Material --> raghavian.github.io/talks

Let’s start with a quiz!

1. Log on to socrative.com
2. Choose “Student Login”
3. Room name: **RAGHAV**

A Sneak Peek into ML for Chemistry

Raghavendra Selvan

Assistant Professor
Dept. of Computer Science
Dept. of Neuroscience
Data Science Lab
raghav@di.ku.dk
raghavian.github.io
@raghavian

UNIVERSITY OF COPENHAGEN

Work in groups for exercise sessions

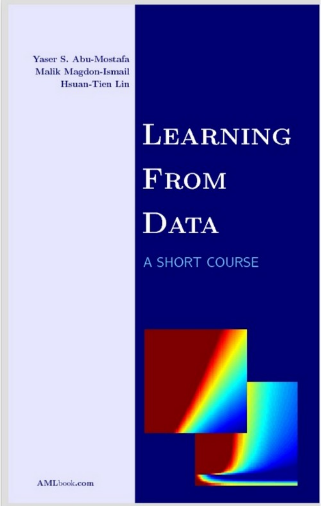
* Ideally one member can parse Python :)

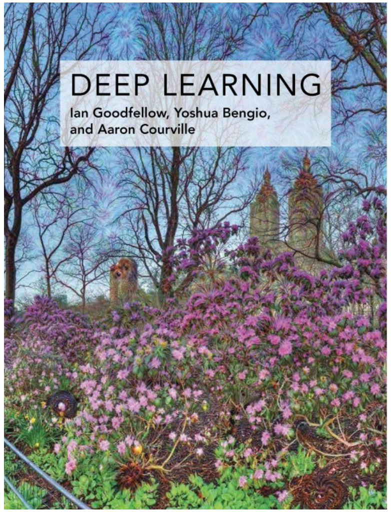
Overview of Day 3 & 4

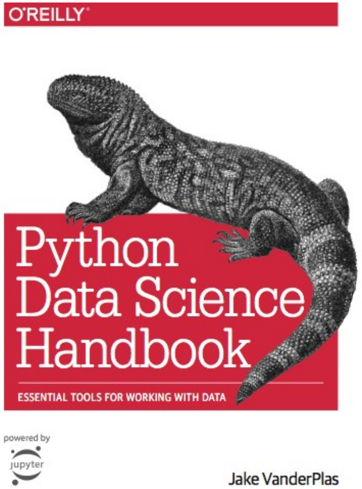
- **Session-1: Learning from Data**
  - Introduction (again)
  - Basics of ML
  - Data preparation (Exercise)
- **Session-2: Perceptron Learning Algorithm**
  - Supervised ML
  - Perceptron algorithm
  - Predicting structure type from PDF (Exercise)
- **Session-3: Multi-Layer Perceptron**
  - Deep Learning
  - Unsupervised Clustering of PDF (Exercise)
- **Session-4: Recent trends in Chem+ML**

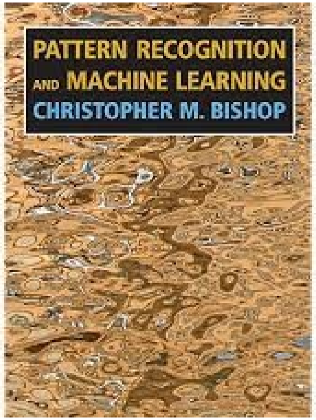
Literature

1. Pattern Recognition and Machine Learning (link)
2. Deep Learning, Goodfellow et al.(link)
3. Learning from Data, Mostafa et al. (link)
4. Python Data Science Handbook (link)





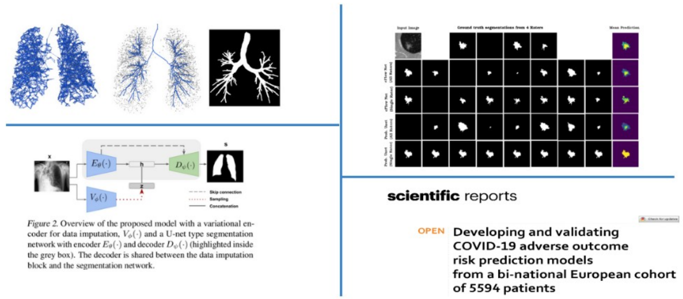




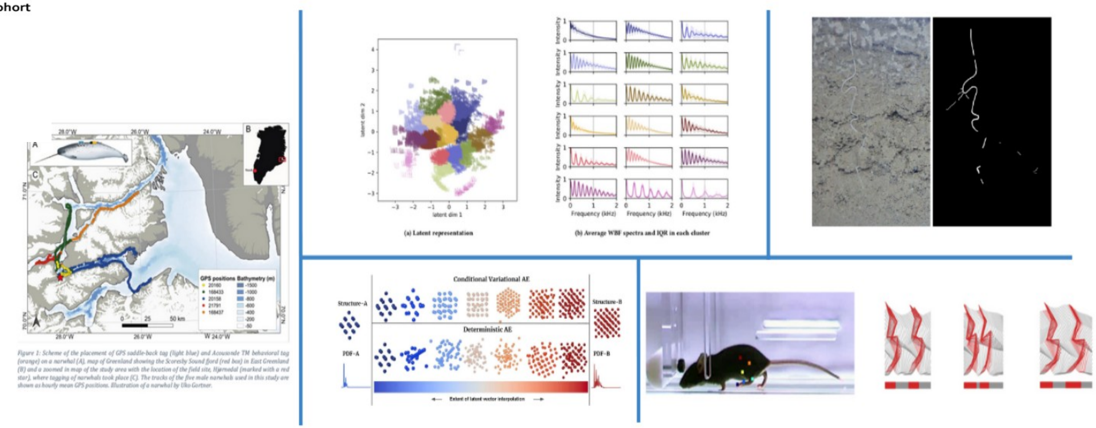
Overview

- Design-based methods
- Learning from data
- Underlying data distributions
- Perceptron Learning Algorithm
- Generalization error

Learning from what type of data?

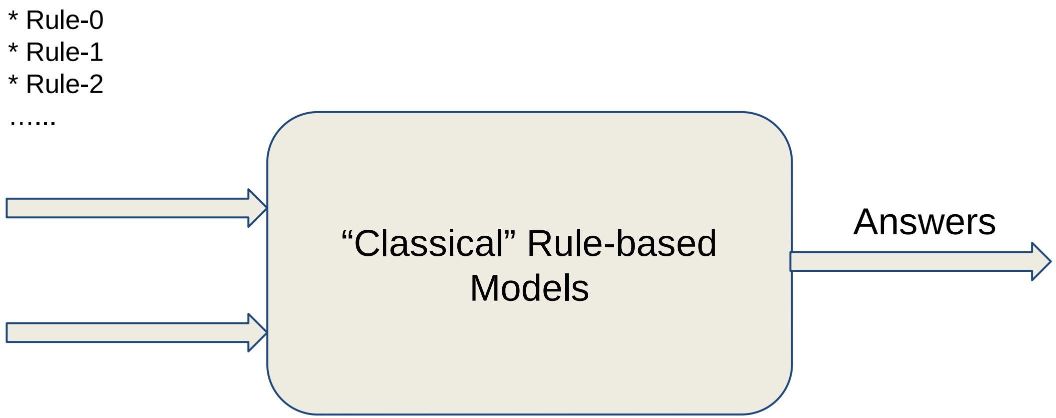


[1] Graph Refinement based Airway Extraction using Mean-Field Networks and Graph Neural Networks (2020), Extraction of Airways from Volumetric Data (2018) - PhD Thesis
[2] Uncertainty quantification in medical image segmentation with Normalizing Flows (2020)
[3] Lung Segmentation from Chest X-rays using Variational Data Imputation (2020)

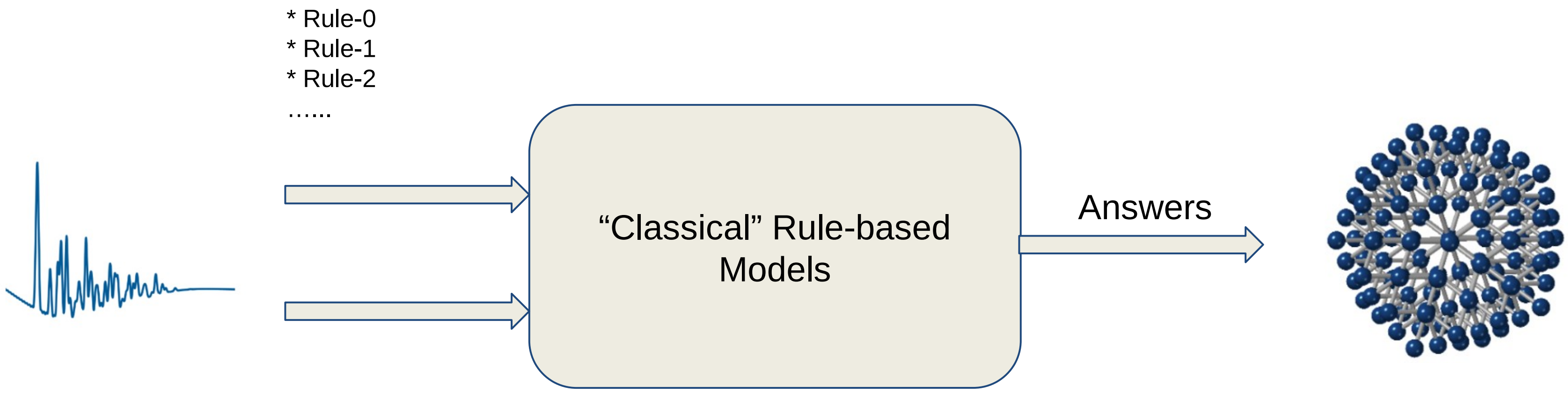


[1] Detection of foraging behavior from accelerometer data using U-Net type convolutional networks (2021)
[2] Dynamic β-VAEs for quantifying biodiversity by clustering optically recorded insect signals (2021)
[3] Segmentation of Roots in Soil with U-Net (2020)
[4] Characterising the atomic structure of mono-metallic nanoparticles from x-ray scattering data using conditional generative models (2020)
[5] Locomotor deficits in ALS mice are paralleled by loss of V1-interneuron-connections onto fast motor neurons (2020)

Design-based methods

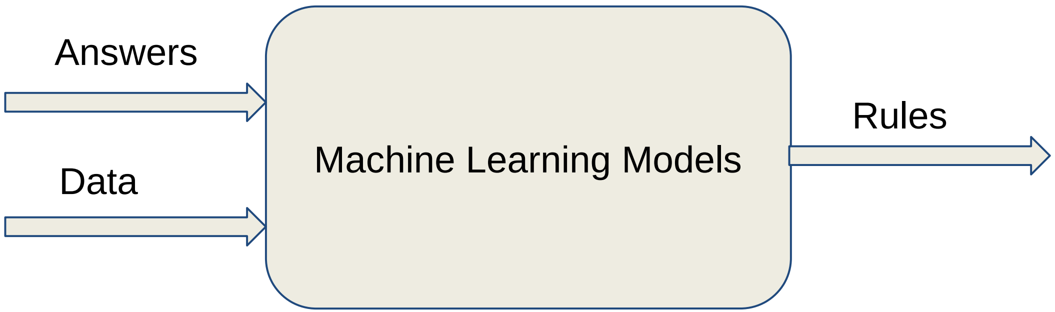


Design-based methods

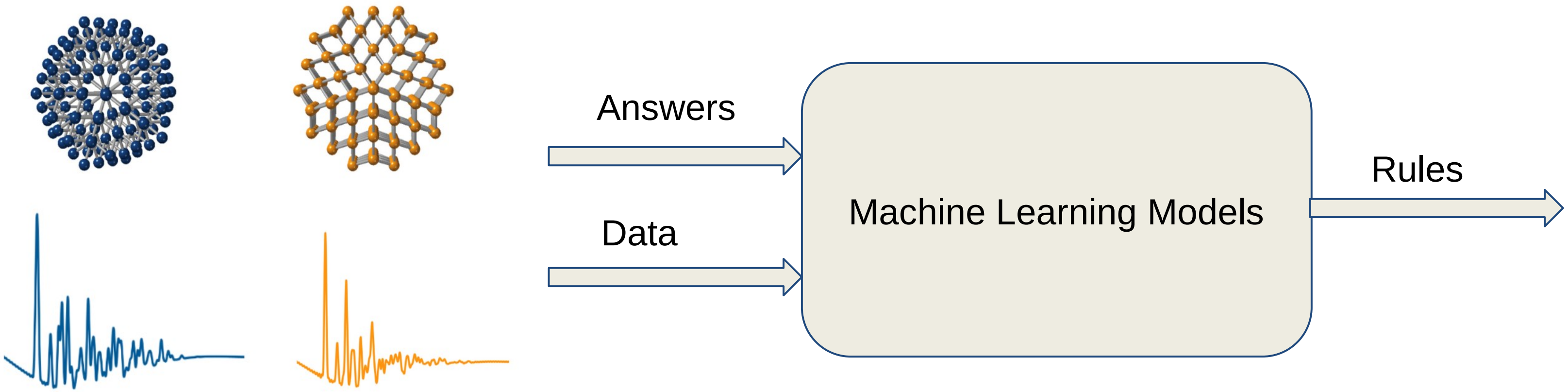


Machine Learning = Learning from Data

Machine Learning = Learning from Data



Machine Learning = Learning from Data



Machine Learning = Learning from Data

- ML is a lot about discovering patterns
  - Big data
  - Big computers
  - More complex patterns, than before

Machine Learning = Learning from Data

- ML is a lot about discovering patterns
  - Big data
  - Big computers
  - More complex patterns, than before
- Learning from examples
  - Natural to humans
  - Temptation to call it AI

Machine Learning = Learning from Data

- ML is a lot about discovering patterns
  - Big data
  - Big computers
  - More complex patterns, than before
- Learning from examples
  - Natural to humans
  - Temptation to call it AI
- What you have is what you get (mostly)
  - Large & diverse datasets
  - Features and flaws are learned

