Intro to Machine Learning (ML) for Earth Scientists

Day 1

Lauren Hoffman
LPHYS2268, Winter 2025

Lecture 1: What is the machine learning?

- Motivation
- Case study
- Types of ML
- How it works
- *Tutorial: Building a ML model*

Lecture 2: ML Applications in Earth and Climate Science

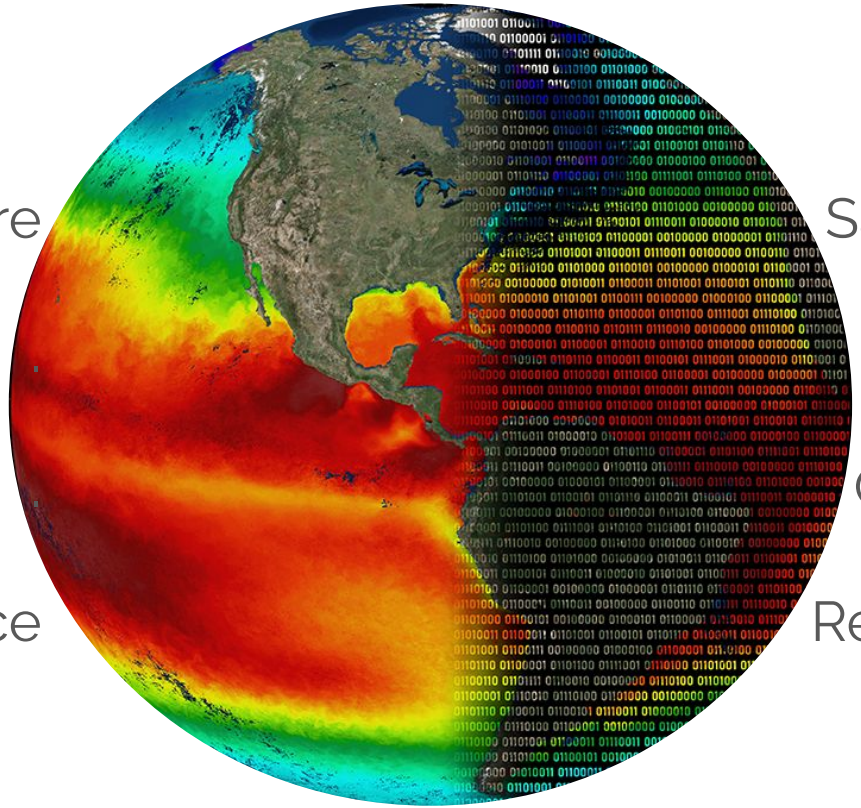
- Applications
- Challenges
- *Activity: literature dive*

https://github.com/lahoffman/ml_lectures_LPHYS2268

“The figurative Cambrian explosion of AI techniques in Earth and climate sciences, however, only began over the last five years and will rapidly continue throughout the coming decades.”

-Irrgang et al. (2021)

The Earth system has a vast (and growing!) array of available data from a variety of observation systems.



Atmosphere

Ocean

Land

Ice

Satellite

In situ

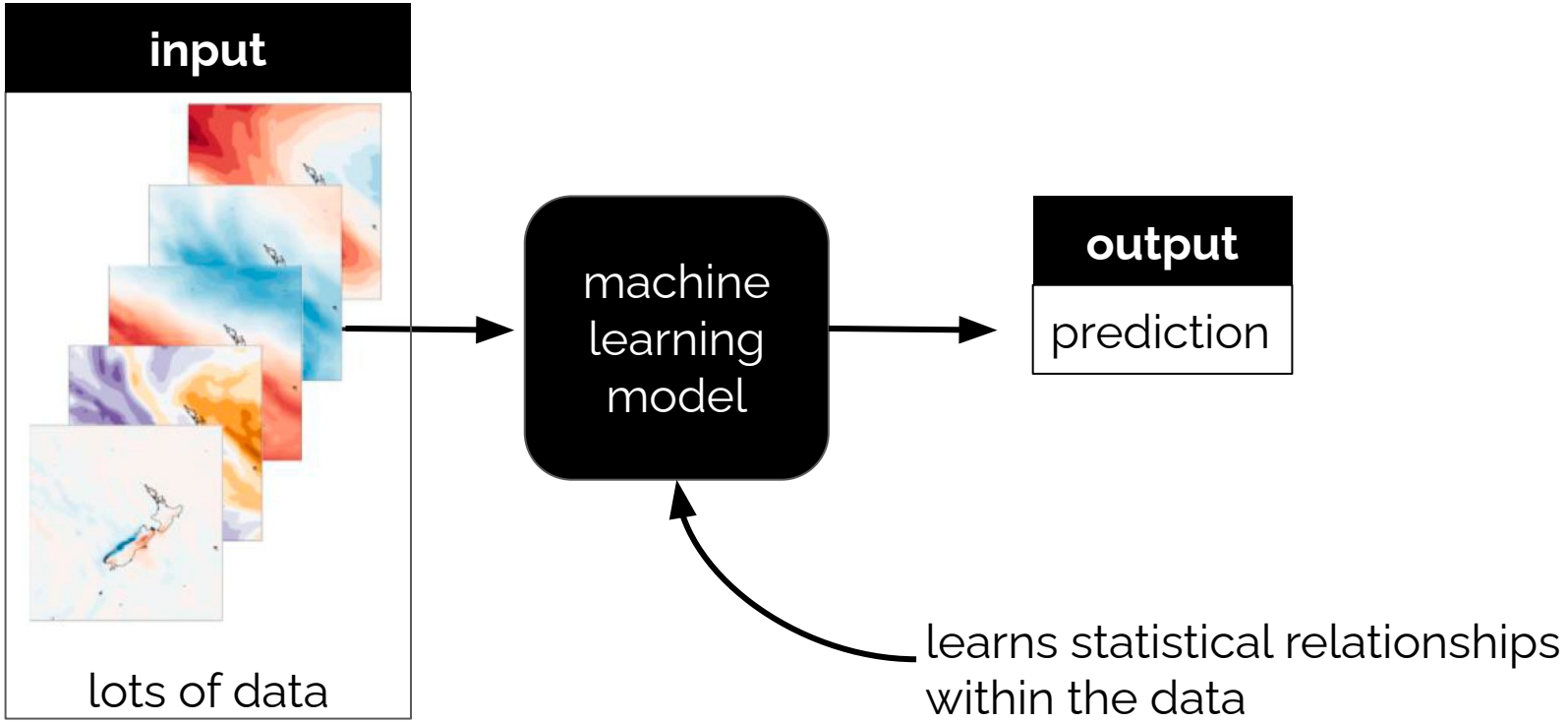
Climate models

Reanalysis products

ML ❤ DATA

Image: Courtesy of Kate Culpepper with design elements provided by Esri, HERE, Garmin, FAO, NOAA, USGS, EPA.

Machine learning is a way for computers to learn from data.

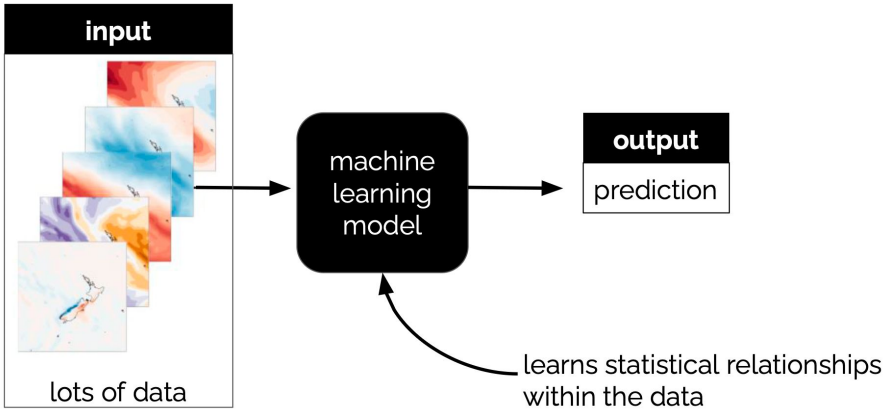


Rampal et al. (2024)

Instead of explicitly programming rules, we use data to teach the system to recognize patterns and make predictions or decisions based on those patterns.

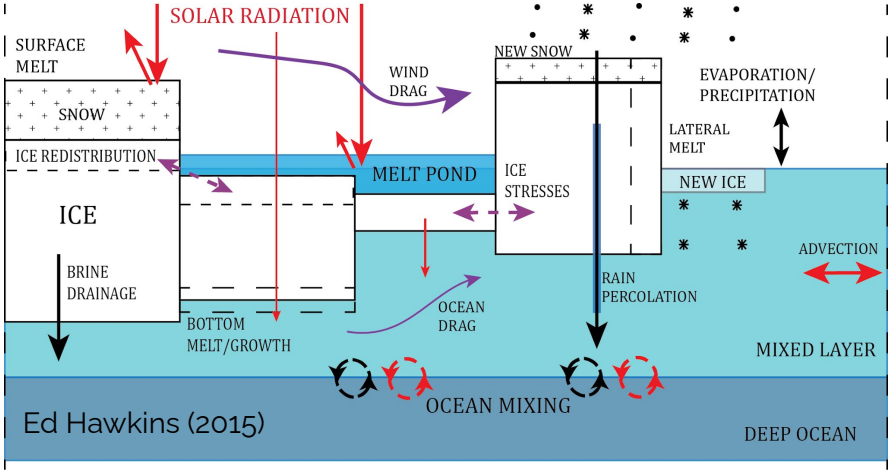
ML models learn statistical relationships in data, whereas numerical models are based on prescribed physics.

Machine Learning Models



Rampal et al. (2024)

Numerical Models



ML models *learn from data* to recognize patterns and make predictions based on those patterns.

Numerical models are *prescribed dynamical equations* and predict based on physics.

Machine learning is a useful tool to predict and understand sea-ice motion in the Arctic.

Lauren Hoffman[1], Matthew Mazloff[1], Sarah Gille[1], Donata Giglio[2], Cecilia Bitz[3], Patrick Heimbach[4]
[1] Scripps Institution of Oceanography, [2] University of Colorado Boulder, [3] University of Washington, [4] University of Texas at Austin

Scan for the manuscript!

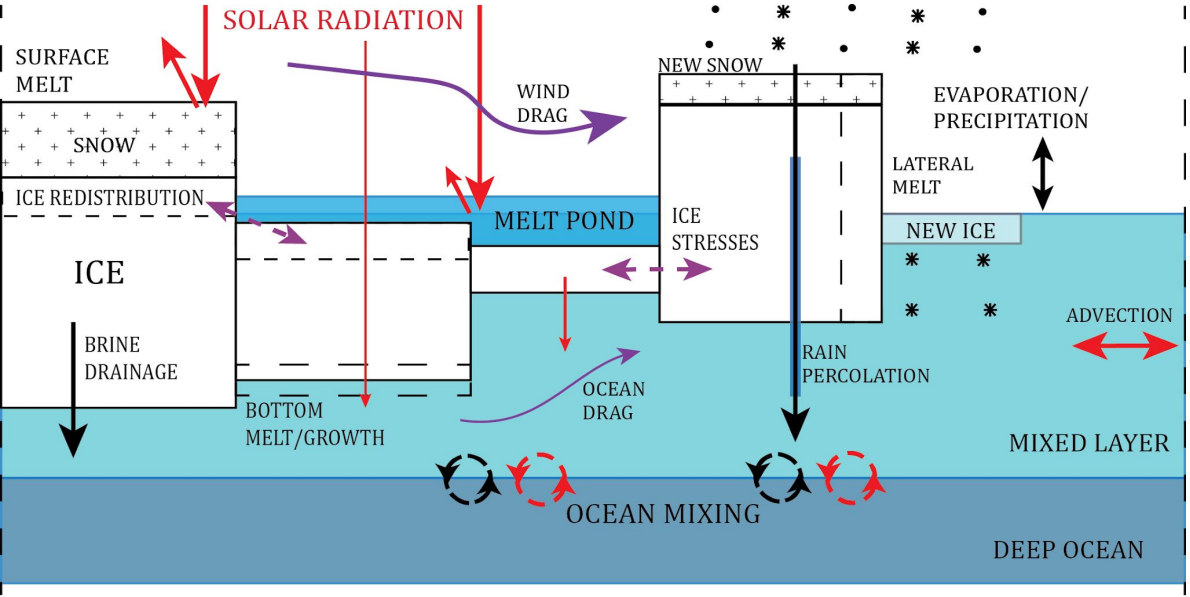
Office of Naval Research Science & Technology

SCRIPPS INSTITUTION OF OCEANOGRAPHY

UC San Diego JACOBS SCHOOL OF ENGINEERING

Scan for the manuscript!

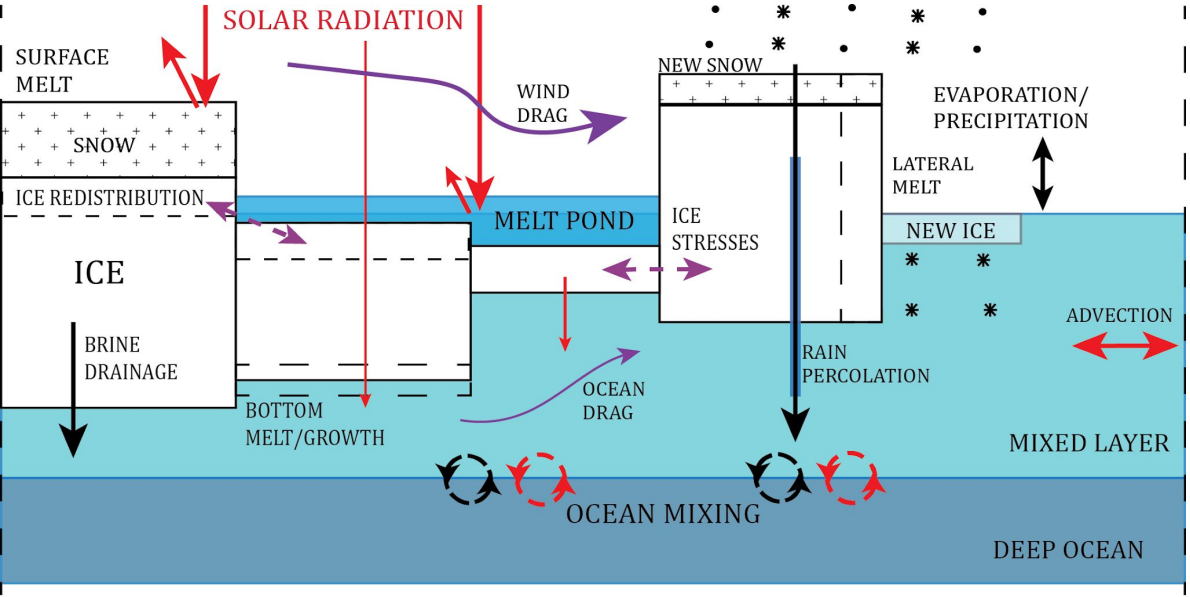
Machine learning models for sea-ice drift have fewer complexities and a lower computational cost than traditional physics-based models.



Physical processes now included in state-of-the-art sea ice models such as CICE (Ed Hawkins, 2015).

Hoffman et al. *AIES.* (2023).

Machine learning models for sea-ice drift have fewer complexities and a lower computational cost than traditional physics-based models.



Physical processes now included in state-of-the-art sea ice models such as CICE (Ed Hawkins, 2015).

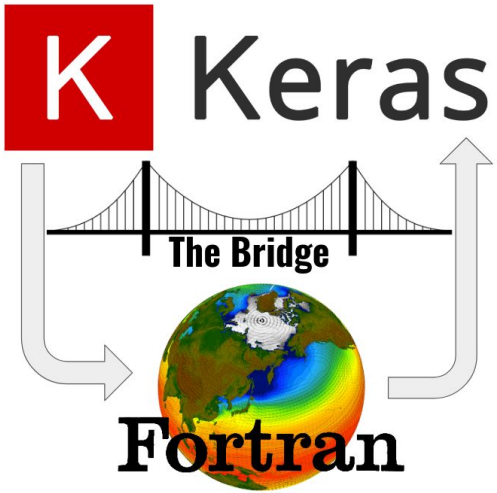
Machine Learning models can be used to understand sea-ice motion because they are drawing information from the data.

Hoffman et al. *AIES.* (2023).

Machine learning models can be useful for...

Modeling

As a surrogate model to parameterize sea-ice motion.

