Perceptron

Sriram Sankararaman

The instructor gratefully acknowledges 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.

Key issues in machine learning

- Modeling
  - How to formulate the problem ?

Supervised learning

$h(x) \rightarrow y$

- Representation
  - What is the input/output space ?
  - What is the model/ hypothesis space?

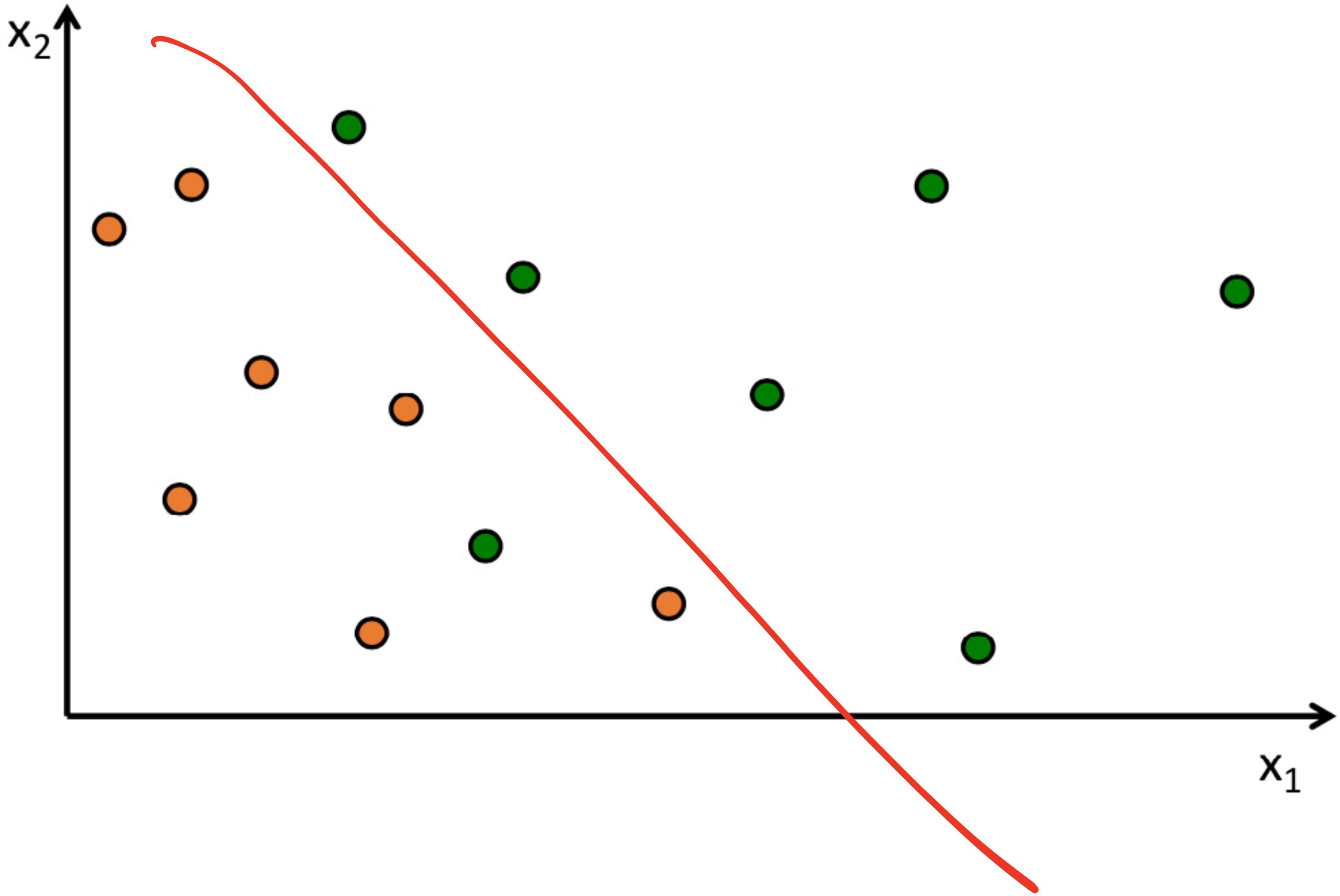
Decision Trees

Nearest neighbors

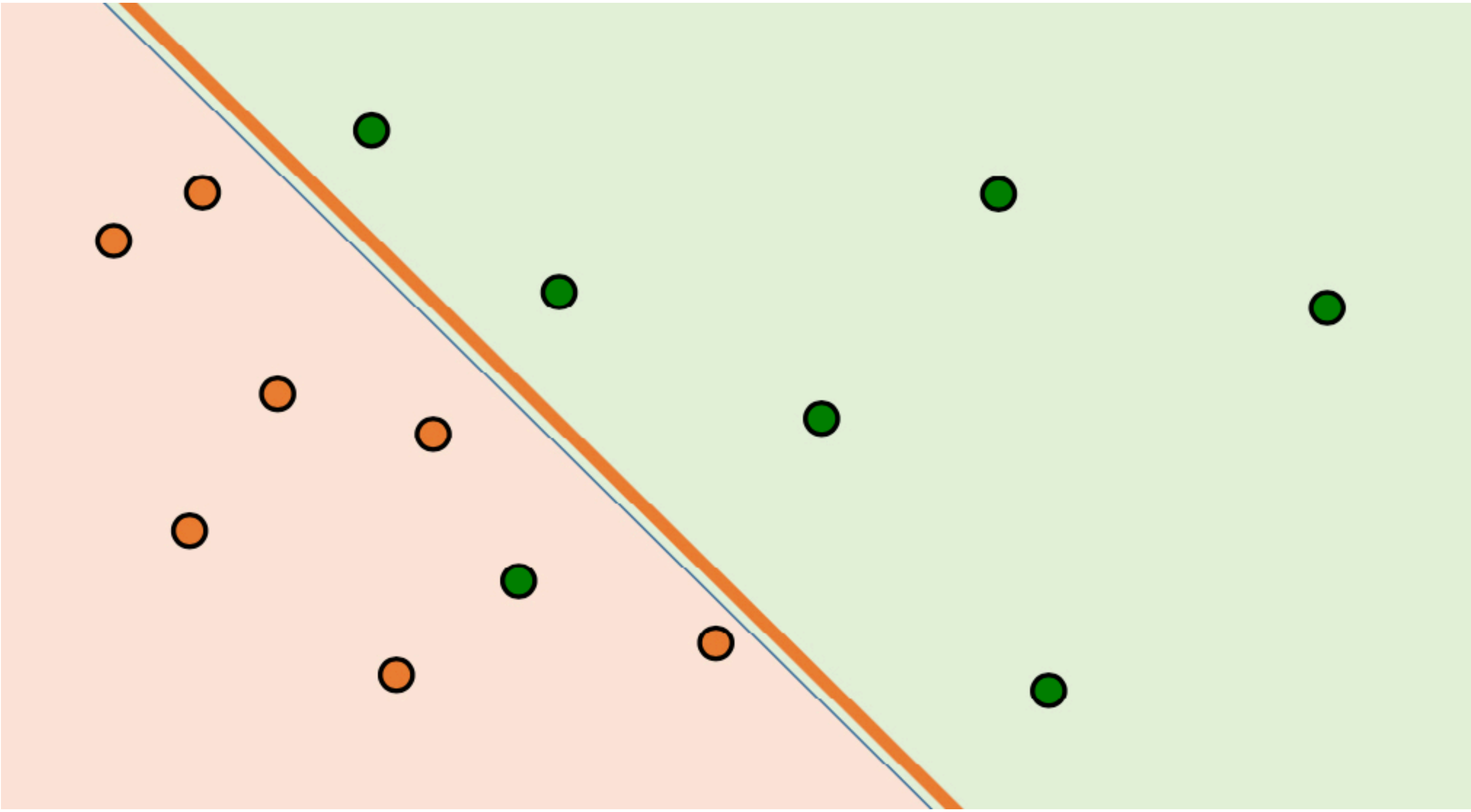
- Algorithms
  - How to find the best hypothesis?