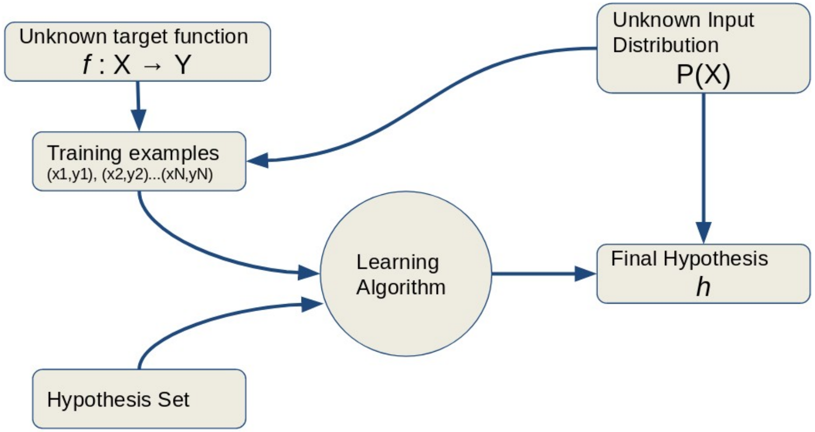
Machine Learning Fundamentals

- Basics of Machine learning
- Types of learning
- Principles of Learning

A learning algorithm

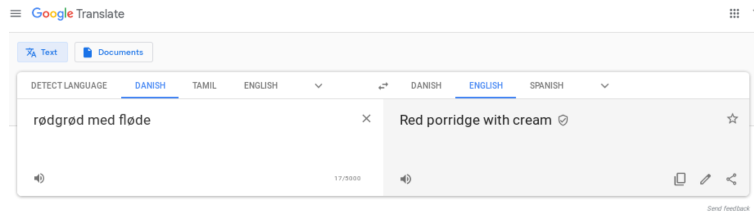
"A computer program is said to learn from experience **E** *with respect to some class of tasks* **T** *and performance measure* **P***, if its performance at tasks in* **T***, as measured by* **P***, improves with experience* **E***."*

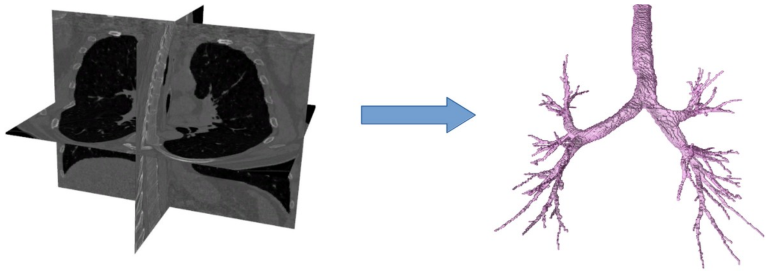
Mitchell, Tom M. Machine learning (1997)

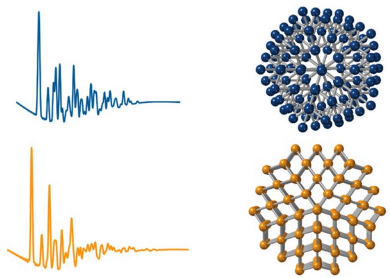


The Task, T

- Classification
- Regression
- Transcription
- Machine translation
- Face recognition
- Anomaly detection
- Synthesis & sampling
- Denoising
- Density estimation
- Self-driving







The Performance measure, **P**

Not always straightforward but most common:

- Accuracy
- Error rates/ losses (0-1 loss)
- Log probability
- KL divergence

https://thispersondoesnotexist.com/

http://www.thisworddoesnotexist.com/

The Experience, **E**

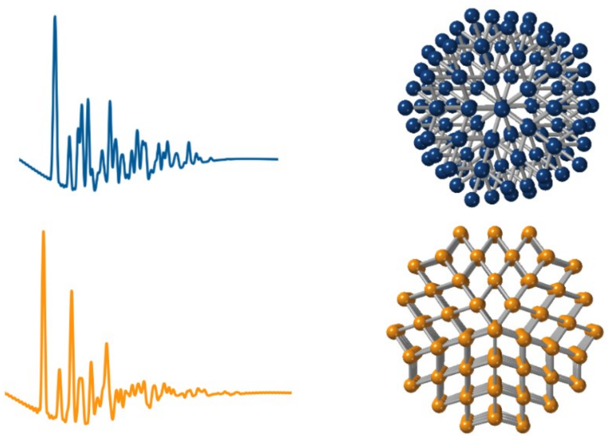
All the ways information can enter the model primarily as:

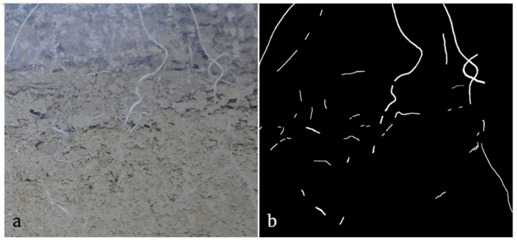
- Prior information
- Hyper-parameters
- Data/ supervision

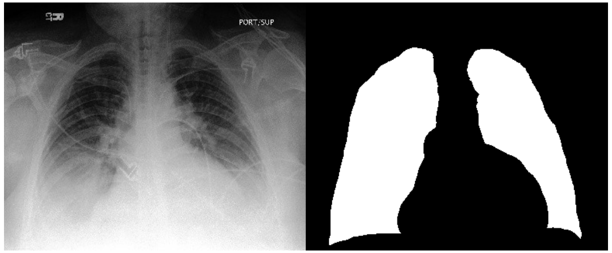
More concrete classification of ML methods is based on **E**

Supervised Learning

- Strong labels for the entire dataset
- (Relatively) Easy to train
- Hard to obtain high quality labels
- Ex: Image Segmentation

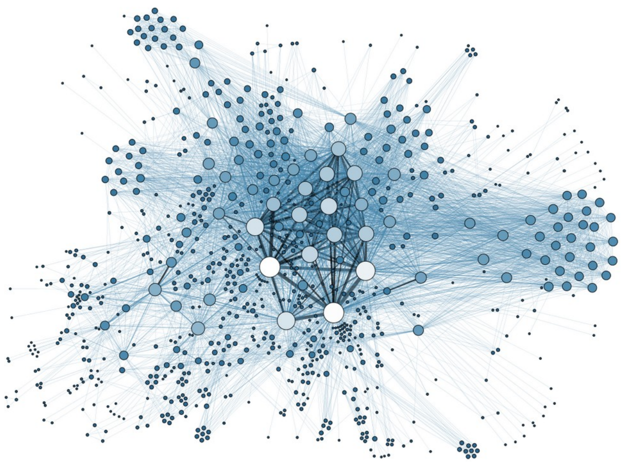


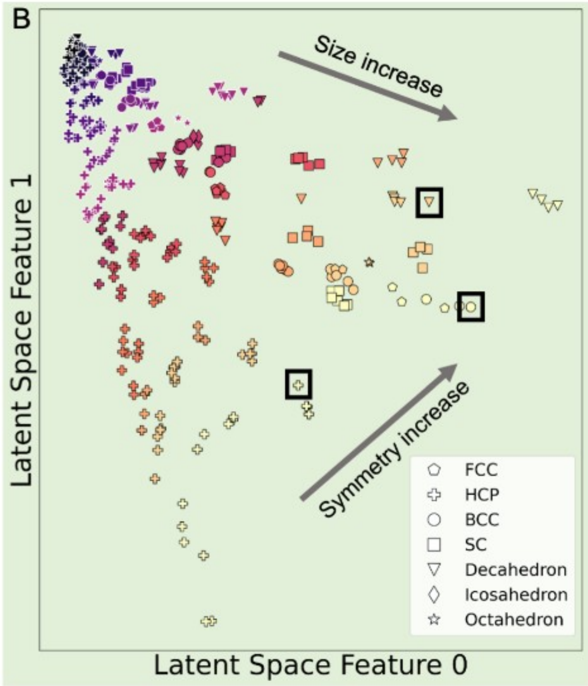




Unsupervised learning

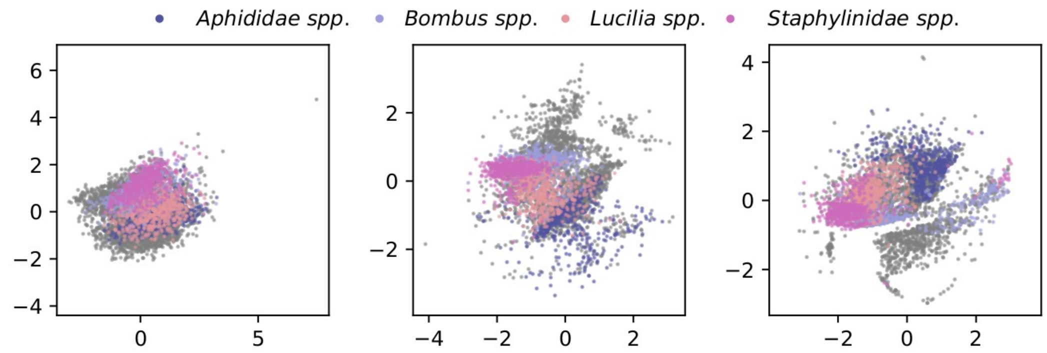
- No labels.
- “Figure it out yourself” model
- Ex: Social networks, Gene expression networks





Semi-supervised learning

- Strong labels for some of the data
- Weak labels for all of the data
- Can be useful in cases where strong labels are hard!
- Ex: Captcha



(a) PCA (b) Unsupervised model (c) Semi-supervised model

Reinforcement learning

- Combination of strong and weak labels
- Online learning
- Constant learning
- Ex: Streaming services recommendation

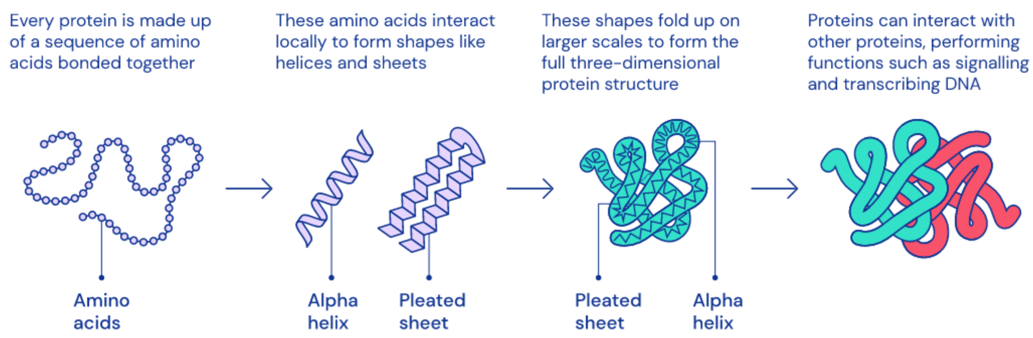


Figure 1: Complex 3D shapes emerge from a string of amino acids.

https://deepmind.com/blog/article/AlphaFold-Using-AI-for-scientific-discovery

More….

- Self-supervised
- Active learning
- Continual learning
- Meta-learning
- ……

More….

- Self-supervised
- Active learning
- Continual learning
- Meta-learning
- ……

We will focus on supervised and unsupervised learning methods.

Formulate your learning task

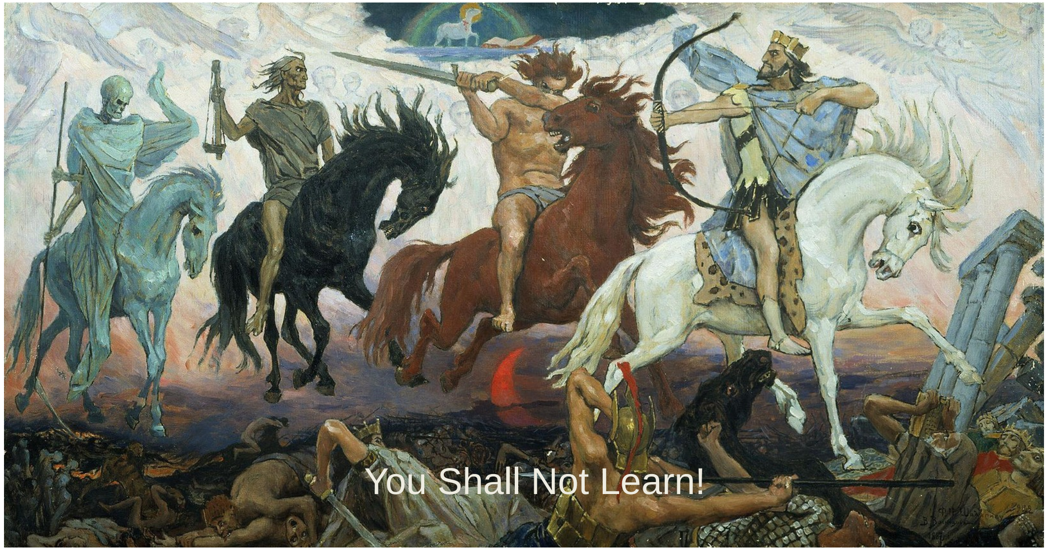
- Task **T**
- Performance **P**
- Experience **E**

We will discuss this in the exercise session.

Principles of Learning

Four horsemen of ML failure

1. Data assumptions
2. Data snooping
3. Underfitting
4. Overfitting



Data assumptions

1. **i.i.d**
    - **Identical:** Data is drawn from the same data distribution
    - **Independent:** Data points independent from each other
2. Sampling/Selection bias

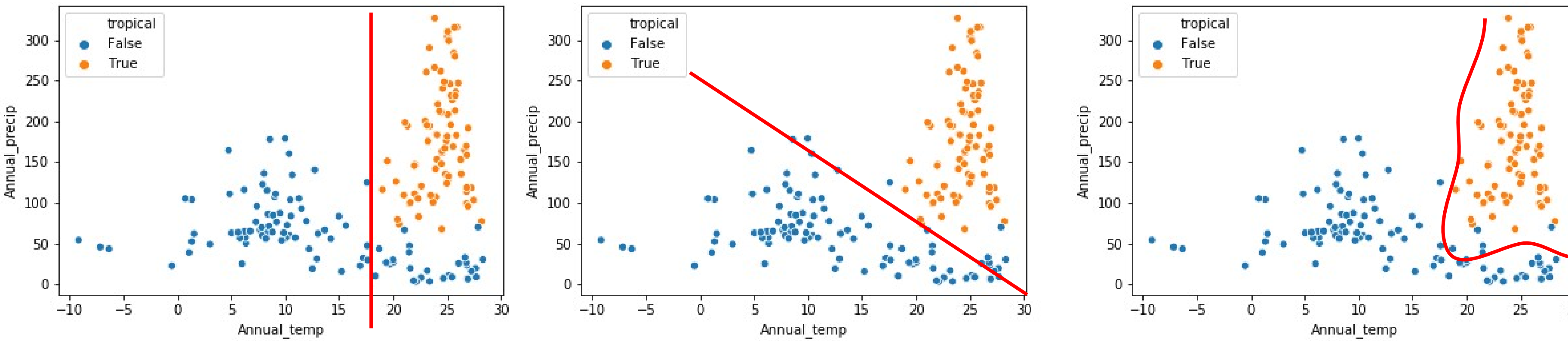
- If i.i.d assumption is violated does learning work?
- How can we overcome?

