About Judith

Born in Aachen, University in Heidelberg

Postdoc in USA (University at Chicago and MIT in Boston)

➢ ROOT user since 1998

Research at DESY, one year at CERN, teaching in Berlin

Experiments at colliders:

PSI (Switzerland), HERA (DESY), RHIC (USA), LHC (CERN)

➢ ML and top-Higgs coupling

Hobbies:

Mountaineering
Swimming
Running
Pilates
Music
Reading novels

Introduction to Machine Learning

Part I

Judith Katzy
Hamburg, September 2024

HELMHOLTZ

Outline

- The big picture
  - Extracting physics knowledge with machine learning
  - Learning frameworks and its ingredients
- The key elements
  - Data sets
  - Hypothesis sets
  - Optimisation
- Example: Neural networks
  - Building functions with perceptrons
  - Universal approximation theorem

Material

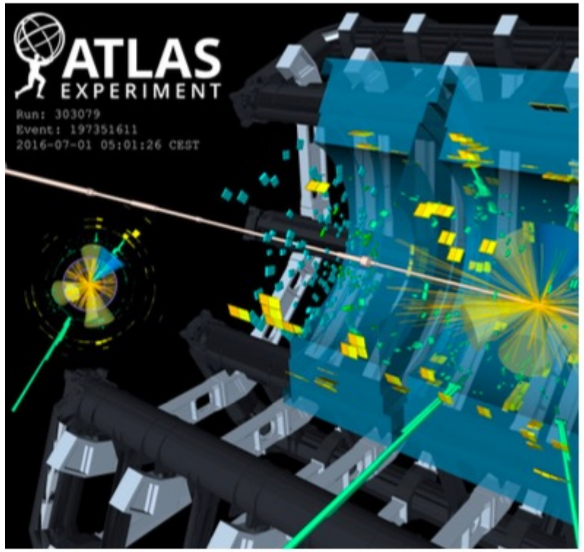
- book: Understanding deep learning from Simon Price
- Deeplearning.org book from Ian Goodfellow
- Pictures from Lukas Heinrich
- Kyle Cranmer, ML Review

ML is NOT a spectator sport – important material in exercises from Peter Steinbach

Why is machine learning relevant for particle physics?

Fundamentals of particle physics analysis

measurement



100 Mio electronic channels

Quantum mechanical nature of physics process

-> Probabilistic distributed events $p(x|\theta)$

Rely on a statistical model p to extract parameters θ from data x:

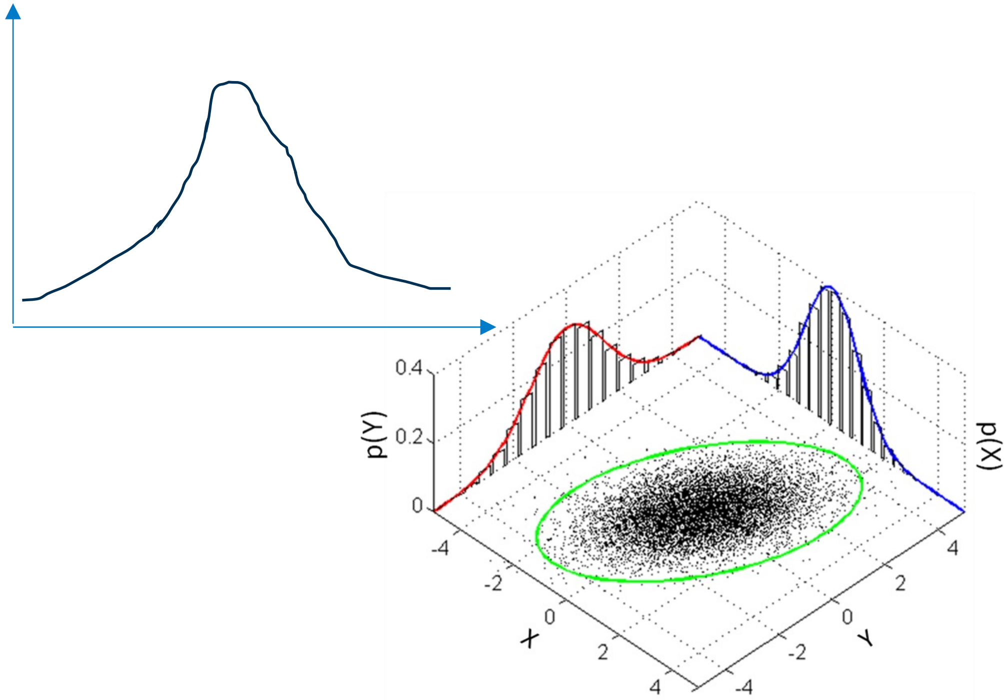
We have high dimensional data

We have large data sets

$$\mathcal{L} = -\frac{1}{4} F_{\mu\nu} F^{\mu\nu} + i\bar{\psi} \not{D} \psi + h.c. + \bar{\psi}_i y_{ij} \psi_j \phi + h.c. + |D_\mu \phi|^2 - V(\phi)$$

Few parameters

Curse of dimensionality



1 dim: Sample N events to describe distribution
2 dim: sample N^2 events to describe distribution

.

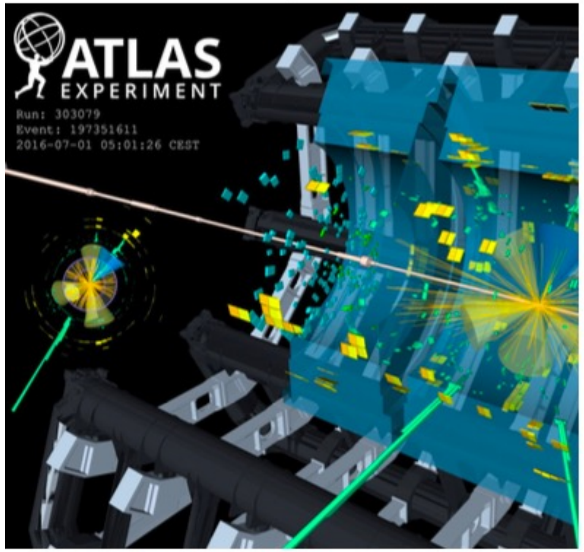
.

.

d dim: sample $\mathcal{O}(N^d)$ events to describe distribution

-> Needs impractical computational resources

Fundamentals of particle physics analysis



100 Mio electronic channels

Quantum mechanical nature of physics process

-> Probabilistic distributed events $p(x|\theta)$

Rely on a statistical model p to extract parameters θ from data x:

We have high dimensional data

We have large data sets

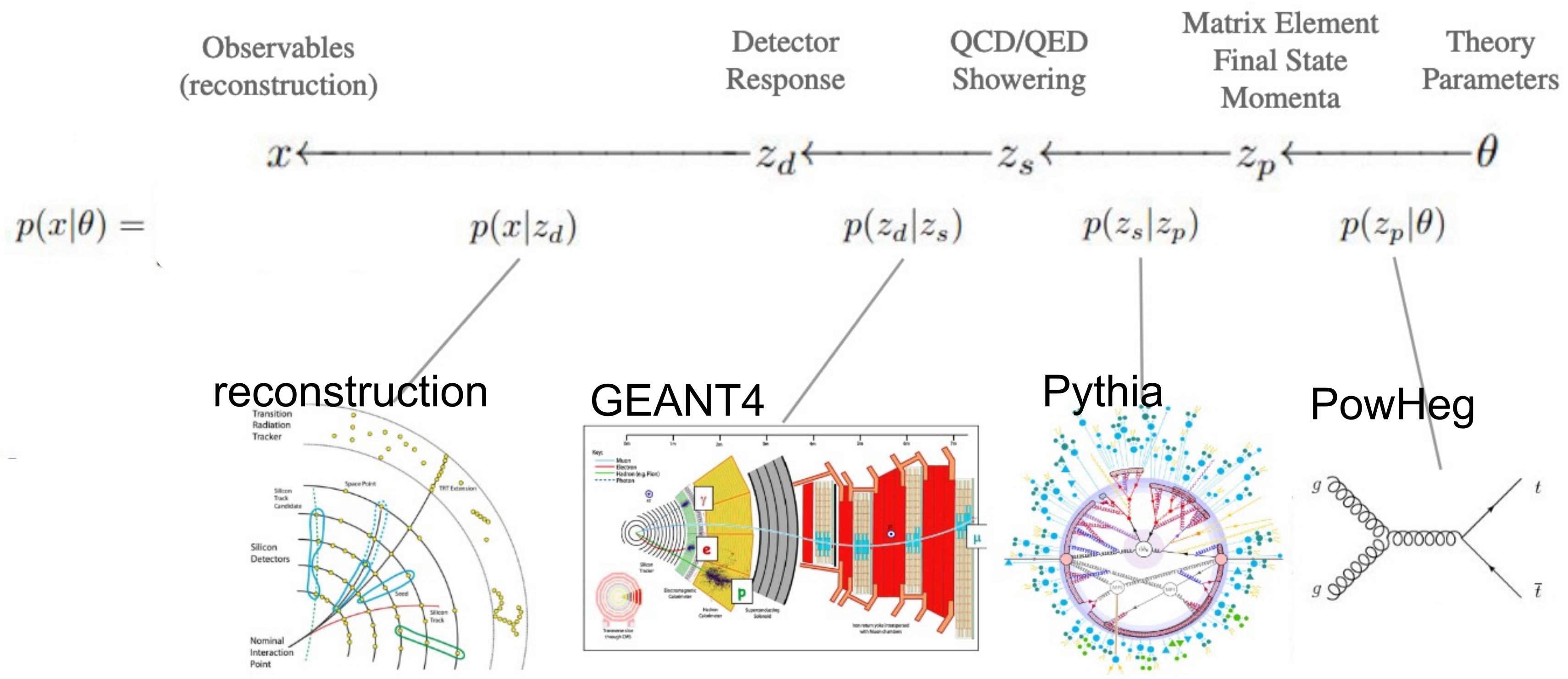
$$\mathcal{L} = -\frac{1}{4} F_{\mu\nu} F^{\mu\nu} + i \bar{\psi} \not{D} \psi + h.c. + \bar{\psi}_i y_{ij} \psi_j \phi + h.c. + |D_\mu \phi|^2 - V(\phi)$$

Few parameters

simulation

The role of simulators

simulators capture the relevant physics on a hierarchy of scales



Data $\{xi\}_{i=1}^{N}$ N samples independently and identically distributed from p(x| θ) with simulator settings θ

➔ Approximate $p(x|\theta) = \int p(x, z|\theta)\, dz$

➔ fixed value of z specifies everything about the simulated event: z = ground truth “label”

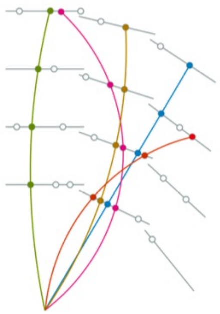
➔ Reconstruction algorithms estimate components from z

➔ data set $\{x_i, z_i\}^{N}_{i=1}$ to study reco algorithms

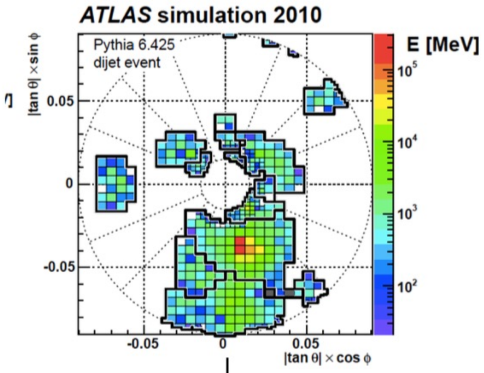
Data representation

Goal: bring the data into a form that is easier to understand and interpret

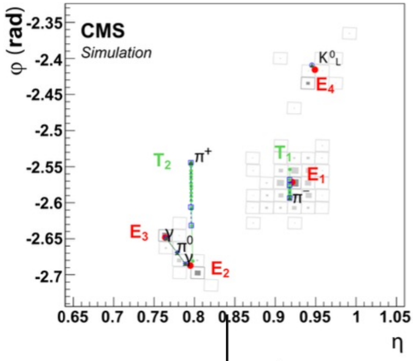
Hits → Tracks



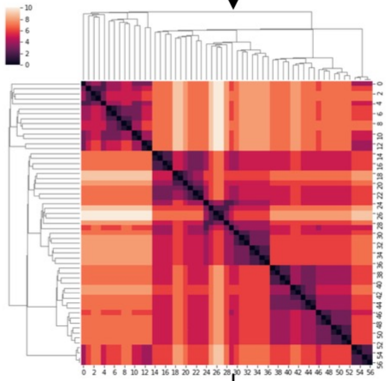
Energy Cells → Energy Clusters



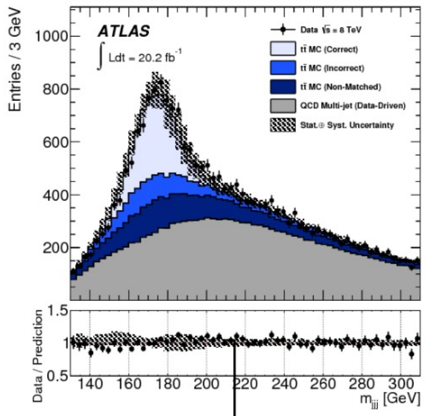
E Clusters & Tracks → Particles



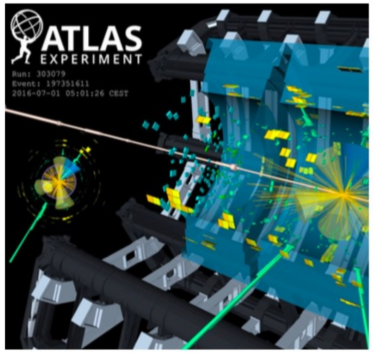
E Clusters → Jets (~ parton)



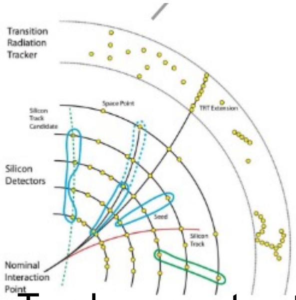
Particles → Resonances



Reducing dimensionality

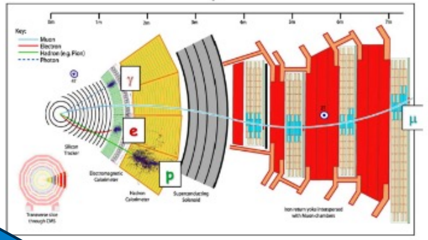


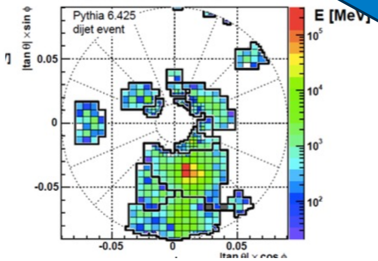
100 Mio electronic channels



Track reconstruction

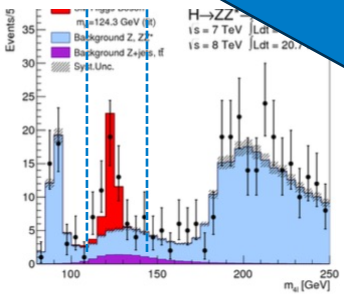
Particle identification





energy reconstruction

$N_{electron} > 4$



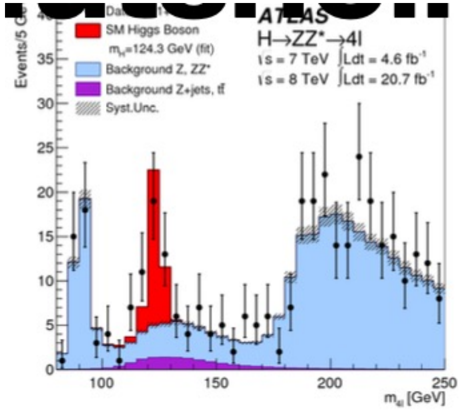
$$\mathcal{L} = -\frac{1}{4} F_{\mu\nu} F^{\mu\nu} + i \bar{\psi} \not{D} \psi + h.c. + \bar{\psi}_i y_{ij} \psi_j \phi + h.c. + |D_\mu \phi|^2 - V(\phi)$$

m_{Higgs} = 125 GeV

Summary

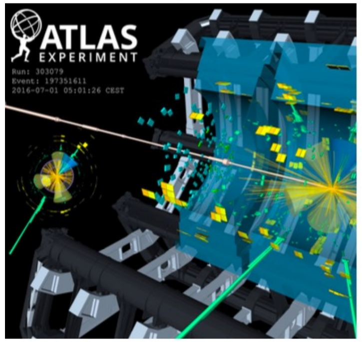
$$\mathcal{L} = -\frac{1}{4} F_{\mu\nu} F^{\mu\nu} + i \bar{\psi} \not{D} \psi + h.c. + \bar{\psi}_i y_{ij} \psi_j \phi + h.c. + |D_\mu \phi|^2 - V(\phi)$$

