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

Hoda Heidari, Henry Chai & Matt Gormley

2/5/24

Front Matter

- Announcements:
  - HW2 released 1/24, due 2/5 (today!) at 11:59 PM
  - HW3 released on 2/5 (today!), due 2/12 at 11:59 PM
    - HW3 is a written-only homework
    - **You may only use at most 2 late days on HW3**

Q & A:

After we do model selection using a validation dataset, should we train a final model using *both* the training and the validation datasets?

- Yes, absolutely! So really the sketch from last lecture should look something like:
  1. Split $\mathcal{D}$ into $\mathcal{D}_{train} \cup \mathcal{D}_{val} \cup \mathcal{D}_{test}$
  2. Learn classifiers using $\mathcal{D}_{train}$
  3. Evaluate models using $\mathcal{D}_{val}$ and choose the one with lowest *validation* error:
  4. **Learn a new classifier from the best model using $\mathcal{D}_{train} \cup \mathcal{D}_{val}$**
  5. Optionally, use $\mathcal{D}_{test}$ to estimate the true error

Q & A:

Can we use kNNs 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:

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

- See HW3 for an example of this

Hyperparameter Optimization

- Given $\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{val} \cup \mathcal{D}_{test}$, suppose we have multiple candidate hyperparameter settings:

$$\theta_1, \theta_2, \dots, \theta_M$$

- Learn a classifier for each setting using only $\mathcal{D}_{train}$:

$$h_1, h_2, \dots, h_M$$

- Evaluate each one using $\mathcal{D}_{val}$ and choose the one with lowest *validation* error:

$$\hat{m} = \underset{m \in \{1,\dots,M\}}{\operatorname{argmin}} \; err(h_m, \mathcal{D}_{val})$$

- Now $err(h_{\hat{m}}, \mathcal{D}_{test})$ is a good estimate of $err(h_{\hat{m}})$!

How to pick hyperparameter settings to try?

- Given $\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{val} \cup \mathcal{D}_{test}$, suppose we have multiple candidate hyperparameter settings:

$$\theta_1, \theta_2, \ldots, \theta_M$$

- Learn a classifier for each setting using only $\mathcal{D}_{train}$:

$$h_1, h_2, \ldots, h_M$$

- Evaluate each one using $\mathcal{D}_{val}$ and choose the one with lowest *validation* error:

$$\hat{m} = \underset{m \in \{1, \ldots, M\}}{\operatorname{argmin}} \; err(h_m, \mathcal{D}_{val})$$

- Now $err(h_{\hat{m}}, \mathcal{D}_{test})$ is a good estimate of $err(h_{\hat{m}})$!

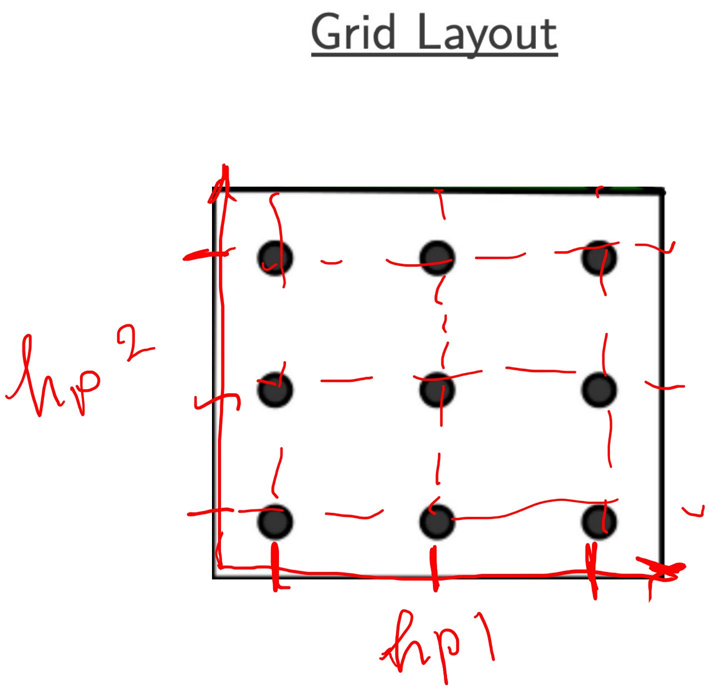
General Methods for Hyperparameter Optimization

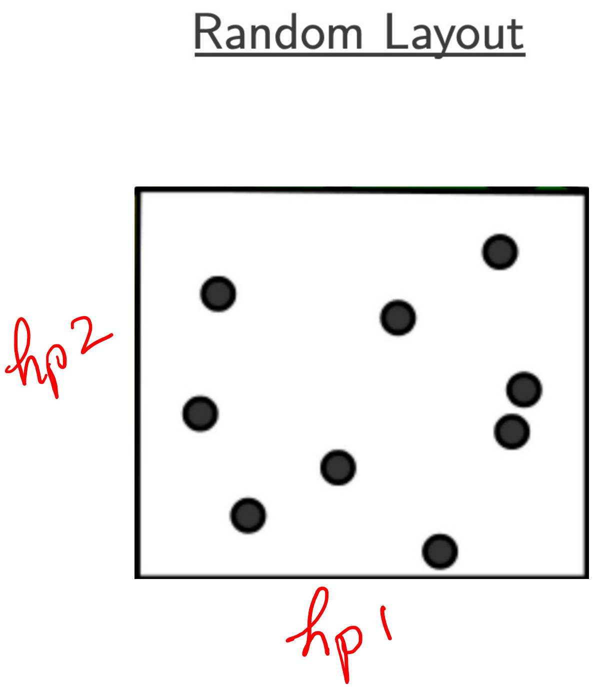
- Idea: set the hyperparameters to optimize some performance metric of the model
- Issue: if we have many hyperparameters that can all take on lots of different values, we might not be able to test all possible combinations
- Commonly used methods:
  - Grid search
  - Random search
  - Bayesian optimization (used by Google DeepMind to optimize the hyperparameters of AlphaGo: https://arxiv.org/pdf/1812.06855v1.pdf)
  - Evolutionary algorithms
  - Graduate-student descent

General Methods for Hyperparameter Optimization

- Idea: set the hyperparameters to optimize some performance metric of the model
- Issue: if we have many hyperparameters that can all take on lots of different values, we might not be able to test all possible combinations
- Commonly used methods:
  - **Grid search**
  - **Random search**
  - Bayesian optimization (used by Google DeepMind to optimize the hyperparameters of AlphaGo: https://arxiv.org/pdf/1812.06855v1.pdf)
  - Evolutionary algorithms
  - Graduate-student descent

Grid Search vs. Random Search (Bergstra and Bengio, 2012)

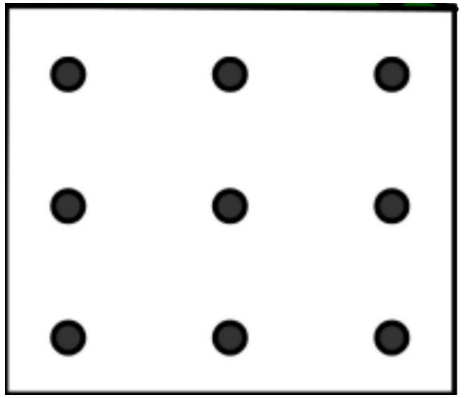




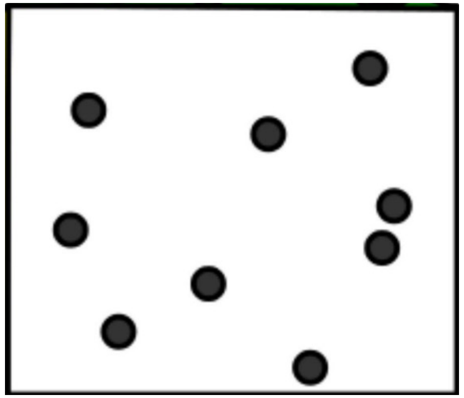
Source: https://jmlr.csail.mit.edu/papers/volume13/bergstra12a/bergstra12a.pdf

Poll Question 1:
Which hyperparameter optimization method do you think will perform better?

