Neural Networks and Machine Learning – 7PAM2021

RAFAEL S. DE SOUZA

COURSE OVERVIEW

THE MODULE TEAM

Lecturers

Dr Rafael S. de Souza (Neural Networks)

Dr Mykola Gordovskyy (Machine Learning)

Tutors

Dr Rob Yates

Dr Vito Graffagnino

Dra Vandana Das

Module Learning Outcomes

Learning Outcomes: Knowledge and Understanding

have a knowledge and understanding of a range of neural networks as models of neural computation;

have a knowledge and understanding of advanced machine learning methods, and their effectiveness for the analysis of numerical data.

Learning Outcomes: Skills and Attributes

be able to analyse and critically evaluate a variety of artificial neural architectures and machine learning techniques;

be able to analyse and critically evaluate a range of neural networks models of brain function and neural development;

be able to program non-trivial machine learning algorithms in a modern programming language;

be able to plan and implement data analysis tasks at a professional or equivalent level;

be able to critically evaluate research papers and methodologies.

Neural Networks - Introduction

Introduction

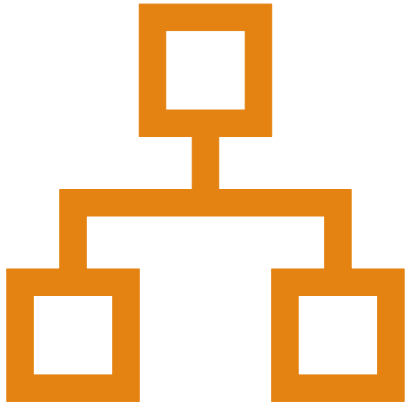
Module structure and assignments

Machine Learning

Landscape

Recap

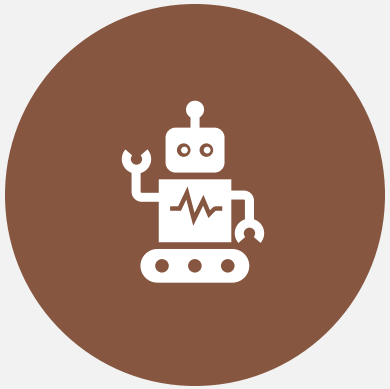
Essential linear algebra and differentiation



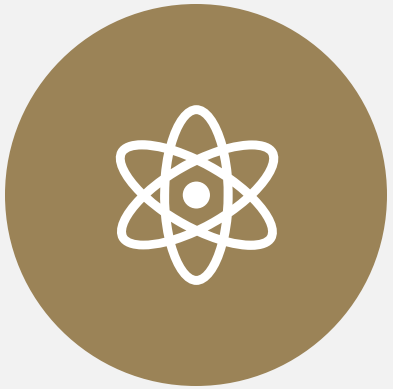
Assessment Structure

Overview

CANVAS QUIZZES – 20% OF TOTAL GRADE



MACHINE LEARNING CASE STUDY – 40% OF TOTAL GRADE



GROUP PROJECT - 40% TOTAL GRADE

Canvas Quizzes (4 x 5%)

Four quizzes, each worth 5% of the final grade.

Quizzes are automatically marked—you will be given 5 questions per quiz, from a pool of 20

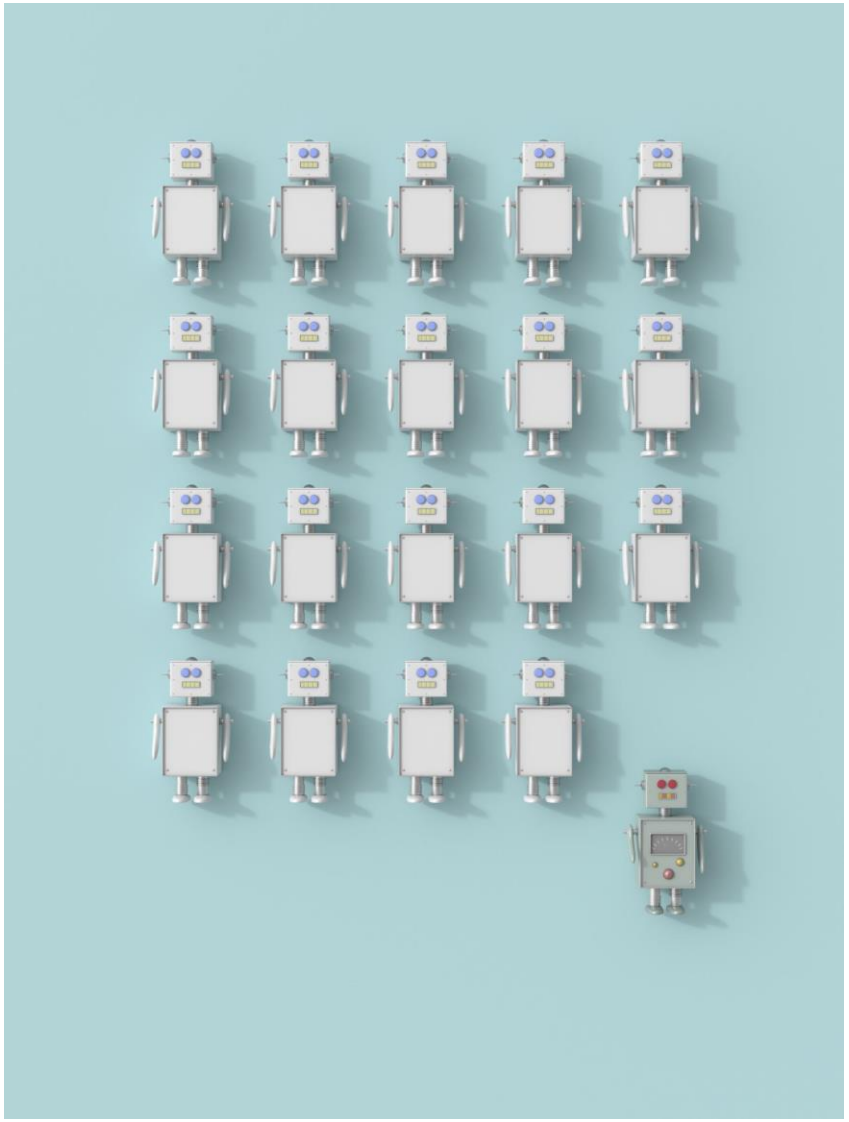
You will have one attempt and 60 minutes to complete the quiz

Quiz 1 (Intro To Neural Nets)

Quiz 2 (Supervised Learning)

Quiz 3 (Deep Learning)

Quiz 4 (Unsupervised Learning)



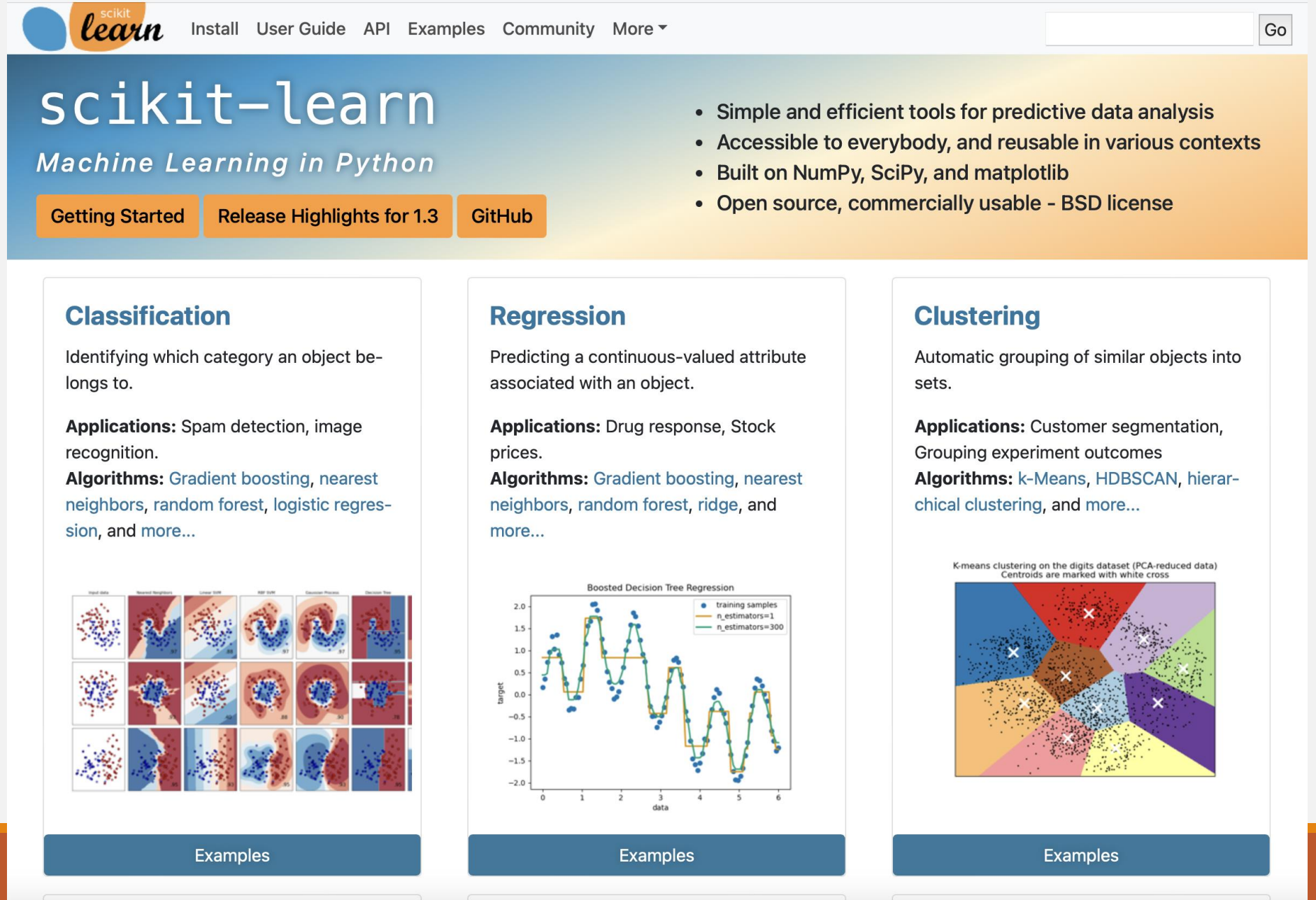
GROUP PROJECT
CNN WITH KERAS
40%

In this assignment, you will utilize the concept of transfer learning for image classification.

You will be provided with a training set to build your model and a validation set for classification. I will retain the true classifications and will evaluate them against your solution.

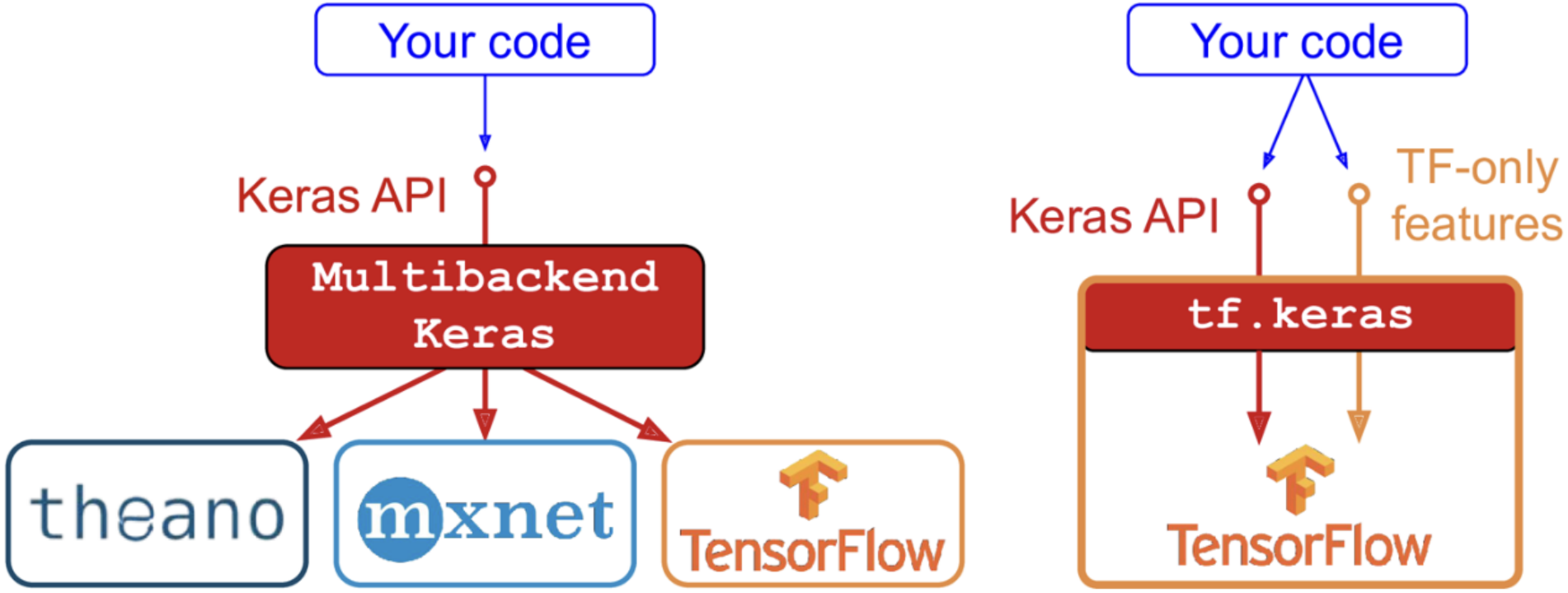
INDIVIDUAL PROJECT (40%) *WITH SCIKIT-LEARN*

- For most of the traditional regression and classification problems, in which your data is a table, you will do just fine using e.g. https://scikit-learn.org/stable/



Keras is a high level deep learning API that allow you to build, train, evaluate and execute different neural networks

Implementing MLPs with Keras



```
model = keras.models.Sequential([
    keras.layers.Conv2D(64, 7, activation="relu", padding="same",
                        input_shape=[28, 28, 1]),
    keras.layers.MaxPooling2D(2),
    keras.layers.Conv2D(128, 3, activation="relu", padding="same"),
    keras.layers.Conv2D(128, 3, activation="relu", padding="same"),
    keras.layers.MaxPooling2D(2),
    keras.layers.Conv2D(256, 3, activation="relu", padding="same"),
    keras.layers.Conv2D(256, 3, activation="relu", padding="same"),
    keras.layers.MaxPooling2D(2),
    keras.layers.Flatten(),
    keras.layers.Dense(128, activation="relu"),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(64, activation="relu"),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(10, activation="softmax")
])
```

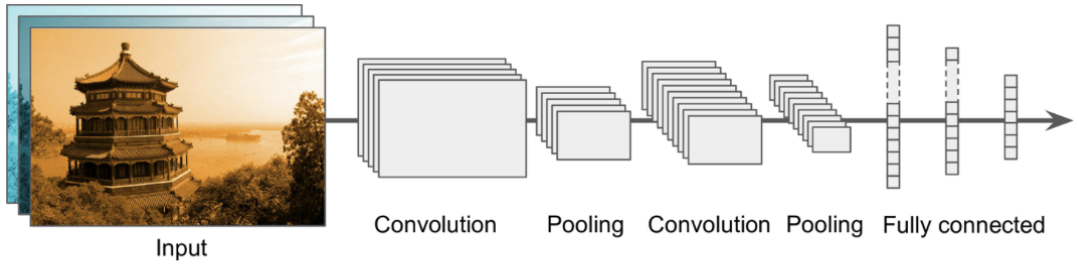
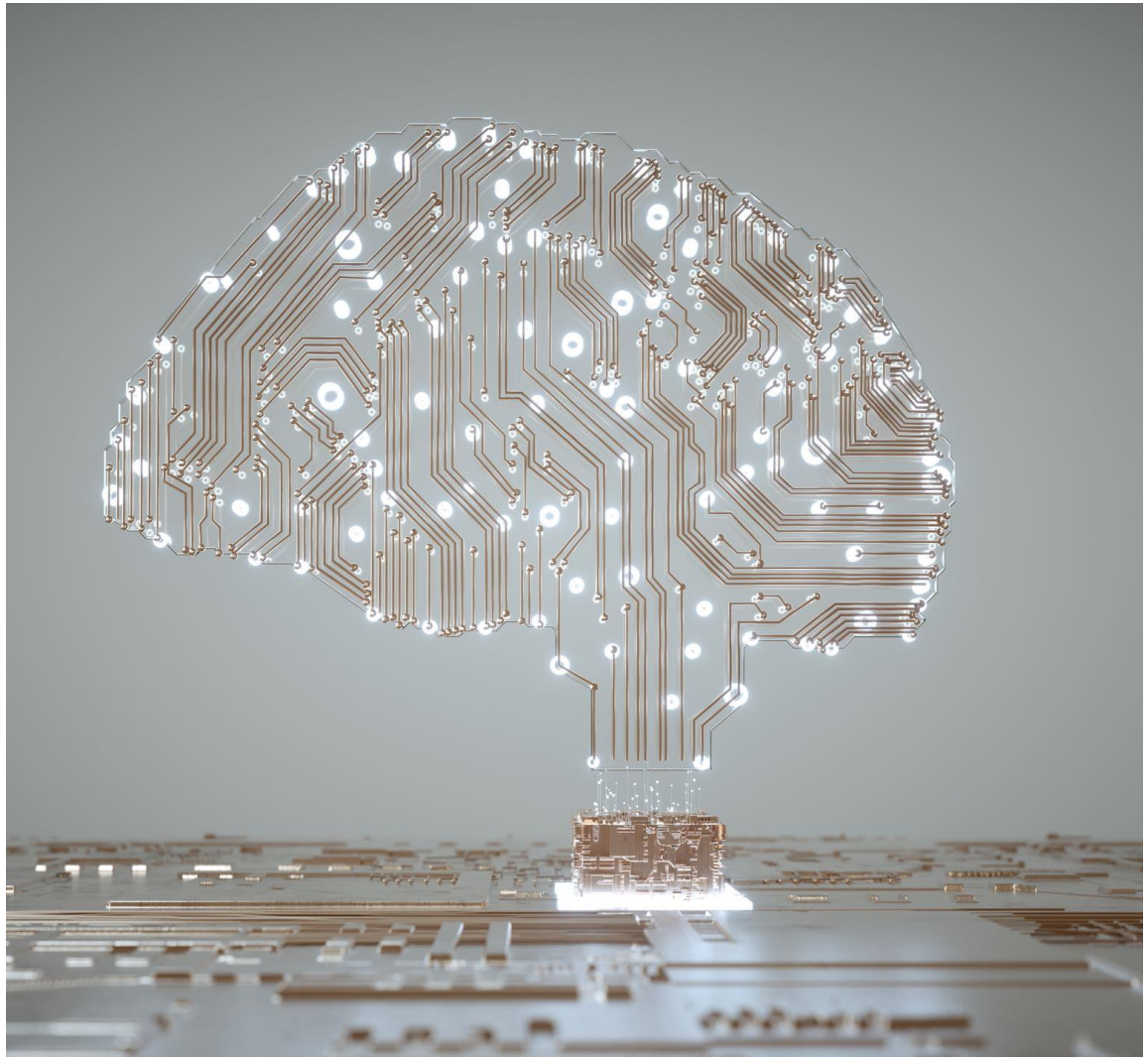


Figure 14-11. Typical CNN architecture

Mock Quizz

Neural Networks and Machine Learning – 7PAM2021

RAFAEL S. DE SOUZA



AI, ML and NN

ARTIFICIAL INTELLIGENCE, MACHINE LEARNING, AND NEURAL NETWORKS

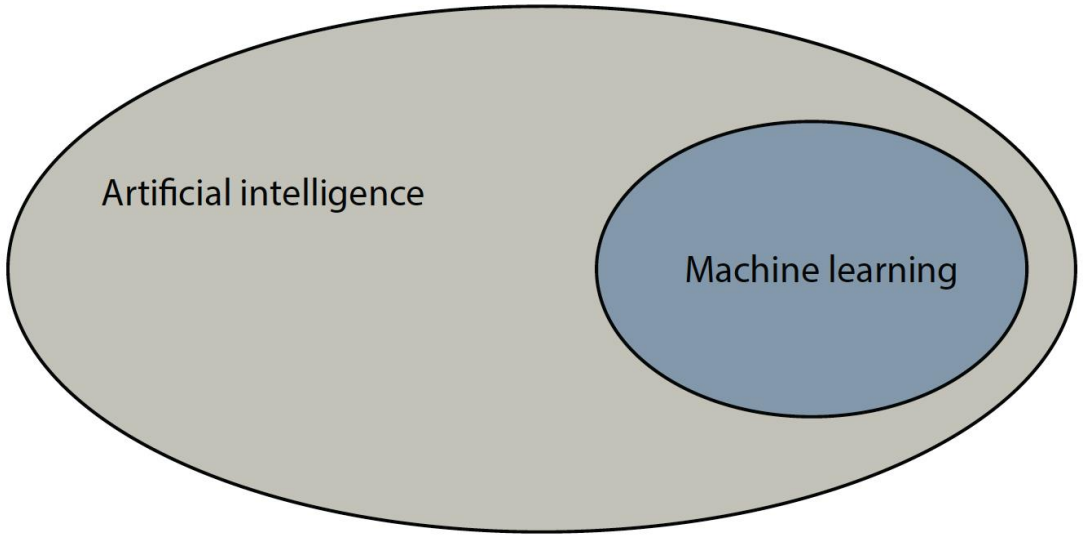
Artificial Intelligence

Artificial Intelligence, within the realm of computer science, aims to construct machines that mimic human thought processes and actions.

The overarching objective for many AI initiatives is to accomplish tasks that traditionally necessitate human intervention.

'Narrow AI' pertains to systems designed for a specific task or a limited set of tasks.

Conversely, 'General AI' references systems that emulate human or rational thinking to tackle a wide variety of challenges.



AI, ML, NN and DL

Machine Learning

A subfield of AI, which concerns itself with algorithms that learn to solve problems by learning from historical data, instead of being programmed to solve them directly

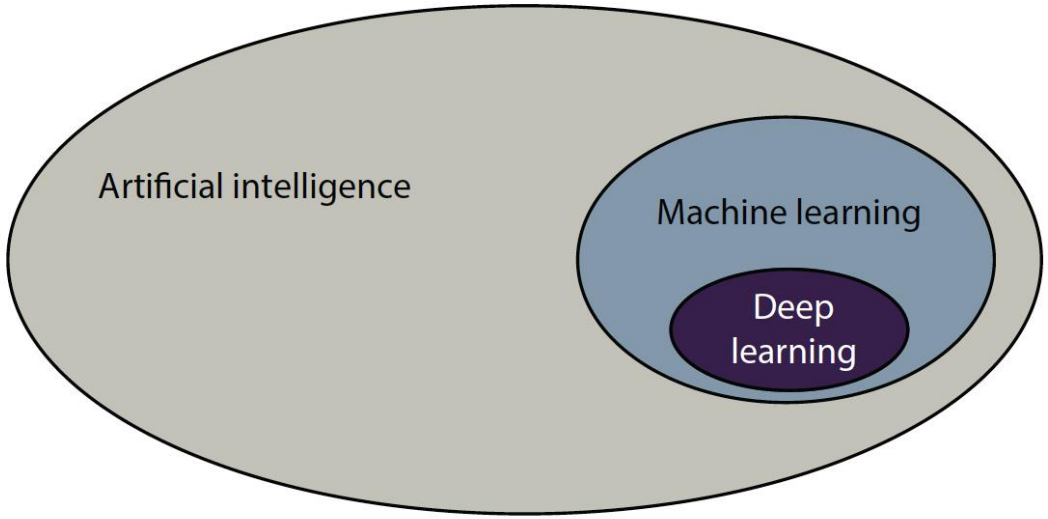
Machine Learning algorithms model a problem and update themselves through *training* to become better at a given task—spoiler alert, this is also what *you* do

Machine Learning

Neural Networks

An approach to machine learning that uses *artificial neural networks*, models which were inspired by how our own brains form connections

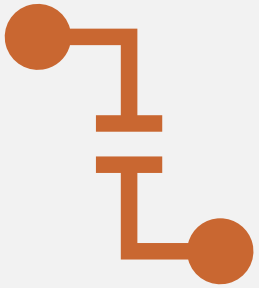
ANNs are made up of computational neurons, connected in layered networks, which feed input data through a series of calculations to predict an output



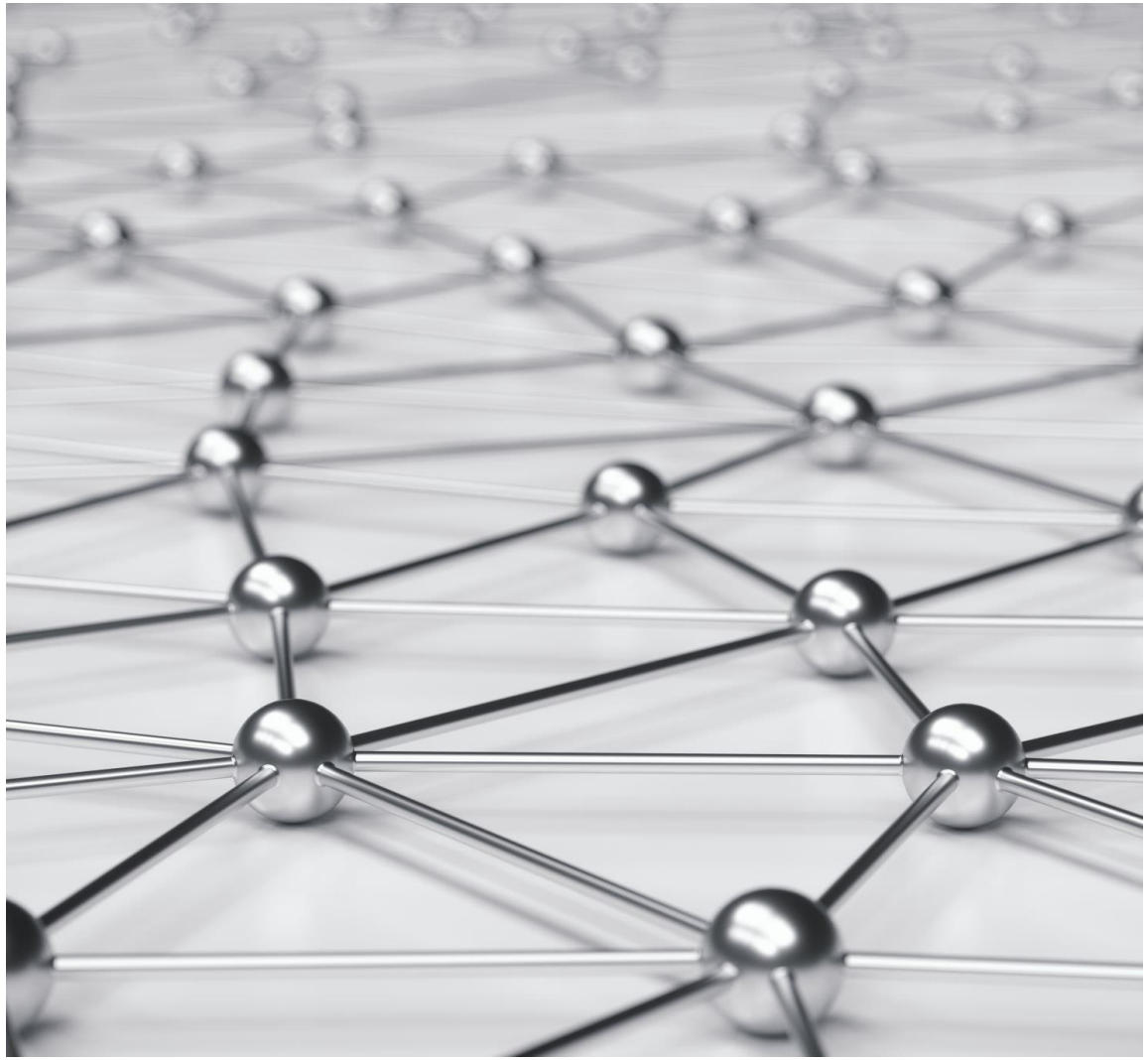
AI, ML, NN and DL

Deep Learning

Deep learning takes neural networks one step further, by constructing models with many hidden layers



Deep neural networks can model complex, non-linear problems, with each layer able to build on the features learned in the previous layer



HOW DO MACHINES LEARN?

01

A model that learns from labelled data is said to be supervised

02

The models make predictions of the labels from the data, then checks its results before updating itself

03

Typically, these are *classification* models

How Do Machines Learn?

How Do Machines Learn?

SUPERVISED LEARNING

A model that learns from labelled data is said to be supervised

The models make predictions of the labels from the data, then checks its results before updating itself

Typically these are *classification* models

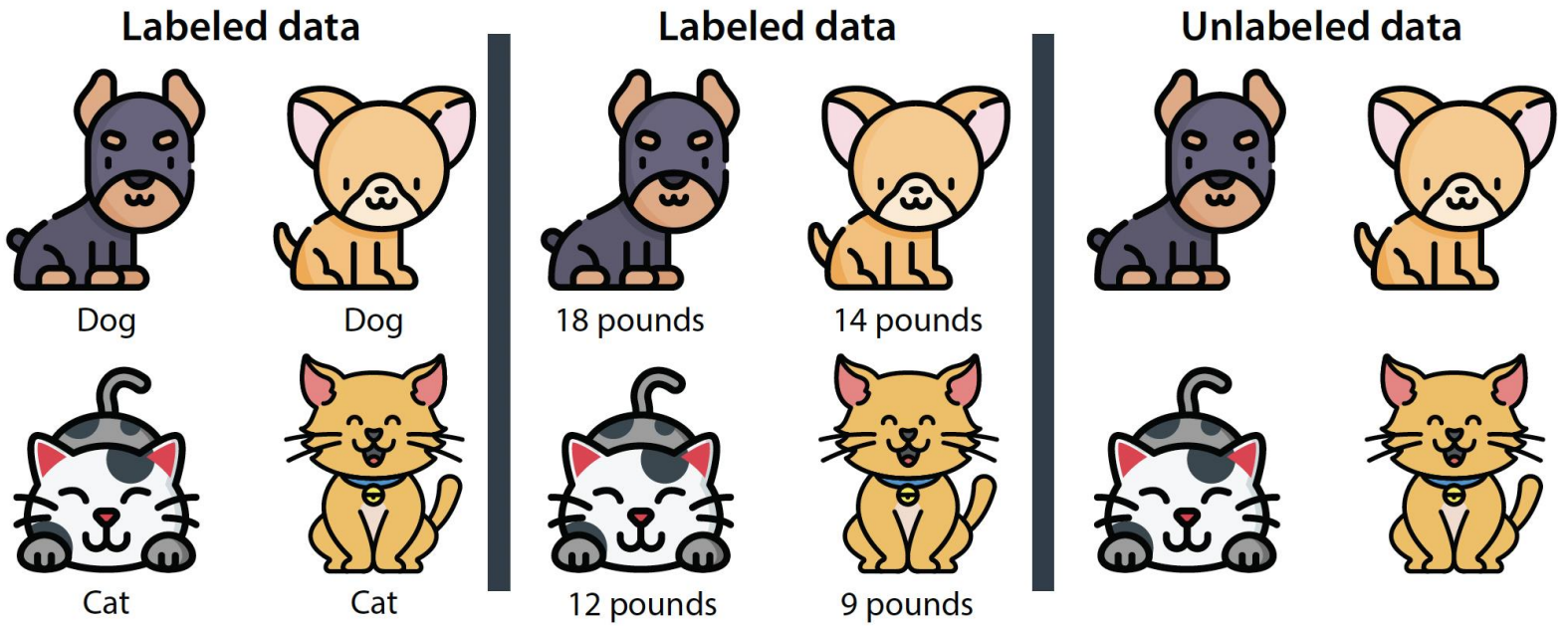
UNSUPERVISED LEARNING

A model that learns from unlabelled data is unsupervised

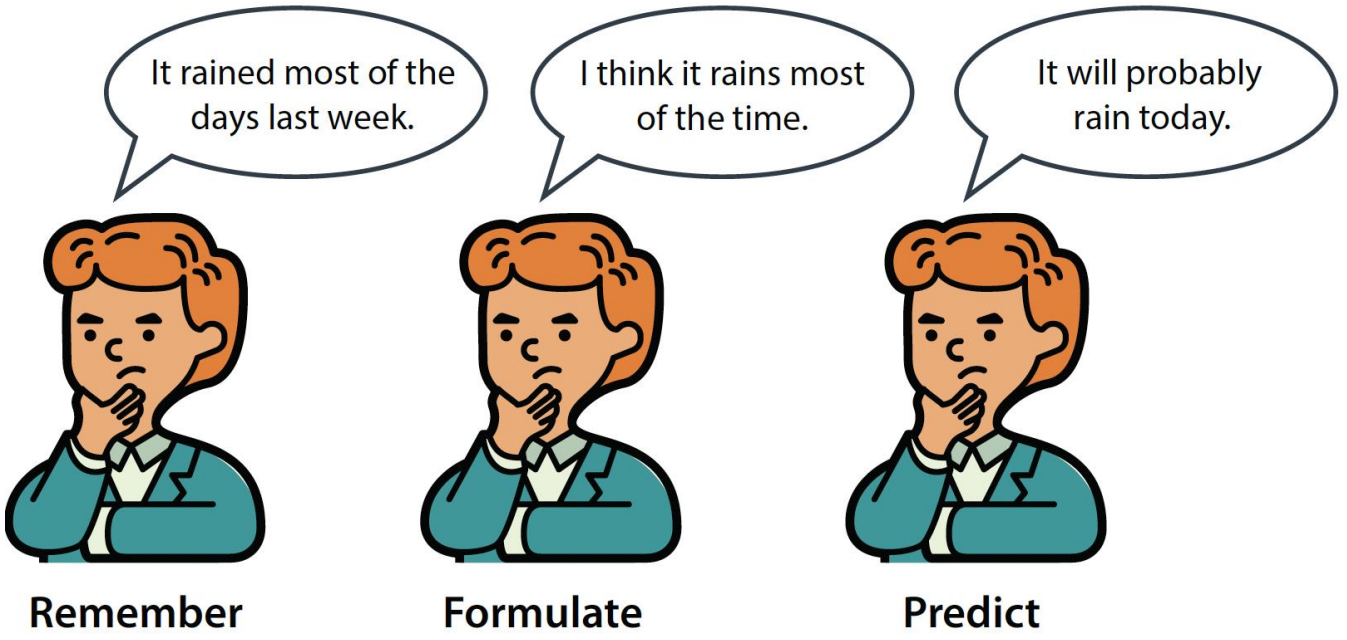
These models search from features in the data that allow them to minimise some function

These models might be *clustering* or *regression* algorithms

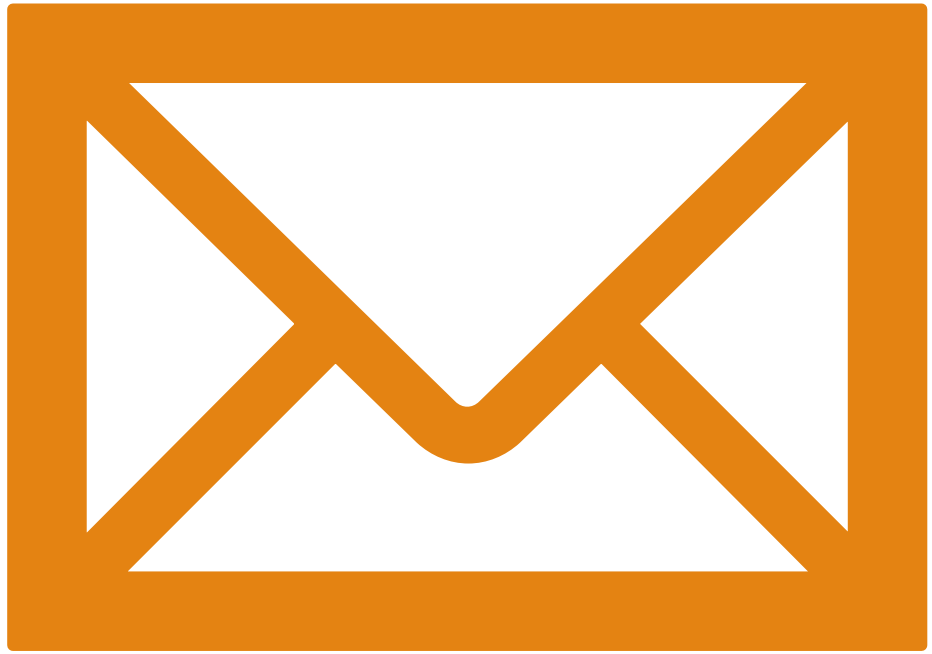
Supervised or Unsupervised



Source: *grokking Machine Learning*



- *How do humans think?*
- *When we, as humans, need to make a decision based on our experience, we normally use the*
- *Fllowing framework:*
- *1. We remember past situations that were similar.*
- *2. We formulate a general rule.*
- *3. We use this rule to predict what may happen in the future.*



TOY CASE

Our friend B likes to send us email. A lot of his emails are spam, in the form of chain letters. We are starting to get a bit annoyed with him. It is Saturday, and we just got a notification of an email from B. Can we guess if this email is spam without looking at it? To figure this out, we use the remember-formulate-predict method.

