Lecture 6: Linear Classification, Perceptron

Winter 2018

Kai-Wei Chang

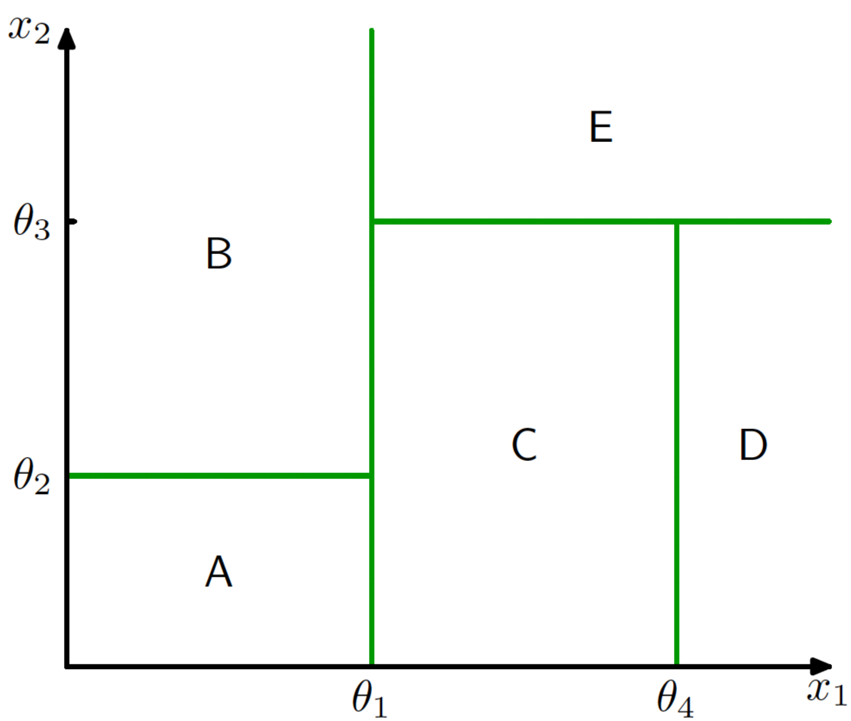
CS @ UCLA

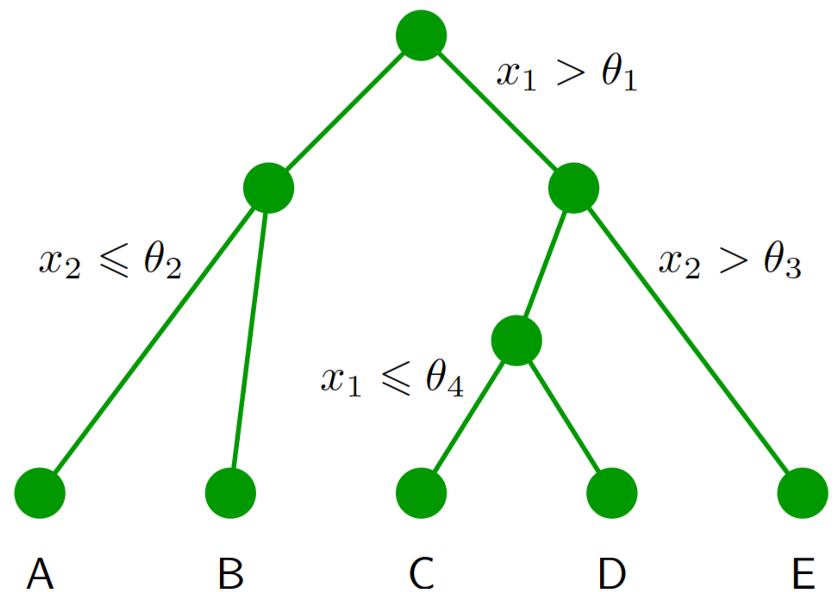
kw+cm146@kwchang.net

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.

Recap: Decision Tree

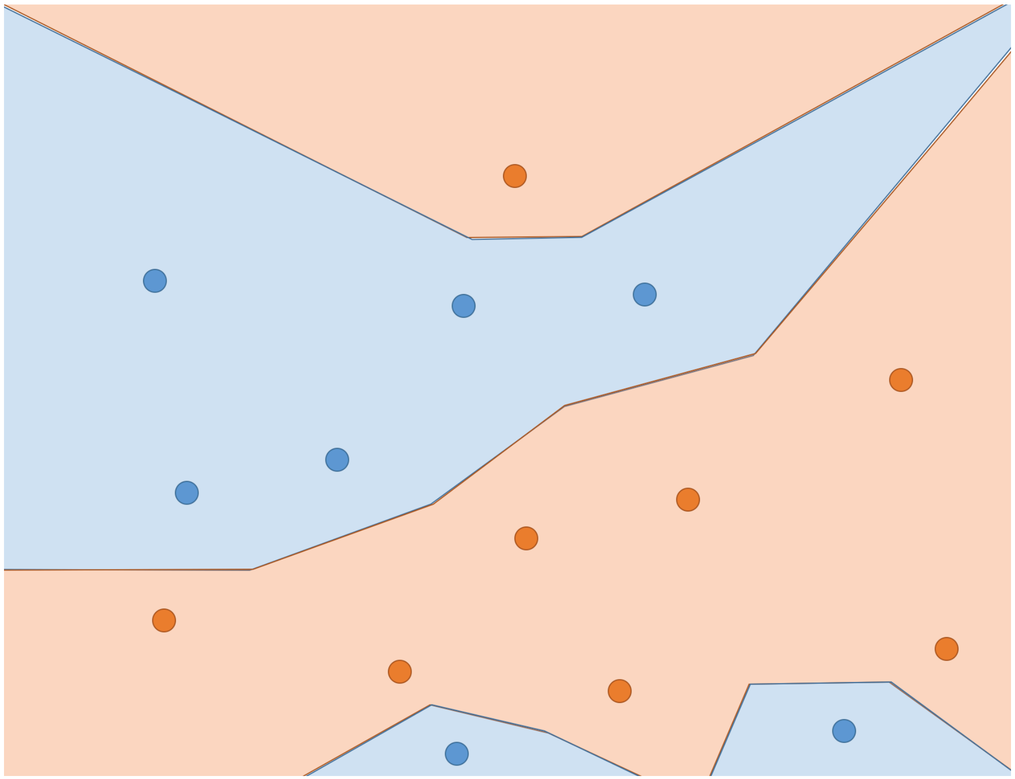
- Learning by splitting input space





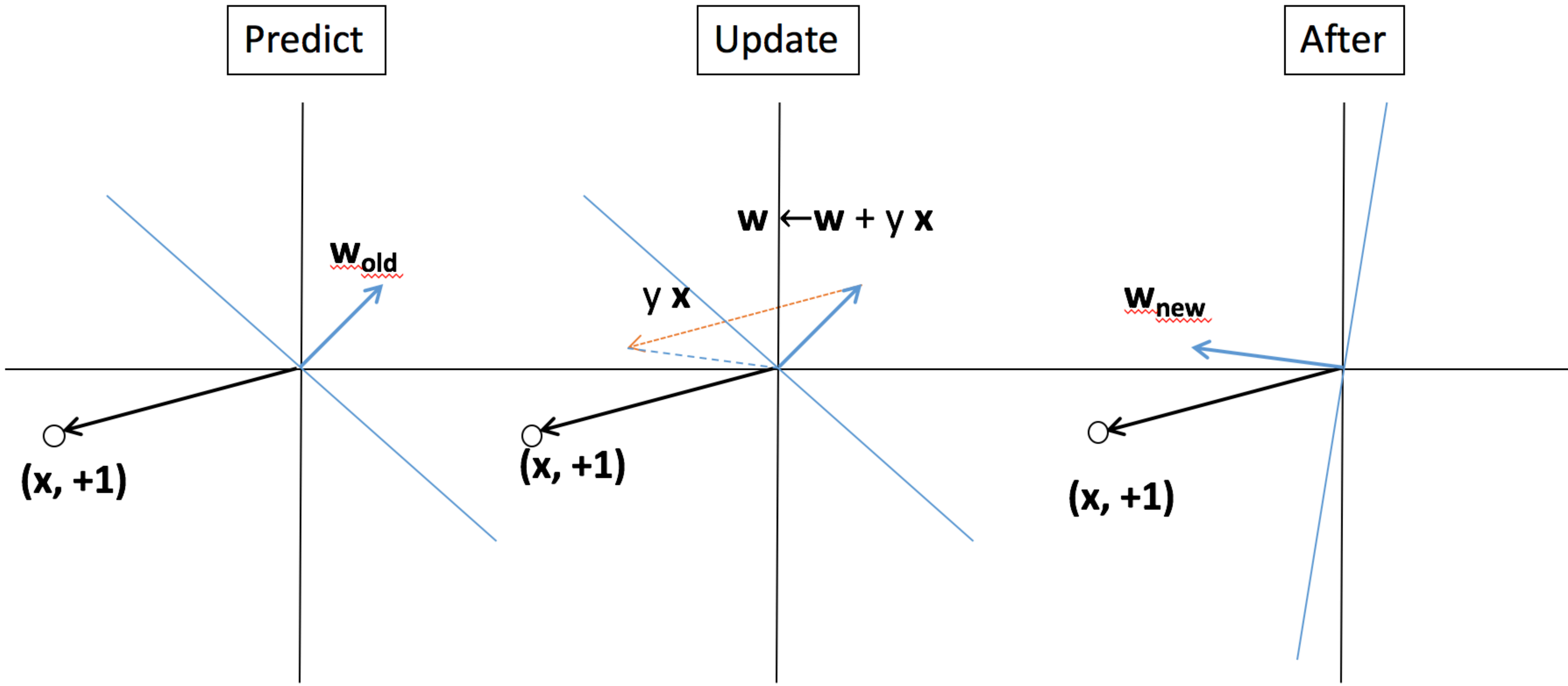
Recap: KNN

❖ Learning by memorization



Today: Perceptron

- Learning by making mistakes



What you will learn today

- Linear models
- The Perceptron Algorithm
- Perceptron Mistake Bound
- Online v.s. Batch learning
- Variants of Perceptron

- Moral for life

mistake

+

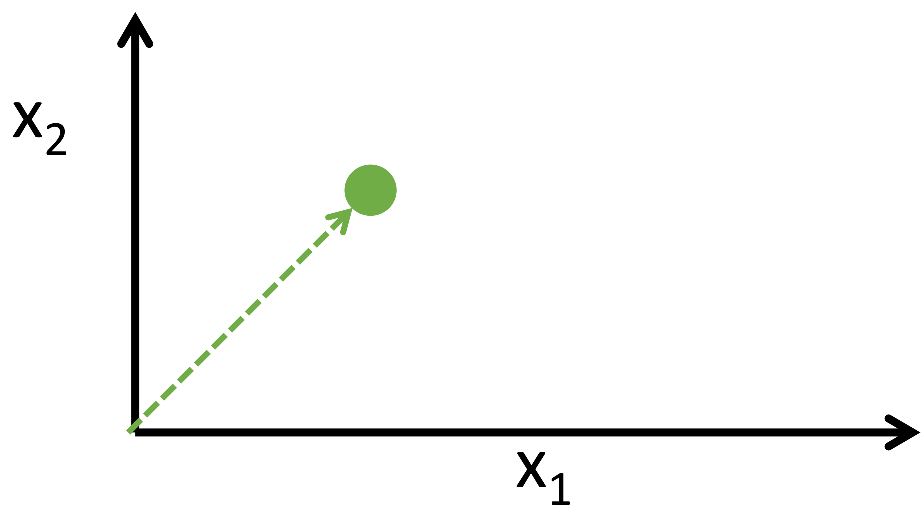
correction

=

learning

Recap: $\mathcal{X}$ as a vector space

- $\mathcal{X}$ is an N-dimensional vector space (e.g. $\mathbb{R}^N$)
  - Each dimension = one feature.
- Each **x** is a feature vector (hence the boldface **x**).
- Think of $\mathbf{x} = [x_1 \ldots x_N]$ as a point in $\mathcal{X}$:



Recap: Linear Classifiers

- Input is a n dimensional vector **x**
- Output is a label $y \in \{-1, 1\}$

- *Linear Threshold Units* (LTUs) classify an example **x** using the following classification rule

  - Output = $\mathrm{sgn}(\mathbf{w}^T\mathbf{x} + b) = \mathrm{sgn}(b + \sum w_i x_i)$