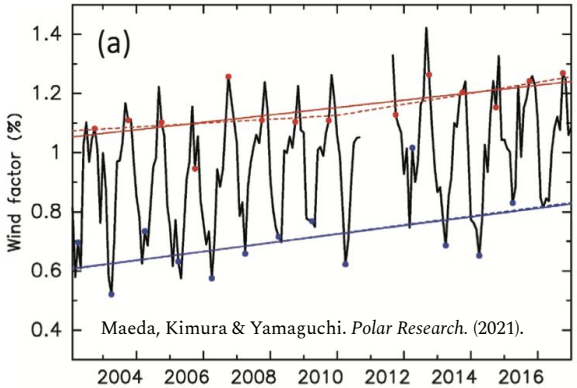


Forecasting

Predictions of sea-ice motion can be used in operational forecasting.

Knowledge Discovery

ML draws information from data, and can perhaps be used to elucidate unrealized physical interactions.

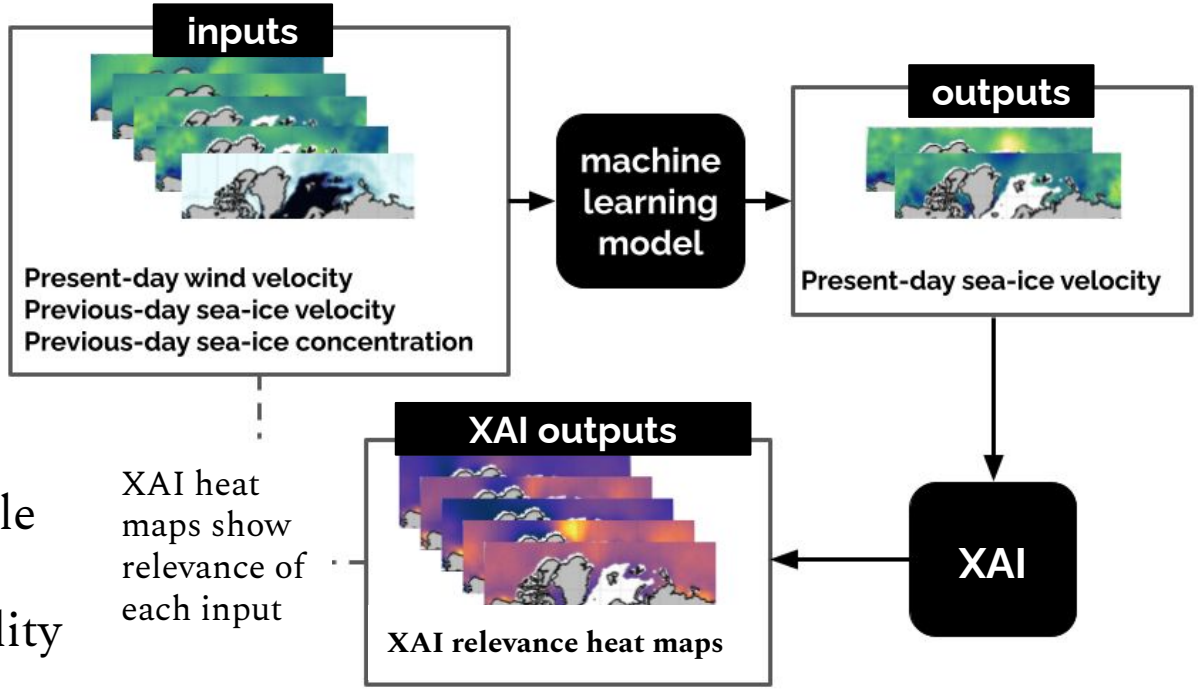


The wind factor is increasing!

This study assesses the viability of using machine learning techniques to predict and understand ice motion.

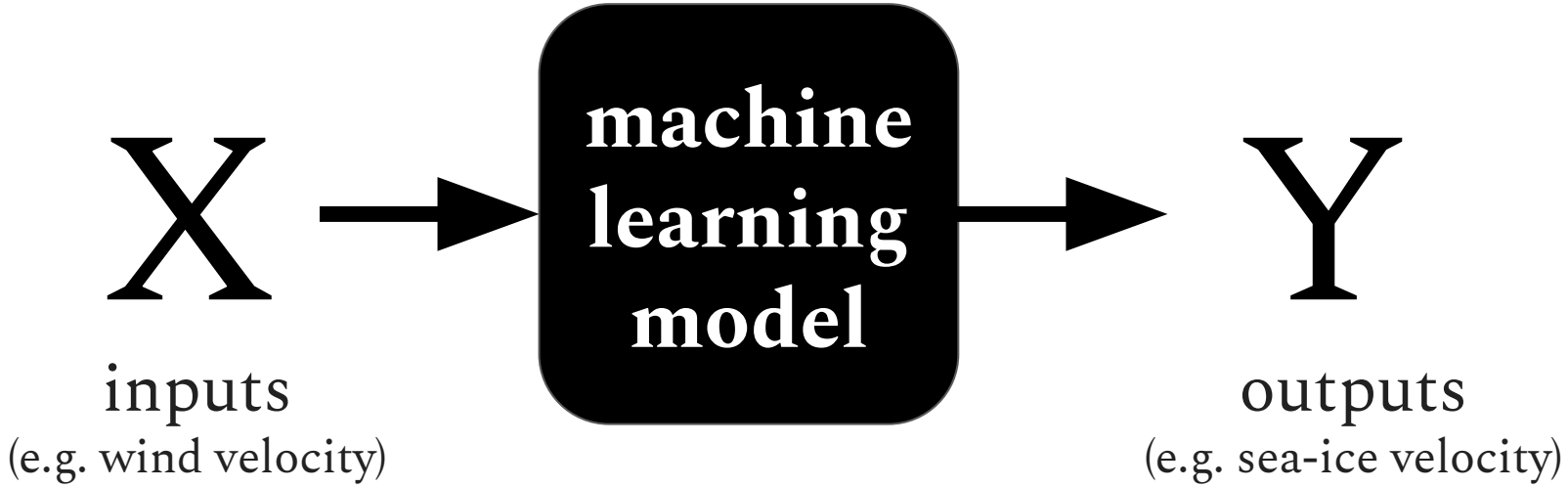
Predictability. Can machine learning models make skillful one-day predictions of sea-ice velocity?

Understanding sea-ice motion. Can we use eXplainable AI (XAI) methods to understand changes in the drivers of variability in ice motion?



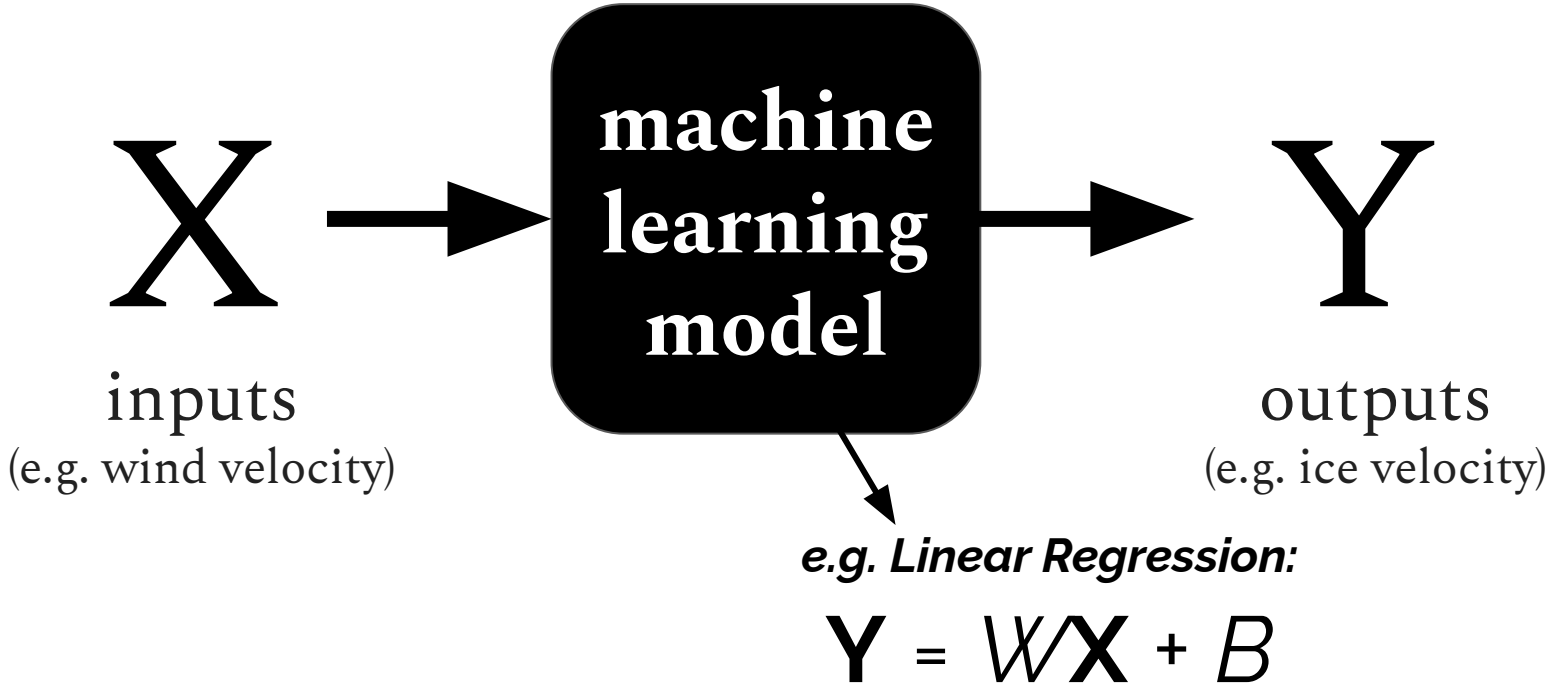
Hoffman et al. *AIES.* (2023).

Machine learning models learn statistical relationships between the inputs and outputs.

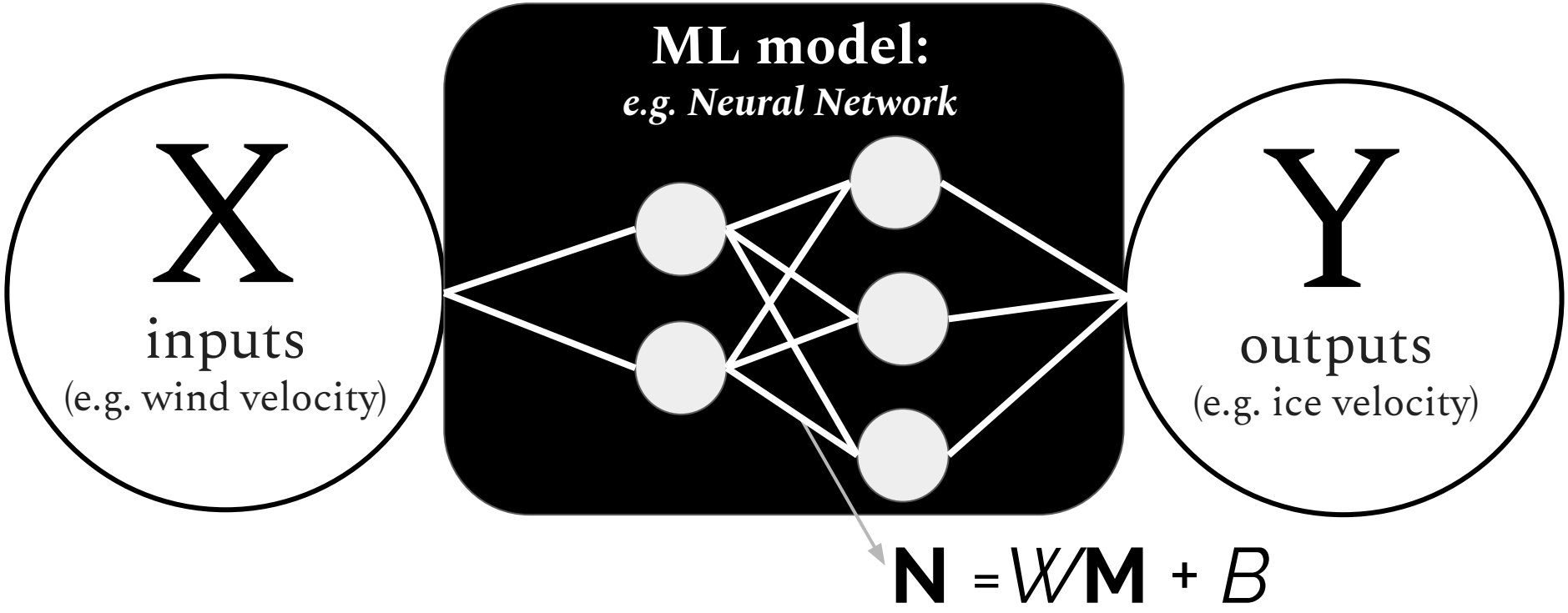


Train | Validate | Test

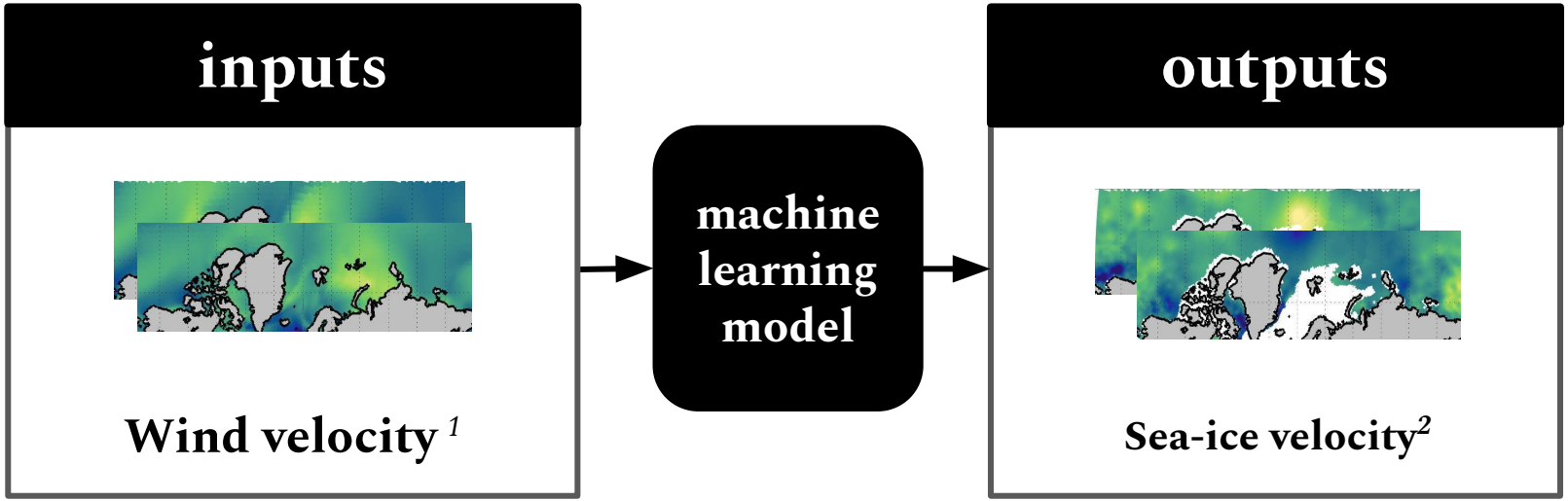
Machine learning models learn statistical relationships between the inputs and outputs.



Machine learning models learn statistical relationships between the inputs and outputs.



A convolutional neural network (CNN) works well with maps and can incorporate *non-linear* and *non-local* interactions within the inputs to predict the output.

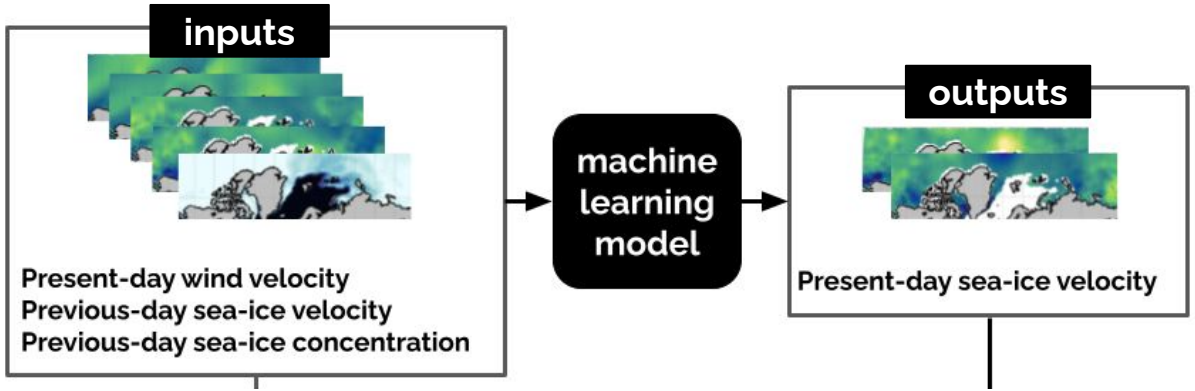


Model Inputs: [1]JRA55; [2]Polar Pathfinder

Hoffman et al. *AIES.* (2023).

