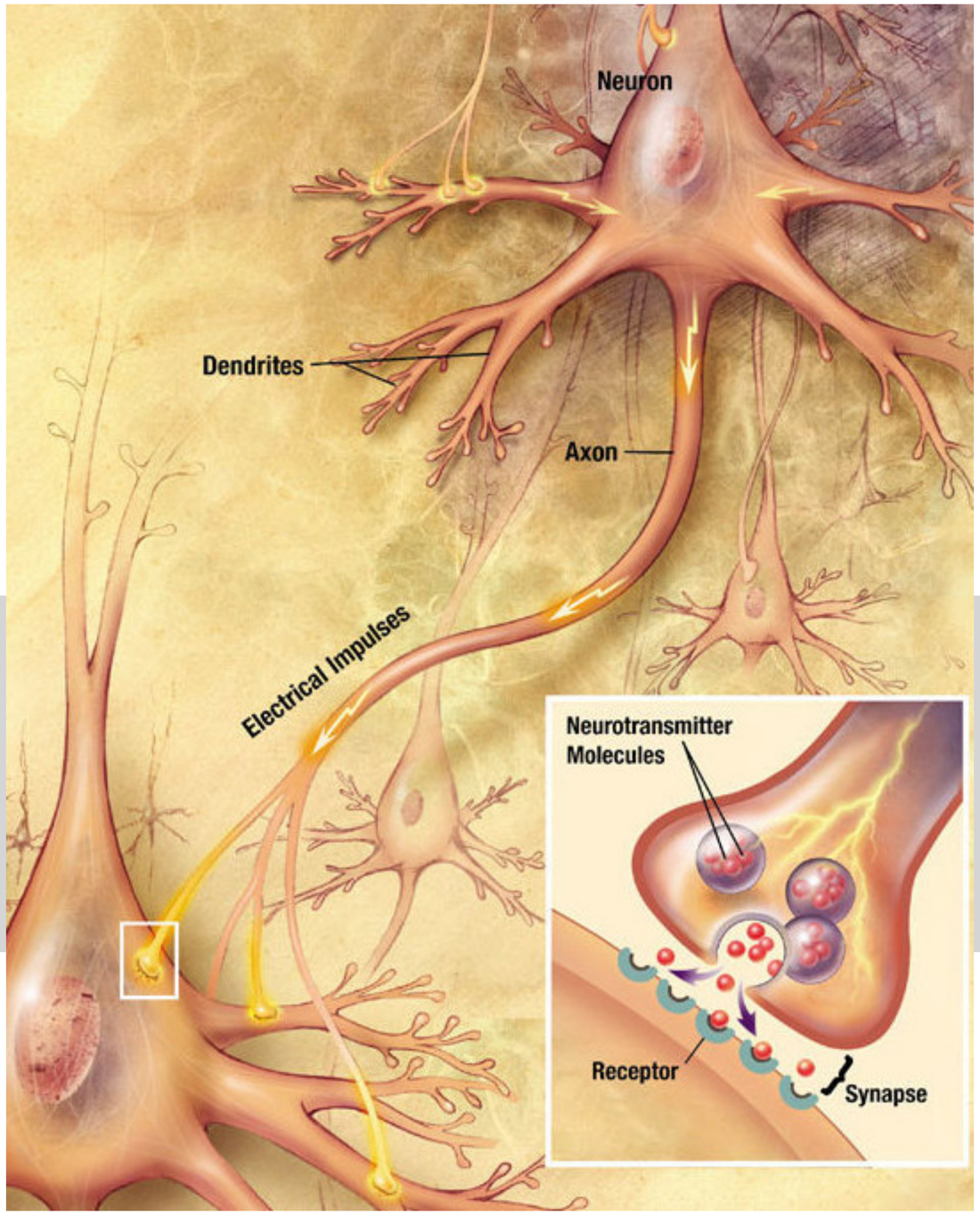
Neural Networks

Jason G. Fleischer, Ph.D.
Asst. Teaching Professor
Department of Cognitive Science, UC San Diego

jfleischer@ucsd.edu

@jasongfleischer

https://jgfleischer.com



https://commons.wikimedia.org/wiki/File:Chemical_synapse_schema_cropped.jpg

Neuroscience 101

Nervous system

COGS 107 in < 10 minutes

- “Materialism”… the functioning of mind is ONLY the actions and states of matter… and therefore the nervous system IS the mind. [so many famous philosphers over centuries]

Nervous system

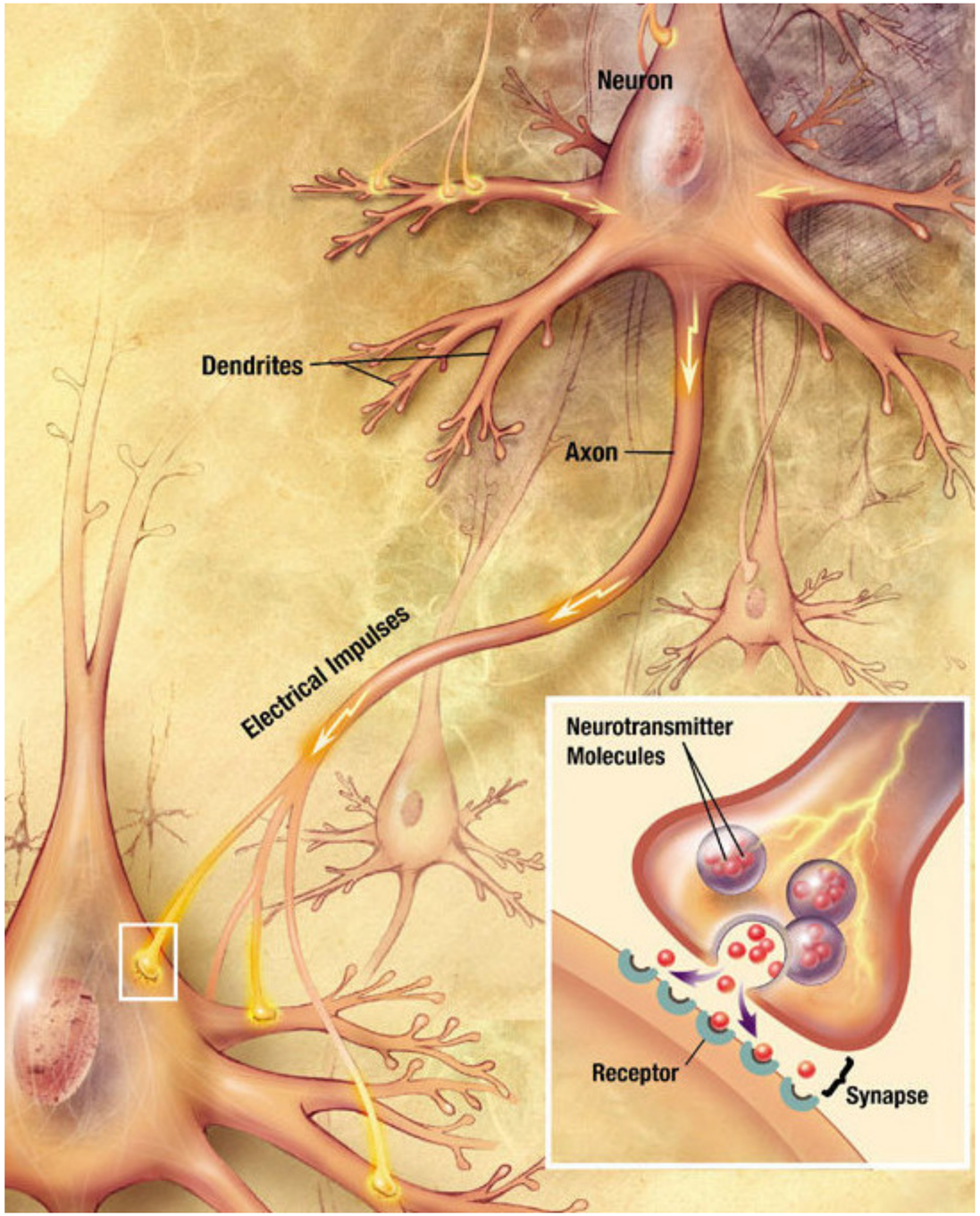
COGS 107 in < 10 minutes

- The “Neuron Doctrine” … the nervous system is made up of a bunch of cells that signal to each other through electrical and chemical means. [1906 Nobel prize to Ramon y Cajal]

Nervous system

COGS 107 in < 10 minutes

- Neurons integrate inputs from other neurons via synaptic connections.
- A neuron will “fire” if it gets enough input, giving input to downstream neurons
- Neurons in your sense organs have receptors for light/ vibration/odors/touch/temperature/etc that begin the cascade of signaling
- Neurons connect to spinal cord or directly to muscles and cause motion
- We know a lot about how things work close to the sense organs or the muscles, but we know a lot less about how neurons a few synapses away from there contribute to behavior and cognition
- There are ~ 100 billion neurons in the human brain, with something like 1,000 trillion connections between them.

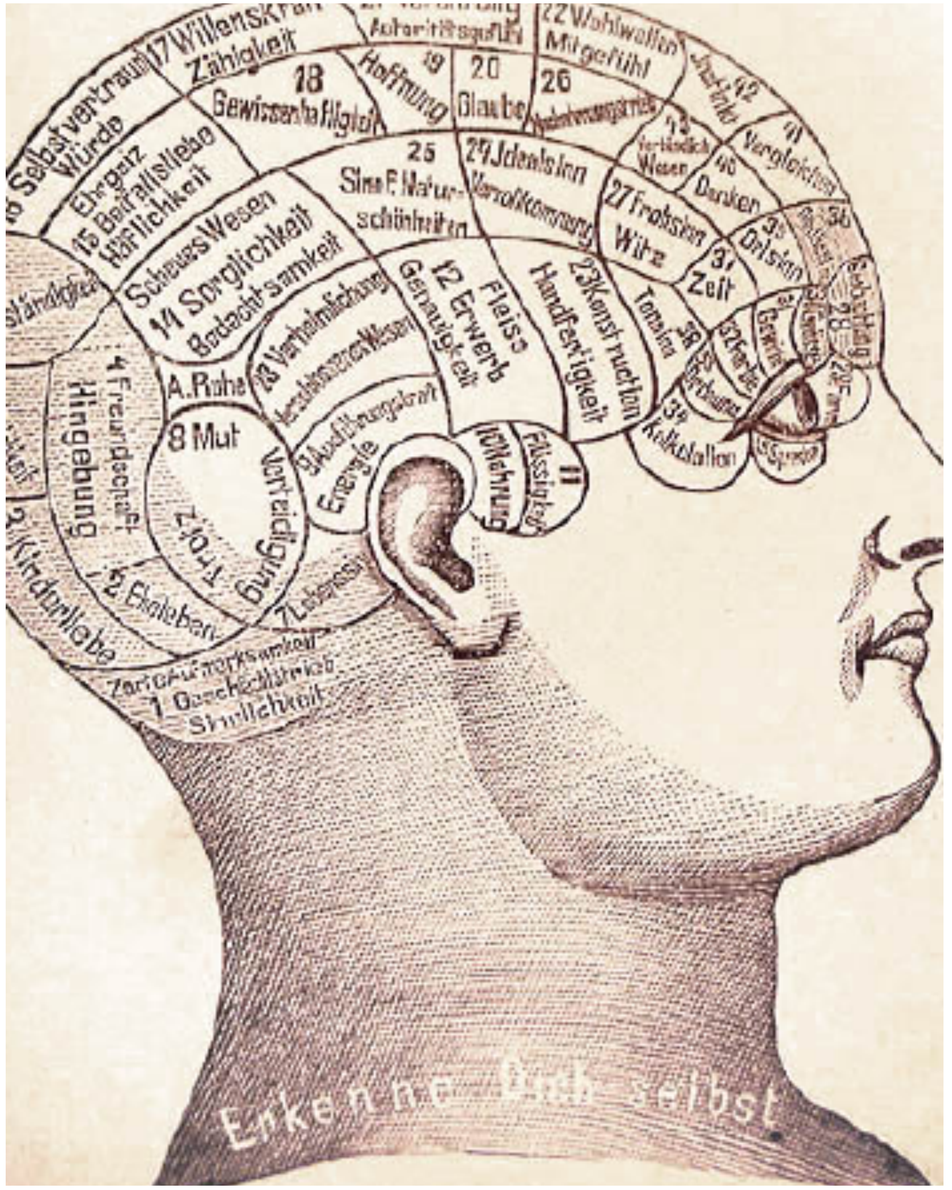


https://commons.wikimedia.org/wiki/File:Chemical_synapse_schema_cropped.jpg

Nervous system

COGS 107 in < 10 minutes

- The brain is functionally segregated (small regions of brain specialize in a function) but also integrated (functions are spread across many regions) and degenerate (more than one set of mechanisms can support a function)

