10-301/601 Introduction to Machine Learning

Machine Learning Department
School of Computer Science
Carnegie Mellon University

Perceptron

Matt Gormley & Henry Chai
Lecture 6
Sep. 17, 2021

Q&A

Q: Is there really another change in the Collaboration Policy for HW3?

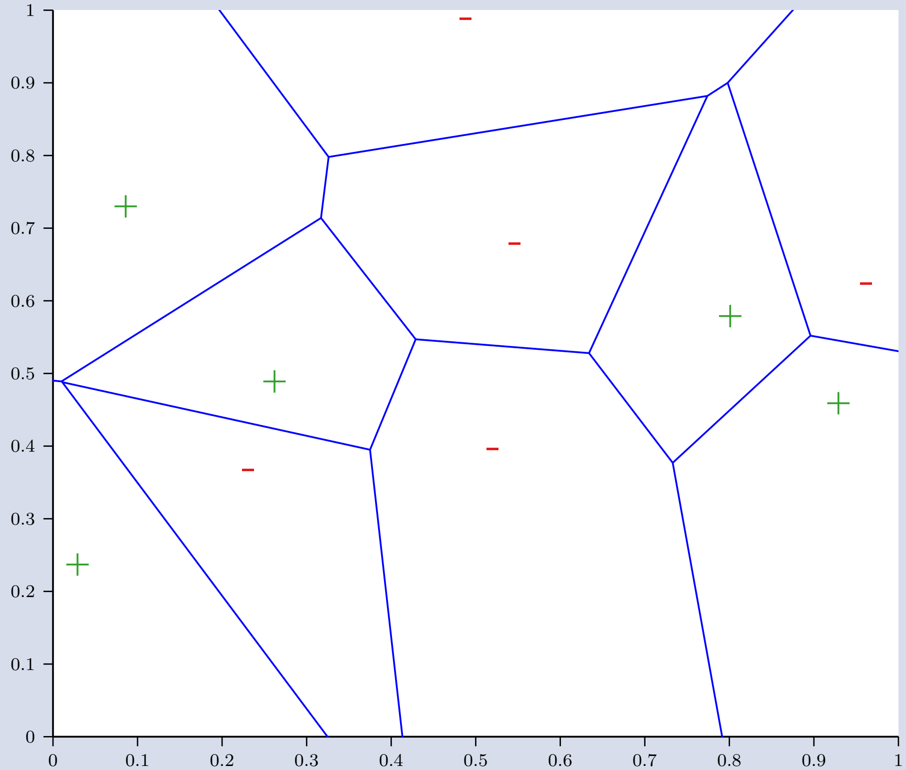
A: Yes! Here are the details:

- For each written assignment, you will be **randomly assigned** to a homework group of 3. These homework groups will be different for each assignment.
- Every problem on each written assignment will be designated as **SOLO** or **GROUP**.
- Within your assigned homework group, you are allowed to collaborate on **GROUP** problems more fully than before. Specifically, you are allowed to share written notes pertaining to **GROUP** problems and you are allowed to write your solutions collectively.
- Our original Collaboration Policy still applies to **SOLO** questions: any written notes about these problems must be taken on an impermanent surface (e.g. whiteboard, chalkboard) and the actual solution to these problems must be written by yourself.

Q&A

Q: Those decision boundary figures for KNN were really cool, how did you make those?

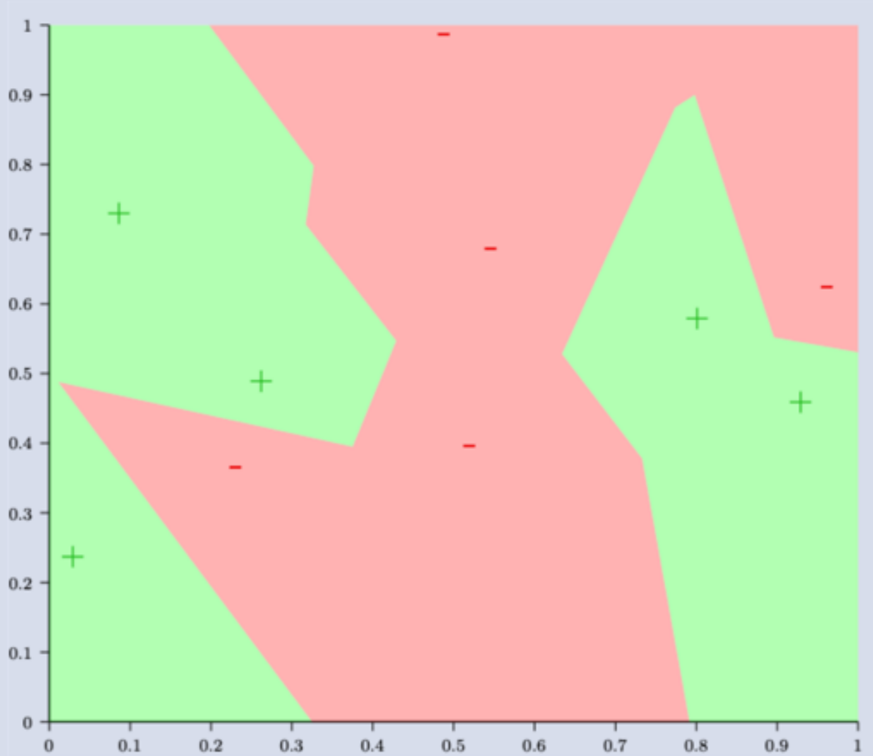
A: Well it's a little complicated for k > 1, but here's a way you can think about decision boundaries for a nearest neighbor hypothesis (k=1)



Q&A

Q: Those decision boundary figures for KNN were really cool, how did you make those?

A: Well it's a little complicated for k > 1, but here's a way you can think about decision boundaries for a nearest neighbor hypothesis (k=1)



Reminders

- Homework 2: Decision Trees
  - Out: Wed, Sep. 8
  - Due: Mon, Sep. 20 at 11:59pm
- Homework 3: KNN, Perceptron, Lin.Reg.
  - Out: Mon, Sep. 20 (+ 1 day)
  - Due: Sun, Sep. 26 at 11:59pm
- Today's In-Class Poll
  - http://poll.mlcourse.org