High level concepts

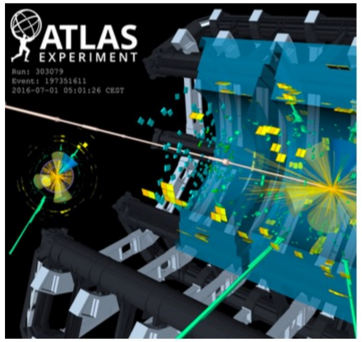


generate low-level, high-dim data from high-level concepts

reconstruct high level concepts from low-level, high-dim data



Low level data



ML excels at both!

street style photo of a woman selling pho at a Vietnamese street market, sunset, shot on fujifilm

generate low-level, high-dim data from high-level concepts

High-Level Concept

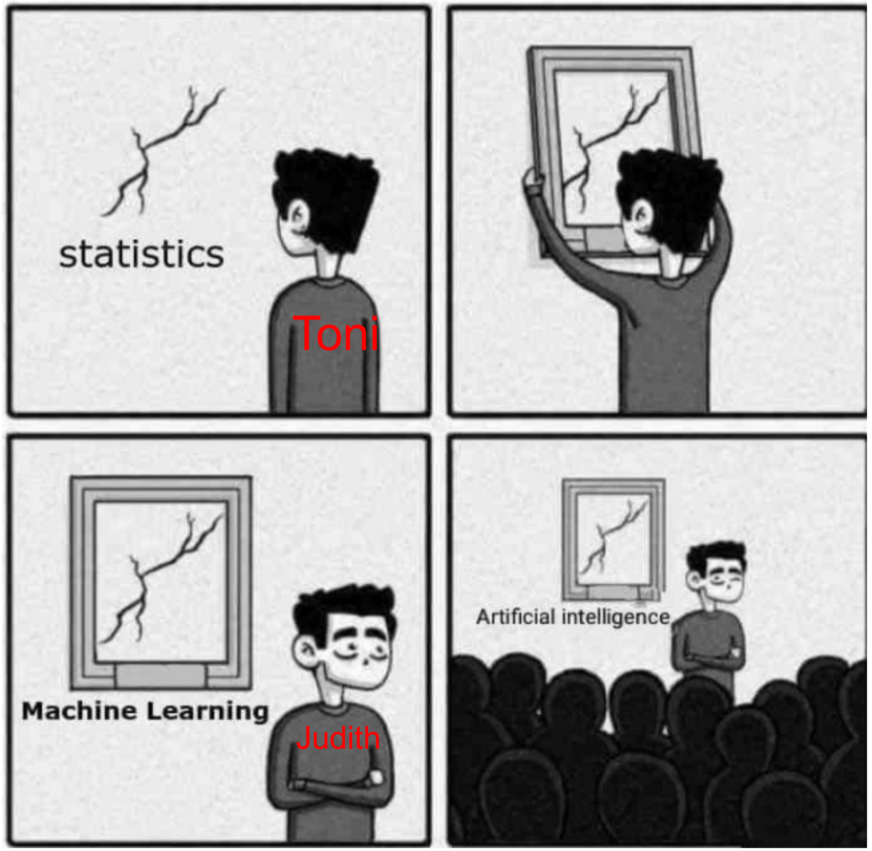
Low-Level Data

This is a picture of Barack Obama.
His foot is positioned on the right side of the scale.
The scale will show a higher weight.

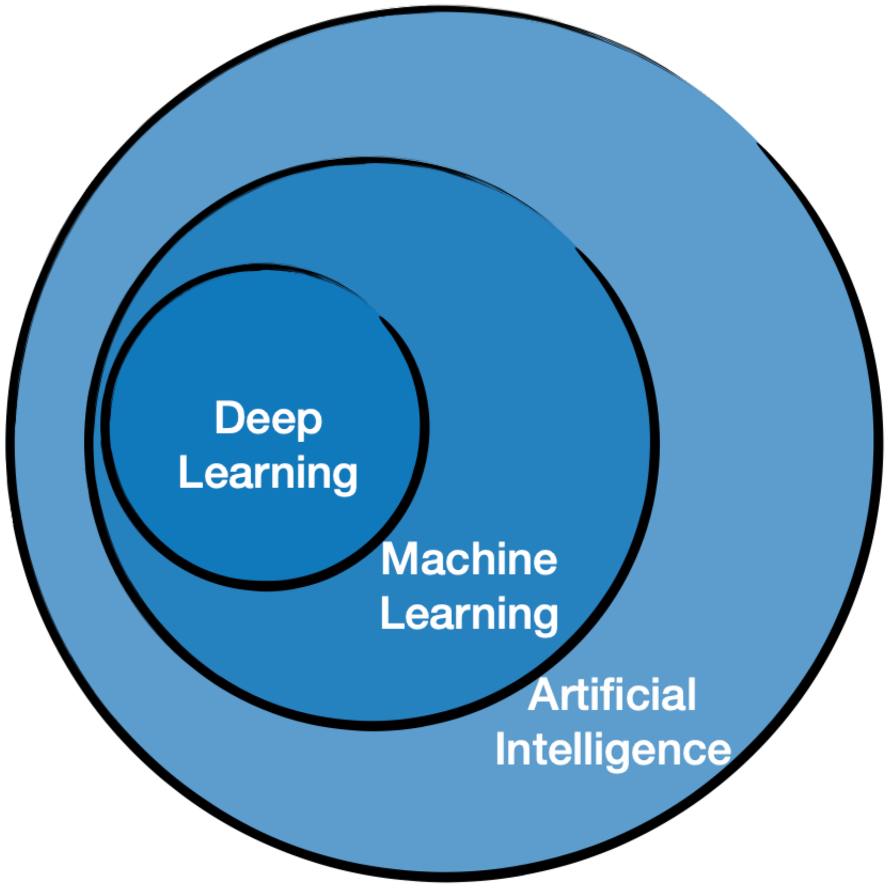
reconstruct high level concepts from low-level, high-dim data

What is machine learning?

What is machine learning?



What is machine learning?



AI: systems that simulate intelligent behavior e.g. via rules, reasoning, symbol manipulation

ML: subset of AI that learns to make decisions or predictions by fitting mathematical models to observed data.

DL: type of machine learning model, that aims at complex pipelines, work on low-level data (e.g. pixels)

ML examples: make decisions

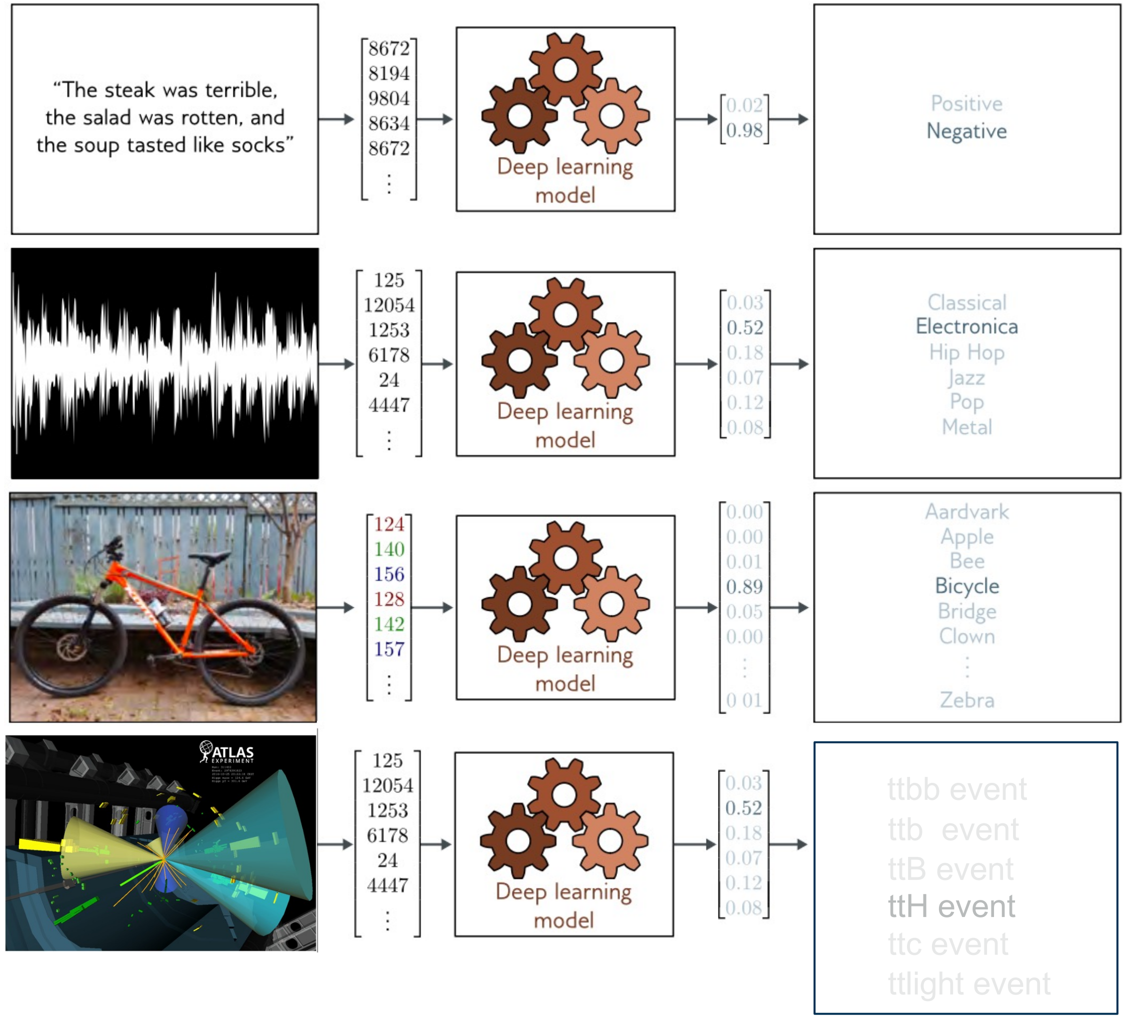
Classification

Variable length structured input

Fixed length structured input

Fixed length structured input

Variable length unstructured input (4vectors of particles)



Binary class

multi class

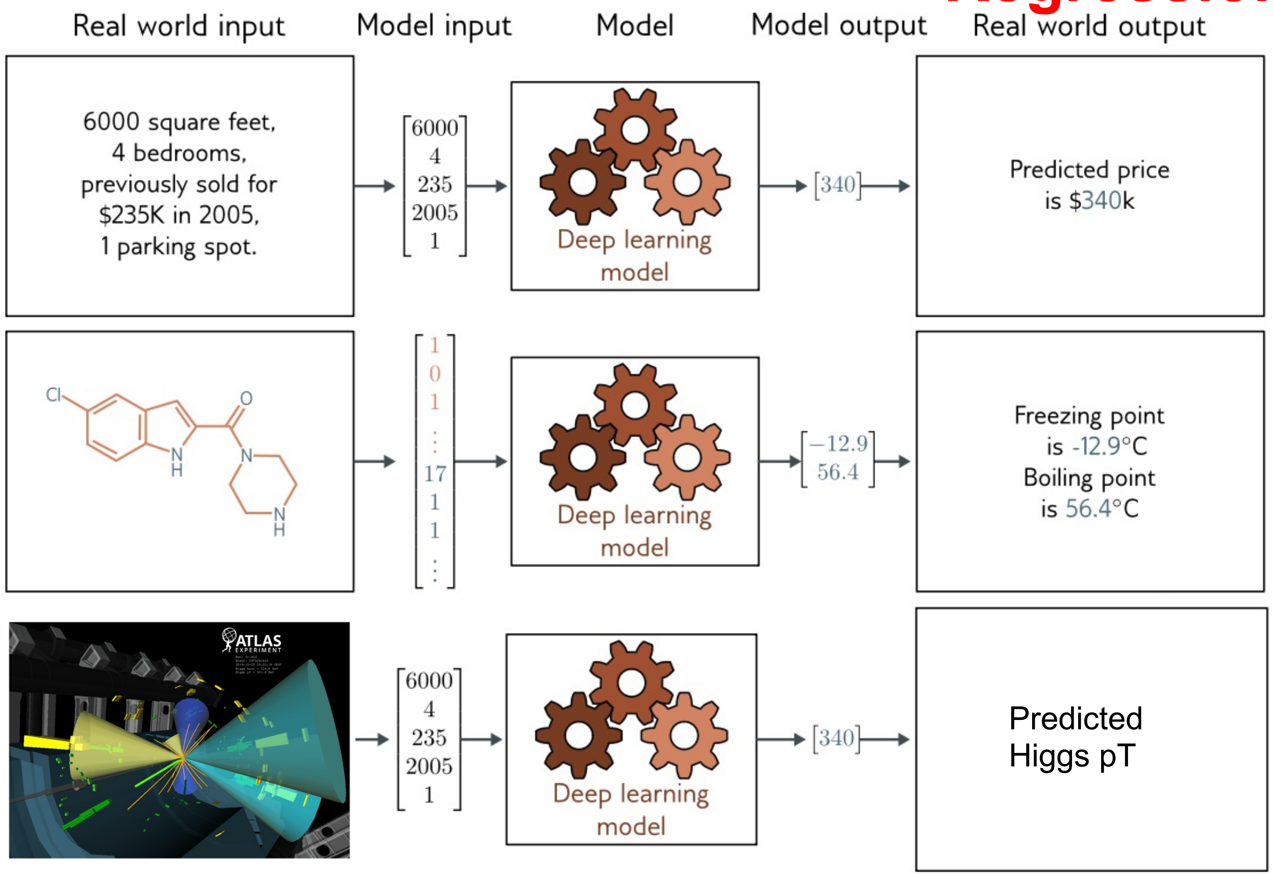
multi class

multi class

Adapted from http://udlbook.com.

ML examples: making predictions

Regression



Fixed length unstructured input — single real value output

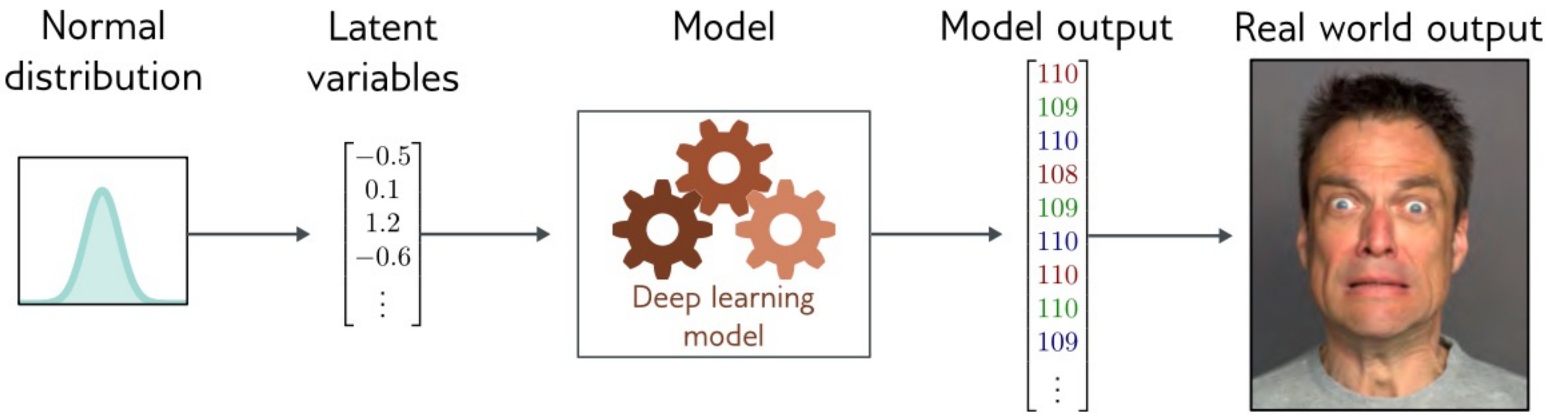
variable length structured input — multiple real value outputs

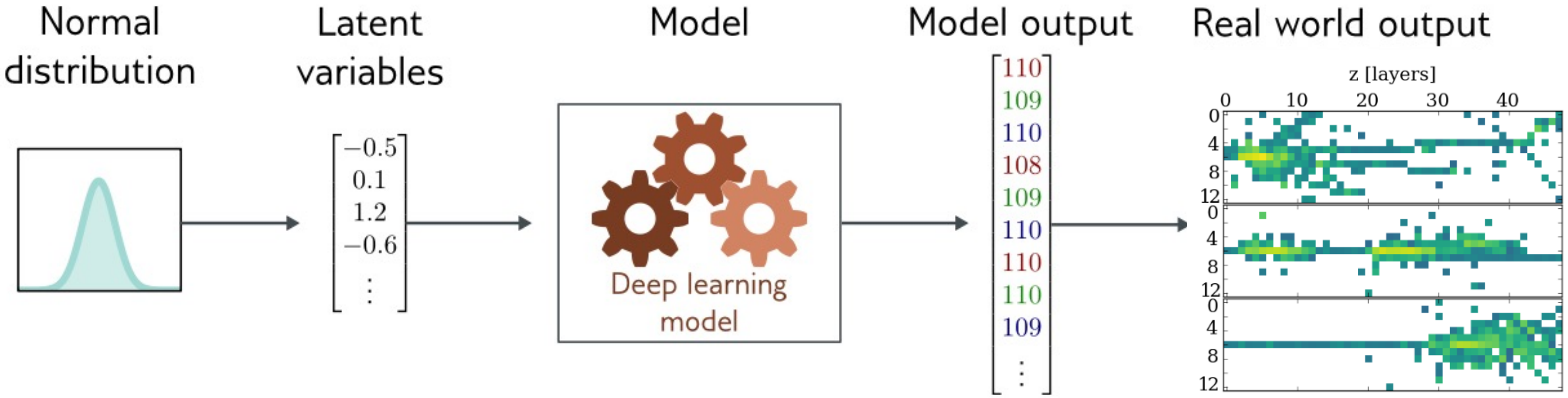
variable length unstructured input (4vectors of particles) — single real value output

Adapted from http://udlbook.com.