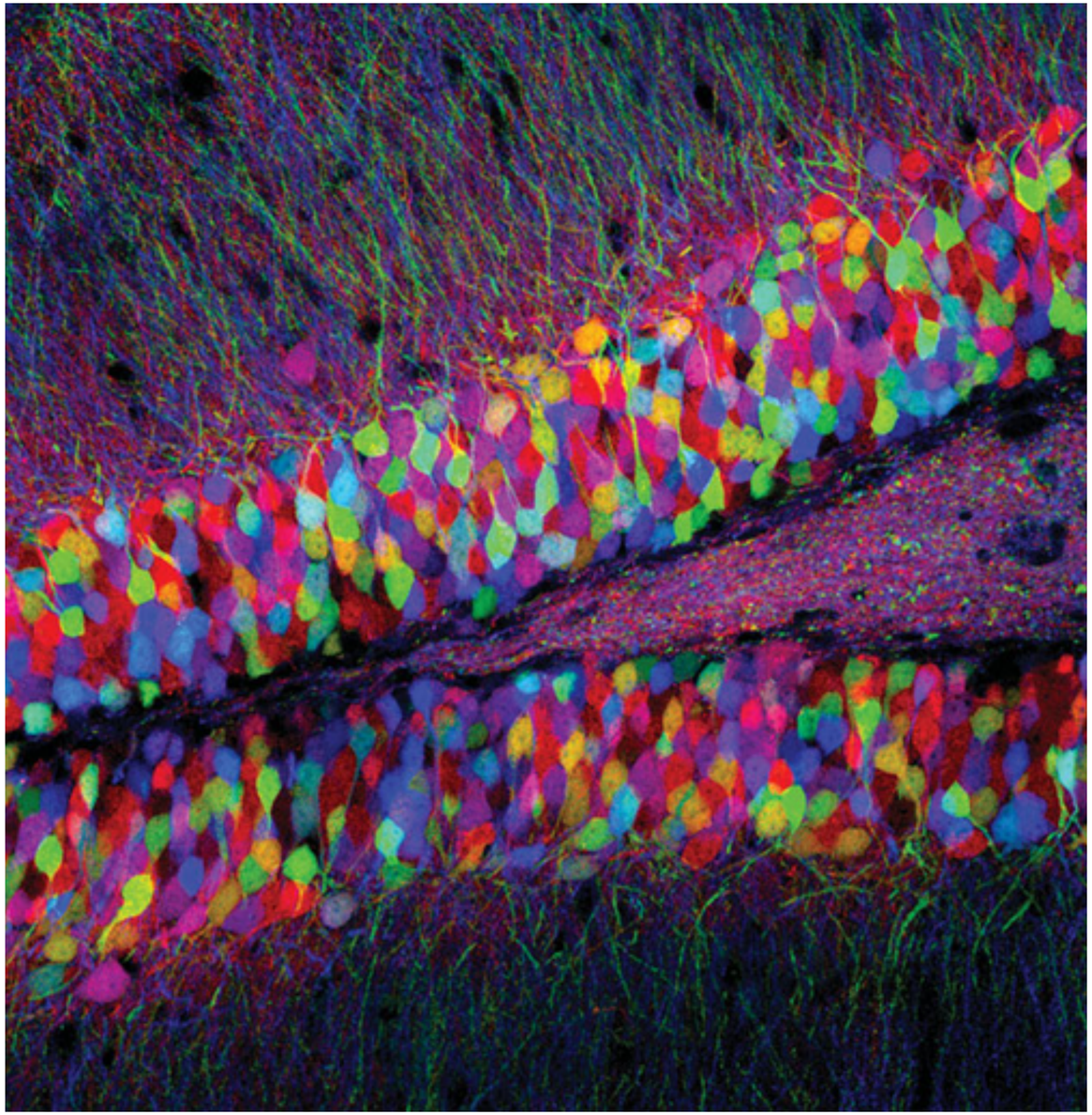


Nervous system

COGS 107 in < 10 minutes

- Circuitry is complicated and dynamic, synapses get stronger and weaker, neurons become more and less excitable. Sometimes signals are transmitted across a synapse, sometimes they aren’t.

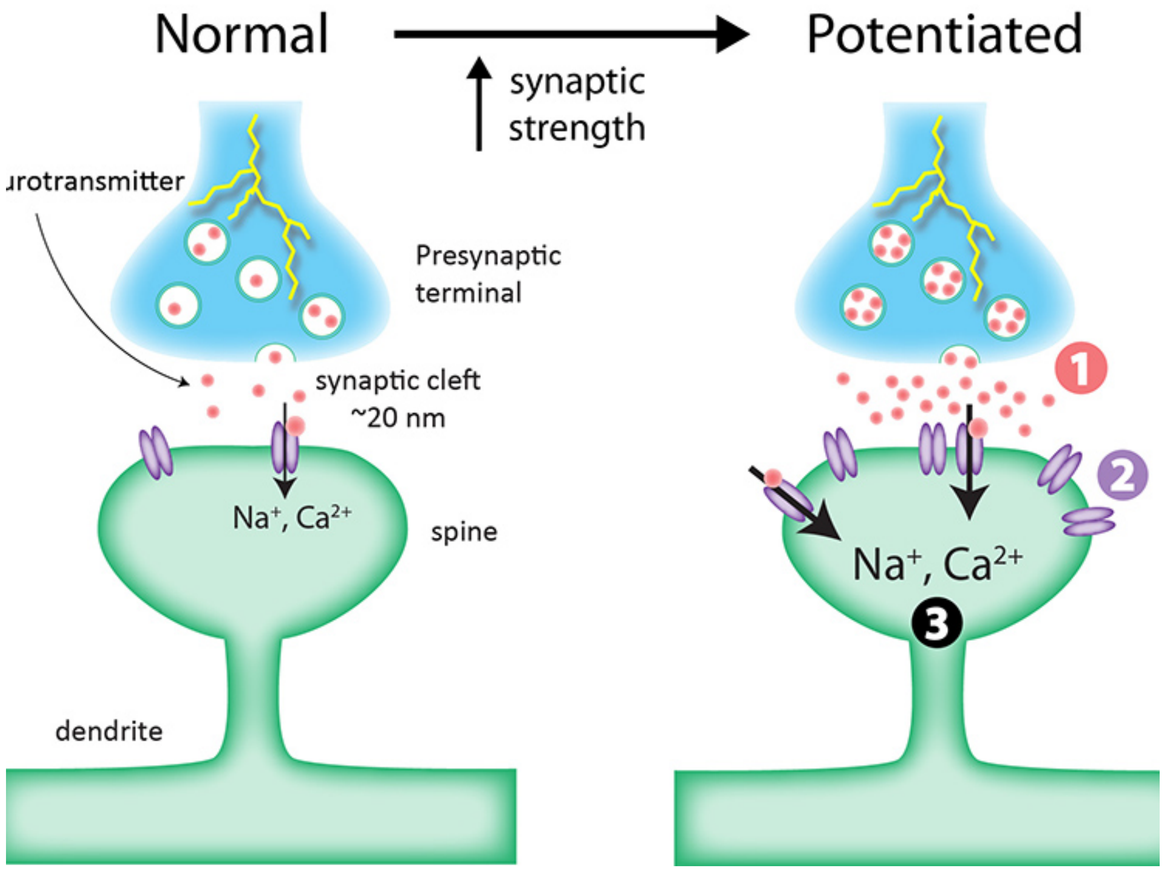
- Circuitry looks more like spaghetti than a straight path. Signals go “backwards” as well as “forwards”



Nervous system

COGS 107 in < 10 minutes

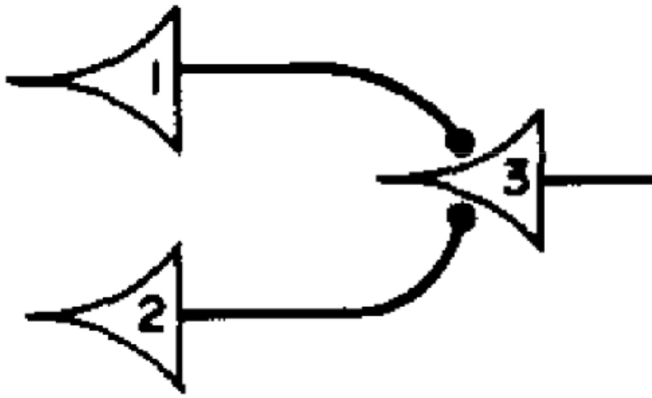
- Learning happens through both changes in synapses and changes in how a neuron responds to synaptic inputs
- There are many learning mechanisms but some important ones are
  - Each neuron wants to keep itself firing a reasonable amount
  - Fire together, wire together
  - If something is rewarding, or if it predicts reward soon, then change synapses to make it happen more frequently in the future

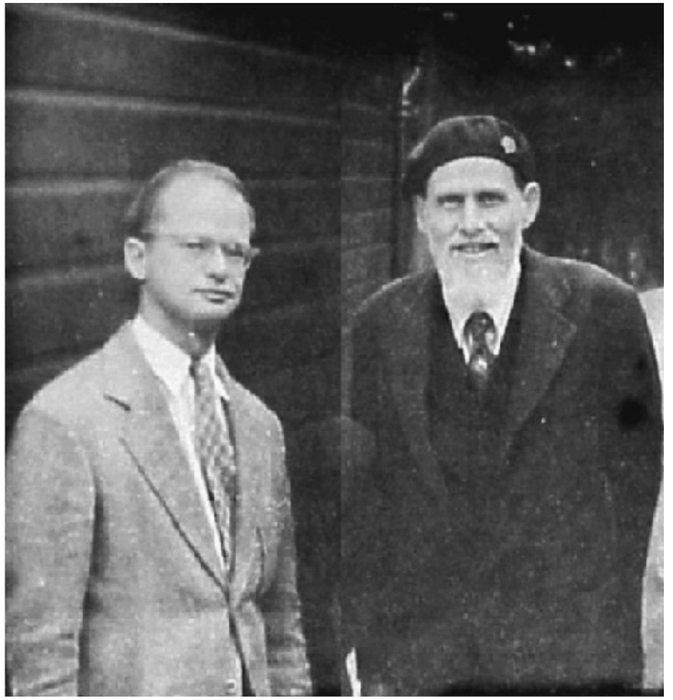


Nervous system

COGS 107 in < 10 minutes

- 1940s: the metaphor of mind shifts to computation
  - Alan Turing - “can machines think?”
  - Norbert Weiner - “how can machines be built to think?”
  - W Grey Walter - “here’s some thinking machines”
  - Warren McCulloch + Walter Pitts - “here’s how thinking happens”



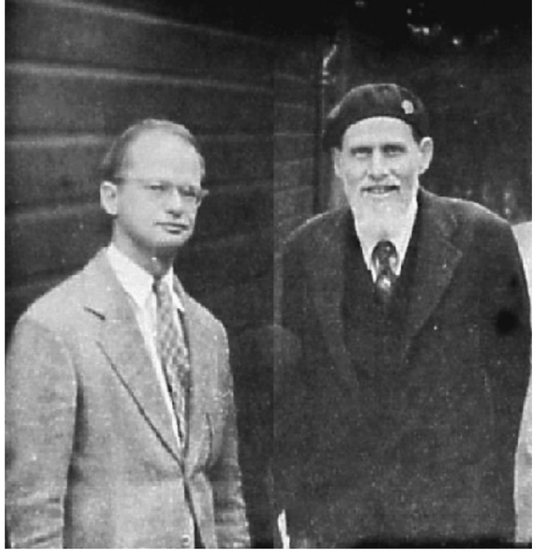


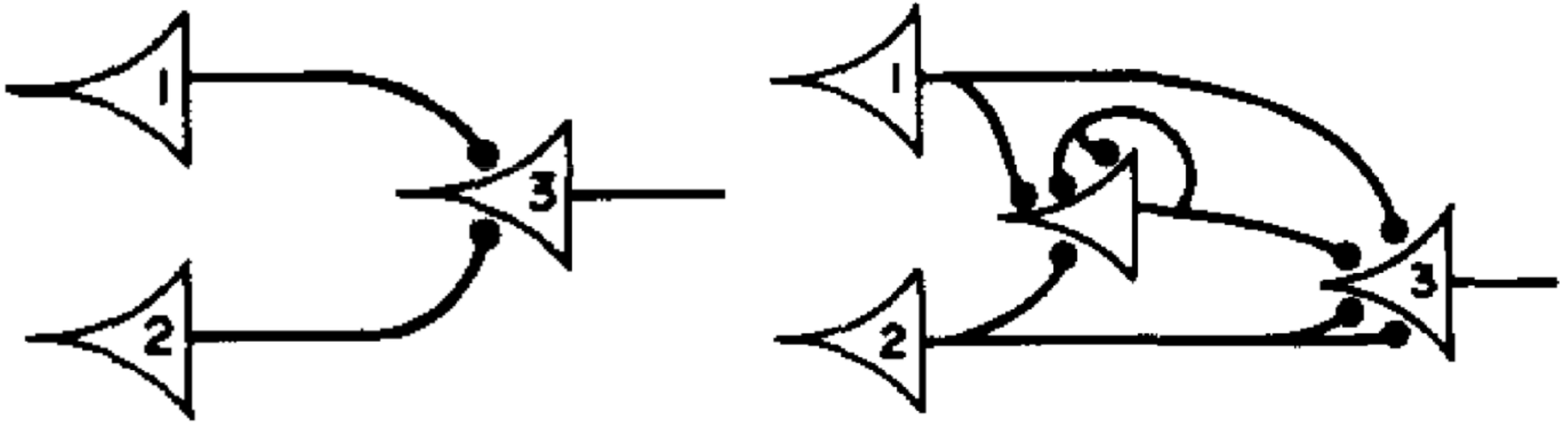
Artificial Neural Networks

A logical calculus of the ideas immanent in nervous activity

A LOGICAL CALCULUS OF THE. **IDEAS IMMANENT IN NERVOUS ACTIVITY**. WARREN S. MCCULLOCH AND WALTER PITTS. FROM THE UNIVERSITY OF ...

by WS McCulloch · 1943 · Cited by 20263 · Related articles



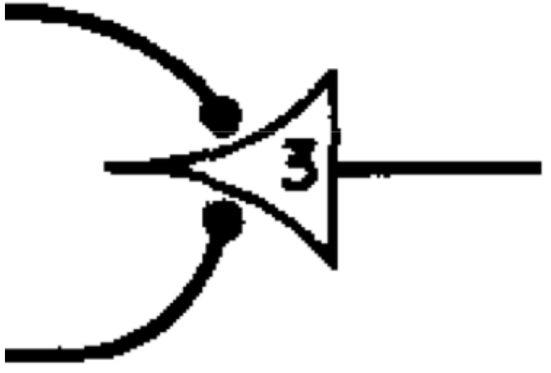


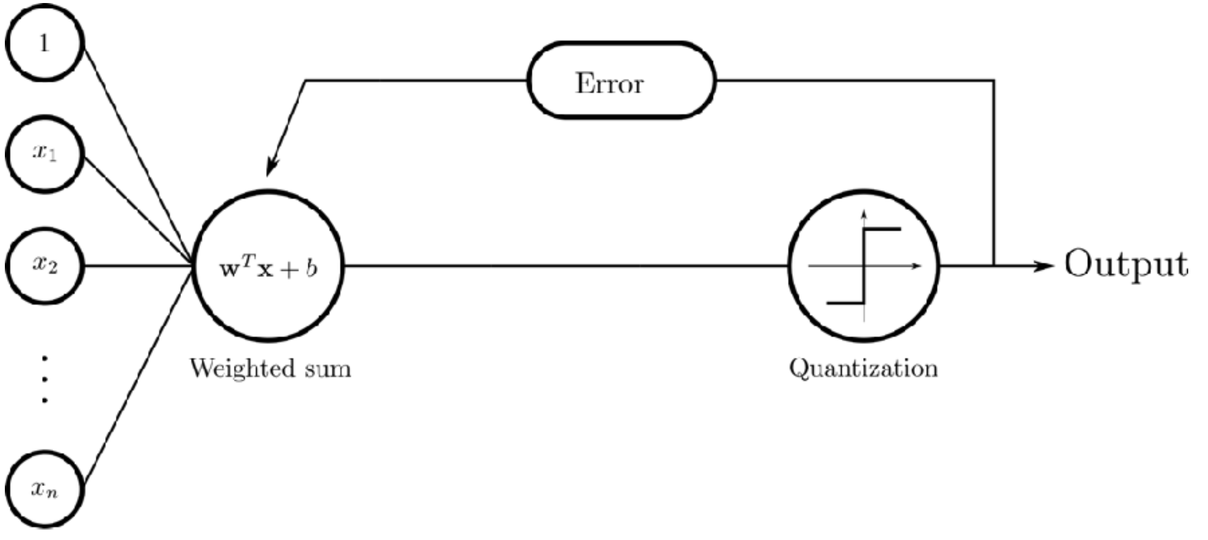
Neurons as bits
Networks as logic gates

A logical calculus of the ideas immanent in nervous activity

A LOGICAL CALCULUS OF THE. **IDEAS IMMANENT IN NERVOUS ACTIVITY**. WARREN S. MCCULLOCH AND WALTER PITTS. FROM THE UNIVERSITY OF ...

