An Introduction to Machine Learning

House Rules

Stop me whenever you want

No such thing as a silly question, and there is no such thing as a badly timed question

Stop underestimating your knowledge

You don't need to be a mathematical genius. Data Science is about asking the right questions, not knowing the right answers

Stop worrying

This course is a whistle-stop tour of the topic, not a assessed lecture series.

Data Science can be counter-intuitive

It will take practice for it to click

Day 1 Contents

1. What is Machine Learning?
2. What tools do I need?
3. Navigating the Machine learning Landscape
4. ML Primitives
5. Shallow ML: Clustering
6. Shallow ML: Classification

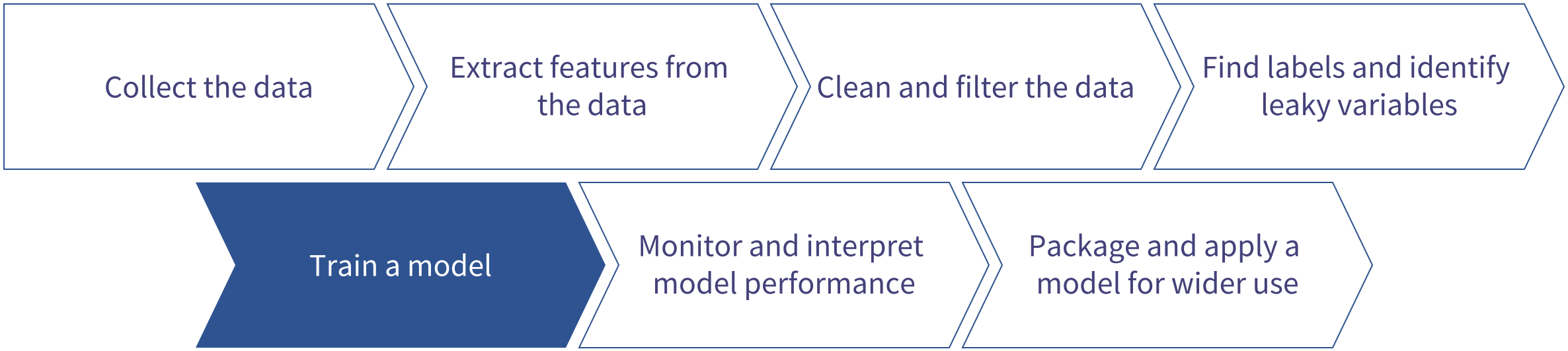
Section 1

What is Machine Learning?

Machine Learning is one element in Data Science

Data Science is the process of turning academic questions into data questions

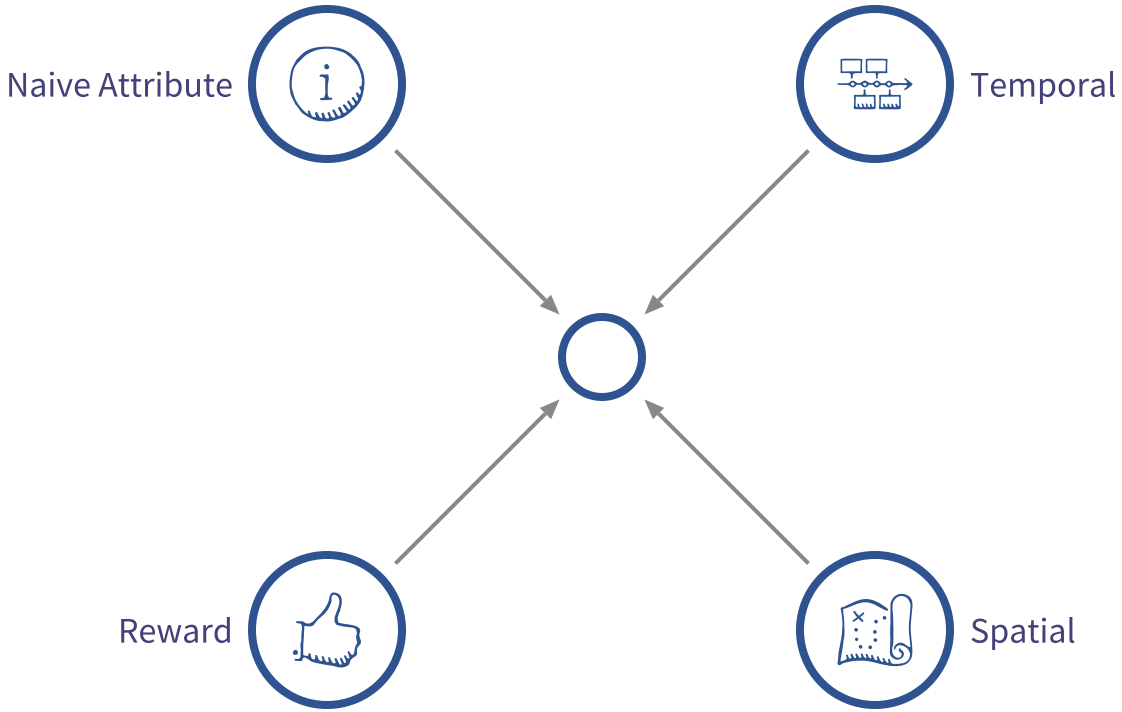
Machine learning is only one piece of the puzzle



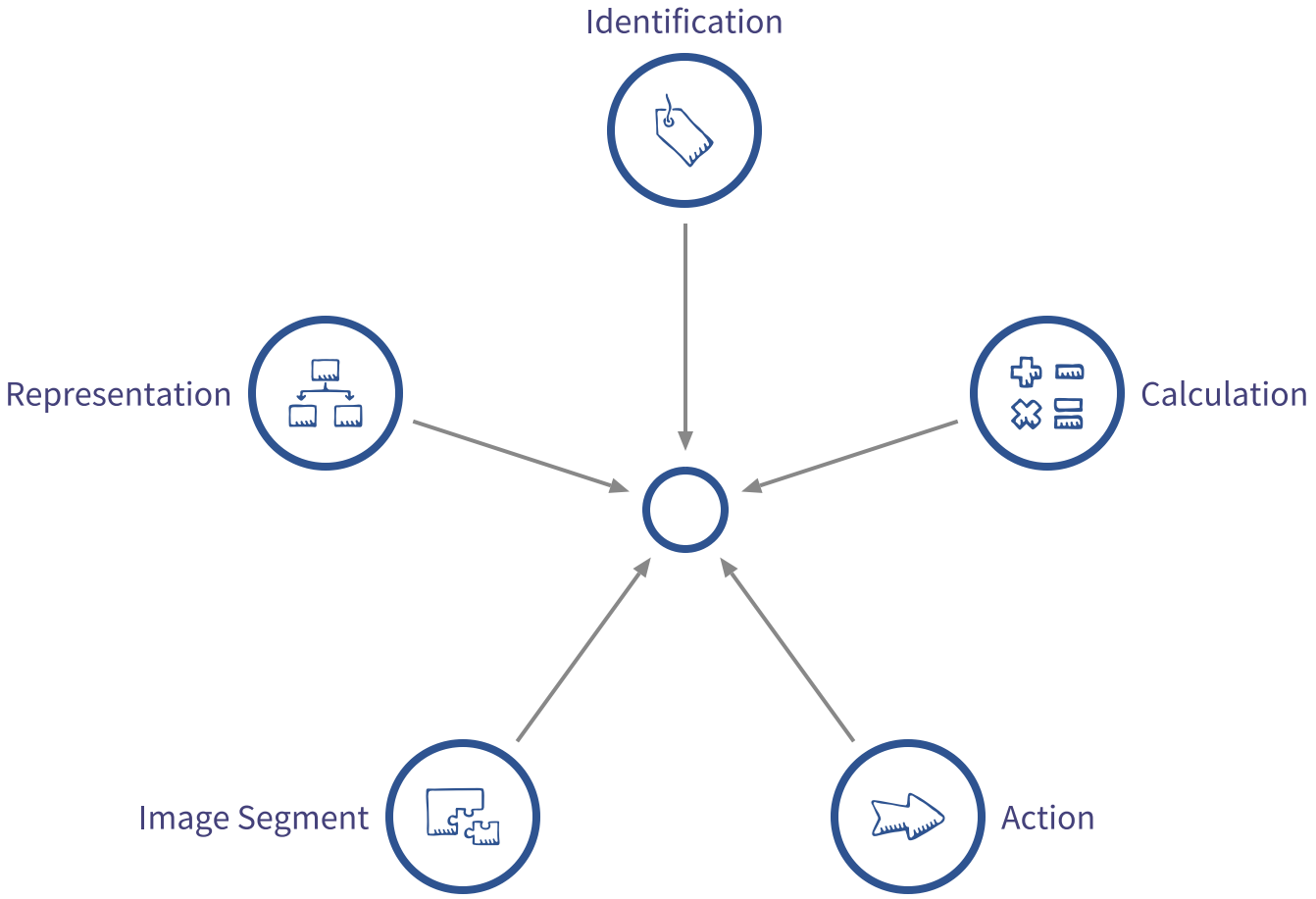
What is a model?

Model primitives

Inputs



Outputs


Identification
Representation
Calculation
Image Segment
Action

Garbage in, garbage out

Your model is only as good as the data it is given

Time taken for a data science project

Machine Learning vs Artificial Intelligence

Machine Learning

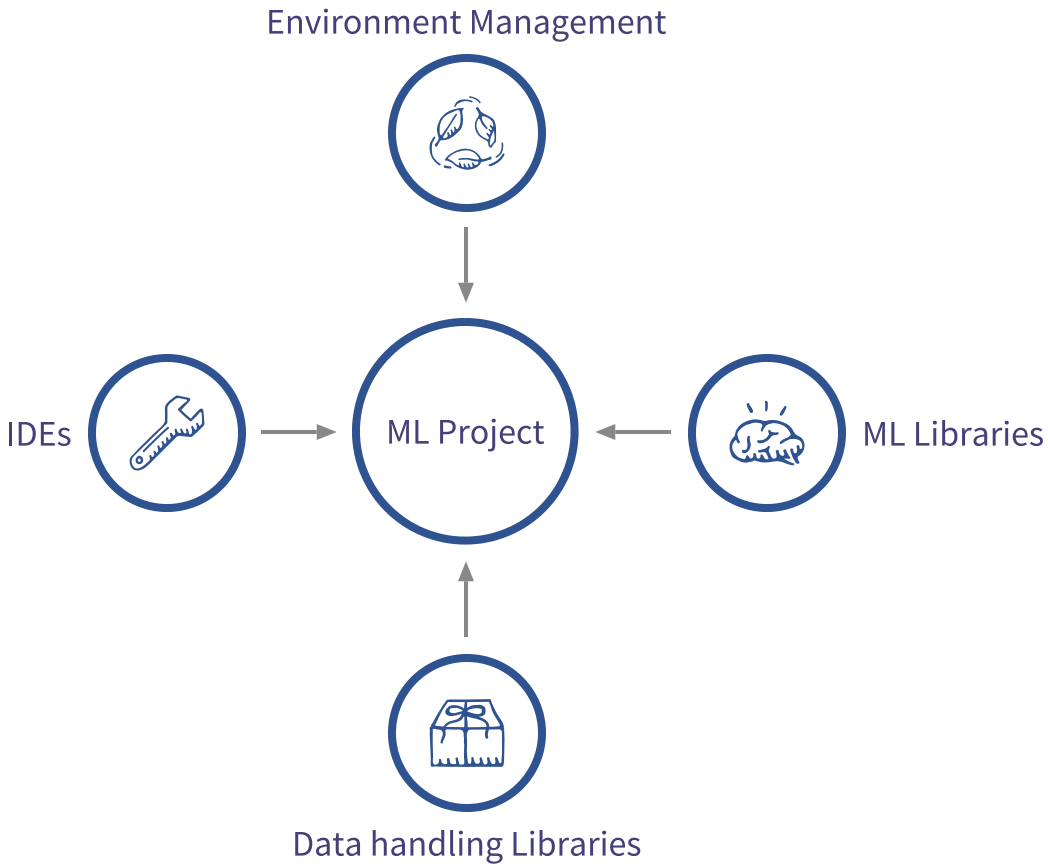
- Machine Learning is the preferred term for data science academics and specialists
- even complex deep learning is considered a “weak” AI
- True “strong” AI is able to understand and apply itself to a variety of unforseen problems and communicate

AI gets grants

Section 2

What tools do I need?

ML Project tooling



Tooling for this course

Environment

PIP

Anaconda

Docker

IDEs

PyCharm

Jupyter

Visual Studio

Atom...

Data Handling

Numpy

Pandas

Dask

Machine Learning

SKLearn

Keras

Tensorflow

Theano

XGBoost

Caffe

Anaconda

- Full virtual environment for python
- Safer than raw pipenvs
- Not as complete, harder to share environments

Jupyter

Julia+Python+R

- A research notebook for data science
- Allows Julia, Python and R code to be mixed
- Markdown supported
- Code is arranged into individual cells with a shared kernel

Numpy and Pandas

- Classic scientific data handling libraries
- Numpy extends the ability of native python to hold complex datatypes needed for scientific data
- Numpy also provides simple data processing functionality
- Pandas (Panel Data) extends pythons ability to store information in R-like dataframes

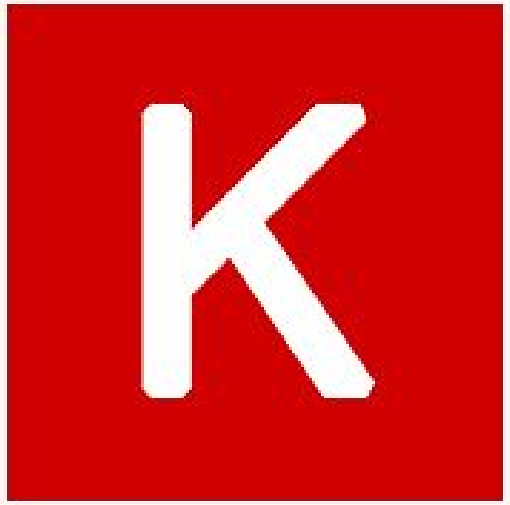
Scikit Learn

sklearn

- A collection of python models and methods for data science
- Great for any non-deep model
- Has data prep and monitoring functionality
- First port-of-call for any ML technique

Keras and TensorFlow

- Keras is a high level API to simply write deep learning models (usually with TensorFlow or Theano)
- TensorFlow is a google tool for writing deep learning code
- Avoids the need to write every method and layer and function individually, when components are largely modular
- Works directly with GPUs



Tooling for this course

Environment

PIP

Anaconda

Docker

IDEs

PyCharm

Jupyter

Visual Studio

Atom...

Data Handling

Numpy

Pandas

Dask

Machine Learning

SKLearn

Keras

Tensorflow

Theano

XGBoost

Caffe

Starting a project

1. Create a new python 3.7 conda environment
2. Install the relevant packages
3. Start Jupyter

Cheat sheet

> conda create -n machinelearning python=3.7

> conda install -c conda-forge [package_name]

packages: jupyter, scikit-learn, pandas, numpy, scipy

Leave Tensorflow and Keras for today

> jupyter notebook

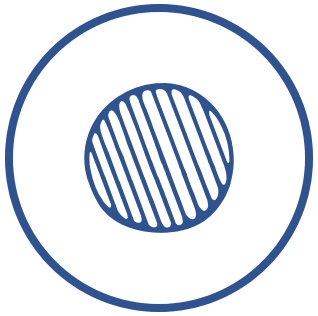
Coffee


TO COMPLETE YOUR REGISTRATION, PLEASE TELL US WHETHER OR NOT THIS IMAGE CONTAINS A STOP SIGN:
STOP
NO
YES
ANSWER QUICKLY—OUR SELF-DRIVING CAR IS ALMOST AT THE INTERSECTION.
SO MUCH OF "AI" IS JUST FIGURING OUT WAYS TO OFFLOAD WORK ONTO RANDOM STRANGERS.

Section 3

Navigating the Machine Learning Landscape

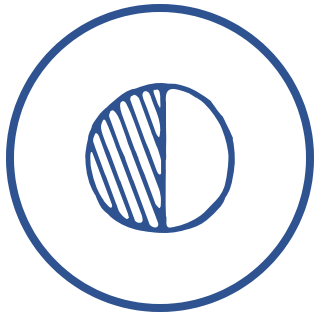
Inputs



Supervised

Every datapoint in the training data has a “label”

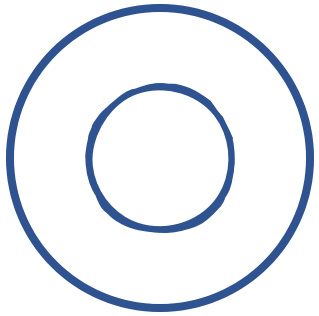
Easy to assess success



Semi-Supervised

Some datapoints in the training set have labels

Harder to assess success



Unsupervised

No datapoints in the training set have labels

Harder to assess success

Reinforcement

Model results change an environment that feeds back to the model

Complex metrics

Outputs

Classification

Item 1 is type B

Item 1 is 70% type A and 30% type B

Item 1 is like items 5, 7, 8

Regression|Calculation

Item 1 is this value

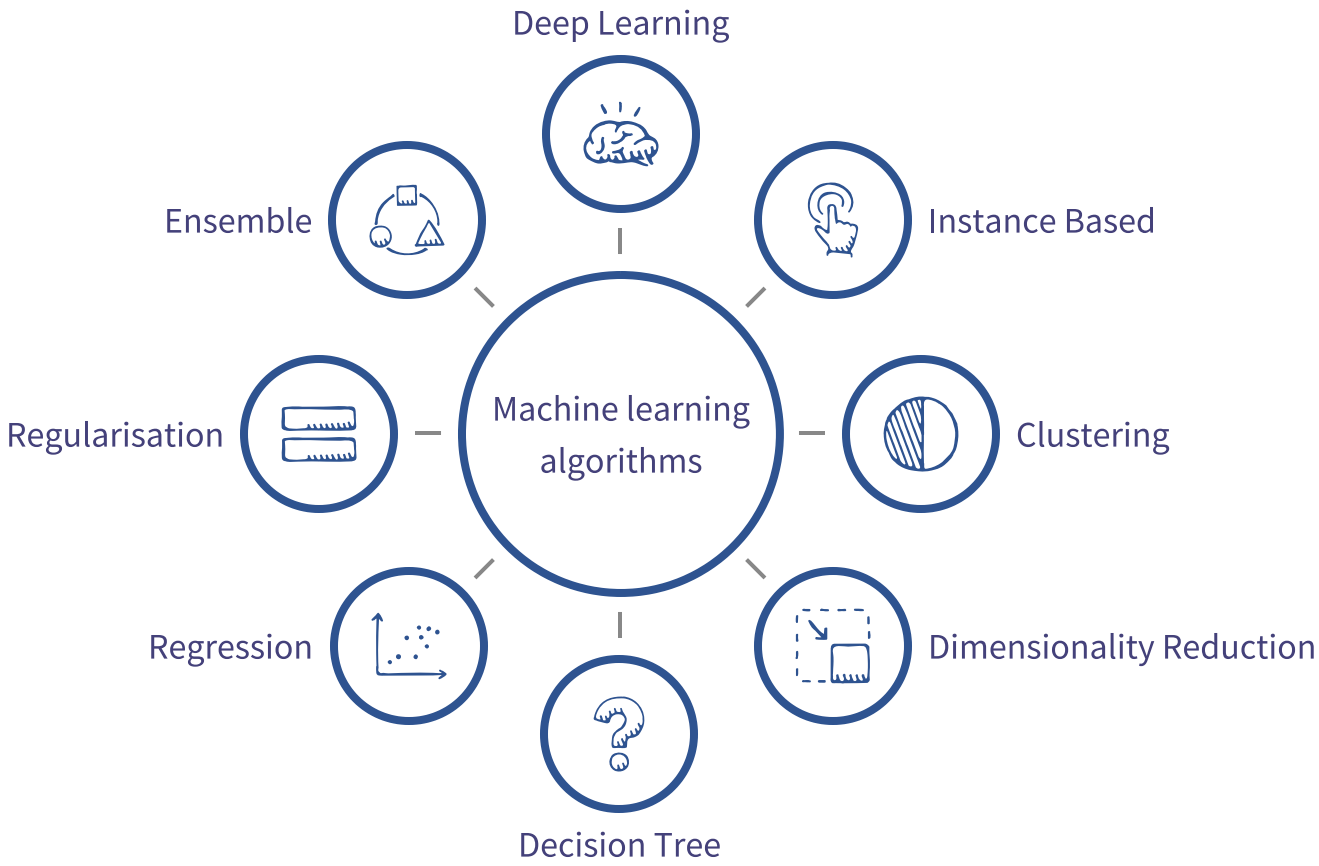
Items 1, 2, 3 are this value

Classification is for discrete categorical outputs

Regression is for continuous outputs

Ordinal discrete outputs require some more thought

Common Algorithms



Decision Tree

Classification

- Typically classifies by constructing a series of quantitative tests
- Often a lower computational cost
- The boundaries and weightings of the quantative tests are tuned iteratively

Classification and Regression Tree

Iterative Dichotomiser 3

Chi-Squared Automatic Interaction Detection

Decision Stump

Conditional Decision Trees

M5, C4.5, C5

Dimensionality Reduction

Classification, Regression, Data Preparation

- Reduces a high dimensional tensor down to a simpler dataset
- Can reduce by selecting the best existing dimensions or crating artificial principal dimensions
- Not always tuned iteratively
- Often part of data prep or part of a more complex model