Machine learning is a useful tool to predict sea ice motion in the Arctic.

Predictability. Can machine learning models make skillful one-day predictions of sea-ice velocity?



Hoffman et al. *AIES.* (2023).

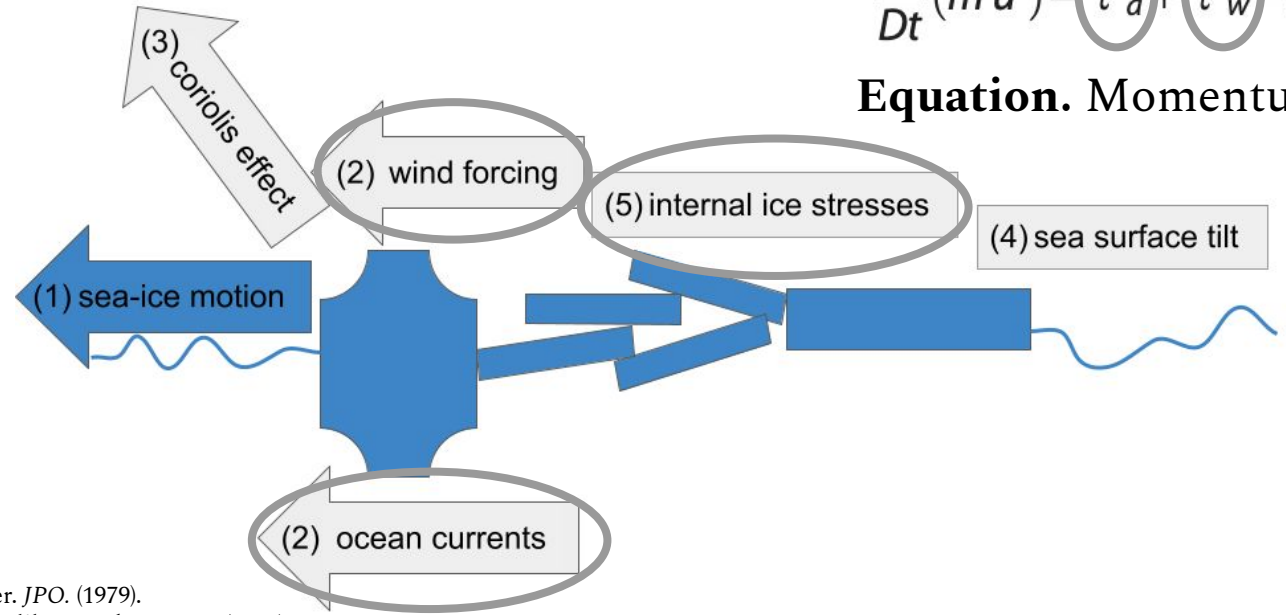
Predictability

A *useful* machine learning model is a *skillful* one.

Sea-ice motion is largely determined by oceanic & atmospheric forces and internal ice stresses.

$$\underbrace{\frac{D}{Dt}(m\vec{u})}_{(1)} = \underbrace{\vec{\tau}_a}_{(2)} + \underbrace{\vec{\tau}_w}_{(2)} - \underbrace{mf\vec{k}\times\vec{u}}_{(3)} - \underbrace{mg\vec{\nabla}H}_{(4)} - \underbrace{\nabla\cdot\sigma}_{(5)}.$$

Equation. Momentum balance for sea ice.



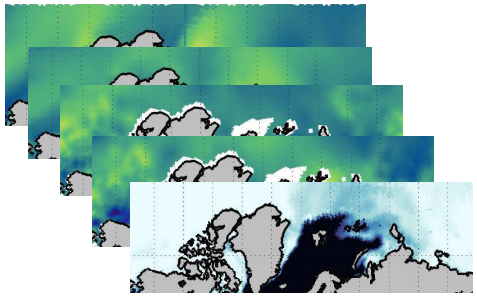
Hibler. *JPO.* (1979).
Thorndike & Colony. *JGR.* (1982).

A machine learning model makes one-day predictions of sea-ice velocity given input data from satellite & reanalysis sources (1989-2021).

$$\frac{D}{Dt}(m\vec{u}) = \vec{\tau}_a + \vec{\tau}_w - mf\vec{k}\times\vec{u} - mg\vec{\nabla}H - \nabla\cdot\sigma.$$

Equation. Momentum balance for sea ice.

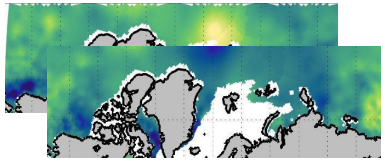
inputs



Present-day wind velocity [1]
Previous-day sea-ice velocity [2]
Previous-day sea-ice concentration [3]

machine learning model

outputs



Present-day sea-ice velocity

Model Inputs: [1]*JRA55;* [2]*Polar Pathfinder;* [3]*Nimbus-7*

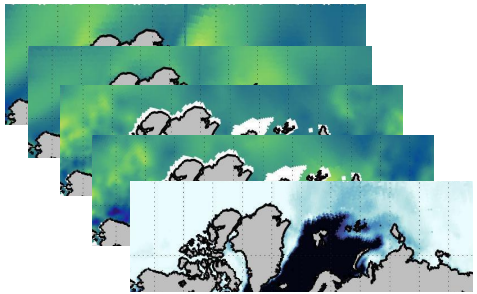
Hoffman et al. *AIES.* (2023).

We compare a suite of machine learning models.

Multiple Linear Regression (LR):

$$\bar{u}_{i,\,t=1} = A\bar{u}_{w,\,t=1} + B\bar{u}_{i,\,t=0} + Cc_{t=0}$$

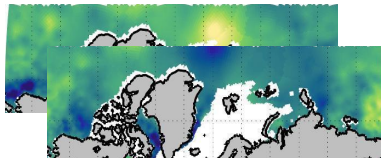
inputs



Present-day wind velocity
Previous-day sea-ice velocity
Previous-day sea-ice concentration

machine learning model

outputs

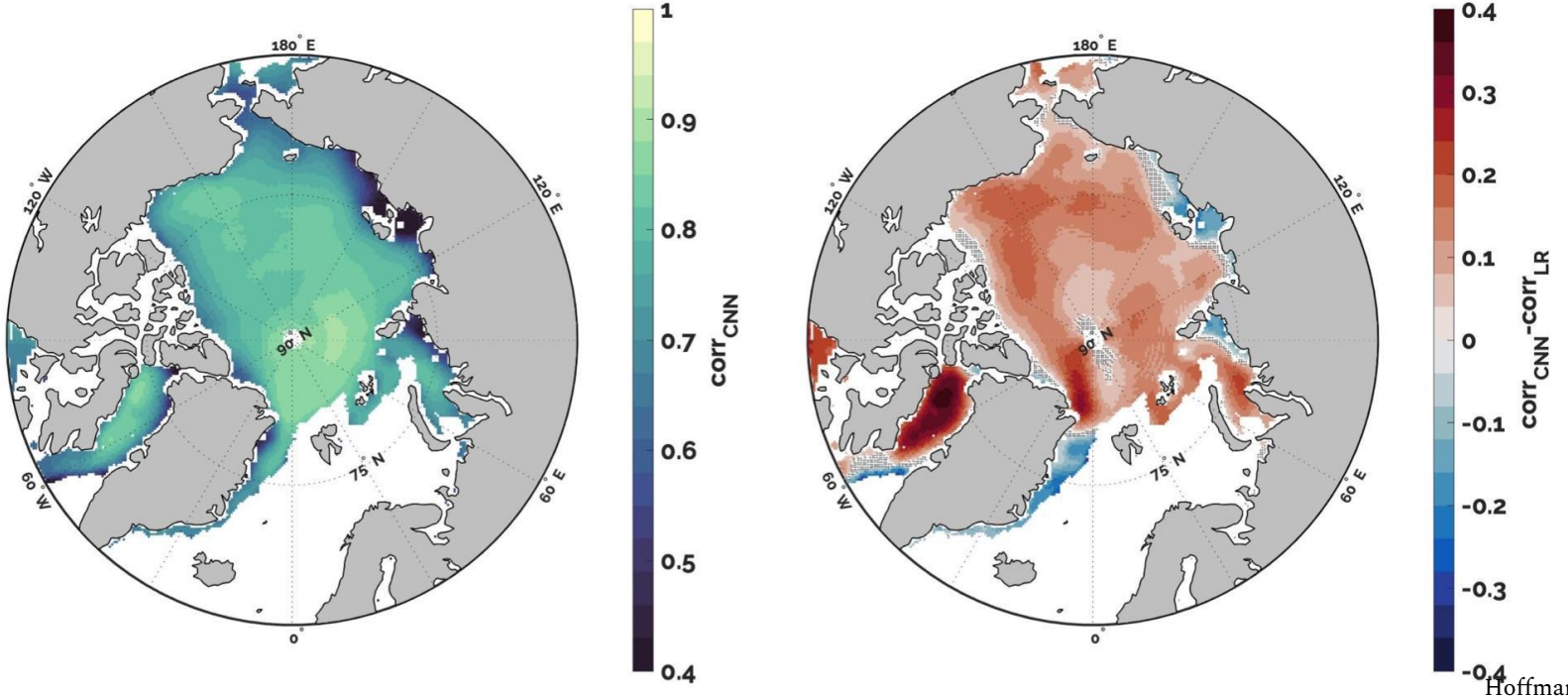


Present-day sea-ice velocity

Convolutional Neural Network (CNN)

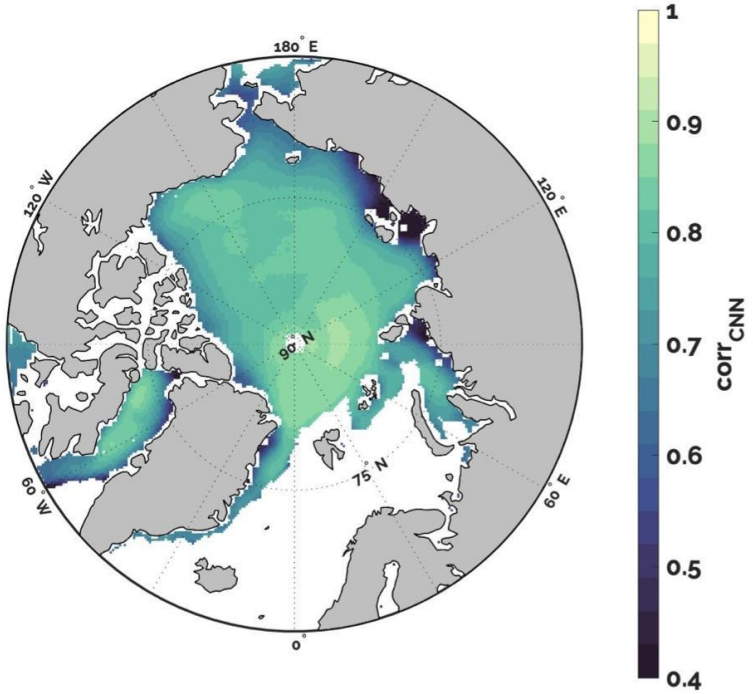
Hoffman et al. *AIES.* (2023).

Models that incorporate non-linear relationships between inputs and non-local information capture important information (i.e. $corr_{CNN} > corr_{LR}$).



Hoffman et al. *AIES.* (2023).

A *useful* machine learning model is a *skillful* one.



Predictability

Machine learning models make *skillful* one-day predictions of ice motion.

Hoffman et al. *AIES.* (2023).

General Introduction to ML

What is Machine Learning?

"Field of study that gives computers the ability to learn without being explicitly programmed."
-coined by Arthur Samuel in 1959