First, let us remember, say, the last 10 emails that we got from B. That is our data. We remember that 6 of them were spam. From this information, we can formulate the following model:

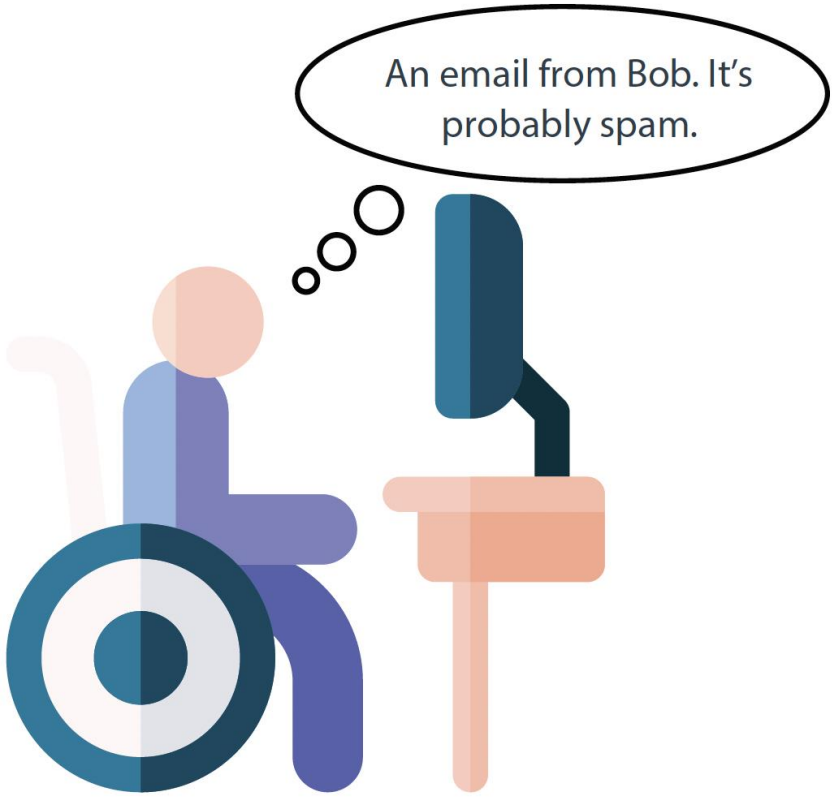
TOY CASE

Our friend B likes to send us email. A lot of his emails are spam, in the form of chain letters. We are starting to get a bit annoyed with him. It is Saturday, and we just got a notification of an email from B. Can we guess if this email is spam without looking at it? To figure this out, we use the remember-formulate-predict method.

First, let us remember, say, the last 10 emails that we got from B. That is our data. We remember that 6 of them were spam. From this information, we can formulate the following model:

Model 1: 6 out of every 10 emails that B sends us are spam. 60% chance B will send a spam.

A very Simple ML model



LET'S ADD MORE FEATURES

FORMULATE A BETTER MODEL:

- Monday: Ham
- Tuesday: Ham
- Saturday: Spam
- Sunday: Spam
- Sunday: Spam
- Wednesday: Ham
- Friday: Ham
- Saturday: Spam
- Tuesday: Ham
- Thursday: Ham

A slightly better model

One day we see B in the street, and he asks, "Why didn't you come to my birthday party?"

We have no idea what he is talking about.
It turns out last Sunday, he sent us an invitation to his birthday party, and we missed it! Why did we miss it?

Because he sent it on the weekend, and we assumed that it would be spam.

It seems that we need a better model. Let's go back to look at Bob's emails—this is our remember step.

Let's see if we can find a pattern.

1 KB: Ham

2 KB: Ham

16 KB: Spam

20 KB: Spam

18 KB: Spam

3 KB: Ham

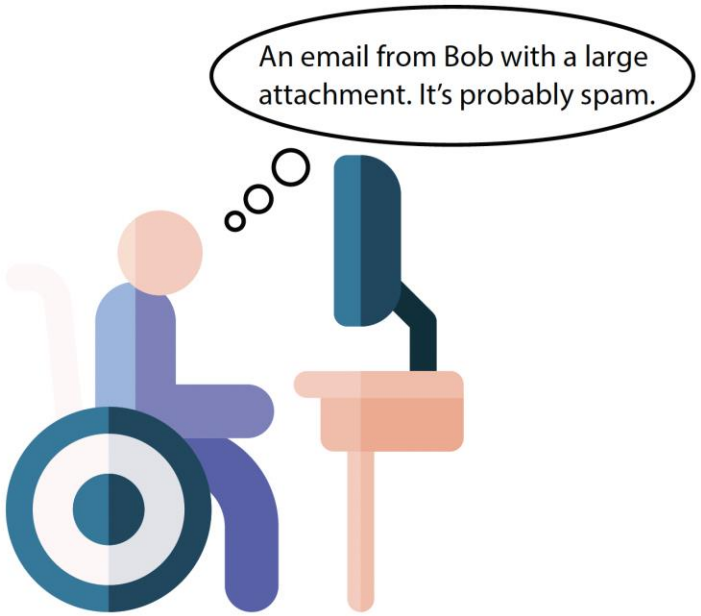
5 KB: Ham

25 KB: Spam

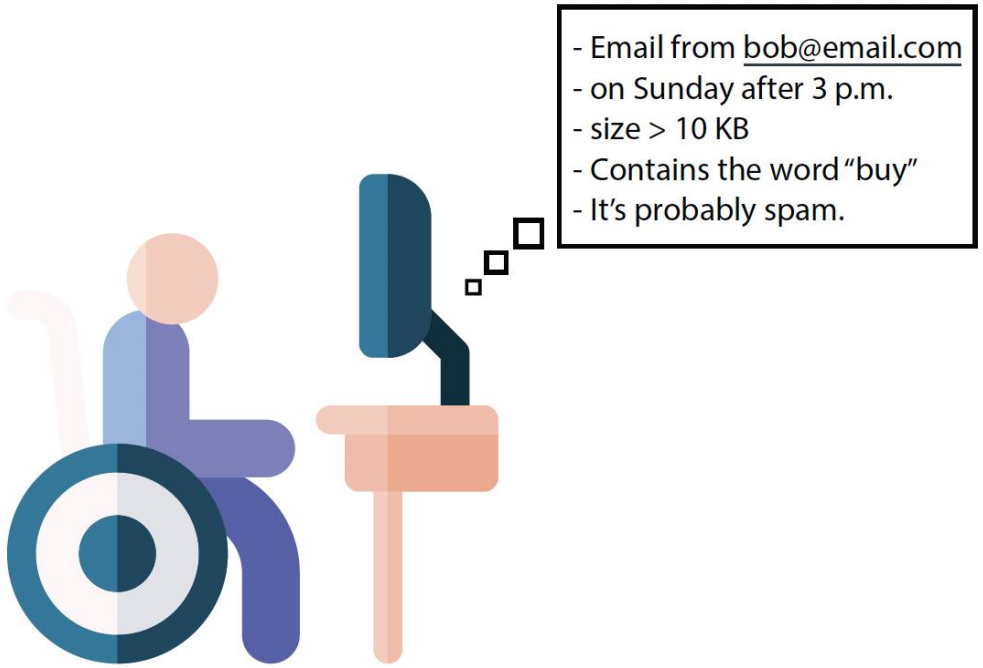
1 KB: Ham

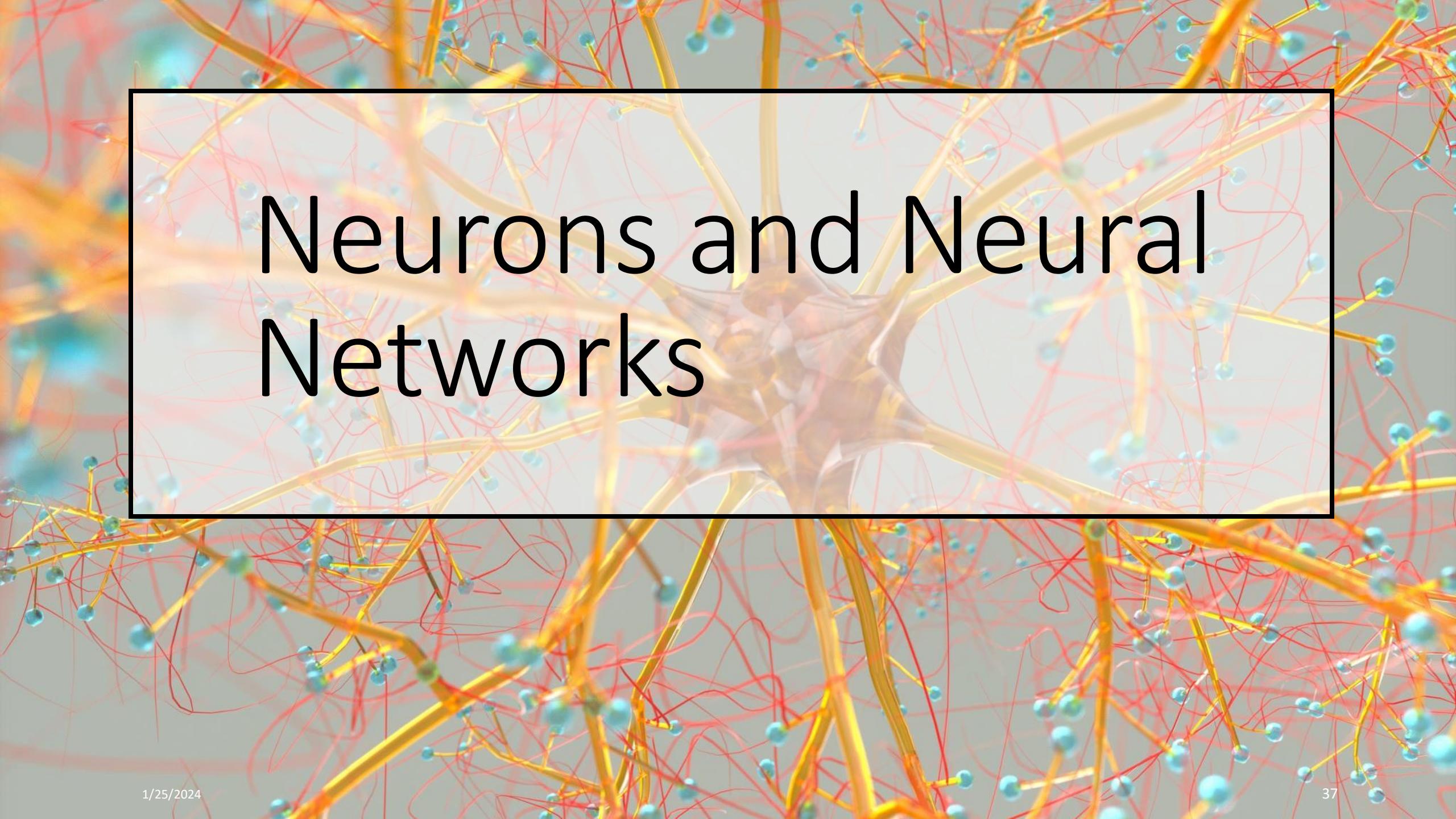
3 KB: Ham

A slightly better model

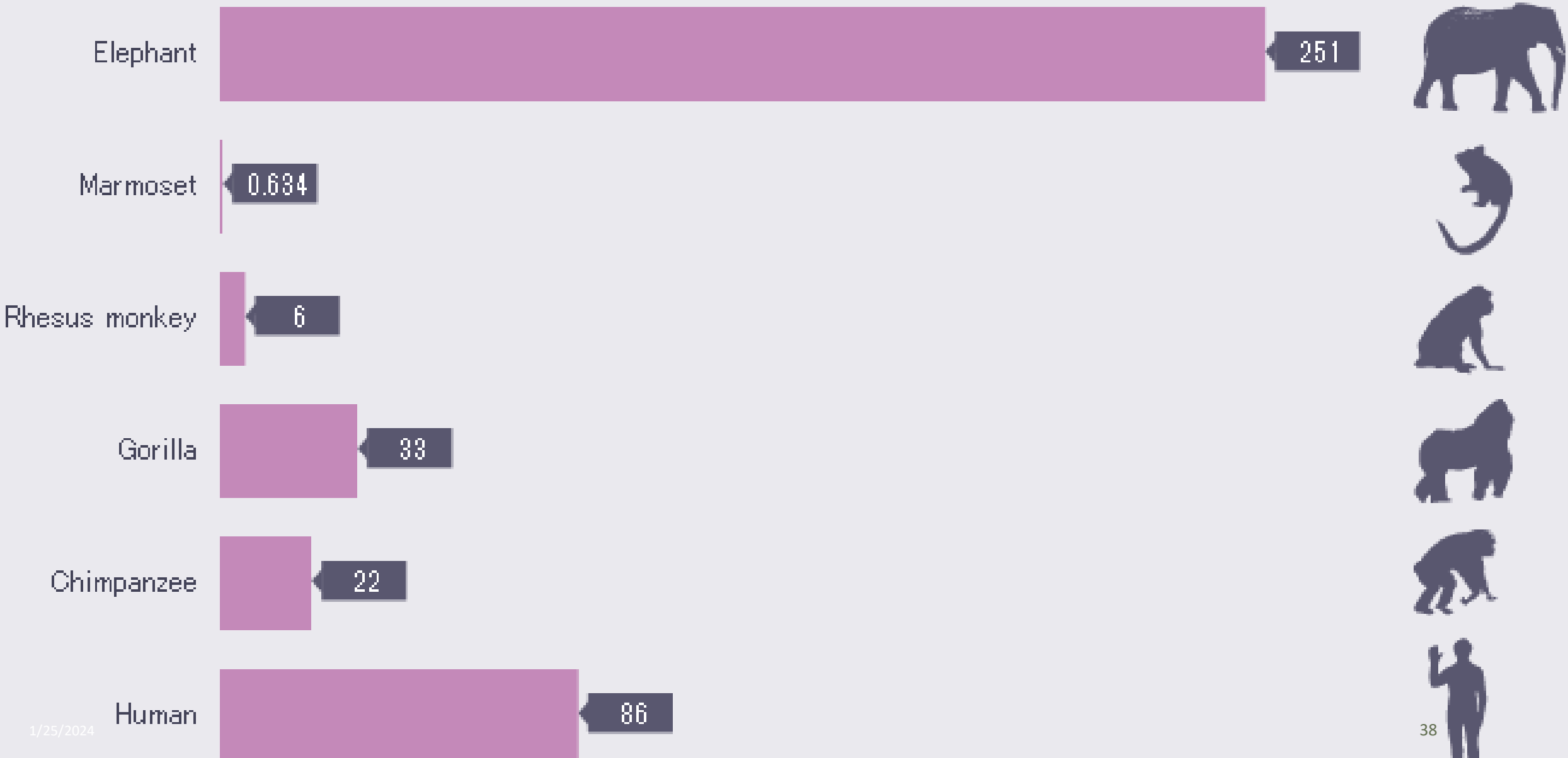


A more complex model




Neurons and Neural Networks

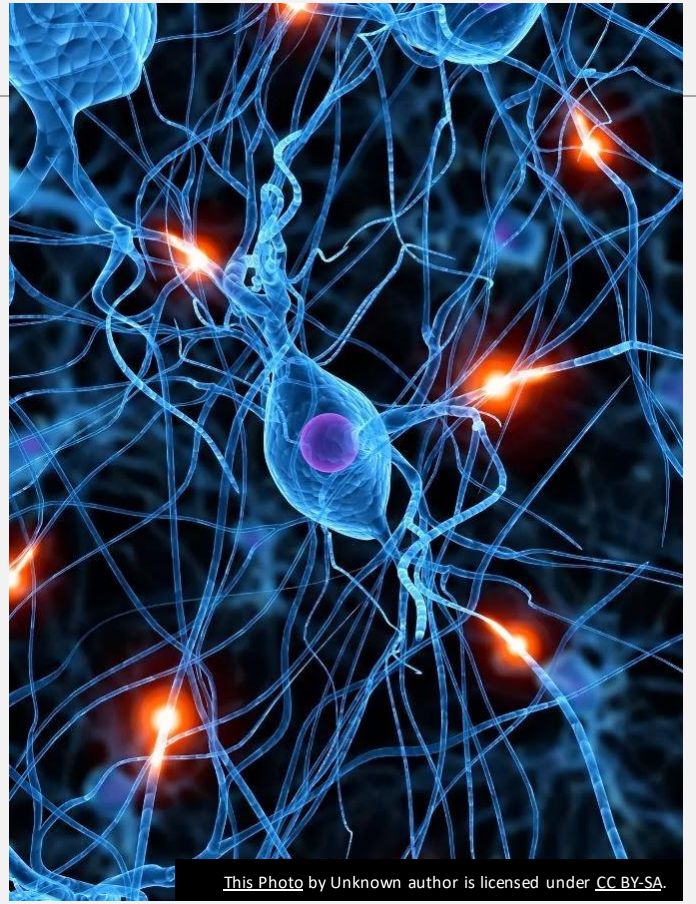
Brain neurons (billions)



Connectionism

A movement in cognitive science that hopes to explain intellectual abilities using artificial neural networks (also known as “neural networks” or “neural nets”).

- Stanford Encyclopedia of Philosophy



Psychological Review
Vol. 65, No. 6, 1958

THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN[1]

F. ROSENBLATT

Cornell Aeronautical Laboratory

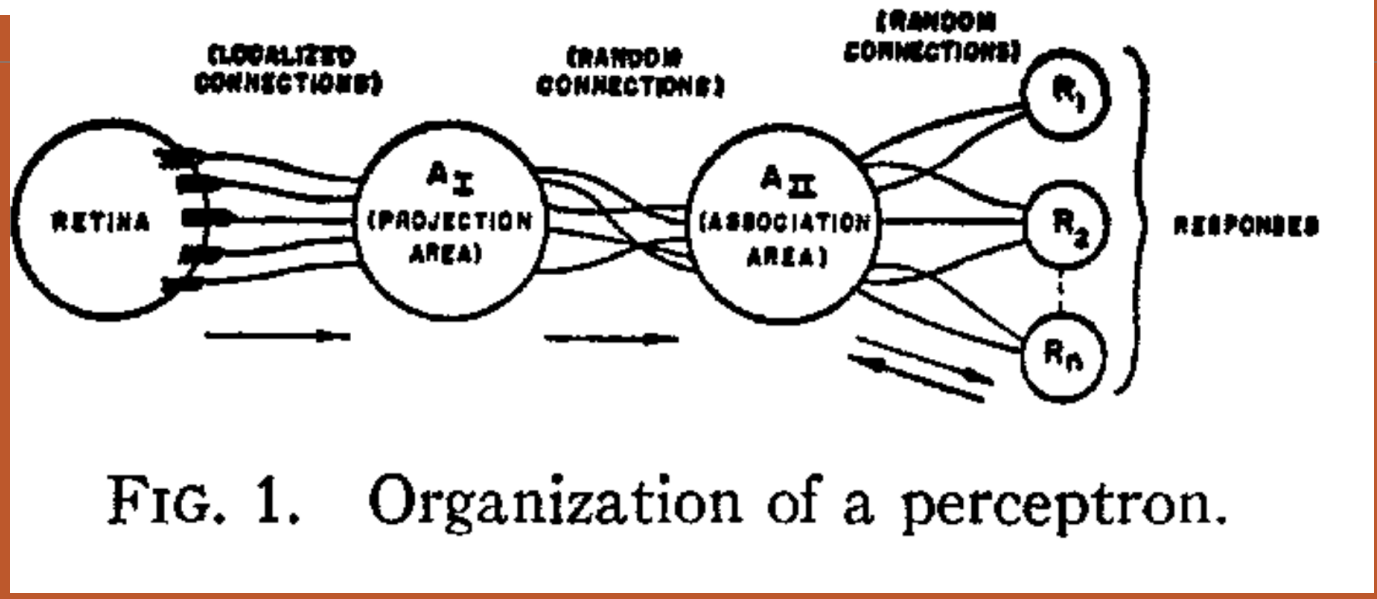
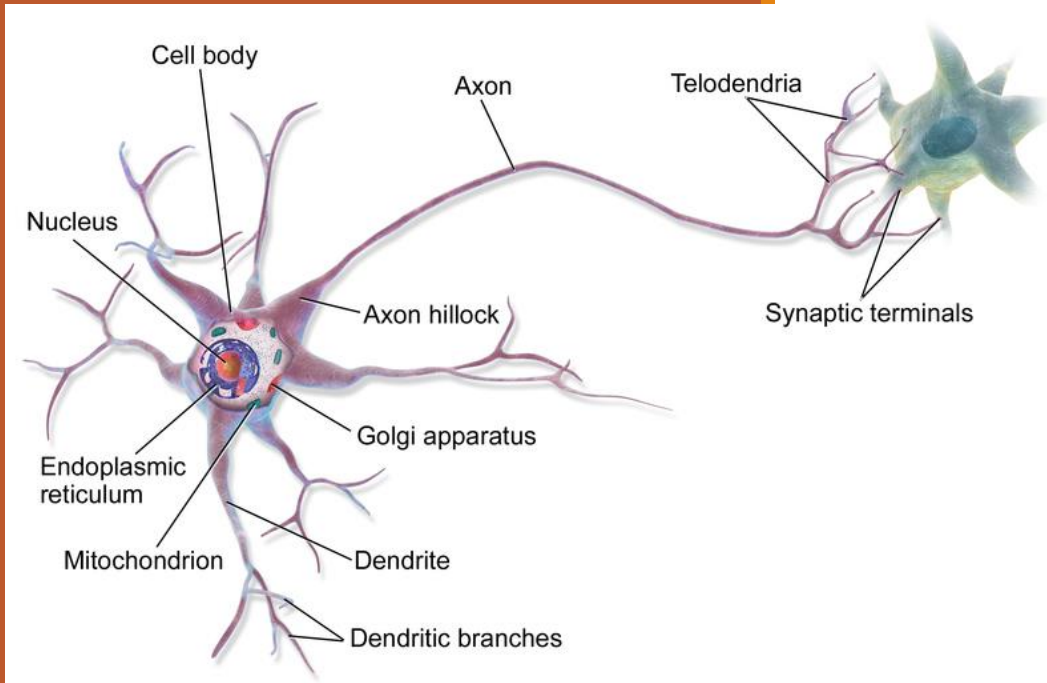


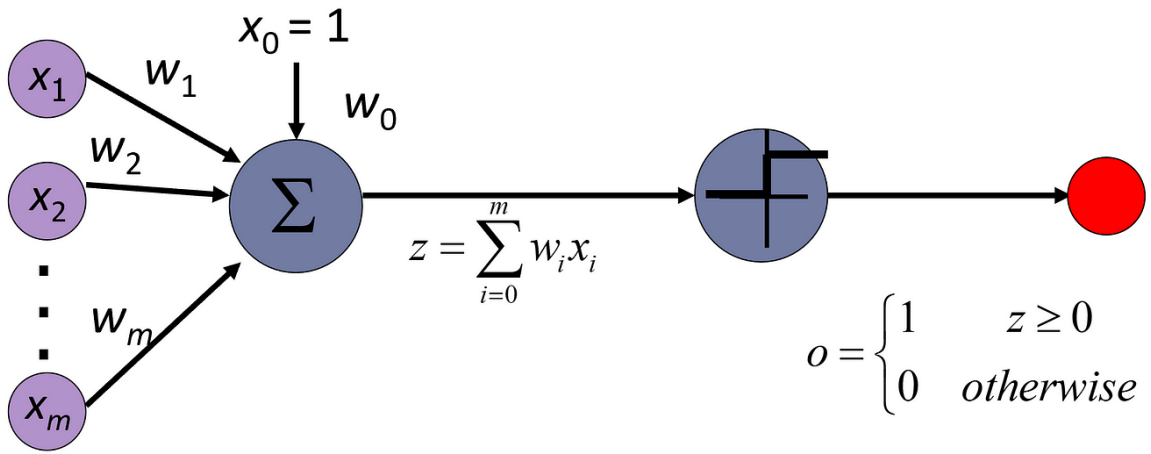
FIG. 1. Organization of a perceptron.

Perceptron

Biological Neuron



PERCEPTRON



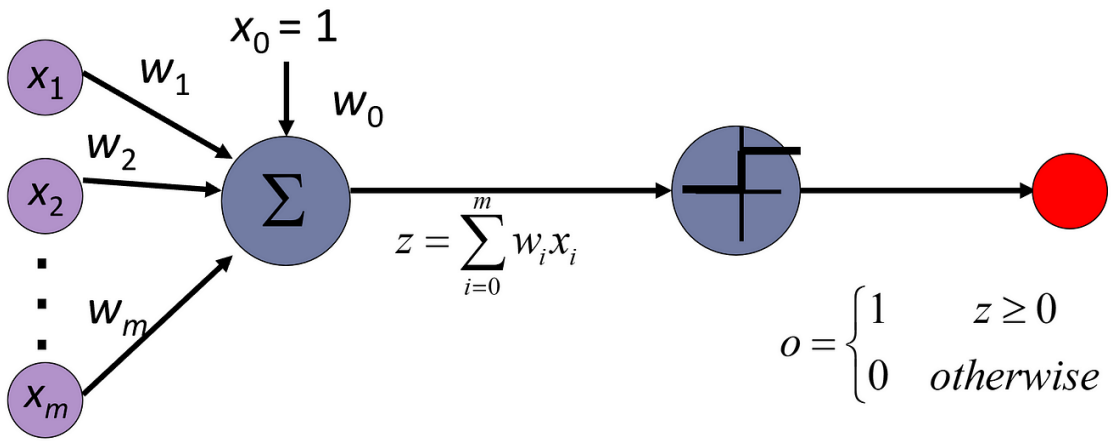
Example

The perceptron first computes the weighted sum of its incoming signals, by multiplying each input by its corresponding weight. This weighted sum is often called **net input** and denoted by *z*:

$$z = w_1 x_1 + \ldots + w_m x_m = \sum_{i=1}^{m} w_i x_i$$

If the net input exceeds some predefined threshold value ϑ, then the perceptron fires (its output is 1), otherwise it doesn't fire (its output is 0). In other words, the perceptron fires if and only if:

$$z = \sum_{i=1}^{m} w_i x_i \geq \theta$$

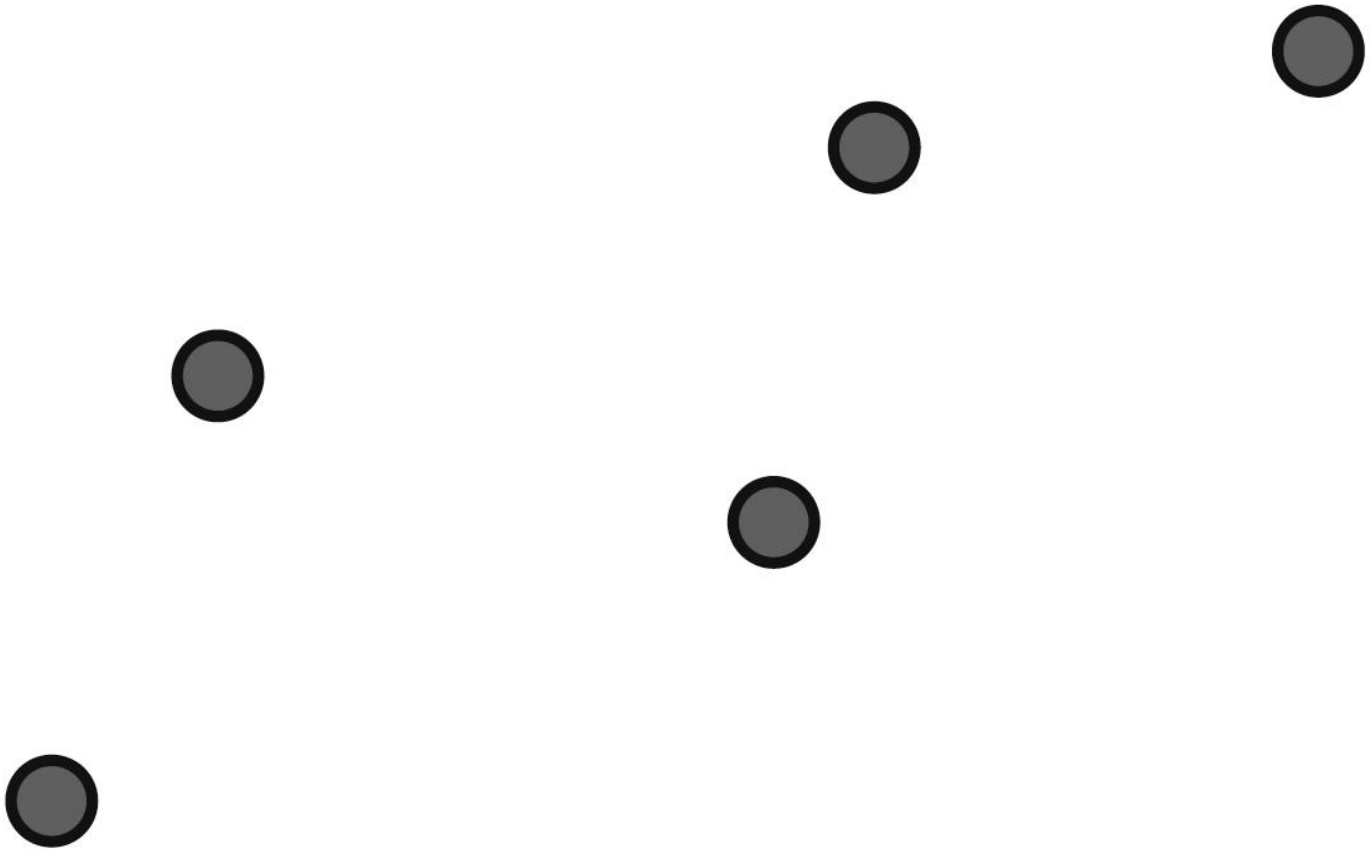


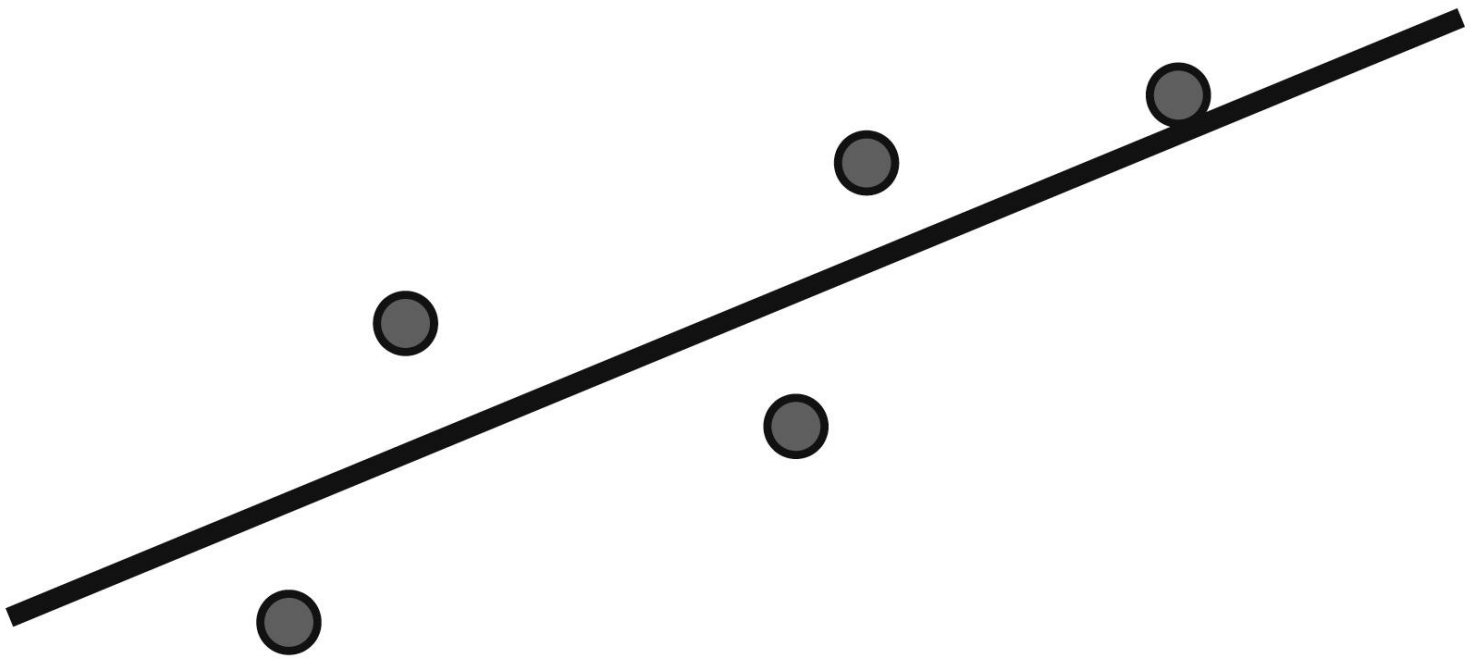
$$\hat{y}_i = \text{step}(w_1 x_1^{(i)} + w_2 x_2^{(i)} + \cdots + w_n x_n^{(i)} + b).$$

More generally

Playground

Let's get some intuition





What do we mean by “points that roughly look like they are forming a line”?