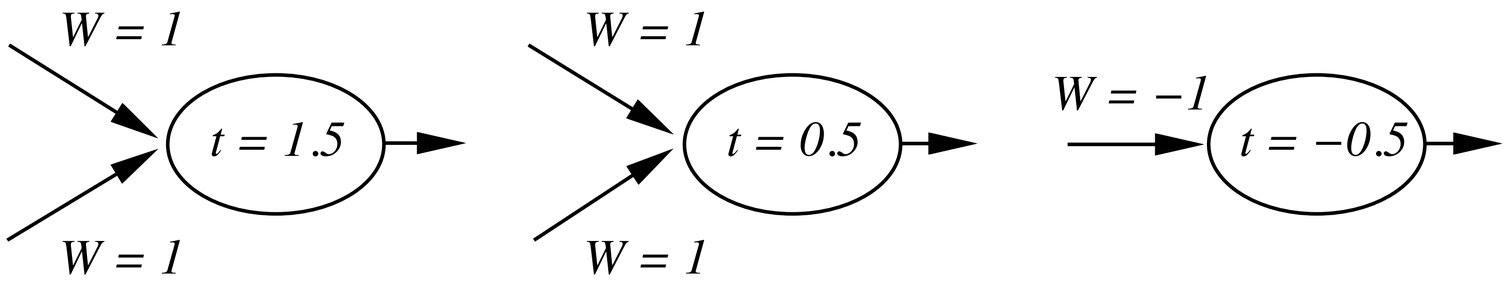
by WS McCulloch · 1943 · Cited by 20263 · Related articles





Neurons as bits
Networks as logic gates

Units as Logic Gates



Activation function: $step_t$

Since units can implement the $\wedge, \vee, \neg$ boolean operators, neural nets are **Turing-complete**: they can implement *any* computable function.

Diagram from Russell & Norvig

SYNTHETIC SAPPHIRES—

A new technique for growing cultured sapphire crystals was announced by R. A. Laudise and A. A. Ballman of the Bell Telephone Laboratories. Large sapphires, up to ¾ of an inch square by ¼ inch thick, have been grown by this technique, they report. The sapphires are made from aluminum oxide—the raw material for the red ruby, blue sapphire, purple amethyst, green emerald and yellow topaz—by subjecting it to pressures of the order of 20,000 to 50,000 pounds per square inch and temperatures of about 395 degrees Centigrade and higher.

N PAGE 2

ulp Hobby (1953-55) and Marion lemming's predecessor.

ariffs when a domestic industry nports. The Tariff Commission increase to the President, who ade act made the final decision.

horez, who is head of the French . M. de Murville is Foreign et and M. Pflimlin have the title te.

Pakistan and Turkey. Britain is of the Pact.

Cortines.

mocrats.

ruman's acceptance of the Presn at the 1948 Presidential conelphia. Mr. Truman announced speech that he was calling the lled Eightieth Congress back he 26th day of July, which out ll Turnip Day." The move was ign against the "do-nothing, nongress."

w from New York to Ireland in ne after being denied permission claimed he had meant to fly had gone the "wrong way" by

There are "inadequate and intriguing hints," he said, "that the synthesis and use of large molecules of nucleic acids (chemicals that con- [illegible] useful in the treatment of bladder cancer for the general population, when the chemical becomes generally available.

Electronic 'Brain' Teaches Itself

The Navy last week demonstrated the embryo of an electronic computer named the Perceptron which, when completed in about a year, is expected to be the first non-living mechanism able to "perceive, recognize and identify its surroundings without human training or control." Navy officers demonstrating a preliminary form of the device in Washington said they hesitated to call it a machine because it is so much like a "human being without life."

Dr. Frank Rosenblatt, research psychologist at the Cornell Aeronautical Laboratory, Inc., Buffalo, N. Y., designer of the Perceptron, conducted the demonstration. The machine, he said, would be the first electronic device to think as the human brain. Like humans, Perceptron will make mistakes at first, "but it will grow wiser as it gains experience," he said.

The first Perceptron, to cost about $100,000, will have about 1,000 electronic "association cells" receiving electrical impulses from an eyelike scanning device with 400 photocells. The human brain has ten billion responsive cells, including 100,000,000 connections with the eye.

Difference Recognized

The concept of the Perceptron was demonstrated on the Weather Bureau's $2,000,000 IBM 704 computer. In one experiment, the 704 computer was shown 100 squares situated at random either on the left or the right side of a field. In 100 trials, it was able to "say" correctly ninety-seven times whether a square was situated on the right or left. Dr. Rosenblatt said that after having seen only thirty to forty squares the device had learned to recognize the difference between right and left, almost the way a child learns.

When fully developed, the Perceptron will be designed to remember images and information it has perceived itself, whereas ordinary computers remember only what is fed into them on punch cards or magnetic tape.

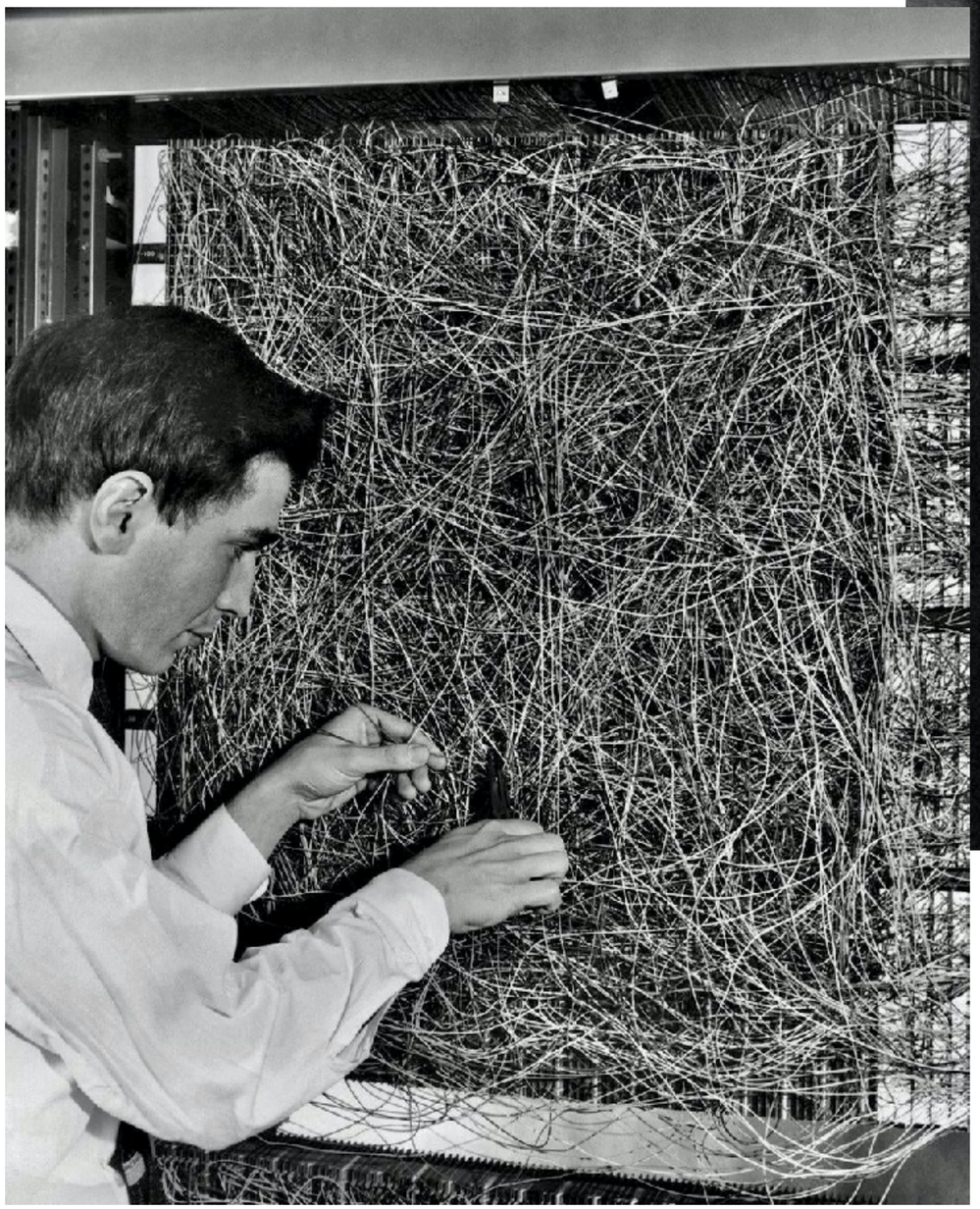
Later Perceptrons, Dr. Rosenblatt said, will be able to recognize people and call out their names. Printed pages, longhand letters and even speech commands are within its reach. Only one more step of development, a difficult step, he said, is needed for the device to hear speech in one language and instantly translate it to speech or writing in another language.

Self-Reproduction

In principle, Dr. Rosenblatt said, it would be possible to build Perceptrons that could reproduce themselves on an assembly line and which would be "conscious" of their existence.

Perceptron, it was pointed out, needs no "priming." It is not necessary to introduce it to surroundings and circumstances, record the data involved and then store them for future comparison as is the case with present "mechanical brains." It literally teaches itself to recognize objects the first time it encounters them. It uses a camera-eye lens to scan objects or survey situations, and an electrical impulse system, patterned point-by-point after the human brain does the interpreting.

The Navy said it would use the principle to build the first Perceptron "thinking machines" that will be able to read or write.



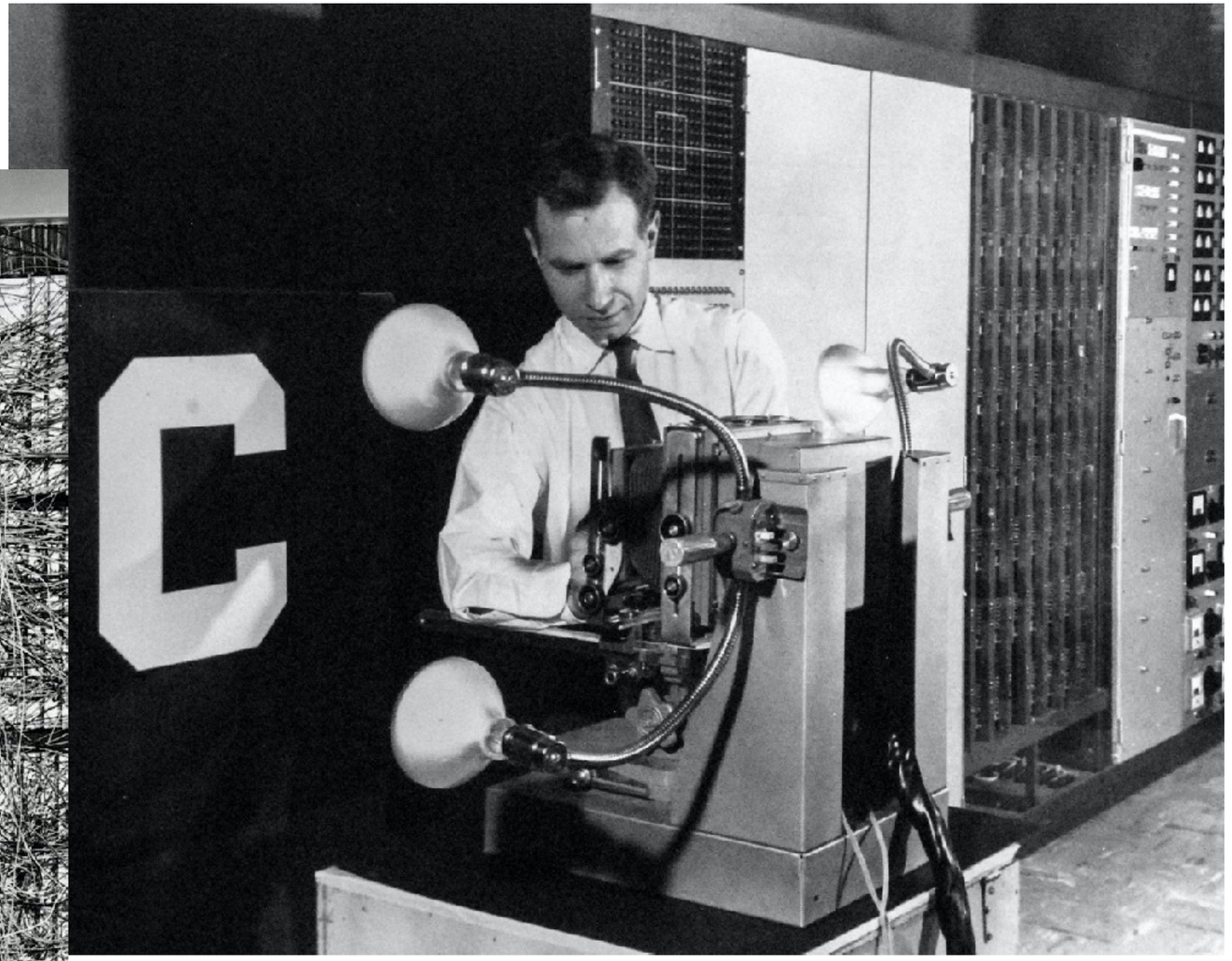

C

FIG. 1 — Organization of a biological brain. (Red areas indicate active cells, responding to the letter X.)