Principal Component Analysis

Partial Least Squares Regression

Sammon Mapping

Multidimensional Scaling

Projection Pursuit

Principal Component Regression

Partial Least Squares Discriminant Analysis

Mixure/Quadratic/Regularised/Flexible /Linear Discriminant Analysis

Regression

Classification, Data Preparation

- Reduces a complex tensor to an output regressed value
- Can be part of a more complex model
- Good baseline model

Linear Regression

Ordinary Least Squares Regression

Multivariate Adaptive Regression Splines

Locally Estimated Scatterplot Smoothing

Logistic regression

Clustering

Classification, Data Preparation

- Looks at patterns in the distribution of datapoints
- Tries to find and organise collections of datapoints into common clusters
- Good baseline model
- Recommended first step for unsupervised problems

K-Means

k-Medians

Expectation Maximisation

Heirachical clustering

X-means

Affinity propagation

Mean Shift

Specral Clustering

DBScan

Regularisation

Regression, Data Preparation

- Corrects for poorer datasets by using additional “beta” weights on each feature
- May result in some dimensionality reduction
- Changes how the model defines its success

Ridge Regression

Least Absolute Shrinkage and Selection Operator

Elastic Net

Leasst Angle Regression

Instance Based

Classification

- Higher computational cost approach
- Creates hypotheses by datapoint or instance, and compares it to previous hypotheses in memory - “walks” to a optimal solution
- Also known as In-Memory based optimisation
- Valuable for unsupervised models

k-Nearest Neighbour

Learning Vector Quantisation

Self-Organising Map

Locally Weighted Learning

Rule System

Classification

- Similar to Decision trees. Sorts inputs down to a range of outputs
- Uses a series of rules rather than quantitative thresholds
- May be more robust than decision trees - results vary

Cubist

One Rule

Zero Rule

Repeated Incremental Pruning to Produce Error Reduction (RIPPER)

Deep Learning

Classification, Regression

- Uses internal “Hidden” layers to introduce cognitive flexibility to how it interprets data
- Can be computationally expensive
- Typically a later model
- **More on these tomorrow**

Deep Boltzmann Machines

Deep Belief Networks

Multi-Layer Perceptrons

Variational Auto-Encoders

Recurrent Neural Networks

Convolutional Neural Networks

Generative Adversarial Networks

Ensemble model

Classification, Regression

- Generic term for a model consisting of multiple component models
- More diverse models usually yield better and more stable results
- component models can be chained; or run in parallel or competition

Random Forests

Gradient Boosting Machines

Boosting

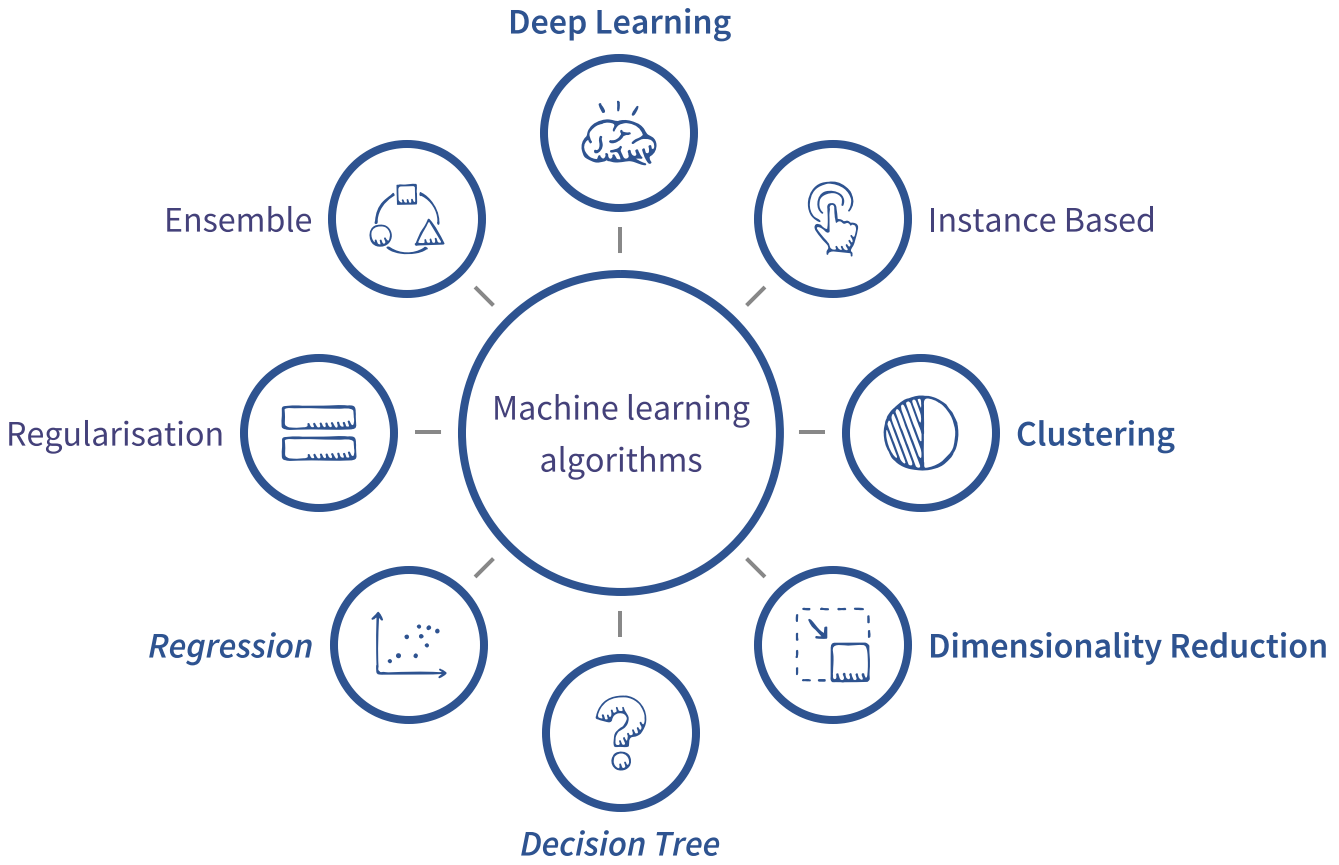
Bootstrapped Aggregation

AdaBoost

Stacked Generalisation

Gradient Boosted Regression Trees

Common Algorithms



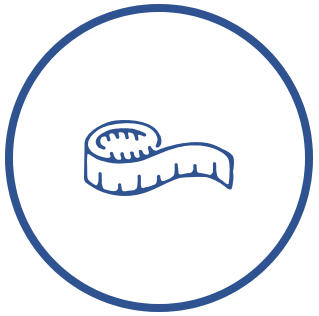
Section 4

Machine Learning Primitives

Questions

Correctness

Entropy



Fitting

Bias-Variance tradeoff

Objectives

Objectives

Models takes inputs, provide outputs, and take steps to improve itself

To improve a model, we need a way to measure its performance

This can be simply accuracy, or more complex statistical measurements

This measurement is the **objective.** A model seeks to optimise an objective.

Loss

The difference between the model performance and maximum performance is the “loss” of the model.

Typical objective for most models

Multiple ways to define and compute loss

Loss definition depends on the behaviour you want to encourage

e.g. Min-Max and Invariance

Loss is not accuracy

Why is accuracy not always suitable?

Correctness

	Is Yes	Is No
Predict Yes	True Positive	False Positive
Predict No	False Negative	True Negative

Correctness

Identifying dogs in pictures

	Is a dog	Is not a dog
Predict a dog	True Positive	False Positive
Predict not a dog	False Negative	True Negative

False Positive: **Type 1 Error** The photo didn't have a dog but we predicted that it did
False Negative: **Type 2 Error** The photo had a dog but we predicted that it didn't

Correctness

Identifying dogs in pictures

	Is a dog	Is not a dog	total
Predict a dog	70	4,930	**5,000**
Predict not a dog	13,930	981,070	**995,000**
total	14,000	986,000	**1,000,000**

What is the test accuracy?

Correctness

Identifying dogs in pictures

	Is a dog	Is not a dog	total
Predict a dog	70	4,930	**5,000**
Predict not a dog	13,930	981,070	**995,000**
total	14,000	986,000	**1,000,000**

Accuracy = # Correct / Total
(70+981,070) / 1,000,000 = **0.98114**

Correctness

Precision and recall

Precision: how accurate the positive predictions are

True Positives / (True and False Positives)

=**0.014**

Recall: how many of the positives were correctly predicted

True Positives / (True Positives + False Negatives)

=**0.005**

Precision and recall often are traded off against each-other

A model that over identifies has good precision but poor recall

A model that is too cautious has poor precision but good recall

Correctness

F1 score

F1 score: the harmonic mean of precision and recall

(2 x Precision x Recall) / (Precision + Recall)

=**0.00736...**

Overfitting

- “Local over-optimisation resulting in poorer global performance”
- A model can find arbitrary rules that work in a training set but don't apply in a wider application.
- The accuracy and loss still improve, but the model ends up working poorly outside the training set.

Correctness

Bias-Variance trade-off

You retrain a model multiple times on different datasets...

1) the models appears to be fairly similar, but fits poorly

High Bias, Low Variance

2) the models vary, but strongly fit the training data

Low Bias, High Variance

The ideal model will generate the same well-fitting model no matter the starting data

Correctness

	High Variance	Low Variance
High Bias	Uh-Oh	**Underfitting**
Low Bias	**Overfitting**	Goldilocks zone

Fix **high bias** by adding more features
Fix **high variance** by removing features or adding data

Entropy

- An abstract measure of the randomness of information
- Does an attribute have meaning?
- A coin flip's output is a high-entropy attribute
- A weighted coin flip's output is a low-entropy attribute

Cross Entropy

- Common loss function for classification
- Measures the difference in distribution of expected (p) and actual (q) outputs
- Underpins logistic regression

Lunch


THE #1 DATA SCIENTIST EXCUSE FOR LEGITIMATELY SLACKING OFF:
"MY MODEL'S TRAINING"
HEY! GET BACK TO WORK!
TRAINING!
OH. CARRY ON.

Section 5

Decision Trees and Random Forests

Decision Trees

- “20 questions” approach to classification
- A strong baseline model
- Can be inflexible with new data
- Divides up datapoints by an attribute per branch

Decision Trees

What divides the dataset up best?

Attributes

Quadruped

Barks

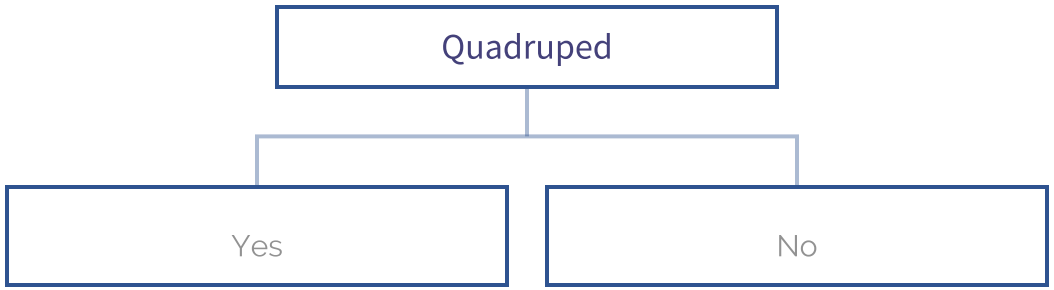
Chases laser pointers

Drinks tea

Has fur

Classes: Human, Chimpanzee, Dog, Tabby cat, Sphynx cat

Decision Trees



Attributes

Quadruped

Barks

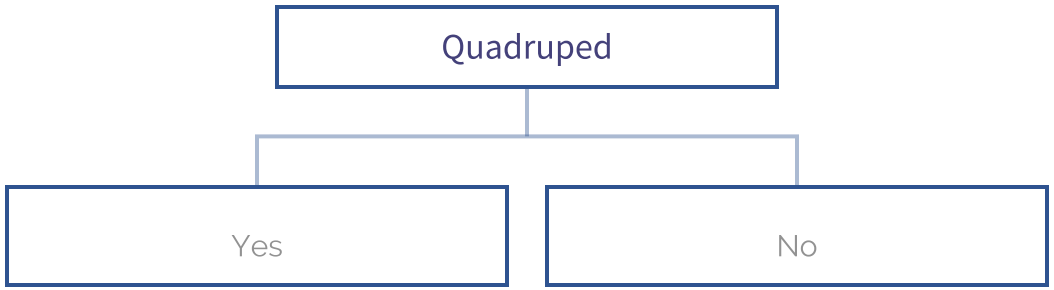
Chases laser pointers

Drinks tea

Has fur

Classes: Human, Chimpanzee, Dog, Tabby cat, Sphynx cat

Decision Trees



Attributes

Quadruped

Barks

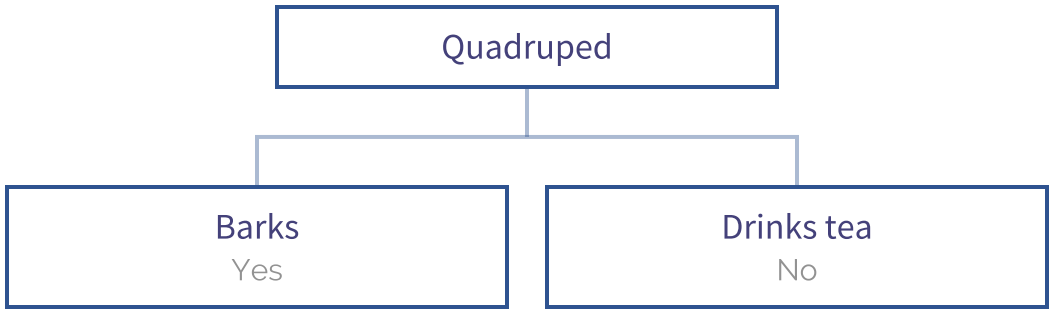
Chases laser pointers

Drinks tea

Has fur

Classes: Human, Chimpanzee, Dog, Tabby cat, Sphynx cat

Decision Trees



Attributes

Quadruped

Barks

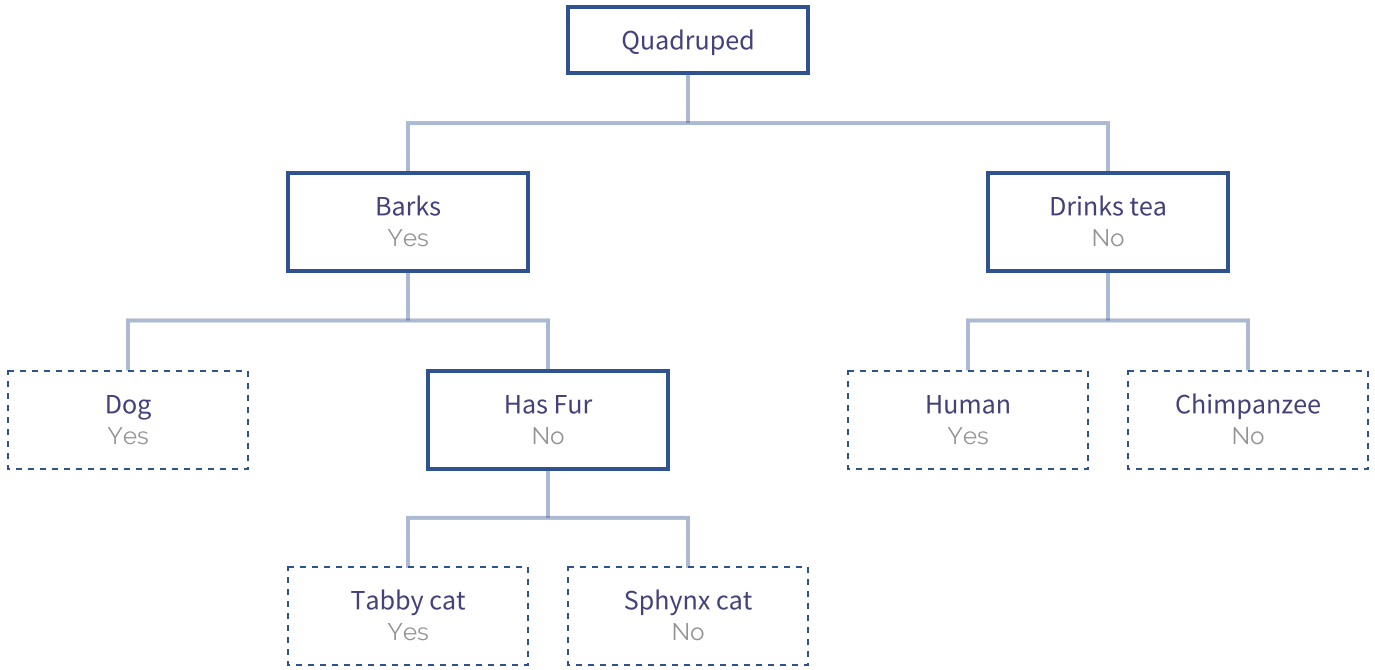
Chases laser pointers

Drinks tea

Has fur

Classes: Human, Chimpanzee, Dog, Tabby cat, Sphynx cat

Decision Trees



Attributes

Quadruped

Barks

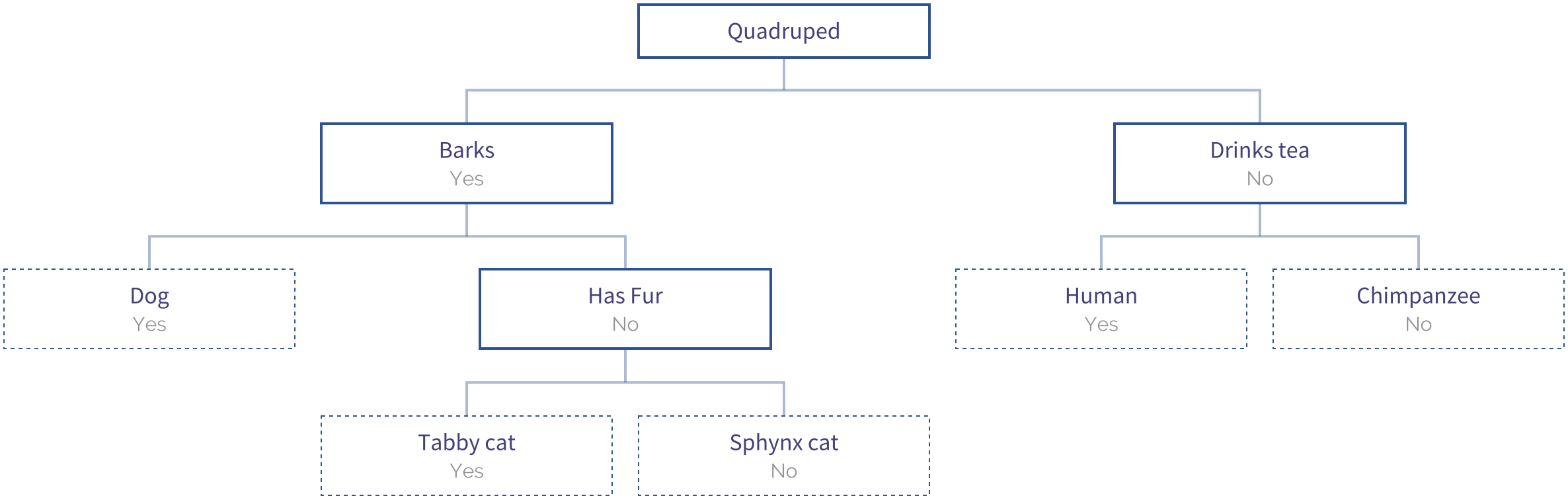
Chases laser pointers

Drinks tea

Has fur

Classes: Human, Chimpanzee, Dog, Tabby cat, Sphynx cat

Decision Trees



Does this work for every case?