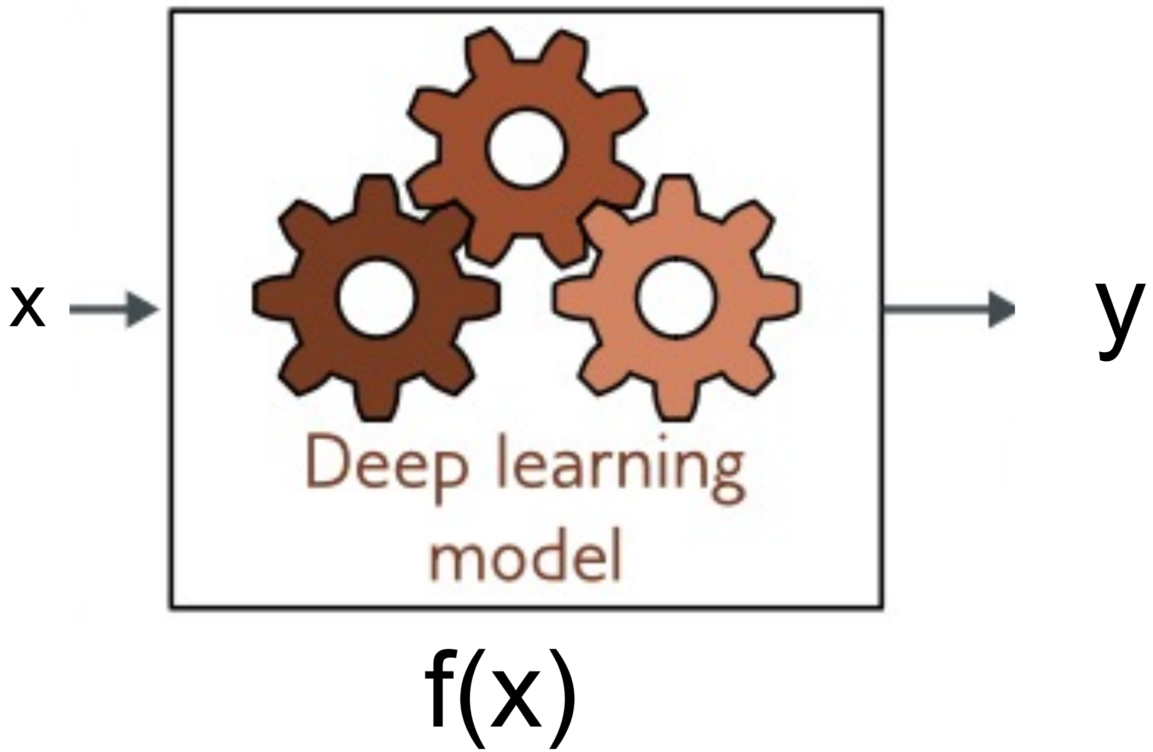
ML example: generate new data

Generation



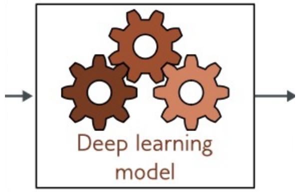


Pion in hadronic calorimeter


x
Deep learning
model
y
f(x)

What does the machine learn?

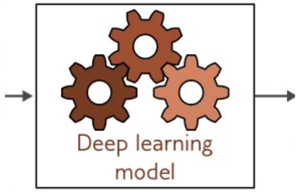
Open the box
or fitting mathematical model to data



output   input

$$y = f(x)$$

Open the box
or fitting mathematical model to data



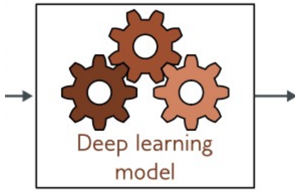
input

family of functions

output

$$y = f(x, \phi)$$

Open the box
or fitting mathematical model to data



output, input, family of functions

$$y = f(x, \phi)$$

Learning = search through a family of functions to let the data guide you to find the best one

Easiest if you have a labeled data set where the input-output relation is known to train and validate

The data

Your connection to the algorithm is the data

- The most important thing in the ML lifecycle

Need to know:

- Where does the existing (labeled) data come from?
- Where will the new data come from?

[src]

The dominant paradigm: statistical learning

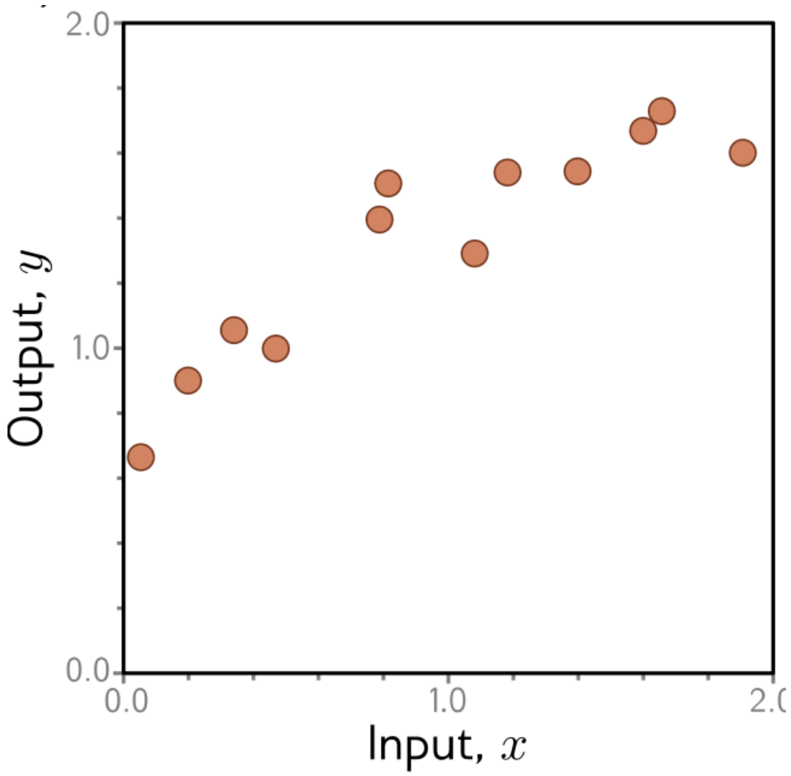
We **assume** the data is drawn i.i.d.

$$\text{data} = \{x_1, x_2, \ldots, x_N\} \qquad x \sim p(x)$$

We **assume** all **existing data and all future data** come from the same distribution.

- Danger: "Out-of-Distribution" samples / Distribution Shift

Example

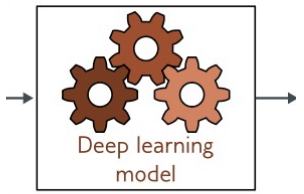


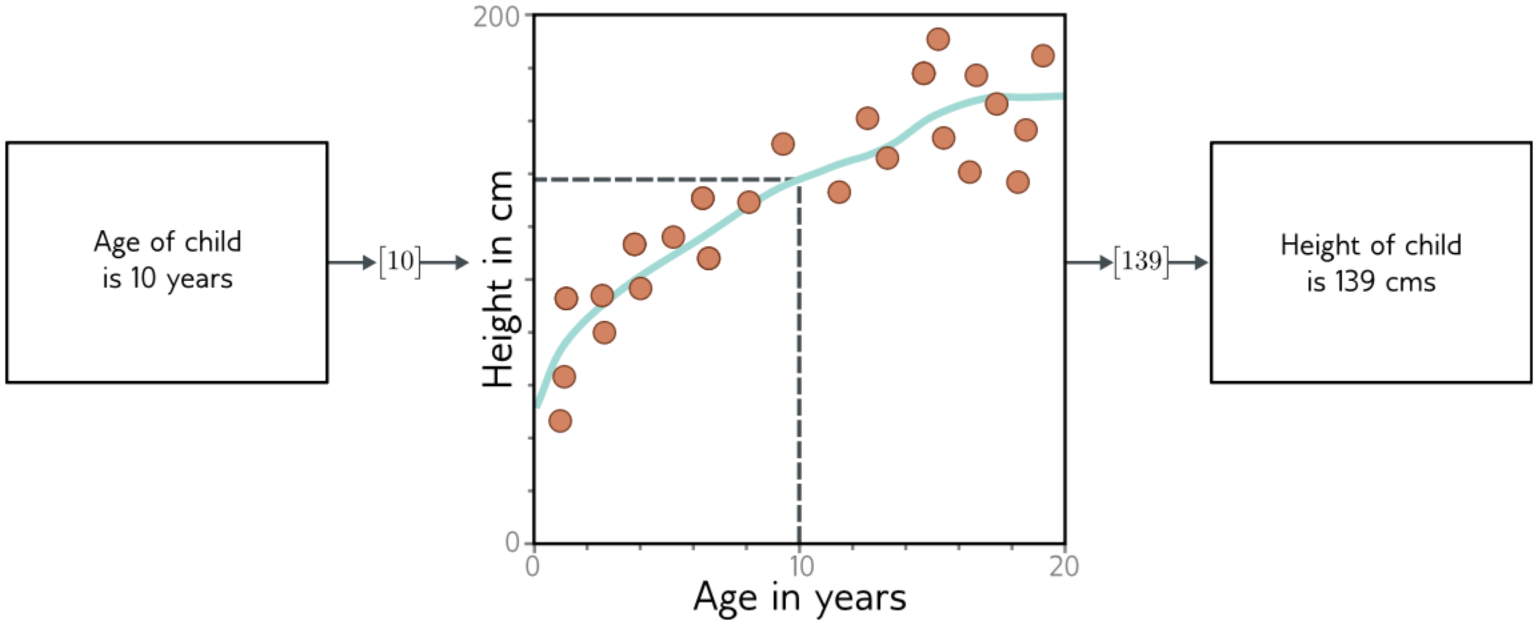
Labeled data set

Let’s try to describe them with a **linear function**, i.s. my set of hypothesis to describe the data is

$$
\begin{aligned}
y &= \mathrm{f}[x, \boldsymbol{\phi}] \\
&= \phi_0 + \phi_1 x.
\end{aligned}
$$

Open the black box
or what’s this “mapping”?



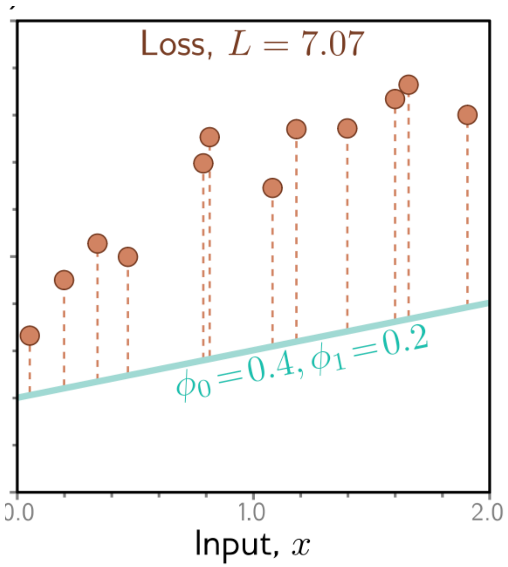


$$y = f_w[x]$$

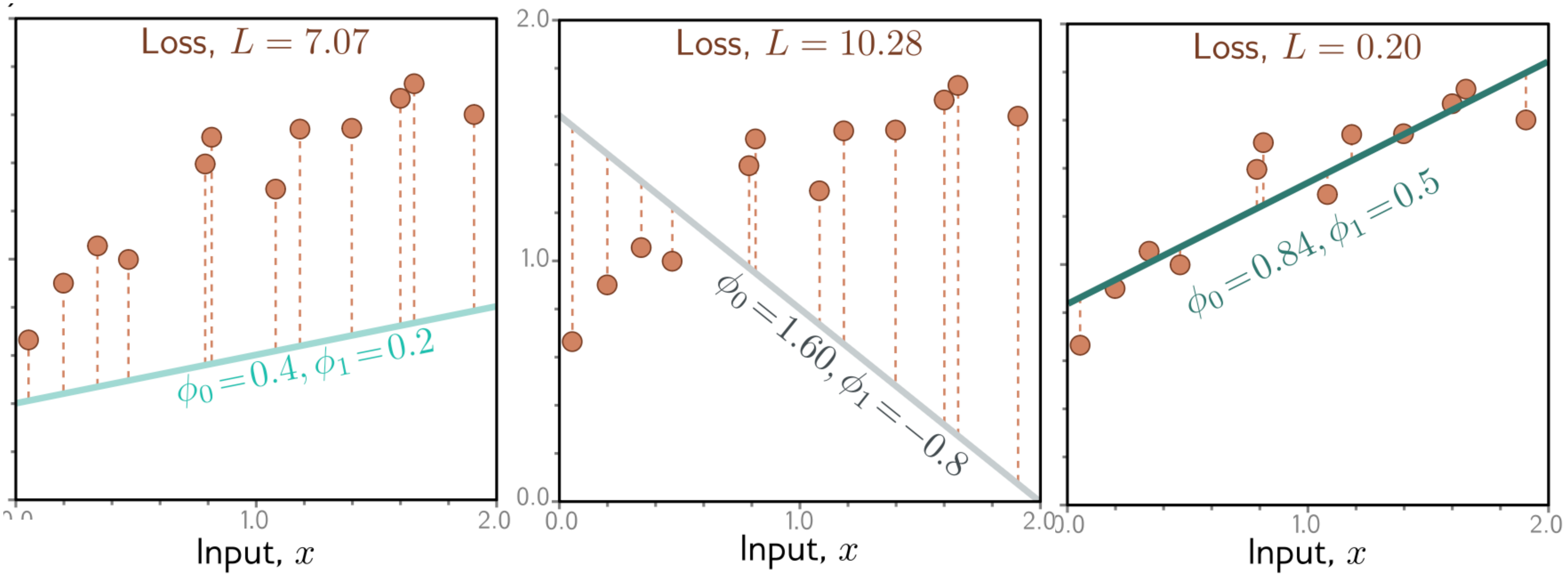
Learning = finding the optimal function from a set of functions to describe known “labeled” data

The Loss

- Need to have a performance measure to quantify what ”best” means: “loss”, “risk”, “cost” function

