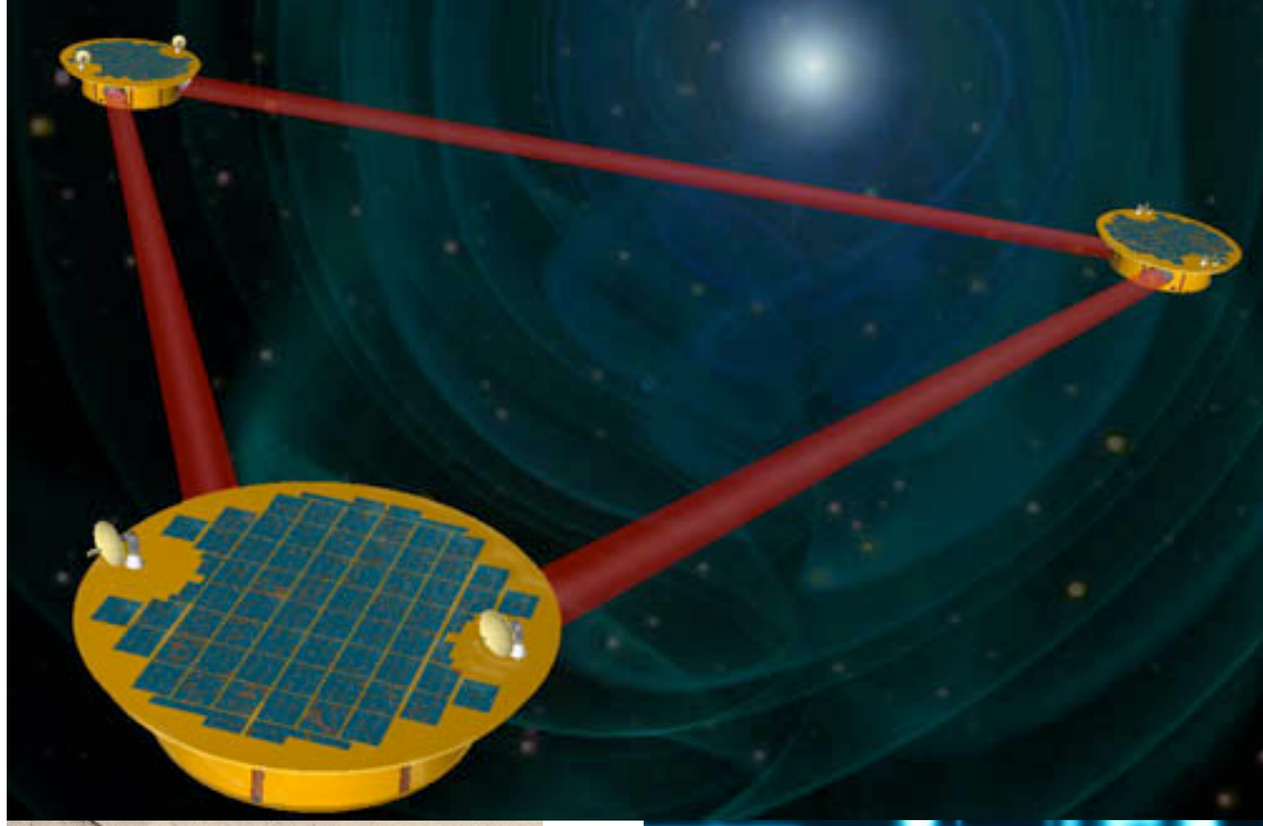
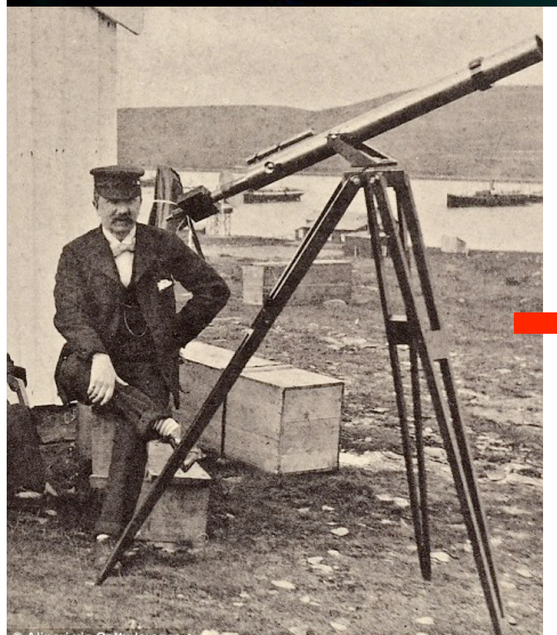
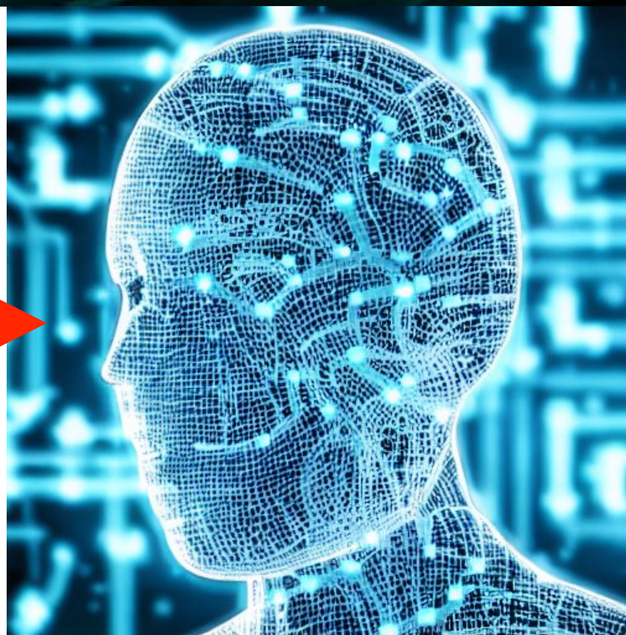
LISA workshop
Traditional methods and machine learning for GW detection

Joe Bayley
Michael Williams

2022-09-21







Introduction

Michael and myself (Joe) are from the University of Glasgow, Scotland.

We primarily work on LIGO data analysis utilising machine learning for many of our projects:

- Detection of Continuous gravitational waves and Compact binary coalescence
- Enhancing Bayesian Parameter estimation for GW signals with ML

Summary

Introduction to tutorial

Traditional methods

Signal processing

Matched filtering

Other search methods

Machine learning

Overview of neural networks

Building, Training, Testing

Application to detection

Aim of tutorial

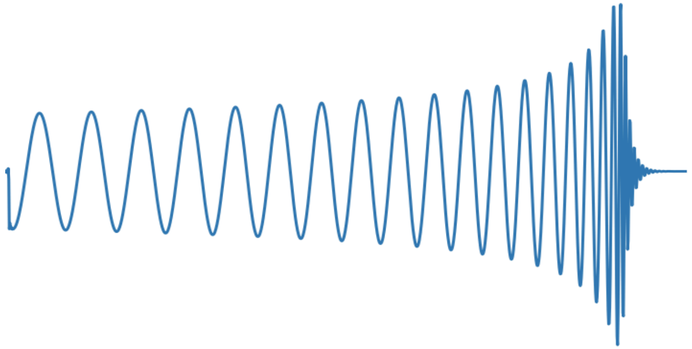
- See an overview of some of the traditional methods used to detect gravitational wave signals
- See how a common detection method is implemented (Matched filtering)
- Understand the background of machine learning (specifically Neural Networks), how they are built, trained and tested.
- Gain some practical experience in building and training a Neural network for gravitational wave detection