Random Forests

Many varied decision trees

Each decision tree only uses a subset of attributes at each stage

Bootstrapping

The dataset is not complete, and items occasionally repeat

Aggregate voting

Results from each tree vote, and the overall answer is the ensemble answer

Random Forests

Has Fur

Attributes

Quadruped

Barks

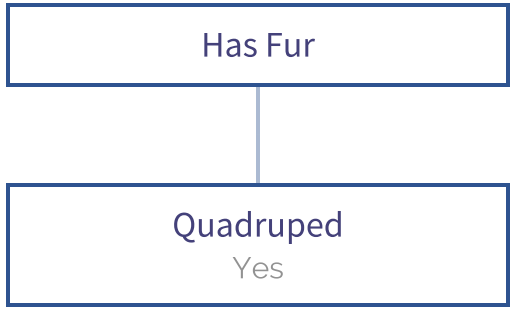
Chases laser pointers

Drinks tea

Has fur

Classes: Human, Chimpanzee, Dog, Tabby cat, Sphynx cat

Random Forests



Attributes

Quadruped

Barks

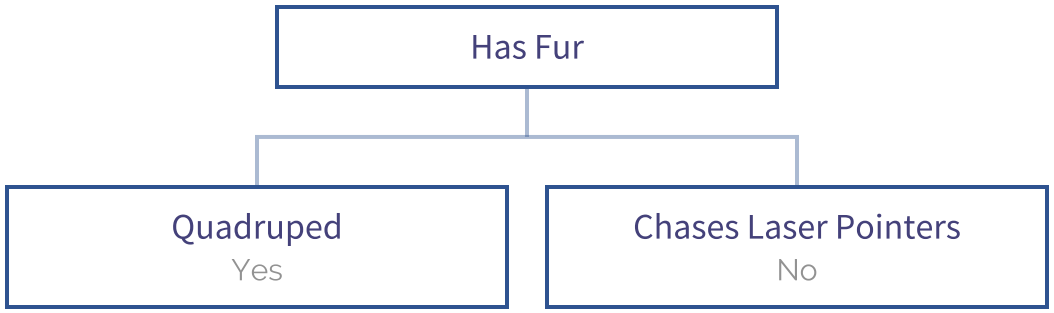
Chases laser pointers

Drinks tea

Has fur

Classes: Human, Chimpanzee, Dog, Tabby cat, Sphynx cat

Random Forests



Attributes

Quadruped

Barks

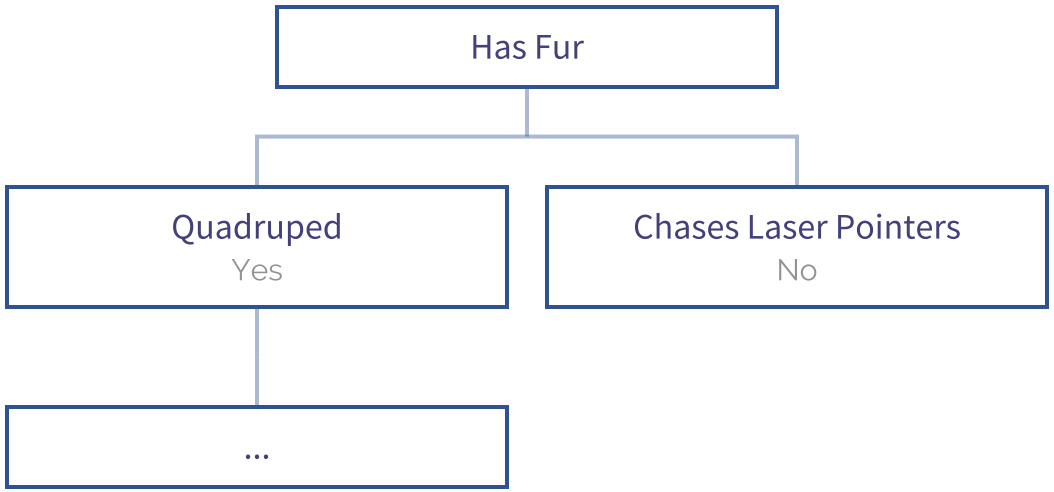
Chases laser pointers

Drinks tea

Has fur

Classes: Human, Chimpanzee, Dog, Tabby cat, Sphynx cat

Random Forests



Attributes

Quadruped

Barks

Chases laser pointers

Drinks tea

Has fur

Classes: Human, Chimpanzee, Dog, Tabby cat, Sphynx cat

Random Forests

1. Create your bootstrap data subset
2. Pick from the random attribute options for your tree
3. Repeat this choice of attribute subsets for multiple trees
4. Run the training data through all the trees
5. Aggregate the results, return the modal answer

Bootstrapping

- For n datapoints, pick n rows. Rows can be picked multiple times
- Normally repetition makes a bad dataset, but we want to introduce heterogeneity and noise
- This adds a second point of randomness to the model
- Rows not picked (typically 1/3 in larger datasets, aka “out of bag”) are used for validation

Bootstrapping + Aggregation = “Bagging”

Random Forests

1. Create your bootstrap data subset
2. Pick from the random attribute options for your tree
3. Repeat this choice of attribute subsets for multiple trees
4. Run the training data through all the trees
5. Aggregate the results, return the modal answer

Hints

The IRIS dataset is available through the sklearn library. No need to download it separately

The GitHub notebook will show you how to put together your model

You will need 30% validation data put to one side before using RF

Think about which metrics you'll need

Random Forests is often the second-best solution to any problem

Coffee


THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

Section 6

Clustering

Clustering

- Typically an unsupervised approach - ideal for data exploration
- Can have semisupervised datasets too
- Not all clustering alogrithms are created equally

k-means

- “naive” k-means has existed for ~60-70 years
- the default clustering algorithm
- most implementations of k-means use the “k-means ++” version of the algorithm
- **Requires you to know how many clusters are in your data already**

  If not, consider x-means algorithms

K-means process

1. create “k” number of centroids at random values
2. Every datapoint finds its nearest centroid
3. The means of the distances are calculated, and the centroids drift to a different value to reduce the overall mean
4. repeat until the total means plateau

link

K-means

- The end model may not be completely optimised - highly reliant on starting points

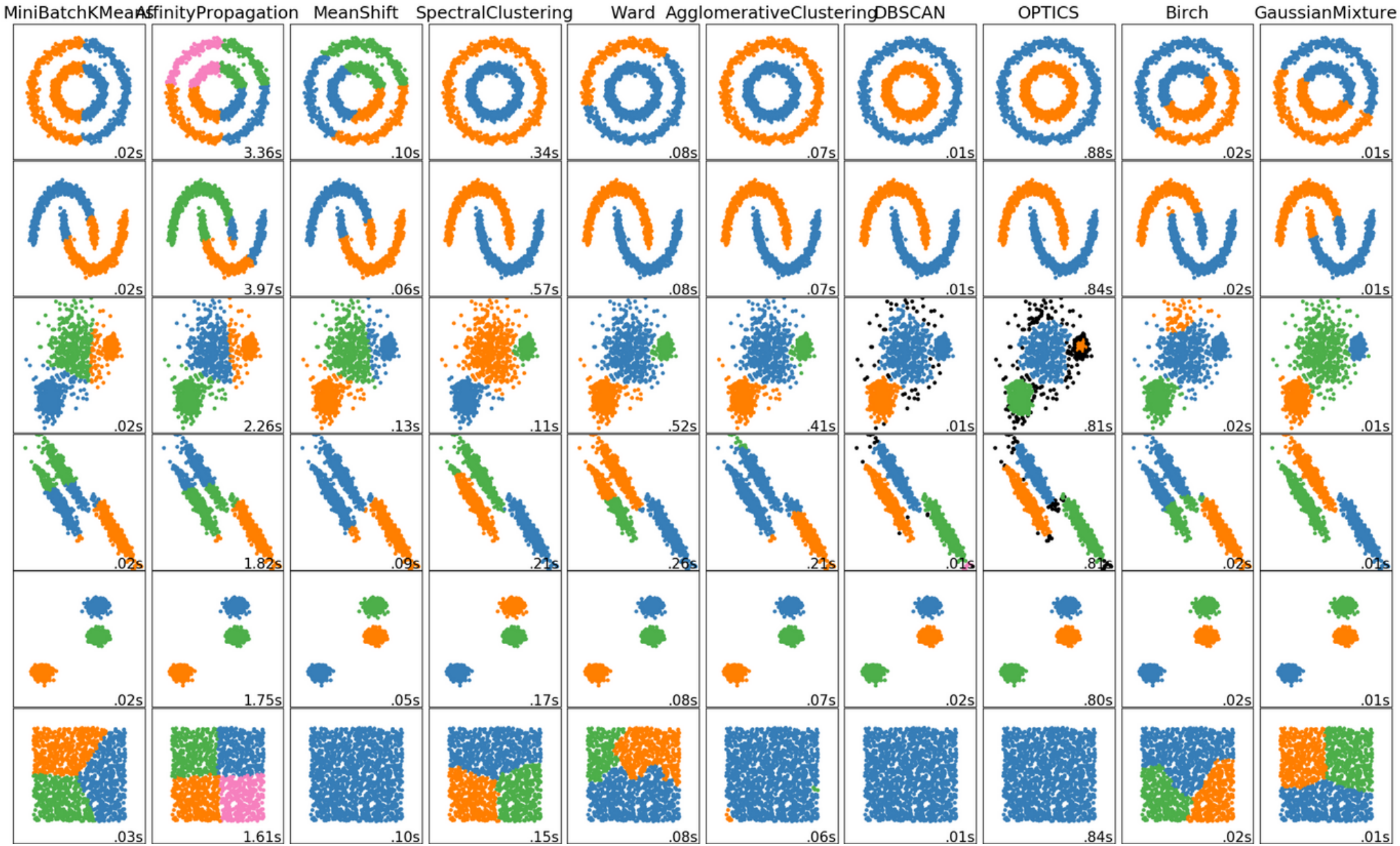
  k-Means ++ picks starting locations more carefully

- There may be hidden sub-clusters that weren't identified that add noise

- May be skewed towards outliers with smaller datasets

  k-Medians, and k-Mediods change this behaviour, with their own advantages and drawbacks

Visualise your data before you train on it


MiniBatchKMeans
AffinityPropagation
MeanShift
SpectralClustering
Ward
AgglomerativeClustering
DBSCAN
OPTICS
Birch
GaussianMixture
.02s
3.36s
.10s
.34s
.08s
.07s
.01s
.88s
.02s
.01s
.02s
3.97s
.06s
.57s
.08s
.07s
.01s
.84s
.02s
.01s
.02s
2.26s
.13s
.11s
.52s
.41s
.01s
.81s
.02s
.01s
.02s
1.82s
.09s
.21s
.26s
.21s
.01s
.81s
.02s
.01s
.02s
1.75s
.05s
.17s
.08s
.07s
.02s
.80s
.02s
.01s
.03s
1.61s
.10s
.15s
.08s
.06s
.01s
.84s
.02s
.01s

K-means exercise

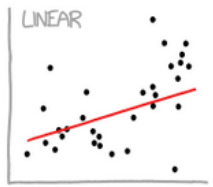
- Use the Wines Dataset from SKlearn
- Visualise when you can - either with physical axes or PCA axes
- Use GitHub for the example code

Recap

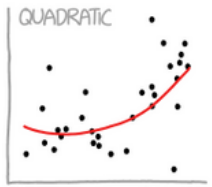
- ML Tooling
- ML Input Types
- ML Output Types
- Decision Trees
- Dimensionality Reduction
- Regression
- Clustering
- Regularisation
- Instance based learning
- Deep Learning
- Ensemble
- Objectives
- Loss
- Correctness
- Entropy
- Random Forest
- K-Means ++

Tomorrow: Deep Learning

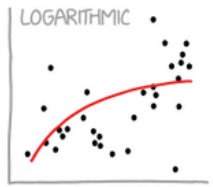
CURVE-FITTING METHODS
AND THE MESSAGES THEY SEND



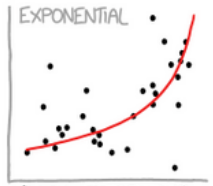
"HEY, I DID A REGRESSION."



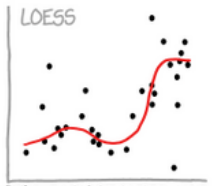
"I WANTED A CURVED LINE, SO I MADE ONE WITH MATH."



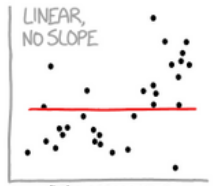
"LOOK, IT'S TAPERING OFF!"



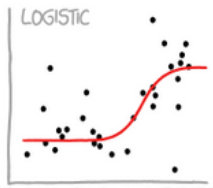
"LOOK, IT'S GROWING UNCONTROLLABLY!"



"I'M SOPHISTICATED, NOT LIKE THOSE BUMBLING POLYNOMIAL PEOPLE."

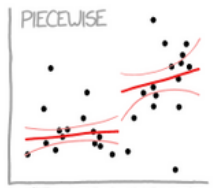


"I'M MAKING A SCATTER PLOT BUT I DON'T WANT TO."

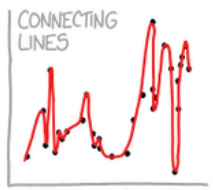


"I NEED TO CONNECT THESE TWO LINES, BUT MY FIRST IDEA DIDN'T HAVE ENOUGH MATH."

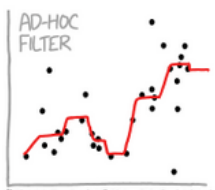
"LISTEN, SCIENCE IS HARD. BUT I'M A SERIOUS PERSON DOING MY BEST."



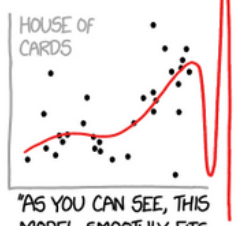
"I HAVE A THEORY, AND THIS IS THE ONLY DATA I COULD FIND.



"I CLICKED 'SMOOTH LINES' IN EXCEL."



"I HAD AN IDEA FOR HOW TO CLEAN UP THE DATA. WHAT DO YOU THINK?"



"AS YOU CAN SEE, THIS MODEL SMOOTHLY FITS THE– *WAIT NO NO DON'T EXTEND IT AAAAAA!!*"

An Introduction to Machine Learning

Day 2

Day 2 outline

1. Ethics and AI
2. What is Deep Learning?
3. Multi-layer perceptrons
4. Convolutional Neural Networks

Section 1

Ethics and AI

Personal Data Mining

AI Shortcomings

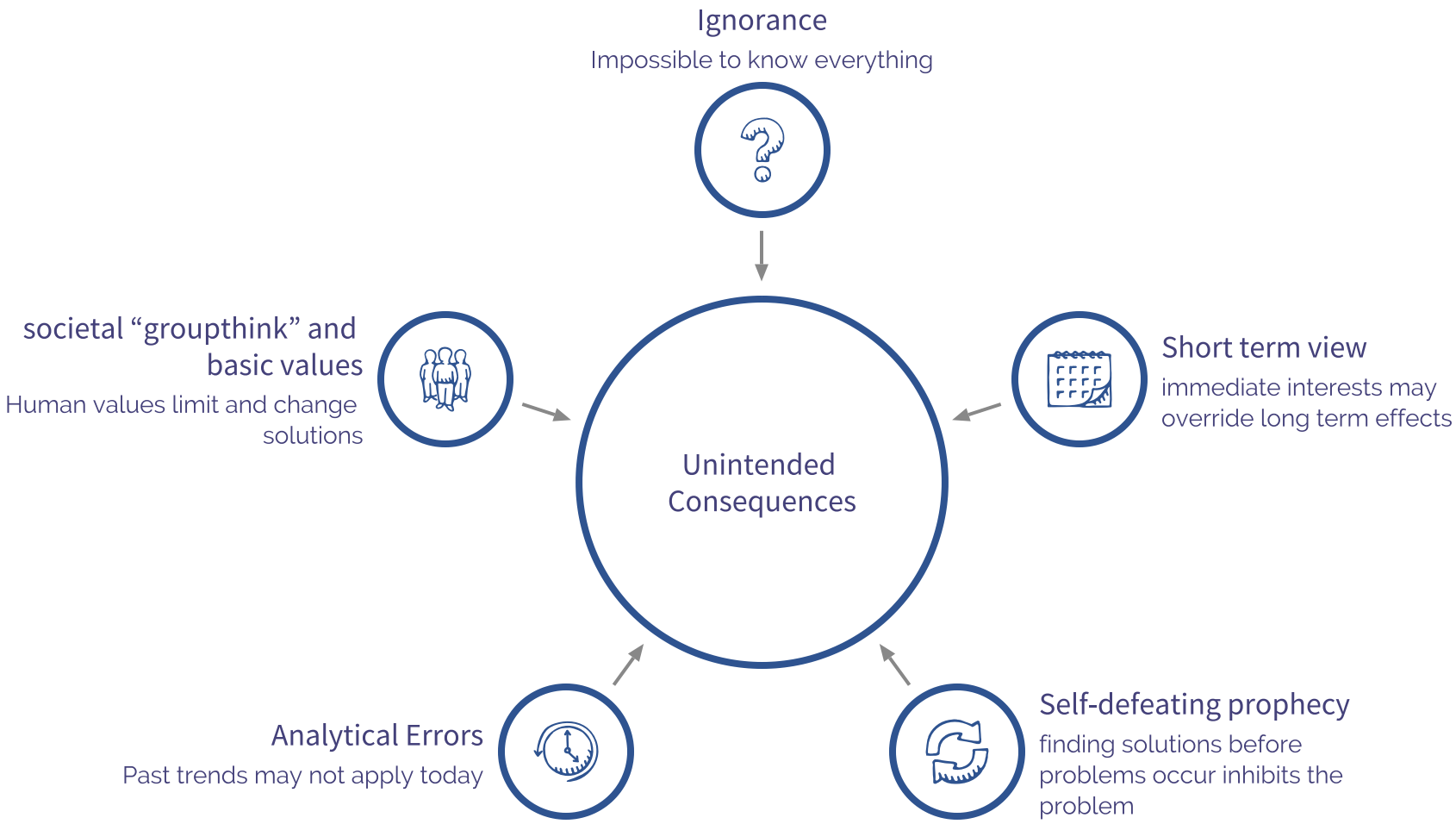
- Vulnerable to reproducing inherent biases
- Can leak personal data

  Heavily reliant on data quality and munging (Mash Until No Good)
- Provides new ways to label and therefore discriminate
- Quantitative analysis without context

The law of unintended consequences

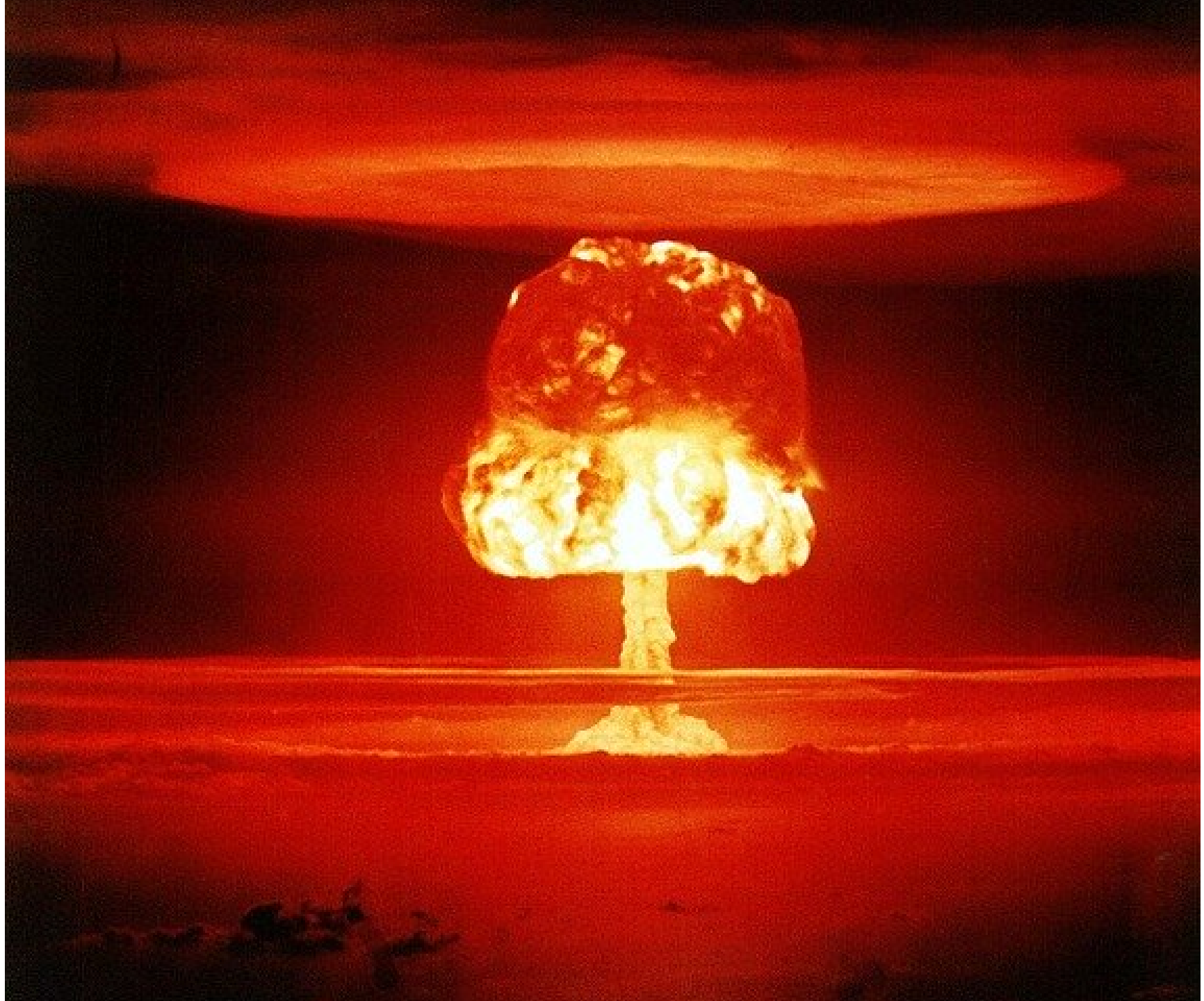
Unexpected benefits, unexpected drawbacks and perverse results

Causes of unintended consequences



Examples

- Computer Vision technology is allowing China to track and identify ethnic minorities
- NASA research for identifying improvised hospitals and schools can be used to identify illegal settlements
- AI assisted Robotics for use in space, safer cars and medical devices is being used for smarter drones and missiles



Changing human interaction

Phones, Internet and Communication

- The internet has changed the nature of political debate and groupthink
- Technology allows malicious interaction with a populous

  vis-a-vis Cambridge Analytica
- Smarter advertising and app design is reinforcing phone addition and consumer control
- Humanity is now using smarter cars, voice assistants, doorbells...

  How will these new ways to interface with information change how we behave?

Will we use these interactions and dependencies to encourage and reward beneficial behaviour?

