



# 10-301/10-601 Introduction to Machine Learning

Machine Learning Department  
School of Computer Science  
Carnegie Mellon University

# Perceptron

Matt Gormley  
Lecture 6  
Feb. 3, 2023

# Q&A

**Q:** How do we define a distance function when the features are categorical (e.g. weather takes values {sunny, rainy, overcast})?

**A:** Step 1: Convert from categorical attributes to numeric features (e.g. binary)

Step 2: Select an appropriate distance function (e.g. Hamming distance)

$$d(\mathbf{x}, \mathbf{x}') = \sum_{d=1}^D \mathbb{1}(x_d \neq x'_d)$$

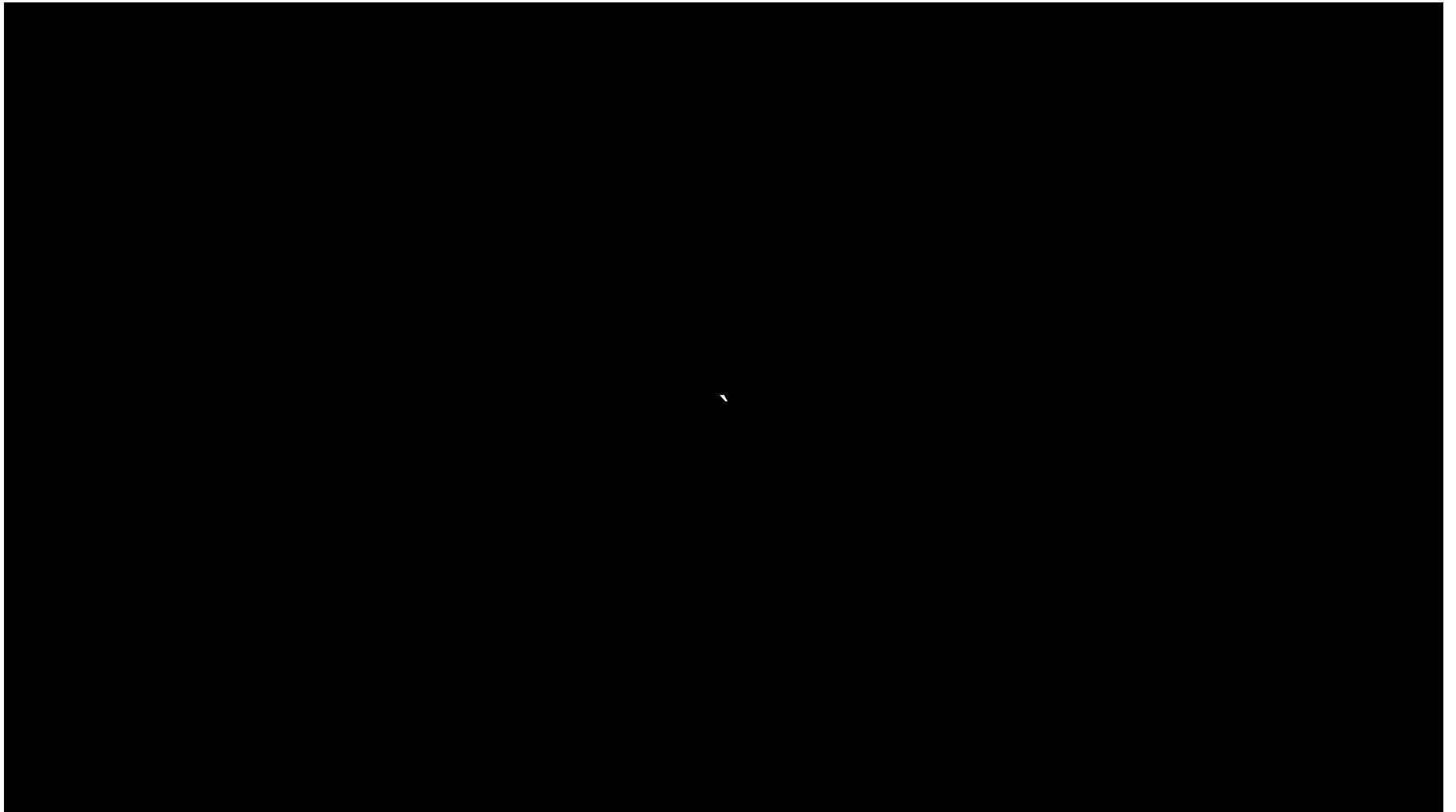
# Reminders

- **Homework 2: Decision Trees**
  - Out: Wed, Jan. 25
  - Due: Fri, Feb. 3 at 11:59pm
- **Recitation: HW3**
  - Wed, Feb. 8
- **Homework 3: KNN, Perceptron, Lin.Reg.**
  - Out: Fri, Feb. 3
  - Due: Fri, Feb. 10 at 11:59pm
  - (only two grace/late days permitted)

# **THE PERCEPTRON ALGORITHM**

# Perceptron: History

Imagine you are trying to build a new machine learning technique... your name is Frank Rosenblatt... and the year is 1957



# Perceptron: History

Imagine you are trying to build a new machine learning technique... your name is Frank Rosenblatt... and the year is 1957

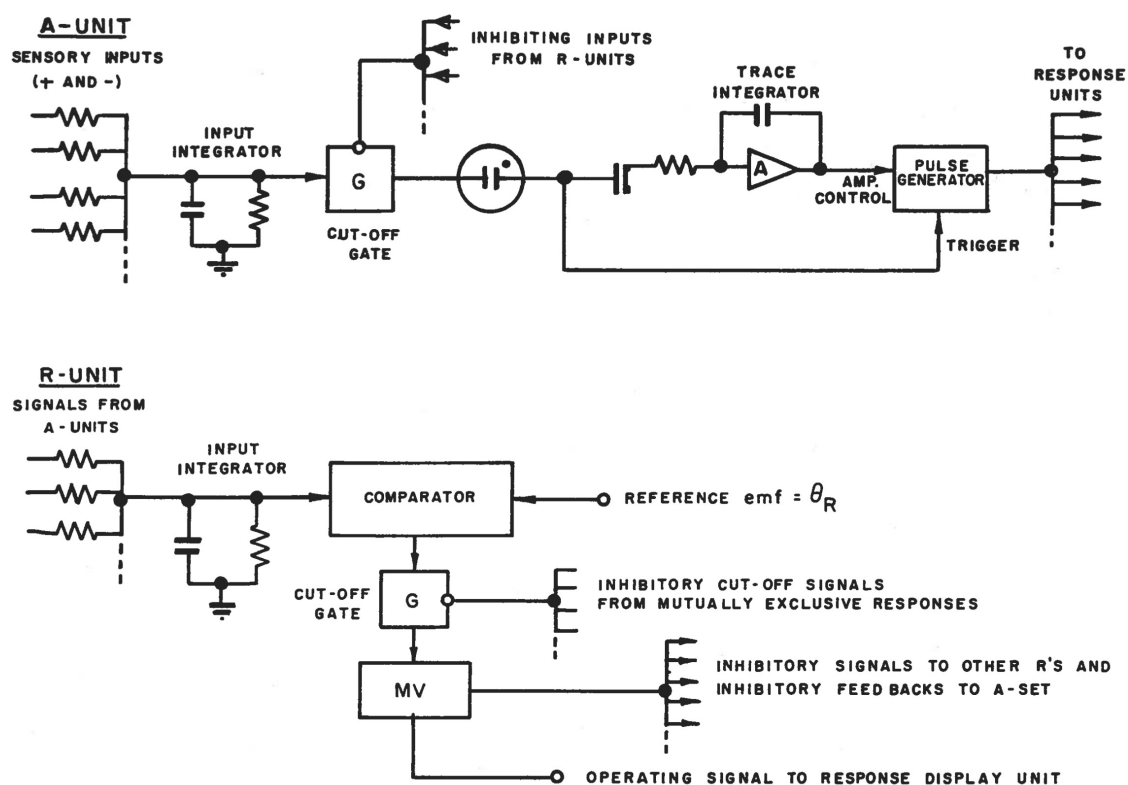
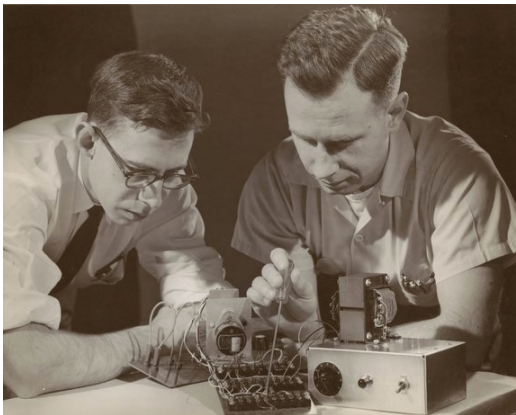


