Machine Learning Department
School of Computer Science
Carnegie Mellon University

Perceptron

Matt Gormley & Henry Chai
Lecture 6
Sep. 13, 2024

Reminders

- **Homework 2: Decision Trees**
  - Out: Wed, Sep. 4
  - Due: Mon, Sep. 16 at 11:59pm
- **Homework 3: KNN, Perceptron, Lin.Reg.**
  - Out: Mon, Sep. 16
  - Due: Mon, Sep. 23 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

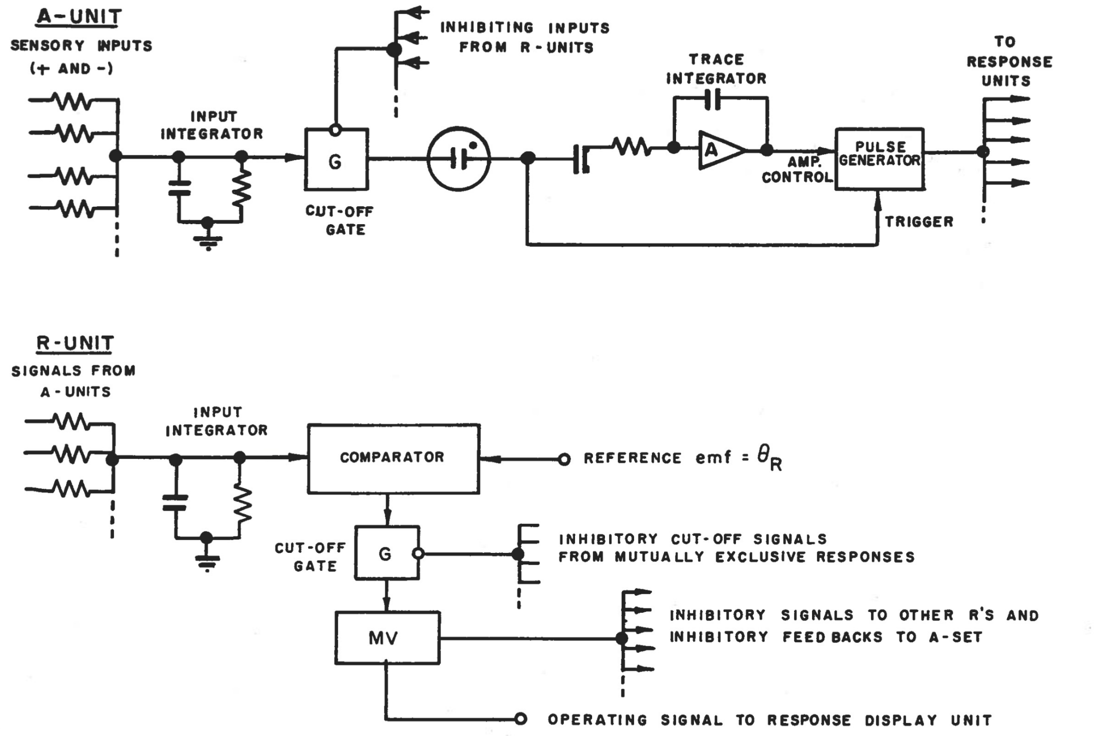
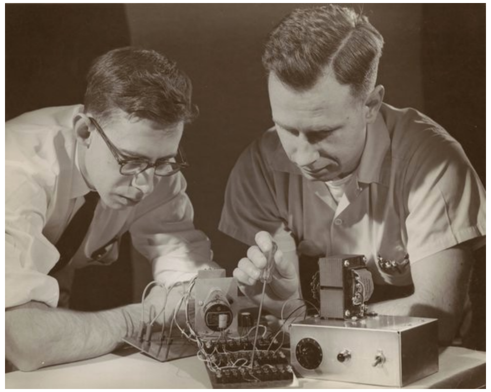
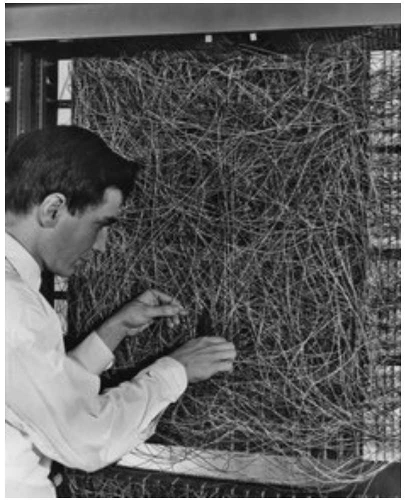