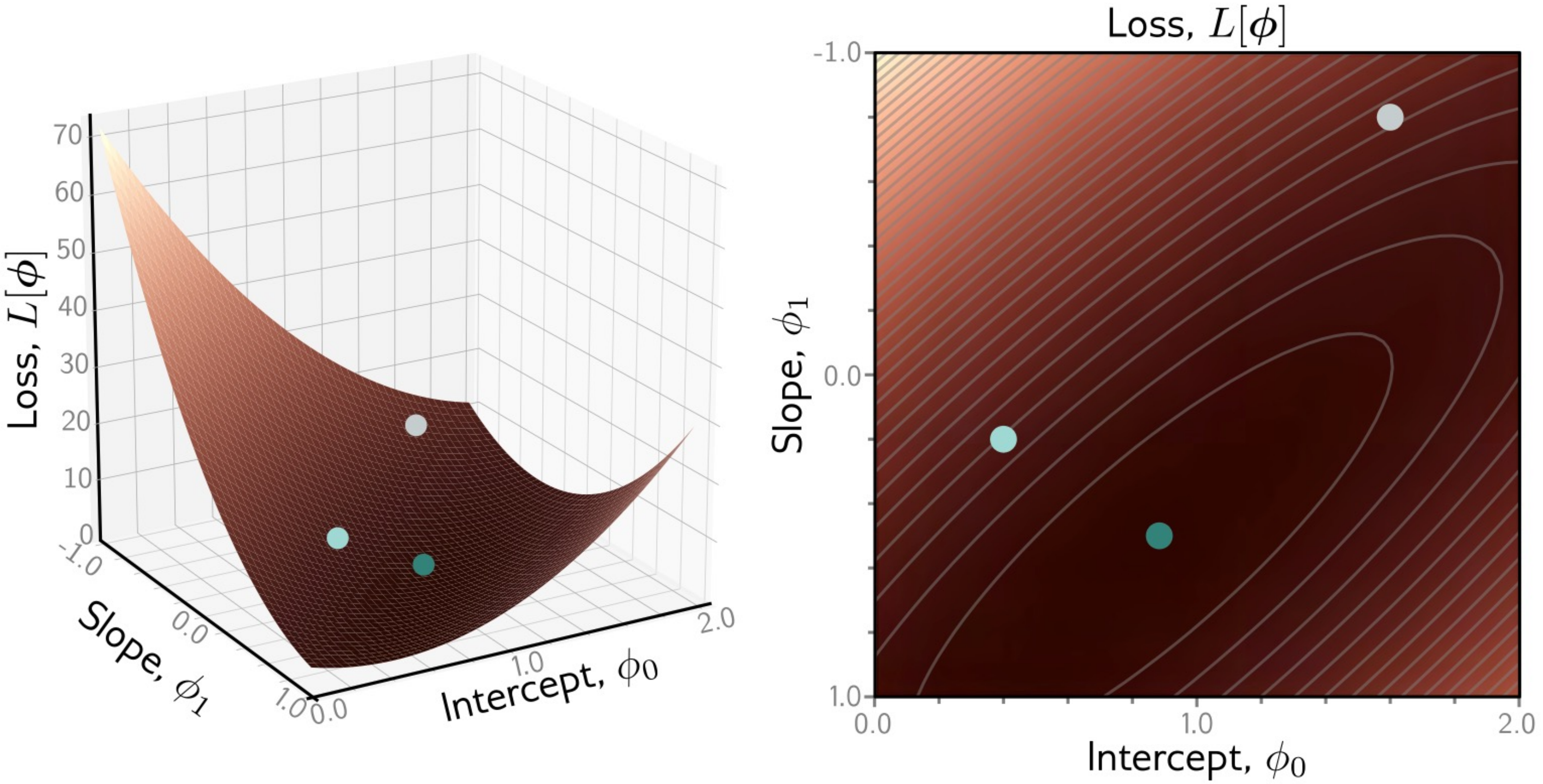


$$\begin{aligned} L[\phi] &= \sum_{i=1}^{I} (\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i)^2 \\ &= \sum_{i=1}^{I} (\phi_0 + \phi_1 x_i - y_i)^2 \end{aligned}$$



Loss, $L = 7.07$
$\phi_0 = 0.4, \phi_1 = 0.2$
Loss, $L = 10.28$
$\phi_0 = 1.60, \phi_1 = -0.8$
Loss, $L = 0.20$
$\phi_0 = 0.84, \phi_1 = 0.5$
2.0
1.0
0.0
0.0
1.0
2.0
Input, x
Input, x
Input, x

The Loss



Loss, $L[\phi]$
70
60
50
40
30
20
10
0
-1.0
0.0
1.0
Slope, ϕ_1
0.0
1.0
2.0
Intercept, ϕ_0
Loss, $L[\phi]$
-1.0
0.0
1.0
Slope, ϕ_1
0.0
1.0
2.0
Intercept, ϕ_0

Learning algorithms

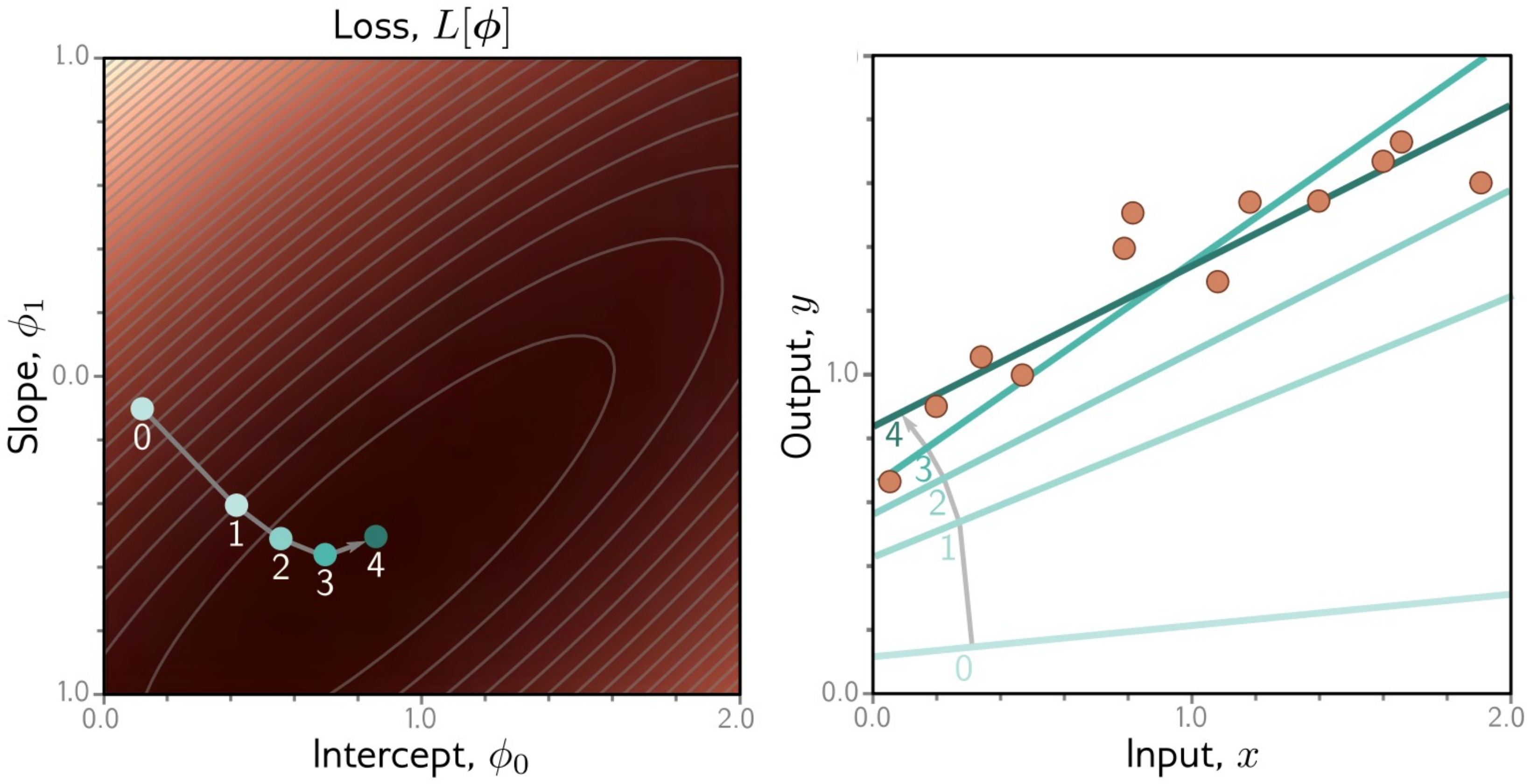
We usually have no idea which of the functions is the best, we need to have a learning algorithm that leads us there

Various possibilities:

- Exhaustive search (discrete functions)
- Closed form solutions (rare)
- Iterative optimization (mostly used)

$$\begin{aligned}\hat{\phi} &= \underset{\phi}{\operatorname{argmin}}\Big[L[\phi]\Big] \\ &= \underset{\phi}{\operatorname{argmin}}\left[\sum_{i=1}^{I}\left(\mathrm{f}[x_i,\phi]-y_i\right)^2\right] \\ &= \underset{\phi}{\operatorname{argmin}}\left[\sum_{i=1}^{I}\left(\phi_0+\phi_1 x_i-y_i\right)^2\right]\end{aligned}$$

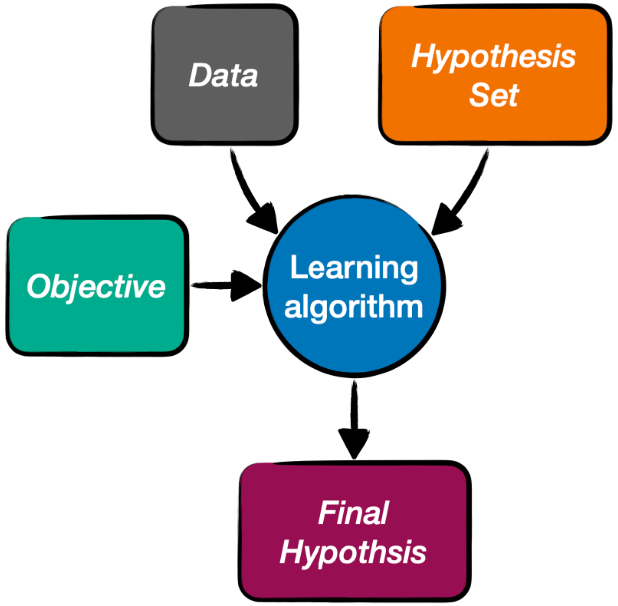
Learning algorithm



This case: exact solutior $\hat{\phi}$ Phi = $(X^TX)^{-1}X^T$ y with X_{ik} = x_i^k (i-th data point, k-th power)

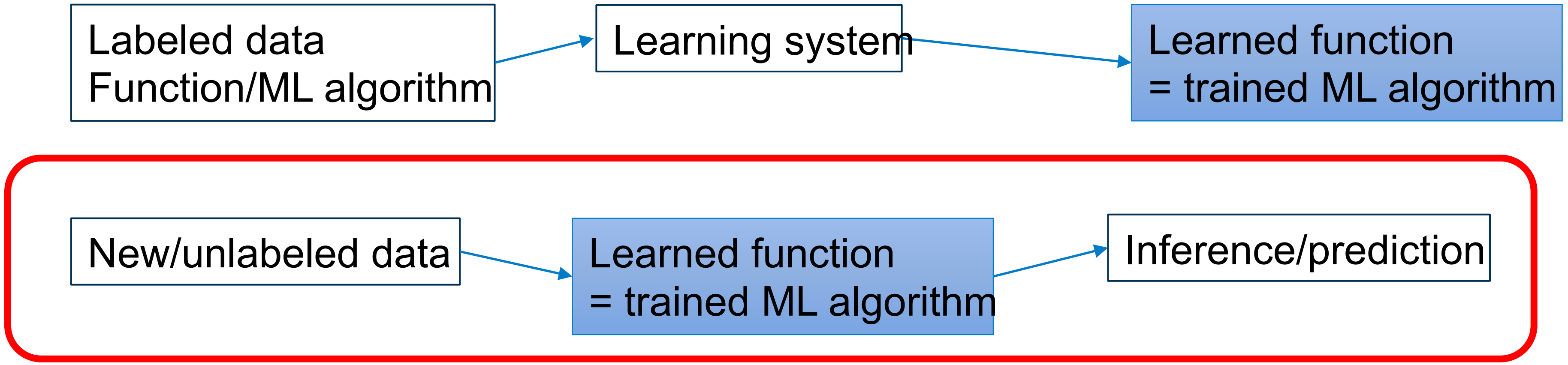
Learning framework
Putting it all together

- Collect and prepare data to be consumed by the machine
- Define the task (objective)
- Choose search space of possible functions (algorithms) aka "hypothesis set"
- Define what "good" means, i.e. a performance measure
- Provide an optimising algorithm to update functions, i.e. change hypothesis
- Decide when to stop and to define the final hypothesis (function)



Supervised learning

mapping from input data to an output prediction



Neural nets

More complex family of functions

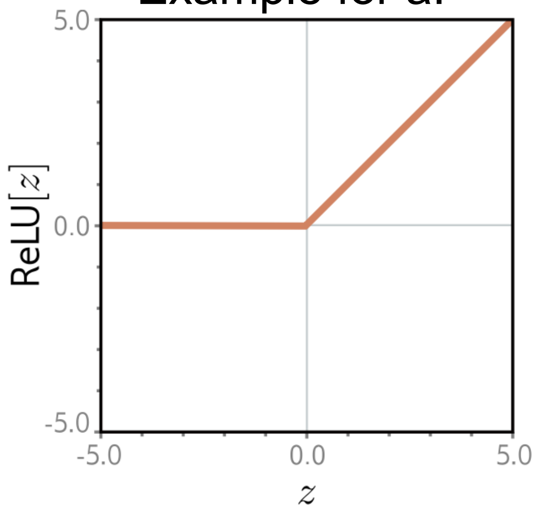
Build complexity by composing very simple building blocks

$$
\begin{aligned}
y &= \mathrm{f}[x, \boldsymbol{\phi}] \\
&= \phi_0 + \phi_1 \mathrm{a}[\theta_{10} + \theta_{11} x] + \phi_2 \mathrm{a}[\theta_{20} + \theta_{21} x] + \phi_3 \mathrm{a}[\theta_{30} + \theta_{31} x]
\end{aligned}
$$

Linear functio
n of
input

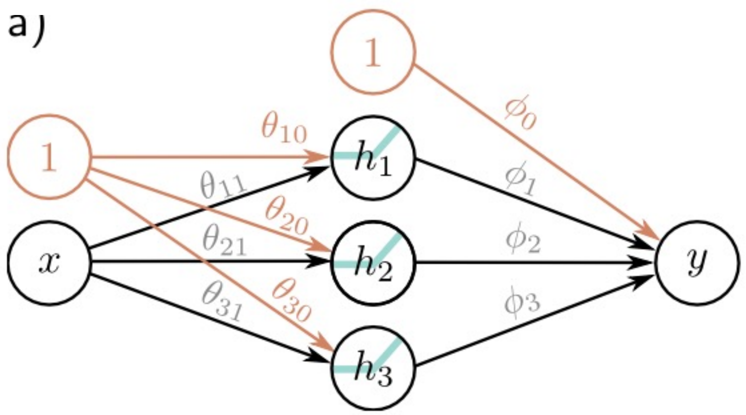
10 parameters ϕ_i, θ_j ,
activation function a

Example for a:



Neural network family of functions

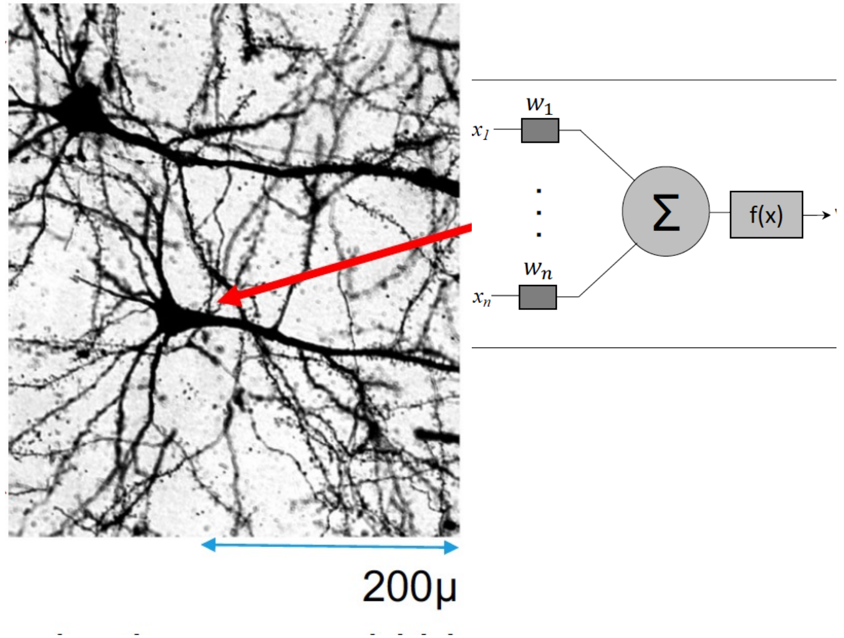
$$
\begin{aligned}
y &= \mathrm{f}[x, \boldsymbol{\phi}] \\
&= \phi_0 + \phi_1 \mathrm{a}[\theta_{10} + \theta_{11} x] + \phi_2 \mathrm{a}[\theta_{20} + \theta_{21} x] + \phi_3 \mathrm{a}[\theta_{30} + \theta_{31} x]
\end{aligned}
$$



input

hidden layer with nodes h_i

output



$$
\begin{aligned}
h_1 &= \mathrm{a}[\theta_{10} + \theta_{11} x] \\
h_2 &= \mathrm{a}[\theta_{20} + \theta_{21} x] \\
h_3 &= \mathrm{a}[\theta_{30} + \theta_{31} x],
\end{aligned}
$$

