10-301/601: Introduction to Machine Learning
Lecture 6 – Perceptron

Geoff Gordon

with thanks to Henry Chai & Matt Gormley

Q & A:

Can we use k-NN with categorical features?

Q & A:

Can we use k-NN with categorical features?

- Yes! We can either:
  1. Convert categorical features into binary ones:

Cholesterol
Normal
Normal
Abnormal

→

Normal Cholesterol?	Abnormal Cholesterol?
1	0
1	0
0	1

  2. Use a distance metric that works over categorical features, e.g., the Hamming distance:

$$d(x, x') = \sum_{i=1}^{M} \mathbb{I}(x_i \neq x'_i)$$

(red square) (red triangle)

Fisher Iris Dataset

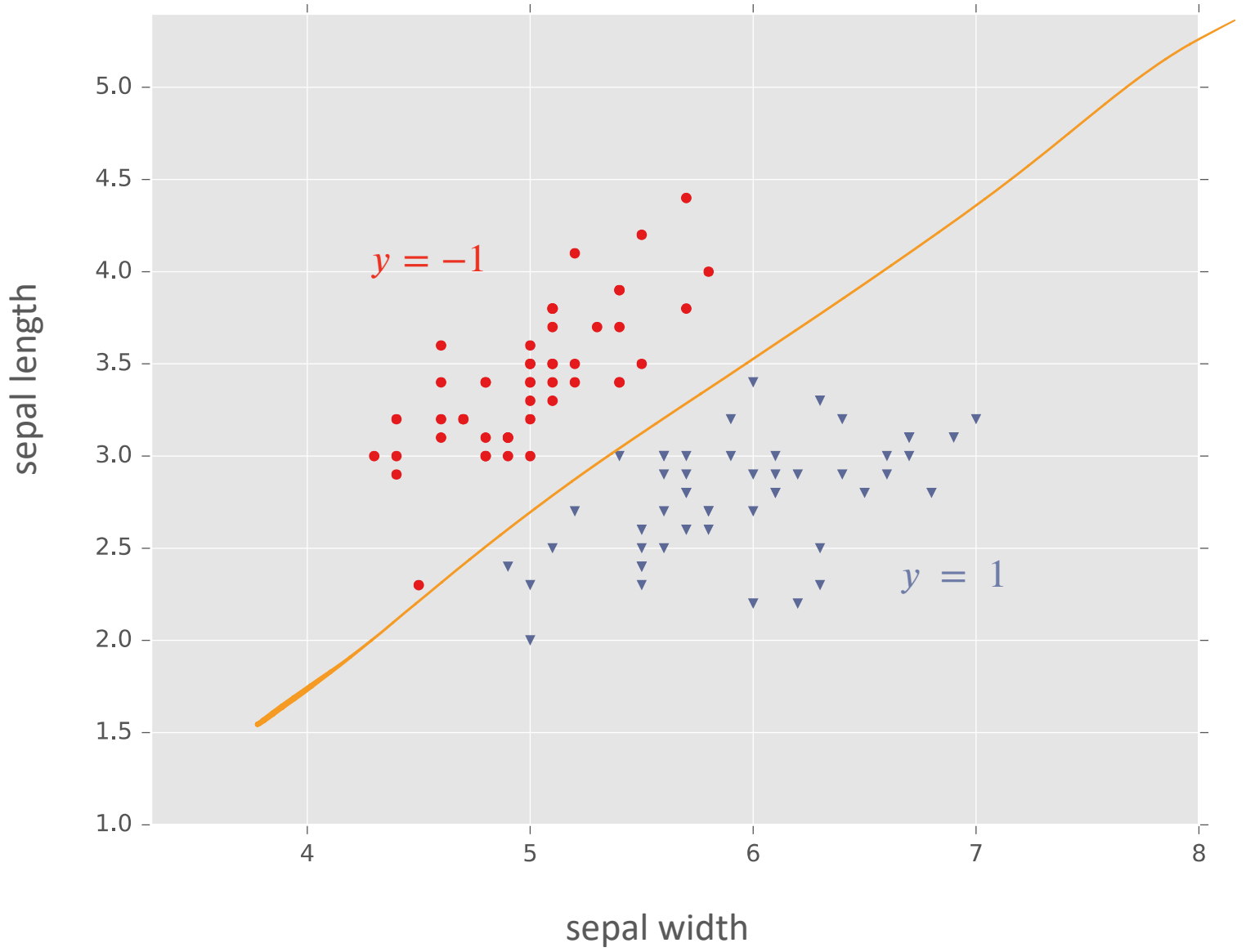


Figure courtesy of Matt Gormley

Linear Algebra Review

- Notation: in this we use column vectors by default, i.e.,

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

- The dot product between two D-dimensional vectors is

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

Linear Algebra Review

- Dot products represent linear functions: $\boldsymbol{w}^\top \boldsymbol{x}$ is linear in $\boldsymbol{x}$ for fixed $\boldsymbol{w}$ (and all linear functions of $\boldsymbol{x}$ that pass through origin can be written this way)
  - include intercept $\boldsymbol{w}^\top \boldsymbol{x} + b$ to not hit origin
- The $L2$-norm of $\boldsymbol{a} = \|\boldsymbol{a}\|_2 = \sqrt{\boldsymbol{a}^T \boldsymbol{a}}$
- Two vectors are *orthogonal* iff $\boldsymbol{a}^T \boldsymbol{b} = 0$

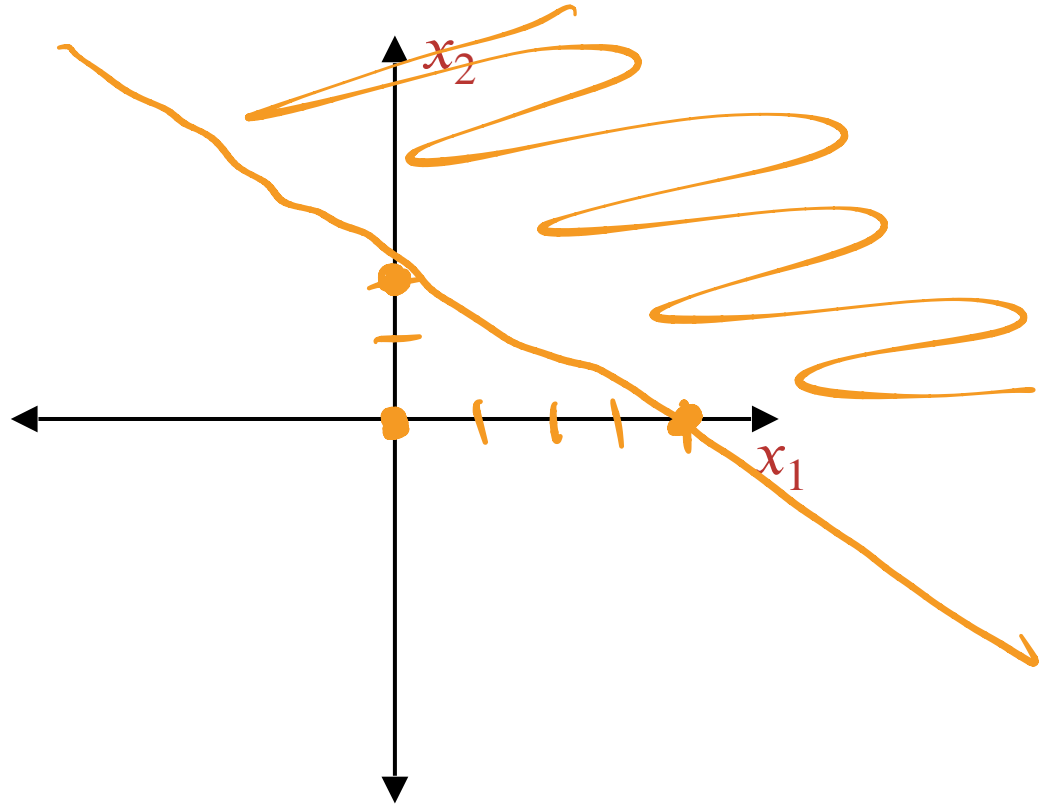
Geometry Warm-up

1. On the axes below, draw the region corresponding to

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

where $w_1 = 1$, $w_2 = 2$ and $b = -4$.

2. Then draw the vector $w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$



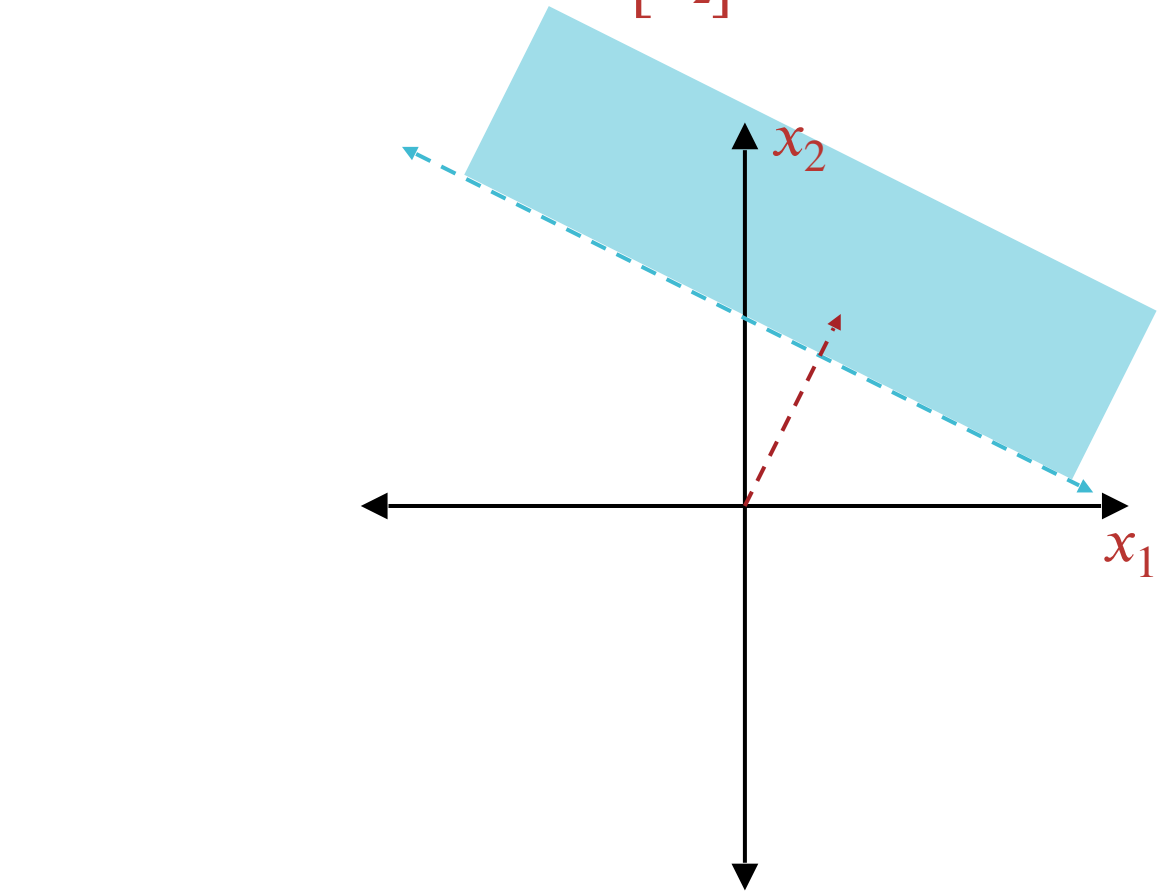
Geometry Warm-up

1. On the axes below, draw the region corresponding to

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

where $w_1 = 1$, $w_2 = 2$ and $b = -4$.