FIGURE 5
DESIGN OF TYPICAL UNITS

Perceptron: History

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

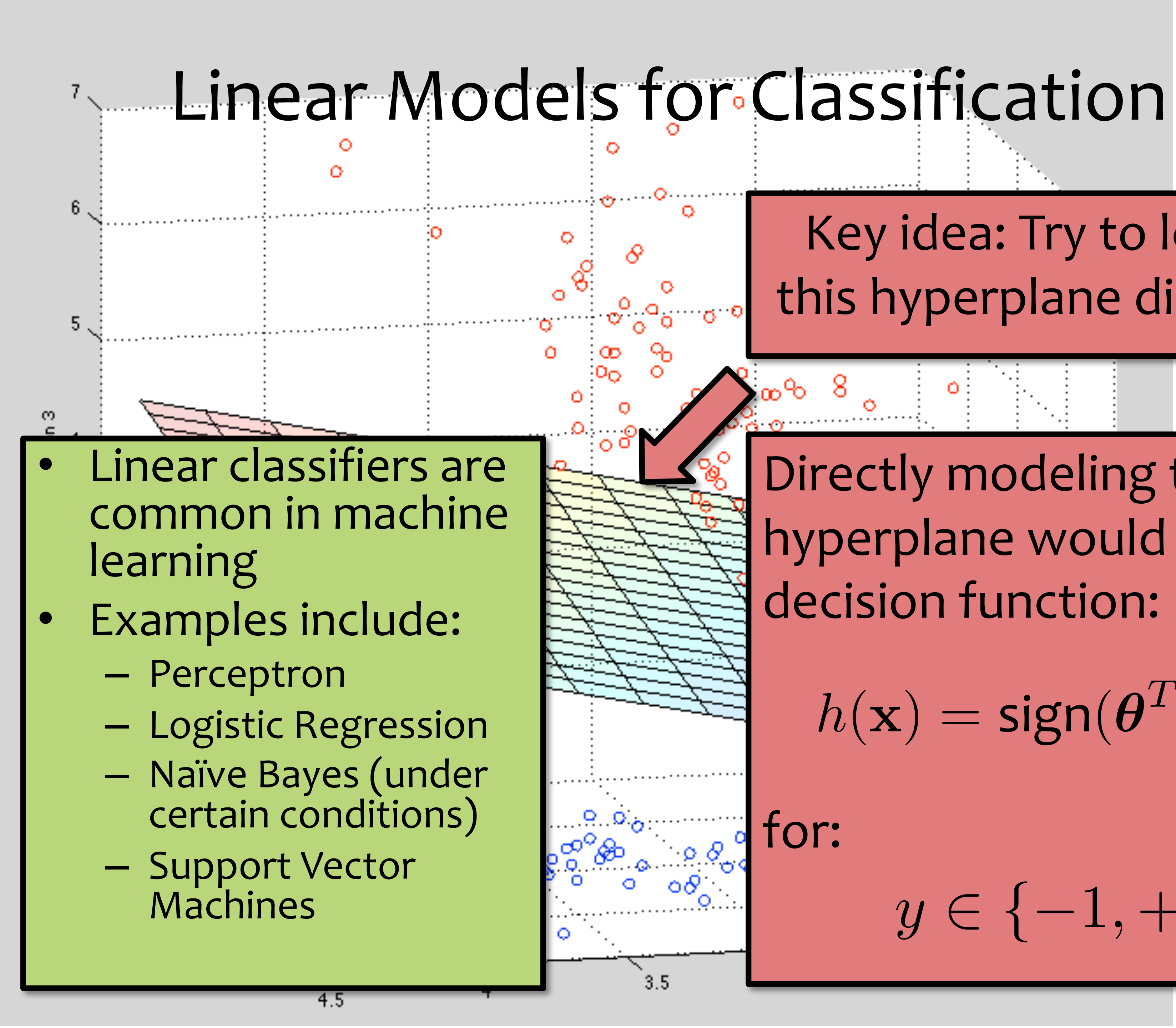




The New Yorker, December 6, 1958 P. 44

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

- Linear classifiers are common in machine learning
- Examples 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

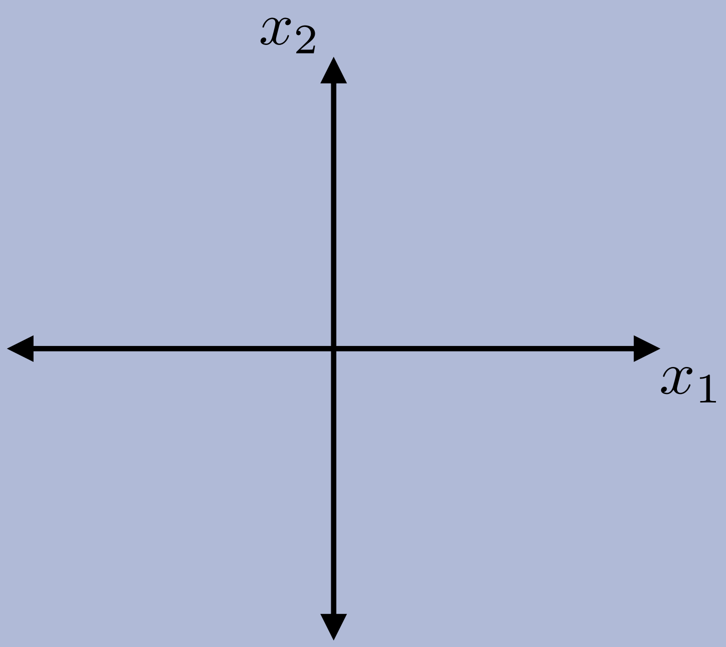
Draw a picture of the region corresponding to:

$$w_1 x_1 + w_2 x_2 + b > 0$$

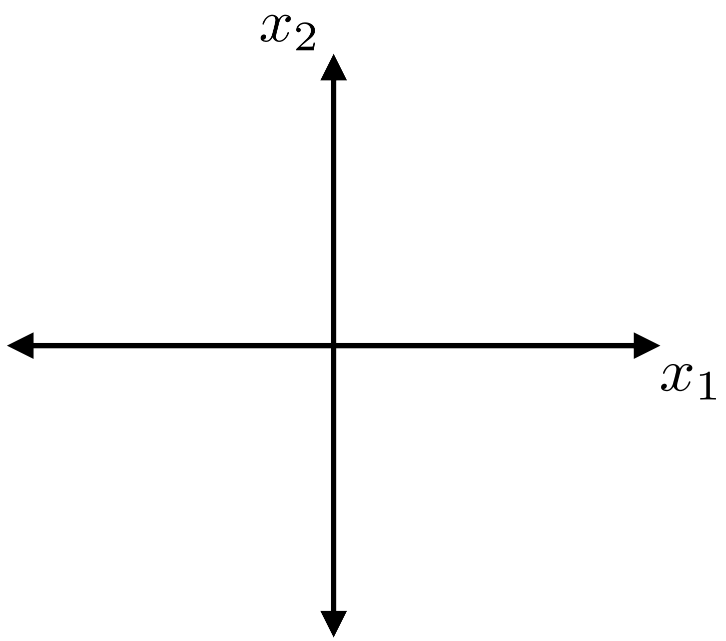
where $w_1 = 2, w_2 = 3, b = 6$

Draw the vector
w = $[w_1, w_2]$

Answer Here:



Geometry Warm-up



Linear Algebra Review

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

$$\boldsymbol{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_D \end{bmatrix} \text{ and } \boldsymbol{a}^T = \begin{bmatrix} a_1 & a_2 & \cdots & a_D \end{bmatrix}$$

- The dot product between two D-dimensional vectors is

$$\boldsymbol{a}^T \boldsymbol{b} = \begin{bmatrix} a_1 & a_2 & \cdots & a_D \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_D \end{bmatrix} = \sum_{d=1}^{D} a_d b_d$$

- The $L2$-norm of $\boldsymbol{a} = \|\boldsymbol{a}\|_2 = \sqrt{\boldsymbol{a}^T \boldsymbol{a}}$
- Two vectors are *orthogonal* iff

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

Vector Projection

Question:

*Which of the following is the projection of a vector **a** onto a vector **b**?*

- A. $\frac{\mathbf{a}^T\mathbf{b}}{\mathbf{b}}\mathbf{a}$
- B. $\frac{\mathbf{a}\cdot\mathbf{b}}{\mathbf{a}^T\mathbf{b}}$
- C. $\frac{(\mathbf{a}^T\mathbf{b})}{||\mathbf{b}||_2}\mathbf{b}$
- D. $\frac{(\mathbf{a}\cdot\mathbf{b})}{||\mathbf{b}||_2}\mathbf{b}$
- E. $\frac{(\mathbf{a}^T\mathbf{b})}{||\mathbf{b}||_2^2}\mathbf{b}$
- F. $\frac{(\mathbf{a}^T\mathbf{b})^2}{||\mathbf{b}||_2}\mathbf{b}$

Vector Projection

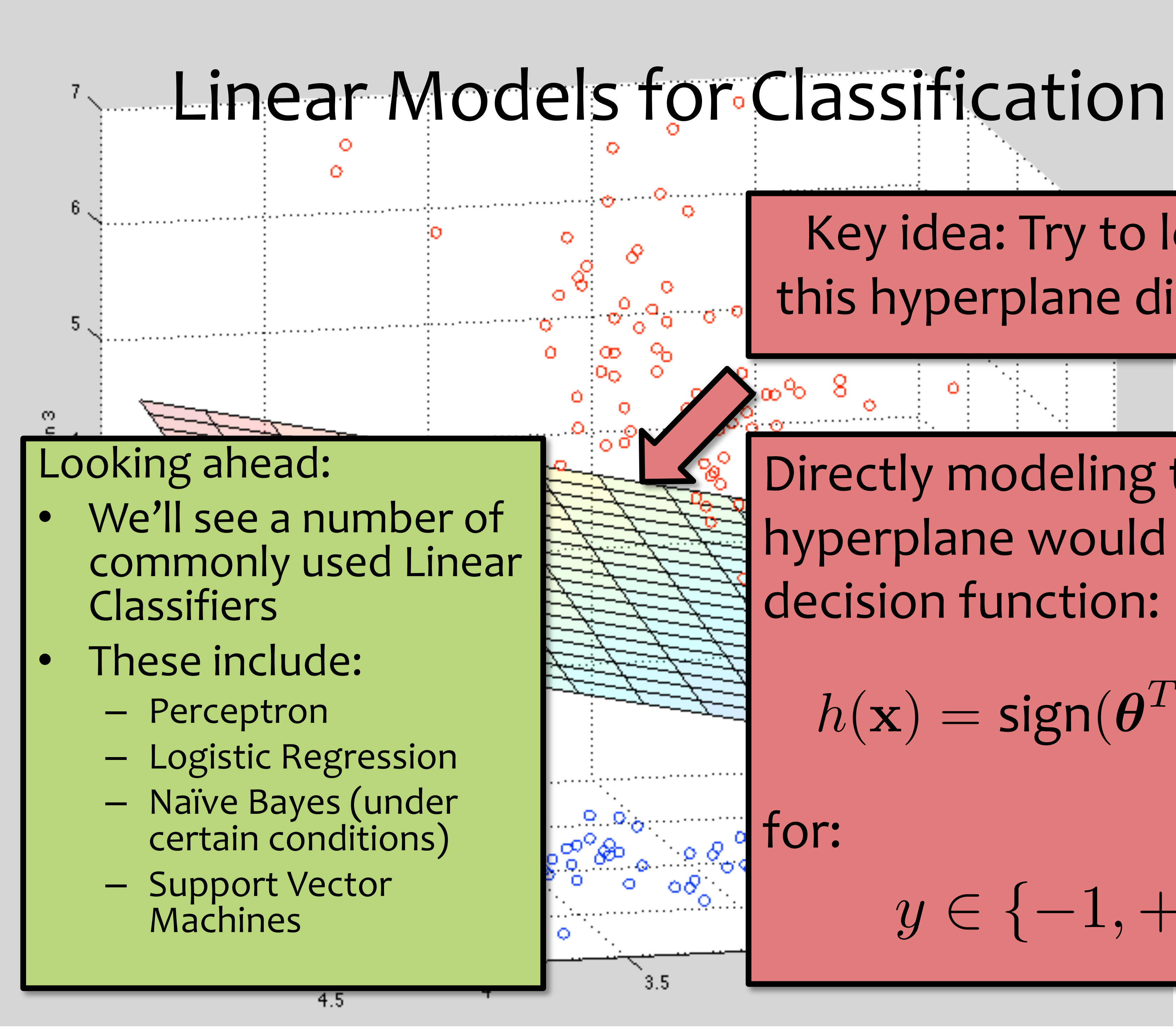
Definition #1:

Definition #2:

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, $\boldsymbol{w}^T\boldsymbol{x} + b = 0$ defines a *hyperplane*
  - The vector $\boldsymbol{w}$ is always orthogonal to this hyperplane and always points in the direction where $\boldsymbol{w}^T\boldsymbol{x} + b > 0$!
- A hyperplane creates two *halfspaces*:
  - $\mathcal{S}_+ = \{\boldsymbol{x}\colon\ \boldsymbol{w}^T\boldsymbol{x} + b > 0\}$ or all $\boldsymbol{x}$ s.t. $\boldsymbol{w}^T\boldsymbol{x} + b$ is positive
  - $\mathcal{S}_- = \{\boldsymbol{x}\colon\ \boldsymbol{w}^T\boldsymbol{x} + b < 0\}$ or all $\boldsymbol{x}$ s.t. $\boldsymbol{w}^T\boldsymbol{x} + b$ 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

Slide adapted from Nina Balcan

Online Learning

For i = 1, 2, 3, … **:**

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

Goal:

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

THE PERCEPTRON ALGORITHM

(Online) Perceptron Algorithm

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.

