

AI4InSync

Optimizing Southern Ocean observing systems using AI.

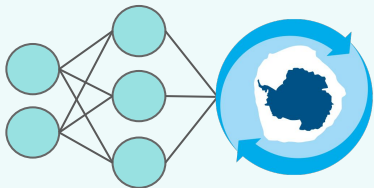
lauren.hoffman@uclouvain.be

Lauren Hoffman, François Massonnet

Earth and Life Institute (ELI), Earth and Climate, Université catholique de Louvain



ANTARCTICA
INSYNC



AI4InSync

A collaborative workshop to support observation system design with machine learning methods.

Workshop Outcomes:

- (a) Identify 1-3 scientific questions
- (b) Identify potential working groups

InSync Webinar // 8-10am // 22 May

Session 1

**Motivation &
Intro to ML**

Session 2

ML Tutorial

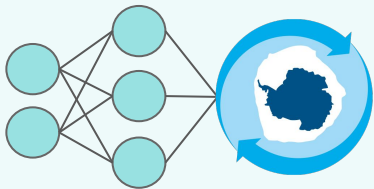
Session 3

**State-of-the-Art:
ML in Climate
Science**

Session 4

Discussion:
Identify science
questions

Define potential
working groups



AI4InSync

A collaborative workshop to support observation system design with machine learning methods.

Workshop Outcomes:

- (a) **Scientific Questions:** Identify 1-3 scientific questions on which we center ML-based observing system design, including key observed variables and the desired spatio-temporal scales.
- (b) **Project Working Groups:** Identify potential working groups to approach these questions, including ML methods best suited for each approach.



ANTARCTICA
INSYNC



The Abdus Salam
International Centre
for Theoretical Physics



6 ICTP
1964-2024

Workshop on the Role of Sea Ice and its Variability in the Climate System



Canadian National Committee for SCOR
Comité national canadien pour SCOR
Scientific Committee on Oceanic Research



29 - 31 July 2024



Trieste, Italy

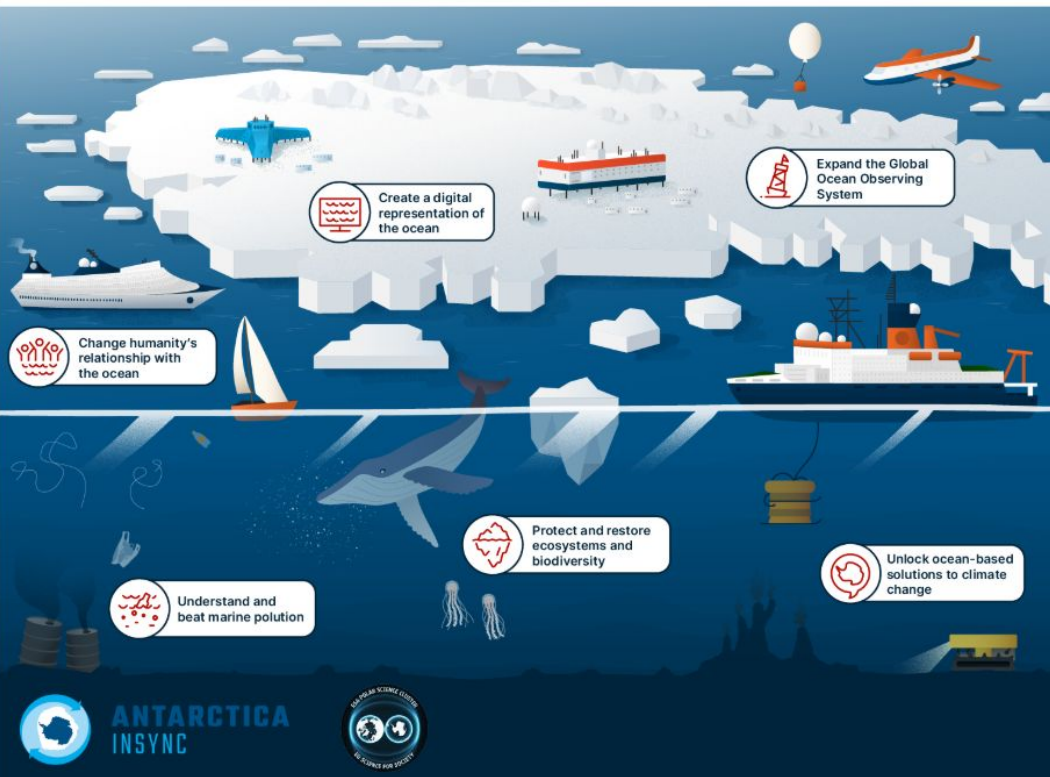


SAVE THE DATE

INTERNATIONAL SCIENCE WEBINAR

19. Mai || 07:00 - 09:00 (UTC)

22. Mai || 15:00 - 17:00 (UTC)



Scan to sign up!

THE OCEAN DECADE

Antarctica InSync

Antarctica International Science &
Infrastructure for Synchronous Observation



unesco

Intergovernmental
Oceanographic
Commission



2021 United Nations Decade
of Ocean Science
2030 for Sustainable Development



Antarctica InSync is an observational campaign to study the dynamics of the connections between ice, ocean, climate, environment and life.

<https://www.antarctica-insync.org/>



ANTARCTICA
INSYNC

Antarctica InSync Themes



Southern Ocean heat, freshwater, and carbon budgets and their response to climate change



Improving knowledge and protection of the unique Antarctic life from land into the deep sea



Rapid sea ice decline and its interdisciplinary consequences



Melting ice shelves and coastal impacts



Anthropogenic signatures in Antarctica: the race against pollution and other pressures

Antarctica InSync timeline

2023

2027

2030

2032

2033

Preparatory phase

Synchronous
scientific observation

Synthesis
and reporting

5th
IPY



ANTARCTICA
INSYNC

Polar regions are notoriously difficult to observe, and limited resources constrain our ability to achieve adequate spatio-temporal coverage.

Moorings

Ocean vessels

Unmanned aerial
or waterborne
vehicles

Submarines



Satellites

Aircraft-borne
instruments

Radiosonde
launches

Drifting buoys

Automatic weather
stations

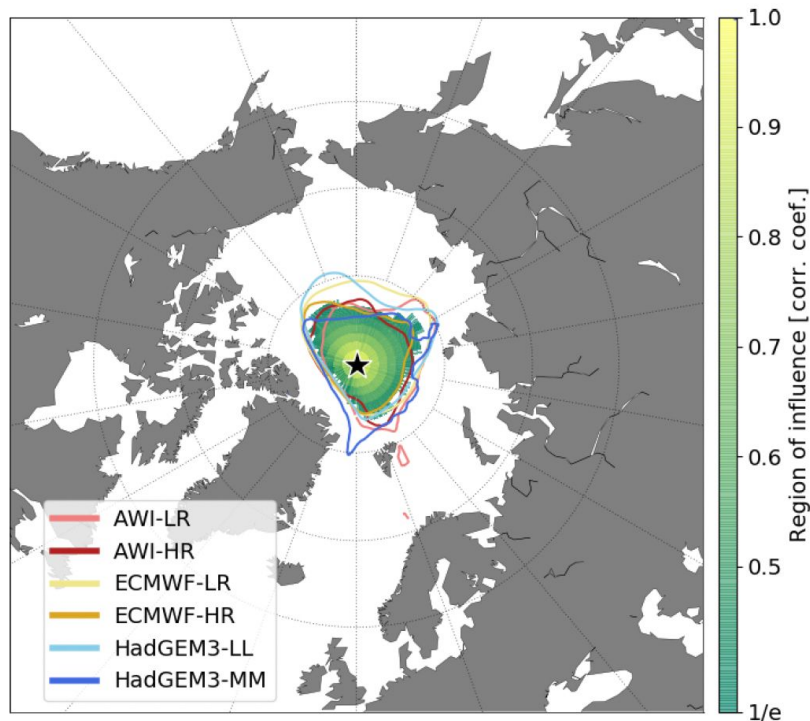
It is not required to monitor all variables at all spaces and times.

A minimal number of
well-chosen stations
targeting *key variables*
could reveal dominant
modes of high-latitude
climate variability in
real world.

It is not required to monitor all variables at all spaces and times.

Ponsoni et al (2019)

A minimal number of *well-chosen stations* targeting *key variables* could reveal dominant modes of high-latitude climate variability in real world.



The *region of influence* is where the correlation between SIT at a select location and all other locations is higher than $1/e$.

How many optimal sites are needed for explaining a substantial amount (e.g. 70%) of original SIV anomaly variance?

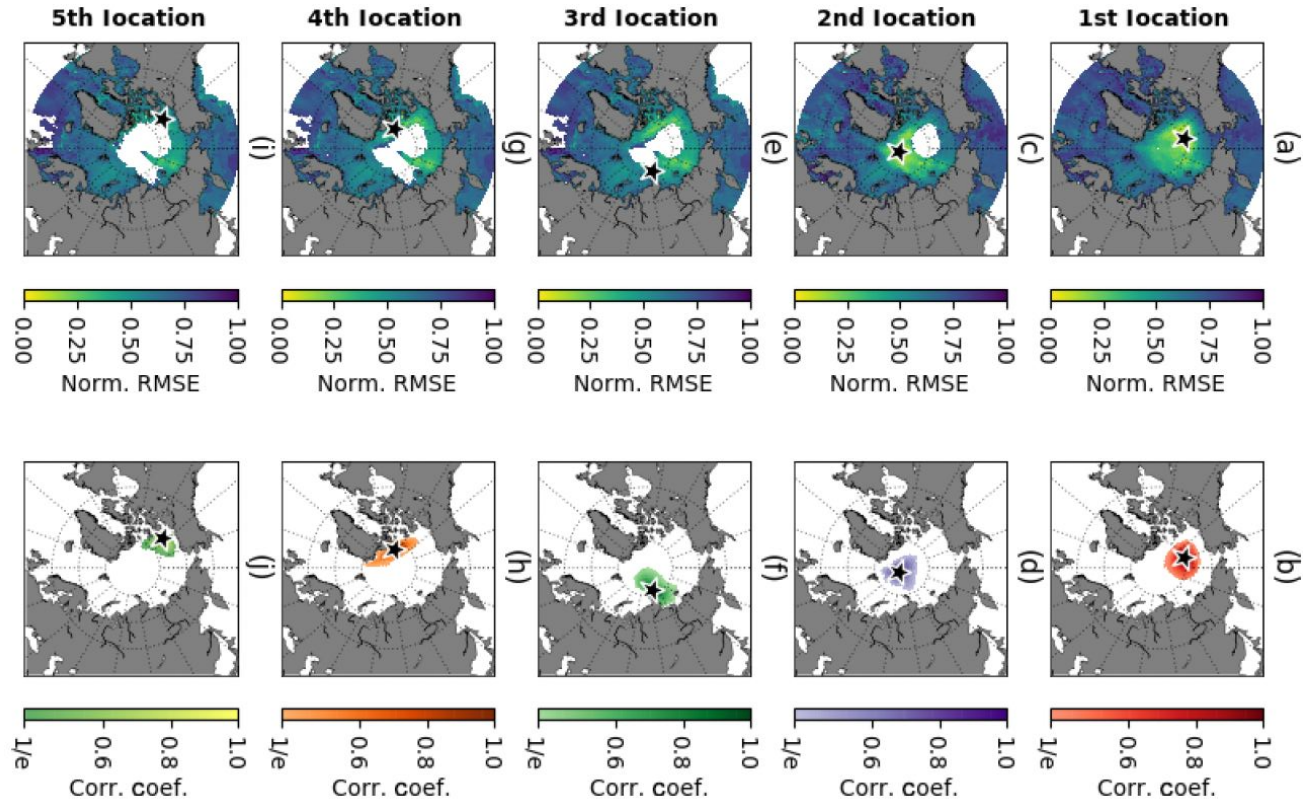
Ponsoni et al (2019)

Variables:

Integrated: pan-Arctic SIV, pan-Arctic SIA, Atlantic basin OHT

Gridded: SIT, SIC, SST, Drift

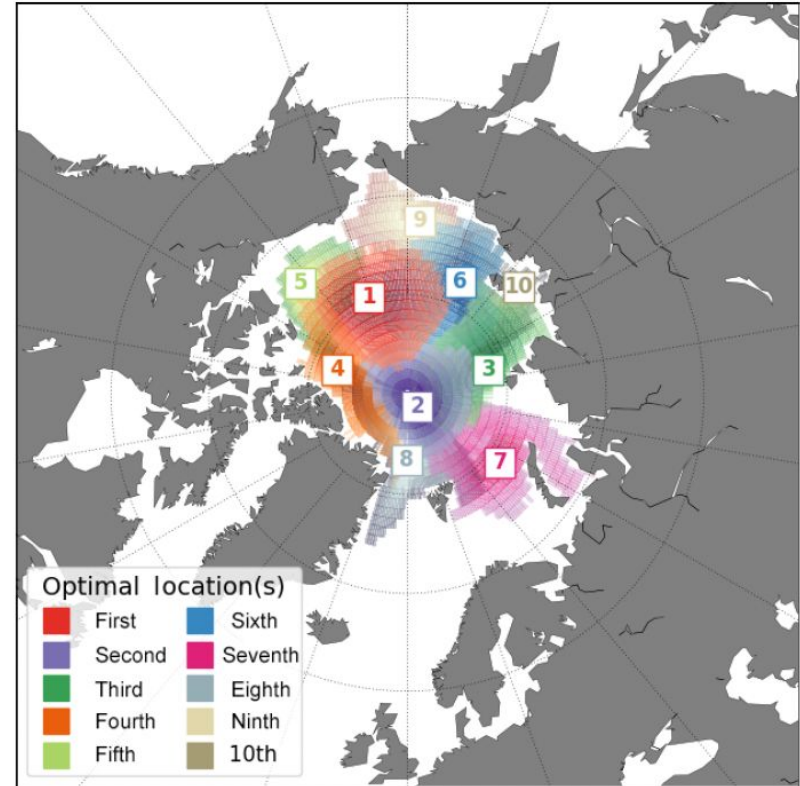
Method: Statistical Empirical Model (multiple linear regression): Iteratively build score maps ($Sc[i,j]$) for selected locations and apply SEM at each location



How many optimal sites are needed for explaining a substantial amount (e.g. 70%) of original SIV anomaly variance?