2. Then draw the vector $w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$



Linear Decision Boundaries

- In 2 dimensions, $w_1 x_1 + w_2 x_2 + b = 0$ defines a *line*
- In 3 dimensions, $w_1 x_1 + w_2 x_2 + w_3 x_3 + b = 0$ 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*:
  - $\mathcal{S}_+ = \left\{ \boldsymbol{x} : \boldsymbol{w}^T \boldsymbol{x} + b > 0 \right\}$ or all $\boldsymbol{x}$ s.t. $\boldsymbol{w}^T \boldsymbol{x} + b$ is positive
  - $\mathcal{S}_- = \left\{ \boldsymbol{x} : \boldsymbol{w}^T \boldsymbol{x} + b < 0 \right\}$ or all $\boldsymbol{x}$ s.t. $\boldsymbol{w}^T \boldsymbol{x} + b$ is negative

Linear Decision Boundaries: Example

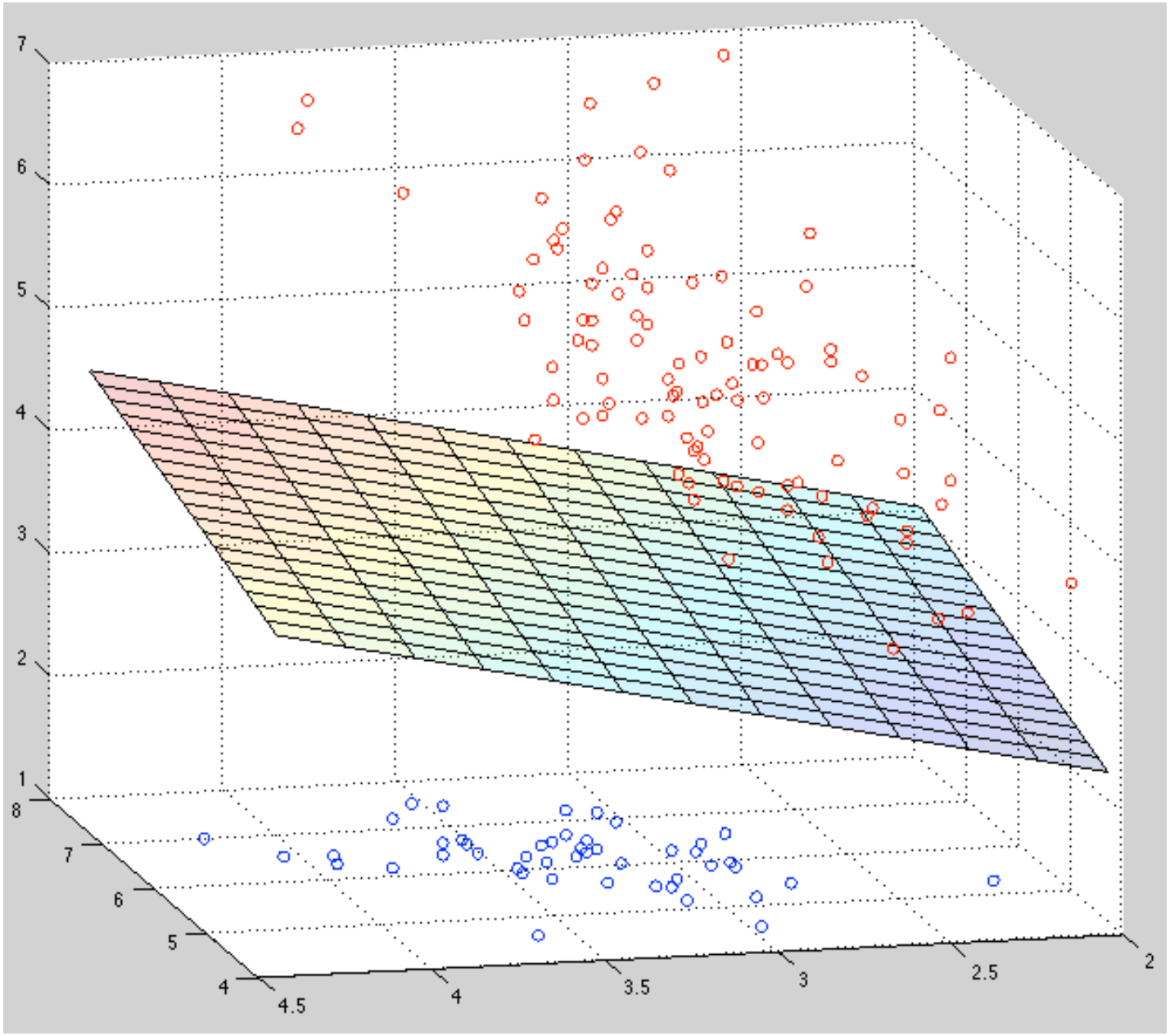
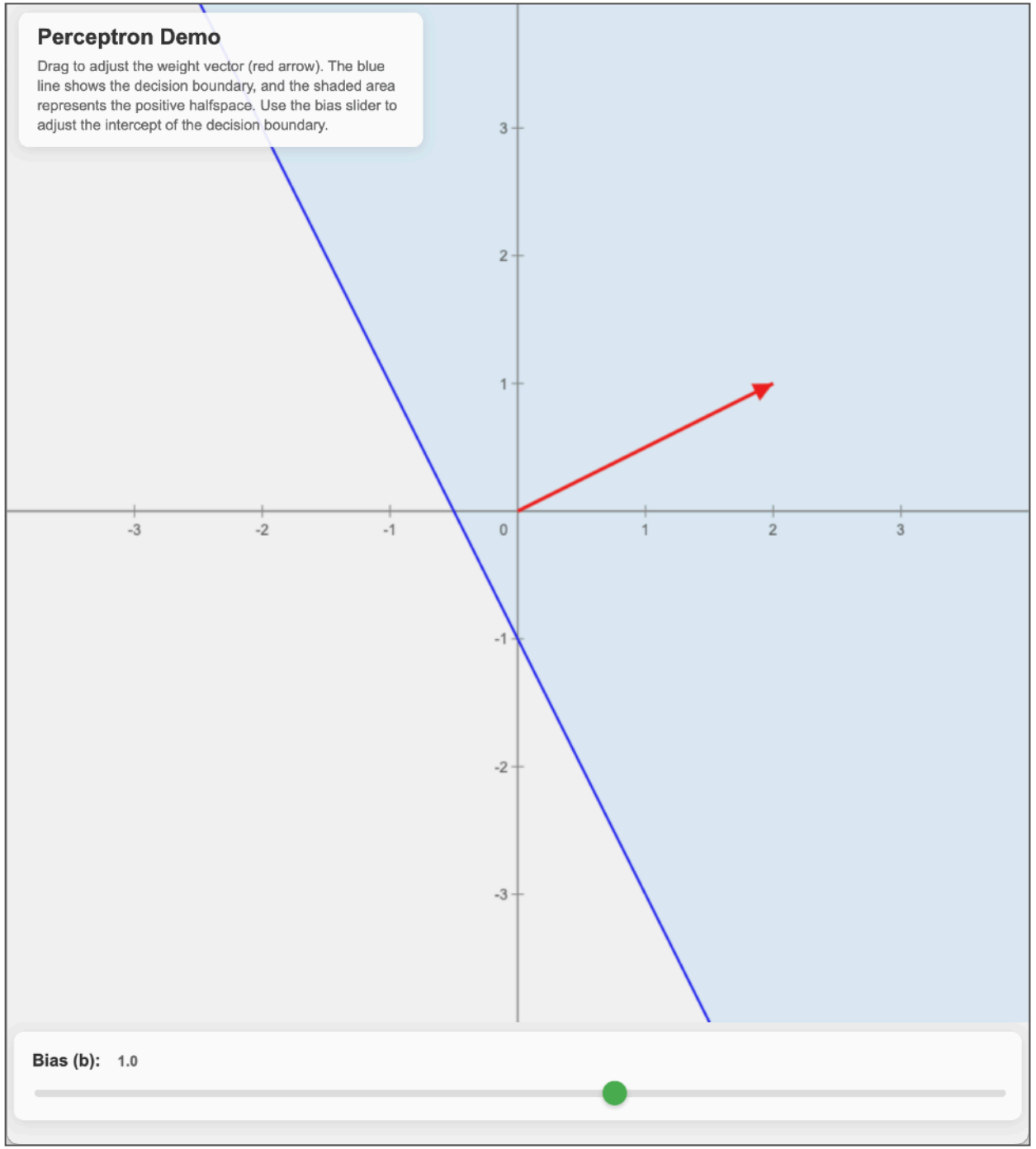
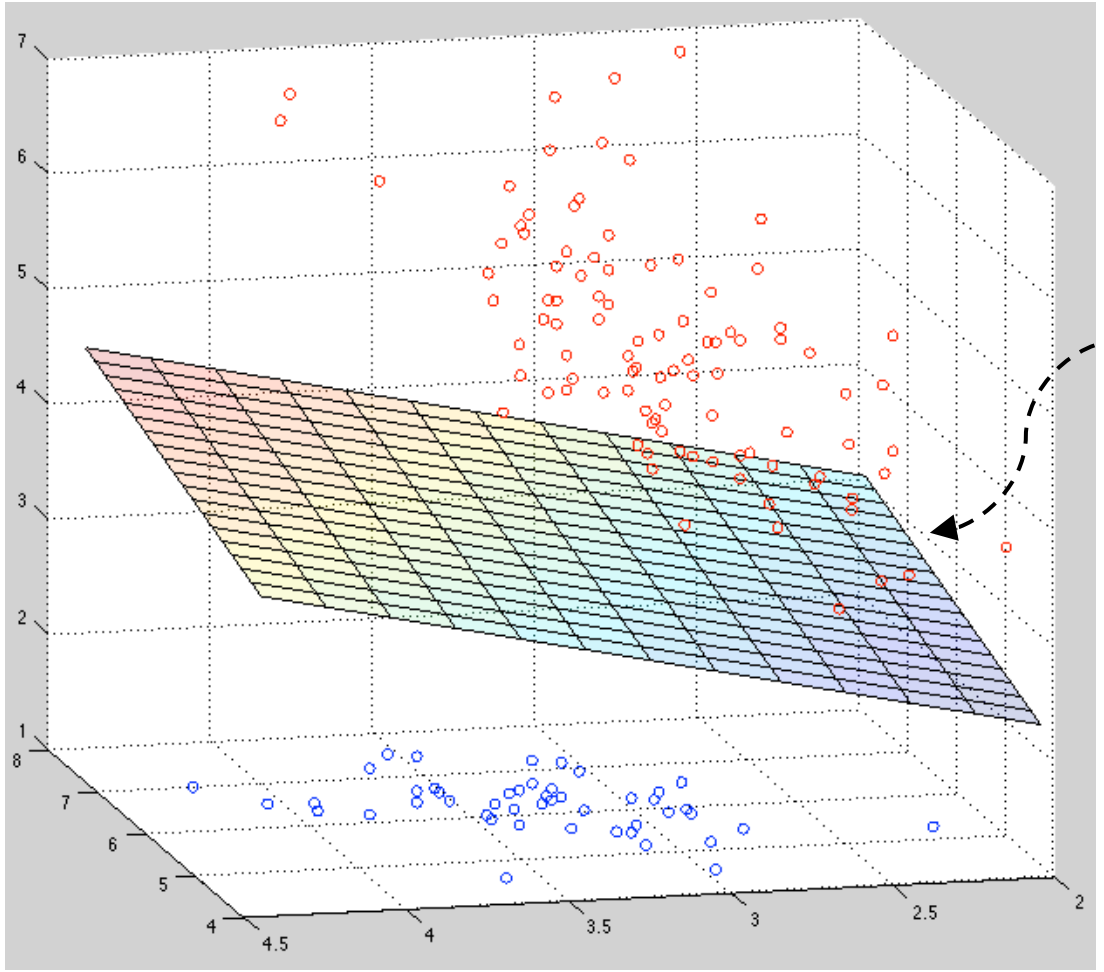


Figure courtesy of Matt Gormley

Interactive decision boundary



Learning a linear classifier



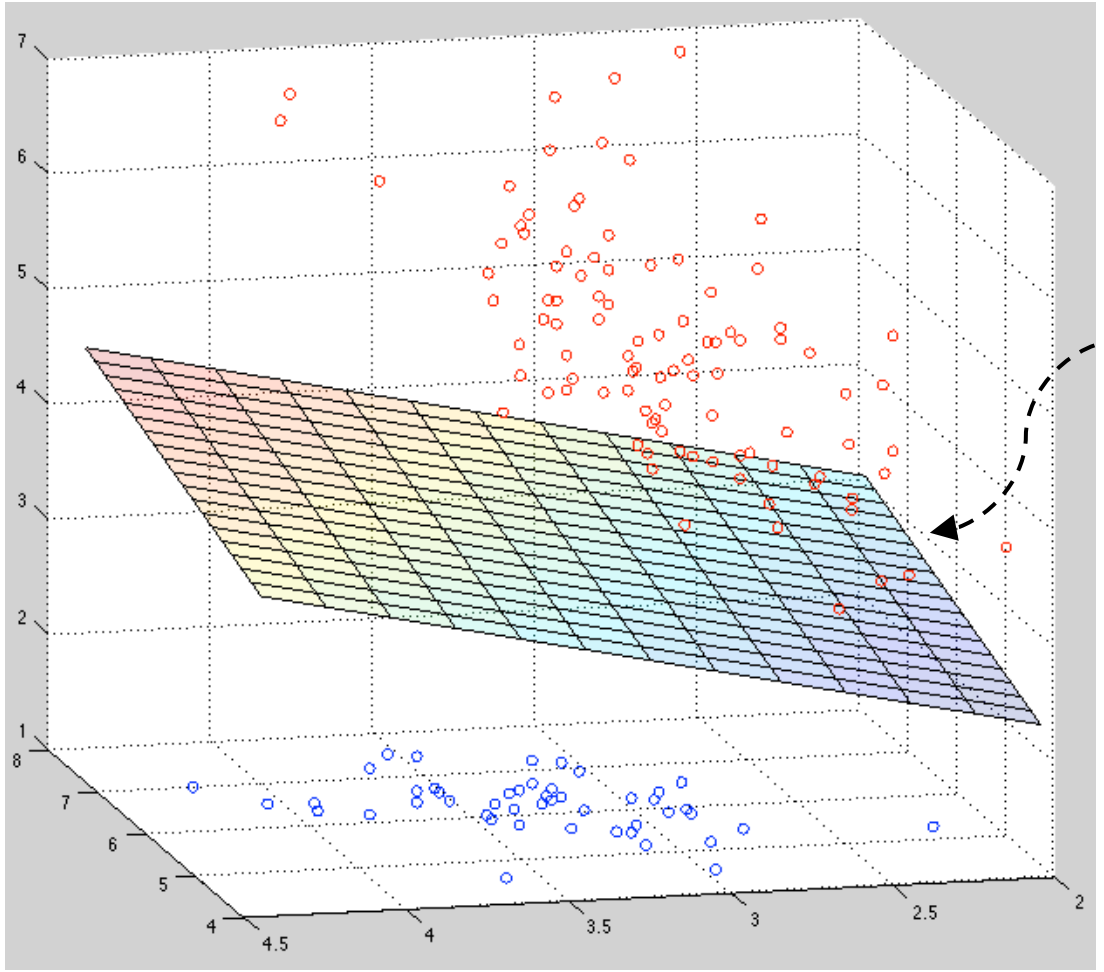
Goal: learn classifiers of the form

$$h(\boldsymbol{x}) = \text{sign}\left(\boldsymbol{w}^T\boldsymbol{x} + b\right)$$