What do we mean by “a line that passes really close to the points”?

How do we find such a line?

Why is this useful in the real world?

The problem: We need to predict the price of a house

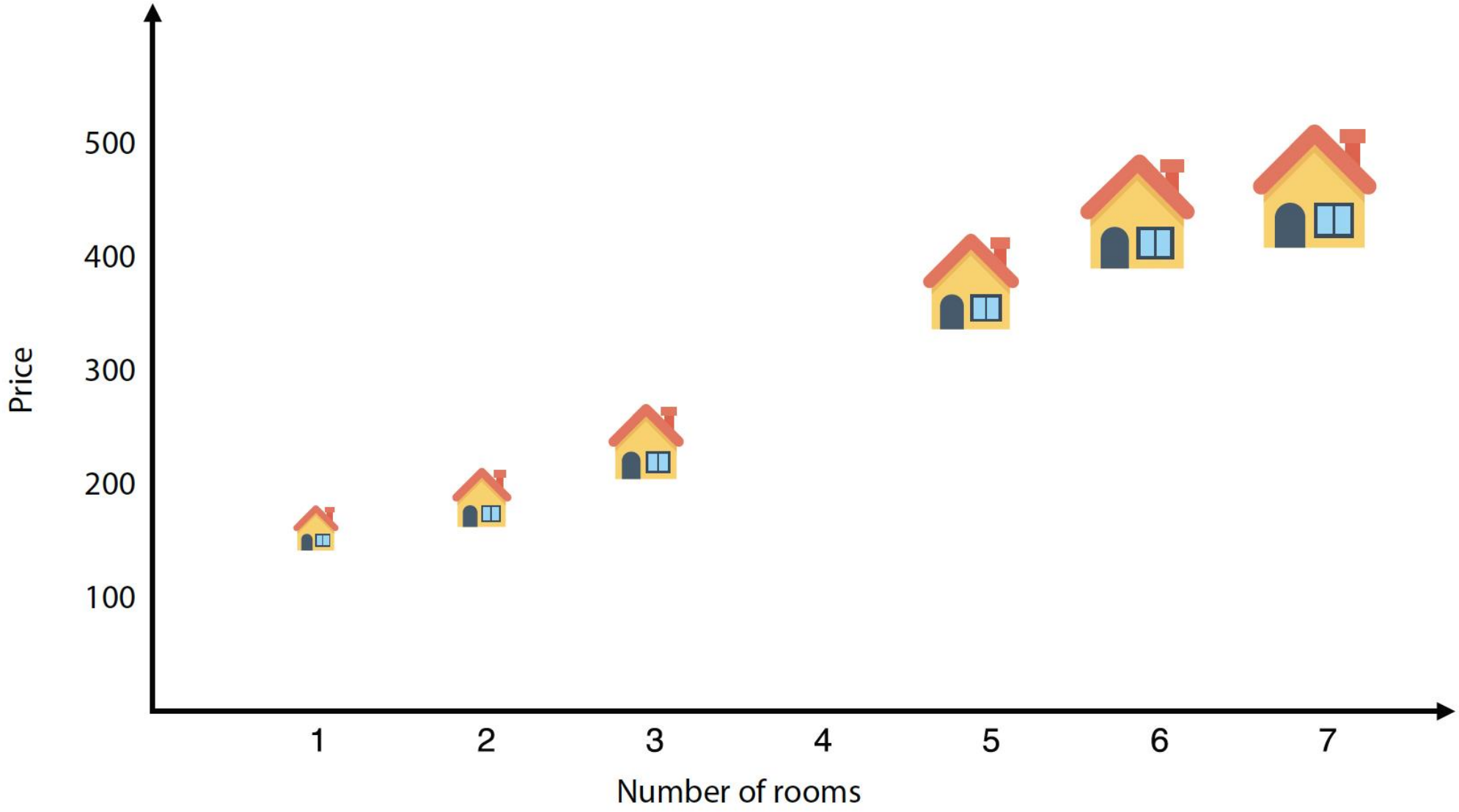
Number of rooms	Price
1	150
2	200
3	250
4	?
5	350
6	400
7	450

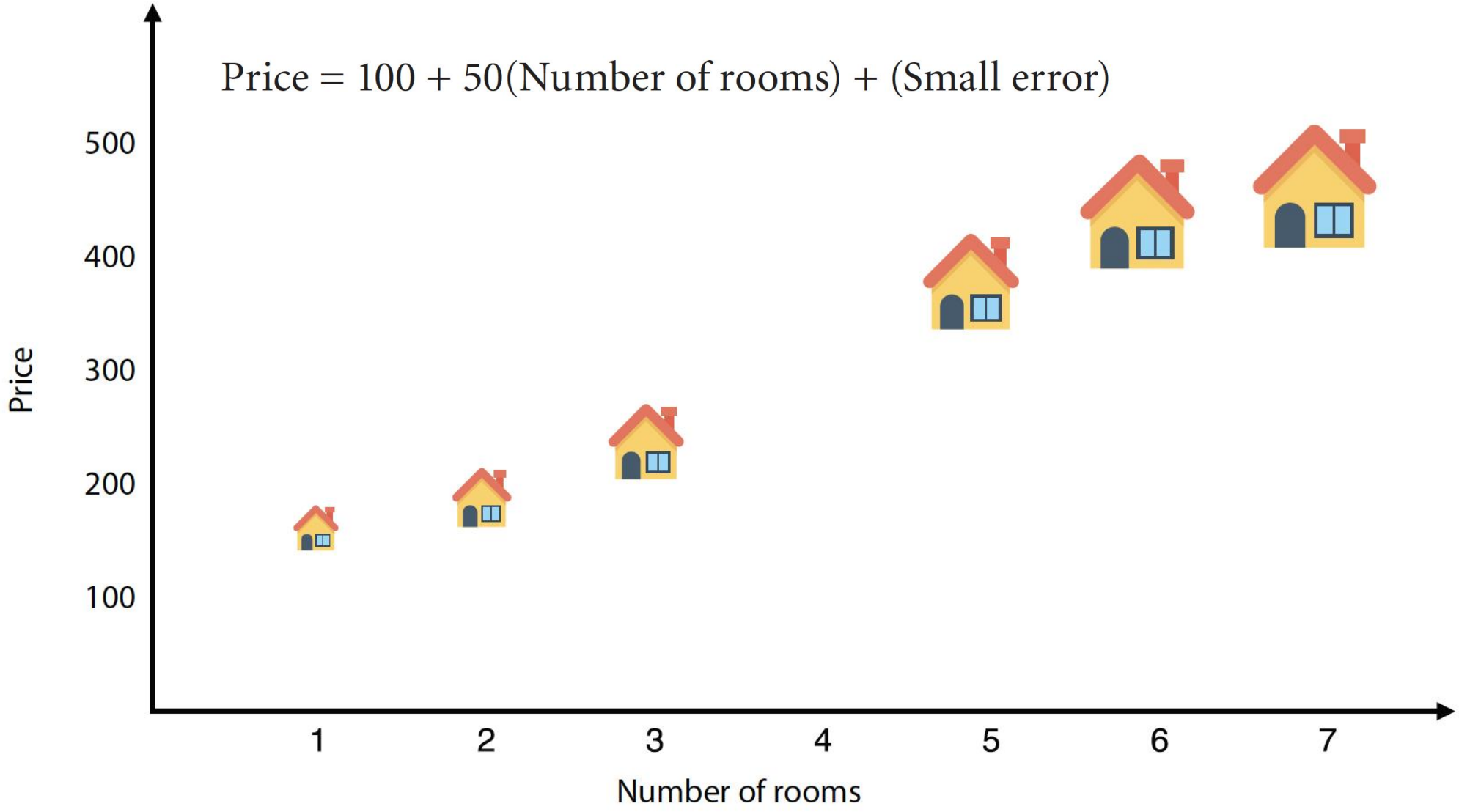
A table of houses with the number of rooms and prices. House 4 is the one whose price we are trying to infer.

Number of rooms	Price
1	150
2	200
3	250
4	?
5	350
6	400
7	450

$$\text{Price} = 100 + 50(\text{Number of rooms})$$

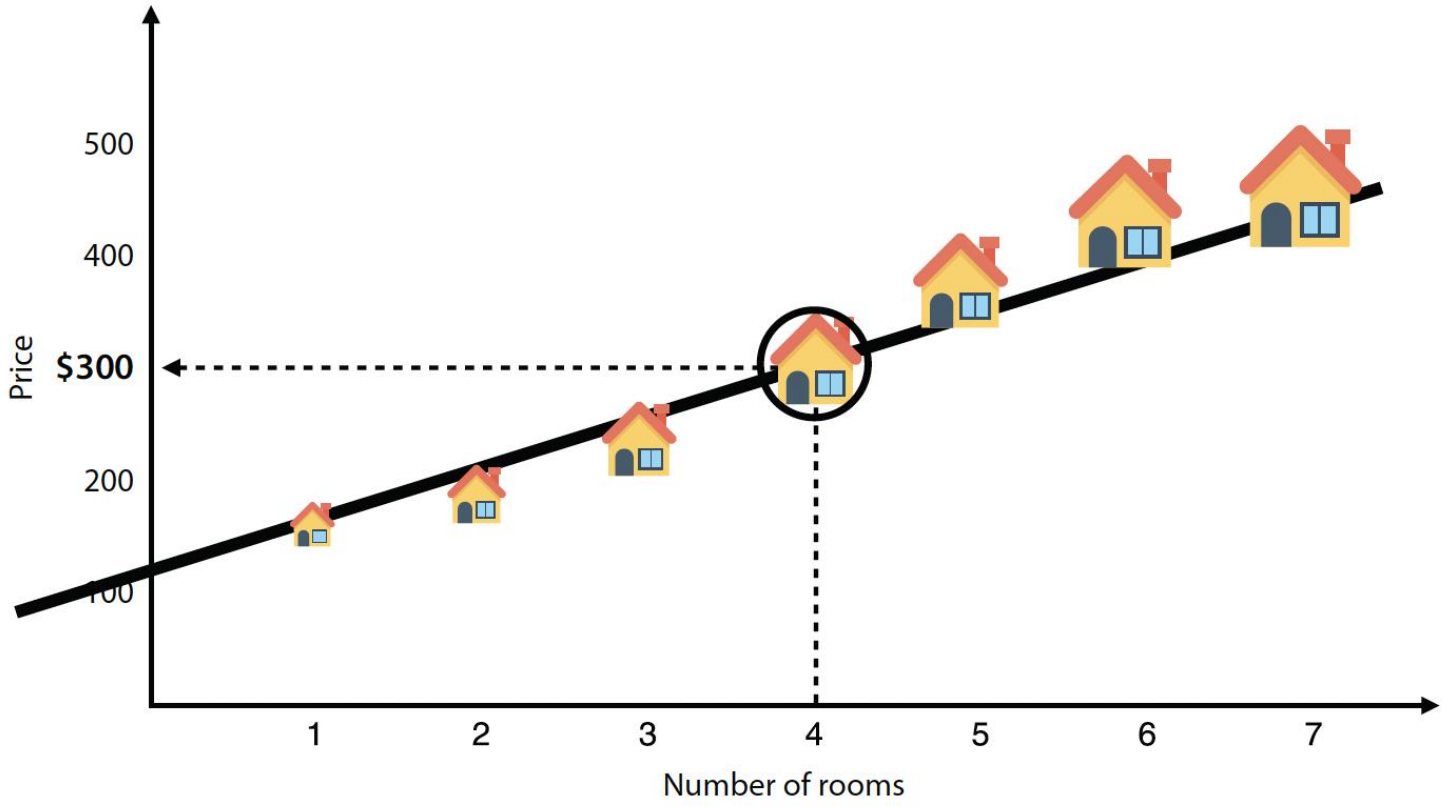
Number of rooms	Price
1	155
2	197
3	244
4	?
5	356
6	407
7	448


Price
500
400
300
200
100
1
2
3
4
5
6
7
Number of rooms


Price = 100 + 50(Number of rooms) + (Small error)
Price
500
400
300
200
100
1
2
3
4
5
6
7
Number of rooms

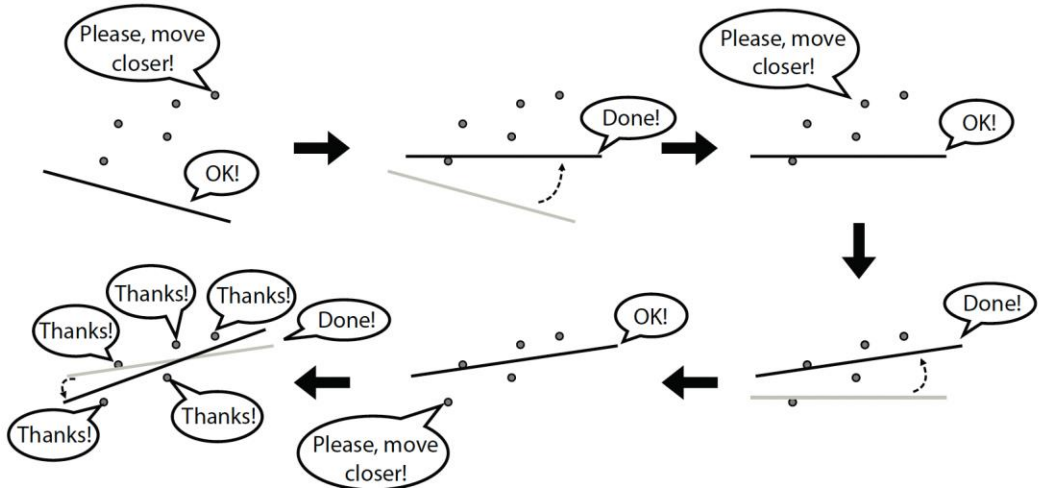

Price = 100 + 50 · (Number of rooms)
500
400
300
200
100
Price
1
2
3
4
5
6
7
Number of rooms

The predict step: What do we do when a new house comes on the market?



What if we have more variables? Multivariate linear regression

$$\text{Price} = 30(\text{number of rooms}) + 1.5(\text{size}) + 10(\text{quality of the schools}) - 2(\text{age of the house}) + 50$$



Pseudocode for the linear regression algorithm (geometric)

Inputs: A dataset of points in the plane

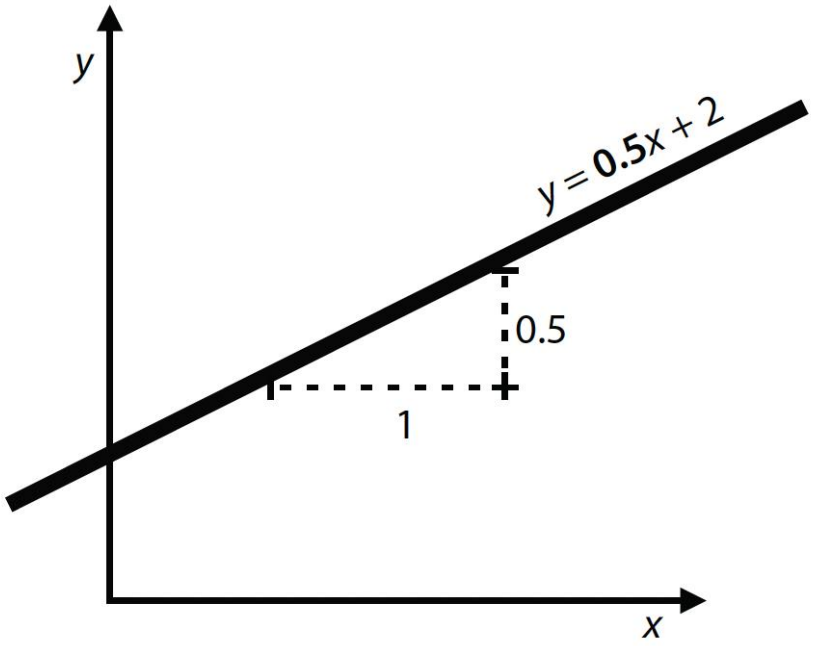
Outputs: A line that passes close to the points

Procedure:

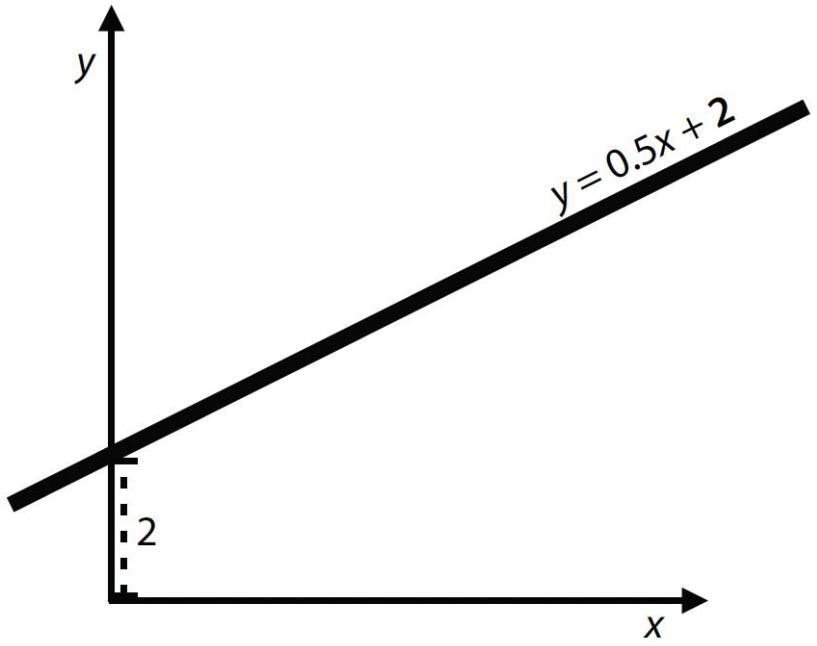
- Pick a random line.
- Repeat many times:
  - Pick a random data point.
  - Move the line a little closer to that point.
- **Return** the line you've obtained.

How to get the computer to draw this line: The linear regression algorithm

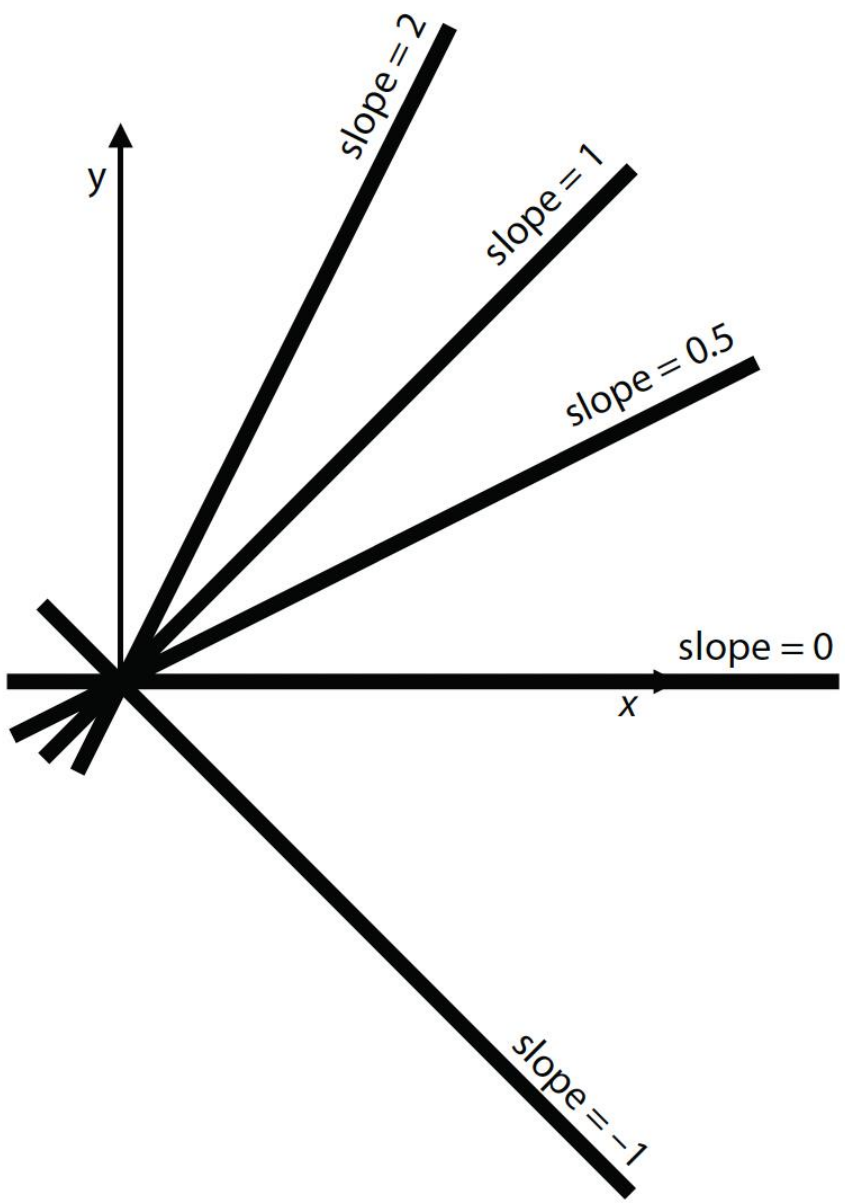
$$y = 0.5x + 2$$

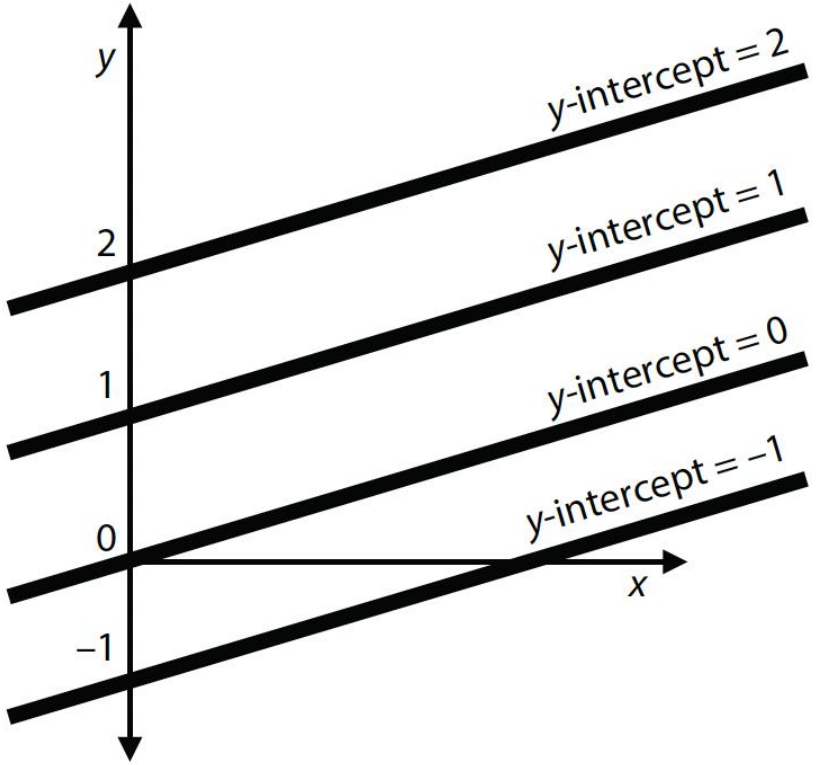


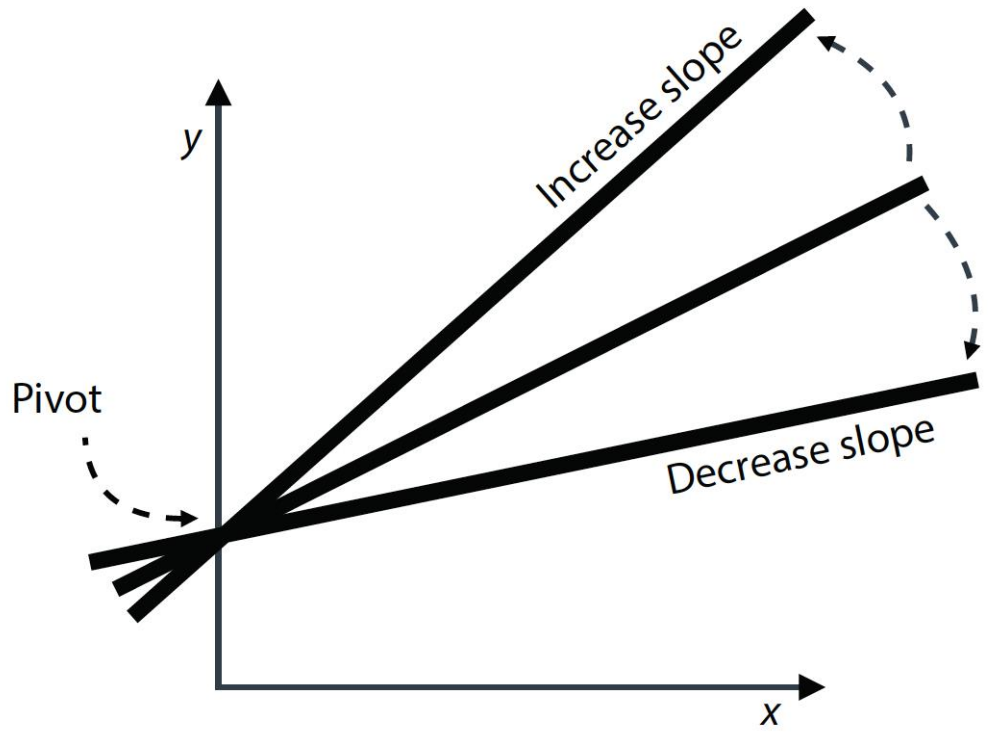
Slope = 0.5



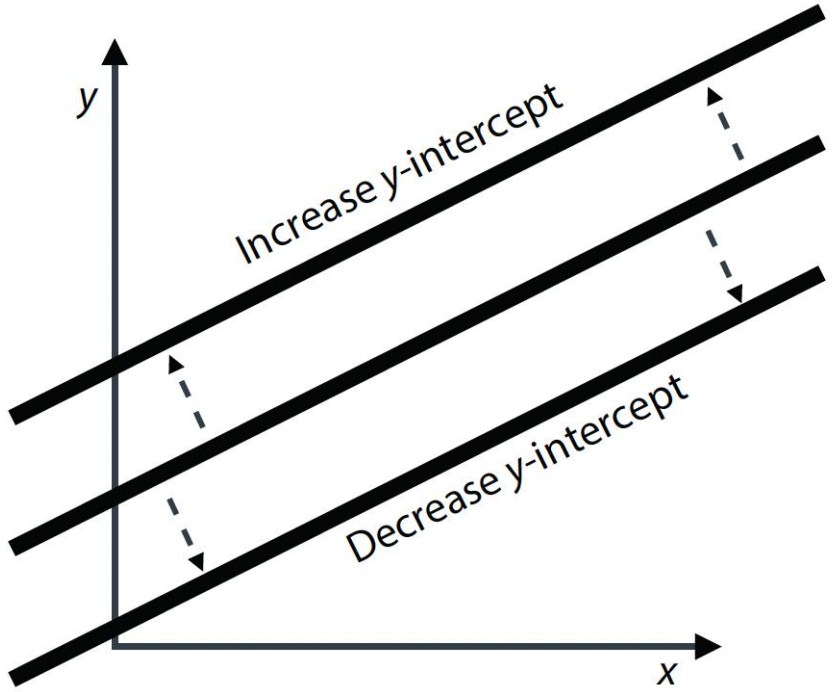
***y*-intercept = 2**


y
slope = 2
slope = 1
slope = 0.5
slope = 0
x
slope = –1

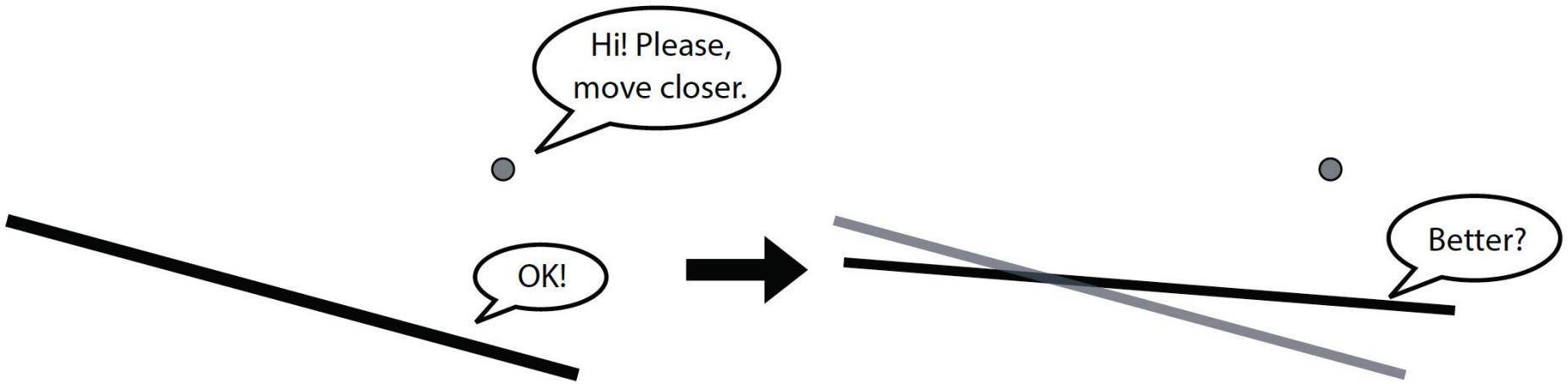

y
y-intercept = 2
y-intercept = 1
y-intercept = 0
y-intercept = –1
2
1
0
–1
x

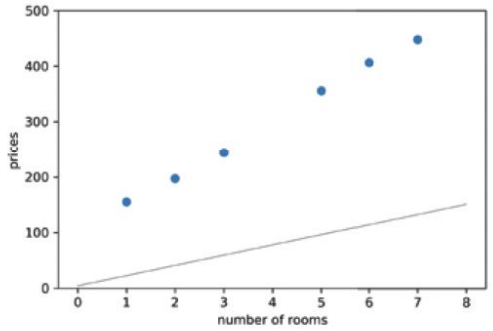


Rotate clockwise and counterclockwise

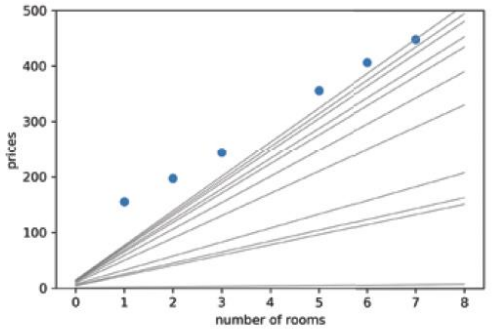


Translate up and down

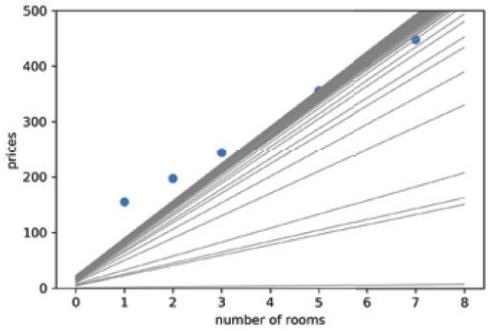

Hi! Please,
move closer.
OK!
Better?