Ponsoni et al (2019)

- Four sites explain more than 70% of temporal variability in SIV
- No equivalent study in Antarctic
- There is no 'best' observing system, but rather good observing systems that can answer specific questions



Models can be useful tools for the development of polar observational networks.

Climate Models as Guidance for the Design of Observing Systems: the Case of Polar Climate and Sea Ice Prediction

François Massonnet¹ 

1. Observing System Simulation Experiments (OSSE) and Quantitative Network Design (QND)
2. Emulation of satellites and retrieval algorithms
3. Constraining long-term projections: identify observational gaps that, if filled, would allow uncertainty reduction in projections
4. Evaluation of observational products
5. Strategic placement of in-situ sampling sites

There are limitations in using models for observing system design.

Climate Model Limitation
Real world obs exhibit broader frequency variability than climate models
Non-stationarity
Optimal number of stations depends on model resolution
No guarantee that models display the correct modes of variability

There are limitations in using models for observing system design.		
Climate Model Limitation	ML Benefits	ML Limitations
Real world obs exhibit broader frequency variability than climate models	ML models can be trained on observations and can learn short and long-term dependencies	<i>Training data must include enough variability for ML to learn from</i>
Non-stationarity	Can adapt through re-training or regime-based learning.	<i>Poor extrapolation on unseen futures</i>
Optimal number of stations depends on model resolution	ML handles high-resolution data quite well, and is often used for downscaling.	<i>May require scale-specific training</i>
No guarantee that models display the correct modes of variability	ML can learn real-world teleconnections from observations	<i>Limited by training data spatio-temporal coverage (i.e. ML needs a lot of data)</i>

Machine learning
models can be
useful tools for
the development
of polar
observational
networks.

~~Climate Models~~ as Guidance for the Design of Observing Systems:
the Case of Polar Climate and Sea Ice Prediction

François Massonnet¹ , Lauren Hoffman, Your Name Here

1. Observing System Simulation Experiments (OSSE) and Quantitative Network Design (QND)
2. Emulation of satellites and retrieval algorithms
3. Constraining long-term projections: identify observational gaps that, if filled, would allow uncertainty reduction in projections
4. Evaluation of observational products
5. Strategic placement of in-situ sampling sites

What are your needs for an observing system?

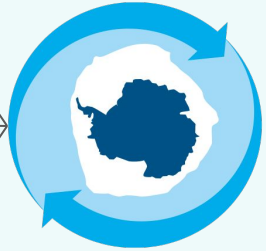
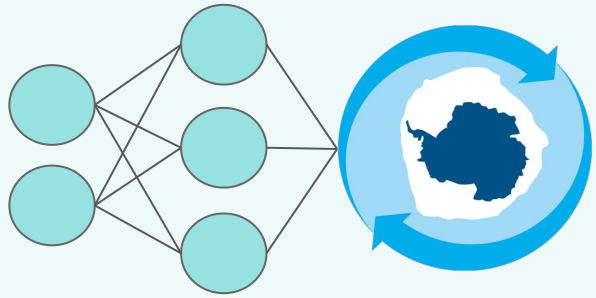
- Variables?
- Spatio-temporal scales?

How do you define the value of an observing system?

What objective function would you optimize?

- Reducing mean state error?
- Constraining long-term trends?
- Improve forecast skills?
- Capture extreme events?
- Uncertainty reduction?

How could optimizing for one metric harm another?



AI4InSync

What is machine learning?

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).

ARTIFICIAL INTELLIGENCE

A program that can sense, reason, act, and adapt.

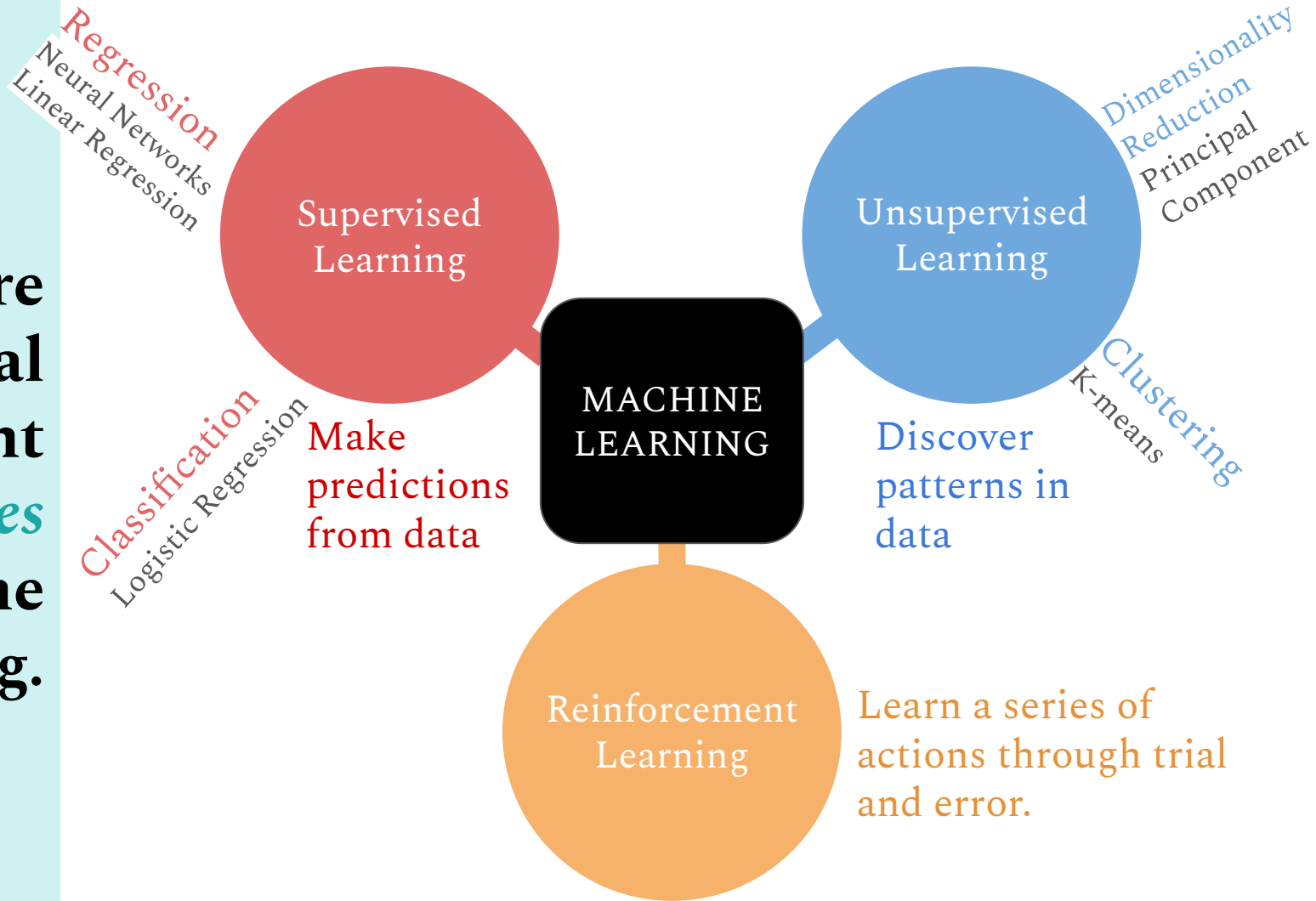
MACHINE LEARNING

Algorithms that learn statistical interactions within data.

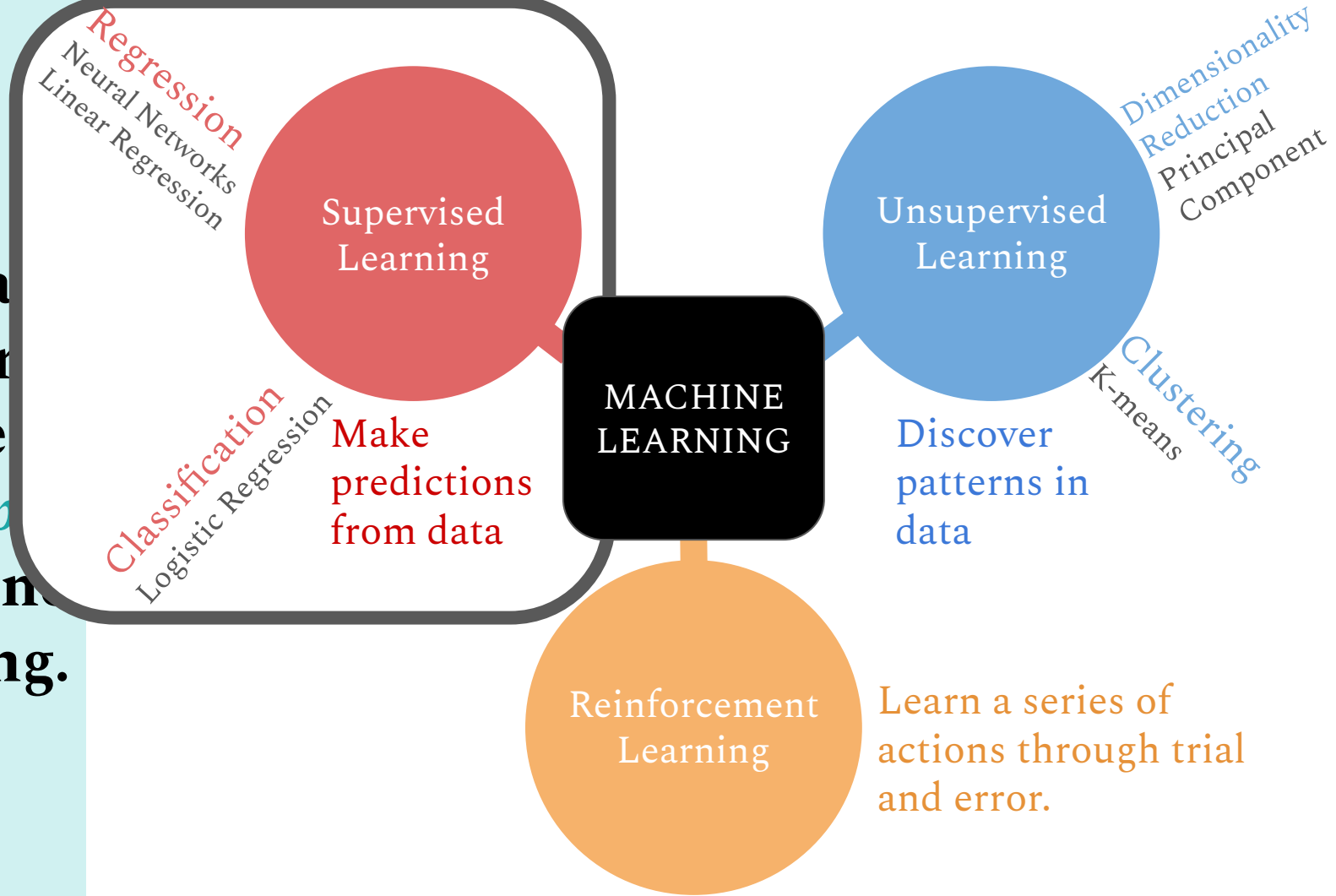
DEEP LEARNING

Subset of ML in which multilayered neural networks learn from vast amounts of data.

There are several different *types* machine learning.

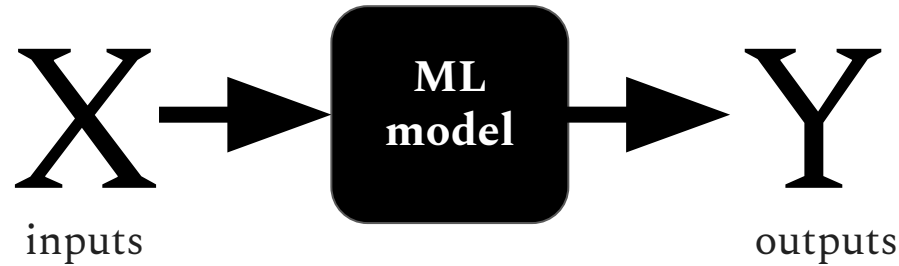


There are several different types of machine learning.