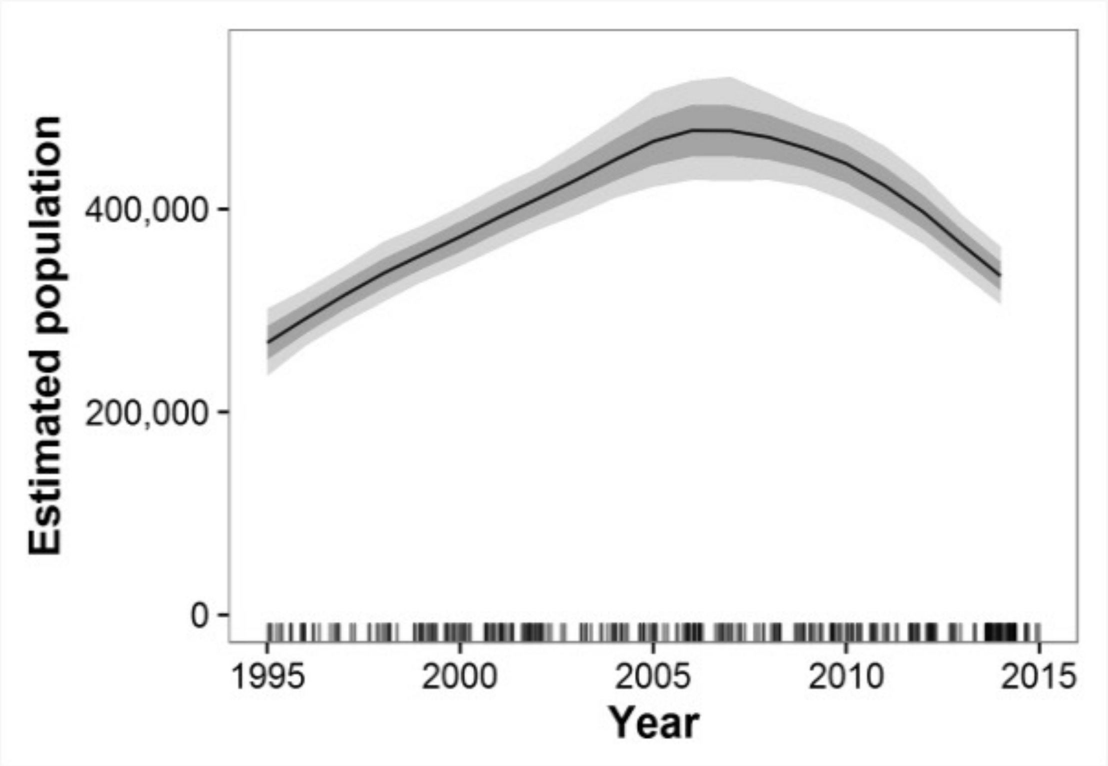


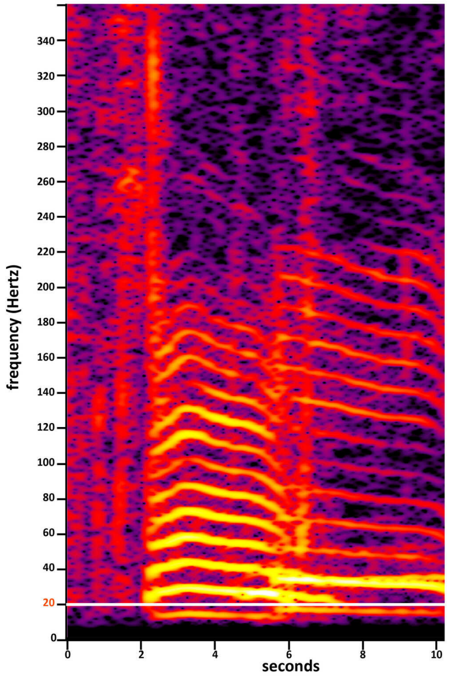
Figure 2: Estimated trends in elephant populations for GEC study areas with historical data available, 1995–2014.
Results are based on 1,000 Monte Carlo replicates. Dark shaded area indicates ±1 SD; light shaded area indicates 95% confidence interval. Tick marks on *x*-axis indicate dates of data points used in model; dates are perturbed slightly to prevent overlap.

Download full-size image

DOI: 10.7717/peerj.2354/fig-2

https://peerj.com/articles/2354/

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

https://www.npr.org/2019/10/25/760487476/elephants-under-attack-have-an-unlikely-ally-artificial-intelligence