Neural network hard wired

- Mark I perceptron

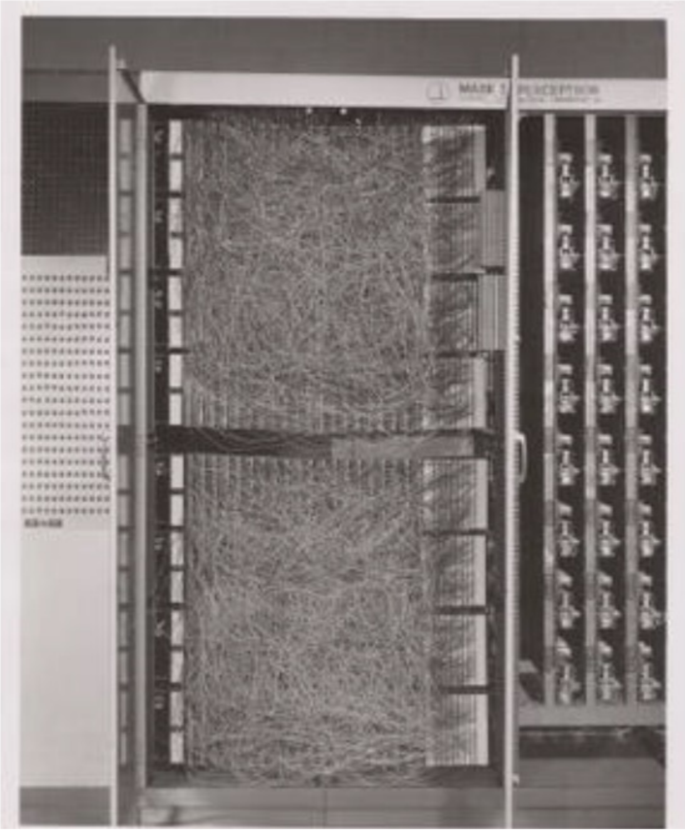
Perceptron:

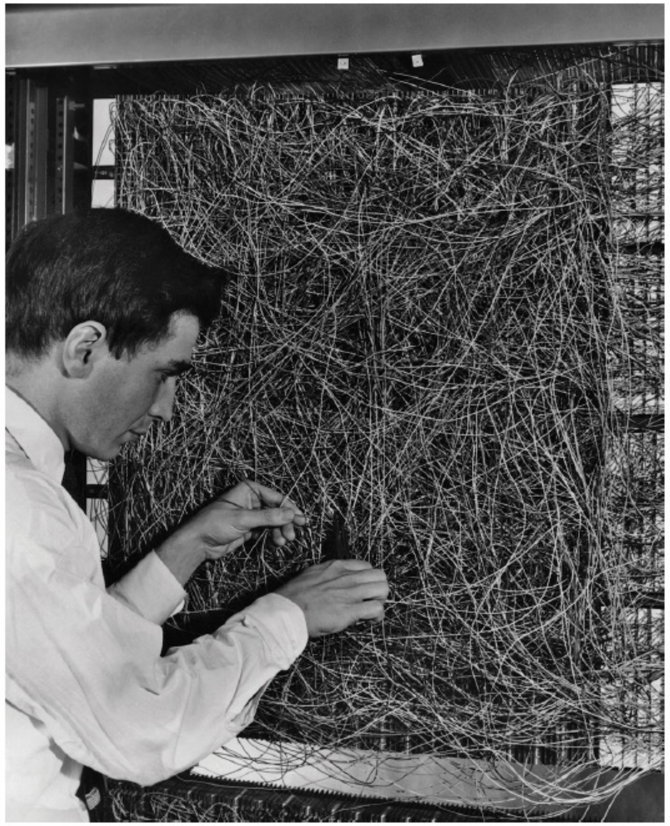
$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0, \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{w} \cdot \mathbf{x} = \sum_{i=1}^{m} w_i x_i$$

Images of 20x20 photo cells were trained for image recognition: “connections” = wires between photo cells and neurons
“weights” = potentiometers moved by electrical motors

1958



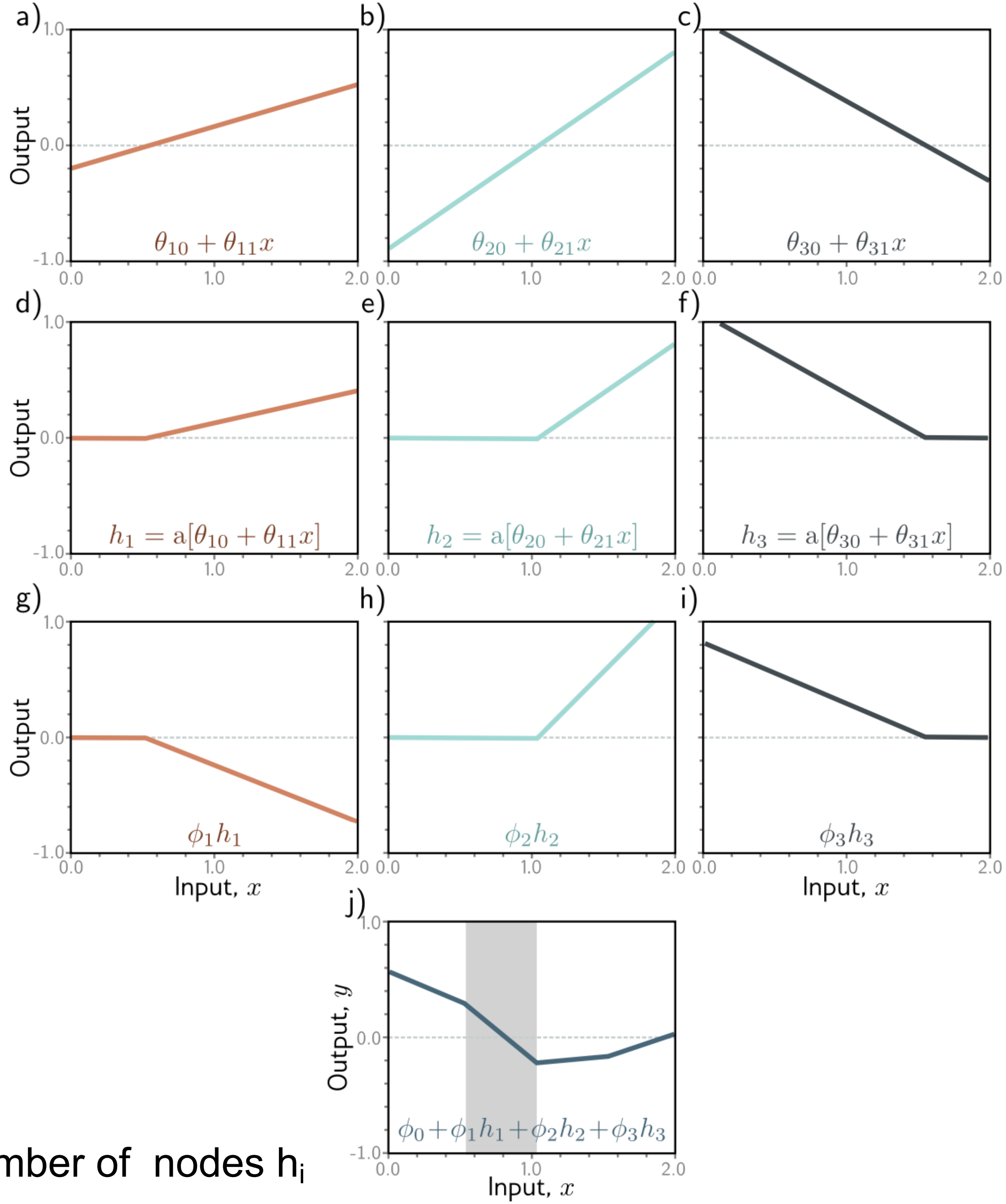


Linear function of the input

Output of the activation function

Weighted output of the activation function

Resulting y = f(x,ϕ)

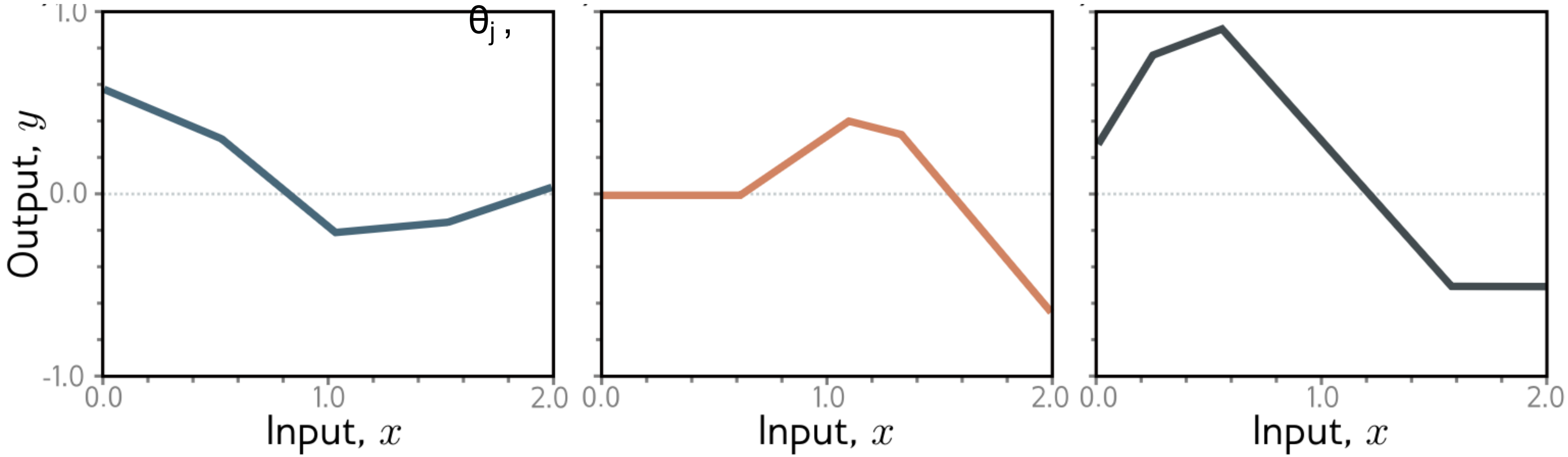


Piecewise linear function

Number of joints given by number of nodes h_i

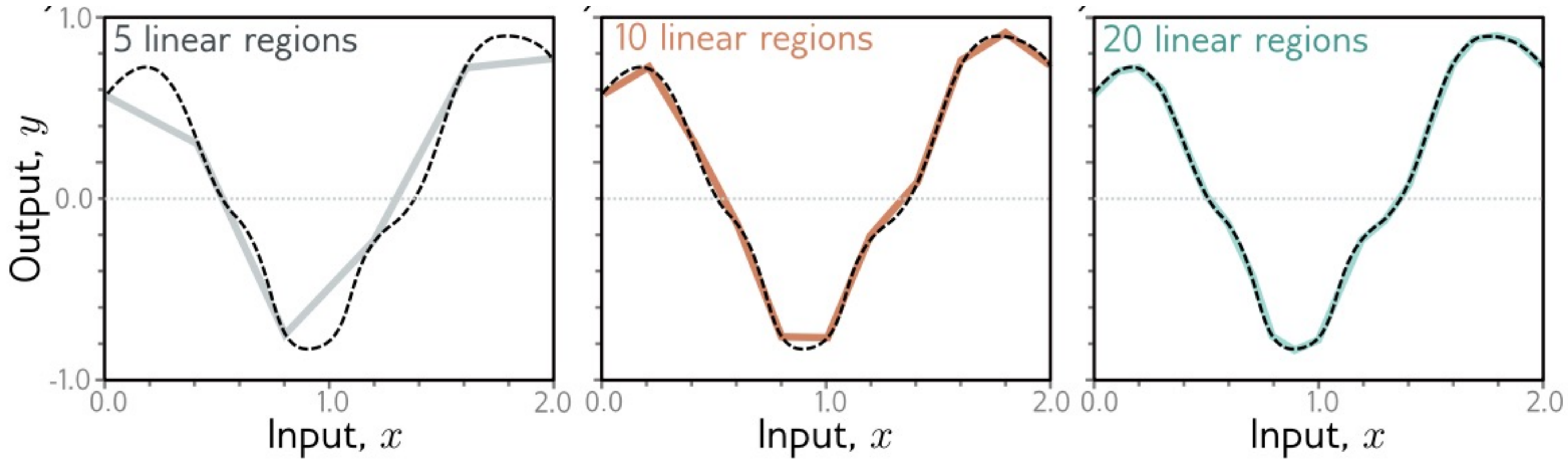
More variability

Different settings of parameters ϕ_i, θ_j ,



Expanding the number of nodes

A lot to gain!

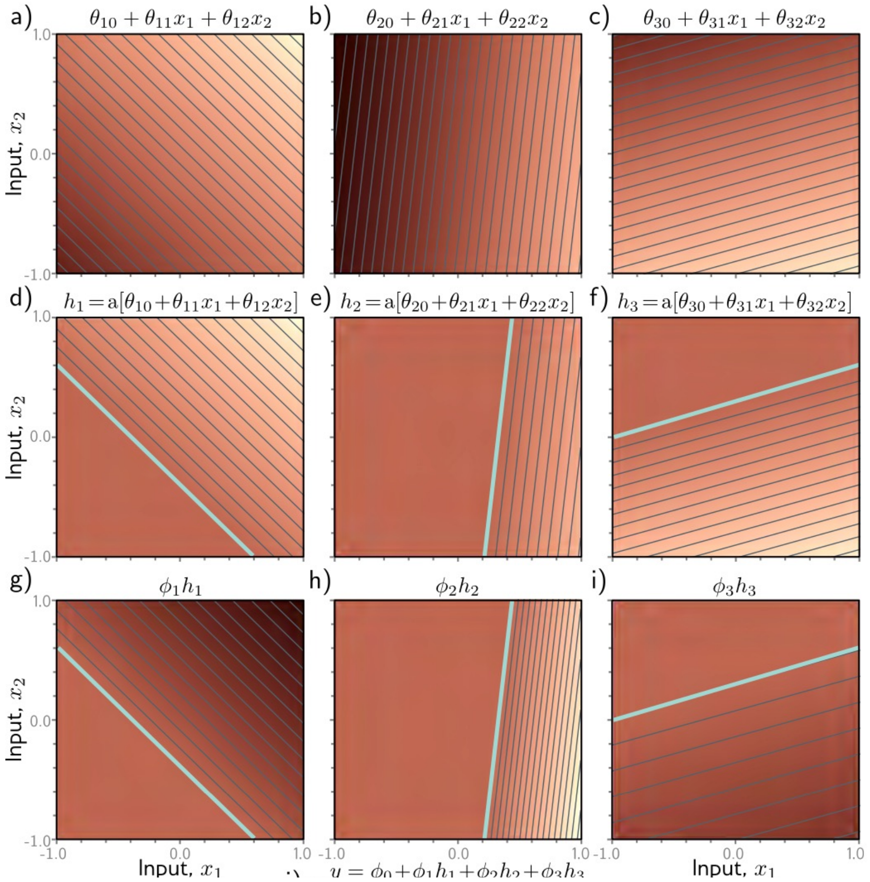


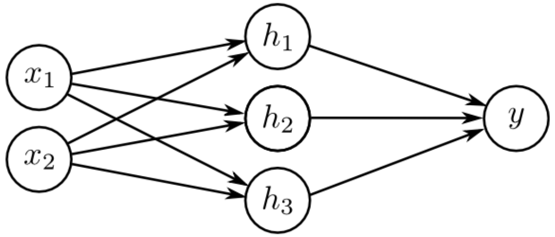
Neural networks with a single hidden layer are universal function approximators

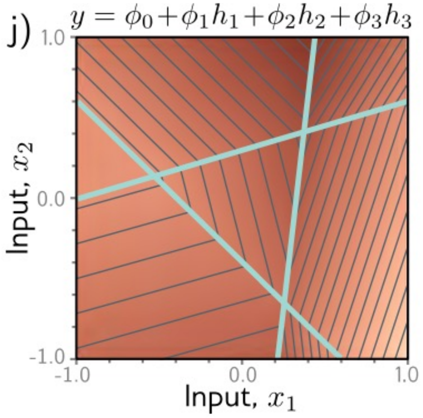
This also holds for multi-dimensional inputs and outputs.