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.**

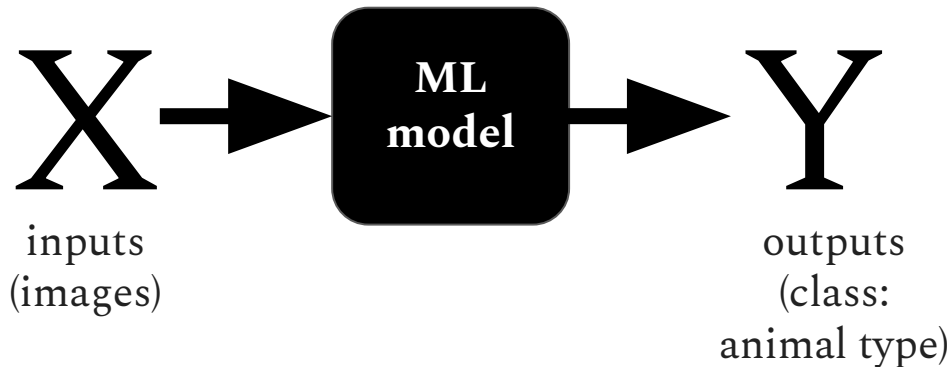


Cat [97%]

Fish [0%]

Deer [3%]

Butterfly [0%]



Supervised
Learning

Classification

**Classification
models
predict
probabilities
of classes.**

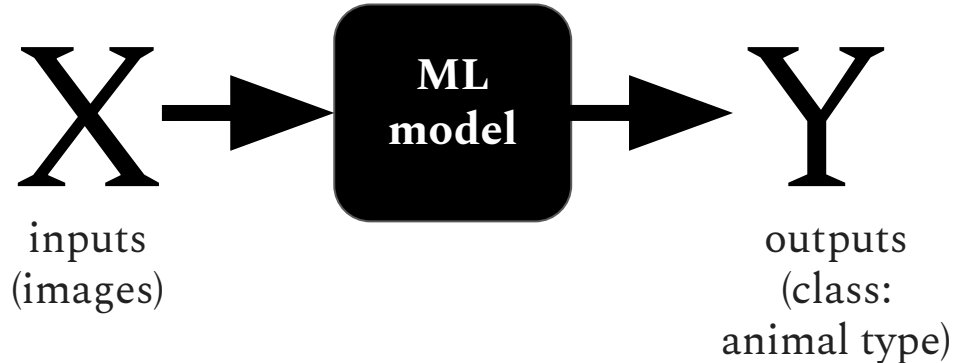


Cat [40%]

Fish [35%]

Deer [10%]

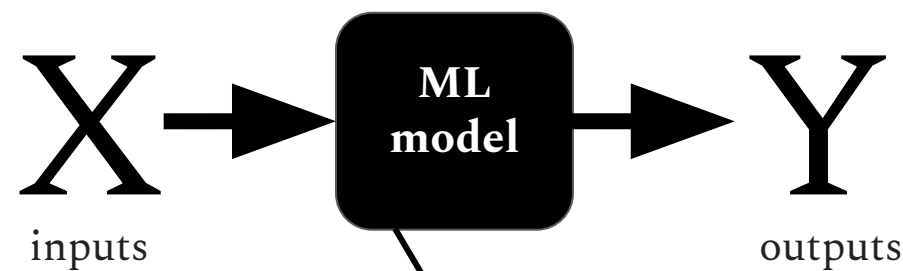
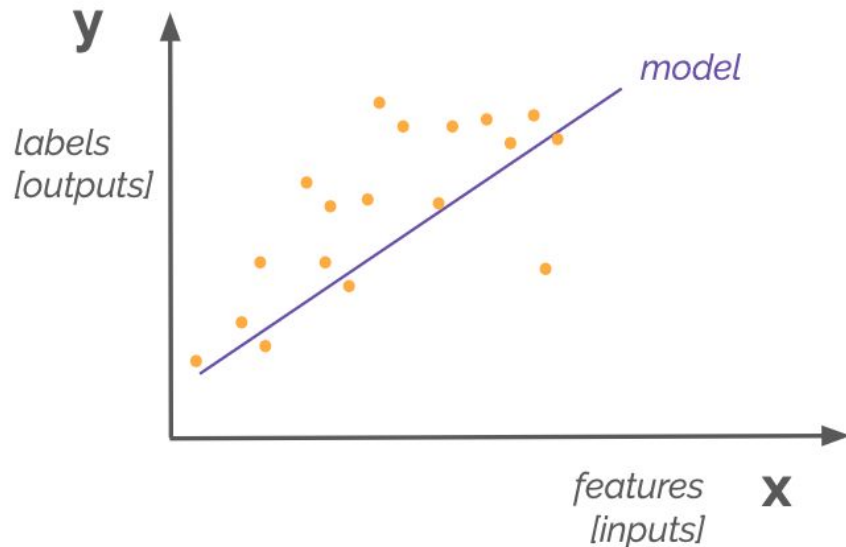
Butterfly [15%]



Supervised
Learning

Classification

Regression models predict real, continuous values.



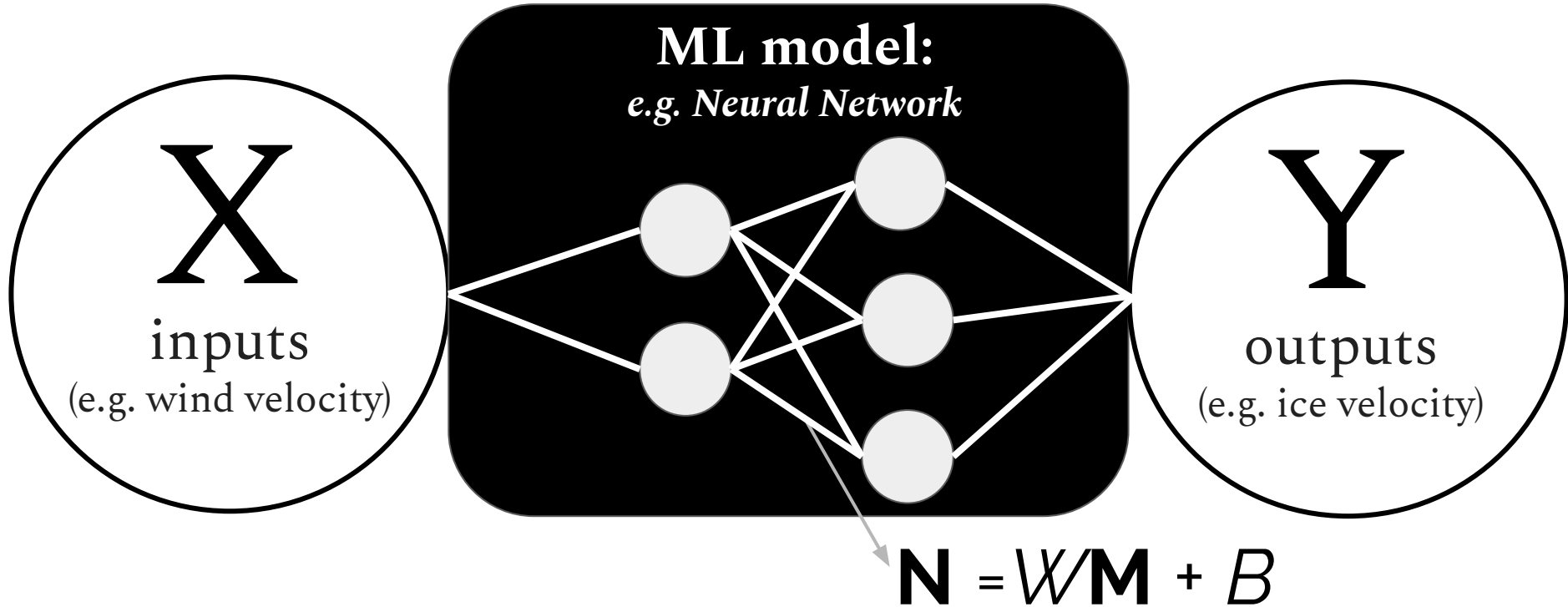
e.g. Linear Regression:

$$Y = WX + B$$

Supervised Learning

Regression:
Linear Regression

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



There are many different types of neural networks.

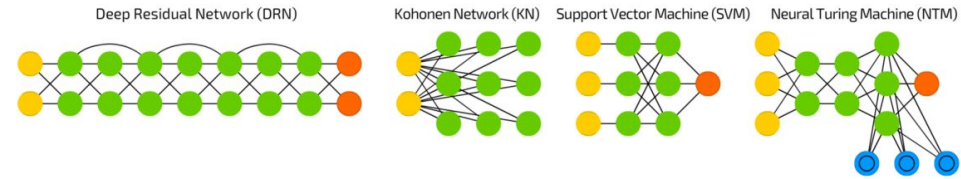
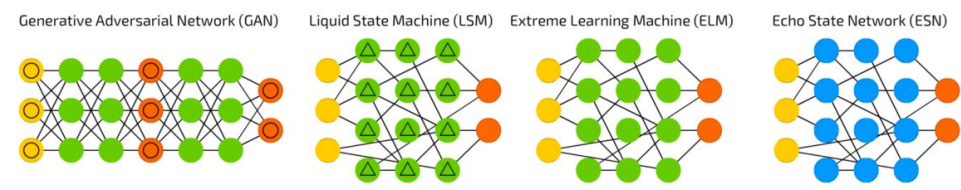
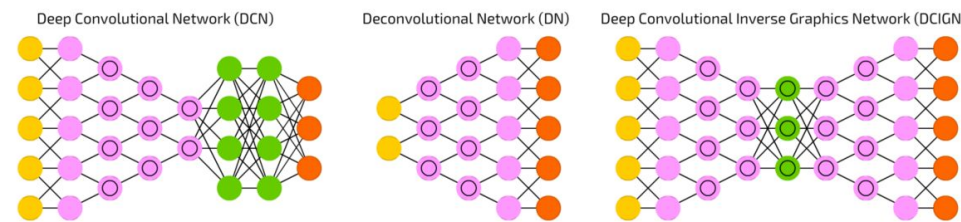
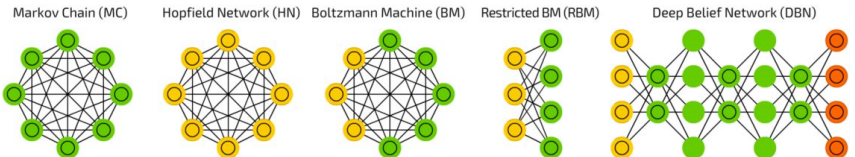
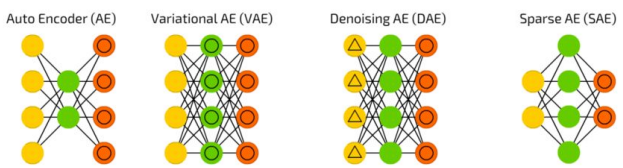
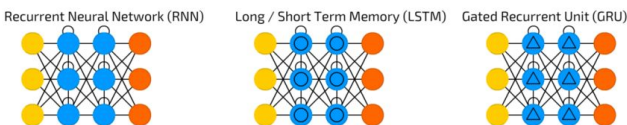
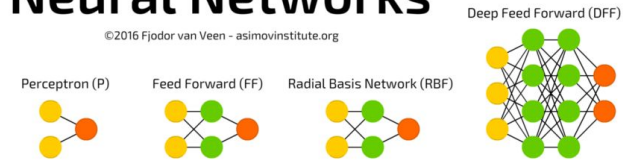
Particular NNs can be beneficial depending on your application.

A mostly complete chart of

Neural Networks

©2016 Fjodor van Veen - asimovinstitute.org

- Backfed Input Cell
- Input Cell
- Noisy Input Cell
- Hidden Cell
- Probabilistic Hidden Cell
- Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- Different Memory Cell
- Kernel
- Convolution or Pool



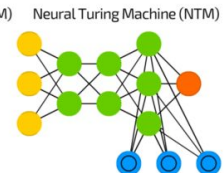
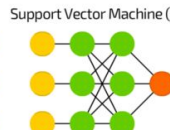
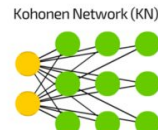
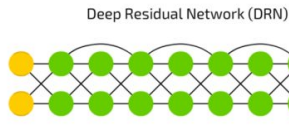
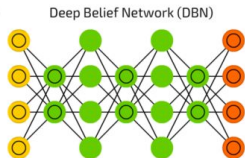
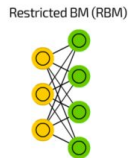
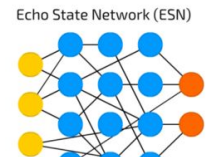
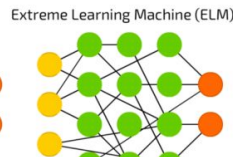
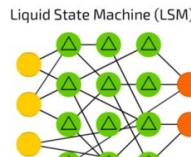
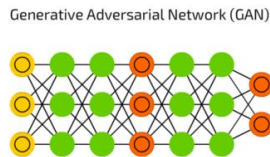
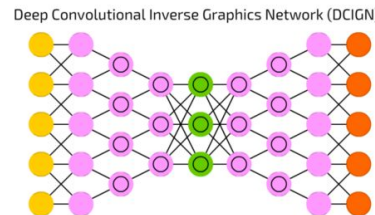
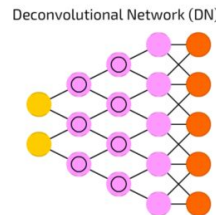
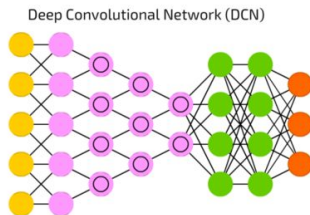
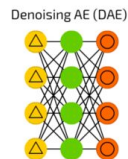
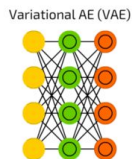
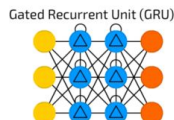
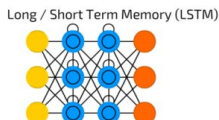
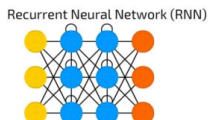
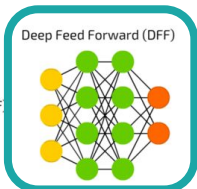
Today we will play with a tutorial that uses a *deep neural network* (here called DFF) with several hidden layers.