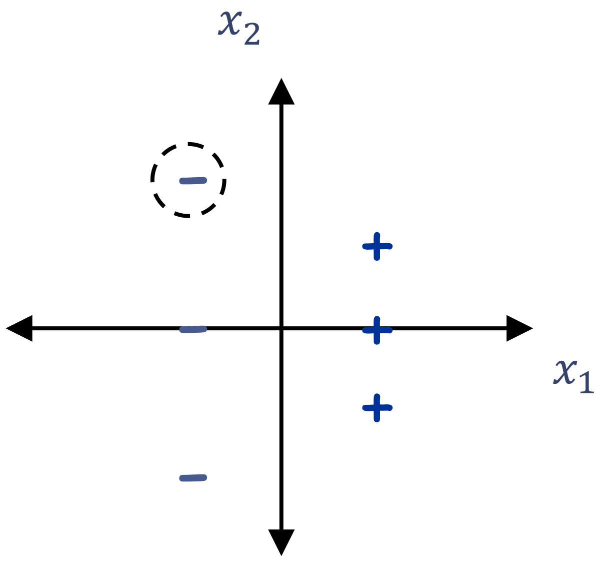
(Online) Perceptron Algorithm: Example

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

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

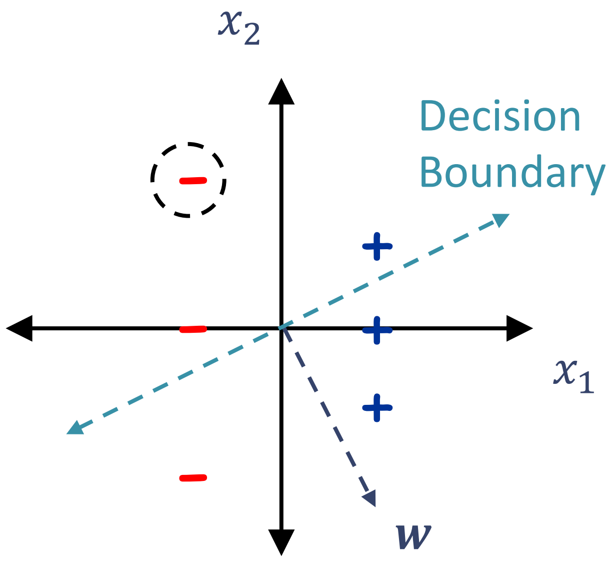


(Online) Perceptron Algorithm: Example

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



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

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

Example courtesy of Nina Balcan

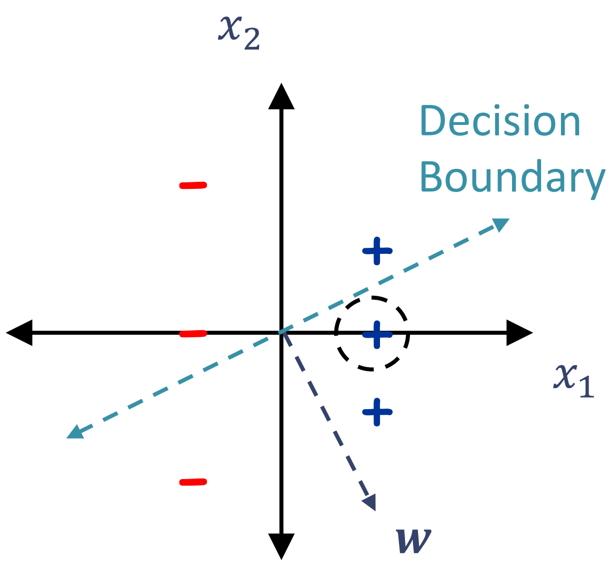
(Online) Perceptron Algorithm: Example

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

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



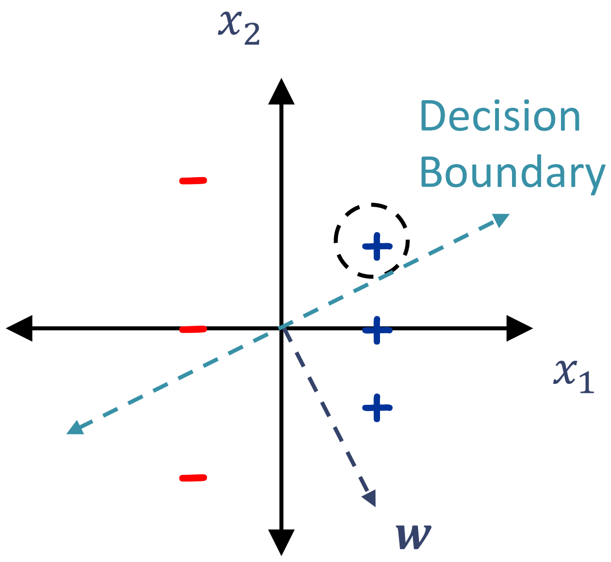
Example courtesy of Nina Balcan

(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



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

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

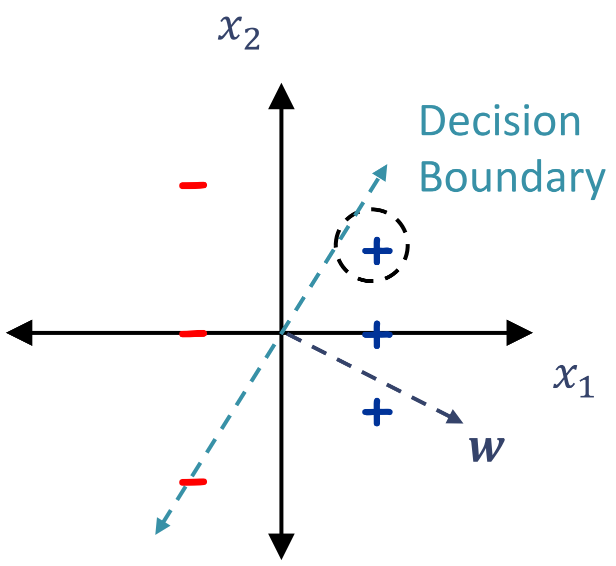
Example courtesy of Nina Balcan

(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



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

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

Example courtesy of Nina Balcan

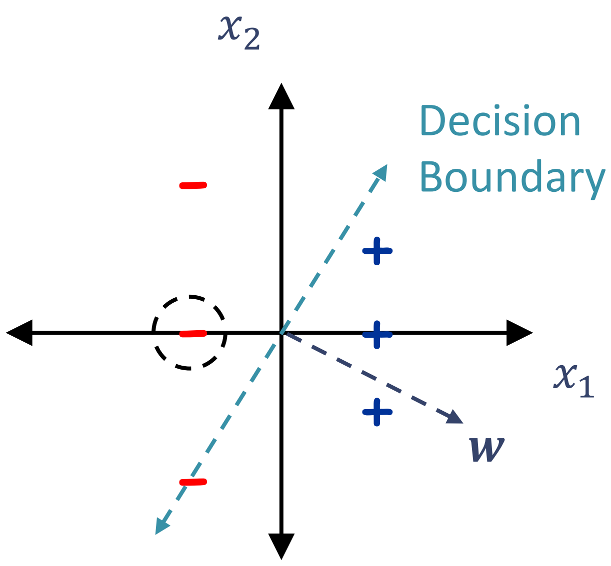
(Online) Perceptron Algorithm: Example