Detection

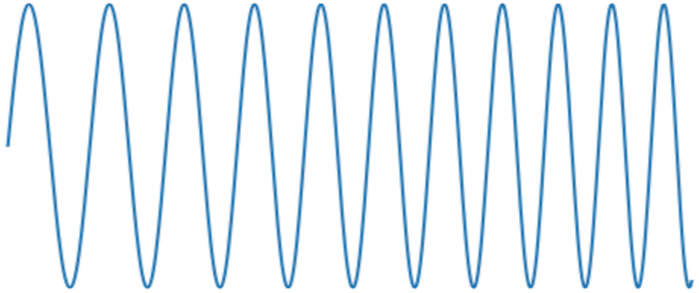
Goal: Detect an astrophysical signal

Different signals may require different approaches for detection.

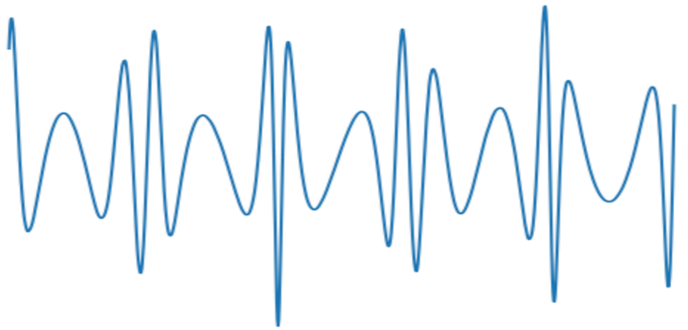
MBHB



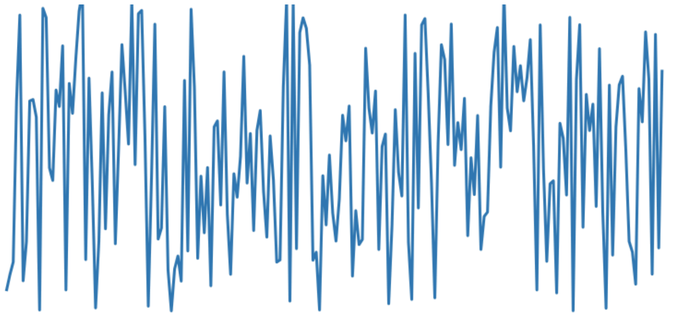
Galactic binaries



EMRIs

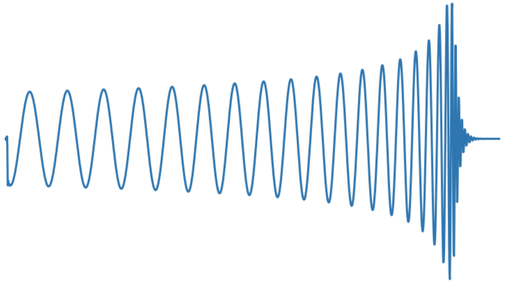


GW background

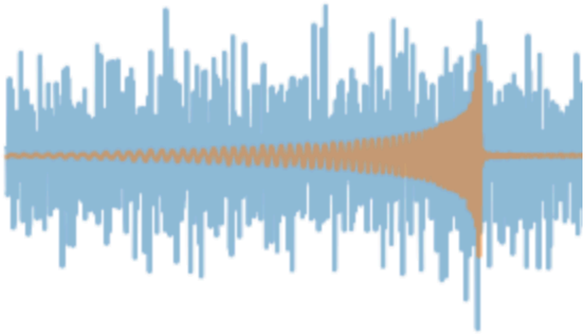


Traditional methods

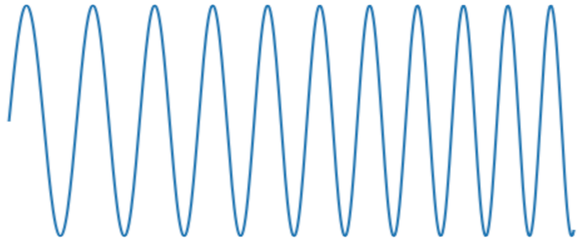
Good knowledge of the waveform



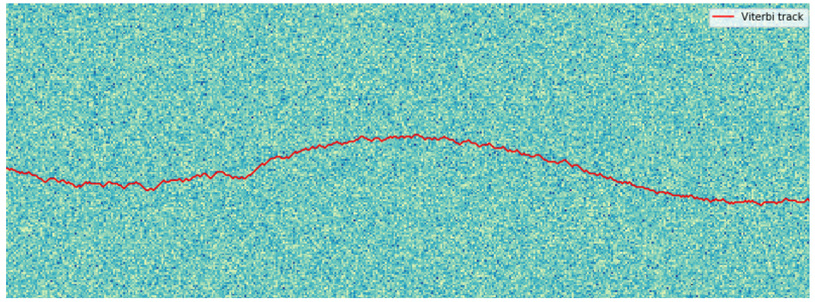
Matched filtering



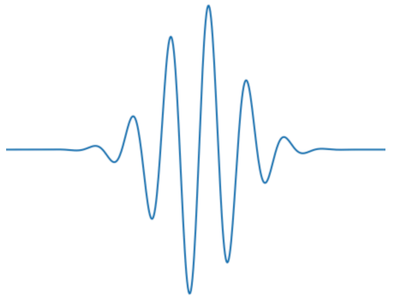
Good knowledge of the waveform but too many templates