Outline

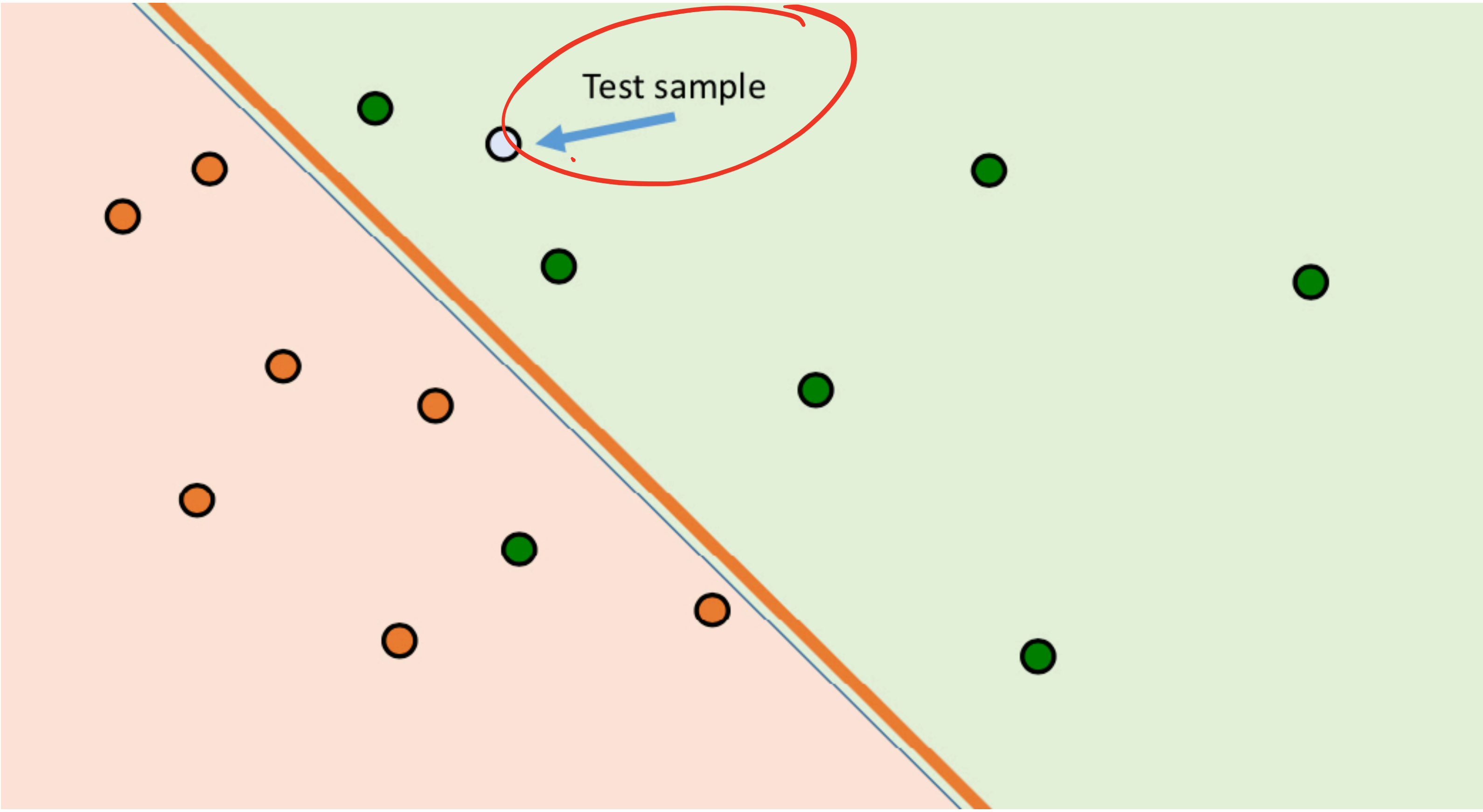
Training data for binary classification



Linear classifier



Prediction using a linear classifier



Representing a linear classifier

$w_1 x_1 + w_2 x_2 + b = 0$

$w = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$

$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$

$w^T x = w_1 x_1 + w_2 x_2$

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

$w^T(a - b) = 0$

$w^T(a-b) = w^T a - w^T b = (-b) - (-b)$

$= 0$

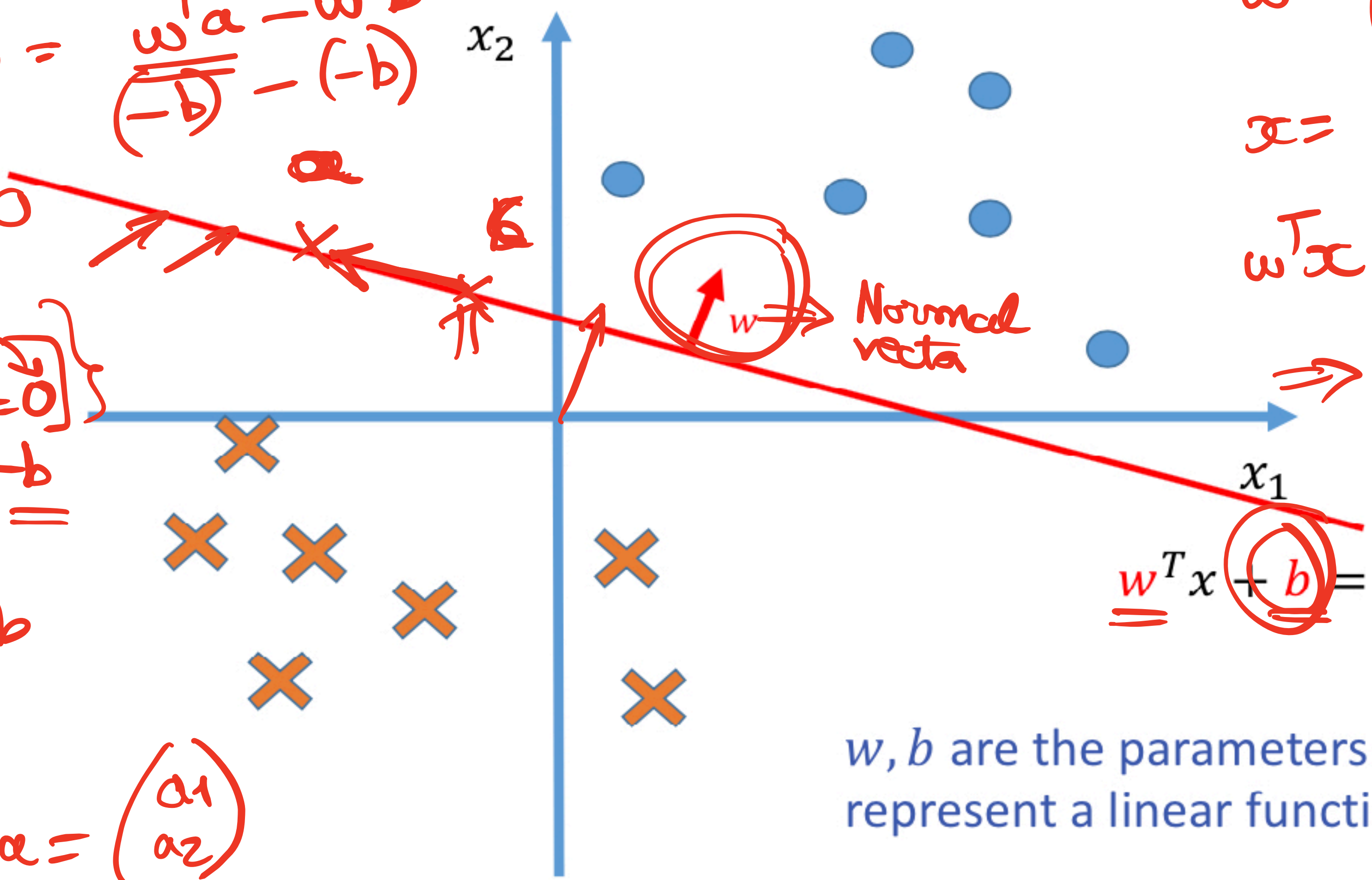
$\{w^T a + b = 0\}$

$\Rightarrow w^T a = -b$

$w^T c = -b$

$a = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$

$b = 0$

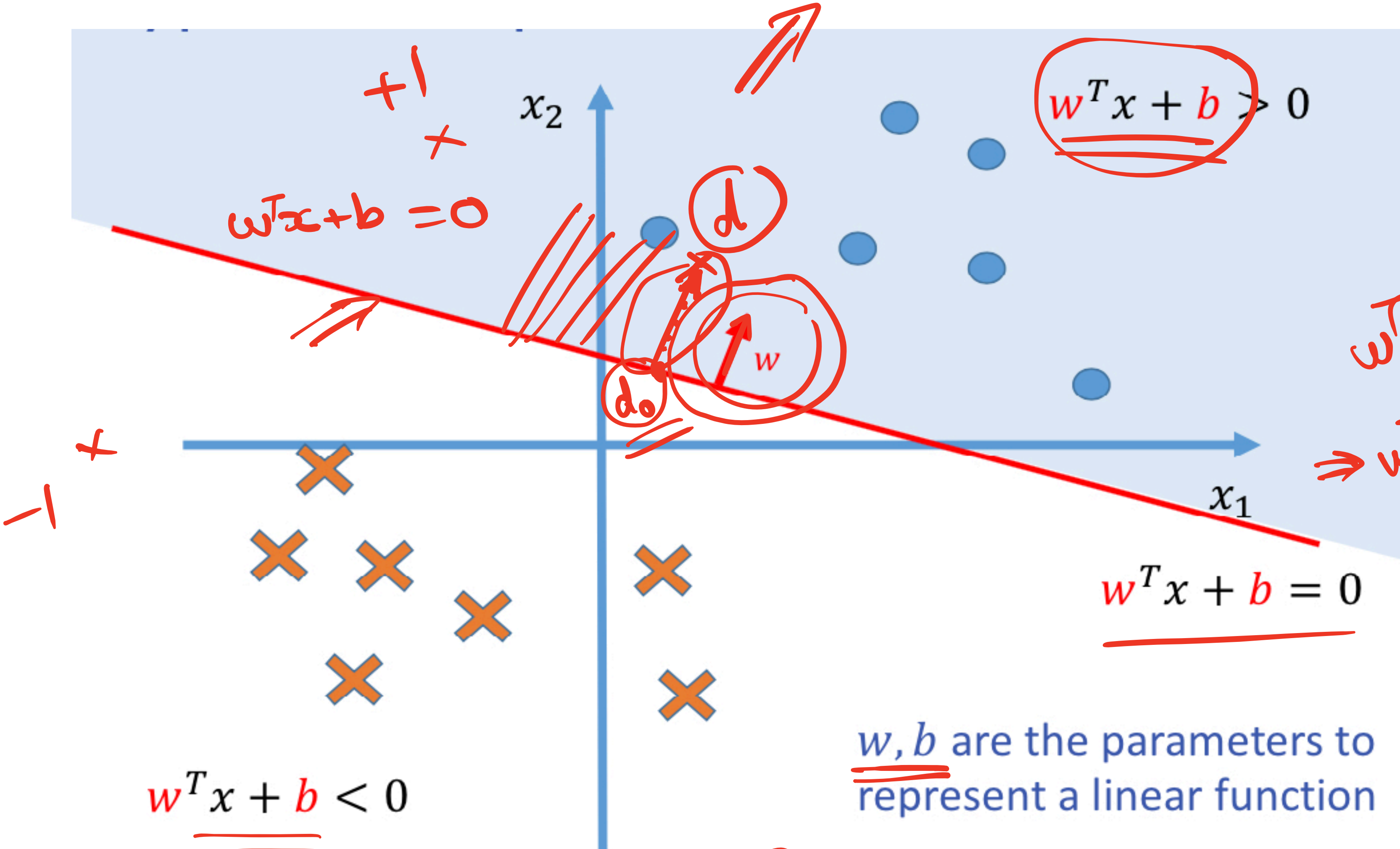


$w^T x + b = 0$

$w^T x = 0$

w, b are the parameters to represent a linear function

Representing a linear classifier



$w^T d_0 + b = 0$

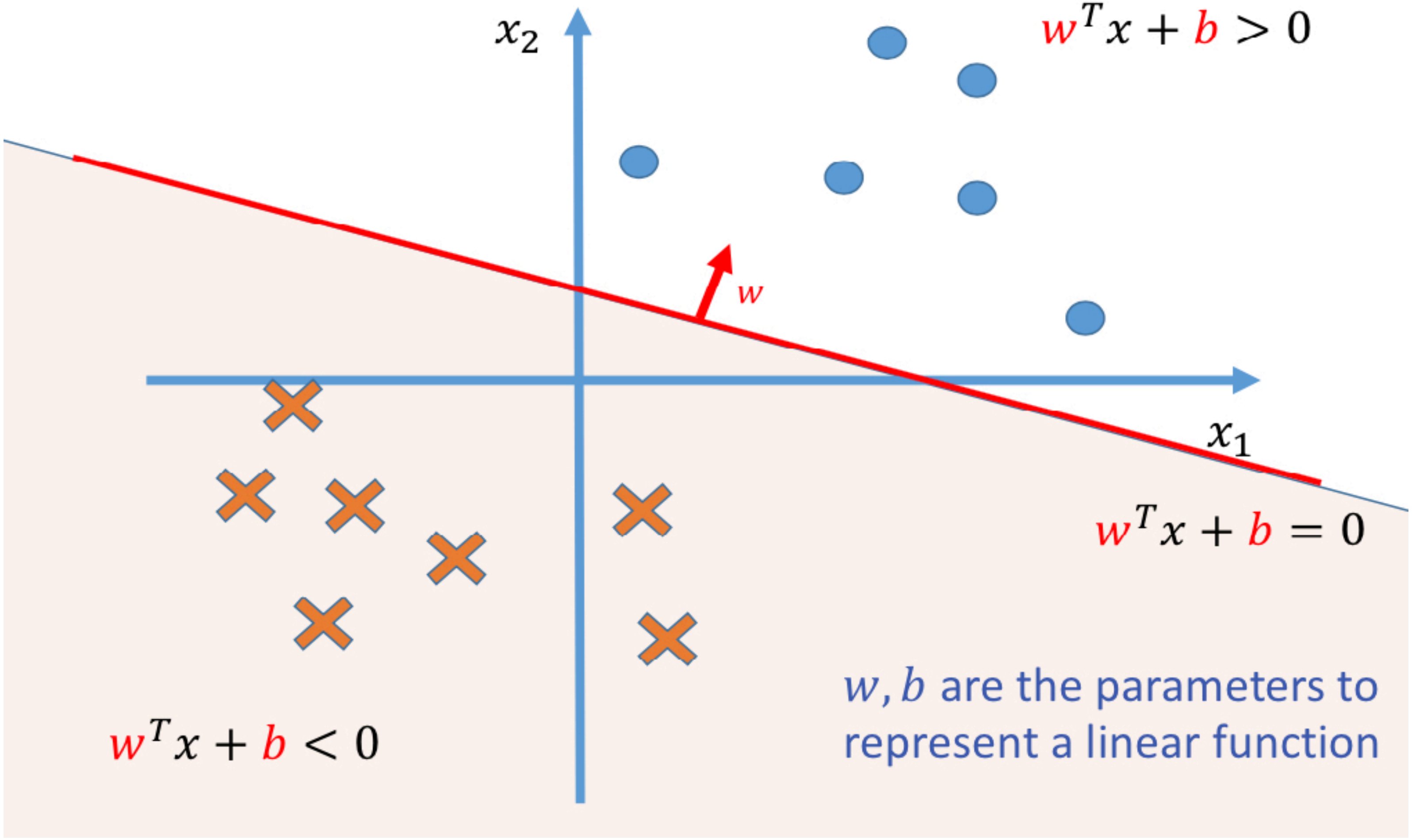
$\Rightarrow w^T d_0 = -b$

w, b are the parameters to represent a linear function

$w^T(d - d_0) > 0$

$\Rightarrow w^T d - w^T d_0 > 0 \Rightarrow w^T d + b > 0$

Classification using a linear classifier



Perceptron learning

$$\text{sign}(x) = \begin{cases} +1 & \text{if } x > 0 \\ -1 & \text{if } x < 0 \\ \text{not determined} & \text{if } x = 0 \end{cases}$$

$$x = \begin{pmatrix} x_1 \\ \vdots \\ x_D \end{pmatrix}$$

Binary classification

- Instance (feature vectors): $\boldsymbol{x} \in \mathbb{R}^D$
- Label: $y \in \{-1, +1\}$
- Model/Hypotheses:
  $H = \{h | h : \mathbb{X} \to \mathbb{Y}, h(\boldsymbol{x}) = sign(\sum_{d=1}^{D} w_d x_d + b)\}$.
- Learning goal: $\hat{y} = h(\boldsymbol{x})$
  - Learn $w_1, \ldots, w_D, b$.
  - Parameters: $w_1, \ldots, w_D, b$.
  - $\boldsymbol{w}$: weights, b: bias

$$(w_1 x_1 + w_2 x_2 + \ldots + w_D x_D + b)$$

Perceptron predict

$x = \begin{pmatrix} x_1 \\ \vdots \\ x_D \end{pmatrix}$

