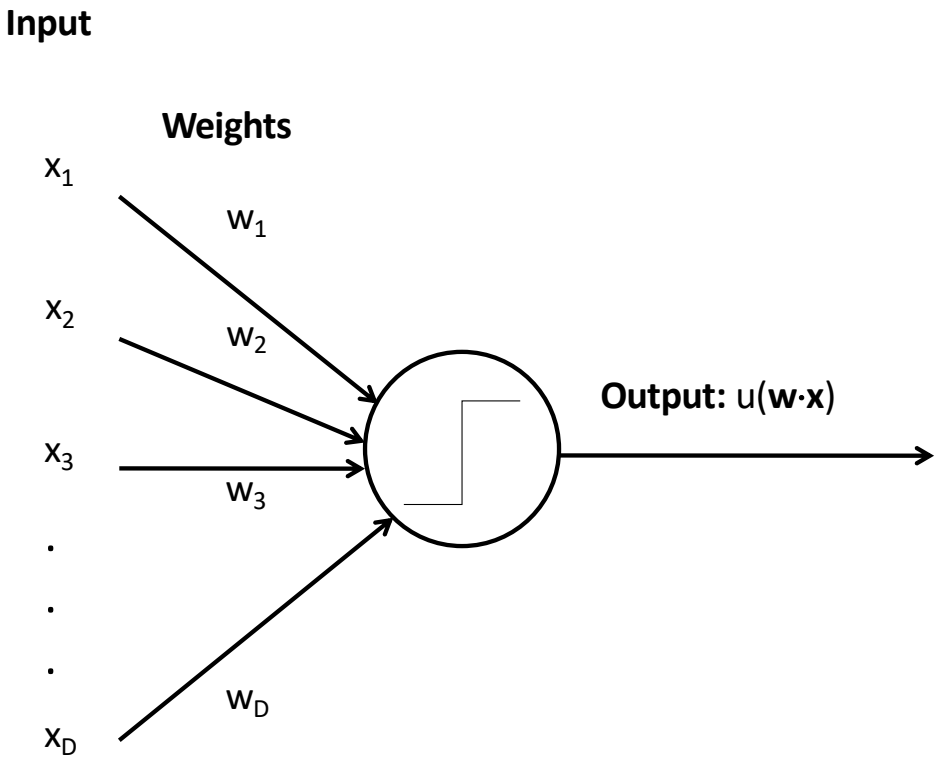
CS440/ECE448 Lecture 6: Perceptron

Mark Hasegawa-Johnson, 1/2022

Outline

- Perceptron: Teaching the McCulloch-Pitts neuron to learn
- Linear classifiers in general
- Relationship between weights and data in a linear classifier
- The perceptron learning algorithm
- Linear separability, yx, and convergence
  - It converges to the right answer, even with $\eta = 1$, if data are linearly separable
  - If data are not linearly separable, it's necessary to use $\eta = 1/n$.
- Multi-class perceptron

McCulloch-Pitts Artificial Neuron, 1943



- In 1943, McCulloch & Pitts proposed that biological neurons have a nonlinear activation function (a step function) whose input is a weighted linear combination of the currents generated by other neurons.
- They showed lots of examples of mathematical and logical functions that could be computed using networks of simple neurons like this.

Biological inspiration: Hebbian learning

"Neurons that fire together, wire together.

…

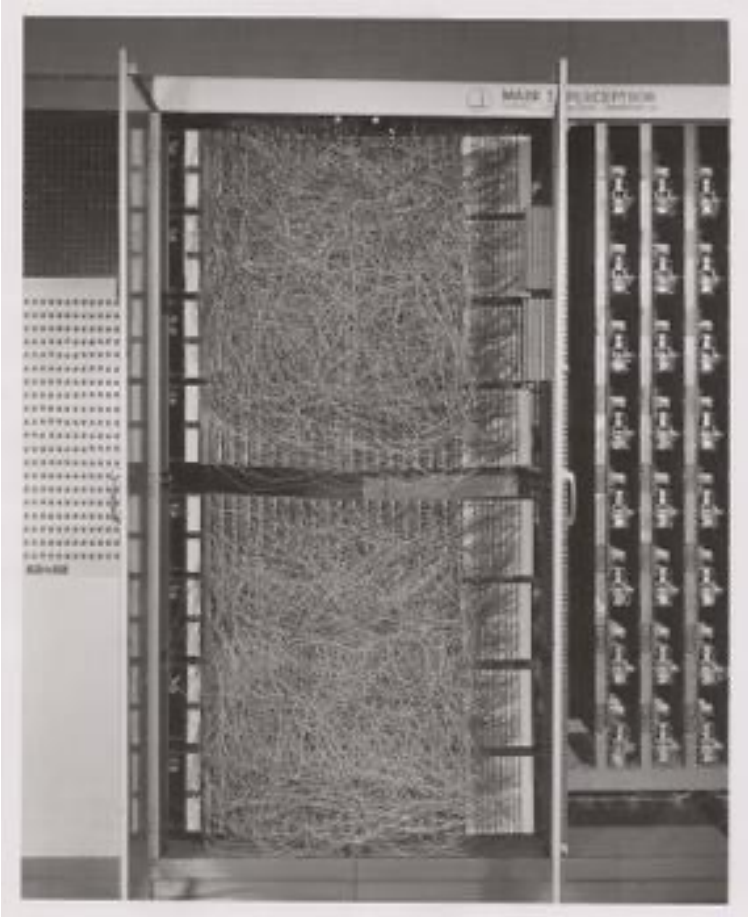
The general idea is an old one, that any two cells or systems of cells that are repeatedly active at the same time will tend to become `associated' so that activity in one facilitates activity in the other."

- D.O. Hebb, 1949

Rosenblatt's interpretation of Hebb's idea: add a supervision signal that tells the neuron what its output should have been.

- If the neuron got the right answer, don't change the weights.
- Else: If the correct answer was "+1", then adjust the weights so that $\sum_i w_i x_i$, in the McCulloch-Pitt neuron, is more positive
- Else: If the correct answer was "-1", then adjust the weights so that $\sum_i w_i x_i$, in the McCulloch-Pitt neuron, is more negative

Perceptron



1959: Rosenblatt is granted a patent for the “perceptron,” an electrical circuit model of a neuron, with the ability to learn from data.

Outline

- Perceptron: Teaching the McCulloch-Pitts neuron to learn
- Linear classifiers in general
- Relationship between weights and data in a linear classifier
- The perceptron learning algorithm
- Linear separability, yx, and convergence
  - It converges to the right answer, even with $\eta = 1$, if data are linearly separable
  - If data are not linearly separable, it's necessary to use $\eta = 1/n$.
- Multi-class perceptron

Classifier example: dogs versus cats

Can you write a program that can tell which ones are dogs, and which ones are cats?

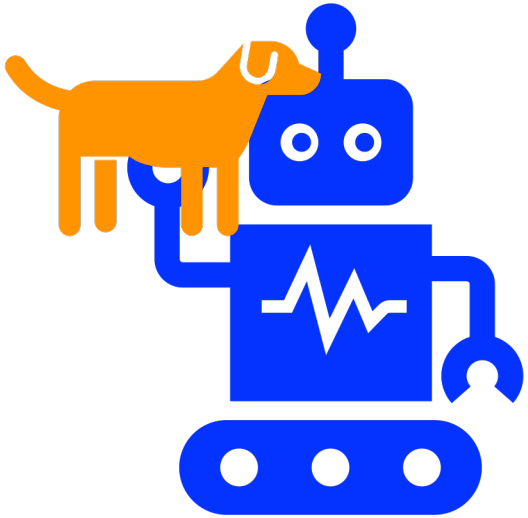
Classifier example: dogs versus cats

Can you write a program that can tell which ones are dogs, and which ones are cats?