FIGURE 5  
DESIGN OF TYPICAL UNITS

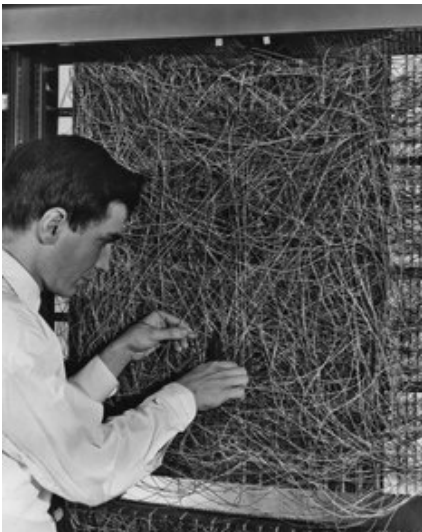
# Perceptron: History

Imagine you are trying to build a new machine learning technique... your name is Frank Rosenblatt... and the year is 1957



*The New Yorker*, December 6, 1958 P. 44

Talk story about the perceptron, a new electronic brain which hasn't been built, but which has been successfully simulated on the I.B.M. 704. Talk with Dr. Frank Rosenblatt, of the Cornell Aeronautical Laboratory, who is one of the two men who developed the prodigy; the other man is Dr. Marshall C. Yovits, of the Office of Naval Research, in Washington. Dr. Rosenblatt defined the perceptron as the first non-biological object which will achieve an organization o its external environment in a meaningful way. It interacts with its environment, forming concepts that have not been made ready for it by a human agent. If a triangle is held up, the perceptron's eye picks up the image & conveys it along a random succession of lines to the response units, where the image is registered. It can tell the difference betw. a cat and a dog, although it wouldn't be able to tell whether the dog was to theleft or right of the cat. Right now it is of no practical use, Dr. Rosenblatt conceded, but he said that one day it might be useful to send one into outer space to take in impressions for us.



# Linear Models for Classification

Key idea: Try to learn this hyperplane directly

Looking ahead:

- We'll see a number of commonly used Linear Classifiers
- These include:
  - Perceptron
  - Logistic Regression
  - Naïve Bayes (under certain conditions)
  - Support Vector Machines

Directly modeling the hyperplane would use a decision function:

$$h(\mathbf{x}) = \text{sign}(\boldsymbol{\theta}^T \mathbf{x})$$

for:

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



# **GEOMETRY & VECTORS**

# Geometry Warm-up

## In-Class Exercise

Draw a picture of the region corresponding to:

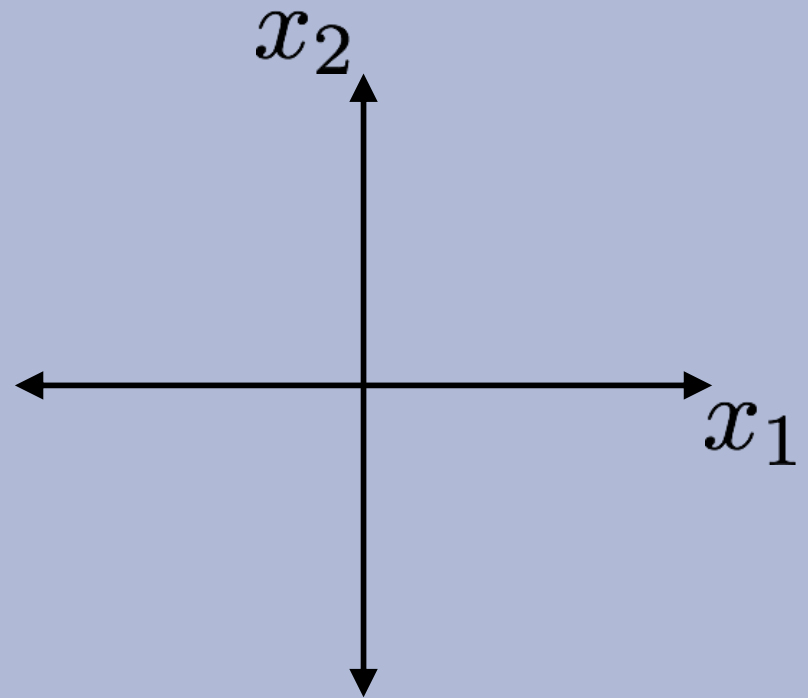
$$w_1x_1 + w_2x_2 + b > 0$$

$$\text{where } w_1 = 2, w_2 = 3, b = 6$$

Draw the vector

$$\mathbf{w} = [w_1, w_2]$$

## Answer Here:



# Linear Algebra Review

- Notation: in this class vectors will be assumed to be column vectors by default, i.e.,

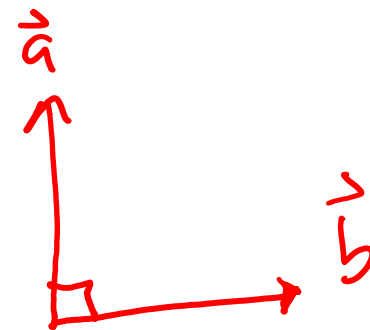
$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_D \end{bmatrix} \text{ and } \mathbf{a}^T = [a_1 \quad a_2 \quad \cdots \quad a_D]$$

- The dot product between two  $D$ -dimensional vectors is

$$\mathbf{a} \cdot \mathbf{b} = \mathbf{a}^T \mathbf{b} = [a_1 \quad a_2 \quad \cdots \quad a_D] \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_D \end{bmatrix} = \sum_{d=1}^D a_d b_d$$

- The  $L_2$ -norm of  $\mathbf{a} = \|\mathbf{a}\|_2 = \sqrt{\mathbf{a}^T \mathbf{a}}$
- Two vectors are *orthogonal* iff

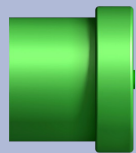
$$\mathbf{a}^T \mathbf{b} = 0$$



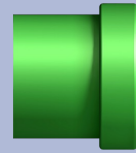
# Vector Projection

Question: Q1

Which of the following is the projection of a vector  $\mathbf{a}$  onto a vector  $\mathbf{b}$ ?