Will we use the technology to pursue political or commercial benefits instead?

technological change has historically changed the distribution of wealth

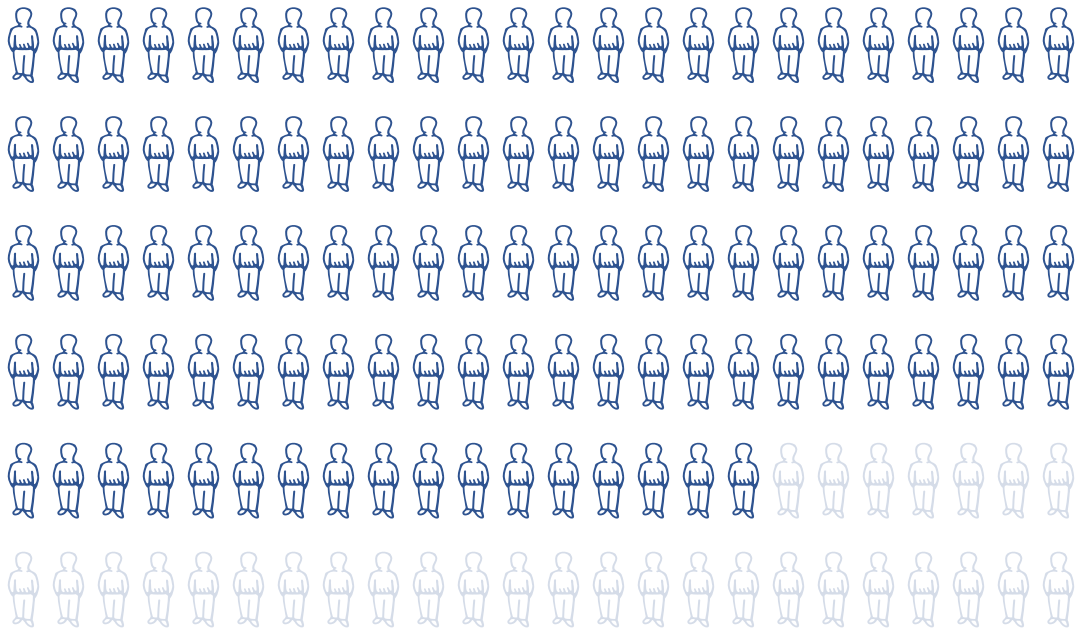
AI in the economy

- The crux of a capitalist system is money in exchange for a service
- AI enabled manufacturing is resulting in up to x10 reductions in workforce. Much of the savings are concentrated in the tech companies that created the technology.
- How do those out of work survive, or thrive
- What does a post-work civilisation look like? Will it be fair?

Workforce at risk

McKinsey&co analysis - by percentage of process time at risk

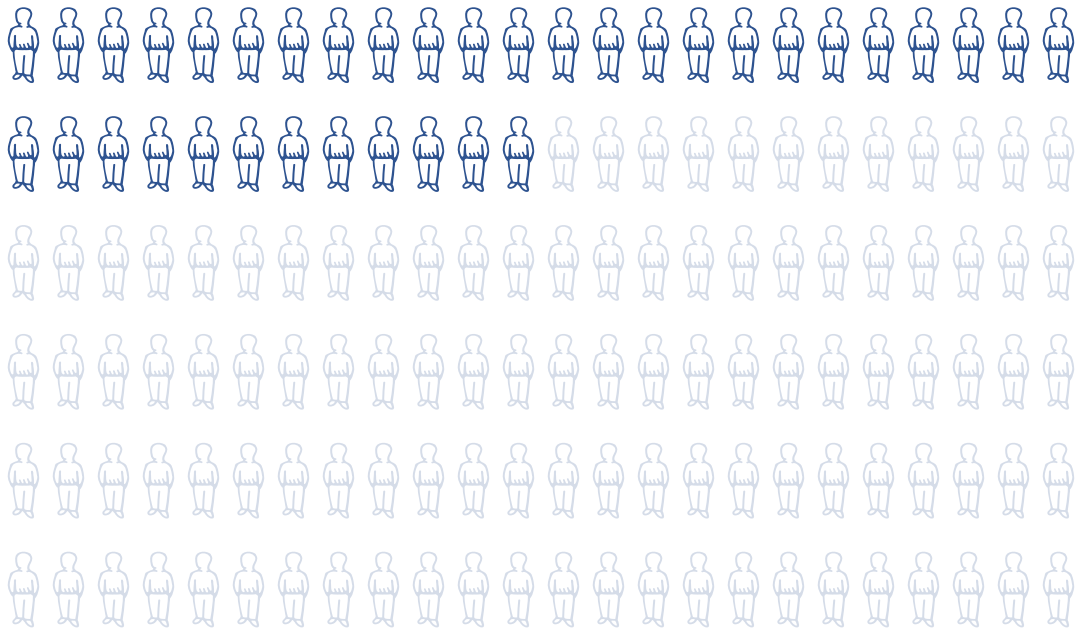
78%



Predicable physical work

Assembly line work, food preparation, packaging

25%



Unpredictable physical work

Construction, livestock farming, forestry

Who is responsible for the decisions an AI makes?

Moral Agents

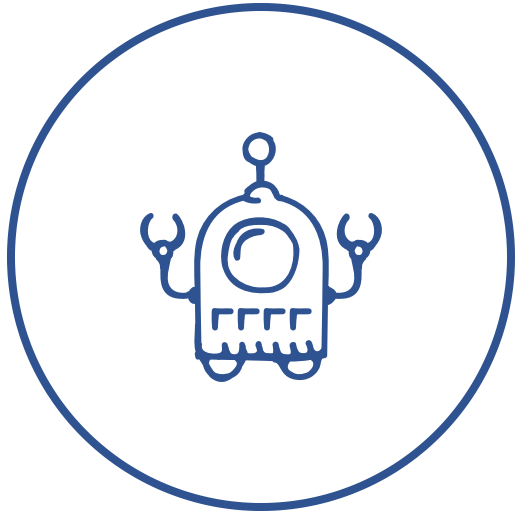
A moral agent is a person who has the ability to discern right from wrong and to be held accountable for his or her own actions. Moral agents have a moral responsibility not to cause unjustified harm.

- Historically, actors capable of decision making are moral agents
- One key exception is a child
- Another is a company

  But this is changing

- We have never delegated decision making so much before

Who is responsible for a morally wrong decision?



The AI

Requires it to have morality scores and complex reasoning

The creator

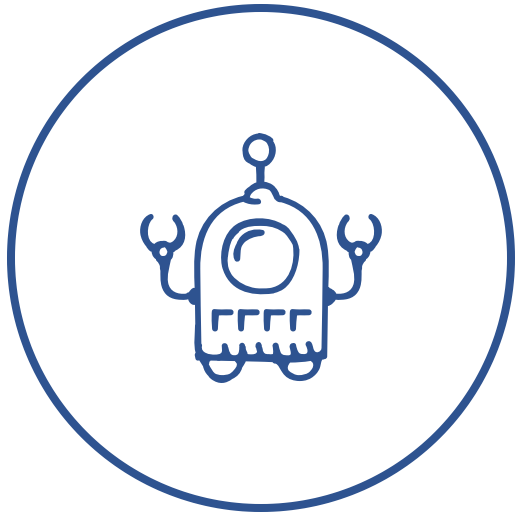
Are parents responsible for their children's bad choices? What if there are multiple creators?

The organisation

Can an organisation catch every error?

Is an organisation culpable, and how can it be made responsible?

Who is responsible for a morally wrong decision?



The AI

Requires it to have morality scores and complex reasoning

The creator

Are parents responsible for their children's bad choices? What if there are multiple creators?

The organisation

Can an organisation catch every error?

Is an organisation culpable, and how can it be made responsible?

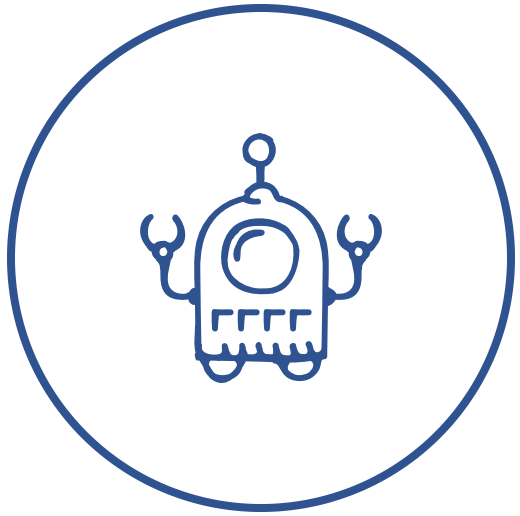
The end users?

Select or pay for an AI tool

Can provide their own data to train on

Apply it to a specific scenario

Scenario: Self driving car kills a pedestrian to save the driver.



The AI

Understood the relative risks and made a preordained decision

The creator

Did the programmer decide to kill the pedestrian by writing functionality to save the driver?

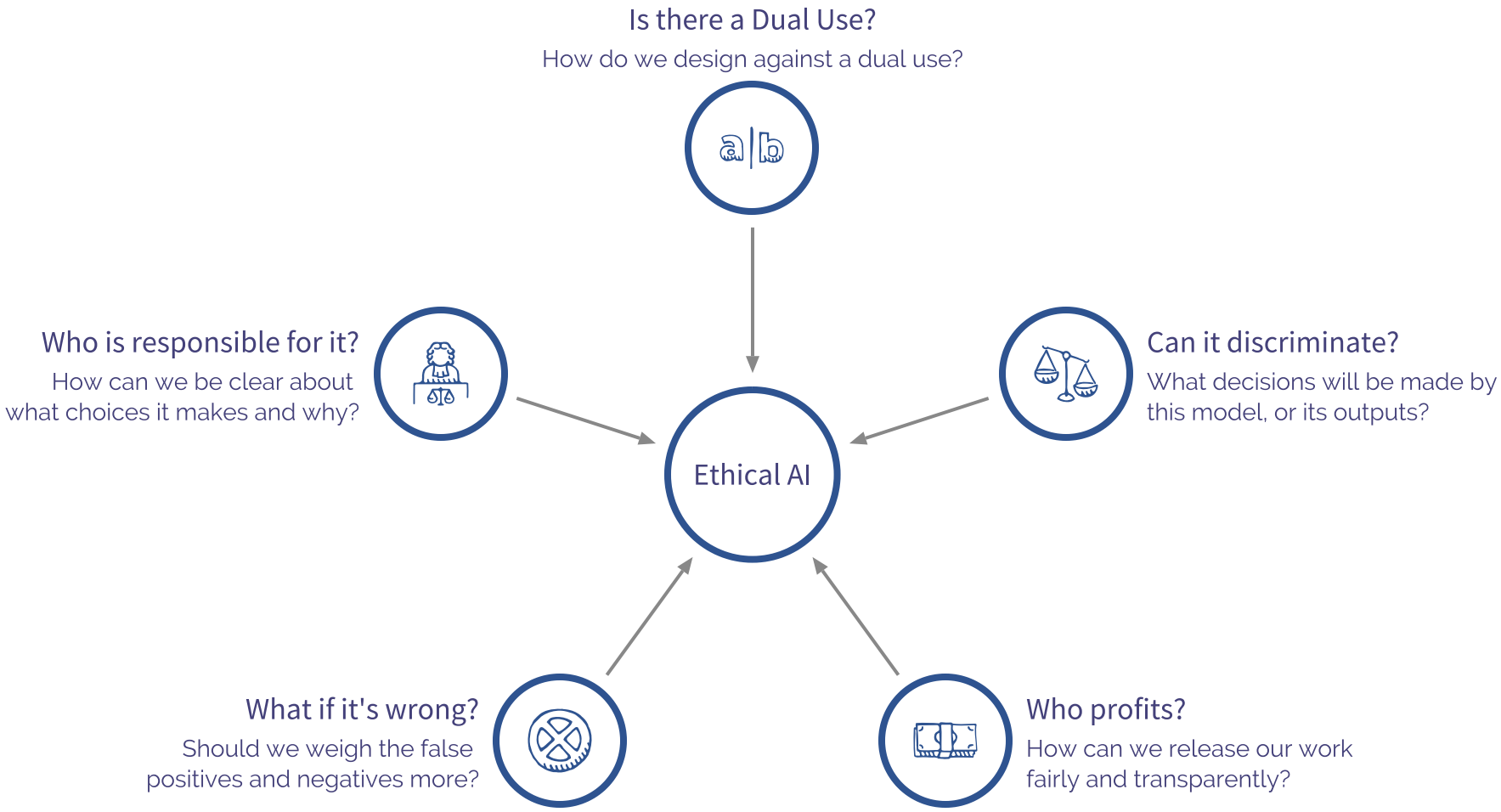
The organisation

Is the organisation evil for the death, or valued for saving the driver?

The end users?

A passenger without control

Questions


Is there a Dual Use?
How do we design against a dual use?
Who is responsible for it?
How can we be clear about what choices it makes and why?
Can it discriminate?
What decisions will be made by this model, or its outputs?
Ethical AI
What if it's wrong?
Should we weigh the false positives and negatives more?
Who profits?
How can we release our work fairly and transparently?

Problem: Parole decisions are often qualitative, biased, and unfair

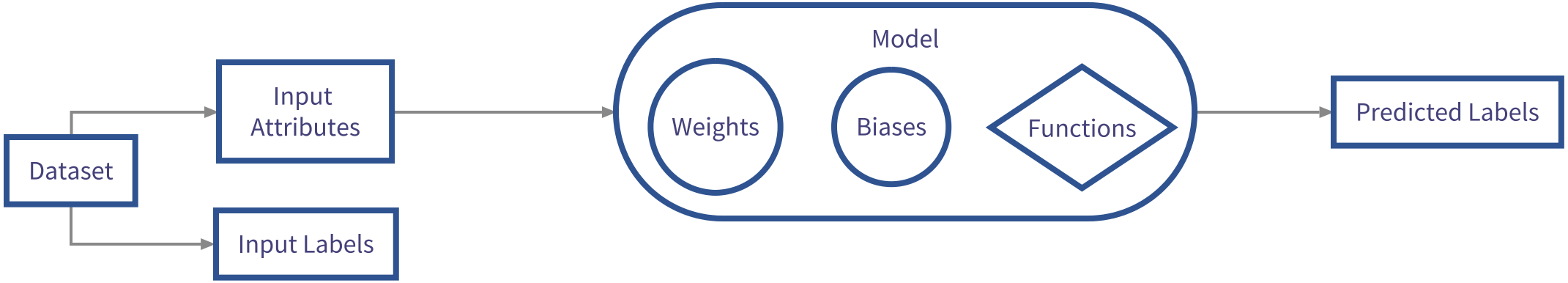
---

Can we use AI to bring a quantitative, evidence-based and fairer approach to these decisions?