A mostly complete chart of

Neural Networks

©2016 Fjodor van Veen - asimovinstitute.org

- Backfed Input Cell
- Input Cell
- Noisy Input Cell
- Hidden Cell
- Probabilistic Hidden Cell
- Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- Different Memory Cell
- Kernel
- Convolution or Pool



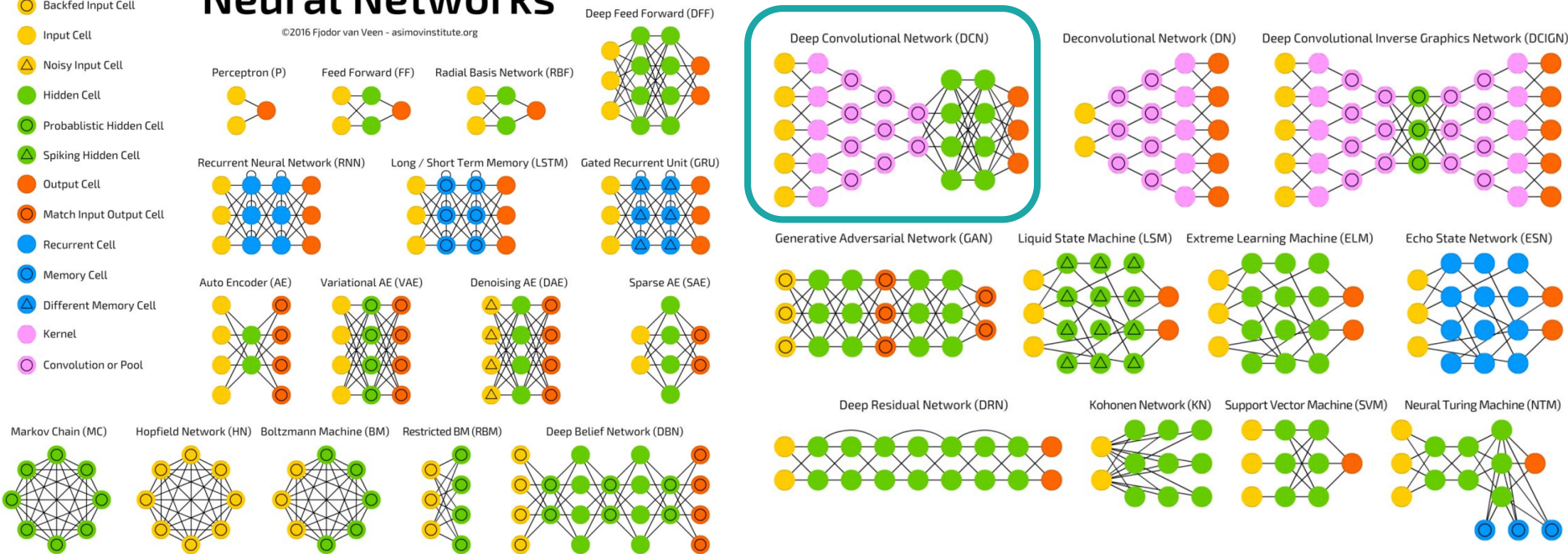
A *Convolutional Neural Network (CNN)* (here labeled DCN) is particularly useful for images or image-like datasets (i.e. mapped products).

A mostly complete chart of

Neural Networks

©2016 Fjodor van Veen - asimovinstitute.org

- Backfed Input Cell
- Input Cell
- △ Noisy Input Cell
- Hidden Cell
- Probabilistic Hidden Cell
- △ Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- △ Different Memory Cell
- Kernel
- Convolution or Pool

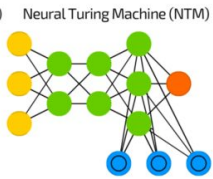
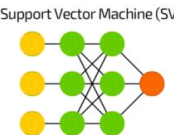
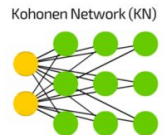
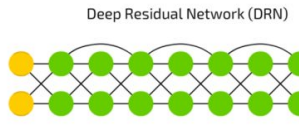
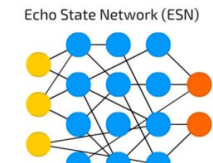
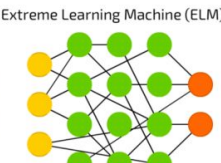
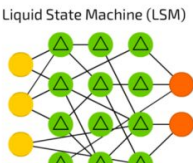
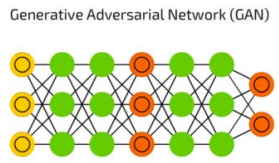
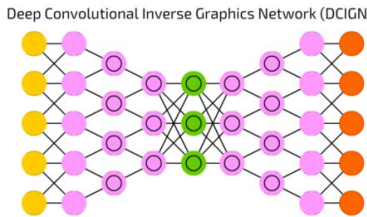
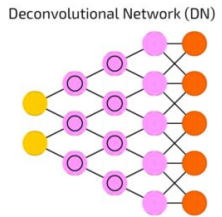
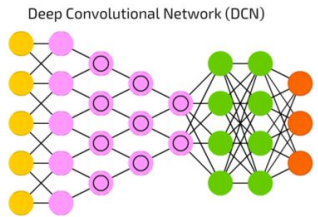
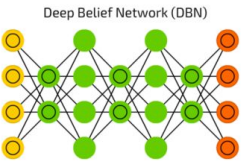
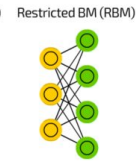
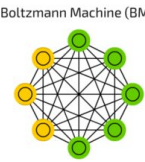
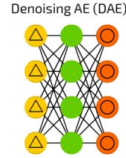
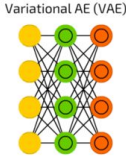
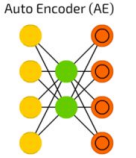
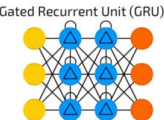
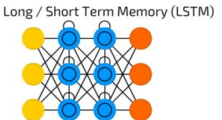
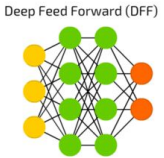
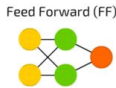
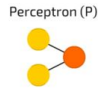
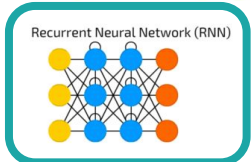


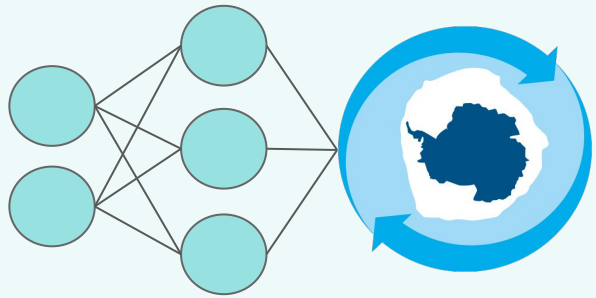
A Recurrent Neural Network (RNN) is particularly useful for time series.

A mostly complete chart of Neural Networks

©2016 Fjodor van Veen - asimovinstitute.org

-  Backfed Input Cell
-  Input Cell
-  Noisy Input Cell
-  Hidden Cell
-  Probabilistic Hidden Cell
-  Spiking Hidden Cell
-  Output Cell
-  Match Input Output Cell
-  Recurrent Cell
-  Memory Cell
-  Different Memory Cell
-  Kernel
-  Convolution or Pool

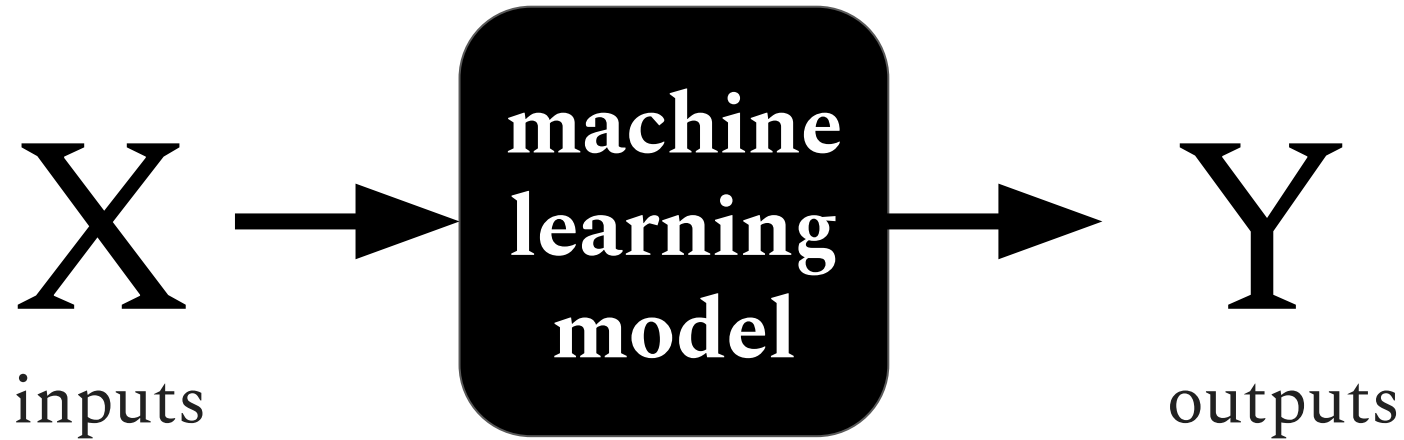




AI4InSync

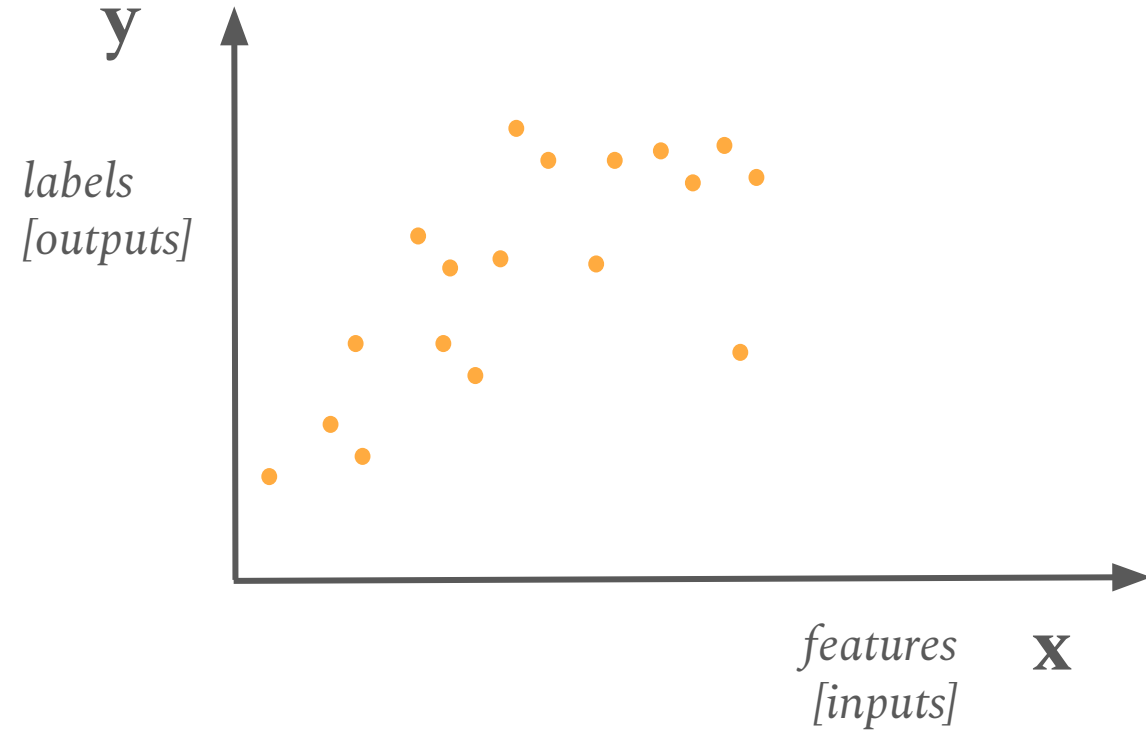
What is *the* machine learning?