Grid Layout



Random Layout



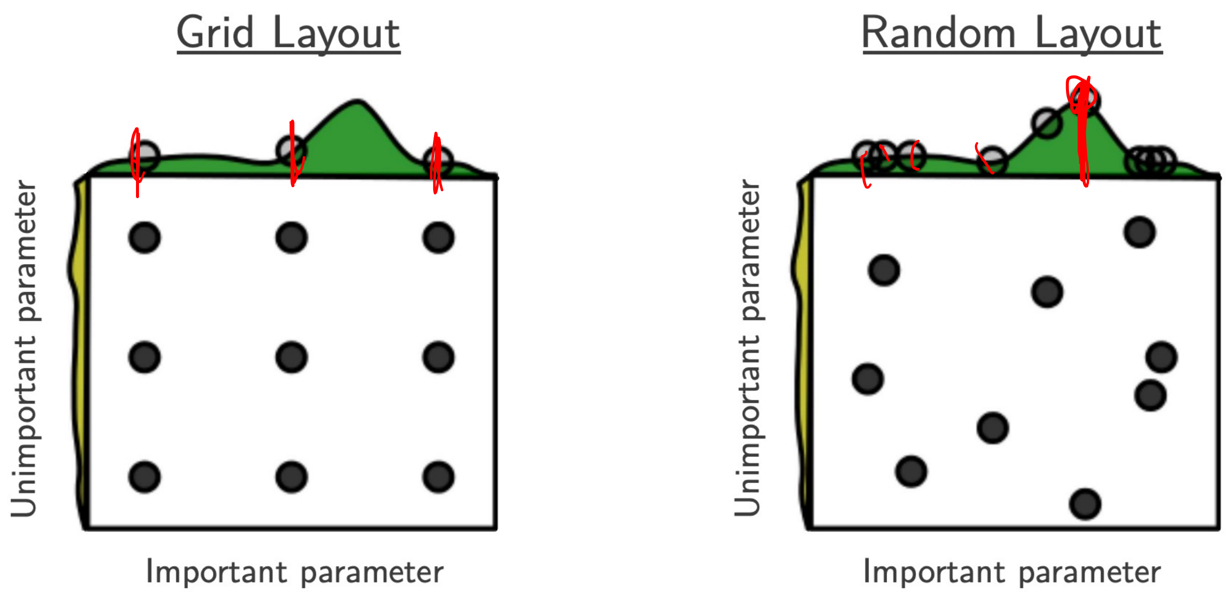
A. Graduate student descent **(TOXIC)**

B. Grid search

C. Random search

Source: https://jmlr.csail.mit.edu/papers/volume13/bergstra12a/bergstra12a.pdf

Grid Search vs. Random Search (Bergstra and Bengio, 2012)



Grid and random search of nine trials for optimizing a function $f(x, y) = g(x) + h(y) \approx g(x)$ with *low effective dimensionality*. Above each square $g(x)$ is shown in green, and left of each square $h(y)$ is shown in yellow. With grid search, nine trials only test $g(x)$ in three distinct places. With random search, all nine trials explore distinct values of g. This failure of grid search is the rule rather than the exception in high dimensional hyper-parameter optimization.

Source: https://jmlr.csail.mit.edu/papers/volume13/bergstra12a/bergstra12a.pdf

Model Selection Learning Objectives

You should be able to…

- Plan an experiment that uses training, validation, and test datasets to predict the performance of a classifier on unseen data (without cheating)
- Explain the difference between (1) training error, (2) validation error, (3) cross-validation error, (4) test error, and (5) true error
- For a given learning technique, identify the model, learning algorithm, parameters, and hyperparamters
- Select an appropriate algorithm for optimizing (aka. learning) hyperparameters