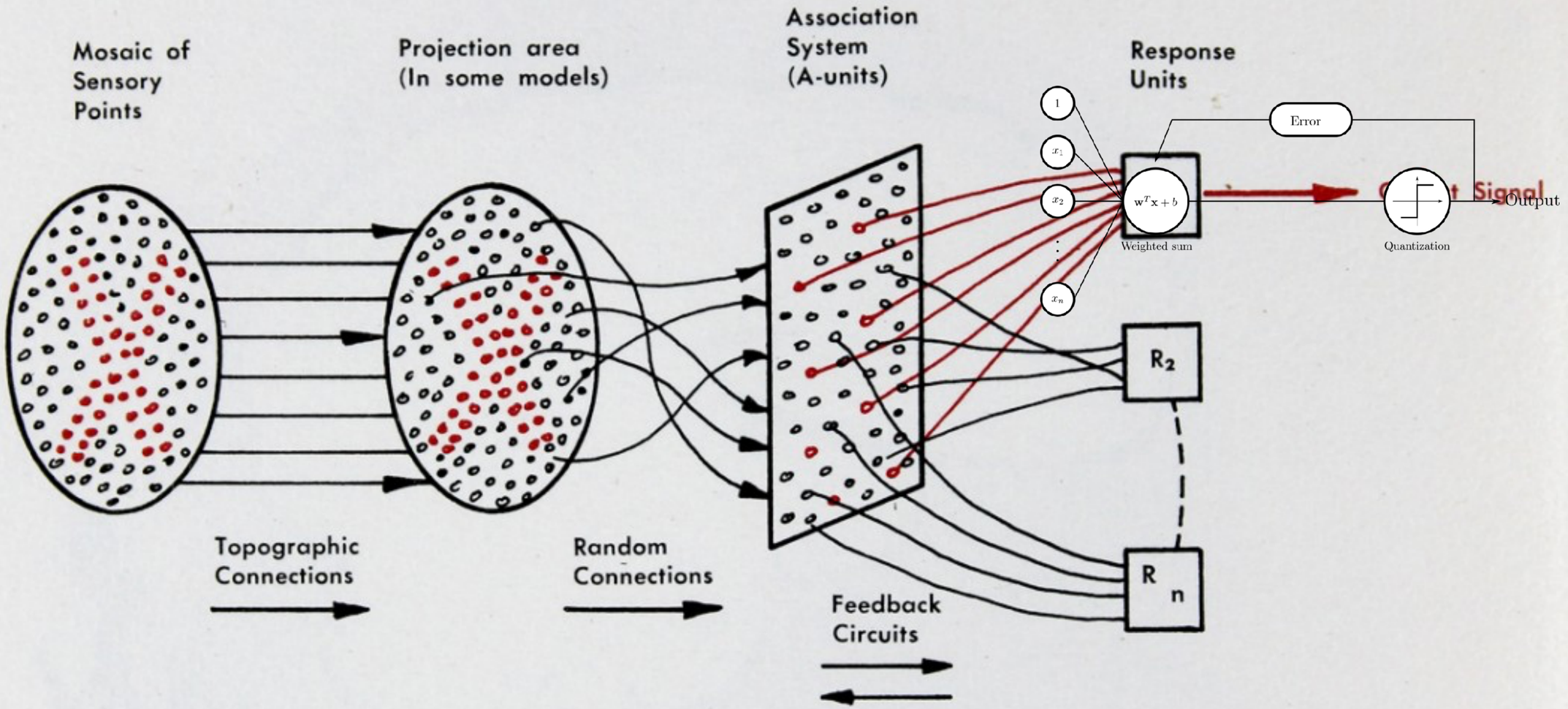
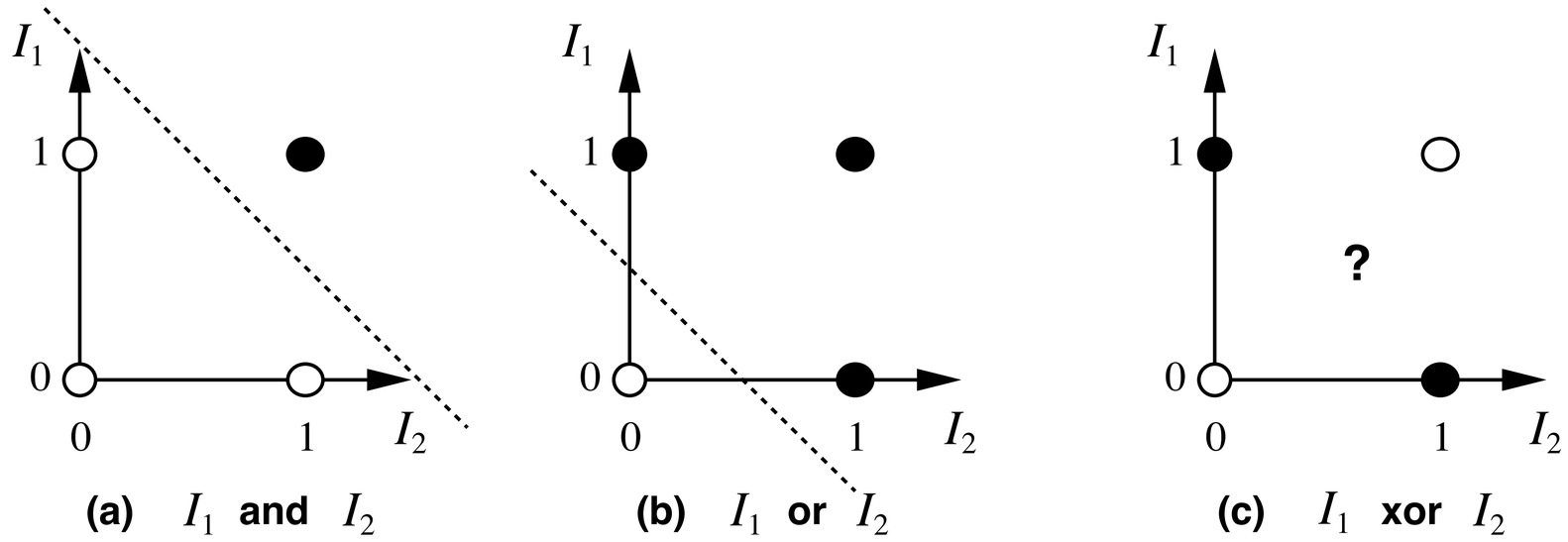


FIG. 2 — Organization of a perceptron.

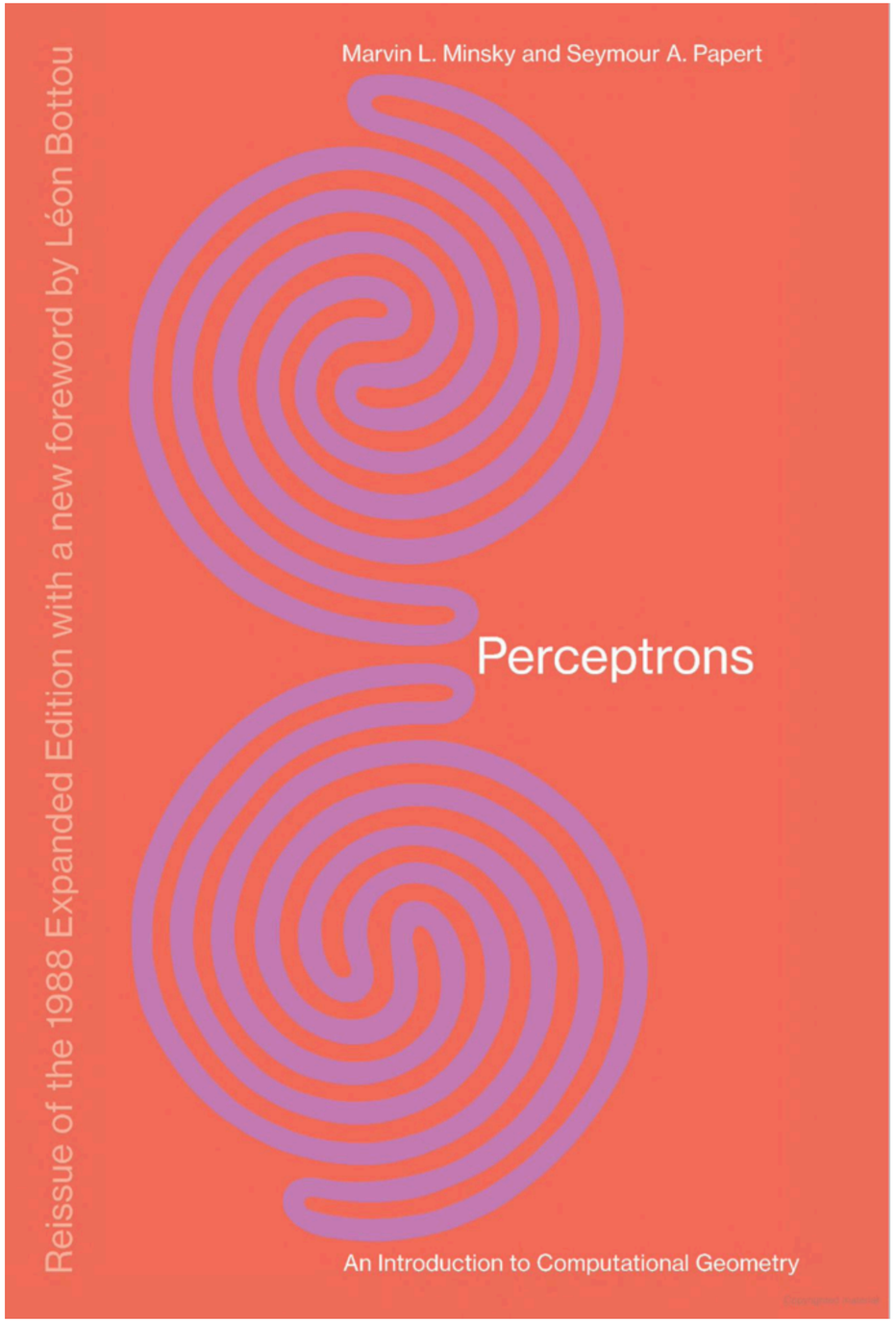
Linearly Separable Functions on a 2-dimensional Space



A black dot corresponds to an output value of 1. An empty dot corresponds to an output value of 0.

Perceptrons

1969 “kills off” NN research?

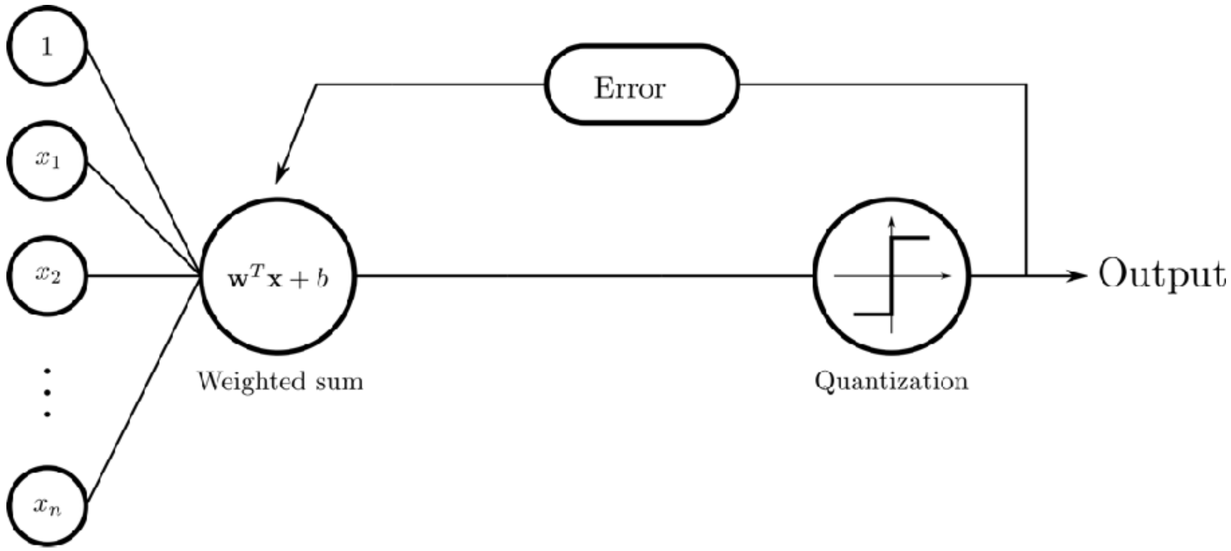


Perceptron

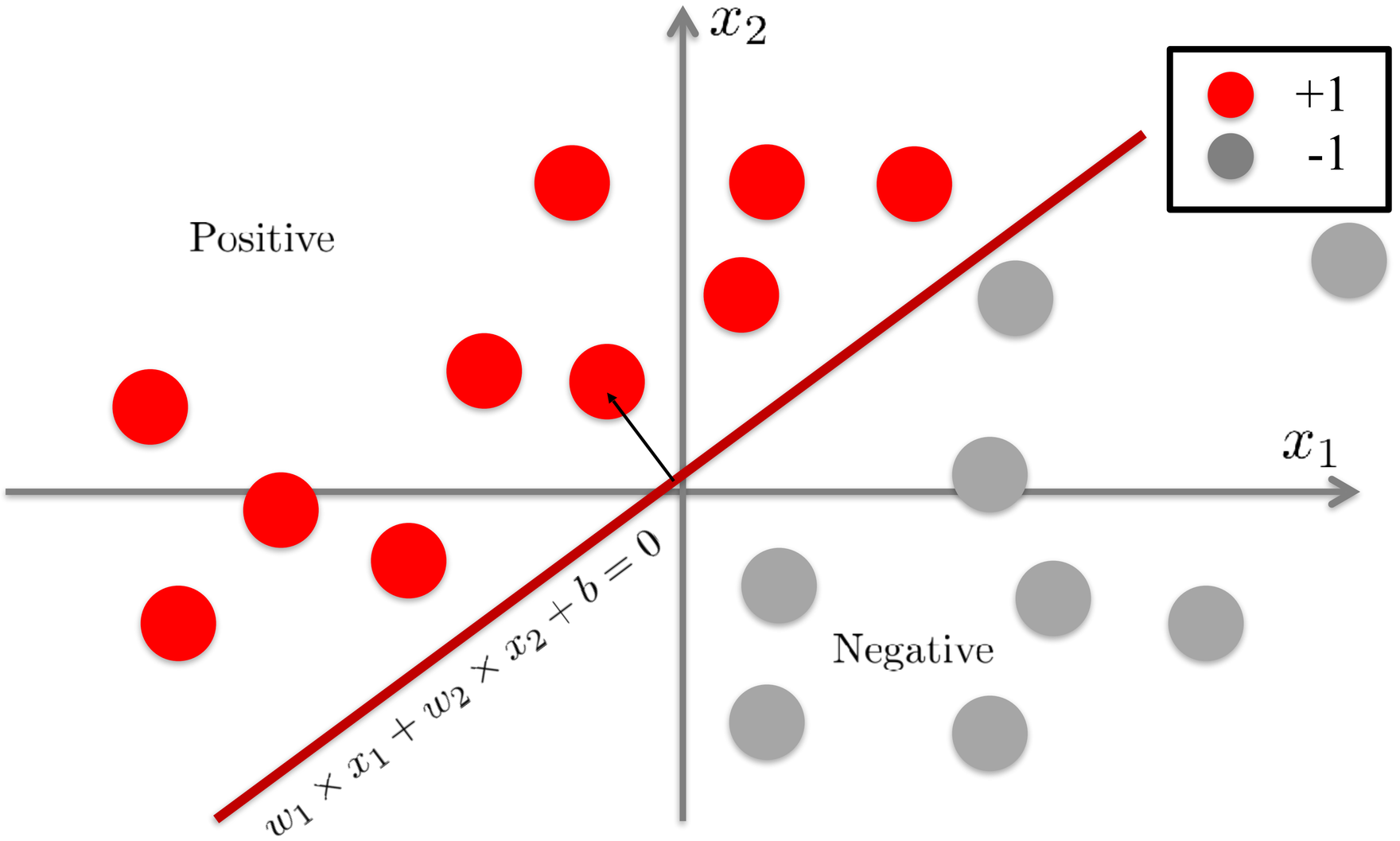
$$\mathbf{x} = (x_1, x_2, ...) \qquad \mathbf{w} = (w_1, w_2, ...) \qquad b$$