(assuming $y \in \{-1, +1\}$)

Figure courtesy of Matt Gormley

Learning a linear classifier



Goal: learn classifiers of the form

$h(\boldsymbol{x}) = \text{sign}(\boldsymbol{w}^T\boldsymbol{x} + b)$

(assuming $y \in \{-1, +1\}$)

Key question: how do we learn the *parameters*, $\boldsymbol{w}, b$?

Figure courtesy of Matt Gormley

Online Learning

- So far, we've been learning in the *batch* setting, where we have access to the entire training dataset at once
- A common alternative is the *online* setting, where examples arrive gradually and we learn continuously
- Examples of online learning:
  - autocorrect — personalization
  - recommender system
  - performance tuning
  - stock prediction

Online Learning: Setup

- For $t = 1, 2, 3, \ldots$
  - Receive an unlabeled example, $\boldsymbol{x}^{(t)}$
  - Predict its label, $\hat{y} = h_{\boldsymbol{w},b}\left(\boldsymbol{x}^{(t)}\right)$
  - Observe its true label, $y^{(t)}$
  - Pay a penalty if we made a mistake, $\hat{y} \neq y^{(t)}$
  - Update the parameters, $\boldsymbol{w}$ and b
- Goal: minimize the number of mistakes made

(Online) Perceptron Learning 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, $\hat{y} = \text{sign}\left(\boldsymbol{w}^T\boldsymbol{x} + b\right) = \begin{cases} +1 \text{ if } \boldsymbol{w}^T\boldsymbol{x} + b \geq 0 \\ -1 \text{ otherwise} \end{cases}$
  - 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 ($y^{(t)} = -1$, $\hat{y} = +1$):
    - $\boldsymbol{w} \leftarrow \boldsymbol{w} - \boldsymbol{x}^{(t)}$
    - $b \leftarrow b - 1$

Notational hack

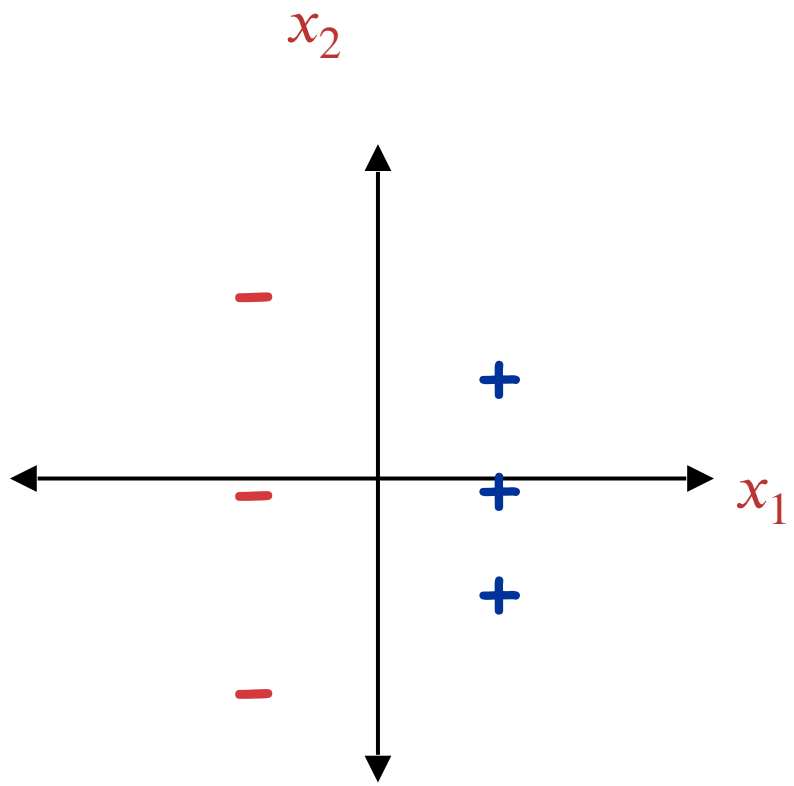
- Initialize the weight vector and intercept to all zeros:

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

- For $t = 1, 2, 3, \ldots$
  - Receive an unlabeled example, $\boldsymbol{x}^{(t)}$
  - Predict its label, $\hat{y} = \text{sign}\left(\boldsymbol{w}^T\boldsymbol{x} + b\right) = \begin{cases} +1 \text{ if } \boldsymbol{w}^T\boldsymbol{x} + b \geq 0 \\ -1 \text{ otherwise} \end{cases}$
  - 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 Learning Algorithm: Example (no Intercept)

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

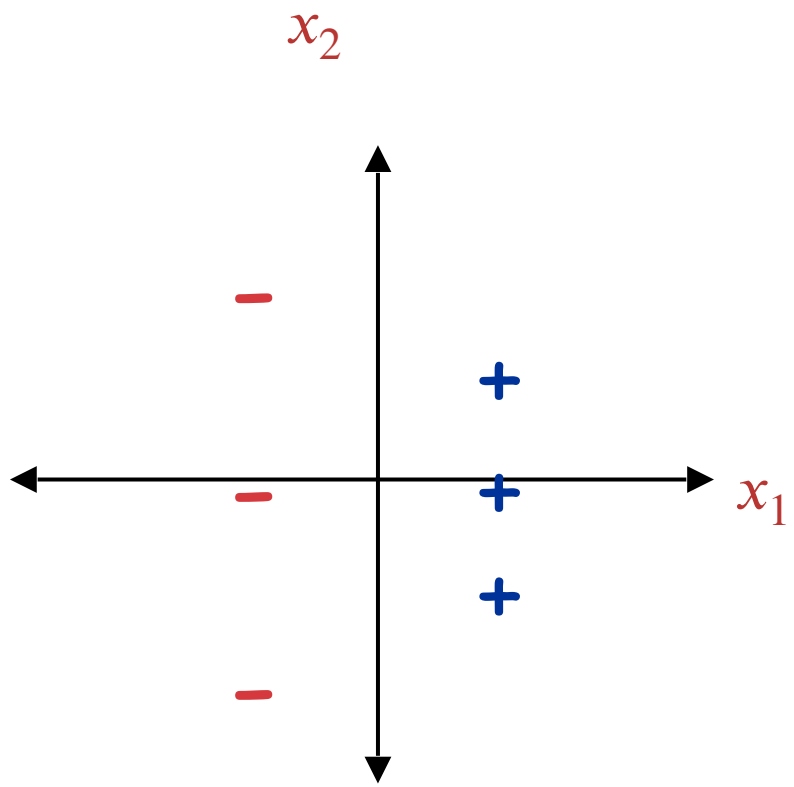


Example courtesy of Nina Balcan

(Online) Perceptron Learning Algorithm: Example (no Intercept)

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

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

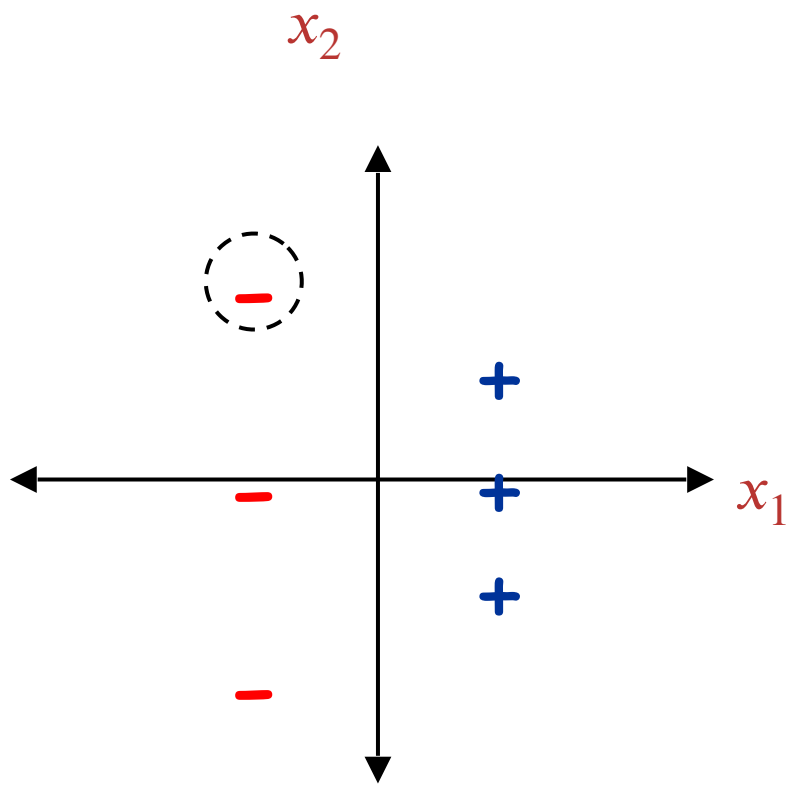


Example courtesy of Nina Balcan

(Online) Perceptron Learning Algorithm: Example (no Intercept)

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

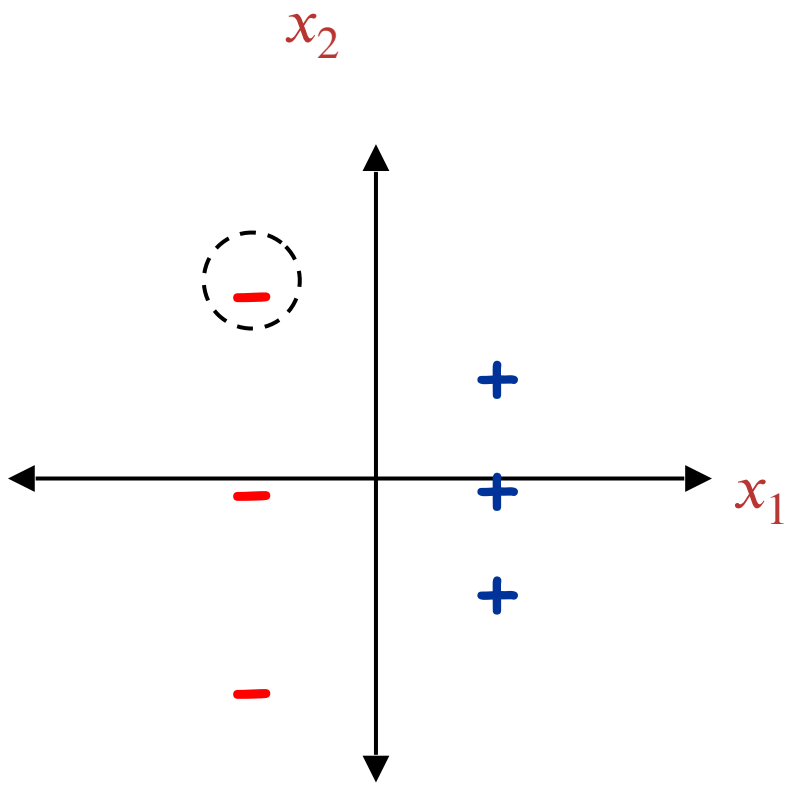
$$\boldsymbol{w} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$



Example courtesy of Nina Balcan

(Online) Perceptron Learning Algorithm: Example (no Intercept)