Recall: Fisher Iris Dataset

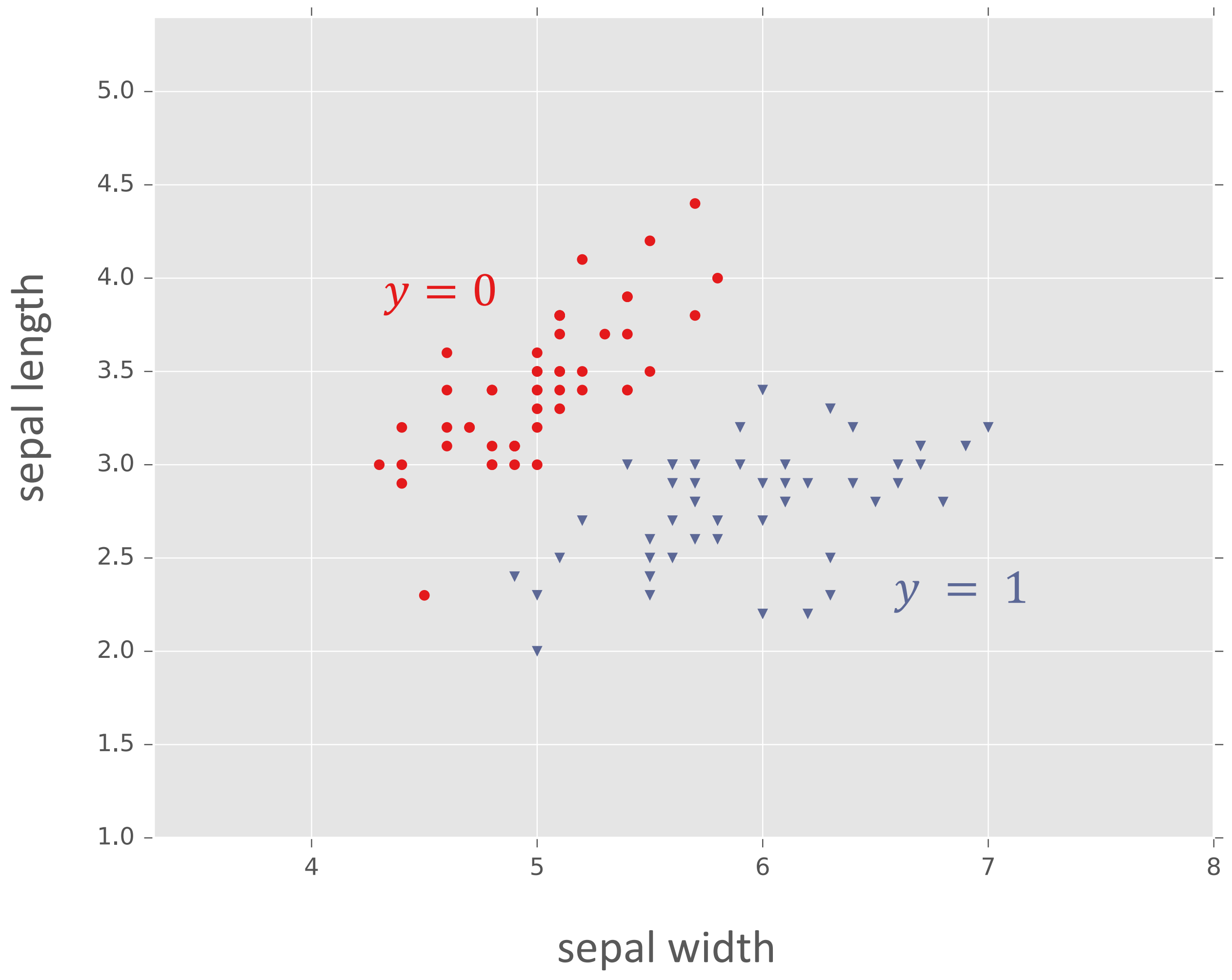


Figure courtesy of Matt Gormley

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

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

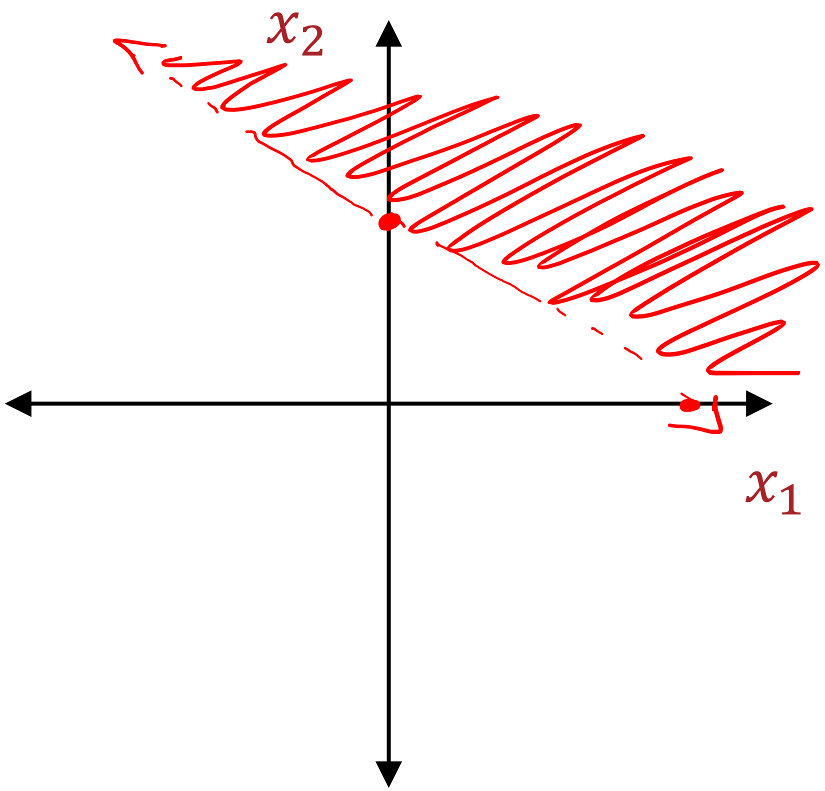
- The $L2$-norm of $\boldsymbol{a} = \|\boldsymbol{a}\|_2 = \sqrt{\boldsymbol{a}^T\boldsymbol{a}}$
- Two vectors are *orthogonal* iff

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

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



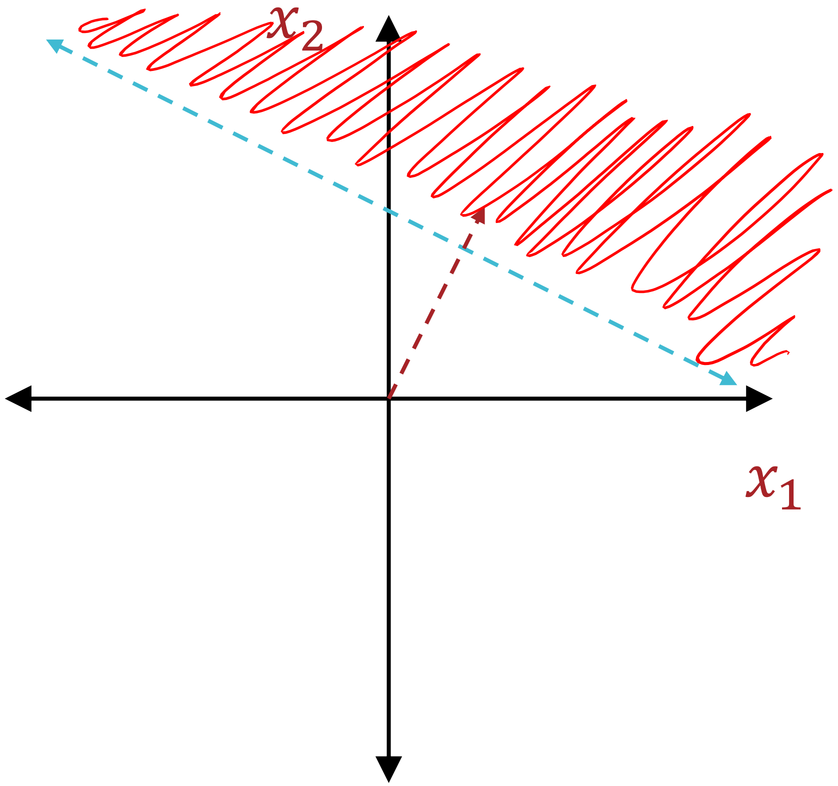
Geometry Warm-up

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 $b = -4$.

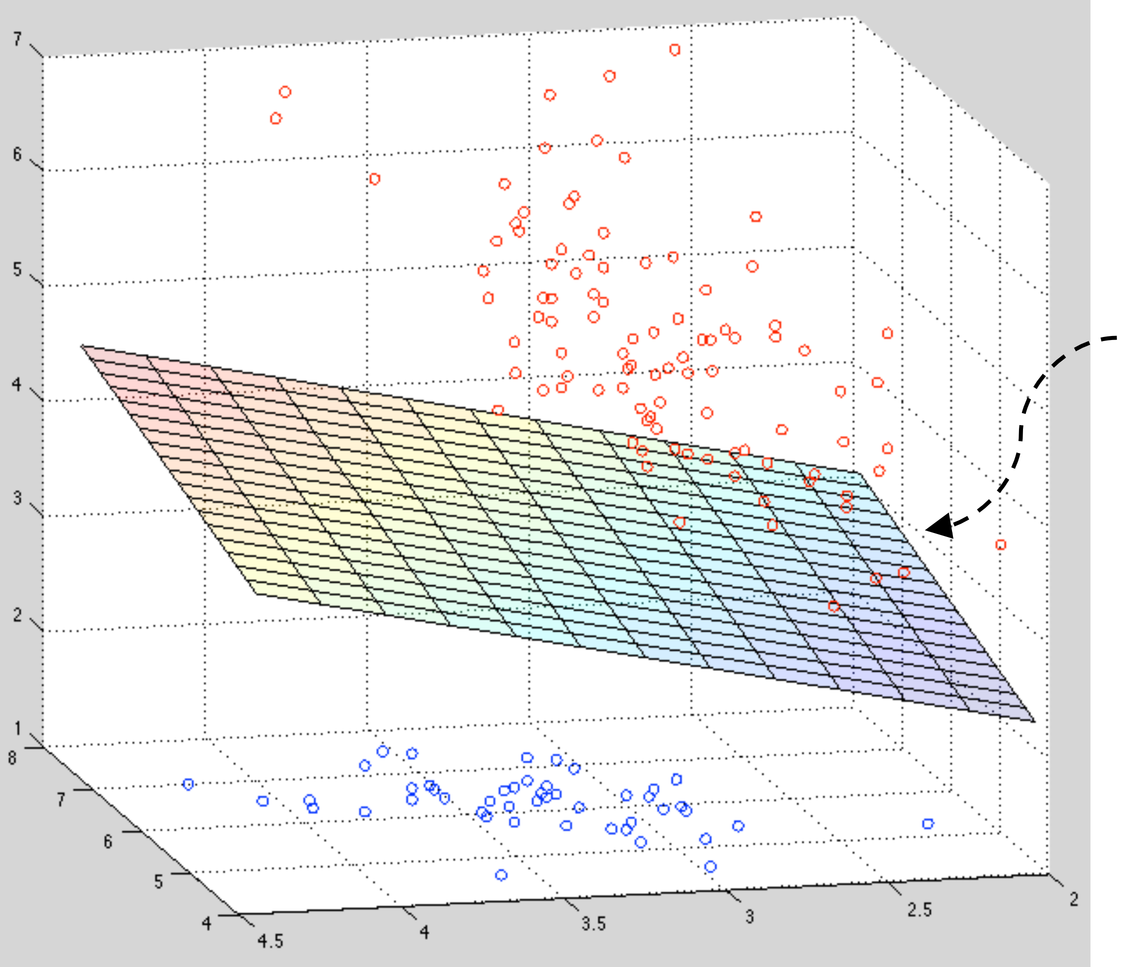
2. Then draw the vector $w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$