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

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



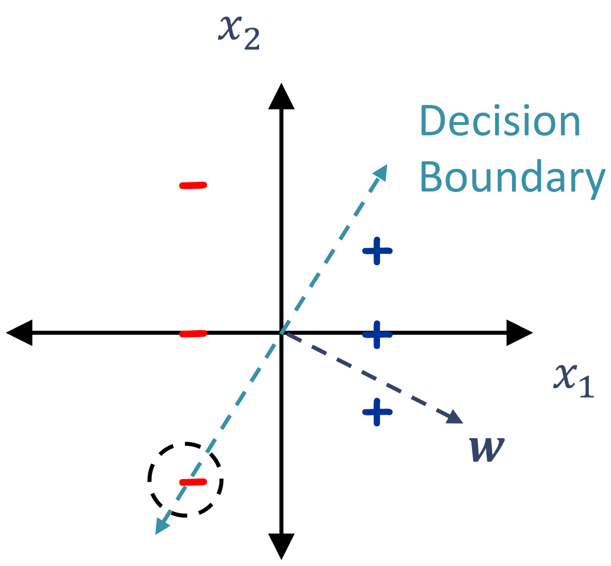
Example courtesy of Nina Balcan

(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



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

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

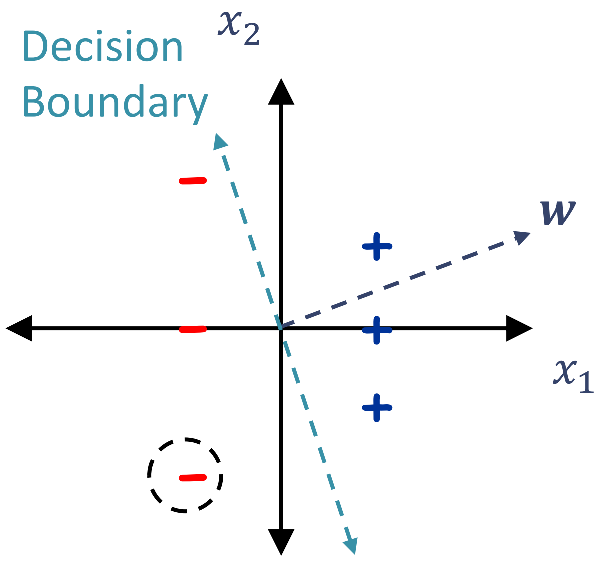
Example courtesy of Nina Balcan

(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



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

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

Example courtesy of Nina Balcan

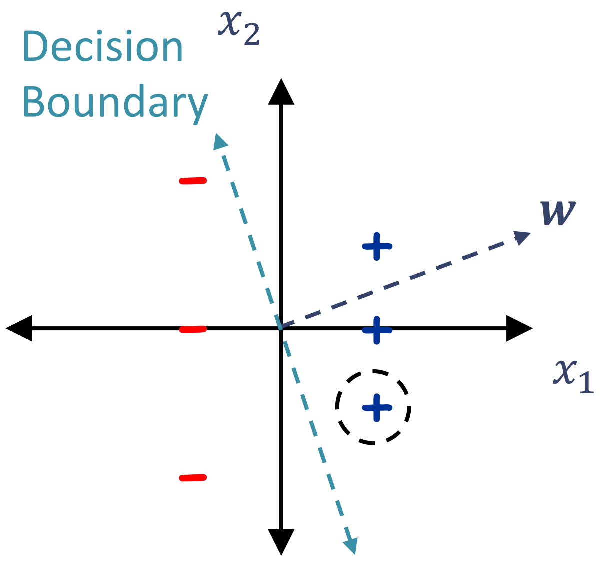
(Online) Perceptron Algorithm: Example

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

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



Example courtesy of Nina Balcan

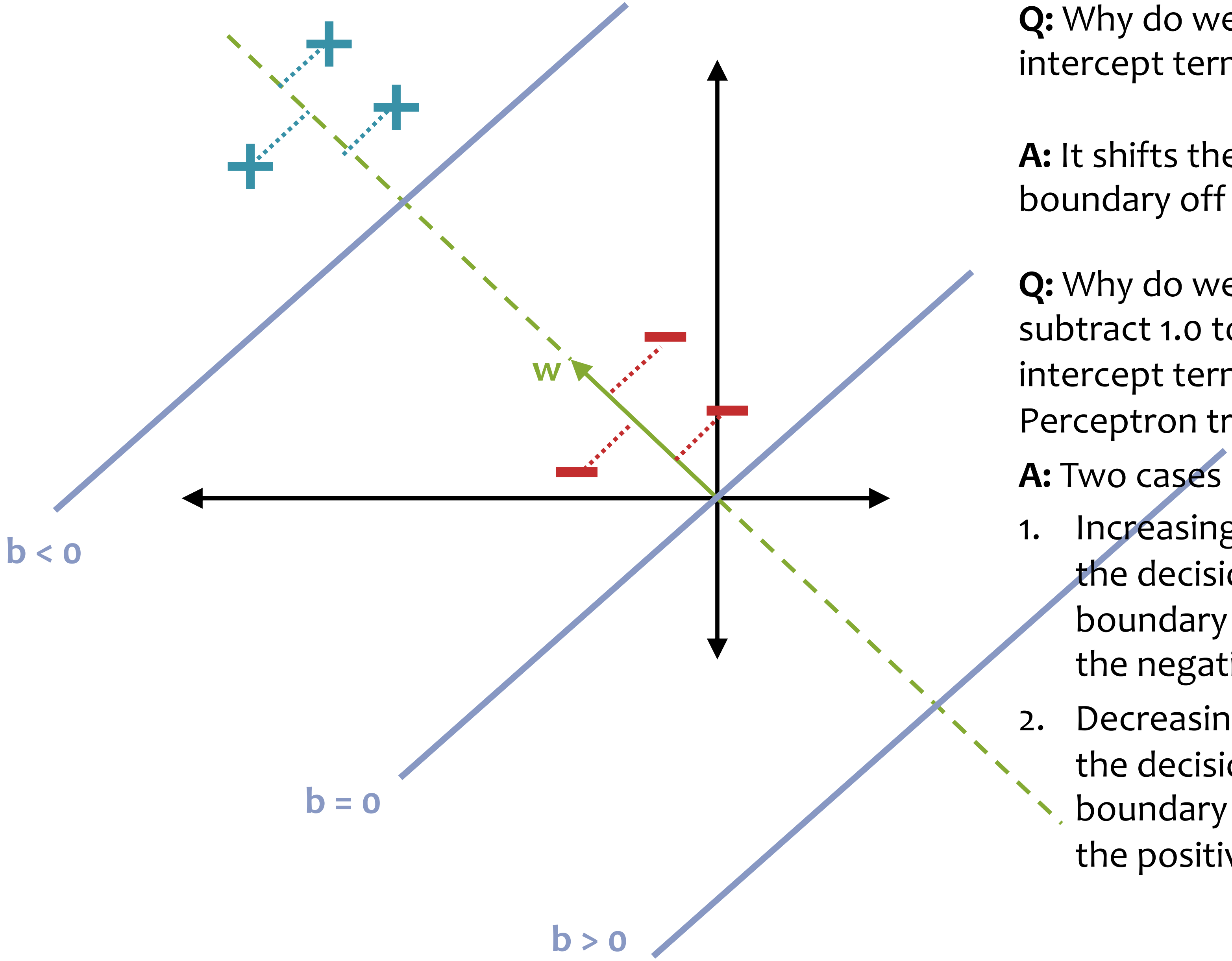
Perceptron Exercises

Question:

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

A. *True, if you replace "linear" with "polynomial" above*
B. *True, for all datasets*
C. *False, for all datasets*
D. *True, but only for certain datasets*
E. *False, but only for certain datasets*

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.