Side remark: it does NOT work for linear activation function, e.g. XOR problem

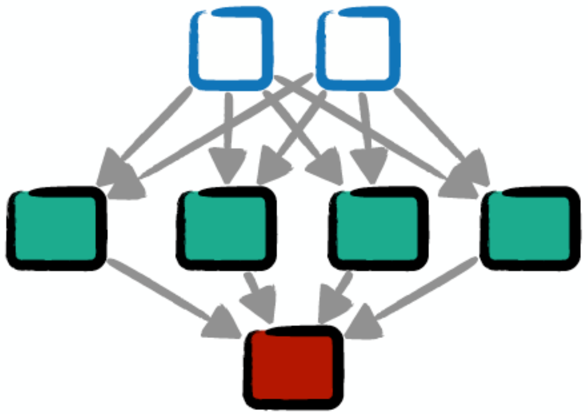
Going more complex

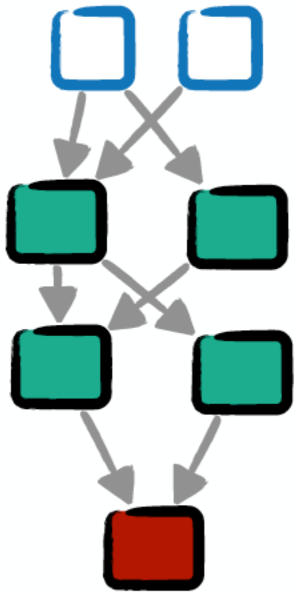






Beyond single layer





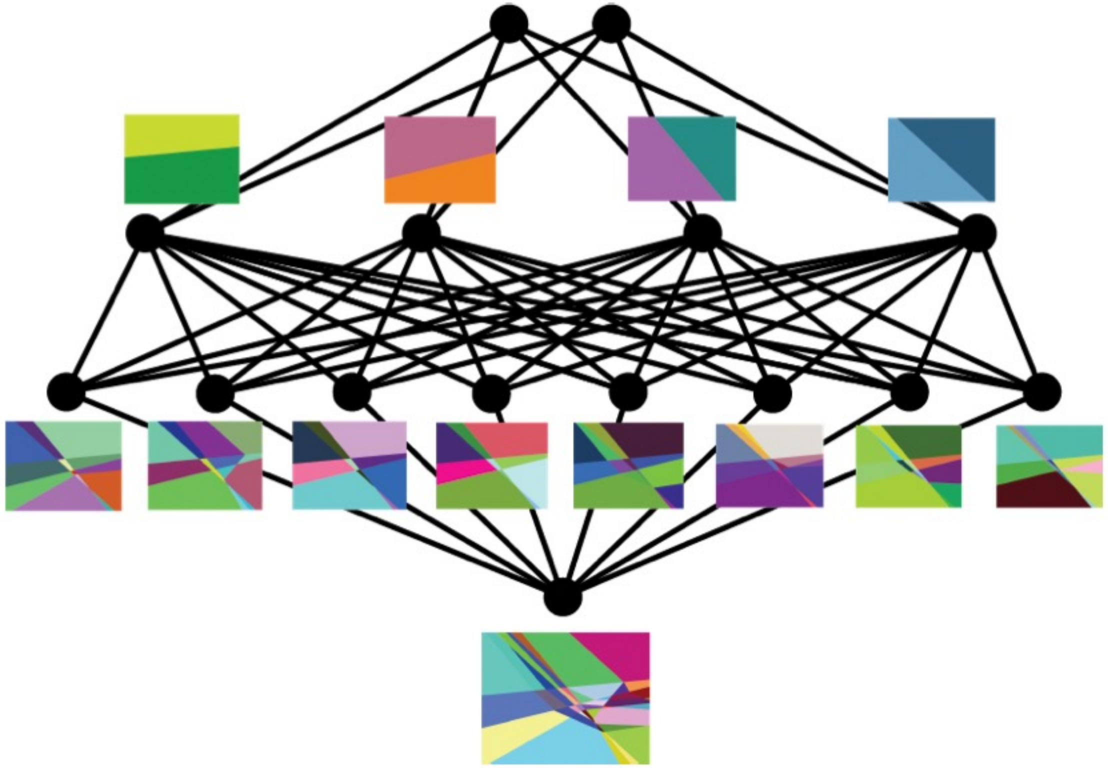
Not forbidden to stack neurons in a different way: go deep instead of wide

->opportunity to build up complex things step by step

Wide or deep?

The relationship between expressivity of swallow networks and deep networks is an active area of research

But empirically:
It seems that deep networks can generate complex patterns with much fewer parameters

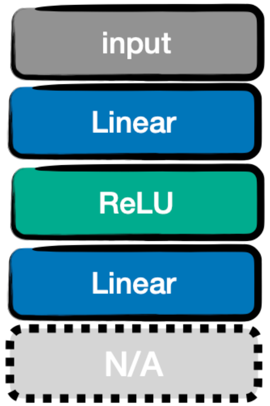


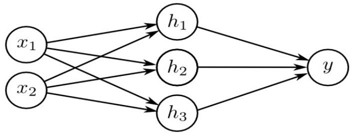
Activation functions

UFA is achieved with any non-linear activation function, but at least for the output activation we need to be careful about the task

Regression

$\mu_\phi(x) \in \mathbb{R}$

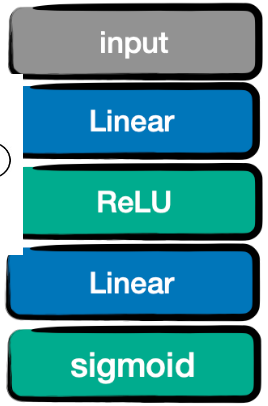




No activation!

Binary Classification

$\theta(x) \in [0,1]$

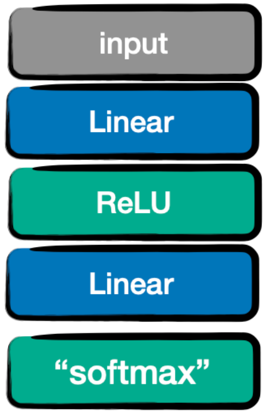


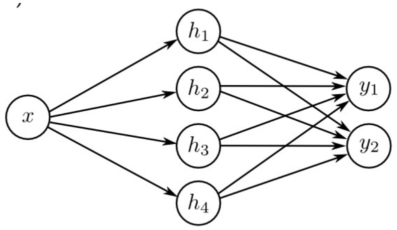
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

22

Multi-class Classification

$p_i(x) \geq 0 \text{ s.t.} \sum p_i = 1$





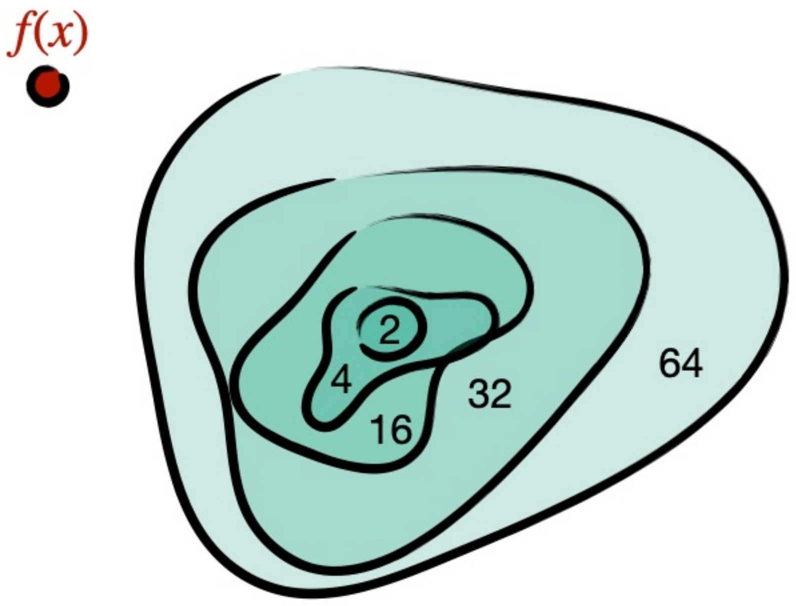
$$\text{softmax}(x) = \frac{e^{x_i}}{\sum_i e^{x_i}}$$

How big should we go?

With increasing size you get a better chance that the actual algorithm you are looking for lives within the family of functions

Bias: the loss $L(f_{min})$ of the overall best function $f \in \mathcal{H}$

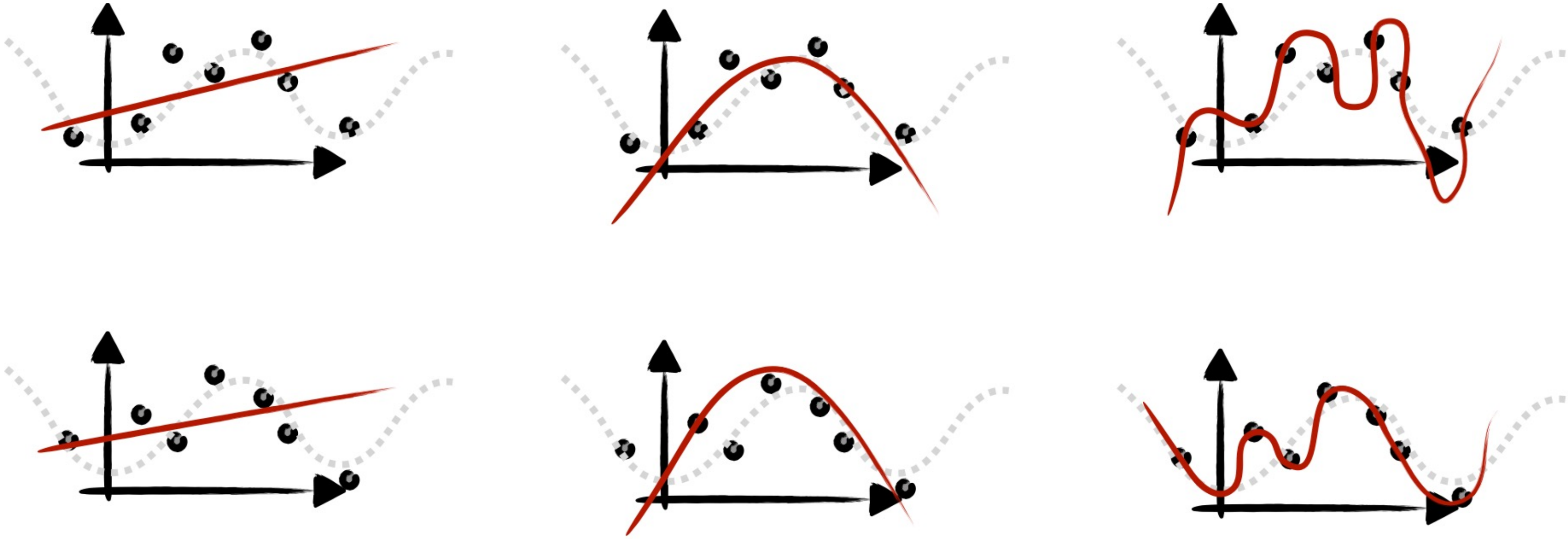
$f_{min} = \langle f(x, \hat{\phi}) \rangle_D$



An argument to make the function family as big as possible

But should we really….?

“With four parameters I can fit an elephant, and with five I can make him wiggle his trunk”
 - John von Neumann



No free lunch theorem of learning

- With limited data you must learn effectively, i.e. you must restrict the "hypothesis set" of functions to perform the task

- Only possible with "inductive bias": constrain on the hypothesis set by adjusting the search space

Empirical risk minimization

The risk we want

In statistical learning we are interested in the expected performance of the algorithm on future data

With assumpumption of i.i.d. distribution of data:

Performance of the hypothesis for a specific input

$$\bar{L} = \int_S p(s)L(s,h) = \mathbb{E}_{p(s)} L(s,h)$$

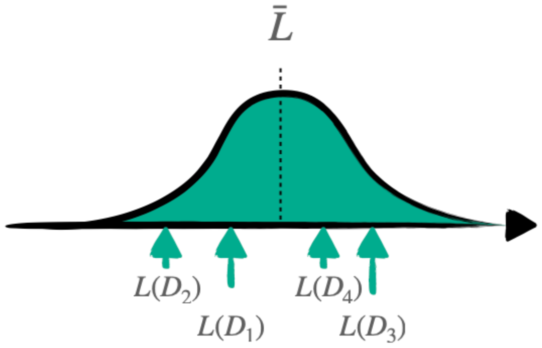
Distribution of possible inputs

The risk we can get

While we don't have p(s) we do have samples s ~ p(s)

- We can (only) estimate the loss empirically as a proxy!

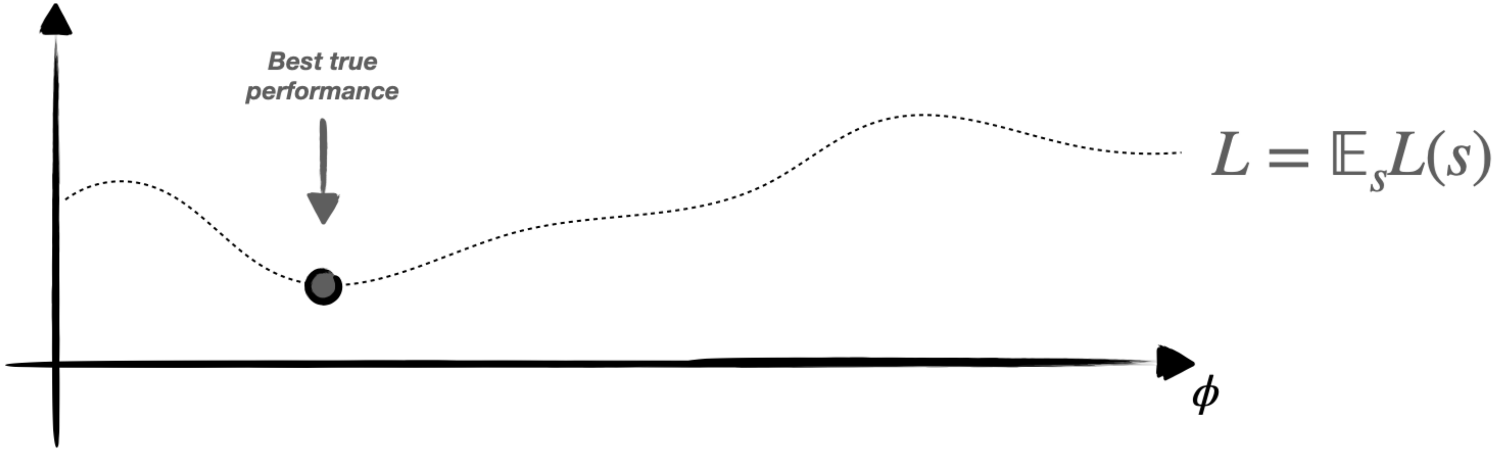
$$\bar{L} = \int_S p(s)L(s,h) \rightarrow \hat{L} = \frac{1}{N}\sum_i L(s_i, h)$$



This difference between what we want and what we get has tricky consequences

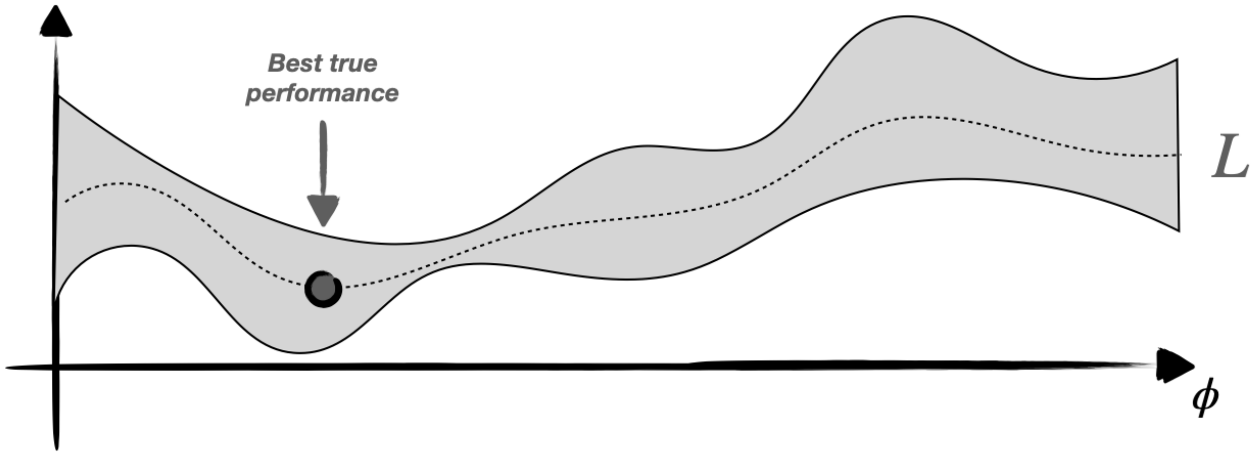
Empirical risk minimisation

Keep in mind that this is just a proxy depending on the training data set we have!