Idea #1: Cats are smaller than dogs.

Our robot will pick up the animal and weigh it.

If it weighs more than 20 pounds, call it a dog. Otherwise, call it a cat.



Classifier example: dogs versus cats

Can you write a program that can tell which ones are dogs, and which ones are cats?

Oops.

Classifier example: dogs versus cats

Can you write a program that can tell which ones are dogs, and which ones are cats?

Idea #2: Dogs are tame, cats are wild.

We'll try the following experiment: 40 different people call the animal's name. Count how many times the animal comes when called.

If the animal comes when called, more than 20 times out of 40, it's a dog.

If not, it's a cat.

Classifier example: dogs versus cats

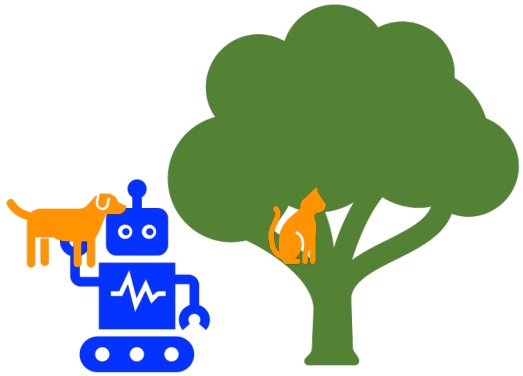
Can you write a program that can tell which ones are dogs, and which ones are cats?

Oops.

By Smok Bazyli - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=16864492

Classifier example: dogs versus cats

Can you write a program that can tell which ones are dogs, and which ones are cats?



Idea #3:

x_1 = # times the animal comes when called (out of 40).

x_2 = weight of the animal, in pounds.

If $0.5x_1 + 0.5x_2 > 20$, call it a dog.

Otherwise, call it a cat.

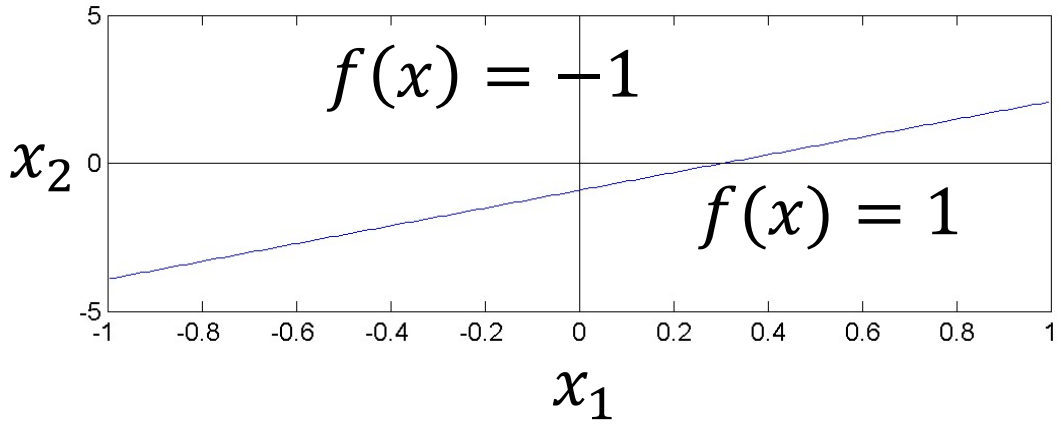
This is called a "linear classifier" because $0.5x_1 + 0.5x_2 = 20$ is the equation for a straight line.

Linear Classifiers in General

Consider the classifier

$$f(x) = 1 \quad \text{if} \quad b + \sum_{j=1}^{D} w_j x_j > 0, \qquad \text{otherwise } f(x) = -1$$

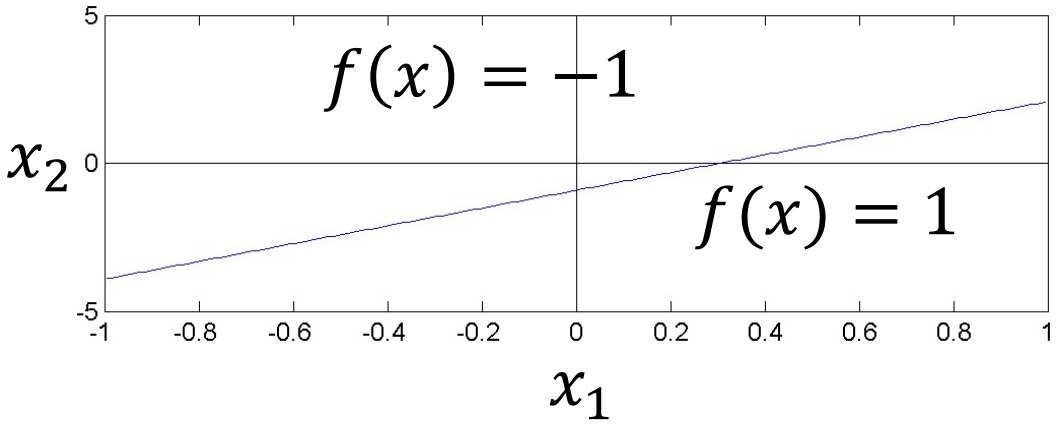
This is called a "D-dimensional linear classifier" because the boundary between the two classes is a line. Here is an example of such a classifier, with its boundary plotted as a line in the two-dimensional space x_1 by x_2:



Simplify notation: use the signum function

We can write this as

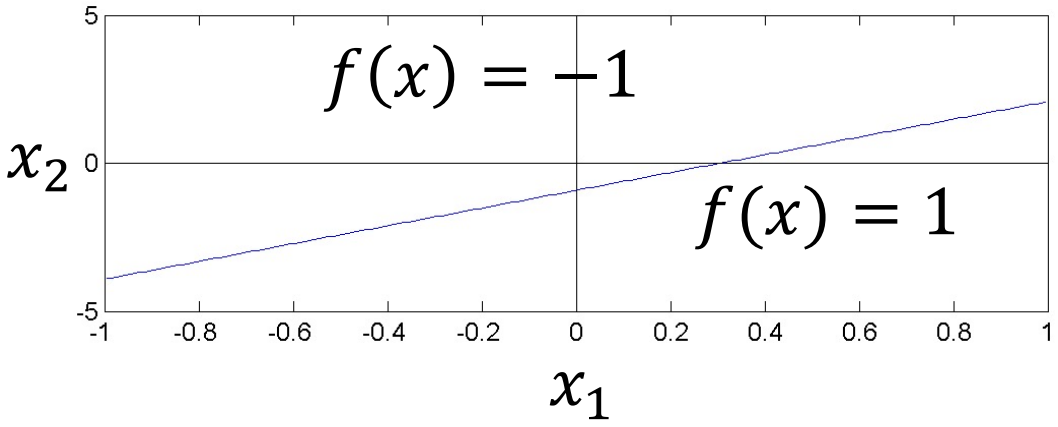
$$f(x) = \text{sign}\left(b + \sum_{j=1}^{D} w_j x_j\right), \qquad \text{sign}(\xi) = \begin{cases} 1 & \xi > 0 \\ -1 & \xi < 0 \end{cases}$$