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

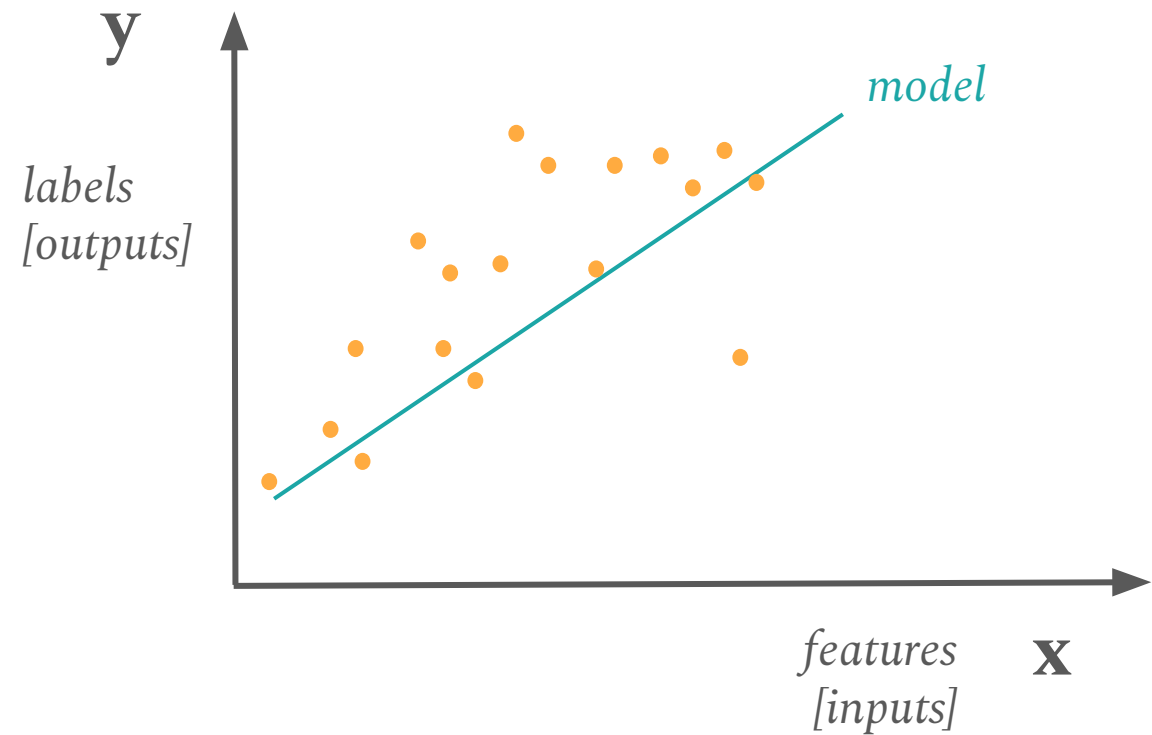


What is in the black box?

How do you predict y from x ?

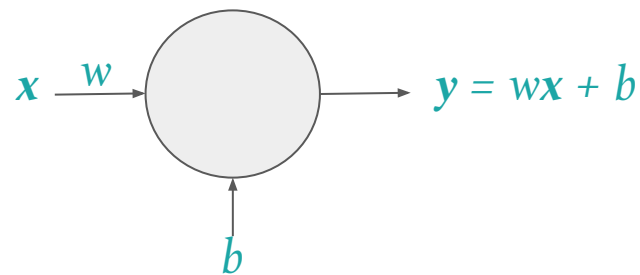


Linear regression is an example of least squares regression.



Linear Regression

Model Architecture:

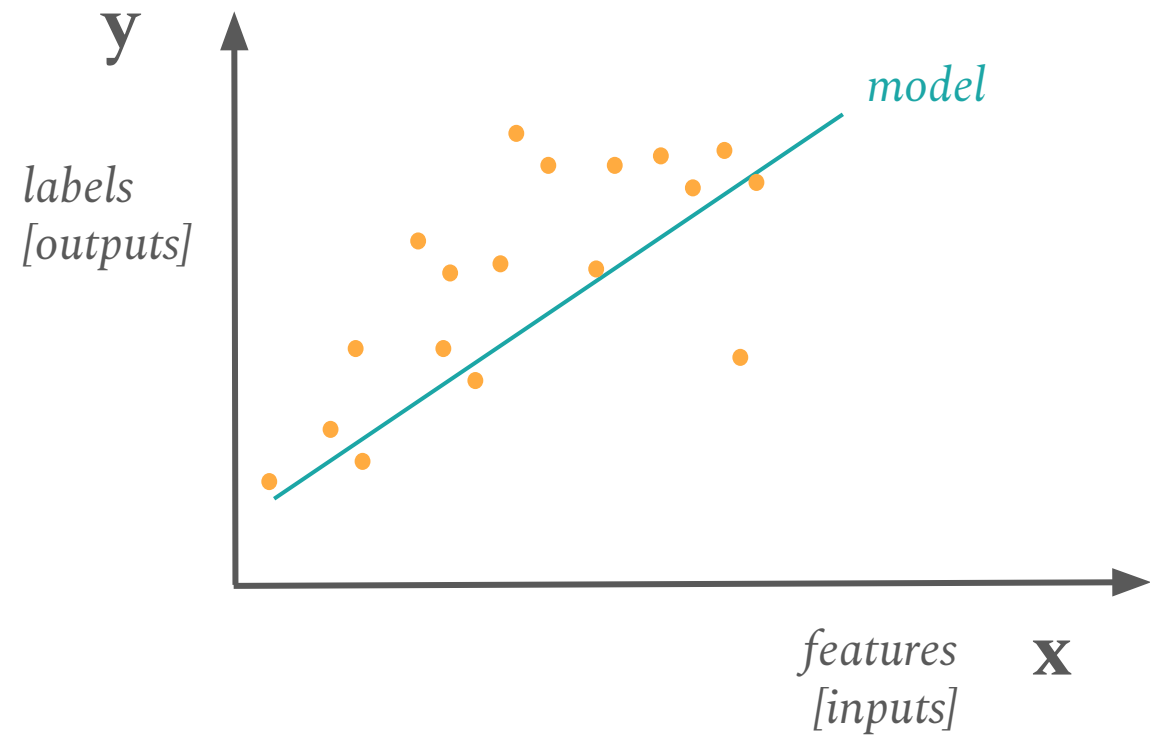


Model:

(w, b)

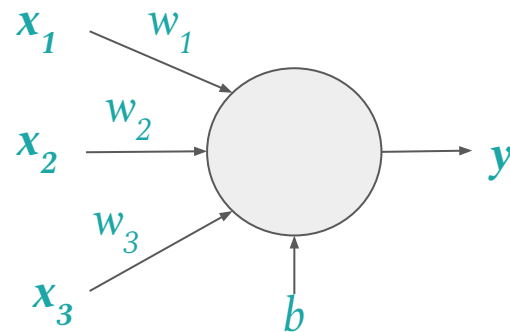
$w = \text{weight}$; $b = \text{bias}$

Linear regression is an example of least squares regression.



Linear Regression

Model Architecture:



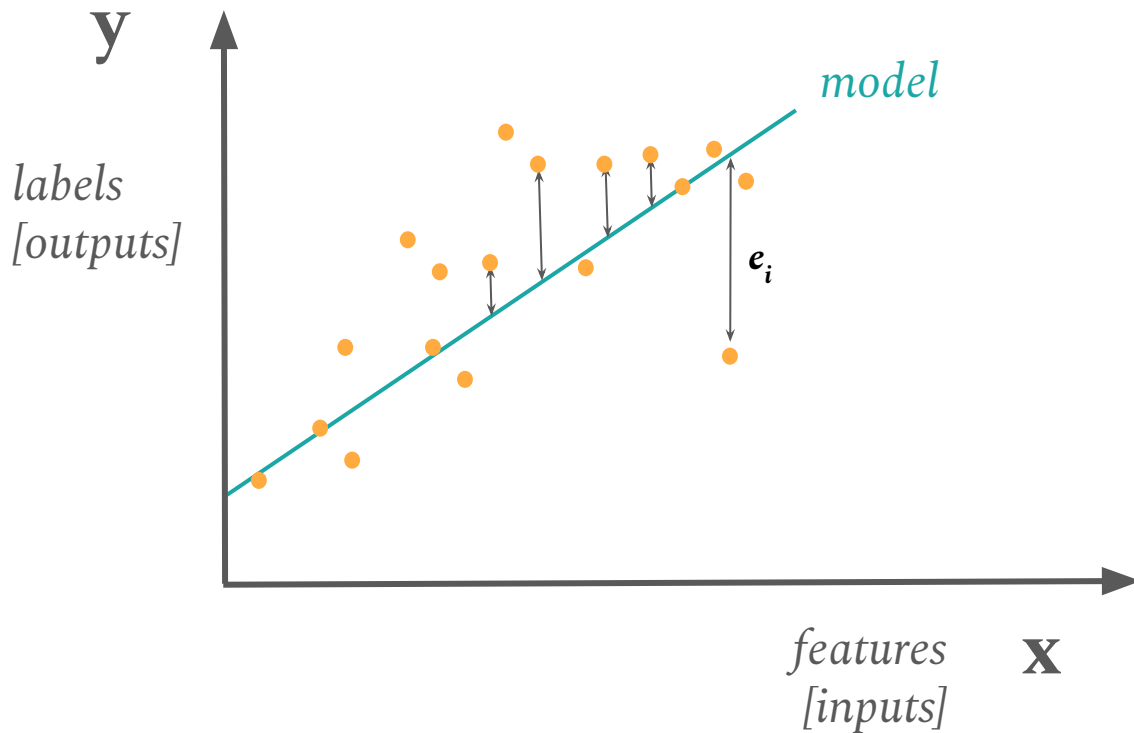
$$y = w_1x_1 + w_2x_2 + w_3x_3 + b$$

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)

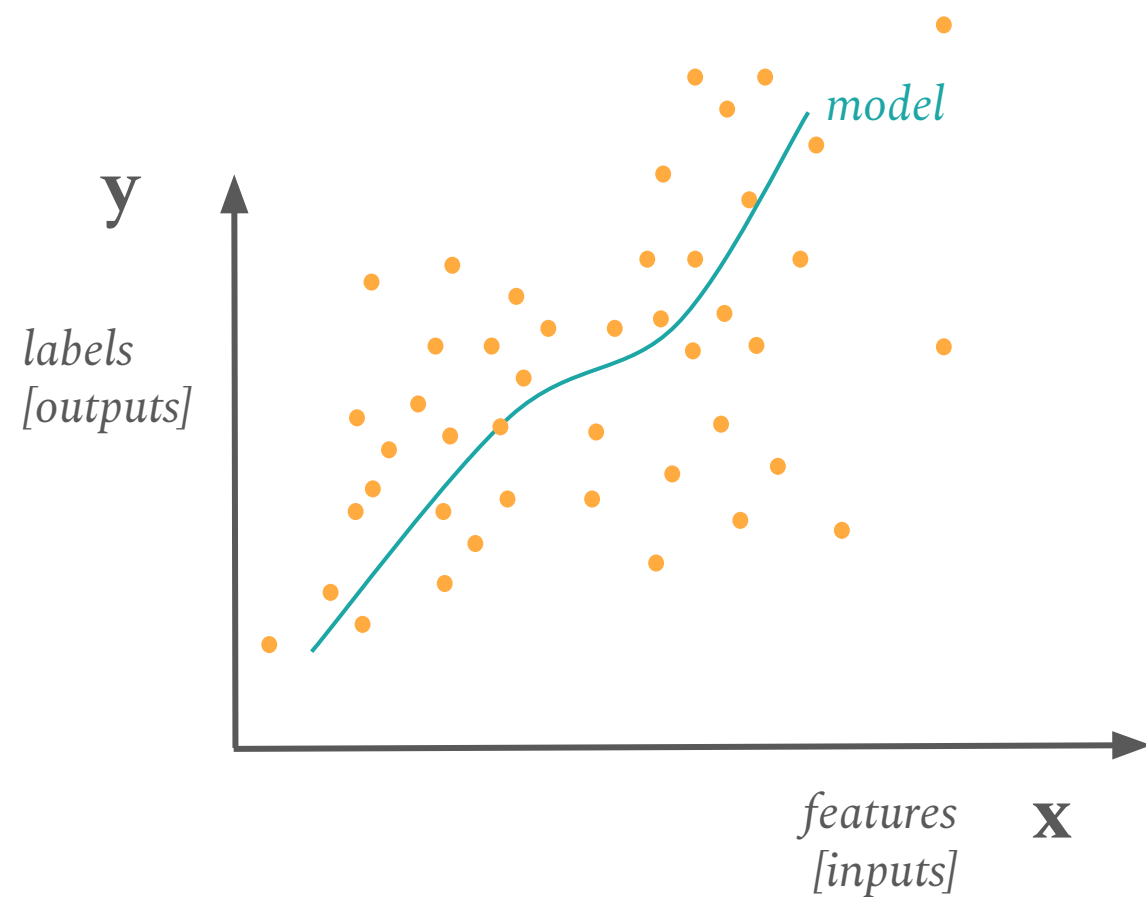
$w = \text{weight}$; $b = \text{bias}$

Model optimization:

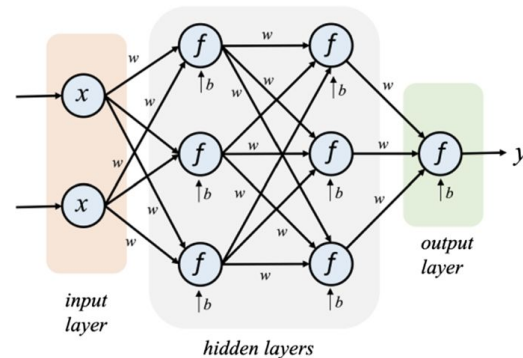
Minimize the cost function

- 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:

