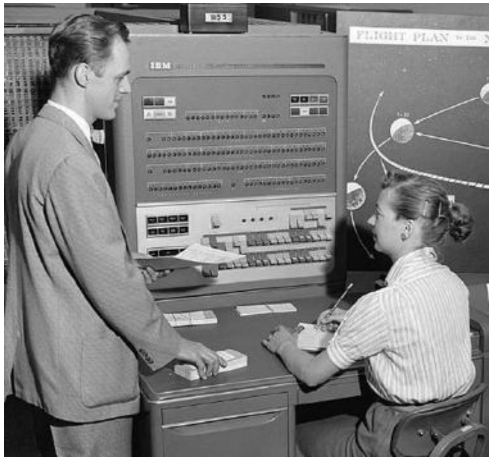
The embryo—the Weather Bureau's $2,000,000 "704" computer—learned to differentiate between right and left after fifty attempts in the Navy's demonstration for newsmen.

The service said it would use this principle to build the first of its Perceptron thinking machines that will be able to read and write. It is expected to be finished in about a year at a cost of $100,000.

The New York Times, July 8 1958

HAVING told you about the giant digital computer known as I.B.M. 704 and how it has been taught to play a fairly creditable game of chess, we'd like to tell you about an even more remarkable machine, the perceptron, which, as its name implies, is capable of what amounts to original thought. The first perceptron has yet to be built,

The New Yorker, December 6, 1958 P. 44



The IBM 704 computer

Note: The application scenario discussed in the video is not ideal.

The Perceptron algorithm

- The goal is to find a separating hyperplane
- An online algorithm
  - Processes one example at a time
- Converges if data is separable
  -- mistake bound

What you will learn about Perceptron

- Perceptron Algorithm
- The intuition behind the algorithm
- Convergence of Perceptron Algorithm
  - Mistake bound

mistake

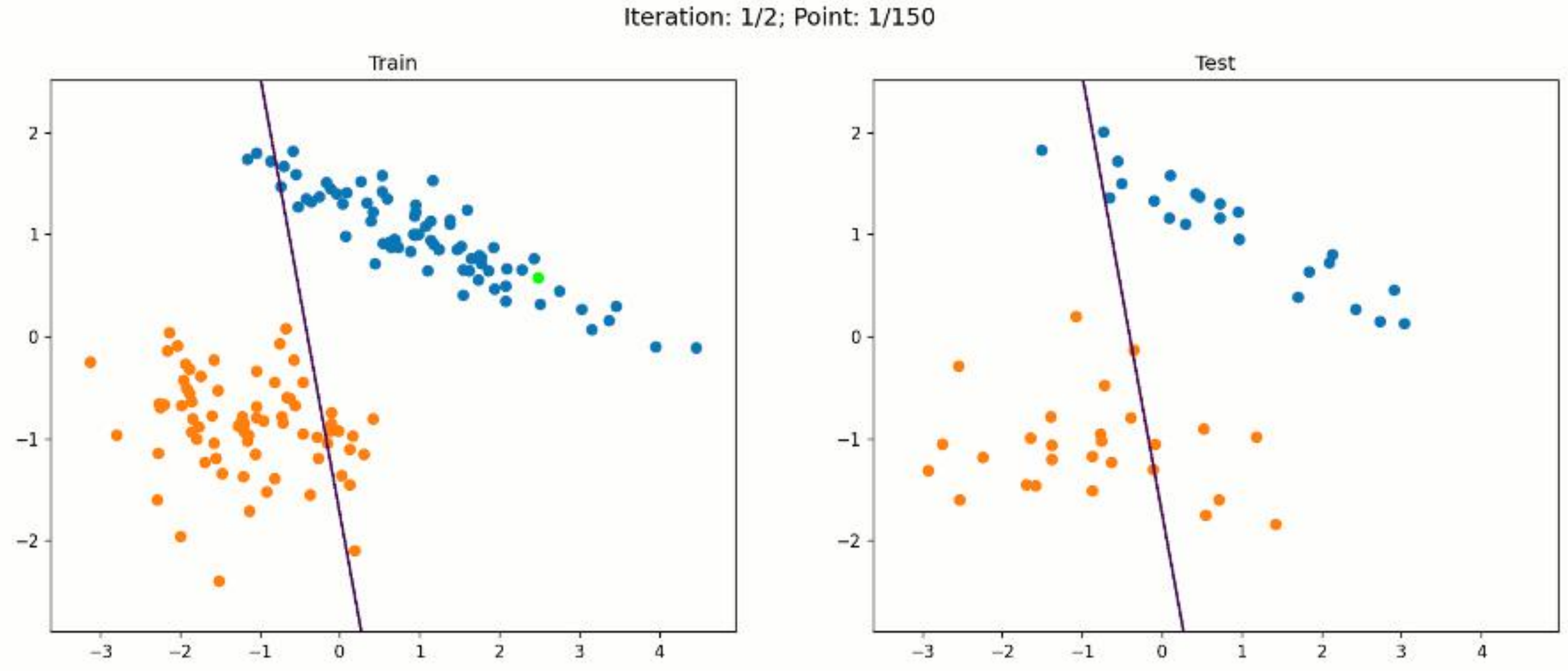
+

correction

=

learning

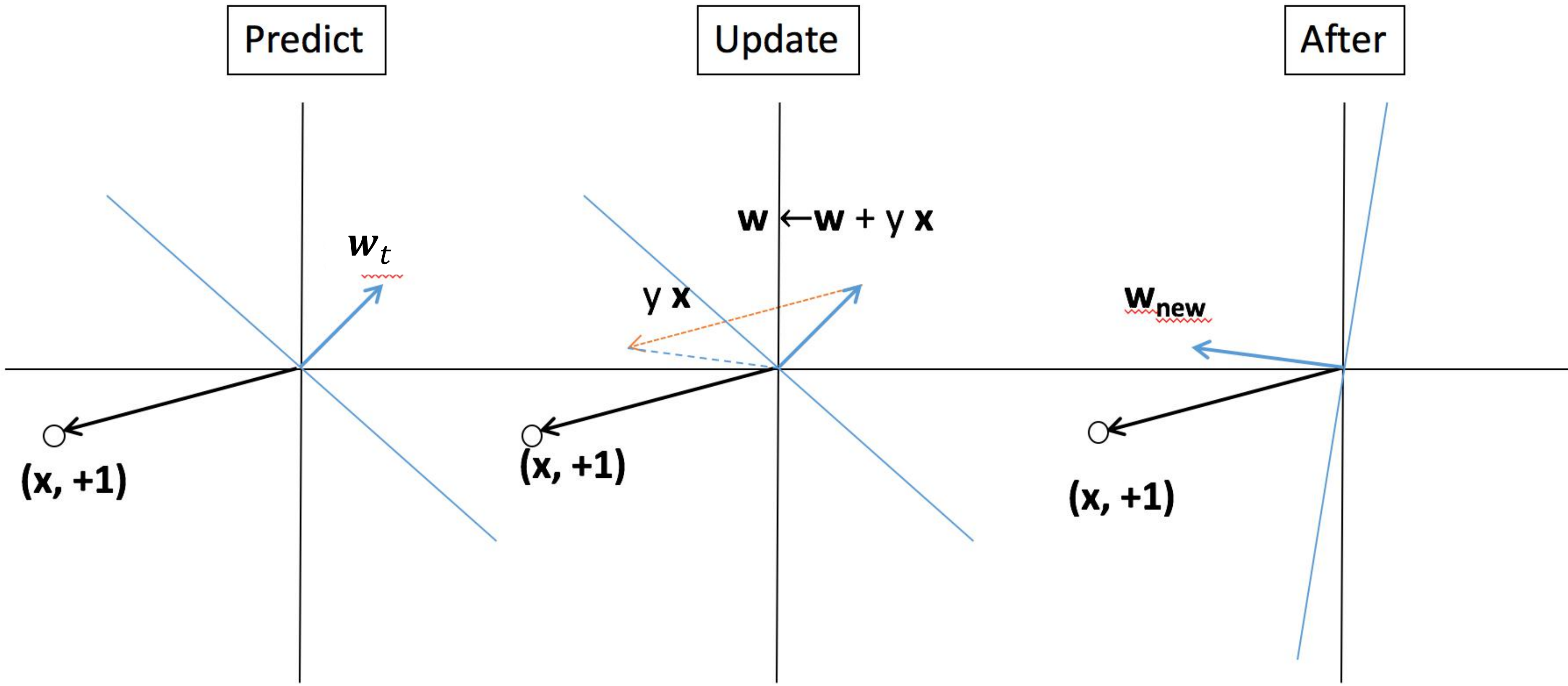
Perceptron in Action



https://towardsdatascience.com/perceptron-explanation-implementation-and-a-visual-example-3c8e76b4e2d1

Perceptron

- Learning by making mistakes



The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}$

Assume $y \in \{1, -1\}$

```
Initialize w ← 0 ∈ ℝ^n
For (x, y) in D:
        ŷ = sgn(wᵀx)                    (predict)
        if ŷ ≠ y,  w ← w + yx           (update)

Return w
```

$$\text{Initialize } \boldsymbol{w} \leftarrow \boldsymbol{0} \in \mathbb{R}^n$$

$$\text{For } (\boldsymbol{x}, y) \text{ in } \mathcal{D}:$$

$$\hat{y} = sgn(\boldsymbol{w}^\top \boldsymbol{x}) \quad \text{(predict)}$$

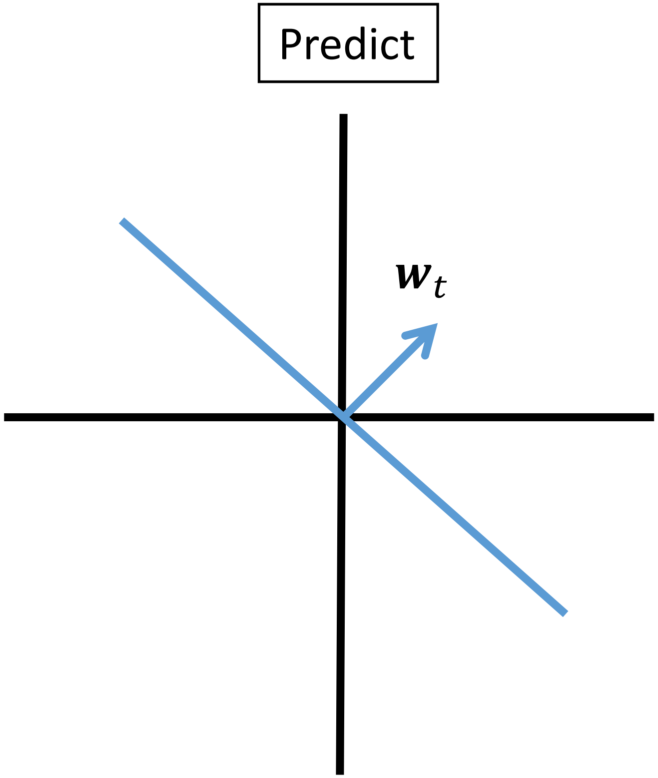
$$\text{if } \hat{y} \neq y, \ \boldsymbol{w} \leftarrow \boldsymbol{w} + y\boldsymbol{x} \quad \text{(update)}$$

$$\text{Return } \boldsymbol{w}$$

Prediction: $y^{\text{test}} \leftarrow sgn(\boldsymbol{w}^\top \boldsymbol{x}^{\text{test}})$