Perceptron:

$$f(\mathbf{x}|\mathbf{w}; b) = \begin{cases} +1 & if\ \mathbf{w}^T\mathbf{x} + b \geq 0 \\ -1 & otherwise \end{cases}$$



Perceptron classifier: $f(x_1, x_2; w_1, w_2, b) = sign(w_1 \times x_1 + w_2 \times x_2 + b)$



Based on Hebbian Learning

“Fire together, wire together”

- Change the weight between neurons i and j (where i is an input to j) proportional to the firing activity of those two neurons

$$\Delta w_{j,i} = \eta x_i x_j$$

- Or to put it in terms of supervised learning, where there are desired outputs y for a given neuron where i is an input

$$\Delta w_i = \eta x_i y$$

- Known to be unstable!

Perceptron Learning Algorithm

- Initialize the weights (however you choose)
- For each epoch of training:
  - For each data point in the data set:
    - Compute the model output for the data point
    - Update the model according to:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + (target_i - output_i) \times \mathbf{x}_i$$
$$b_{t+1} \leftarrow b_t + (target_i - output_i)$$

Typically we keep running epochs until the total number of errors is less than a threshold, or until a maximum number of epochs

```python
import numpy as np


class Perceptron(object):

    # assumes that no_of_inputs does not include bias!
    def __init__(self, no_of_inputs, threshold, learning_rate):

        self.threshold = threshold

        self.learning_rate = learning_rate

        self.weights = np.zeros(no_of_inputs + 1)


    # predicts for a single observation in training set
    def predict(self, inputs):

        summation = np.dot(inputs, self.weights[1:]) + self.weights[0]

        if summation > self.threshold:

          activation = 1

        else:

          activation = 0

        return activation


    # trains a training set through just once
    def train(self, training_inputs, labels):

        for inputs, label in zip(training_inputs, labels):

           prediction = self.predict(inputs)

           self.weights[1:] += self.learning_rate * (label - prediction) * inputs

           self.weights[0] += self.learning_rate * (label - prediction)
```

$$f(\mathbf{x}|\mathbf{w};b) = \begin{cases} +1 & if\ \mathbf{w}^T\mathbf{x} + b \geq 0 \\ -1 & otherwise \end{cases}$$

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + (target_i - output_i) \times \mathbf{x}_i$$
$$b_{t+1} \leftarrow b_t + (target_i - output_i)$$

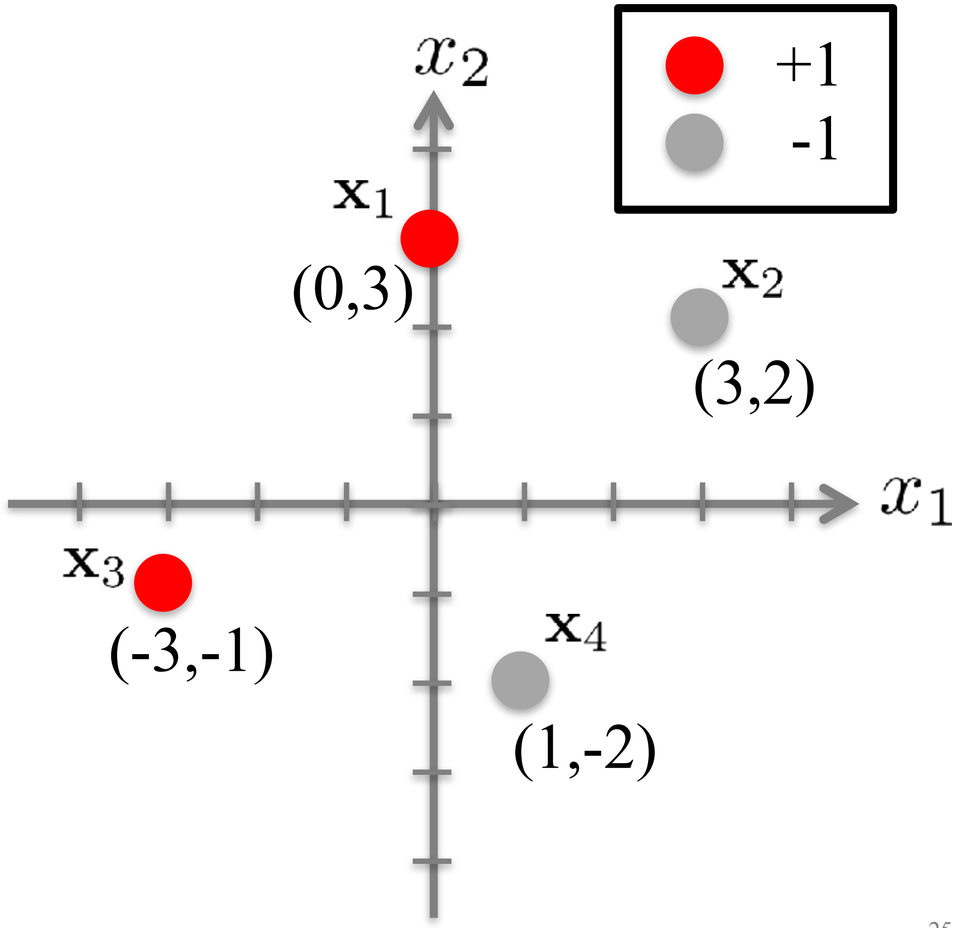
When does it work?

- Guarantee it will converge for linearly separable problems.
- Not necessarily a “good” separator, c.f. SVM
- Linear separator is not well-defined because reorderings of the same input lead to different outputs

How does it work?

Given a training dataset:

$$S = \{(\mathbf{x}_1 = (0, 3), y_1 = +1),$$
$$(\mathbf{x}_2 = (3, 2), y_2 = -1),$$
$$(\mathbf{x}_3 = (-3, -1), y_3 = +1),$$
$$(\mathbf{x}_4 = (1, -2), y_4 = -1)\}$$



Initialize the weights

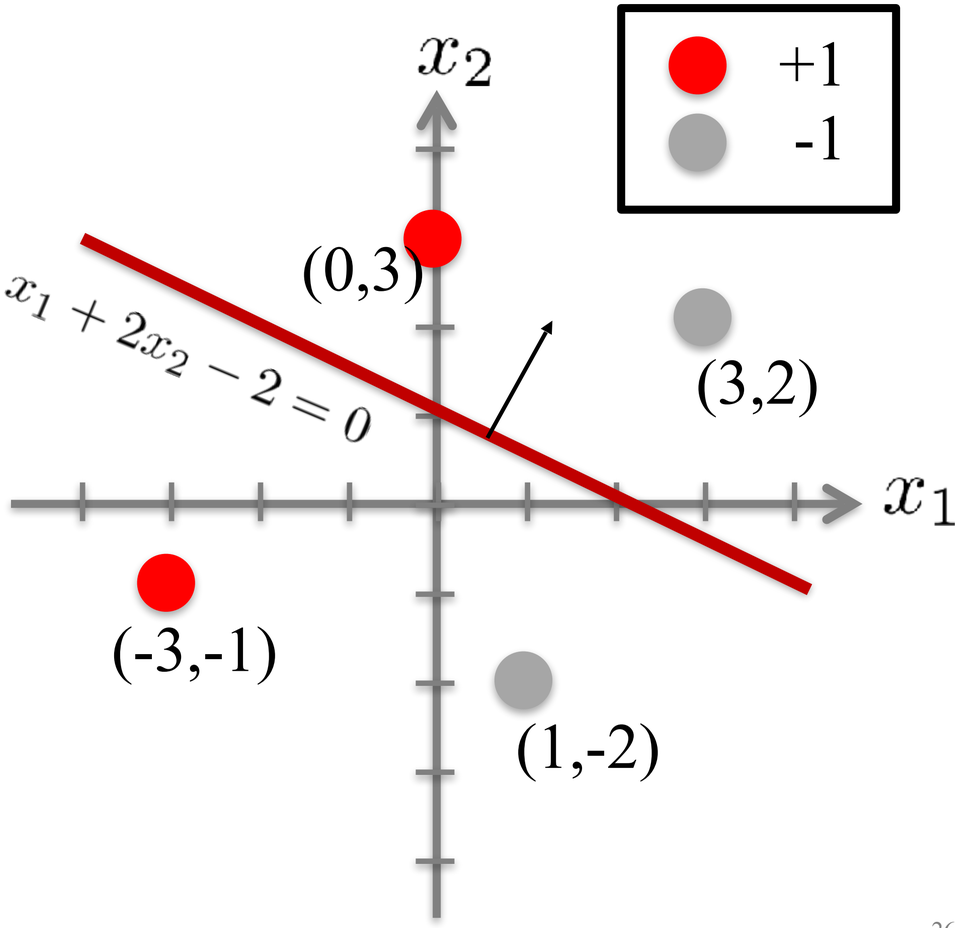
- Choose randomly – but start the weights small!

- We'll choose:

Decision boundary:
$w_1 x_1 + w_2 x_2 + b = 0$

$w_1 = 1 \quad w_2 = 2 \quad b = -2$

$$x_1 + 2x_2 - 2 = 0$$

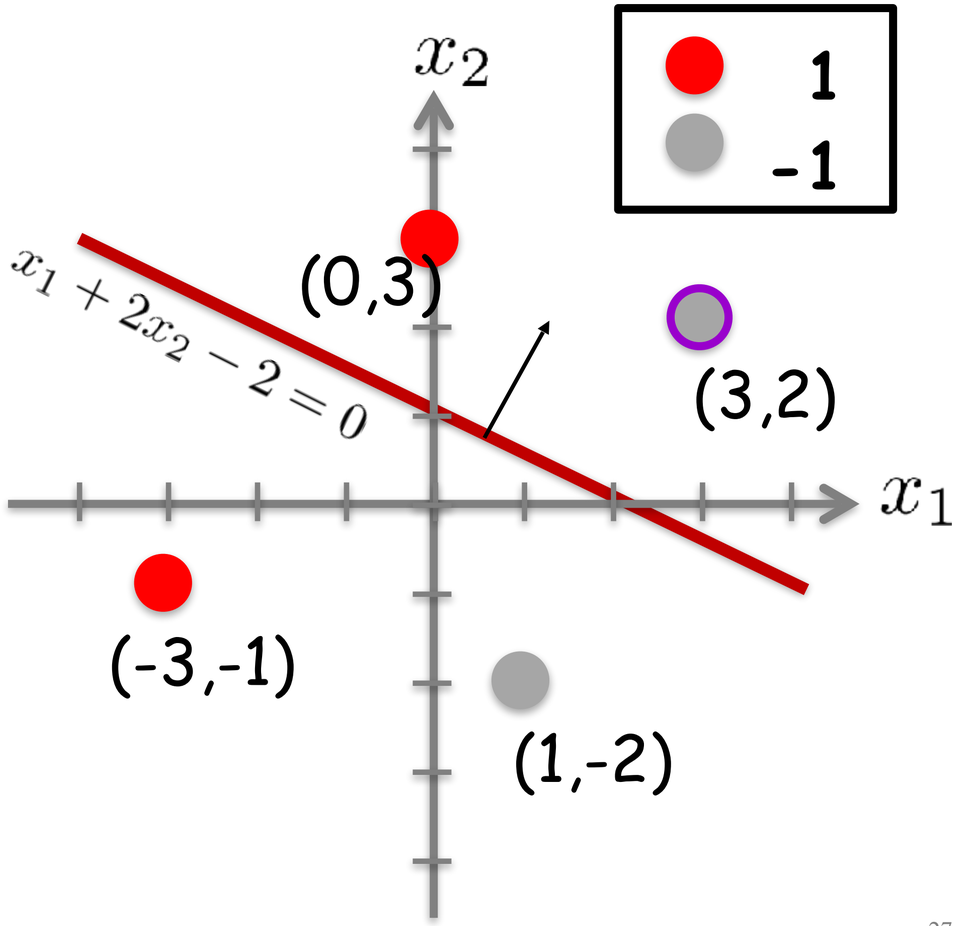


Step 1: Choose a point

- Choose randomly (or sequentially, either can work)

Say: $(\mathbf{x}_2 = (3, 2), y_2 = -1)$

It's ground-truth label (target) $= -1$: a negative sample.