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

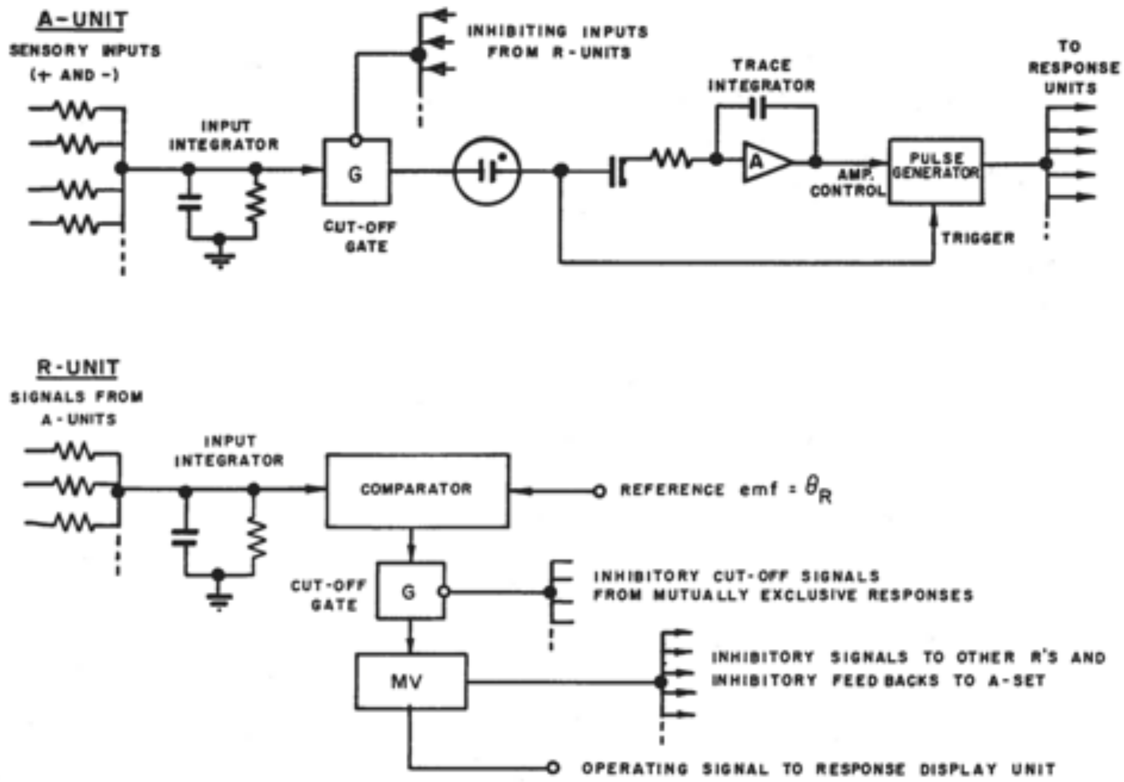
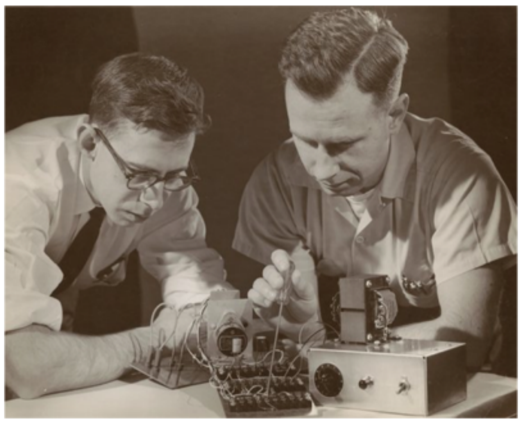
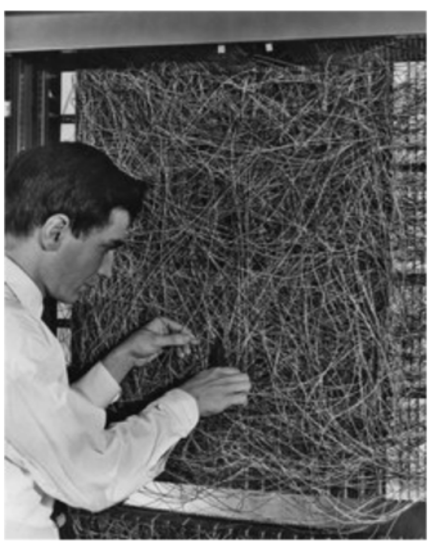