Geometry of the perceptron update

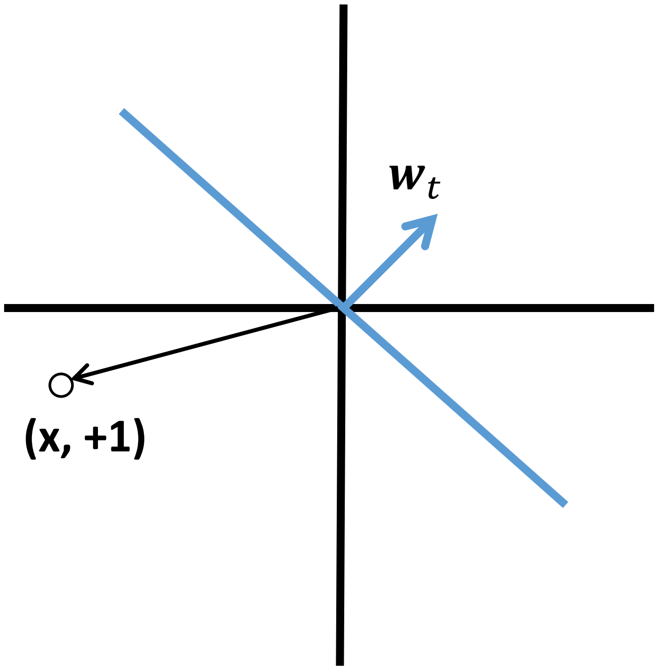
Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}_i$
Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}_i$



Geometry of the perceptron update

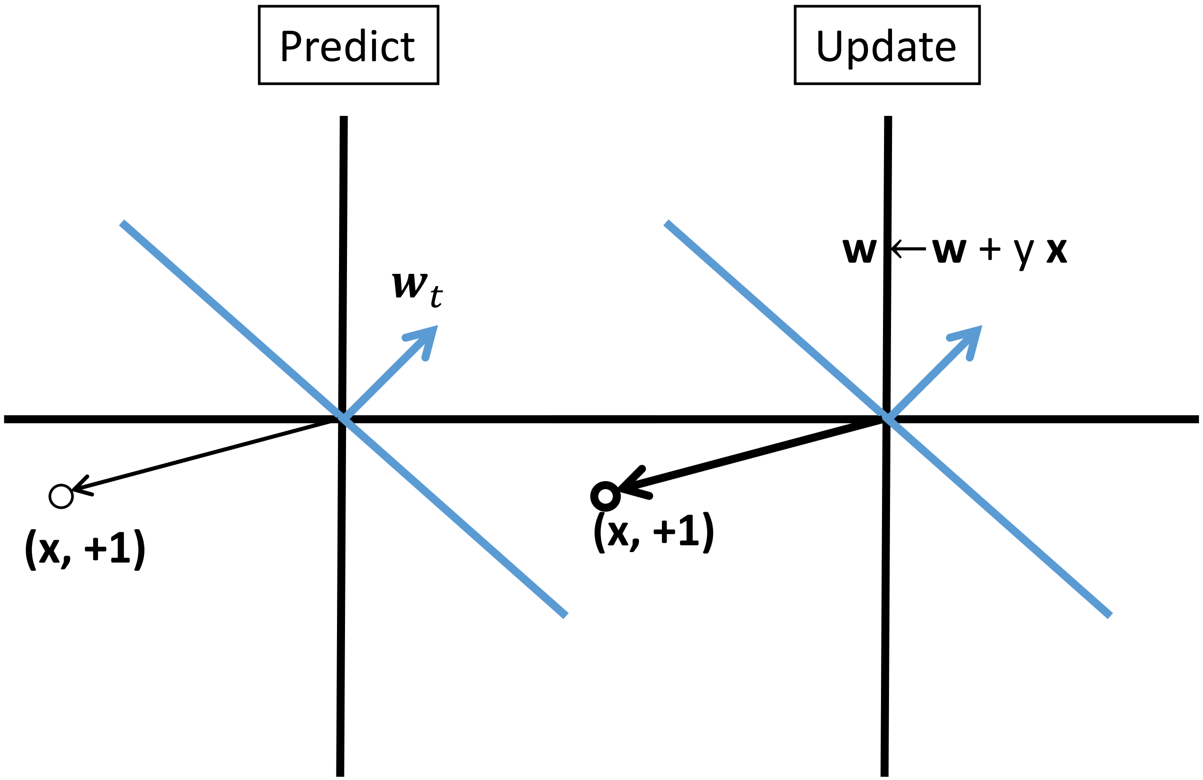
Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}_i$
Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}_i$

Predict



Geometry of the

Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}_i$
Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}_i$