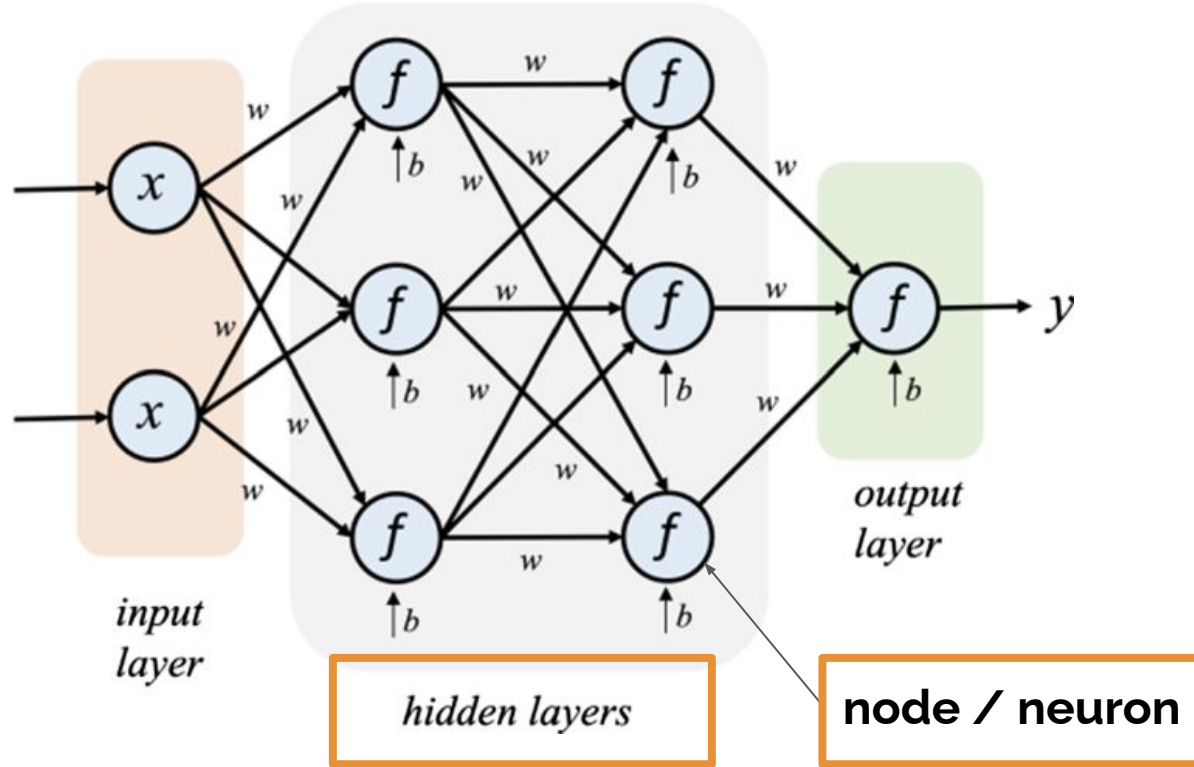


Model:



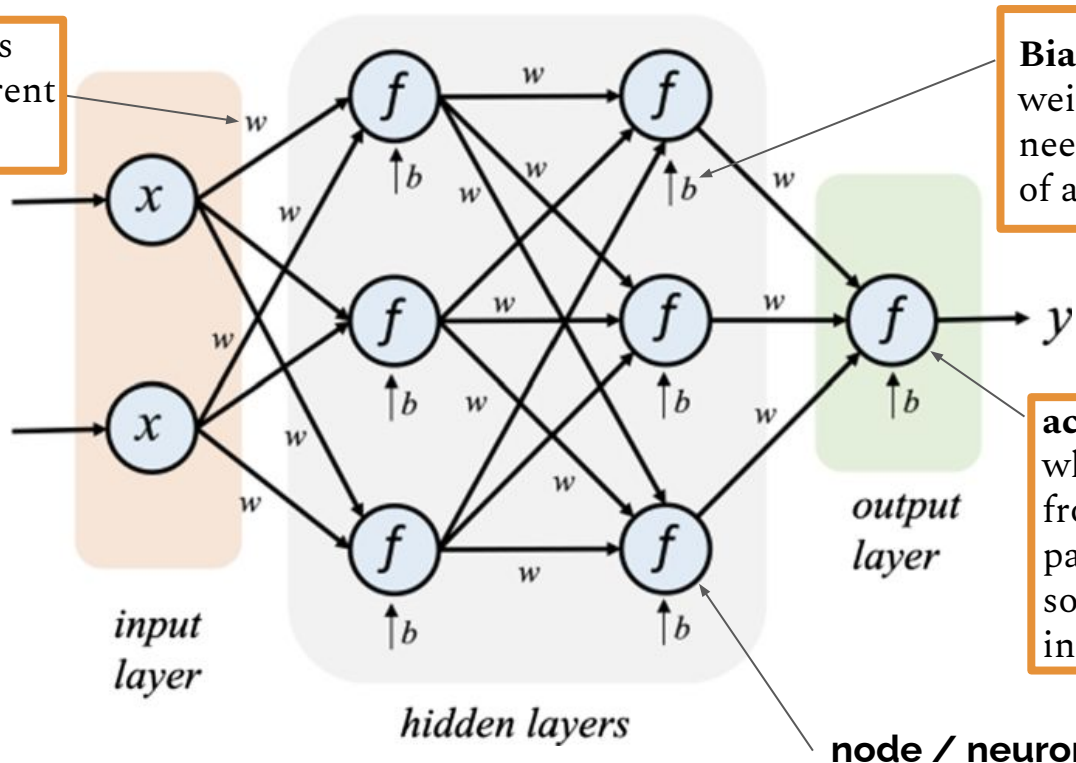
Model optimization: Minimize the **cost function** through the iterative process of gradient descent.

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*.

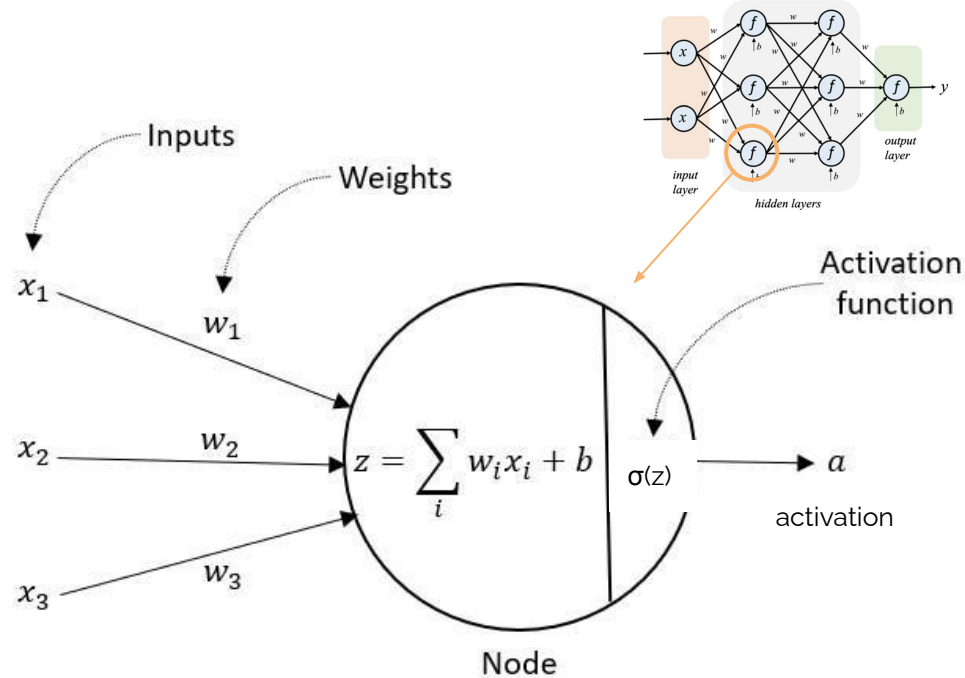
weights, w : connections between nodes of different layers



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?



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

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

Parameters:

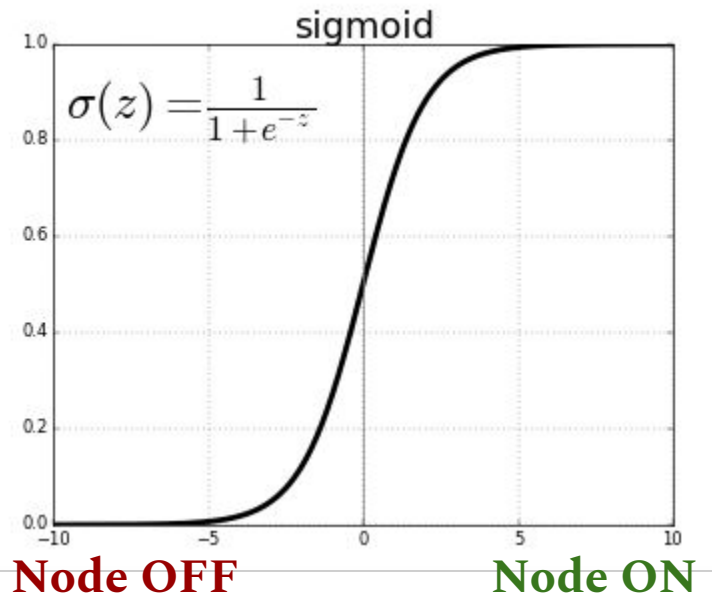
- **Inputs, x_i**
- **Weights, w_i** : connections between neurons
- **Biases, b** : how high the weighted sum of $x_i w_i$ needs to be for activation of a neuron
- **Activations, a** : number inside each neuron; determines whether or not neuron is “lit up”

Hyperparameters:

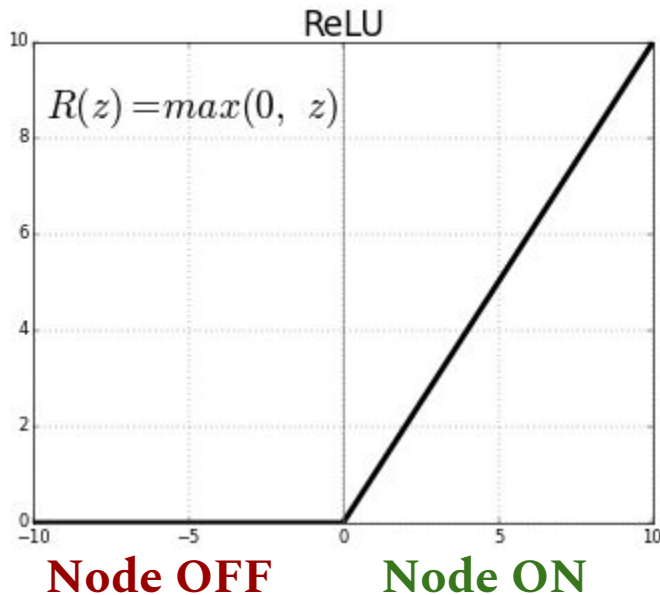
- **Activation function, σ** (sigmoid, ReLU, tanh, etc.): determines if information from specific node is propagated forward in the NN
- **Number of neurons / nodes**
- **Number of layers**
- **Number of epochs / iterations**

The activation function determines whether information from a node will be passed forward in the neural network.

Let some of the signal through depending on its strength

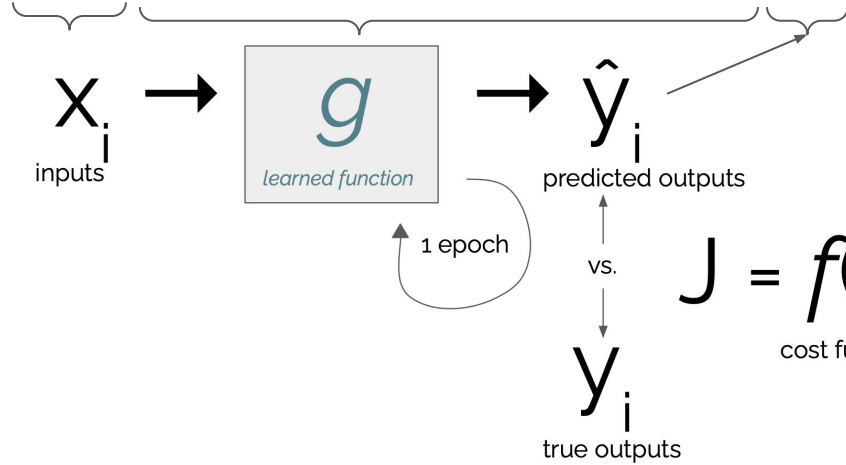
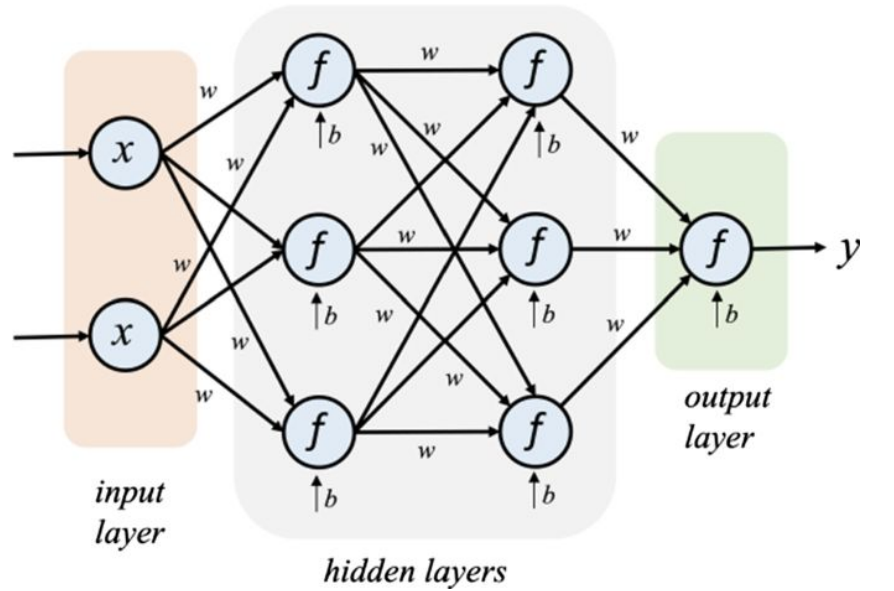