- Input: $\boldsymbol{x} \in \mathbb{R}^{\mathrm{D}}$, $\boldsymbol{w} \in \mathbb{R}^{\mathrm{D}}$, $b \in \mathbb{R}$.

$(w_1 x_1 + w_2 x_2 + \ldots + w_D x_D + b)$

$$a = \sum_{d=1}^{D} w_d x_d + b = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x} + b$$

$$\hat{y} = sign(a) \quad \in \{+1, -1\}$$

- Output: $\hat{y}$.
- $\sum_{d=1}^{D} w_d x_d + b = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x} + b = 0$: hyperplane in D dimensions with parameters $(\boldsymbol{w}, b)$.
- $\boldsymbol{w}$: weights, b: bias
- a: activation
- $sign(\sum_{d=1}^{D} w_d x_d + b)$: Linear Threshold Unit (LTU)

Hyperplanes through the origin

Consider $\boldsymbol{x}$ that satisfies $g(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b = 0$. These $\boldsymbol{x}$ define a hyperplane in D dimensions.

We can always write this as a hyperplane passing through the origin in $D+1$ dimensions.

$$\tilde{\boldsymbol{x}} \equiv \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_D \end{pmatrix} \quad \tilde{\boldsymbol{w}} \equiv \begin{pmatrix} b \\ w_1 \\ \vdots \\ w_D \end{pmatrix}$$

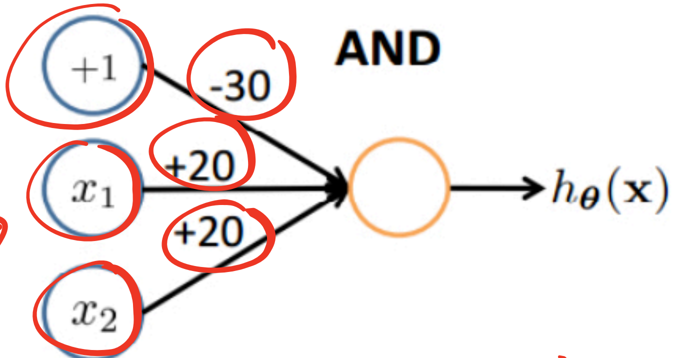
$$\begin{aligned} \tilde{g}(\tilde{\boldsymbol{x}}) &= \tilde{\boldsymbol{w}}^T\tilde{\boldsymbol{x}} = 0 \\ &= \sum_{d=1}^{D} w_d x_d + b = \\ &= g(\boldsymbol{x}) = 0 \end{aligned}$$

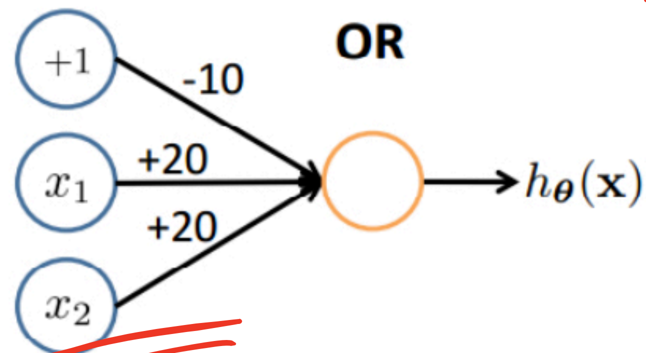
For simplicity, I may write $\tilde{\boldsymbol{w}}$ and $\tilde{\boldsymbol{x}}$ as $\boldsymbol{w}$ and $\boldsymbol{x}$ when there is no confusion

Representing Boolean functions

AND

x_1	x_2	y	
0	0	0	-1
0	1	0	-1 ✓
1	0	0	-1 ✓
1	1	1	$+1$





$a = 20x_1 + 20x_2 - 30$

$\hat{y} = \text{sign}(a)$

$x_1 = 1, x_2 = 1$

$a = 20 + 20 - 30 = 10$

$\hat{y} = +1$

$x_1 = 0, x_2 = 0$

$a = -30$

$\hat{y} = -1$

$x_1 = 0, x_2 = 1$

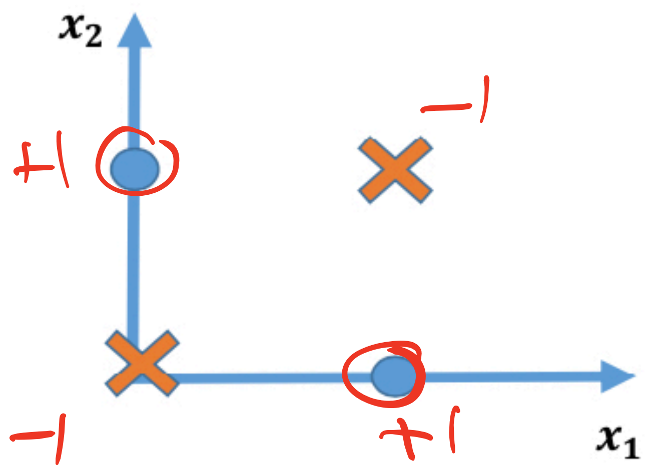
$a = 20 - 30 = -10$

$\hat{y} = -1$

Representing Boolean functions

Can linear model represent XOR?