Fitting a model to data

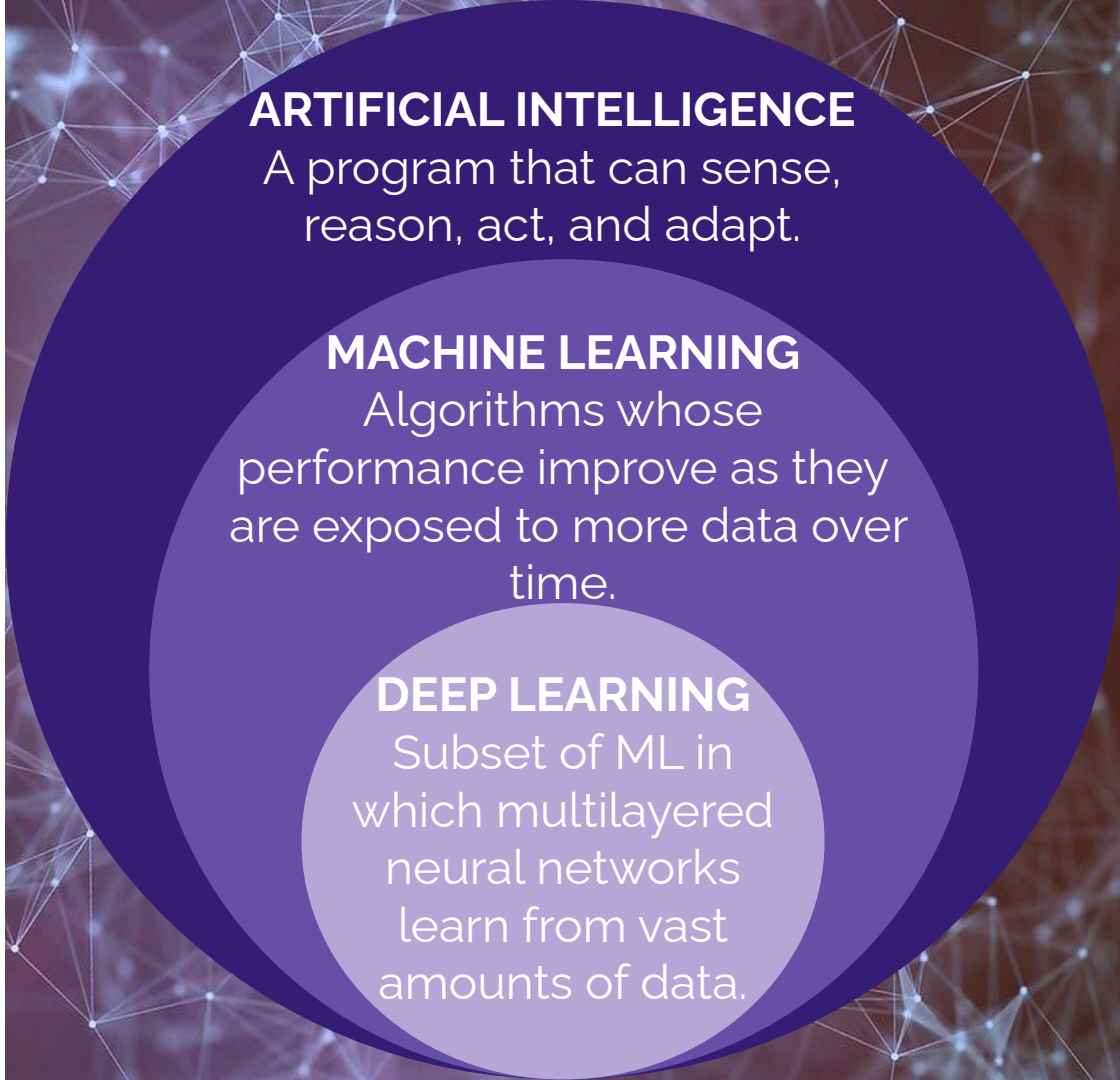
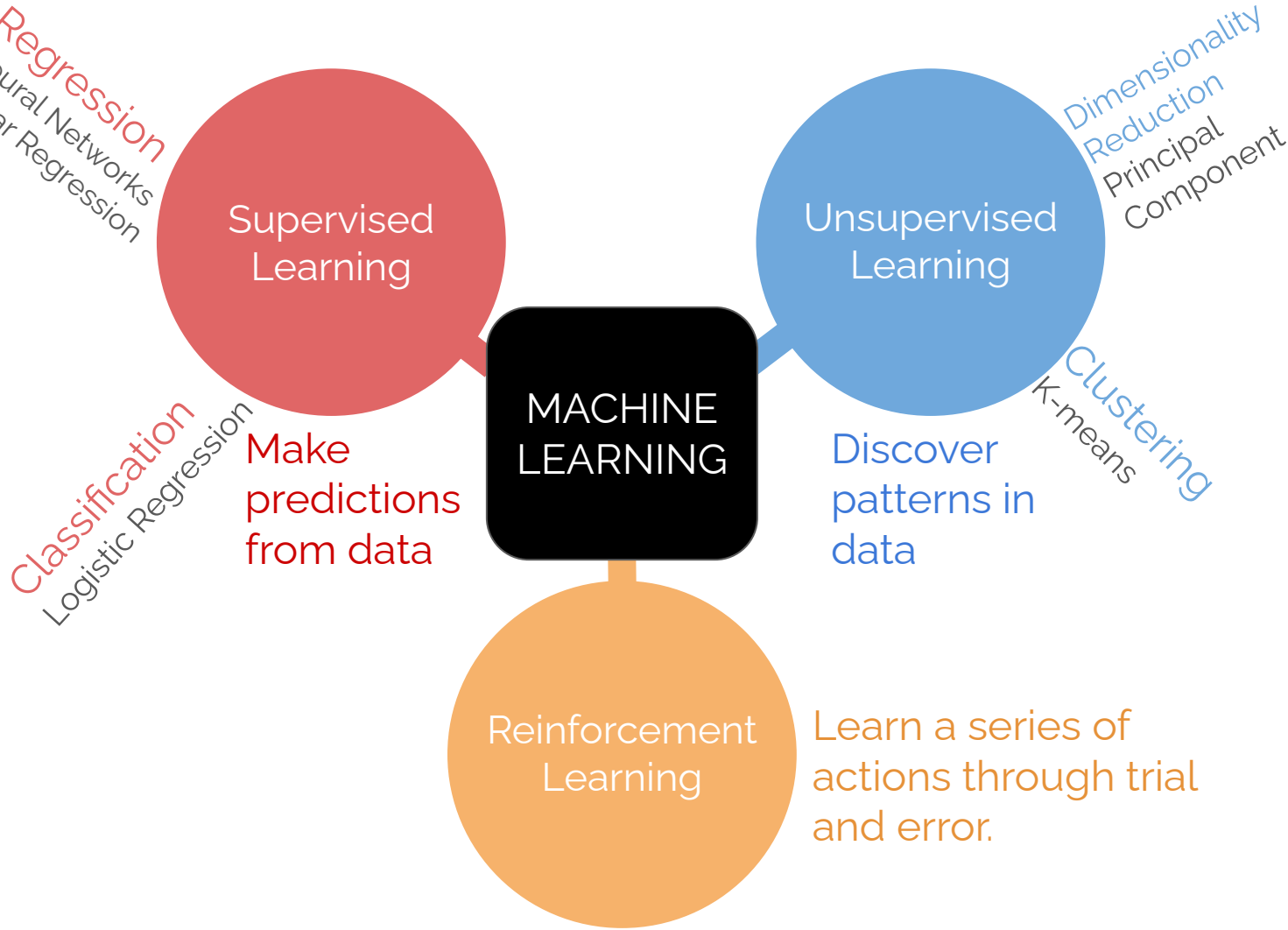
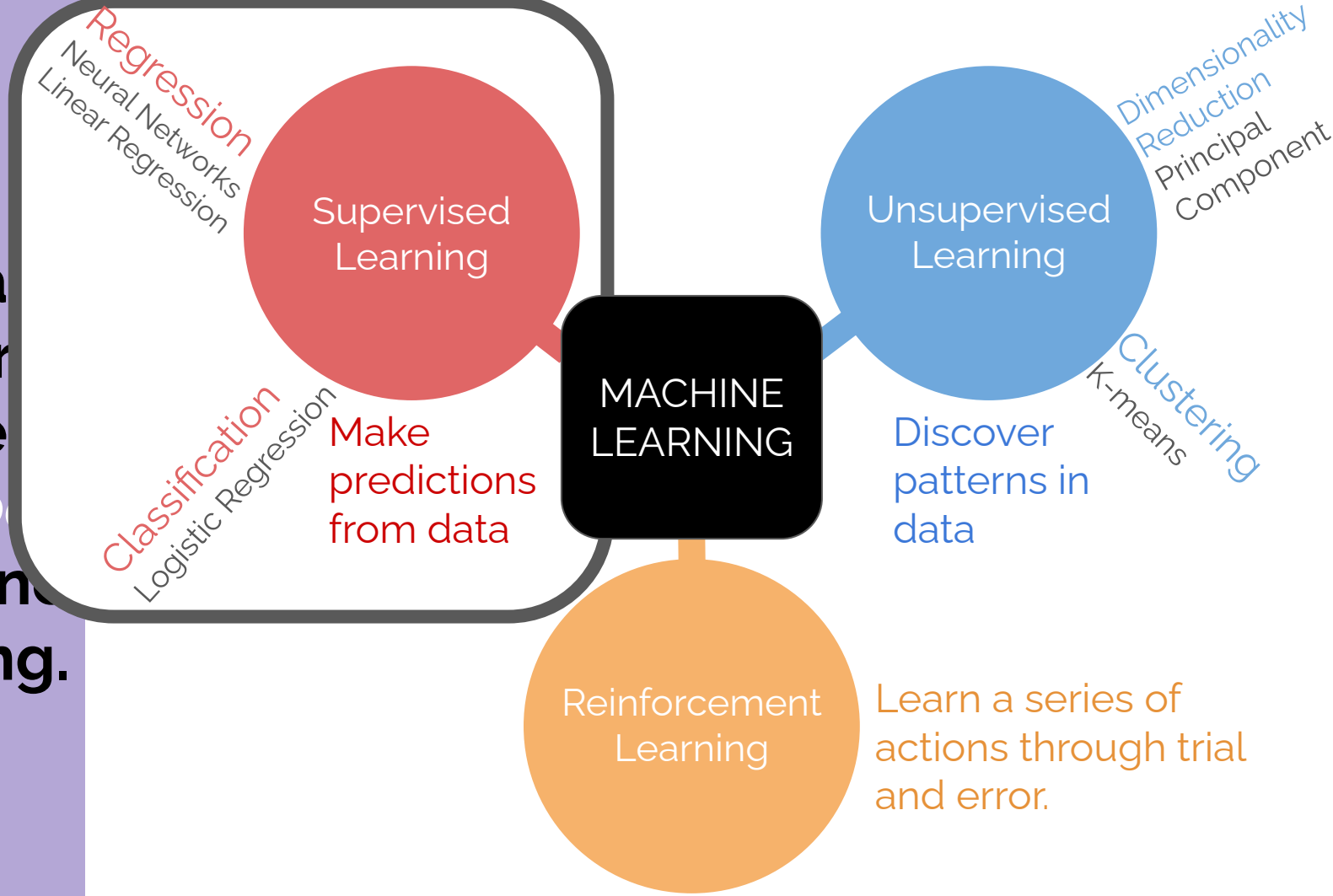


Figure adapted from: Balodi, Arun. (2020).

There are several different *types* machine learning.

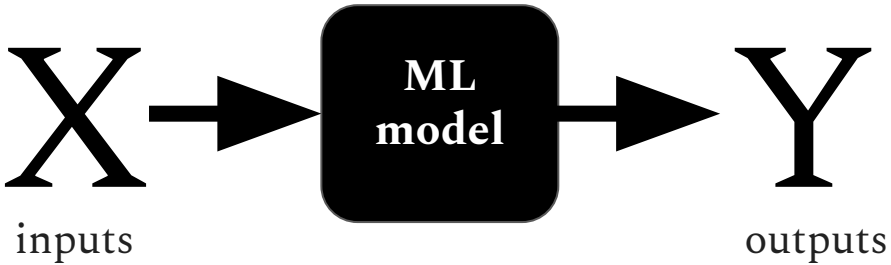


There a
sever
differe
typ
machin
learning.



Supervised learning is used for *making predictions* from a set of *input and output* data.

Models learn statistical relationships between the inputs and outputs.



Supervised Learning

Classification

&

Regression

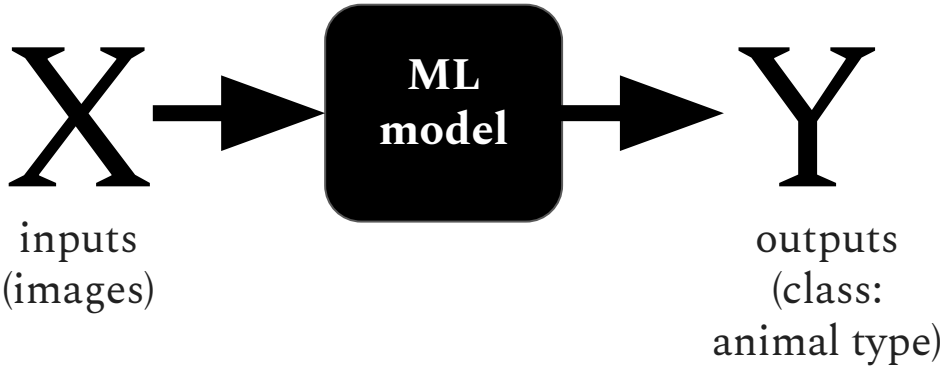
Classification models predict probabilities of classes.

Cat [97%]

Fish [0%]

Deer [3%]

Butterfly [0%]



Supervised Learning

Classification

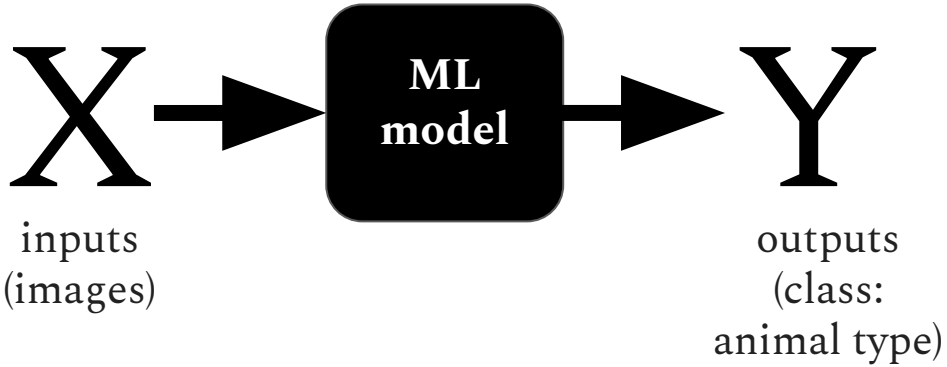
Classification models predict probabilities of classes.

Cat [40%]

Fish [35%]

Deer [10%]

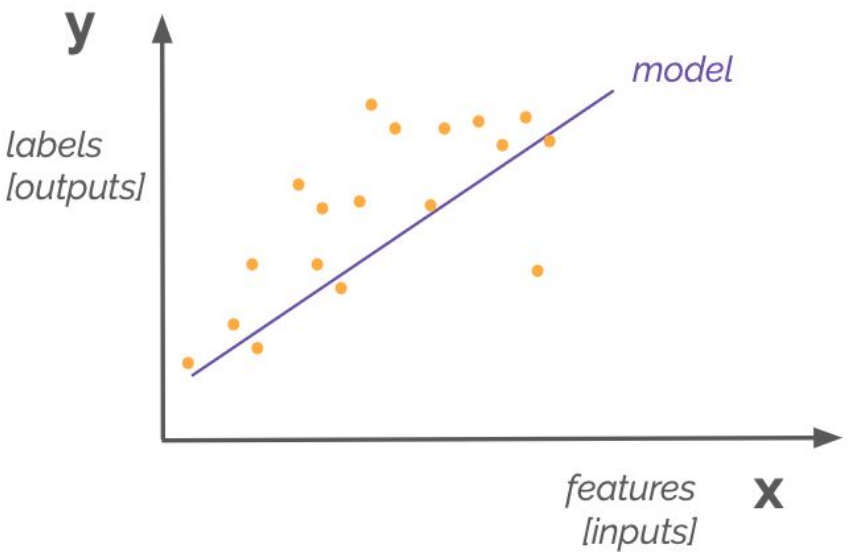
Butterfly [15%]

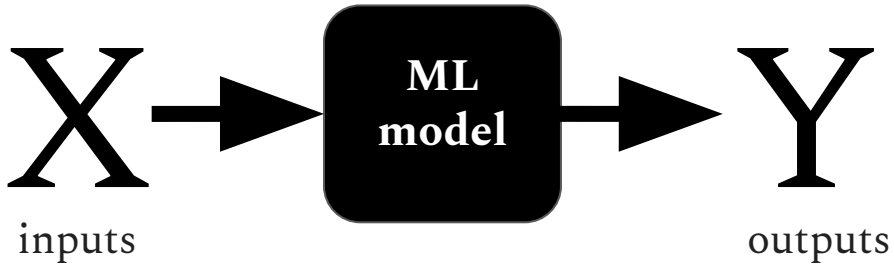


Supervised Learning

Classification

Regression models predict real, continuous values.



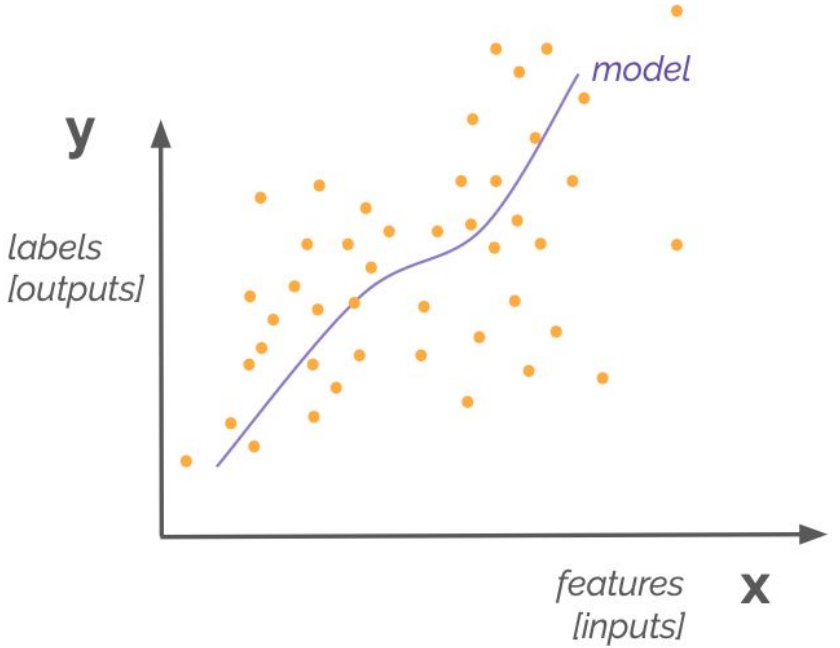


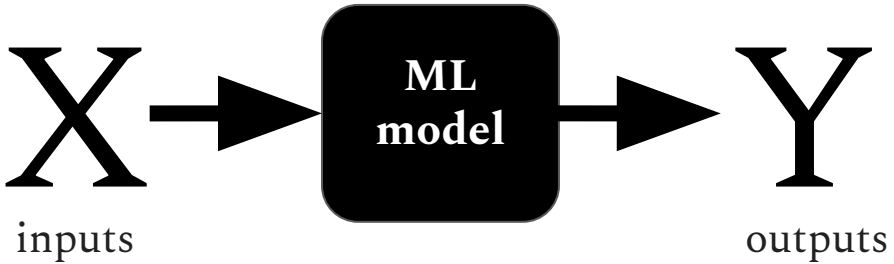
Supervised Learning

Regression:

Linear Regression

Regression models predict real, continuous values.