Empirical risk minimization

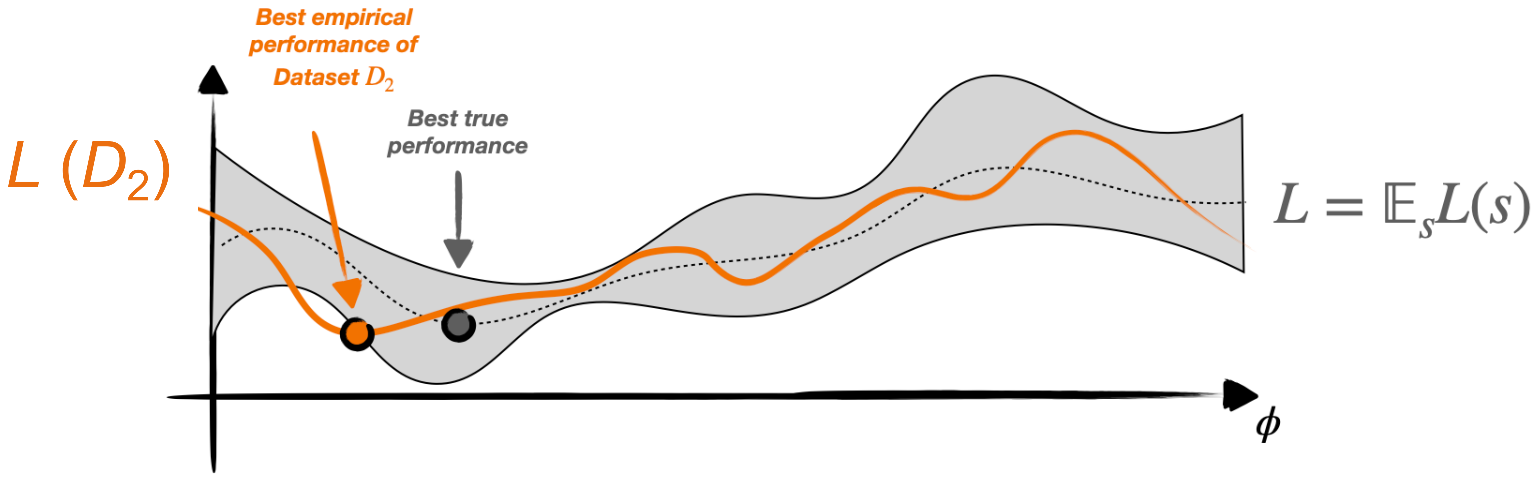
Keep in mind that this is just a proxy depending on the training data set we have!



$L = \mathbb{E}_s L(s)$

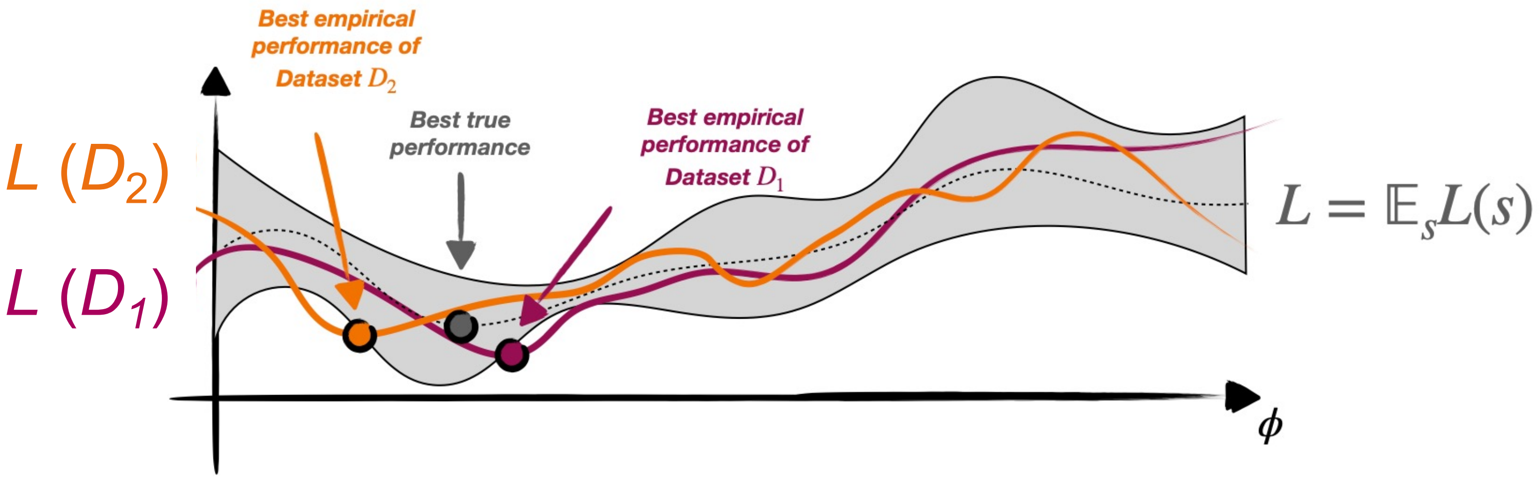
Empirical risk minimization

Keep in mind that this is just a proxy depending on the training data set we have!

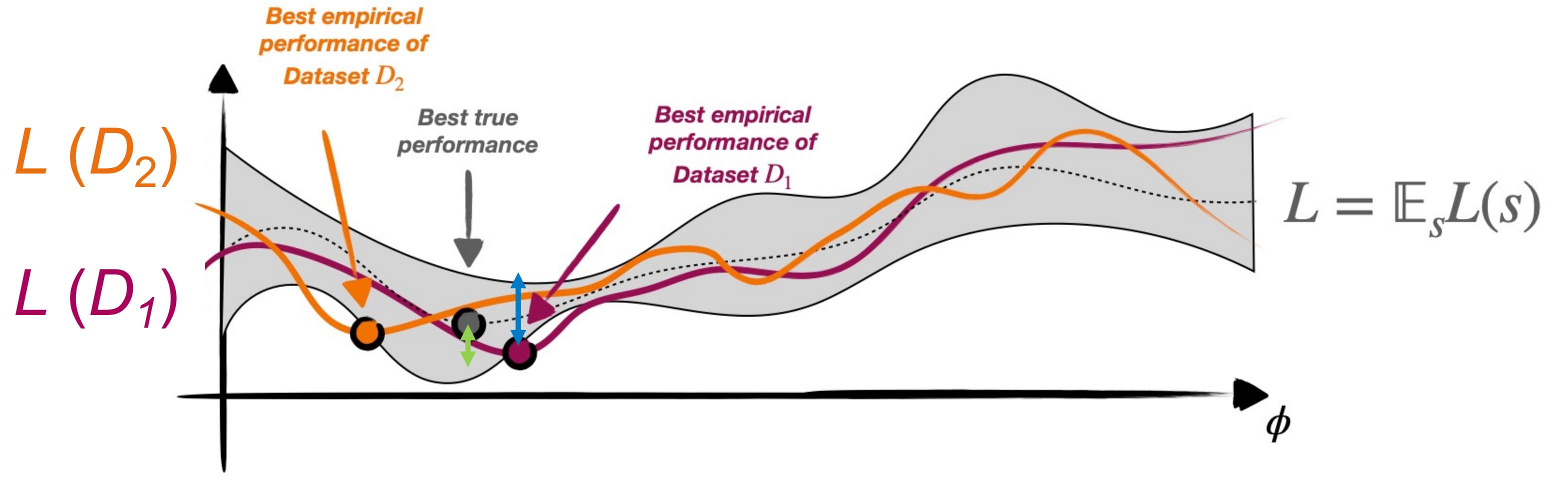


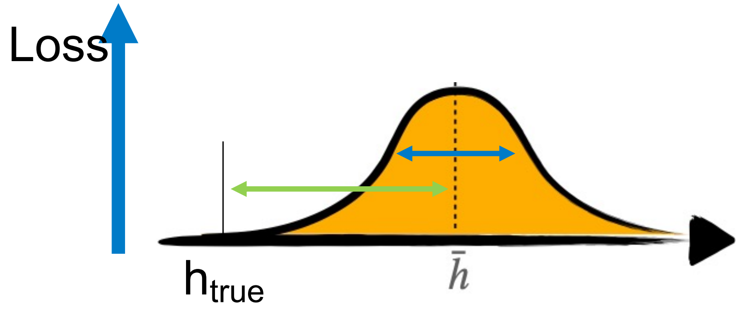
Empirical risk minimization

Keep in mind that this is just a proxy depending on the training data set we have!



Variance and bias





variance

Variance: spread of the the loss of h* in $\mathcal{H}$

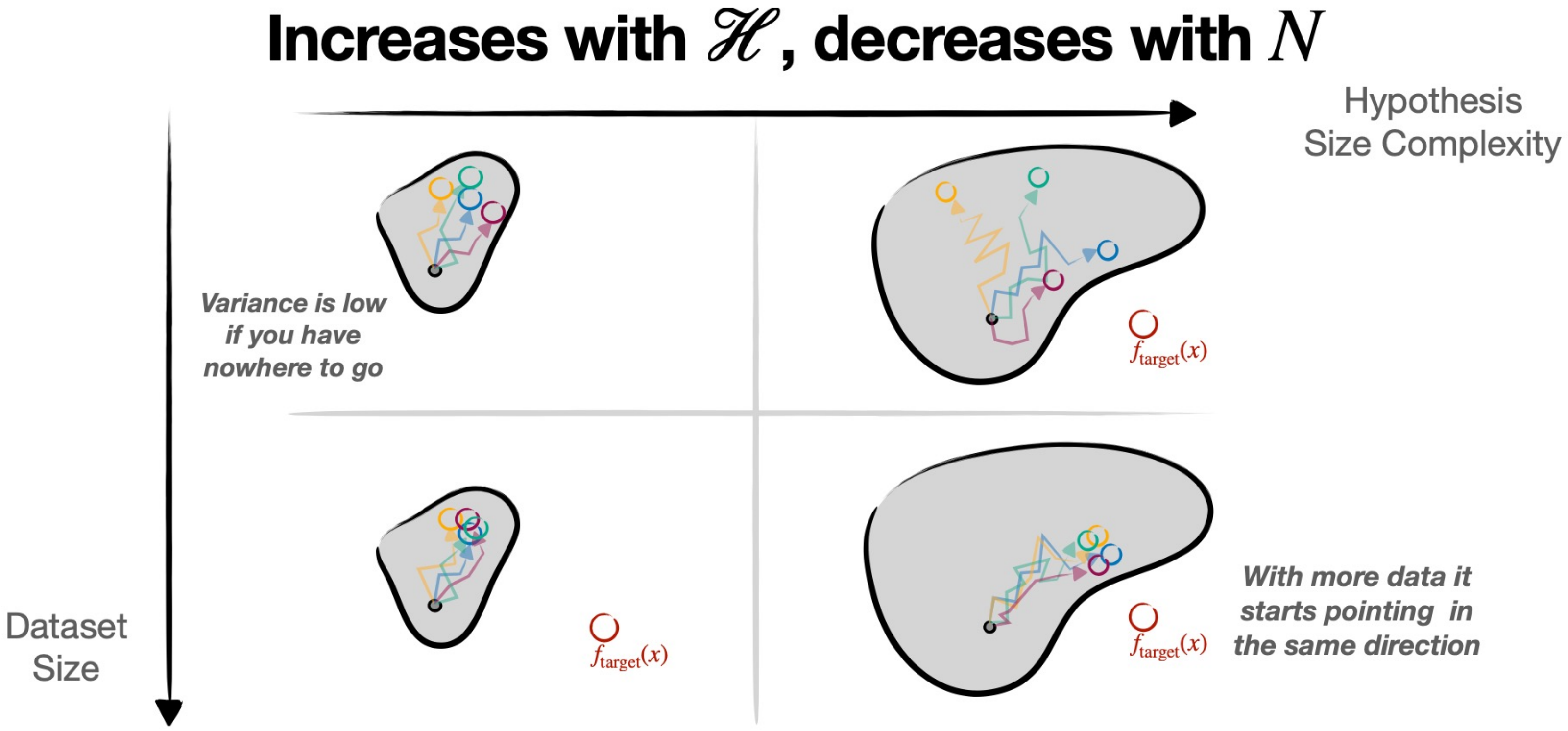
bias

offset of the loss of h* with respect to the best true loss

h* best hypothesis/function on a given data set

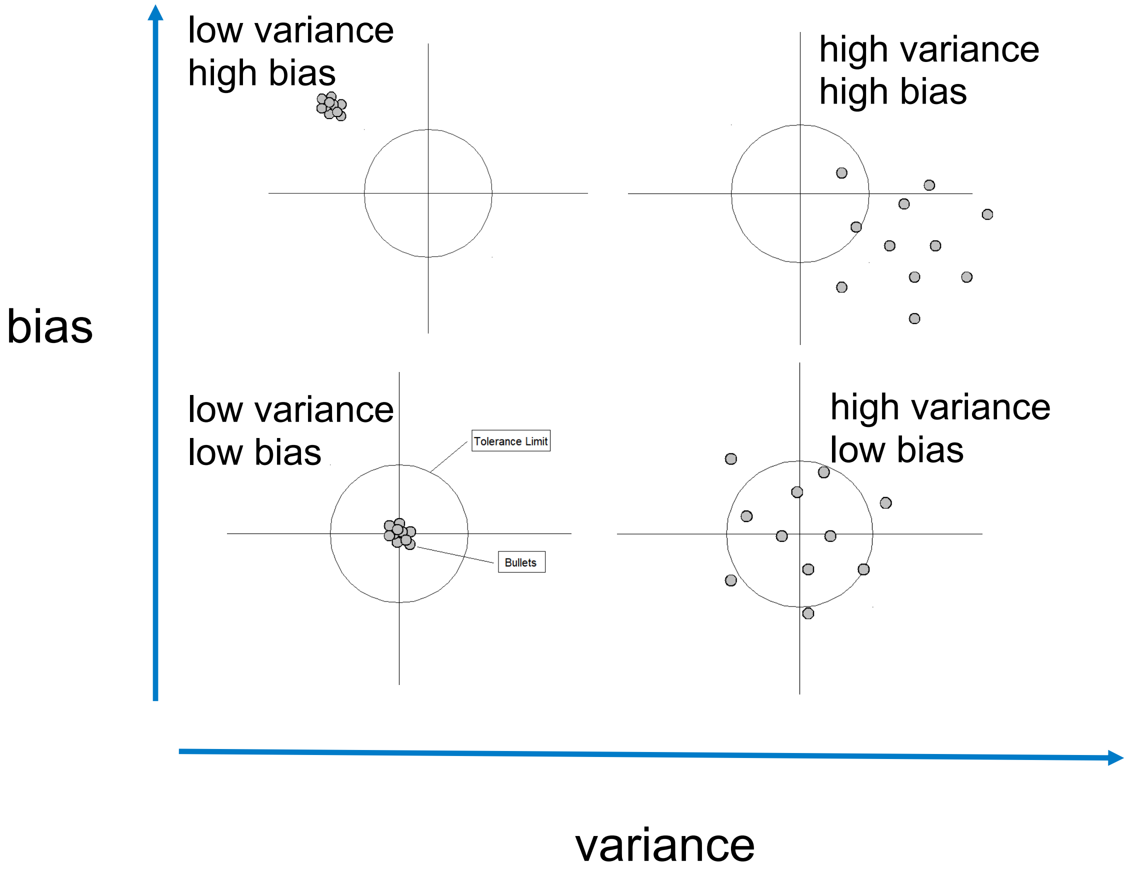
$$L_{true}(h^*) = Bias(L(h^*))^2 + Var(L(h^*))$$