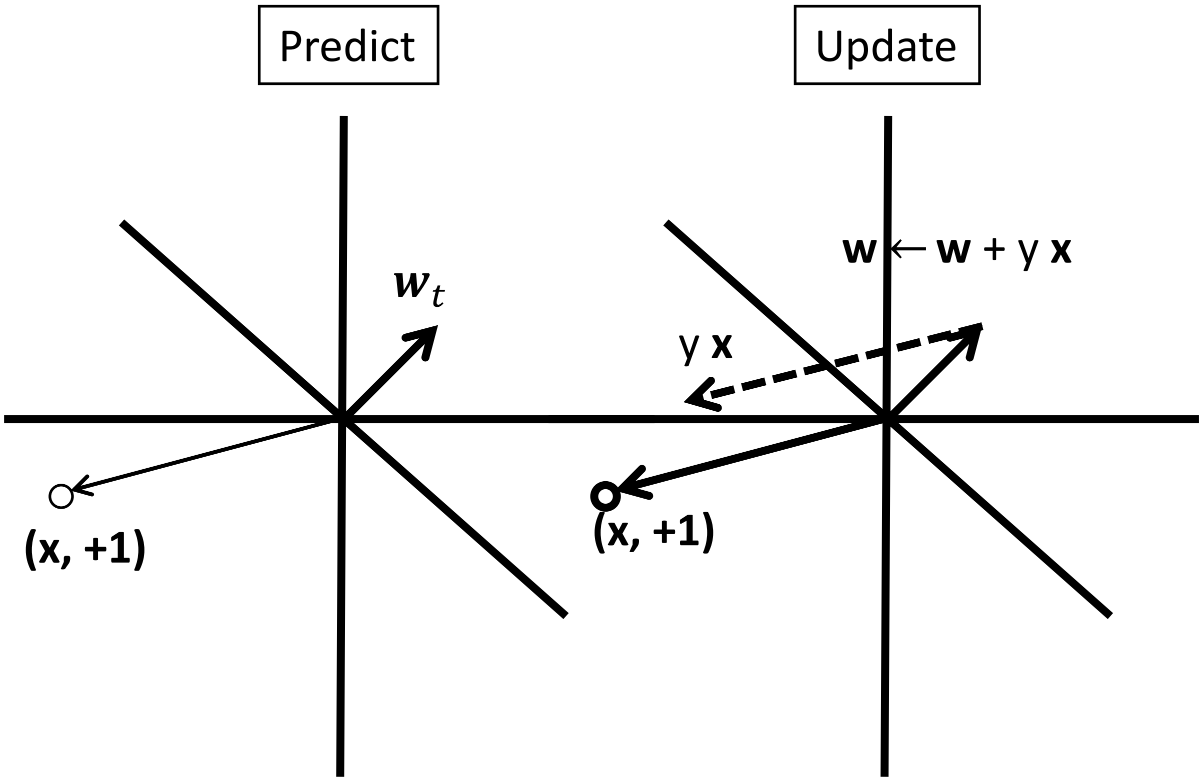


For a mistake on a positive example

Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}_i$

Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}_i$

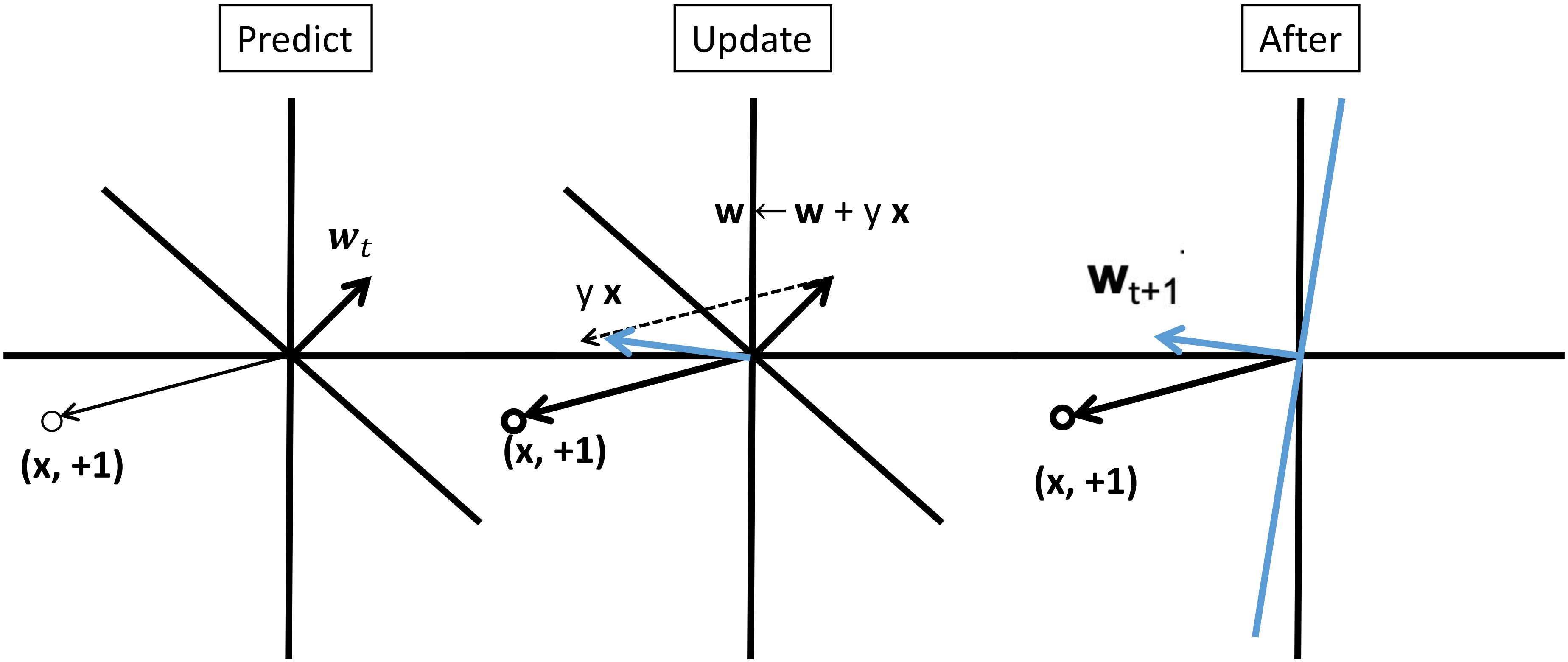
Geometry of the



For a mistake on a positive example

Geometry of the

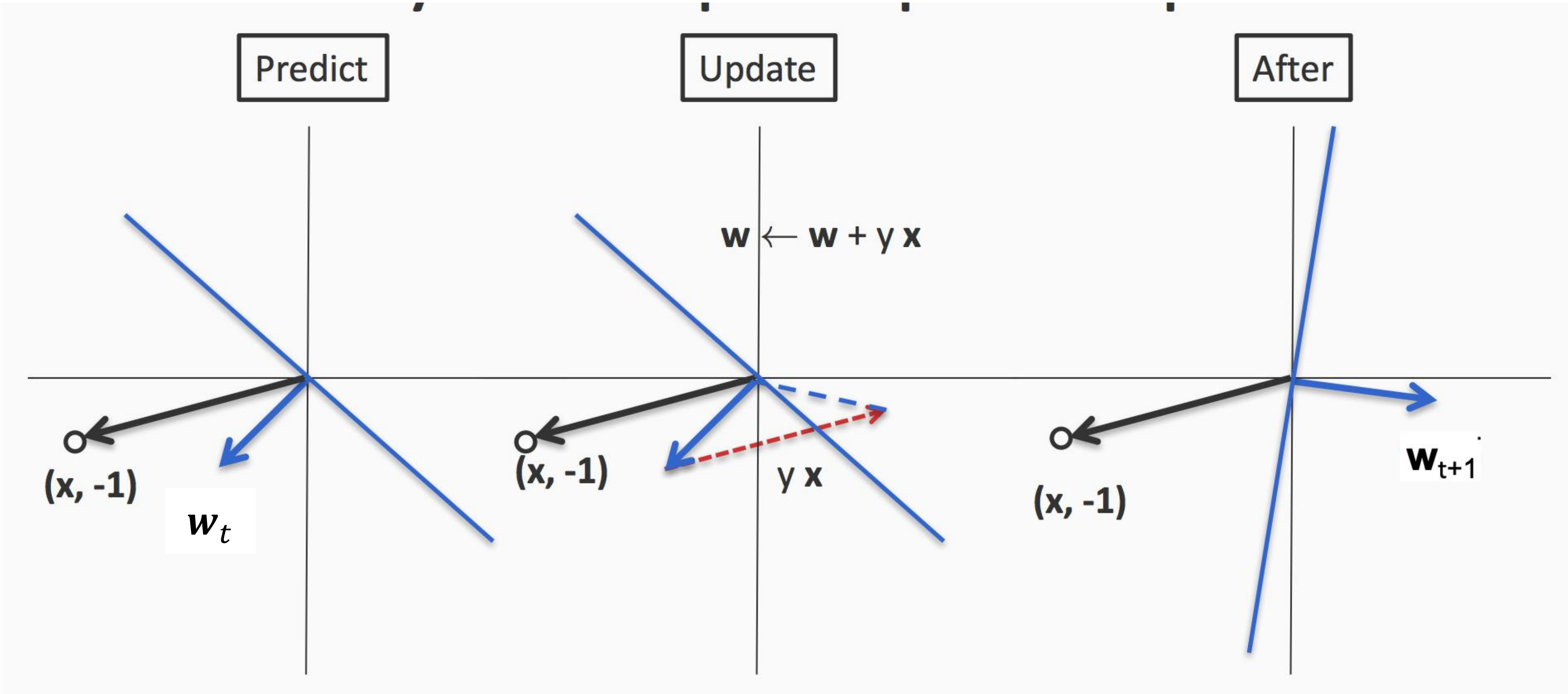
Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}_i$
Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}_i$



For a mistake on a positive example

Geometry of the

Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}_i$
Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}_i$



For a mistake on a negative example

Intuition behind the update

Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}_i$
Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}_i$

Suppose we have made a mistake on a positive example

That is, y = +1 and $\mathbf{w}_t^T\mathbf{x} \leq 0$

Call the new weight vector $\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{x}$

The new dot product will be

$$\mathbf{w}_{t+1}^T\mathbf{x} = (\mathbf{w}_t + \mathbf{x})^T\mathbf{x} = \mathbf{w}_t^T\mathbf{x} + \mathbf{x}^T\mathbf{x}$$

For a positive example, the Perceptron update will increase the score assigned to the same input

Similar reasoning for negative examples

The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}$

Assume $y \in \{1, -1\}$

1. Initialize $\boldsymbol{w} \leftarrow \boldsymbol{0} \in \mathbb{R}^n$
2. For $(\boldsymbol{x}, y)$ in $\mathcal{D}$:
3. $\qquad \hat{y} = sgn(\boldsymbol{w}^\top \boldsymbol{x})$ (predict)
4. $\qquad$ if $\hat{y} \neq y$, $\boldsymbol{w} \leftarrow \boldsymbol{w} + y\boldsymbol{x}$ (update)
5.
6. Return $\boldsymbol{w}$

Prediction: $y^{\text{test}} \leftarrow sgn(\boldsymbol{w}^\top \boldsymbol{x}^{\text{test}})$

The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}$

Initialize $\boldsymbol{w} \leftarrow \mathbf{0} \in \mathbb{R}^n$

For $(\boldsymbol{x}, y)$ in $\mathcal{D}$:

    if $y(\boldsymbol{w}^\top \boldsymbol{x}) \leq \mathbf{0}$

        $\boldsymbol{w} \leftarrow \boldsymbol{w} + y\boldsymbol{x}$

Return $\boldsymbol{w}$

Assume $y \in \{1, -1\}$

Prediction: $y^{\text{test}} \leftarrow sgn(\boldsymbol{w}^\top \boldsymbol{x}^{\text{test}})$

The Perceptron Algorithm [Rosenblatt 1958]

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}$

1. Initialize $\boldsymbol{w} \leftarrow \mathbf{0} \in \mathbb{R}^n$
2. For $(\boldsymbol{x}, y)$ in $\mathcal{D}$:
3. if $y(\boldsymbol{w}^\top \boldsymbol{x}) \leq \mathbf{0}$
4. $\boldsymbol{w} \leftarrow \boldsymbol{w} + y\boldsymbol{x}$
5.
6. Return $\boldsymbol{w}$

Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}_i$
Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}_i$

Prediction: $y^{\text{test}} \leftarrow sgn(\boldsymbol{w}^\top \boldsymbol{x}^{\text{test}})$

The Perceptron Algorithm (batch)

Given a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}$

1. Initialize $\boldsymbol{w} \leftarrow \mathbf{0} \in \mathbb{R}^n$
2. For epoch $1 \ldots T$:
3. For $(\boldsymbol{x}, y)$ in $\mathcal{D}$:
4. if $y(\boldsymbol{w}^\top \boldsymbol{x}) \leq \mathbf{0}$
5. $\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta y \boldsymbol{x}$
6. Return $\boldsymbol{w}$

T is a hyper-parameter to the algorithm

η is a hyper-parameter to the algorithm

Prediction: $y^{\text{test}} \leftarrow sgn(\boldsymbol{w}^\top \boldsymbol{x}^{\text{test}})$

Check point: What you need to know

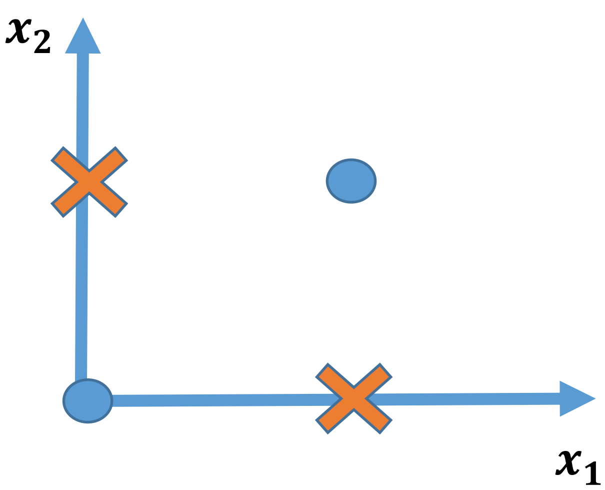
- The Perceptron algorithm

- The geometry of the update

Perceptron Learnability

- Perceptron cannot learn what it cannot represent
  - Only linearly separable functions
    -- Minsky and Papert (1969)
  - E.g., Parity functions can't be learned (XOR)

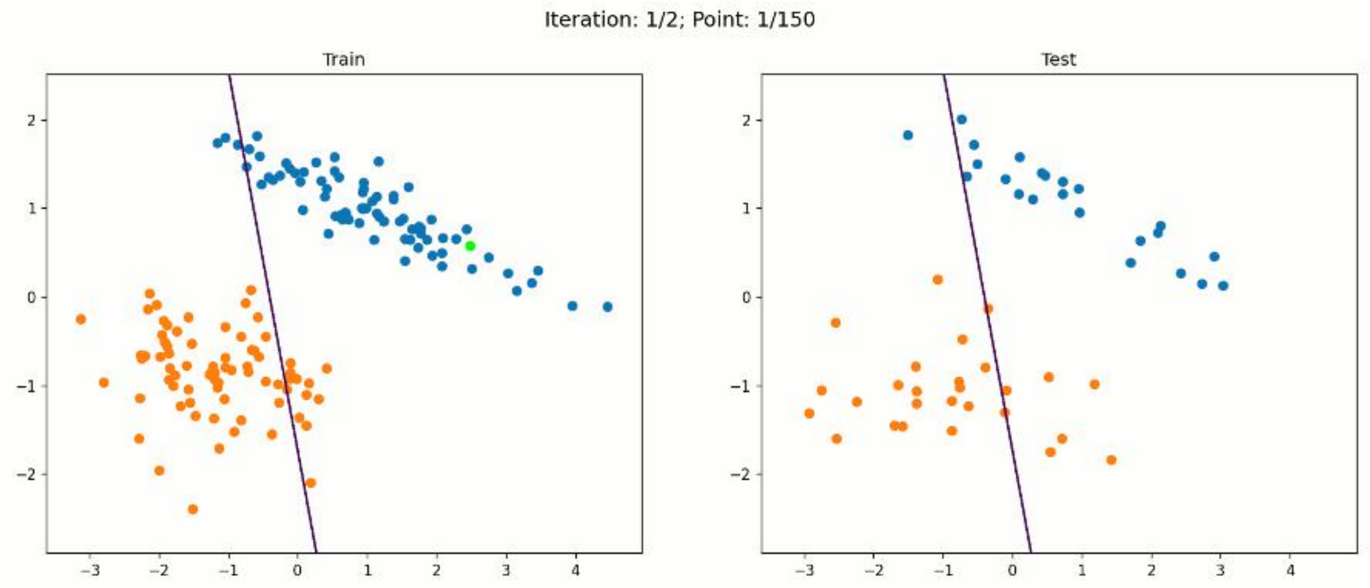
x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	0



Convergence

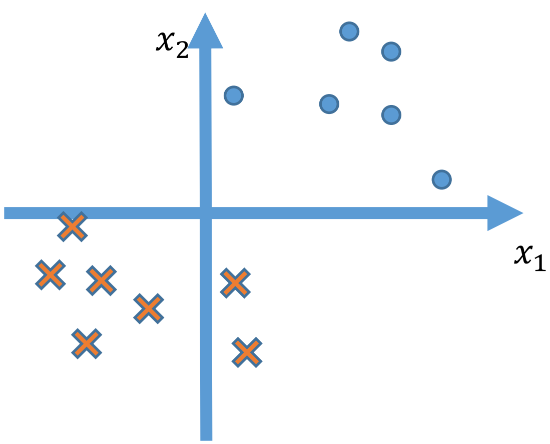
Convergence theorem

- If there exists a model that is consistent with the data (i.e. the data is linearly separable), the perceptron algorithm will converge.