Only let the signal through if it's strong enough



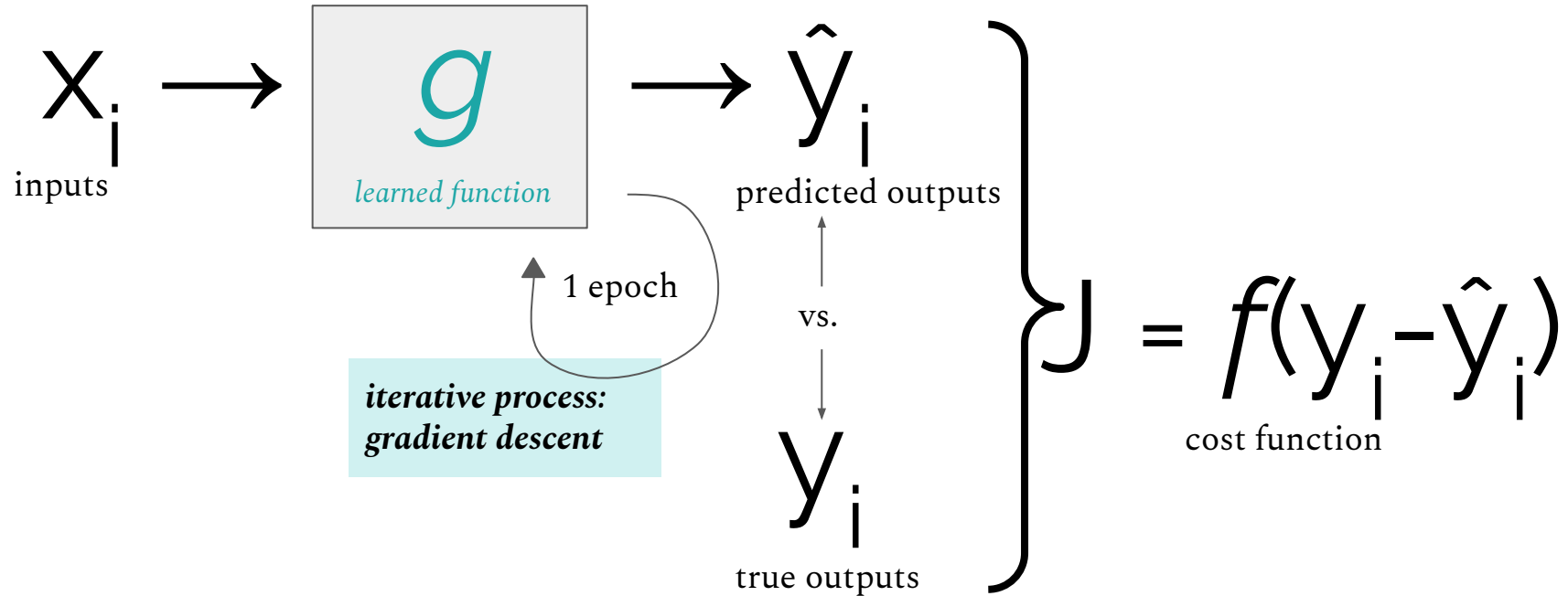
Node partially ON/OFF

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

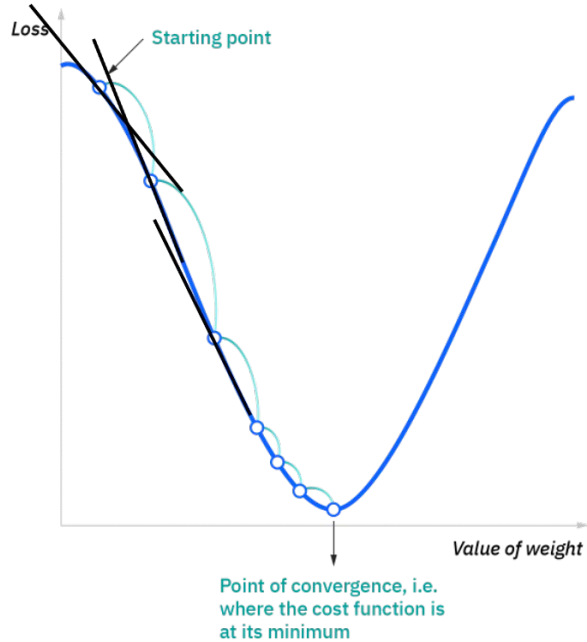


Neural Networks are trained through an *iterative process** that works to minimizing the cost function.

*iterative process = gradient descent



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.

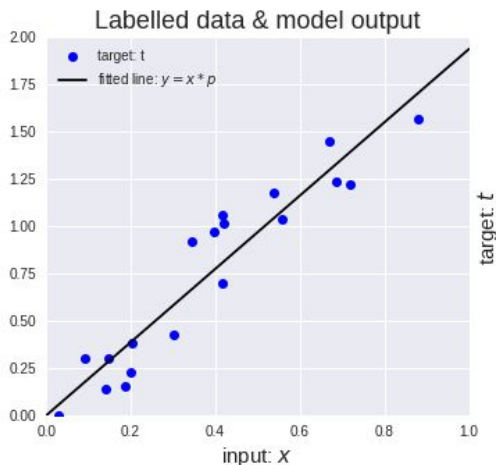
The negative of that gradient is followed to the next point and so on.

This is repeated until the minimum is reached.

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

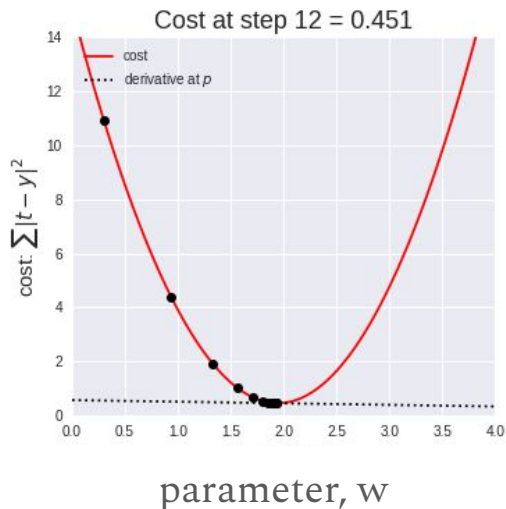
A linear model:

$$y = wx + b$$



The cost function:

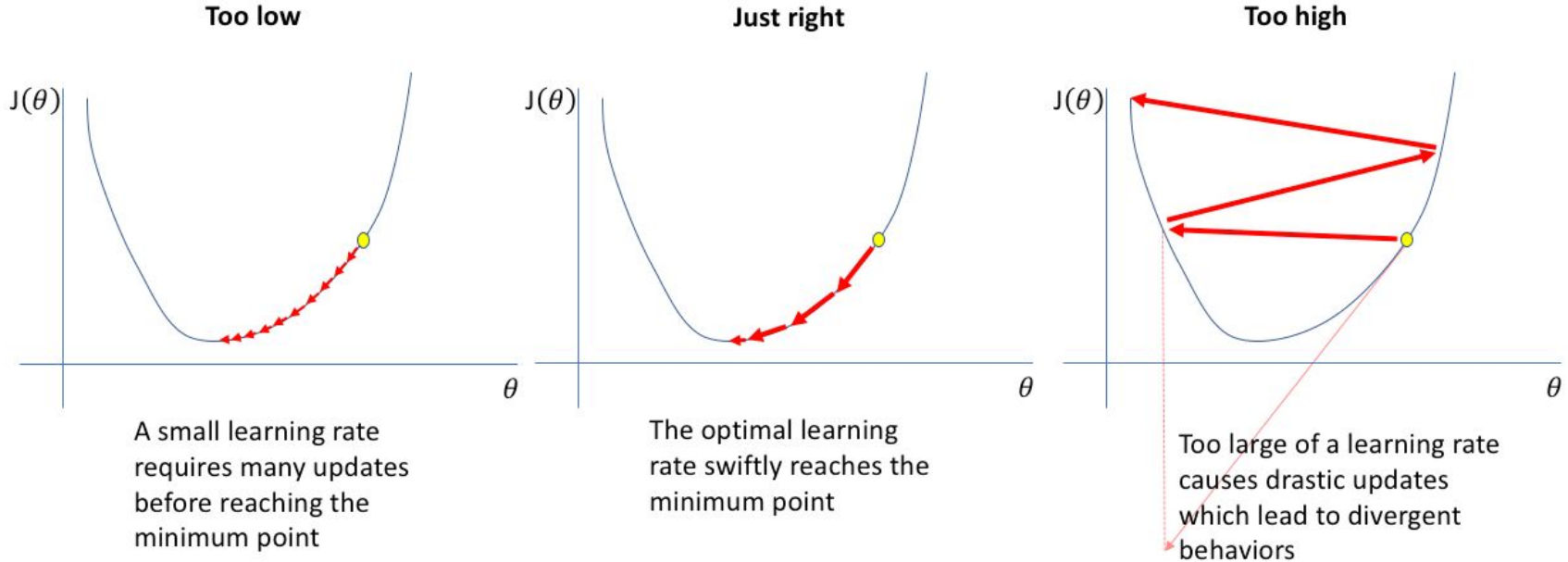
$$J = f(y_i - \hat{y}_i)$$



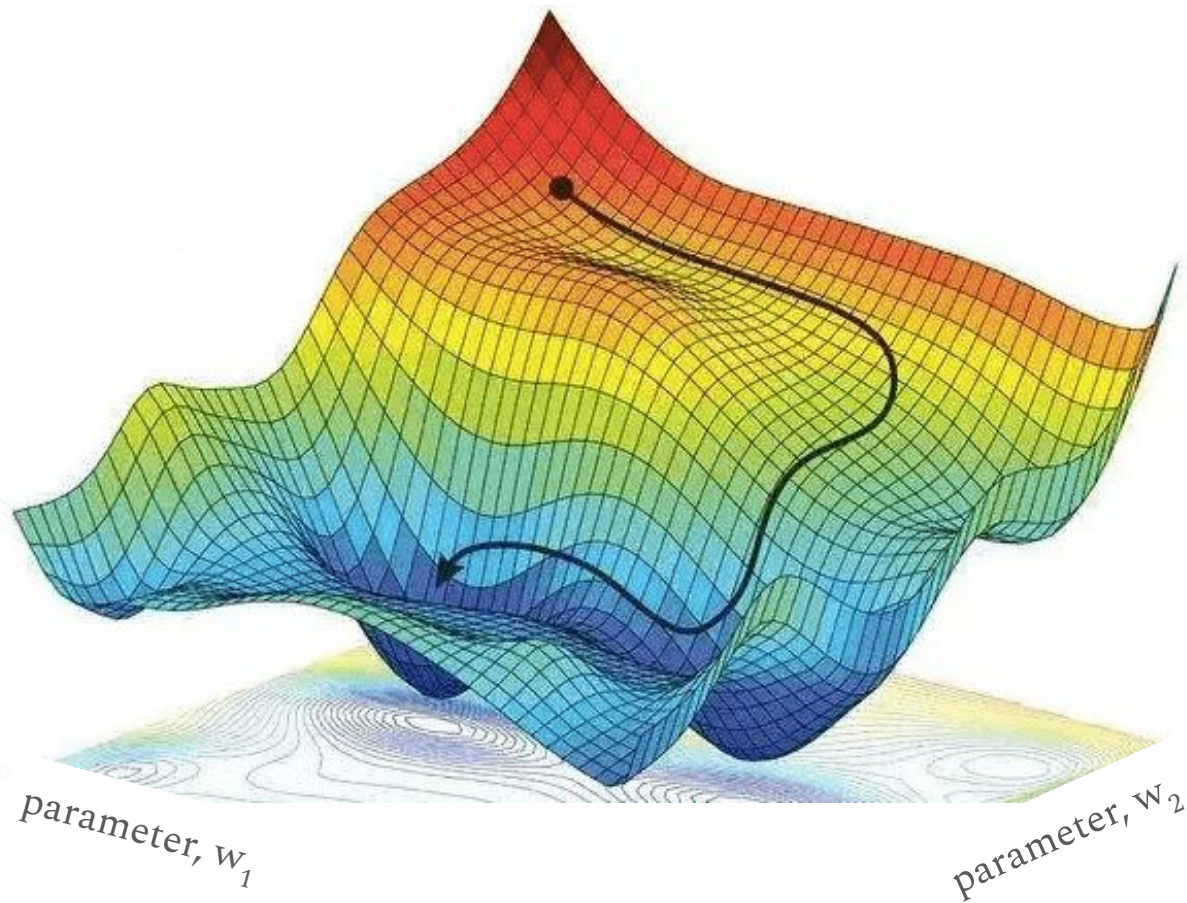
The cost function is updated iteratively based on the *gradient* and the *step size*.

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.



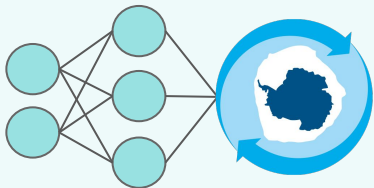
The take-home messages...