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



Learning a linear classifier

Several algorithms

- Perceptron
- Logistic regression
- (Linear) Support Vector Machines

Based on different assumptions

Outline

Perceptron learning

Learning by making mistakes

Perceptron learning

If we have only one training example $(\boldsymbol{x}_n, y_n)$.

Assume $b = 0$. (w)

How can we change $\boldsymbol{w}$ such that

$$y_n = \text{sign}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n)$$

Two cases

- If $y_n = \text{sign}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n)$, do nothing.
- If $y_n \neq \text{sign}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n)$,

$$\boldsymbol{w}^{\text{NEW}} \leftarrow \boldsymbol{w} + y_n\boldsymbol{x}_n$$

Perceptron learning

If we have only one training example $(\boldsymbol{x}_n, y_n)$.

Assume $b = 0$.

How can we change $\boldsymbol{w}$ such that

$$y_n = \text{sign}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n)$$

Another way of saying the same thing

- $a = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n$
- If $y_n a > 0$, do nothing.
- If $y_n a \leq 0$,

$$\boldsymbol{w}^{\mathrm{NEW}} \leftarrow \boldsymbol{w} + y_n \boldsymbol{x}_n$$

$y_n = +1 \quad a > 0$

$y_n = -1 \quad a < 0$

$\Rightarrow y_n a > 0$

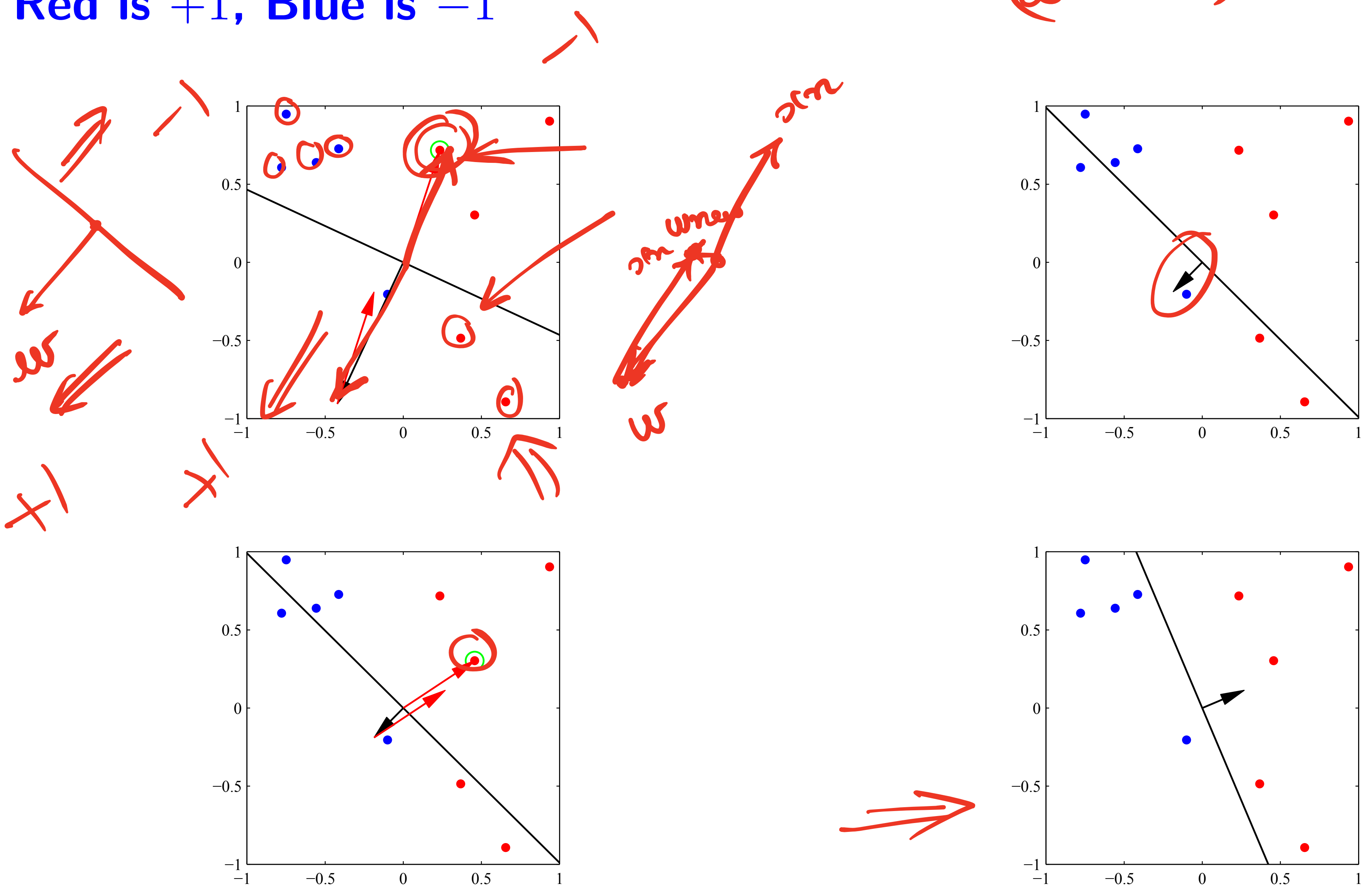
Example of perceptron update

$w_{new} \leftarrow w + y_n x_n$

$(w + x_n)$

$y_n = +1$

Red is $+1$, Blue is -1



Why would it work?

$a = w^T x_n$

If $y_n a \leq 0$, then

$$y_n(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n) \leq 0$$

Why would it work?

$y_n \in \{+1, -1\}$

$y_n^2 x_n^T x_n = x_n^T x_n \geq 0$

If $y_n a \leq 0$, then

$$y_n(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n) \leq 0$$

What would happen if we change to new $\boldsymbol{w}^{\mathrm{NEW}} = \boldsymbol{w} + y_n \boldsymbol{x}_n$?

$$y_n[(\boldsymbol{w} + y_n \boldsymbol{x}_n)^{\mathrm{T}} \boldsymbol{x}_n] = \underbrace{y_n \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n}_{\leq 0} + y_n^2 \boldsymbol{x}_n^{\mathrm{T}} \boldsymbol{x}_n$$

$$y_n(w_{new}^T x_n) = y_n((w + y_n x_n)^T x_n)$$

$$= y_n(w^T x_n + (y_n x_n)^T x_n)$$

$$= y_n(w^T x_n + y_n x_n^T x_n)$$

$$= y_n w^T x_n + y_n y_n x_n^T x_n$$

Why would it work?

If $y_n a \leq 0$, then

$$y_n(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n) \leq 0$$

What would happen if we change to new $\boldsymbol{w}^{\mathrm{NEW}} = \boldsymbol{w} + y_n\boldsymbol{x}_n$?

$$y_n[(\boldsymbol{w} + y_n\boldsymbol{x}_n)^{\mathrm{T}}\boldsymbol{x}_n] = y_n\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n + y_n^2\boldsymbol{x}_n^{\mathrm{T}}\boldsymbol{x}_n$$

We are adding a positive number, so it is possible that

$$y_n(\boldsymbol{w}^{\mathrm{NEW}^{\mathrm{T}}}\boldsymbol{x}_n) > 0$$

i.e., we are more likely to classify correctly

Perceptron learning

i) Not linearly separable

Iteratively solving one case at a time

- REPEAT
- Pick a data point $\boldsymbol{x}_n$
- Compute $a = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n$ using the *current* $\boldsymbol{w}$
- If $ay_n > 0$, do nothing. Else,

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + y_n\boldsymbol{x}_n$$

- UNTIL converged.

Perceptron training/learning

$data$ **= N samples/instances:** $= \{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_N, y_N)\}$

Algorithm 1 PerceptronTrain $(data, maxIter)$

```
w ← 0
for iter = 1 .. MaxIter do
    for (x, y) ∈ data do
        a ← wᵀx
        if ay ≤ 0 then
            w ← w + yx
        end if
    end for
end for
return w
```

Prediction: $sign(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x})$.