Data Snooping

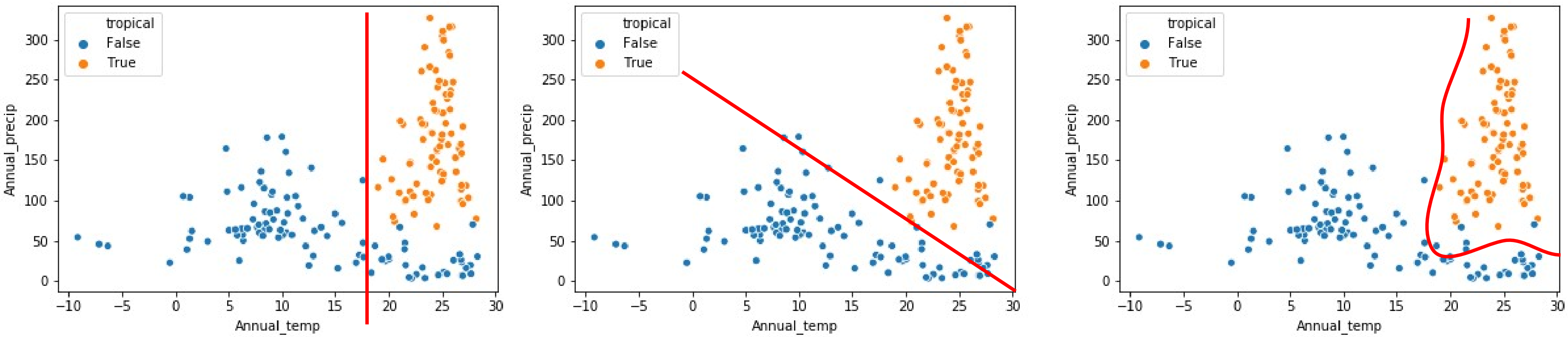
- Test data has informed the model selection
- Generalization suffers

“If you want an unbiased assessment of your learning performance, you should keep a test set in a vault and never use it for learning in any way”
Mostafa et al. Learning from data (book)

Underfitting & Overfitting



Underfitting & Overfitting



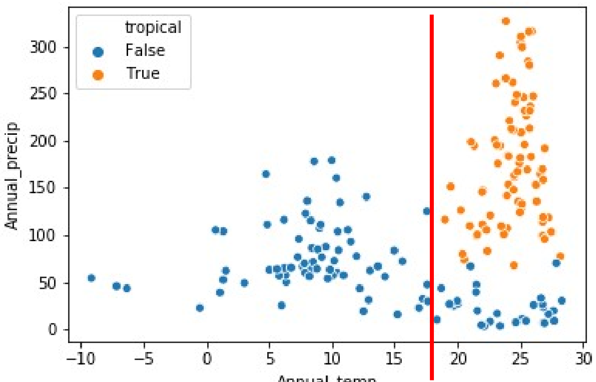
Underfitting

Appropriate capacity

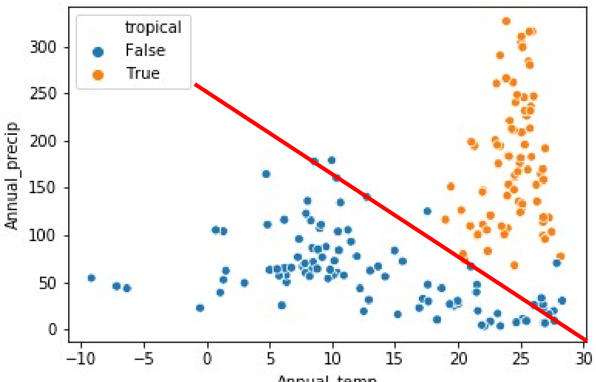
Overfitting

Underfitting & Overfitting

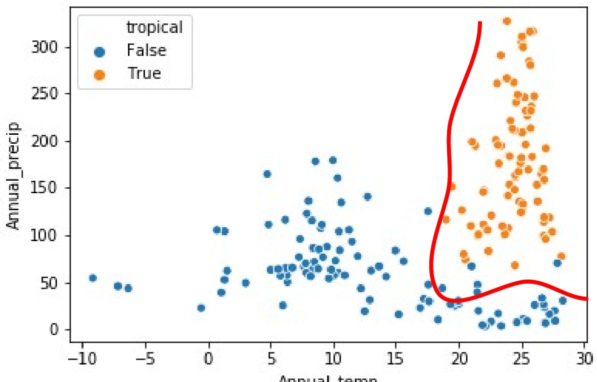
- Models are chosen based on training error
- Test error ≥ Training error



Underfitting



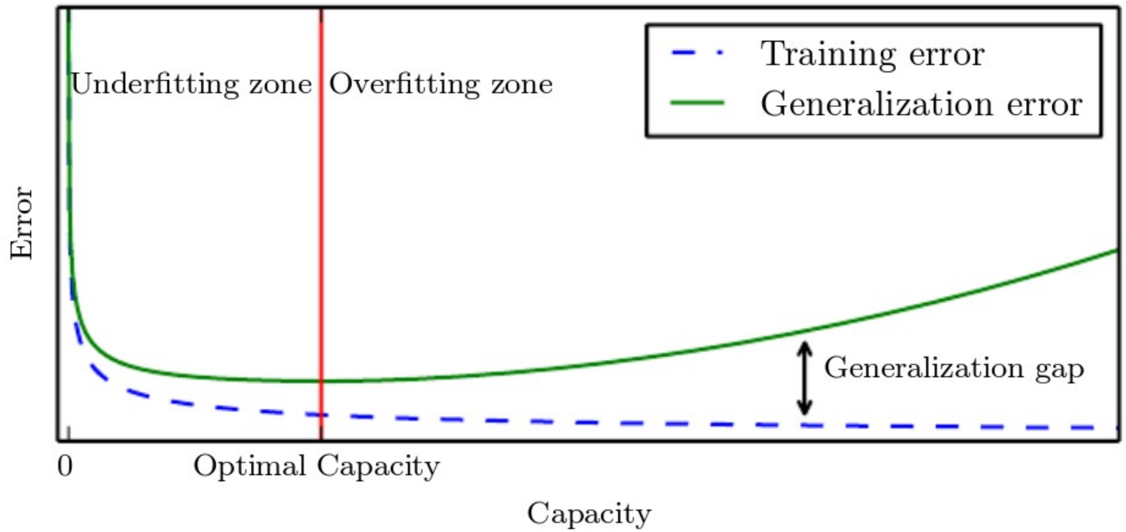
Appropriate capacity



Overfitting

Handling overfitting

- Representational capacity
    - **Occam's Razor:** *"The simplest model that fits the data is also the most plausible."*

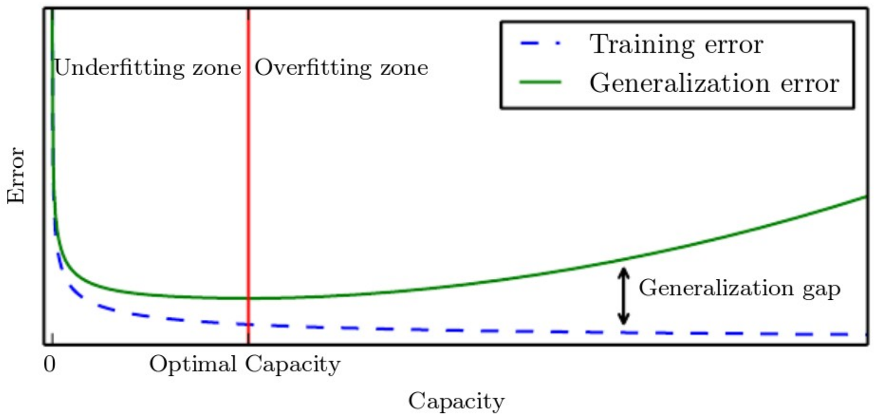


Summary of Learning Principles

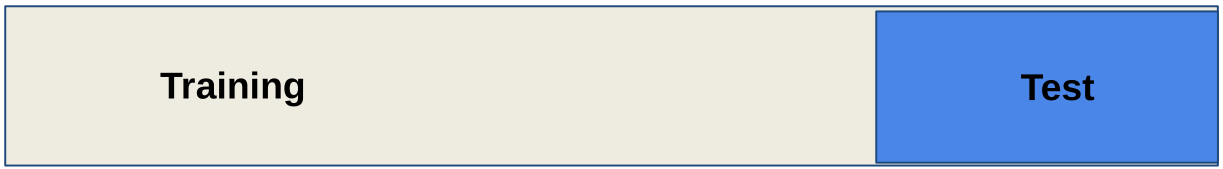
- Data is not ideal
- Lock away test data
- Low generalization error is the *Holy Grail* of all ML
- Model capacity is hard to decide, even with Occam's Razor
- Underfitting & Overfitting can hamper performance

Model Selection & Validation

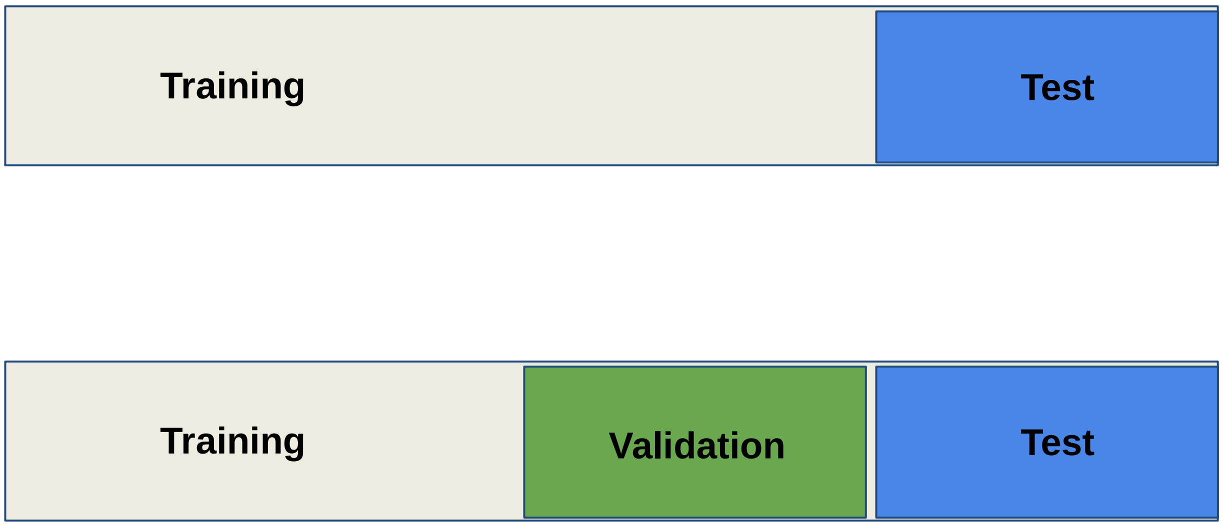
- How to avoid Overfitting
- How to pick models based on training error



Validation Set comes to the rescue

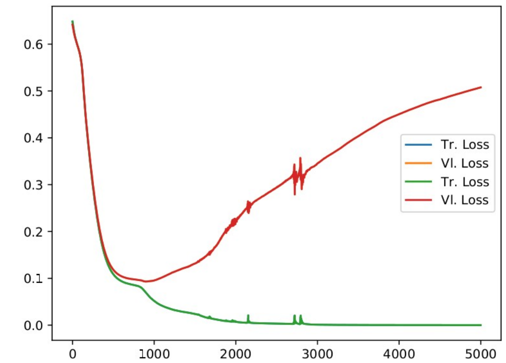


Validation Set comes to the rescue



Validation Set comes to the rescue

- Training data for training
- Validation data for model selection
- Hyper-parameters can be selected with it
- Rule of thumb: 60-20-20



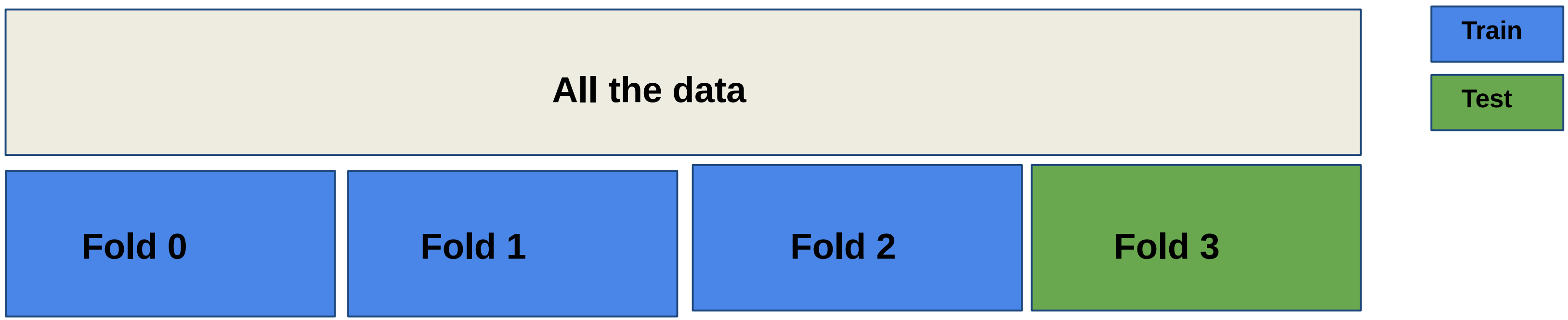
Consequences:

- Reduction in training data
- Computational overhead

Cross-validation gives more training data

All the data

Cross-validation gives more training data



Cross-validation gives more training data

Summary

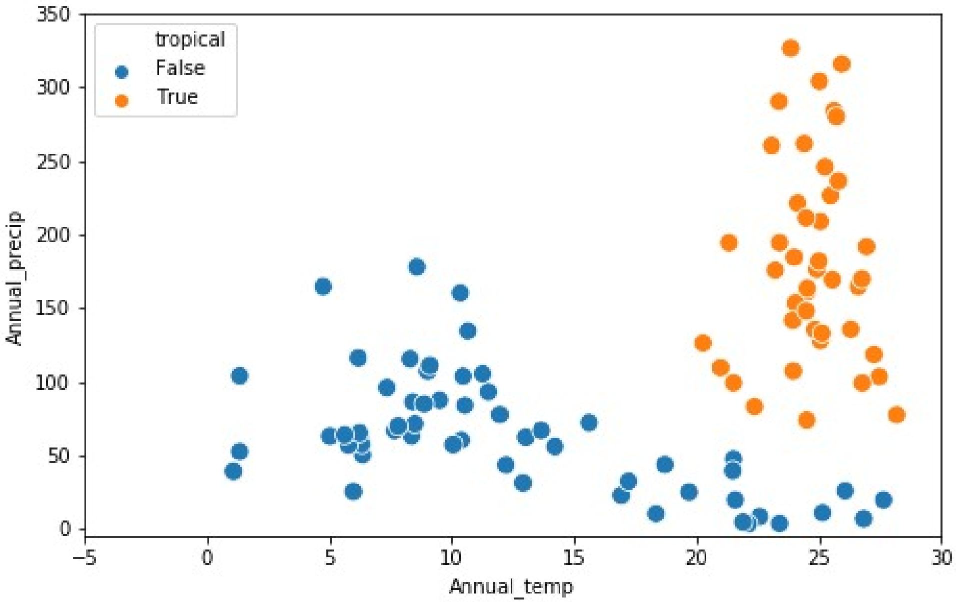
- Models selection is not straightforward
- Pick a class of models -> Tune hyper-parameters
- Training data to select models
- Generalization suffers if only based on training data
- Use part of training data for validation
- Cross validation to the rescue (?)