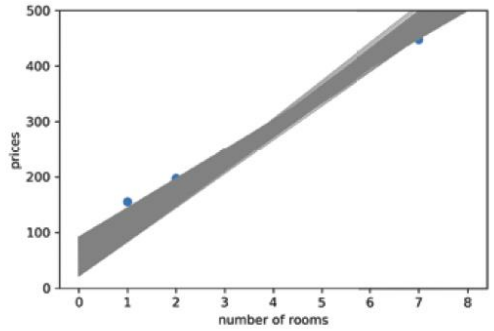
Starting point



Epochs 1–10

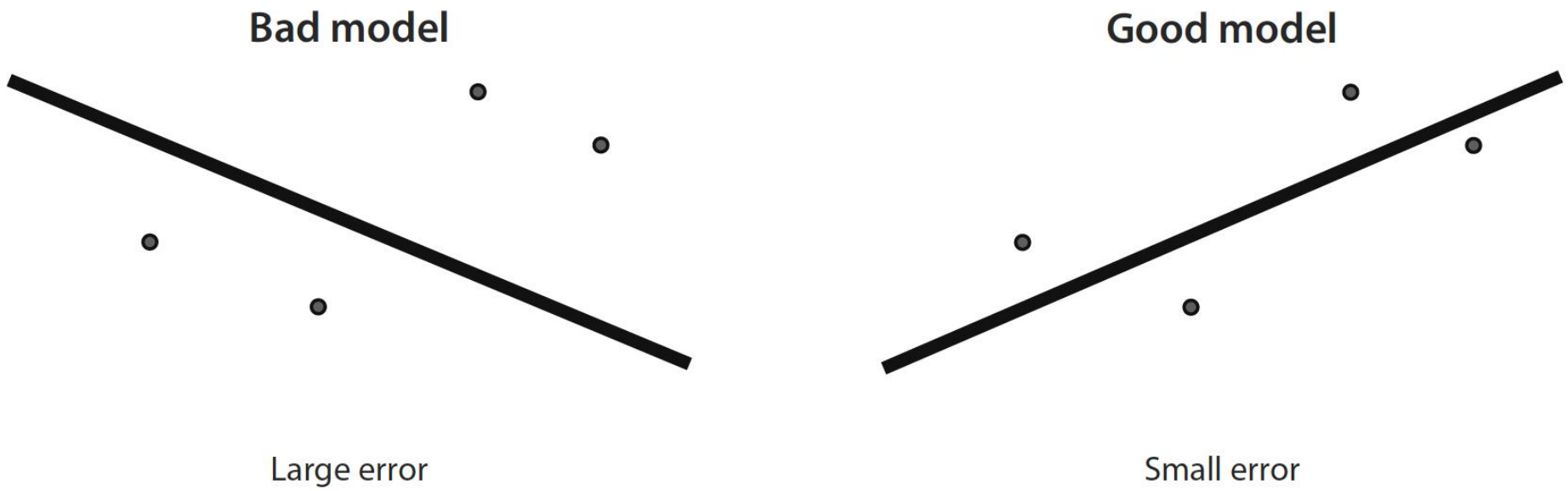


Epochs 1–50

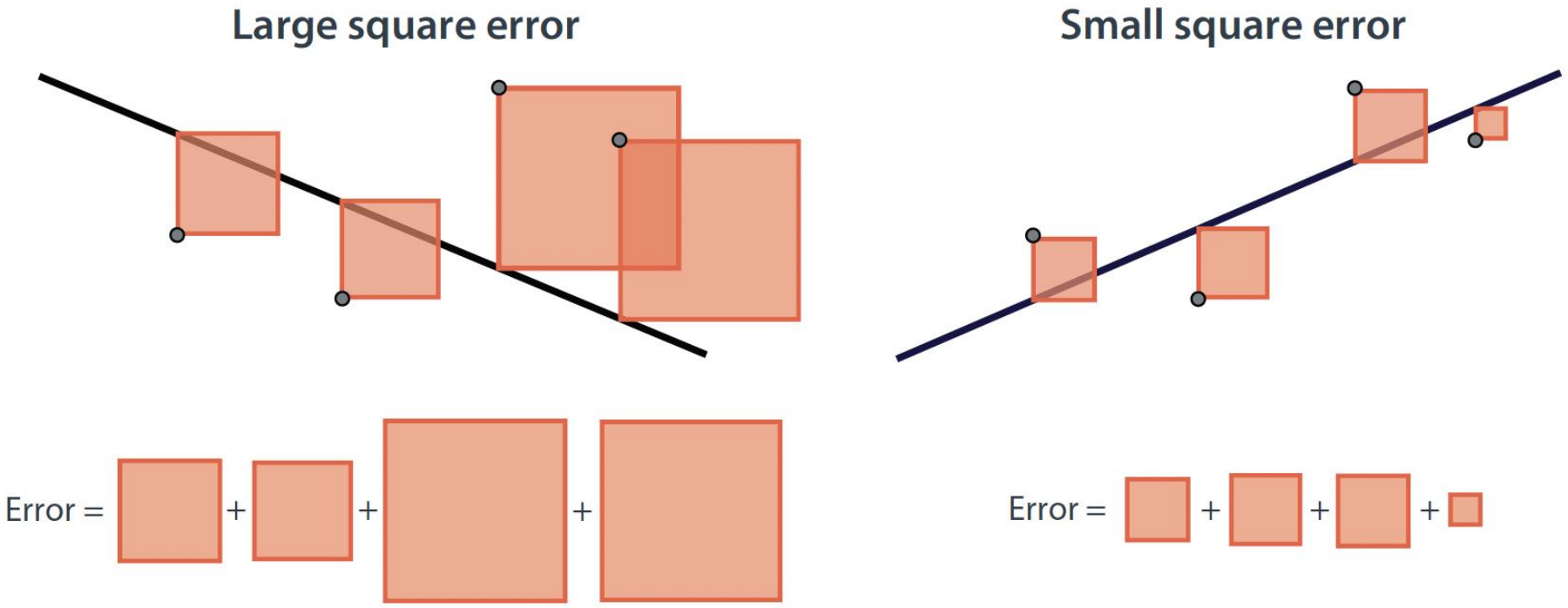


Epochs 51–10,000

How do we measure our results? The error function



The square error: A metric that tells us how good our model is by adding squares of distances



We are on an alien planet,
and we don't know their language!

Dataset:

- Alien 1
  - Mood: Happy
  - Sentence: "*Aack, aack, aack*!"
- Alien 2:
  - Mood: Sad
  - Sentence: "*Beep beep*!"
- Alien 3:
  - Mood: Happy
  - Sentence: "*Aack beep aack*!"
- Alien 4:
  - Mood: Sad
  - Sentence: "*Aack beep beep beep*!"

Data


Aack aack aack!


Beep beep!


Aack beep aack!


Aack beep beep beep!

Prediction

Is this alien happy or sad?

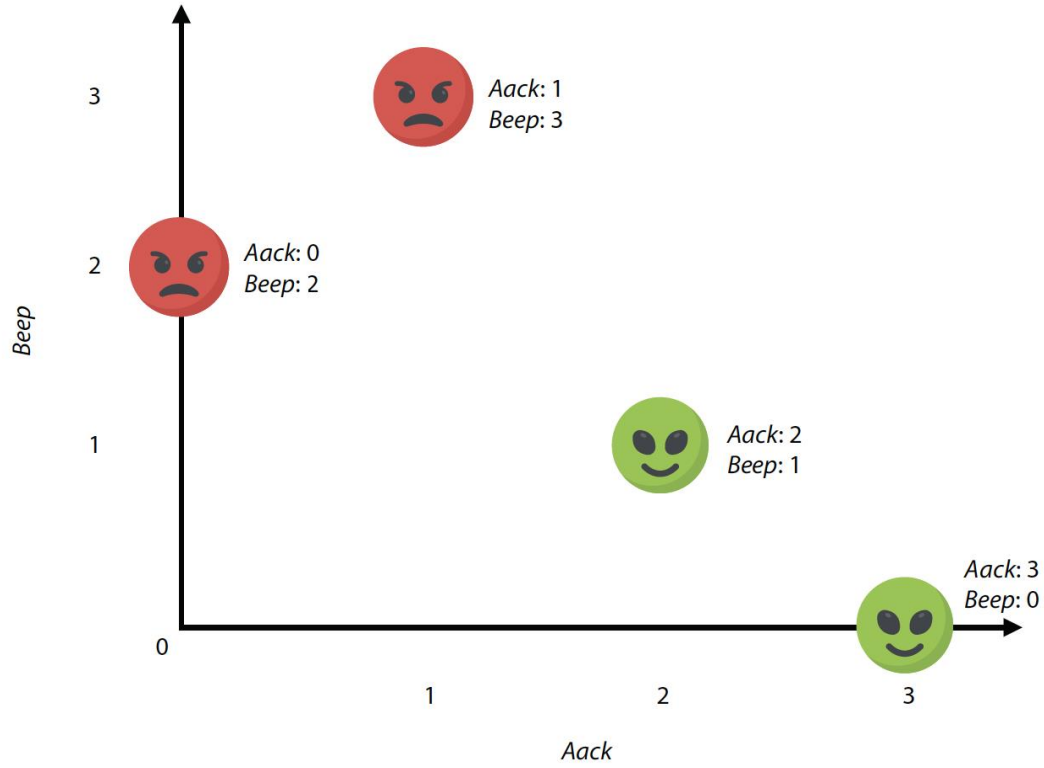

Aack beep aack aack!

Scores:

- *Aack*: 1 point
- *Beep*: –1 points

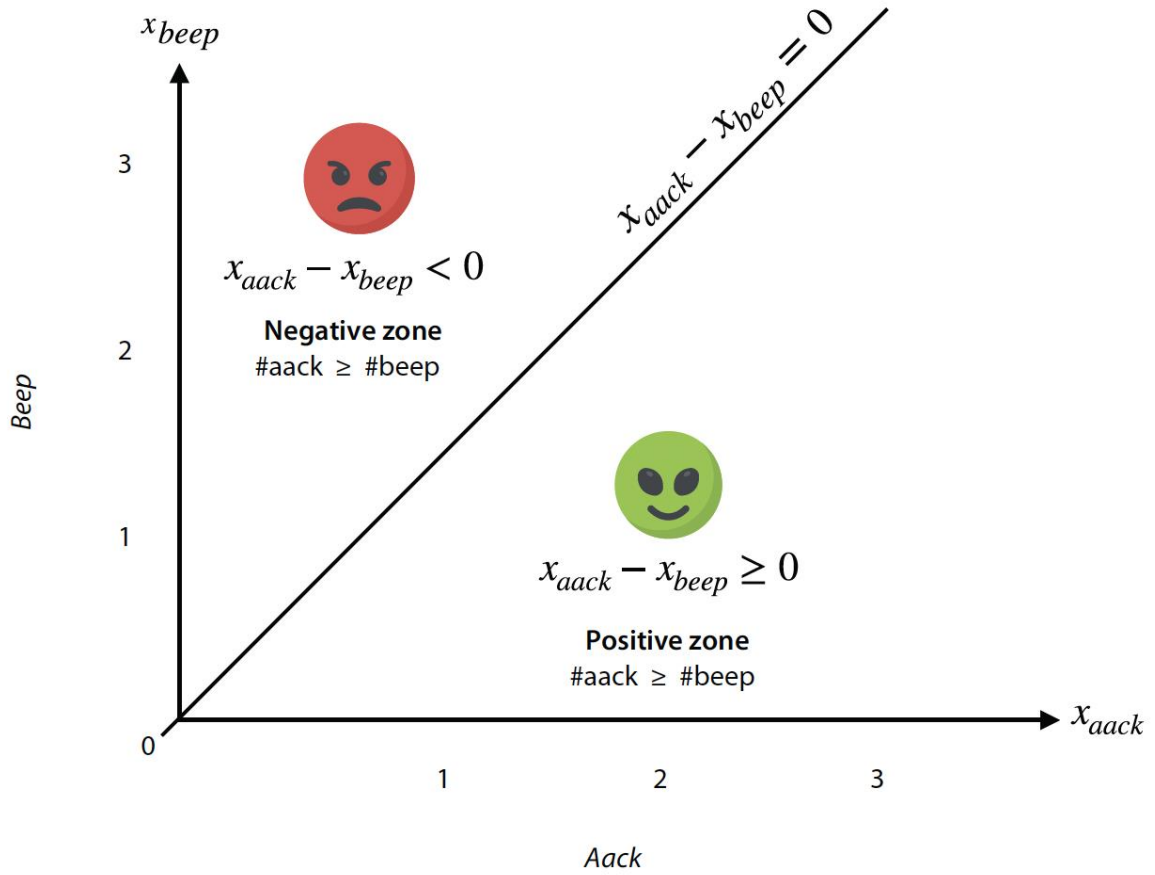
Feature Space

Can you see the classification line already?



Is this alien happy or sad? _____

Aack beep aack aack!



A bit more complex case

Sentence	*Crack*	*Doink*	Mood
Crack!	1	0	Sad
Doink doink!	0	2	Sad
Crack doink!	1	1	Sad
Crack doink crack!	2	1	Sad
Doink crack doink doink!	1	3	Happy
Crack doink doink crack!	2	2	Happy
Doink doink crack crack crack!	3	2	Happy
Crack doink doink crack doink!	2	3	Happy

The idea for this classifier is to count the number of words in a sentence. Notice that the sentences with one, two, or three words are all sad, and the sentences with four and five words are happy. That is the classifier!

Sentiment analysis classifier

Given a sentence, assign the following scores to the words:

Scores:

- *Crack*: one point
- *Doink*: one point

Rule:

Calculate the score of the sentence by adding the scores of all the words on it.

- If the score is four or more, predict that the sentence is happy.
- If the score is three or less, predict that the sentence is sad.

To make it simpler, let's slightly change the rule by using a cutoff of 3.5.

Rule:

Calculate the score of the sentence by adding the scores of all the words on it.

- If the score is 3.5 or more, predict that the sentence is happy.
- If the score is less than 3.5, predict that the sentence is sad.

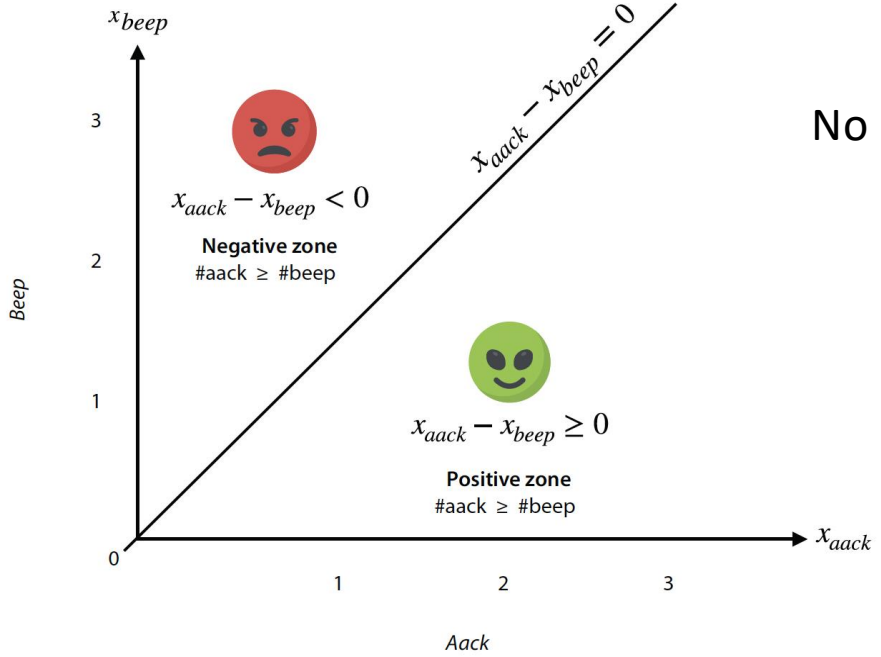
This classifier again corresponds to a line, and that line is illustrated in figure 5.5.

Given a sentence, assign the following weights and bias to the words:

Weights:

- *Crack*: one point
- *Doink*: one point

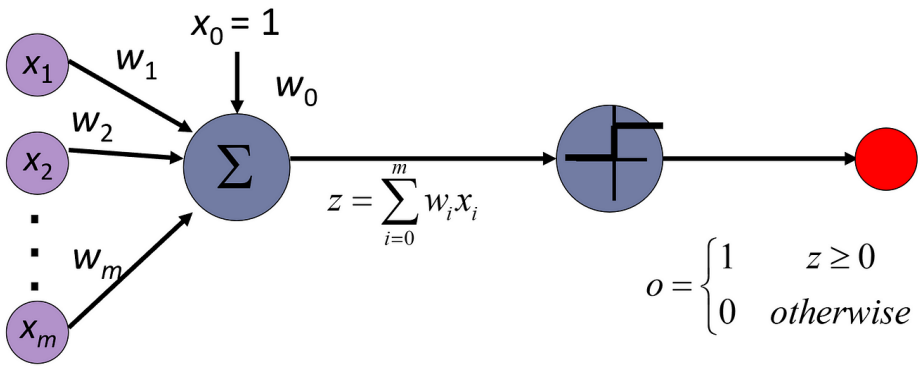
Bias: –3.5 points

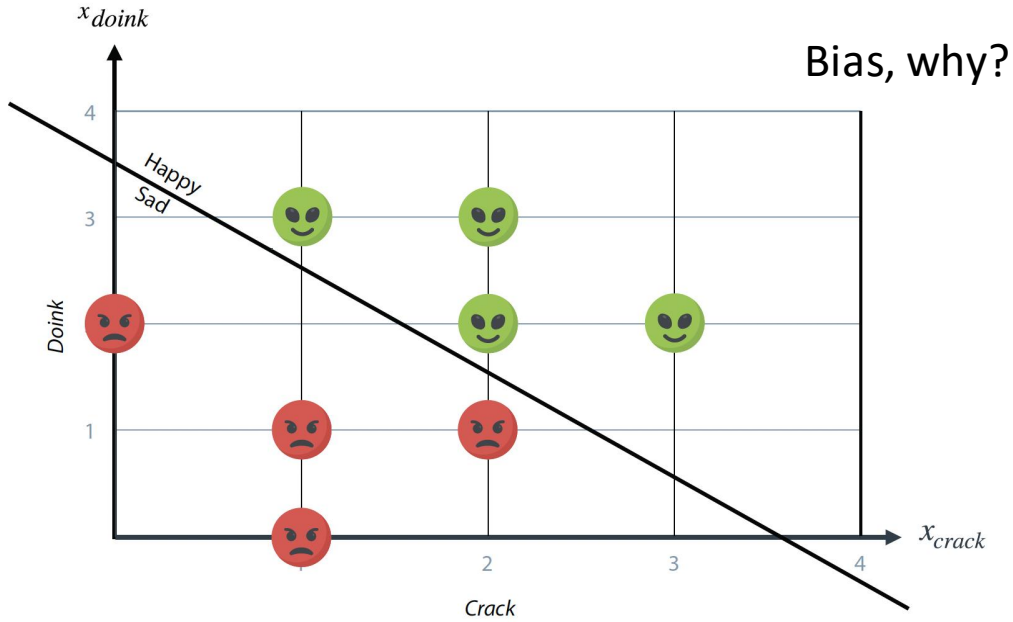


No bias

Positive zone: the area on the plane for which $x_{crack} + x_{doink} - 3.5 \geq 0$

Negative zone: the area on the plane for which $x_{crack} + x_{doink} - 3.5 < 0$





Bias, why?

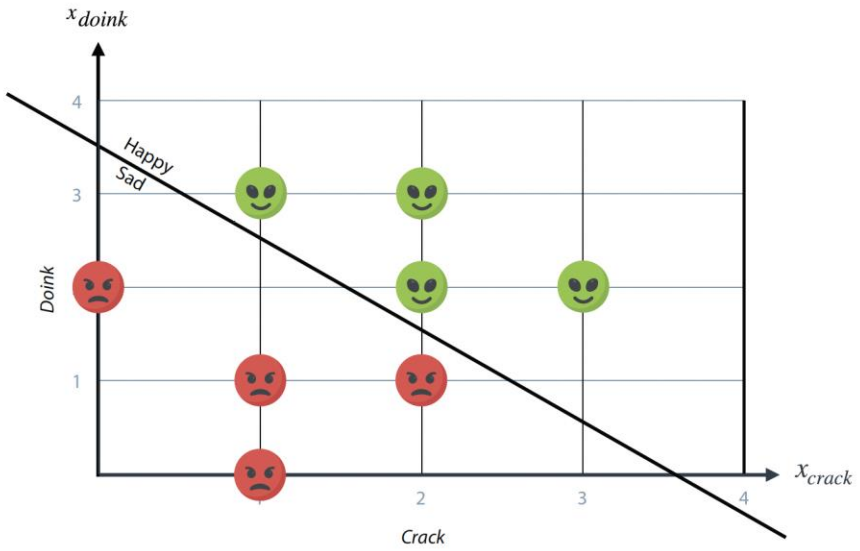
Positive zone: the zone on the plane for which $ax_1 + bx_2 + c \geq 0$

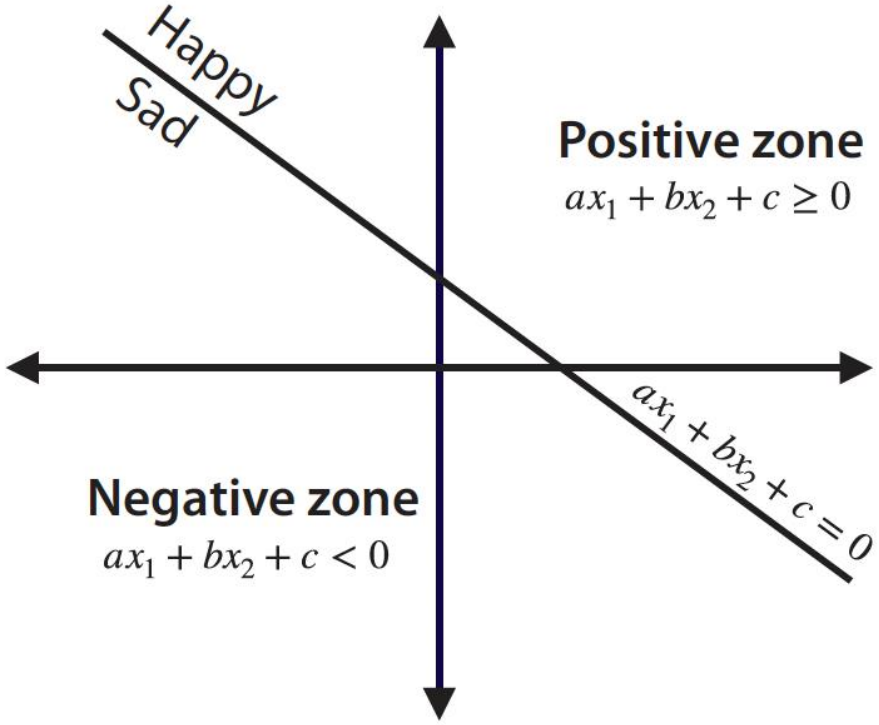
Negative zone: the zone on the plane for which $ax_1 + bx_2 + c < 0$

$$x_1 - x_2 = 0$$

$$x_1 + x_2 - 3.5 = 0$$

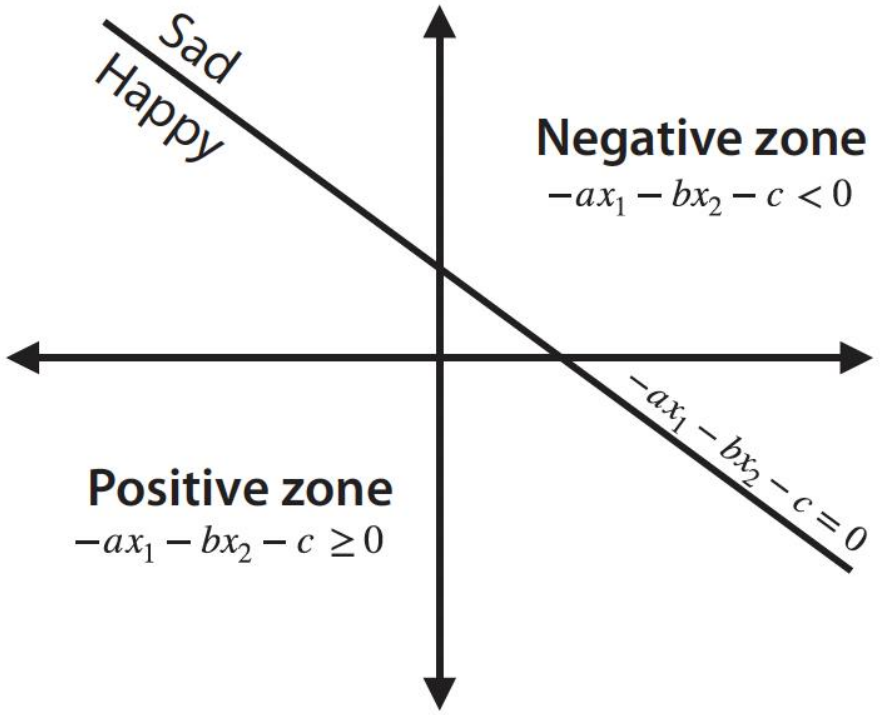
A bit of M





Classifier with equation

$$ax_1 + bx_2 + c = 0$$



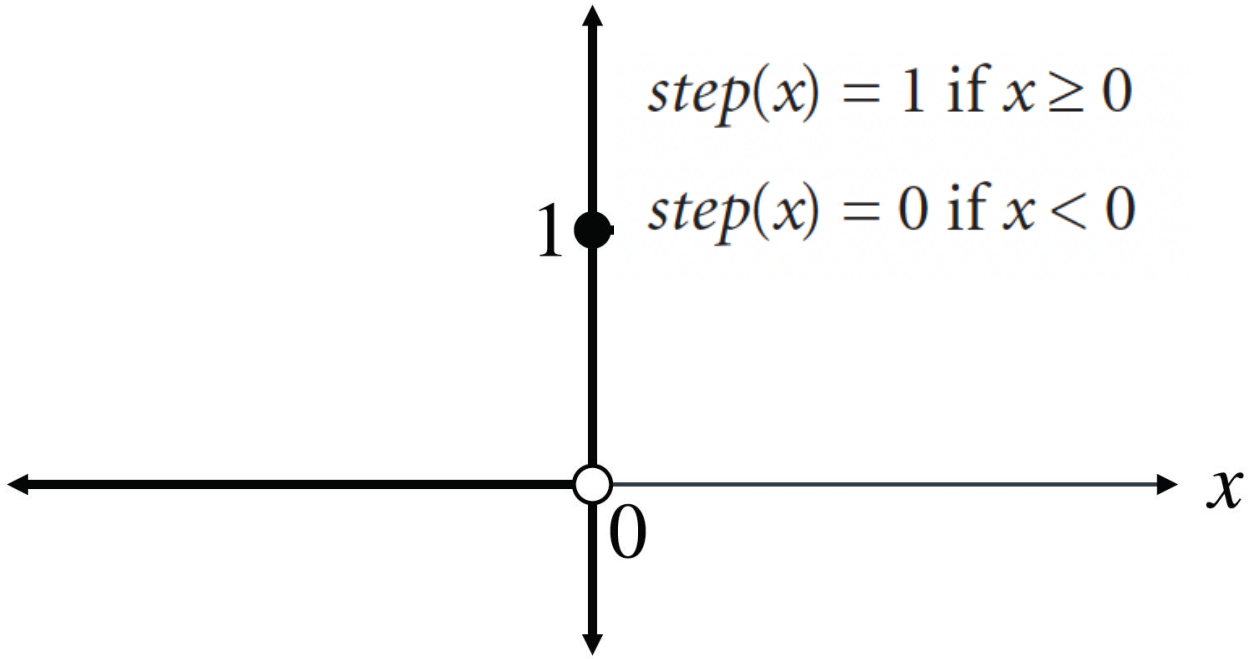
Classifier with equation

$$-ax_1 - bx_2 - c = 0$$

Activation Function

The perceptron classified we built is based on an if statement, we say the alien is happy or sad if the score is higher or lower than zero.

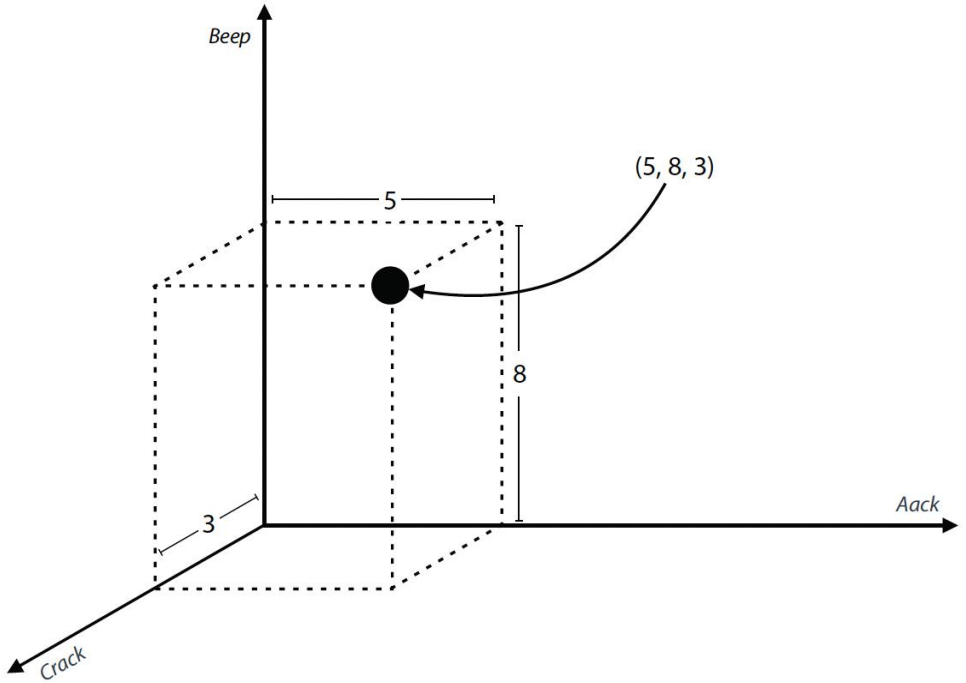
This last part can be made by the activation function



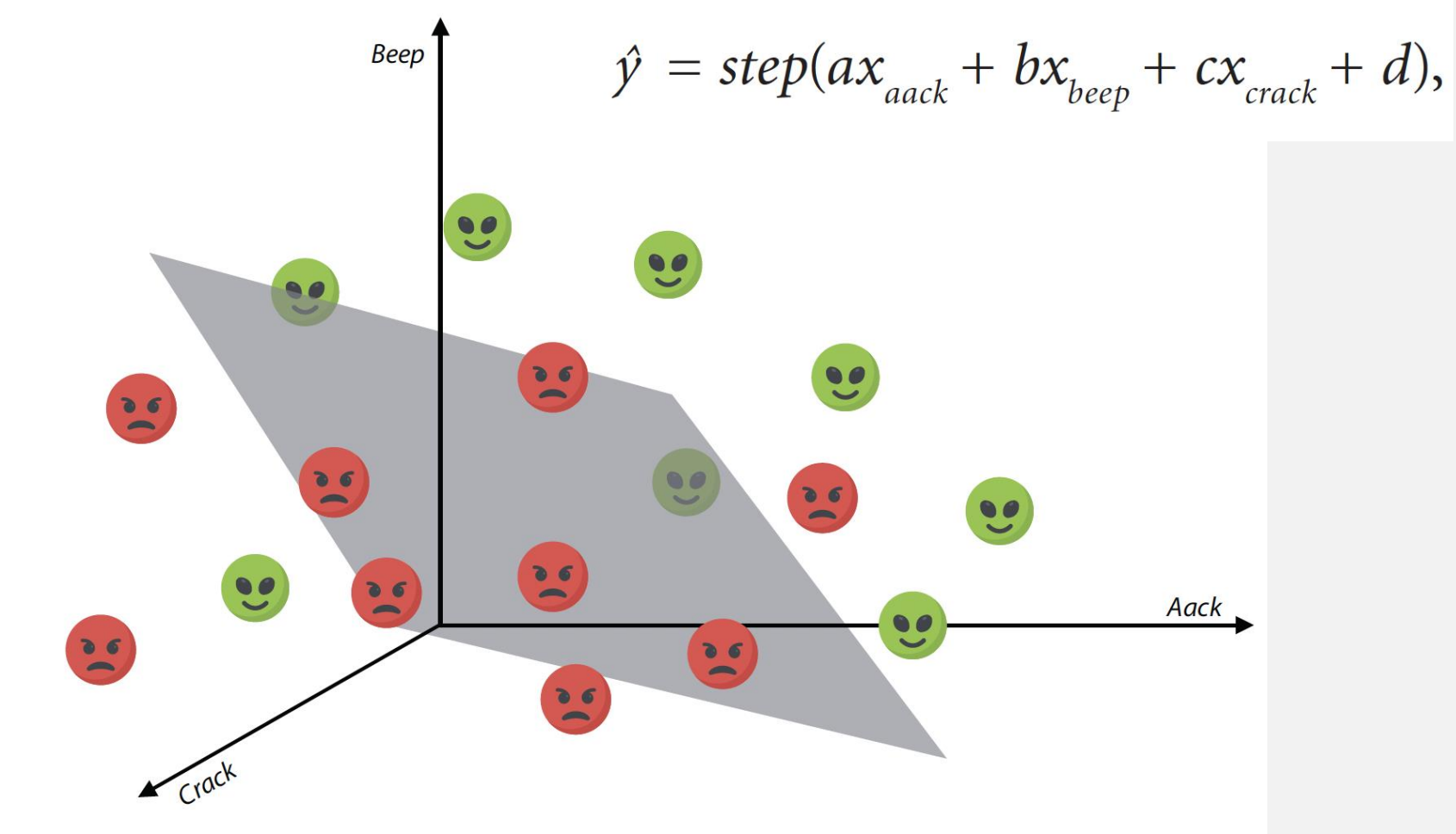
Perceptron Model

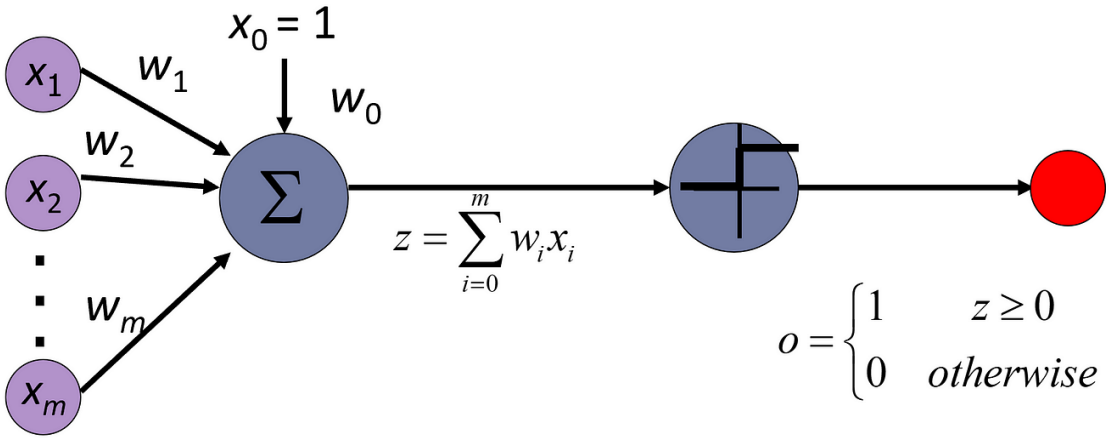
$$\hat{y} = step(ax_1 + bx_2 + c).$$

Perceptron Model, more dimensions



$$\hat{y} = step(ax_{aack} + bx_{beep} + cx_{crack} + d),$$


$\hat{y} = step(ax_{aack} + bx_{beep} + cx_{crack} + d),$
Beep
Aack
Crack



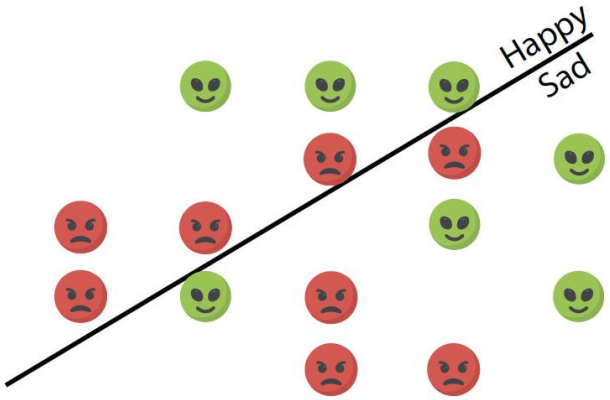
$$\hat{y}_i = \text{step}(w_1 x_1^{(i)} + w_2 x_2^{(i)} + \cdots + w_n x_n^{(i)} + b).$$