Design decisions

- $MaxIter$: Hyperparameter
- How to loop over the data?
  - Constant.
  - Permuting once
  - Permuting in each iteration

Properties of perceptron learning

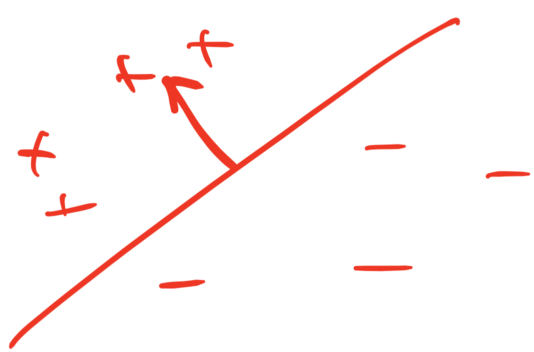
- This is an online algorithm – looks at one instance at a time.
- Does the algorithm terminate (convergence)?

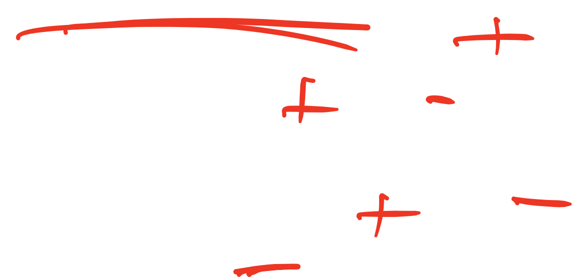
Properties of perceptron learning

- This is an online algorithm – looks at one instance at a time.
- Does the algorithm terminate (convergence)?
  - If training data is not linearly separable, the algorithm does not converge.
  - If the training data is linearly separable, the algorithm stops in a finite number of steps (converges).

Properties of perceptron learning

- This is an online algorithm – looks at one instance at a time.
- Does the algorithm terminate (convergence)?
  - If training data is not linearly separable, the algorithm does not converge.
  - If the training data is linearly separable, the algorithm stops in a finite number of steps (converges).
- How long to convergence ?
  - Depends on the difficulty of the problem (margin).





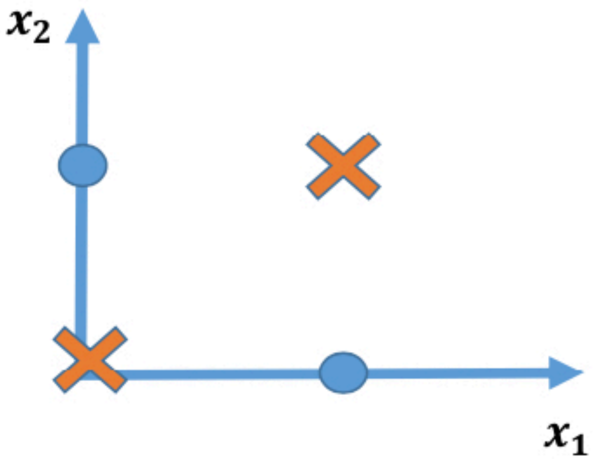
Outline

Perceptron learnability

Perceptron cannot learn what it cannot represent

- Only linearly separable functions (Minsky and Papaert 1969).
- Parity function (XOR) cannot be learned.

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



Convergence

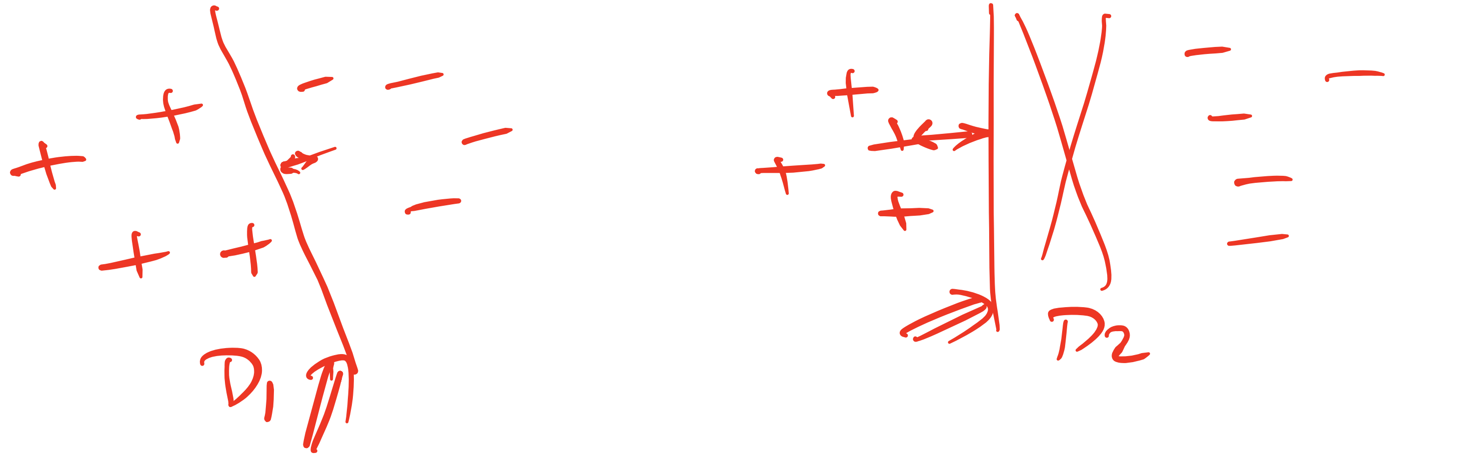
Convergence theorem

- If the data is linearly separable, the perceptron algorithm will converge after making mistakes that depend on the difficulty of the problem (margin).

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

Margin

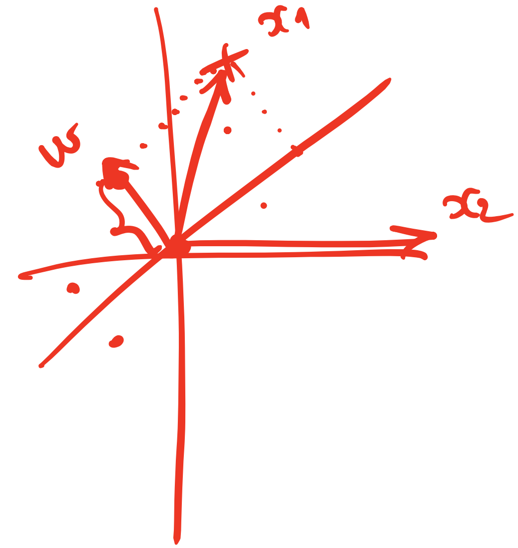


- The margin of a separating hyperplane for a dataset is the distance between the hyperplane and the data point nearest to it.
- The margin of a data set is the maximum margin possible for that dataset using any weight vector.

Margin

$$\mathcal{D} = \left\{ \begin{array}{c} (x_1, y_1) \\ \vdots \\ (x_N, y_N) \end{array} \right\}$$

$y_n w^T x_n$



$$\text{Margin}(\mathcal{D}, \boldsymbol{w}) = \begin{cases} min_{(\boldsymbol{x},y)\in\mathcal{D}} y_n \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n & \text{for a separating hyperplane } \boldsymbol{w} \\ -\infty & \text{else} \end{cases}$$

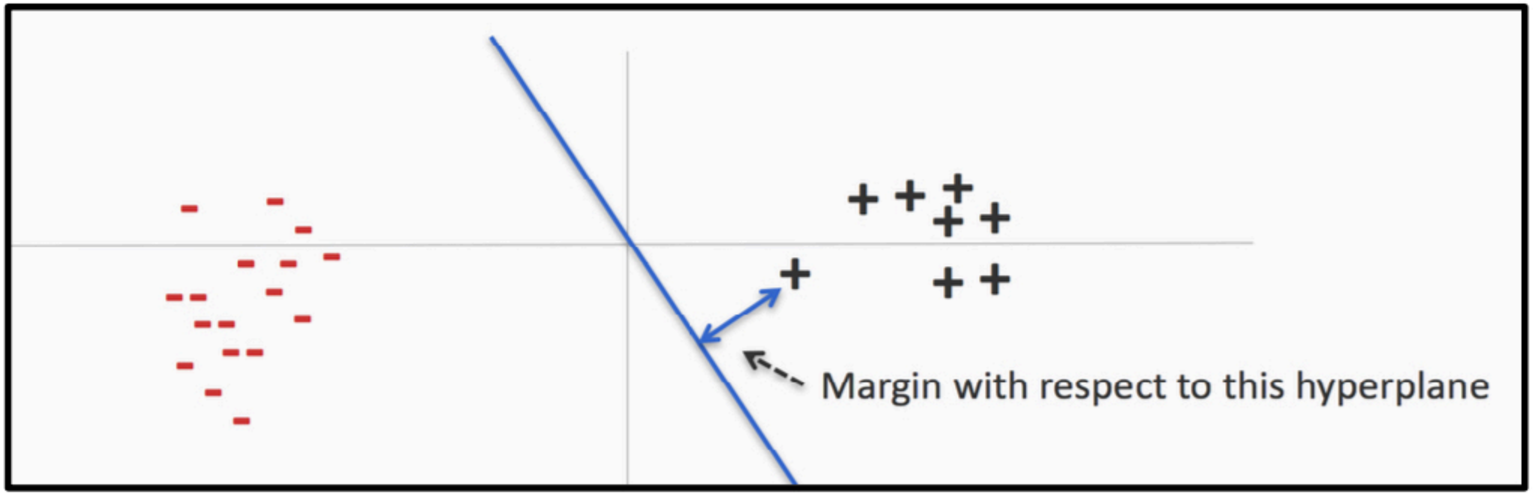
x_n

$$\text{Margin}(\mathcal{D}) = sup_{\boldsymbol{w}} \text{Margin}(\mathcal{D}, \boldsymbol{w})$$