Step 1: Choose a point

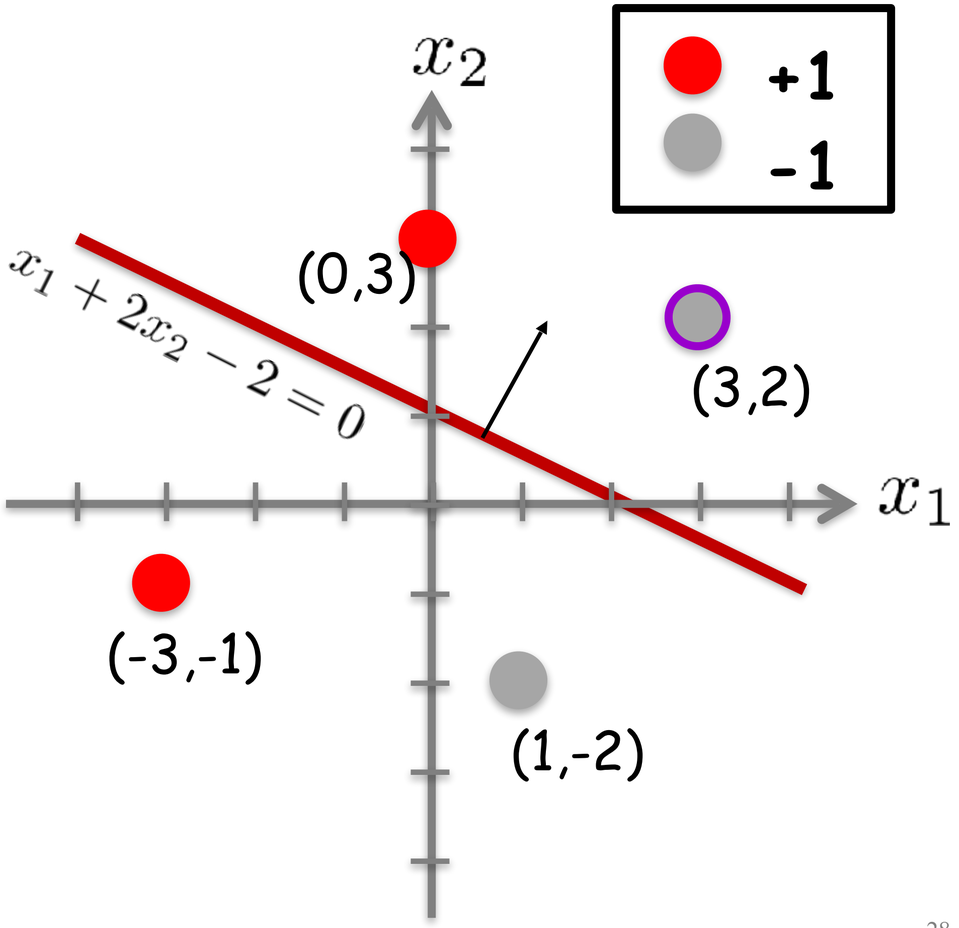
Say: $(\mathbf{x}_2 = (3, 2), y_2 = -1)$

$w_1 = 1 \quad w_2 = 2 \quad b = -2$

It's ground-truth label (target) $= -1$: a negative sample.

Perceptron:

$$f(\mathbf{x}|w_1, w_2, b) = \begin{cases} +1 & if\ x_1 + 2x_2 - 2 \geq 0 \\ -1 & otherwise \end{cases}$$



Step 1: Choose a point at random

Say: $(\mathbf{x}_2 = (3,2), y_2 = -1)$

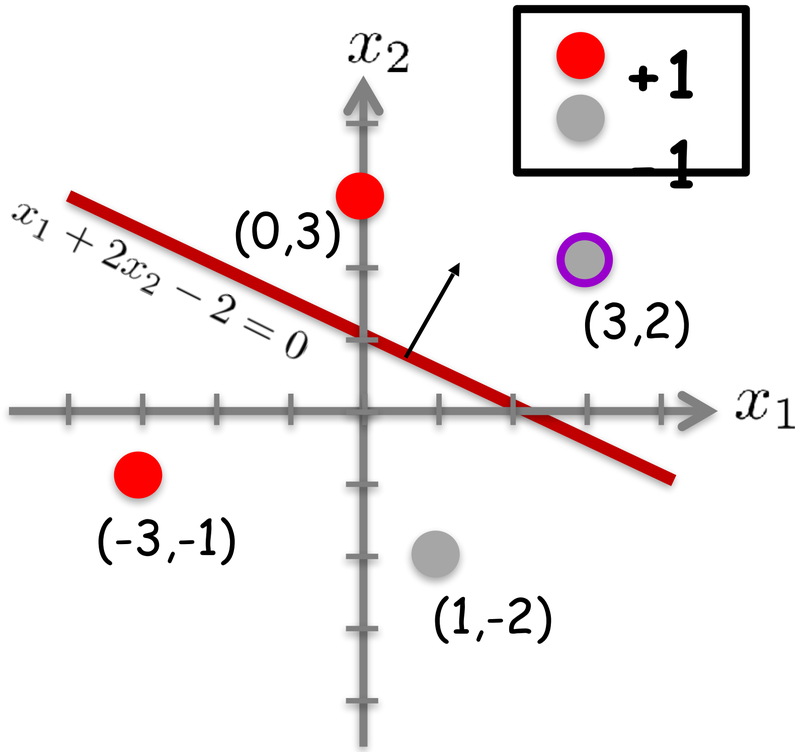
$w_1 = 1 \quad w_2 = 2 \quad b = -2$

Perceptron:

$$f(\mathbf{x}|w_1, w_2, b) = \begin{cases} +1 & if\ x_1 + 2x_2 - 2 \geq 0 \\ -1 & otherwise \end{cases}$$

We plug in $\mathbf{x}_2 = (3,2)$:

$3 + 2 \times 2 - 2 = 3 + 4 - 2 = 5$



Classification:

$$f(\mathbf{x}_2 = (3,2)|w_1 = 1, w_2 = 2, b = -2) = sign(5) = +1$$

$+1 \neq -1$

$y_2 \neq f(\mathbf{x}_2 = (3,2)|w_1 = 1, w_2 = 2, b = -2)$

We need to change (**w**,b)!

Step 1: Update

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + (target_i - output_i) \times \mathbf{x}_i$$
$$b_{t+1} \leftarrow b_t + (target_i - output_i)$$

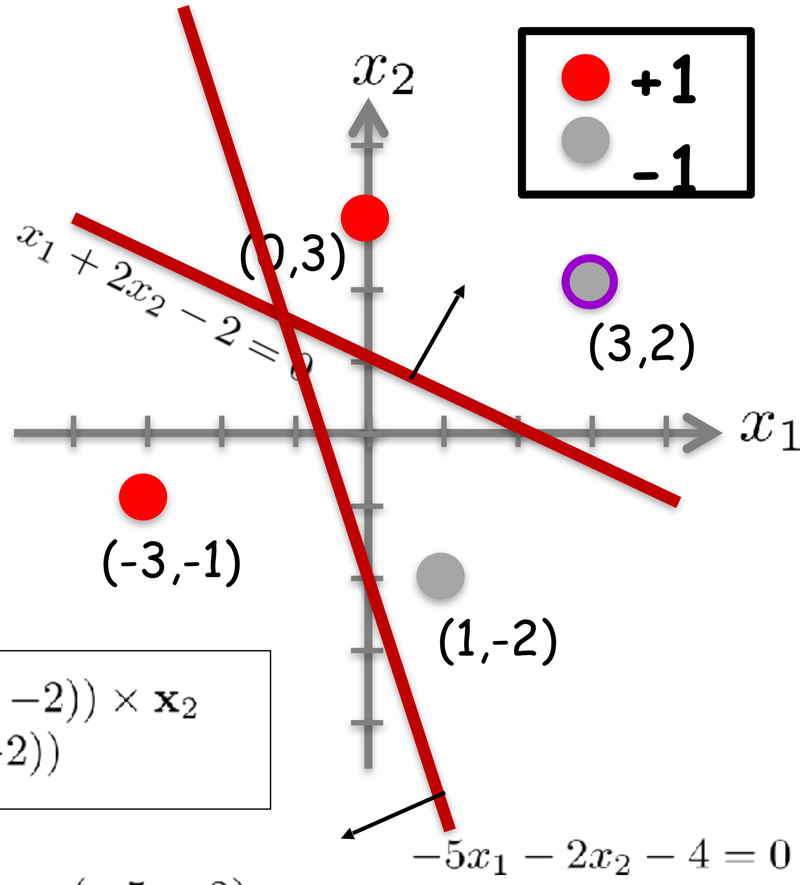
Say: $(\mathbf{x}_2 = (3,2), y_2 = -1)$

$w_1 = 1 \quad w_2 = 2 \quad b = -2$

Perceptron:

$$f(\mathbf{x}|w_1, w_2, b) = \begin{cases} +1 & if\ x_1 + 2x_2 - 2 \geq 0 \\ -1 & otherwise \end{cases}$$

$y_2 \neq f(\mathbf{x}_2 = (3,2)|w_1 = 1, w_2 = 2, b = -2)$



Updating rule:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + (y_2 - f(\mathbf{x}_2 = (3,2)|w_1 = 1, w_2 = 2, b = -2)) \times \mathbf{x}_2$$
$$b_{t+1} \leftarrow b_t + (y_2 - f(\mathbf{x}_2 = (3,2)|w_1 = 1, w_2 = 2, b = -2))$$

$$\mathbf{w}_{t+1} \leftarrow (1,2) + (-1-1) \times (3,2)$$
$$b_{t+1} \leftarrow -2 + (-1-1)$$

$$\mathbf{w}_{t+1} = (-5,-2)$$
$$b_{t+1} = -4$$

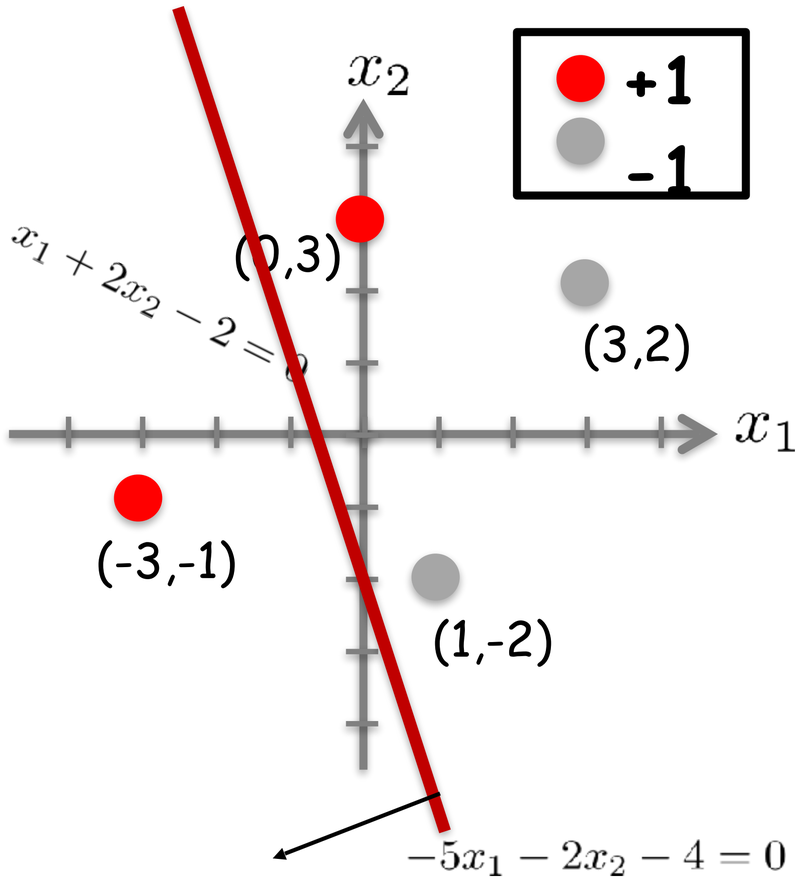
Step 1: Update

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + (target_i - output_i) \times \mathbf{x}_i$$
$$b_{t+1} \leftarrow b_t + (target_i - output_i)$$

$$w_1 = -5 \quad w_2 = -2 \quad b = -4$$

Updated Perceptron:

$$f(\mathbf{x}|w_1, w_2, b) = \begin{cases} +1 & if \ -5x_1 - 2x_2 - 4 \geq 0 \\ -1 & otherwise \end{cases}$$

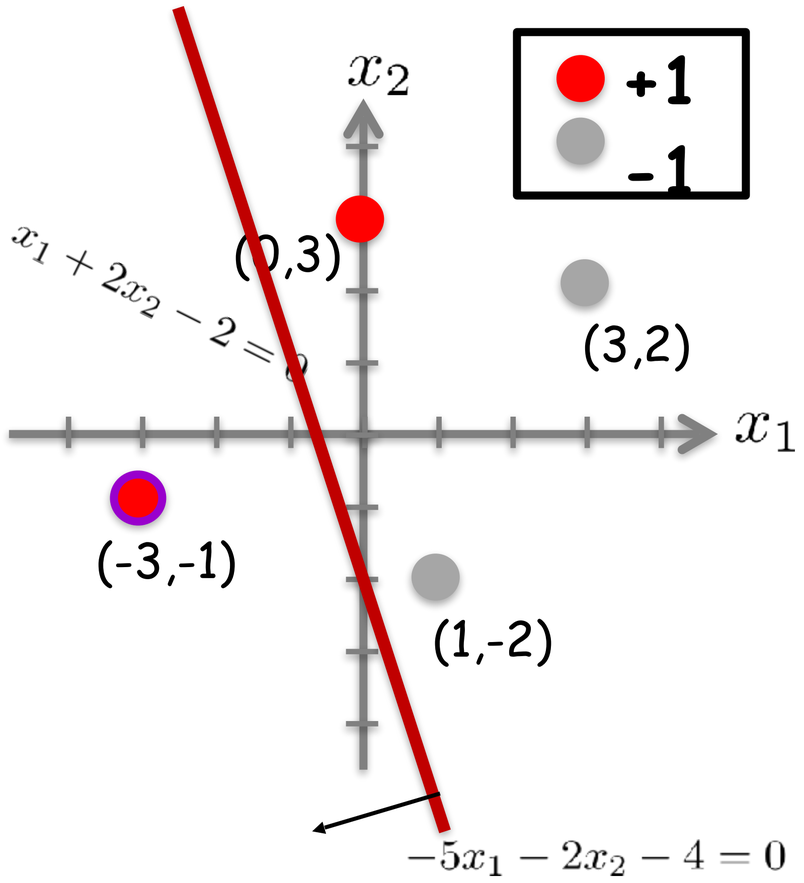


Step 2: Choose another point

$$w_1 = -5 \quad w_2 = -2 \quad b = -4$$

Perceptron:

$$f(\mathbf{x}|w_1, w_2, b) = \begin{cases} +1 & if \ -5x_1 - 2x_2 - 4 \geq 0 \\ -1 & otherwise \end{cases}$$