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



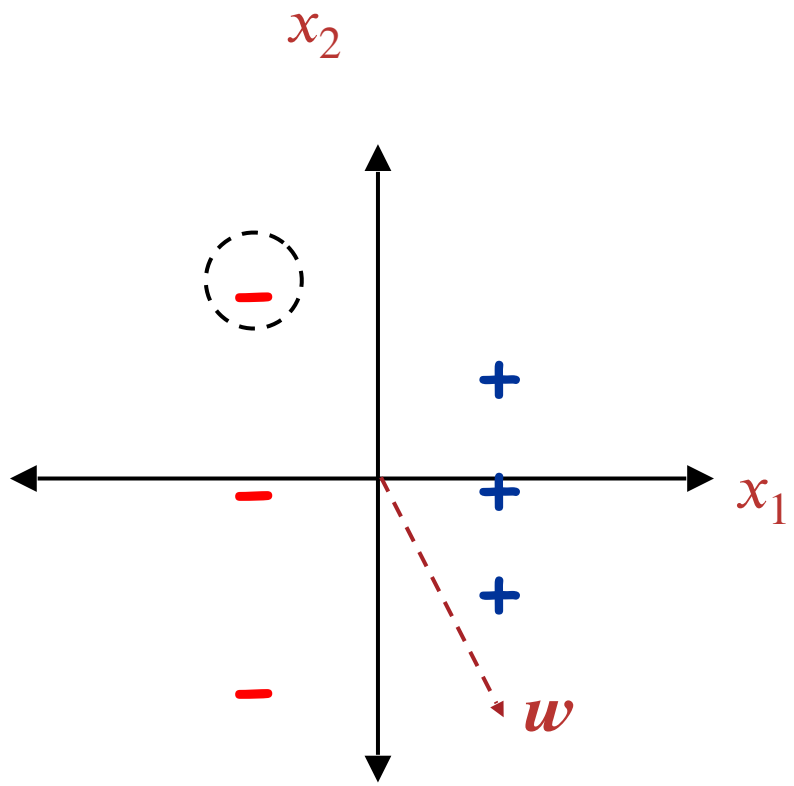
$$\boldsymbol{w} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

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

Example courtesy of Nina Balcan

(Online) Perceptron Learning Algorithm: Example (no Intercept)

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



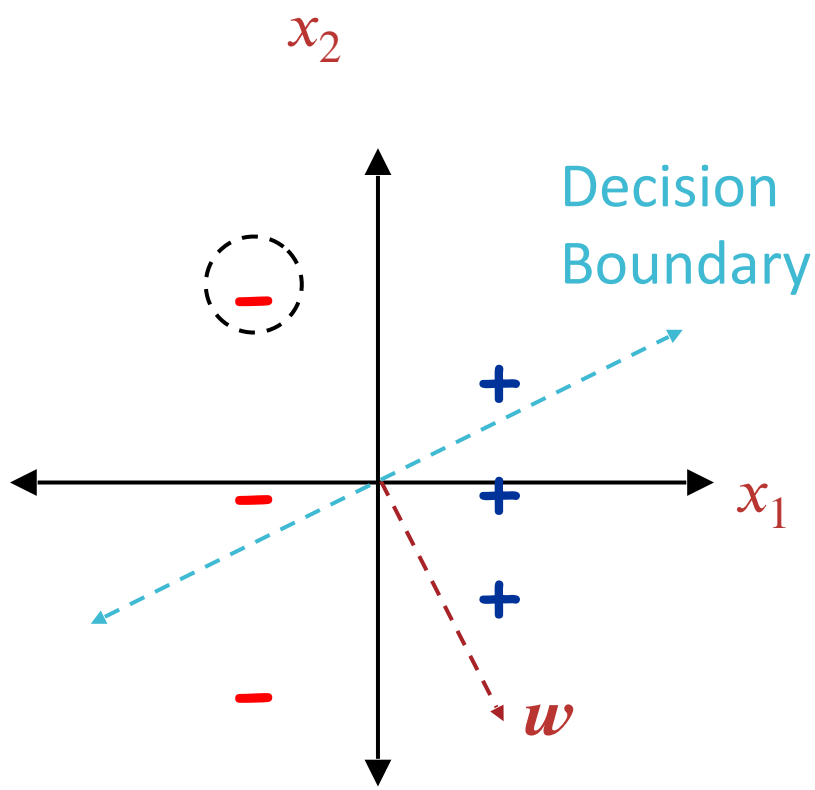
$$\boldsymbol{w} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

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

Example courtesy of Nina Balcan

(Online) Perceptron Learning Algorithm: Example (no Intercept)

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



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

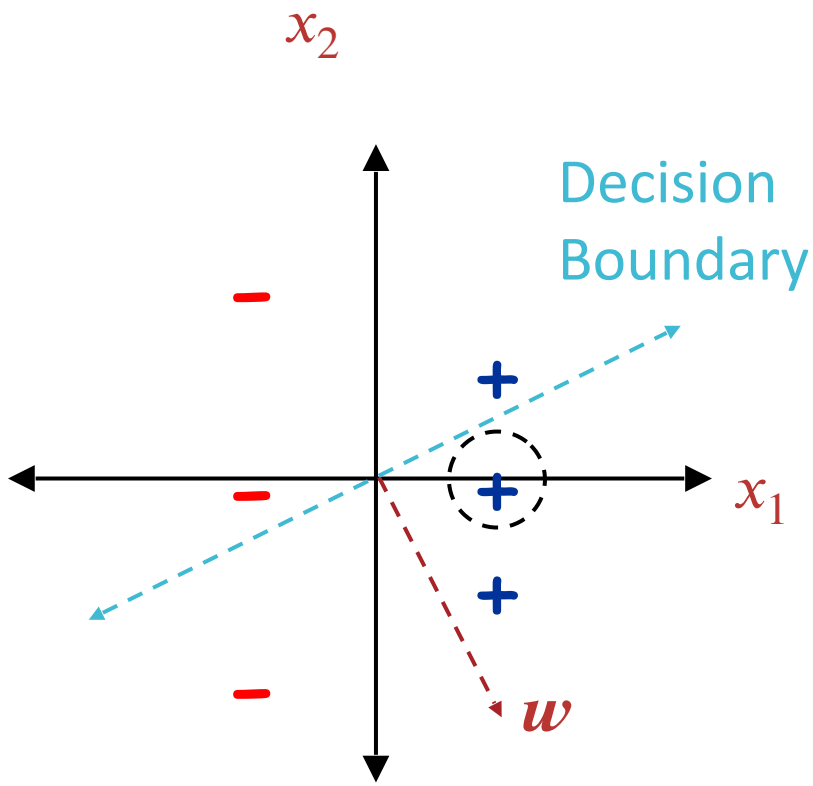
$$\boldsymbol{w} \leftarrow \boldsymbol{w} + y^{(1)}\boldsymbol{x}^{(1)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

Example courtesy of Nina Balcan

(Online) Perceptron Learning Algorithm: Example (no Intercept)

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

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

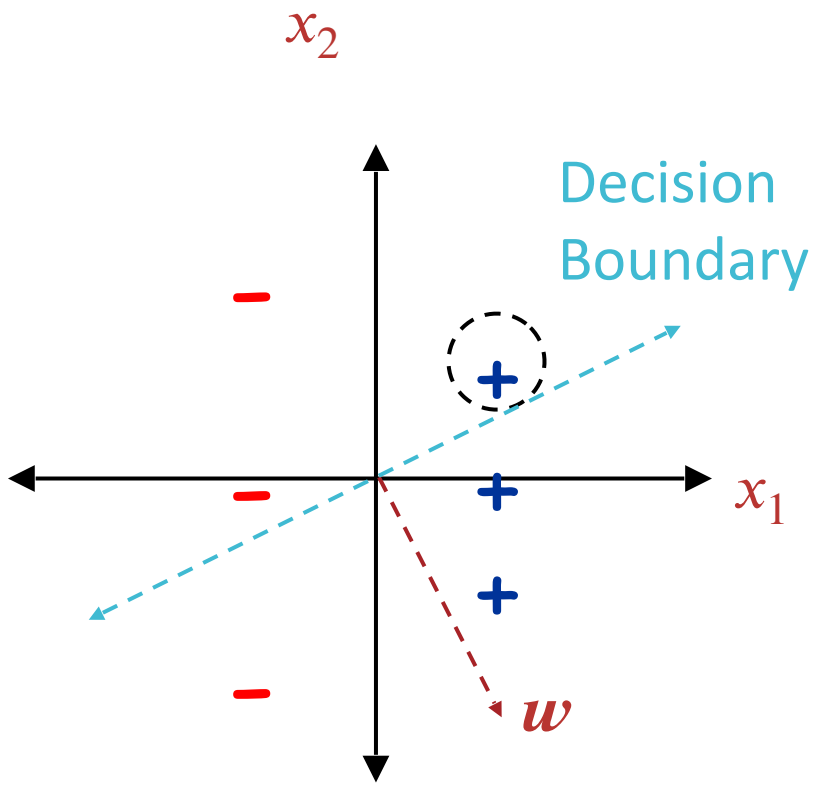


Example courtesy of Nina Balcan

(Online) Perceptron Learning Algorithm: Example (no Intercept)

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

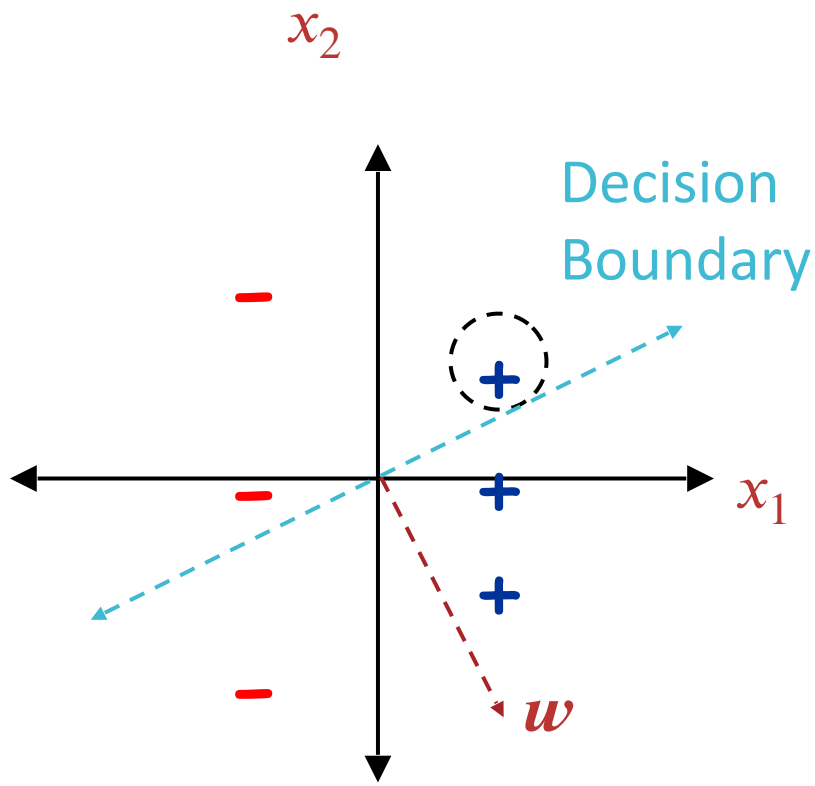
$$\boldsymbol{w} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$



Example courtesy of Nina Balcan

(Online) Perceptron Learning Algorithm: Example (no Intercept)

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



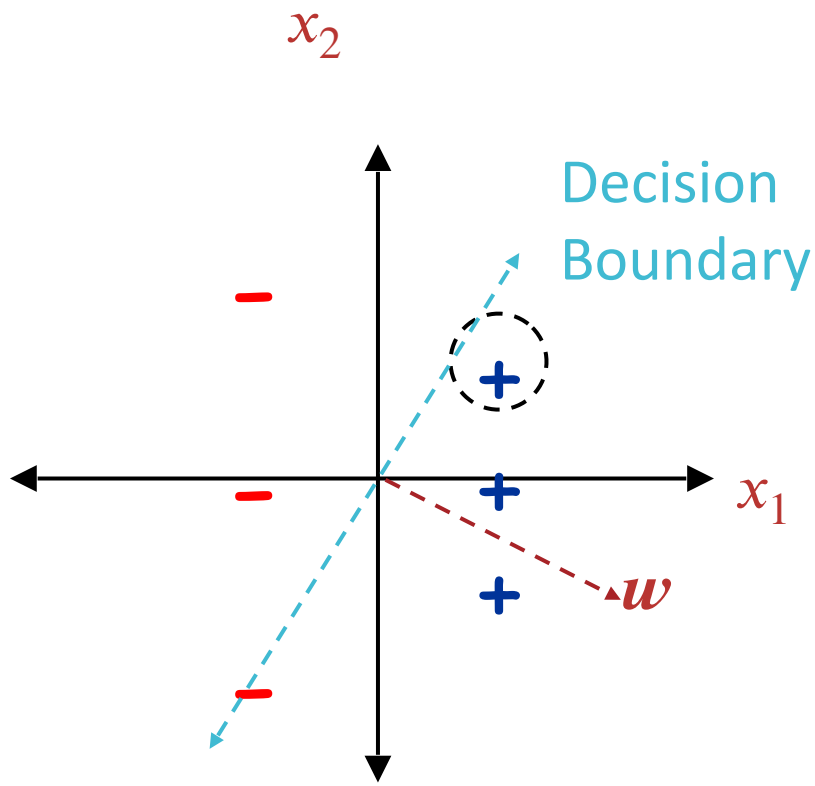
$$\boldsymbol{w} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