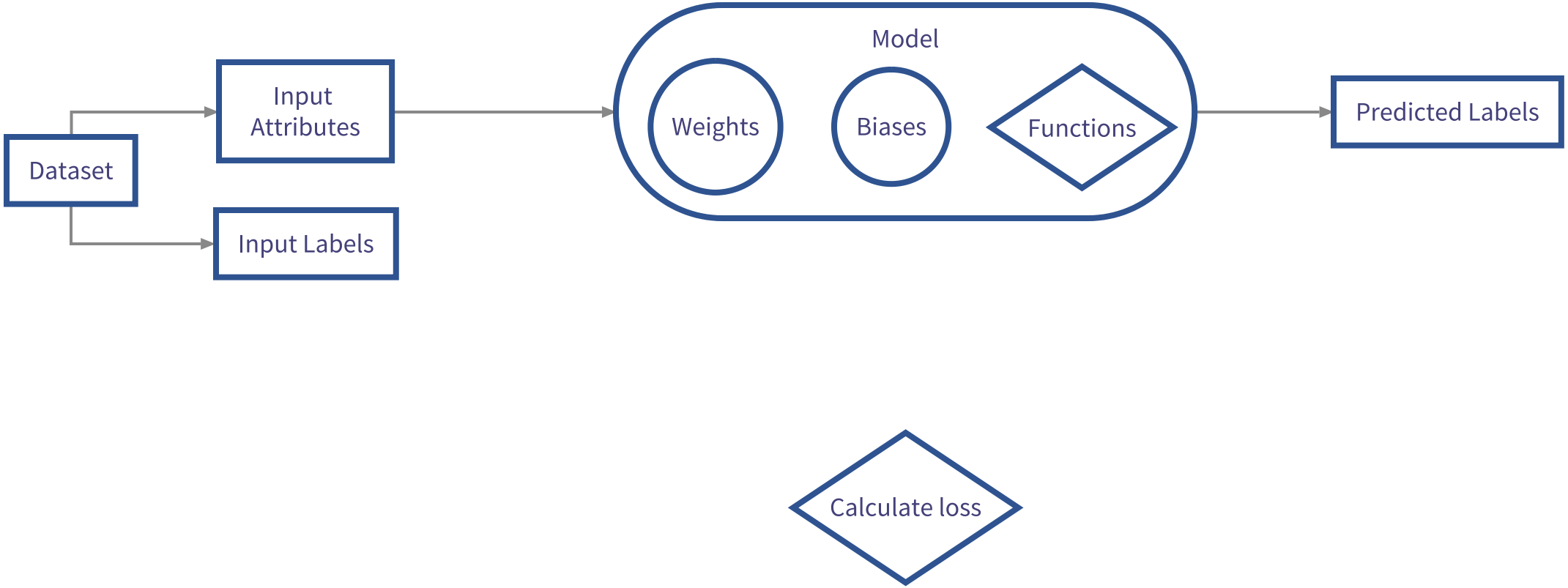
Section 2

Deep Learning

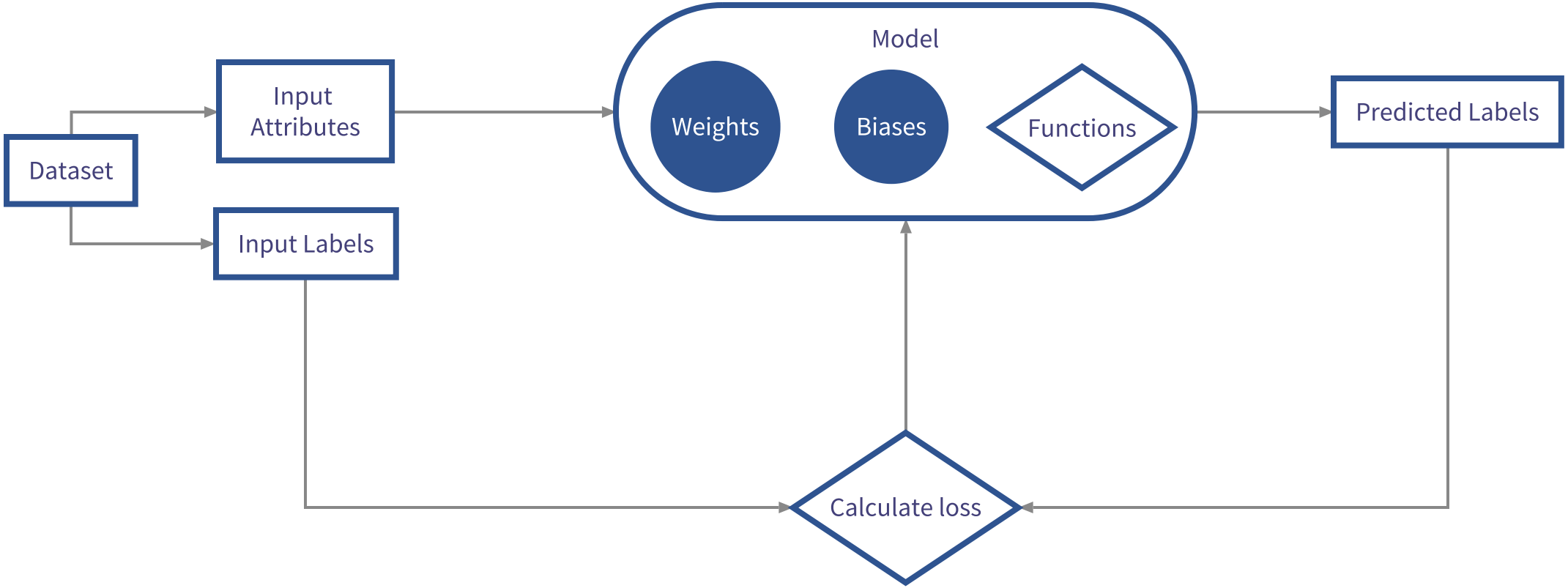
Deep Learning Flow

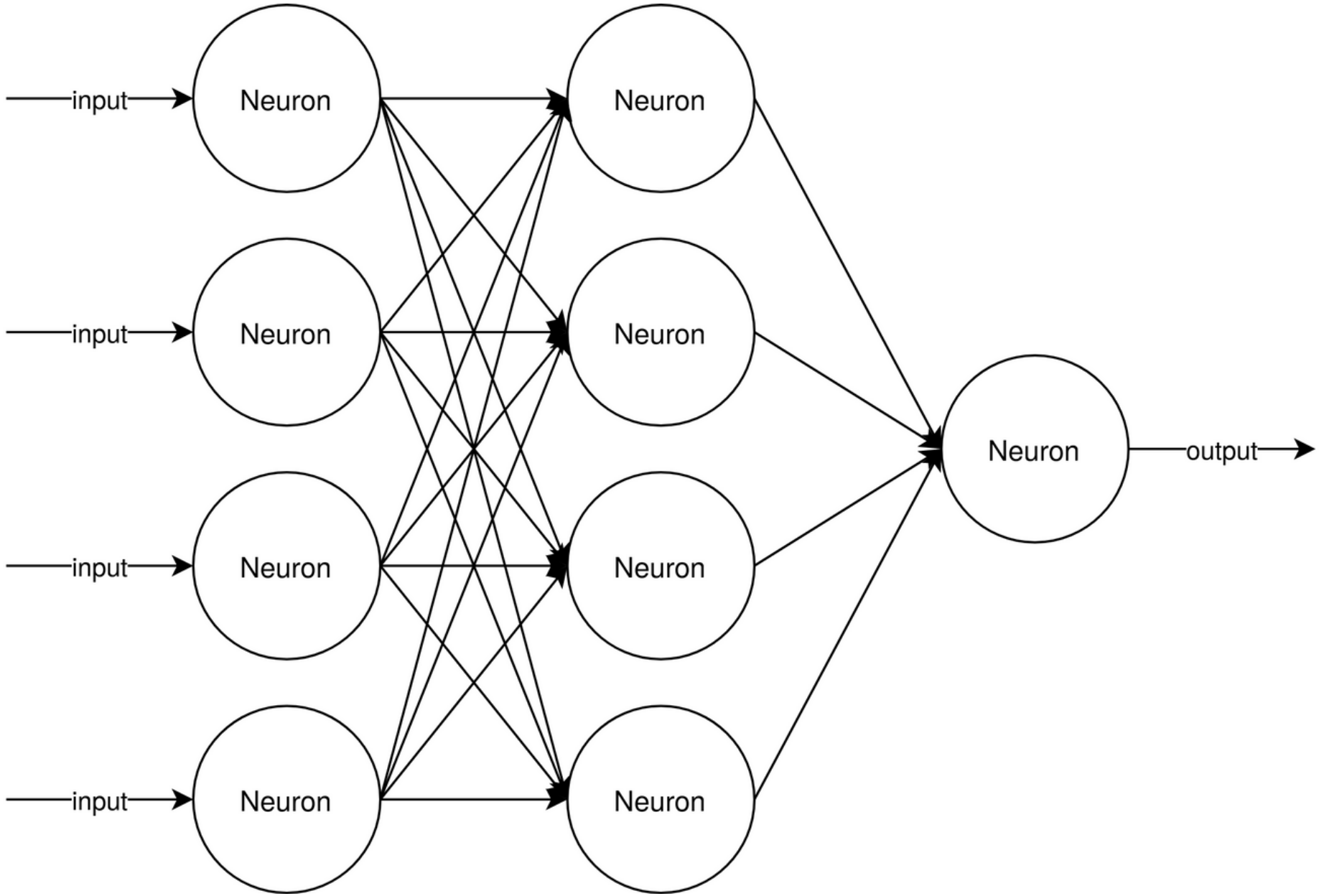


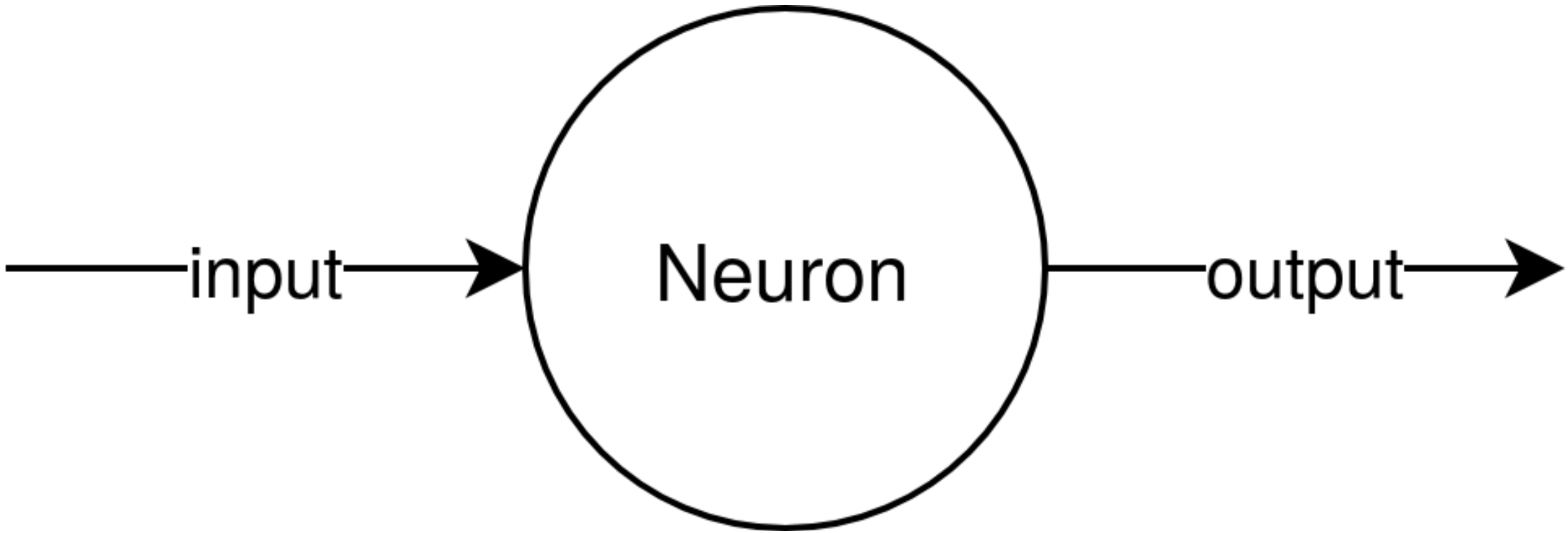
Deep Learning Flow



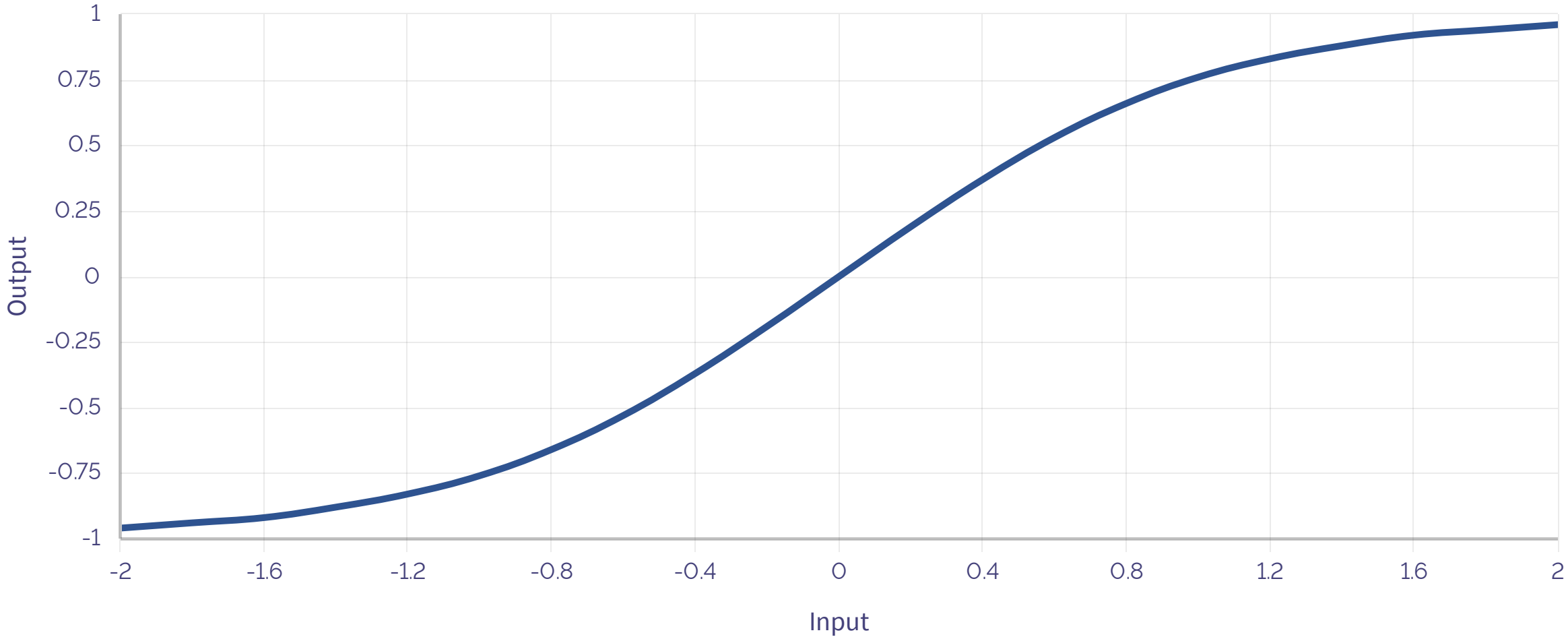
Deep Learning Flow

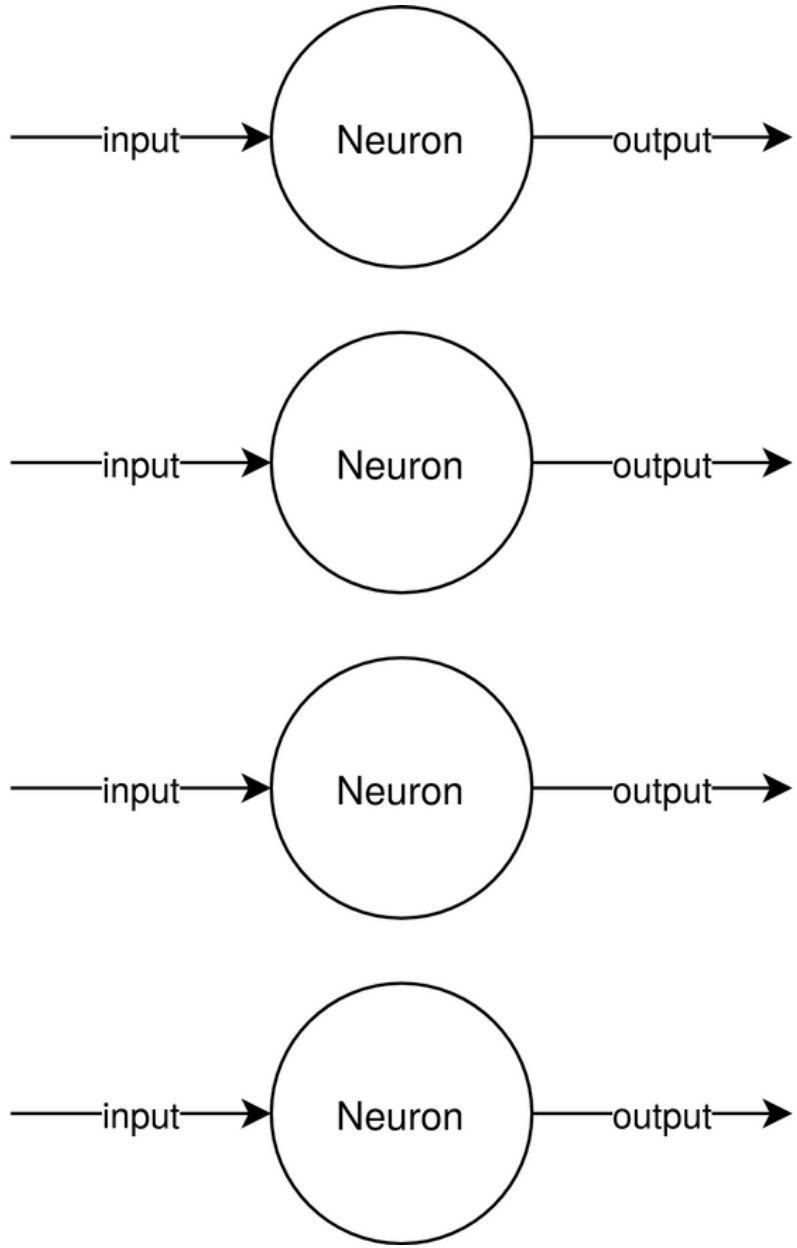


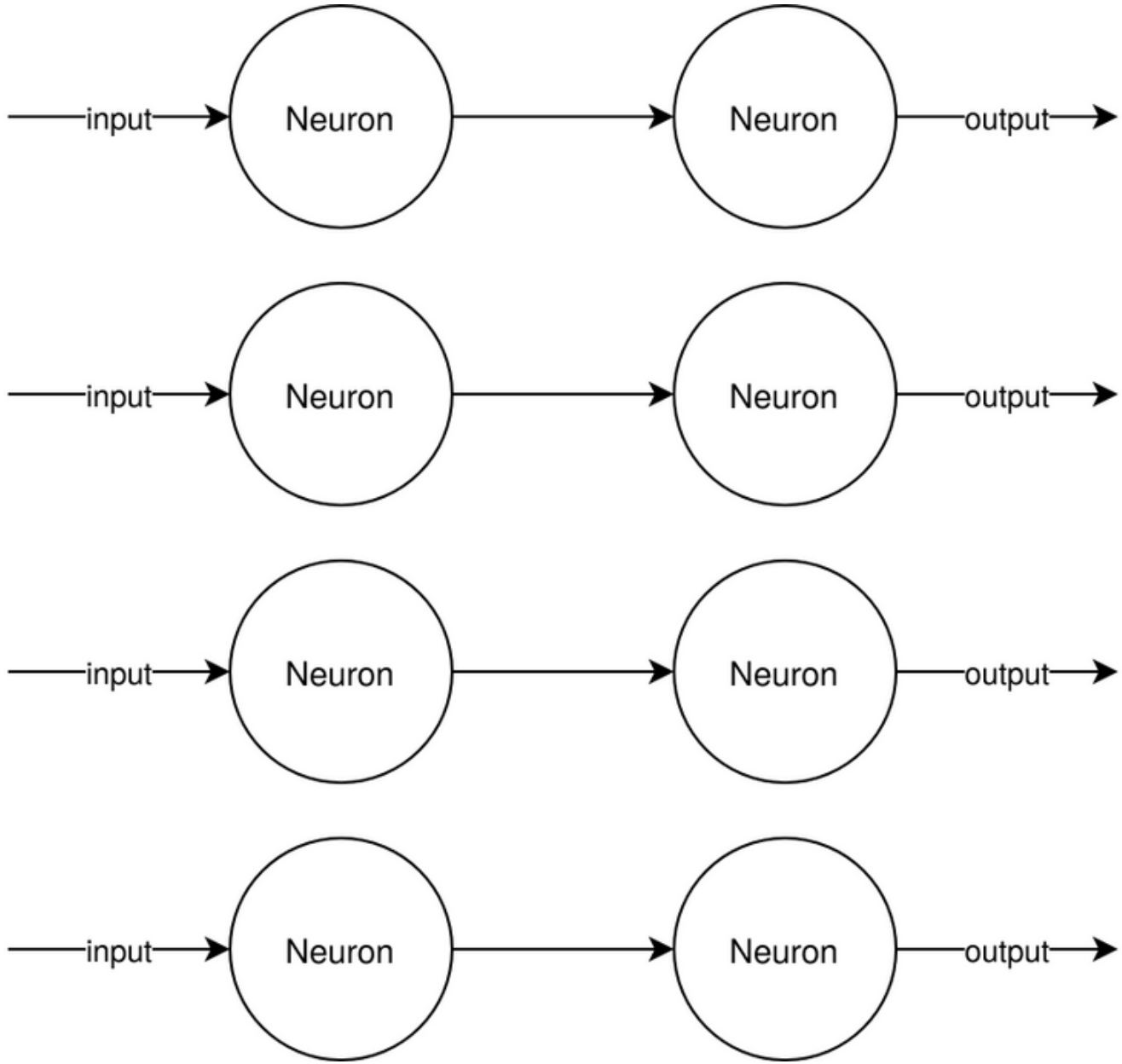

input
input
input
input
Neuron
Neuron
Neuron
Neuron
Neuron
Neuron
Neuron
Neuron
Neuron
output

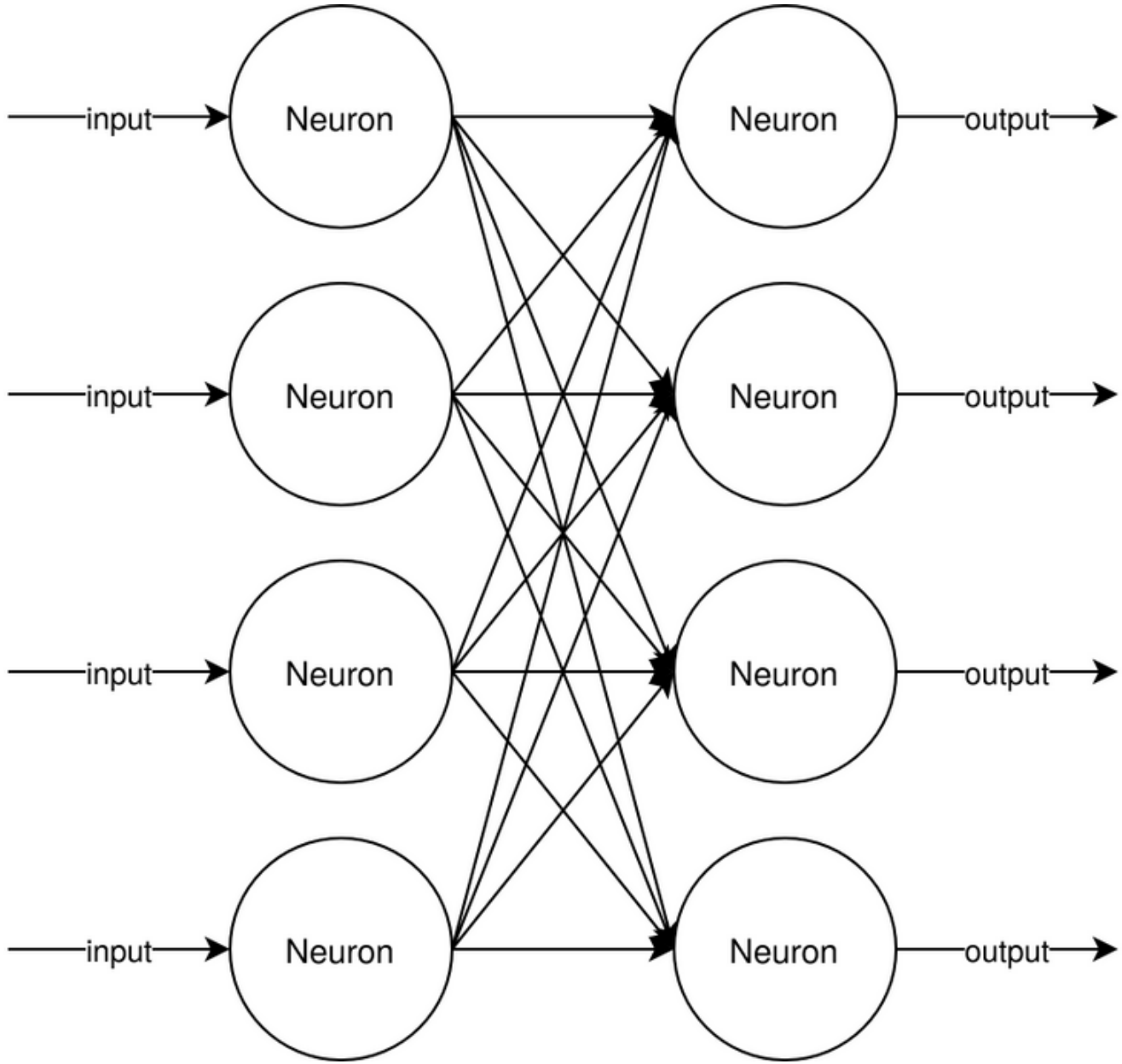

input
Neuron
output

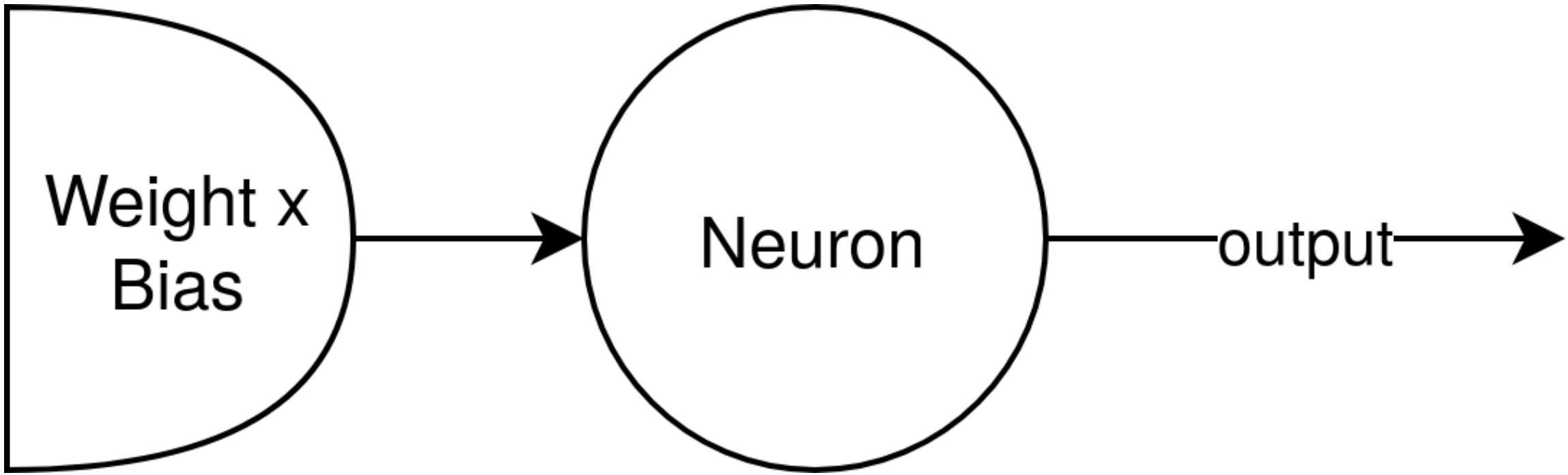
Tanh(x)