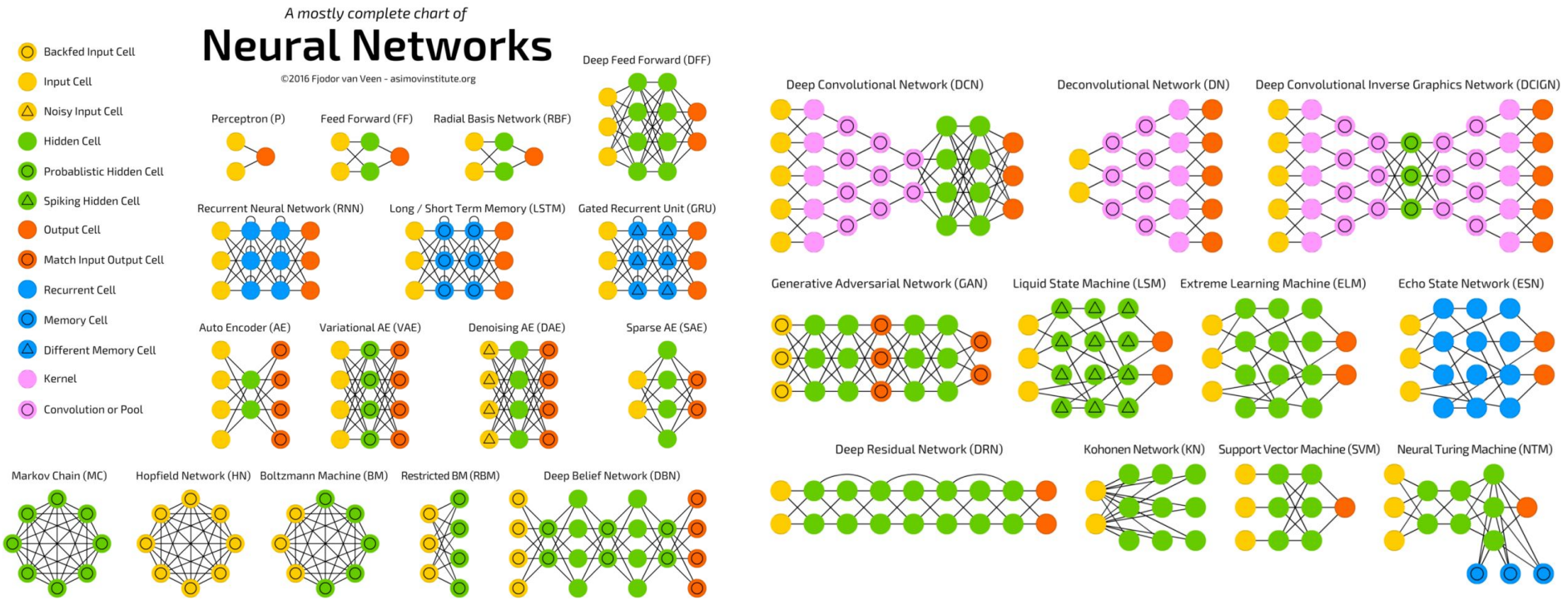
Supervised Learning

Regression:

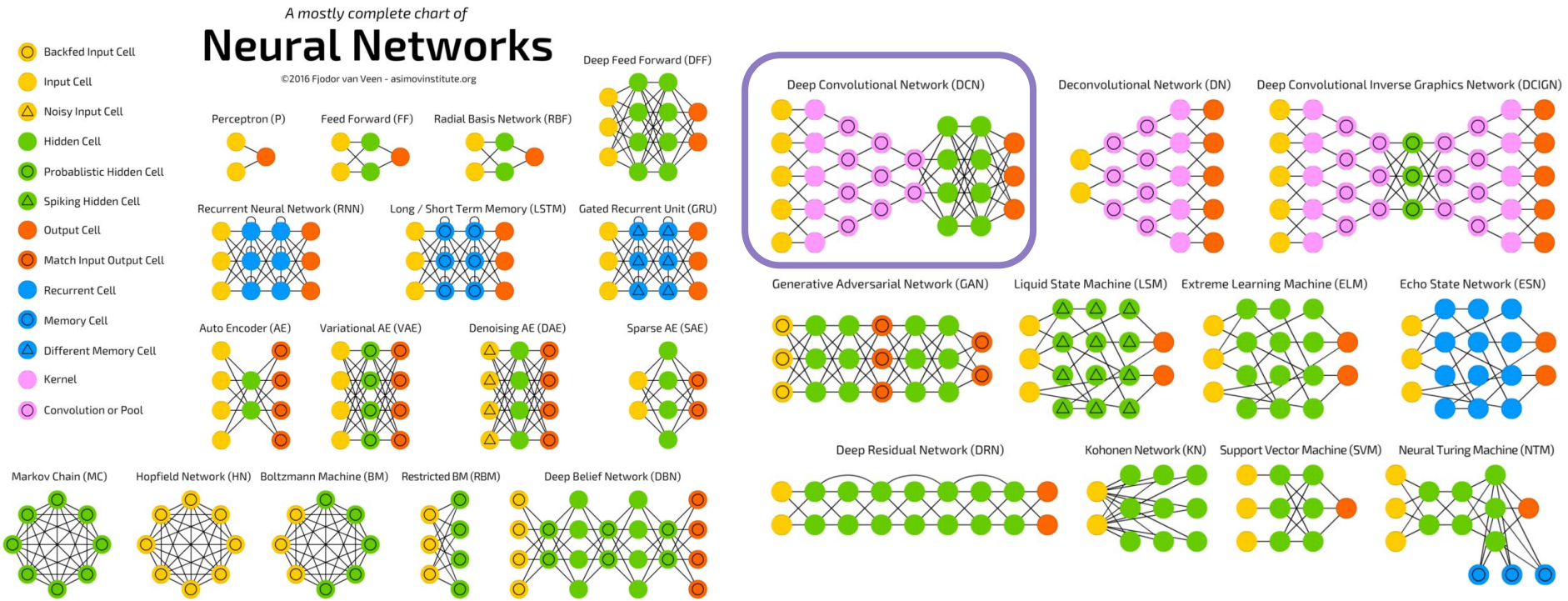
Non-linear Regression,

Neural networks

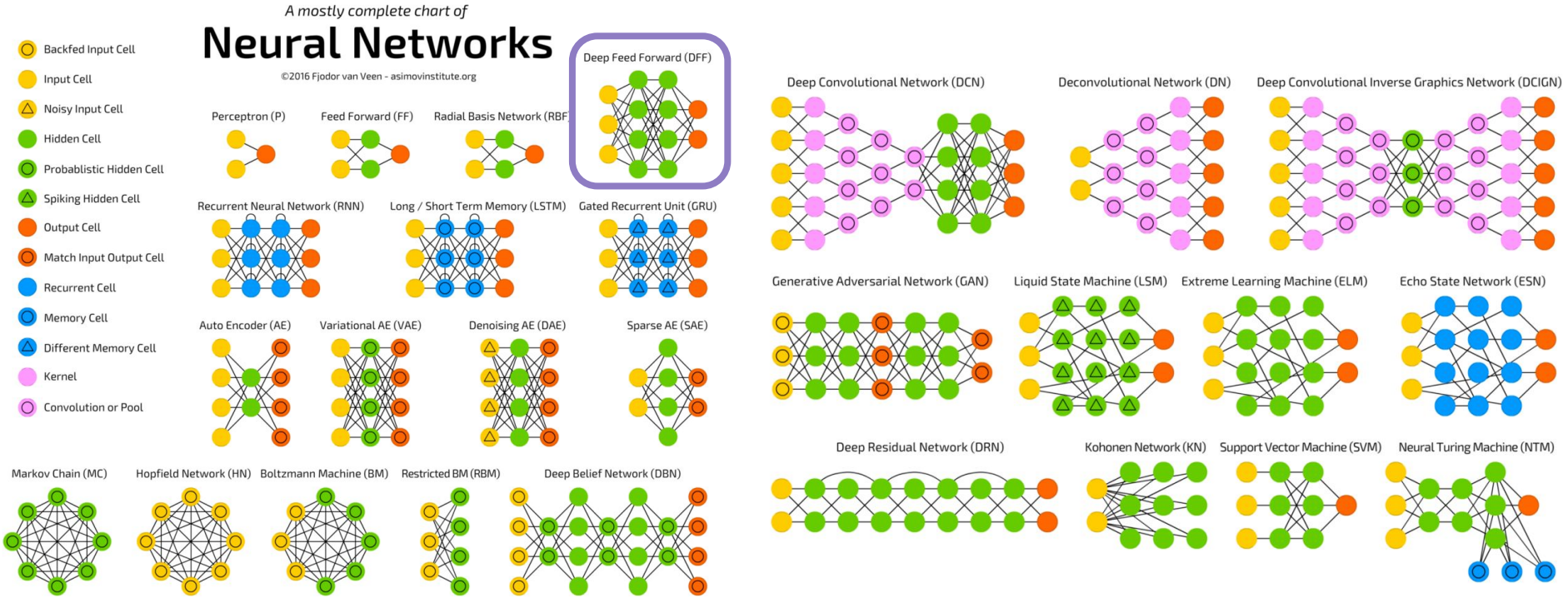
There are many different types of neural networks. Particular NNs can be beneficial depending on your application.



For example, my research on sea-ice motion used a *Convolutional Neural Network (CNN)* (here labeled DCN), which is particularly useful for images or image-like datasets (i.e. mapped products).

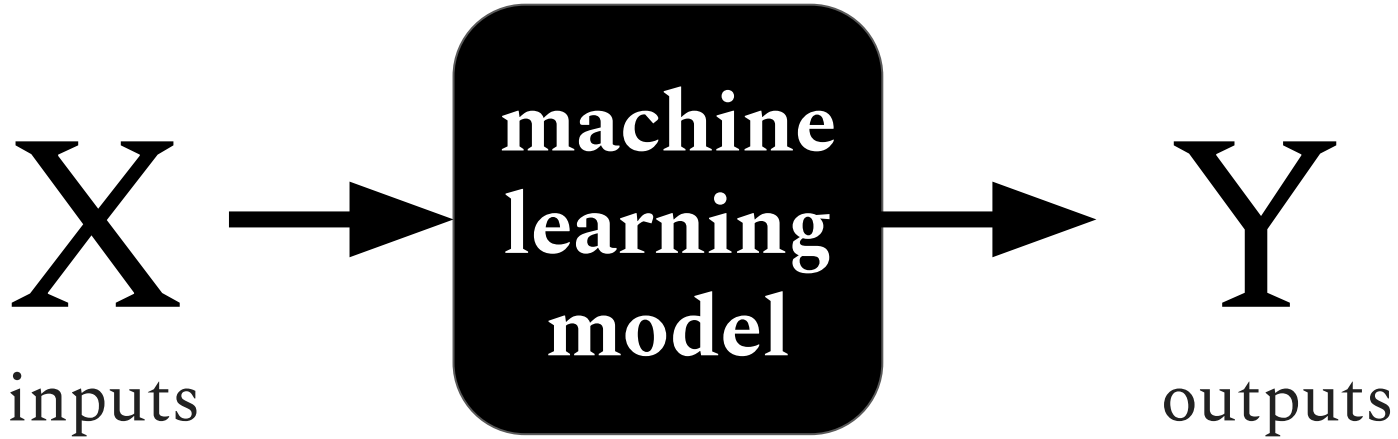


Today we will play with a tutorial that uses a *deep neural network* (here called DFF) with several hidden layers.



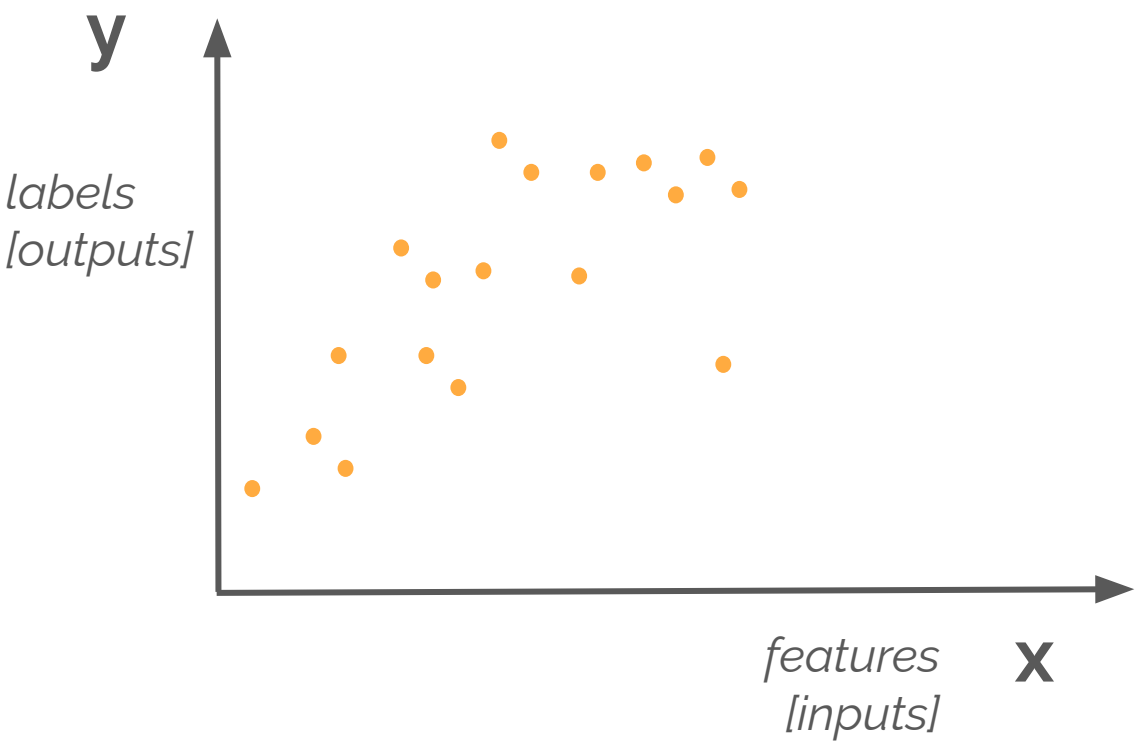
What is the machine learning?

In supervised learning, machine learning models learn statistical relationships between the inputs and outputs.

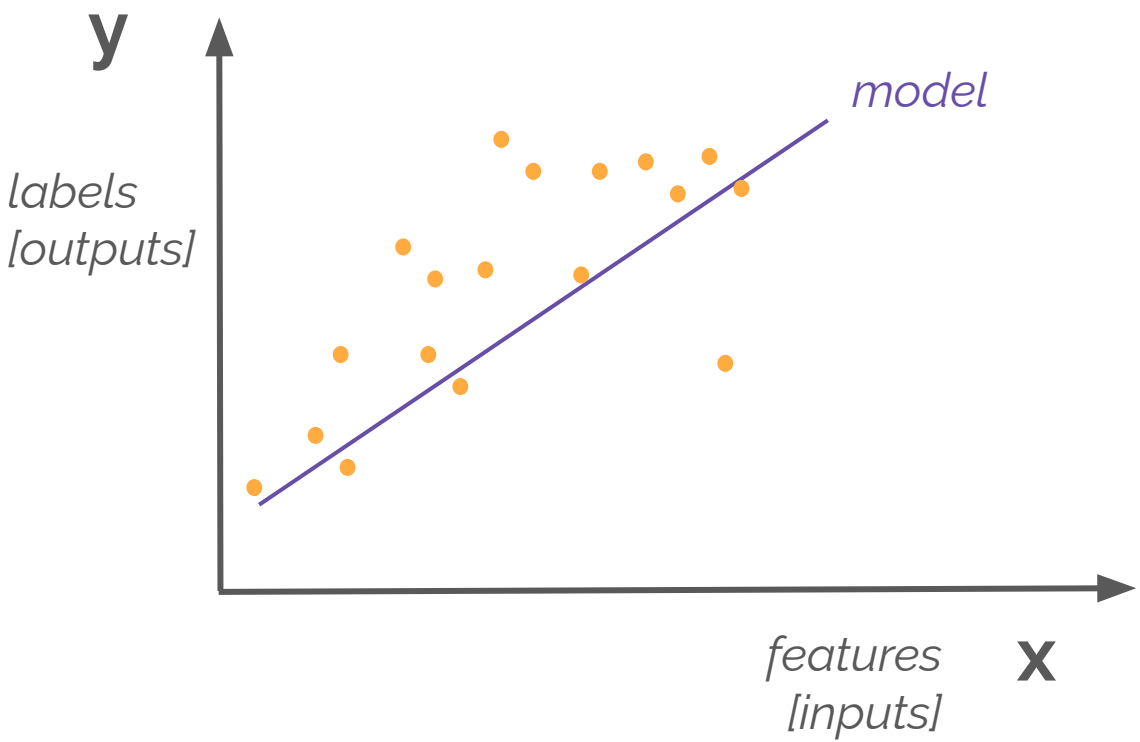


What is in the black box?

How do you predict y from x?

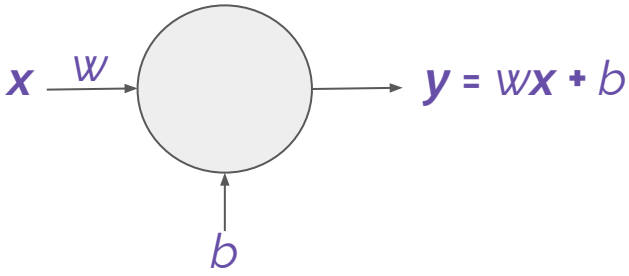


Linear regression is an example of least squares regression.