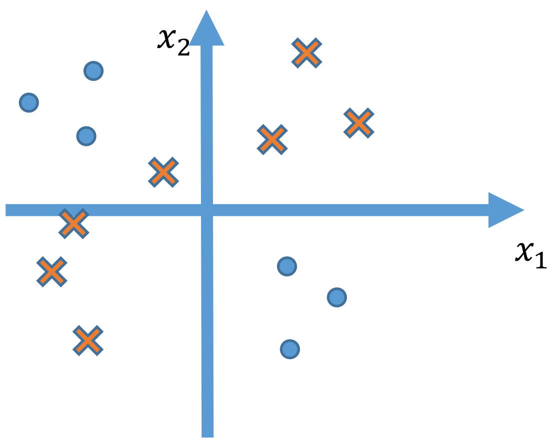


Convergence theorem

- If there exist a set of weights that are consistent with the data (i.e. the data is linearly separable), the perceptron algorithm will converge.
- Note: this is the condition of the data we may not know what the hyperplane is



Linearly separable data

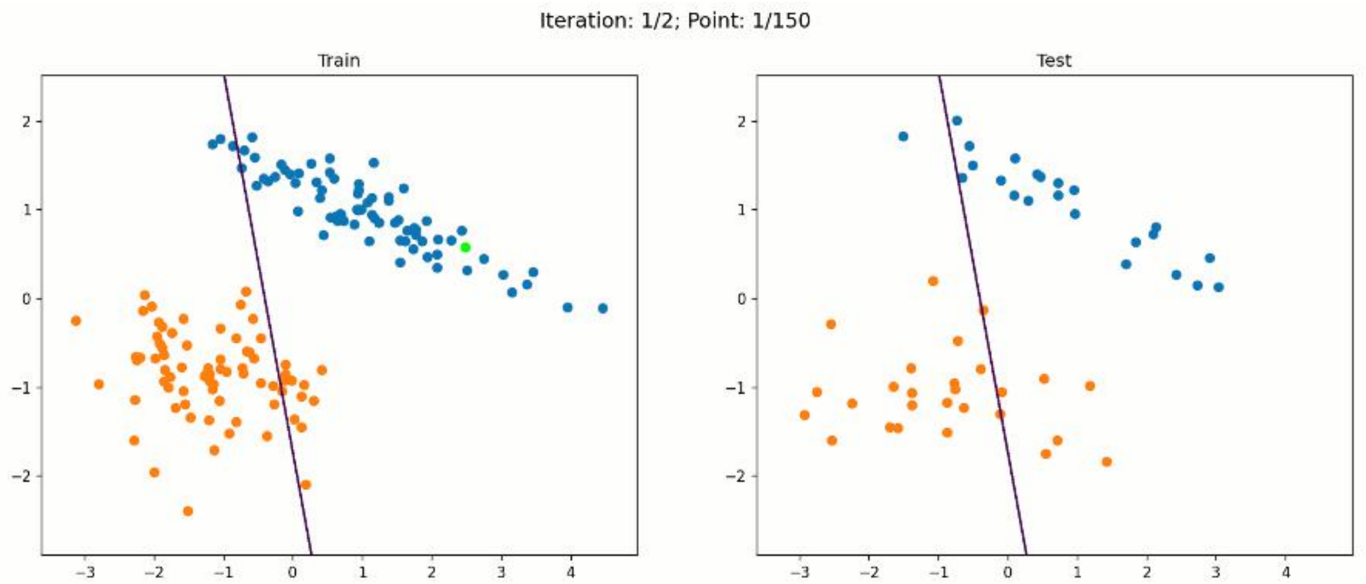


Linearly non-separable data

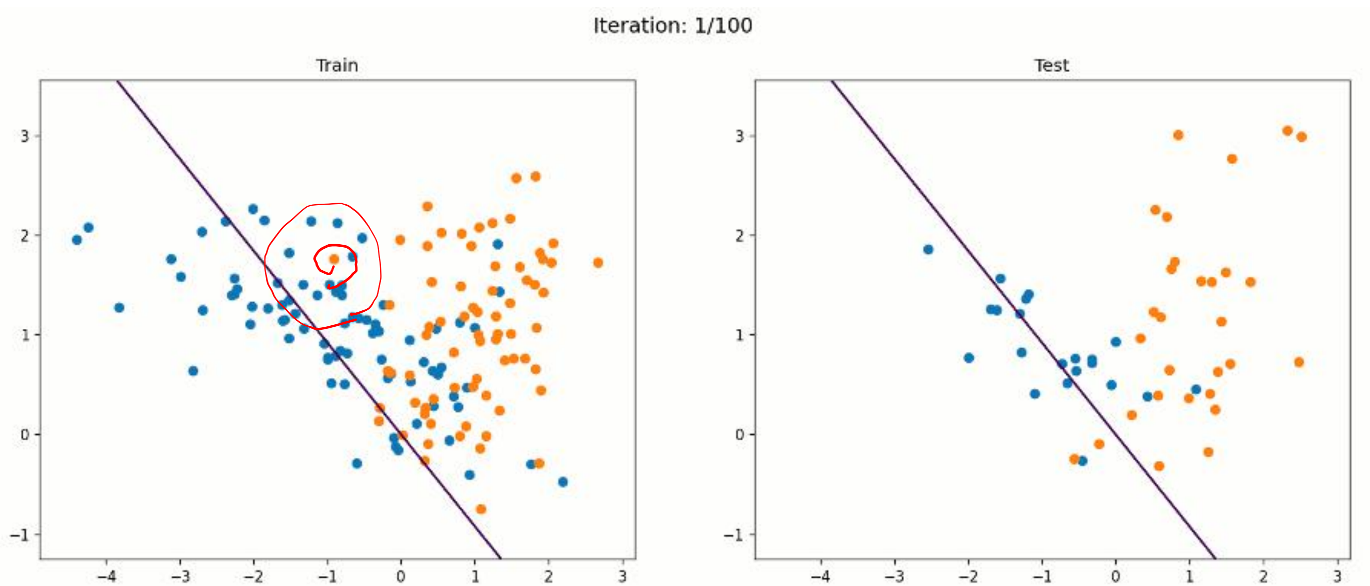
Convergence theorem

- If there exist a set of weights that are consistent with the data (i.e. the data is linearly separable), the perceptron algorithm will converge
  - The update stops after making a finite number of mistakes.
  - The convergence rate depends on the difficulty of the problem (explain later)

- If the training data is *not* linearly separable, then the learning algorithm will eventually repeat the same set of weights and enter an infinite loop



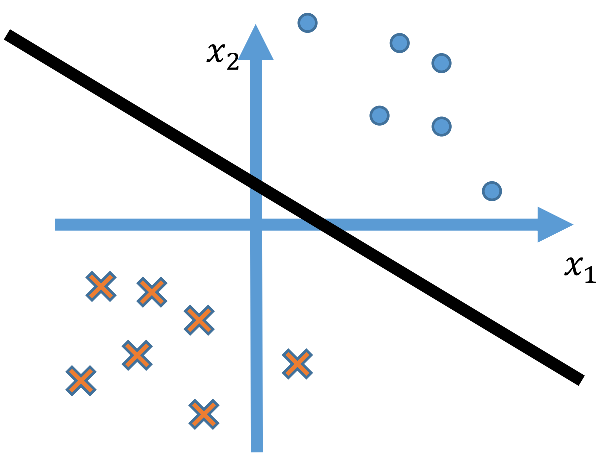
Linearly Separatable

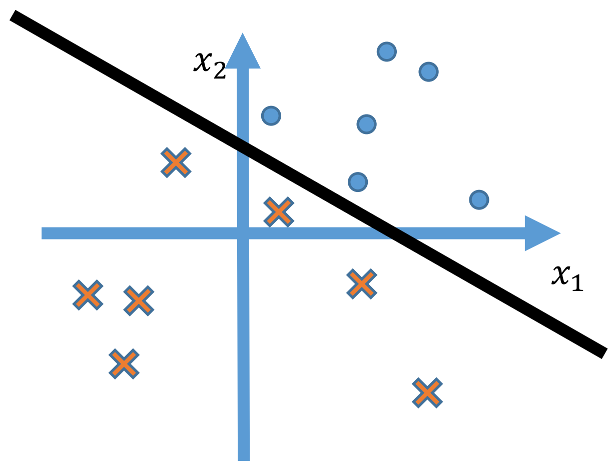


Not Linearly Separatable

Convergence theorem

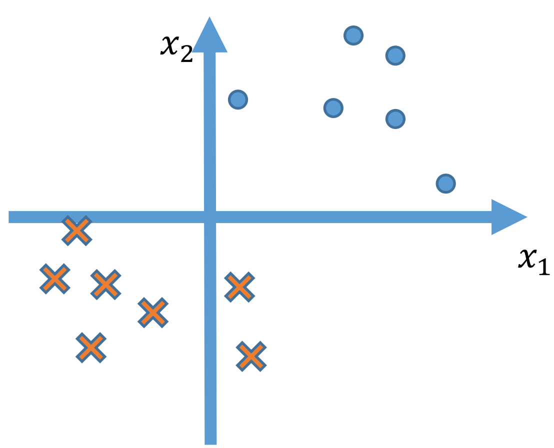
- If there exist a set of weights that are consistent with the data (i.e. the data is linearly separable), the perceptron algorithm will converge
  - The update stops after making a finite number of mistakes.
  - The convergence rate depends on the difficulty of the problem



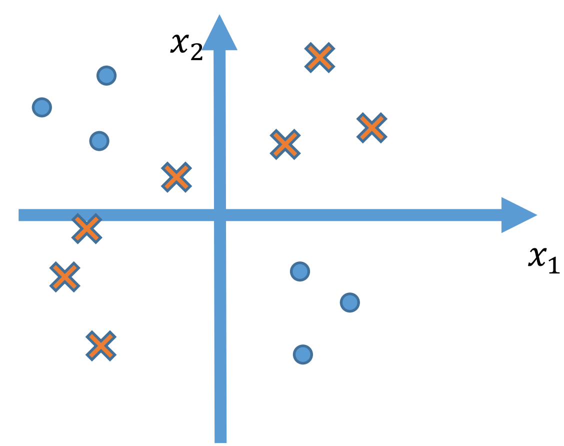


Convergence theorem

- If there exist a set of weights that are consistent with the data (i.e. the data is linearly separable), the perceptron algorithm will converge.
- Note: this is the condition of the data
  we may not know what the hyperplane is