Semi-coherent search



Poor knowledge of the waveform



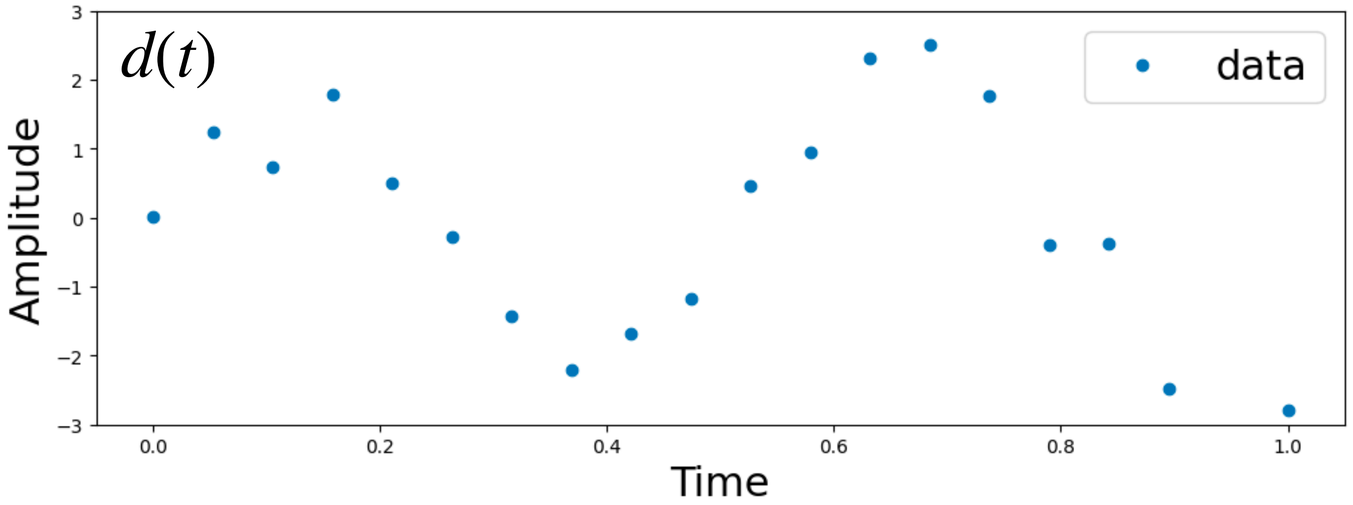
Correlate data between detectors

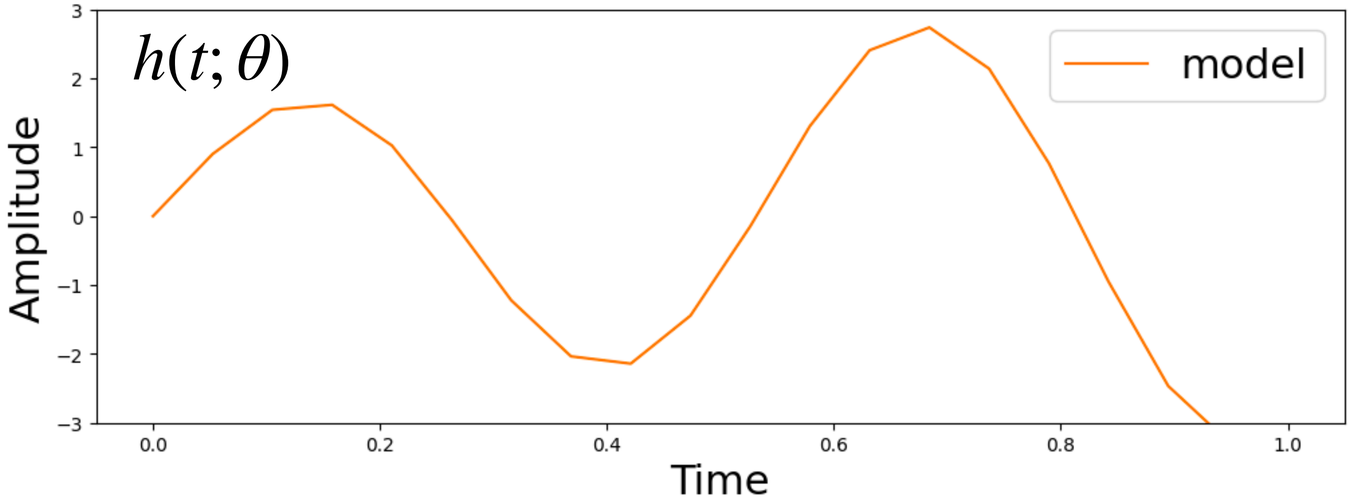
Detector data

Assume additive noise

$$d(t) = n(t) + h(t;\theta)$$

We want to search through data $d(t)$ to identify if a signal model $h(t;\theta)$ is present.





Noise

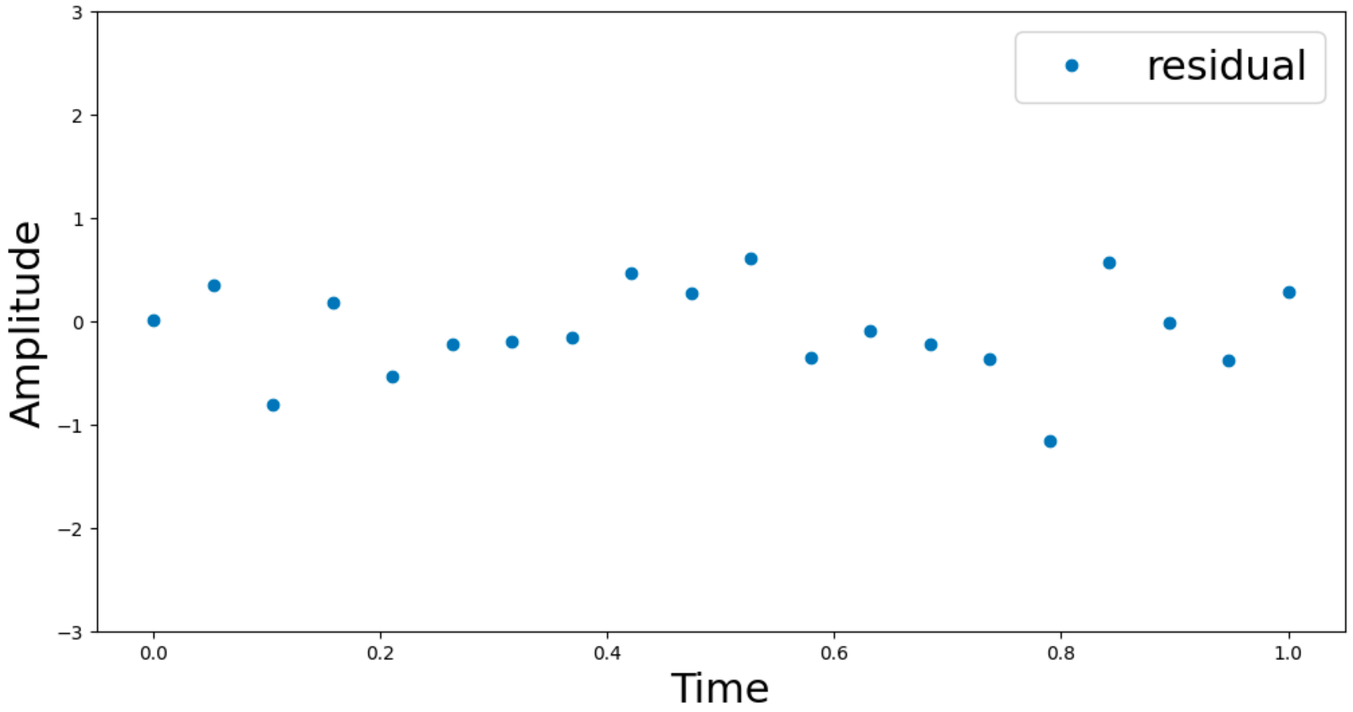
$$d(t) = n(t) + h(t)$$

$$d(t) - h(t) = n(t)$$

Stationary white noise : In this case noise is uncorrelated and drawn from a Gaussian with fixed variance