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*:
  - $\mathcal{S}_+ = \{\boldsymbol{x} : \boldsymbol{w}^T \boldsymbol{x} + b > 0\}$ or all $\boldsymbol{x}$ s.t. $\boldsymbol{w}^T \boldsymbol{x} + b$ is positive
  - $\mathcal{S}_- = \{\boldsymbol{x} : \boldsymbol{w}^T \boldsymbol{x} + b < 0\}$ or all $\boldsymbol{x}$ s.t. $\boldsymbol{w}^T \boldsymbol{x} + b$ is negative

Linear Decision Boundaries: Example



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*, $\boldsymbol{w}$ and 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 data points arrive gradually over time and we learn continuously
- Examples of online learning:
  - sensor data or internet-of-things
  - recommendations for social data
  - stock prices
  - self-driving car/robotics

Online Learning: Setup

- For $t = 1, 2, 3, \ldots$
  - Receive an unlabeled data point, $\boldsymbol{x}^{(t)}$
  - 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 data point, $\boldsymbol{x}^{(t)}$
  - 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 point ($y^{(t)} = +1, \hat{y} = -1$):
    - $\boldsymbol{w} \leftarrow \boldsymbol{w} + \boldsymbol{x}^{(t)}$
    - $b \leftarrow b + 1$
  - If we misclassified a negative point ($y^{(t)} = -1, \hat{y} = +1$):
    - $\boldsymbol{w} \leftarrow \boldsymbol{w} - \boldsymbol{x}^{(t)}$
    - $b \leftarrow b - 1$

(Online) Perceptron Learning Algorithm

- Initialize the weight vector and intercept to all zeros:

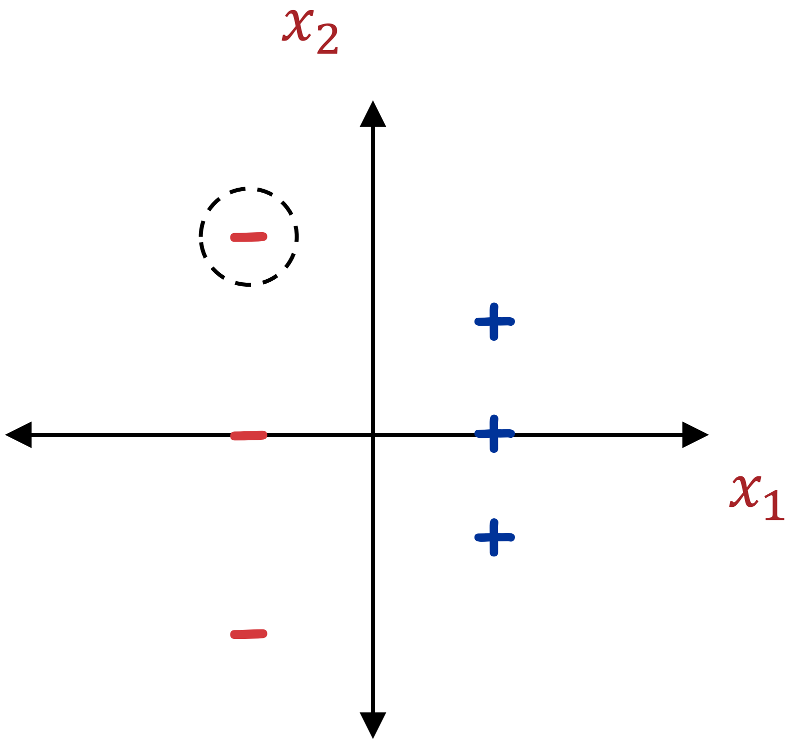
$$\boldsymbol{w} = [0 \quad 0 \quad \cdots \quad 0] \text{ and } b = 0$$

- For $t = 1, 2, 3, \dots$
  - Receive an unlabeled data point, $\boldsymbol{x}^{(t)}$
  - 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 point ($y^{(t)} \neq \hat{y}$):
    - $\boldsymbol{w} \leftarrow \boldsymbol{w} + y^{(t)}\boldsymbol{x}^{(t)}$
    - $b \leftarrow b + y^{(t)}$

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

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

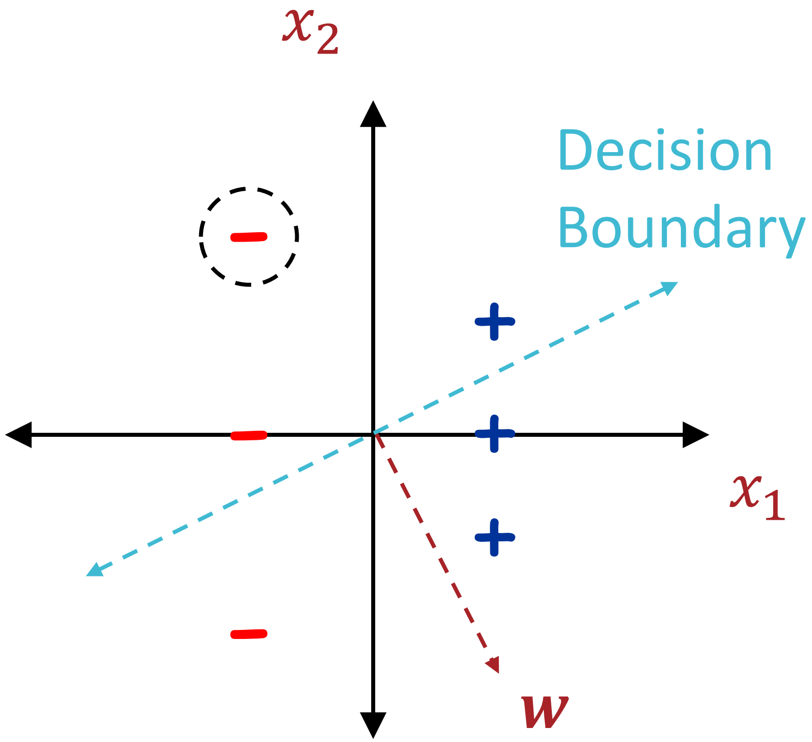
$$\boldsymbol{w} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$