FIGURE 5
DESIGN OF TYPICAL UNITS

Perceptron: History

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

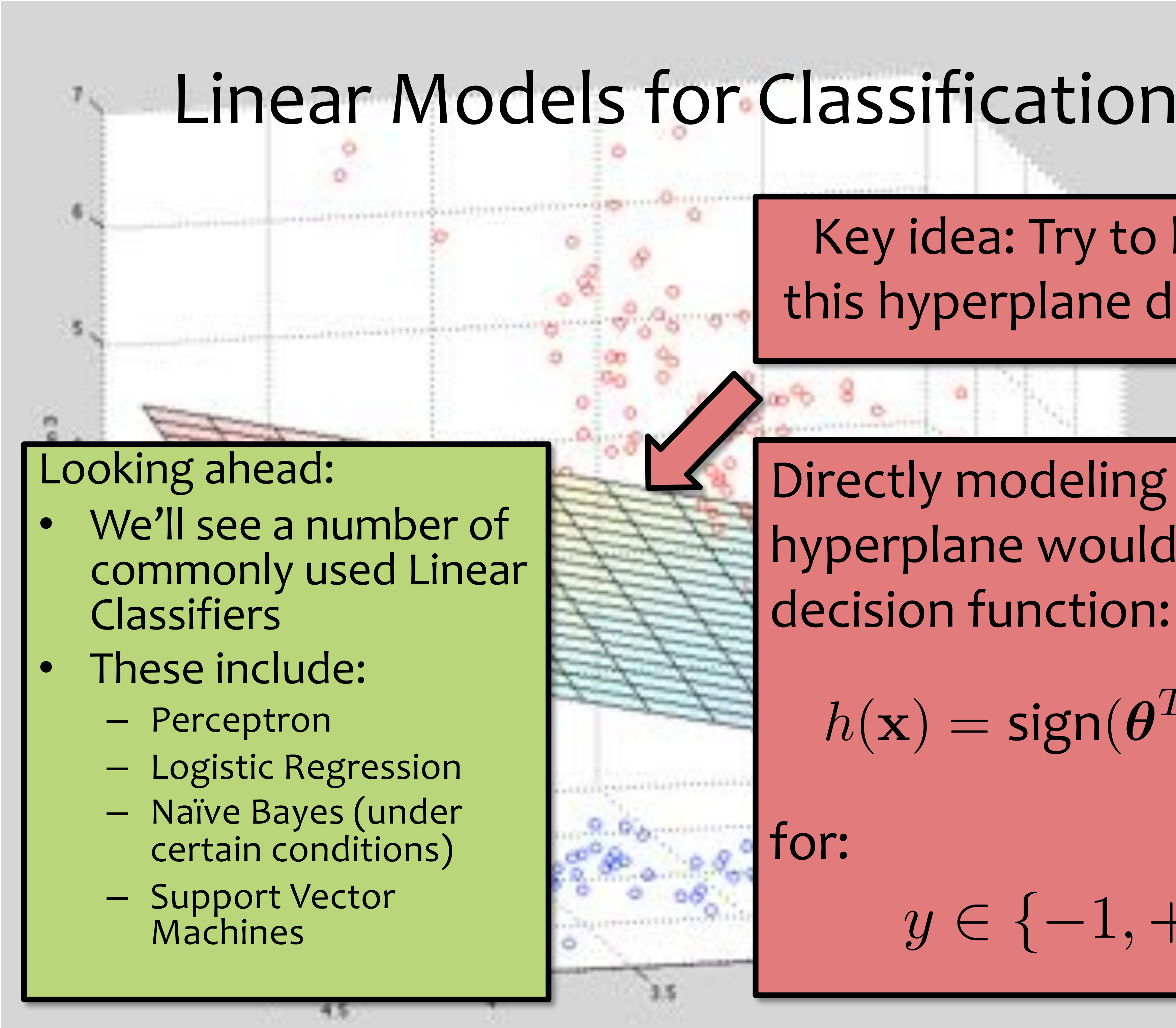




The New Yorker, December 6, 1958 P. 44

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

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

In-Class Exercise

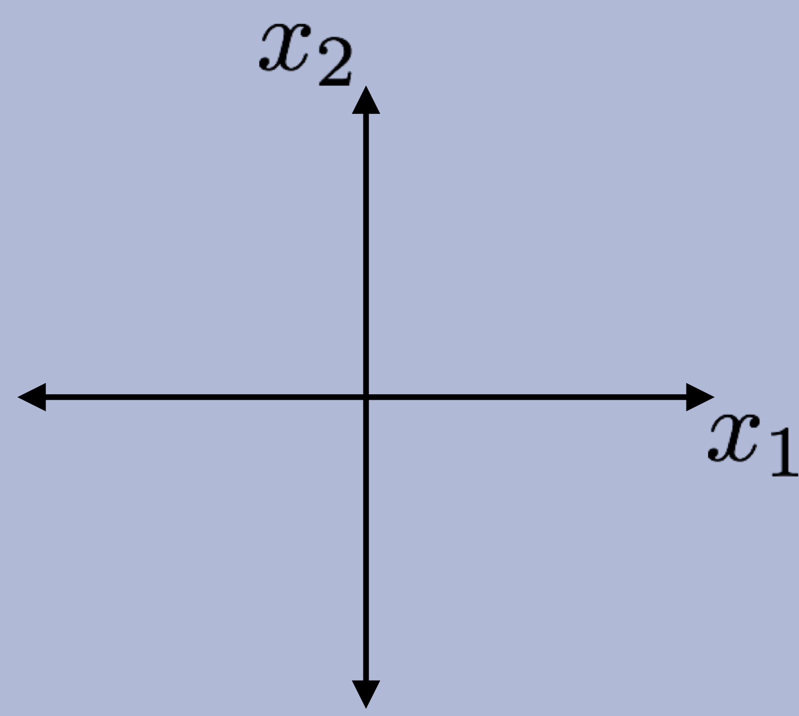
Draw a picture of the region corresponding to:

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

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

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

Answer Here:



Visualizing Dot-Products

Whiteboard:

- definition of dot product
- definition of L2 norm
- definition of orthogonality

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

⚠ When survey is active, respond at **pollev.com/10301601polls**

Lecture 6: In-Class Poll

1 done

2 underway

When poll is active, respond at **pollev.com/10301601polls**

Question 1

A

B

C

D

E

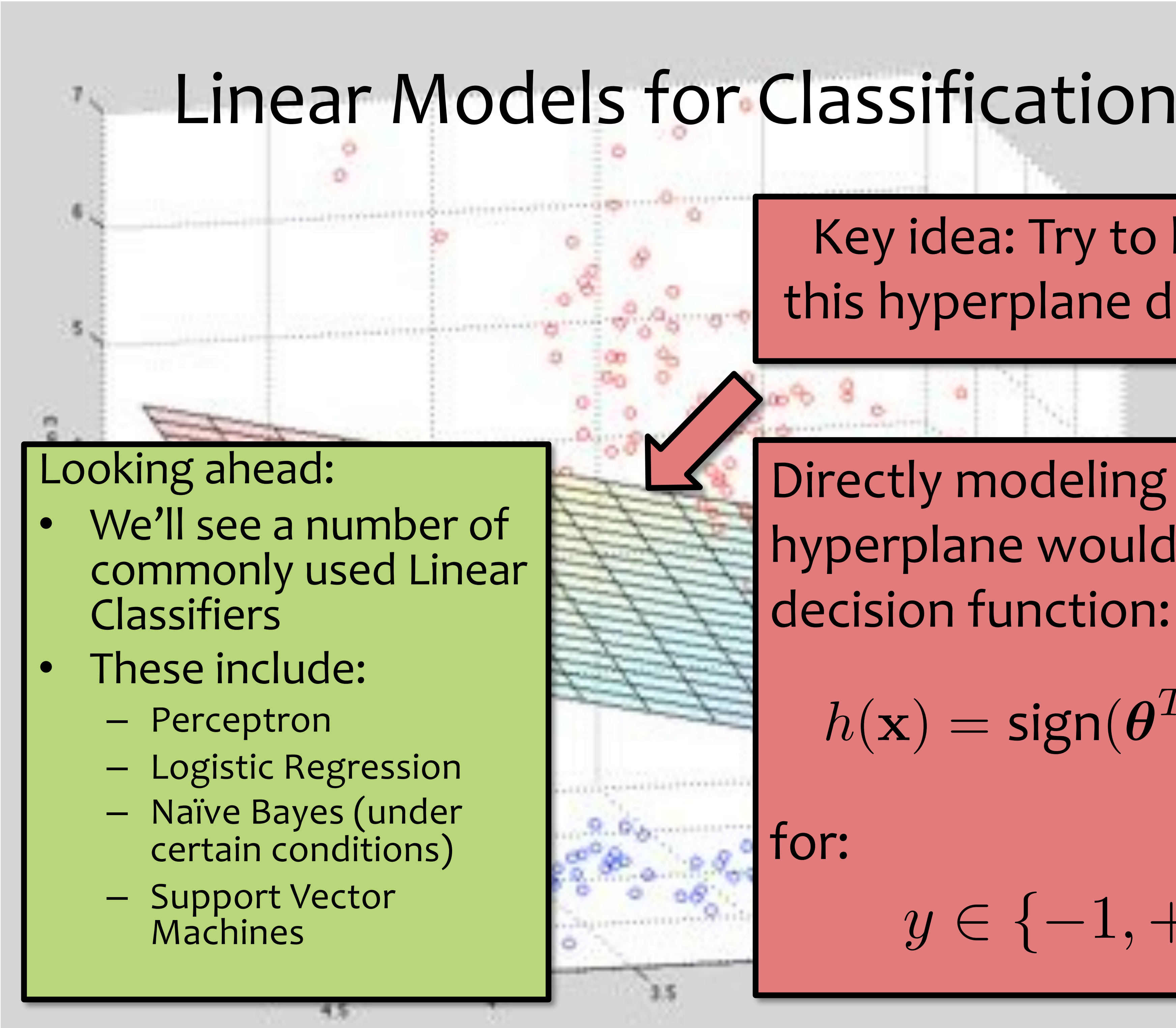
F

Visualizing Dot-Products

Whiteboard:

- – vector projection
- – hyperplane definition
- – half-space definitions

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 vs. Batch Learning

Batch Learning

Learn from all the examples at once

Online Learning

Gradually learn as each example is received

Online Learning

Examples

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 a mistake was made, $y' \neq y^{(i)}$
- **Update** parameters $\boldsymbol{\theta}$

Goal:

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

THE PERCEPTRON ALGORITHM

Perceptron

Whiteboard:

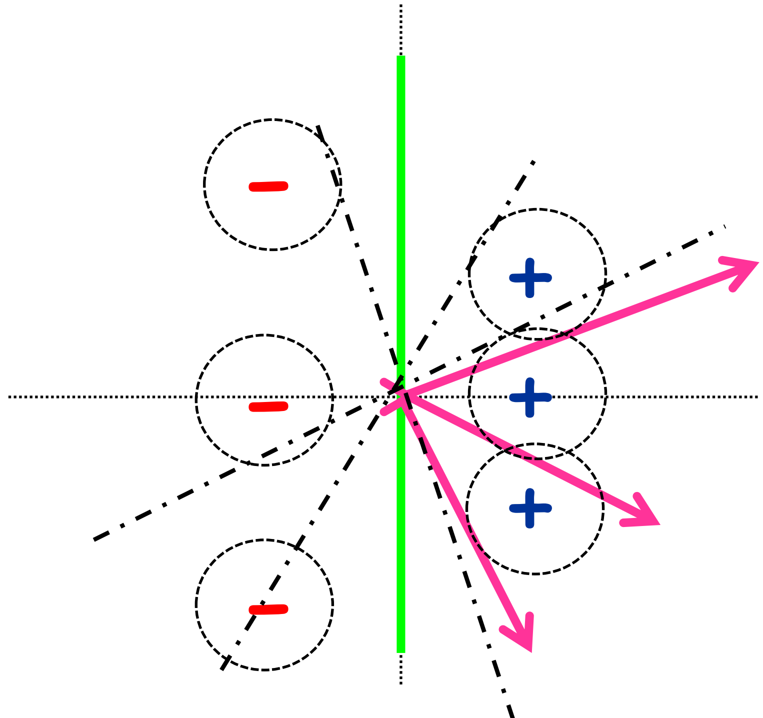
– (Online) Perceptron Algorithm

– Hypothesis class for Perceptron

– 2D Example of Perceptron

Perceptron Algorithm: Example

Example: $(-1,2) -$ ✗
$(1,0) +$ ✓
$(1,1) +$ ✗
$(-1,0) -$ ✓
$(-1,-2) -$ ✗
$(1,-1) +$ ✓