Exercise on Data Preparation

Session 2

A first ML algorithm…..

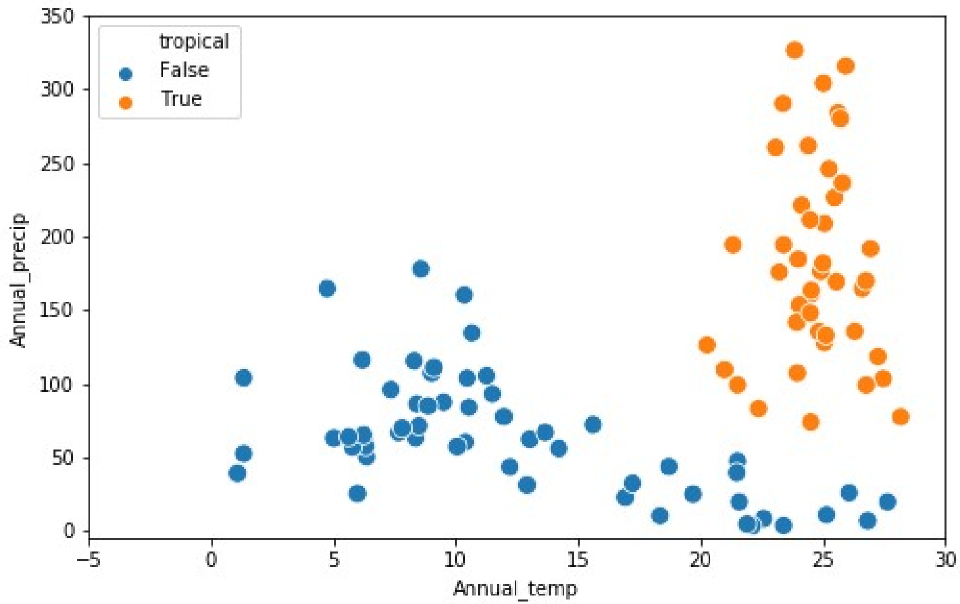
Linear Separability



https://datacatalog.worldbank.org/dataset/climate-change-knowledge-portal-historical-data

Linear Classification

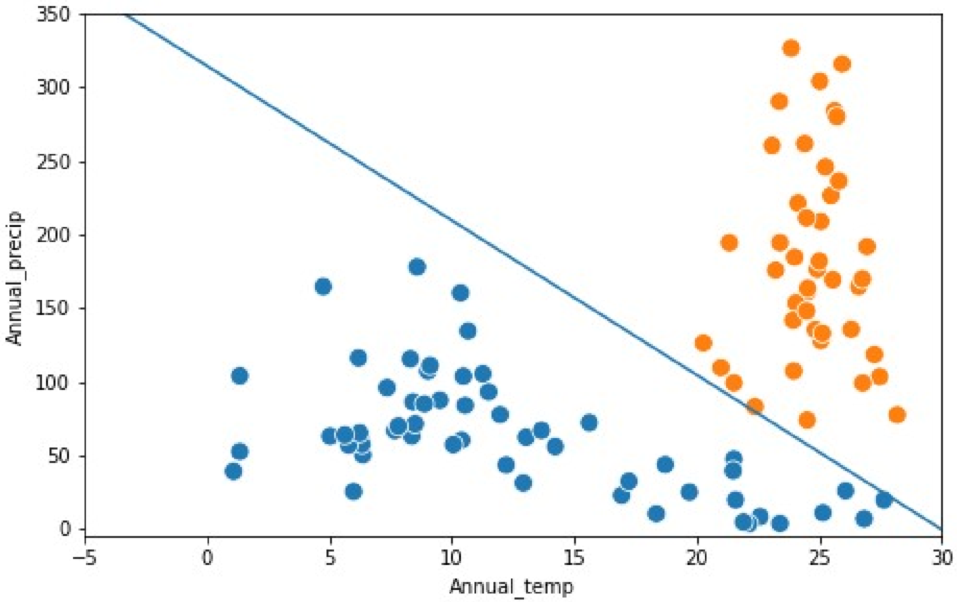
- Given a training set, with binary labels
- Model a linear classifier based on the training data
- Predict classification on new data



https://datacatalog.worldbank.org/dataset/climate-change-knowledge-portal-historical-data

Linear Separability

If the d-dimensional data is linearly separable, then there exists at least one (d-1) dimensional hyperplane that is a classifier.



https://datacatalog.worldbank.org/dataset/climate-change-knowledge-portal-historical-data

Linear Separability

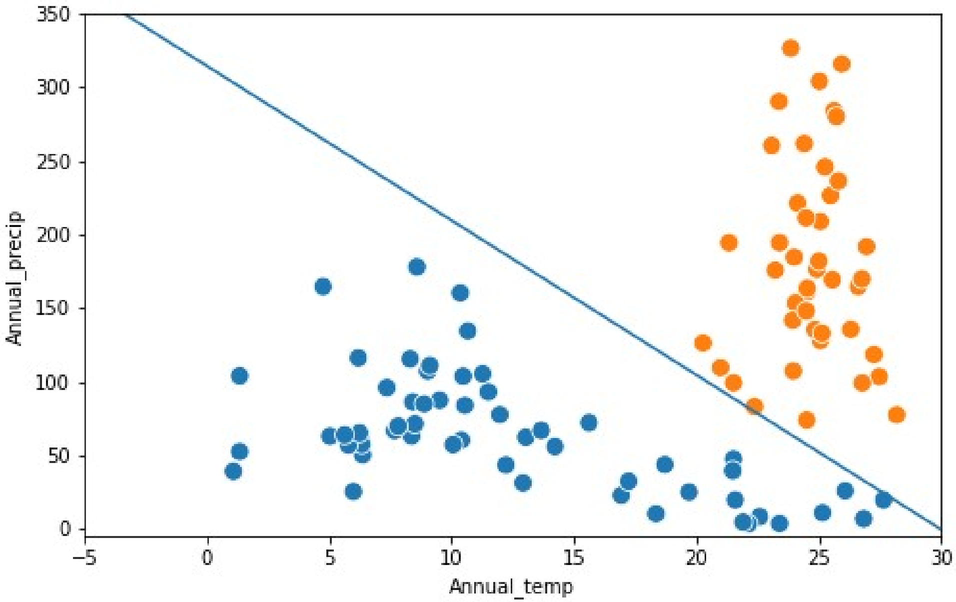
If the d-dimensional data is linearly separable, then there exists at least one (d-1) dimensional hyperplane that is a classifier.

Mathematically, the hyperplane (in this case) a line is given as:

$$w_0 + w_1 x_1 + w_2 x_2 = 0$$

Or in vector form,

$$\mathbf{w}^T \mathbf{x} = 0$$

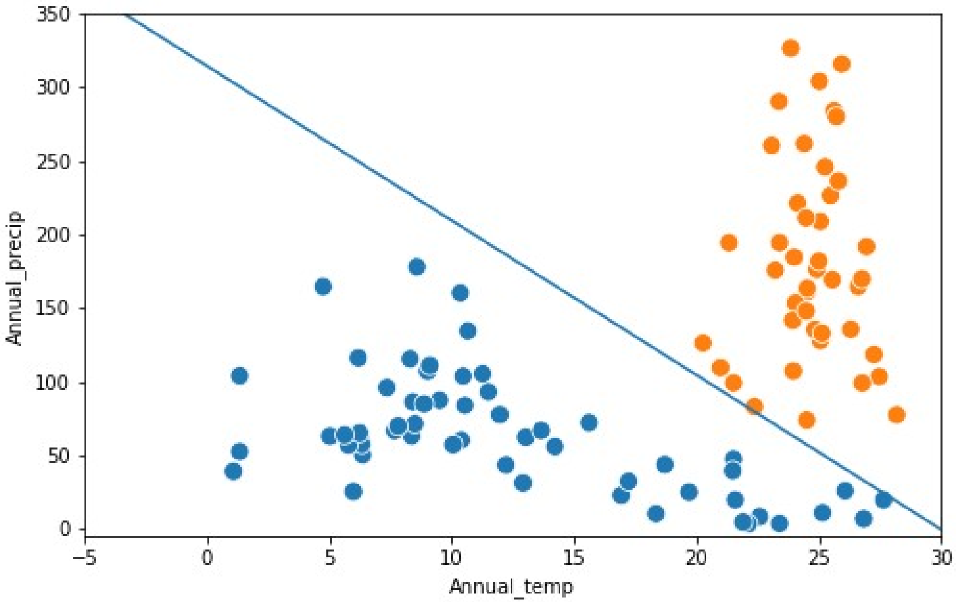


https://datacatalog.worldbank.org/dataset/climate-change-knowledge-portal-historical-data

A first ML Algorithm

Given the training data,

$$\mathbf{X} = \{\mathbf{x_i}\} : \mathbf{x_i} = [x_0, \ldots, x_d]^T$$
$$\mathbf{Y} = \{y_i\} : y_i \in \{+1, -1\}$$



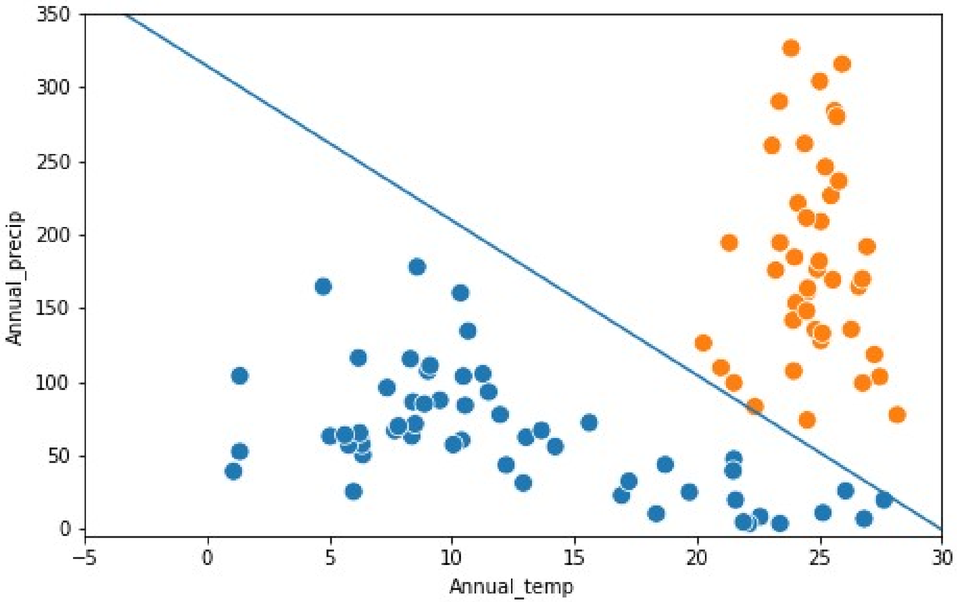
A first ML Algorithm

Given the training data,

$$\mathbf{X} = \{\mathbf{x_i}\} : \mathbf{x_i} = [x_0, \ldots, x_d]^T$$
$$\mathbf{Y} = \{y_i\} : y_i \in \{+1, -1\}$$

The model should be of the form:

$$h(\mathbf{x}) = \begin{cases} +1 & \text{if } \mathbf{w}^T\mathbf{x} \geq 0 \\ -1 & \text{if } \mathbf{w}^T\mathbf{x} < 0 \end{cases}$$



A first ML Algorithm

Given the training data,

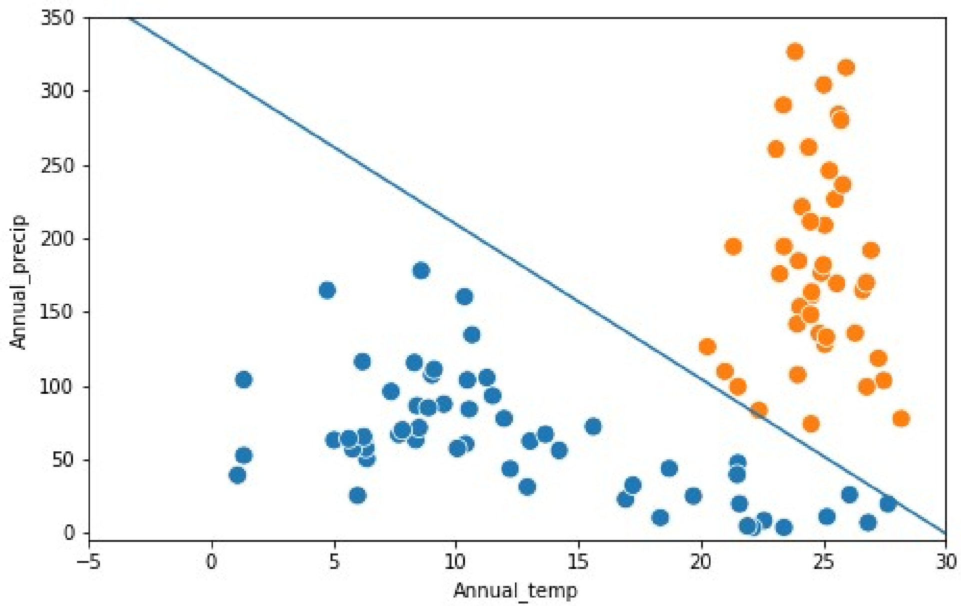
$$\mathbf{X} = \{\mathbf{x_i}\} : \mathbf{x_i} = [x_0, \ldots, x_d]^T$$
$$\mathbf{Y} = \{y_i\} : y_i \in \{+1, -1\}$$