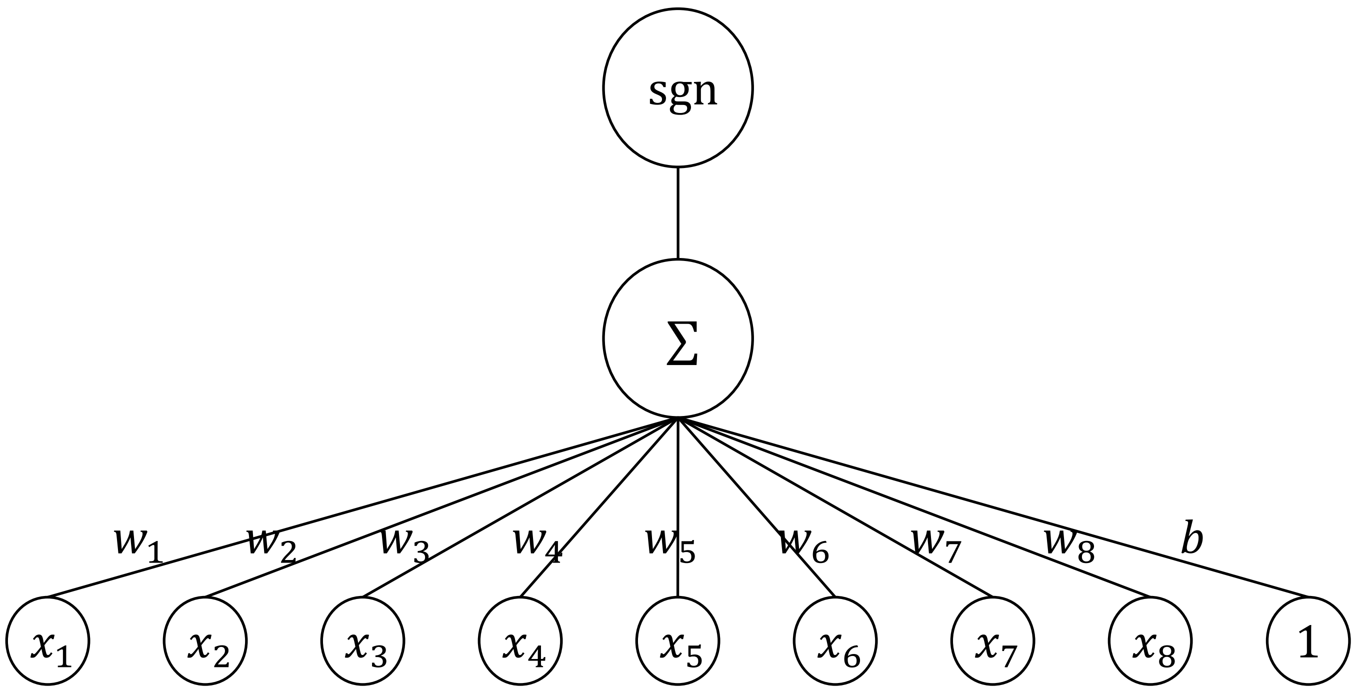
  - $\mathbf{w}^T\mathbf{x} + b \geq 0$ Predict $y = 1$
  - $\mathbf{w}^T\mathbf{x} + b < 0$ Predict $y = -1$

  b is called the bias term

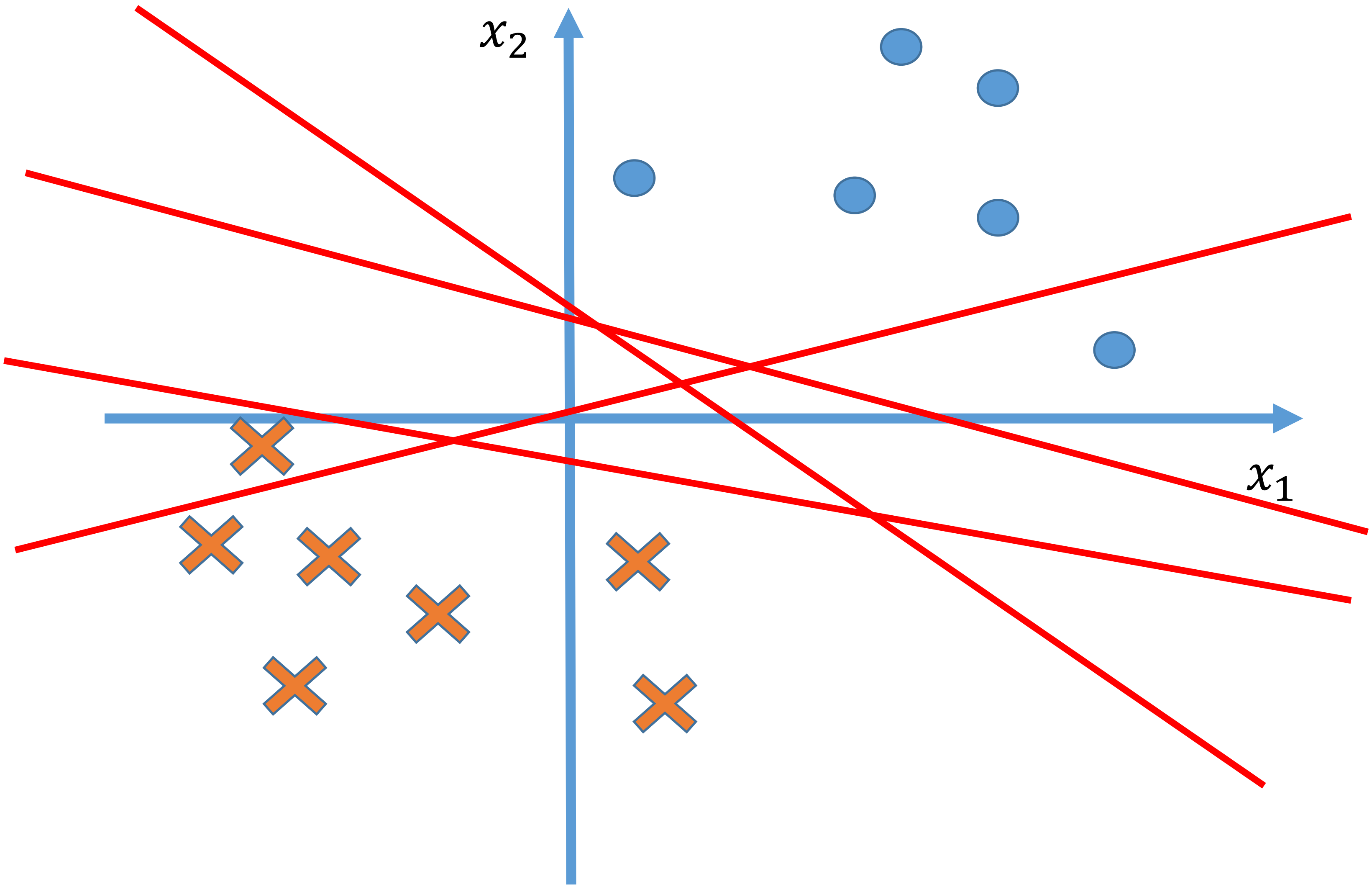
Tie is broken arbitary

Recall: Linear Classifiers

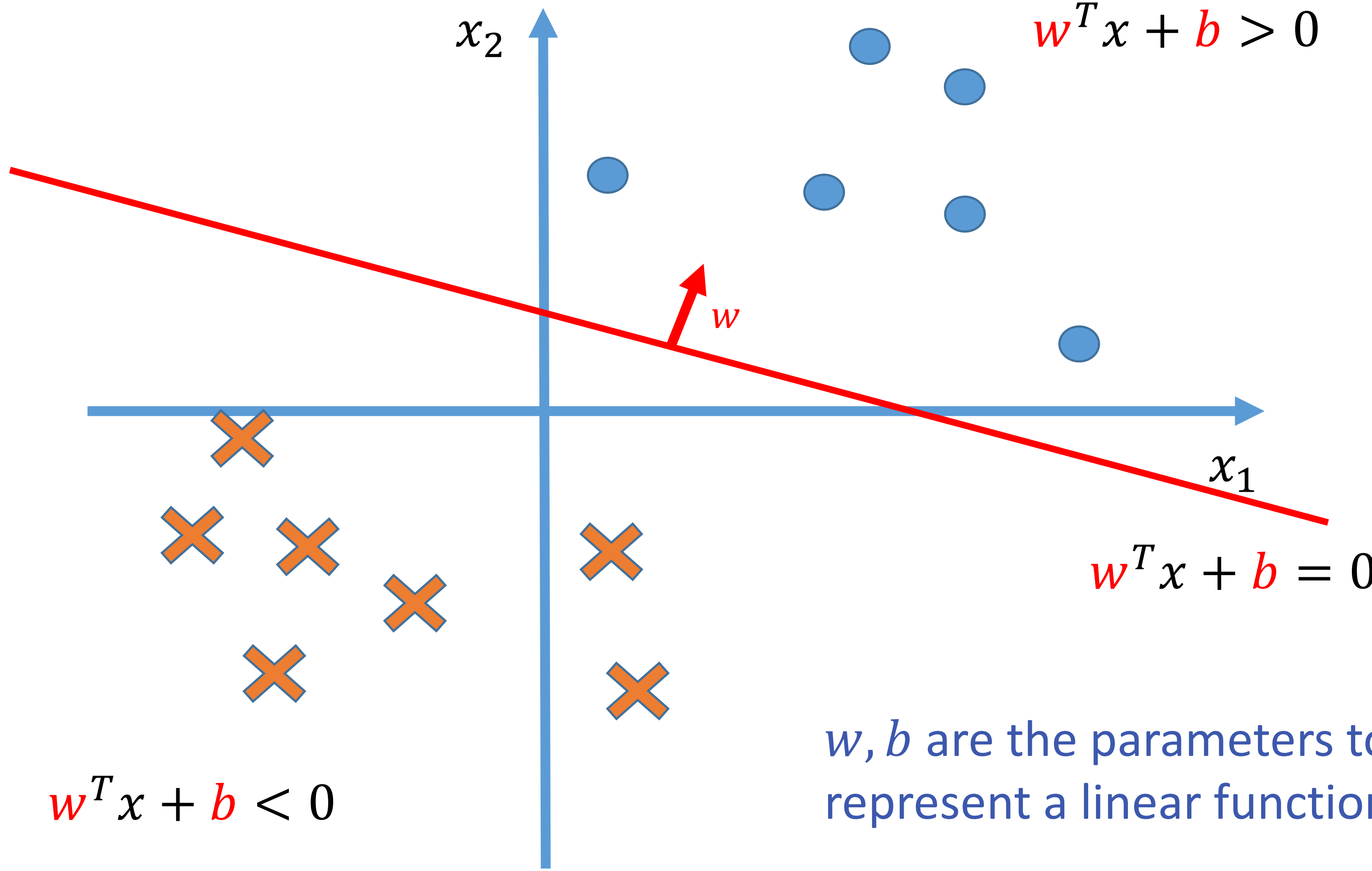
- *Linear Threshold Units* (LTUs) classify an example **x** using the following classification rule