Variance



An argument to make the hypothesis set as small as possible given the data set size

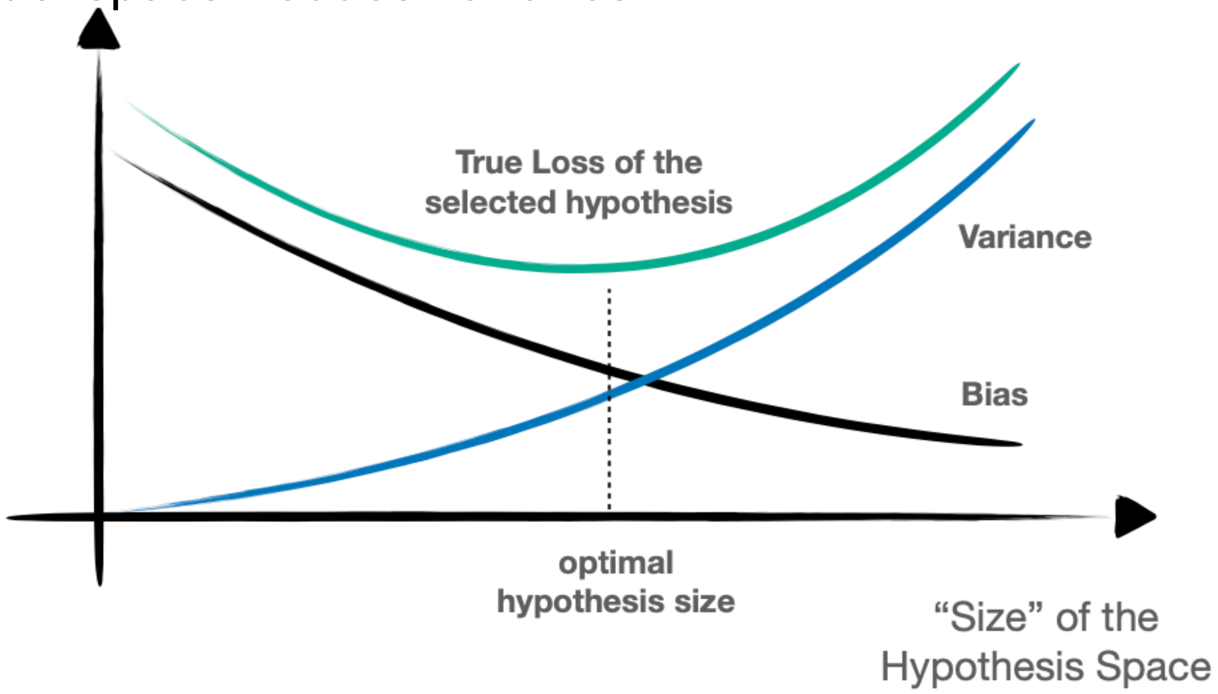
Bias - Variance trade-off



Bias-Variance trade off

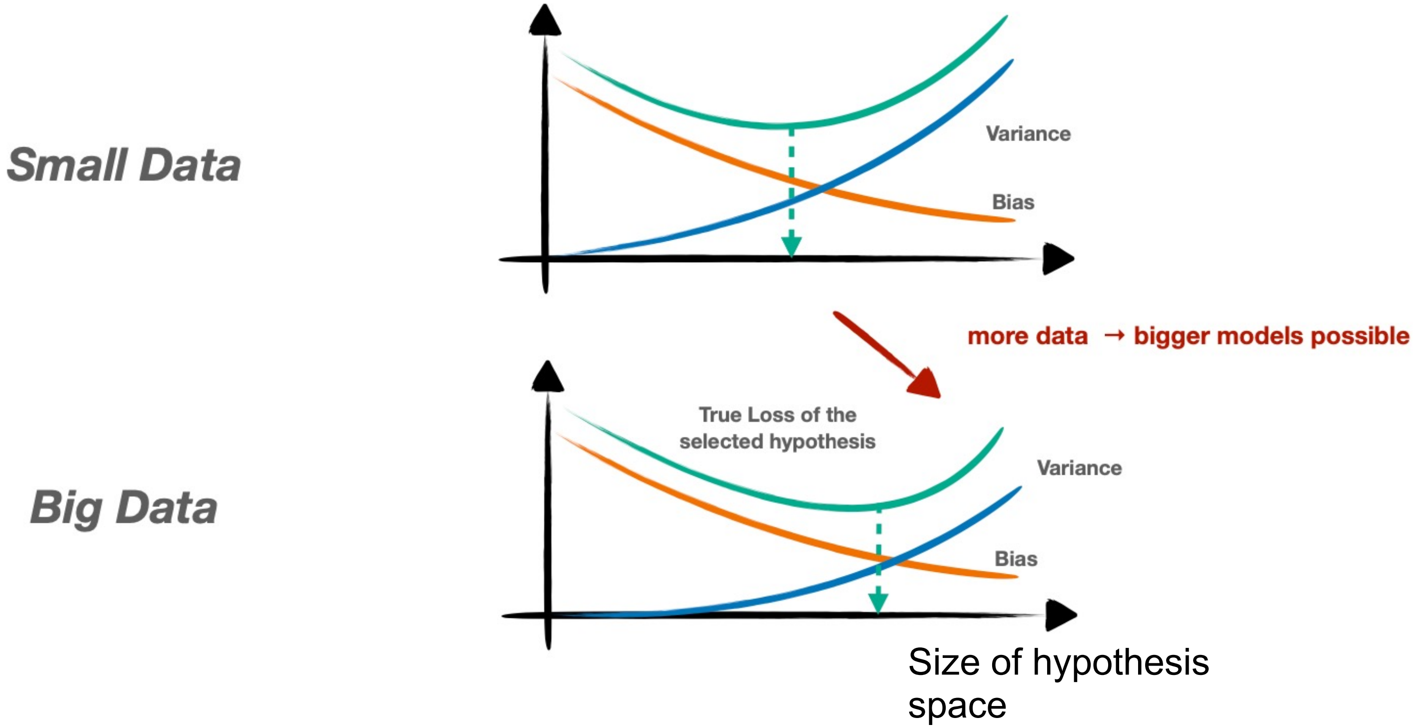
We now have two competing forces

- Make model space as big as possible: reduce bias
- Constrain the model space: reduce variance



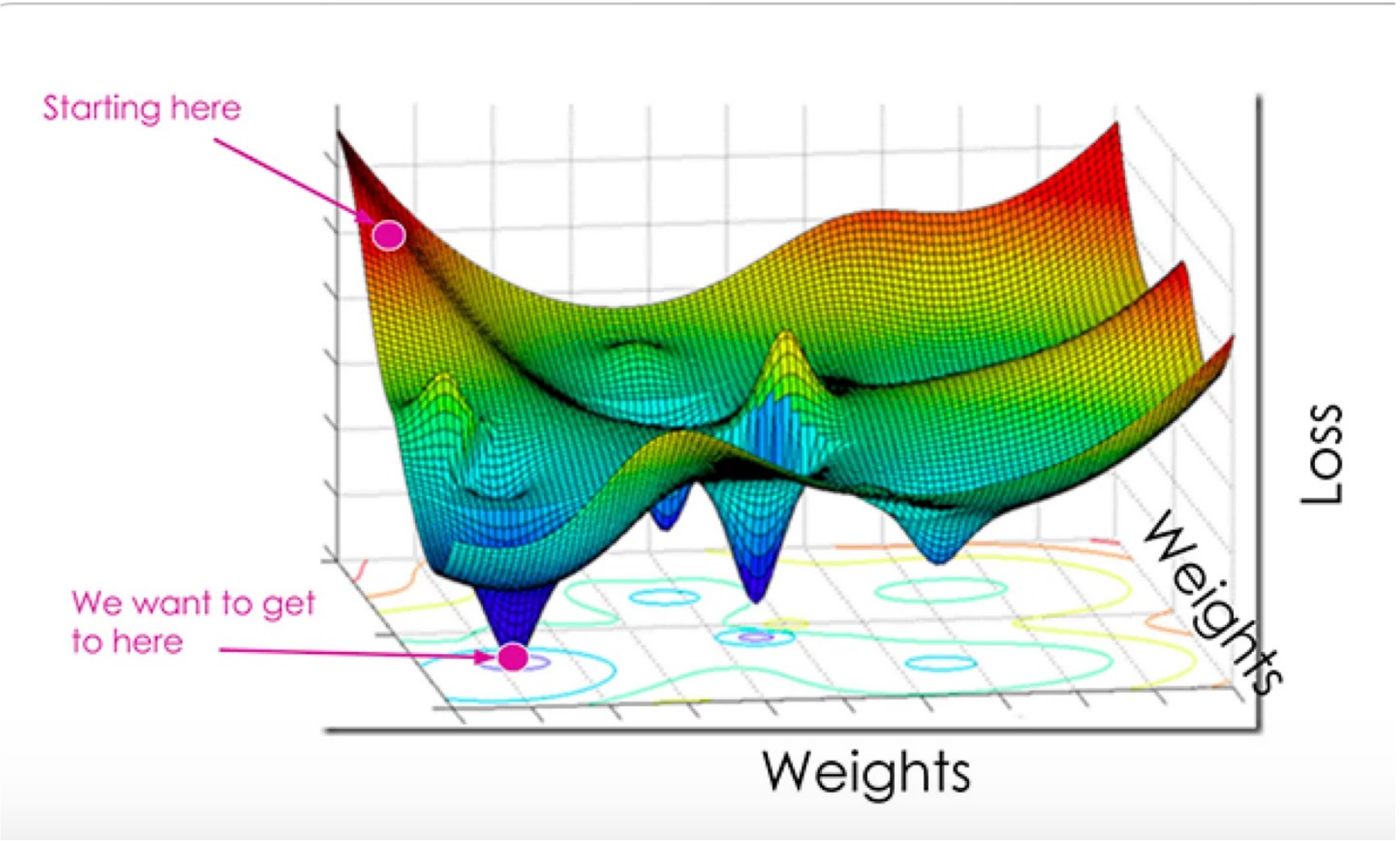
Big networks require big data!

If you don't have enough of it, you simply cannot afford to train a billion parameter model!



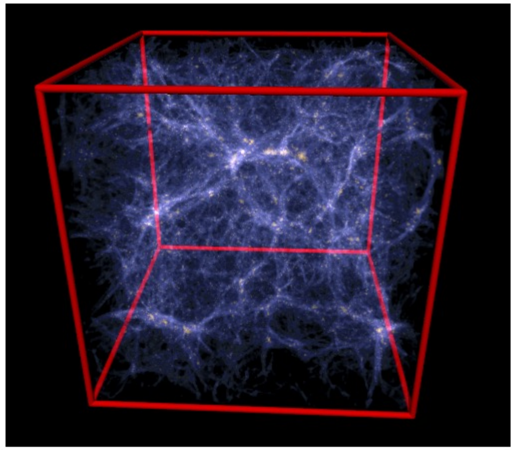
Back-up

Loss becomes also more complex

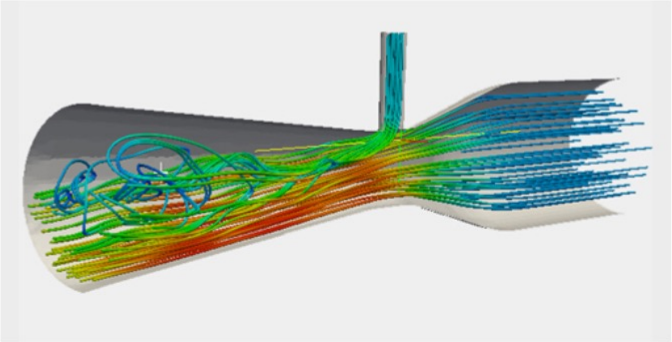


Possible sources of labelled data

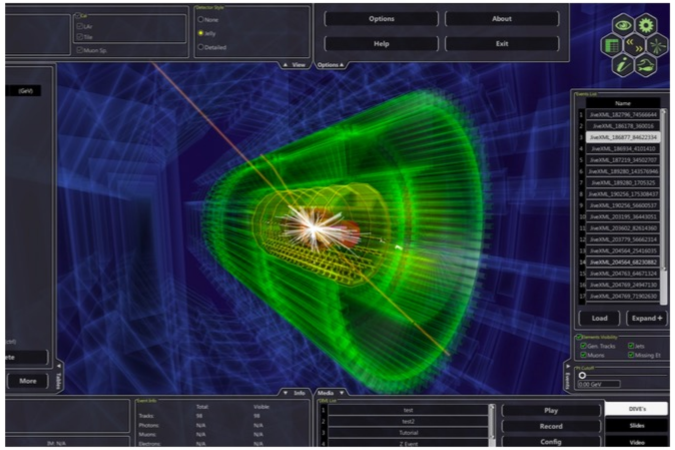
Huge advantage of ML in science: high-fidelity simulators



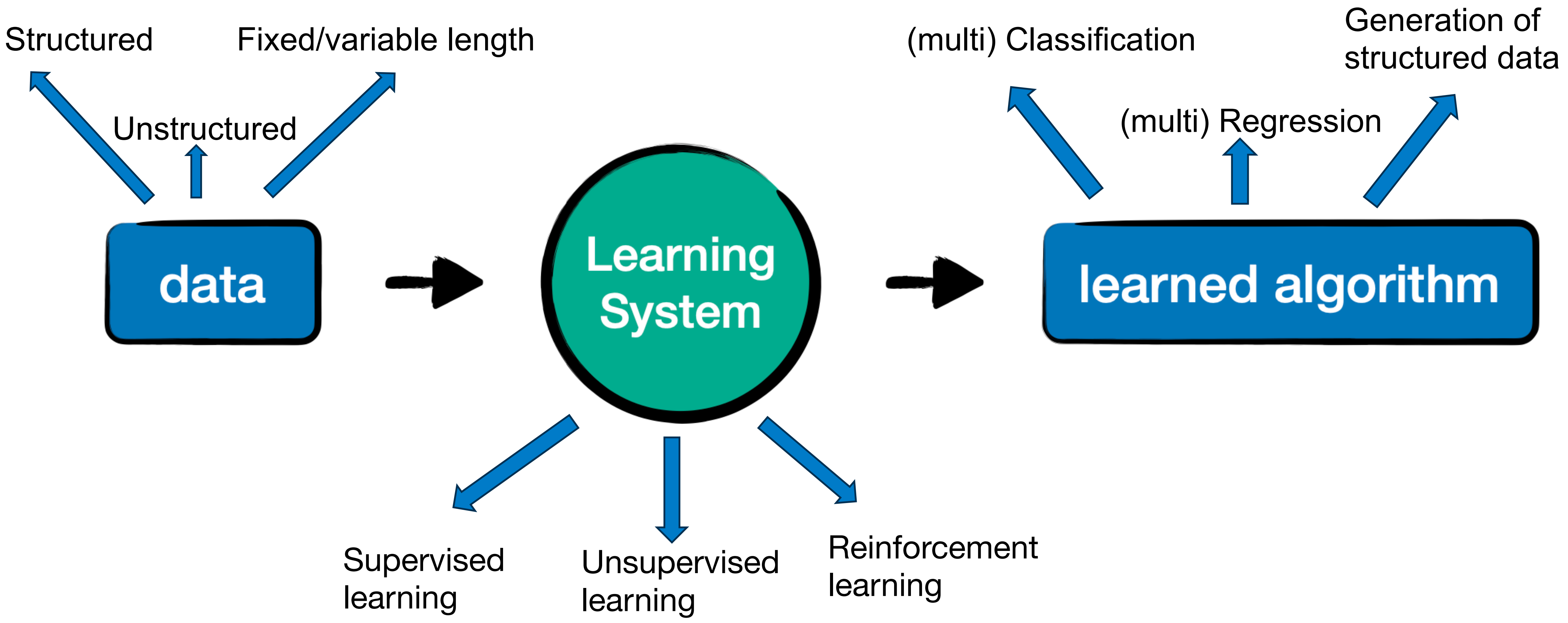
simulated cosmology



simulated fluid dynamics



simulated particle physics


Structured
Fixed/variable length
Unstructured
data
Learning System
(multi) Classification
Generation of structured data
(multi) Regression
learned algorithm
Supervised learning
Unsupervised learning
Reinforcement learning