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

Example courtesy of Nina Balcan

(Online) Perceptron Learning Algorithm: Example (no Intercept)

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



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

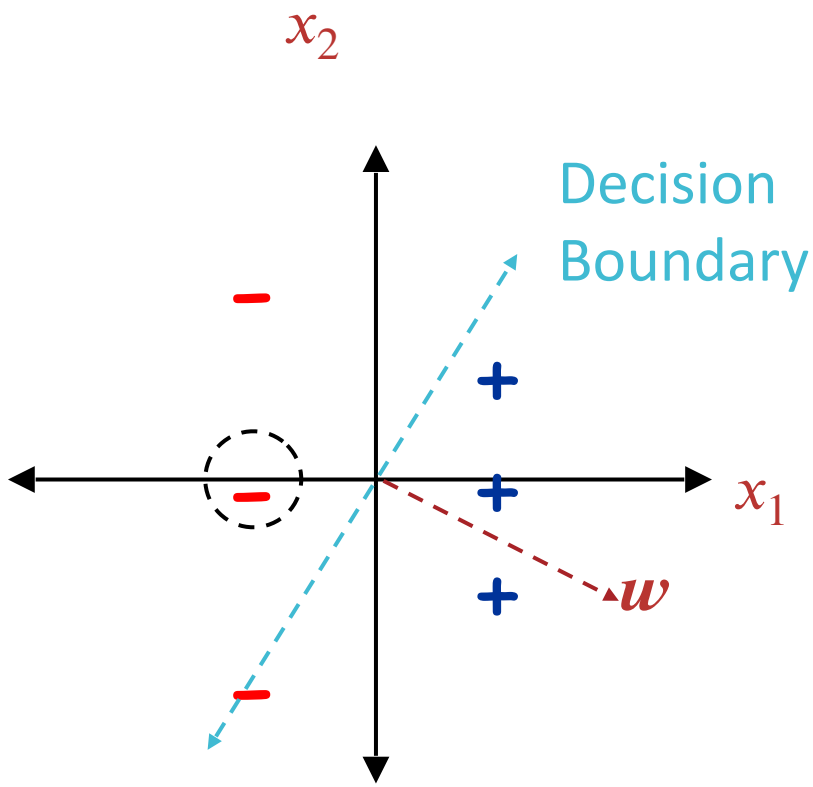
$$\boldsymbol{w} \leftarrow \boldsymbol{w} + y^{(3)}\boldsymbol{x}^{(3)} = \begin{bmatrix} 1 \\ -2 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$$

Example courtesy of Nina Balcan

(Online) Perceptron Learning Algorithm: Example (no Intercept)

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

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

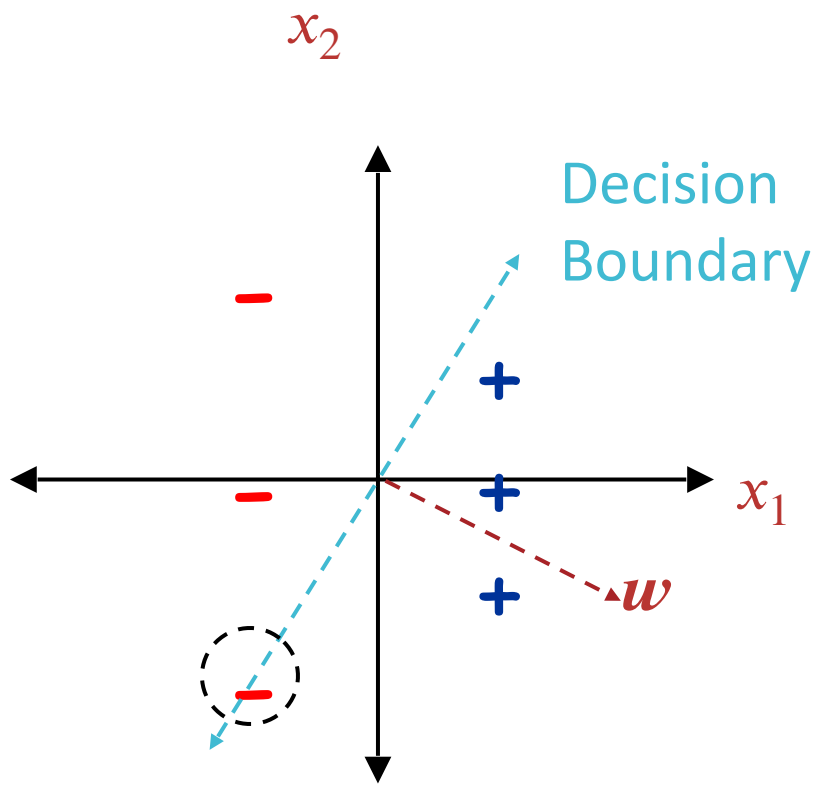


Example courtesy of Nina Balcan

(Online) Perceptron Learning Algorithm: Example (no Intercept)

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

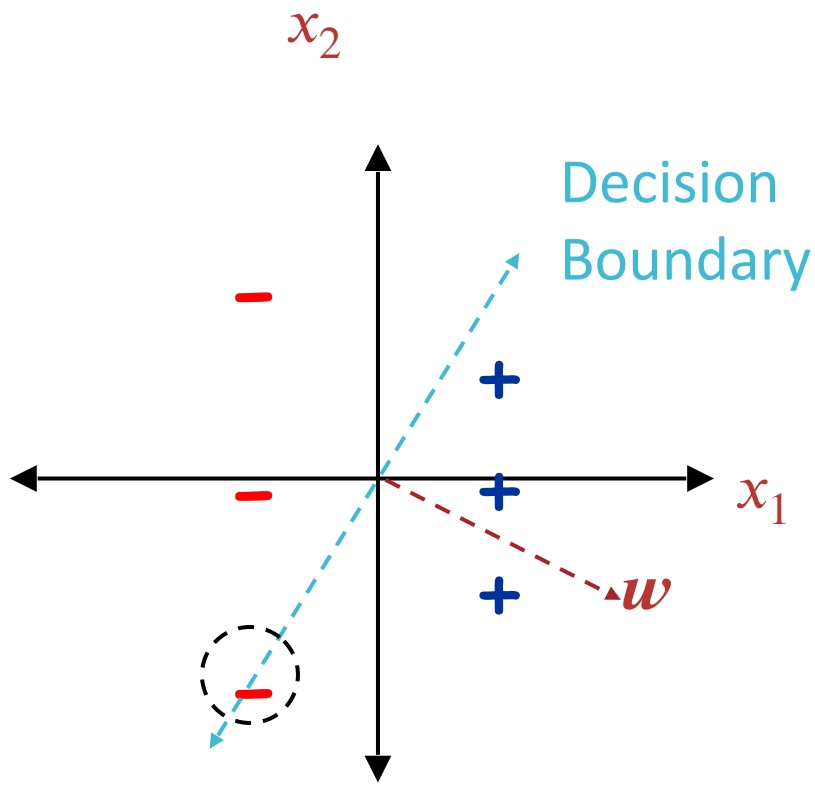
$$\boldsymbol{w} = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$$



Example courtesy of Nina Balcan

(Online) Perceptron Learning Algorithm: Example (no Intercept)

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



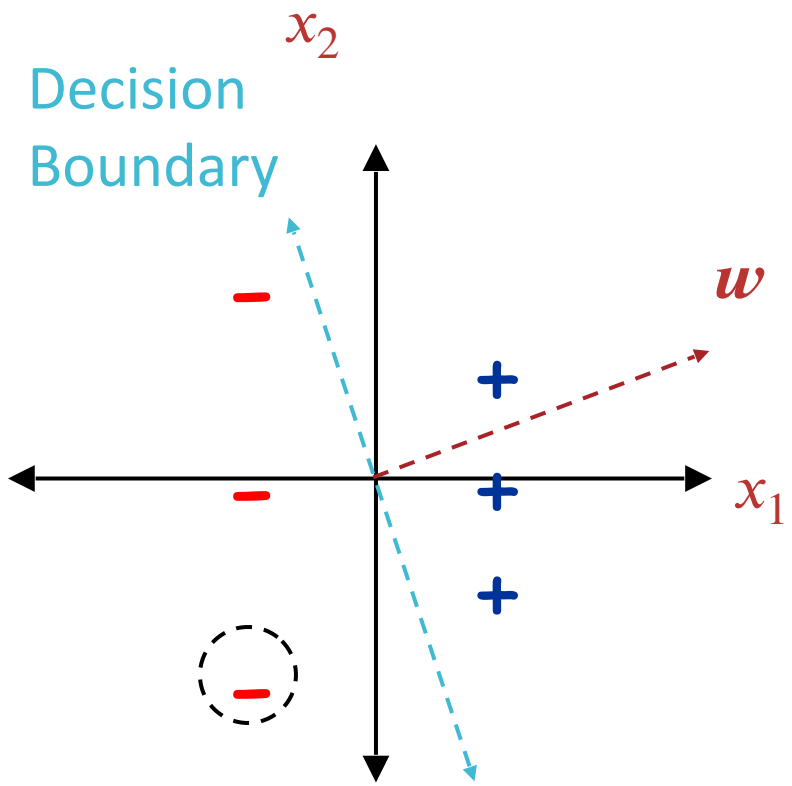
$$\boldsymbol{w} = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$$

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

Example courtesy of Nina Balcan

(Online) Perceptron Learning Algorithm: Example (no Intercept)

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



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

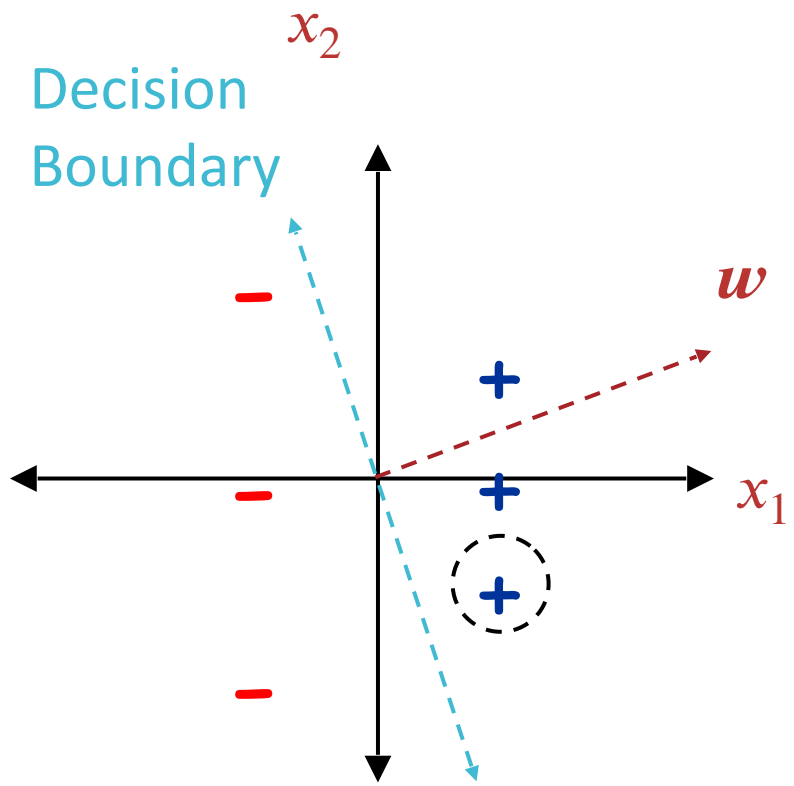
$$\boldsymbol{w} \leftarrow \boldsymbol{w} + y^{(5)}\boldsymbol{x}^{(5)} = \begin{bmatrix} 2 \\ -1 \end{bmatrix} - \begin{bmatrix} -1 \\ -2 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

Example courtesy of Nina Balcan

(Online) Perceptron Learning Algorithm: Example (no Intercept)

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

Thought experiment

- [No bias term for now]
- What if we scaled up every training example?
  - $\boldsymbol{x}^{(i)} \rightarrow 2\boldsymbol{x}^{(i)}$ for all i

nothing!