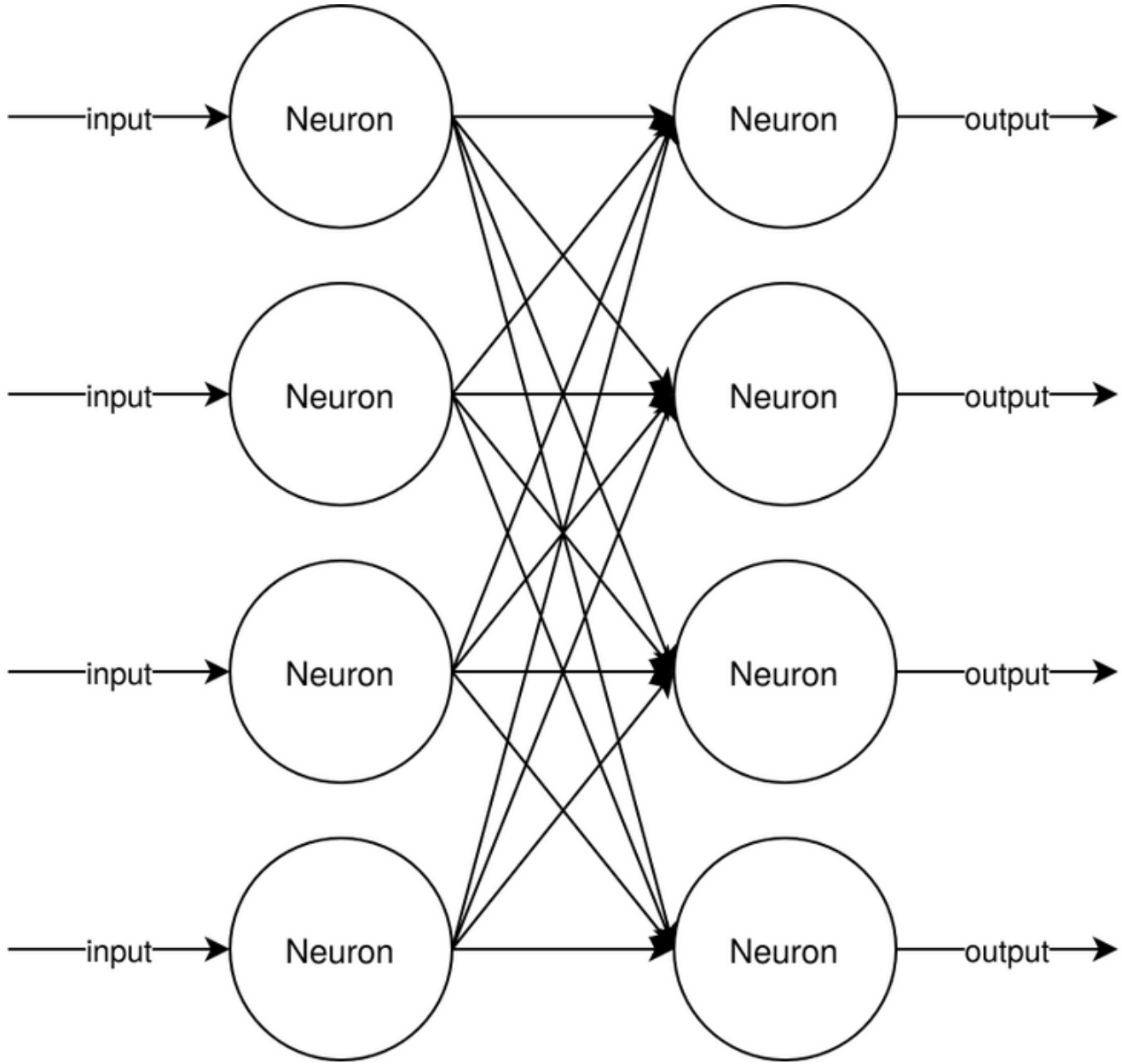


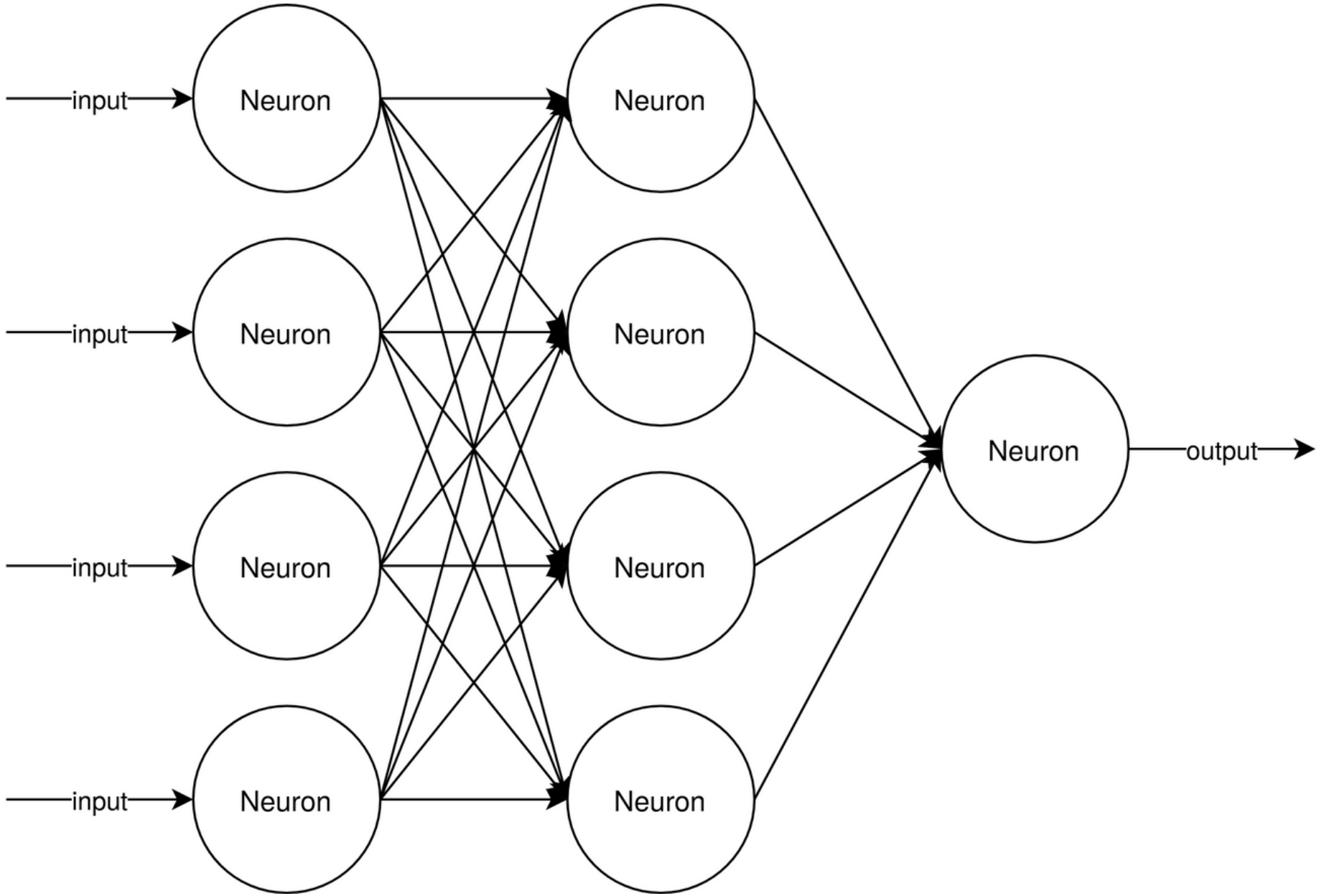

input
Neuron
output
input
Neuron
output
input
Neuron
output
input
Neuron
output


input
Neuron
Neuron
output
input
Neuron
Neuron
output
input
Neuron
Neuron
output
input
Neuron
Neuron
output

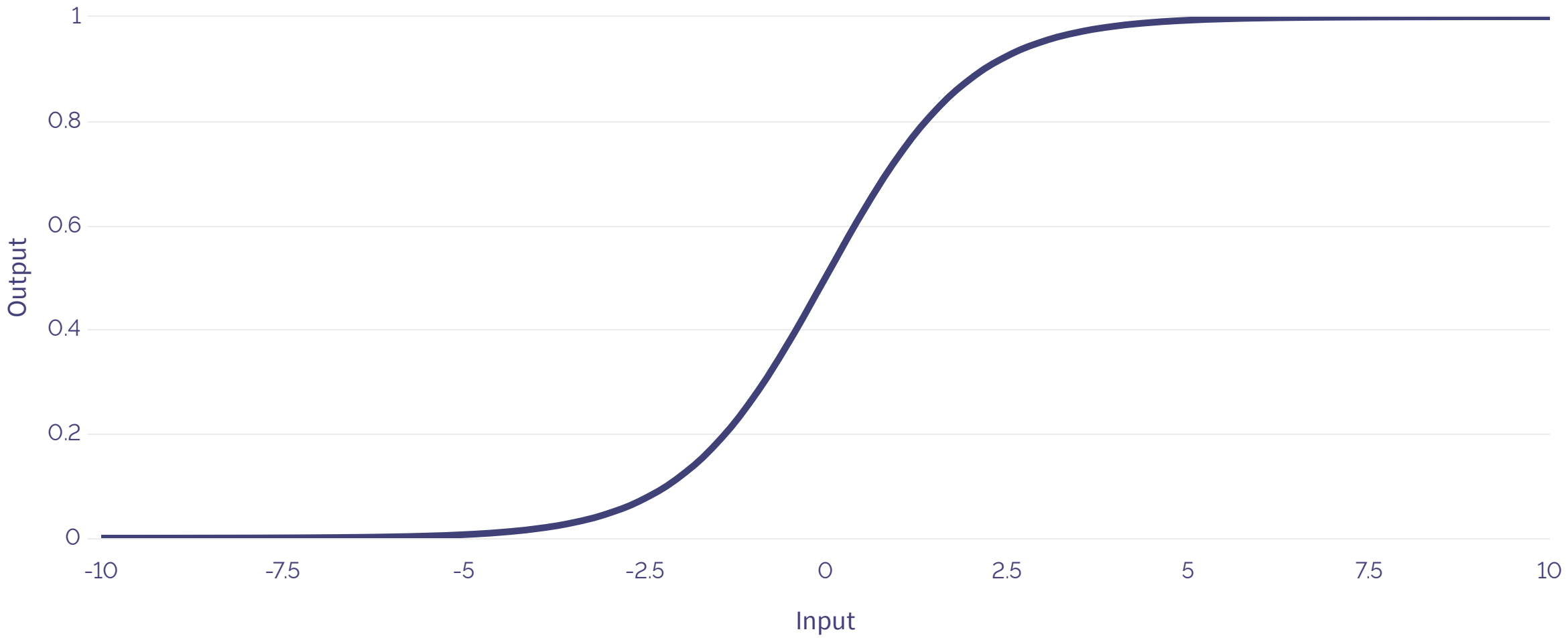

input
input
input
input
Neuron
Neuron
Neuron
Neuron
Neuron
Neuron
Neuron
Neuron
output
output
output
output

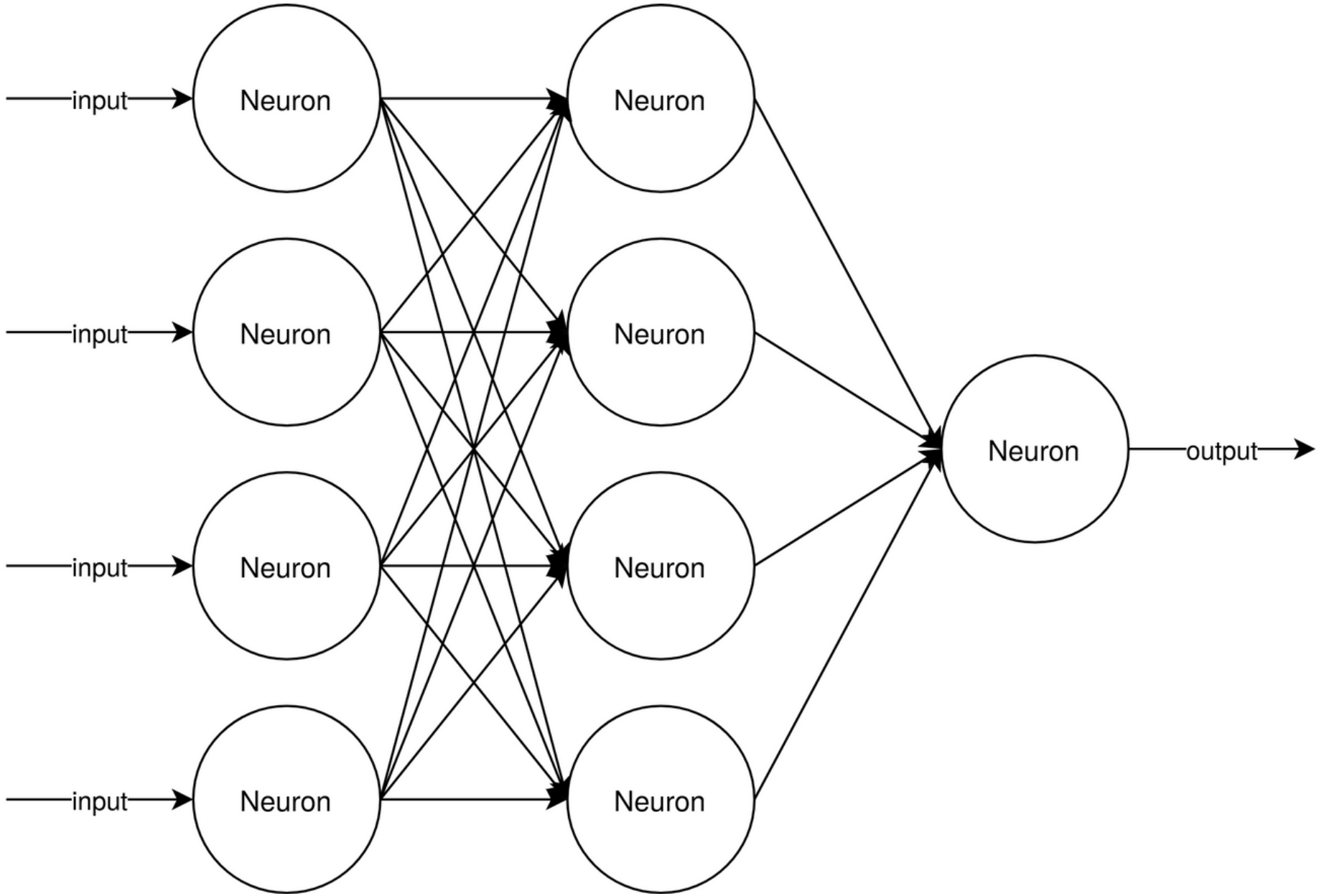

Weight x
Bias
Neuron
output


input
input
input
input
Neuron
Neuron
Neuron
Neuron
Neuron
Neuron
Neuron
Neuron
output
output
output
output


input
input
input
input
Neuron
Neuron
Neuron
Neuron
Neuron
Neuron
Neuron
Neuron
Neuron
output

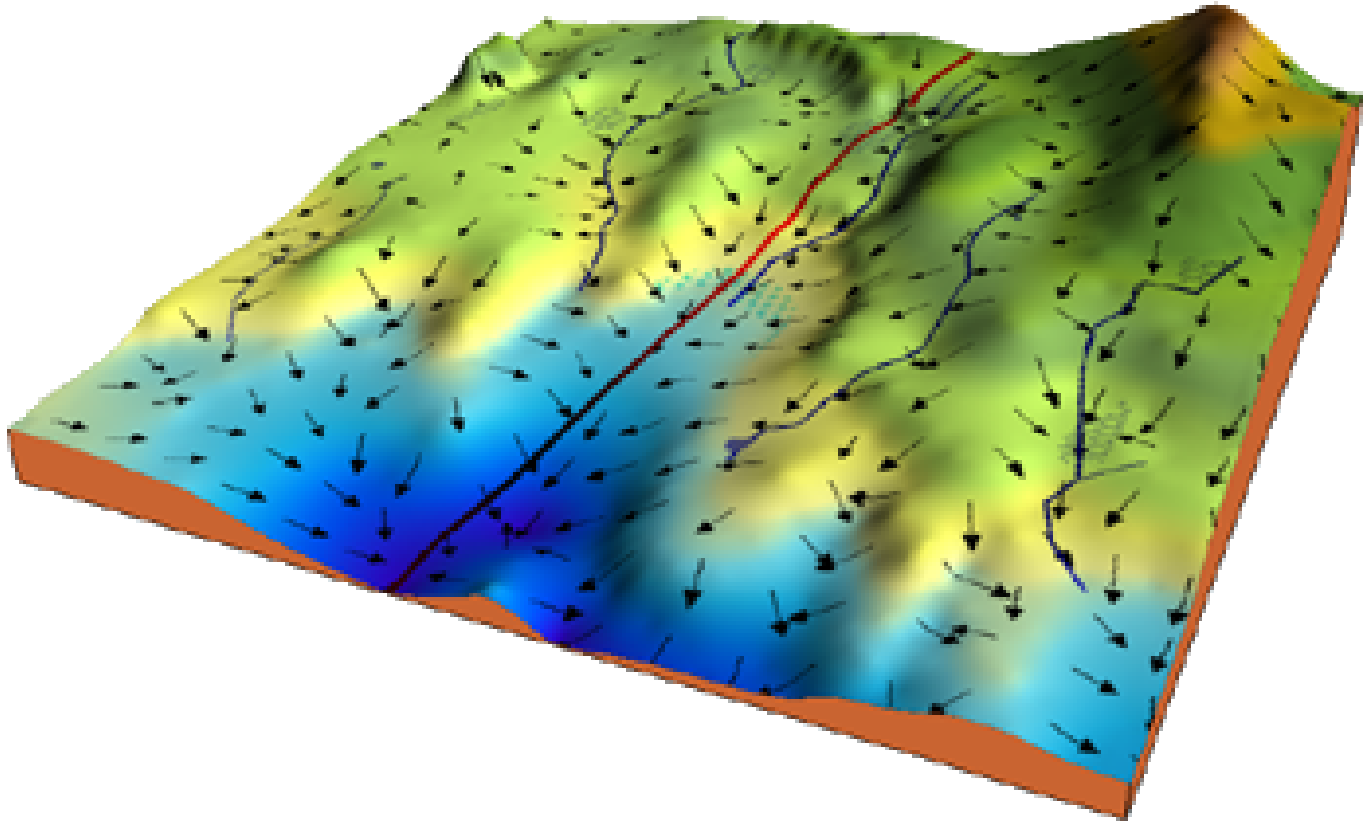
Sigmoid(x)



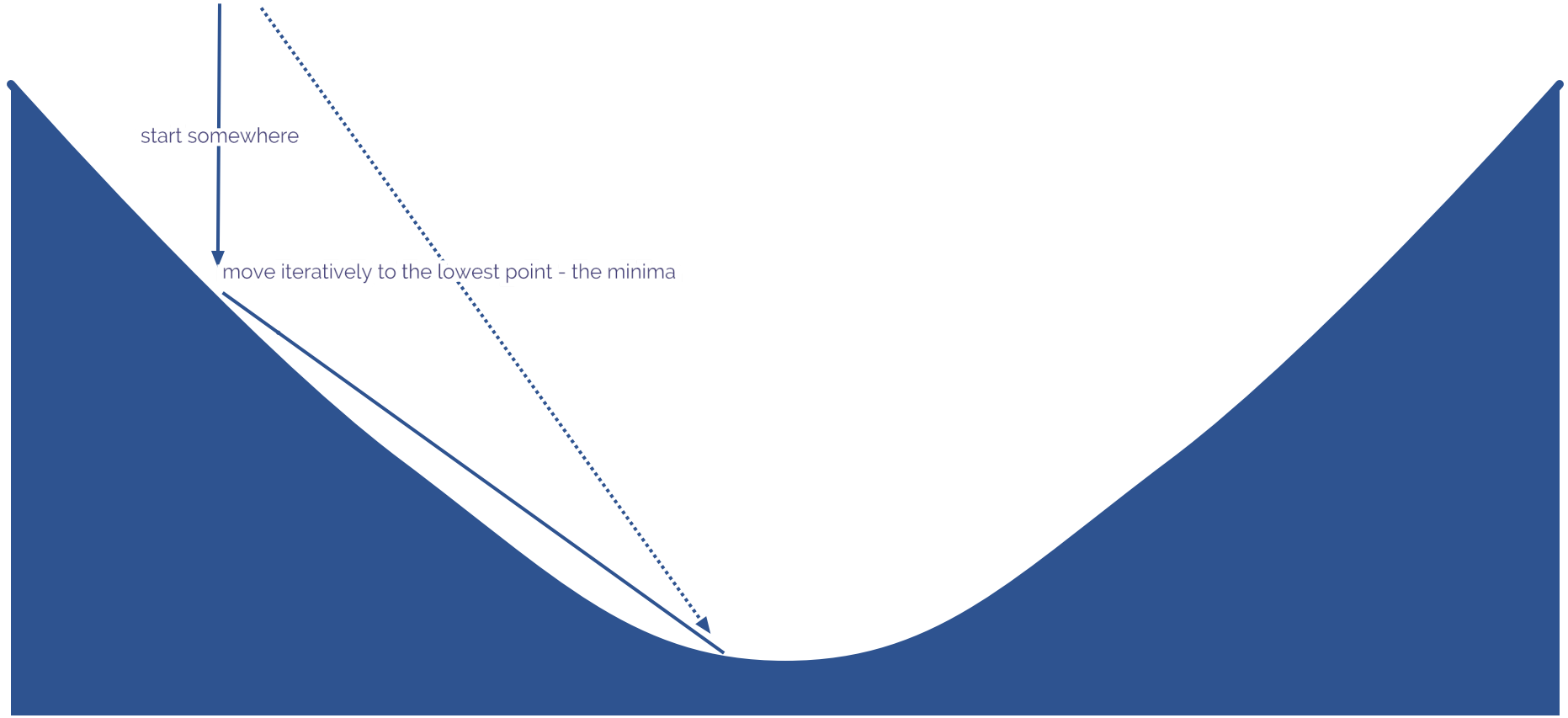

input
input
input
input
Neuron
Neuron
Neuron
Neuron
Neuron
Neuron
Neuron
Neuron
Neuron
output