Example courtesy of Nina Balcan

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

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



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

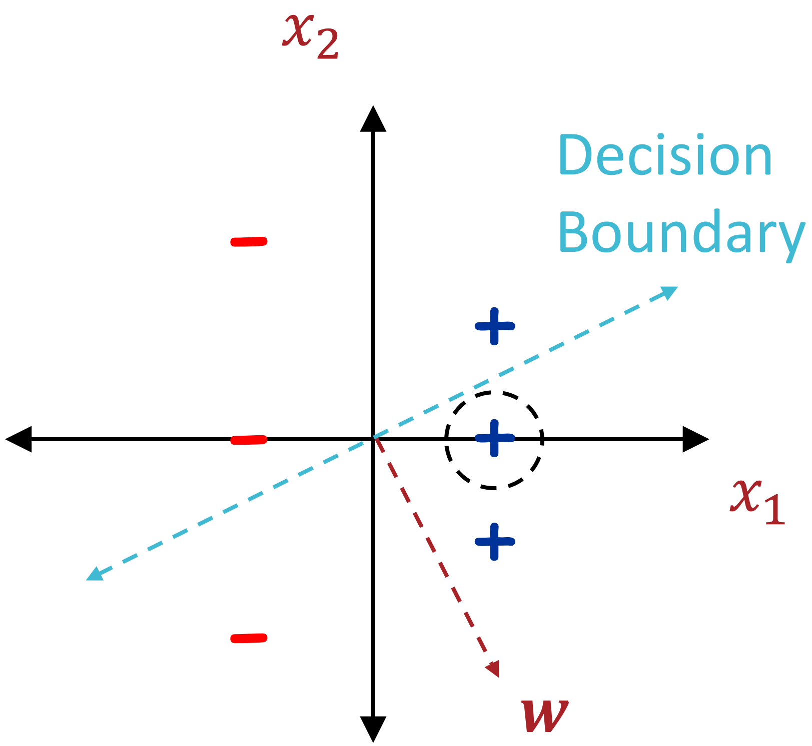
$$\boldsymbol{w} \leftarrow \boldsymbol{w} + y^{(1)}\boldsymbol{x}^{(1)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

Example courtesy of Nina Balcan

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

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

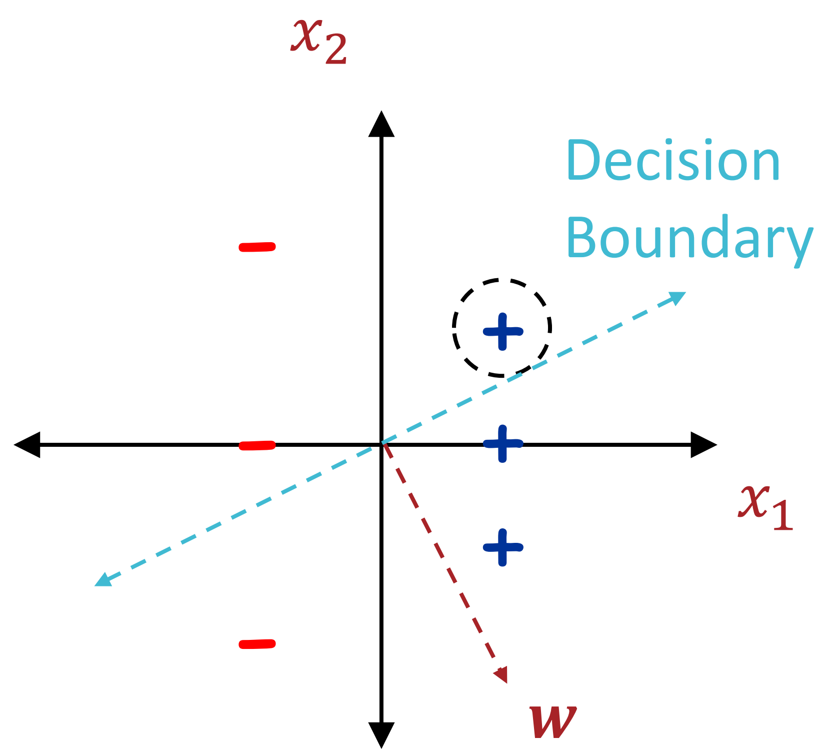
$$\boldsymbol{w} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$



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



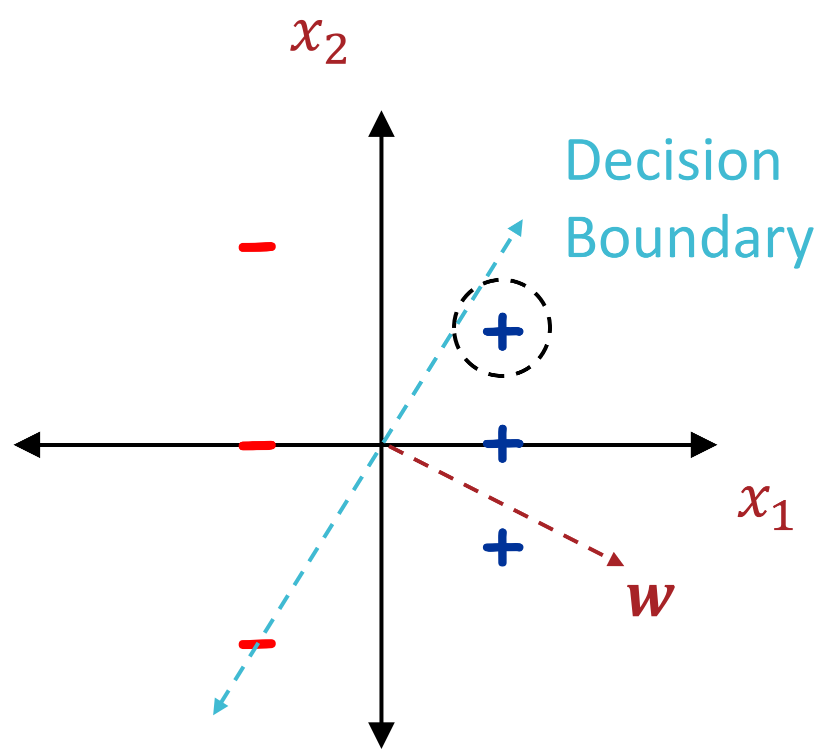
$$\boldsymbol{w} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