Hypothesis space: linear model

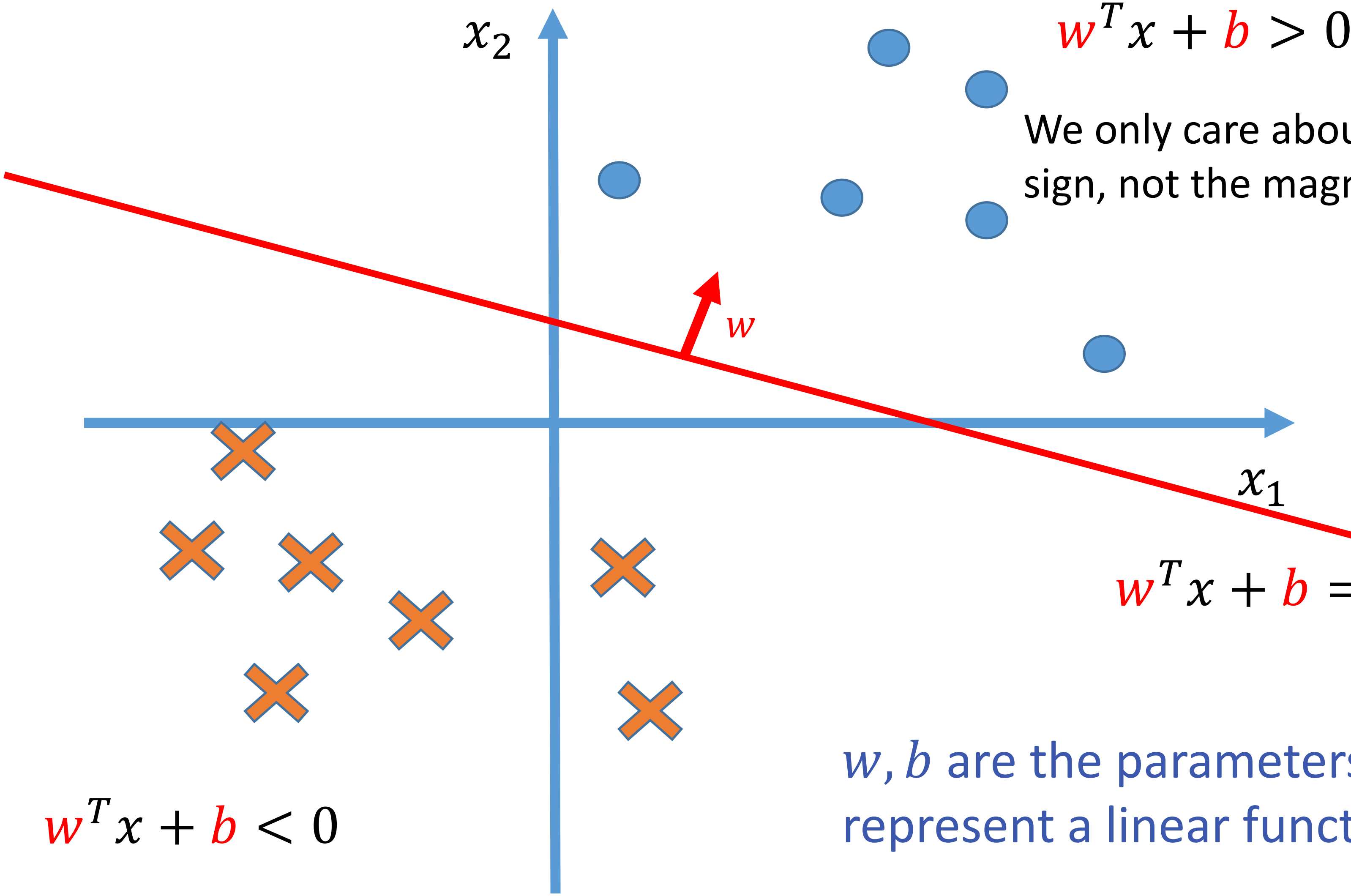


Hypothesis space: linear model



w, b are the parameters to represent a linear function

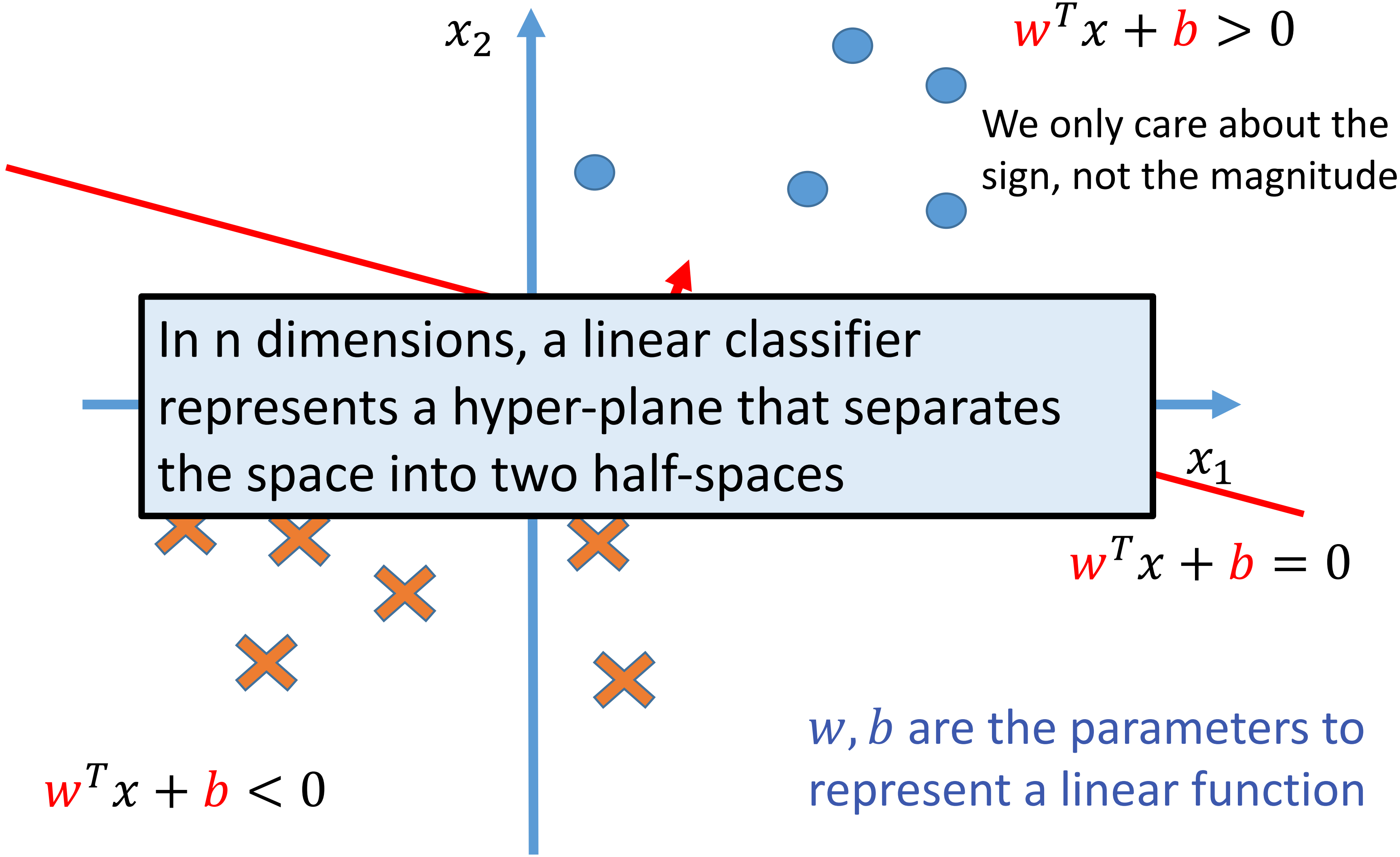
Hypothesis space: linear model



We only care about the sign, not the magnitude

w, b are the parameters to represent a linear function

Hypothesis space: linear model



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

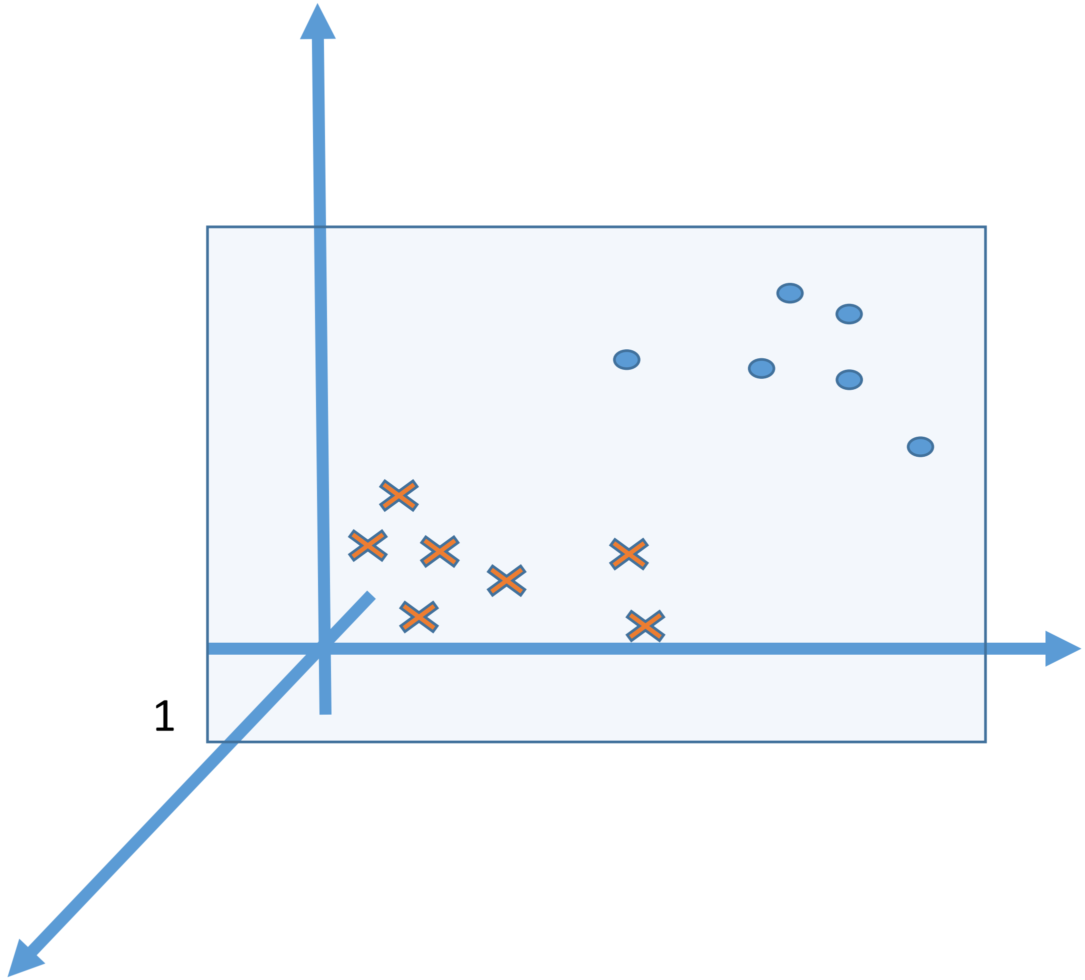
w, b are the parameters to represent a linear function

A simple trick to remove the bias term b

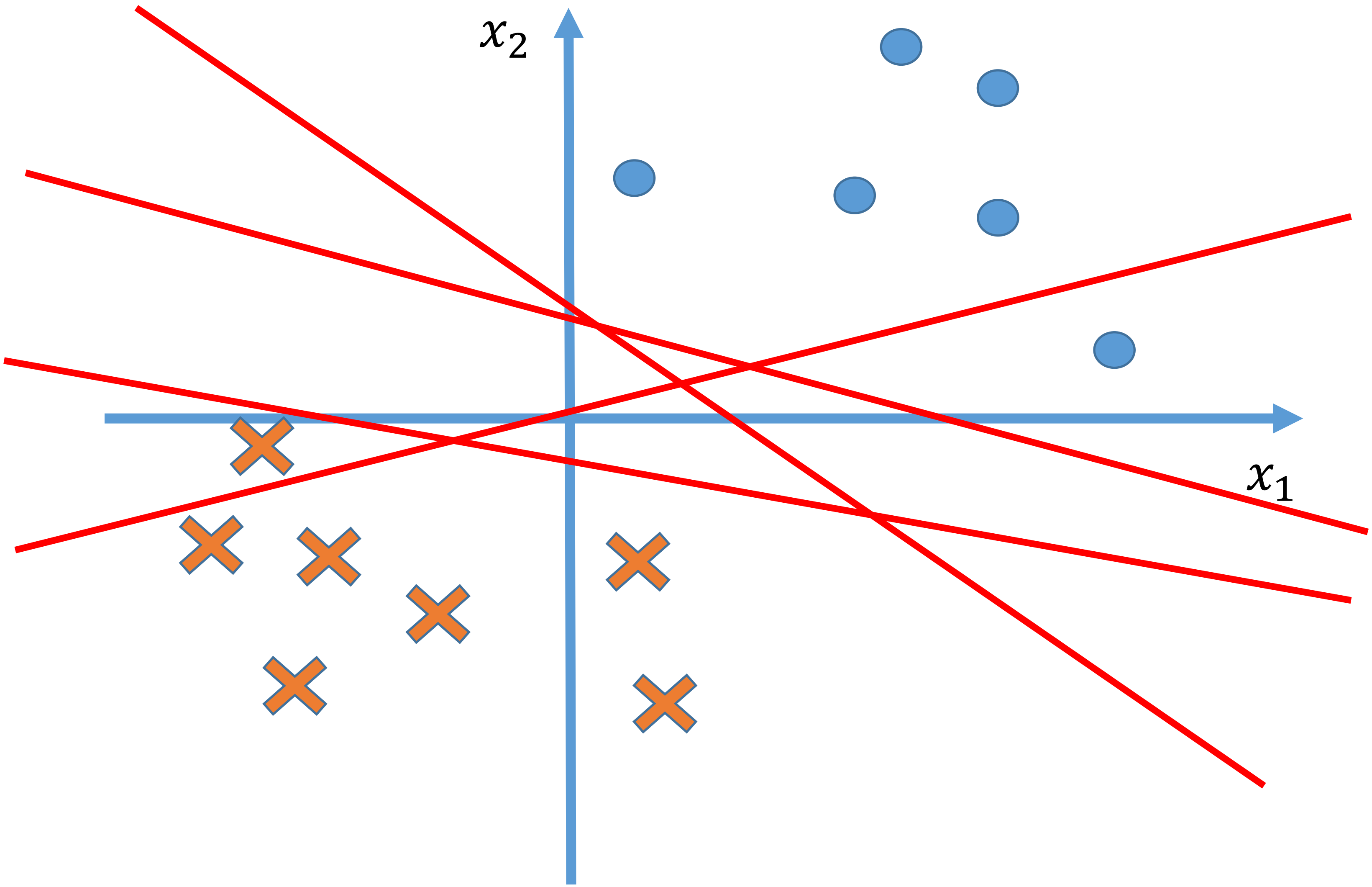
$$
\begin{aligned}
& w^T x + b \\
&= [w^T\ b] \cdot \begin{bmatrix} x \\ 1 \end{bmatrix} \\
&= \tilde{w} \cdot \tilde{x}
\end{aligned}
$$

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

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


1

How to find the best linear model?