Perceptron Algorithm: (without the bias 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 $w_{t+1} \leftarrow w_t - x$

$w_1 = (0,0)$

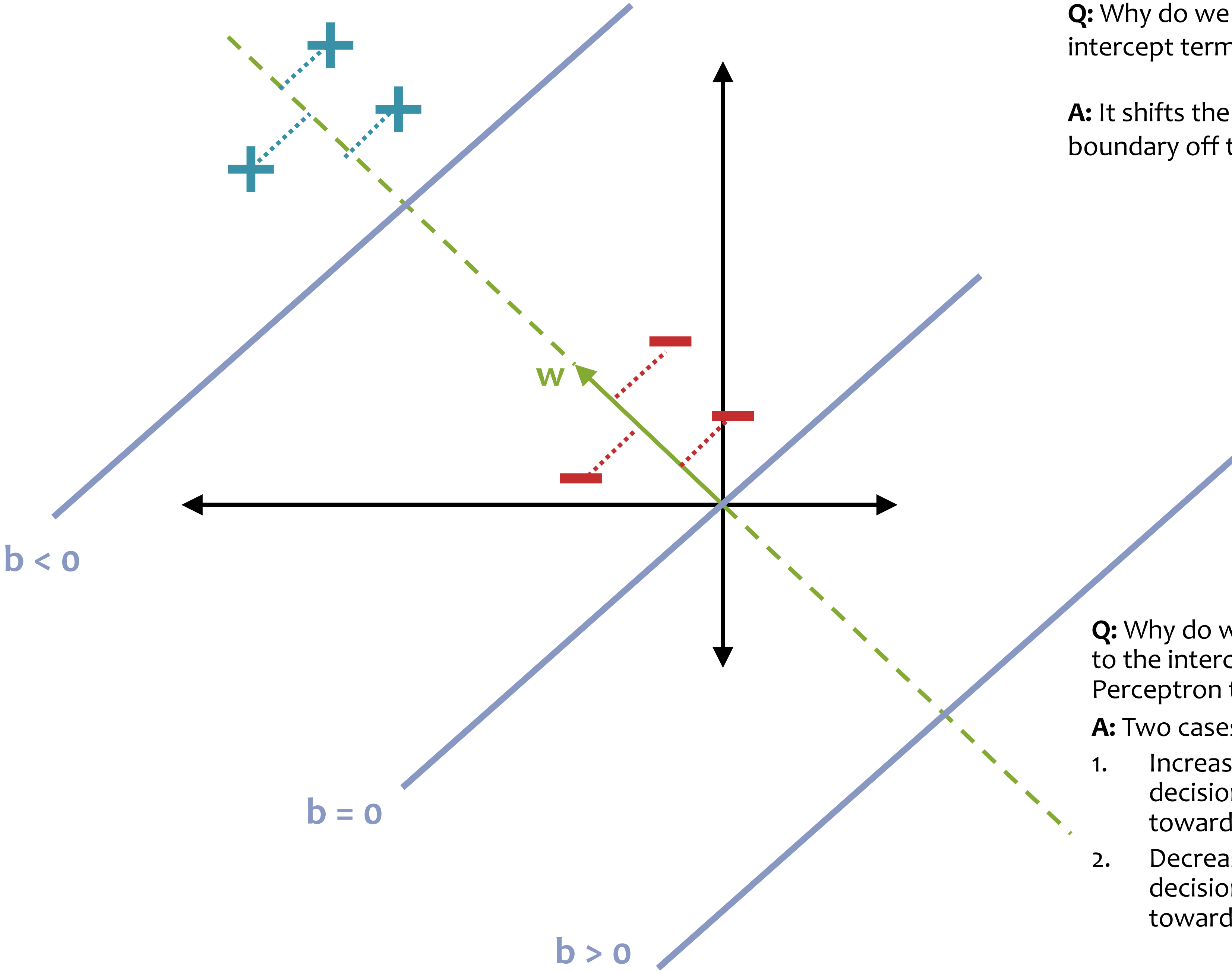
$w_2 = w_1 - (-1,2) = (1,-2)$

$w_3 = w_2 + (1,1) = (2,-1)$

$w_4 = w_3 - (-1,-2) = (3,1)$

Slide adapted from Nina Balcan

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?

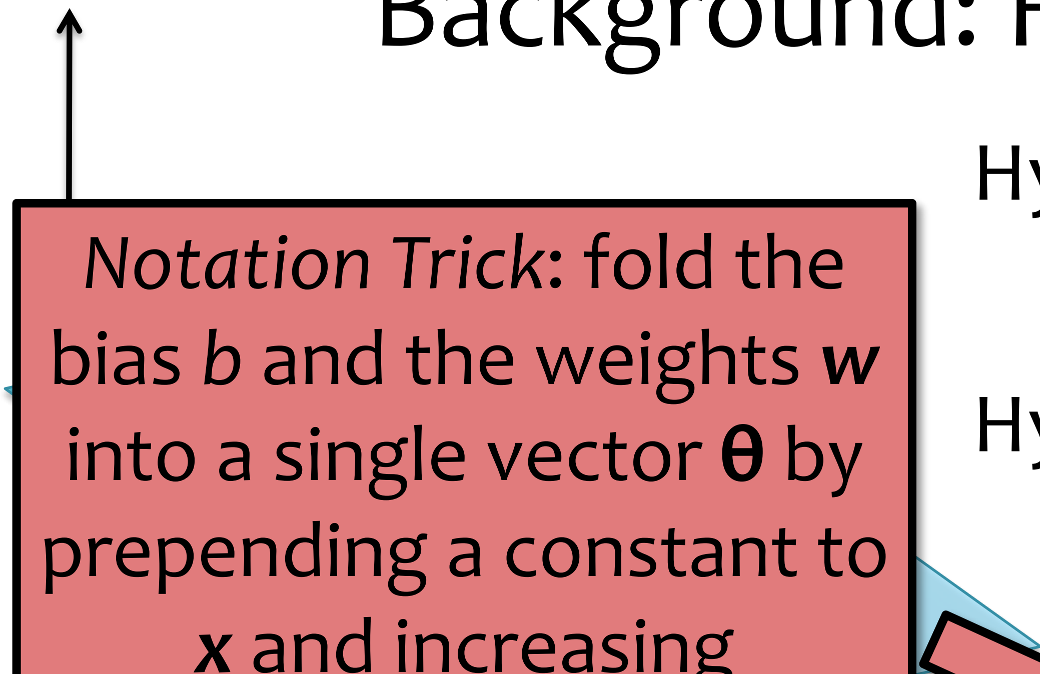
A: Two cases

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

Perceptron Inductive Bias

1. Decision boundary should be linear
2. Most recent mistakes are most important (and should be corrected)

Background: Hyperplanes



Notation Trick: fold the bias b and the weights $\mathbf{w}$ into a single vector $\boldsymbol{\theta}$ by prepending a constant to $\mathbf{x}$ and increasing dimensionality by one to get $\mathbf{x}'$!