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

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



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

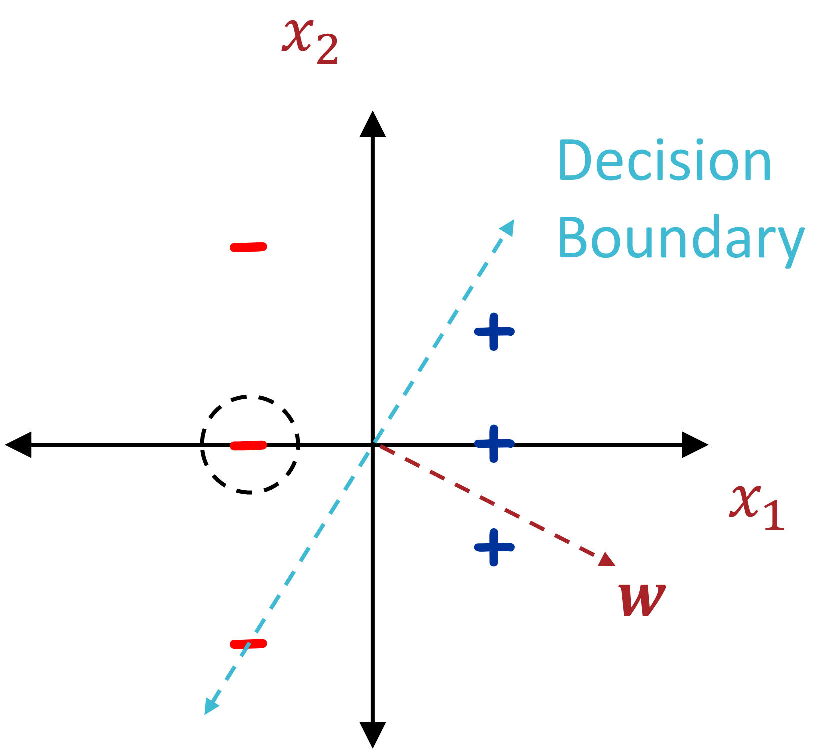
$$\boldsymbol{w} \leftarrow \boldsymbol{w} + y^{(3)}\boldsymbol{x}^{(3)} = \begin{bmatrix} 1 \\ -2 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$$

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

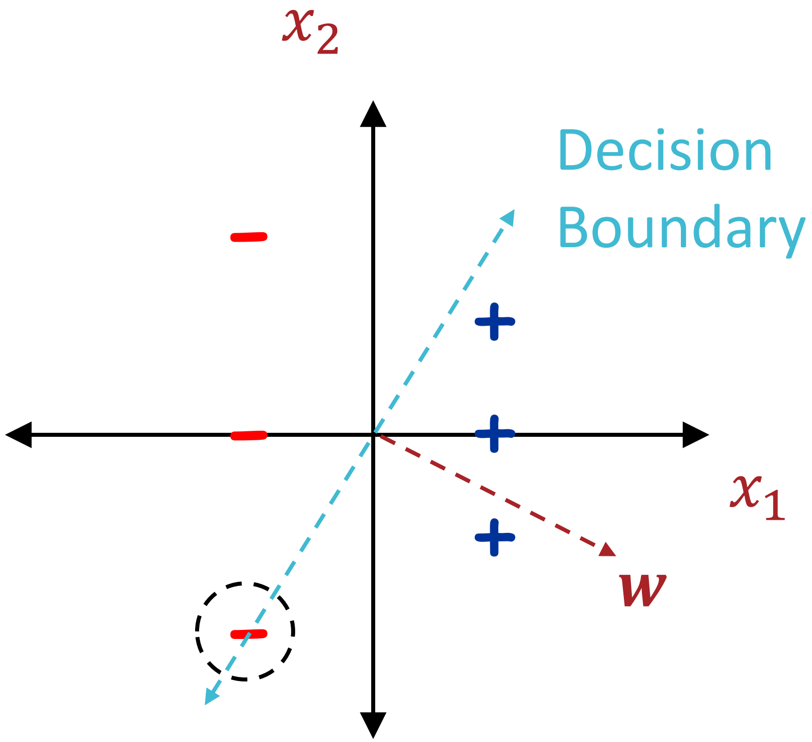
$$\boldsymbol{w} = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$$



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



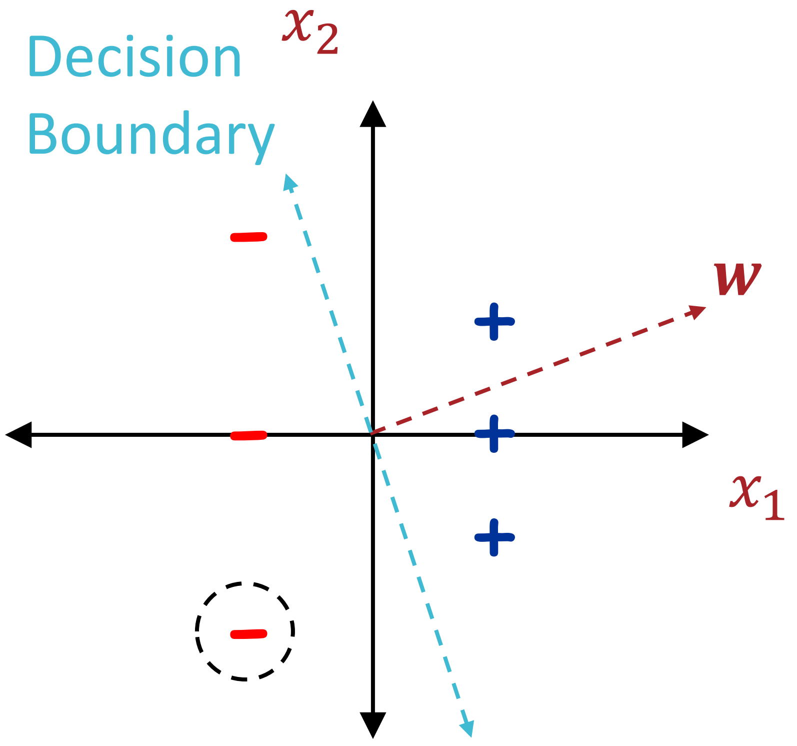
$$\boldsymbol{w} = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$$

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

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



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

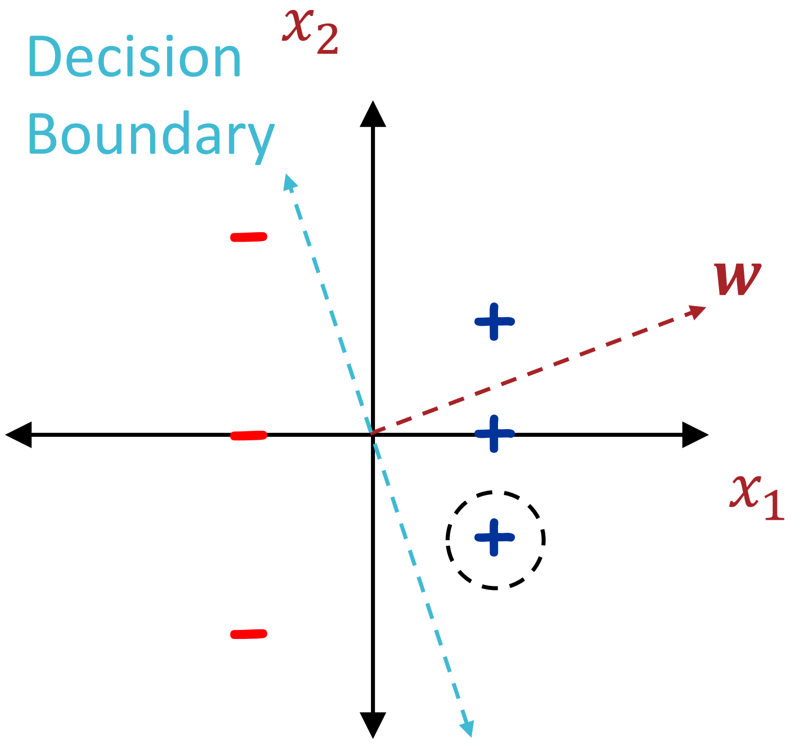
$$\boldsymbol{w} \leftarrow \boldsymbol{w} + y^{(5)}\boldsymbol{x}^{(5)} = \begin{bmatrix} 2 \\ -1 \end{bmatrix} - \begin{bmatrix} -1 \\ -2 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