Linear classifiers

- There are several algorithms/models

- Perceptron
- (Linear) Support Vector Machines
- Logistic Regression
- …

- Based on different assumptions, you get different linear models

The 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.

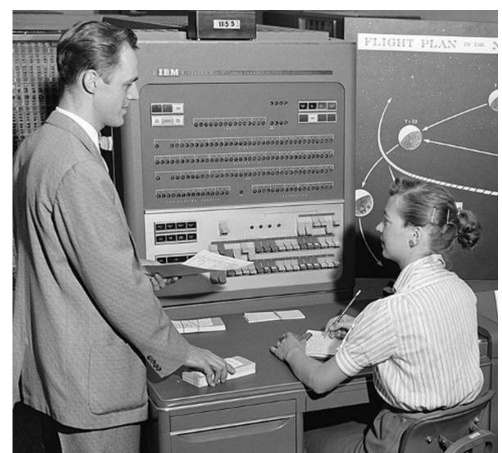
The embryo—the Weather Bureau's $2,000,000 "704" computer—learned to differentiate between right and left after fifty attempts in the Navy's demonstration for newsmen.

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

The Perceptron algorithm

- Rosenblatt 1958
- The goal is to find a separating hyperplane
  - For separable data, guaranteed to find one
- An online algorithm
  - Processes one example at a time
- Converges if data is separable
  -- mistake bound
- Several variants exist (will discuss briefly at towards the end)

The Perceptron Algorithm [Rosenblatt 1958]

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

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

Return w
```

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

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

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

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

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 w ← 0 ∈ ℝ^n
For (x, y) in D:
    if y(wᵀx) ≤ 0
        w ← w + yx

Return w
```

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

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

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.

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

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.

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 $y(\boldsymbol{w}^\top \boldsymbol{x}) < 0$
4. $\boldsymbol{w} \leftarrow \boldsymbol{w} + y\boldsymbol{x}$
5.
6. Return $\boldsymbol{w}$

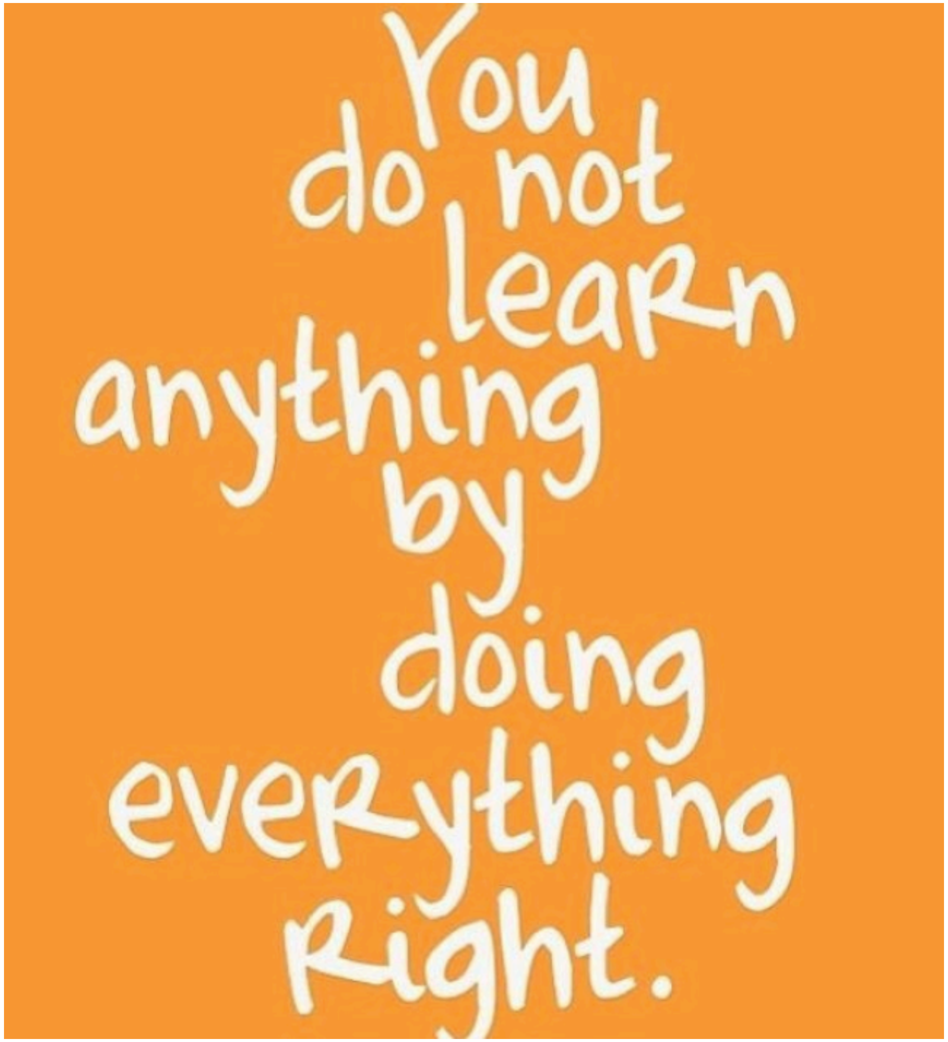
Update only on error. A mistake-driven algorithm

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

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.

Moral quotes for life



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 $\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{x}$

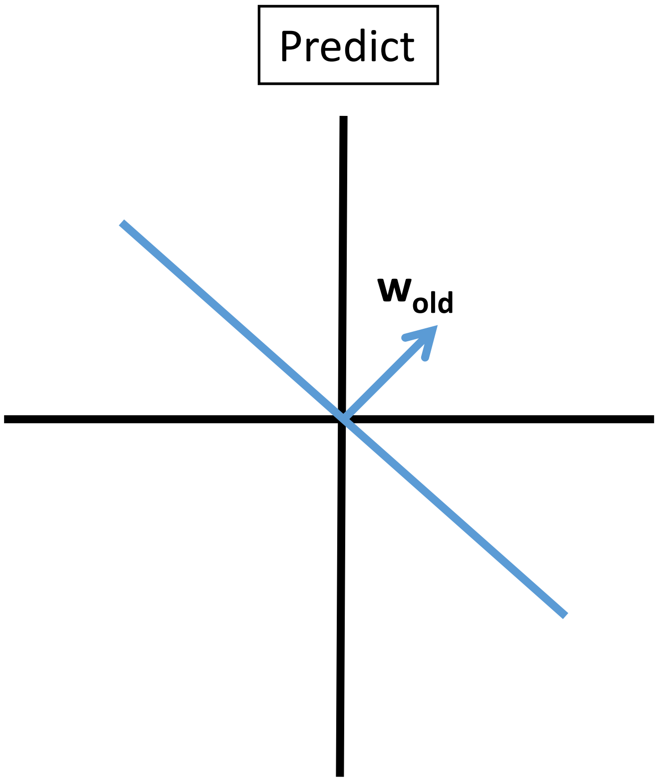
The new dot product will be

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

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

Similar reasoning for negative examples

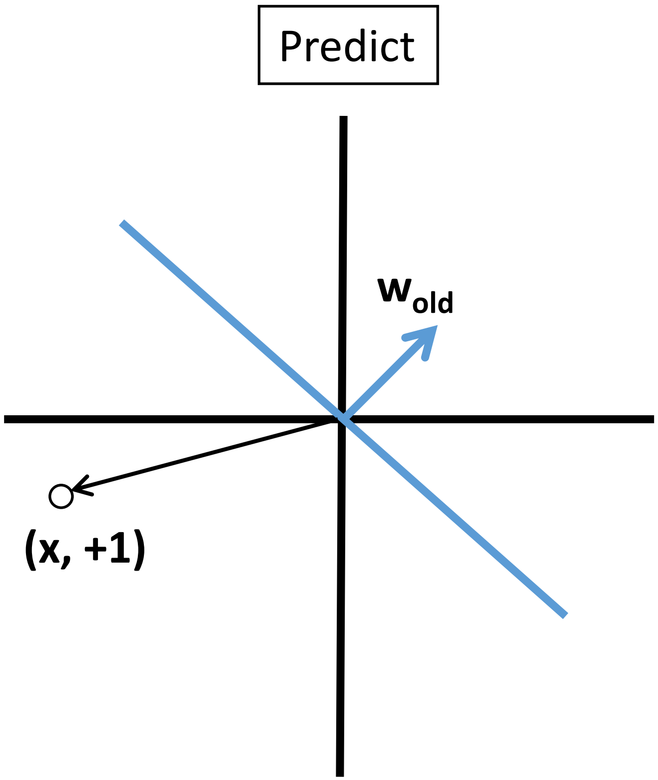
Geometry of the perceptron update



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

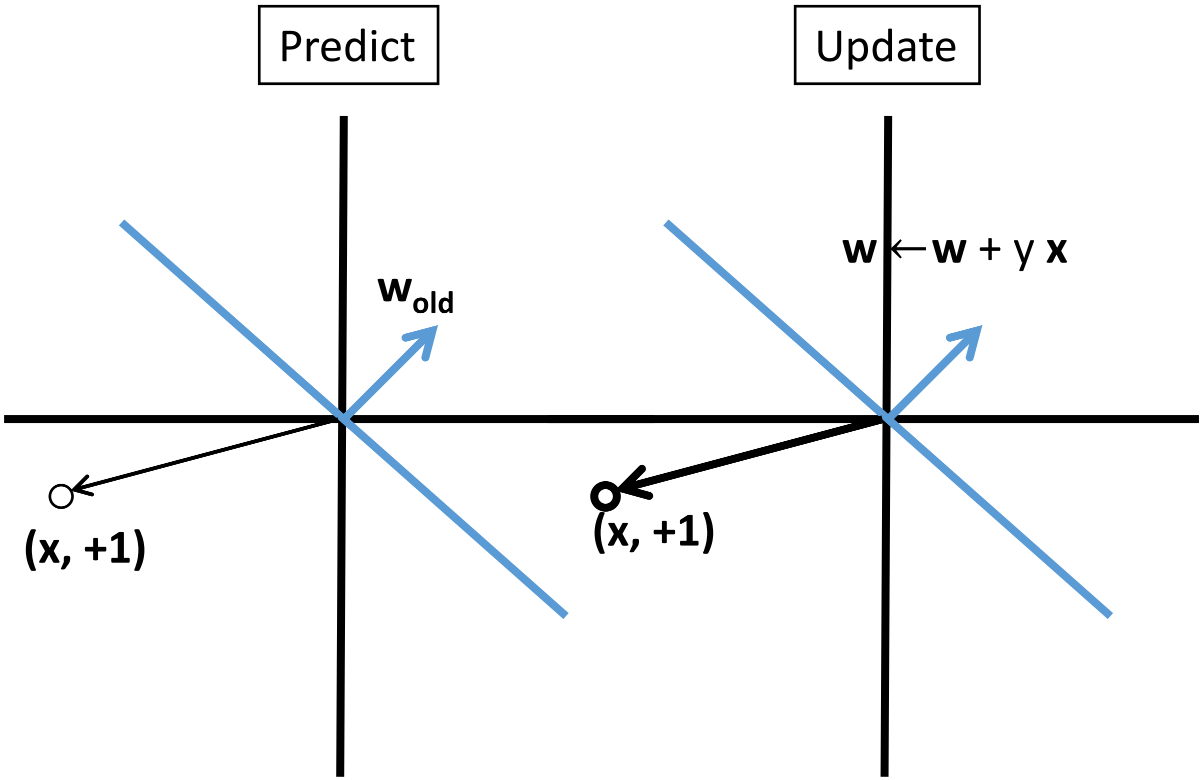
Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}_i$
Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}_i$