The model should be of the form:

$$h(\mathbf{x}) = \begin{cases} +1 & \text{if } \mathbf{w}^T\mathbf{x} \geq 0 \\ -1 & \text{if } \mathbf{w}^T\mathbf{x} < 0 \end{cases}$$

Or, more compactly:

$$h(\mathbf{x}) = \text{sign}\left(\mathbf{w}^T\mathbf{x}\right)$$



A first ML Algorithm

Given the training data,

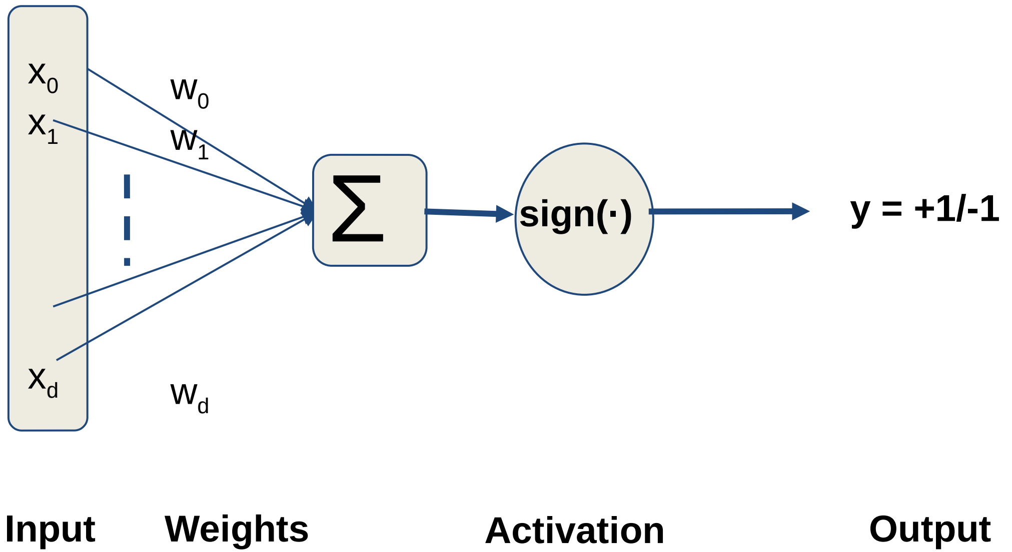
$$\mathbf{X} = \{\mathbf{x_i}\} : \mathbf{x_i} = [x_0, \ldots, x_d]^T$$
$$\mathbf{Y} = \{y_i\} : y_i \in \{+1, -1\}$$

The model should be of the form:

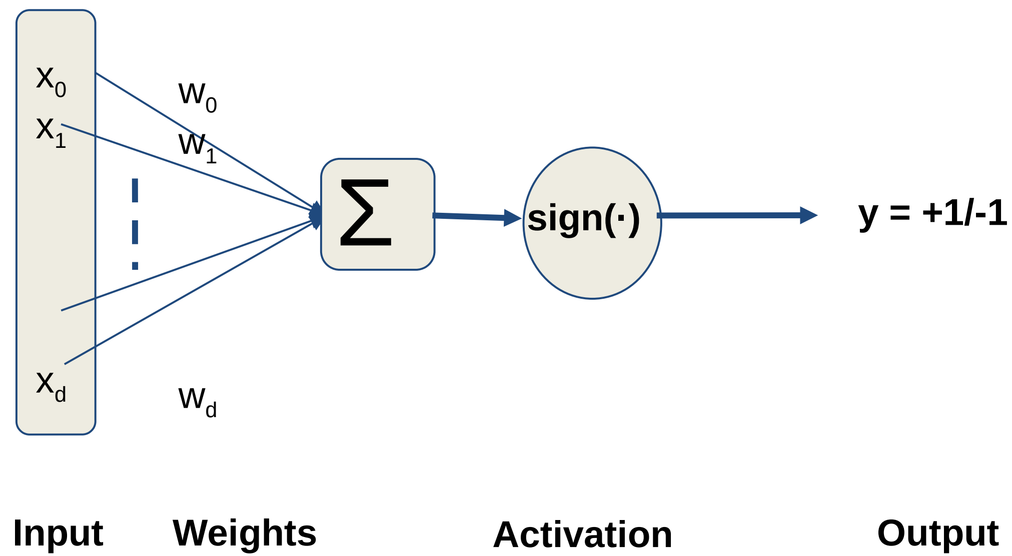
$$h(\mathbf{x}) = \begin{cases} +1 & \text{if } \mathbf{w}^T\mathbf{x} \geq 0 \\ -1 & \text{if } \mathbf{w}^T\mathbf{x} < 0 \end{cases}$$

Or, more compactly:

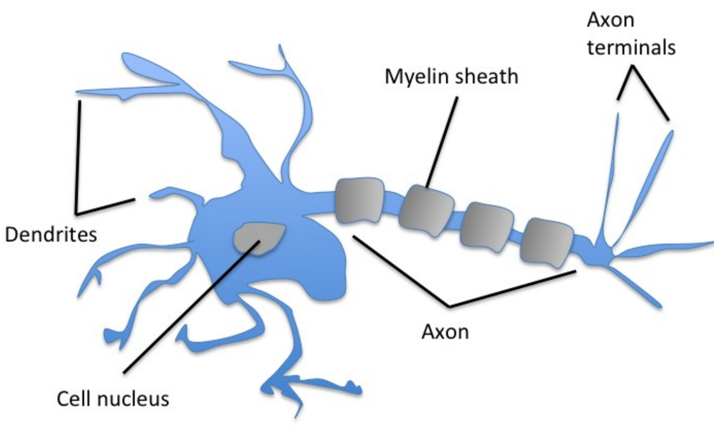
$$h(\mathbf{x}) = \operatorname{sign}\left(\mathbf{w}^T\mathbf{x}\right)$$



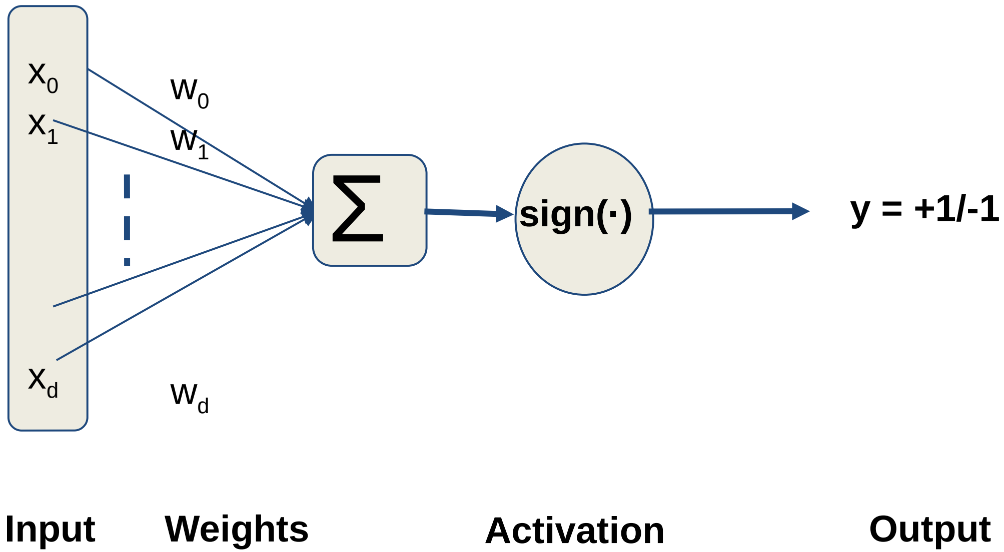
Perceptron Learning Algorithm



Perceptron Learning Algorithm



Schematic of a biological neuron.



https://sebastianraschka.com/Articles/2015_singlelayer_neurons.html

Perceptron Learning Algorithm

Algorithm: Perceptron Learning Algorithm

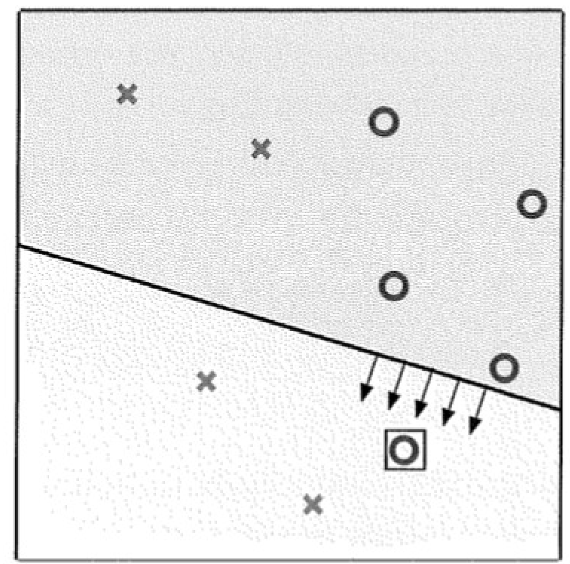
$P \leftarrow inputs \quad with \quad label \quad 1;$
$N \leftarrow inputs \quad with \quad label \quad 0;$
Initialize $\mathbf{w}$ randomly;
while $!convergence$ **do**
    Pick random $\mathbf{x} \in P \cup N$;
    if $\mathbf{x} \in P \quad and \quad \mathbf{w}.\mathbf{x} < 0$ **then**
        $\mathbf{w} = \mathbf{w} + \mathbf{x}$;
    end
    if $\mathbf{x} \in N \quad and \quad \mathbf{w}.\mathbf{x} \geq 0$ **then**
        $\mathbf{w} = \mathbf{w} - \mathbf{x}$;
    end
end
//the algorithm converges when all the inputs are classified correctly

Perceptron Learning Algorithm

Algorithm: Perceptron Learning Algorithm

$P \leftarrow$ *inputs with label* 1;
$N \leftarrow$ *inputs with label* 0;
Initialize $\mathbf{w}$ randomly;
while !*convergence* **do**
    Pick random $\mathbf{x} \in P \cup N$;
    if $\mathbf{x} \in P$ *and* $\mathbf{w}.\mathbf{x} < 0$ **then**
        $\mathbf{w} = \mathbf{w} + \mathbf{x}$;
    end
    if $\mathbf{x} \in N$ *and* $\mathbf{w}.\mathbf{x} \geq 0$ **then**
        $\mathbf{w} = \mathbf{w} - \mathbf{x}$;
    end
end
//the algorithm converges when all the inputs are classified correctly

$$\mathbf{w}(t+1) = \mathbf{w}(t) + y(t)\mathbf{x}(t).$$



PLA Task in Notebook

Summary

- Can learn from data!
- Overcomes tedious model designs
- Perceptrons mimic biological neurons

Summary

- Can learn from data!
- Overcomes tedious model designs
- Perceptrons mimic biological neurons

  However,
- Depends on the data
  - Strong assumptions (iid)
  - Distribution (no shift)
  - Number of samples
  - High quality labels
- Many (equally better/worse) models to choose from
- Hard to generalize

Exercise on Perceptron

Recipe for rest of the exercises

0. Create training & test sets

1. Instantiate the model

2. Fit the model to training set

3. Predict using trained model

Perceptron Learning Algorithm

Given the training data,

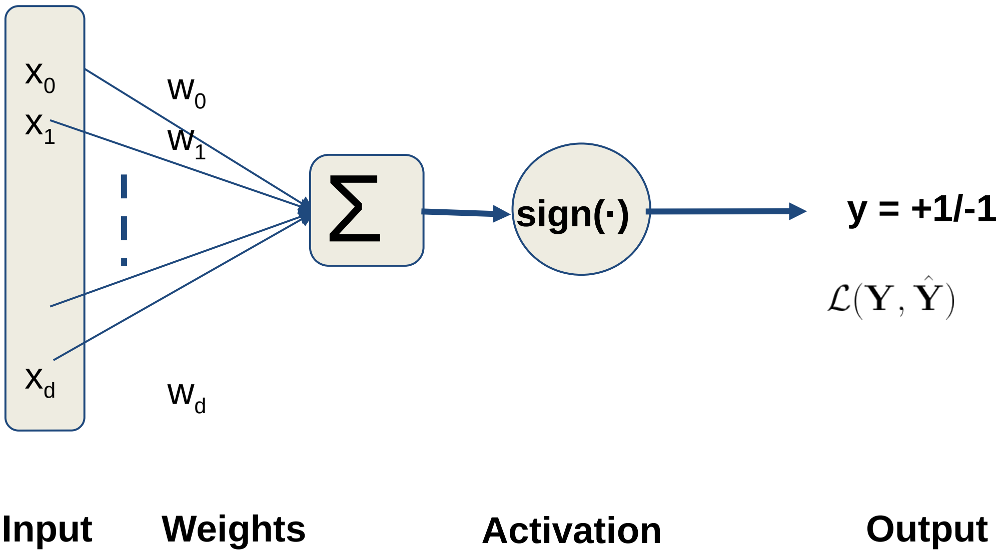
$$\mathbf{X} = \{\mathbf{x_i}\} : \mathbf{x_i} = [x_0, \ldots, x_d]^T$$
$$\mathbf{Y} = \{y_i\} : y_i \in \{+1, -1\}$$

The model should be of the form:

$$h(\mathbf{x}) = \begin{cases} +1 & \text{if } \mathbf{w}^T\mathbf{x} \geq 0 \\ -1 & \text{if } \mathbf{w}^T\mathbf{x} < 0 \end{cases}$$