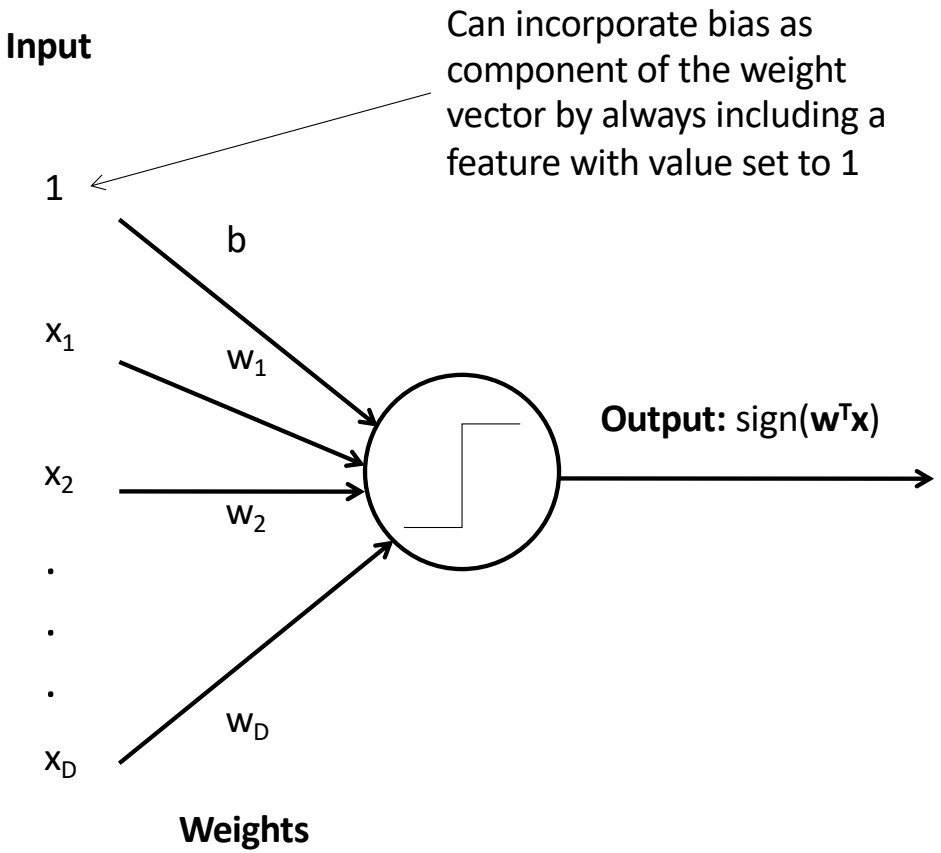
Simplify notation: use a vector dot product

We can save some space if we write this as a vector dot product:

$$f(\vec{x}) = \text{sign}(\vec{w}^T\vec{x}), \qquad \vec{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_D \\ 1 \end{bmatrix}, \vec{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_D \\ b \end{bmatrix}, \qquad \vec{w}^T\vec{x} = b + \sum_{j=1}^{D} w_j x_j$$



McCulloch-Pitt Neuron is a Linear Classifier



In particular, the M-P neuron inspires these two names:

- The neuron's **excitation** is

$$\vec{w}^T \vec{x} = b + \sum_{j=1}^{D} w_j x_j$$

- The neuron's **activation** is

$$f(x) = \text{sign}(\vec{w}^T \vec{x})$$

Outline

- Perceptron: Teaching the McCulloch-Pitts neuron to learn
- Linear classifiers in general
- Relationship between weights and data in a linear classifier
- The perceptron learning algorithm
- Linear separability, yx, and convergence
  - It converges to the right answer, even with $\eta = 1$, if data are linearly separable
  - If data are not linearly separable, it's necessary to use $\eta = 1/n$.
- Multi-class perceptron

Relationship between weights and data

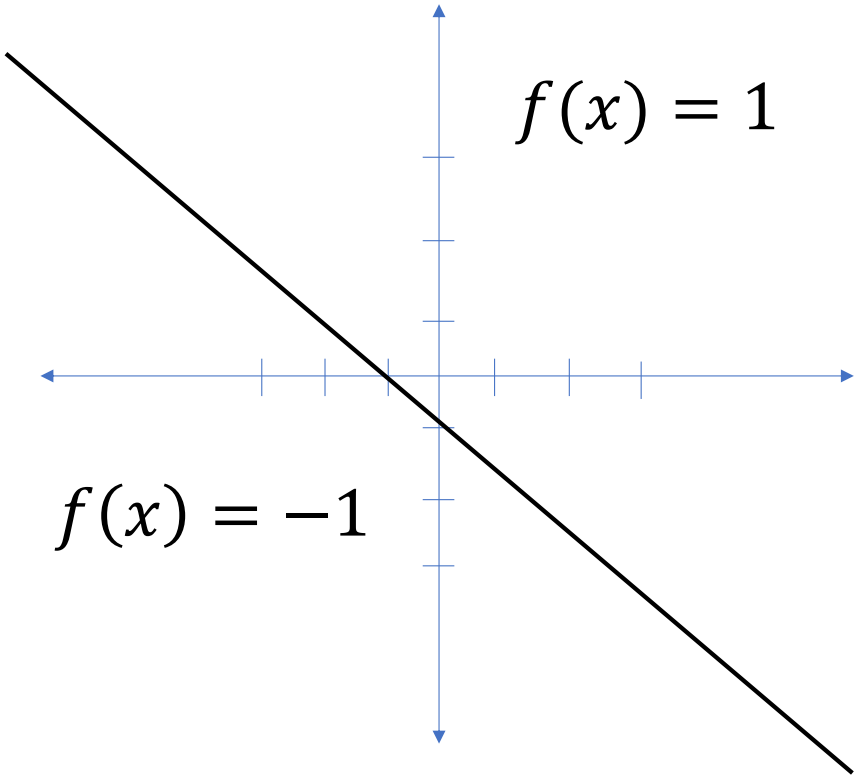
Consider the classifier

$$f(x) = \text{sign}\left(1 + \sum_{j=1}^{D} x_j\right)$$

i.e.,

$$\vec{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_D \\ 1 \end{bmatrix}, \vec{w} = \begin{bmatrix} 1 \\ \vdots \\ 1 \\ 1 \end{bmatrix}$$