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

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

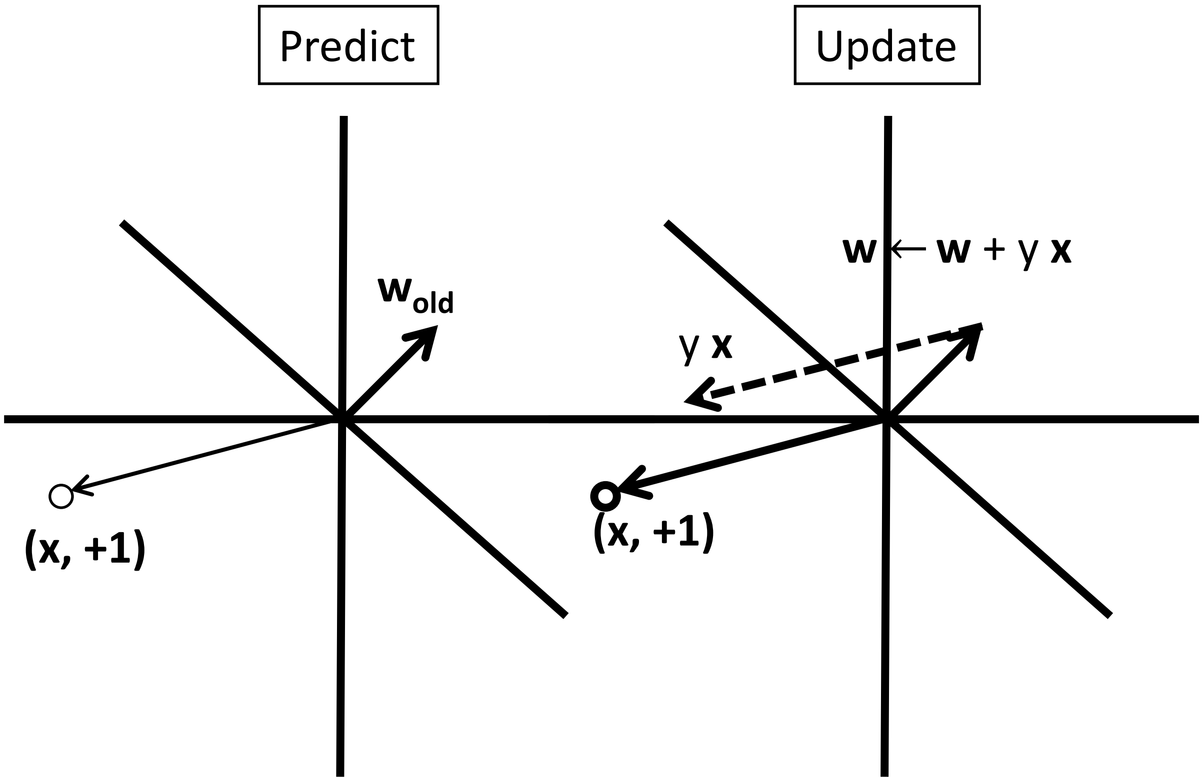
Geometry of the



For a mistake on a positive example

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

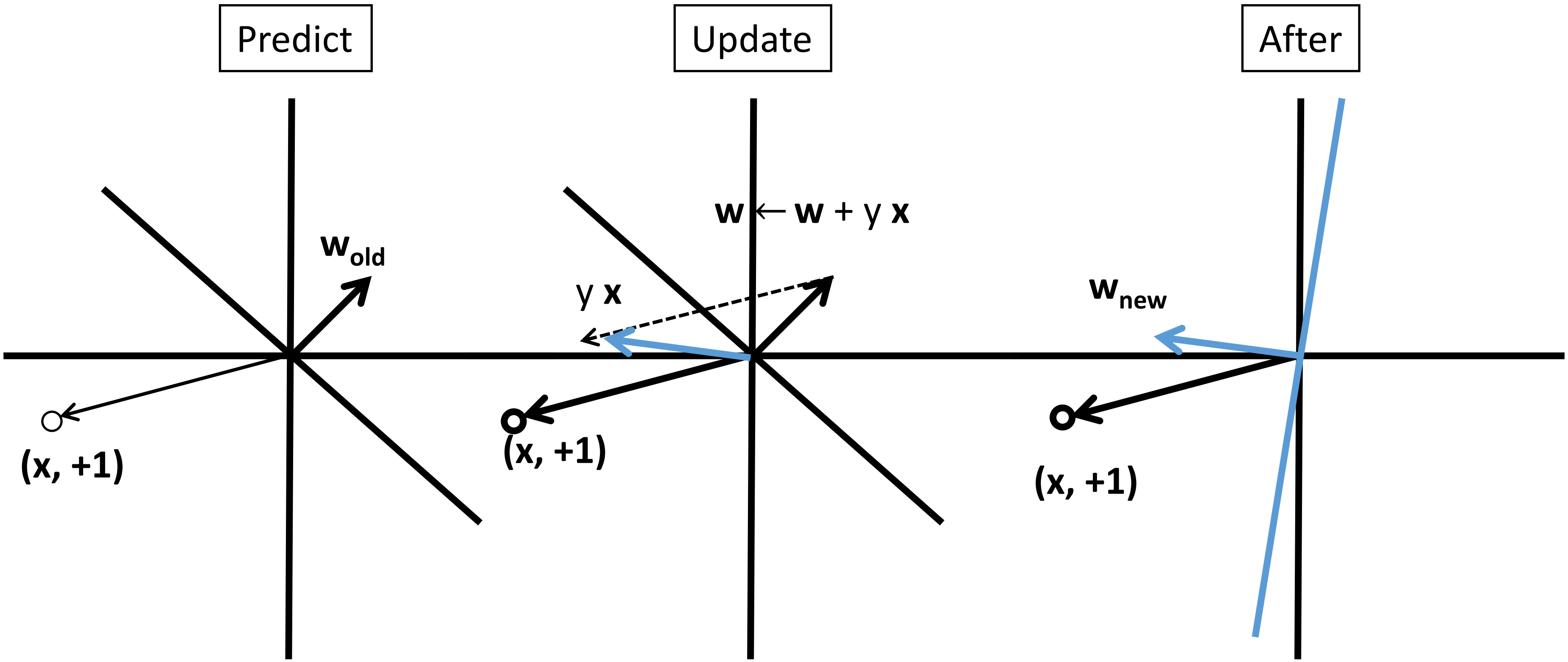
Geometry of the



For a mistake on a positive example

Geometry of the

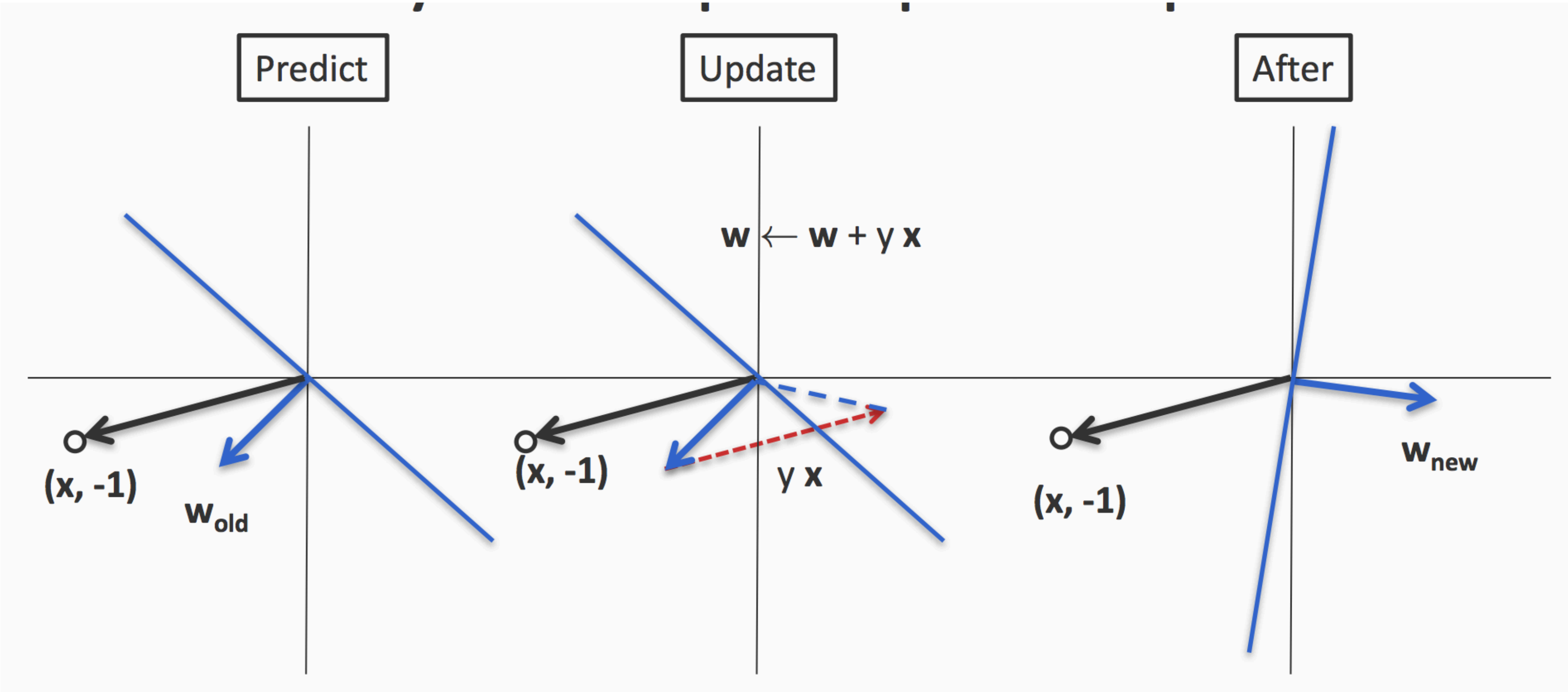
Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta\, \mathbf{x}_i$
Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta\, \mathbf{x}_i$



For a mistake on a positive example

Geometry of the

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



For a mistake on a negative example

Perceptron Learnability

- Obviously Perceptron cannot learn what it cannot represent
  - Only linearly separable functions
- Minsky and Papert (1969) wrote an influential book demonstrating Perceptron's representational limitations
  - Parity functions can't be learned (XOR)
    - But we already know that XOR is not linearly separable

Check point: What you need to know

- The Perceptron algorithm
- The geometry of the update
- What can it represent

What you will learn today

- Linear models
- The Perceptron Algorithm
- Perceptron Mistake Bound
- Variants of Perceptron
- Online v.s. Batch learning

Convergence

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.

Convergence

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.

Cycling theorem

- 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

Moral Quotes

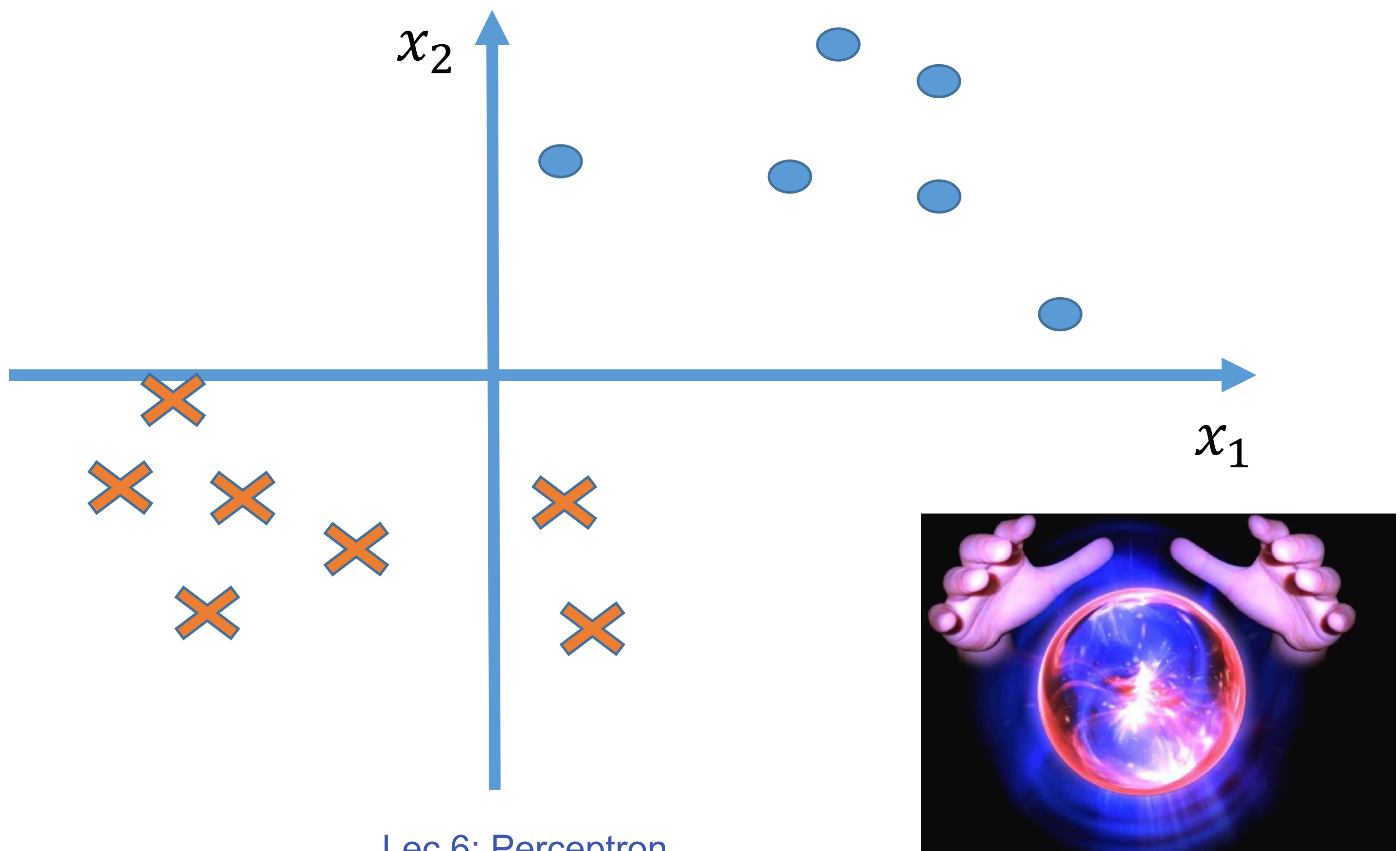
THE MOST DISTURBING PART OF WHAT HAS OCCURRED IS THE VULTURES ARE CIRCLING.