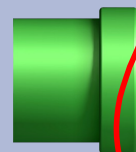
A.  $\frac{\mathbf{a}^T \mathbf{b}}{\mathbf{b}} \mathbf{a}$



D.  $\frac{(\mathbf{a} \cdot \mathbf{b})}{\|\mathbf{b}\|_2} \mathbf{b}$



~~B.  $\frac{\mathbf{a} \cdot \mathbf{b}}{\mathbf{a}^T \mathbf{b}} \mathbf{a}$~~  toxic



E.  $\frac{(\mathbf{a}^T \mathbf{b})}{\|\mathbf{b}\|_2^2} \mathbf{b}$  40%



C.  $\frac{(\mathbf{a}^T \mathbf{b})}{\|\mathbf{b}\|_2} \mathbf{b}$  36%



F.  $\frac{(\mathbf{a}^T \mathbf{b})^2}{\|\mathbf{b}\|_2} \mathbf{b}$

# Linear Decision Boundaries

- In 2 dimensions,  $w_1x_1 + w_2x_2 + b = 0$  defines a *line*
- In 3 dimensions,  $w_1x_1 + w_2x_2 + w_3x_3 + b = 0$  defines a *plane*
- In 4+ dimensions,  $\mathbf{w}^T \mathbf{x} + b = 0$  defines a *hyperplane*
  - The vector  $\mathbf{w}$  is always orthogonal to this hyperplane and always points in the direction where  $\mathbf{w}^T \mathbf{x} + b > 0$ !
- A hyperplane creates two *halfspaces*:
  - $\mathcal{S}_+ = \{\mathbf{x}: \mathbf{w}^T \mathbf{x} + b > 0\}$  or all  $\mathbf{x}$  s.t.  $\mathbf{w}^T \mathbf{x} + b$  is positive
  - $\mathcal{S}_- = \{\mathbf{x}: \mathbf{w}^T \mathbf{x} + b < 0\}$  or all  $\mathbf{x}$  s.t.  $\mathbf{w}^T \mathbf{x} + b$  is negative

# Linear Models for Classification

Key idea: Try to learn this hyperplane directly

Looking ahead:

- We'll see a number of commonly used Linear Classifiers
- These include:
  - Perceptron
  - Logistic Regression
  - Naïve Bayes (under certain conditions)
  - Support Vector Machines

Directly modeling the hyperplane would use a decision function:

$$h(\mathbf{x}) = \text{sign}(\boldsymbol{\theta}^T \mathbf{x})$$

for:

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

# ONLINE LEARNING

# Online Learning

- **Batch Learning:** So far, we've been learning in the *batch* setting, where we have access to the entire training dataset at once
- **Online Learning:** A common alternative is the *online* setting, where examples arrive gradually and we learn continuously
- **Examples of online learning:**
  1. **Stock market** prediction (what will the value of Alphabet Inc. be tomorrow?)
  2. **Email** classification (distribution of both spam and regular mail changes over time, but the target function stays fixed - last year's spam still looks like spam)
  3. **Recommendation** systems. Examples: recommending movies; predicting whether a user will be interested in a new news article
  4. **Ad placement** in a new market



# Online Learning

**For  $i = 1, 2, 3, \dots$ :**

- **Receive** an unlabeled instance  $\mathbf{x}^{(i)}$
- **Predict**  $y' = h_{\theta}(\mathbf{x}^{(i)})$
- **Receive** true label  $y^{(i)}$
- **Suffer loss** if we made a mistake,  $y' \neq y^{(i)}$
- **Update** parameters  $\theta$

**Goal:**

- **Minimize** the number of **mistakes**

# **THE PERCEPTRON ALGORITHM**

# Perceptron

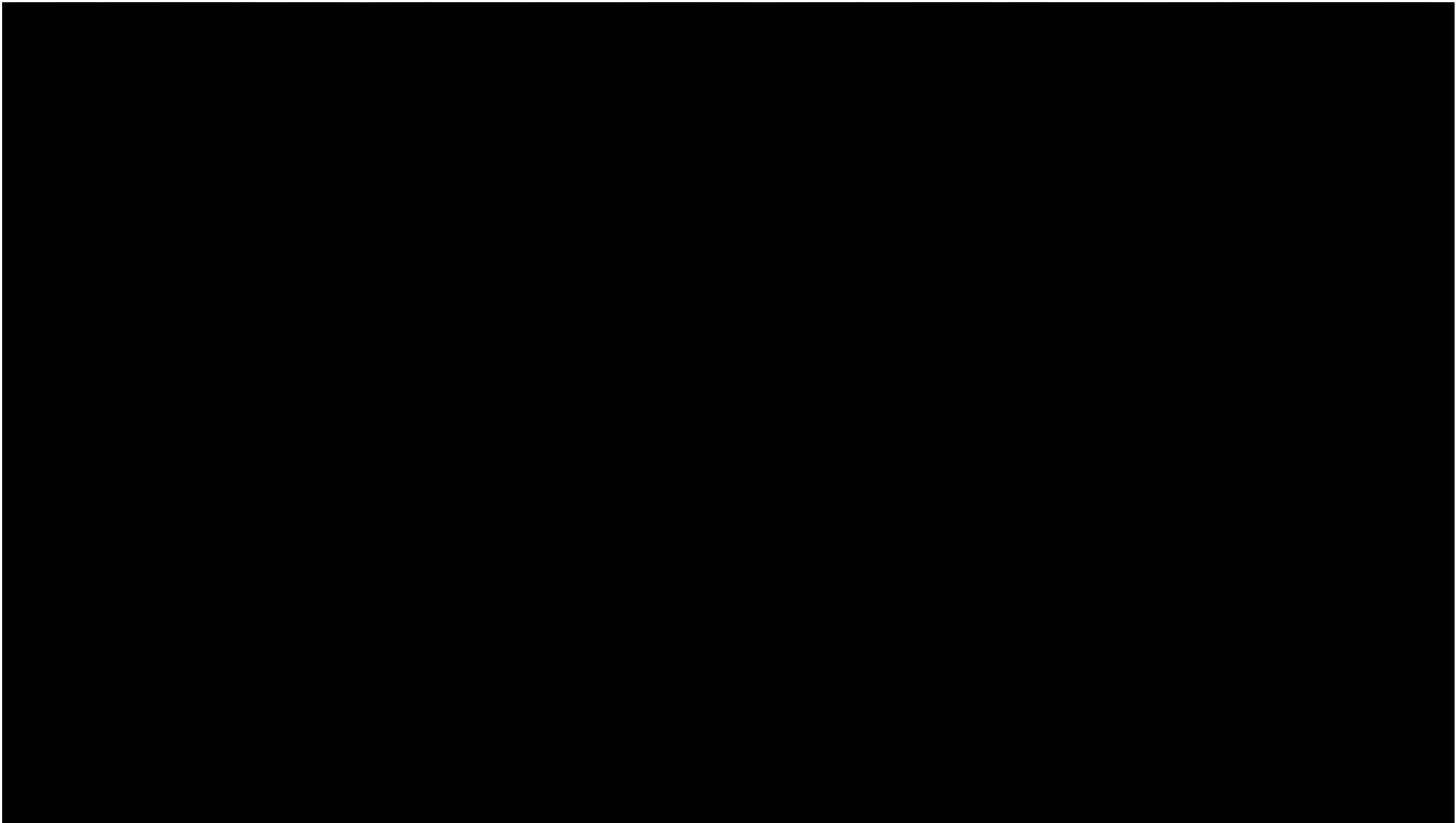
## *Whiteboard:*

- (Online) Perceptron Algorithm
- Hypothesis class for Perceptron
- 2D Example of Perceptron