Hyperplane (Definition 1):

$$\mathcal{H} = \{\mathbf{x} : \mathbf{w}^T \mathbf{x} = b\}$$

Hyperplane (Definition 2):

$$\mathcal{H} = \{\mathbf{x}' : \boldsymbol{\theta}^T \mathbf{x}' = 0$$

$$\text{and } x'_1 = 1\}$$

$$\boldsymbol{\theta} = [b, w_1, \ldots, w_M]^T$$

Half-spaces:

$$\mathcal{H}^+ = \{\mathbf{x} : \boldsymbol{\theta}^T \mathbf{x} > 0 \text{ and } x_1 = 1\}$$

$$\mathcal{H}^- = \{\mathbf{x} : \boldsymbol{\theta}^T \mathbf{x} < 0 \text{ and } x_1 = 1\}$$

(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

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:

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** $\boldsymbol{\theta}$

(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 determine

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

Assume $\boldsymbol{\theta} = [b, w_1, \ldots, w_M]$

Learning:

Implementation Trick: same behavior as our "*add on positive mistake and subtract on negative mistake*" version, because $y^{(i)}$ takes care of the sign

Algorithm 1 Perceptron Learning Alg

```
procedure PERCEPTRON(D = {(x
    θ ← 0                                  ▷ Initialize parameters
    for i ∈ {1, 2, ...} do                 ▷ For each example
        ŷ ← sign(θ^T x^(i))                ▷ Predict
        if ŷ ≠ y^(i) then                  ▷ If mistake
            θ ← θ + y^(i) x^(i)            ▷ Update parameters
    return θ
```

(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($\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(N)}, y^{(N)})\}$)
  $\boldsymbol{\theta} \leftarrow \mathbf{0}$ ▷ Initialize parameters
  while not converged **do**
    for $i \in \{1, 2, \ldots, N\}$ **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.

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

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

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*

When poll is active, respond at **pollev.com/10301601polls**

Question 2

A

B

C

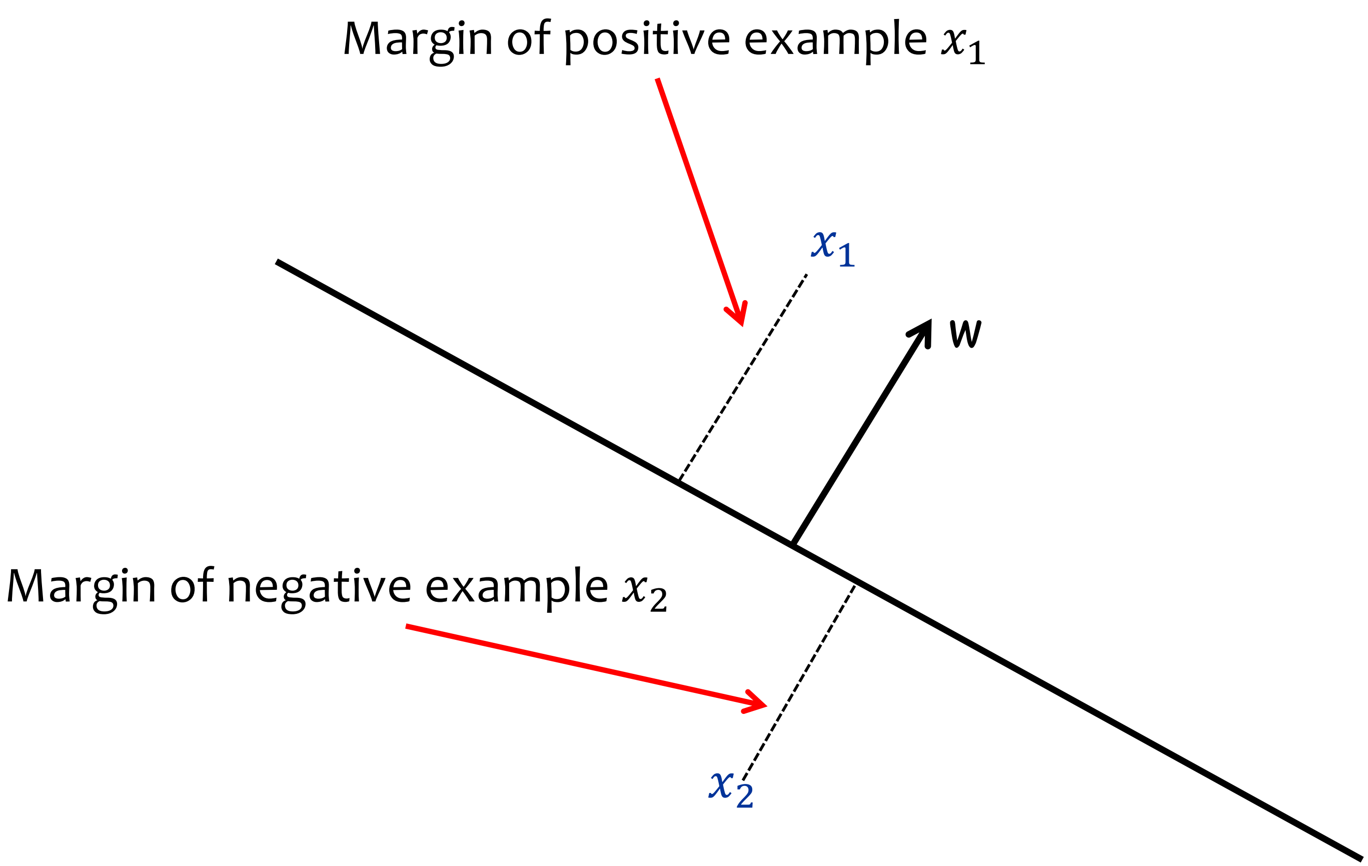
D

E

ANALYSIS OF PERCEPTRON

Geometric Margin

Definition: The margin of example x w.r.t. a linear separator w is the distance from x to the plane $w \cdot x = 0$ (or the negative if on wrong side)