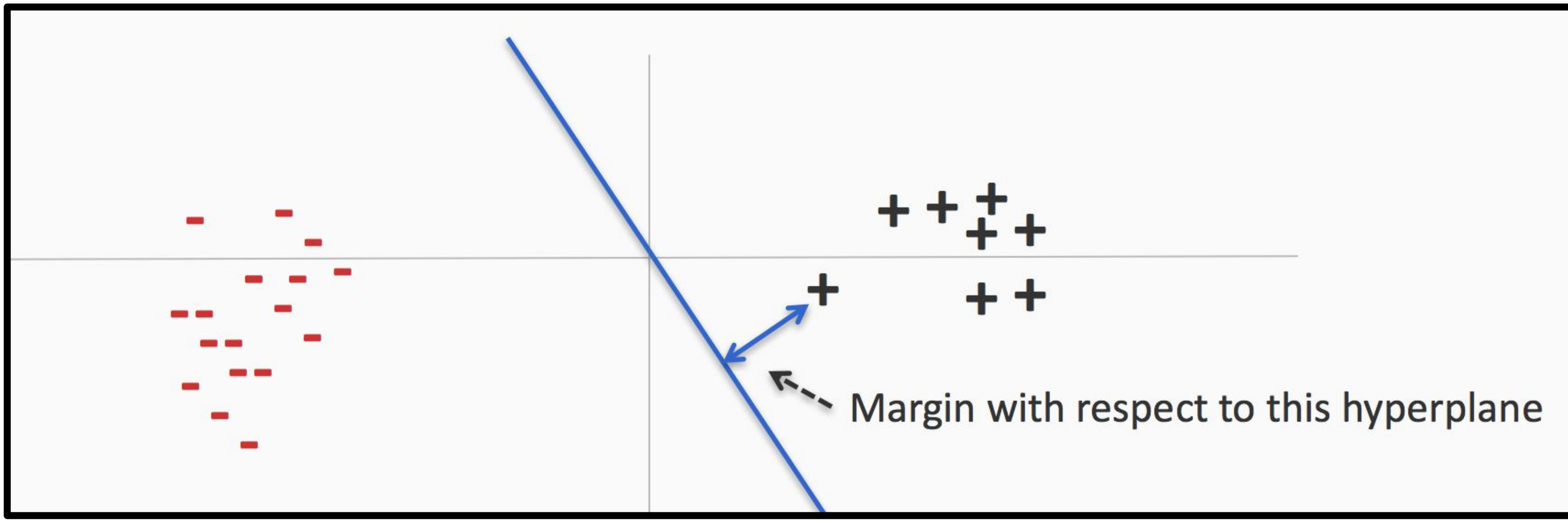
Linearly separable data



Linearly non-separable data

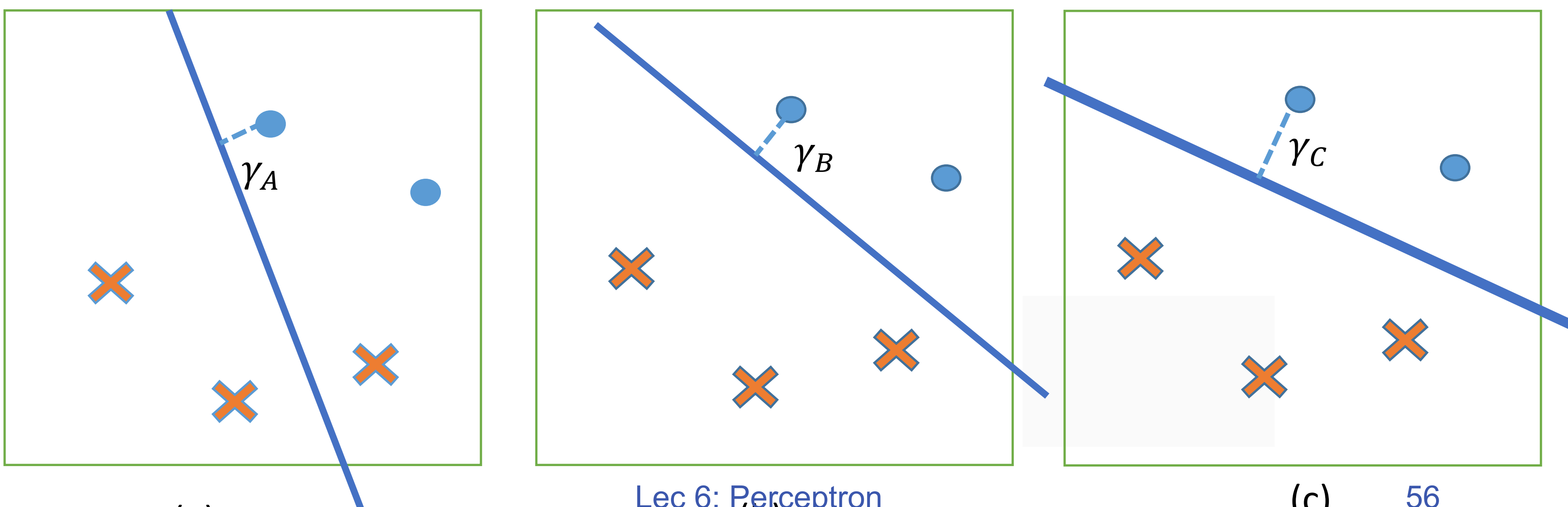
Margin

If a hyperplane can separate the data. The margin of a hyperplane for a dataset is the distance between the hyperplane and the data point nearest to it.



Margin

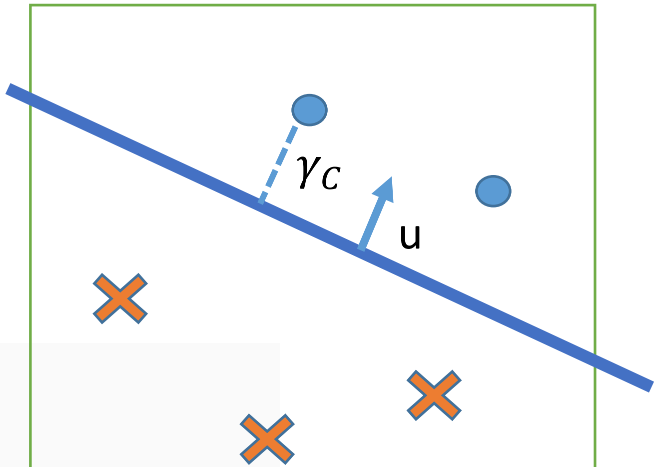
- If a hyperplane can separate the data. The margin of a hyperplane for a dataset is the distance between the hyperplane and the data point nearest to it.
- The margin of a data set (γ) is the maximum margin possible for that dataset using any weight vector. Which γ is the margin of data?



(a) (b) (c)

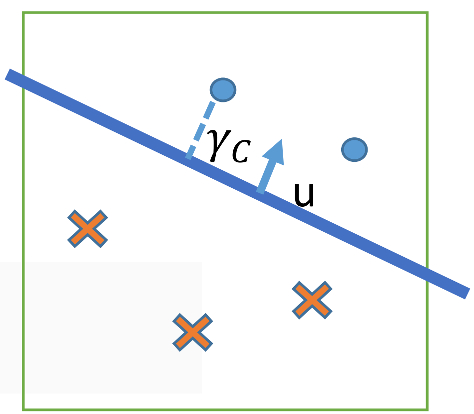
Margin

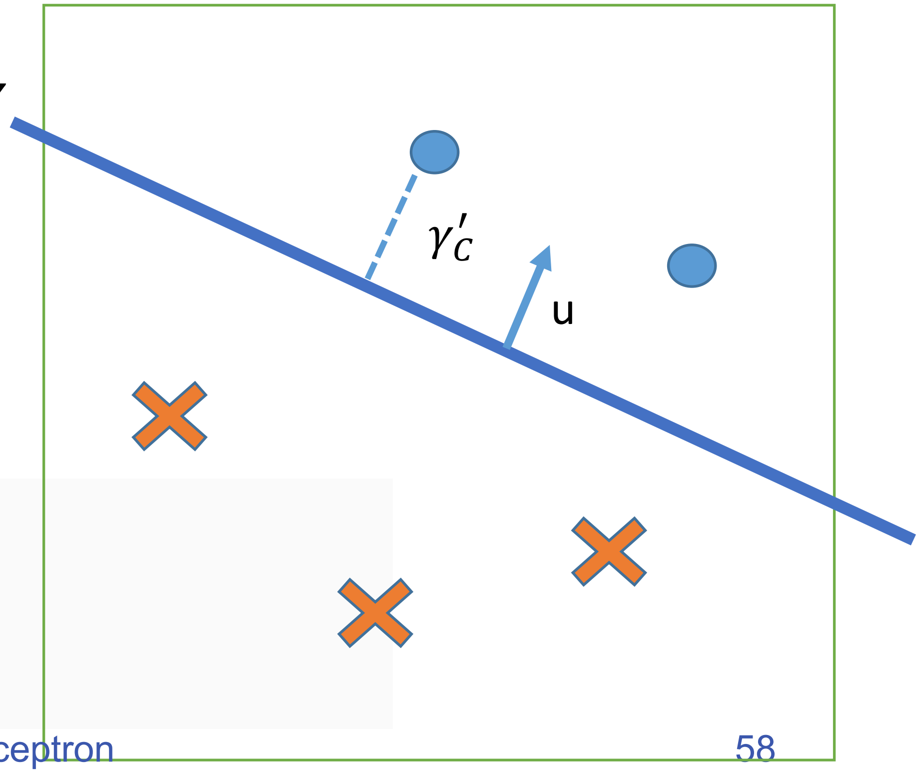
- The margin of a data set (γ) is the maximum margin possible for that dataset using any weight vector.
- Let $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_m, y_m)\}$ be a set of training data, if there exists a unit vector **u** such that $y_i\,(\mathbf{u}^T\,\mathbf{x}_i) \geq \gamma$, for all data points (x_i, y_i) in the training set
- γ is the margin



Margin is not scale invariance

- Let $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_m, y_m)\}$ be a set of training data, if there exists a unit vector $\mathbf{u}$ such that
  $y_i\,(\mathbf{u}^T\,\mathbf{x}_i) \geq \gamma$, for all data points $(\boldsymbol{x_i}, y_i)$ in the training set
- If we double the size of every input $\mathbf{x_i}$, the margin γ is also double, because
  $y_i\,(\mathbf{u}^T\,\mathbf{x}_i) \geq \gamma \Rightarrow y_i\,(\mathbf{u}^T\,\mathbf{2x}_i) \geq 2\gamma$





Margin is not scale invariance

- The "difficulty" of the problem, can be captured by $\frac{R}{\gamma}$, $||\mathbf{x}_i|| \leq R, \forall i$
- Perceptron makes $\leq \left(\frac{R}{\gamma}\right)^2$ mistakes if data has margin γ and the size of all the input vectors $||\mathbf{x}_i|| \leq R, \forall i$