sup = supremum $\quad x_1^T w$

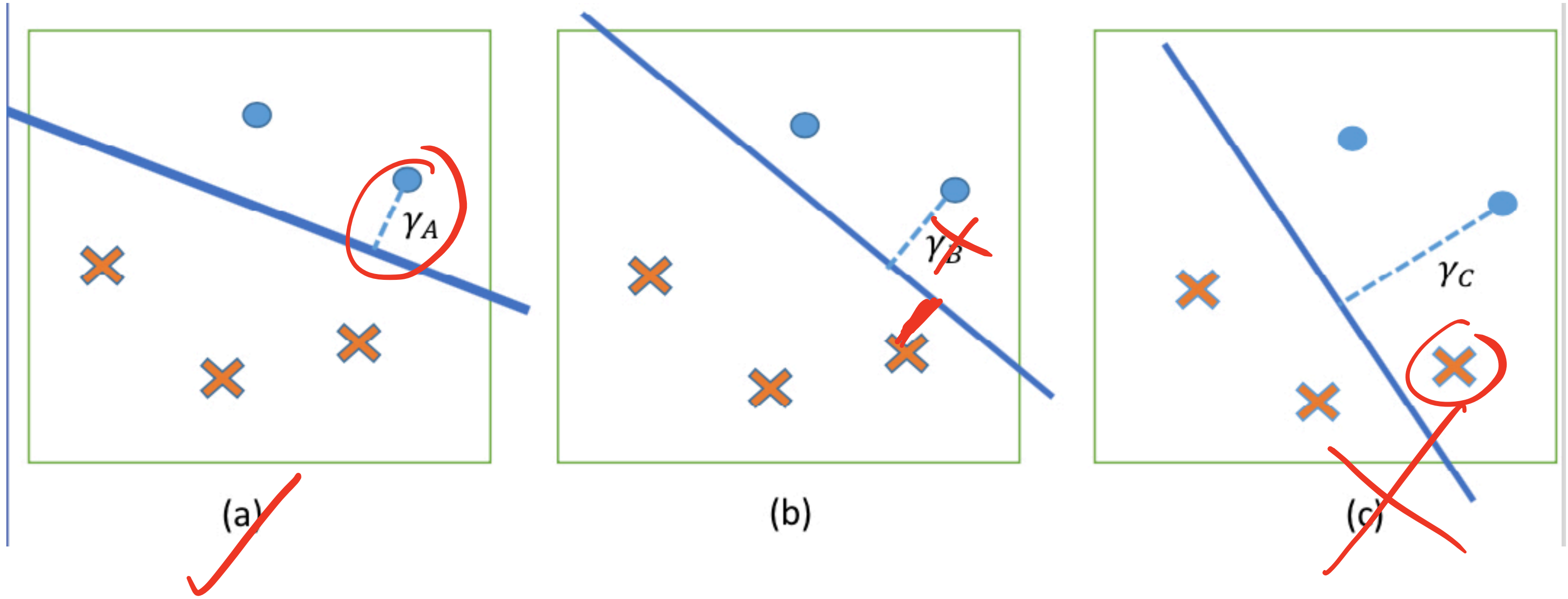
$w^T x = x^T w$

Margin



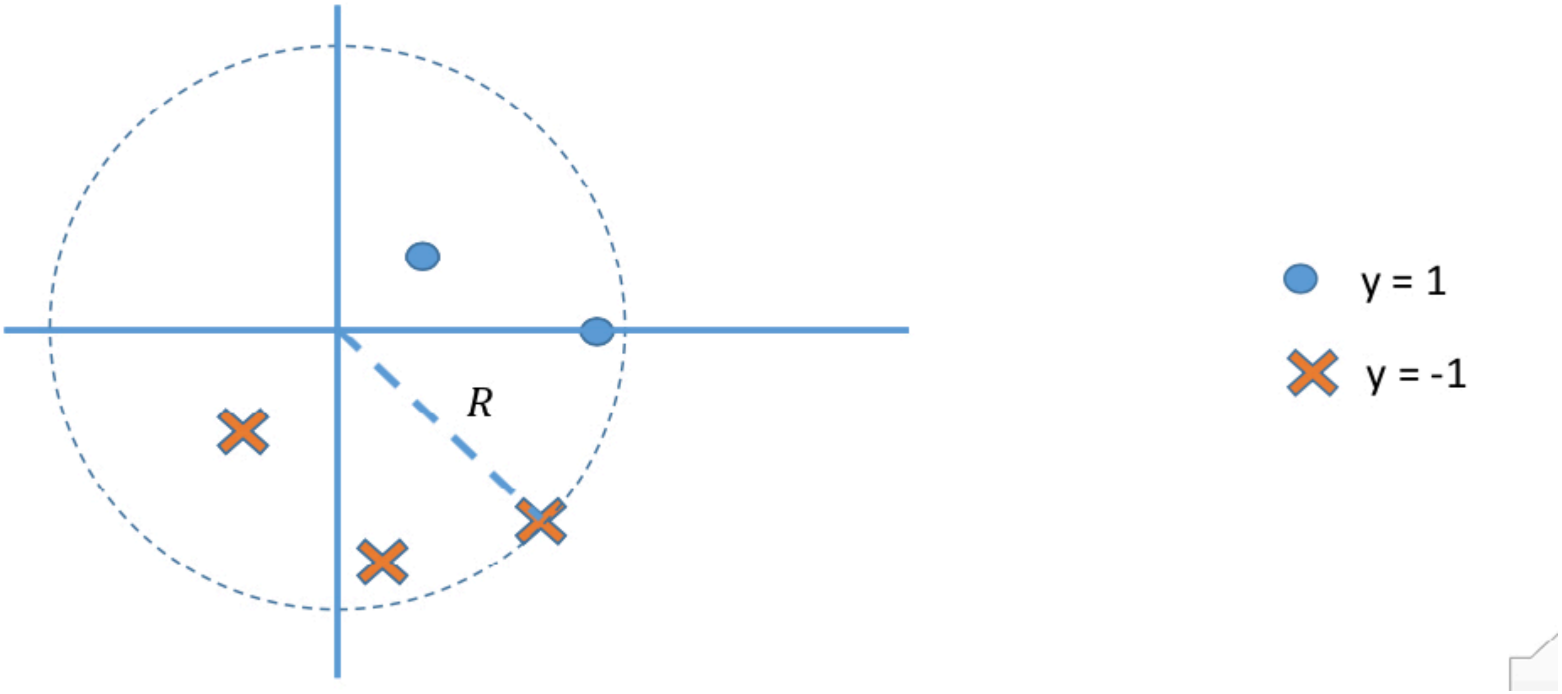
Margin

Which γ is the margin of the data?



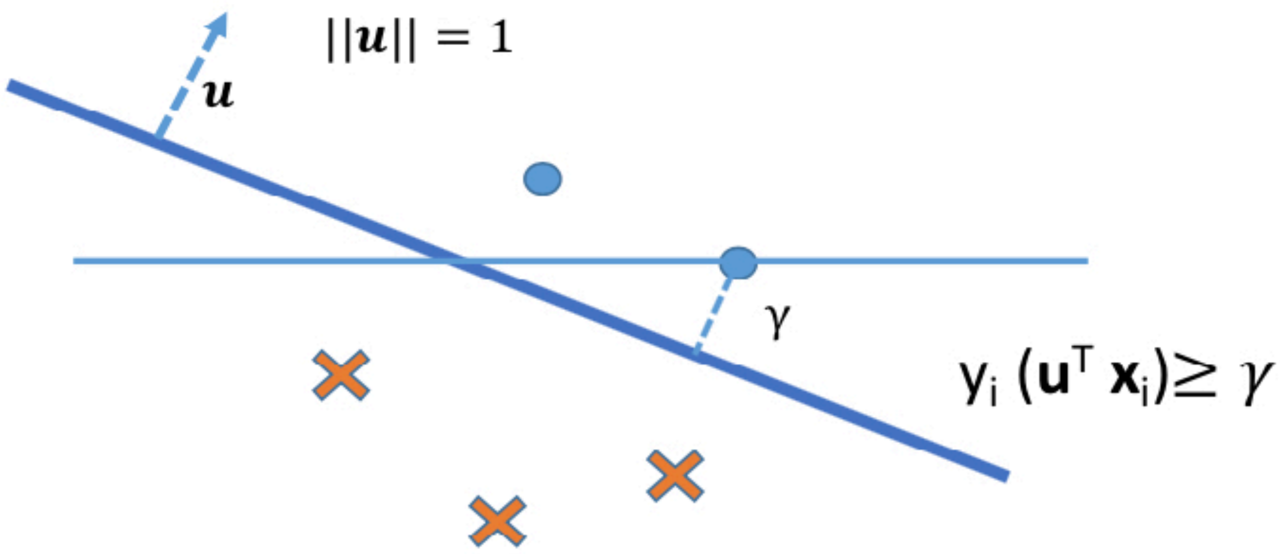
The Mistake Bound Theorem (Novikoff 1962, Block 1962)

- Let $\{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_\mathsf{N}, y_\mathsf{N})\}$ be a sequence of training examples such that $\|\boldsymbol{x}_n\|_2 \leq R$ and label $y_n \in \{-1, +1\}$.



The Mistake Bound Theorem (Novikoff 1962, Block 1962)

- Let $\{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_\mathsf{N}, y_\mathsf{N})\}$ be a sequence of training examples such that $\|\boldsymbol{x}_n\|_2 \leq R$ and label $y_n \in \{-1, +1\}$.
- Suppose there exists a unit vector $\boldsymbol{u} \in \mathbb{R}^D$ such that for some $\gamma > 0$, we have $y_n \boldsymbol{u}^\mathrm{T} \boldsymbol{x}_n \geq \gamma$.