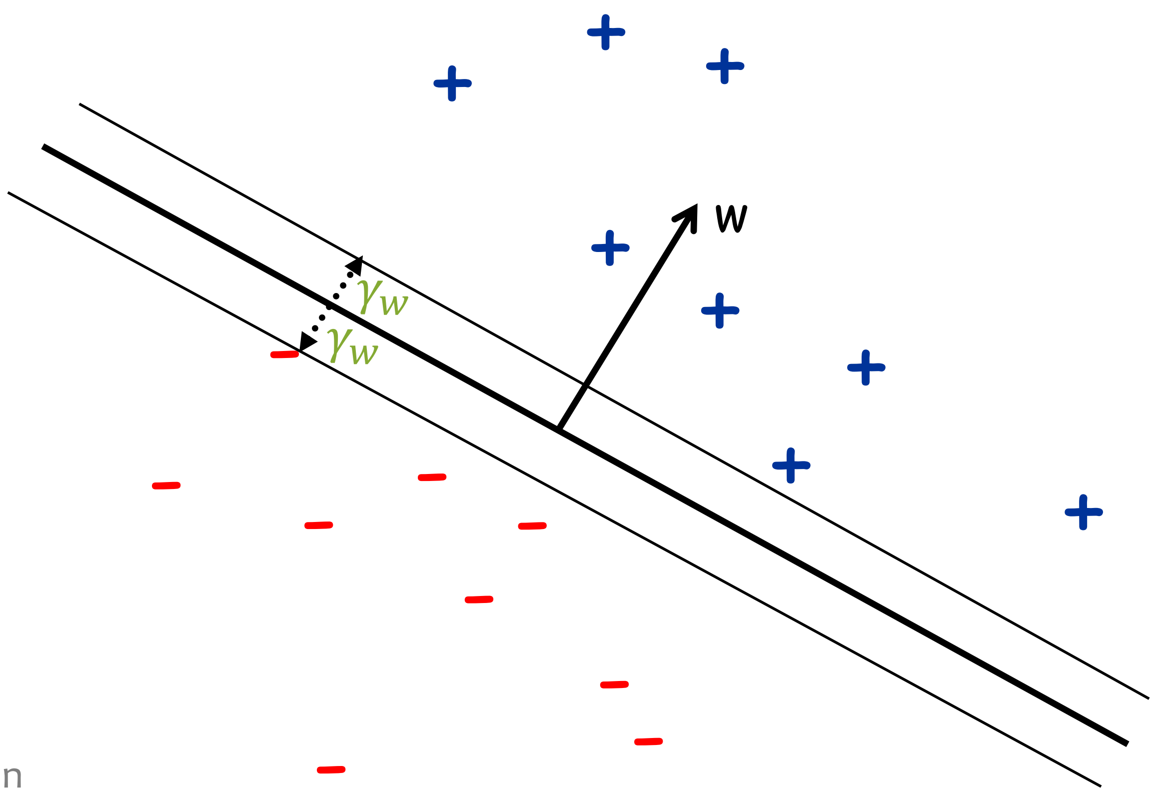


Slide from Nina Balcan

Geometric Margin

Definition: The margin of example x w.r.t. a linear separator w is the distance from x to the plane $w \cdot x = 0$ (or the negative if on wrong side)

Definition: The margin γ_w of a set of examples S w.r.t. a linear separator w is the smallest margin over points $x \in S$.



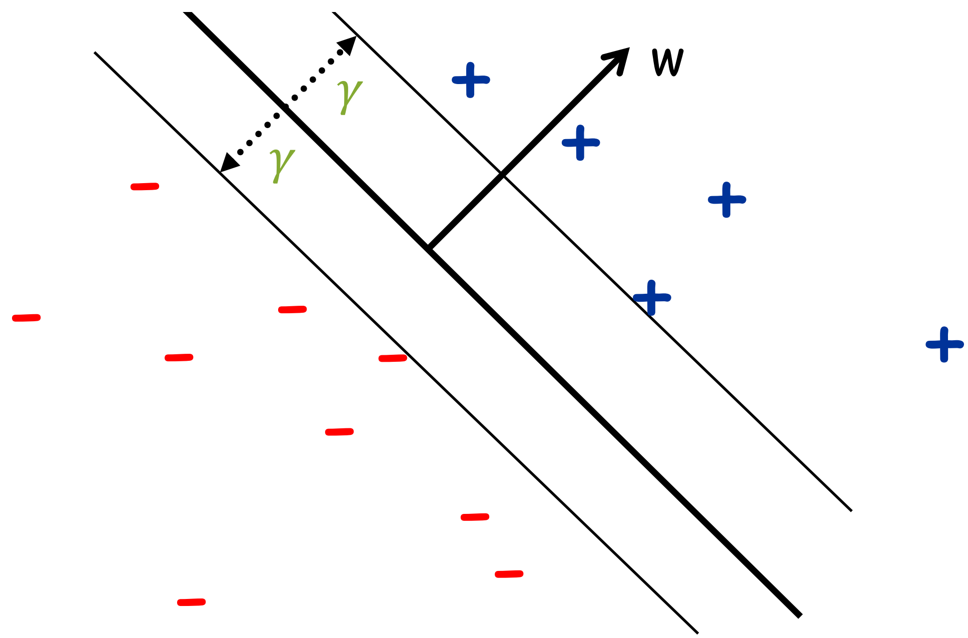
Slide from Nina Balcan

Geometric Margin

Definition: The margin of example x w.r.t. a linear separator w is the distance from x to the plane $w \cdot x = 0$ (or the negative if on wrong side)

Definition: The margin γ_w of a set of examples S w.r.t. a linear separator w is the smallest margin over points $x \in S$.

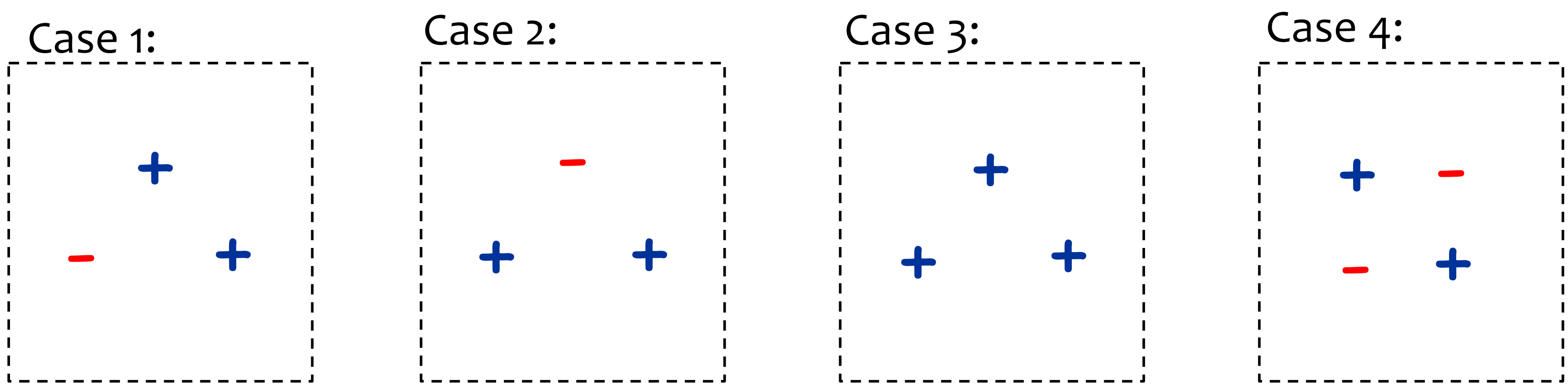
Definition: The margin γ of a set of examples S is the maximum γ_w over all linear separators w.