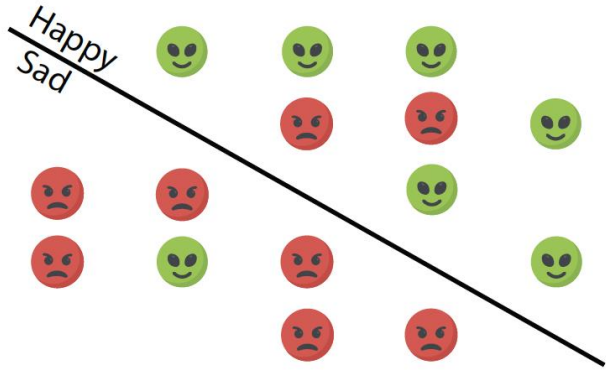
More generally

Playground

$$\text{Precision} = \frac{tp}{tp + fp}$$

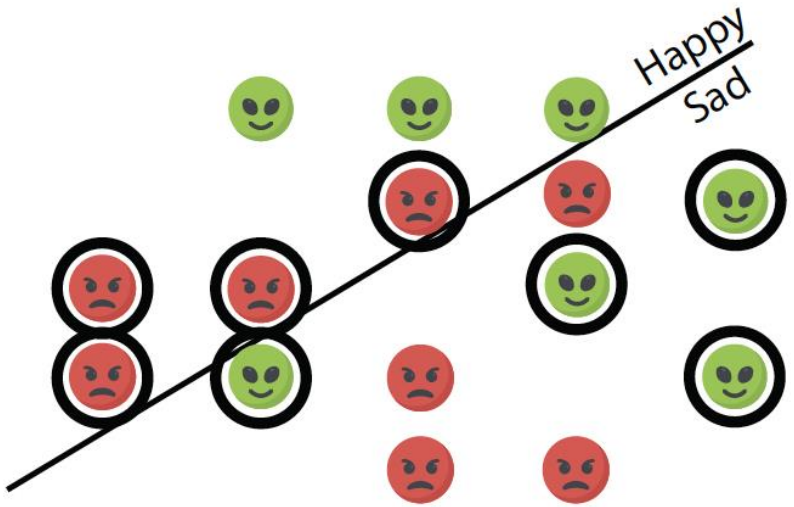
$$\text{Recall} = \frac{tp}{tp + fn}$$



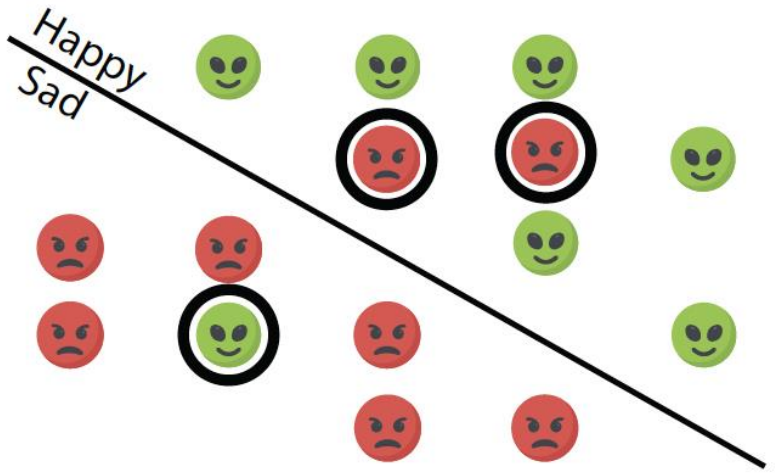


How do we determine whether a classifier is good or bad? The error function

Model evaluation (loss function)

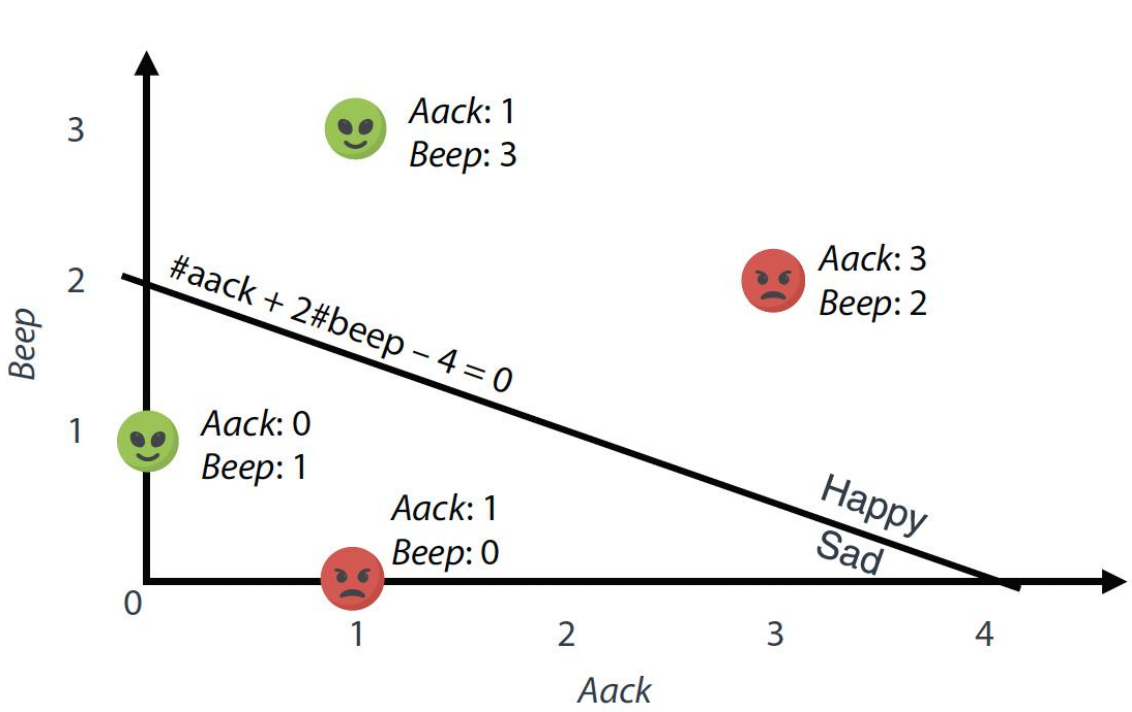


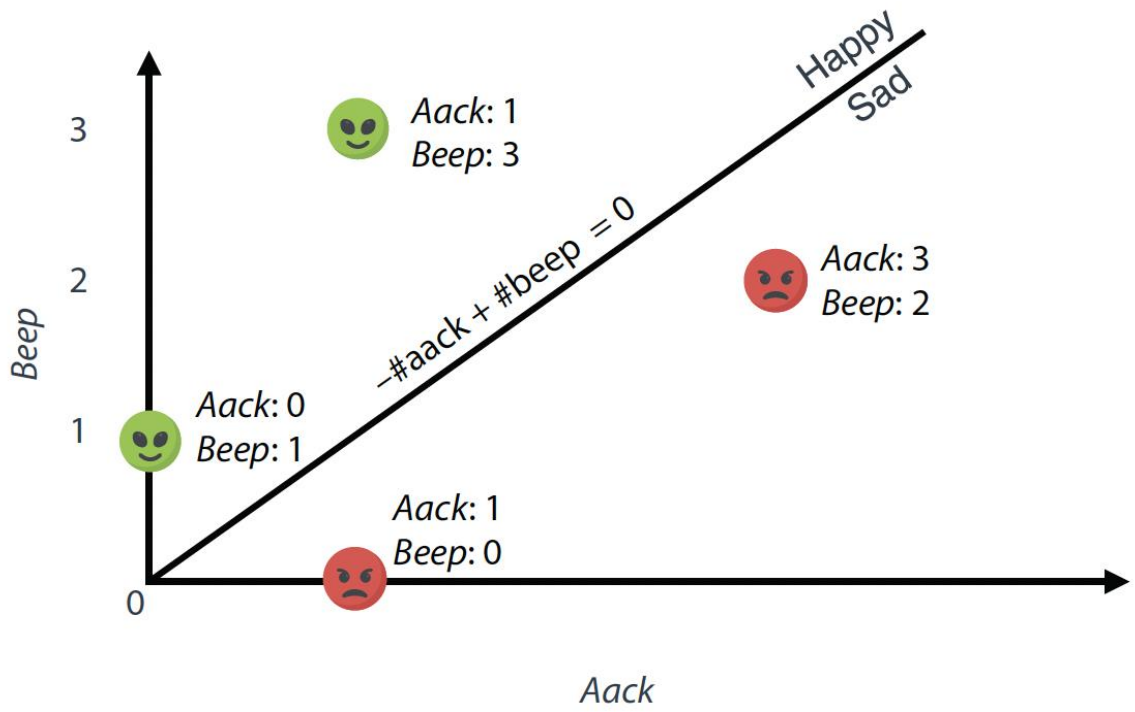
Bad classifier
Error: 8



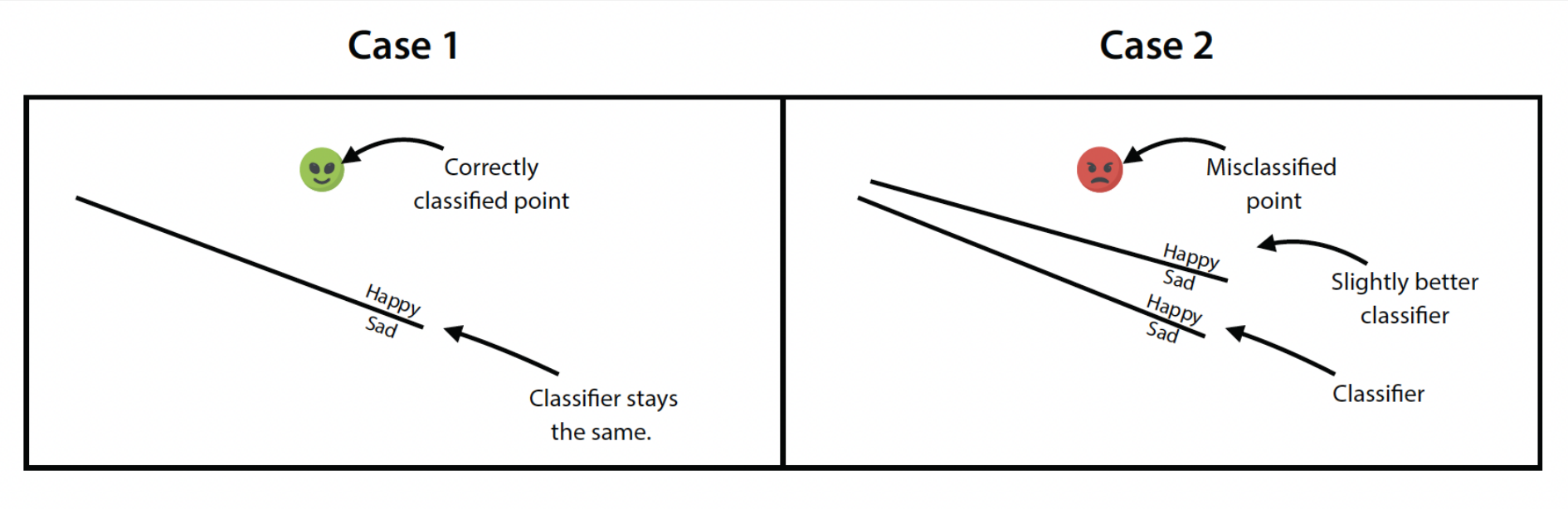
Good classifier
Error: 3

Model evaluation (loss function)

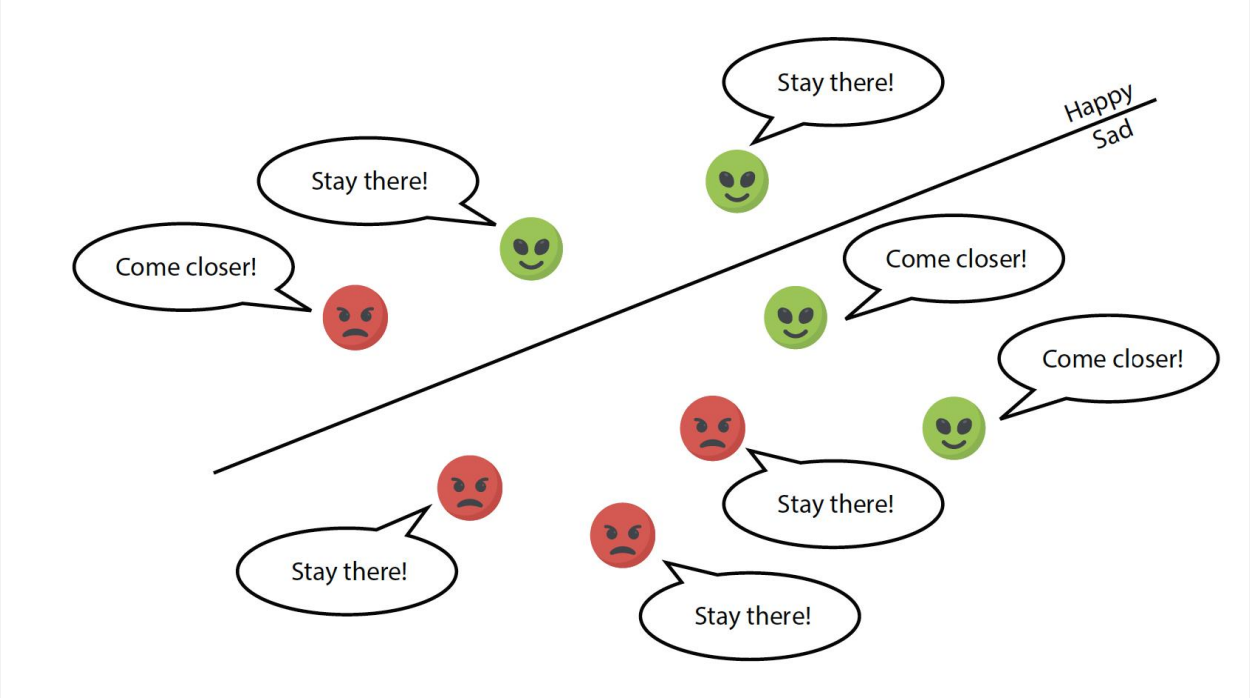




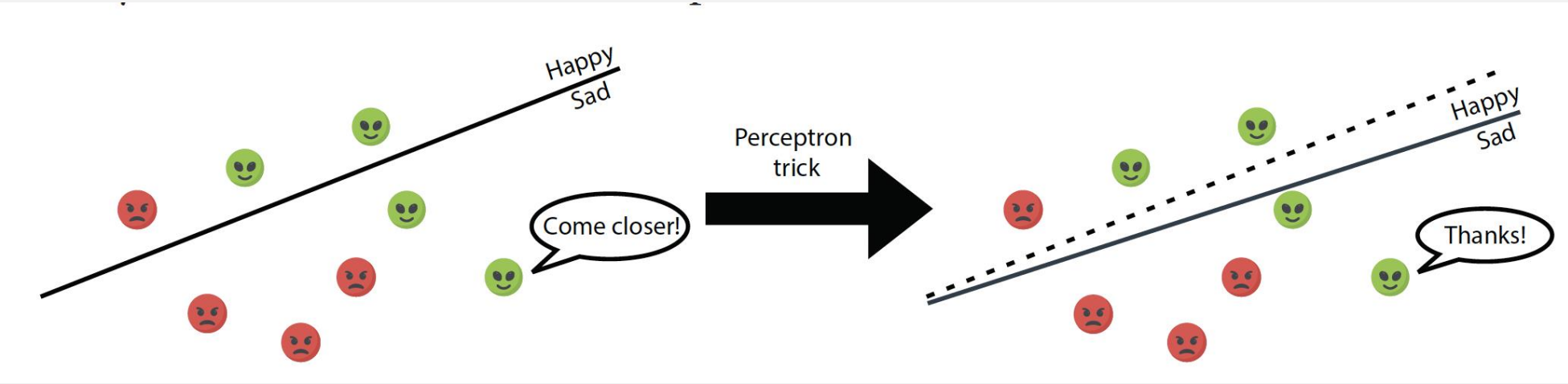
How to find a good classifier?
The perceptron algorithm



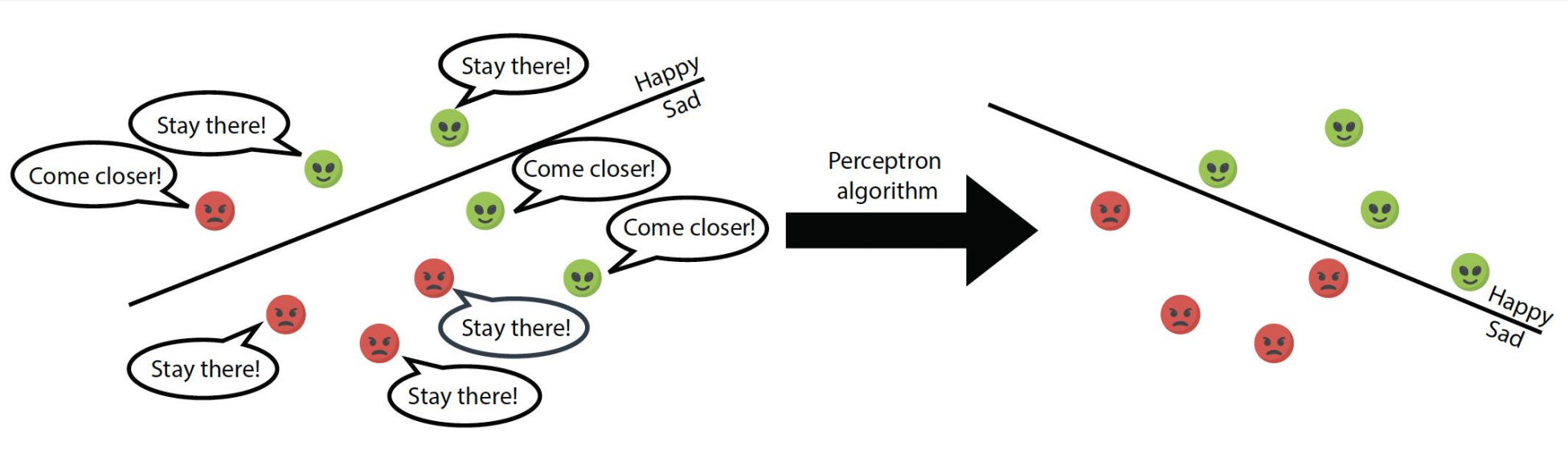
Repeating the perceptron trick many times: The perceptron algorithm

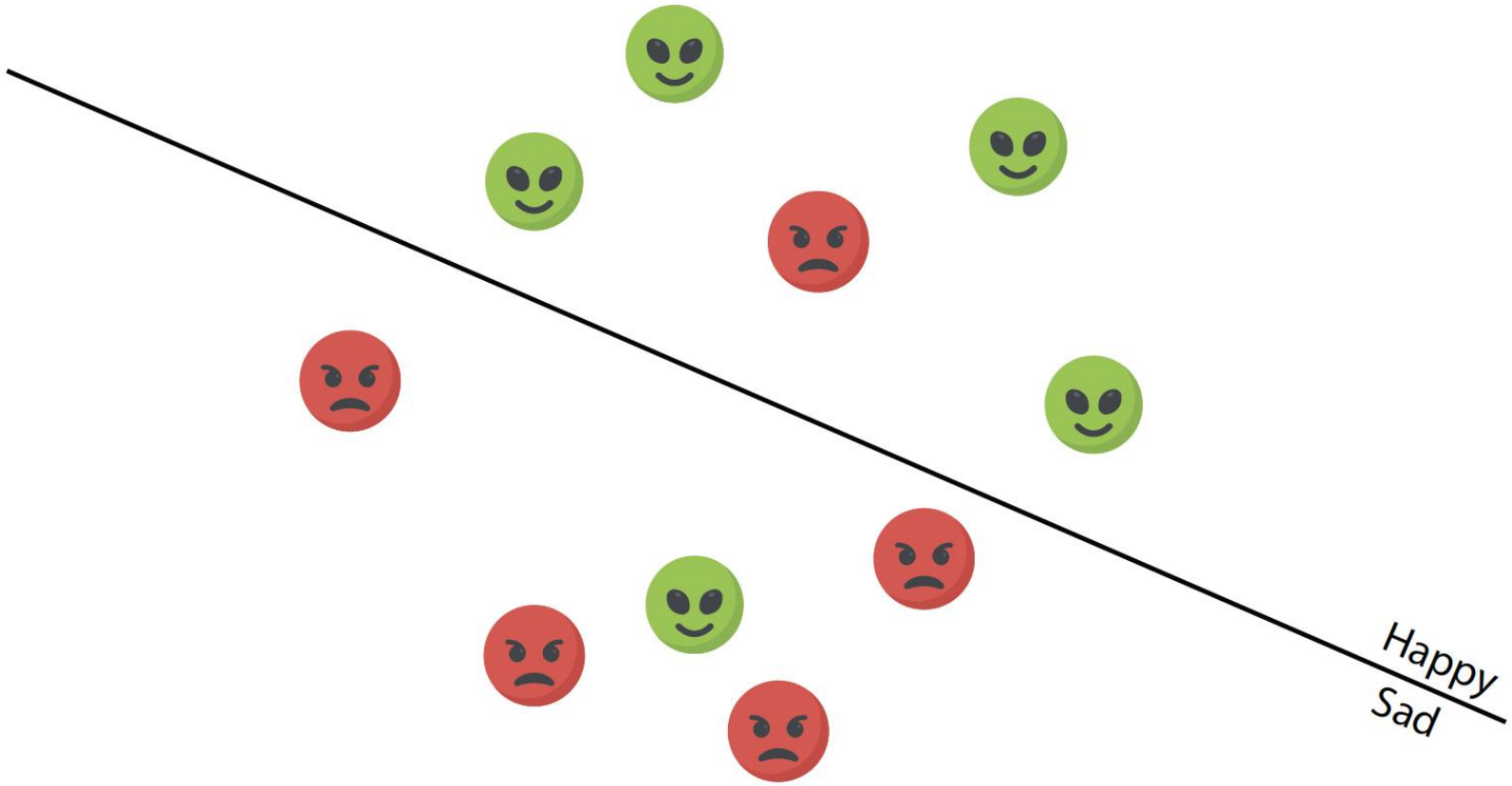


Repeating the perceptron trick many times: The perceptron algorithm



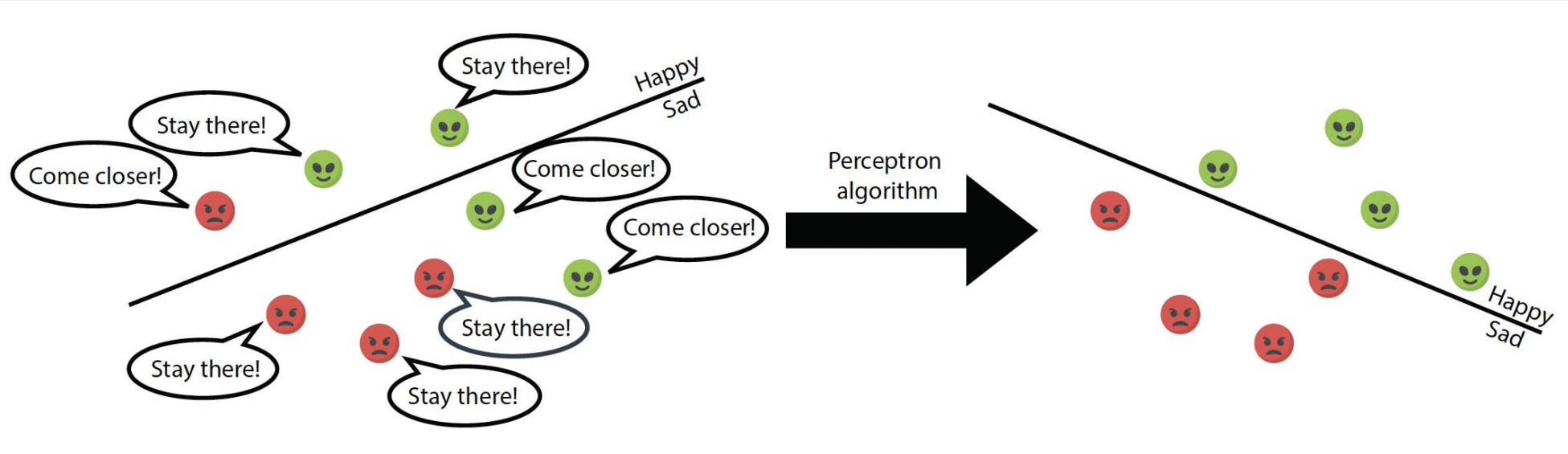
Repeating the perceptron trick many times: The perceptron algorithm




Happy
Sad

Perceptron Learning Rule

Repeating the perceptron trick many times: The perceptron algorithm



Math behind - Learning the Correct Weights

To learn a set of hetero-associations, we desire a weight vector, $\boldsymbol{w} = (w_1, w_2, \dots, w_N)$ that correctly maps an input pattern, $\boldsymbol{x}^P = (x_1^P, x_2^P, \dots, x_N^P)$, to a target output, $\boldsymbol{t}^P = (t_1^P, t_2^P, \dots, t_N^P)$.

Algorithm might look like this:

```
Repeat
        for each association, x^P, t^P
                calculate y^P = f(x^P, w)
                if (y^P ≠ t^P)
                        update w = w + Δw
                end
        end
Until (y^P = t^P)
```

Learning the Correct Weights

To learn a set of hetero-associations, we desire a weight vector, $\boldsymbol{w} = (w_1, w_2, \dots, w_N)$ that correctly maps an input pattern, $\boldsymbol{x}^P = (x_1^P, x_2^P, \dots, x_N^P)$, to a target output, $\boldsymbol{t}^P = (t_1^P, t_2^P, \dots, t_N^P)$.

Algorithm might look like this:

```
Repeat
        for each association, x^P, t^P
                calculate y^P = f(x^P, w)
                if (y^P ≠ t^P)
                        update w = w + Δw
                end
        end
Until (y^P = t^P)
```

How do we calculate $\Delta \boldsymbol{w}$?

Perceptron Learning Rule (Rosenblatt 1962)

We need to consider how each of the weights, w_i contributed to the weighted sum, $\sum w_i x_i$

If $x_i^P = 0$, w_i does not contribute to $\sum w_i x_i$

If $y^P = t^P$, then we should do nothing

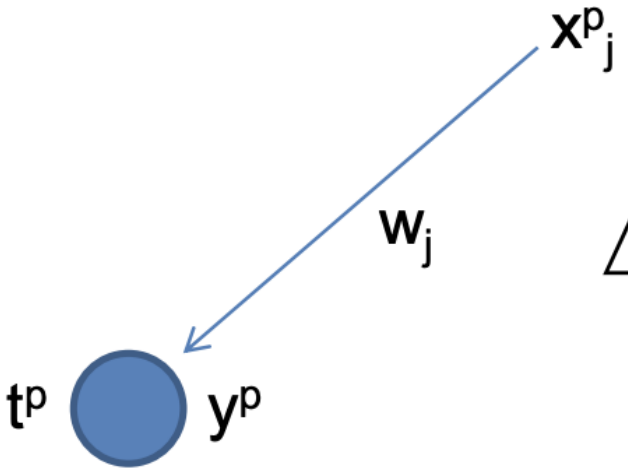
If $y^P = 1$ and $t^P = 0$, then $\sum w_i x_i$ is too large, so we decrease all w_i where $x_i = 1$

If $y^P = 0$ and $t^P = 1$, then $\sum w_i x_i$ is too large, so we increase all w_i where $x_i = 1$

$$\Delta w_i = \eta(t^P - y^P)x_i^P$$

η is called the "learning rate", it scales the update

Features of the Perceptron Rule



$$\Delta w_i = \eta (t^p - y^p) x_i^p$$

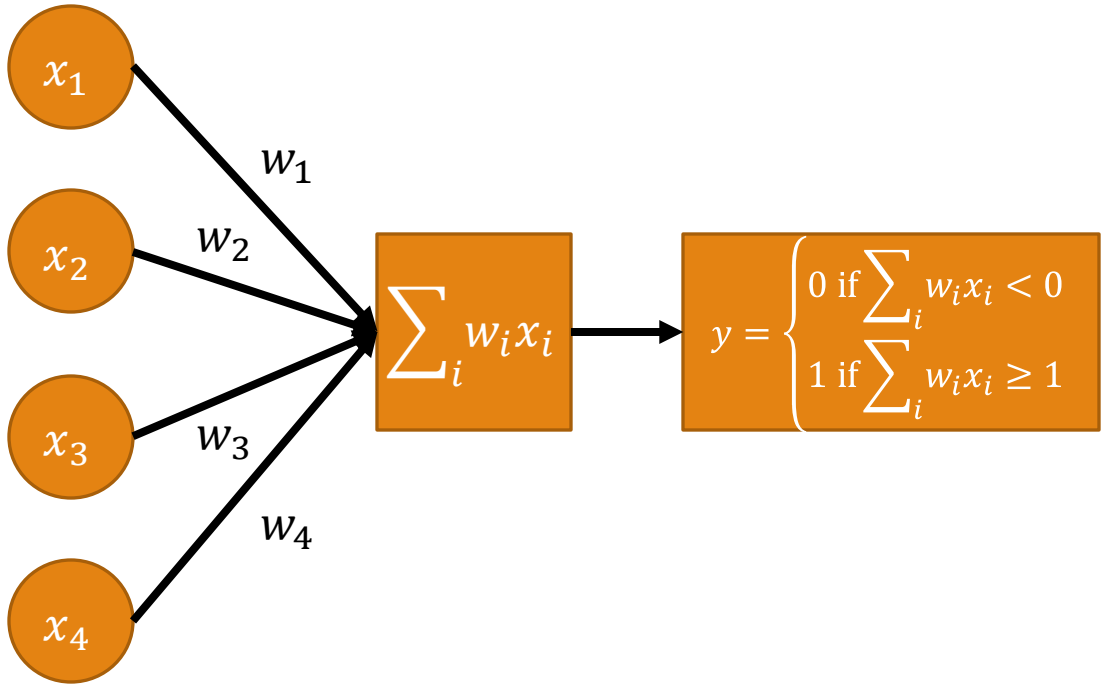
The rule is local: it uses information that is locally available to the weight

It is an error correction rule, so it is stable when there are no errors

If a set of weights exist that separate the pattern—i.e. the task is linearly separable—then the rule is **guaranteed** to find the correct weights: Perceptron Convergence Theorem

Example

Consider a linear perceptron with four inputs, $x = (x_1, x_2, x_3, x_4) = (0.3, 1, 0, 0.7)$, weights, $w = (w_1, w_2, w_3, w_4) = (1, 0.4, 0.6, 0)$, and a threshold, $\theta = 0.5$ The expected output is, $t = 0$

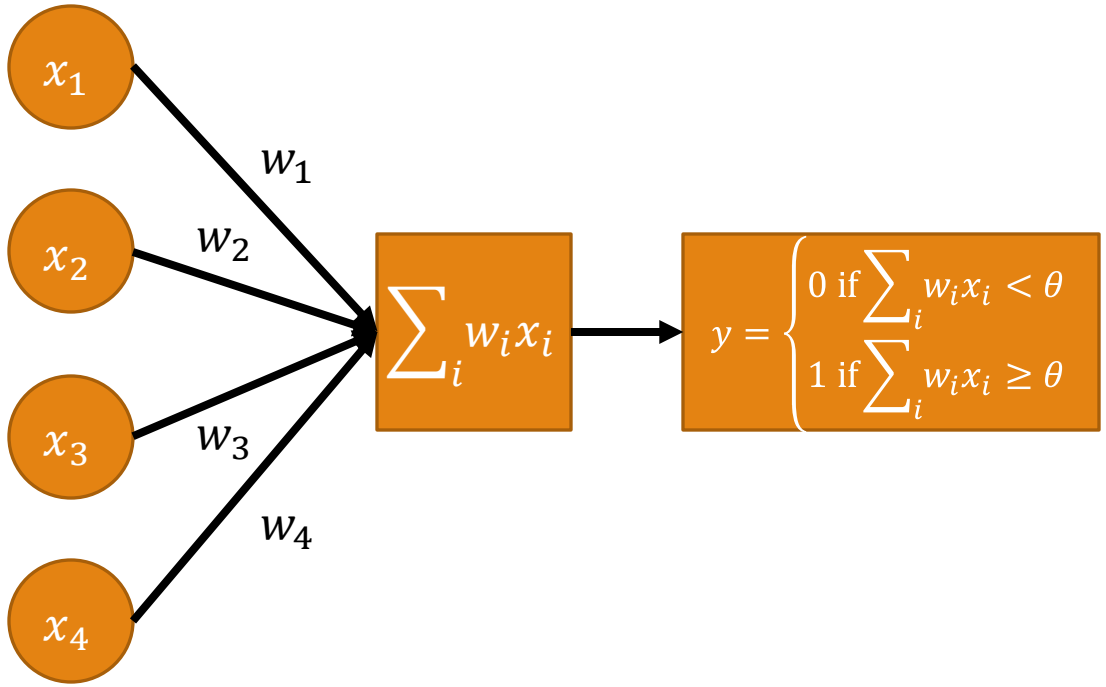


Example

Consider a linear perceptron with four inputs, $x = (x_1, x_2, x_3, x_4) = (0.3, 1, 0, 0.7)$, weights, $w = (w_1, w_2, w_3, w_4) = (1, 0.4, 0.6, 0)$, and a threshold, $\theta = 0.5$ The expected output is, $t = 0$

The weighted sum of the inputs is:

$$\sum_i w_i x_i = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4$$
$$= 1 \cdot 0.3 + 0.4 \cdot 1 + 0.6 \cdot 0 + 0 \cdot 0.7$$
$$= 0.7$$