Linear Regression

Model Architecture:

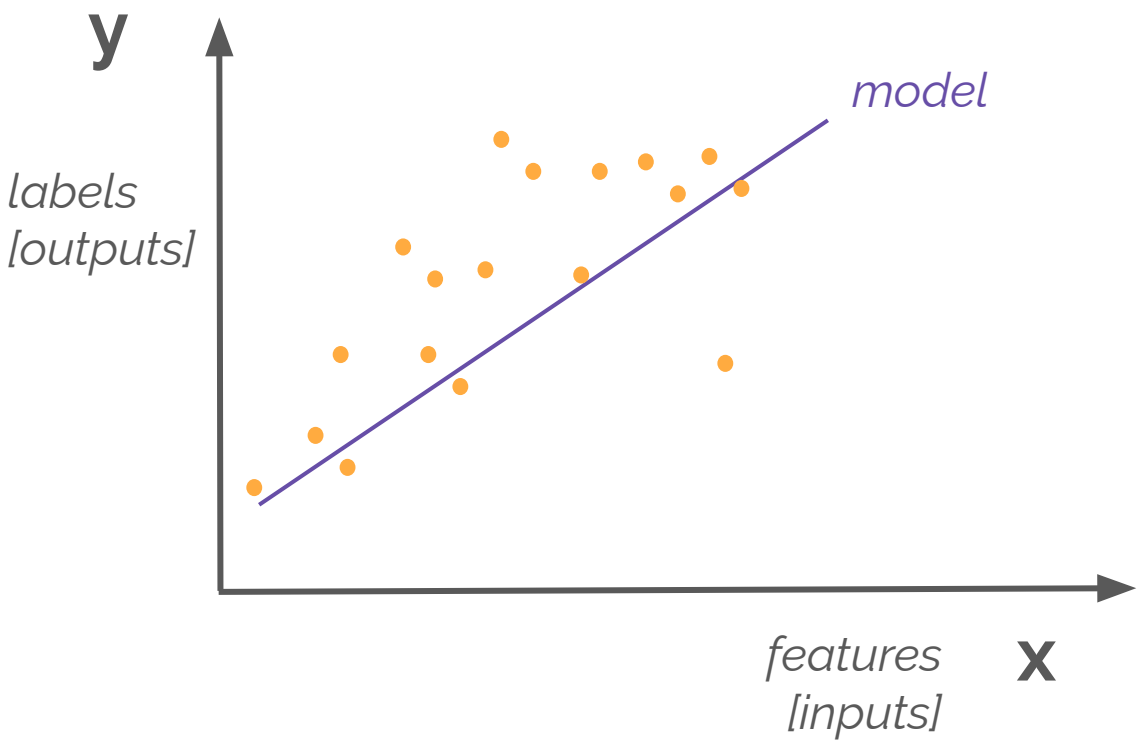


Model:

(w, b)

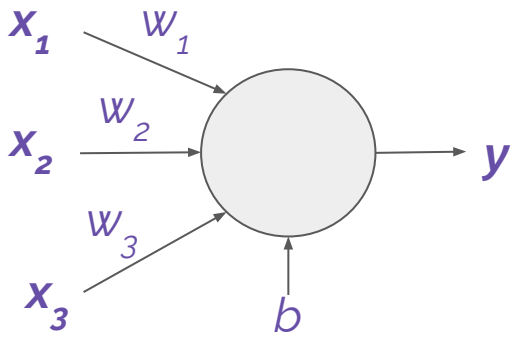
w = weight ; b = bias

Linear regression is an example of least squares regression.



Linear Regression

Model Architecture:



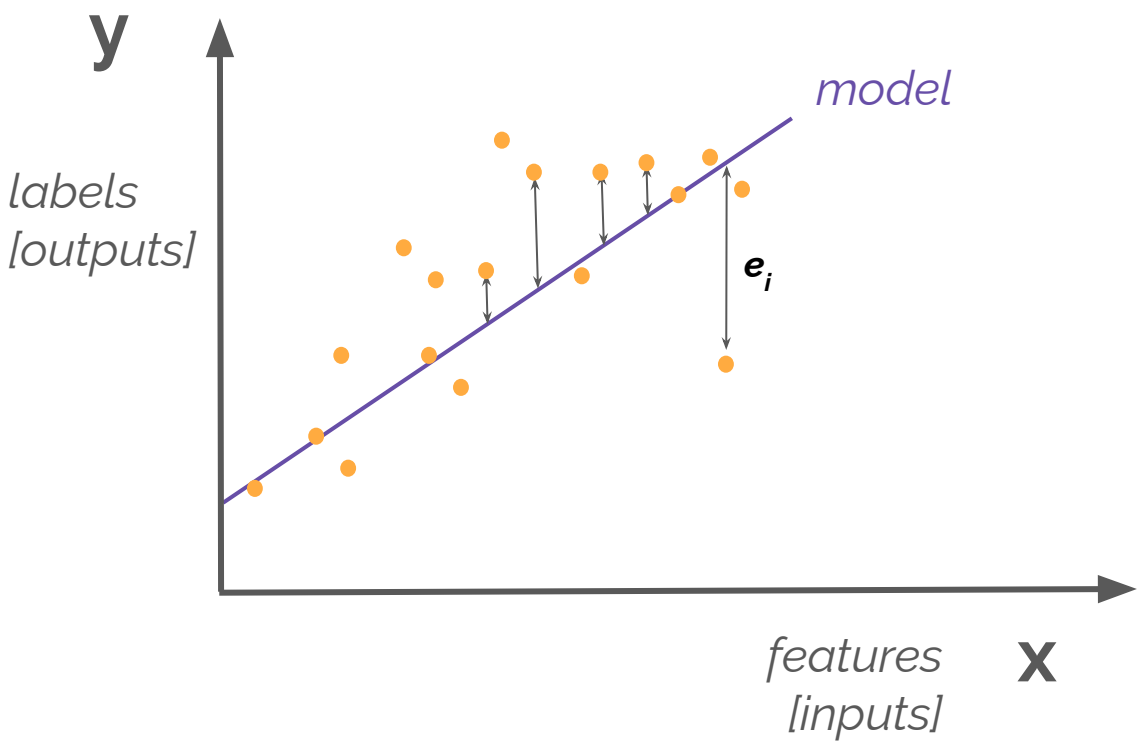
$$y = w_1x_1 + w_2x_2 + w_3x_3 + b$$

Model:

(w_1, w_2, w_3, b)

w = weight ; b = bias

A linear regression model *learns the weights and biases* by minimizing the cost function.



Linear Regression

Model Architecture:

$y = wx + b$

Model:

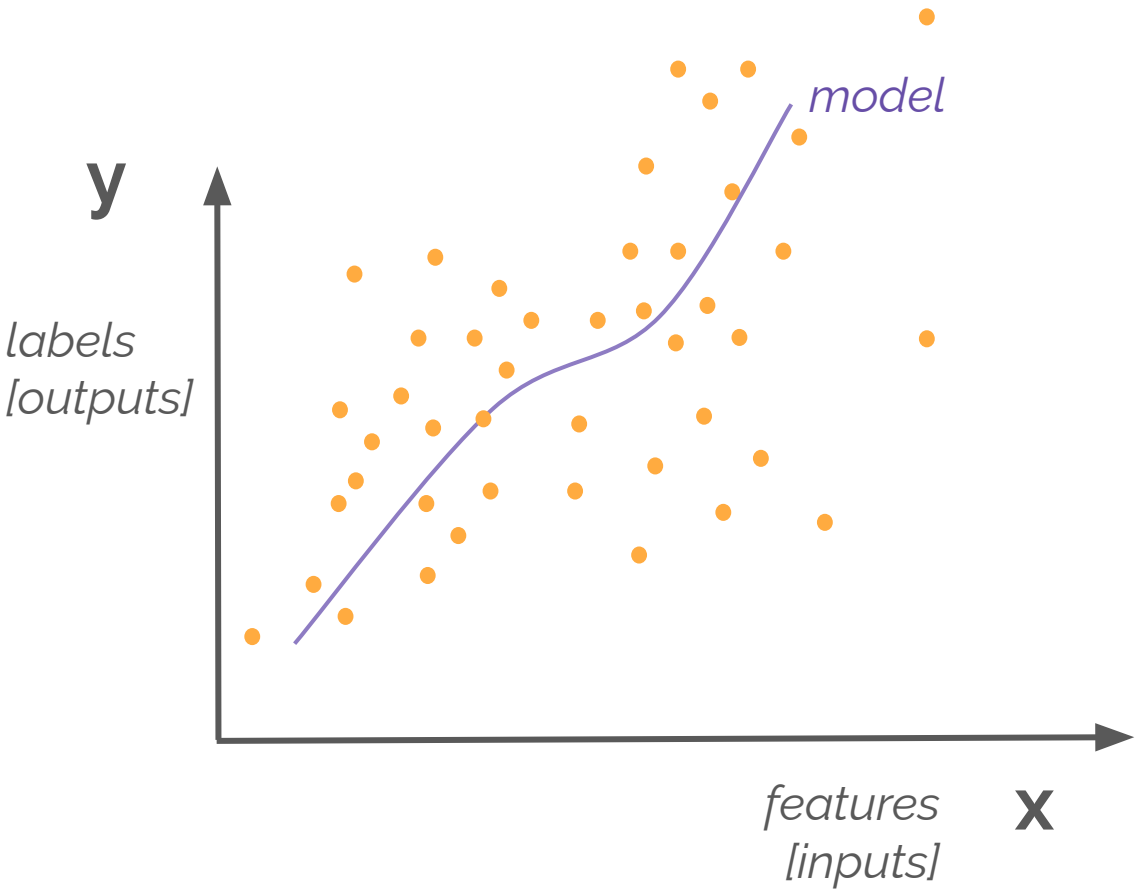
(w, b)

w = weight ; b = bias

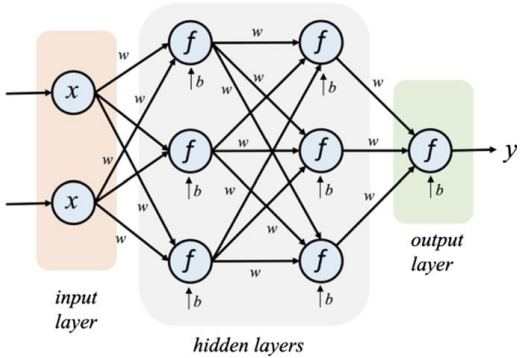
Model optimization: Minimize the **cost function**

- L_2 norm: $||e||_2 = \left(\sum_{i=1}^{N} e_i^2\right)^{1/2}$

Neural Networks are also trained to minimize a cost function.



Model Architecture:

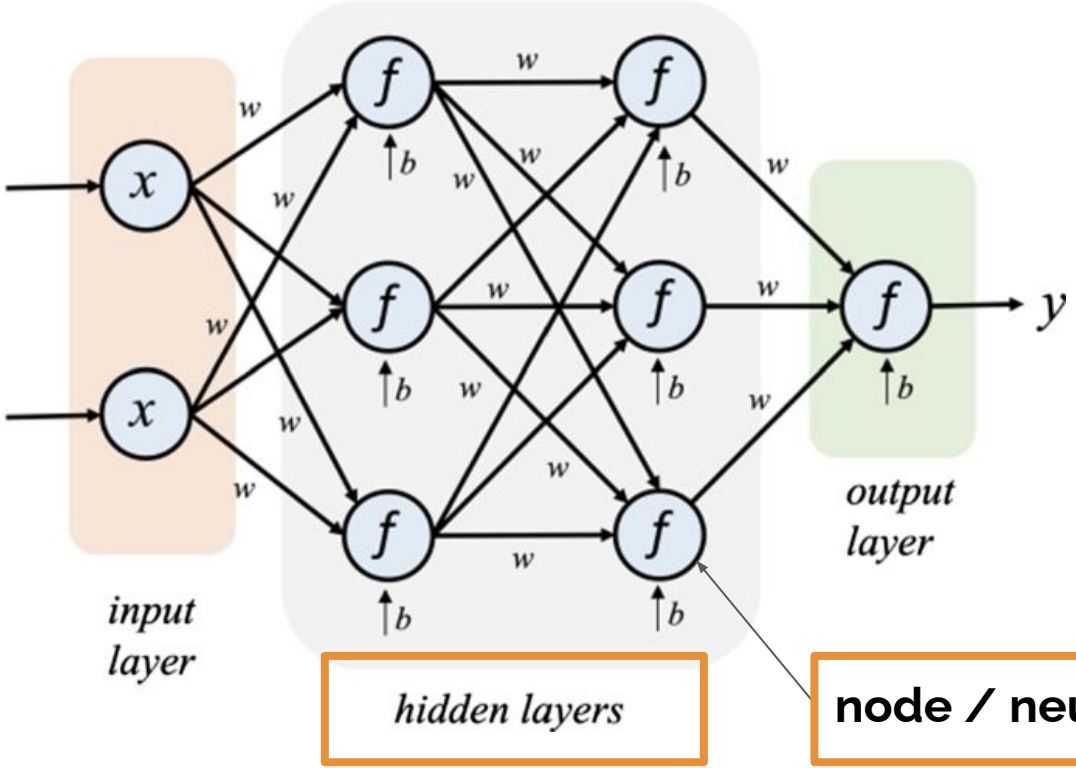


Model:

$$x \rightarrow \boxed{g} \rightarrow \hat{y}$$

Model optimization: Minimize the **cost function** through the *iterative* process of *gradient descent*.

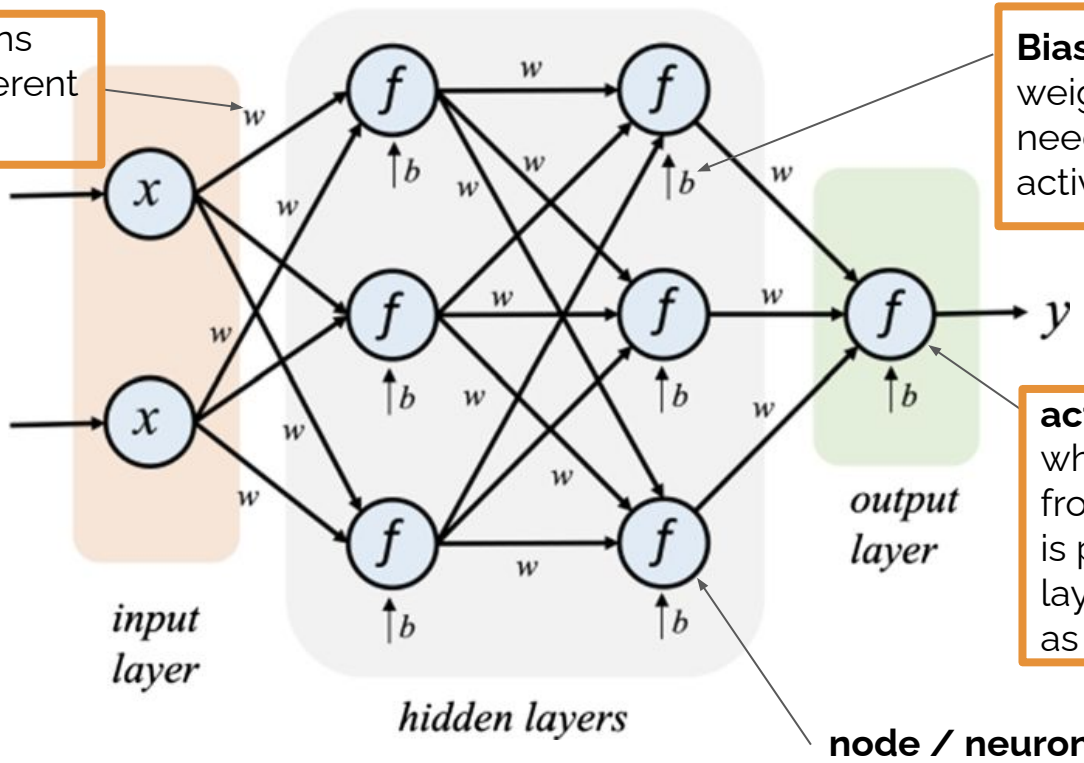
Today we will focus on neural networks (NNs).
This is an example of the architecture of a NN, which consists of several *nodes* embedded within multiple *hidden layers*.



Information is passed through a NN based on the *weights*, *biases*, and *activations*. These are the NN *parameters*.