Or, more compactly:

$$h(\mathbf{x}) = \mathrm{sign}\left(\mathbf{w}^T\mathbf{x}\right)$$



PLA to Linear regression

Linear Regression

$\mathbf{X} \in \mathbb{R}^{N \times d}$
$\mathbf{Y} \in \mathbb{R}^{N \times 1}$
Then, we are interested in a function

$$h(\cdot) : \mathbf{X} \to \mathbf{Y}$$

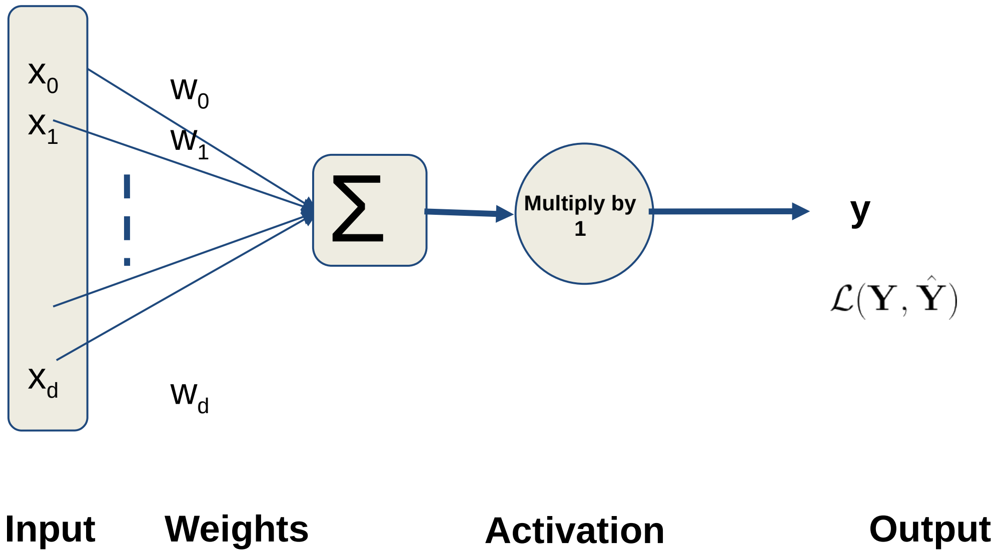
$$\hat{y} \triangleq h(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$$

Linear Regression

$\mathbf{X} \in \mathbb{R}^{N \times d}$
$\mathbf{Y} \in \mathbb{R}^{N \times 1}$
Then, we are interested in a function

$$h(\cdot) : \mathbf{X} \rightarrow \mathbf{Y}$$

$$\hat{y} \triangleq h(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$$



Analytical solution obtained by minimizing mean squared error loss $\mathcal{L}(\mathbf{Y}, \hat{\mathbf{Y}})$

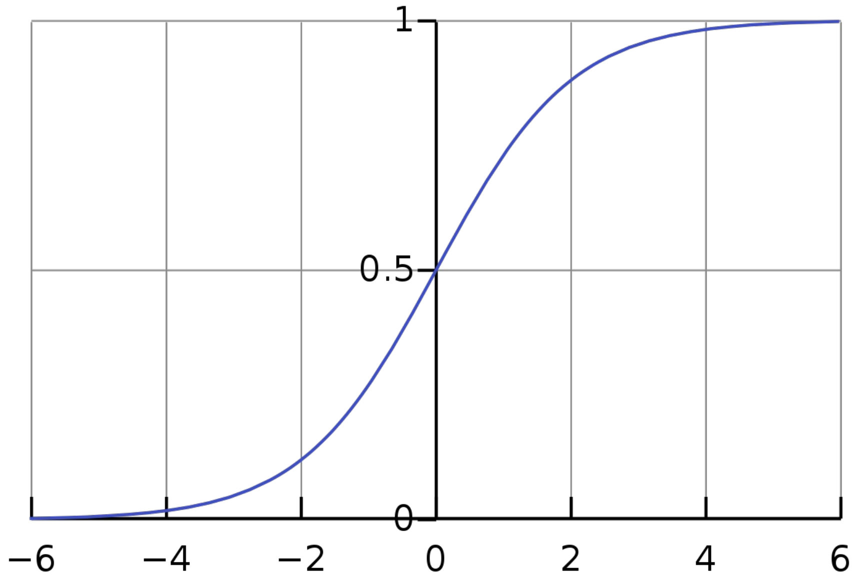
$$\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}$$

PLA to Logistic regression

Sigmoid function

$$\sigma(\cdot) : x \in \mathbb{R} \rightarrow (0, 1)$$

$$\sigma(x) = \frac{1}{(1 + \exp^{-x})}$$



Logistic Regression

$\mathbf{X} \in \mathbb{R}^{N \times d}$
$\mathbf{Y} \in (0, 1)^{N \times 1}$
Then, we are interested in a function

$$h(\cdot) : \mathbf{X} \to \mathbf{Y}$$

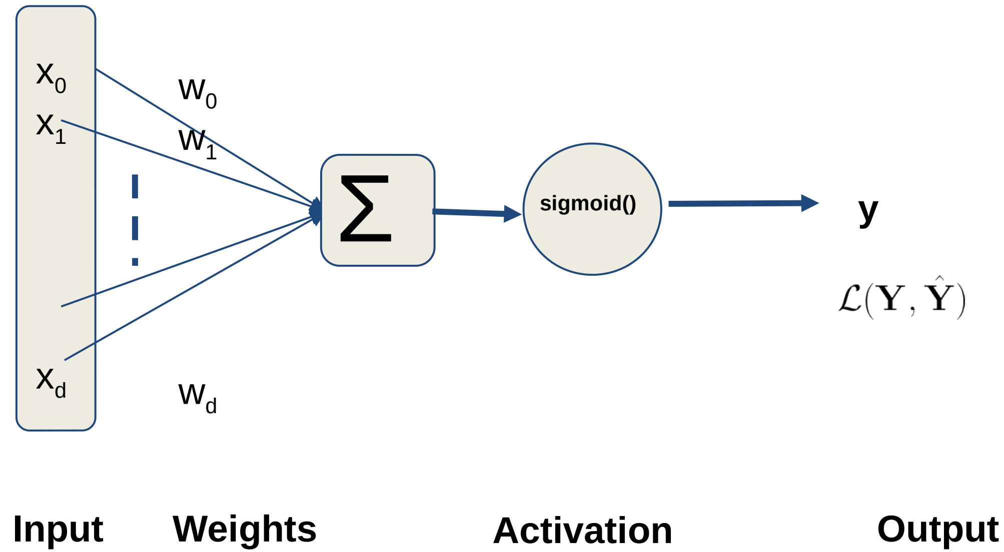
$$\hat{y} \triangleq h(\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x})$$

Logistic Regression

$\mathbf{X} \in \mathbb{R}^{N \times d}$
$\mathbf{Y} \in (0,1)^{N \times 1}$
Then, we are interested in a function

$$h(\cdot) : \mathbf{X} \to \mathbf{Y}$$

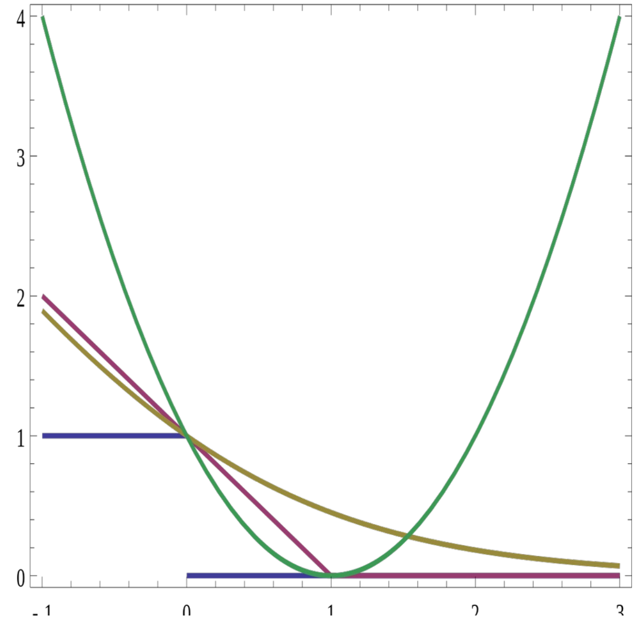
$$\hat{y} \triangleq h(\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x})$$



Gradient based update for Logistic Regression

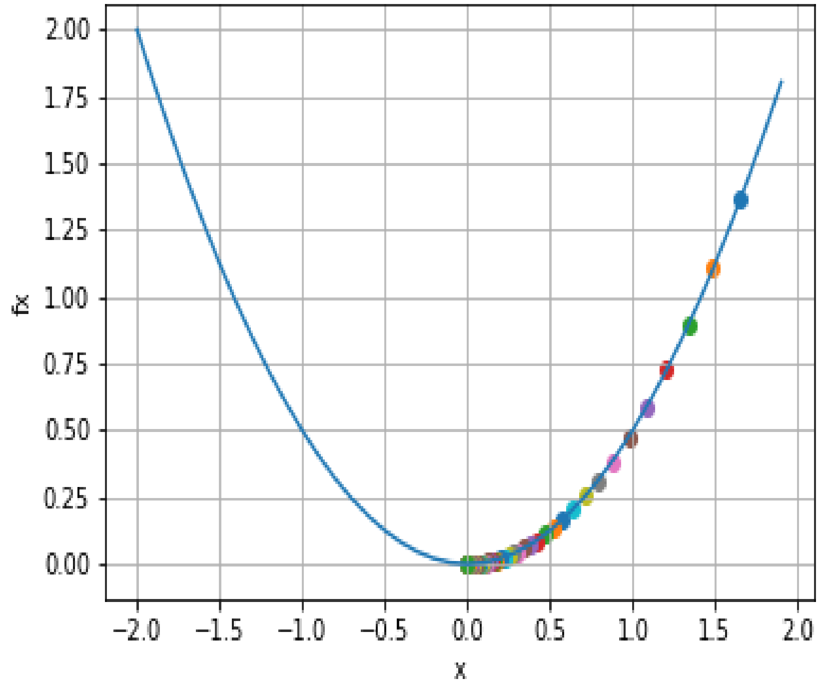
Desired forms of loss functions

- Smooth
- Convex
- Analytical gradients
- If not convex, with feasible set of local minima



Gradient descent

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \gamma \Delta f(\mathbf{x}_t)$$



Summary

- Perceptron with “multiply by 1” activation -> Linear Regression
- Perceptron with “sigmoid” activation -> Logistic Regression
- Analytical solutions seldom exist
- Gradient descent can be used when gradients can be computed
- What if analytical gradients cannot be computed?

Home Exercise on Gradient Descent

Session 3

DL: Massively parameterised function approximator

DL: Massively parameterised function approximator

$$\mathbf{X} \in \mathbb{R}^{N \times D},$$
$$\mathbf{Y} \in \mathbb{R}^{N \times D}, \mathbb{R}^{N}, (0,1)^{N}$$
$$f(\cdot; \mathbf{W}) : \mathbf{X} \to \mathbf{Y}$$

where, the number of parameters are approximately:

$$|\mathbf{W}| \geq \mathcal{O}(N \times D)$$

DL: Massively parameterised function approximator

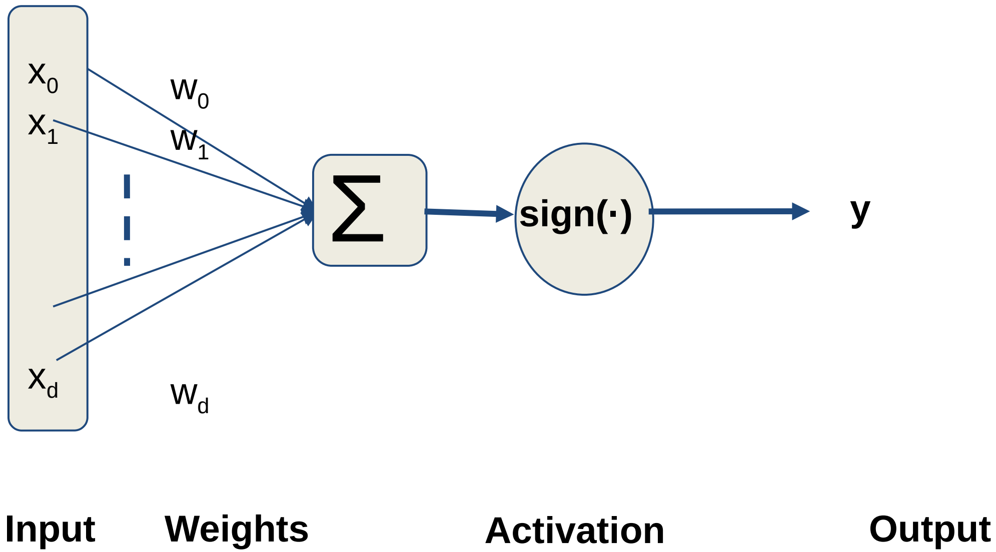
$$\mathbf{X} \in \mathbb{R}^{N \times D},$$
$$\mathbf{Y} \in \mathbb{R}^{N \times D}, \mathbb{R}^{N}, (0,1)^{N}$$
$$f(\cdot;\mathbf{W}) : \mathbf{X} \rightarrow \mathbf{Y}$$

where, the number of parameters are approximately:

$$|\mathbf{W}| \geq \mathcal{O}(N \times D)$$

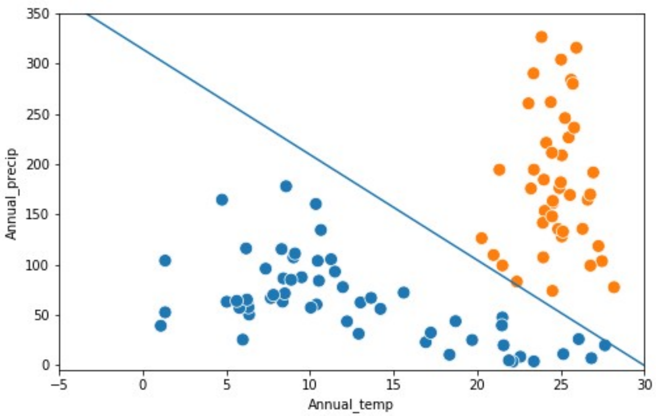
- **X**: Images, videos, time series, tabular, text, density functions, graphs, etc…
- **Y**: Segmentations, alignments, regression, translation, classification, synthesis

Good old PLA again!