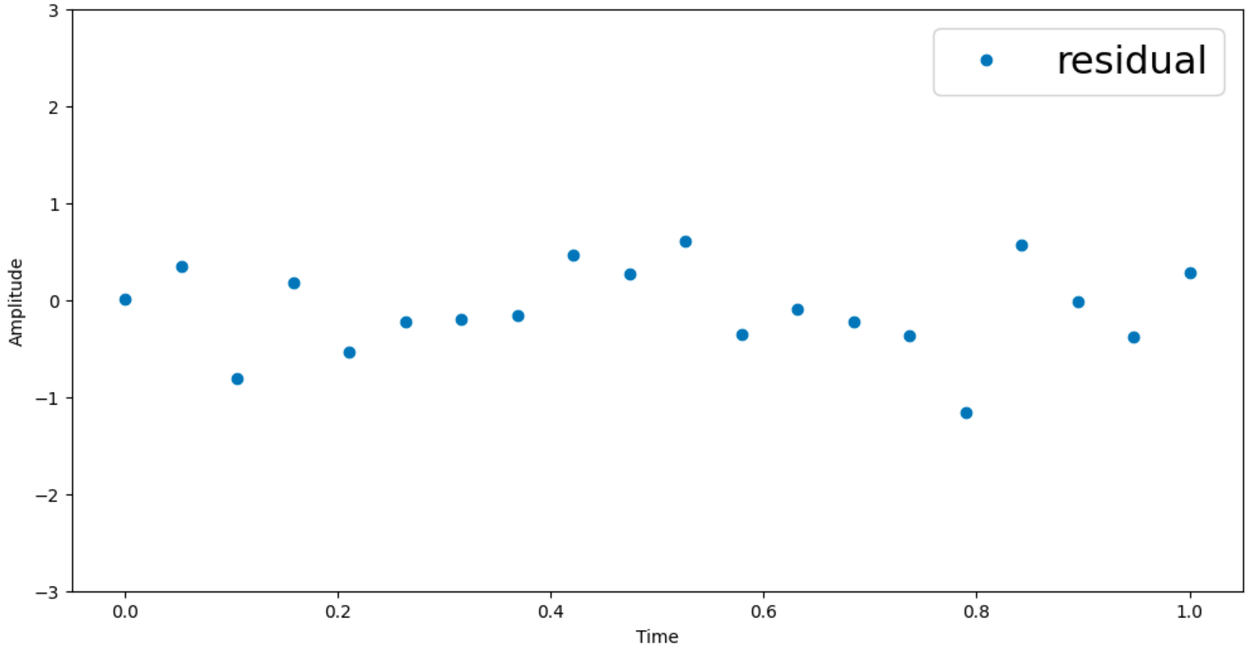
$$p(n_i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{n_i^2}{2\sigma^2}}$$

$$p(n) = \prod_i p(n_i)$$



Likelihood

$$n(t) = d(t) - h(t)$$



The likelihood function is then

$$p(d\,|\,h) = \frac{1}{(2\pi\sigma^2)^{N/2}} \exp\left(-\sum_i^N \frac{(d_i - h_i)^2}{2\sigma^2} \right)$$

White noise

$$C_{ij} = \delta_{ij}\sigma^2$$

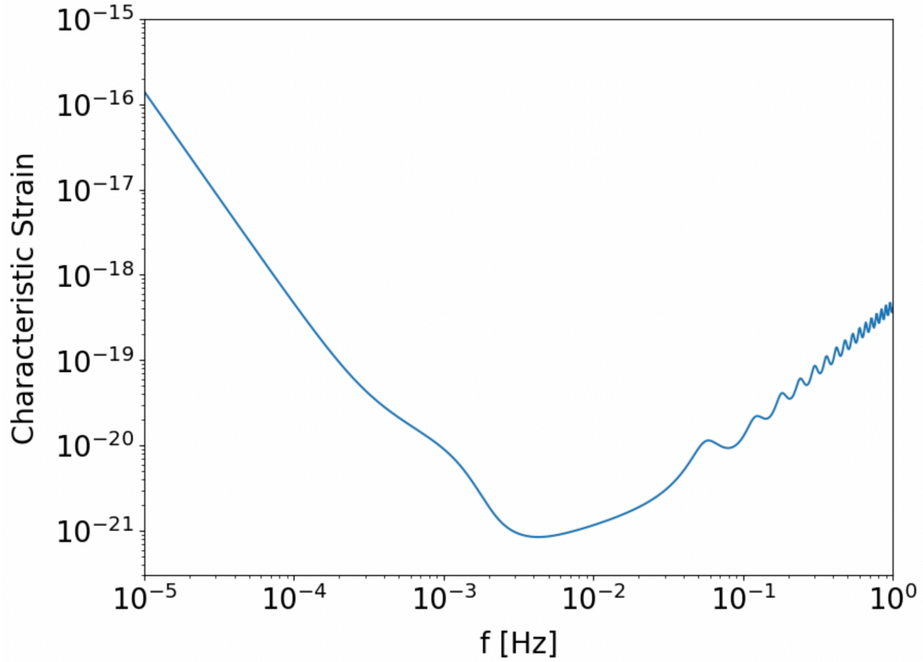
More generally

$$p(d\,|\,h) = \frac{1}{\det(2\pi C)^{N/2}} \exp\left(-\frac{1}{2} \sum_{ij} (d_i - h_i) C_{ij}^{-1} (d_j - h_j) \right)$$

Noise model

In simple cases we can assume that the noise is stationary and Gaussian distributed following some PSD, i.e.

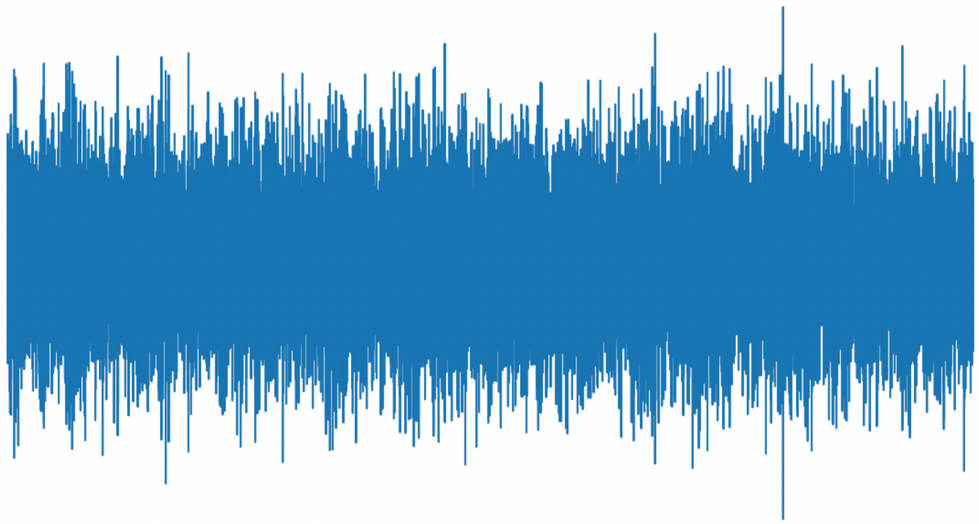
$$C_{ij} = \delta_{ij} S_n(f_i)$$



https://arxiv.org/abs/1803.01944

Multiple possible noise components make up the PSD:

- Instrument noise
- Signal confusion noise

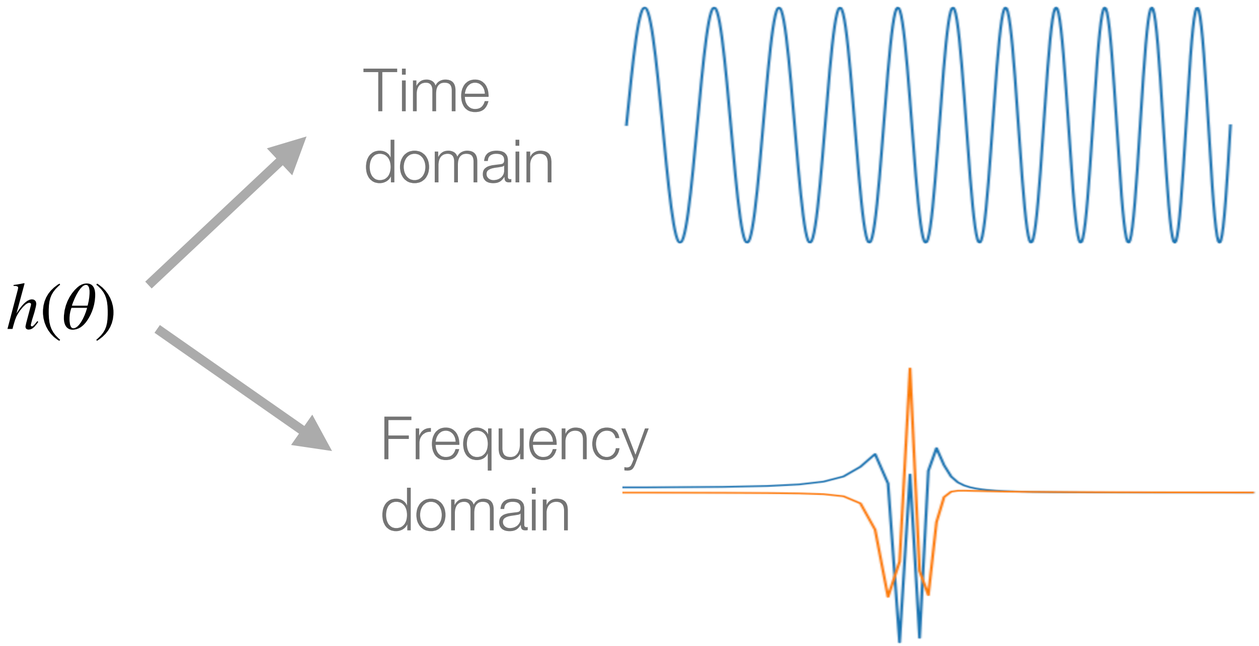


Signal model

Signal model can be anything you like

In the tutorial tomorrow we will use a Massive black hole binary model in the LISA band.

The faster these can be generated the better (see Michael Katz's talk later)



For tutorial tomorrow we will use

IMRPhenomD

Compact binaries, Aligned spin, non precessing

Matched filter

Want to find an optimal filter $K(t)$

$$\hat{d} = \int_{-\infty}^{\infty} d(t)K(t)dt$$

That maximises the SNR

$$\frac{S}{N} = \frac{\left\langle \hat{d} \right\rangle_{h=h}}{\left[\left\langle \hat{d}^2 \right\rangle_{h=0} - \left\langle \hat{d} \right\rangle_{h=0}^2 \right]^{1/2}}$$

Expectation value of $\left\langle \hat{d} \right\rangle$ when signal is present

Root mean square of $\left\langle \hat{d} \right\rangle$ when no signal is present

Matched filter

Want to find an optimal filter $K(t)$

$$S = \left\langle \hat{d} \right\rangle_{h=h} = \int_{-\infty}^{\infty} \langle d(t) \rangle \, K(t) dt$$

$$= \int_{-\infty}^{\infty} \left(\langle n(t) \rangle + \langle h(t) \rangle \right) K(t) dt$$

$$= \int_{-\infty}^{\infty} h(t) K(t) dt$$

$$= \int_{-\infty}^{\infty} \tilde{h}(f) \tilde{K}^*(f) df$$

$$N = \left[\left\langle \hat{d}^2 \right\rangle_{h=0} - \left\langle \hat{d} \right\rangle_{h=0}^2 \right]^{1/2}$$

$$= \left\langle \hat{d}^2 \right\rangle_{h=0}^{1/2} = \left\langle \hat{n}^2 \right\rangle^{1/2}$$

$$= \left[\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \langle n(t) n(t') \rangle \, K(t) K(t') dt dt' \right]^{1/2}$$

$$= \left[\int_{-\infty}^{\infty} \frac{1}{2} S_n(f) \, | \, \tilde{K}^*(f) \, |^2 df \right]^{1/2}$$

Matched filter

We can define the noise weighted inner product

$$(a\,|\,b) = 4\mathscr{R}e\int_0^\infty \frac{\tilde{a}(f)\tilde{b}^*(f)}{S_n(f)}df$$

So we can write the SNR as

$$\frac{S}{N} = \frac{(u\,|\,h)}{(u\,|\,u)^{1/2}}$$

Assume the optimal filter $K \propto h$ so can define the optimal SNR

$$u = \frac{1}{2}S_n(f)\tilde{K}(f)$$

$$\left(\frac{S}{N}\right)_{\text{opt}} = \frac{(h\,|\,h)}{(h\,|\,h)^{1/2}} = (h\,|\,h)^{1/2}$$

We have assumed so far that $\langle d\rangle = h$ however in practice we only have the detector output d not its expectation value, so we can approximate