Step 2: Choose another point

Say: $(\mathbf{x}_3 = (-3, -1), y_3 = +1)$

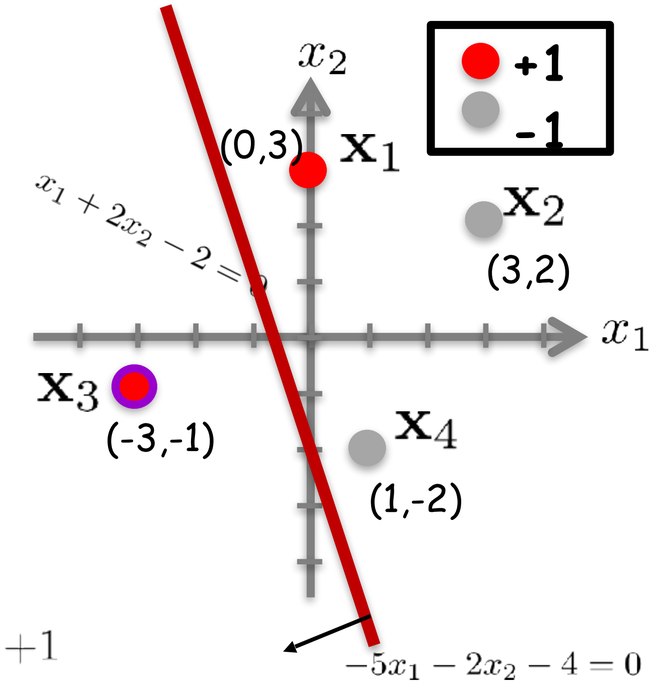
$w_1 = -5 \quad w_2 = -2 \quad b = -4$

Perceptron:

$$f(\mathbf{x}|w_1, w_2, b) = \begin{cases} +1 & if \ -5x_1 - 2x_2 - 4 \geq 0 \\ -1 & otherwise \end{cases}$$

We plug in $\mathbf{x}_3 = (-3, -1)$:

$-5 \times (-3) + (-2) \times (-1) - 4 = 15 + 2 - 4 = 13$



Classification:

$$f(\mathbf{x}_2 = (-3, -1)|w_1 = -5, w_2 = -2, b = -4) = sign(13) = +1$$

$+1 = +1$

$y_3 = f(\mathbf{x}_3 = (-3, -1)|w_1 = -5, w_2 = -2, b = -4)$

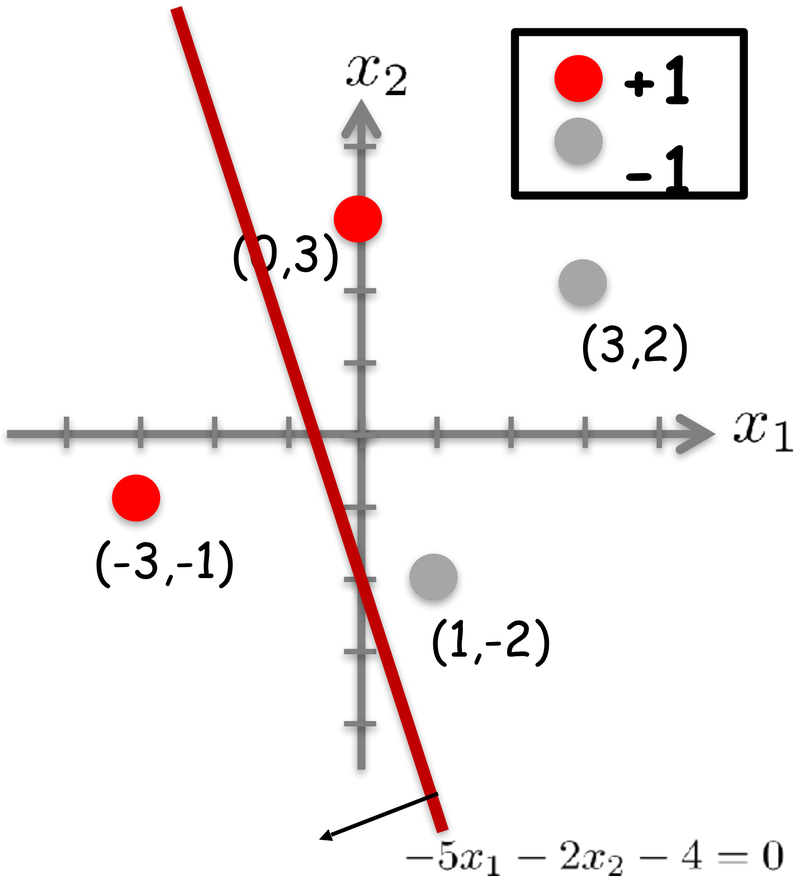
We don't need to change (**w**,b)!

Step 2: Keep

$$w_1 = -5 \quad w_2 = -2 \quad b = -4$$

Updated Perceptron:

$$f(\mathbf{x}|w_1, w_2, b) = \begin{cases} +1 & if \ -5x_1 - 2x_2 - 4 \geq 0 \\ -1 & otherwise \end{cases}$$



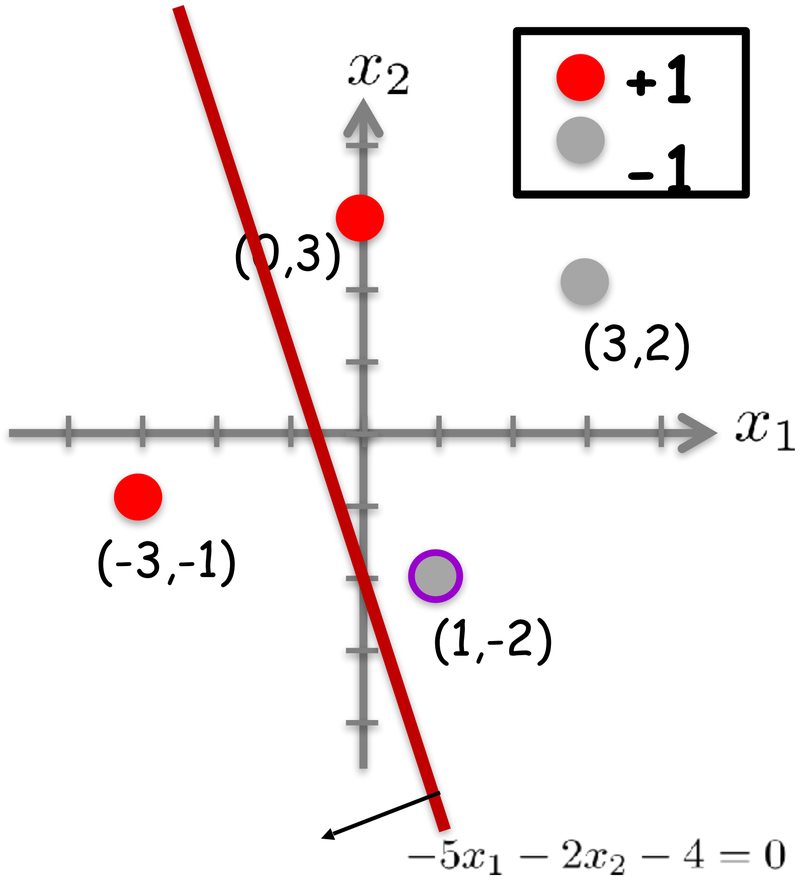
Step 3: Choose another point

Say: $(\mathbf{x}_4 = (1, -2), y_4 = -1)$

$w_1 = -5 \quad w_2 = -2 \quad b = -4$

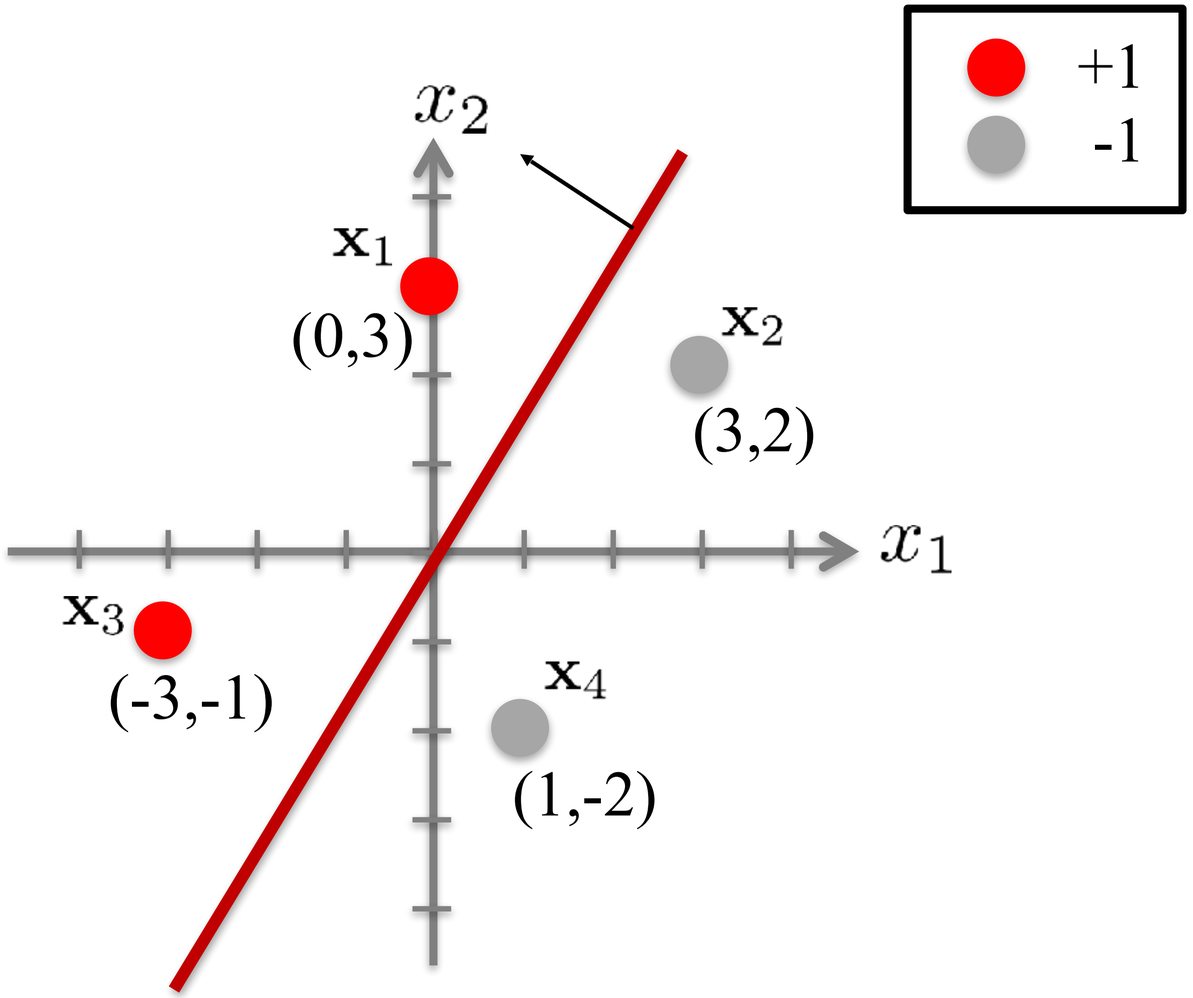
Updated Perceptron:

$$f(\mathbf{x}|w_1, w_2, b) = \begin{cases} +1 & if \ -5x_1 - 2x_2 - 4 \geq 0 \\ -1 & otherwise \end{cases}$$



Iterate Until Converge

. . . . . .



Widrow-Hoff
aka ‘delta’ learning rule
aka Rescorla-Wagner rule for Pavlovian conditioning
aka generic gradient descent version of perceptron

Make a mistake on a single data point?

Move the weights part of the way towards generating the correct answer via GD.

Let us assume:

- η is the learning rate (some positive constant)
- y is the output of the model
- o is the target (desired) output

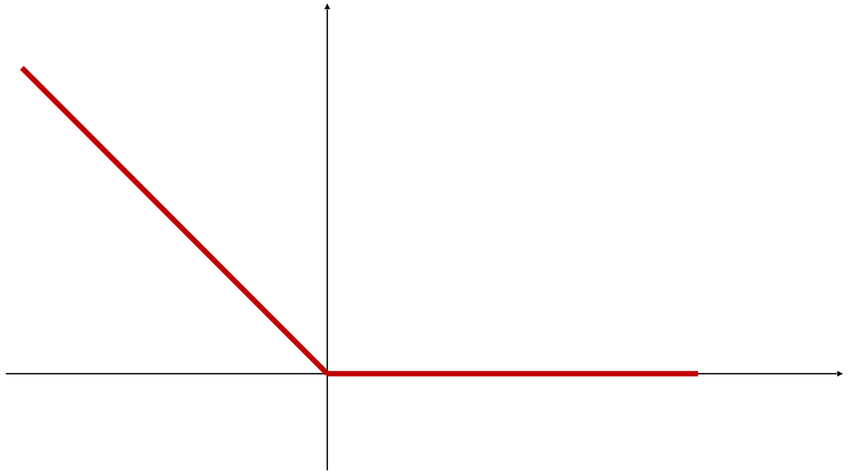
then the weights are updated as follows $w \leftarrow w + \eta(o - y)x$. The ADALINE converges to the least squares error which is $E = (o - y)^2$.[6] This update rule is in fact the stochastic gradient descent update for linear regression.[7]

Perceptron training by gradient descent (general case)

Perceptron: $f(\mathbf{x};\mathbf{w}) = sign(\mathbf{w}^T\mathbf{x} + b)$

$S = \{(\mathbf{x}_i, y_i), i = 1..n\} \quad y_i \in -1, +1$

Training: Minimize $\mathcal{L}(\mathbf{w}, b) = \frac{1}{2}(y_i - sign(\mathbf{w}^T\mathbf{x} + b))$



$$\frac{\mathcal{L}(\mathbf{w},\mathbf{b})}{\partial \mathbf{w}} = \begin{cases} 0 & if\ y_i = sign(\mathbf{w}^T\mathbf{x}_i + b) \\ -y_i\mathbf{x}_i & otherwise \end{cases}$$