Example courtesy of Nina Balcan

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

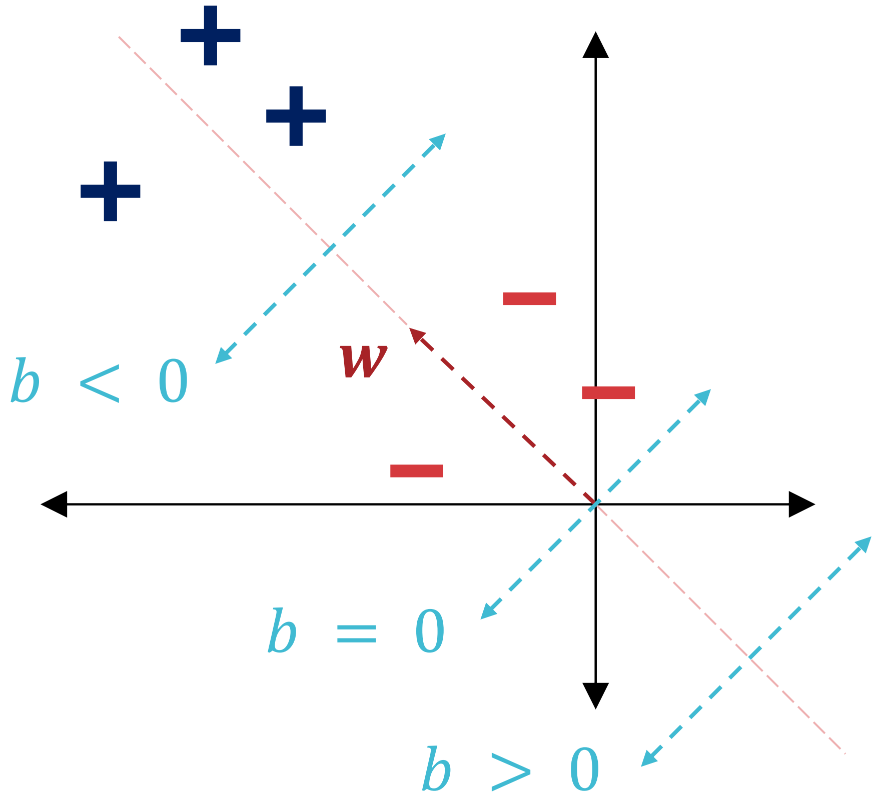
x_1	x_2	$\hat{y}$	y	Mistake?
-1	2	$+$	$-$	Yes
1	0	$+$	$+$	No
1	1	$-$	$+$	Yes
-1	0	$-$	$-$	No
-1	-2	$+$	$-$	Yes
1	-1	$+$	$+$	No

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



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 2:

- **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 the ~~training~~ validation data.

  A. True

  B. True and False **(TOXIC)**

  C. False

Notational Hack

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

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

- … 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 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 $t = 1, 2, 3, \dots$
  - Receive an unlabeled data point, $\boldsymbol{x}^{(t)}$
  - 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 point ($y^{(t)} \neq \hat{y}$):
    - $\boldsymbol{w} \leftarrow \boldsymbol{w} + y^{(t)}\boldsymbol{x}^{(t)}$
    - $b \leftarrow b + y^{(t)}$

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

Automatically handles updating the intercept

(Online) Perceptron Learning Algorithm: Inductive Bias

The true decision boundary is linear
and more recent mistakes are more
important to correct

(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 data point, $\boldsymbol{x}^{(t)}$
  - Predict its label, $\hat{y} = \text{sign}\left(\boldsymbol{\theta}^T \boldsymbol{x}'^{(t)}\right)$
  - Observe its true label, $y^{(t)}$
  - If we misclassified a point ($y^{(t)} \neq \hat{y}$):
    - $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + y^{(t)} \boldsymbol{x}'^{(t)}$

(Batch) Perceptron Learning Algorithm

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

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

- While NOT CONVERGED
  - For $t \in \{1, \dots, N\}$
    - 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 $\boldsymbol{x}'^{(t)}$ $(y^{(t)} \neq \hat{y})$:
      - $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + y^{(t)} \boldsymbol{x}'^{(t)}$

~~Poll Question 3:~~ (SKIPPED)

- **True or False**: The parameter vector $\boldsymbol{w}$ learned by the batch Perceptron Learning Algorithm can be **written as a linear combination** of the examples, i.e.,

$$\boldsymbol{w} = c_1 \boldsymbol{x}^{(1)} + c_2 \boldsymbol{x}^{(2)} + \cdots + c_N \boldsymbol{x}^{(M)}$$

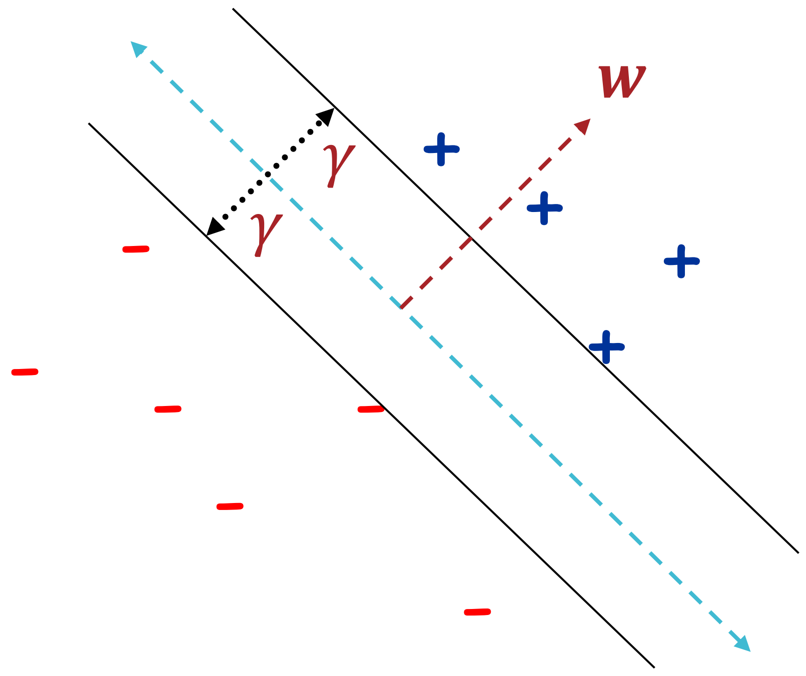
A. True and False **(TOXIC)**

B. True

C. False

Perceptron Mistake Bound

- Definitions:
  - A dataset $\mathcal{D}$ is *linearly separable* if ∃ a linear decision boundary that perfectly classifies the data points in $\mathcal{D}$
  - The margin, γ, of a dataset $\mathcal{D}$ is the greatest possible distance between a linear separator and the closest data point in $\mathcal{D}$ to that linear separator