Jim Miller

We know! Because vultures are not linear separable!

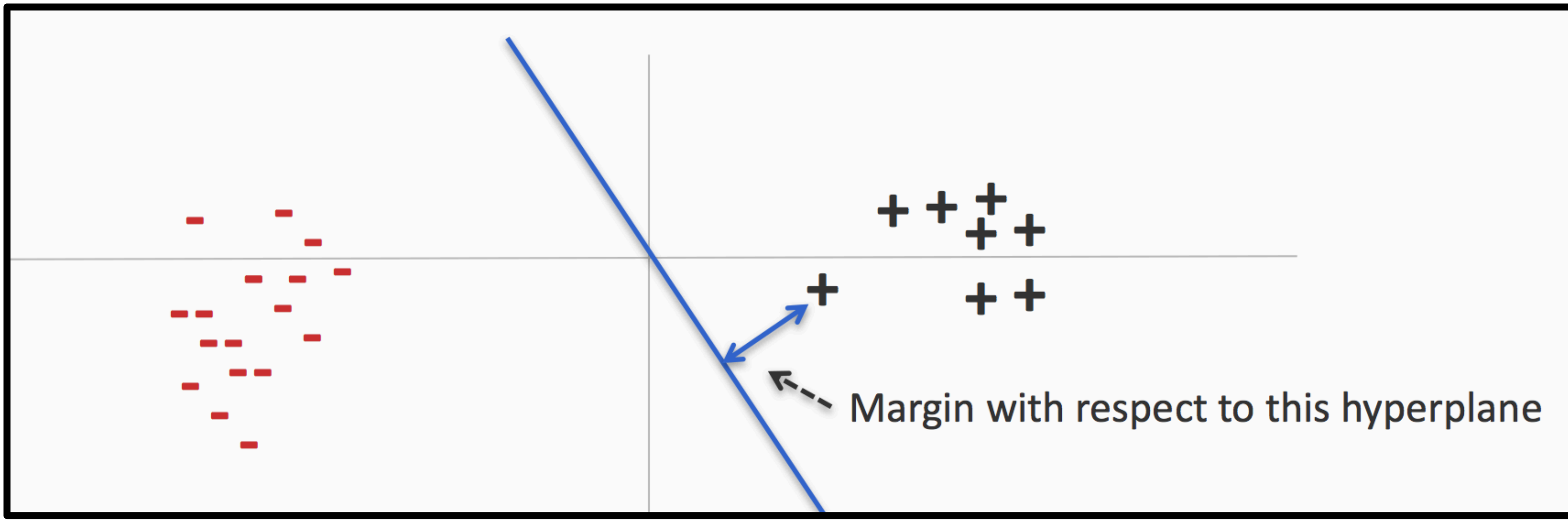
Linear separable

There exists a hyper-plane that can separate the data
(we may not know where it is)



Margin

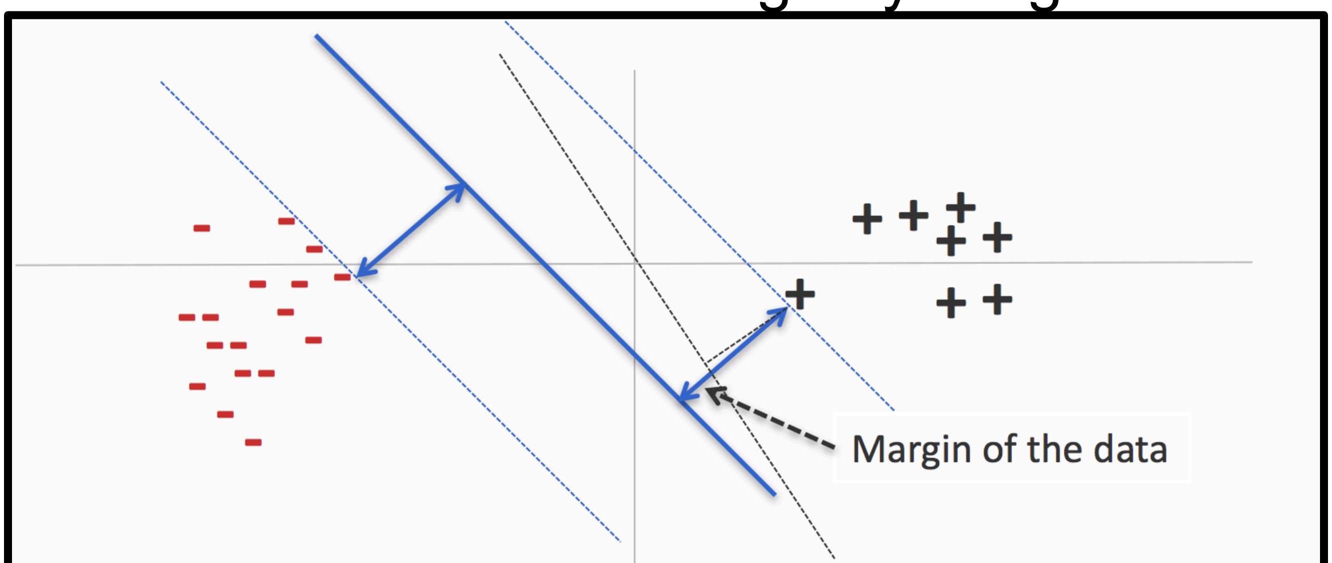
The margin of a hyperplane for a dataset is the distance between the hyperplane and the data point nearest to it.



Margin

- 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.



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 $y_i\ (\mathbf{u}^\top \mathbf{x}_i) \geq \gamma$.

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

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 we have a binary classification dataset with n dimensional inputs.

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

If the data is separable,…

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

…then the Perceptron algorithm will find a separating hyperplane after making a finite number of mistakes

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 $y_i \in \{-1, +1\}$.

We can always find such an R. Just look for the farthest data point from the origin.

Suppose there exists a unit vector $\mathbf{u} \in R^n$ (i.e $\|\mathbf{u}\| = 1$) such that for some $\gamma \in R$ and $\gamma > 0$ 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.

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 $y_i \in \{-1, +1\}$.

We can always find such an R. Just look for the farthest data point from the origin.

Suppose there exists a unit vector $\mathbf{u} \in R^n$ (i.e $\|\mathbf{u}\| = 1$) such that for some $\gamma \in R$ and $\gamma > 0$ 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.

The data has a margin γ.
Importantly, the data is *separable*.
γ is the complexity parameter that defines the separability of data.

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 $y_i \in \{-1, +1\}$.

We can always find such an R. Just look for the farthest data point from the origin.

Suppose there exists a unit vector $\mathbf{u} \in R^n$ (i.e $\|\mathbf{u}\| = 1$) such that for some $\gamma \in R^n$ and $\gamma > 0$ 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.

If **u** hadn't been a unit vector, then we could scale γ in the mistake bound. This will change the final mistake bound to $(\|\mathbf{u}\|R/\gamma)^2$

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 we have a binary classification dataset with n dimensional inputs.

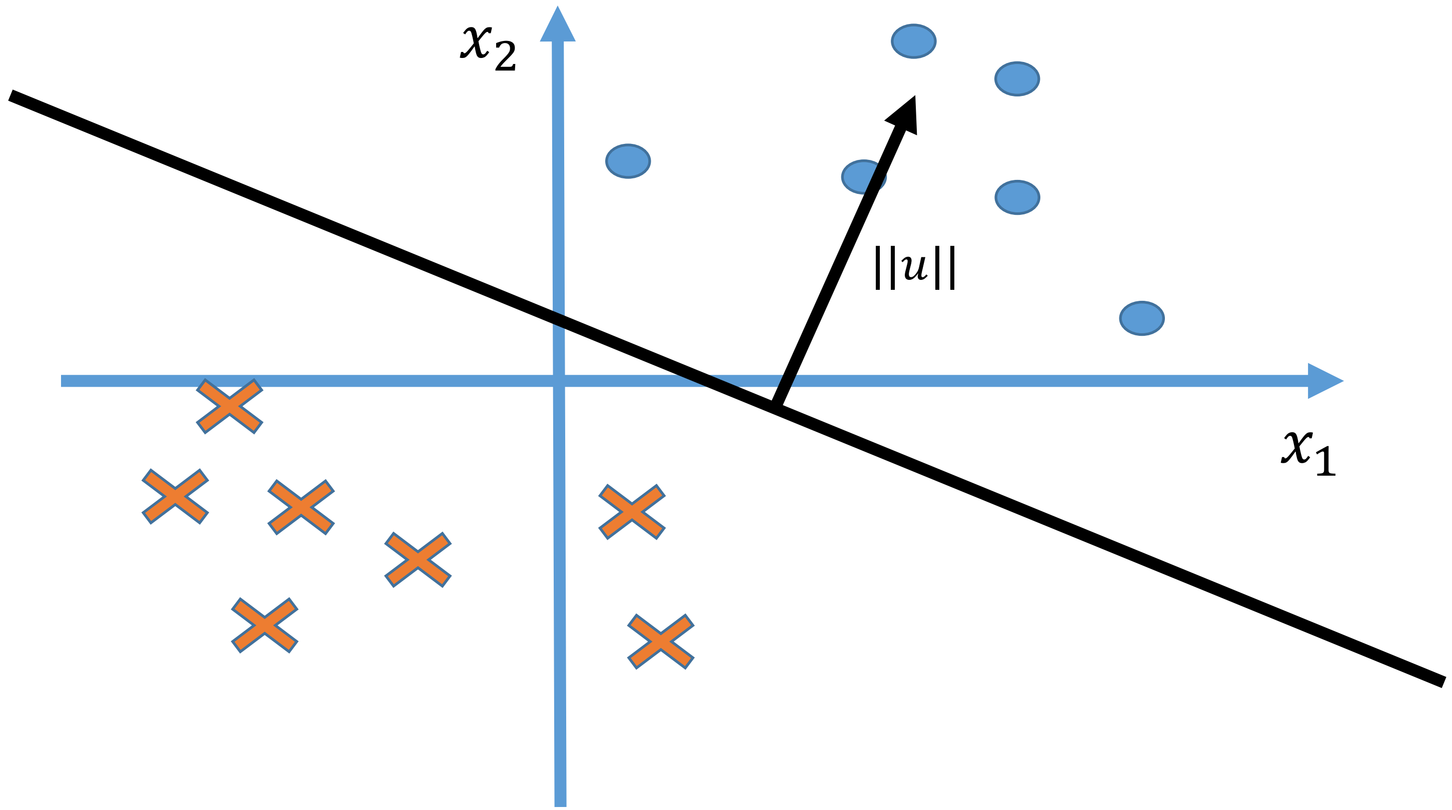
Suppose there exists a unit vector $\mathbf{u} \in R^n$ (i.e $\|\mathbf{u}\| = 1$) such that for some $\gamma \in R^n$ and $\gamma > 0$ we have $y_i (\mathbf{u}^T \mathbf{x}_i) \geq \gamma$.