$$\rho = \frac{S}{N} \approx \frac{(h\,|\,d)}{(h\,|\,h)^{1/2}}$$

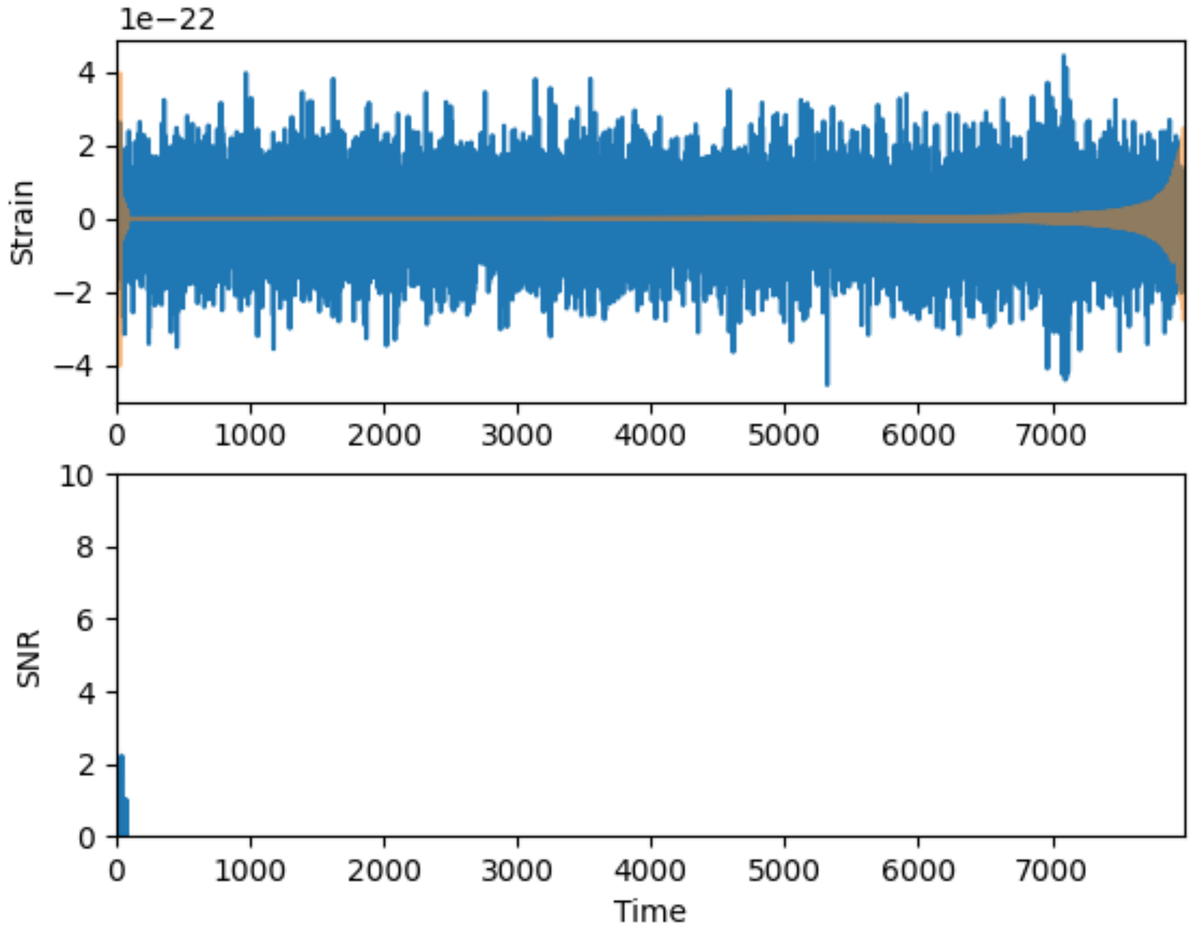
Matched filter

The matched filter time series is defined by

$$(d\,|\,h)(t) = 4\mathscr{R}e\int_0^\infty \frac{\tilde{d}(f)\tilde{h}^*(f)}{S_n(f)}e^{-2\pi i f t}df$$

The matched filter SNR can then be found with

$$\rho(t) = \frac{(d\,|\,h)(t)}{\sqrt{(h\,|\,h)}}$$



Matched filter search

For a search we want to compare many different templates to the data

- How many templates?
- Where in parameter space?

Have a template h and noise n

$$E[(n + \mathscr{A}h \,|\, h)] = \mathscr{A}$$

If the signal s is different from the template h

$$E[(n + \mathscr{A}s \,|\, h)] = \mathscr{A}(s \,|\, h)$$

The match between two templates is

$$M = (h(\theta) \,|\, h(\theta + \Delta\theta))$$

and is a measure of the fraction of SNR retained by using a template $h(\theta)$ to filter a signals with parameters $\theta + \Delta\theta$

$$M \approx (h(\theta) \,|\, h(\theta)) + \frac{1}{2}\frac{\partial^2 M}{\partial\theta^i \partial\theta^j}\Delta\theta^i\Delta\theta^j$$

Mismatch between neighbouring templates

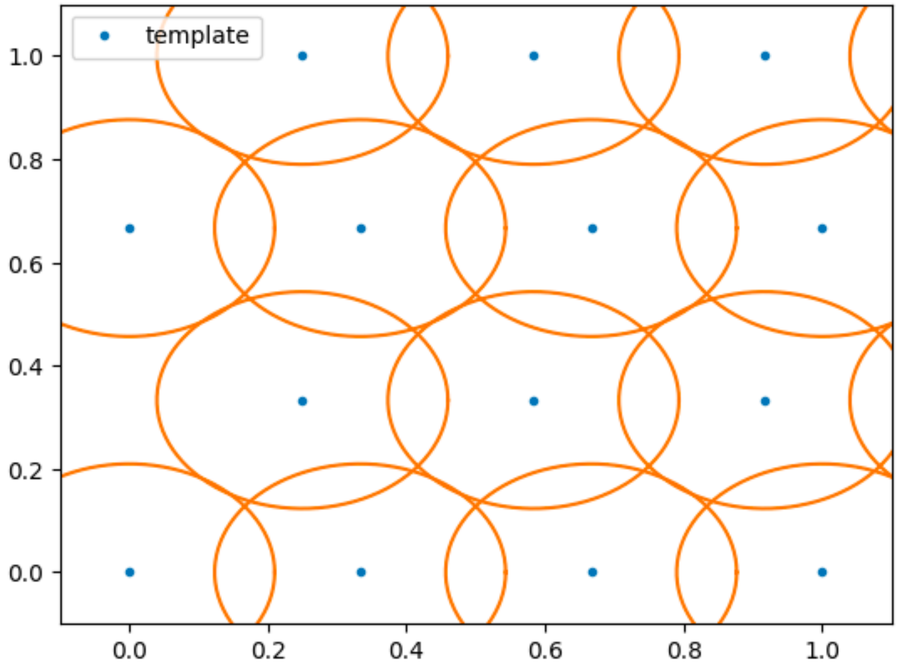
$$1 - M \approx g_{ij}\Delta\theta^i\Delta\theta^j$$

https://link.aps.org/doi/10.1103/PhysRevD.53.6749

Template placement

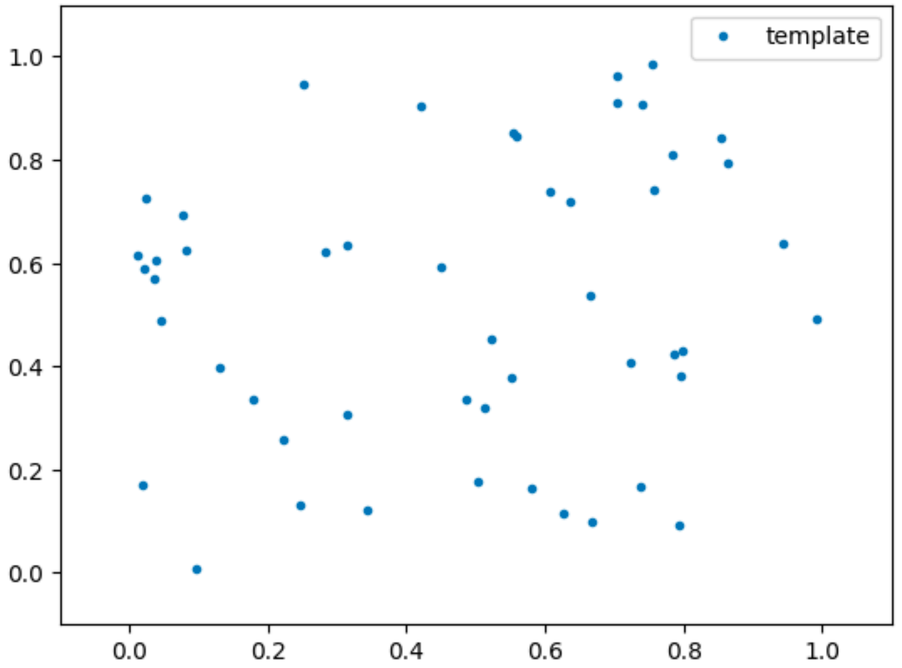
Grid placement

- Placement is defined by metric g_{ij}
- Spaced according to some mismatch



Random placement

- Number of templates defined by metric g_{ij}
- Can be more efficient in high dimensions