Weights x loss gradient

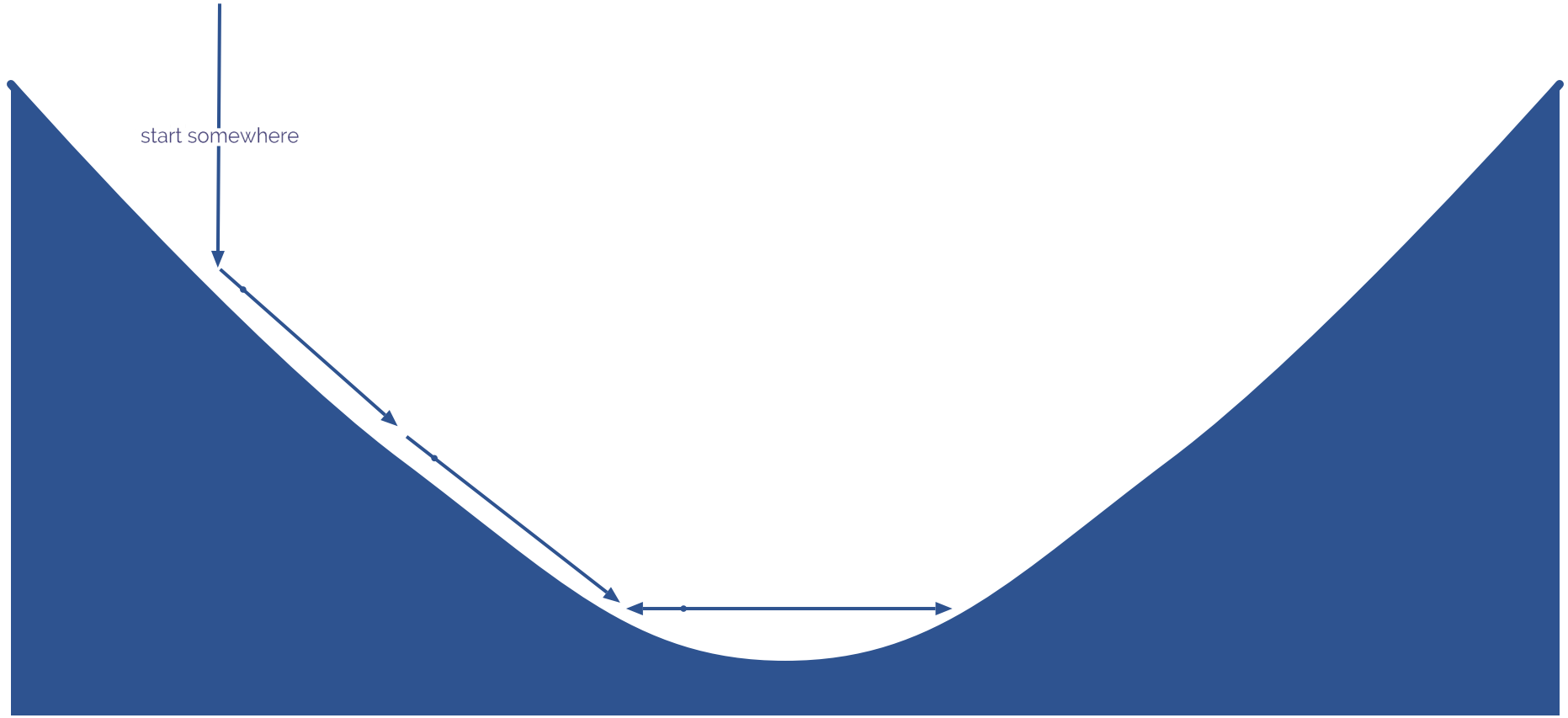
Bias x loss gradient



loss vs weights + bias



loss vs weights + bias

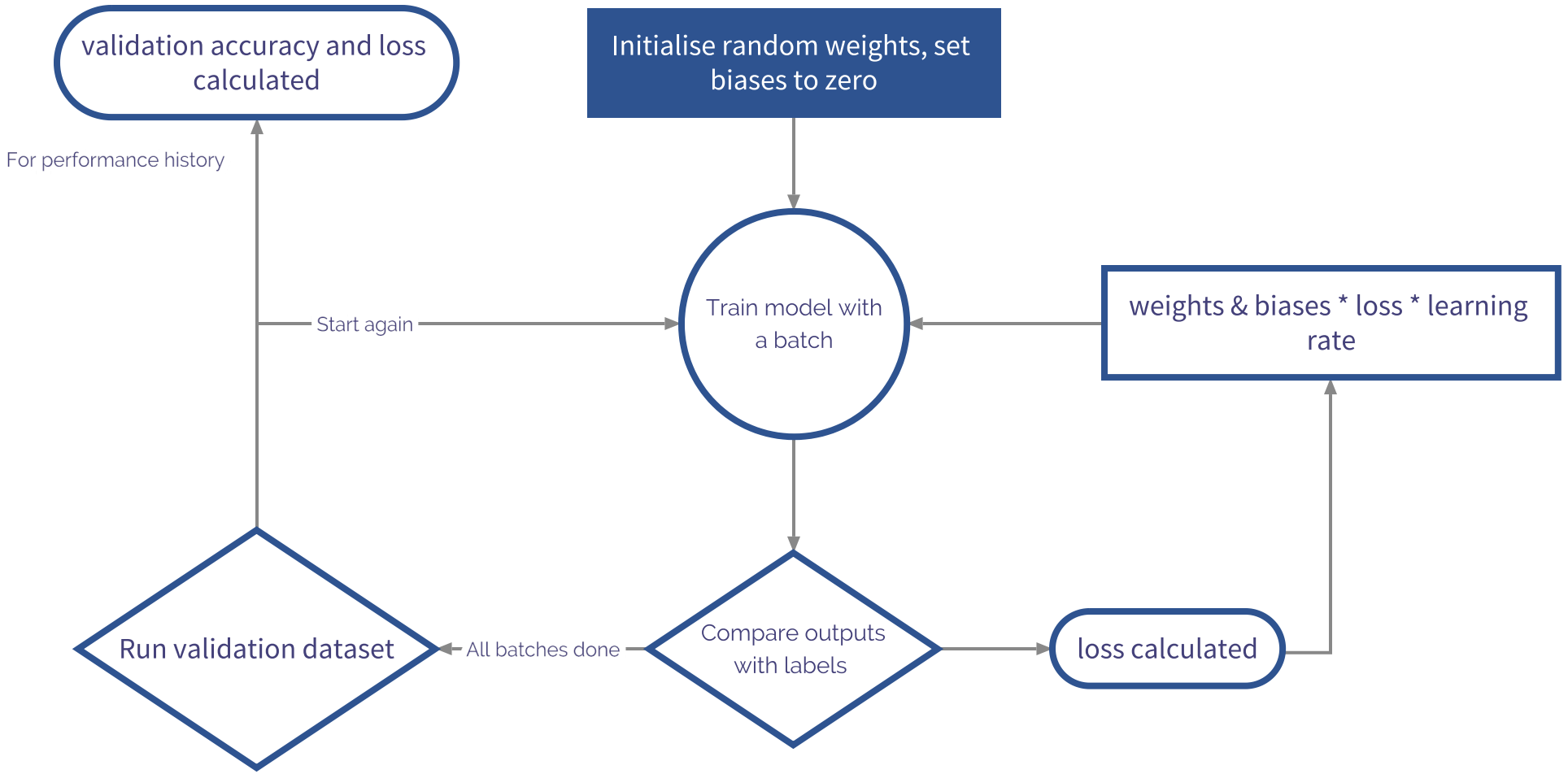


Weights x loss gradient
x learning rate

Bias x loss gradient x learning rate

cost vs weights + bias

full learning cycle (epoch)



When your performance plateaus, stop

Coffee

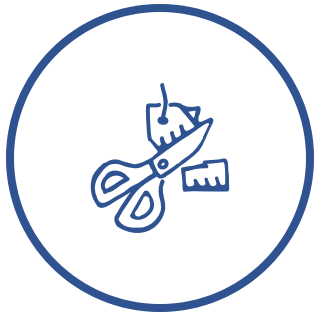
Toolkit

Activation Functions

What is the best activator for this?

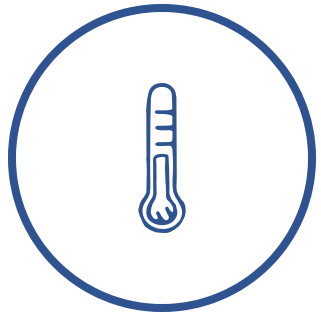
Topologies

How many layers? How many dimensions in the layers?



Interventions

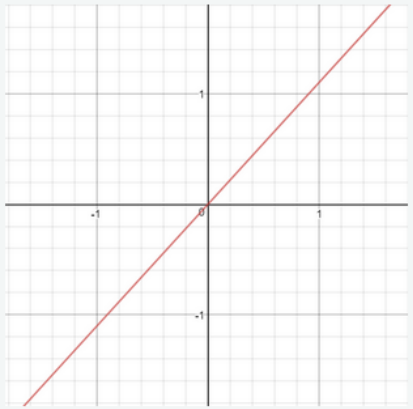
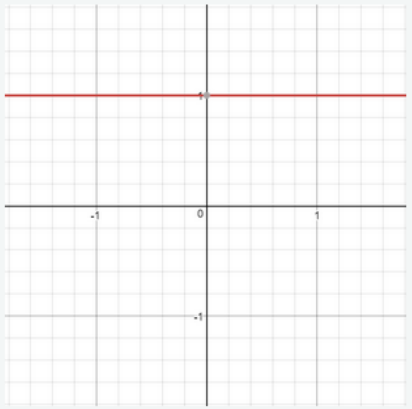
What do I do to keep my neural net behaving properly?



Loss Functions

How do I measure the performance of my Neural Network?

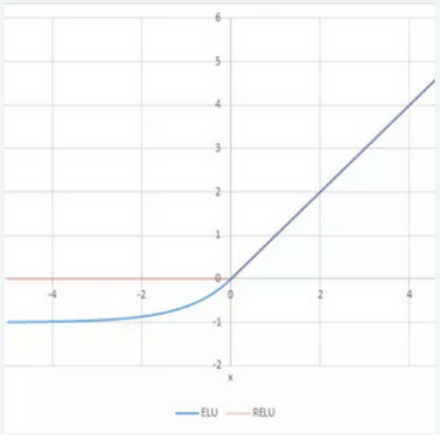
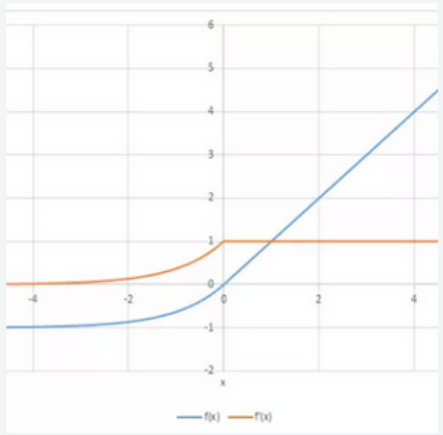
Linear

Function	Derivative
$R(z,m) = \{ z * m \}$	$R'(z,m) = \{ m \}$
	
`def linear(z,m):` `  return m*z`	`def linear_prime(z,m):` `  return m`

- Simplest function
- Relies only on the weight and bias
- You can probably do better

Exponential Linear Unit

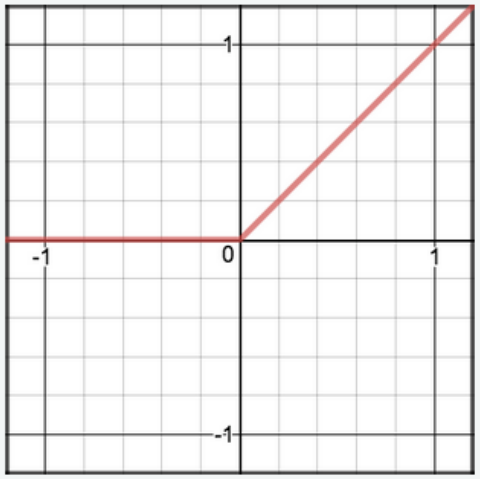
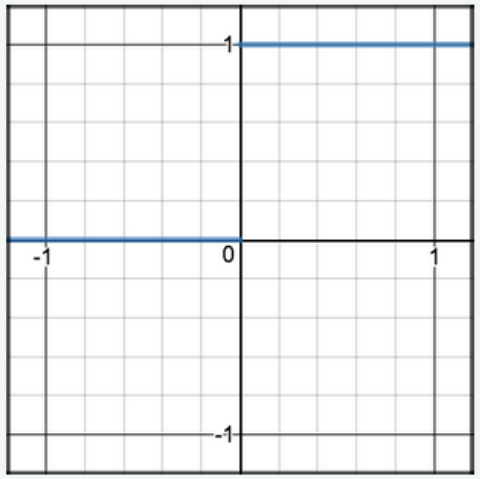
ELU

Function	Derivative
$R(z) = \begin{cases} z & z > 0 \\ \alpha.(e^z - 1) & z <= 0 \end{cases}$	$R'(z) = \begin{cases} 1 & z > 0 \\ \alpha.e^z & z < 0 \end{cases}$
	
`def elu(z,alpha):` `  return z if z >= 0 else alpha*(e^z -1)`	`def elu_prime(z,alpha):` `  return 1 if z > 0 else alpha*np.exp(z)`

- Encourages inputs to resolve faster towards zero
- needs an additional (positive) alpha constant
- Has a habit of exploding with positive z

Rectified Linear Unit

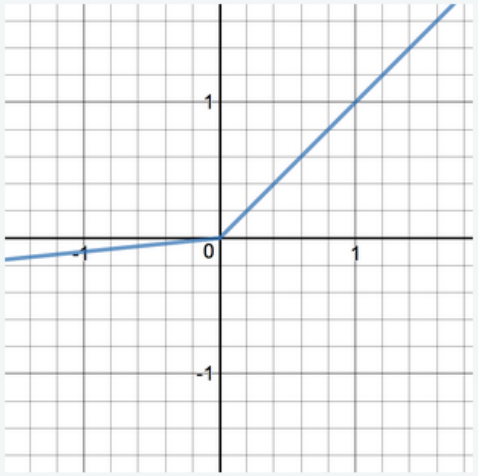
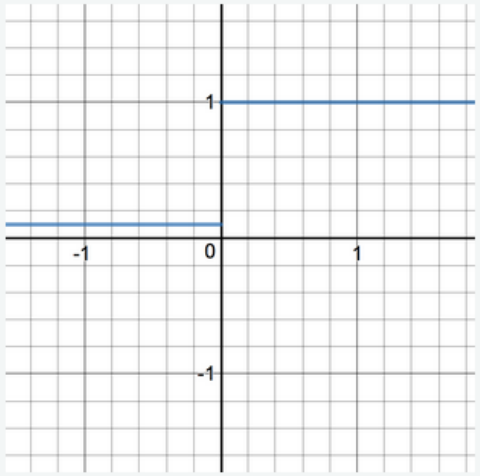
ReLU

Function	Derivative
$R(z) = \begin{cases} z & z > 0 \\ 0 & z <= 0 \end{cases}$	$R'(z) = \begin{cases} 1 & z > 0 \\ 0 & z < 0 \end{cases}$
	
`def relu(z):` `  return max(0, z)`	`def relu_prime(z):` `  return 1 if z > 0 else 0`

- A recent fix to the vanishing gradient problem
- Cheap to calculate
- Has a habit of exploding with positive z
- Can jam open or closed if the weight adjusts too much

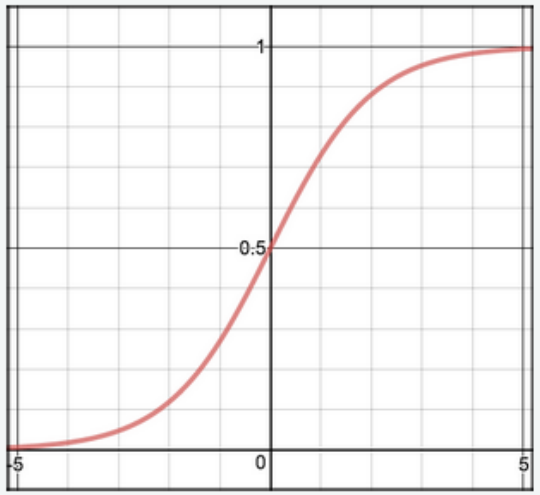
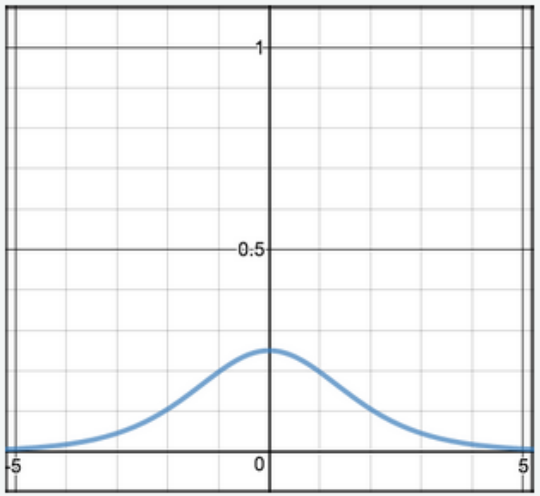
Leaky Rectified Linear Unit

LReLU

Function	Derivative
$R(z) = \begin{cases} z & z > 0 \\ \alpha z & z <= 0 \end{cases}$	$R'(z) = \begin{cases} 1 & z > 0 \\ \alpha & z < 0 \end{cases}$
	
`def leakyrelu(z, alpha):` `  return max(alpha * z, z)`	`def leakyrelu_prime(z, alpha):` `  return 1 if z > 0 else alpha`

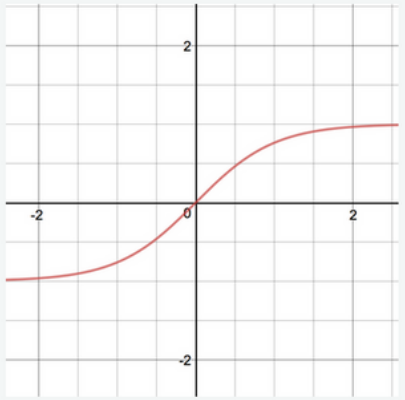
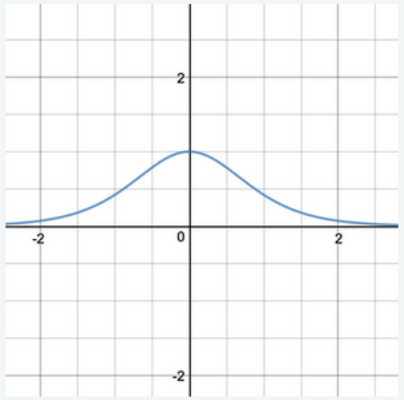
- Fixes dying ReLUs with a small alpha constant
- Cheap to calculate
- Has a habit of exploding with positive z
- No good for complex classification

Sigmoid

Function	Derivative
$S(z) = \frac{1}{1+e^{-z}}$	$S'(z) = S(z) \cdot (1 - S(z))$
	
`def sigmoid(z):` `  return 1.0 / (1 + np.exp(-z))`	`def sigmoid_prime(z):` `  return sigmoid(z) * (1-sigmoid(z))`

- Plots any value as a value between 0 and 1
- Smooth gradient, won't blow up
- Causes “vanishing” gradient
- Not zero-centered, can be slow to train

Tanh

Function	Derivative
$tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$tanh'(z) = 1 - tanh(z)^2$
	
`def tanh(z):` `  return (np.exp(z) - np.exp(-z)) / (np.exp(z) + np.exp(-z))`	`def tanh_prime(z):` `  return 1 - np.power(tanh(z), 2)`

- Plots any value as a value between -1 and 1
- Smooth gradient, won't blow up
- Zero centered!
- Causes “vanishing” gradient

What do I use?