If the data is separable,…

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

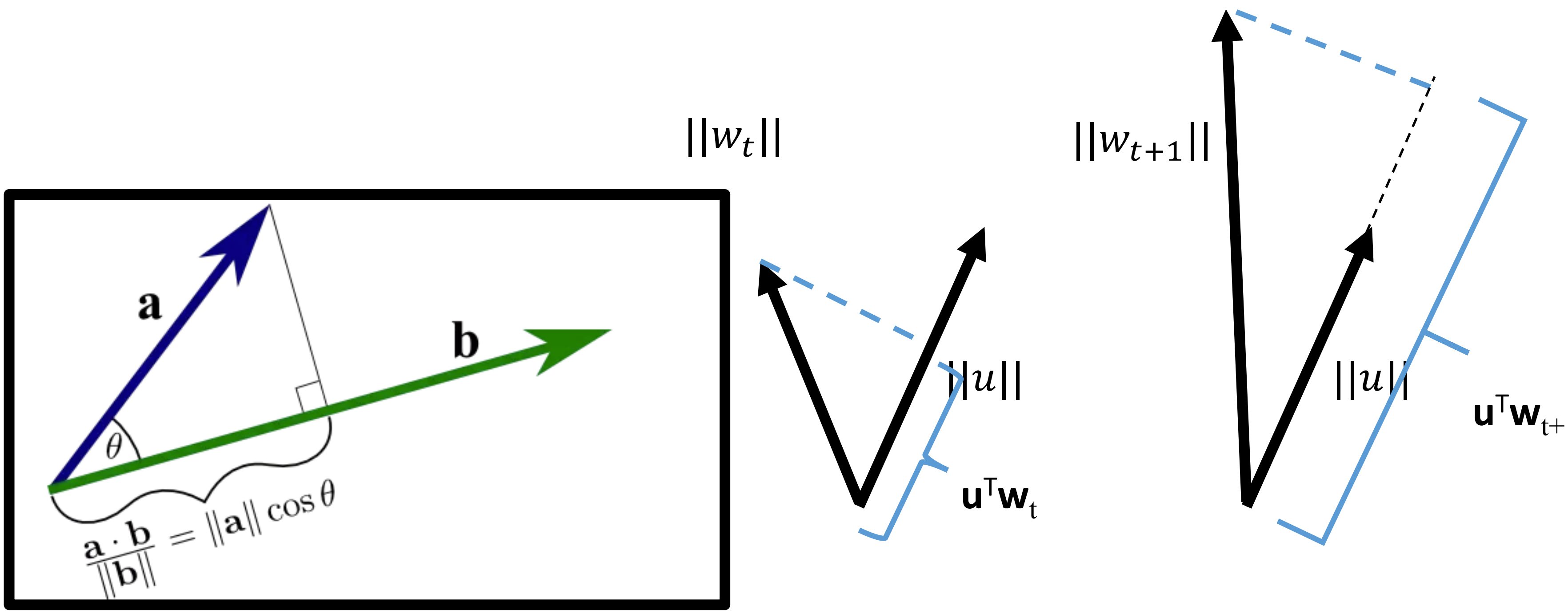
…then the Perceptron algorithm will find a separating hyperplane after making a finite number of mistakes

Intuition



Intuition

1. After update, $\mathbf{u}^{\mathsf{T}}\mathbf{w}_{t+1}$ gets longer than $\mathbf{u}^{\mathsf{T}}\mathbf{w}_{t}$
2. The size of $||w_{t+1}||$ may increase, but not much



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

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

- 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 (1/3)

1. C

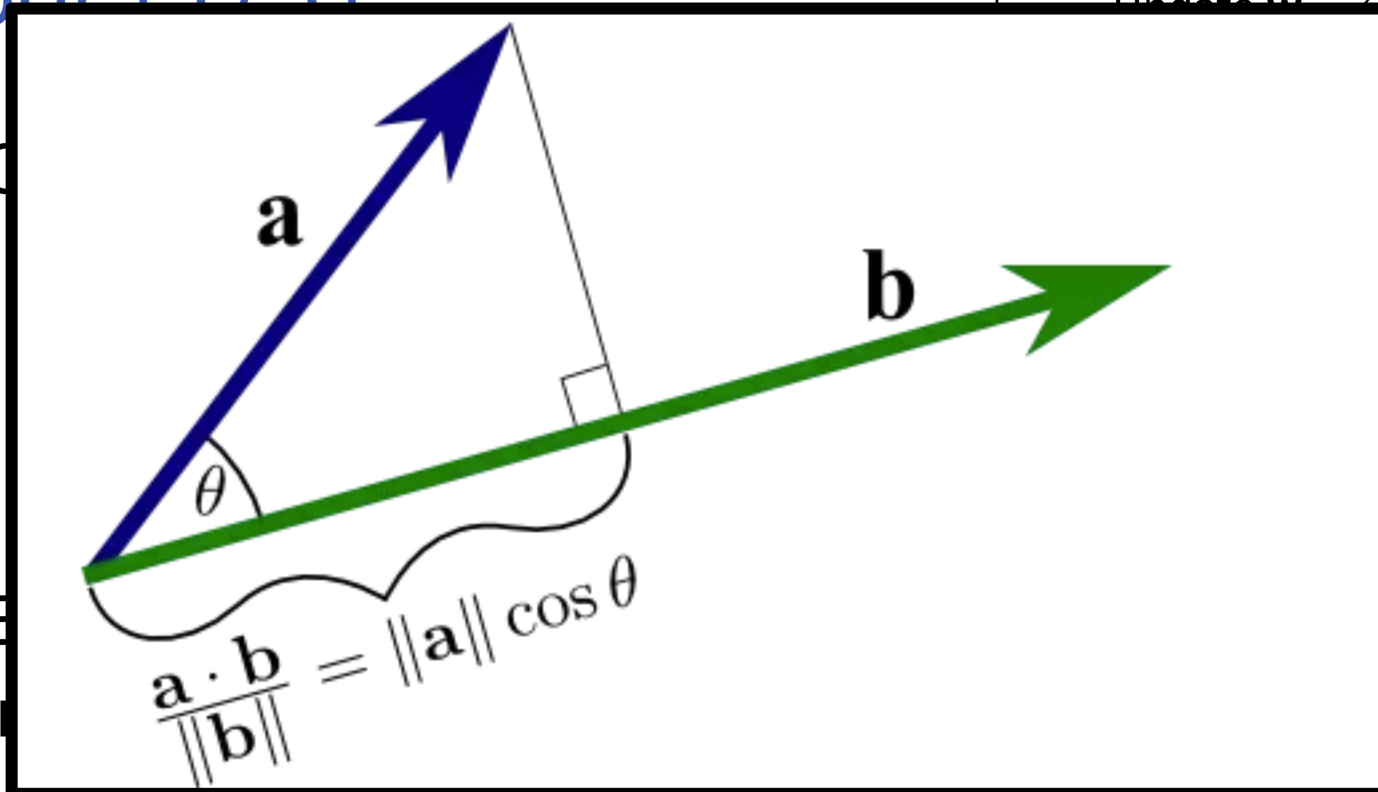
data is

y a

B

i

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



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

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

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

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

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

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)

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

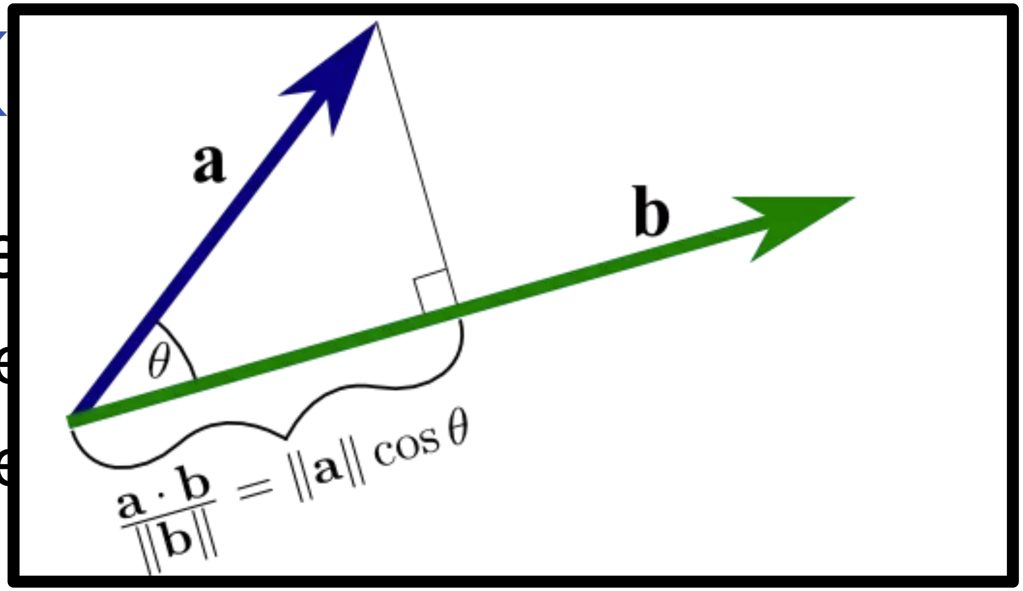
But $\|\mathbf{u}\| = 1$ and cosine is less than 1

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)

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

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

So $\mathbf{u}^{\mathrm{T}}\,\mathbf{w}_{\mathrm{t}} \leq \|\mathbf{w}_{\mathrm{t}}\|$ *(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, $\|\mathbf{w}_t\|^2 \leq tR^2$

$$\underbrace{R\sqrt{t} \geq \|\mathbf{w}_t\|}_{\text{From (2)}} \geq \underbrace{\mathbf{u}^T\mathbf{w}_t \geq t\gamma}_{\text{From (1)}}$$

Proof (3/3)

mistakes != # seen data points

What we know:

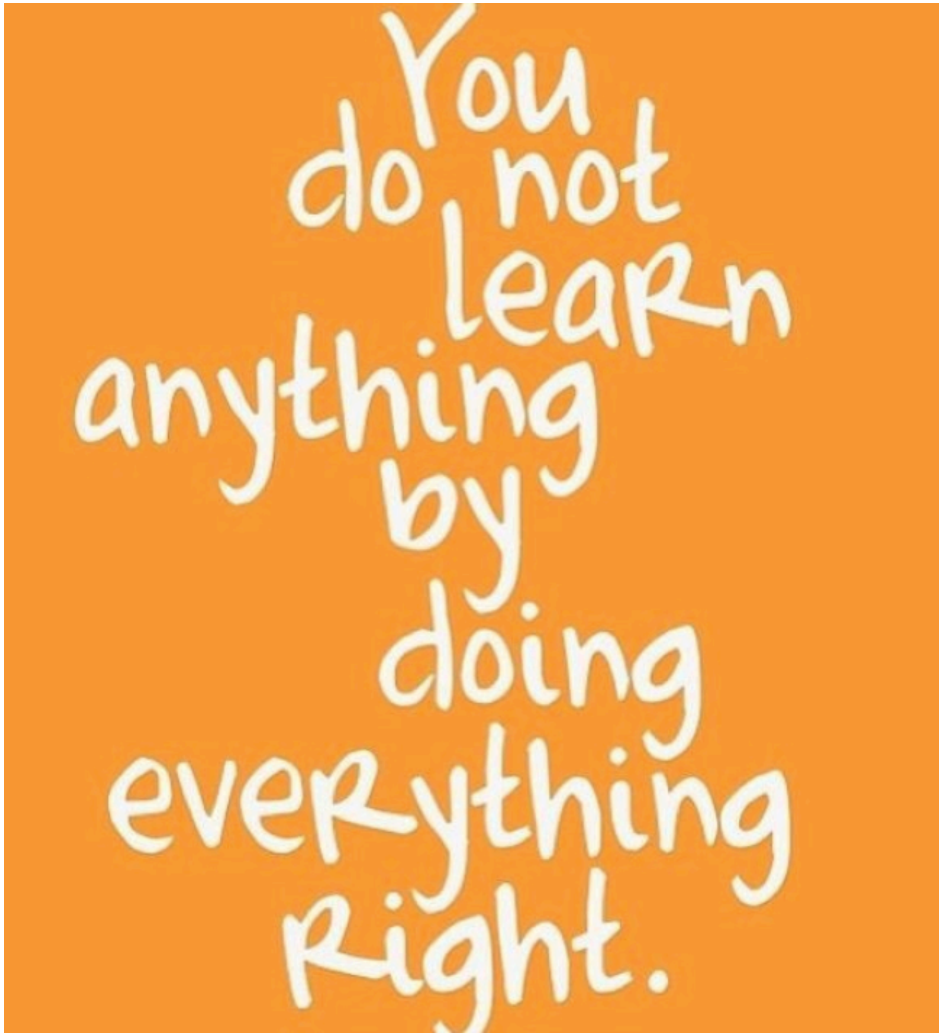
1. After t mistakes, $\mathbf{u}^{\mathrm{T}}\mathbf{w}_{\mathrm{t}} \geq t\gamma$
2. After t mistakes, $\|\mathbf{w}_{\mathrm{t}}\|^2 \leq tR^2$

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

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

Bounds the total number of mistakes!