Then the boundary between class +1 and class -1 is…

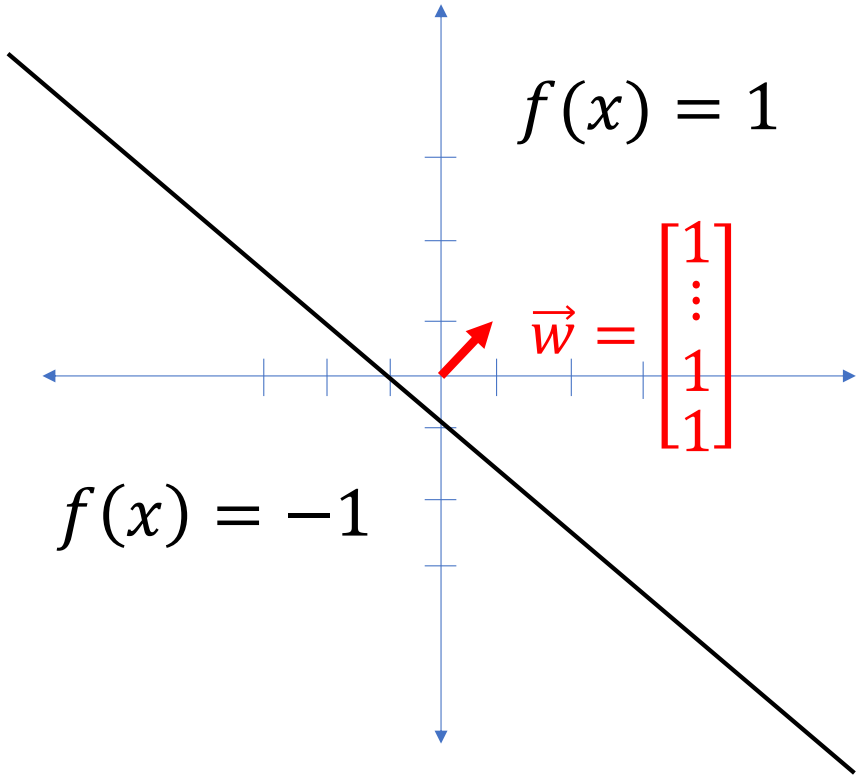


Relationship between weights and data

Notice that the vector

$$\vec{w} = \begin{bmatrix} 1 \\ \vdots \\ 1 \\ 1 \end{bmatrix}$$

… is perpendicular to the boundary between the two classes.

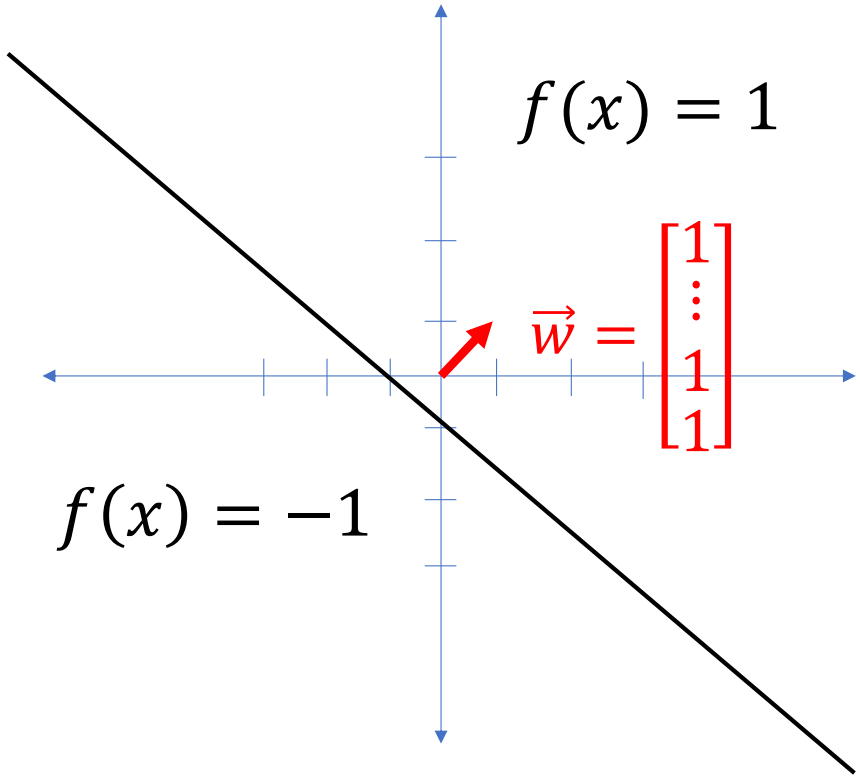


Relationship between weights and data

This is a fundamental geometry fun-fact! If

$$f(x) = \text{sign}(\vec{w}^T \vec{x})$$

… then the boundary between $f(x) = 1$ and $f(x) = -1$ is always a hyperplane perpendicular to the vector $\vec{w}$.

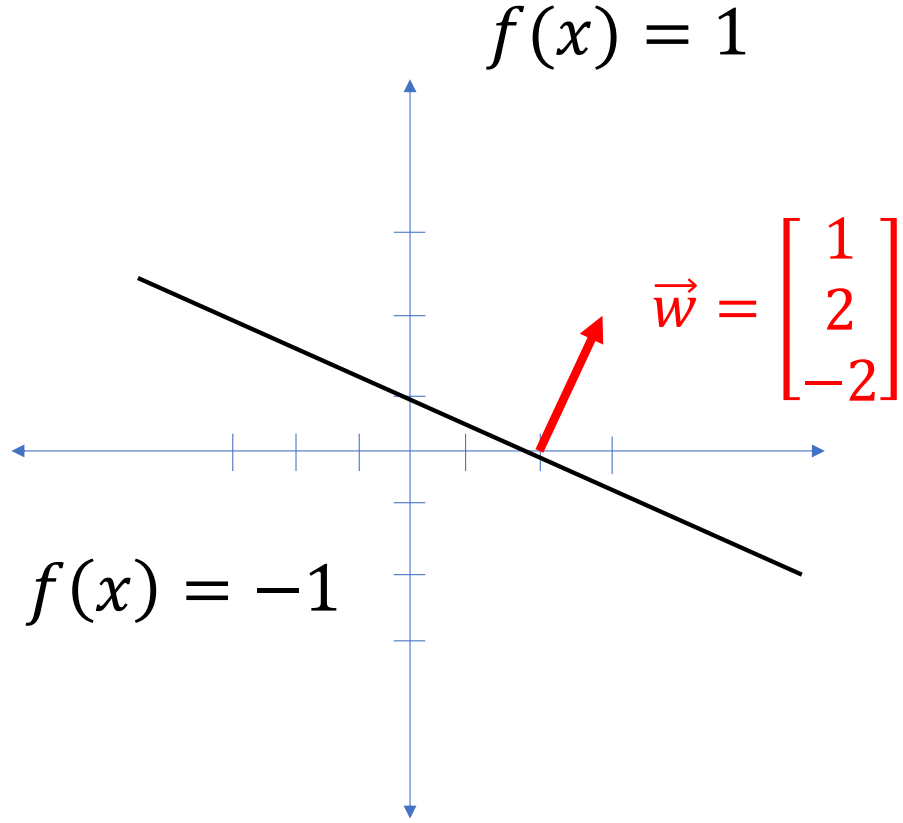


Another example

$$f(x) = \text{sign}(-2 + x_1 + 2x_2)$$

i.e.,

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}, \vec{w} = \begin{bmatrix} 1 \\ 2 \\ -2 \end{bmatrix}$$