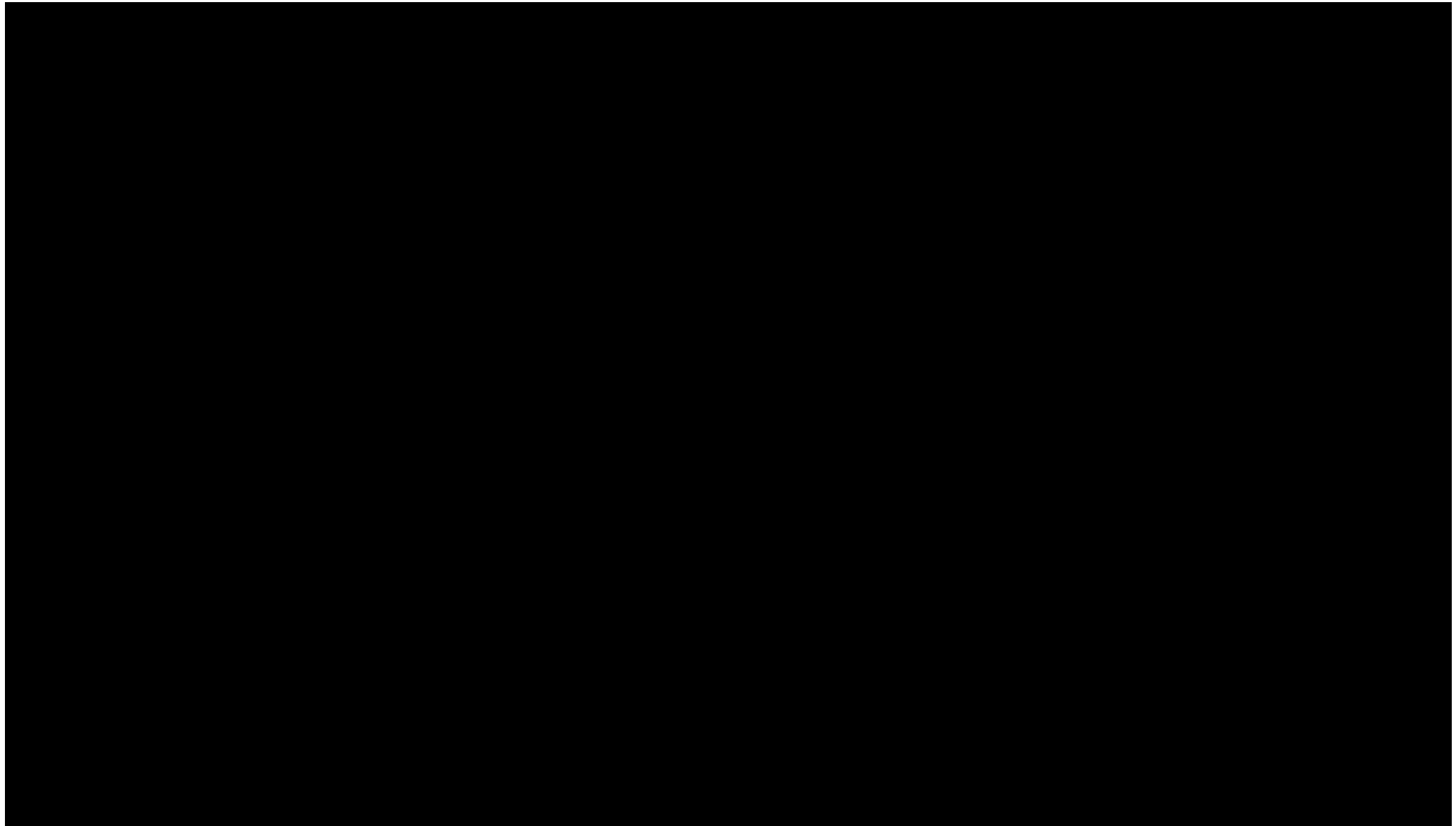




# AI for Wildlife Conservation



# AI for Wildlife Conservation



# AI for Wildlife Conservation

The **Great Elephant Census** of 2014 revealed that elephant populations were **trending downward** at an alarming rate.

**Poaching** is known to be one of the main threats to elephants.

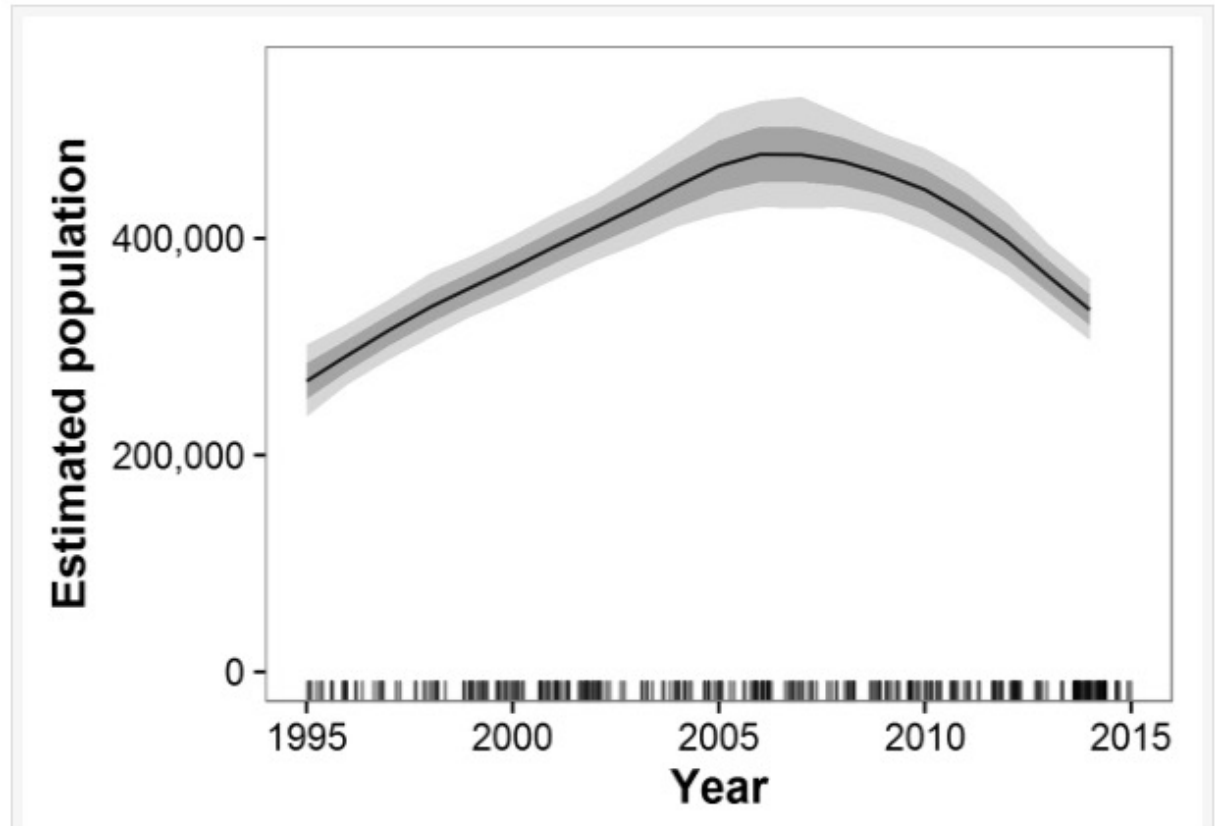



Figure 2: Estimated trends in elephant populations for GEC study areas with historical data available, 1995–2014.

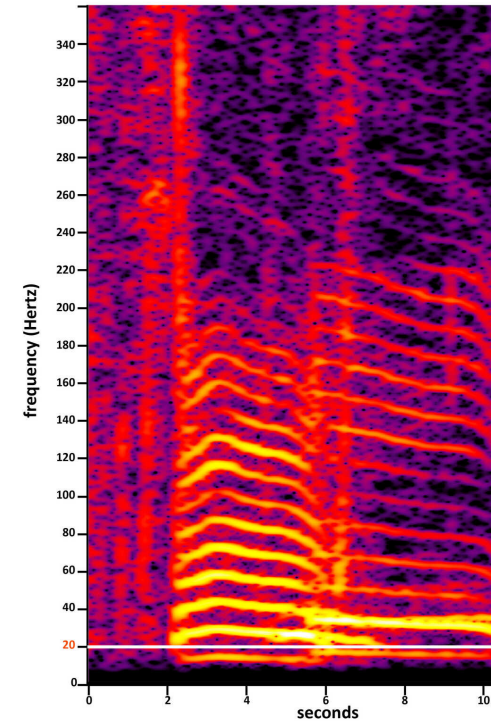
Results are based on 1,000 Monte Carlo replicates. Dark shaded area indicates  $\pm 1$  SD; light shaded area indicates 95% confidence interval. Tick marks on x-axis indicate dates of data points used in model; dates are perturbed slightly to prevent overlap.