Thought experiment

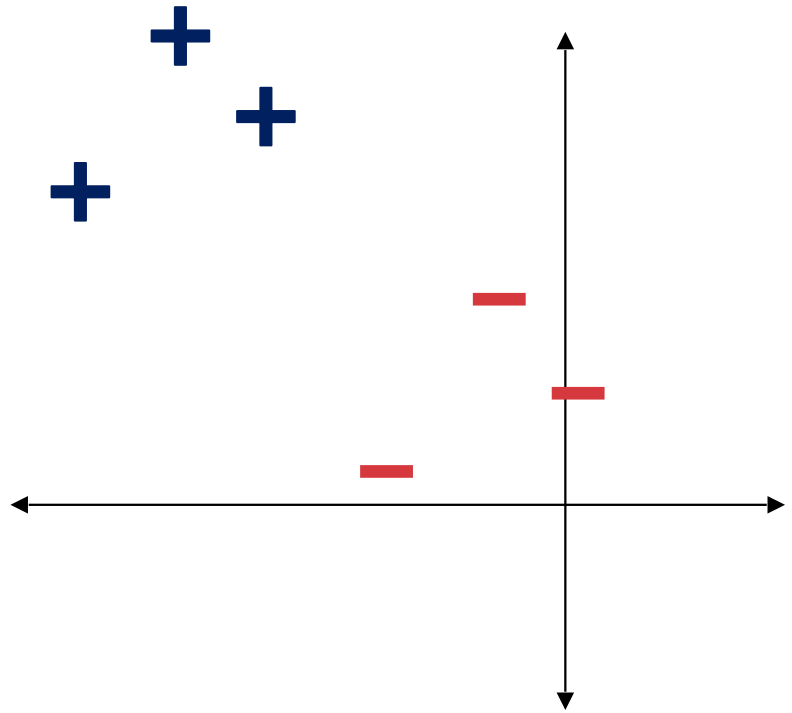
- What if we scaled up ***half*** of the training examples?
  - $\boldsymbol{x}^{(i)} \rightarrow 2\boldsymbol{x}^{(i)}$ for every ***even*** value of i

↳ more important

Scale a feature → classifier depends on it more

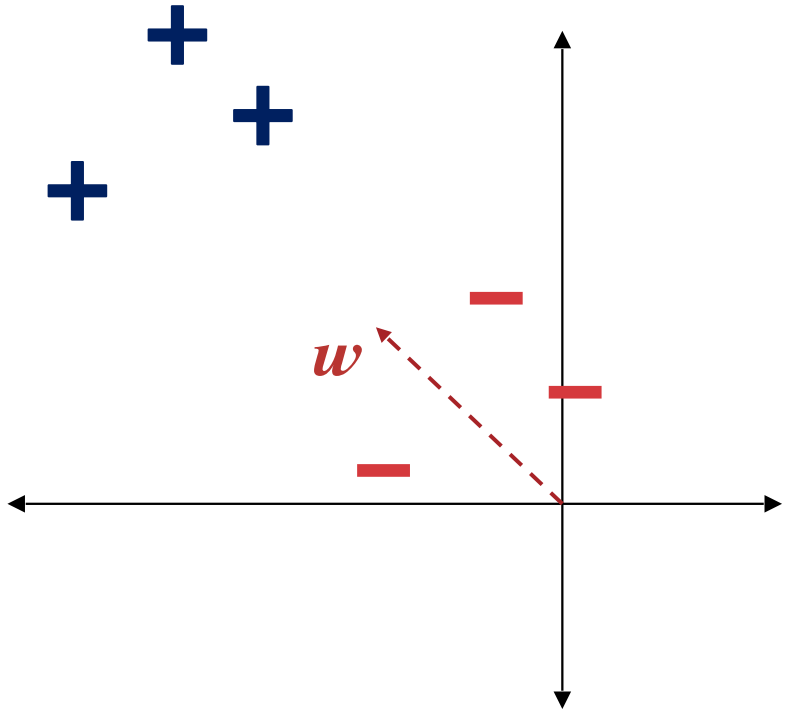
Updating the Intercept

- The intercept shifts the decision boundary off the origin



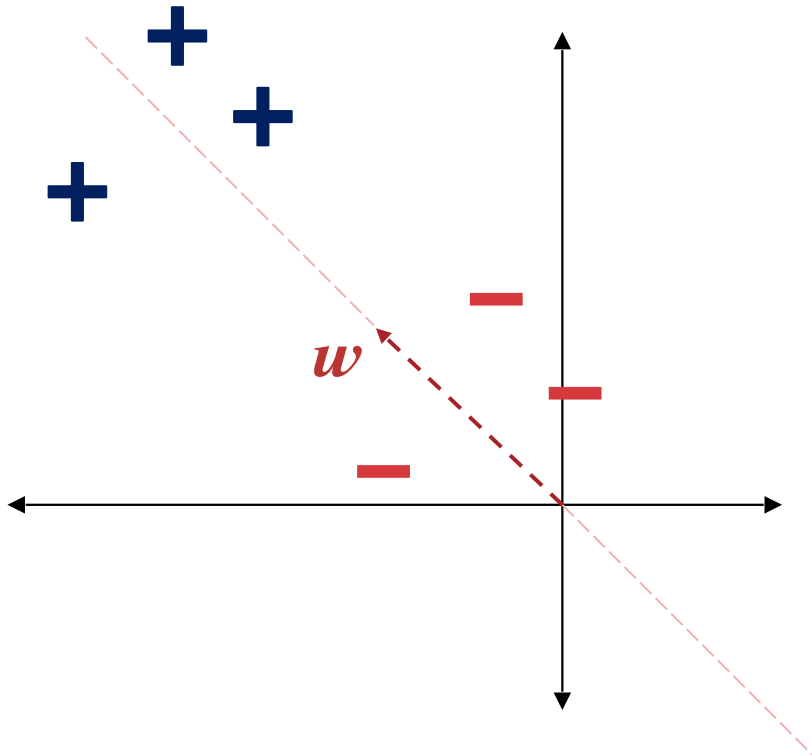
Updating the Intercept

- The intercept shifts the decision boundary off the origin



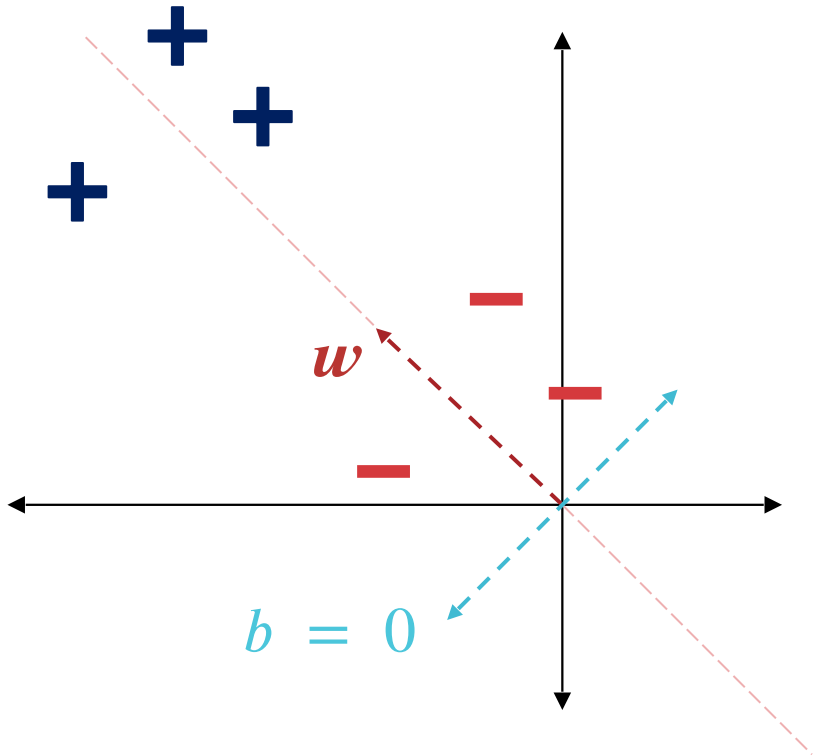
Updating the Intercept

- The intercept shifts the decision boundary off the origin



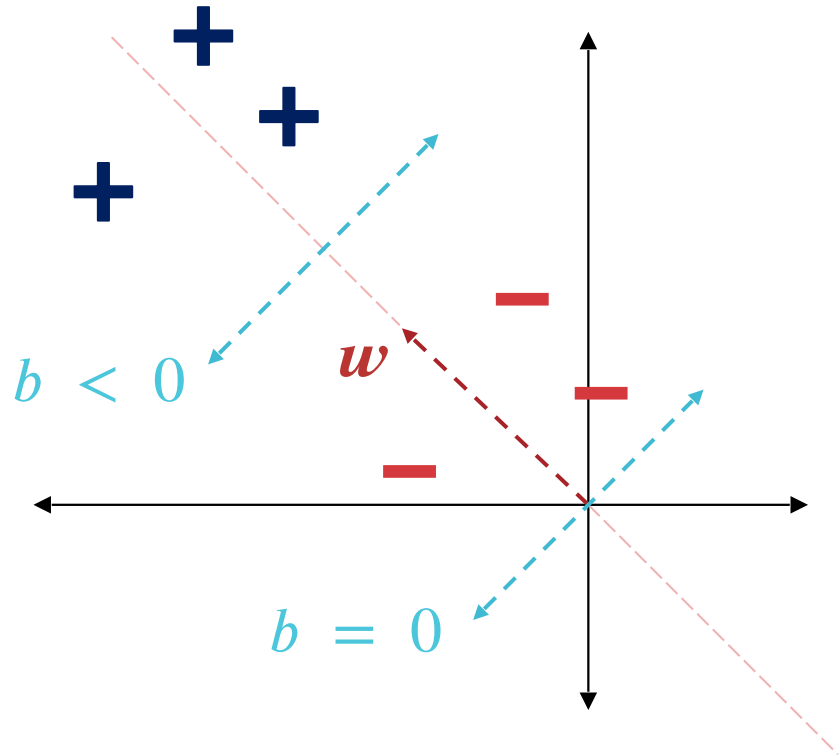
Updating the Intercept

- The intercept shifts the decision boundary off the origin



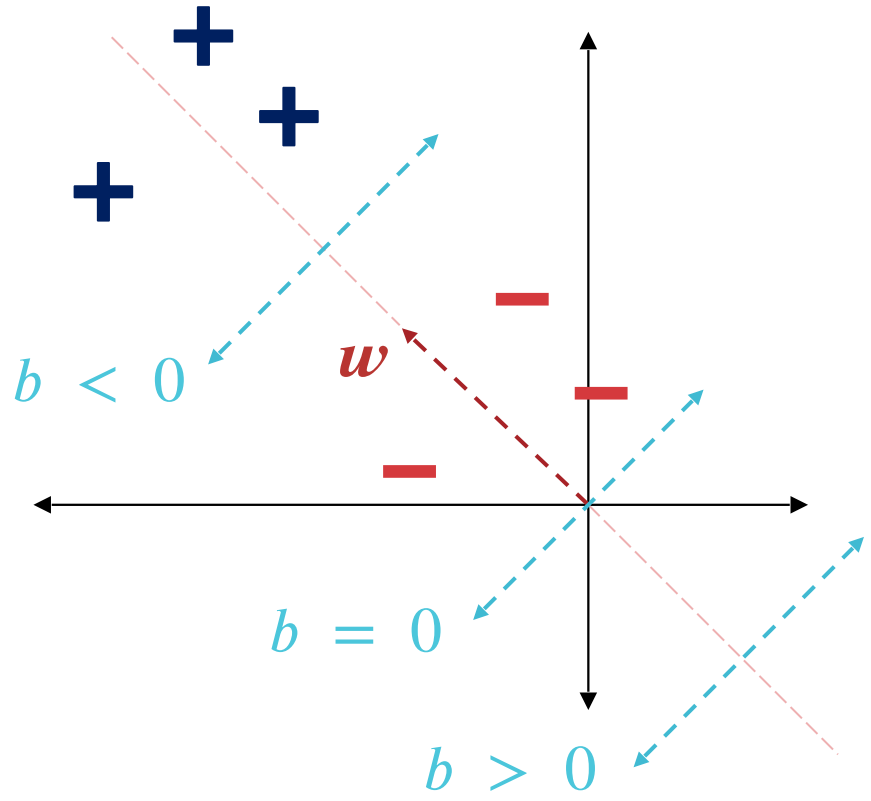
Updating the Intercept

- The intercept shifts the decision boundary off the origin
  - Increasing b shifts the decision boundary towards the negative side



Updating the Intercept

- The intercept shifts the decision boundary off the origin
  - Increasing b shifts the decision boundary towards the negative side
  - Decreasing b shifts the decision boundary towards the positive side



Poll Question 1

poll.mlcourse.org

- **True or False**: Unlike Decision Trees and k-Nearest Neighbors, the Perceptron learning algorithm does not suffer from overfitting because it does not have any hyperparameters that could be over-tuned on a validation dataset.