Monte caro - MCMC

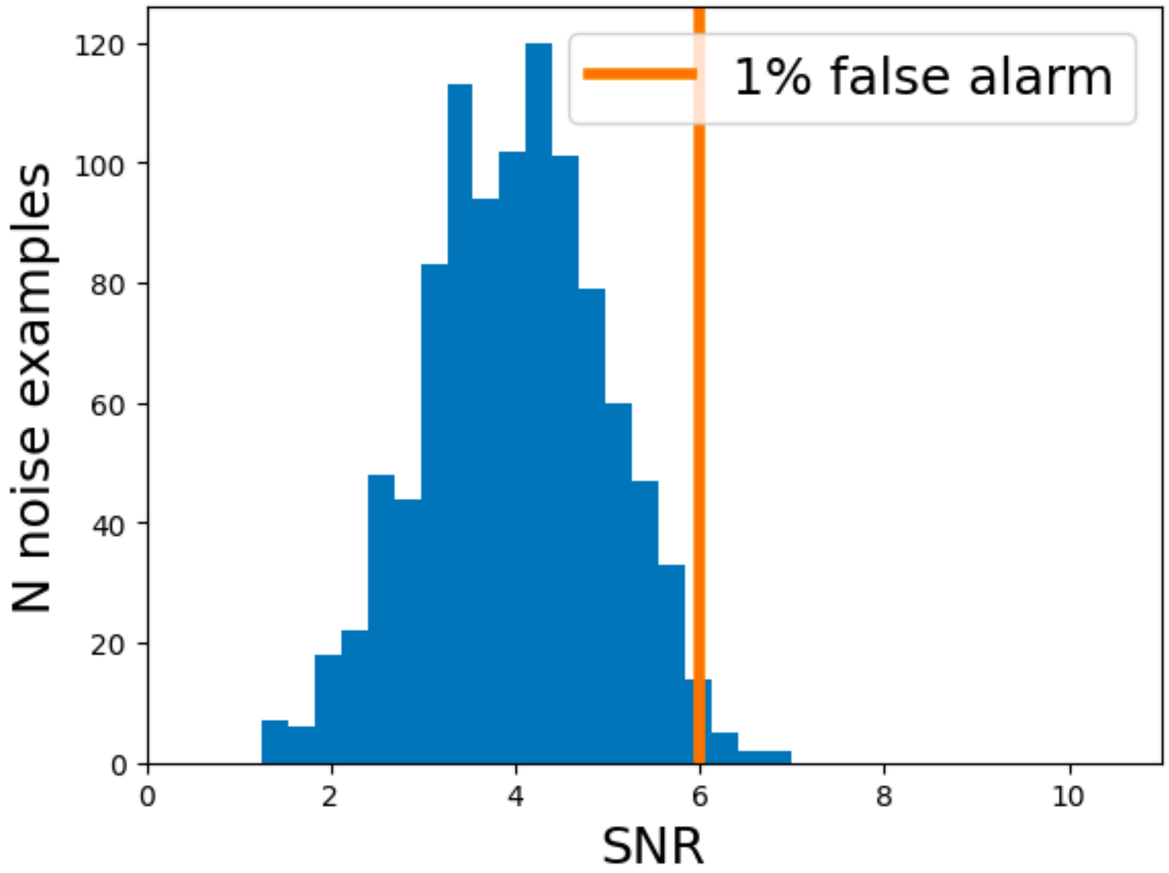
Significance

We have just been using stationary white Gaussian noise.

Data is not often exactly distributed like this but has many other artefacts

We can empirically estimate the significance of a matched filter SNR by estimating noise background

Then set a False Alarm Rate

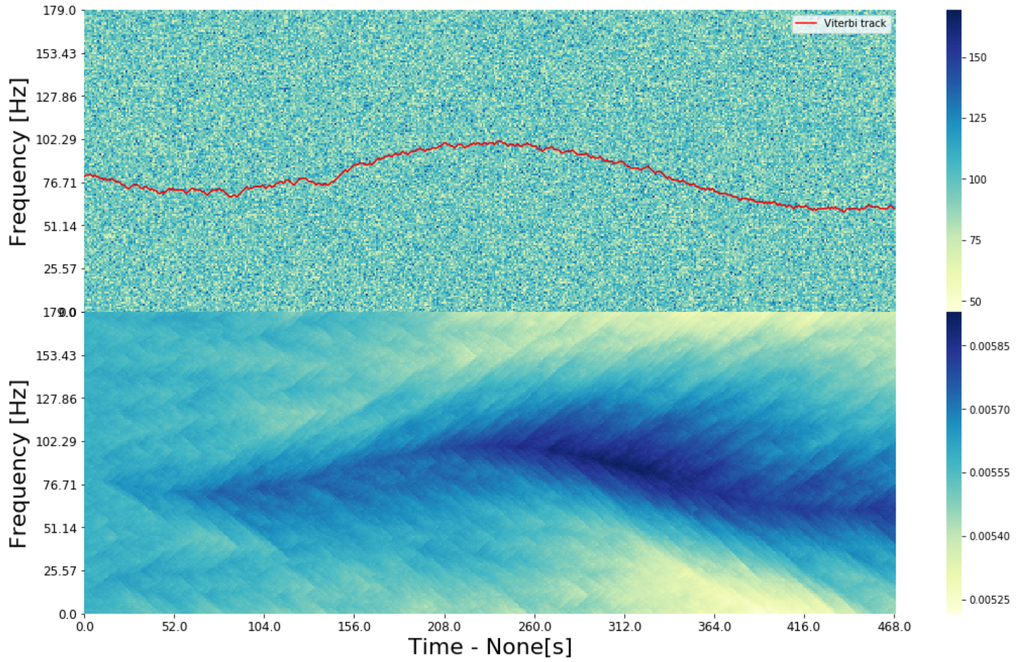


Other search methods

Semi-coherent methods

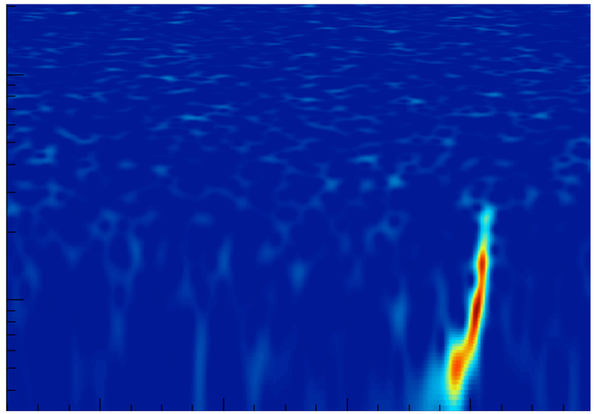
- Break data into smaller segments and run matched filter on segments