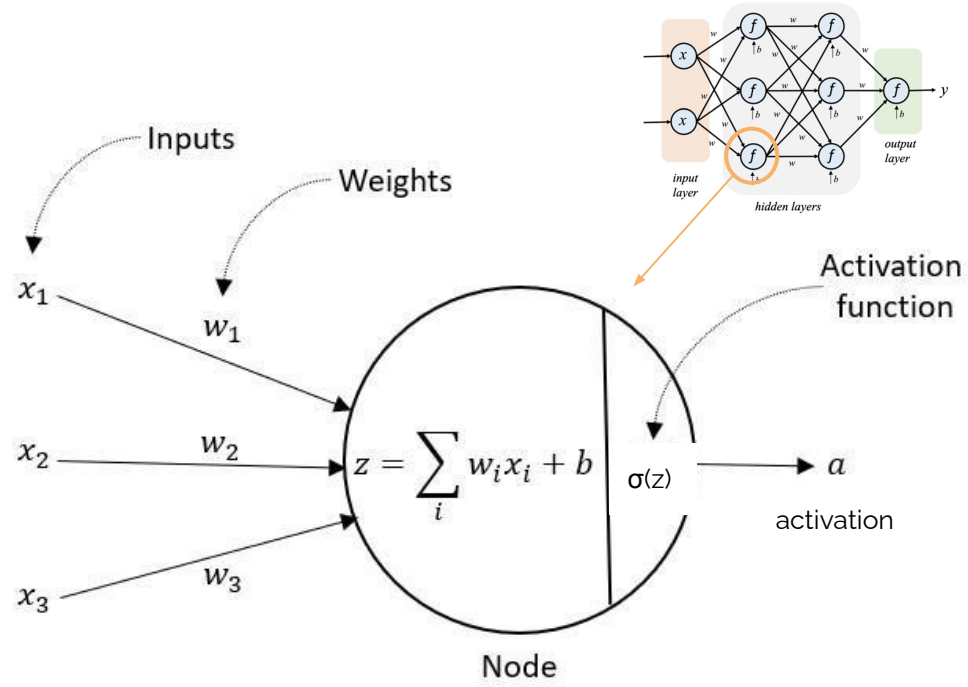
weights, w: connections between nodes of different layers

Biases, b: how high the weighted sum of $x_i w_i$ needs to be for activation of a neuron



activations, f: determine whether the information from the particular node is passed on to the next layer; sometimes labeled as ***a*** instead of ***f***

What happens within each node?



$$a = \sigma(\Sigma w_i x_i + b)$$

Parameters:

- **Inputs, x_i**
- **Weights, w_i**: connections between neurons
- **Biases, b:** how high the weighted sum of $x_i w_i$ needs to be for activation of a neuron
- **Activations, a**: number inside each neuron; determines whether or not neuron is "lit up"

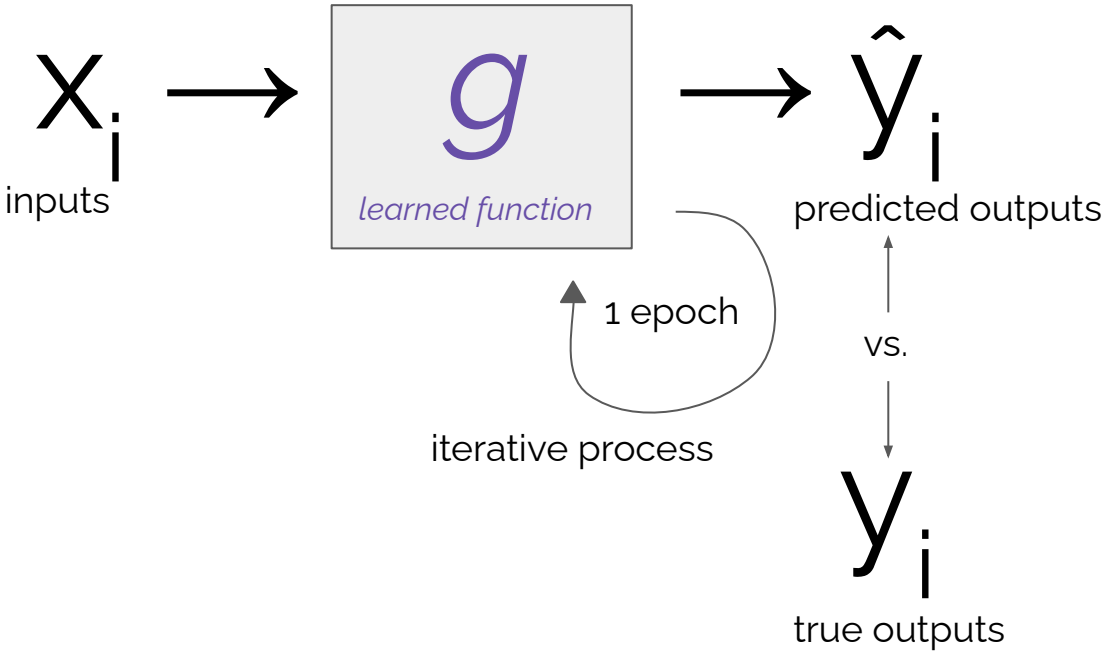
Hyperparameters:

- **Activation function, σ** (sigmoid, ReLU, tanh, etc.): determines if information from specific node is propagated forward in the NN
- **Number of neurons / nodes**
- **Number of layers**
- **Number of epochs / iterations**

Activation of each neuron = activation function (sum (weights * inputs) + biases)

Neural Networks are trained through an *iterative process** that works to minimizing the cost function.

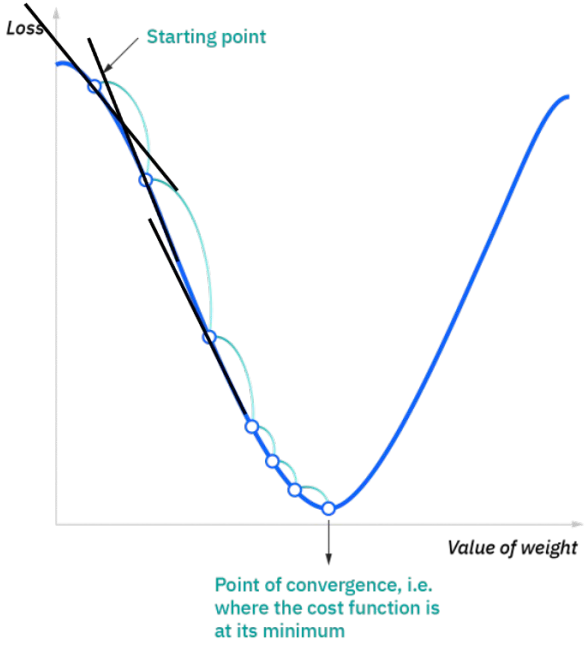
**iterative process = gradient descent*



$$J = f(y_i - \hat{y}_i)$$

cost function

Gradient descent is a technique to find the weight that minimizes the cost function.



An example cost function (MSE)

$$J = \frac{1}{N}\sum_{i=1}^{N}(\tilde{y}_i - y_i)^2$$

This is done by starting with a random point,

The gradient (the black lines) is calculated at that point.

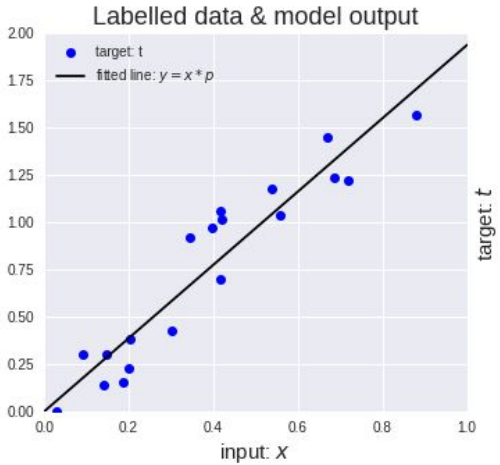
The negative of that gradient is followed to the next point and so on.

This is repeated until the minimum is reached.

Gradient descent is a technique to find the weight that minimizes the cost function.

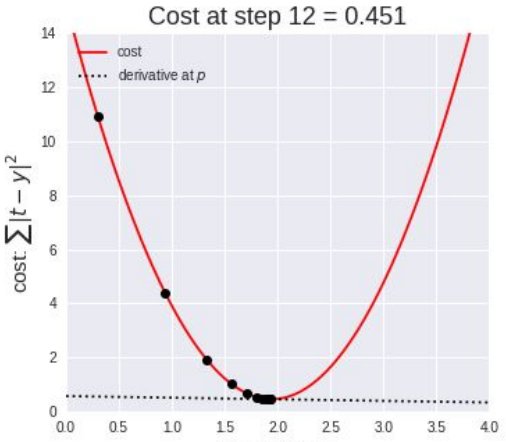
A linear model:

$y = wx + b$



The cost function:

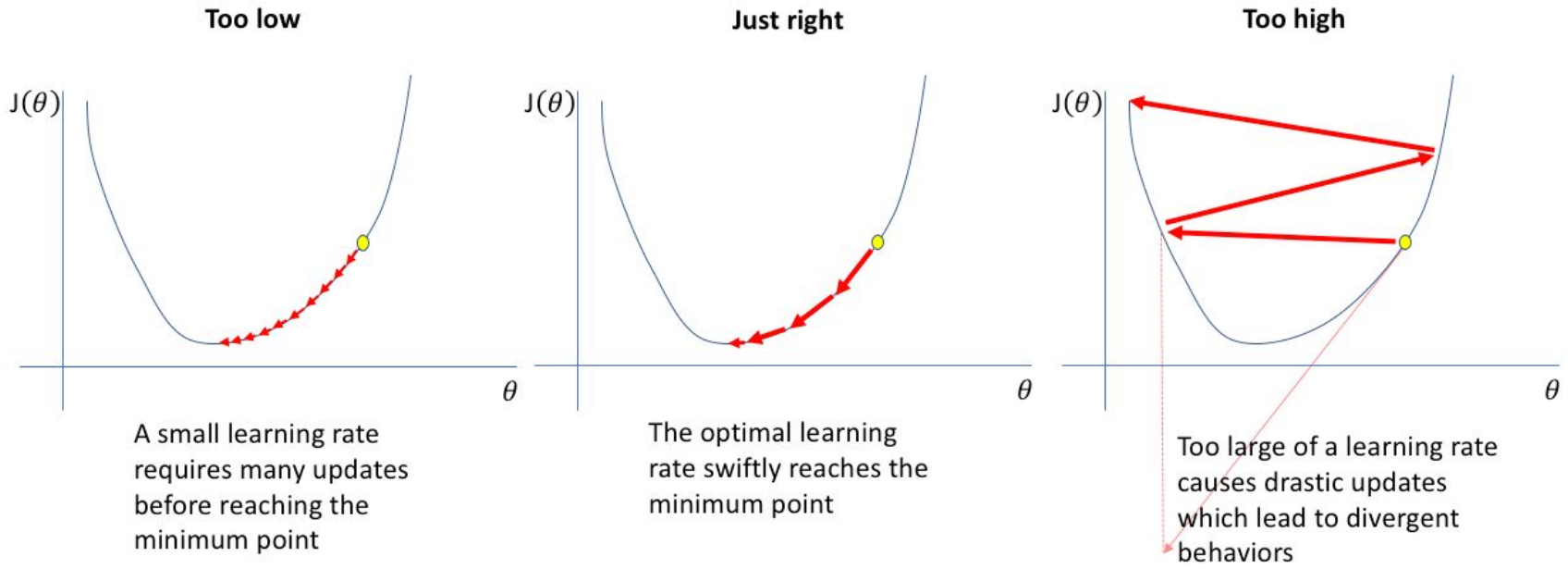
$J = f(y_i - \hat{y}_i)$



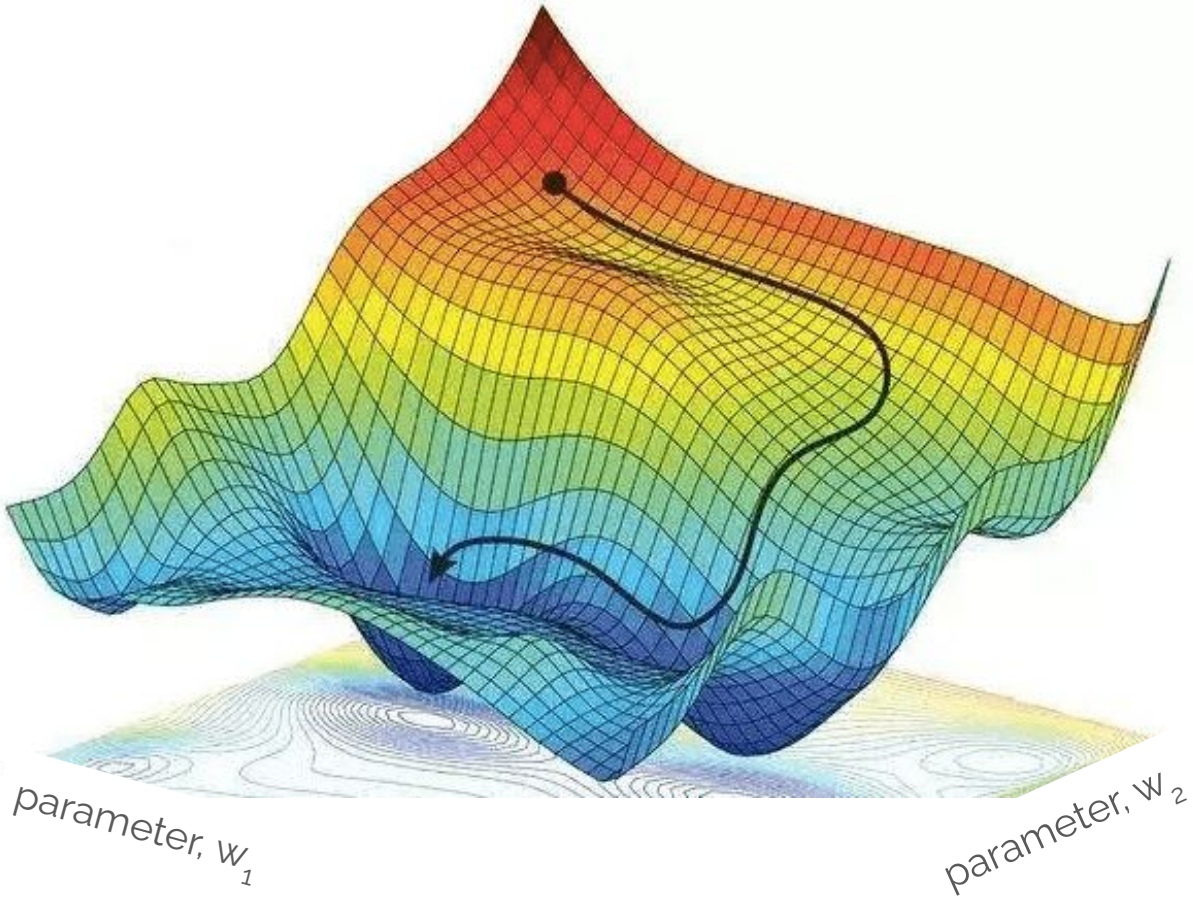
The cost function is updated iteratively based on the ***gradient*** and the ***step size.***

The step size is also called the ***learning rate.***

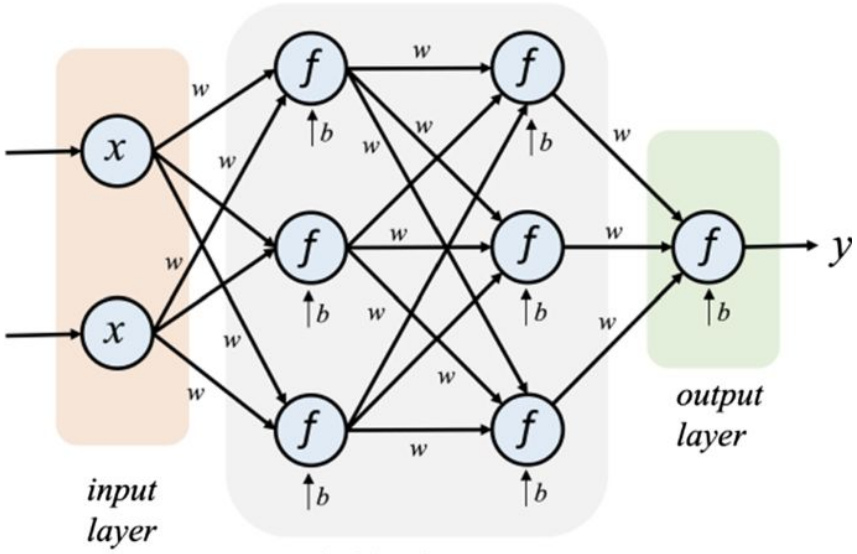
We choose a *learning rate (α)* that allows the model to optimally reach the minimum of the cost function.