ML models learn statistical relationships in the data.

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.

<https://github.com/lahoffman/AI4InSync>



AI4InSync

A collaborative workshop to support observation system design with machine learning methods.

Workshop Outcomes:

- (a) Identify 1-3 scientific questions
- (b) Identify potential working groups

InSync Webinar // 8-10am // 22 May

Session 1

**Motivation &
Intro to ML**

Session 2

ML Tutorial

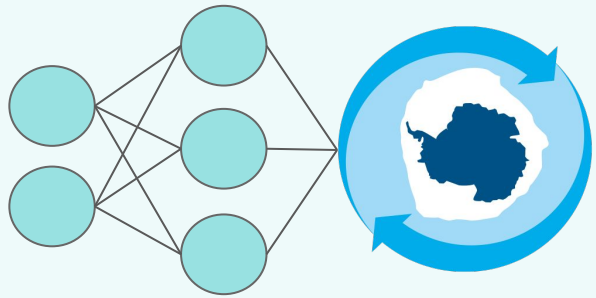
Session 3

**State-of-the-Art:
ML in Climate
Science**

Session 4

Discussion:
Identify science
questions

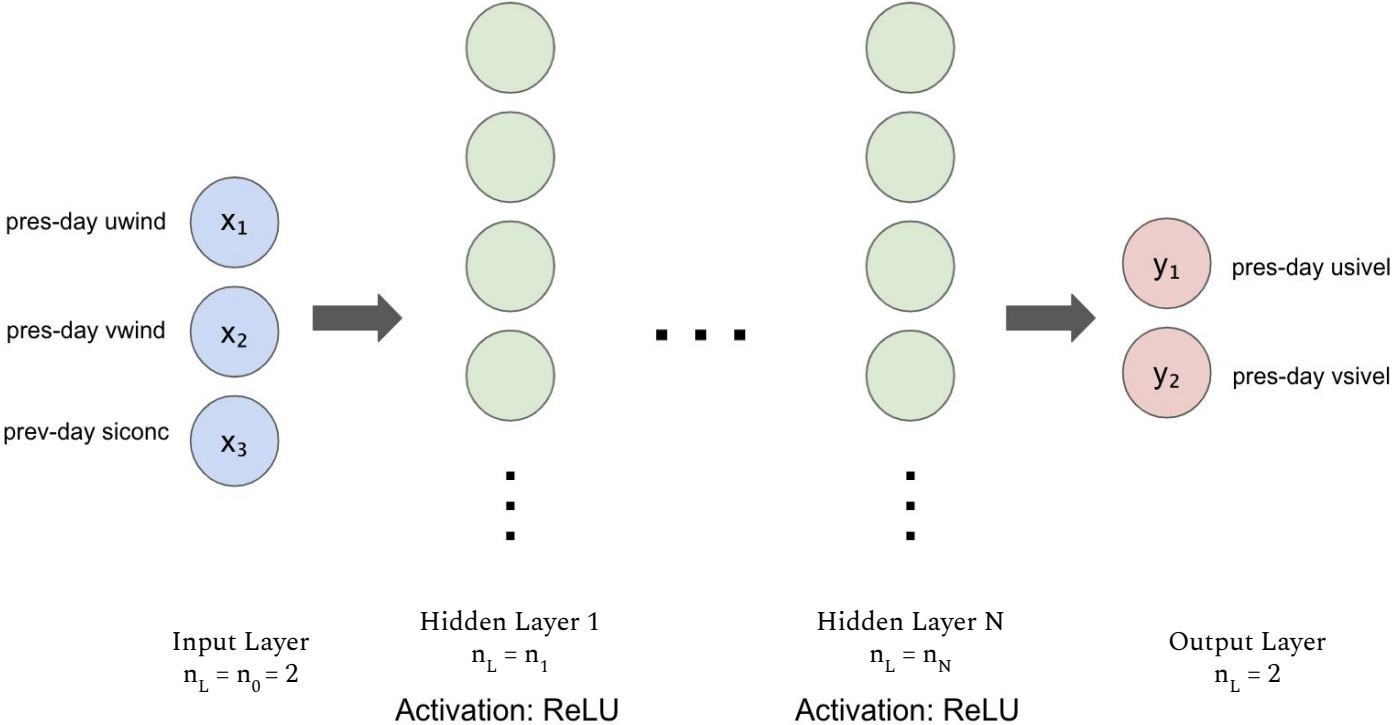
Define potential
working groups



AI4InSync

Tutorial

General NN architecture



Building a Machine Learning Model

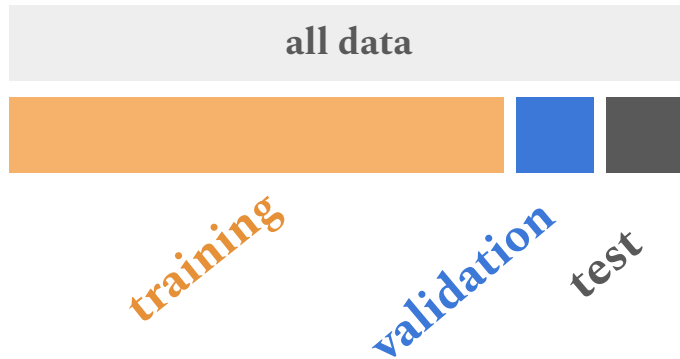
1. Set up environment
2. Data Preparation
 - a. Separate into 'Train-Validation-Test'
 - b. Standardize
3. Build the Model
 - a. Define parameters and hyperparameters
 - i. Loss function
 - ii. Number of nodes & layers
 - iii. Learning rate
 - iv. Activation function
 - b. Build the model
 - c. Compile 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
$$\text{data}_S = (\text{data} - \mu) / \sigma$$
- **Normalization:** from 0 to 1
$$\text{data}_N = [\text{data} - \text{min}] / [\text{max} - \text{min}]$$

Preparing the data.

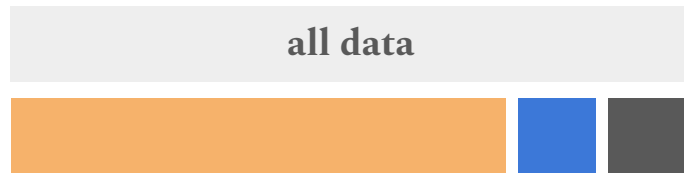
Data are split into training, validation and testing sets.



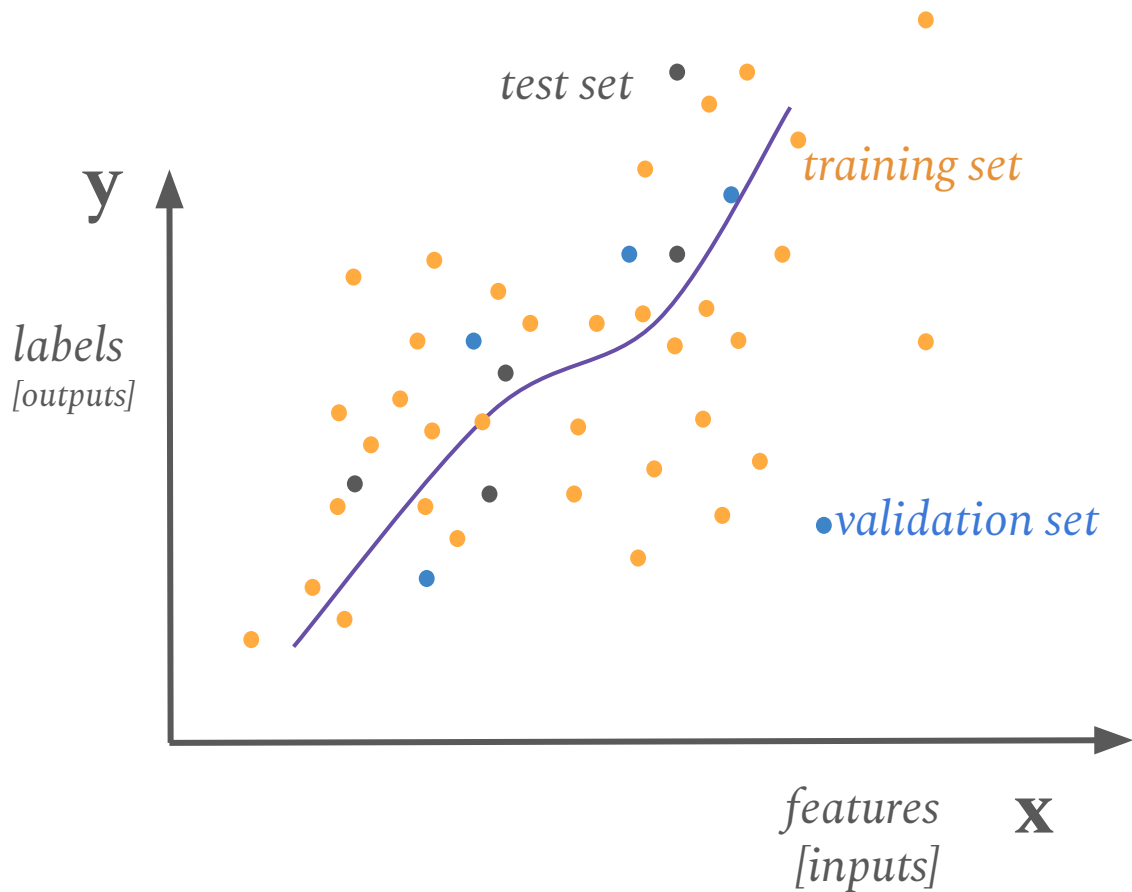
training: Data used to fit the model. The subset of data that the model uses to *learn* (i.e. optimize the cost function)

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.



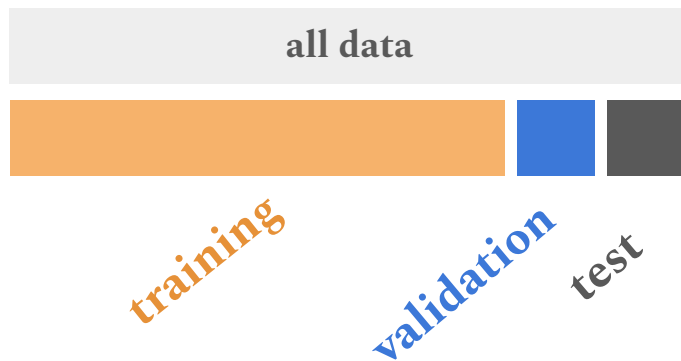
Data are split into training, validation and testing sets.



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.

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

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...

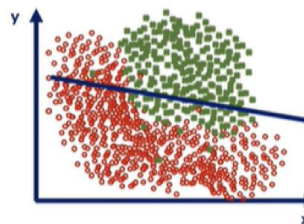
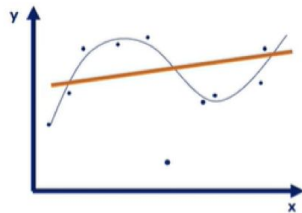
Parameters:

- **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”

There are several ways to reduce under / overfitting in ML models.

- Additional data
- Data splitting: train/val/test
- Regularization
- Dropout
- Hyperparameter tuning:
 - Learning Rate
 - Number of nodes & layers

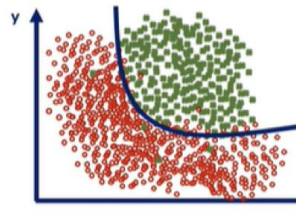
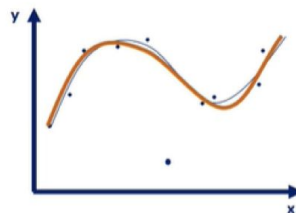
An **underfitted** model



Doesn't capture any logic

- High loss
- Low accuracy

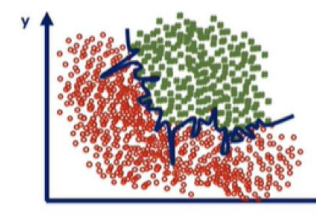
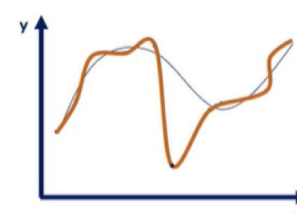
A **good** model



Captures the underlying logic of the dataset

- Low loss
- High accuracy

An **overfitted** model



Captures all the noise, thus "missed the point"

- Low loss
- Low accuracy

The *learning curve* gives us a sense of whether or not our model is over or underfitting.

	Underfitting	Just right	Overfitting
Symptoms	<ul style="list-style-type: none"> • High training error • Training error close to test error • High bias 	<ul style="list-style-type: none"> • Training error slightly lower than test error 	<ul style="list-style-type: none"> • Very low training error • Training error much lower than test error • High variance
Regression illustration			
Classification illustration			
Deep learning illustration			
Possible remedies	<ul style="list-style-type: none"> • Complexify model • Add more features • Train longer 		<ul style="list-style-type: none"> • Perform regularization • Get more data

Tutorial: global reconstructions of sea-ice velocity using a Neural Network.

Everything required to run this tutorial can be found at:

<https://github.com/lahoffman/AI4InSync>

Open the Google Colaboratory Notebook ('nn_reconstruction.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).

What are your needs for an observing system?

- Variables?
- Spatio-temporal scales?

How do you define the *value* of an observing system?

What objective function would you optimize?

- Reducing mean state error?
- Constraining long-term trends?
- Improve forecast skills?
- Capture extreme events?
- Uncertainty reduction?

How could optimizing for one metric harm another?