Slide from Nina Balcan

Linear Separability

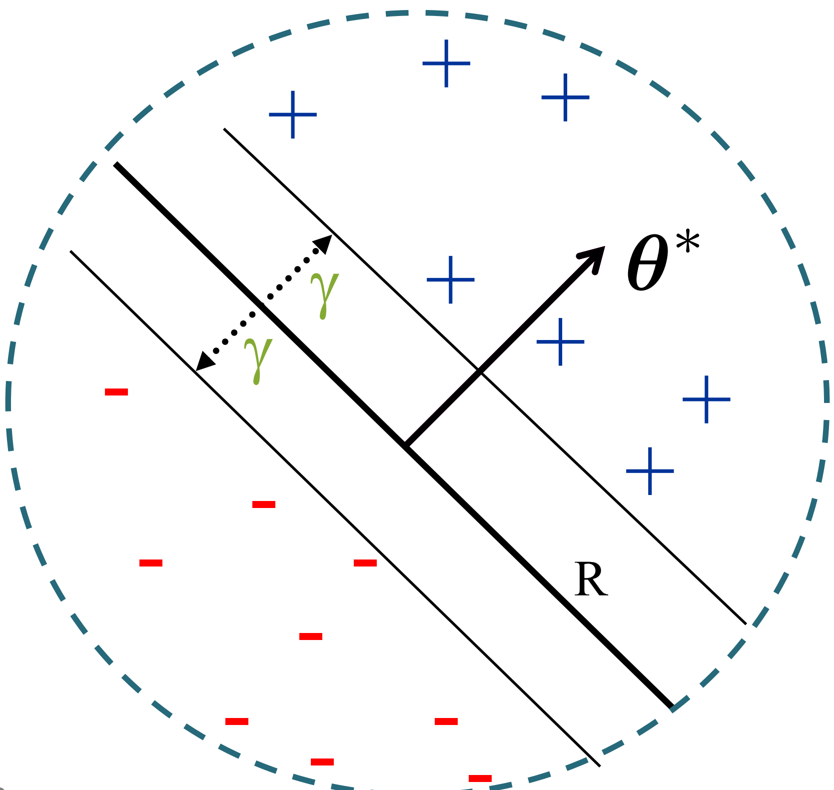
Def: For a **binary classification** problem, a set of examples S is **linearly separable** if there exists a linear decision boundary that can separate the points



Perceptron Mistake Bound

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

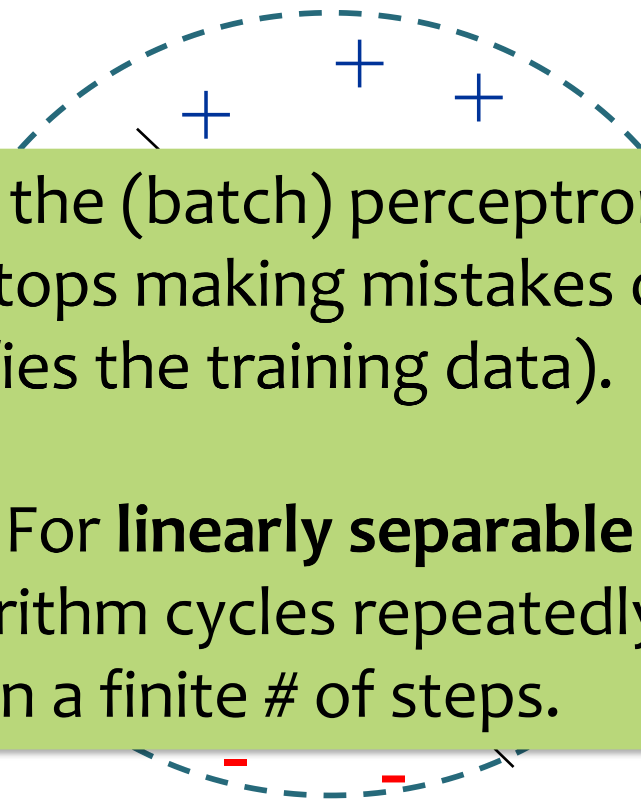
(Normalized margin: multiplying all points by 100, or dividing all points by 100, doesn't change the number of mistakes! The algorithm is invariant to scaling.)



Perceptron Mistake Bound

Guarantee: if some data has margin γ and all points lie inside a ball of radius R, 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.)



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

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

ANALYSIS OF PERCEPTRON (PROOF)

Analysis: Perceptron

Perceptron Mistake Bound

Theorem 0.1 (Block (1962), Novikoff (1962)).
Given dataset: $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$.
Suppose:

1. *Finite size inputs:* $||x^{(i)}|| \leq R$
2. *Linearly separable data:* $\exists \boldsymbol{\theta}^*$ *and* $\gamma > 0$ *s.t.* $||\boldsymbol{\theta}^*|| = 1$ *and* $y^{(i)}(\boldsymbol{\theta}^* \cdot \mathbf{x}^{(i)}) \geq \gamma, \forall i$

Then: The number of mistakes made by the Perceptron algorithm on this dataset is

$$k \leq (R/\gamma)^2$$

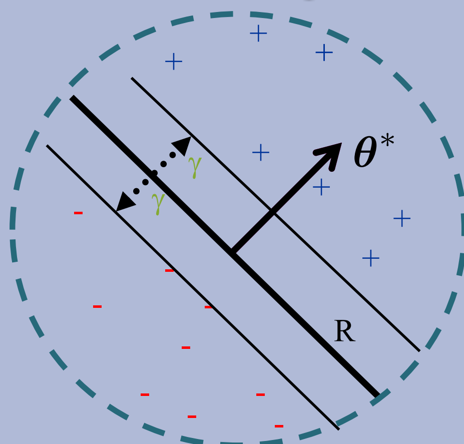


Figure from Nina Balcan

Analysis: Percept

Common *Misunderstanding*:
The **radius** is **centered at the origin,** not at the center of the points.

Perceptron Mistake Boun

Theorem 0.1 (Block (1962), Novikoff (1

Given dataset: $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$

Suppose:

1. *Finite size inputs:* $||x^{(i)}|| \leq R$
2. *Linearly separable data:* $\exists \boldsymbol{\theta}^*$ *and* $\gamma > 0$ *s.t.* $||\boldsymbol{\theta}^*|| = 1$ *and* $y^{(i)}(\boldsymbol{\theta}^* \cdot \mathbf{x}^{(i)}) \geq \gamma, \forall i$

Then: The number of mistakes made by the Perceptron algorithm on this dataset is

$$k \leq (R/\gamma)^2$$

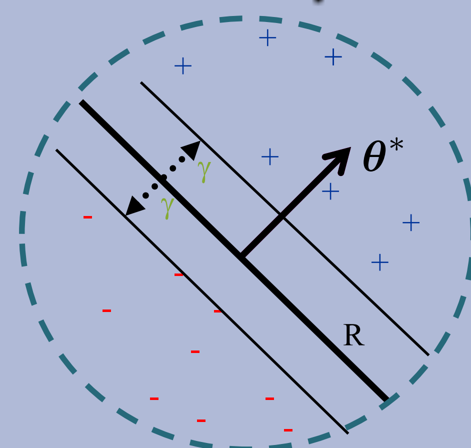


Figure from Nina Balcan

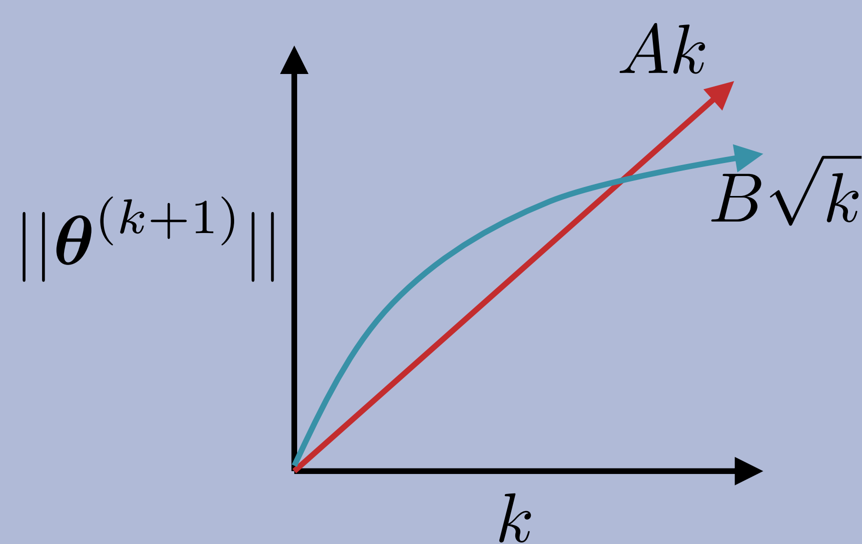
Analysis: Perceptron

Proof of Perceptron Mistake Bound:

We will show that there exist constants A and B s.t.

$$Ak \leq ||\boldsymbol{\theta}^{(k+1)}|| \leq B\sqrt{k}$$

parameters after k'th mistake



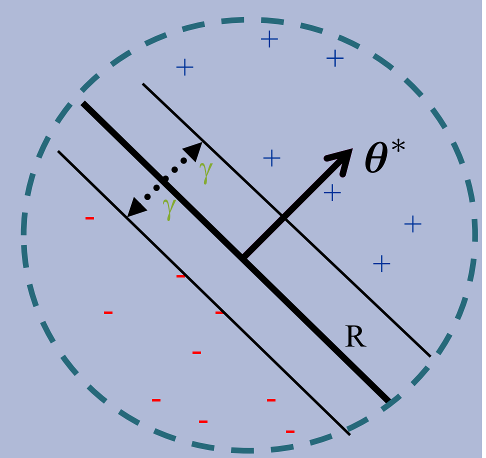
Analysis: Perceptron

Theorem 0.1 (Block (1962), Novikoff (1962)).
Given dataset: $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$.
Suppose:

1. *Finite size inputs:* $||x^{(i)}|| \leq R$
2. *Linearly separable data:* $\exists \boldsymbol{\theta}^*$ *s.t.* $||\boldsymbol{\theta}^*|| = 1$ *and* $y^{(i)}(\boldsymbol{\theta}^* \cdot \mathbf{x}^{(i)}) \geq \gamma, \forall i$

Then: The number of mistakes made by the Perceptron algorithm on this dataset is

$$k \leq (R/\gamma)^2$$



Algorithm 1 Perceptron Learning Algorithm (Online)

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

Analysis: Perceptron

Proof of Perceptron Mistake Bound:

Part 1: for some A, $Ak \leq ||\boldsymbol{\theta}^{(k+1)}||$

$$\boldsymbol{\theta}^{(k+1)} \cdot \boldsymbol{\theta}^* = (\boldsymbol{\theta}^{(k)} + y^{(i)}\mathbf{x}^{(i)})\boldsymbol{\theta}^*$$

by Perceptron algorithm update

$$= \boldsymbol{\theta}^{(k)} \cdot \boldsymbol{\theta}^* + y^{(i)}(\boldsymbol{\theta}^* \cdot \mathbf{x}^{(i)})$$

$$\geq \boldsymbol{\theta}^{(k)} \cdot \boldsymbol{\theta}^* + \gamma$$

by assumption

$$\Rightarrow \boldsymbol{\theta}^{(k+1)} \cdot \boldsymbol{\theta}^* \geq k\gamma$$

by induction on k since $\theta^{(1)} = \mathbf{0}$

$$\Rightarrow ||\boldsymbol{\theta}^{(k+1)}|| \geq k\gamma$$

since $||\mathbf{w}|| \times ||\mathbf{u}|| \geq \mathbf{w} \cdot \mathbf{u}$ and $||\theta^*|| = 1$

Cauchy-Schwartz inequality

Analysis: Perceptron

Proof of Perceptron Mistake Bound:

Part 2: for some B, $||\boldsymbol{\theta}^{(k+1)}|| \leq B\sqrt{k}$

$$||\boldsymbol{\theta}^{(k+1)}||^2 = ||\boldsymbol{\theta}^{(k)} + y^{(i)}\mathbf{x}^{(i)}||^2$$

by Perceptron algorithm update

$$= ||\boldsymbol{\theta}^{(k)}||^2 + (y^{(i)})^2||\mathbf{x}^{(i)}||^2 + 2y^{(i)}(\boldsymbol{\theta}^{(k)} \cdot \mathbf{x}^{(i)})$$

$$\leq ||\boldsymbol{\theta}^{(k)}||^2 + (y^{(i)})^2||\mathbf{x}^{(i)}||^2$$

since kth mistake $\Rightarrow y^{(i)}(\boldsymbol{\theta}^{(k)} \cdot \mathbf{x}^{(i)}) \leq 0$

$$= ||\boldsymbol{\theta}^{(k)}||^2 + R^2$$

since $(y^{(i)})^2||\mathbf{x}^{(i)}||^2 = ||\mathbf{x}^{(i)}||^2 = R^2$ by assumption and $(y^{(i)})^2 = 1$

$$\Rightarrow ||\boldsymbol{\theta}^{(k+1)}||^2 \leq kR^2$$

by induction on k since $(\theta^{(1)})^2 = 0$

$$\Rightarrow ||\boldsymbol{\theta}^{(k+1)}|| \leq \sqrt{k}R$$

Analysis: Perceptron

Proof of Perceptron Mistake Bound:

Part 3: Combining the bounds finishes the proof.

$$k\gamma \leq ||\boldsymbol{\theta}^{(k+1)}|| \leq \sqrt{k}R$$

$$\Rightarrow k \leq (R/\gamma)^2$$

The total number of mistakes must be less than this

Analysis: Perceptron

What if the data is *not* linearly separable?

1. Perceptron will **not converge** in this case (it can't!)
2. However, Freund & Schapire (1999) show that by projecting the points (hypothetically) into a higher dimensional space, we can achieve a similar bound on the number of mistakes made on **one pass** through the sequence of examples

Theorem 2. *Let* $\langle(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\rangle$ *be a sequence of labeled examples with* $\|\mathbf{x}_i\| \leq R$. *Let* $\mathbf{u}$ *be any vector with* $\|\mathbf{u}\| = 1$ *and let* $\gamma > 0$. *Define the deviation of each example as*

$$d_i = \max\{0, \gamma - y_i(\mathbf{u} \cdot \mathbf{x}_i)\},$$

and define $D = \sqrt{\sum_{i=1}^m d_i^2}$. *Then the number of mistakes of the online perceptron algorithm on this sequence is bounded by*

$$\left(\frac{R + D}{\gamma}\right)^2.$$

Perceptron Exercises

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*

When poll is active, respond at **pollev.com/10301601polls**

Question 3

A

B

C

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Summary: Perceptron

- Perceptron is a **linear classifier**
- **Simple learning algorithm**: when a mistake is made, add / subtract the features
- Perceptron will converge if the data are **linearly separable**, it will **not** converge if the data are **linearly inseparable**
- For linearly separable and inseparable data, we can **bound the number of mistakes** (geometric argument)
- **Extensions** support nonlinear separators and structured prediction

Perceptron Learning Objectives

You should be able to…

- Explain the difference between online learning and batch learning
- Implement the perceptron algorithm for binary classification [CIML]
- Determine whether the perceptron algorithm will converge based on properties of the dataset, and the limitations of the convergence guarantees
- Describe the inductive bias of perceptron and the limitations of linear models
- Draw the decision boundary of a linear model
- Identify whether a dataset is linearly separable or not
- Defend the use of a bias term in perceptron