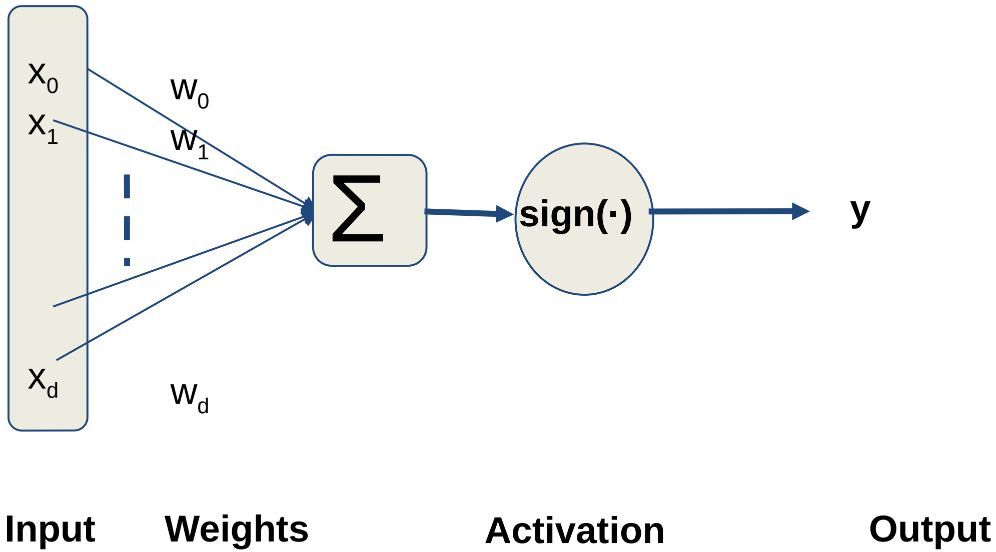


Perceptron Learning Algorithm

$$f(\cdot;\mathbf{W}):\mathbf{X}\rightarrow\mathbf{Y}$$



Good old PLA again!

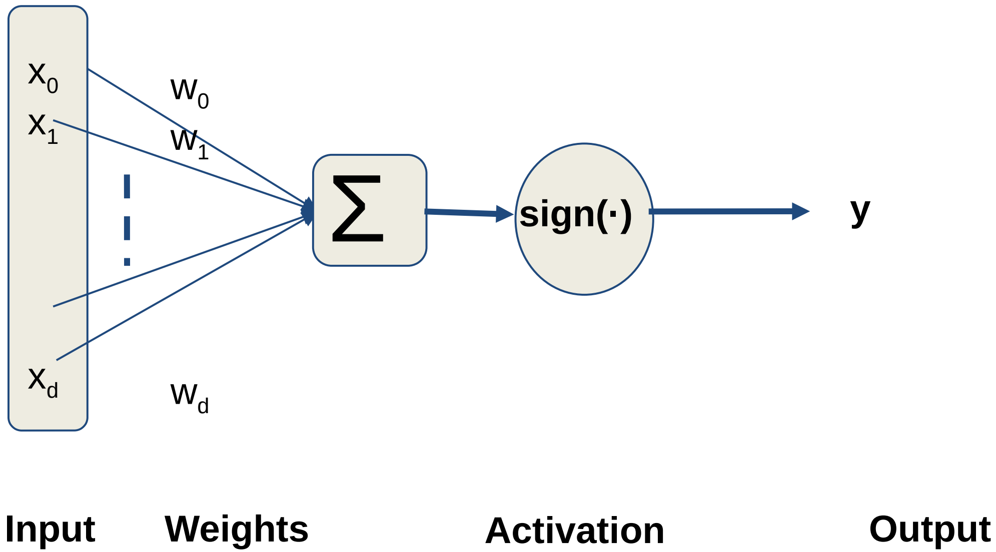


$$f(\cdot; \mathbf{W}) : \mathbf{X} \to \mathbf{Y}$$

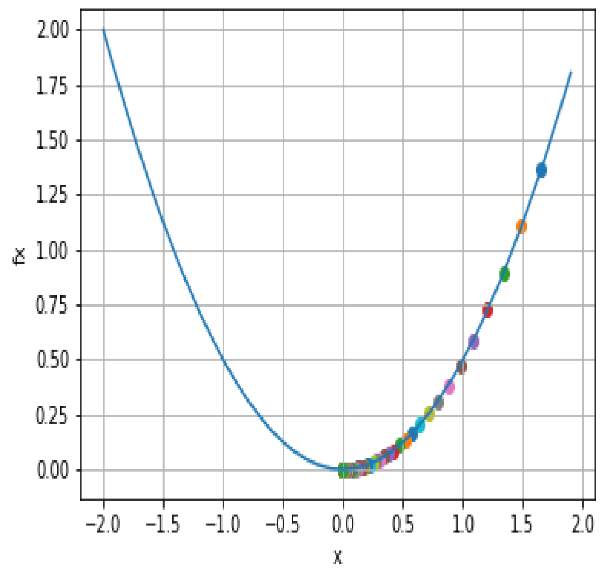
- How many parameters?
- How do we obtain **W***?
- What are the challenges?

Perceptron Learning Algorithm

Good old PLA again!



Perceptron Learning Algorithm



$$\mathbf{x}_{t+1} = \mathbf{x}_t - \gamma \Delta f(\mathbf{x}_t)$$

Gradient based optimization for different scenarios

Solve for the analytical model

Linear Regression

Gradient based optimization for different scenarios

Solve for the analytical model

Analytical gradients for iterative gradient descent

Linear Regression

Logistic Regression

Gradient based optimization for different scenarios

Solve for the analytical model

Linear Regression

Analytical gradients for iterative gradient descent

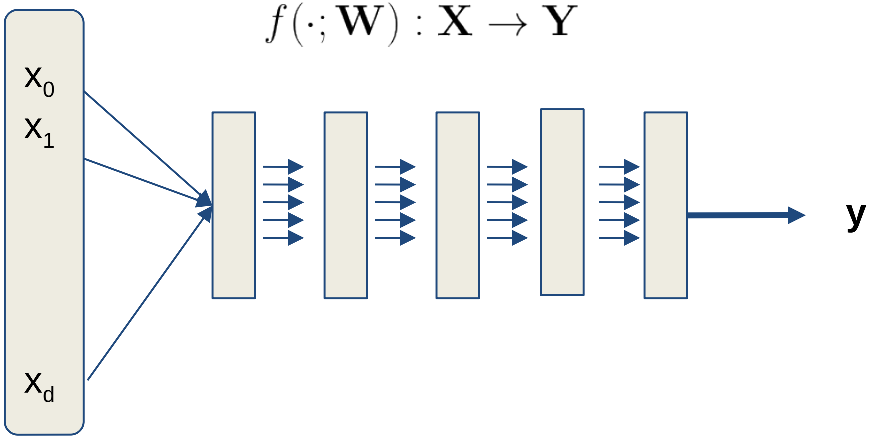
Logistic Regression

Automatic Differentiation

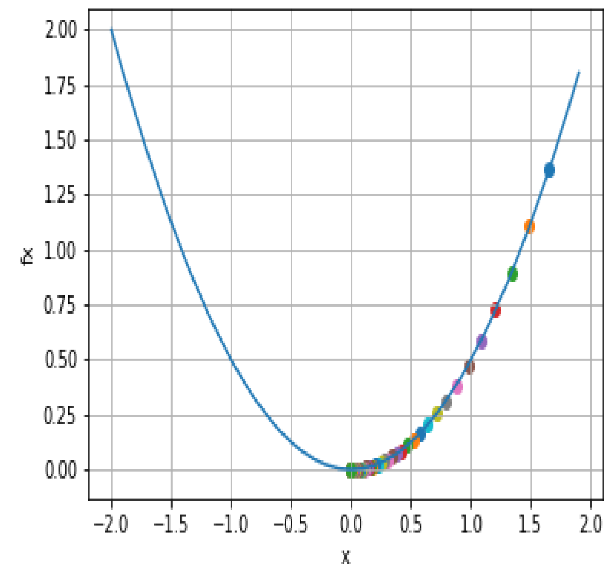
Almost everything else.
Including DL

Increasing tunable parameters gives more flexibility

$$f(\cdot;\mathbf{W}) : \mathbf{X} \rightarrow \mathbf{Y}$$



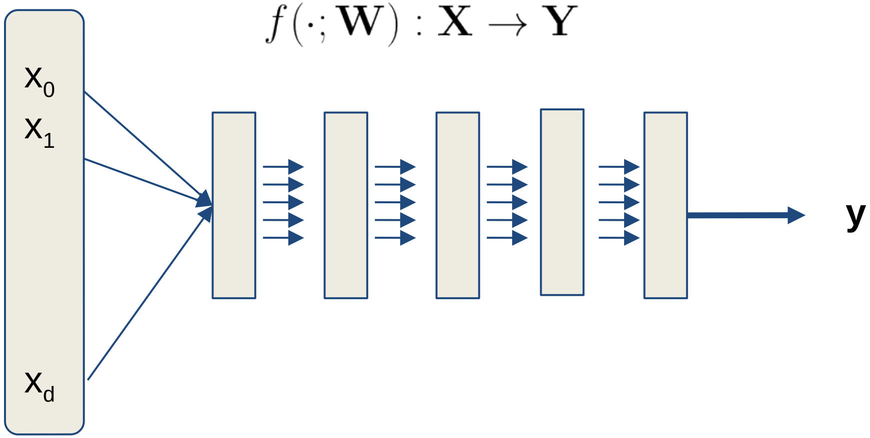
Input **Weights** **Output**



Deep Learning Models

Increasing tunable parameters gives more flexibility

$$f(\cdot;\mathbf{W}) : \mathbf{X} \rightarrow \mathbf{Y}$$

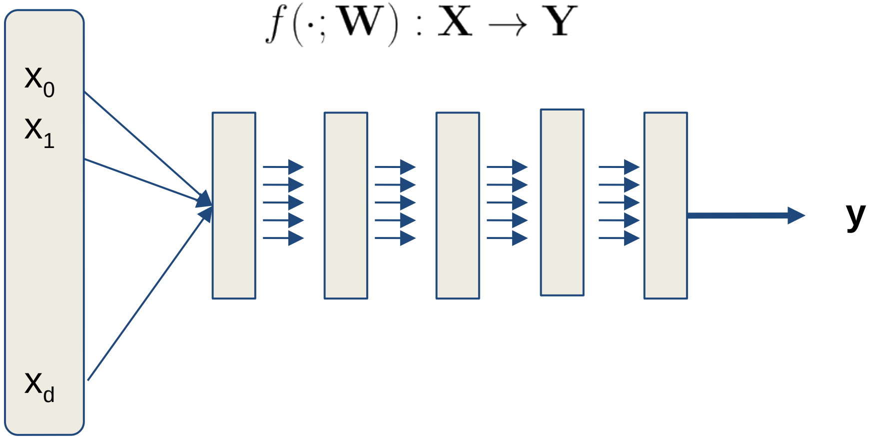


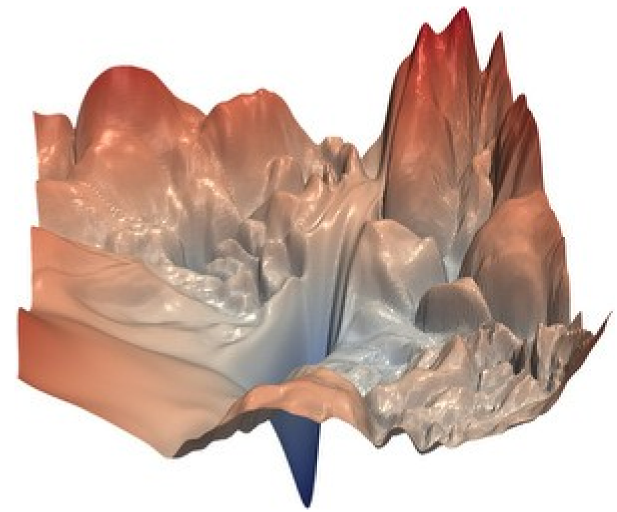
Input **Weights** **Output**

Deep Learning Models

Increasing tunable parameters gives more flexibility, but...

$$f(\cdot; \mathbf{W}) : \mathbf{X} \to \mathbf{Y}$$



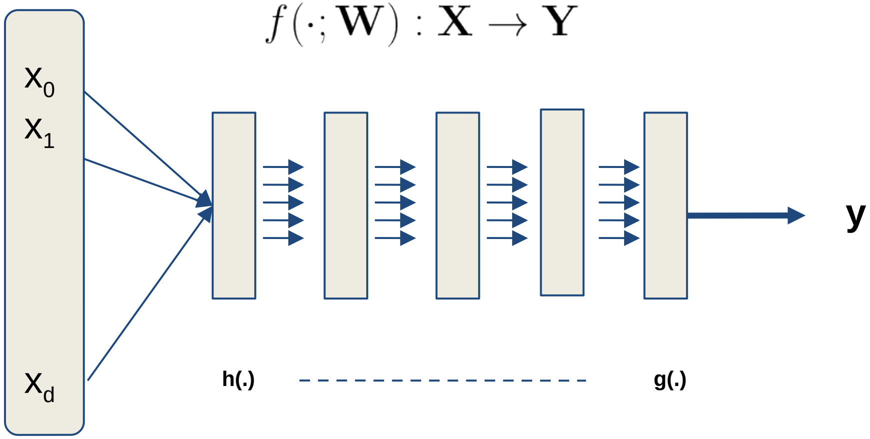


Input **Weights** **Output**

Deep Learning Models

Automatic differentiation to navigate such loss-scapes

$$f(\cdot;\mathbf{W}) : \mathbf{X} \rightarrow \mathbf{Y}$$



$$y = f(g(h(x))) = f(g(w_1)) = f(w_2)$$

Then, using chain rule

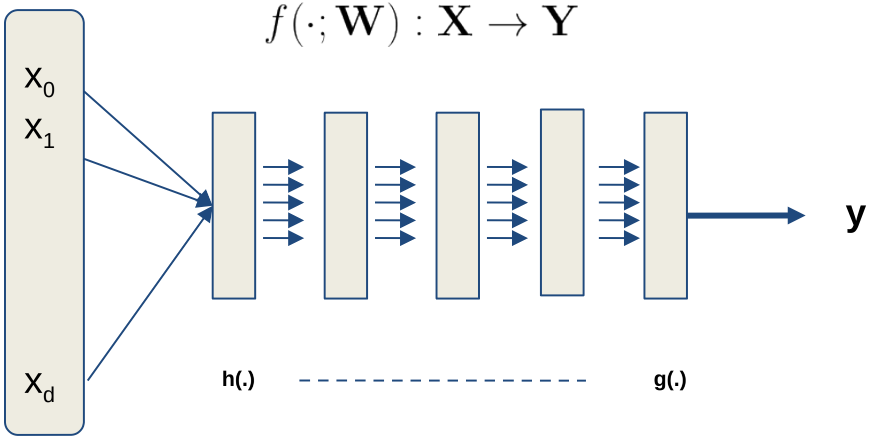
$$\frac{dy}{dx} = \frac{dy}{dw_2}\frac{dw_2}{dw_1}\frac{dw_1}{dx}$$

Input **Weights** **Output**

Deep Learning Models

Automatic differentiation to navigate such loss-scapes

$$f(\cdot;\mathbf{W}) : \mathbf{X} \rightarrow \mathbf{Y}$$



Input **Weights** **Output**

Deep Learning Models

$$y = f(g(h(x))) = f(g(w_1)) = f(w_2)$$

Then, using chain rule

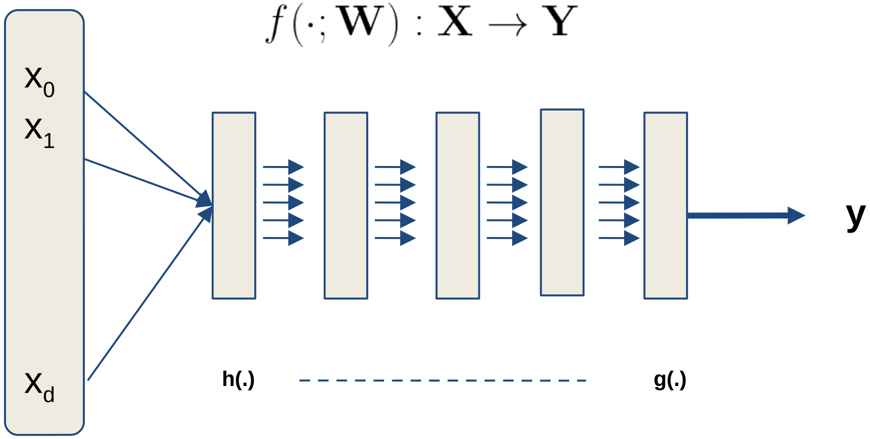
$$\frac{dy}{dx} = \frac{dy}{dw_2}\frac{dw_2}{dw_1}\frac{dw_1}{dx}$$

AD in ML is Backpropagation!