 Download full-size image

DOI: [10.7717/peerj.2354/fig-2](https://doi.org/10.7717/peerj.2354/fig-2)



# AI for Wildlife Conservation



- Researchers at Cornell planted **50 audio recording devices** high in the jungle -- each one covering a 25 square km grid cell
- Recordings revealed two large creatures making noise: **elephants** and **poachers**
- So they built **classifiers** to detect these

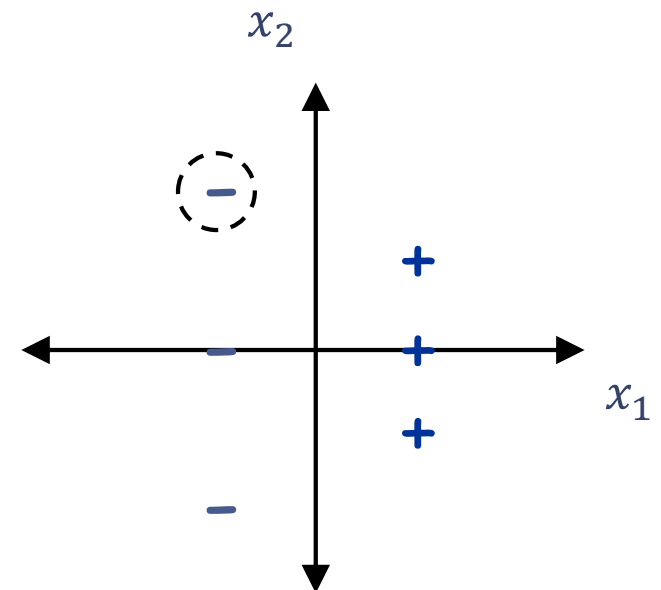
# (Online) Perceptron Algorithm: Example

## Perceptron Algorithm: (without the intercept term)

- Set  $t=1$ , start with all-zeroes weight vector  $w_1$ .
- Given example  $x$ , predict positive iff  $w_t \cdot x \geq 0$ .
- On a mistake, update as follows:
  - Mistake on positive, update  $w_{t+1} \leftarrow w_t + x$
  - Mistake on negative, update  $w_{t+1} \leftarrow w_t - x$

$x_1$	$x_2$	$\hat{y}$	$y$	Mistake?
-1	2	+	-	Yes

$$w = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

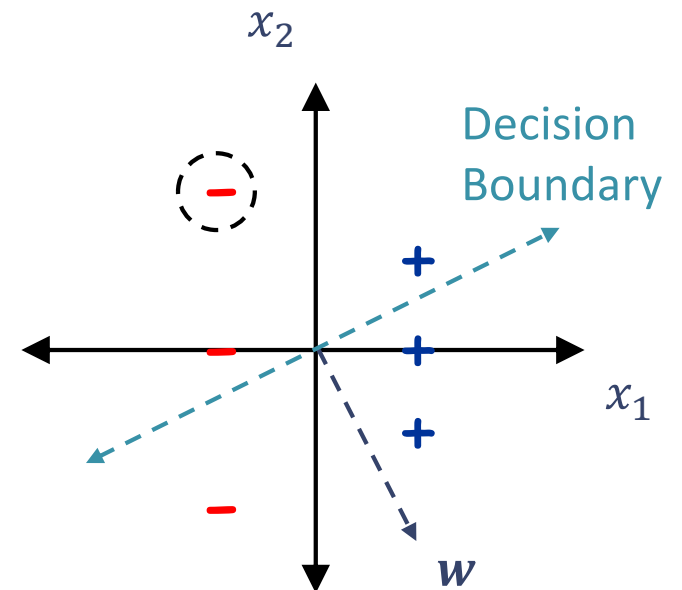


# (Online) Perceptron Algorithm: Example

## Perceptron Algorithm: (without the intercept term)

- Set  $t=1$ , start with all-zeroes weight vector  $w_1$ .
- Given example  $x$ , predict positive iff  $w_t \cdot x \geq 0$ .
- On a mistake, update as follows:
  - Mistake on positive, update  $w_{t+1} \leftarrow w_t + x$
  - Mistake on negative, update  $w_{t+1} \leftarrow w_t - x$

$x_1$	$x_2$	$\hat{y}$	$y$	Mistake?
-1	2	+	-	Yes



$$w = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$w \leftarrow w + y^{(1)}x^{(1)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

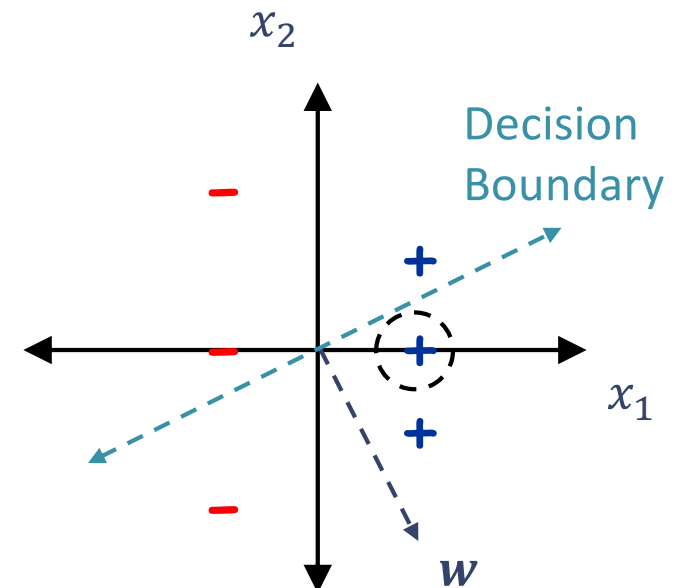
# (Online) Perceptron Algorithm: Example

## Perceptron Algorithm: (without the intercept term)

- Set  $t=1$ , start with all-zeroes weight vector  $w_1$ .
- Given example  $x$ , predict positive iff  $w_t \cdot x \geq 0$ .
- On a mistake, update as follows:
  - Mistake on positive, update  $w_{t+1} \leftarrow w_t + x$
  - Mistake on negative, update  $w_{t+1} \leftarrow w_t - x$

$x_1$	$x_2$	$\hat{y}$	$y$	Mistake?
-1	2	+	-	Yes
1	0	+	+	No

$$w = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

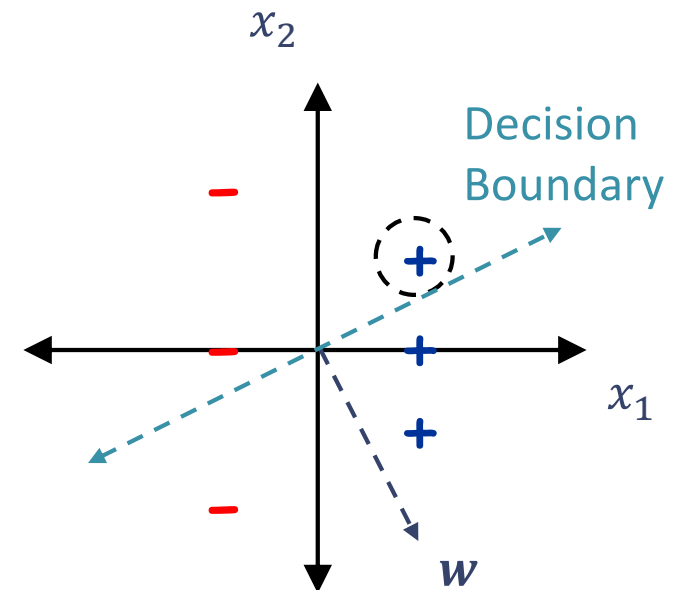


# (Online) Perceptron Algorithm: Example

## Perceptron Algorithm: (without the intercept term)

- Set  $t=1$ , start with all-zeroes weight vector  $w_1$ .
- Given example  $x$ , predict positive iff  $w_t \cdot x \geq 0$ .
- On a mistake, update as follows:
  - Mistake on positive, update  $w_{t+1} \leftarrow w_t + x$
  - Mistake on negative, update  $w_{t+1} \leftarrow w_t - x$