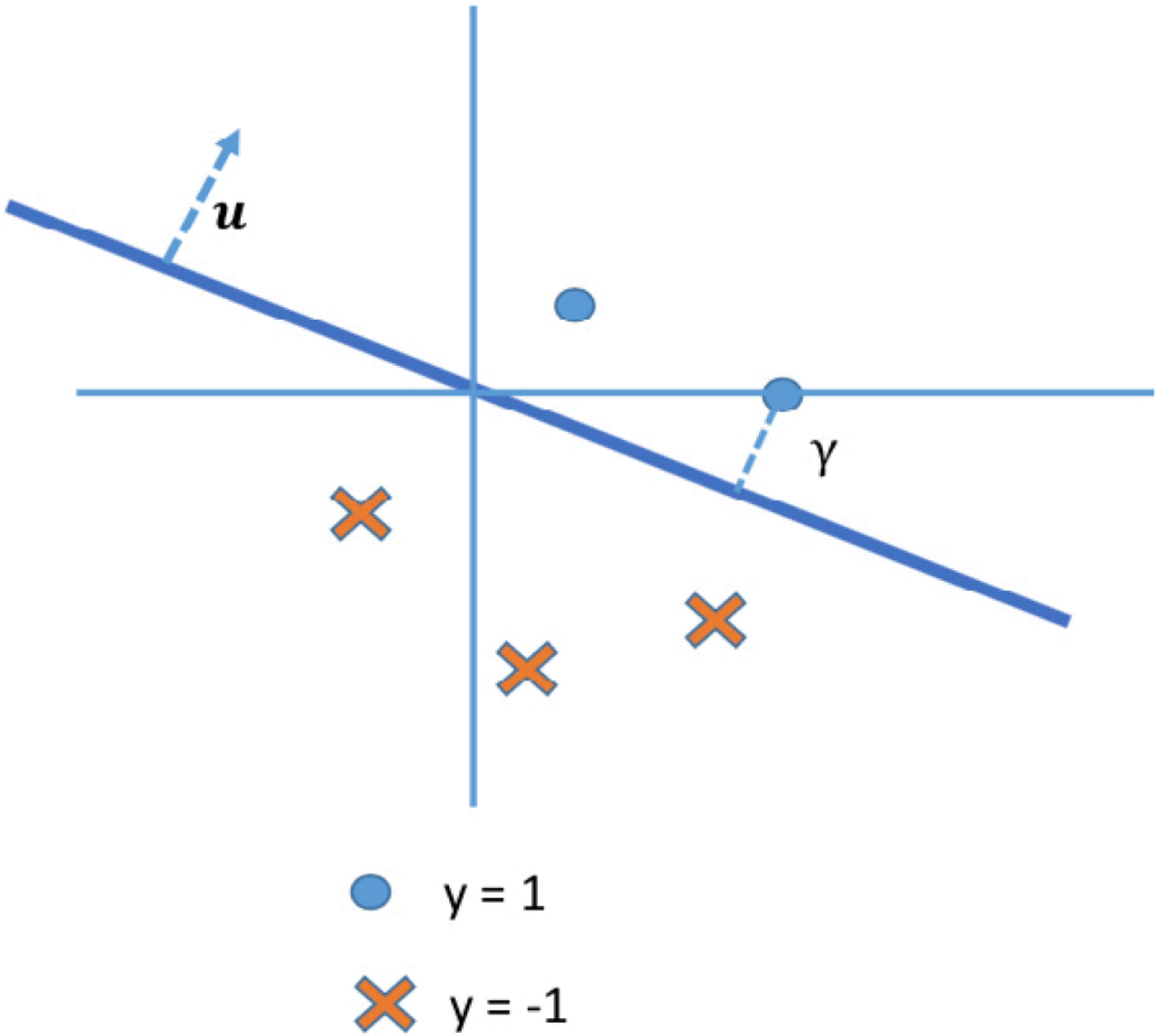
The Mistake Bound Theorem (Novikoff 1962, Block 1962)

- Let $\{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_\mathsf{N}, y_\mathsf{N})\}$ be a sequence of training examples such that $\|\boldsymbol{x}_n\|_2 \leq R$ and label $y_n \in \{-1, +1\}$.
- Suppose there exists a unit vector $\boldsymbol{u} \in \mathbb{R}^D$ such that for some $\gamma > 0$, we have $y_n \boldsymbol{u}^\mathrm{T} \boldsymbol{x}_n \geq \gamma$.
- Then the Perceptron algorithm will make at most $\frac{R^2}{\gamma^2}$ mistakes on the training sequence.

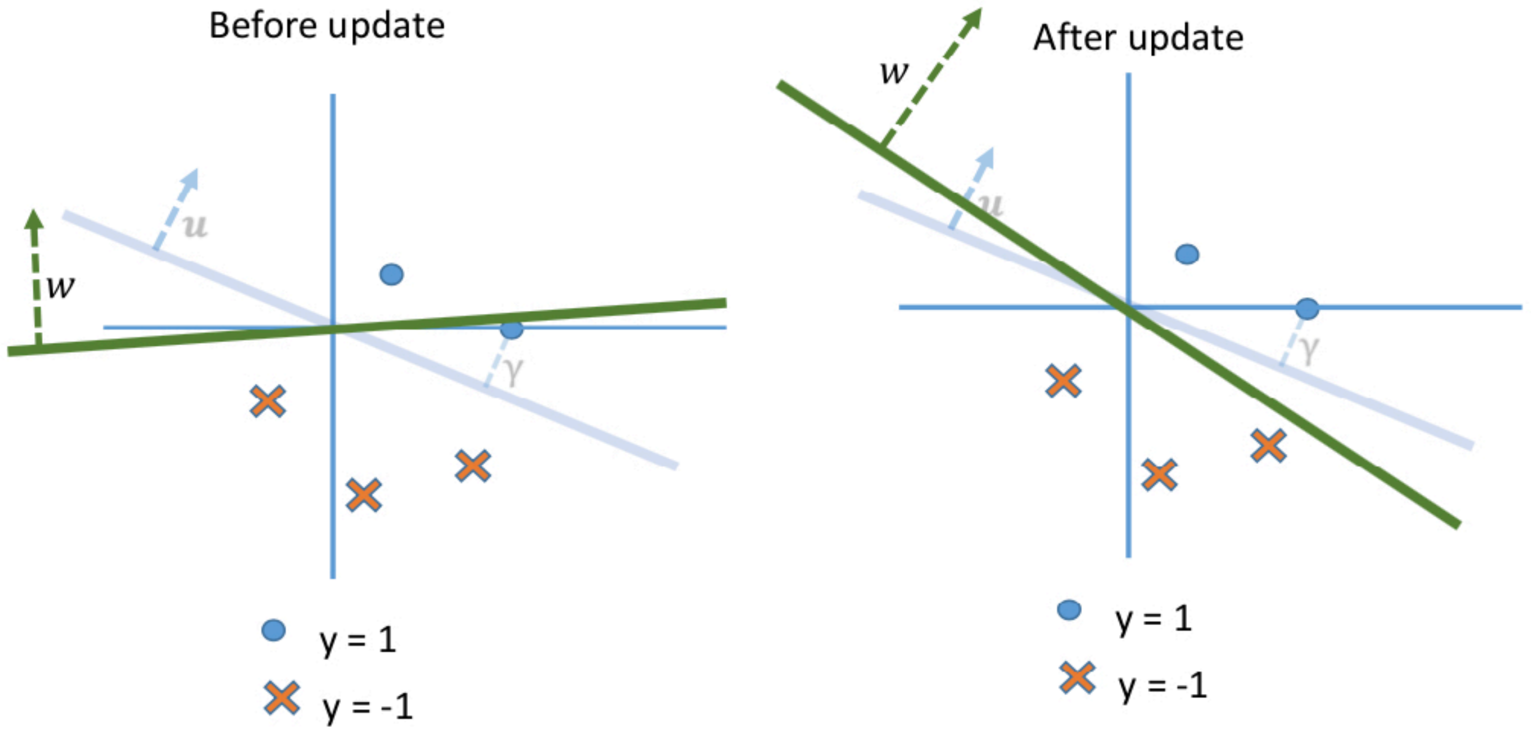
The Mistake Bound Theorem (Novikoff 1962, Block 1962)

- If the data is separable....
- then the perceptron algorithm will find a separating hyperplane after making a finite number of mistakes.