1. Forward accumulation (wrt input)
2. Reverse accumulation (wrt loss)

Automatic differentiation to navigate such loss-scapes

$$f(\cdot;\mathbf{W}) : \mathbf{X} \to \mathbf{Y}$$



Input **Weights** **Output**

Deep Learning Models

$$y = f(g(h(x))) = f(g(w_1)) = f(w_2)$$

Then, using chain rule

$$\frac{dy}{dx} = \frac{dy}{dw_2}\frac{dw_2}{dw_1}\frac{dw_1}{dx}$$

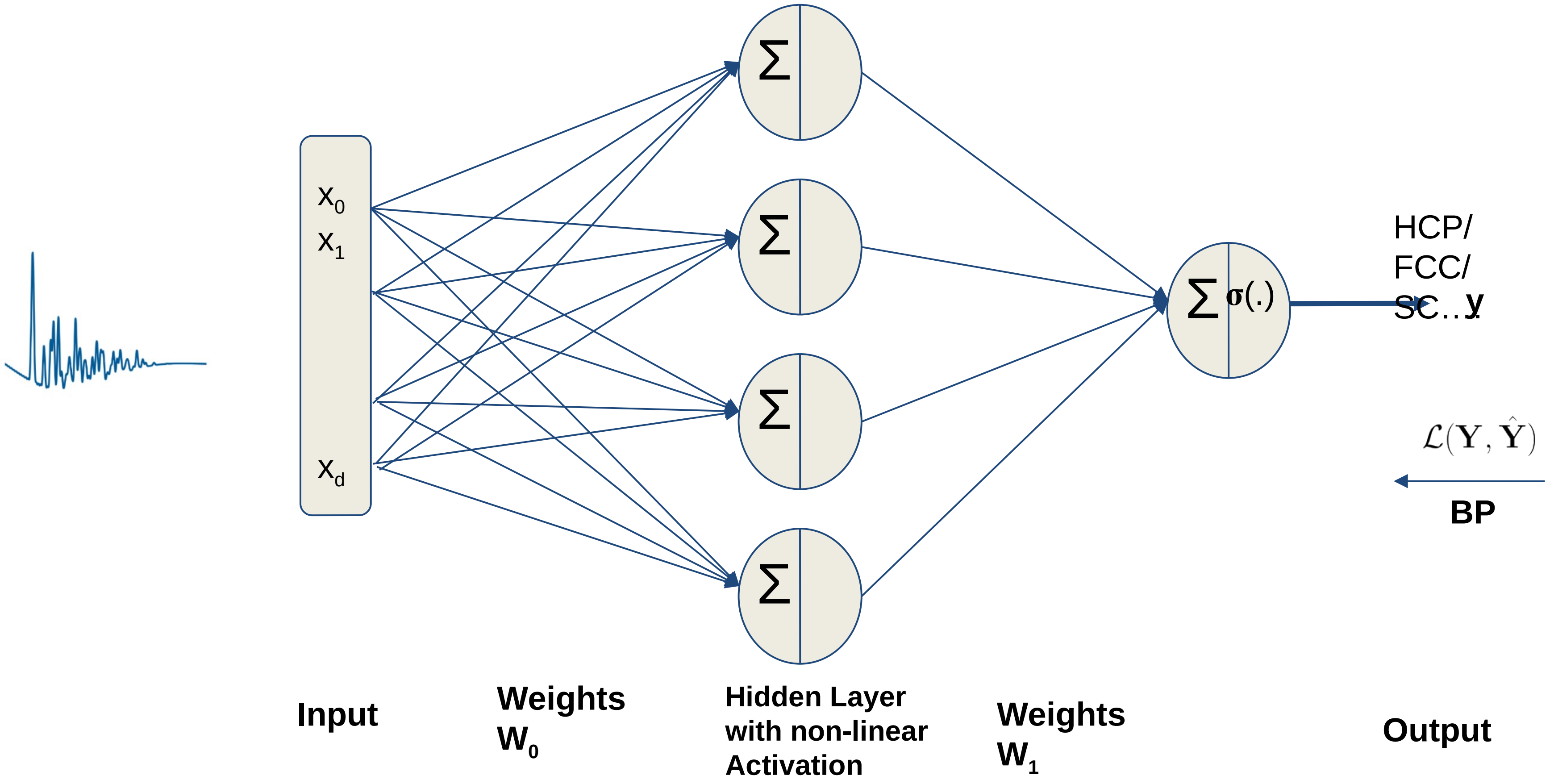
AD in ML is Backpropagation!

1. Forward accumulation (wrt input)
2. Reverse accumulation (wrt loss)

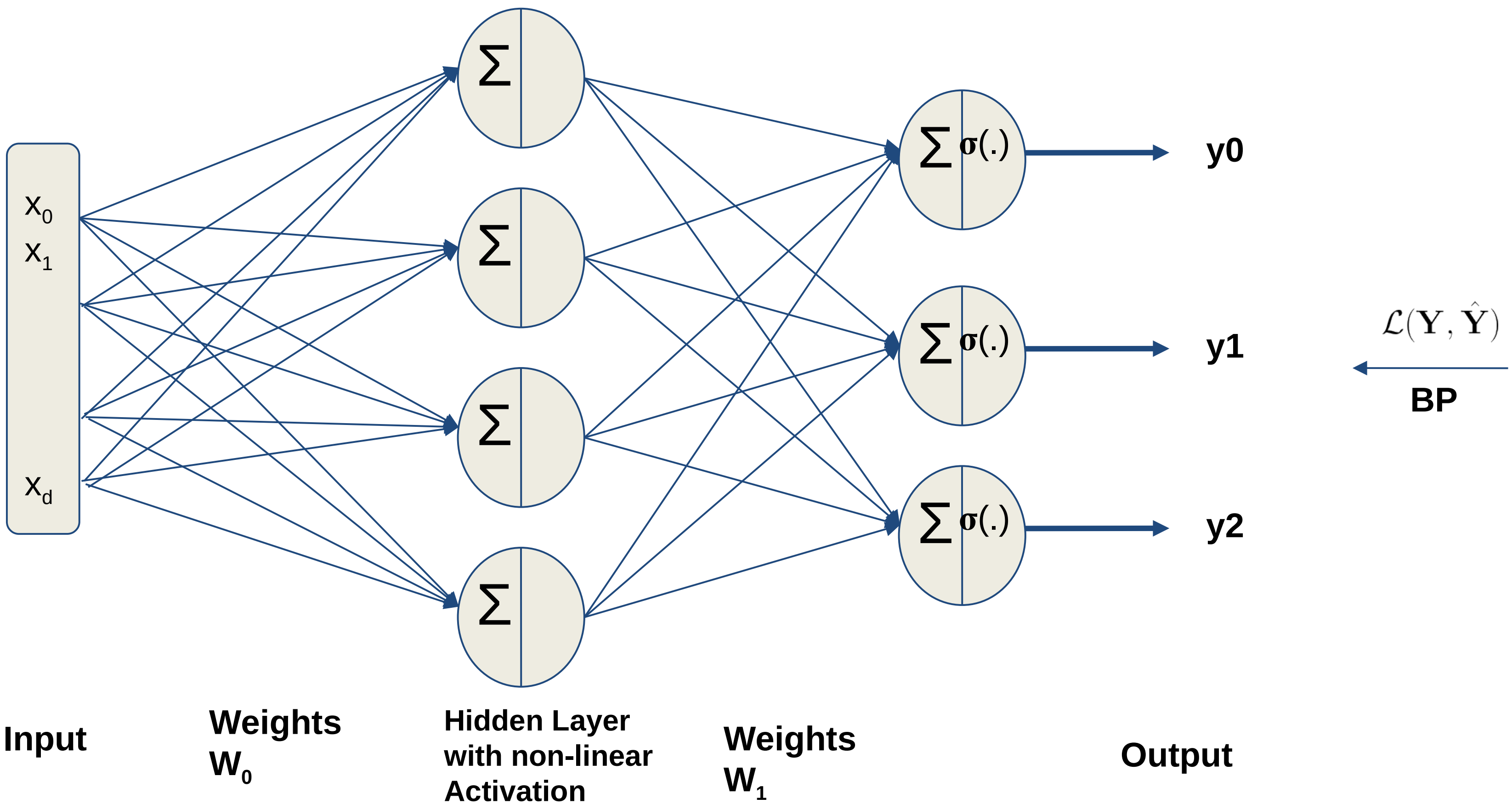
And, don't worry. It is by now efficiently implemented in several packages!

First DL model: Multi Layer Perceptron (MLP)

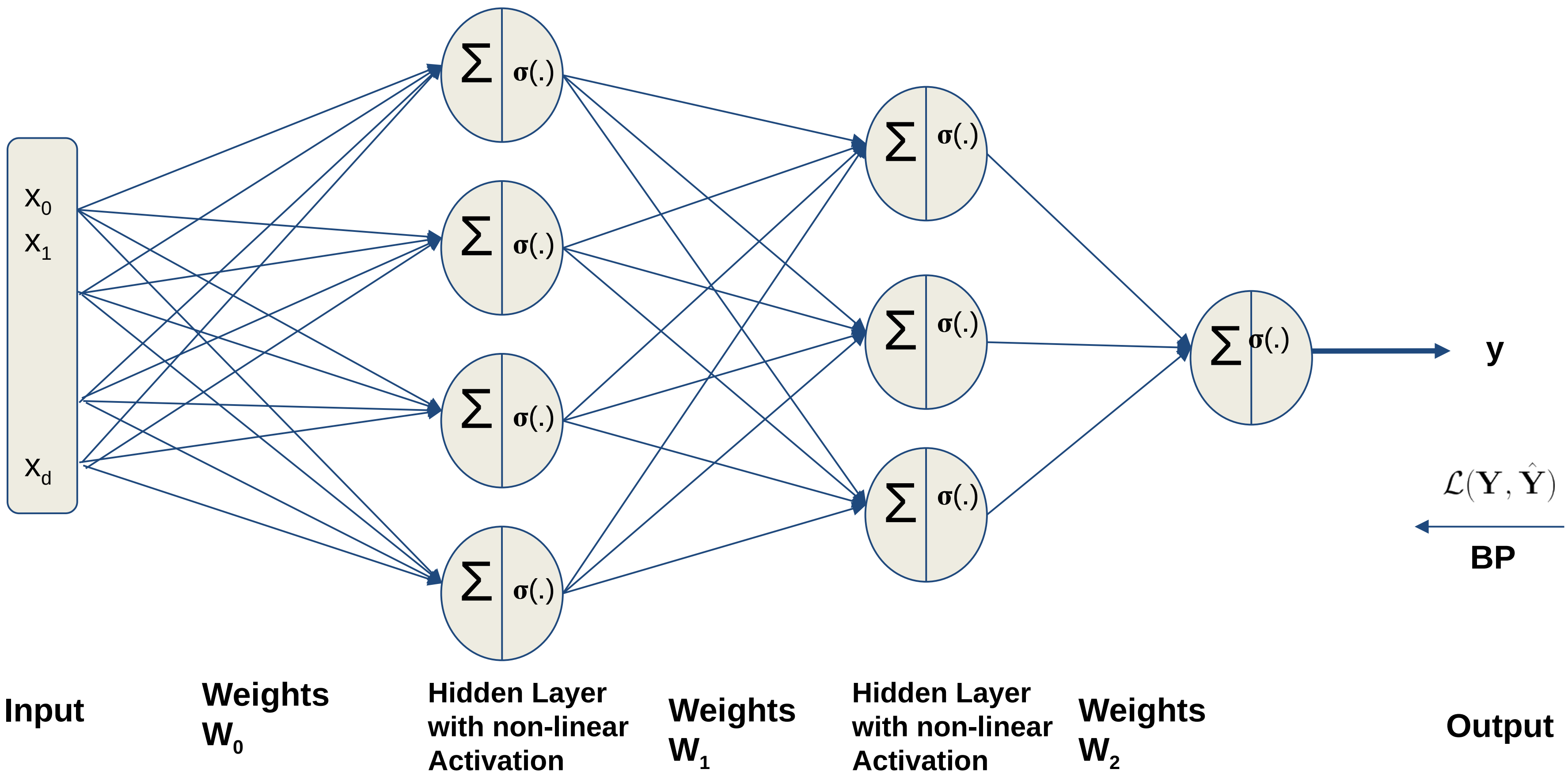
First DL model: Multi Layer Perceptron (MLP)



MLP with multiple outputs



Multi Layered Perceptron with multiple outputs



MLPs everywhere!

- Highly flexible components
- Can approximate highly non-linear functions
- Classification/Regression/Segmentation
- Non-linearities are critical
- “Small” compared to other DL models
- Deeper or Wider?
- No obvious way to decide architectures

Summary

- Design based methods
- Learning from data is possible*
- Some form of experience must be given to the ML models
- Perceptron as the fundamental unit
- MLPs already can approximate complex functions
- Automatic differentiation is handy!
- CNNs can learn complex filters
- CNNs can harvest information from different scales
- Occam's Razor

Exercise on MLPs