Example

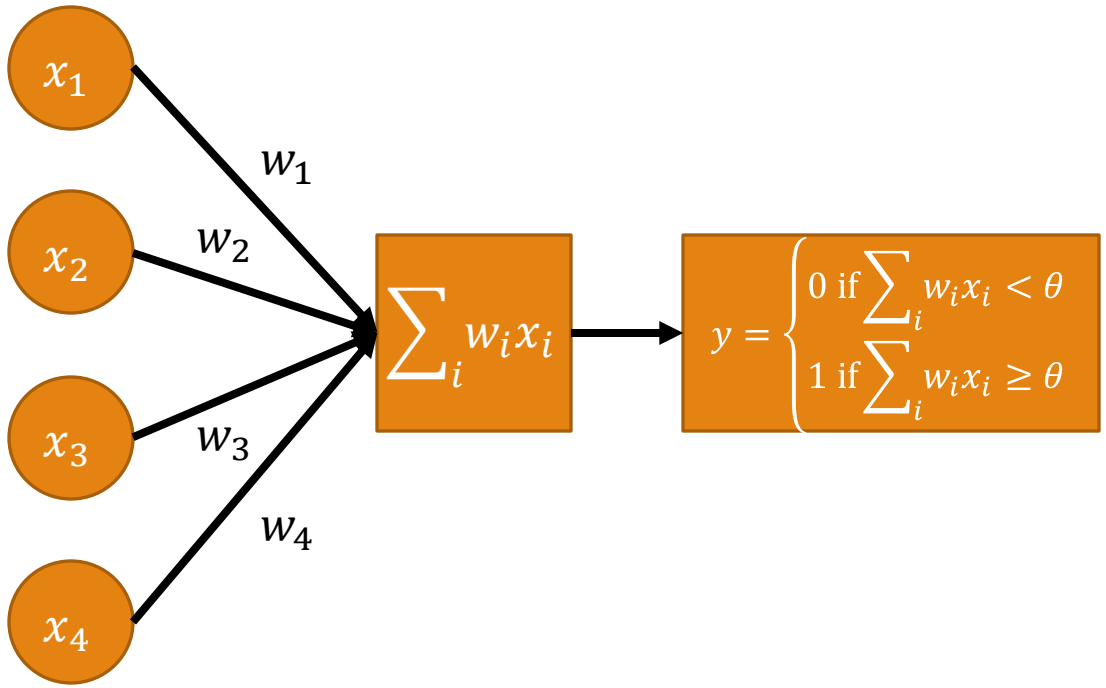
Consider a linear perceptron with four inputs, $x = (x_1, x_2, x_3, x_4) = (0.3, 1, 0, 0.7)$, weights, $w = (w_1, w_2, w_3, w_4) = (1, 0.4, 0.6, 0)$, and a threshold, $\theta = 0.5$. The expected output is, $t = 0$

The weighted sum of the inputs is:

$$\sum_i w_i x_i = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4$$
$$= 1 \cdot 0.3 + 0.4 \cdot 1 + 0.6 \cdot 0 + 0 \cdot 0.7$$
$$= 0.7 > \theta$$

The output is then:

$$y = 1$$

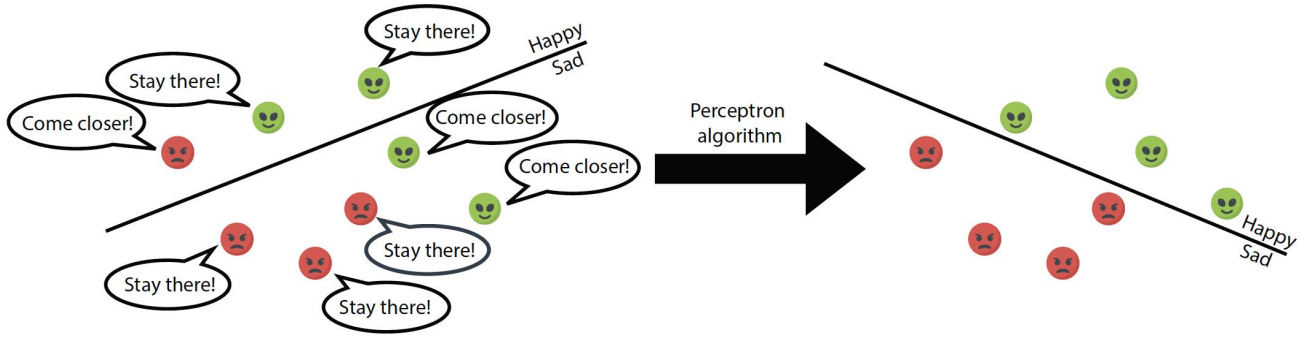


Example – Weight Update

Calculate Δw_i, for a learning rate, $\eta = 0.1$:

$$\Delta w_1 = 0.1 \cdot (0 - 1) \cdot 0.3 = -0.03$$
$$\Delta w_2 = 0.1 \cdot (0 - 1) \cdot 1 = -0.1$$
$$\Delta w_3 = 0.1 \cdot (0 - 1) \cdot 0 = -0$$
$$\Delta w_4 = 0.1 \cdot (0 - 1) \cdot 0.7 = -0.07$$

$$\Delta w_i = \eta(t^P - y^P)x_i^P$$



Example – Weight Update

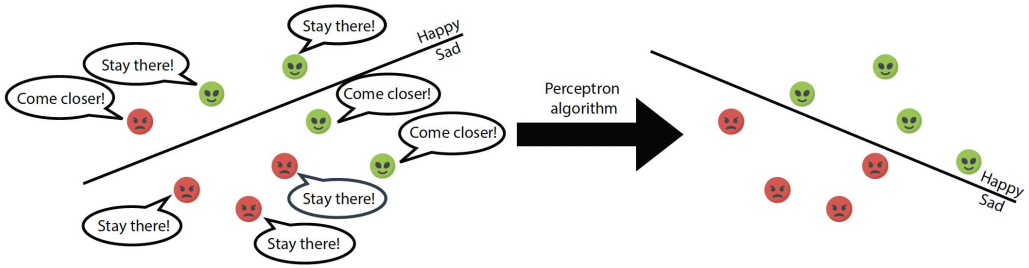
Calculate Δw_i, for a learning rate, $\eta = 0.1$:

$$\Delta w_1 = 0.1 \cdot (0 - 1) \cdot 0.3 = -0.03$$
$$\Delta w_2 = 0.1 \cdot (0 - 1) \cdot 1 = -0.1$$
$$\Delta w_3 = 0.1 \cdot (0 - 1) \cdot 0 = 0$$
$$\Delta w_4 = 0.1 \cdot (0 - 1) \cdot 0.7 = -0.07$$

Compute the new weights:

$$w = (w_1, w_2, w_3, w_4) = (0.97, 0.3, 0.6, -0.07)$$

$$\Delta w_i = \eta(t^P - y^P)x_i^P$$

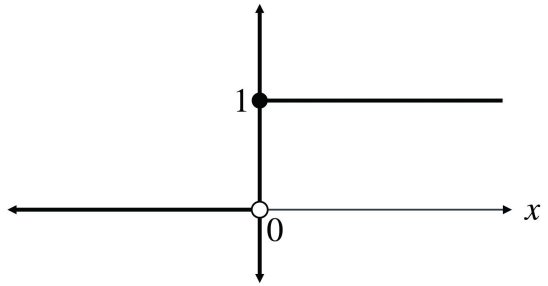


Consider the following dataset with 10 observations. Each observation consists of two binary input features X_1, X_2 and a binary target variable Y.

Observation	X_1	X_2	Y
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	1
5	0	0	0
6	0	1	0
7	1	0	1
8	1	1	0
9	0	0	1
10	1	1	1

Initial Settings:

- Initial weights: $w_1 = 0.5, w_2 = -0.5$
- Bias: $b = 0.0$
- Learning rate: $\alpha = 0.1$



- Select three observations at random from the dataset.
- For each selected observation, calculate the perceptron output. If the prediction is incorrect, update the weights and bias using the perceptron learning rule.
- Using the updated weights and bias from Task 1, calculate the perceptron output for the remaining observations.
- Identify which observations are correctly classified and which are not.

$$\hat{y}_i = \text{step}(w_1 x_1^{(i)} + w_2 x_2^{(i)} + \cdots + w_n x_n^{(i)} + b).$$

$$\Delta w_i = \eta(t^P - y^P)x_i^P$$

$step(x) = 1$ if $x \geq 0$

$step(x) = 0$ if $x < 0$

[Tensor Flow Playground]

Extras

Example – Weight Update

Calculate Δw_i, for a learning rate, $\eta = 0.1$:

$$\Delta w_1 = 0.1 \cdot (0 - 1) \cdot 0.3 = -0.03$$
$$\Delta w_2 = 0.1 \cdot (0 - 1) \cdot 1 = -0.1$$
$$\Delta w_3 = 0.1 \cdot (0 - 1) \cdot 0 = 0$$
$$\Delta w_4 = 0.1 \cdot (0 - 1) \cdot 0.7 = -0.07$$

Compute the new weights:

$$w = (w_1, w_2, w_3, w_4) = (0.97, 0.3, 0.6, -0.07)$$

Compute the new output:

$$y = 1$$

$$\Delta w_i = \eta(t^P - y^P)x_i^P$$

Example – Continue Updates

Since $y \neq t$, we continue updating the weights. As Δw_i only depends on x_i, the weight updates are the same in each step—this would not be the case if we had multiple input samples

Updating the weights once more gives:
$w = (w_1, w_2, w_3, w_4) = (0.94, 0.2, 0.6, -0.14)$

Example – Continue Updates

Since $y \neq t$, we continue updating the weights. As Δw_i only depends on x_i, the weight updates are the same in each step—this would not be the case if we had multiple input samples

Updating the weights once more gives:

$$w = (w_1, w_2, w_3, w_4) = (0.94, 0.2, 0.6, -0.14)$$

The new weighted sum is then:

$$\sum_i w_i x_i$$
$$= 0.94 \cdot 0.3 + 0.2 \cdot 1 + 0.6 \cdot 0 - 0.14 \cdot 0.7$$
$$= 0.384 < \theta$$

Example – Continue Updates

Since $y \neq t$, we continue updating the weights. As Δw_i only depends on x_i, the weight updates are the same in each step—this would not be the case if we had multiple input samples

Updating the weights once more gives:

$$w = (w_1, w_2, w_3, w_4) = (0.94, 0.2, 0.6, -0.14)$$

The new weighted sum is then:

$$\sum_i w_i x_i = 0.94 \cdot 0.3 + 0.2 \cdot 1 + 0.6 \cdot 0 - 0.14 \cdot 0.7$$
$$= 0.384 < \theta$$

And thus, the output is:

$$y = 0 = t$$

It’s Maths All the Way Down

Neural Networks might seem to be complex; how could we understand something that contains millions of calculations?

That is all they are—calculations.

To understand how neural networks function, there are two crucial elements to understand:

Linear Algebra: vectors and matrices

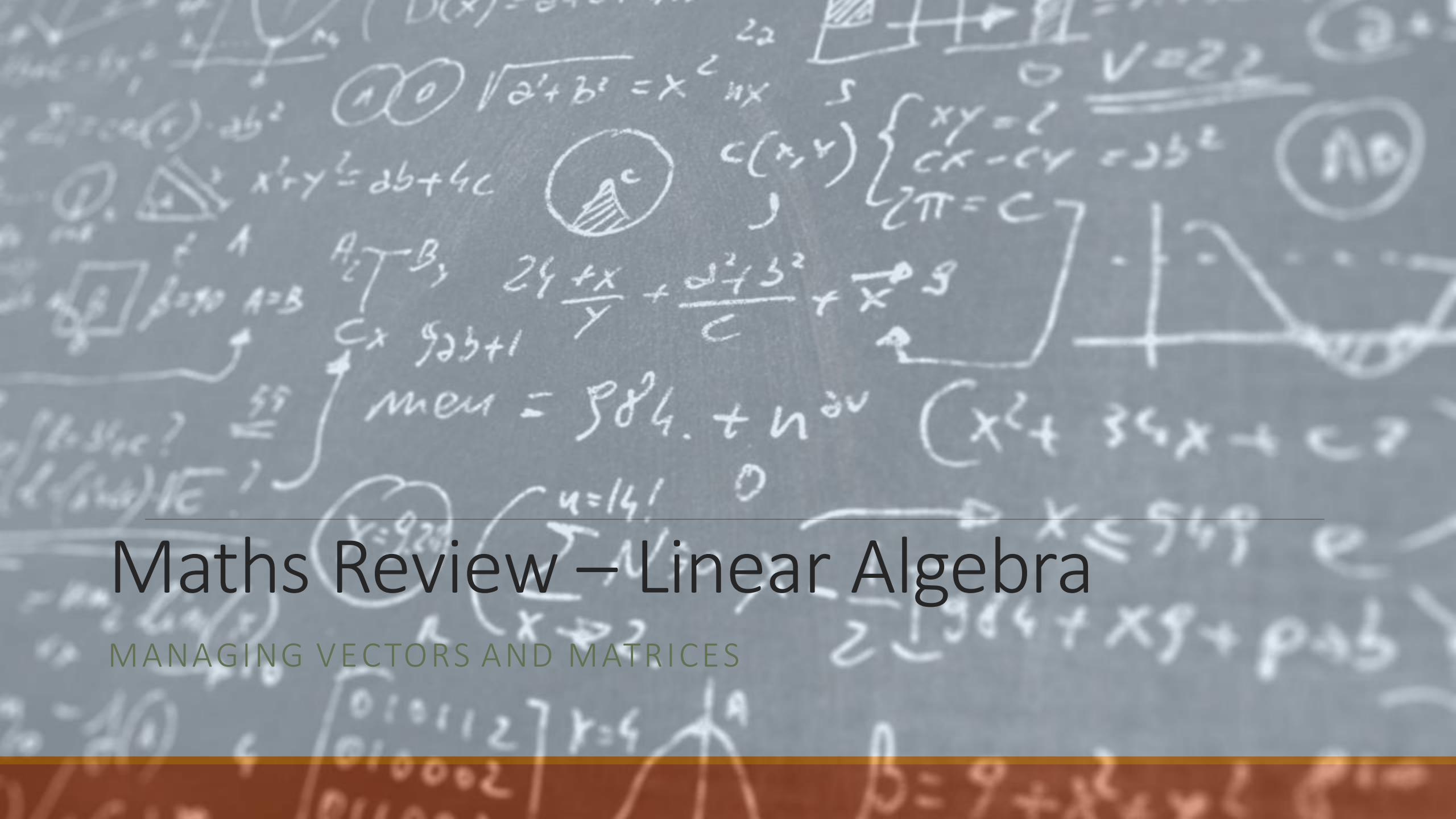
Differential Calculus: the rate at which properties change

It’s Maths All the Way Down

As we will learn throughout this course, a neural network is made up of neurons that perform some mathematical function.

Neurons contain vectors and matrices of weights and biases. (Linear Algebra)

And we want to understand how changing those weights and biases contribute to the output of the network. (Differential Calculus)



Maths Review – Linear Algebra

MANAGING VECTORS AND MATRICES

What is a Vector?

In it's simplest form, a vector is a list of numbers. For example:

Five students take a maths test and score the grades, [82, 75, 64, 91, 77].

This is a vector, $\boldsymbol{x} = (82, 75, 64, 91, 77)$, where each element of the vector can be given a unique subscript,

$\boldsymbol{x} = (x_1, x_2, x_3, x_4, x_5)$

In programming we might call this a linear array or list. For example, in python:

```
>>> x = [82, 75, 64, 91, 77]
```

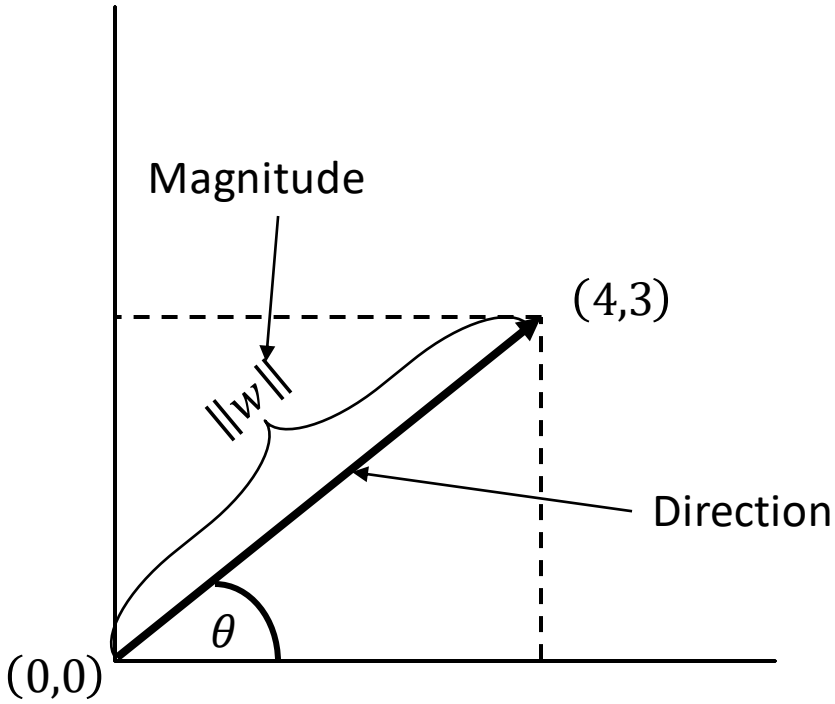
or,

```
>>> x = np.array([82, 75, 64, 91, 77])
```

Vectors in Physics

Vectors are used in physics to denote properties like Velocity and Acceleration

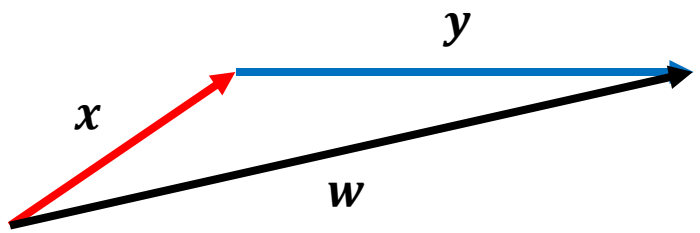
They have Magnitude (or length), Direction and a reference point.



Properties of Vectors

Vectors can be added together by summing their elements

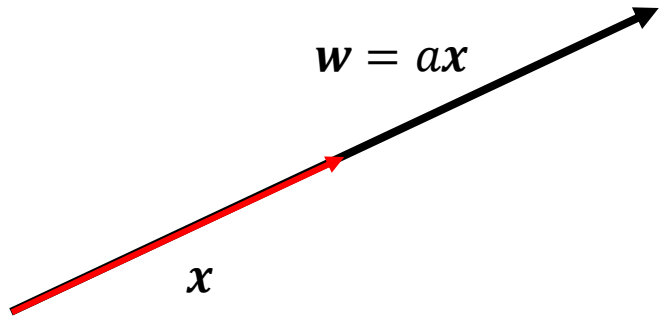
$$\boldsymbol{w} = \boldsymbol{x} + \boldsymbol{y} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} x_1 + y_1 \\ x_2 + y_2 \\ x_3 + y_3 \end{pmatrix}$$



Properties of Vectors

Vectors can be multiplied by non-vector numbers, called scalars, to increase or decrease their magnitudes

$$\boldsymbol{w} = \boldsymbol{x} \times a = \begin{pmatrix} ax_1 \\ ax_2 \\ ax_3 \end{pmatrix}$$



Matrix Addition

$$x = \begin{pmatrix} 5 \\ 2 \\ 1 \end{pmatrix}, y = \begin{pmatrix} 3 \\ 7 \\ 9 \end{pmatrix}, w = x + y = ?$$

$$w = \begin{pmatrix} 5+3 \\ 2+7 \\ 1+9 \end{pmatrix} = \begin{pmatrix} 8 \\ 9 \\ 10 \end{pmatrix}$$

Matrix Multiplication

$$\boldsymbol{w} = a\boldsymbol{x}, \boldsymbol{x} = \begin{pmatrix} 5 \\ 2 \\ 1 \end{pmatrix}, a = 4$$

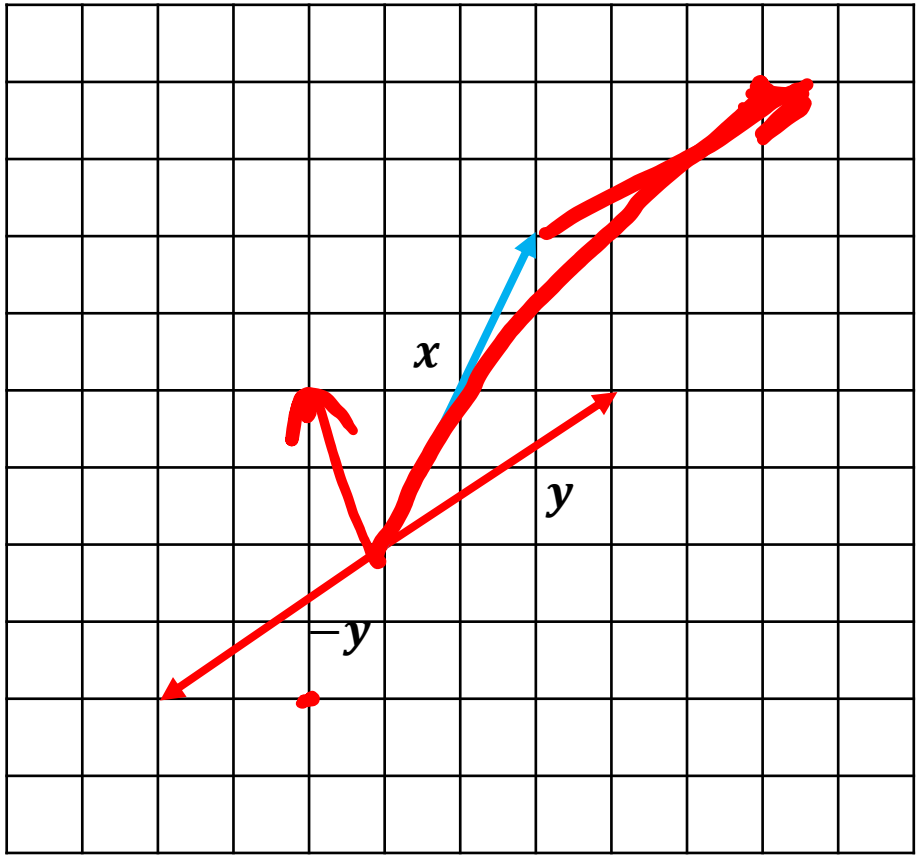
$$\begin{pmatrix} 5\times 4 \\ 2\times 4 \\ 1\times 4 \end{pmatrix} = \begin{pmatrix} 20 \\ 8 \\ 4 \end{pmatrix}$$

Example

$\boldsymbol{x} = (2,4), \boldsymbol{y} = (3,2), \boldsymbol{x} + \boldsymbol{y} = ?, \boldsymbol{x} - \boldsymbol{y} = ?$

$\begin{pmatrix} 5 \\ 6 \end{pmatrix}$

$\begin{pmatrix} -1 \\ 2 \end{pmatrix}$



Vector Dot Product

The dot product (or inner product) is how we multiply two arbitrary vectors together.

Consider two vectors, in d-dimensions ($\mathbb{R}^d$), $\boldsymbol{x} = (x_1, x_2, x_3, \ldots, x_d)$ and $\mathbf{w} = (w_1, w_2, w_3, \ldots, w_d)$

The dot product of these vectors is then:

$$\boldsymbol{w} \cdot \boldsymbol{x} = (x_1 w_1 + x_2 w_2 + x_3 w_3 + \cdots + x_d w_d) = \sum_{i=1}^{d} x_i w_i$$

Vector Norm

The norm—or length—of a vector, $\boldsymbol{w}$, in $\mathbb{R}^d$ is calculated as the non-negative square root of $\boldsymbol{w} \cdot \boldsymbol{w}$.

$$\|\boldsymbol{w}\| = \sqrt{\boldsymbol{w} \cdot \boldsymbol{w}} = \sqrt{w_1^2 + w_2^2 + w_3^2 + \cdots + w_d^2}$$

$$\sum_{i=1}^{d} w_i w_i$$

Example – Distance between two points

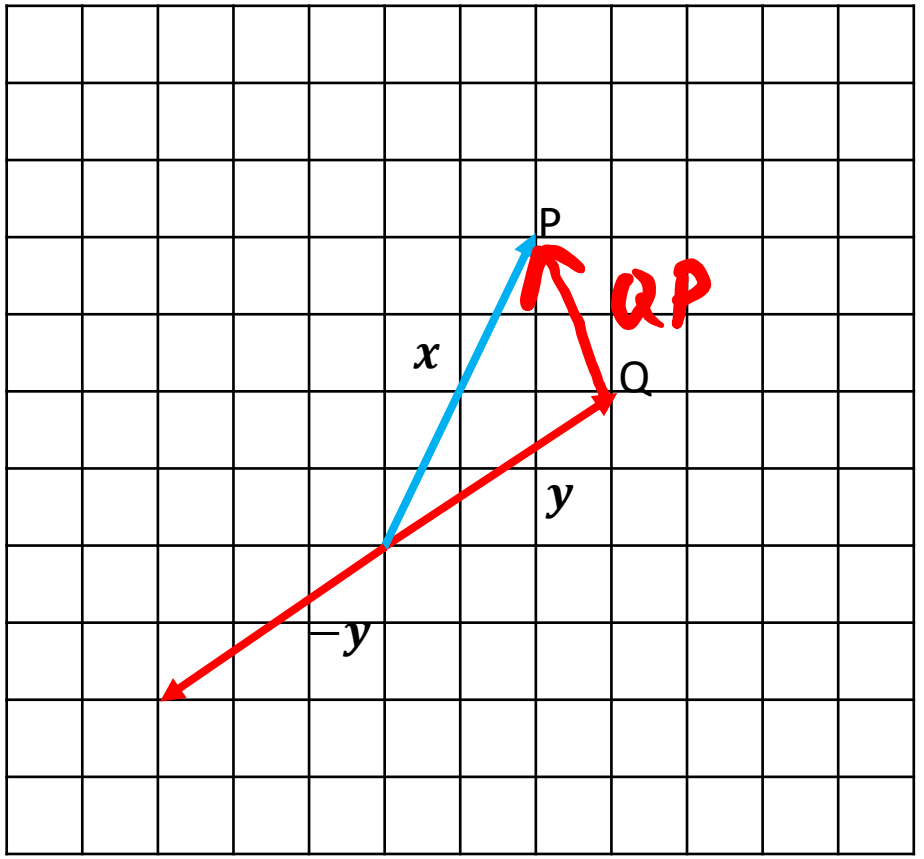
What is the distance between points Q and P?

$\boldsymbol{x} = (2,4), \boldsymbol{y} = (3,2)$

$QP = \boldsymbol{x} - \boldsymbol{y}$

$$\begin{pmatrix} -1 \\ 2 \end{pmatrix}$$

$$\|QP\| = \sum_{i=1} QP_i = \sqrt{1+4} = \sqrt{5}$$



Vector Magnitude and Direction in $\mathbb{R}^2$

Suppose we have a vector $\boldsymbol{u} = (a, b)$

Magnitude, $\|\boldsymbol{u}\| = \sqrt{\boldsymbol{u} \cdot \boldsymbol{u}} = \sqrt{a^2 + b^2}$

Direction, $\theta = \tan^{-1}\left(\frac{b}{a}\right)$

Components are then:

- $a = \|\boldsymbol{u}\| \cos\theta \, , b = \|\boldsymbol{u}\| \sin\theta$

What is a Matrix?

A rectangular arrangement of numbers with m rows and n columns.

A matrix with the same number of rows and columns is called a square matrix

The elements of a matrix are defined with two indices:

$$\boldsymbol{M} = \begin{bmatrix} M_{1,1} & M_{1,2} \\ M_{2,1} & M_{2,2} \\ M_{3,1} & M_{3,2} \end{bmatrix}$$

Matrix Addition

The sum of two matrices, $\boldsymbol{M}$ and $\boldsymbol{N}$, where both matrices must have the same number of rows and columns, is equal to the sum of the corresponding elements:

$$\boldsymbol{M} + \boldsymbol{N} = \begin{bmatrix} M_{1,1} + N_{1,1} & M_{1,2} + N_{1,2} \\ M_{2,1} + N_{2,1} & M_{2,2} + N_{2,2} \\ M_{3,1} + N_{3,1} & M_{3,2} + N_{3,2} \end{bmatrix}$$

Scalar Multiplication

As with vectors, a matrix can be multiplied by a scalar, such that the multiplication acts on every element of the matrix

$$a\boldsymbol{M} = \begin{bmatrix} aM_{1,1} & aM_{1,2} \\ aM_{2,1} & aM_{2,2} \\ aM_{3,1} & aM_{3,2} \end{bmatrix}$$

Example

Suppose, $\boldsymbol{M} = \begin{bmatrix} 1 & 3 \\ 5 & 8 \\ 4 & 7 \end{bmatrix}$ and $\boldsymbol{N} = \begin{bmatrix} 3 & 2 \\ 8 & 8 \\ 9 & 1 \end{bmatrix}$

Find:

$2\boldsymbol{M} + 4\boldsymbol{N} =$

$3\boldsymbol{M} - 2\boldsymbol{N} =$

Matrix Multiplication

Taking the product, AB, of two matrices, A and B, can be quite complex.

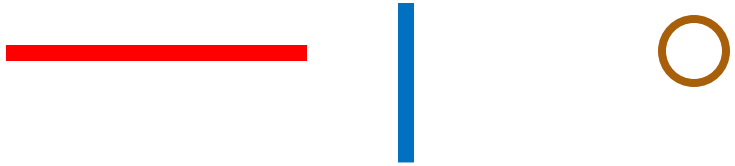
$$AB = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \end{bmatrix} \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \\ b_{3,1} & b_{3,2} \end{bmatrix} = \begin{bmatrix} c_{1,1} & c_{1,2} \\ c_{2,1} & c_{2,2} \end{bmatrix}$$

Where each element, $c_{i,j}$, is the sum of the products of each element in row $[a_i]$ and column $[b_j]$

$$c_{1,1} = a_{1,1} \times b_{1,1} + a_{1,2} \times b_{2,1} + a_{1,3} \times b_{3,1}$$

Matrix Multiplication

$$AB = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \end{bmatrix} \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \\ b_{3,1} & b_{3,2} \end{bmatrix} = \begin{bmatrix} c_{1,1} & c_{1,2} \\ c_{2,1} & c_{2,2} \end{bmatrix}$$

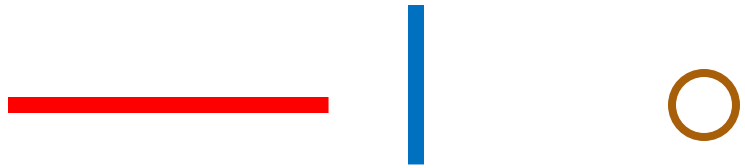


Matrix Multiplication

$$AB = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \end{bmatrix} \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \\ b_{3,1} & b_{3,2} \end{bmatrix} = \begin{bmatrix} c_{1,1} & c_{1,2} \\ c_{2,1} & c_{2,2} \end{bmatrix}$$

Matrix Multiplication

$$AB = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \end{bmatrix} \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \\ b_{3,1} & b_{3,2} \end{bmatrix} = \begin{bmatrix} c_{1,1} & c_{1,2} \\ c_{2,1} & c_{2,2} \end{bmatrix}$$



Matrix Multiplication

$$AB = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \end{bmatrix} \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \\ b_{3,1} & b_{3,2} \end{bmatrix} = \begin{bmatrix} c_{1,1} & c_{1,2} \\ c_{2,1} & c_{2,2} \end{bmatrix}$$

Matrix Multiplication

To take the product of two matrices, the number of *columns* in the first matrix must be equal to the number of *rows* in the second matrix

The resulting matrix will have the same number of *rows* as the first matrix, and the same number of *columns* as the second matrix

Example

Let $\mathbf{X} = \begin{bmatrix} 2 & 1 & 4 \\ 3 & 3 & 5 \end{bmatrix}$, and $\mathbf{Y} = \begin{bmatrix} 6 & 2 \\ 5 & 1 \\ 3 & 4 \end{bmatrix}$. Find $\boldsymbol{XY}$:

$$\begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

$c_{11} = 12+5+12 = 29$

$c_{12} = 4+1+16 = 21$

$c_{21} = 18+15+15 = 48$

$c_{22} = 6+3+20 = 29$

$$XY = \begin{bmatrix} 29 & 21 \\ 48 & 29 \end{bmatrix}$$

Inverse Matrices

Not all matrices can be inverted. For a given matrix, $\boldsymbol{A}$, we wish to find the inverse, $\boldsymbol{A}^{-1}$, such that the following is true:

$$(\boldsymbol{A}\boldsymbol{A}^{-1}) = (\boldsymbol{A}^{-1}\boldsymbol{A}) = \boldsymbol{I}$$

Where $\boldsymbol{I}$ is the Identity Matrix…

Identity Matrix

A square matrix where all diagonal elements are equal to 1 and all off-diagonal elements are equal to 0:

$$\boldsymbol{I}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \boldsymbol{I}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \boldsymbol{I}_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The product of a matrix and an identity matrix is simply equal to the matrix itself:

$$\boldsymbol{AI} = \boldsymbol{IA} = \boldsymbol{A}$$

Matrix Inversion

Only square matrices can be inverted

But not all square matrices can be inverted

Matrix Inversion – Partial Proof

$$(\boldsymbol{A}\boldsymbol{A}^{-1}) = \boldsymbol{I}$$

$$\begin{pmatrix} \boldsymbol{a} & \boldsymbol{b} \\ \boldsymbol{c} & \boldsymbol{d} \end{pmatrix} \begin{pmatrix} \boldsymbol{w} & \boldsymbol{x} \\ \boldsymbol{y} & \boldsymbol{z} \end{pmatrix} = \begin{pmatrix} \boldsymbol{1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{1} \end{pmatrix}$$

Matrix Determinant

The value used to derive the inverse of a matrix is called the determinant, For a 2×2 matrix, it is:

$$det(A) = det\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) = ad - bc$$

The determinant tells us about the linear mapping of matrices, and in solving sets of linear equations

Other Matrix Properties

The transpose of a matrix is taken by swapping the rows and columns:

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix}, \boldsymbol{A}^T = \begin{bmatrix} a & c & e \\ b & d & f \end{bmatrix}$$

Orthogonal matrices are those where $\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{I}$, i.e. $\boldsymbol{A}^T = \boldsymbol{A}^{-1}$:

The Trace of a matrix is the sum of its diagonal elements:

$$Tr(\boldsymbol{A}) = a_{11} + a_{22} + a_{33} + a_{44} + \cdots a_{nn}$$

Maths Review – Differential Calculus

FINDING GRADIENTS OF SLOPES

Lecture 1 - Introduction

Learning Outcomes

Get and Overview of the Course Structure and Assignments

Review maths skills for the module

- Use Linear Algebra to manipulate vectors and matrices
- Calculate differentials using the chain rule

Define what we mean by Machine Learning, Neural Networks and AI

What is a derivative?