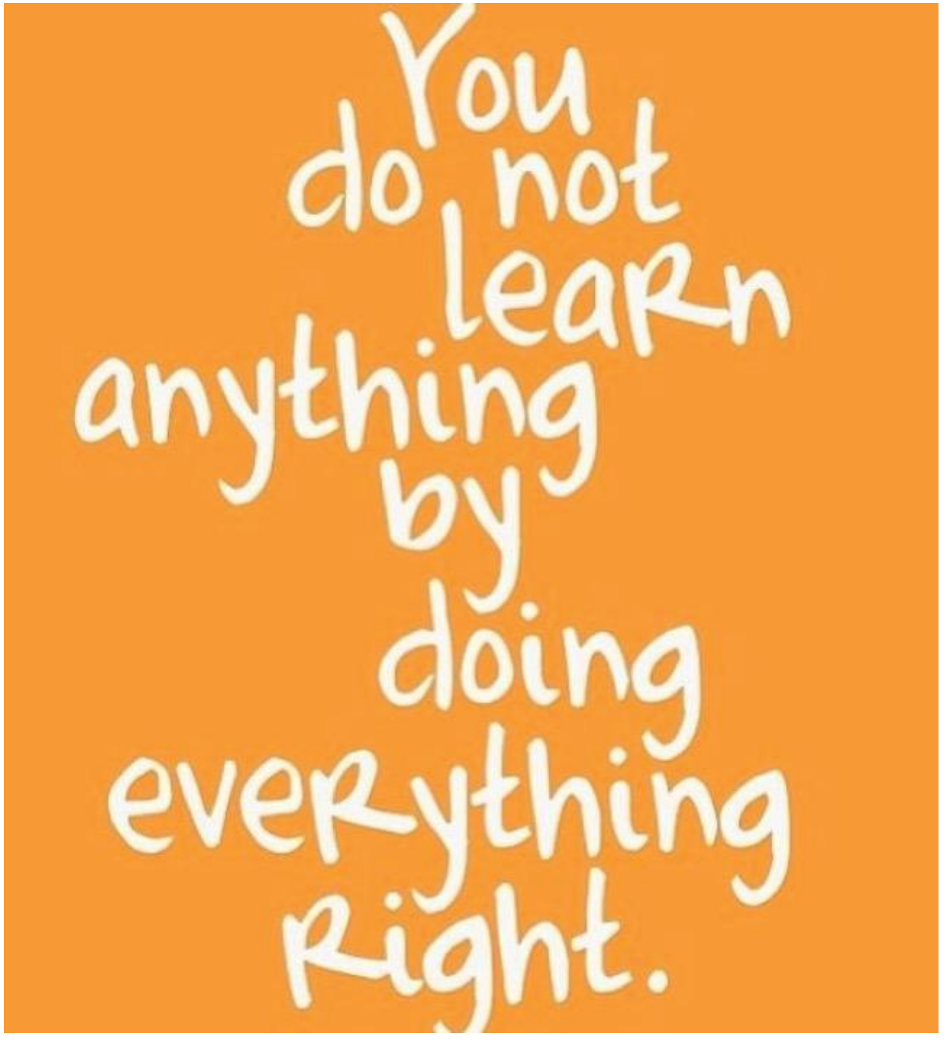
Mistake Bound Theorem [Novikoff 1962, Block 1962]

Let $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_m, y_m)\}$ be a sequence of training examples such that for all i, the feature vector $\mathbf{x}_i \in R^n$, $||\mathbf{x}_i|| \leq R$ and the label $y_i \in \{-1, +1\}$.

Suppose there exists a unit vector $\mathbf{u} \in R^n$ (i.e $||\mathbf{u}|| = 1$) such that for some $\gamma > 0$ we have $y_i (\mathbf{u}^T \mathbf{x}_i) \geq \gamma$.

Then, the perceptron algorithm will make at most $(R/\gamma)^2$ mistakes on the training sequence.

NOTE!! # mistakes!= # seen data points



Beyond the separable case

- Good news
  - Perceptron makes no assumption about data distribution, could be even adversarial
- Bad news:
  - Real-world data are often not linearly separable