NOTE!! # mistakes!= # seen data points



Mistake Bound Theorem [Novikoff 1962, Block 1962]

Let $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots\}$ 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 $\gamma \in R^n$ and $\gamma > 0$ we have $y_i\,(\mathbf{u}^\top \mathbf{x}_i) \geq \gamma$.

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

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 we have a binary classification dataset with n dimensional inputs.

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

If the data is separable,…

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

…then the Perceptron algorithm will find a separating hyperplane after making a finite number of mistakes

The Perceptron Mistake bound

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

- R is a property of the dimensionality. How?
  - For Boolean functions with n attributes, show that $R^2 = n$.
- γ is a property of the data

Beyond the separable case

- Good news
  - Perceptron makes no assumption about data distribution, could be even adversarial
  - After a fixed number of mistakes, you are done. Don't even need to see any more data
- Bad news: Real world is not linearly separable
  - Can't expect to *never* make mistakes again
  - What can we do: more features, try to be linearly separable if you can, use averaging

Check point: What you need to know

- What is the perceptron mistake bound?

- How to prove it

What you will learn today

- Linear models
- The Perceptron Algorithm
- Perceptron Mistake Bound
- Variants of Perceptron
- Online v.s. Batch learning

Practical use of the Perceptron algorithm

1. Using the Perceptron algorithm with a finite dataset

2. Voting and Averaging

3. Margin Perceptron

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 $\boldsymbol{w}$

Update only on error. A mistake-driven algorithm

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

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.

The Perceptron Algorithm

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

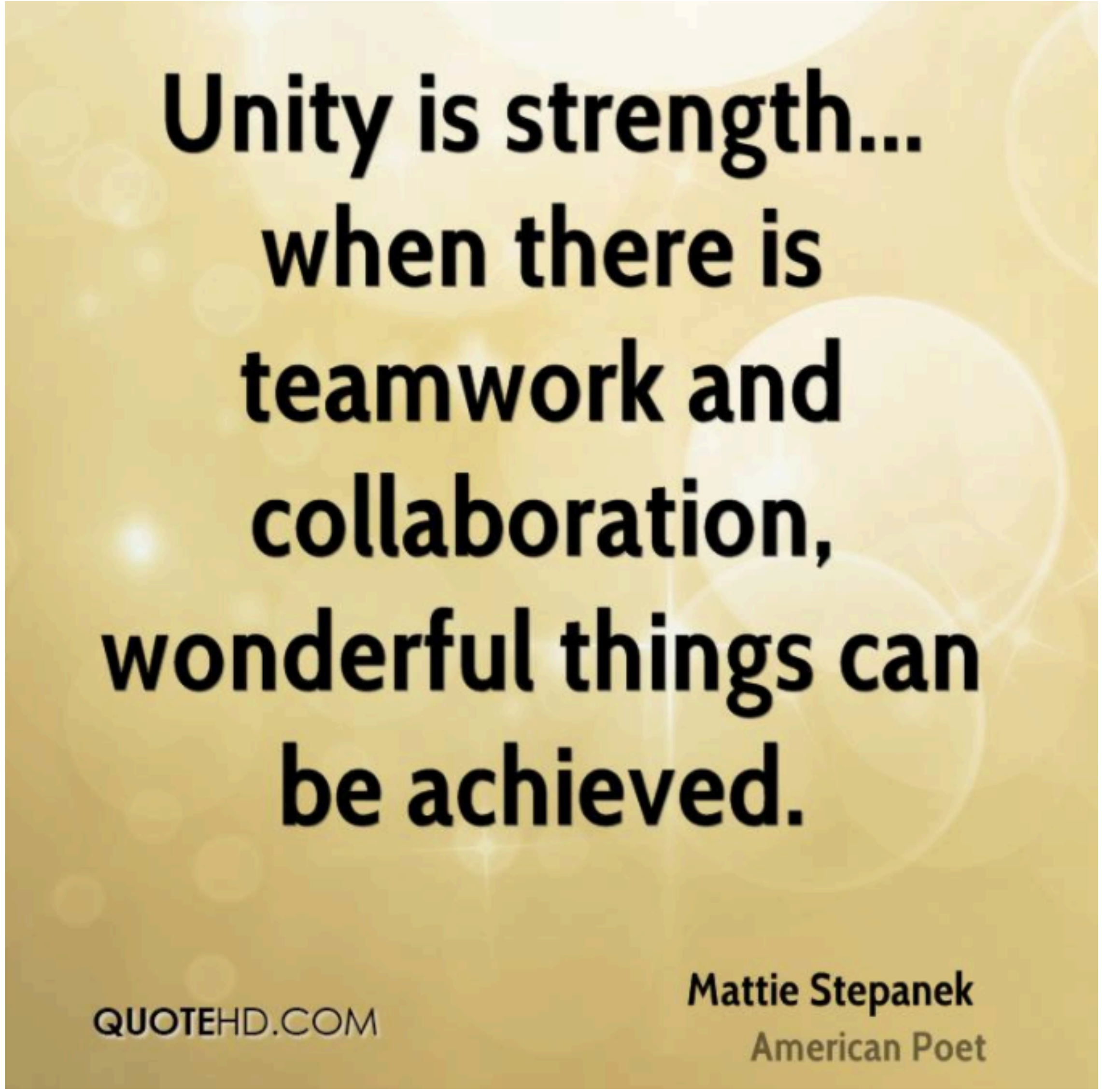
Prediction: $y^{\text{test}} \leftarrow sgn(\boldsymbol{w}^\top \boldsymbol{x}^{\text{test}})$

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.

2. Voting and Averaging

- So far: We return the final weight vector
- Aggregating the models on the learning path may give better results
  - Especially, when data is not sperable

- Voted perceptron
- Averaged perceptron


Unity is strength...
when there is
teamwork and
collaboration,
wonderful things can
be achieved.
Mattie Stepanek
American Poet
QUOTEHD.COM

Voted perceptron

- Remember every weight vector in your sequence of updates.
- At final prediction time, each weight vector gets to vote on the label.
- The number of votes it gets is the number of iterations it survived before being updated
- Comes with strong theoretical guarantees about generalization, impractical because of storage issues

2. Voting and Averaging

- So far: We return the final weight vector

- Averaged perceptron
  - Instead of using all weight vectors, use the average weight vector (i.e longer surviving weight vectors get more say)

  - More practical alternative and widely used

Averaged Perceptron

Given a training set D = $\{(x_i, y_i)\}$, $x_i \in \Re^n$, $y_i \in \{-1,1\}$

1. Initialize $\mathbf{w} = 0 \in \Re^n$ and $\mathbf{a} = 0 \in \Re^n$

2. For epoch = 1 … T:
   - For each training example $(x_i, y_i) \in D$:
     - If $y_i \mathbf{w}^T x_i \leq 0$
       - update $\mathbf{w} \leftarrow \mathbf{w} + r\, y_i\, x_i$
     - $\mathbf{a} \leftarrow \mathbf{a} + \mathbf{w}$

3. Return **a**

Prediction: sgn($\mathbf{a}^T\mathbf{x}$)

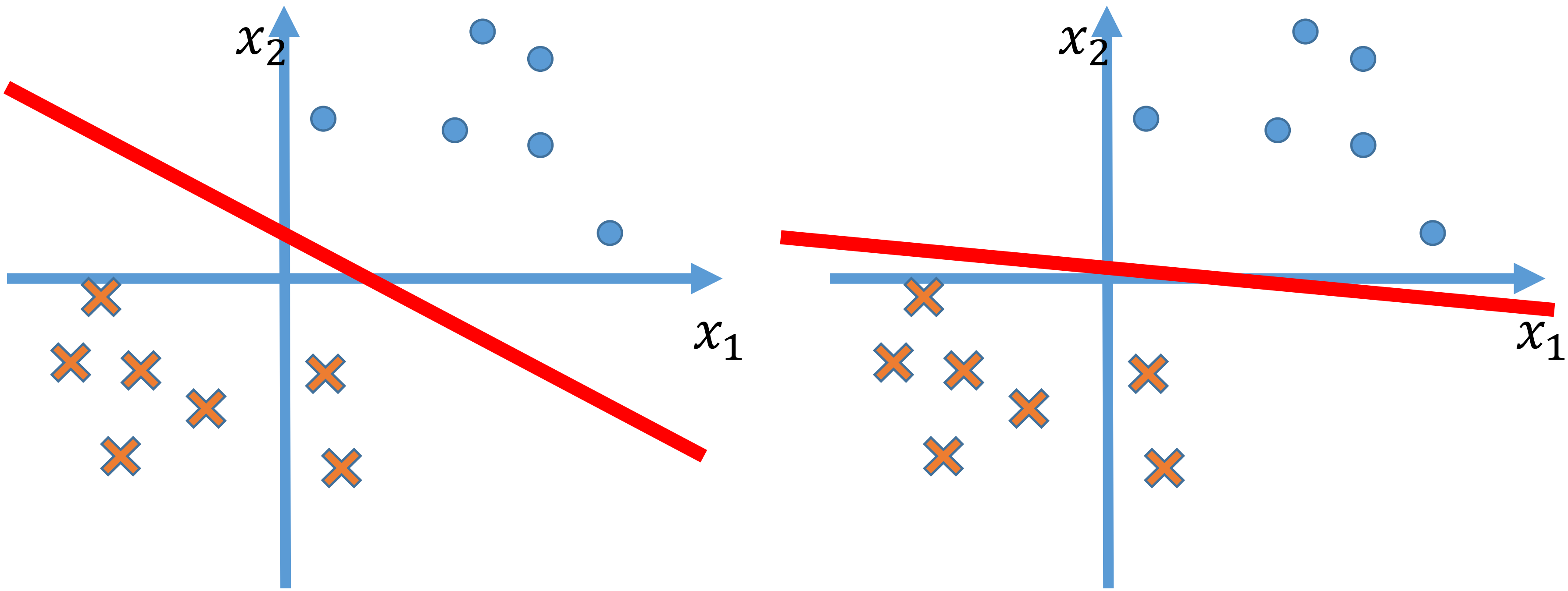
Extremely popular

If you want to use the Perceptron algorithm, use averaging

See details in. http://www.ciml.info/dl/v0_8/ciml-v0_8-ch03.pdf

3. Marginal Perceptron

- Which one is better



3. Marginal Perceptron

- Perceptron makes updates only when the prediction is incorrect

$$y_i \mathbf{w}^T \mathbf{x}_i \leq 0$$

- What if the prediction is close to being incorrect? That is, Pick a small positive ϵ and update when

$$y_i \mathbf{w}^T \mathbf{x}_i \leq \epsilon$$

- Can generalize better, but need to choose ´
  - Exercise: Why is this a good idea?

What you will learn today

- Linear models
- The Perceptron Algorithm
- Perceptron Mistake Bound
- Variants of Perceptron
- Online v.s. Batch learning

Batch Learning

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

All the data instances are presented in the training time

- Train your model

Training is usually done by minimizing training error (w/ regularization).

- Predict on the test set $\mathcal{D}' = \{(\boldsymbol{x}, y)\}$

Note: We can use online learning algorithm in the batch setting

Online Learning

- Initialize a model
- Loop

  Data instance is provided one at a time
  - Given one data point $(\boldsymbol{x}, y)$
  - Update the model
- Return the final model

Spam filtering

- Data distribution can shift
  - Spammers become smarter
- Data comes in stream
  - New spam mails are invented everyday
- Data set is huge
  - May be larger than the memory capacity

Online v.s. Batch

- Different learning protocols
  - The assumption & learning goal may different
  - E.g., batch learning assumes data are i.i.d (independent and identically distributed) while online learning may provide worst-case bound under adversarial data.
- Computation
  - Space: online learning consider one instance at a time
  - Convergence rate: some fast converged optimization method requires access to the entire dataset

Perceptron learns from mistakes
It converges when data is separable

Homework 2 is coming soon

Exercises:

- How many mistakes will the Perceptron algorithm make for disjunctions with n attributes?
  - What are R and γ?

- How many mistakes will the Perceptron algorithm make for k-disjunctions with n attributes?

- Find a sequence of examples that will force the Perceptron algorithm to make O(n) mistakes for a concept that is a k-disjunction.