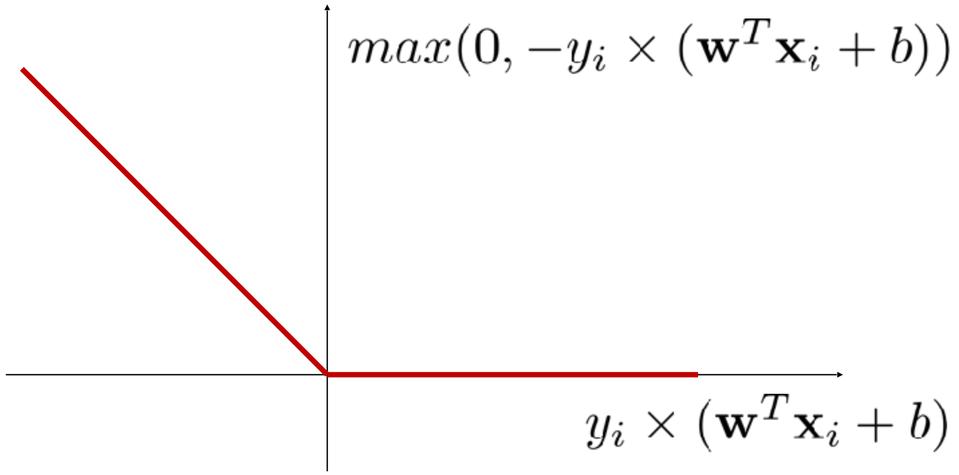
Perceptron training by gradient descent for binary classification

Perceptron: $f(\mathbf{x};\mathbf{w}) = sign(\mathbf{w}^T\mathbf{x} + b)$

$S = \{(\mathbf{x}_i, y_i), i = 1..n\} \quad y_i \in -1, +1$

no penalty if y_i and $(\mathbf{w}^T\mathbf{x}_i + b)$ have the same sign

Training: Minimize $\mathcal{L}(\mathbf{w}, b) = \sum_i max(0, -y_i \times (\mathbf{w}^T\mathbf{x}_i + b))$

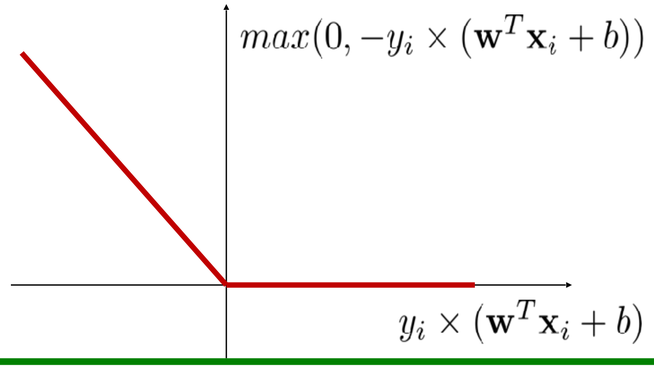


$$\frac{\mathcal{L}(\mathbf{w},\mathbf{b})}{\partial \mathbf{w}} = \begin{cases} 0 & if\ y_i = sign(\mathbf{w}^T\mathbf{x}_i + b) \\ -y_i\mathbf{x}_i & otherwise \end{cases}$$

Swapping loss functions

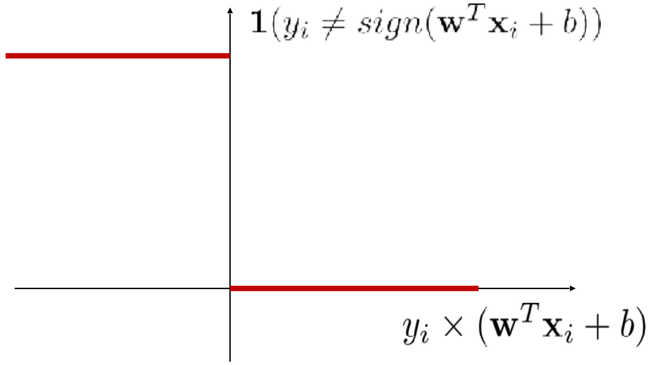
Hinge loss: gradient feedback when the target (ground-truth label) and the output (classification) are different).

Training: Minimize $\mathcal{L}(\mathbf{w}, b) = \sum_i max(0, -y_i \times (\mathbf{w}^T \mathbf{x}_i + b))$



Perceptron's 0/1 loss has the gradient always 0 except where infinity => no gradient feedback

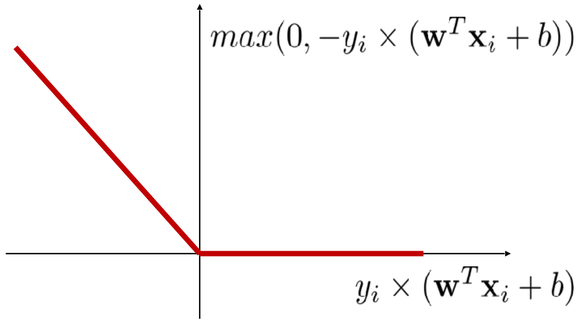
Training: Minimize $\mathcal{L}(\mathbf{w}, b) = \sum_i \mathbf{1}(y_i \neq sign(\mathbf{w}^T \mathbf{x}_i + b))$

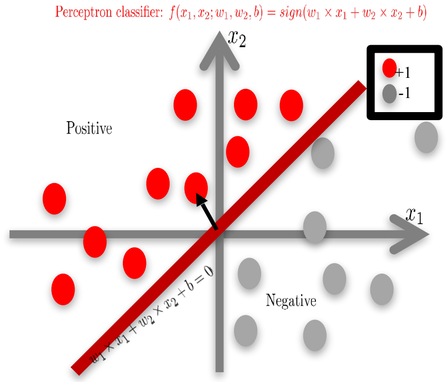


Swapping loss functions

Hinge loss: gradient feedback when the target (ground-truth label) and the output (classification) are different).

Training: Minimize $\mathcal{L}(\mathbf{w}, b) = \sum_i max(0, -y_i \times (\mathbf{w}^T \mathbf{x}_i + b))$





- Perceptron is a linear classifier.
- It replaces the 0/1 loss by a relaxed loss.
- It updates the model parameters (**w**,b) based on a single sample, whereas standard gradient descent algorithm computes the gradient by taking ALL the training samples into account.

Perceptron training is a special gradient descent algorithm

Perceptron: $f(\mathbf{x};\mathbf{w}) = sign(\mathbf{w}^T\mathbf{x} + b)$

$S = \{(\mathbf{x}_i, y_i), i = 1..n\}$

Training: Minimize $\mathcal{L}(\mathbf{w}, b) = \sum_i max(0, -y_i \times (\mathbf{w}^T\mathbf{x}_i + b))$

no penalty if y_i and $(\mathbf{w}^T\mathbf{x}_i + b)$ have the same sign

$$\frac{\partial \mathcal{L}(\mathbf{w},b)}{\partial \mathbf{w}} = \sum_i -\frac{1}{2}(target_i - output_i)\mathbf{x}_i$$

$target_i = y_i$

$output_i = sign(\mathbf{w}^T\mathbf{x}_i + b)$

Gradient decent: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \lambda \frac{\partial \mathcal{L}(\mathbf{w},b)}{\partial \mathbf{w}}$

Note the update rule for the perceptron is:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + (target_i - output_i)\mathbf{x}_i$$

Perceptron training is a special gradient descent algorithm

Perceptron: $f(\mathbf{x};\mathbf{w}) = sign(\mathbf{w}^T\mathbf{x} + b)$

$S = \{(\mathbf{x}_i, y_i), i = 1..n\}$

Training: Minimize $\mathcal{L}(\mathbf{w}, b) = \sum_i max(0, -y_i \times (\mathbf{w}^T\mathbf{x}_i + b))$

no penalty if y_i and $(\mathbf{w}^T\mathbf{x}_i + b)$ have the same sign

$$\frac{\mathcal{L}(\mathbf{w},b)}{\partial b} = \sum_i -\frac{1}{2}(target_i - output_i)$$

$$target_i = y_i$$

$$output_i = sign(\mathbf{w}^T\mathbf{x}_i + b)$$

Gradient decent: $b_{t+1} \leftarrow b_t - \lambda \frac{\partial \mathcal{L}(\mathbf{w},b)}{\partial b}$

Note the update rule for the perceptron is:

$$b_{t+1} \leftarrow b_t + (target_i - output_i)$$

Perceptron Learning

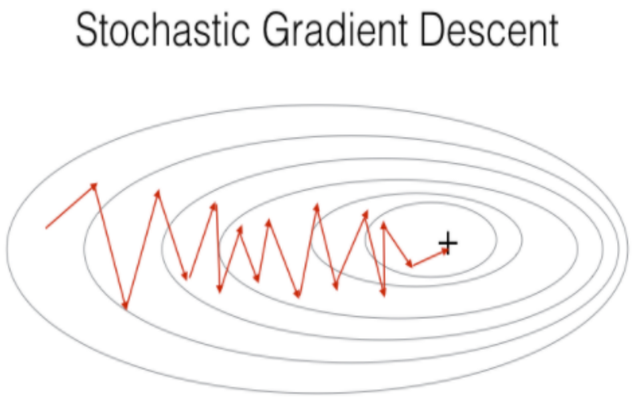
Perceptron:

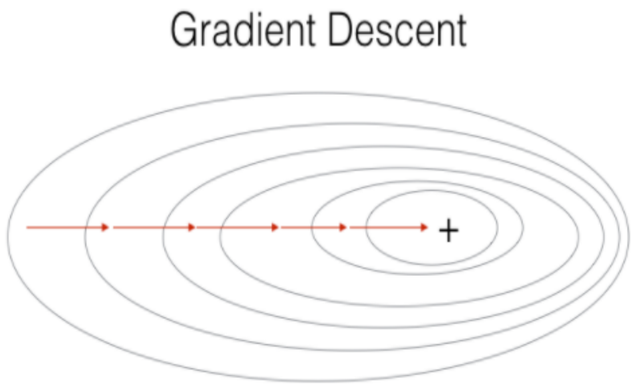
Note that the learning process is not strictly gradient descent.

$$\mathbf{w}_{t+1} = \mathbf{w}_t + (target_i - output_i)\mathbf{x}_i$$
$$b_{t+1} = b_t + (target_i - output_i)$$

It's a stochastic gradient descent algorithm!

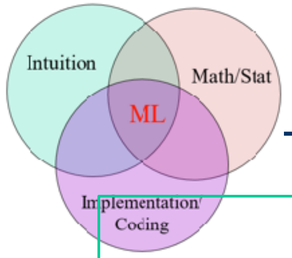
Stochastic vs Batch Gradient Descent





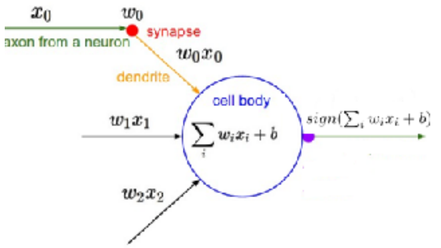
- Random order of updates means you can take different paths with same data
- Online approximation to batch
- Subsample training set for faster progress

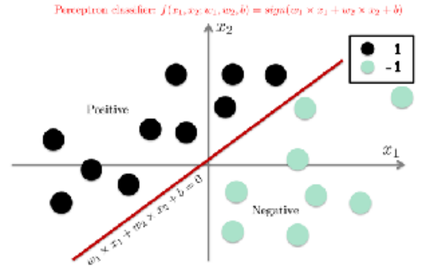
- Always the same path for the same data, order irrelevant
- Offline



Recap: Perceptron

Intuition: Perceptron is a linear classifier predicting via sign of the output. Originally an ad hoc iterative algorithm, it can be recast as gradient descent when replacing 0/1 loss with hinge loss. Perceptrons cannot calculate XOR or other problems that are not linearly separable. It was therefore ignored for many years until the modern age of multi-layer and then deep multi-layer neural networks demonstrated their power.

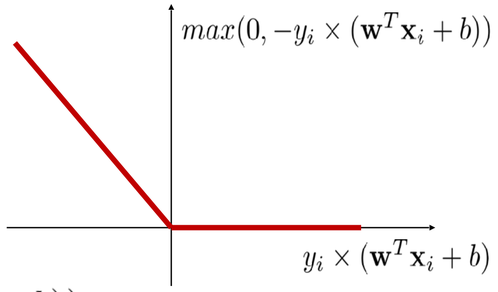




Math:

$$f(\mathbf{x};\mathbf{w},b) = \begin{cases} +1 & if\ \mathbf{w}^T\mathbf{x}+b \geq 0 \\ -1 & otherwise \end{cases}$$

$$S = \{(\mathbf{x}_i, y_i), i = 1..n\}$$

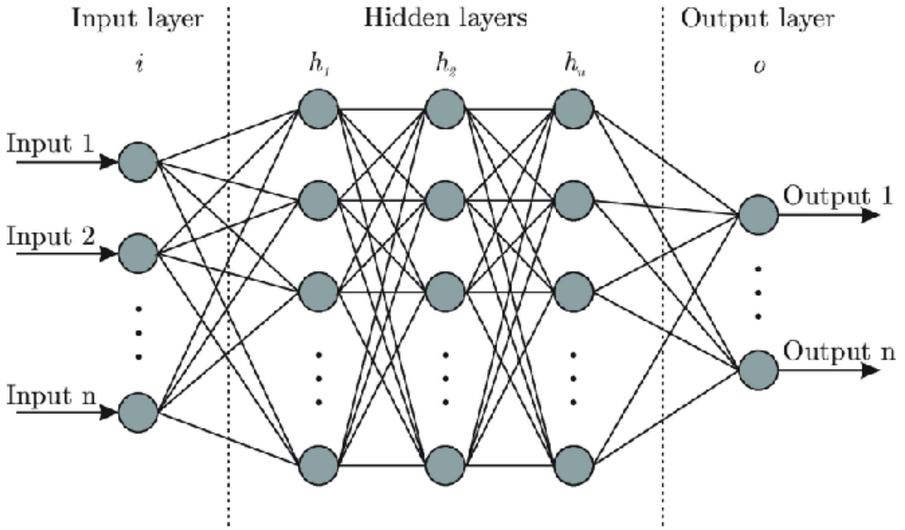


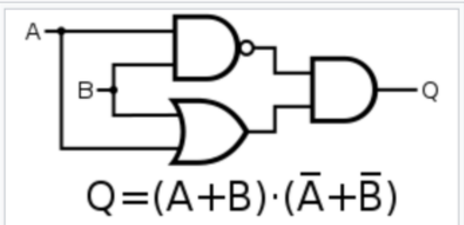
Training: Minimize $\mathcal{L}(\mathbf{w},b) = \sum_i max(0, -y_i \times (\mathbf{w}^T\mathbf{x}_i + b))$

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + (target_i - output_i)\mathbf{x}_i \qquad target_i = y_i$$
$$b_{t+1} \leftarrow b_t + (target_i - output_i) \qquad output_i = sign(\mathbf{w}^T\mathbf{x}_i + b)$$

Multi-layer perceptrons and even deeper

- Solve the XOR, and in fact are general function approximations
- Need a special kind of gradient descent “back propagation”





XOR gate circuit using three mixed gates