- Tanh was the original best choice for activator functions, but had problems in larger networks (vanishing gradient issues)
- ReLU isn't as effective but is faster to train and avoids gradient issues. Only used in hidden layers
- Sigmoid is best used as an end activator

Topologies

- Start with the numbers of dimensions/neurons as your tensor dimensions
- Adding and removing dimensions will change how the network behaves
- Deeper networks take exponentially longer to train, but can be more effective if used properly (watch out for vanishing gradients)

Changing dimensionality

Reducing dimensions

By lowering the number of neurons in a consecutive layer, the neural network loses information

This can encourage it to lose noise and prioritise useful information

Maintaining dimensions

By keeping the dimensionality the same in a consecutive layer, the network preserves information

This can encourage the network to further develop on the information it has

Increasing dimensions

By increasing the number of neurons in a consecutive layer, the neural network extrapolates on information

While no new information is created, it encourages a neural network to widen its introspection of existing information

Changing dimensionality

Reducing dimensions

By lowering the number of neurons in a consecutive layer, the neural network loses information

This can encourage it to lose noise and prioritise useful information

Maintaining dimensions

By keeping the dimensionality the same in a consecutive layer, the network preserves information

This can encourage the network to further develop on the information it has

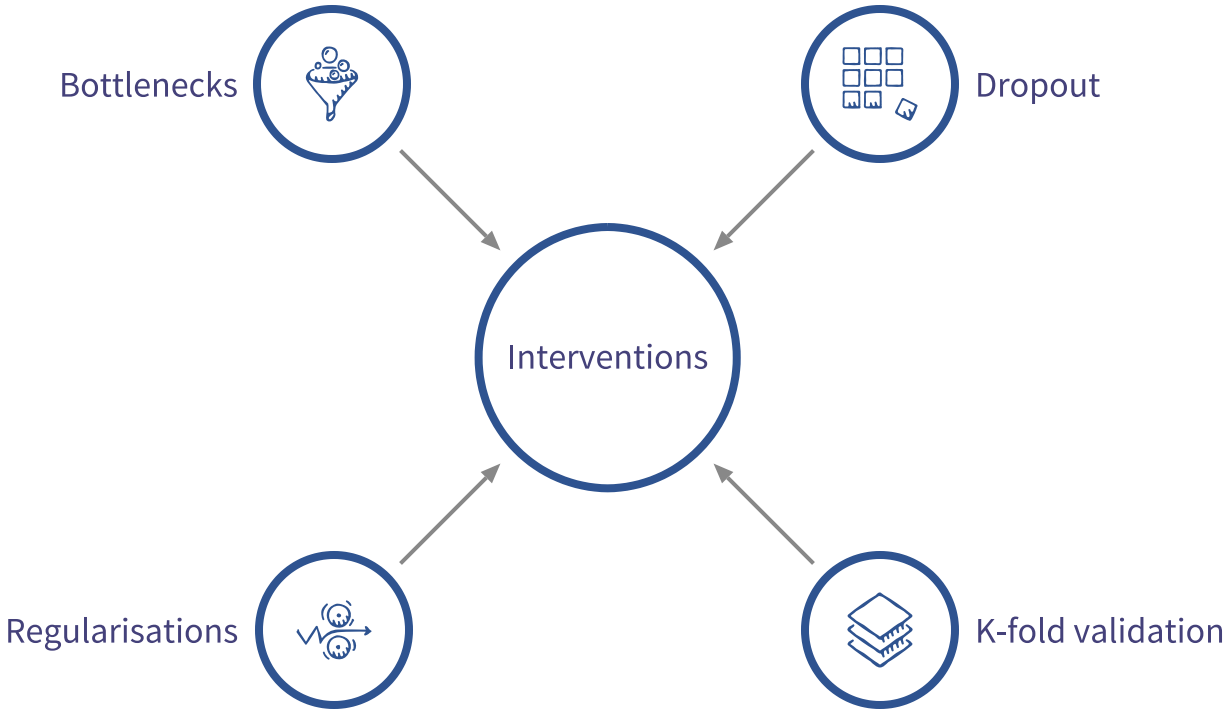
Increasing dimensions

By increasing the number of neurons in a consecutive layer, the neural network extrapolates on information

While no new information is created, it encourages a neural network to widen its introspection of existing information

The first two are more typically used. More dimensions also means more processing.

Interventions



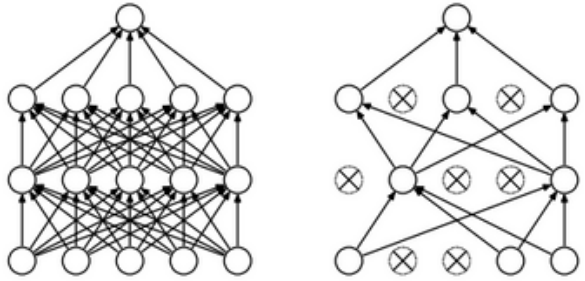
Bottlenecks

- A significant drop in dimension
- Meant to force unimportant dimensions out
- Good for reducing noise in the system
- Side benefit of faster computation

K-fold validation

- Avoids the risk of validation and traning sets being randomly different
- Takes a random percentage of the dataset as validation, k number of times. Runs the models in parallel
- expensive!

Drop-out

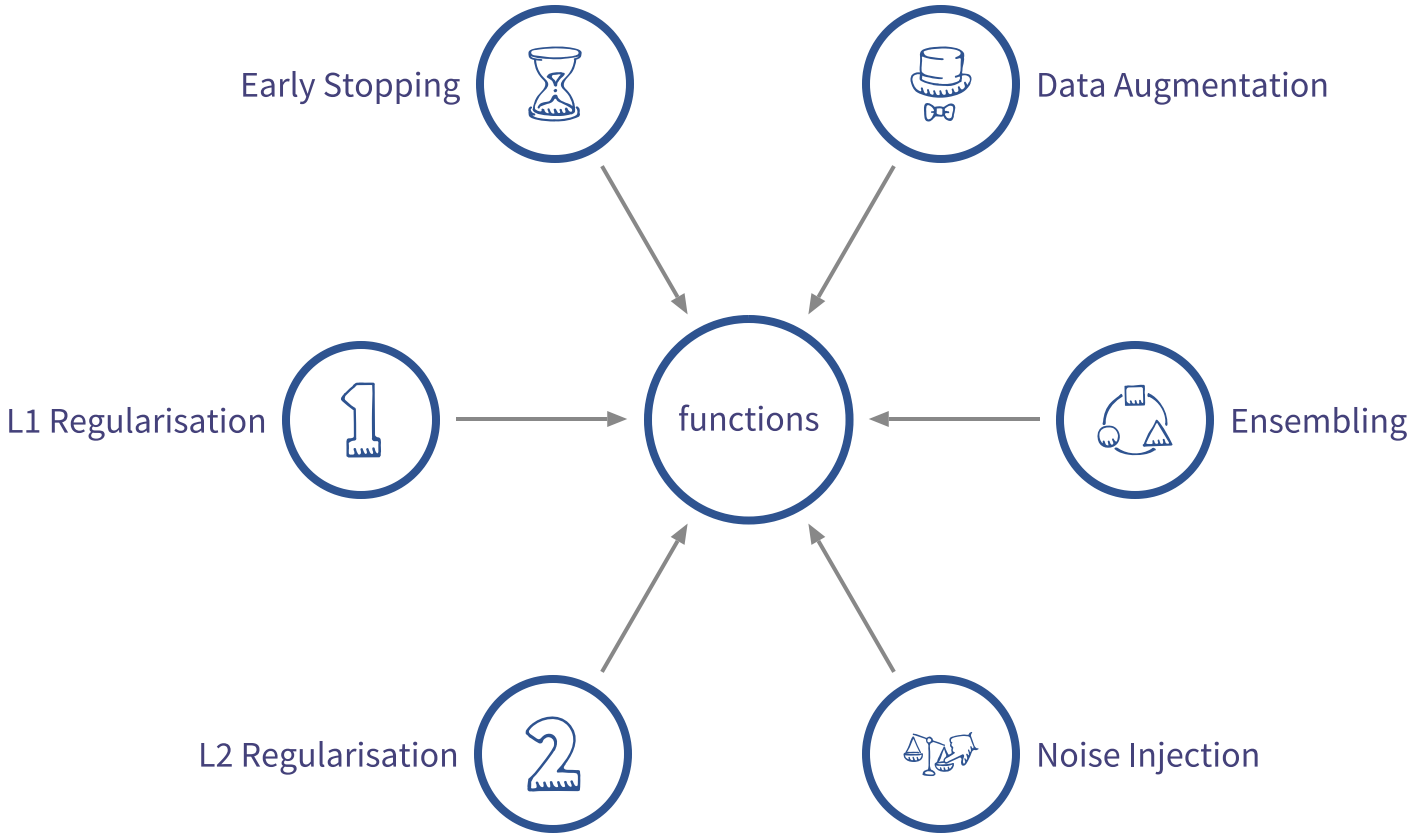


(a) Standard Neural Net (b) After applying dropout.

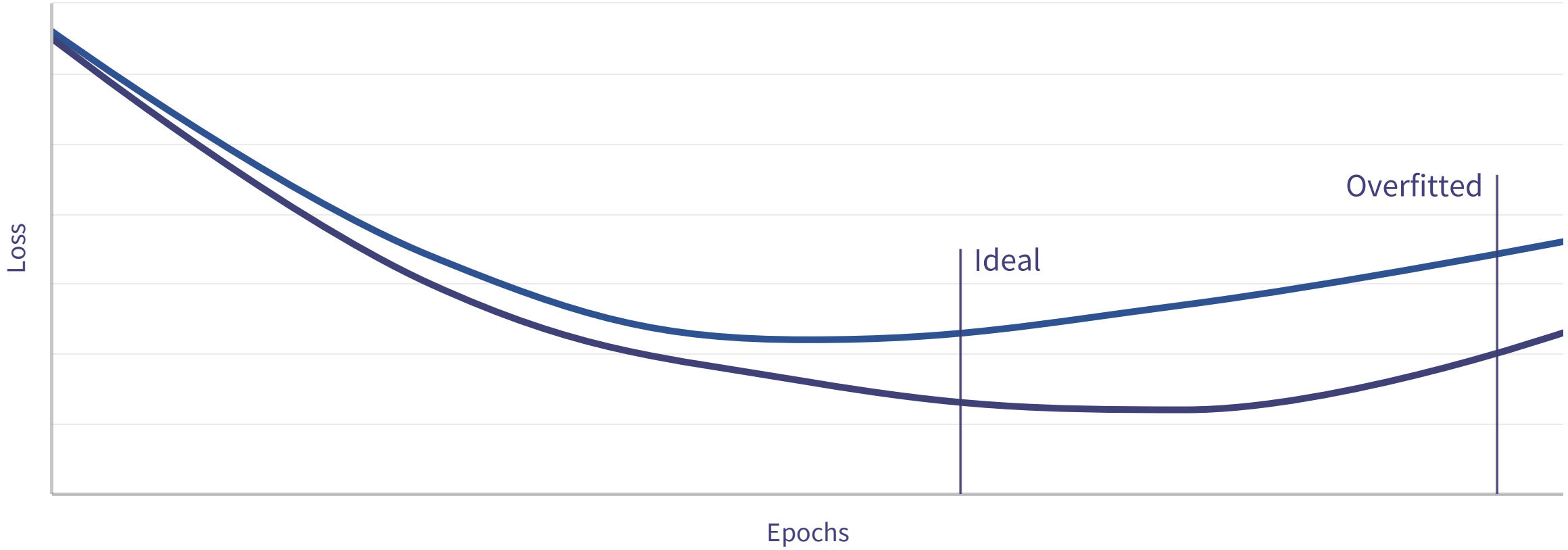
Figure 1: Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

- At a specific layer in the model, kill x% of the nodes
- Forces the model to work well in imperfect condition
- Counteracts overfitting effectively
- Avoids the model relying on only one or two attributes or trends

Regularisations



Early Stopping



Run multiple models with different epoch values.

Data Augmentation

- Instead of new datasets, change the existing ones
- Replace similar words with synonyms for NLP
- Flip photos horizontally

Noise Injection

- A subset of data augmentation
- Add random statistical noise to inputs or to specific layers
- Discourages a Neural Network from memorising training samples
- Only on training, not evaluation

You can also add noise to the known labels or to the weights too

Ensembling

- Bagging - Run multiple Neural Networks together and average the results

  Reduce variance on strong learners

- Boosting - Run multiple Neural Networks end-to-end

  Create a strong learner from weak learners

Strong and weak learners

aka Unconstrained and Constrained models

- Neural Networks are usually strong learners - learn everything about their inputs
- You can hobble a neural network by their inputs, noise or by their topologies, so they only “learn” one thing

L1 Regularisation

- Penalises model complexity
- Encourages weights towards zero
- Good if the model uses a lot of features
- There are sklearn tools that do this for you

L2 Regularisation

L1 squared...

- Penalises model complexity and collinear attributes
- Encourages weights towards zero
- Good if the model uses a lot of features
- There are sklearn tools that do this for you

Strong L1 and L2 regularisation leads to underfitting

Binary Classification Loss Functions

Cross Entropy (log loss)

Go-to method

compares the probability distribution of the model output labels against the true labels

Hinge

Better for SVMs

where outputs are -1<x<1 (tanh not sigmoid). polarity is more important.

max(0, 1-y*q)

Squared Hinge

max(0, (1-y*q)^2)

smoother than hinge, works in the same way and setting

Multiclass Classification Loss Functions

Multiclass Cross Entropy

Go-to method

compares the probability distribution of the model output labels against the true labels, averaged across classes

Sparse MCE

Better for when you have hundreds of classes. Doesn't need one-hot encoding

Needs a softmax output

Kullback Liebler Divergence

Similar to MCE, but not originally intended for Multiclass Classification. Better for representation.

Regression Loss Functions

MSE - L2

Go-to method

compares the match between a sequence of continuous values (yHat and y). Punishes mistakes more than L1. Expects gaussian PDFs

MSLogE

Mimicks L2, but doesn't punish larger distances between yHat and y as much. Ideal for regressing to much higher values.

MAE - L1

“bumpier” than L2. Ideal fro semi-gaussian PDFs with large outliers.

Lunch

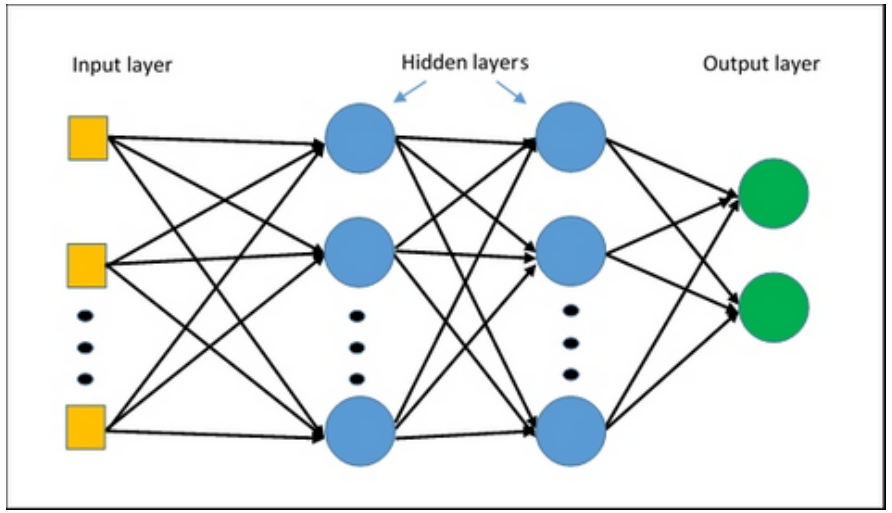

THE #1 DATA SCIENTIST EXCUSE FOR LEGITIMATELY SLACKING OFF:
"MY MODEL'S TRAINING"
HEY! GET BACK TO WORK!
TRAINING!
OH. CARRY ON.

Give it a go - Boston housing dataset

Neural Networks

- **ANN**
  Artificial Neural Network
- **MLP**
  Multi-Layer Perceptron
- **RNN**
  Recurrent Neural Network
- **CNN**
  Convolutional Neural Network
- **VAE**
  Variational Autoencoder
- **GAN**
  Generative Adversarial Network

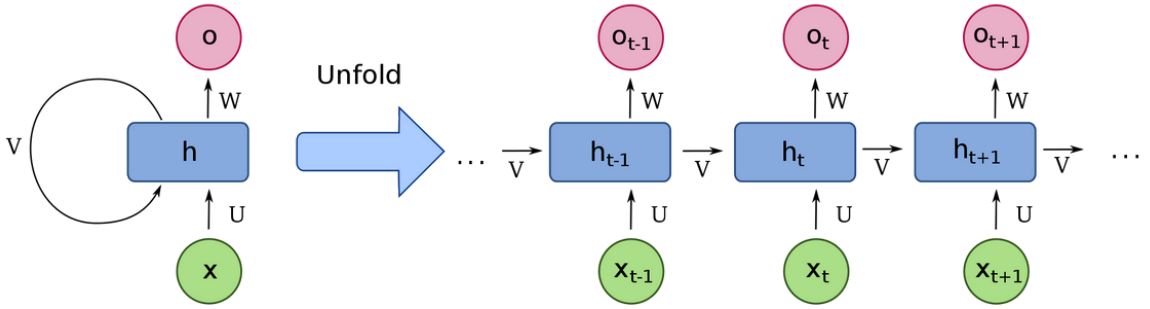
Multi - Layer Perceptron



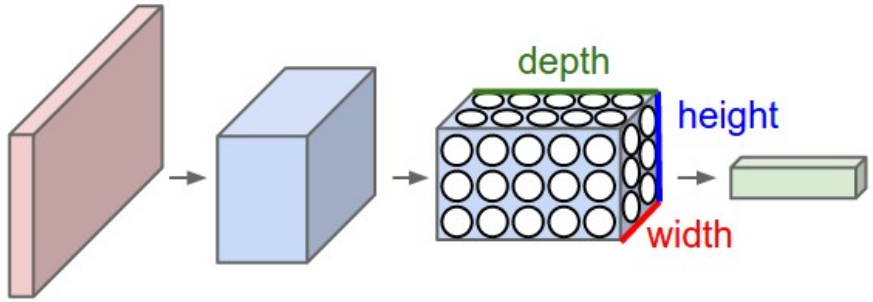
- Fully Connected Layers
- At least one hidden layer
- Sequential, forward only
- Simpler Deep Learning model

Recurrent Neural Network

- Walk along sequences of events
- Can process varied-length datasets
- Often holds significant information from previous data points
- defined by its inputs and outputs (one-many, many-many, many-one)



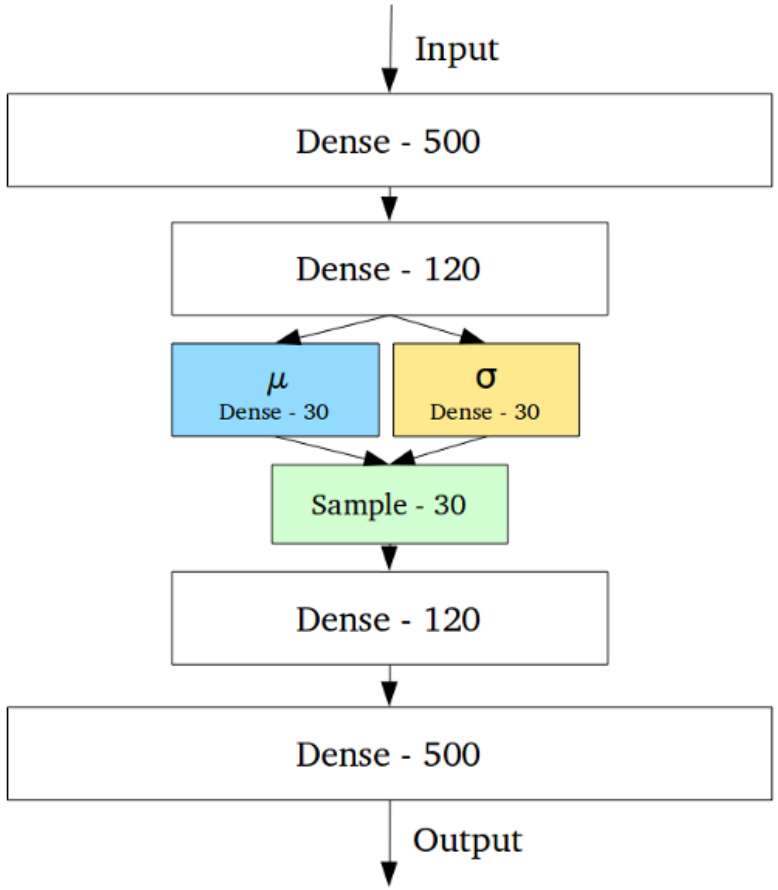
Convolutional Neural Network



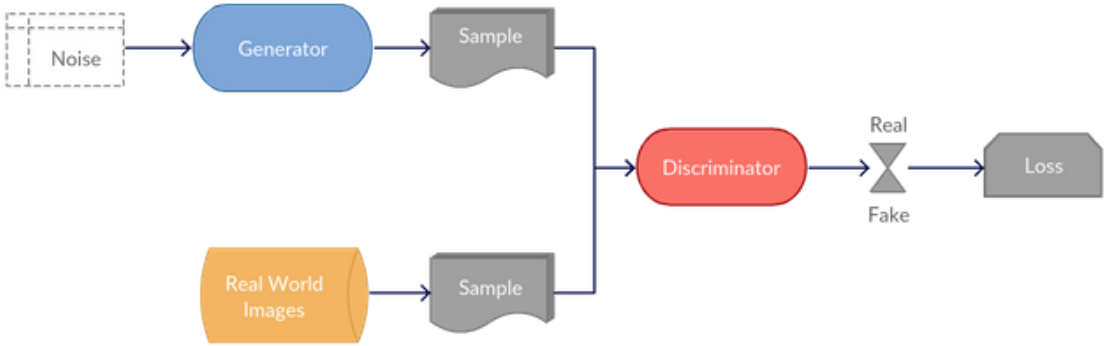
- Uses mathematical convolution as part of its data preparation
- Pools pixels to reduce dimensionality down-stream
- Fully connected
- Reliant on kernels to spot patterns

Variational AutoEncoder

- simpler architecture
- generative in purpose - samples from latent space
- reconstruct fuzzily



Generative Adversarial Network



- Two competing AIs
- One invents fake data, the other polices against it
- Generative “photo-realism”
- Deepfakes

CNN example

Garbage in, garbage out


and
breathe