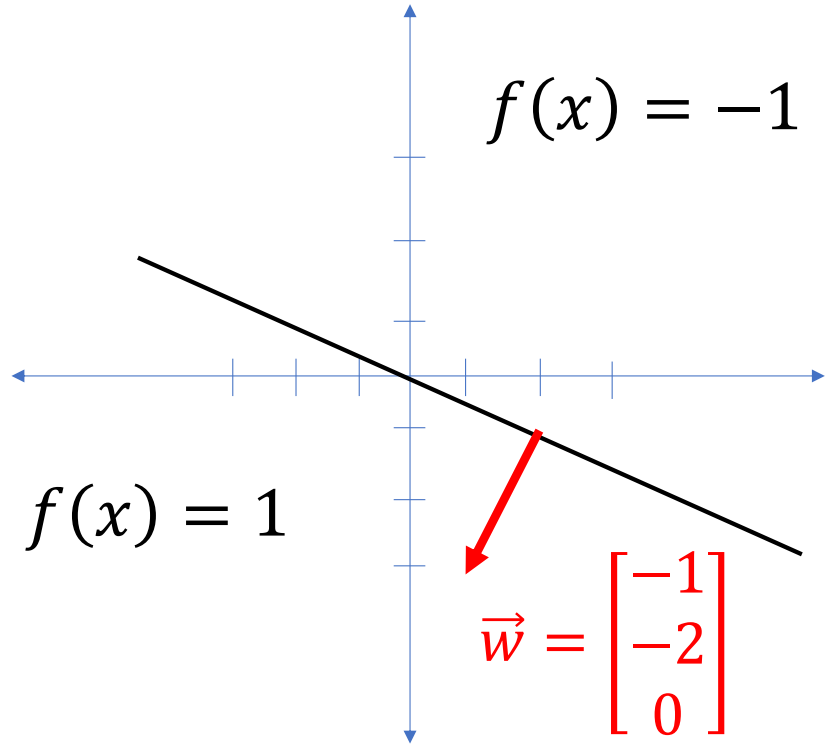


Another example

$$f(x) = \text{sign}(-x_1 - 2x_2)$$

i.e.,

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}, \vec{w} = \begin{bmatrix} -1 \\ -2 \\ 0 \end{bmatrix}$$



Outline

- Perceptron: Teaching the McCulloch-Pitts neuron to learn
- Linear classifiers in general
- Relationship between weights and data in a linear classifier
- The perceptron learning algorithm
- Linear separability, yx, and convergence
  - It converges to the right answer, even with $\eta = 1$, if data are linearly separable
  - If data are not linearly separable, it's necessary to use $\eta = 1/n$.
- Multi-class perceptron

Perceptron Learning Algorithm

Rosenblatt's interpretation of Hebb's idea: add a supervision signal that tells the neuron what its output should have been.

- If the neuron got the right answer, don't change the weights.
- Else: If the correct answer was "+1", then adjust the weights so that $\sum_i w_i x_i$, in the McCulloch-Pitt neuron, is more positive
- Else: If the correct answer was "-1", then adjust the weights so that $\sum_i w_i x_i$, in the McCulloch-Pitt neuron, is more negative

Perceptron Learning Algorithm

Rosenblatt's interpretation of Hebb's idea: add a supervision signal that tells the neuron what its output should have been.

- If the neuron got the right answer, don't change the weights.
- Else: If the correct answer was "+1", then adjust the weights so that $\sum_i w_i x_i$, in the McCulloch-Pitt neuron, is more positive by setting

$$\vec{w} = \vec{w} + \vec{x}$$

- Else: If the correct answer was "-1", then adjust the weights so that $\sum_i w_i x_i$, in the McCulloch-Pitt neuron, is more negative:

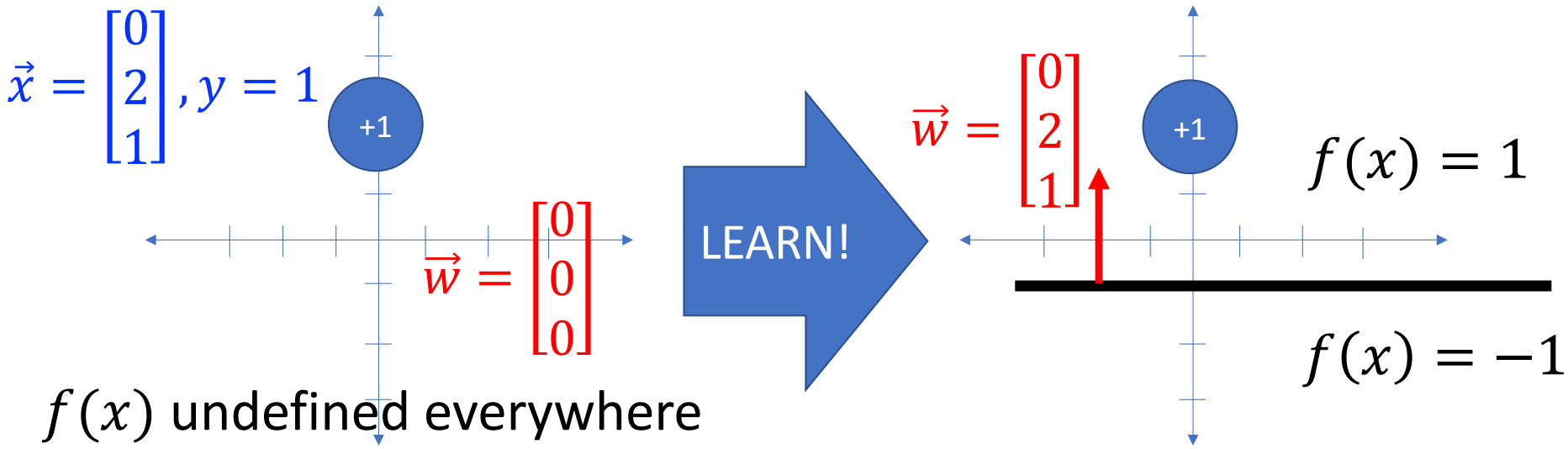
$$\vec{w} = \vec{w} - \vec{x}$$

Example

Start with $\vec{w}^T = [0,0,0]$. Then, no matter what $\vec{x}$ is, $f(x) = \text{sign}(\vec{w}^T\vec{x})$ will be wrong, because it's undefined.

Suppose that $\vec{x}^T = [0,2,1]$, with the label $y = +1$. Since the $f(x)$ was wrong, we update:

$$\vec{w} = \vec{w} + \vec{x}$$



Example

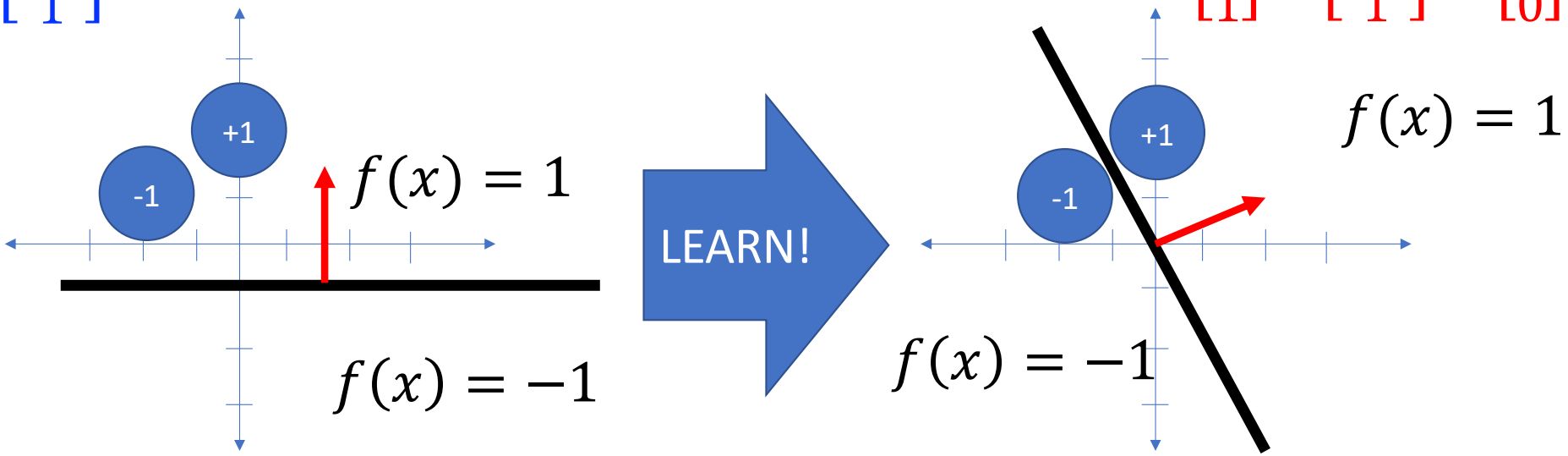
Now we have $\vec{w}^T = [0,2,1]$.

Suppose the next token is $\vec{x}^T = [-2,1,1]$, with the label $y = -1$. Since $f(x)$ is wrong, we update:

$$\vec{w} = \vec{w} - \vec{x}$$

$$\vec{x} = \begin{bmatrix} -2 \\ 1 \\ 1 \end{bmatrix}, y = -1$$

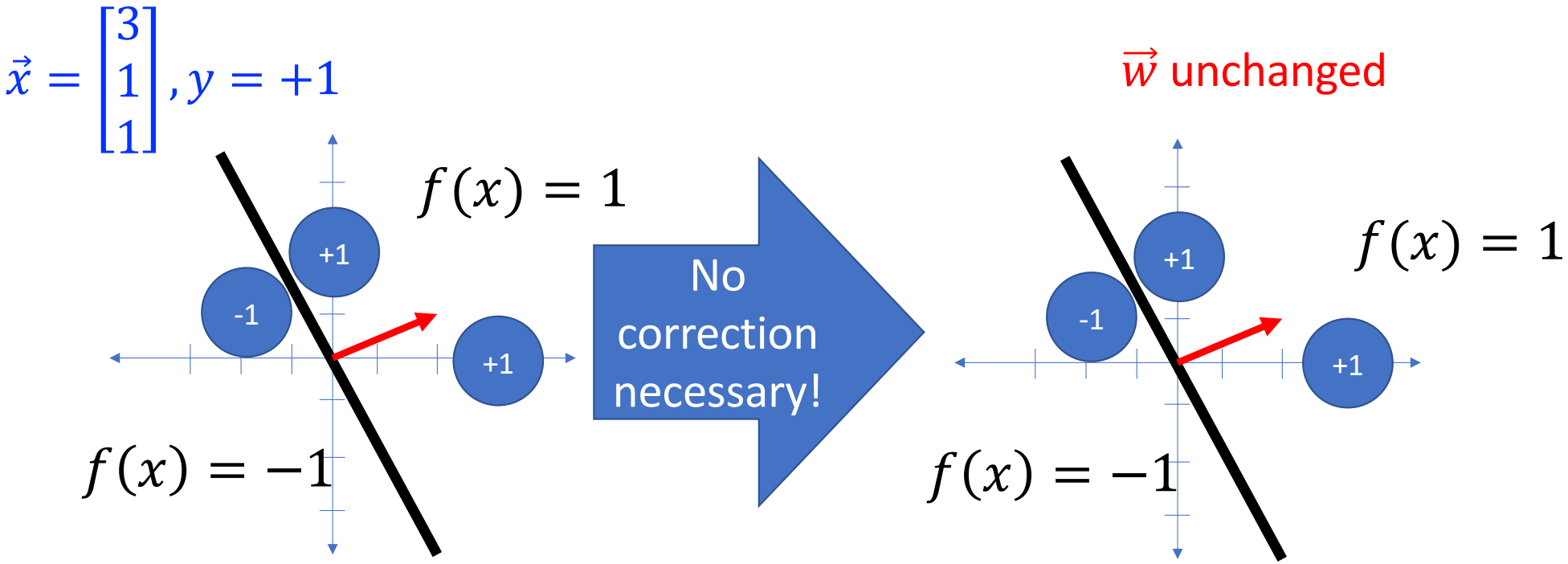
$$\vec{w} = \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix} - \begin{bmatrix} -2 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix}$$



Example

Suppose the next token is $\vec{x}^T = [3,0,1]$, with the label $y = +1$. Since $f(x)$ is right, the weights don't need to be updated:

$$\vec{w} = \vec{w}$$



Perceptron Learning Algorithm

For each training instance $\vec{x}$ with ground truth label $y \in \{-1,1\}$:

- Classify with current weights: $f(\vec{x}) = \text{sgn}(\vec{w}^T \vec{x})$
- Update weights:
  - If $f(\vec{x}) = y$ then do nothing
  - If $f(\vec{x}) \neq y$ then

$$\vec{w} = \vec{w} + y\vec{x} = \begin{cases} \vec{w} + \vec{x} & y = +1 \\ \vec{w} - \vec{x} & y = -1 \end{cases}$$

Outline

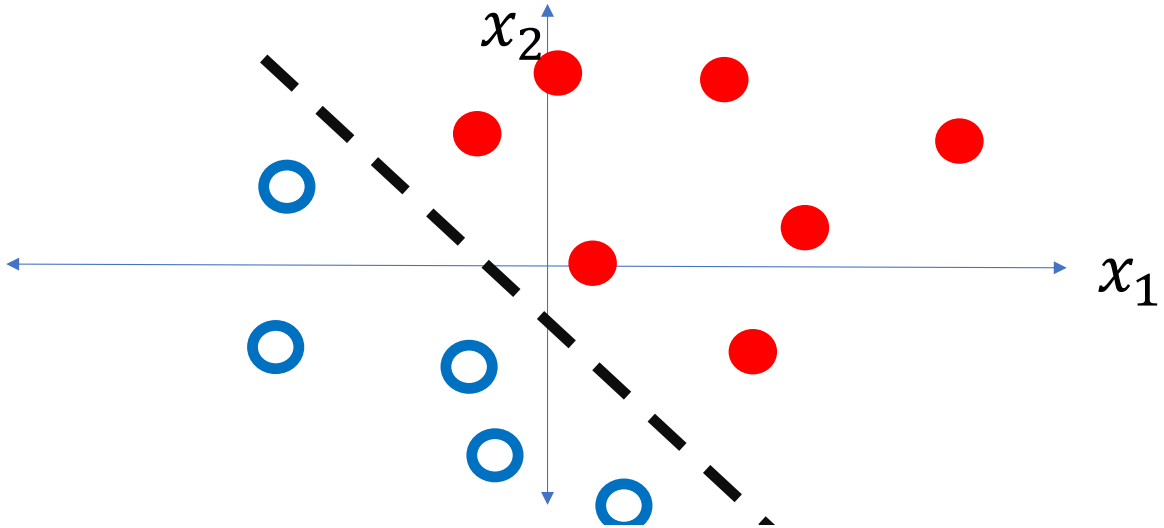
- Perceptron: Teaching the McCulloch-Pitts neuron to learn
- Linear classifiers in general
- Relationship between weights and data in a linear classifier
- The perceptron learning algorithm
- Linear separability, yx, and convergence
  - It converges to the right answer, even with $\eta = 1$, if data are linearly separable
  - If data are not linearly separable, it's necessary to use $\eta = 1/n$.
- Multi-class perceptron

Does the perceptron find an answer? Does it find the correct answer?