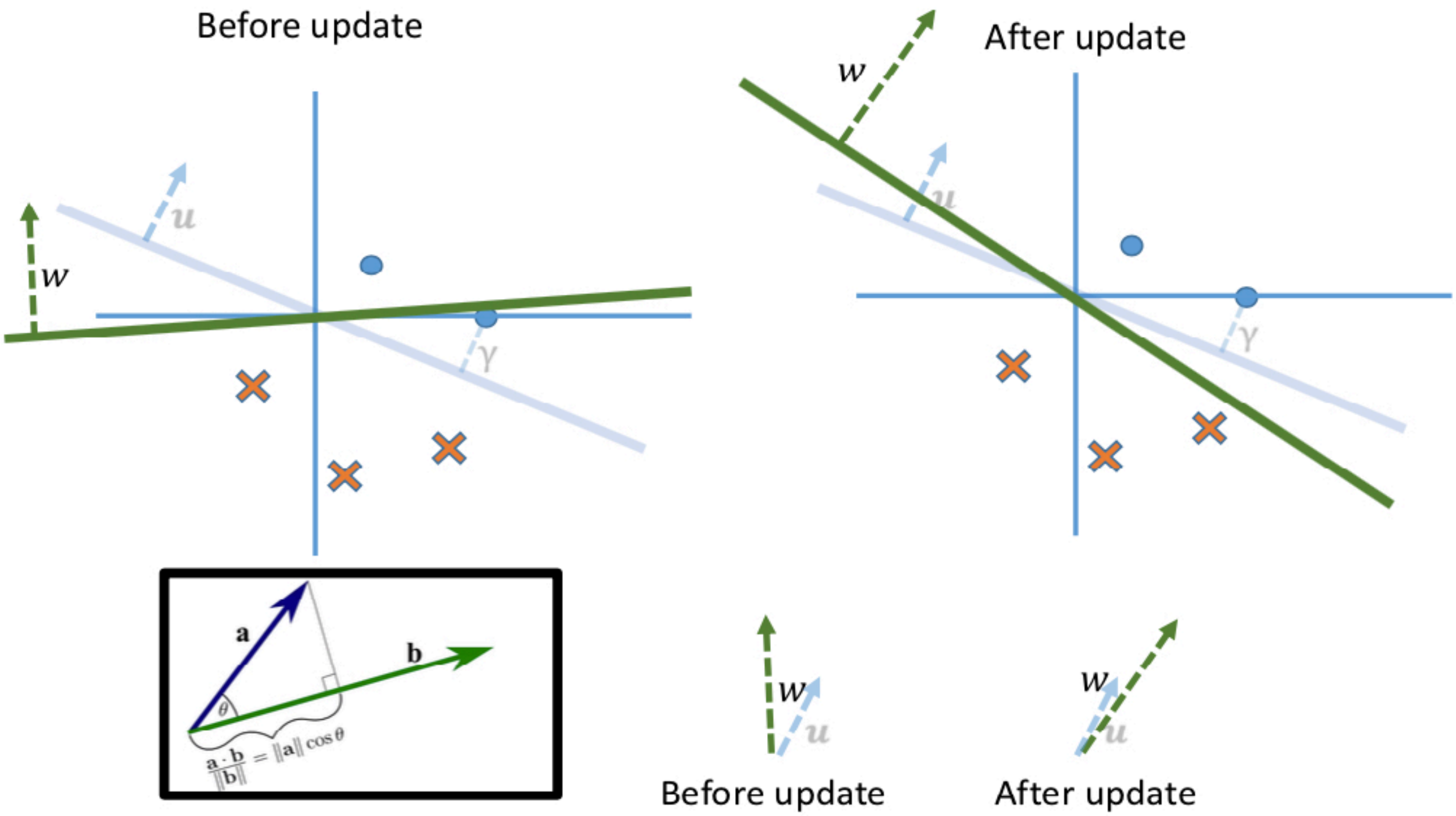
Intuition



Intuition



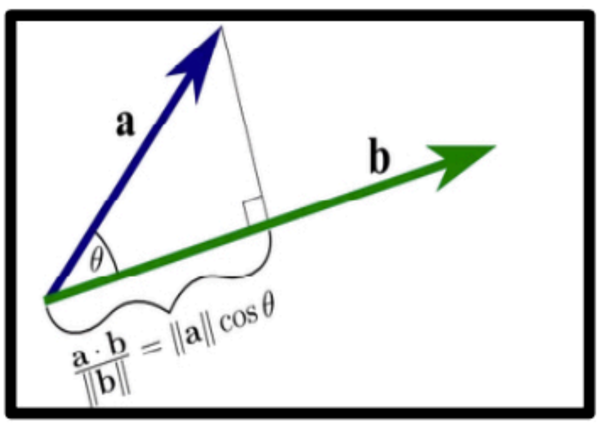
Intuition



Before update
After update
w
u
γ
a
b
θ
$\frac{\mathbf{a}\cdot\mathbf{b}}{\|\mathbf{b}\|} = \|\mathbf{a}\| \cos\theta$
Before update
After update

Intuition

- After update, $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{w}_{t+1}$ is larger than $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{w}_t$.
  - After t mistakes, $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{w}_t \geq t\gamma$.
- The size of $\|\boldsymbol{w}_{t+1}\|$ may increase but not too much.
  - After t mistakes, $\|\boldsymbol{w}_t\|^2 \leq tR^2$.



Proof (Preliminaries)

Setting

- Initial weight vector $\boldsymbol{w}_0 = \boldsymbol{0}$.
- All training examples are contained in a ball of size R.
  $\|\boldsymbol{x}_n\| \leq R$.
- The training data is separable by a margin γ using a unit vector $\boldsymbol{u}$.
  $y_n \boldsymbol{u}^{\mathrm{T}} \boldsymbol{x}_n \geq \gamma$.

Proof (1/3)

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

$$\begin{aligned}\boldsymbol{u}^{\mathrm{T}}\boldsymbol{w}_{t+1} &= \boldsymbol{u}^{\mathrm{T}}(\boldsymbol{w}_t + y_n\boldsymbol{x}_n)\\ &\geq \boldsymbol{u}^{\mathrm{T}}\boldsymbol{w}_t + \gamma\end{aligned}$$

Because $\boldsymbol{w}_0 = 0$, simple induction gives us: $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{w}_t \geq t\gamma$.

$a \leftarrow \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$
if $ay \leq 0$ **then**
  $\boldsymbol{w} \leftarrow \boldsymbol{w} + y\boldsymbol{x}$

Proof (1/3)

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

$$
\begin{aligned}
\boldsymbol{u}^{\mathrm{T}}\boldsymbol{w}_{t+1} &= \boldsymbol{u}^{\mathrm{T}}(\boldsymbol{w}_t + y_n\boldsymbol{x}_n) \\
&\geq \boldsymbol{u}^{\mathrm{T}}\boldsymbol{w}_t + \gamma
\end{aligned}
$$

Because $\boldsymbol{w}_0 = 0$, simple induction gives us: $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{w}_t \geq t\gamma$.

$a \leftarrow \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$
if $ay \leq 0$ **then**
  $\boldsymbol{w} \leftarrow \boldsymbol{w} + y\boldsymbol{x}$

The inner product between the true underlying model and the current model is non-decreasing after each update. This could be because the directions of $\boldsymbol{w}$ and $\boldsymbol{u}$ align or because the length of $\boldsymbol{w}$ increases.

Proof (2/3)

Claim 2: After t mistakes, $\|w_t\|^2 \leq tR^2$

$$\begin{aligned}\|\boldsymbol{w}_{t+1}\|^2 &= \|\boldsymbol{w}_t + y_n \boldsymbol{x}_n\|^2 \\ &= \|\boldsymbol{w}_t\|^2 + 2\boldsymbol{w}_t^{\mathrm{T}}(y_n \boldsymbol{x}_n) + \|y_n \boldsymbol{x}_n\|^2\end{aligned}$$

Proof (2/3)

Claim 2: After t mistakes, $\|w_t\|^2 \leq tR^2$

$$\begin{aligned}\|\boldsymbol{w}_{t+1}\|^2 &= \|\boldsymbol{w}_t + y_n \boldsymbol{x}_n\|^2 \\ &= \|\boldsymbol{w}_t\|^2 + 2\boldsymbol{w}_t^{\mathrm{T}}(y_n \boldsymbol{x}_n) + \|y_n \boldsymbol{x}_n\|^2 \\ &= \|\boldsymbol{w}_t\|^2 + 2y_n(\boldsymbol{w}_t^{\mathrm{T}} \boldsymbol{x}_n) + y_n^2 \|\boldsymbol{x}_n\|^2\end{aligned}$$

$a \leftarrow \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}$
if $ay \leq 0$ **then**
  $\boldsymbol{w} \leftarrow \boldsymbol{w} + y\boldsymbol{x}$

Proof (2/3)

Claim 2: After t mistakes, $\|w_t\|^2 \leq tR^2$

$$\begin{aligned}\|\boldsymbol{w}_{t+1}\|^2 &= \|\boldsymbol{w}_t + y_n \boldsymbol{x}_n\|^2 \\ &= \|\boldsymbol{w}_t\|^2 + 2\boldsymbol{w}_t^{\mathrm{T}}(y_n \boldsymbol{x}_n) + \|y_n \boldsymbol{x}_n\|^2 \\ &\leq \|\boldsymbol{w}_t\|^2 + R^2\end{aligned}$$

Because $\boldsymbol{w}_0 = 0$, simple induction gives us: $\|\boldsymbol{w}_t\|^2 \leq tR^2$.

Proof (3/3)

What we know

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

The inner product between the true underlying model and the current model is non-decreasing after each update. This could be because the directions of $\boldsymbol{w}$ and $\boldsymbol{u}$ align or because the length of $\boldsymbol{w}$ increases.

But the length of $\boldsymbol{w}$ does not increase too much!.

Proof (3/3)

What we know

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

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

Proof (3/3)

What we know

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

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

Proof (3/3)

What we know

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

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

Proof (3/3)

What we know

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

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

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

Bounds the total number of mistakes!

Beyond the separable case

- Good news
  - Perceptron makes no assumptions about the data, could be even adversarial.
  - After a fixed number of mistakes, you are done. Do not need to see any more data.
- Bad news
  - Real world data is often not linearly separable.

Outline

Voting and averaging

- Vanilla perceptron returns final weight vector.
- Might lose good weight vectors that were learned during training.
- Aggregating the models (or weight vectors) seen during training may give better results (especially when data is not separable).

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

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 training/learning

$data$ **= N samples/instances:** $= \{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_\mathsf{N}, y_\mathsf{N})\}$

Algorithm 2 AveragedPerceptronTrain $(data, maxIter)$

```
w ← 0.  μ ← 0.
for iter = 1 ... MaxIter do
    for (x, y) ∈ data do
        a ← wᵀx
        if ay ≤ 0 then
            w ← w + yx
        end if
        μ ← μ + w
    end for
end for
return μ
```

Prediction: $sign(\boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{x})$.

Perceptron

- Extensions
  - Voting
  - Averaging
- Limitations
  - Linear separability
- Interpreting the importance of features
  - The values of weight w_d tells us the importance of feature x_d.

Outline

Summary

- You should now be able to understand the differences between decision trees, perceptrons and nearest neighbors.
- Given data, use training, development and test splits (or cross-validation).
- Use training and development to tune hyperparameters that trades off overfitting and underfitting.
- Use test to get an estimate of generalization or accuracy on unseen data.