What you learned today

- Linear models
- The Perceptron Algorithm
- Perceptron Mistake Bound

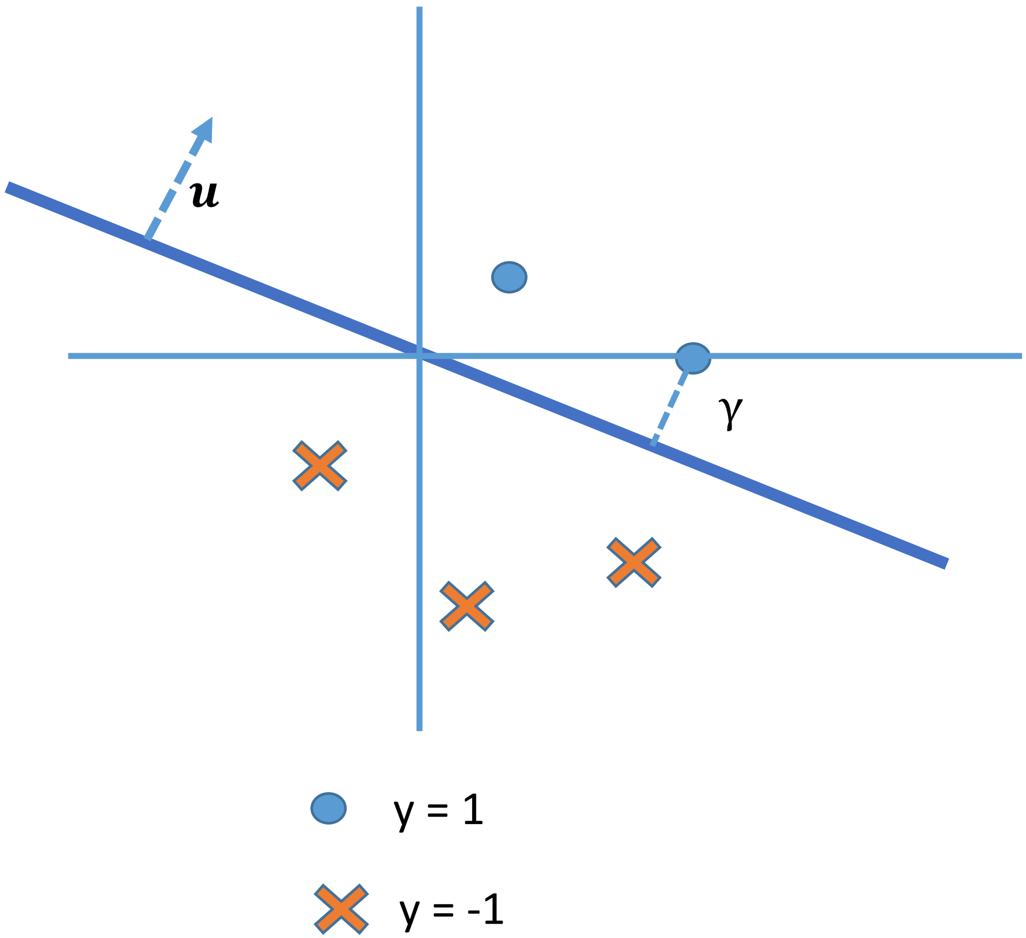
Not cover in the lecture and the exams

Appendix: Proof of mistake bound

See details at https://arxiv.org/pdf/1305.0208.pdf

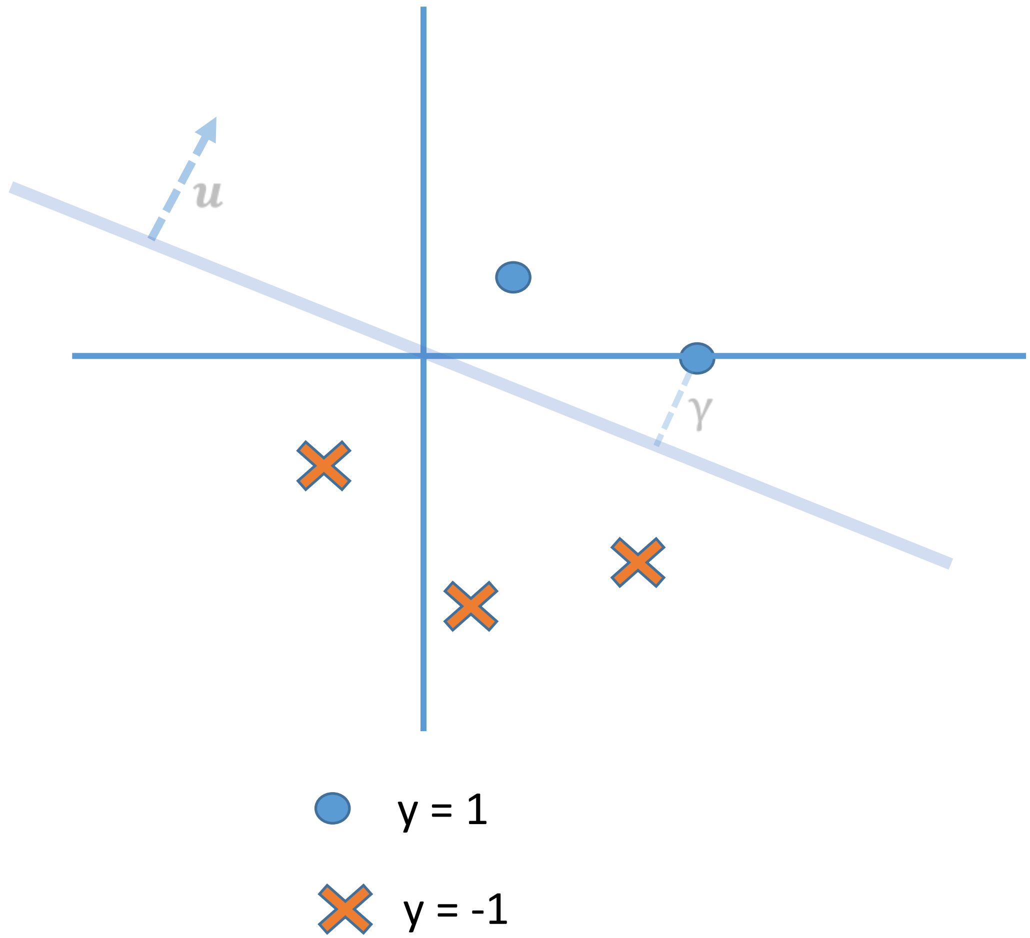
Intuition

$||\boldsymbol{u}|| = 1 \quad y_i \, (\mathbf{u}^T \mathbf{x}_i) \geq \gamma$



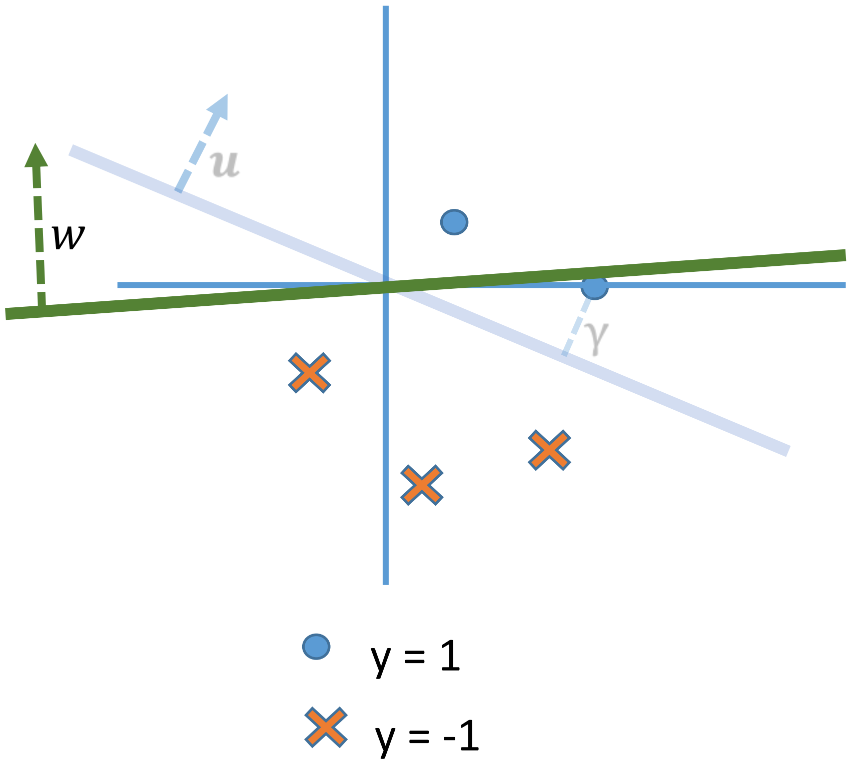
Intuition

$||\boldsymbol{u}|| = 1 \quad y_i\ (\mathbf{u}^\top \mathbf{x}_i) \geq \gamma$



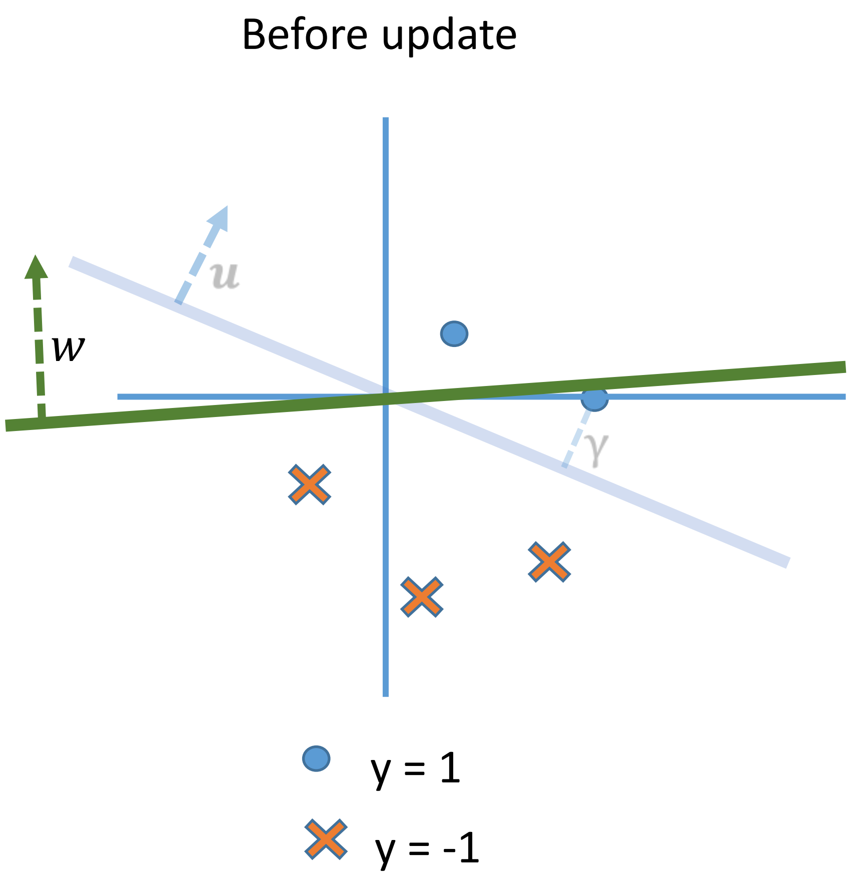
Intuition

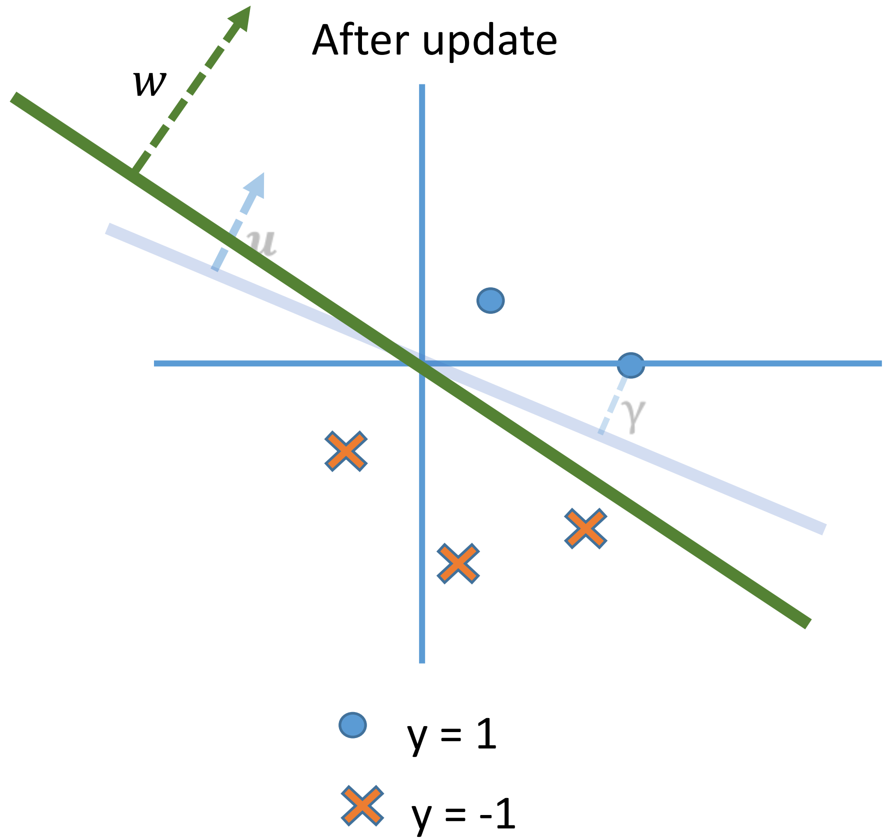
$||\boldsymbol{u}|| = 1 \quad y_i\,(\mathbf{u}^{\top}\,\mathbf{x}_i) \geq \gamma$



Intuition

$||\boldsymbol{u}|| = 1 \quad y_i\,(\mathbf{u}^T\,\mathbf{x}_i) \geq \gamma$

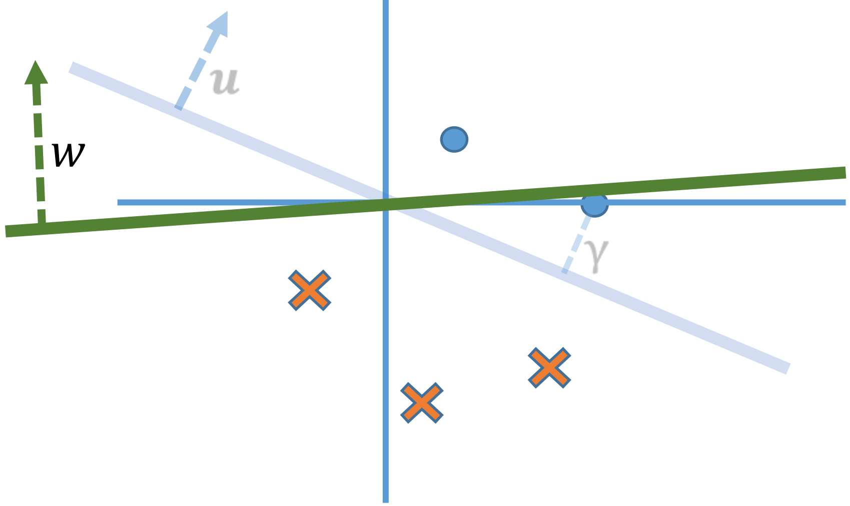




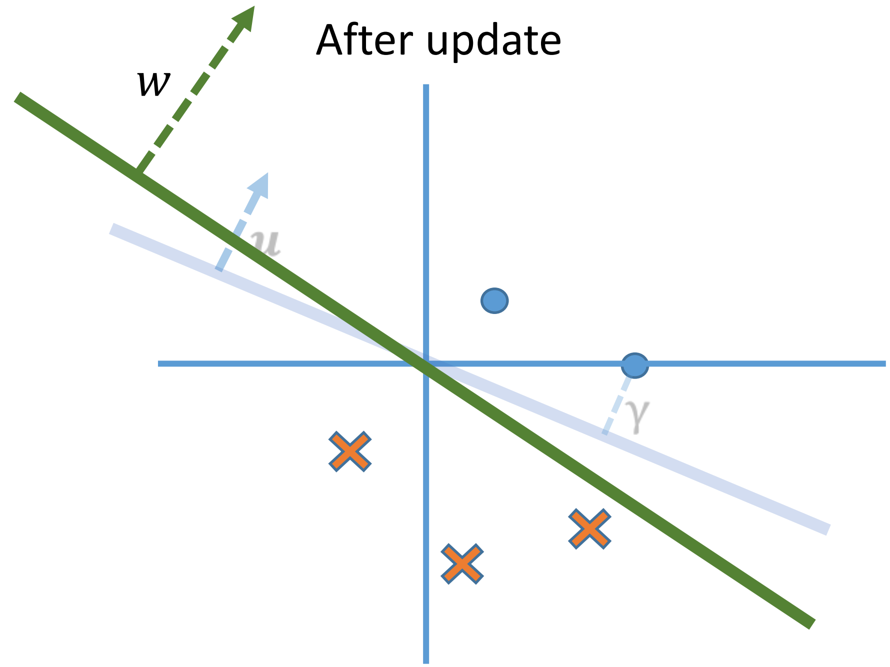
Intuition

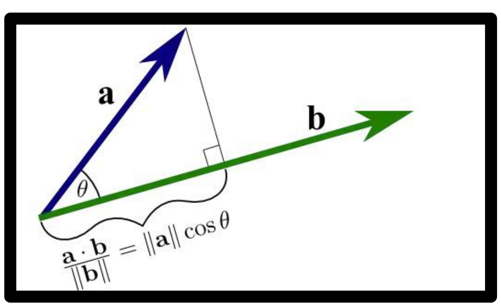
$||\boldsymbol{u}|| = 1 \quad y_i\,(\mathbf{u}^{\mathrm{T}}\,\mathbf{x}_i) \geq \gamma$

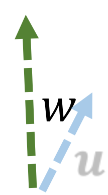
Before update



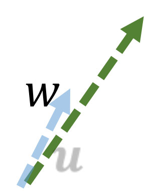
After update







Before update



After update

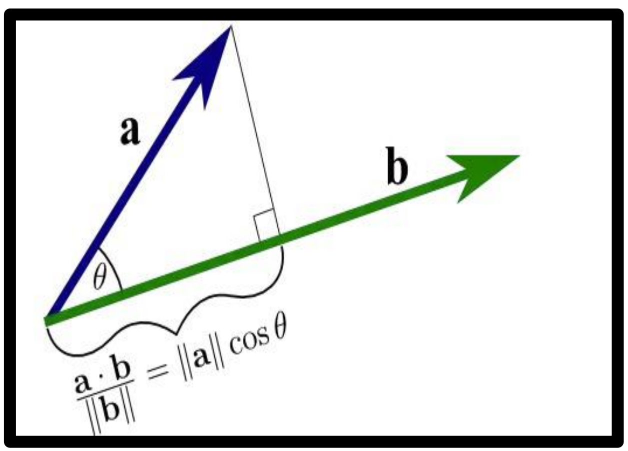
Intuition

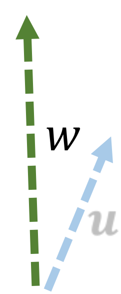
1. After update, $\mathbf{u}^\mathsf{T}\mathbf{w}_{t+1}$ is larger than $\mathbf{u}^\mathsf{T}\mathbf{w}_t$