Perceptron Mistake Bound

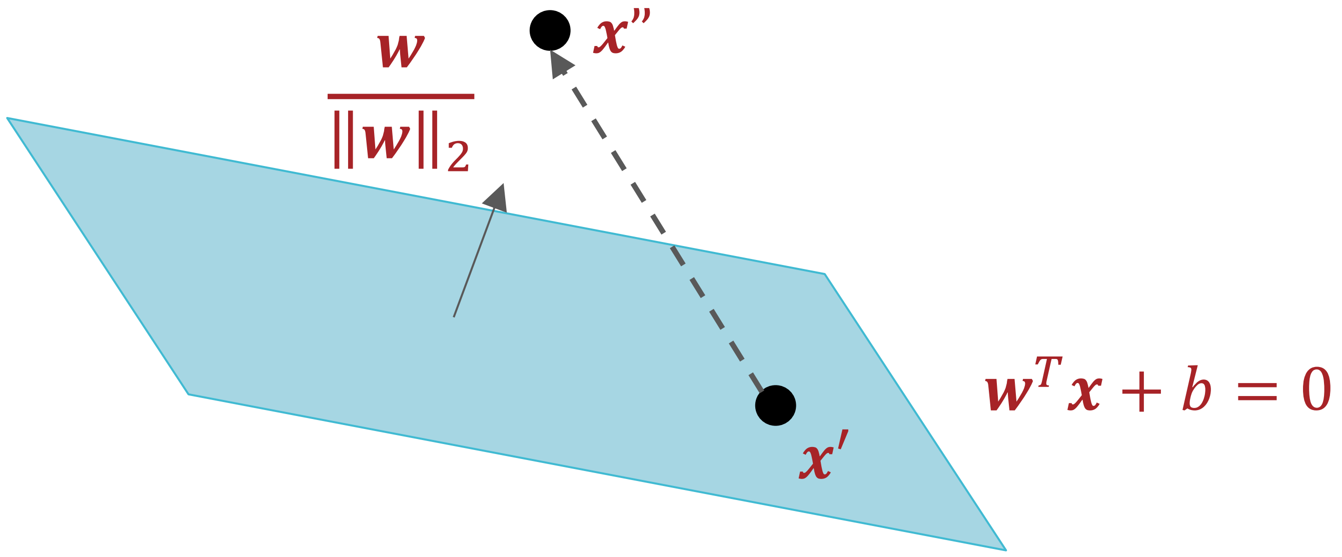
- Theorem: if the data points seen by the Perceptron Learning Algorithm (online and batch)
  1. lie in a ball of radius R (centered around the origin)
  2. have a margin of γ

then the algorithm makes at most $(R/\gamma)^2$ mistakes.

- 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!

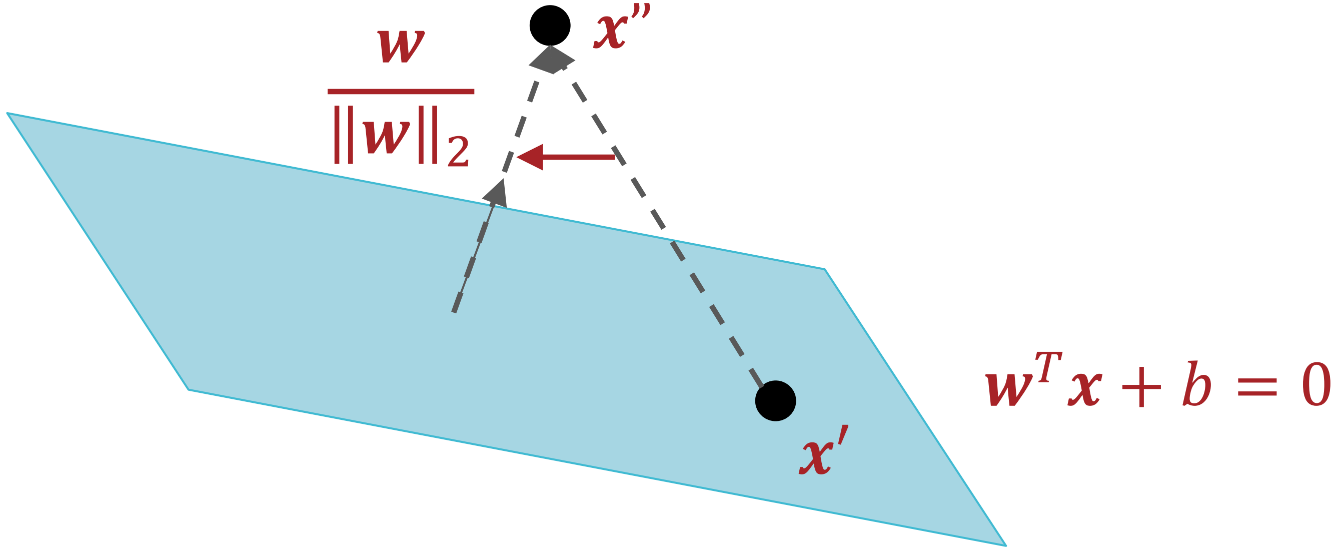
Computing the Margin

- 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



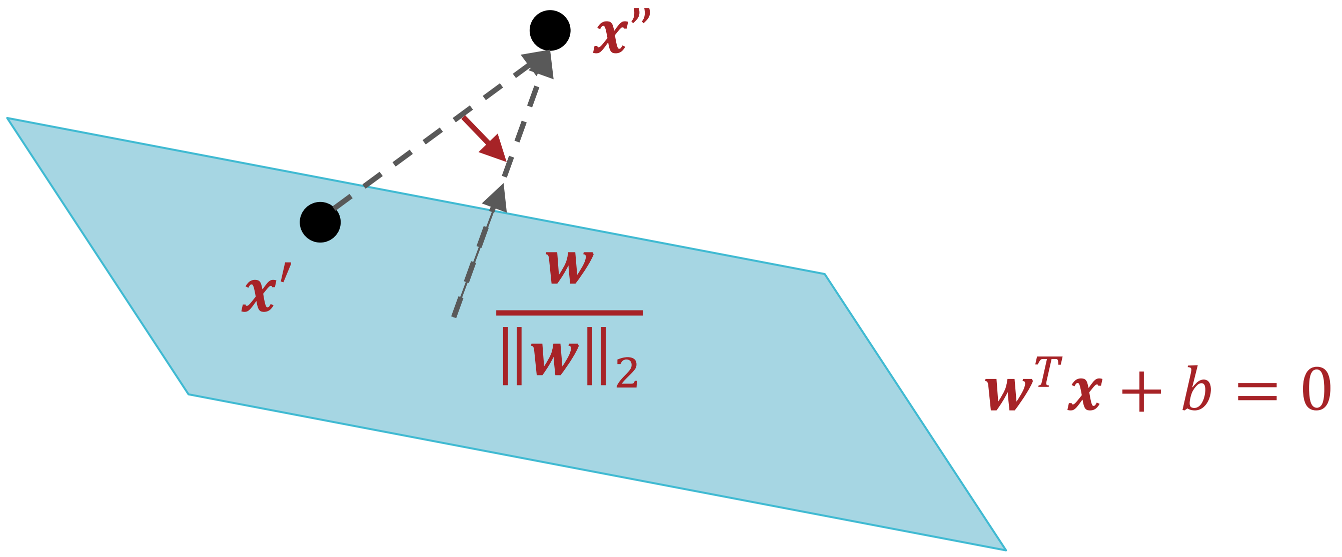
Computing the Margin

- 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



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



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

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)