$x_1$	$x_2$	$\hat{y}$	$y$	Mistake?
-1	2	+	-	Yes
1	0	+	+	No
1	1	-	+	Yes



$$w = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

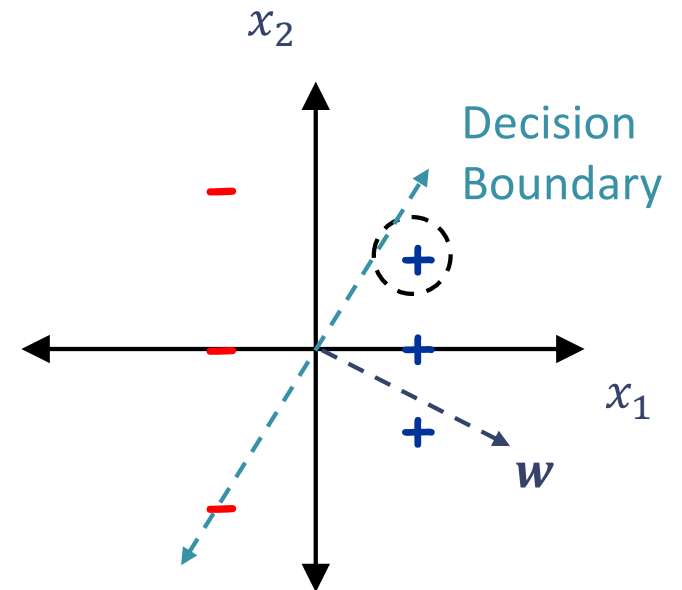
$$w \leftarrow w + y^{(3)}x^{(3)} = \begin{bmatrix} 1 \\ -2 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$$

# (Online) Perceptron Algorithm: Example

## Perceptron Algorithm: (without the intercept term)

- Set  $t=1$ , start with all-zeroes weight vector  $w_1$ .
- Given example  $x$ , predict positive iff  $w_t \cdot x \geq 0$ .
- On a mistake, update as follows:
  - Mistake on positive, update  $w_{t+1} \leftarrow w_t + x$
  - Mistake on negative, update  $w_{t+1} \leftarrow w_t - x$

$x_1$	$x_2$	$\hat{y}$	$y$	Mistake?
-1	2	+	-	Yes
1	0	+	+	No
1	1	-	+	Yes



$$w = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

$$w \leftarrow w + y^{(3)}x^{(3)} = \begin{bmatrix} 1 \\ -2 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$$

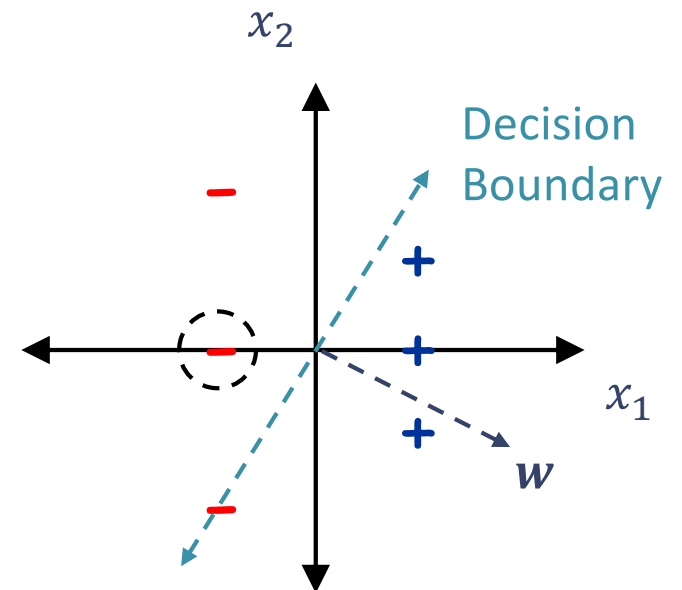
# (Online) Perceptron Algorithm: Example

## Perceptron Algorithm: (without the intercept term)

- Set  $t=1$ , start with all-zeroes weight vector  $w_1$ .
- Given example  $x$ , predict positive iff  $w_t \cdot x \geq 0$ .
- On a mistake, update as follows:
  - Mistake on positive, update  $w_{t+1} \leftarrow w_t + x$
  - Mistake on negative, update  $w_{t+1} \leftarrow w_t - x$

$x_1$	$x_2$	$\hat{y}$	$y$	Mistake?
-1	2	+	-	Yes
1	0	+	+	No
1	1	-	+	Yes
-1	0	-	-	No

$$w = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$$

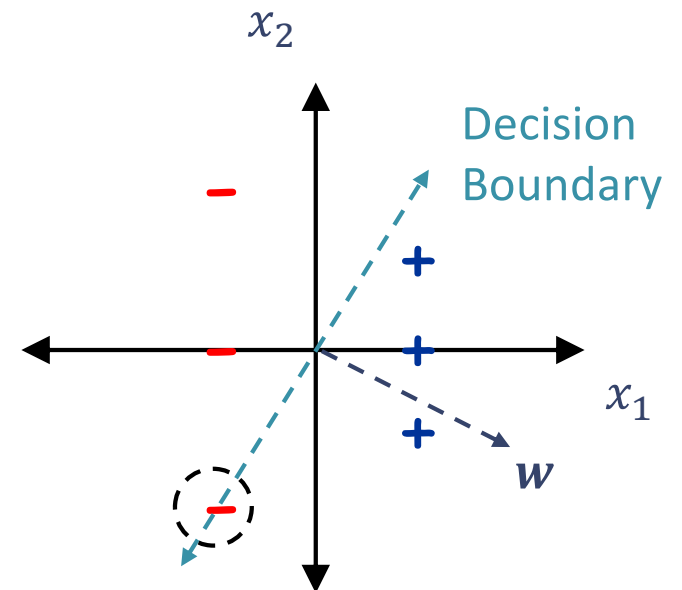


# (Online) Perceptron Algorithm: Example

## Perceptron Algorithm: (without the intercept term)

- Set  $t=1$ , start with all-zeroes weight vector  $w_1$ .
- Given example  $x$ , predict positive iff  $w_t \cdot x \geq 0$ .
- On a mistake, update as follows:
  - Mistake on positive, update  $w_{t+1} \leftarrow w_t + x$
  - Mistake on negative, update  $w_{t+1} \leftarrow w_t - x$

$x_1$	$x_2$	$\hat{y}$	$y$	Mistake?
-1	2	+	-	Yes
1	0	+	+	No
1	1	-	+	Yes
-1	0	-	-	No
-1	-2	+	-	Yes



$$w = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$$

$$w \leftarrow w + y^{(5)}x^{(5)} = \begin{bmatrix} 2 \\ -1 \end{bmatrix} - \begin{bmatrix} -1 \\ -2 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

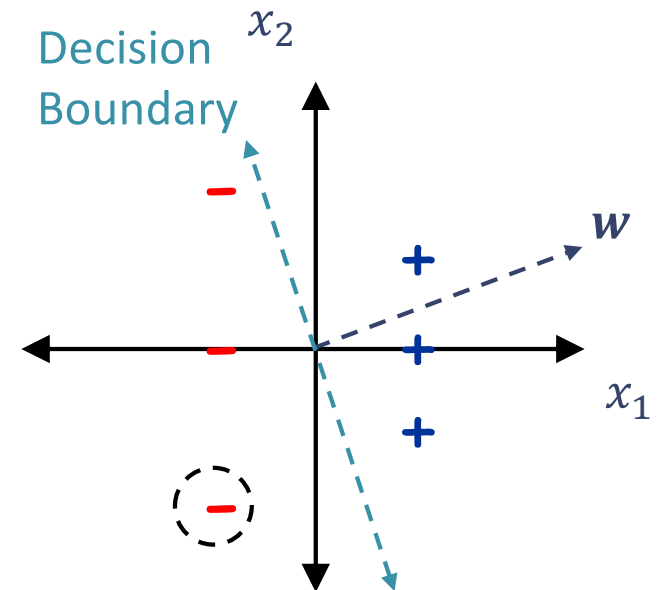


# (Online) Perceptron Algorithm: Example

## Perceptron Algorithm: (without the intercept term)

- Set  $t=1$ , start with all-zeroes weight vector  $w_1$ .
- Given example  $x$ , predict positive iff  $w_t \cdot x \geq 0$ .
- On a mistake, update as follows:
  - Mistake on positive, update  $w_{t+1} \leftarrow w_t + x$
  - Mistake on negative, update  $w_{t+1} \leftarrow w_t - x$