A: False
C: True

B: toxic

Notational hack #2

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

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

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

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

Notational hack #2

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

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

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

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

… and now if we update $\boldsymbol{\theta} \rightarrow \boldsymbol{\theta} + y\boldsymbol{x}'$, that's the same as $\boldsymbol{w} \rightarrow \boldsymbol{w} + y\boldsymbol{x}$ and $b \rightarrow b + y$

(Online) Perceptron Learning Algorithm

- Initialize the parameters to all zeros:

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

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

(Online) Perceptron Learning 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 ($y^{(t)} \neq \hat{y}$):
    - $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + y^{(t)} \boldsymbol{x}'^{(t)}$

(Online) Perceptron Learning 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

(Online) Perceptron Learning Algorithm: Intuition

- Suppose $(x', y) \in \mathcal{D}$ is a misclassified training example and $y = +1$ (the $y = -1$ case is similar)

$$\theta_{new} = \theta + yx'$$

$$\theta_{new} \cdot x' = (\theta + yx') \cdot x'$$

$$= \underbrace{\theta \cdot x'}_{\text{old prediction}} + \underbrace{yx' \cdot x'}_{\text{added something}}$$

$yx' \cdot x' \to +1 > 0$

(Online) Perceptron Learning Algorithm: Inductive Bias

don't change too fast

linear decision bd.

outlier sensitive – higher scale ex more important

continuous wts represent bd.

higher scale features more important

(Batch) Perceptron Learning Algorithm

- Input: $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(1)}, y^{(1)}\right), \left(\boldsymbol{x}^{(2)}, y^{(2)}\right), \ldots, \left(\boldsymbol{x}^{(N)}, y^{(N)}\right)\right\}$
- Initialize the parameters to all zeros:

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

- While NOT CONVERGED
  - For $t \in \{1, \ldots, N\}$ *[optionally: permute each epoch]*
    - Predict the label of $\boldsymbol{x}'^{(t)}$, $\hat{y} = \text{sign}\left(\boldsymbol{\theta}^T \boldsymbol{x}'^{(t)}\right)$
    - Observe its true label, $y^{(t)}$
    - If we misclassified $\boldsymbol{x}'^{(t)}$ ($y^{(t)} \neq \hat{y}$):
      - $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + y^{(t)} \boldsymbol{x}'^{(t)}$

(Batch) Perceptron Learning Algorithm

- Input: $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(1)}, y^{(1)}\right), \left(\boldsymbol{x}^{(2)}, y^{(2)}\right), \ldots, \left(\boldsymbol{x}^{(N)}, y^{(N)}\right)\right\}$
- Initialize the parameters to all zeros:

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

no mistakes

- While NOT CONVERGED ← what does this mean?
  - For $t \in \{1, \ldots, N\}$ *[optionally: permute each epoch]*
    - Predict the label of $\boldsymbol{x}'^{(t)}$, $\hat{y} = \text{sign}\left(\boldsymbol{\theta}^T \boldsymbol{x}'^{(t)}\right)$
    - Observe its true label, $y^{(t)}$
    - If we misclassified $\boldsymbol{x}'^{(t)}$ ($y^{(t)} \neq \hat{y}$):
      - $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + y^{(t)} \boldsymbol{x}'^{(t)}$

Linearly separable

- A dataset $\mathscr{D}$ is *linearly separable* if $\exists$ a linear decision boundary $\boldsymbol{\theta} = (\boldsymbol{w}, b)$ that perfectly classifies all examples in $\mathscr{D}$

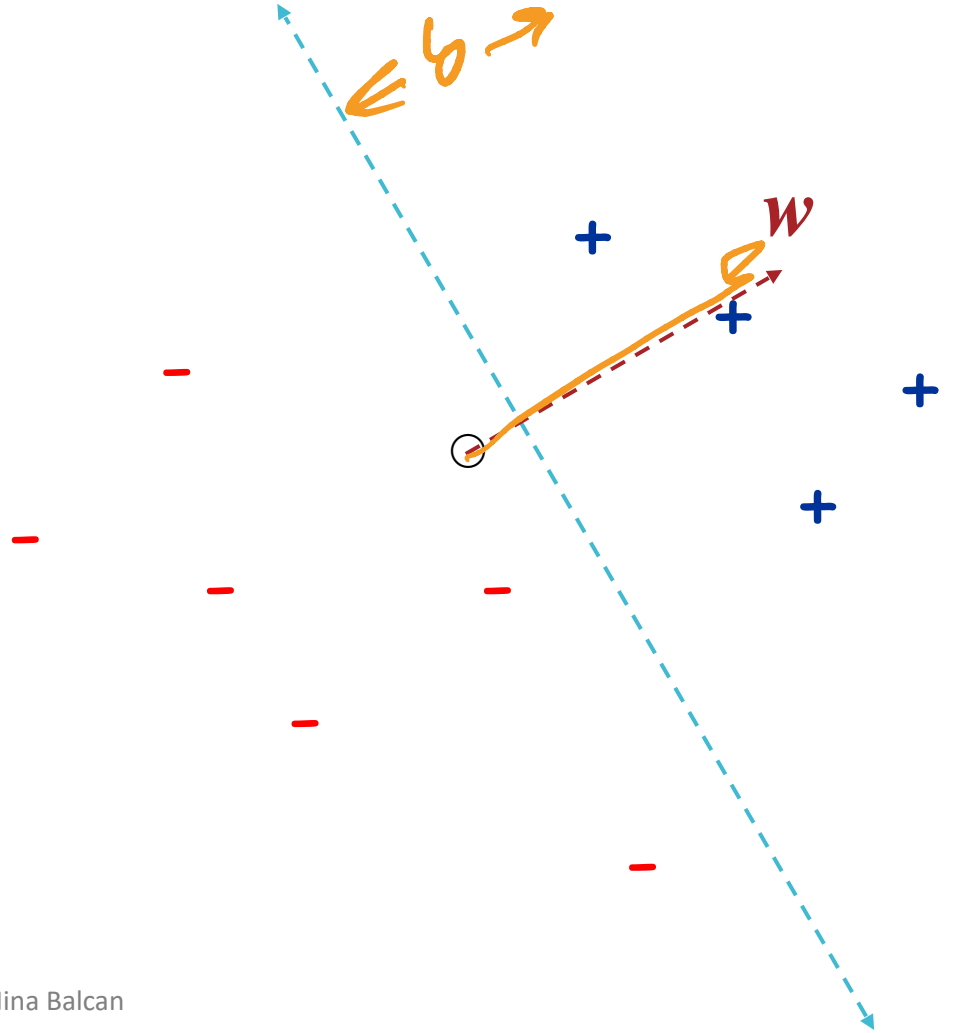
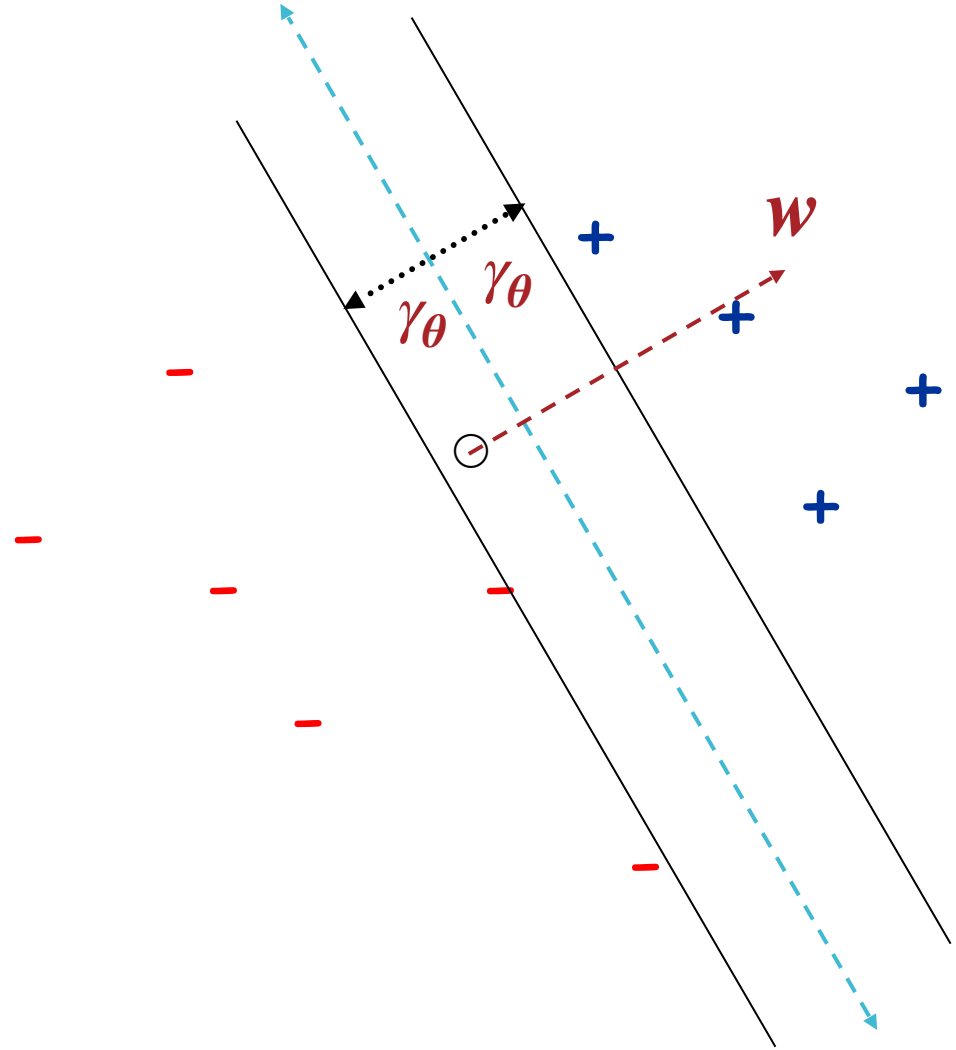


Figure based on one by Nina Balcan

Margin

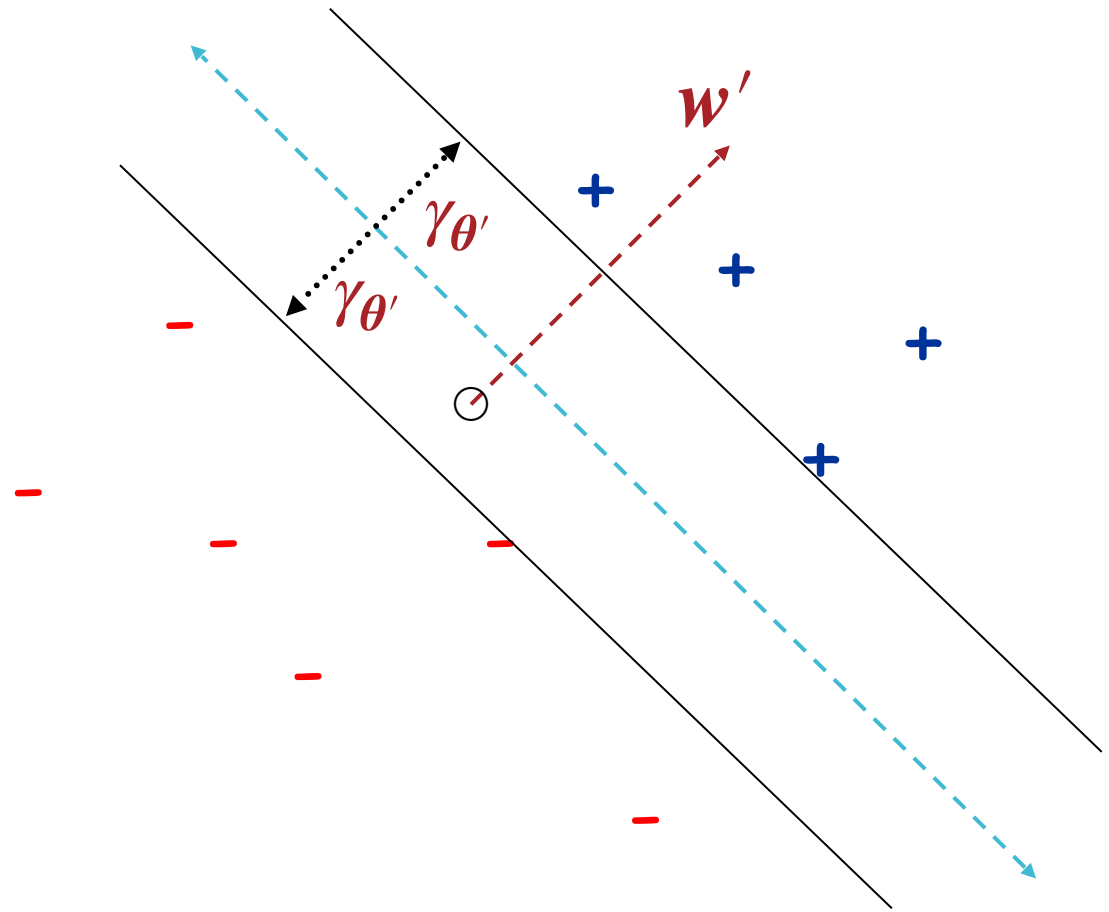
- The margin, γ_{θ}, of any separator $\boldsymbol{\theta}$ is the distance of the closest example in $\mathcal{D}$ to that separator



expand a slab around the separator until it touches an example on one side or the other