Neural networks have multiple weights.



The learned function, g, is representative of the *learned parameters* within the layers of the model (i.e. weights, biases, activations).



x_i inputs → *g* learned function → $\hat{y}_i$ predicted outputs

1 epoch

vs.

y_i true outputs

$J = f(y_i - \hat{y}_i)$ cost function

The take-home messages...

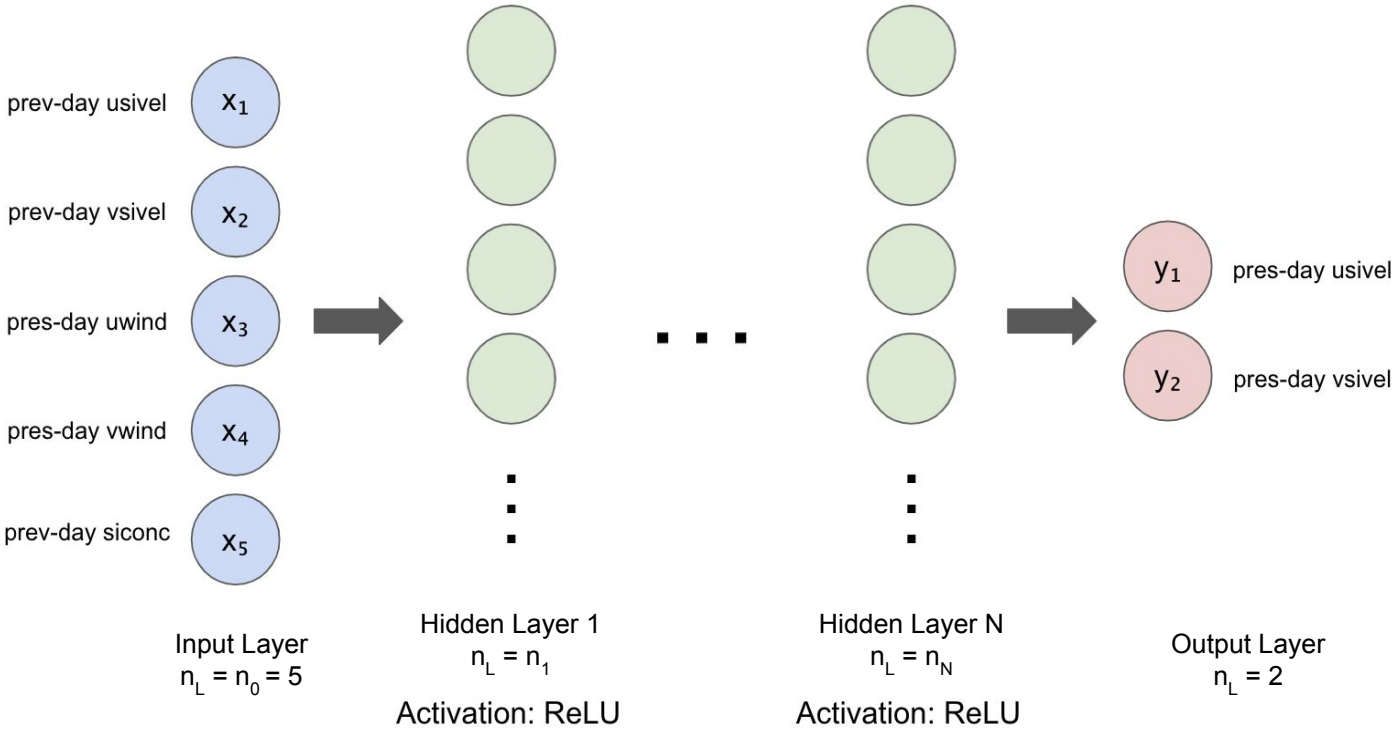
ML models learn statistical relationships in the data, whereas numerical models are based on prescribed physical equations.

There are different ML types (supervised, unsupervised, reinforcement), problems (regression, classification), and algorithms (linear regression, neural networks).

ML models are trained to learn the weights and biases by minimizing a loss function.

Exercise

General NN architecture



Building a Machine Learning Model

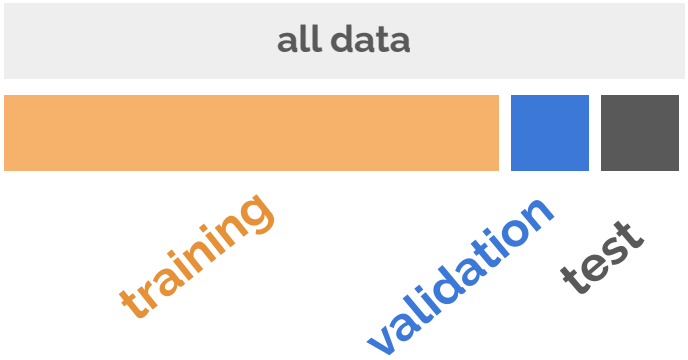
1. Set up environment
2. Data Preparation
    a. Separate into ‘Train-Validation-Test’
    b. Standardize
3. Build the Model
    a. Define parameters and hyperparameters
        i. Loss function
        ii. Number of nodes & layers
        iii. Learning rate
        iv. Activation function
    b. Build the model
    c. Compile and train the model
    d. Plot the model predictions

Building a Machine Learning Model

1. Set up environment
2. **Data Preparation**
    a. **Separate into 'Train-Validation-Test'**
    b. **Standardize**
3. Build the Model
    a. Define parameters and hyperparameters
        i. Loss function
        ii. Number of nodes & layers
        iii. Learning rate
        iv. Activation function
    b. Build the model
    c. Compile and train the model
    d. Plot the model predictions

Before training your model it is important to apply *data splitting* and *feature scaling* to your dataset.

Data Splitting



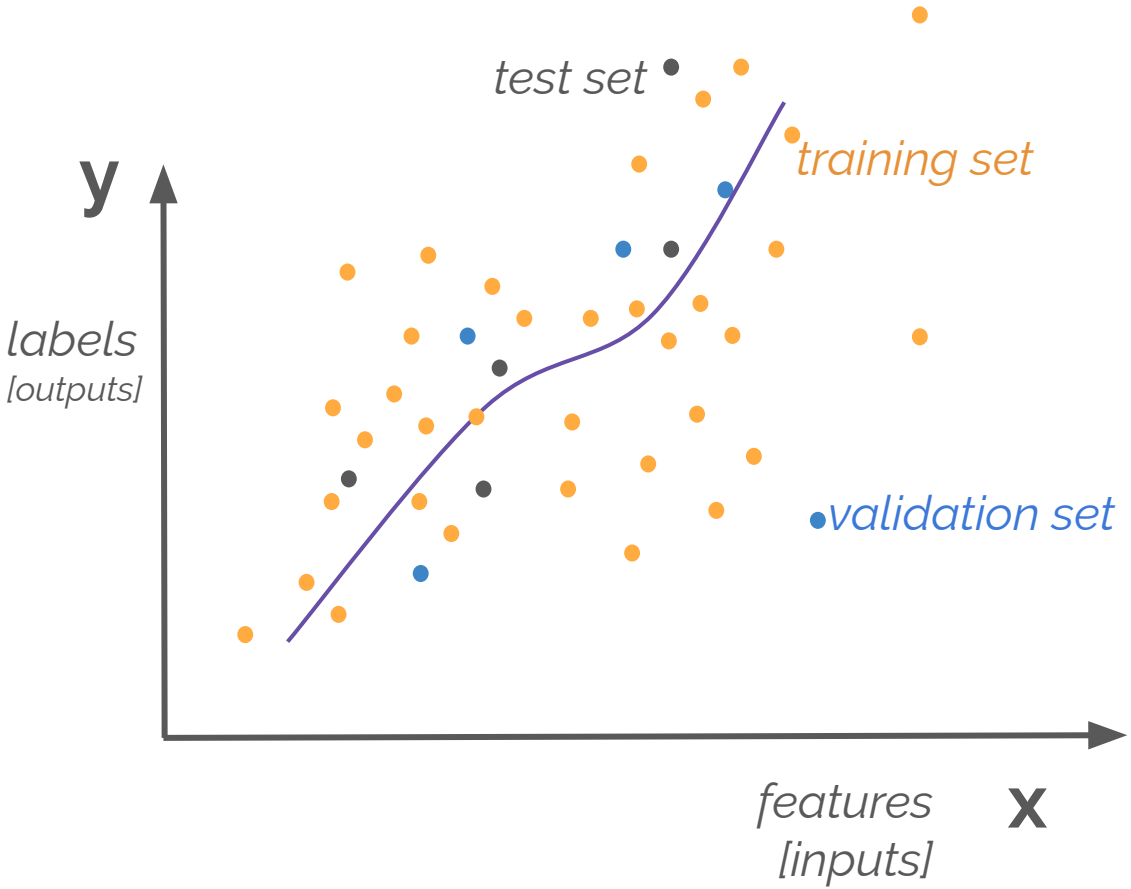
Feature Scaling makes it so inputs are on similar scales.

Apply either:

- **Standardization:** zero mean, one standard deviation
  $data_S = (data-\mu)/\sigma$
- **Normalization:** from 0 to 1
  $data_N$ = [data - max] / [min - max]

Preparing the data.

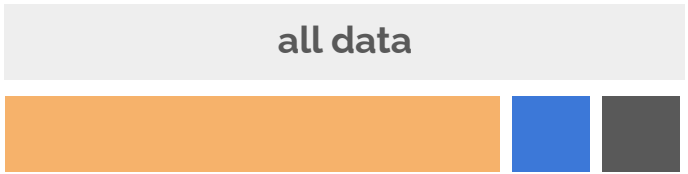
Data are split into training, validation and testing sets.



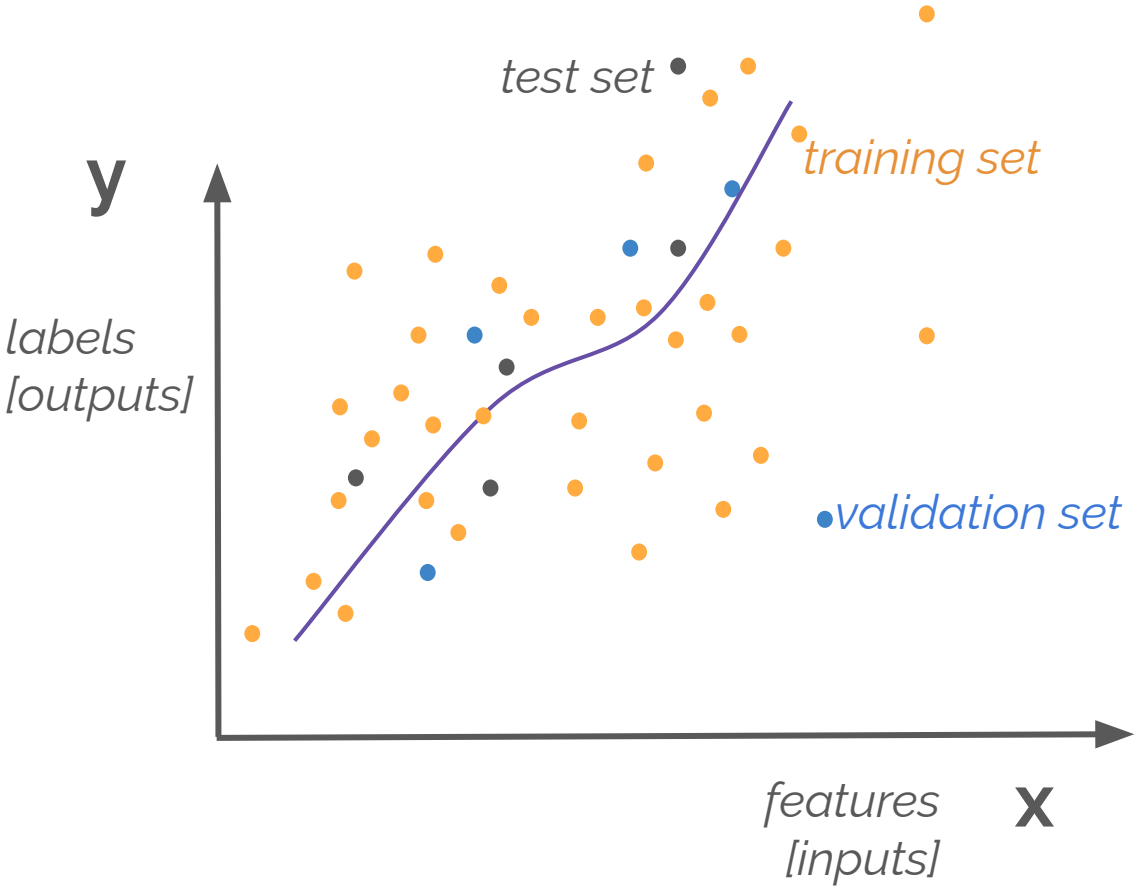
training: Data used to fit the model. The subset of data that the model uses to *learn* (i.e. optimize the cost function)

validation: Data subset used to optimize model *hyperparameters* during training.

test: Data subset used to evaluate the model on data it has not seen before.



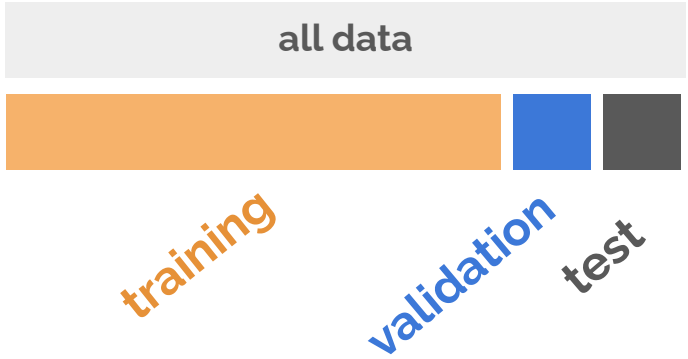
Data are split into training, validation and testing sets.



Data Splitting

Why do we split data?

- Reduce overfitting
- Optimize *hyperparameters*



Building a Machine Learning Model

1. Set up environment
2. Data Preparation
    a. Separate into 'Train-Validation-Test'
    b. Standardize
3. **Build the Model**
    a. **Define parameters and hyperparameters**
        i. **Loss function**
        ii. **Number of nodes & layers**
        iii. **Learning rate**
        iv. **Activation function**
    b. Build the model
    c. Compile and train the model
    d. Plot the model predictions

The *cost function* evaluates the difference between the model prediction and the true data during training.

$$\text{error} = \frac{1}{N}\Sigma (y_i - \hat{y}_i)^2$$

true outputs — predicted outputs

- mean squared error (MSE)
- root MSE (RMSE)
- normalized RMSE (NRMSE)
- mean absolute error (MAE)

The cost function is applied and evaluated during ***training*** and ***validation***.

Other metrics can be analyzed as well, but the model will not use them to train.

Metrics applied to the ***testing*** data are not used in training, but merely to investigate the performance of the model on unseen data.

Choices you can make...

Model Inputs & Outputs:

- Under "**Section 1.3.3 Create Input & Output Data**" you can choose different inputs and outputs depending on the question you want to ask.

Hyperparameters:

- Under **"Section 2.2 Define Model Hyperparameters"** you can make changes to the following hyperparameters:
  - **Activation function, σ** (sigmoid, ReLU, tanh, etc.): determines if information from specific node is propagated forward in the NN
  - **Number of neurons / nodes**
  - **Number of layers**
  - **Number of epochs / iterations**
  - **Batch size**
  - **Learning rate**
  - **Gradient descent algorithm**
  - **Regularization**

What the NN chooses for you...

Parameters:

- **Weights, w_i**: connections between neurons
- **Biases, b:** how high the weighted sum of $x_i w_i$ needs to be for activation
- **Activations, a**: number inside each neuron; determines whether or not neuron is "lit up"

Tutorial: *predicting sea-ice velocity using a Neural Network.*

Everything required to run this tutorial can be found at:
https://github.com/lahoffman/ml_lectures_LPHYS2268

Open the Google Colaboratory Notebook ('nn_regression.ipynb') and follow the instructions.

If you don't have a google account, use the login info:

Username: lphys2268
Password: coriolis7

Make sure to read and follow the instructions under "0. Set up" before you begin (i.e. make a copy of the notebook that you can edit & get the data in your drive).