$x_1$	$x_2$	$\hat{y}$	$y$	Mistake?
-1	2	+	-	Yes
1	0	+	+	No
1	1	-	+	Yes
-1	0	-	-	No
-1	-2	+	-	Yes



$$w = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$$

$$w \leftarrow w + y^{(5)}x^{(5)} = \begin{bmatrix} 2 \\ -1 \end{bmatrix} - \begin{bmatrix} -1 \\ -2 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

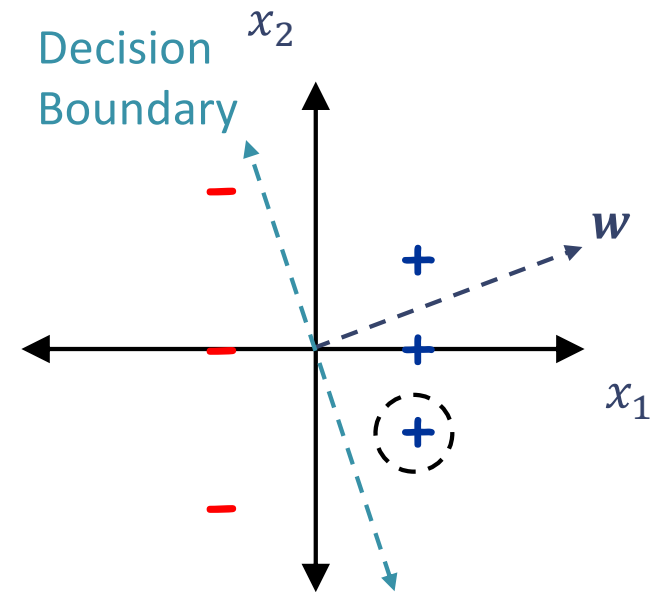
# (Online) Perceptron Algorithm: Example

## Perceptron Algorithm: (without the intercept term)

- Set  $t=1$ , start with all-zeroes weight vector  $w_1$ .
- Given example  $x$ , predict positive iff  $w_t \cdot x \geq 0$ .
- On a mistake, update as follows:
  - Mistake on positive, update  $w_{t+1} \leftarrow w_t + x$
  - Mistake on negative, update  $w_{t+1} \leftarrow w_t - x$

$x_1$	$x_2$	$\hat{y}$	$y$	Mistake?
-1	2	+	-	Yes
1	0	+	+	No
1	1	-	+	Yes
-1	0	-	-	No
-1	-2	+	-	Yes
1	-1	+	+	No

$$w = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$



# Perceptron Exercises

Question: Q2

The parameter vector  $w$  learned by the Perceptron algorithm can be **written as a linear combination** of the feature vectors  $x^{(1)}, x^{(2)}, \dots, x^{(N)}$ .

A. True, if you replace “linear” with “polynomial” above

B. True, for all datasets 79%

C. False, for all datasets

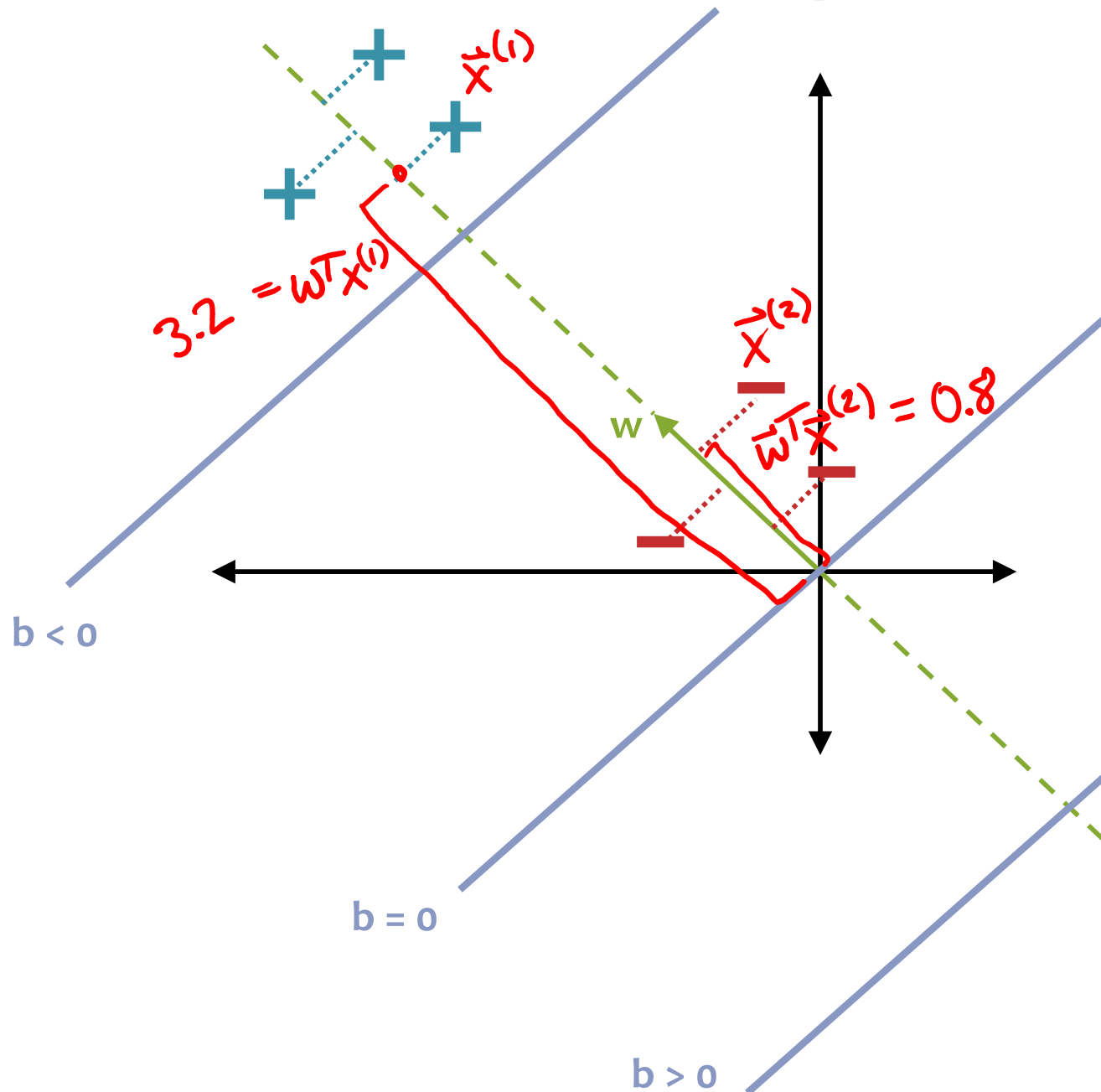
~~D. True, but only for certain datasets~~

E. False, but only for certain datasets

toxic

If  $\|w\|_2 = 1$ ,  $\vec{w}^T \vec{x}$  is the length of the projection of  $\vec{x}$  onto  $\vec{w}$

## Intercept Term



**Q:** Why do we need an intercept term?

**A:** It shifts the decision boundary off the origin

**Q:** Why do we add / subtract 1.0 to the intercept term during Perceptron training?

**A:** Two cases

1. Increasing  $b$  shifts the decision boundary towards the negative side
2. Decreasing  $b$  shifts the decision boundary towards the positive side

# (Online) Perceptron Algorithm

**Data:** Inputs are continuous vectors of length  $M$ . Outputs are discrete.

$$(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \dots$$

$$\text{where } \mathbf{x} \in \mathbb{R}^M \text{ and } y \in \{+1, -1\}$$

**Prediction:** Output determined by hyperplane.

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = \text{sign}(\boldsymbol{\theta}^T \mathbf{x})$$

$$\text{sign}(a) = \begin{cases} 1, & \text{if } a \geq 0 \\ -1, & \text{otherwise} \end{cases}$$

$$\text{Assume } \boldsymbol{\theta} = [b, w_1, \dots, w_M]^T \text{ and } x_1 = 1$$