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

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

Prediction: Output determined by hyperplane.

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

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

Assume $\boldsymbol{\theta} = [b, w_1, \ldots, w_M]^T$ 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:
  $$\boldsymbol{w} = [0 \quad 0 \quad \cdots \quad 0] \text{ and } b = 0$$
- For $t = 1, 2, 3, \ldots$
  - Receive an unlabeled example, $\boldsymbol{x}^{(t)}$
  - Predict its label, $\hat{y} = \text{sign}(\boldsymbol{w}^T\boldsymbol{x} + b) = \begin{cases} +1 \text{ if } \boldsymbol{w}^T\boldsymbol{x} + b \geq 0 \\ -1 \text{ otherwise} \end{cases}$
  - Observe its true label, $y^{(t)}$
  - If we misclassified a positive example ($y^{(t)} = +1, \hat{y} = -1$):
    - $\boldsymbol{w} \leftarrow \boldsymbol{w} + \boldsymbol{x}^{(t)}$
    - $b \leftarrow b + 1$
  - If we misclassified a negative example ($y^{(t)} = -1, \hat{y} = +1$):
    - $\boldsymbol{w} \leftarrow \boldsymbol{w} - \boldsymbol{x}^{(t)}$
    - $b \leftarrow b - 1$

(Online) Perceptron Algorithm

- Initialize the weight vector and intercept to all zeros:

  $$\boldsymbol{w} = [0 \quad 0 \quad \cdots \quad 0] \text{ and } b = 0$$

- For $t = 1, 2, 3, \ldots$
  - Receive an unlabeled example, $\boldsymbol{x}^{(t)}$
  - Predict its label, $\hat{y} = \text{sign}(\boldsymbol{w}^T\boldsymbol{x} + b) = \begin{cases} +1 \text{ if } \boldsymbol{w}^T\boldsymbol{x} + b \geq 0 \\ -1 \text{ otherwise} \end{cases}$
  - Observe its true label, $y^{(t)}$
  - If we misclassified an example ($y^{(t)} \neq \hat{y}$):
    - $\boldsymbol{w} \leftarrow \boldsymbol{w} + y^{(t)}\boldsymbol{x}^{(t)}$
    - $b \leftarrow b + y^{(t)}$

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.,

$$\boldsymbol{x}' = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix} \ldots$$

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

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

(Online) Perceptron Algorithm

- Initialize the weight vector and intercept to all zeros:

  $\boldsymbol{w} = [0 \quad 0 \quad \cdots \quad 0]$ and $b = 0$

- For $t = 1, 2, 3, \ldots$
  - Receive an unlabeled example, $\boldsymbol{x}^{(t)}$
  - Predict its label, $\hat{y} = \operatorname{sign}(\boldsymbol{w}^T \boldsymbol{x} + b) = \begin{cases} +1 \text{ if } \boldsymbol{w}^T \boldsymbol{x} + b \geq 0 \\ -1 \text{ otherwise} \end{cases}$
  - Observe its true label, $y^{(t)}$
  - If we misclassified an example ($y^{(t)} \neq \hat{y}$):
    - $\boldsymbol{w} \leftarrow \boldsymbol{w} + y^{(t)} \boldsymbol{x}^{(t)}$
    - $b \leftarrow b + y^{(t)}$

(Online) Perceptron Algorithm

- Initialize the parameters to all zeros:

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

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

- For $t = 1, 2, 3, \ldots$
  - Receive an unlabeled example, $\boldsymbol{x}^{(t)}$
  - Predict its label, $\hat{y} = \text{sign}\left(\boldsymbol{\theta}^T \boldsymbol{x}'^{(t)}\right) = \begin{cases} +1 \text{ if } \boldsymbol{\theta}^T \boldsymbol{x}'^{(t)} \geq 0 \\ -1 \text{ otherwise} \end{cases}$
  - Observe its true label, $y^{(t)}$
  - If we misclassified an example ($y^{(t)} \neq \hat{y}$):
    - $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + y^{(t)} \boldsymbol{x}'^{(t)}$

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)

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

(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)

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

(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($\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(N)},$
2: $\quad\boldsymbol{\theta} \leftarrow \mathbf{0}$ ▷ Initialize para
3: $\quad$**while** not converged **do**
4: $\quad\quad$**for** $i \in \{1, 2, \ldots, N\}$ **do** ▷ For each e
5: $\quad\quad\quad\hat{y} \leftarrow \text{sign}(\boldsymbol{\theta}^T \mathbf{x}^{(i)})$ ▷
6: $\quad\quad\quad$**if** $\hat{y} \neq y^{(i)}$ **then** ▷ If
7: $\quad\quad\quad\quad\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + y^{(i)} \mathbf{x}^{(i)}$ ▷ Update para
8: $\quad$**return** $\boldsymbol{\theta}$

Def: We say that the (batch) perceptron algorithm has **converged** if it stops making mistakes on the training data (perfectly classifies the training data).

(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:

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.*

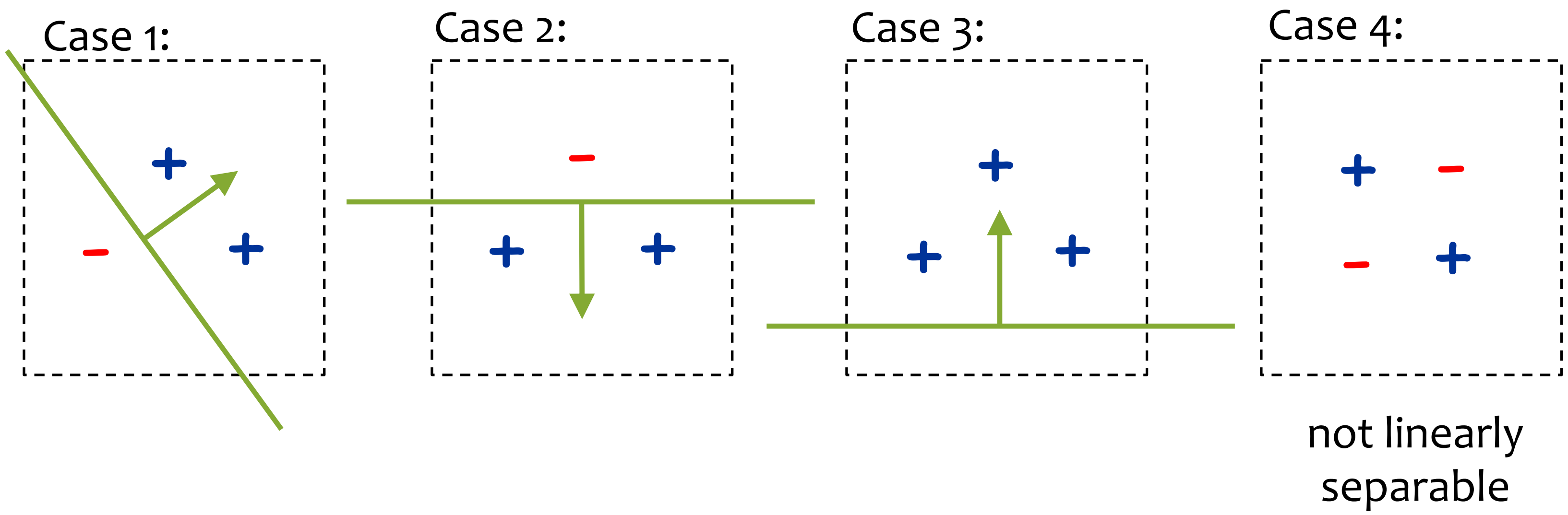
A. *True*
B. *False*
C. *True and False*

Answer:

PERCEPTRON MISTAKE BOUND

Definitions

***Def*: For a **binary classification** problem, a set of examples S is **linearly separable** if there exists a linear decision boundary that can separate the points**



Definitions

Def: The **margin** γ for a dataset D is the greatest possible distance between a linear separator and the closest data point in D to that linear separator

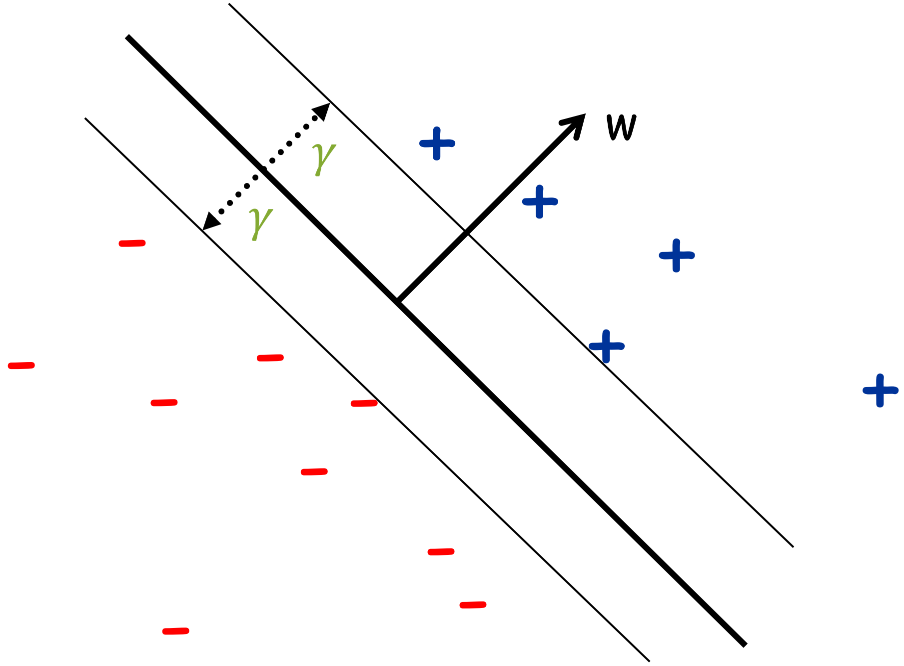
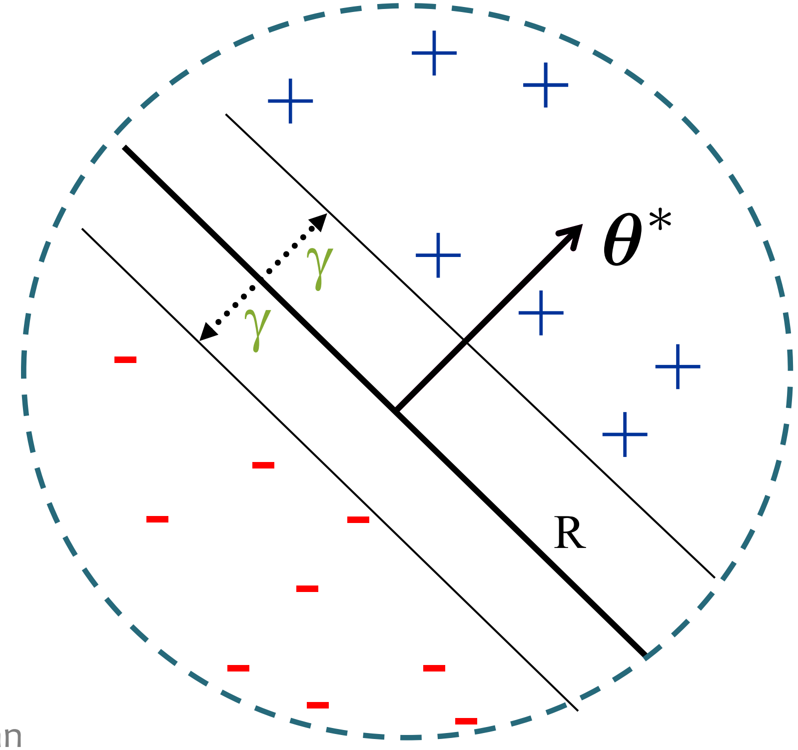


Figure from Nina Balcan

Perceptron Mistake Bound

Guarantee: if some data has margin γ and all points lie inside a ball of radius R rooted at the origin, then the online Perceptron algorithm makes $\leq (R/\gamma)^2$ mistakes

(Normalized margin: multiplying all points by 100, or dividing all points by 100, doesn't change the number of mistakes! The algorithm is invariant to scaling.)



Main Takeaway: For **linearly separable** data, if the perceptron algorithm cycles repeatedly through the data, it will **converge** in a finite # of steps.

Slide adapted from Nina Balcan

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