   After t mistakes, $\mathbf{u}^\mathsf{T}\mathbf{w}_t \geq t\,\gamma$

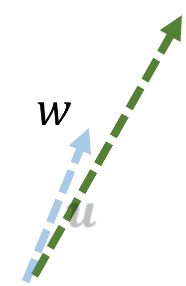
2. The size of $||w_{t+1}||$ may increase, but not much

   After t mistakes, $||\mathbf{w}_t||^2 \leq tR^2$





Before update



After update

- Receive an input ($\mathbf{x}_i$, y_i)
- if $\mathrm{sgn}(\mathbf{w}_t^T \mathbf{x}_i) \neq y_i$:
  Update $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_i \mathbf{x}_i$

Proof (preliminaries)

The setting

- Initial weight vector **w** is all zeros
- All training examples are contained in a ball of size R
  - $||\mathbf{x}_i|| \leq R$
- The training data is separable by margin γ using a unit vector **u**
  - $y_i (\mathbf{u}^T \mathbf{x}_i) \geq \gamma$

Proof (1/3)

- Receive an input ($\mathbf{x}_i$, y_i)
- if $\text{sgn}(\mathbf{w}_t^T \mathbf{x}_i) \neq y_i$:
  Update $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_i \, \mathbf{x}_i$

1. Claim: After t mistakes, $\mathbf{u}^T\mathbf{w}_t \geq t\,\gamma$

$$\begin{aligned} \mathbf{u}^T\mathbf{w}_{t+1} &= \mathbf{u}^T\mathbf{w}_t + y_i\mathbf{u}^T\mathbf{x}_i \\ &\geq \mathbf{u}^T\mathbf{w}_t + \gamma \end{aligned}$$

Because the data is separable by a margin γ

$y_i\,(\mathbf{u}^T\,\mathbf{x}_i) \geq \gamma$

Because $\mathbf{w}_0 = \mathbf{0}$ (i.e $\mathbf{u}^T\mathbf{w}_0 = 0$), straightforward induction gives us

$\mathbf{u}^T\mathbf{w}_t \geq t\,\gamma$

Intuition: the inner product between the underlying true model and the current model is non-decreasing after each update 1). The directions of u, w align or 2). $||w||$ increases

Proof (2/3)

- Receive an input ($\mathbf{x}_i$, y_i)
- if $\text{sgn}(\mathbf{w}_t^T\mathbf{x}_i) \neq y_i$:
  Update $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_i\, \mathbf{x}_i$

2. Claim: After t mistakes, $||\mathbf{w}_t||^2 \leq tR^2$

$$
\begin{aligned}
||\mathbf{w}_{t+1}||^2 &= ||\mathbf{w}_t + y_i\mathbf{x}_i||^2 \\
&= ||\mathbf{w}_t||^2 + 2y_i(\mathbf{w}_t^T\mathbf{x}_i) + ||\mathbf{x}_i||^2
\end{aligned}
$$

- Receive an input ($\mathbf{x}_i$, y_i)
- if $\text{sgn}(\mathbf{w}_t^T\mathbf{x}_i) \neq y_i$:
  Update $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_i\, \mathbf{x}_i$

Proof (2/3)

2. Claim: After t mistakes, $||\mathbf{w}_t||^2 \leq tR^2$

$$
\begin{aligned}
\left|\left|\mathbf{w}_{t+1}\right|\right|^2 &= \left|\left|\mathbf{w}_t + y_i\mathbf{x}_i\right|\right|^2 \\
&= \left|\left|\mathbf{w}_t\right|\right|^2 + 2y_i(\mathbf{w}_t^T\mathbf{x}_i) + \left|\left|\mathbf{x}_i\right|\right|^2
\end{aligned}
$$

The weight is updated only when there is a mistake. That is when $y_i\, \mathbf{w}_t^T\, \mathbf{x}_i < 0$.

$||\mathbf{x}_i|| \cdot R$, by definition of R

Proof (2/3)

- Receive an input ($\mathbf{x}_i$, y_i)
- if $\text{sgn}(\mathbf{w}_t^T\mathbf{x}_i) \neq y_i$:
  Update $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_i\, \mathbf{x}_i$

2. Claim: After t mistakes, $||\mathbf{w}_t||^2 \leq tR^2$

$$
\begin{aligned}
\left\|\mathbf{w}_{t+1}\right\|^2 &= \left\|\mathbf{w}_t + y_i \mathbf{x}_i\right\|^2 \\
&= \left\|\mathbf{w}_t\right\|^2 + 2y_i(\mathbf{w}_t^T \mathbf{x}_i) + \left\|\mathbf{x}_i\right\|^2 \\
&\leq \left\|\mathbf{w}_t\right\|^2 + R^2
\end{aligned}
$$

Because $\mathbf{w}_0 = \mathbf{0}$ (i.e $\mathbf{u}^T\mathbf{w}_0 = 0$),
straightforward induction gives us $||\mathbf{w}_t||^2 \leq tR^2$

Proof (2/3)

- Receive an input ($\mathbf{x}_i$, y_i)
- if $\text{sgn}(\mathbf{w}_t^T\mathbf{x}_i) \neq y_i$:
  Update $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_i\,\mathbf{x}_i$

2. Claim: After t mistakes, $\|\mathbf{w}_t\|^2 \leq tR^2$

Intuition: $||w||$ is bounded

$$
\begin{aligned}
\|\mathbf{w}_{t+1}\| &= \|\mathbf{w}_t + y_i\mathbf{x}_i\|^2 \\
&= \|\mathbf{w}_t\|^2 + 2y_i(\mathbf{w}_t^T\mathbf{x}_i) + \|\mathbf{x}_i\|^2 \\
&\leq \|\mathbf{w}_t\|^2 + R^2
\end{aligned}
$$

Because $\mathbf{w}_0 = \mathbf{0}$ (i.e $\mathbf{u}^T\mathbf{w}_0 = 0$),
straightforward induction gives us $\|\mathbf{w}_t\|^2 \leq tR^2$

Proof (3/3)

What we know:

1. After t mistakes, $\mathbf{u}^T\mathbf{w}_t \geq t\gamma$
2. After t mistakes, $||\mathbf{w}_t||^2 \leq tR^2$

Intuition 1: the inner product between the underlying true model and the current model is non-decreasing after each update 1). The directions of u, w align or 2). $||w||$ increases

Intuition 2: $||w||$ is bounded

Proof (3/3)

What we know:

1. After t mistakes, $\mathbf{u}^\mathsf{T}\mathbf{w}_t \geq t\gamma$
2. After t mistakes, $\|\mathbf{w}_t\|^2 \leq tR^2$

$$R\sqrt{t} \geq \|\mathbf{w}_t\|$$

From (2)

Proof (3/3)

What we know:

1. After t mistakes, $\mathbf{u}^T\mathbf{w}_t \geq t\gamma$
2. After t mistakes, $\|\mathbf{w}_t\|^2 \leq tR^2$

$$R\sqrt{t} \geq \|\mathbf{w}_t\| \geq \mathbf{u}^T\mathbf{w}_t$$

From (2)

$\mathbf{u}^T \mathbf{w}_t = \|\mathbf{u}\| \, \|\mathbf{w}_t\| \cos($<angle between them>$)$

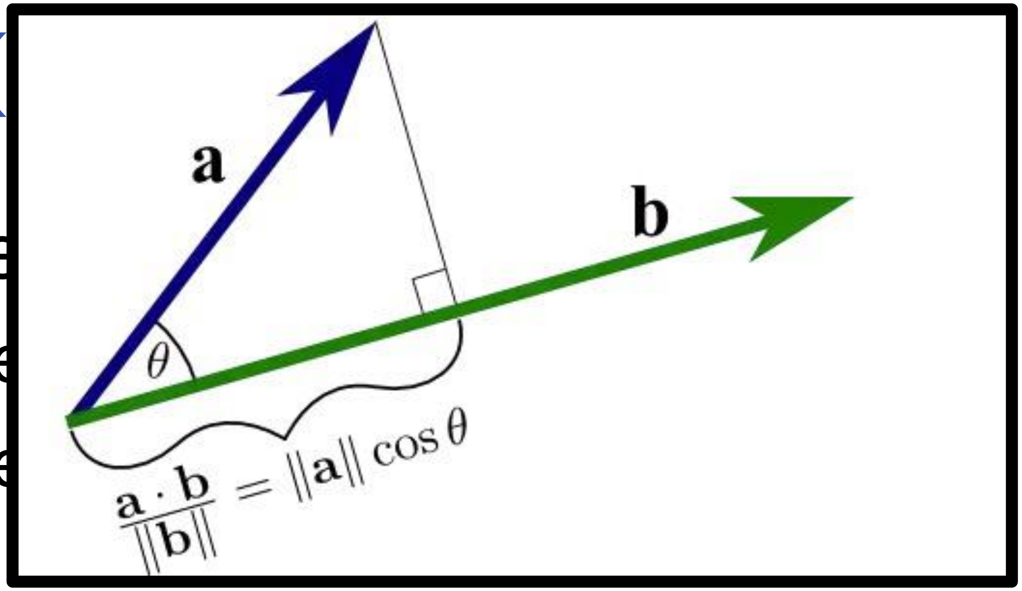
But $\|\mathbf{u}\| = 1$ and cosine is less than 1

So $\mathbf{u}^T \mathbf{w}_t \leq \|\mathbf{w}_t\|$ *(Cauchy-Schwarz inequality)*

Proof (

What we

1. Afte
2. Afte



$$R\sqrt{t} \geq \|\mathbf{w}_t\| \geq \mathbf{u}^T \mathbf{w}_t$$

From (2)

$\mathbf{u}^T \mathbf{w}_t = \|\mathbf{u}\|\ \|\mathbf{w}_t\| \cos($<angle between them>$)$

But $\|\mathbf{u}\| = 1$ and cosine is less than 1

So $\mathbf{u}^T \mathbf{w}_t \leq \|\mathbf{w}_t\|$ *(Cauchy-Schwarz inequality)*

Proof (3/3)

What we know:

1. After t mistakes, $\mathbf{u}^T\mathbf{w}_t \geq t\gamma$
2. After t mistakes, $\|\mathbf{w}_t\|^2 \leq tR^2$

$$R\sqrt{t} \geq \|\mathbf{w}_t\| \geq \mathbf{u}^T\mathbf{w}_t \geq t\gamma$$

From (2)

From (1)

Proof (3/3)

mistakes != # seen data points

What we know:

1. After t mistakes, $\mathbf{u}^T\mathbf{w}_t \geq t\gamma$
2. After t mistakes, $\|\mathbf{w}_t\|^2 \leq tR^2$

$$R\sqrt{t} \geq \|\mathbf{w}_t\| \geq \mathbf{u}^T\mathbf{w}_t \geq t\gamma$$

Number of mistakes $t \leq \frac{R^2}{\gamma^2}$

Bounds the total number of mistakes!