**Learning:** Iterative procedure:

- **initialize** parameters to vector of all zeroes
- **while** not converged
  - **receive** next example  $(\mathbf{x}^{(i)}, y^{(i)})$
  - **predict**  $y' = h(\mathbf{x}^{(i)})$
  - **if** positive mistake: **add**  $\mathbf{x}^{(i)}$  to parameters
  - **if** negative mistake: **subtract**  $\mathbf{x}^{(i)}$  from parameters

# (Online) Perceptron Algorithm

- Initialize the weight vector and intercept to all zeros:

$$\mathbf{w} = [0 \quad 0 \quad \dots \quad 0] \text{ and } b = 0$$

- For  $t = 1, 2, 3, \dots$

- Receive an unlabeled example,  $\mathbf{x}^{(t)}$

- Predict its label,  $\hat{y} = \text{sign}(\mathbf{w}^T \mathbf{x} + b) = \begin{cases} +1 & \text{if } \mathbf{w}^T \mathbf{x} + b \geq 0 \\ -1 & \text{otherwise} \end{cases}$

- Observe its true label,  $y^{(t)}$

- If we misclassified a positive example ( $y^{(t)} = +1, \hat{y} = -1$ ):

- $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}^{(t)}$

- $b \leftarrow b + 1$

- If we misclassified a negative example ( $y^{(t)} = -1, \hat{y} = +1$ ):

- $\mathbf{w} \leftarrow \mathbf{w} - \mathbf{x}^{(t)}$

- $b \leftarrow b - 1$

# (Online) Perceptron Algorithm

- Initialize the weight vector and intercept to all zeros:

$$\mathbf{w} = [0 \quad 0 \quad \dots \quad 0] \text{ and } b = 0$$

- For  $t = 1, 2, 3, \dots$

- Receive an unlabeled example,  $\mathbf{x}^{(t)}$

- Predict its label,  $\hat{y} = \text{sign}(\mathbf{w}^T \mathbf{x} + b) = \begin{cases} +1 & \text{if } \mathbf{w}^T \mathbf{x} + b \geq 0 \\ -1 & \text{otherwise} \end{cases}$

- Observe its true label,  $y^{(t)}$

- If we misclassified an example ( $y^{(t)} \neq \hat{y}$ ):

- $\mathbf{w} \leftarrow \mathbf{w} + y^{(t)} \mathbf{x}^{(t)}$

- $b \leftarrow b + y^{(t)}$

Implementation trick: Multiplying by  $y^{(t)}$  gives us a simple update rule for both positive *and* negative mistakes

# Notational Hack

- If we add a 1 to the beginning of every example e.g.,

$$\mathbf{x}' = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix} \dots$$

- ... we can just fold the intercept into the weight vector!

$$\boldsymbol{\theta} = \begin{bmatrix} b \\ w_1 \\ w_2 \\ \vdots \\ w_D \end{bmatrix} \rightarrow \boldsymbol{\theta}^T \mathbf{x}' = \mathbf{w}^T \mathbf{x} + b$$



# (Online) Perceptron Algorithm

- Initialize the weight vector and intercept to all zeros:

$$\mathbf{w} = [0 \quad 0 \quad \dots \quad 0] \text{ and } b = 0$$

- For  $t = 1, 2, 3, \dots$

- Receive an unlabeled example,  $\mathbf{x}^{(t)}$

- Predict its label,  $\hat{y} = \text{sign}(\mathbf{w}^T \mathbf{x} + b) = \begin{cases} +1 & \text{if } \mathbf{w}^T \mathbf{x} + b \geq 0 \\ -1 & \text{otherwise} \end{cases}$

- Observe its true label,  $y^{(t)}$

- If we misclassified an example ( $y^{(t)} \neq \hat{y}$ ):

- $\mathbf{w} \leftarrow \mathbf{w} + y^{(t)} \mathbf{x}^{(t)}$

- $b \leftarrow b + y^{(t)}$

# (Online) Perceptron Algorithm

- Initialize the parameters to all zeros:

$$\boldsymbol{\theta} = [0 \quad 0 \quad \dots \quad 0]$$

- For  $t = 1, 2, 3, \dots$

- Receive an unlabeled example,  $\mathbf{x}^{(t)}$

1  
prepended  
to  $\mathbf{x}^{(t)}$

- Predict its label,  $\hat{y} = \text{sign}(\boldsymbol{\theta}^T \mathbf{x}'^{(t)}) = \begin{cases} +1 & \text{if } \boldsymbol{\theta}^T \mathbf{x}'^{(t)} \geq 0 \\ -1 & \text{otherwise} \end{cases}$

- Observe its true label,  $y^{(t)}$

- If we misclassified an example ( $y^{(t)} \neq \hat{y}$ ):

- $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + y^{(t)} \mathbf{x}'^{(t)}$



Automatically handles  
updating the intercept

# Perceptron Inductive Bias

1. Decision boundary should be linear
2. Recent mistakes are more important than older ones (and should be corrected immediately)

# (Online) Perceptron Algorithm

---

## Algorithm 1 Perceptron Learning Algorithm (Online)

---

```
1: procedure PERCEPTRON( $\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \dots\}$ )
2:    $\theta \leftarrow \mathbf{0}$  ▷ Initialize parameters
3:   for  $i \in \{1, 2, \dots\}$  do ▷ For each example
4:      $\hat{y} \leftarrow \text{sign}(\theta^T \mathbf{x}^{(i)})$  ▷ Predict
5:     if  $\hat{y} \neq y^{(i)}$  then ▷ If mistake
6:        $\theta \leftarrow \theta + y^{(i)} \mathbf{x}^{(i)}$  ▷ Update parameters
7:   return  $\theta$ 
```

---

# (Batch) Perceptron Algorithm

Learning for Perceptron also works if we have a fixed training dataset,  $D$ . We call this the “batch” setting in contrast to the “online” setting that we’ve discussed so far.

---

## Algorithm 1 Perceptron Learning Algorithm (Batch)

---

```
1: procedure PERCEPTRON( $\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \dots, (\mathbf{x}^{(N)}, y^{(N)})\}$ )
2:    $\theta \leftarrow \mathbf{0}$  ▷ Initialize parameters
3:   while not converged do
4:     for  $i \in \{1, 2, \dots, N\}$  do ▷ For each example
5:        $\hat{y} \leftarrow \text{sign}(\theta^T \mathbf{x}^{(i)})$  ▷ Predict
6:       if  $\hat{y} \neq y^{(i)}$  then ▷ If mistake
7:          $\theta \leftarrow \theta + y^{(i)} \mathbf{x}^{(i)}$  ▷ Update parameters
8:   return  $\theta$ 
```

---

# (Batch) Perceptron Algorithm

Learning for Perceptron also works if we have a fixed training dataset,  $D$ . We call this the “batch” setting in contrast to the “online” setting that we’ve discussed so far.

## Discussion:

The Batch Perceptron Algorithm can be derived in two ways.

1. By extending the online Perceptron algorithm to the batch setting (as mentioned above)
2. By applying **Stochastic Gradient Descent (SGD)** to minimize a so-called **Hinge Loss** on a linear separator

# Perceptron Exercise

Question:

Q3: SKIP

Unlike Decision Trees and K-Nearest Neighbors, the Perceptron algorithm **does not suffer from overfitting** because it does not have any hyperparameters that could be over-tuned on the training data.

- A. True
- B. False
- C. True and False

Answer:

# Extensions of Perceptron

- **Voted Perceptron**
  - generalizes better than (standard) perceptron
  - memory intensive (keeps around every weight vector seen during training, so each one can vote)
- **Averaged Perceptron**
  - empirically similar performance to voted perceptron
  - can be implemented in a memory efficient way (running averages are efficient)
- **Kernel Perceptron**
  - Choose a kernel  $K(x', x)$
  - Apply the **kernel trick** to Perceptron
  - Resulting algorithm is **still very simple**
- **Structured Perceptron**
  - Basic idea can also be applied when  $y$  ranges over an exponentially large set
  - Mistake bound **does not** depend on the size of that set