Suppose we run the perceptron algorithm for a very long time. Will it converge to an answer? Will it converge to the correct answer?

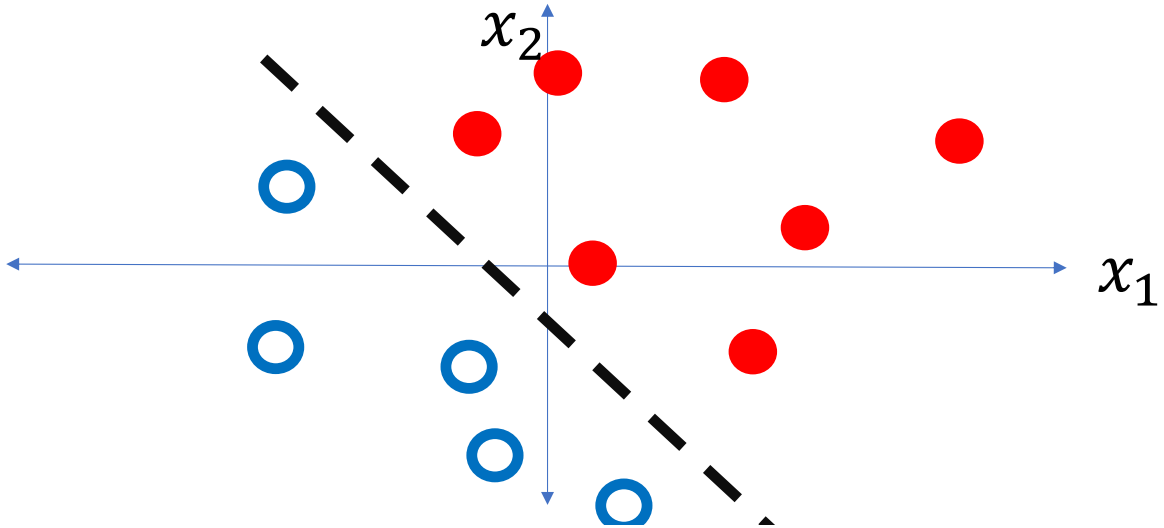
That depends on whether or not the data are linearly separable.

”Linearly separable” means it’s possible to find a line (a hyperplane, in D-dimensional space) that separates the two classes, like this:



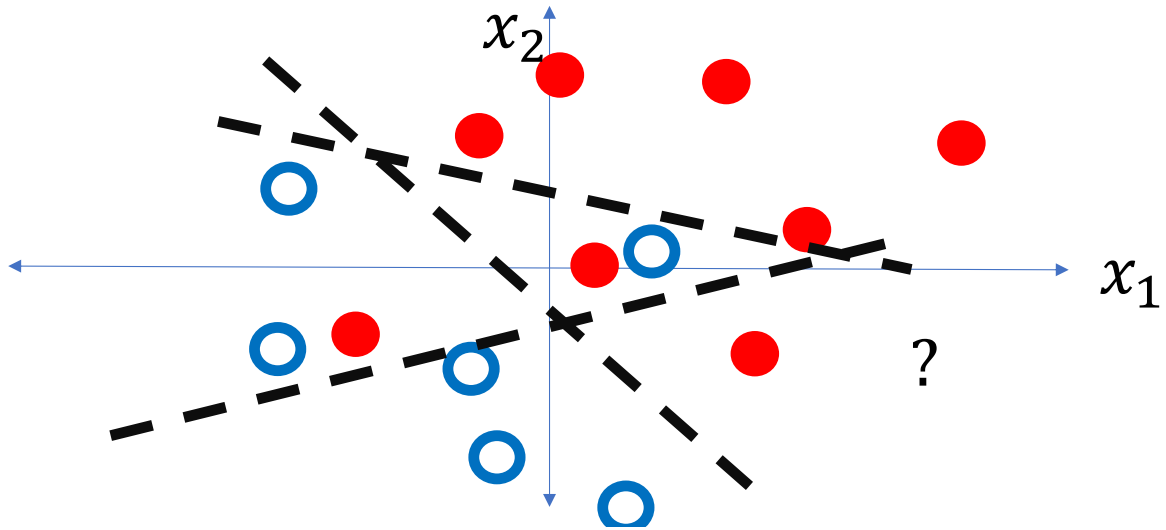
Convergence of the Perceptron

- If the data are linearly separable, as shown below, then the perceptron is guaranteed to reach a value of $\vec{w}$ that classifies the training data with 100% accuracy.
- When it reaches 100% accuracy, there are no more mistakes, therefore $\vec{w}$ stops changing! We say that it has "converged."



Convergence of the Perceptron

- If the data are NOT linearly separable, as shown below, then every possible value of $\vec{w}$ will make some mistakes.
- Therefore $\vec{w}$ never stops changing.
- The perceptron algorithm never converges.



Perceptron with learning rate

- We can force the perceptron to converge to a reasonably good answer, even if the data aren't linearly separable, by multiplying the update by a learning rate, η:
  - If $f(\vec{x}) = y$ then do nothing
  - If $f(\vec{x}) \neq y$ then

$$\vec{w} = \vec{w} + \eta y \vec{x} = \begin{cases} \vec{w} + \eta \vec{x} & y = +1 \\ \vec{w} - \eta \vec{x} & y = -1 \end{cases}$$

- We force it to converge by choosing a value of η that gets gradually smaller. For example, on the n^{th} iteration of training, we can set $\eta = \frac{1}{n}$.