For a function, $f(x)$, the derivative of that function, $f'(x)$ or $\frac{df}{dx}$, is the rate of change of that function.

The derivative can be thought of as the slope of a line that is tangent to a curve.

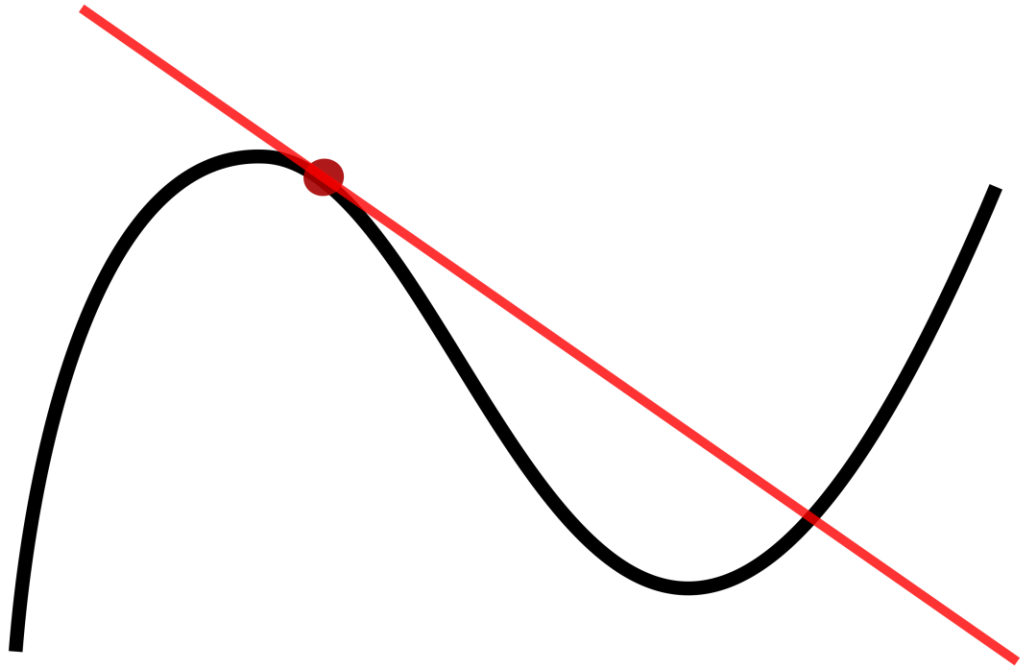


Image Source: https://en.wikipedia.org/wiki/Derivative

Limit Definition

For a given function to be differentiable at some point, a, we must be able to calculate the limit:

$$L = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}$$

The derivative at point a, $f'(a)$, is given by this limit

Example

Suppose we have the function: $f(x) = 4x + 9$

From the limit definition we know:

$$f'(x) = \lim_{h \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

Differentials

$\Delta x = dx$ is referred to as an increment in x. The increment in $y = f(x)$ can be expressed as:

$$\Delta y = f(x + \Delta x) - f(\Delta x)$$

If a derivative can be found at every point in the function, then the function is said to be differentiable.

Combing this with the limit theorem, we can see that:

$$\mathrm{d}y = f'(x)dx$$

$$\frac{dy}{dx} = f'(x)$$

Rules of Differentiation

$$\frac{d}{dx}x^a = ax^{a-1}$$

$$\frac{d}{dx}ax = a$$

$$\frac{d}{dx}|x| = \frac{|x|}{x},\ x \neq 0$$

$$\frac{d}{dx}\left(\frac{1}{x^c}\right) = \frac{d}{dx}(x^{-c}) = -cx^{c-1} = -\frac{c}{x^{c+1}}$$

$$\frac{d}{dx}\sqrt{x} = \frac{d}{dx}x^{\frac{1}{2}} = \frac{1}{2}x^{-\frac{1}{2}} = \frac{1}{2\sqrt{x}},\ x > 0$$

$$\frac{d}{dx}e^x = e^x$$

$$\frac{d}{dx}a^x = a^x \ln a\,,\ a > 0$$

$$\frac{d}{dx}\ln x = \frac{1}{x},\ x > 0$$

$$\frac{d}{dx}\log_a x = \frac{1}{x \ln a},\ x, a > 0$$

$$\frac{d}{dx}\sin x = \cos x$$

$$\frac{d}{dx}\cos x = -\sin x$$

$$\frac{d}{dx}\tan x = \sec^2 x = \frac{1}{\cos^2 x} = 1 + \tan^2 x$$

Examples

Find the derivatives of the following equations:

$y = 2x^2 + 4x + 7$

$\frac{dy}{dx} = 4x + 4$

$y = \frac{1}{\sqrt[3]{4x}}$

$y = 4x^{-\frac{1}{3}}$

$\frac{dy}{dx} = -\frac{4}{3}x^{-\frac{4}{3}}$

$y = e^{5x+3}$

#

Chain Rule

In the study of neural networks, an important property of differentials is the Chain Rule. This rule is a formula for computing the composition of two or more functions.

Suppose $y = h(x)$ and $z = f(y)$, then $z = f\big(h(x)\big)$

The differential of the function z with respect to x is then:

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx} = f'(y)h'(x) = f'\big(h(x)\big)h'(x)$$

Chain Rule Explained

If a car travels twice as fast as a bicycle, and a bicycle is four times as fast as a walking person, then the car travels $2 \times 4 = 8$ times faster than the person.

Let z, y, x be the relative positions of the car, bike and person, respectively. The rates of change of the relative positions of the car and bike is: $\frac{dz}{dy} = 2$. Similarly, $\frac{dy}{dx} = 4$.

We want to know $\frac{dz}{dx}$, which is just the product of $\frac{dz}{dy}$ and $\frac{dy}{dx}$.

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx} = 2 \cdot 4 = 8$$

Adapted from: https://en.wikipedia.org/wiki/Chain_rule

Chain Rule Explained

If we want to know the speed of the car, that is the rate of change of position with respect to time, we can write:

$$\frac{dz}{dx} = \frac{\frac{dz}{dt}}{\frac{dx}{dt}}$$

That is, the rate of change in positions is equivalent to the ratio of the speeds. Thus the speed of car can be written as:

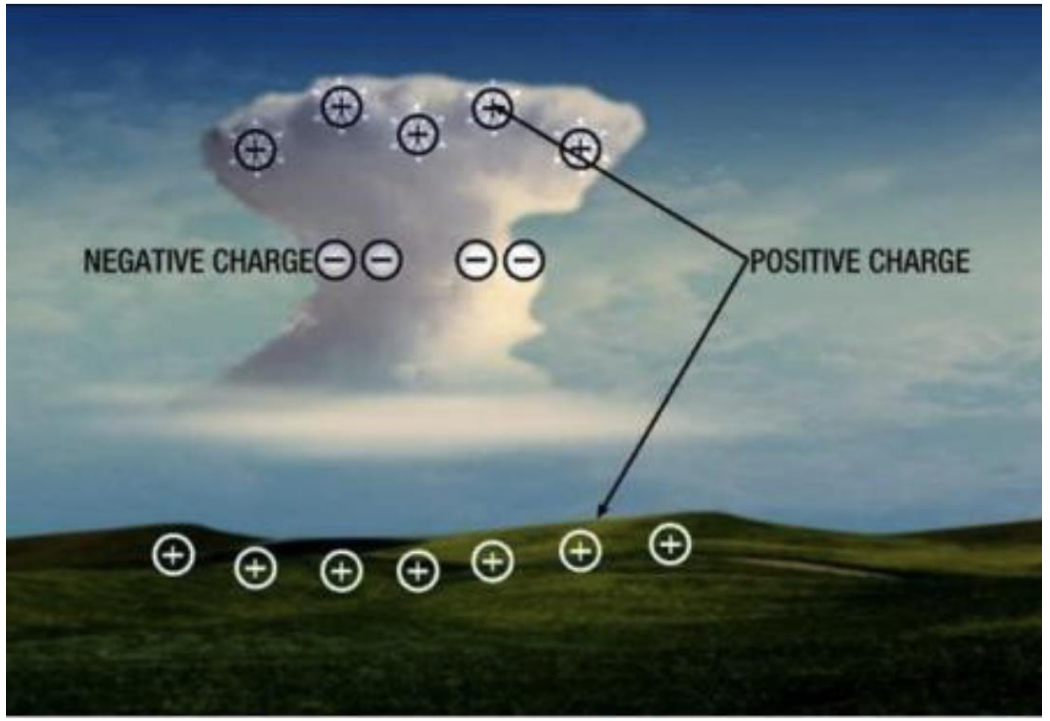
$$\frac{dz}{dt} = \frac{dz}{dx} \cdot \frac{dx}{dt}$$

How Do Neurons Work?

Neurons operate based on the principle of charge separation.

When positive and negative electrical charges are separated, they generate voltage gradients and currents.

The same principle is what drives thunderstorms.



Basic Neuron Structure

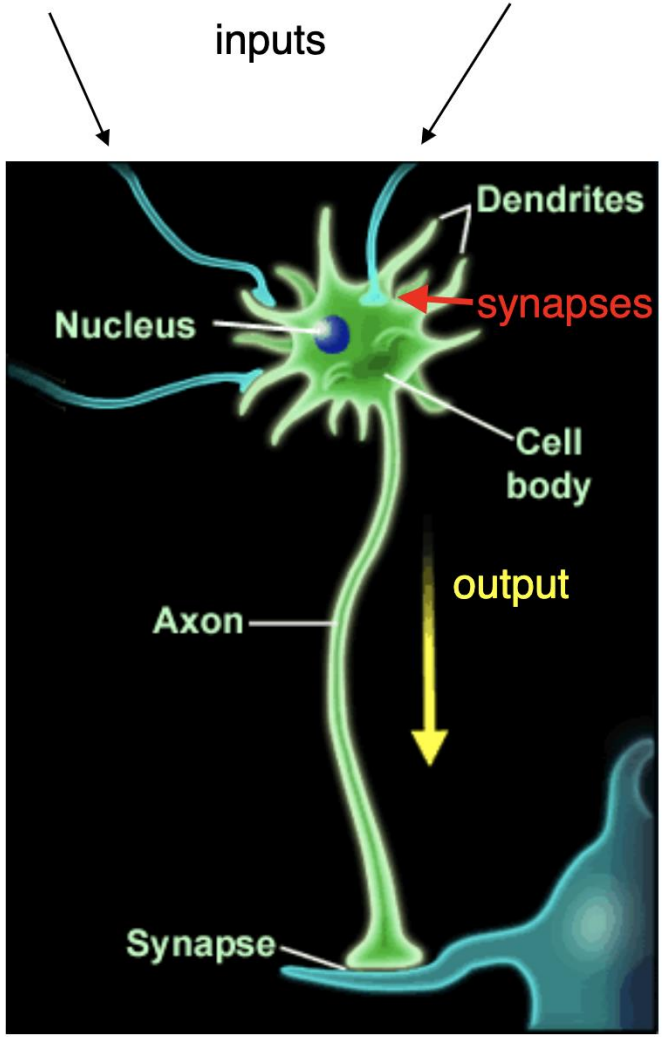
A typical neuron consists of three made sections:

Dendrites: receive and process inputs.

Soma: Cell body of the neuron that integrates the inputs.

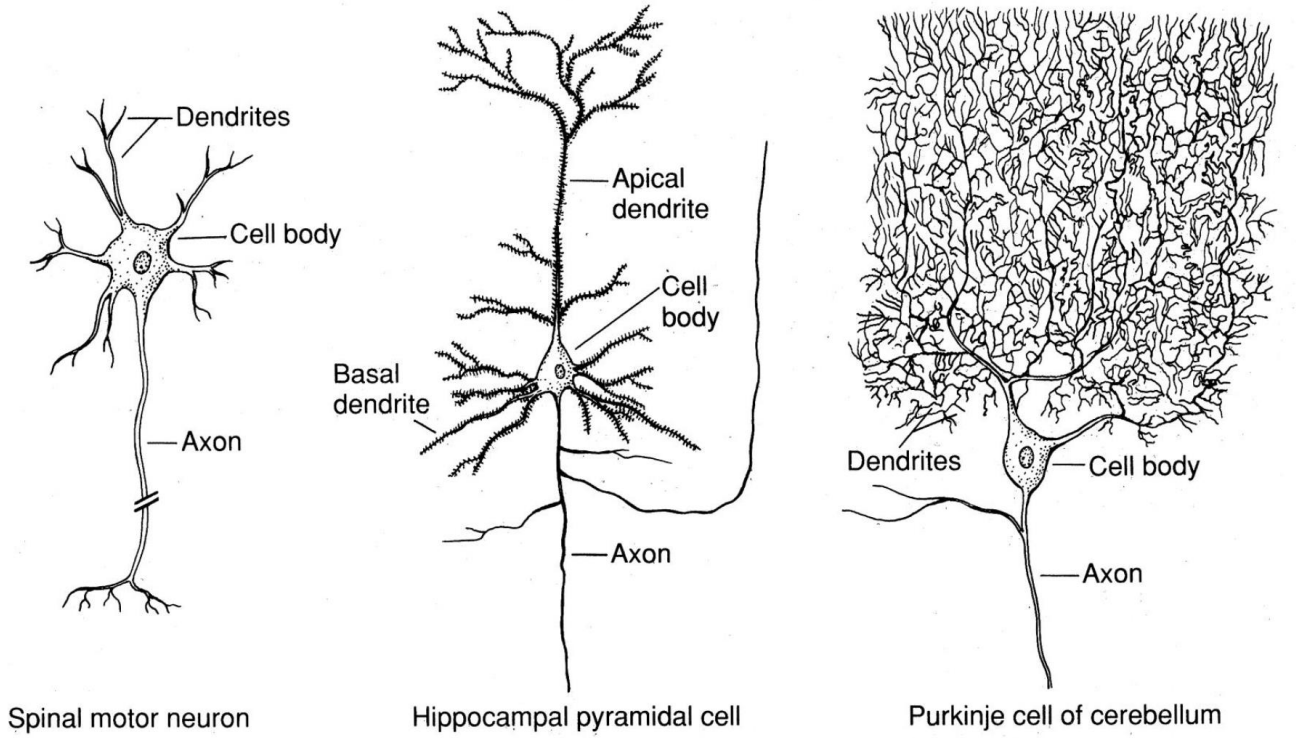
Axon: Generates the output and transmits it to the target neurons.

The connection between the axon of a neuron and the dendrite of another is called a **synapse.**



Other Structures

Neuronal structures are incredibly varied. In addition to synapses that connect axons to dendrites, there are also synapses that connect axons directly to cell bodies, and onto other axons.

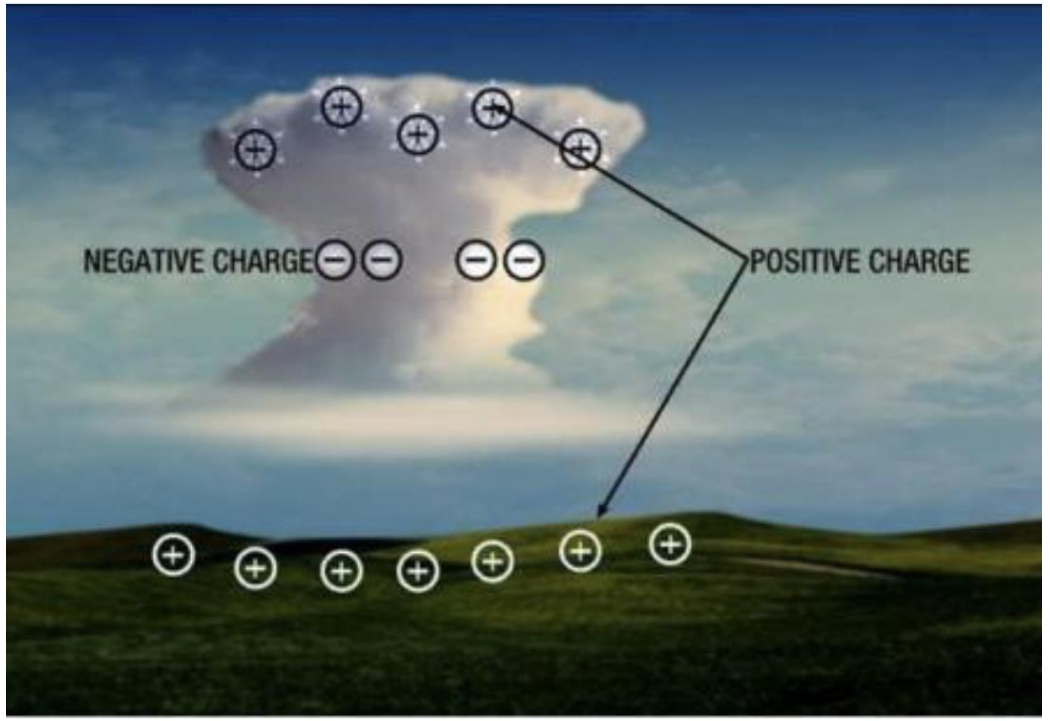


How Do Neurons Work?

Neurons operate based on the principle of charge separation.

When positive and negative electrical charges are separated, they generate voltage gradients and currents.

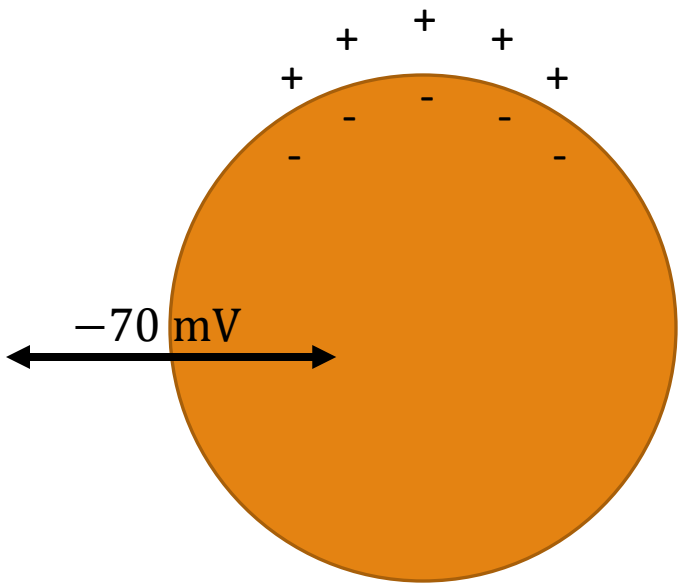
The same principle is what drives thunderstorms.



Neurons as Electrical Devices

The cell membrane—which separates the interior of the cell with the exterior—is able to maintain a separation of charges.

Most of the time the inside of the cell is negatively charged, and the outside of the cell is positively charged. In the absence of any input, the voltage of the membrane is around -70mV (millivolts).

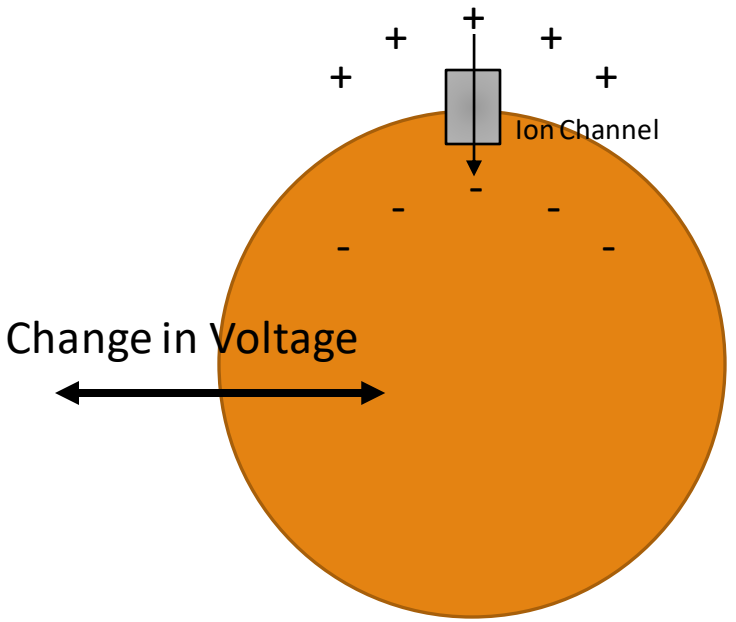


Simplified model of a neurons cell membrane. The outside of the cell is positively charge, and the inside of the cell is negatively charged.

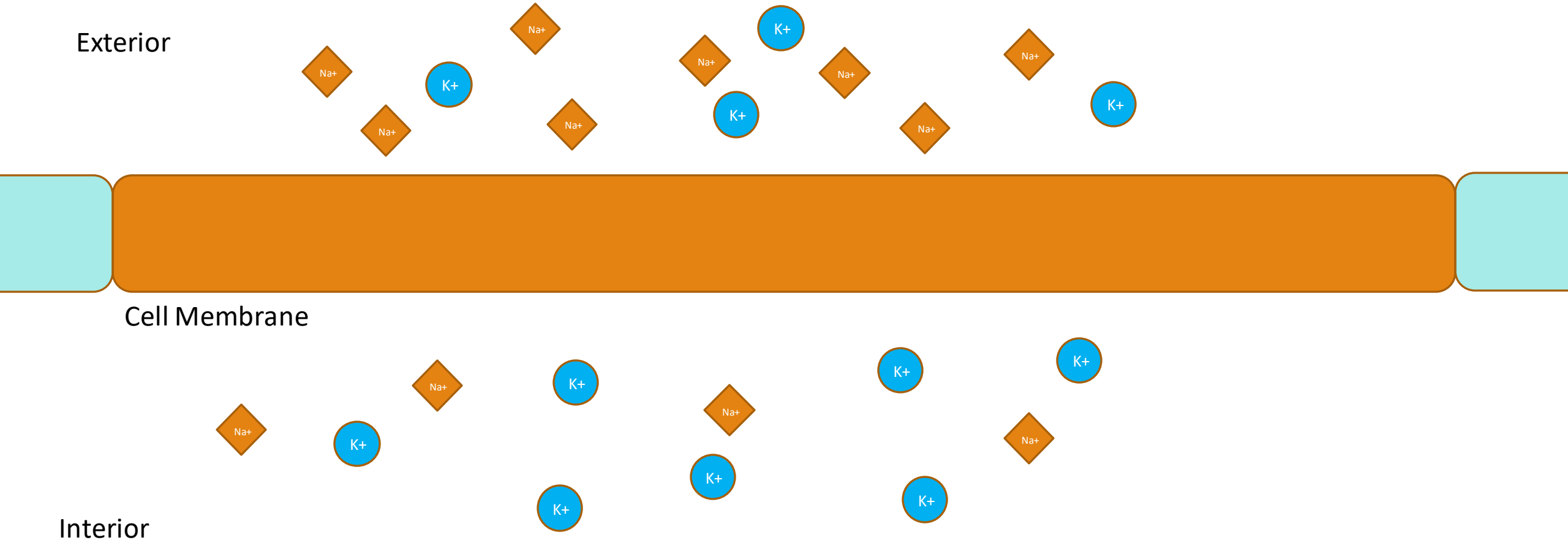
Neurons as electrical devices

Neurons transfer information via the temporary changes in their membrane potential.

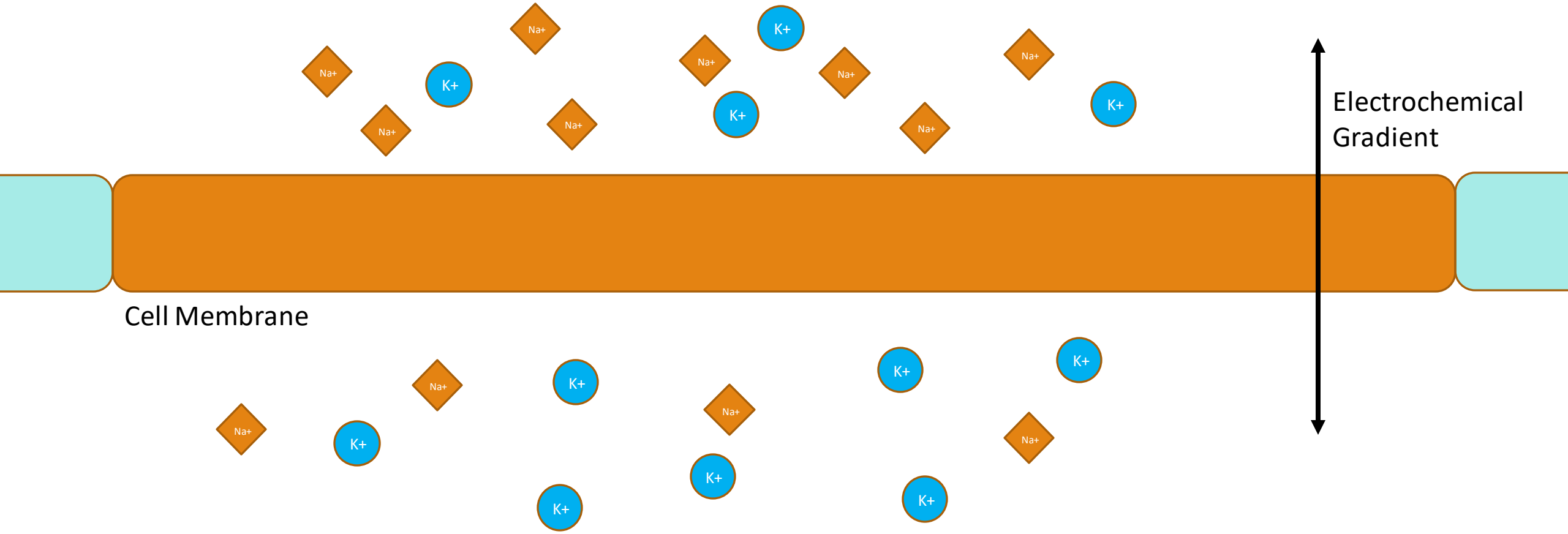
Small pores—biological devices called ion channels and pumps—open and close in response to inputs, allowing the flow of ions (charge) across the membrane.



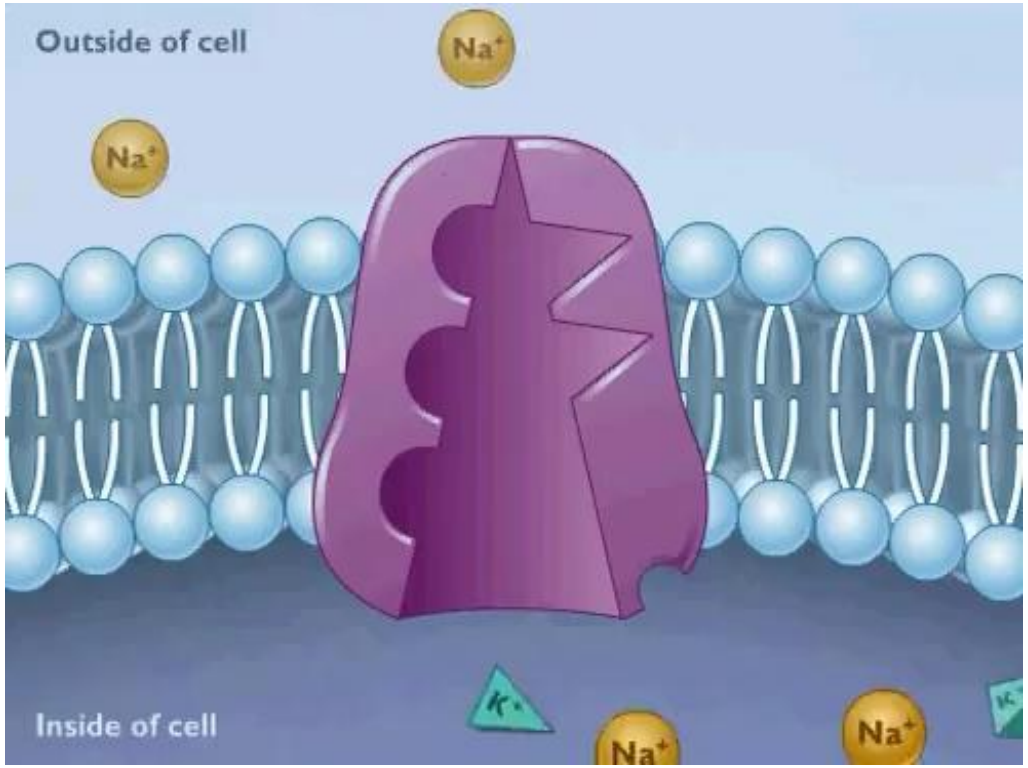
The Neuron At Rest



The Neuron At Rest



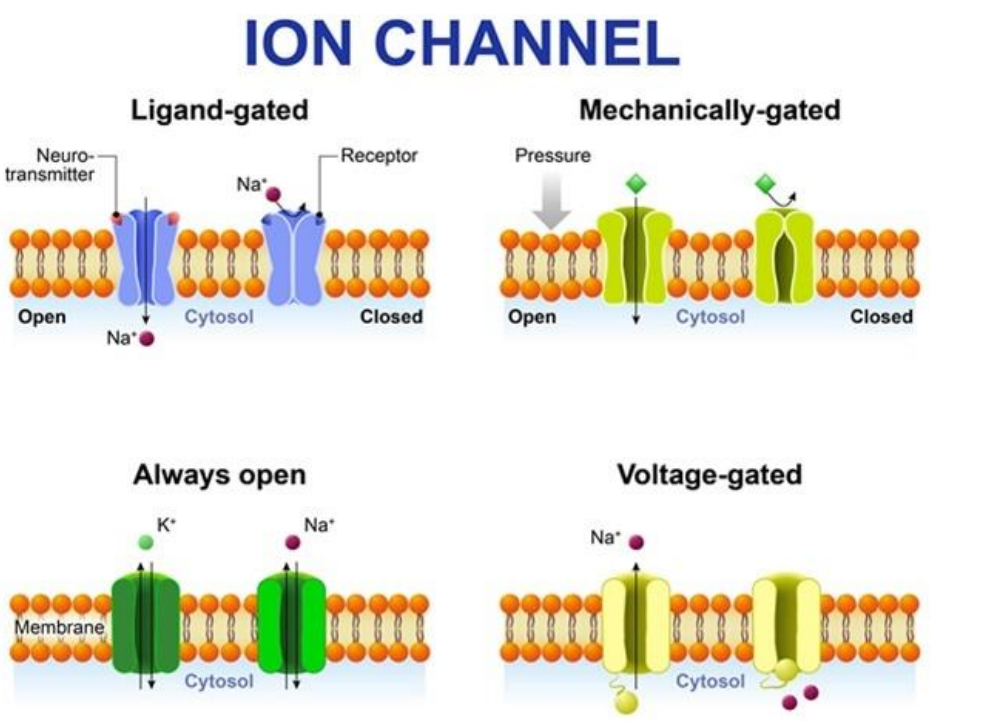
Sodium Potassium Pump



The sodium-potassium pump is one of the most important structures in the body, and is responsible for about 20% of your brains energy use.

The pump actively moves three sodium ions from inside the cell to the exterior, then moves two potassium ions inside—this maintains the resting potential of the neuron.

Ion Channels – the bodies response



Types of ion channel. Classification by gating. mechanism of action. Voltage-Gated, Ligand-gated, Mechanically-gated and Always open ion channels. Image Credit: Designua / Shutterstock

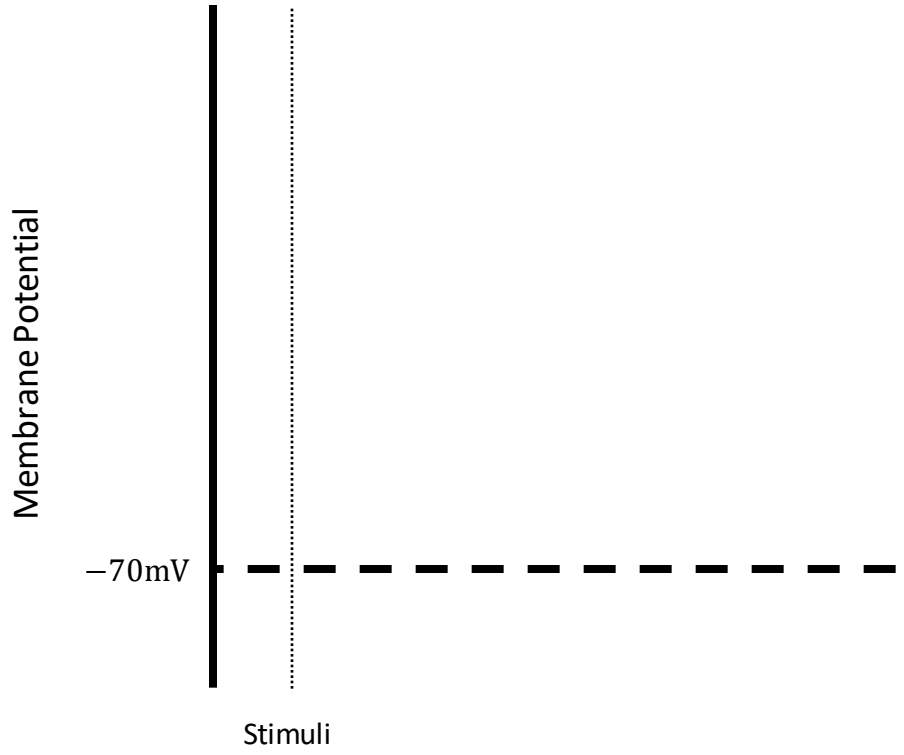
Ion channels facilitate the movement of sodium and potassium across the cell membrane, and normally only open under certain conditions.

Ligand-gated channels open in response to molecules—such as serotonin—while mechanically-gated channels open in response to physical stimuli.

Voltage-gated channels open in response to a change in the membrane potential, and are what propagate the neural signal.