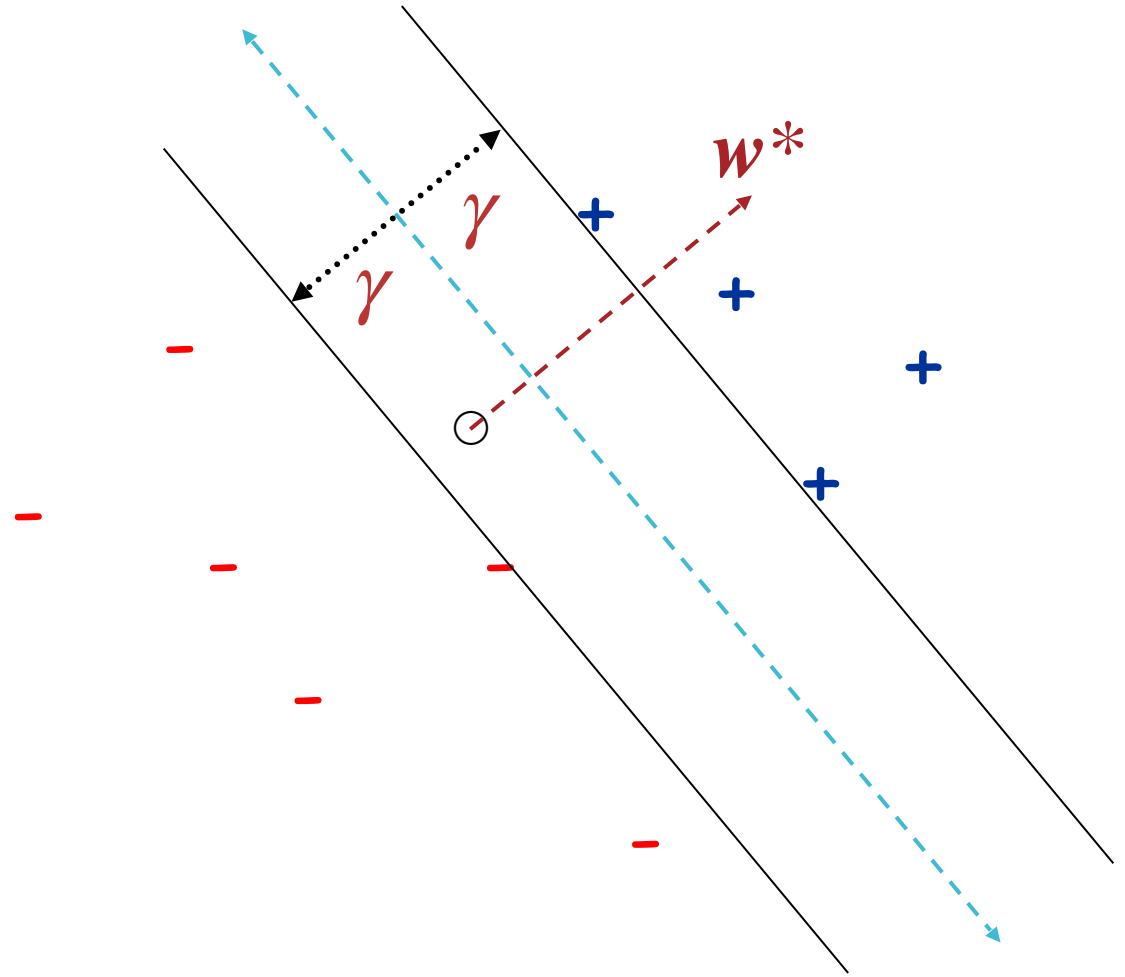
Margin

- Different separators can have different margins



Margin

- The margin, γ, of the entire dataset $\mathcal{D}$ is the largest margin of any linear separator



Perceptron Mistake Bound

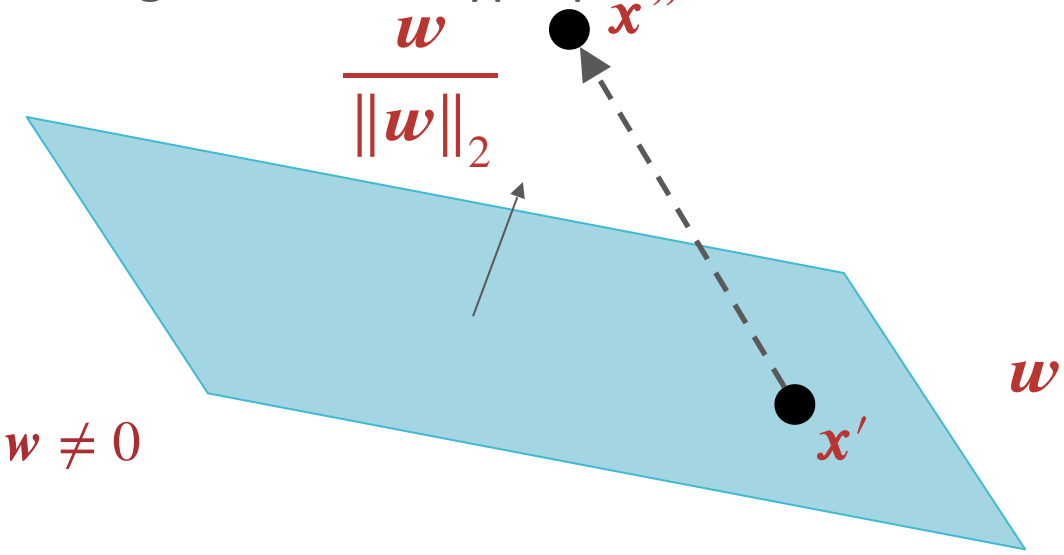
- Theorem: if the examples seen by the Perceptron Learning Algorithm (either online or batch)
  1. lie in a ball of radius R (centered around the origin)
  2. have a margin of $\gamma >$ 

then the algorithm makes at most $\left(R/\gamma\right)^2$ mistakes total!

- Key Takeaway: if the training dataset is linearly separable, the batch Perceptron Learning Algorithm will converge (i.e., stop making mistakes on the training dataset or achieve 0 training error) in a finite number of steps!

Computing the Margin

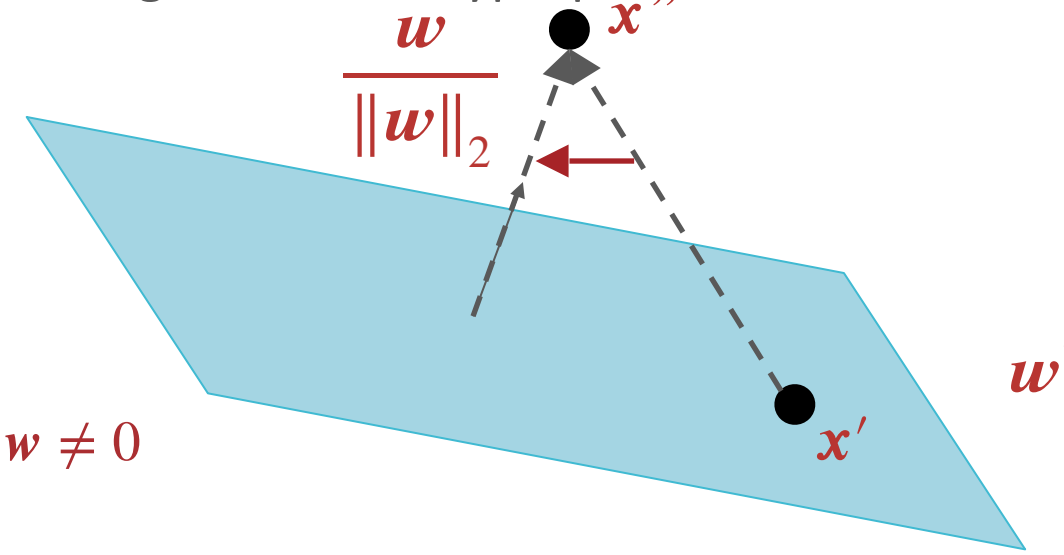
- Let $\boldsymbol{x}'$ be an arbitrary point on the hyperplane $\boldsymbol{w}^T\boldsymbol{x} + b = 0$ and let $\boldsymbol{x}''$ be an arbitrary point
- The distance between $\boldsymbol{x}''$ and $\boldsymbol{w}^T\boldsymbol{x} + b = 0$ is equal to the magnitude of the projection of $\boldsymbol{x}'' - \boldsymbol{x}'$ onto $\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|_2}$, the unit vector orthogonal to the hyperplane



assume $\boldsymbol{w} \neq 0$

Computing the Margin

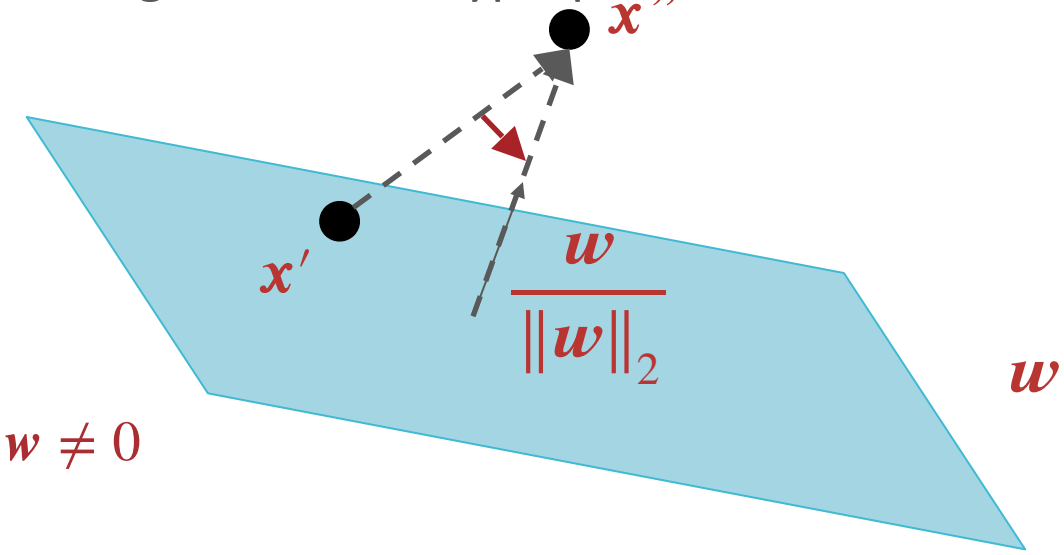
- Let $\boldsymbol{x}'$ be an arbitrary point on the hyperplane $\boldsymbol{w}^T\boldsymbol{x} + b = 0$ and let $\boldsymbol{x}''$ be an arbitrary point
- The distance between $\boldsymbol{x}''$ and $\boldsymbol{w}^T\boldsymbol{x} + b = 0$ is equal to the magnitude of the projection of $\boldsymbol{x}'' - \boldsymbol{x}'$ onto $\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|_2}$, the unit vector orthogonal to the hyperplane



assume $\boldsymbol{w} \neq 0$

Computing the Margin

- Let $\boldsymbol{x}'$ be an arbitrary point on the hyperplane $\boldsymbol{w}^T\boldsymbol{x} + b = 0$ and let $\boldsymbol{x}''$ be an arbitrary point
- The distance between $\boldsymbol{x}''$ and $\boldsymbol{w}^T\boldsymbol{x} + b = 0$ is equal to the magnitude of the projection of $\boldsymbol{x}'' - \boldsymbol{x}'$ onto $\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|_2}$, the unit vector orthogonal to the hyperplane



assume $\boldsymbol{w} \neq 0$

Computing the Margin

- Let $\boldsymbol{x}'$ be an arbitrary point on the hyperplane and let $\boldsymbol{x}''$ be an arbitrary point
- The distance between $\boldsymbol{x}''$ and $\boldsymbol{w}^T\boldsymbol{x} + b = 0$ is equal to the magnitude of the projection of $\boldsymbol{x}'' - \boldsymbol{x}'$ onto $\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|_2}$, the unit vector orthogonal to the hyperplane

$$\left| \frac{\boldsymbol{w}^T(\boldsymbol{x}'' - \boldsymbol{x}')}{\|\boldsymbol{w}\|_2} \right| = \frac{\left|\boldsymbol{w}^T\boldsymbol{x}'' - \boldsymbol{w}^T\boldsymbol{x}'\right|}{\|\boldsymbol{w}\|_2} = \frac{\left|\boldsymbol{w}^T\boldsymbol{x}'' + b\right|}{\|\boldsymbol{w}\|_2}$$

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 (shifting points after projection onto weight vector)