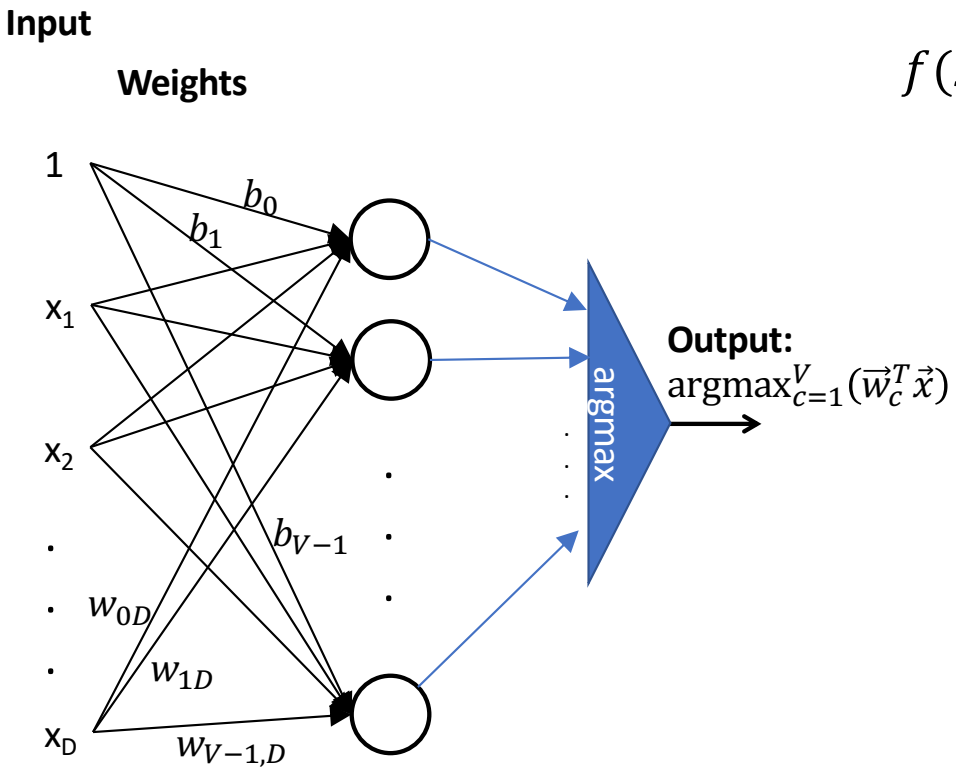
Outline

- Perceptron: Teaching the McCulloch-Pitts neuron to learn
- Linear classifiers in general
- Relationship between weights and data in a linear classifier
- The perceptron learning algorithm
- Linear separability, yx, and convergence
  - It converges to the right answer, even with $\eta = 1$, if data are linearly separable
  - If data are not linearly separable, it's necessary to use $\eta = 1/n$.
- Multi-class perceptron

Multi-Class Perceptron

True class is $y \in \{1,2,\ldots,V\}$.

Classifier output is

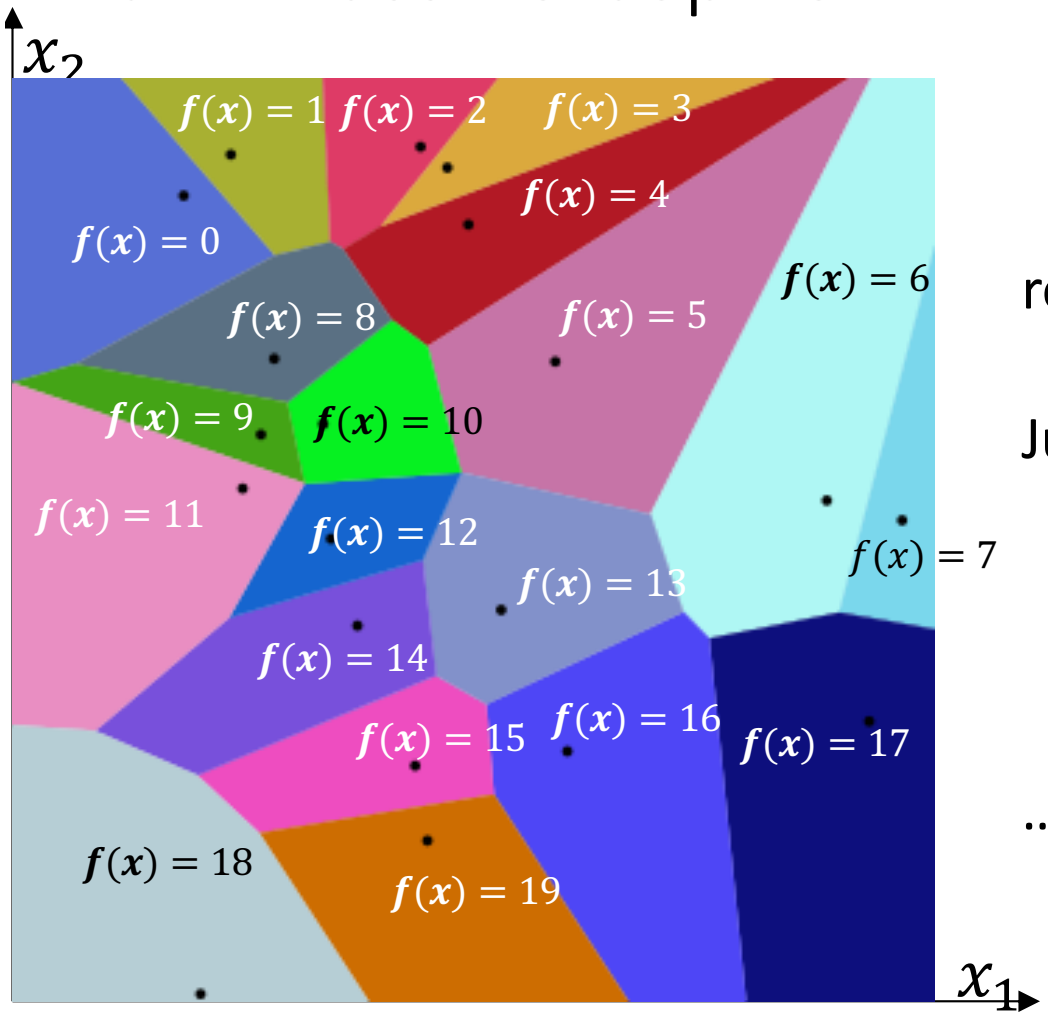


$$f(x) = \text{argmax}_{c=0}^{V-1}\left(w_{c,1}x_1 + \cdots + w_{c,D}x_D + b_c\right)$$

$$= \text{argmax}_{c=1}^{V}(\vec{w}_c^T \vec{x})$$

$$\vec{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_D \\ 1 \end{bmatrix}, \vec{w}_c = \begin{bmatrix} w_{c,1} \\ \vdots \\ w_{c,D} \\ b_c \end{bmatrix}$$

Multi-Class Perceptron



The argmax classifier,

$$f(\vec{x}) = \text{argmax}_{c=1}^{V}(\vec{w}_c^T \vec{x})$$

results in classification boundaries that are piece-wise linear.

Just think about why that is: the boundary between the regions for y=1 and y=2 is the line where

$$\vec{w}_1^T \vec{x} = \vec{w}_2^T \vec{x}$$

… or in other words, $(\vec{w}_1 - \vec{w}_2)^T \vec{x} = 0$, which is the equation for a straight line!

Training a Multi-Class Perceptron

First, classify a training token, $f(\vec{x}) = \text{argmax}_{c=1}^{V}(\vec{w}_c^T \vec{x})$. Then:

- If $f(\vec{x}) = y$ then do nothing
- If $f(\vec{x}) \neq y$ then
  - **Add** x to the vector that **should** have been the winner:

$$\vec{w}_y = \vec{w}_y + \eta\vec{x}$$

  - **Subtract** x from the vector that **shouldn't** have won, but did:

$$\vec{w}_{f(x)} = \vec{w}_{f(x)} - \eta\vec{x}$$

  - Don't change any of the other classes

Conclusions

- Perceptron as a model of a biological neuron: $f(x) = \text{sign}(w^T x)$
- The perceptron learning algorithm: if $y = f(x)$ then do nothing, else $w = w + \eta y x$.
- Linear separability, yx, and convergence
  - It converges to the right answer, even with $\eta = 1$, if data are linearly separable
  - If data are not linearly separable, it's necessary to use $\eta = 1/n$.
- Multi-class perceptron: if $y = f(x)$ then do nothing, else $w_y = w_y + \eta x$, and $w_{f(x)} = w_{f(x)} - \eta x$.