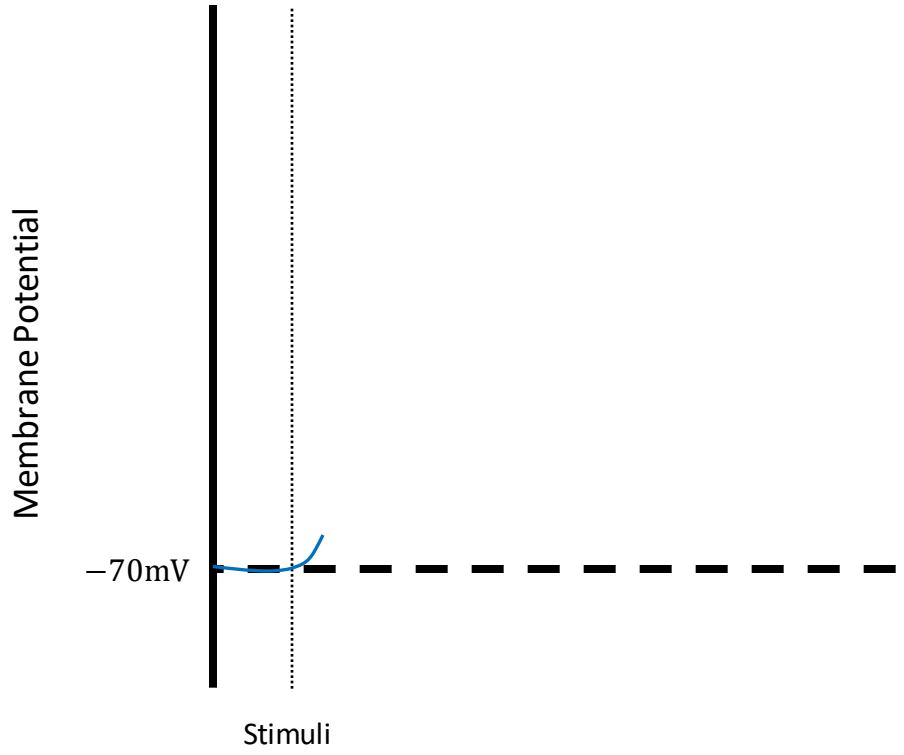
The Action Potential

Given a sufficiently strong input, the ion channels of the neuron open and allow the flow of sodium ions into the cell.



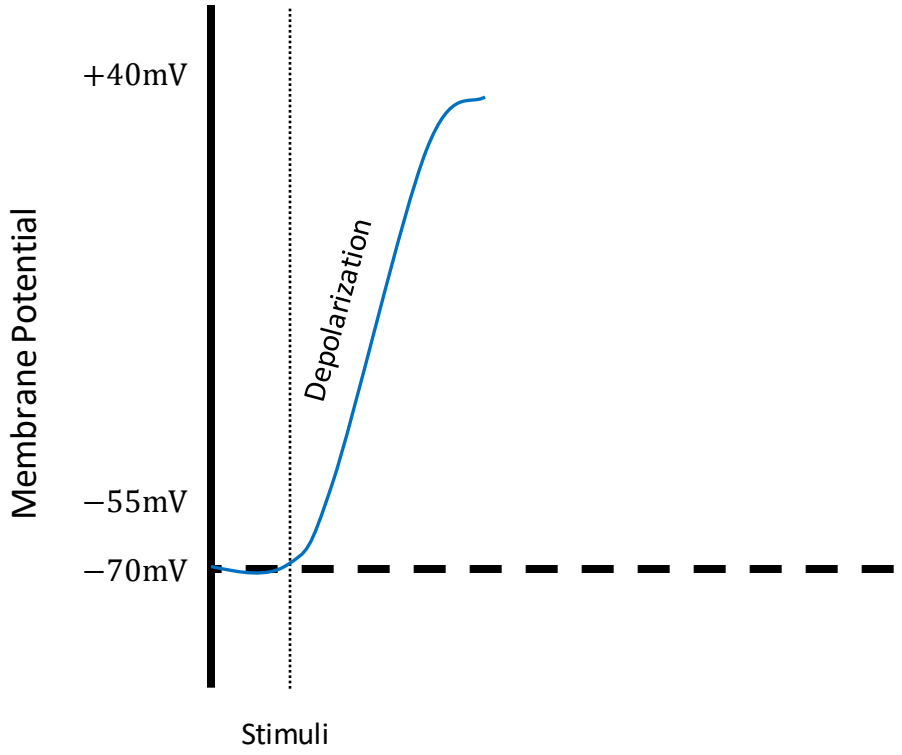
The Action Potential

Given a sufficiently strong input, the ion channels of the neuron open and allow the flow of sodium ions into the cell.



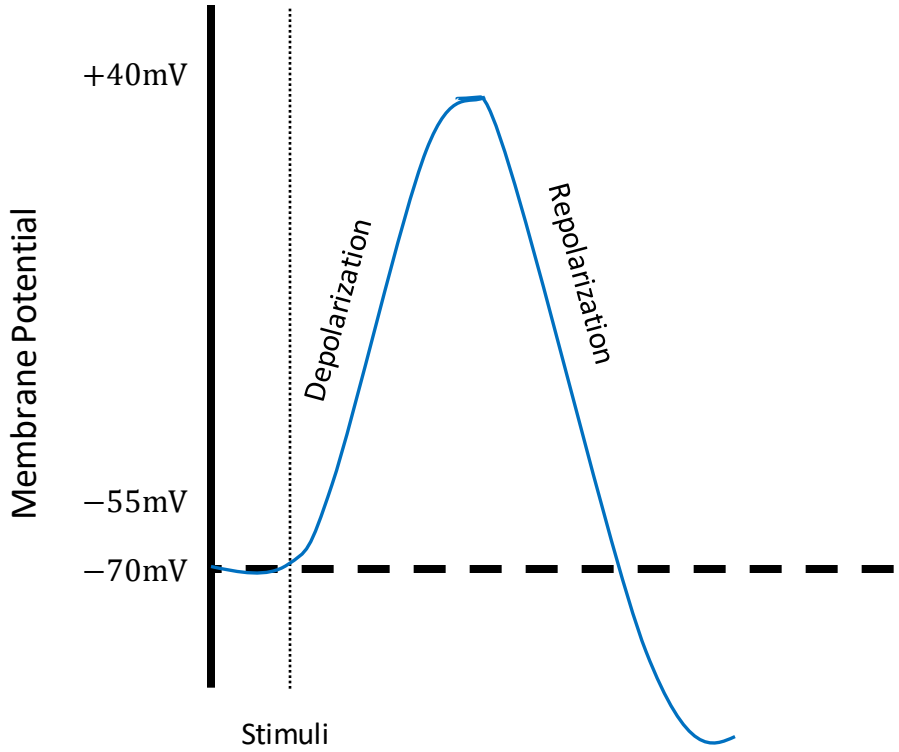
The Action Potential

The membrane potential becomes less negative, and at a threshold voltage of -55mV, the voltage-gated channels open—flooding the cell with positive sodium ions.



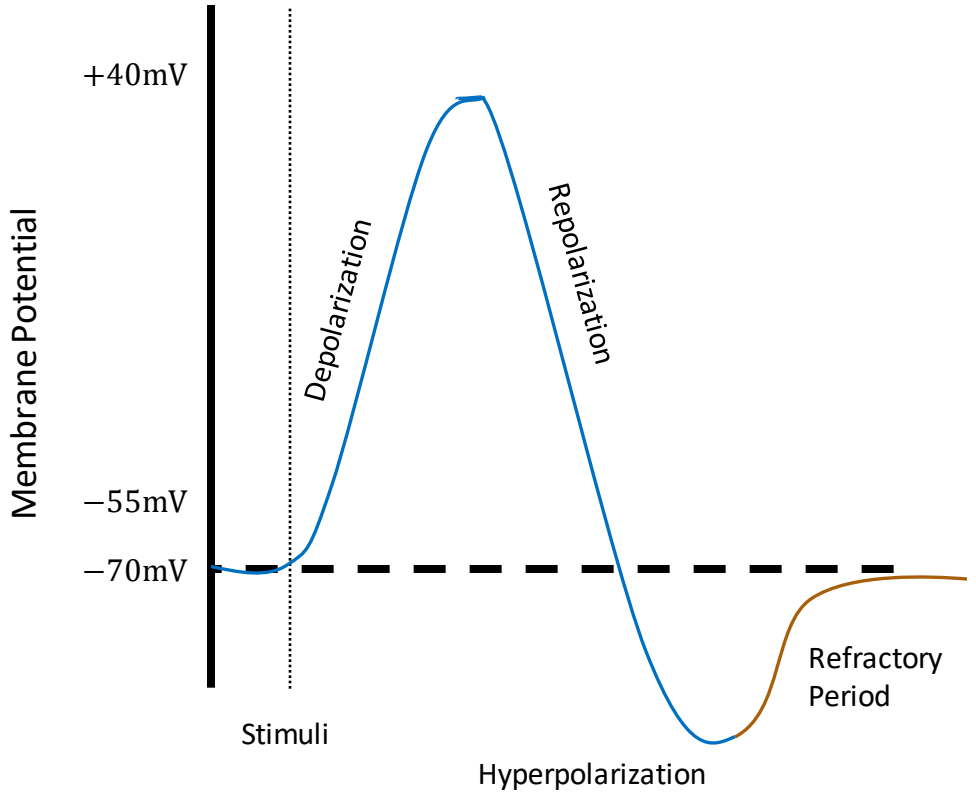
The Action Potential

At around $+30 - 40\text{mV}$, the voltage channels close. Slower opening potassium voltage channels open, which normalise the membrane potential.



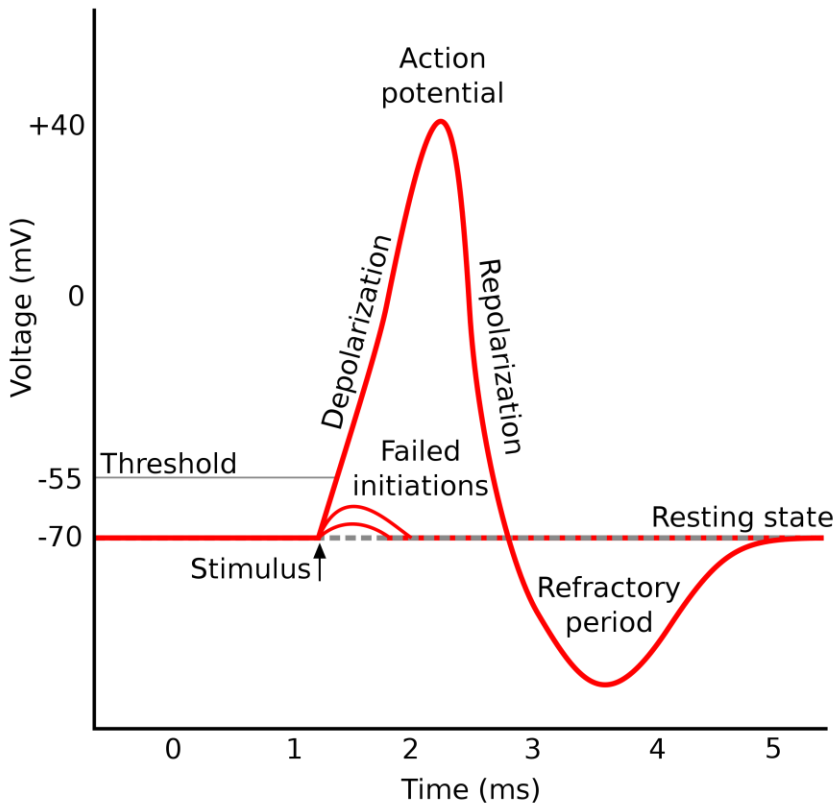
The Action Potential

The membrane potential overshoots the resting potential, becoming hyperpolarised. The sodium-potassium pumps then work to return the neuron to the resting potential.



The Action Potential

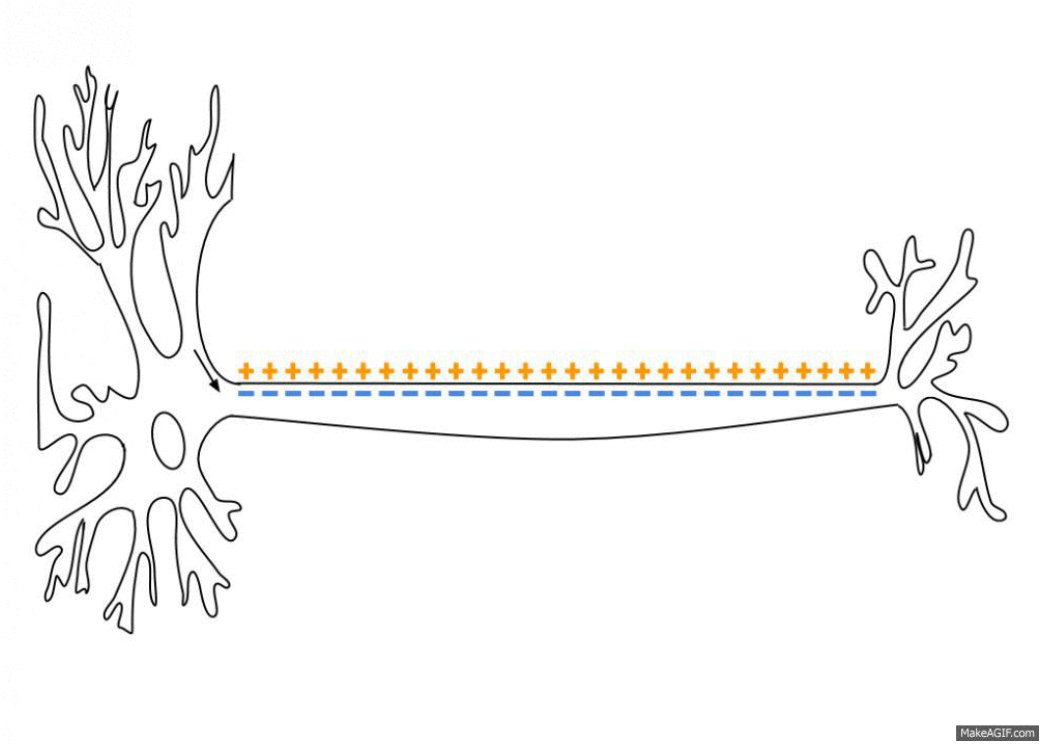
Action potentials are all or nothing. If the stimuli is not strong enough to reach the threshold voltage of the ion channels, the sodium-potassium pumps simply reset the resting potential—this is called a graded potential.



Propagation along the Neuron

Voltage gates along the axon of a neuron are triggered in sequence, due to the relative spatial electrochemical potential.

The refractory period prevents the neurons from firing repeatedly, meaning the signal can only travel one way. This is important for how our brains recognise different stimuli.

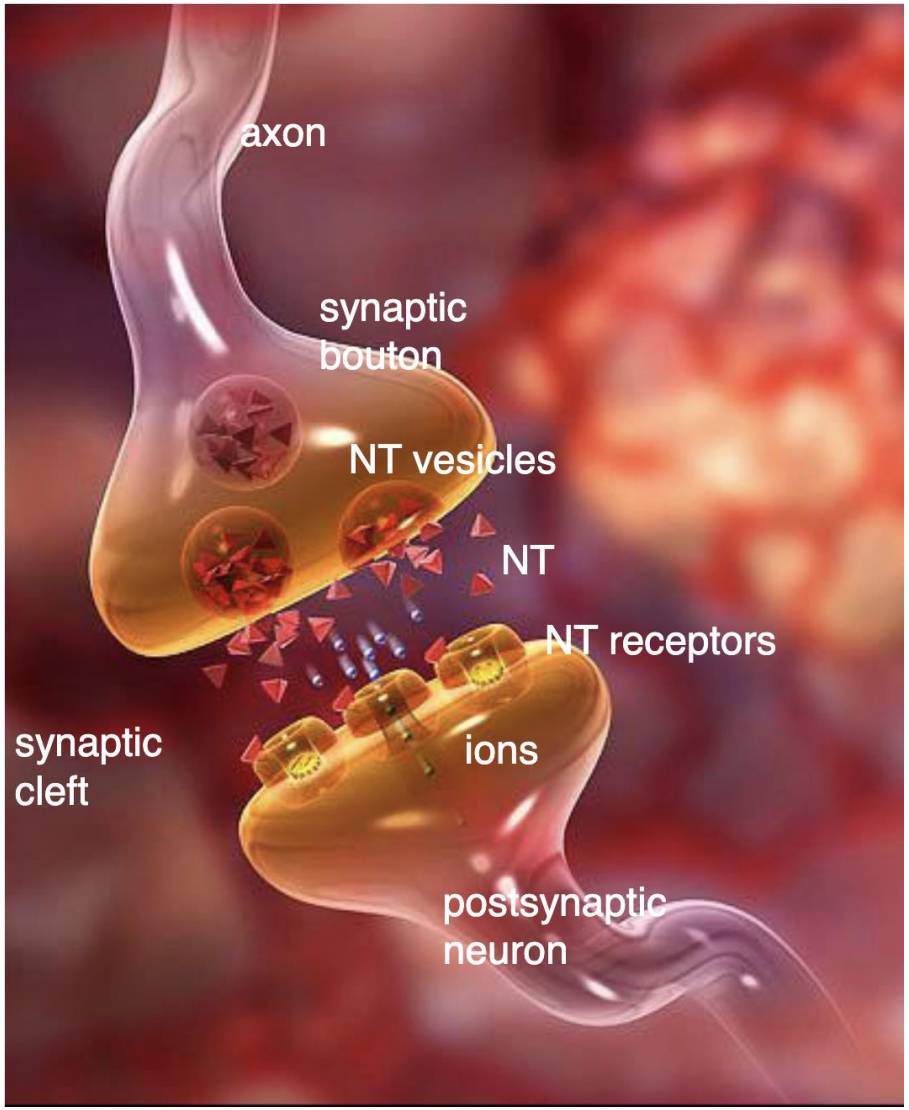


Neurotransmitters and Synapses

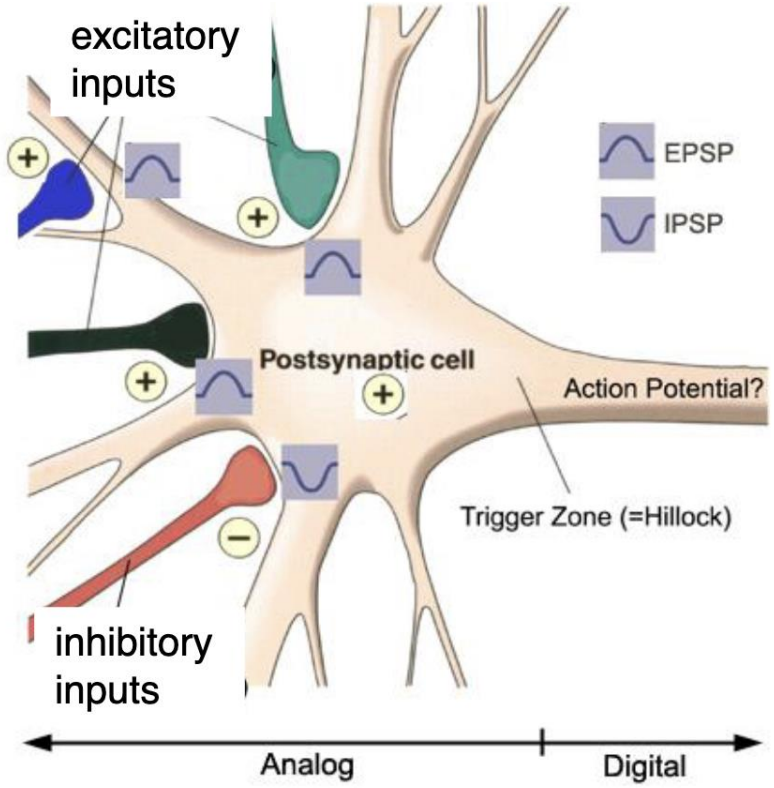
When the AP reaches the end of the axon, the signal is transmitted across a synapse.

Neurotransmitters—such as serotonin and dopamine—are released from vesicles into the synaptic cleft.

The NT bind with receptors on the postsynaptic membrane—these can be ion channels that generate a postsynaptic potential.



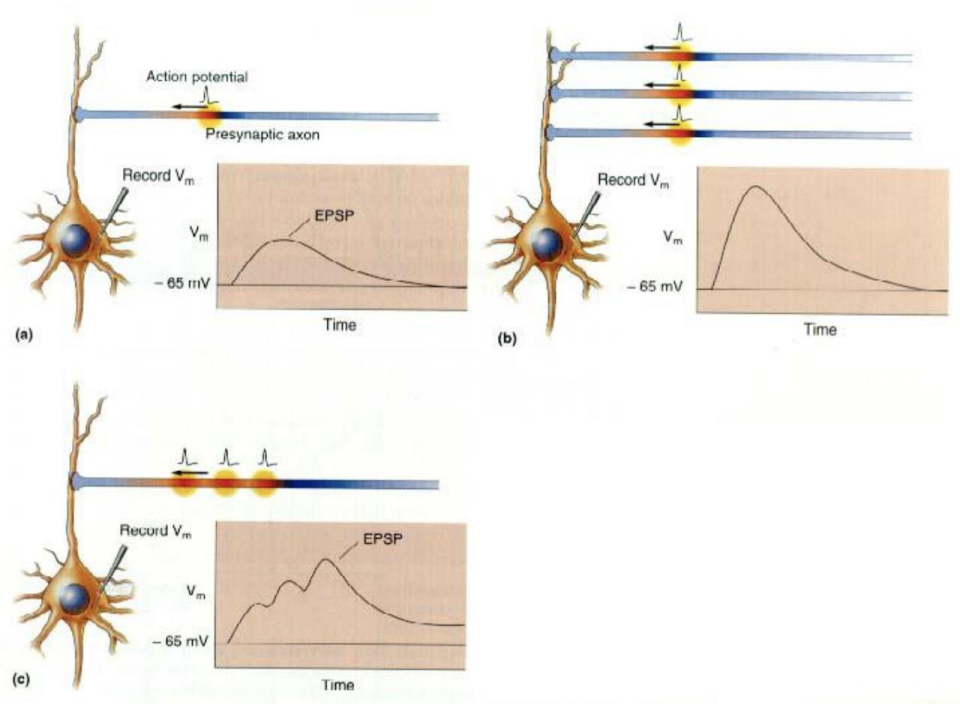
Postsynaptic potentials



Postsynaptic potentials (PSPs) spread along the dendrites to the soma and axon hillock, where they may trigger a new action potential.

PSPs can be excitatory (EPSPs) or inhibitory (IPSPs), which increase or decrease the likelihood of a new action potential, respectively.

Summing up PSPs



The postsynaptic neuron can sum up the incoming action potentials based on spatial and temporal information.

APs arriving at synapses on different locations along the dendrite, or at different times, combine together to form the PSP.

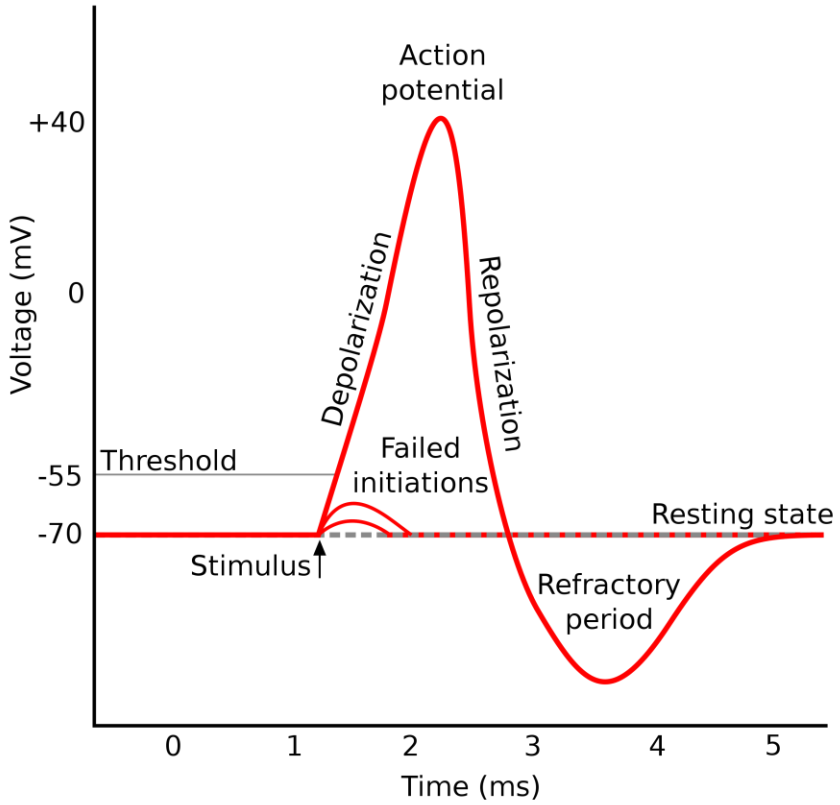
PSPs are continuous, analogue signals—while APs are pulsed, digital signals.

Problem – APs are always the same

Action potentials are distinct, all or nothing, digital pulses. The resting potential is always $-70\mathrm{mV}$, the threshold $-55\mathrm{mV}$, and the peak voltage $+40\mathrm{mV}$. For sodium based signals, they always last a few milliseconds.

This is the same if a you are picking up a can or a boulder!

How does our nervous system distinguish between different stimuli?



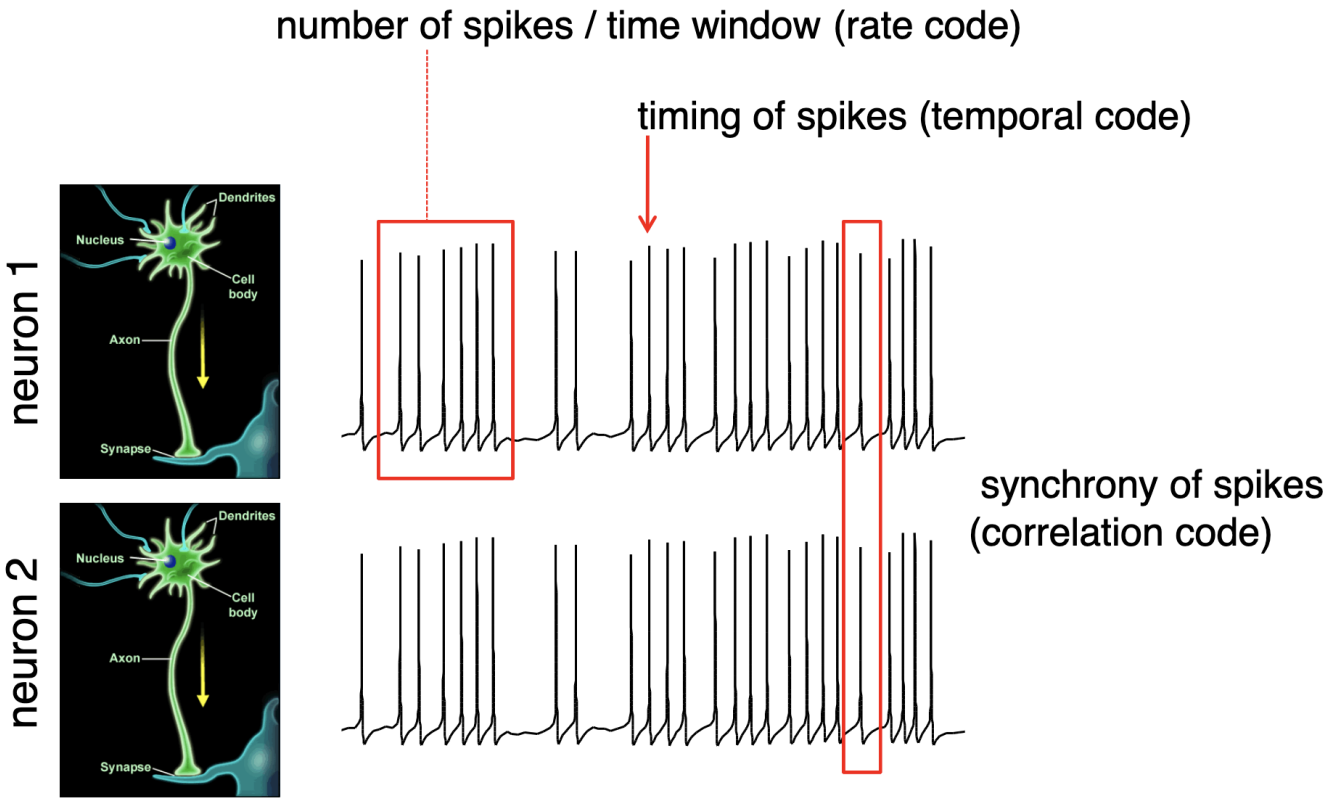
Neural coding

Neurons fire off action potentials with different rates and patterns in response to different stimuli. Information is encoded in these action potential trains.

Rate coding refers to the mean rate of APs in a given amount of time.

Temporal coding refers to the timing of individual AP spikes.

Correlation coding refers to the synchrony of signals in multiple neurons.

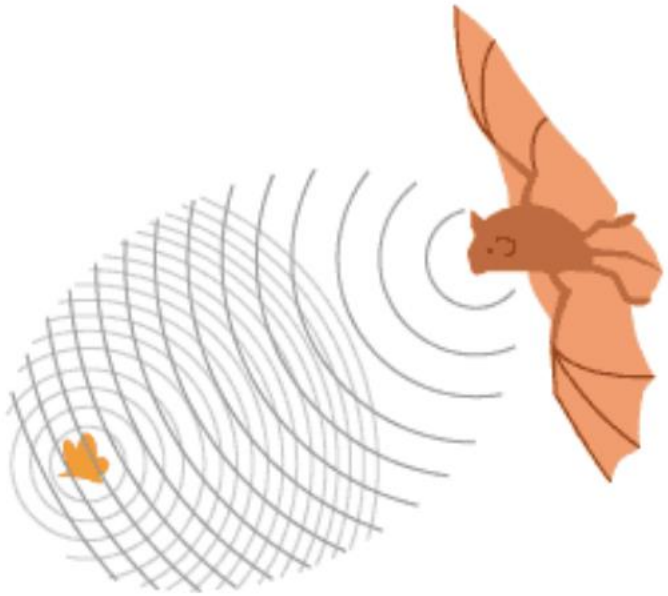


Neural Coding in Biological Systems

Rate Coding
- Measurement of joint angles
- Detection of light levels

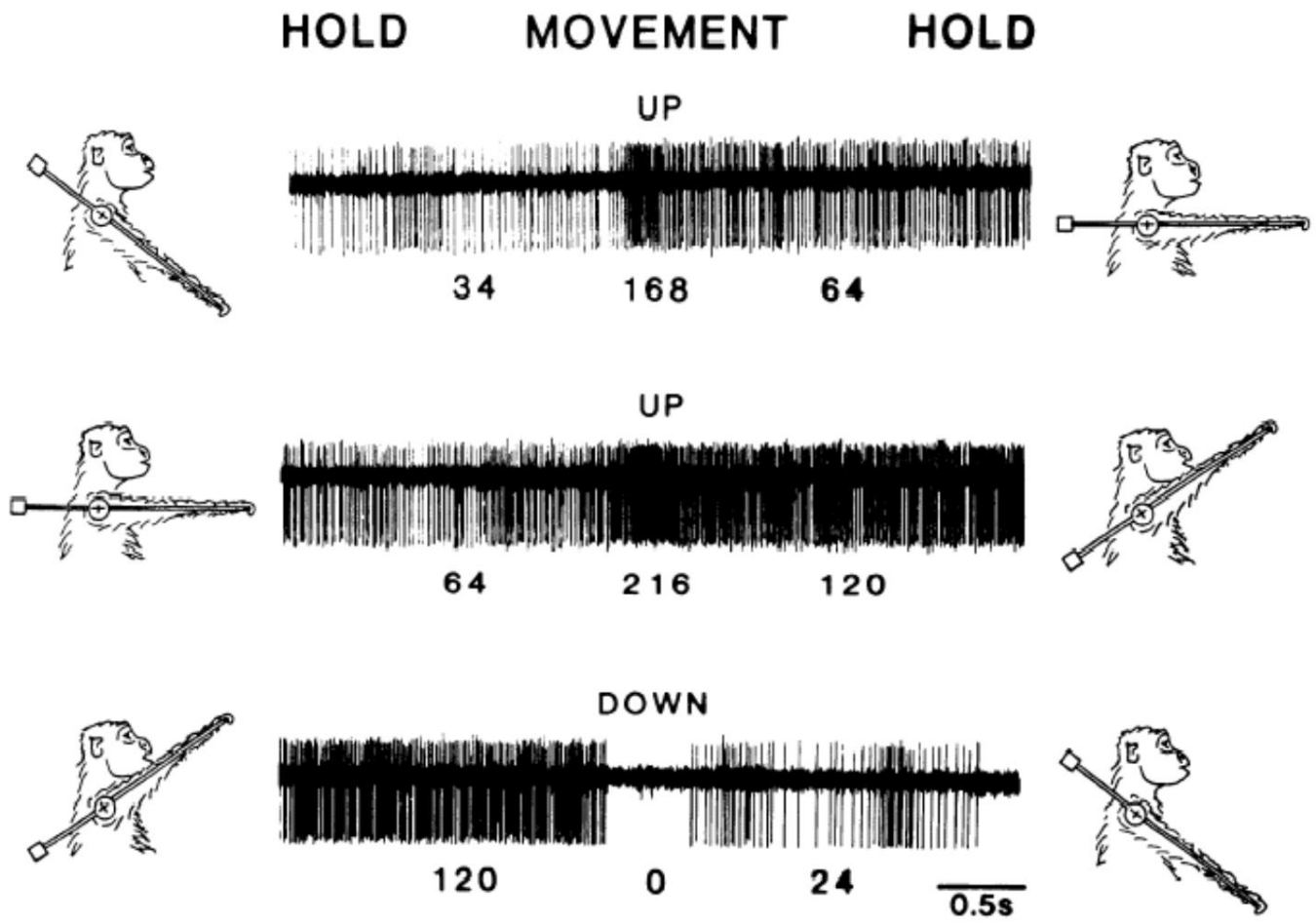
Temporal coding
- Binaural sound localisation
- Echolocation in bats

Correlation coding
- Odour detection in insects
- Pattern recognition



Measurement of Joint Angles

Van Kan et al (1997): Found that the mossy fibres in the cerebellum use rate coding to represent joint angles.



Binaural sound encoding – Jeffress Model

Jeffress (1948) proposed a time-delay neural network that uses delay lines and coincidence detectors to detect the direction of a sound.

1. temporally-coded input signals consisting of spikes that are time-locked to the waveform of the acoustic stimulus,
2. two sets of tapped conduction delay lines that differentially delay these monaural neural patterns that are then fed into
3. an array of binaural spike coincidence detectors, whose outputs are then inputs into
4. coincidence counters that provide the number of coincidences as a function of relative delay.