- Search for tracks in power spectrum



CWb

- Uses a wavelet transform and correlates power between detectors

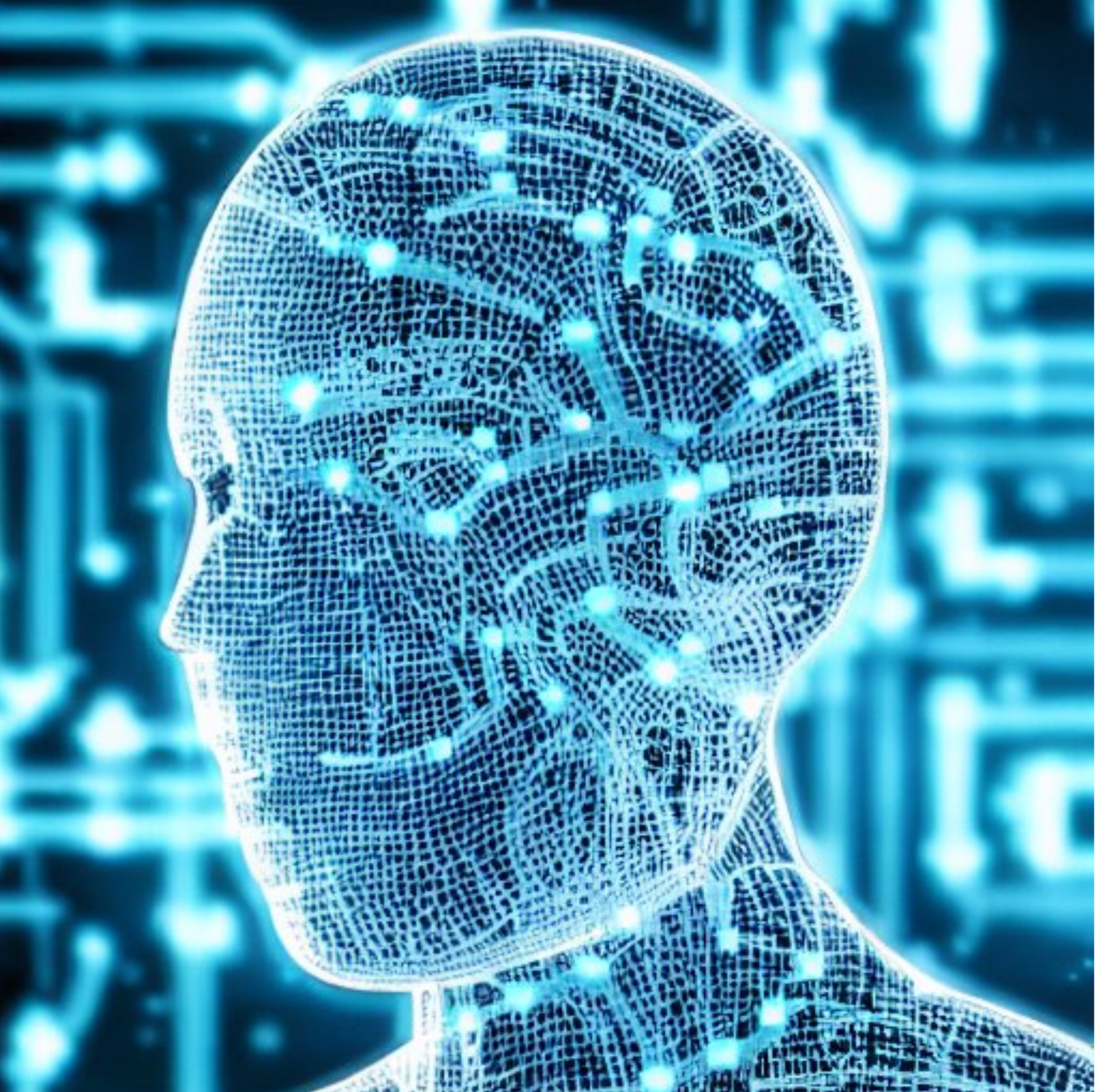


Limitations

Matched filtering can be limited if signal model is not known well

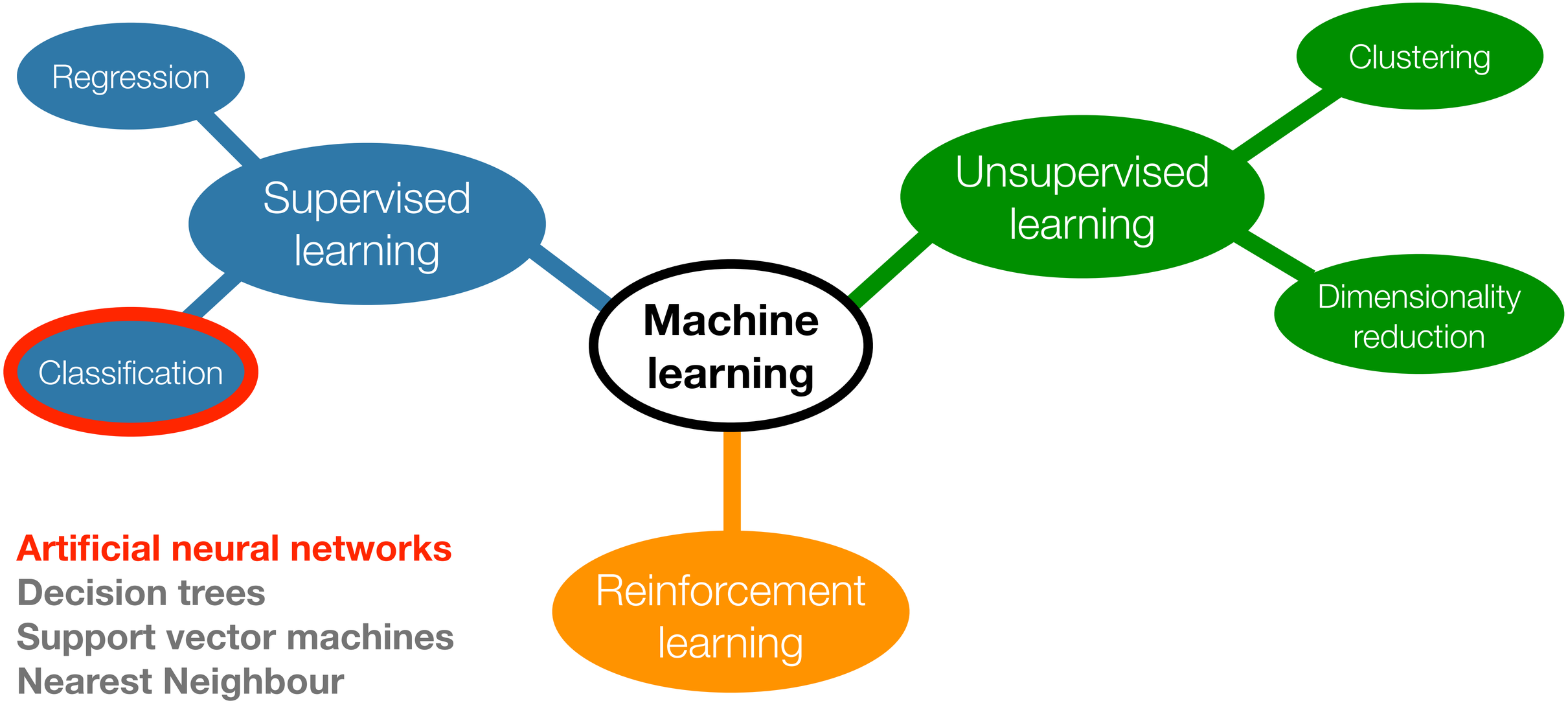
If too many templates need to be generated then it is computationally not feasible to run a complete templated matched filter search.

Other semi-coherent methods are limited in their sensitivity and ability to extract parameters



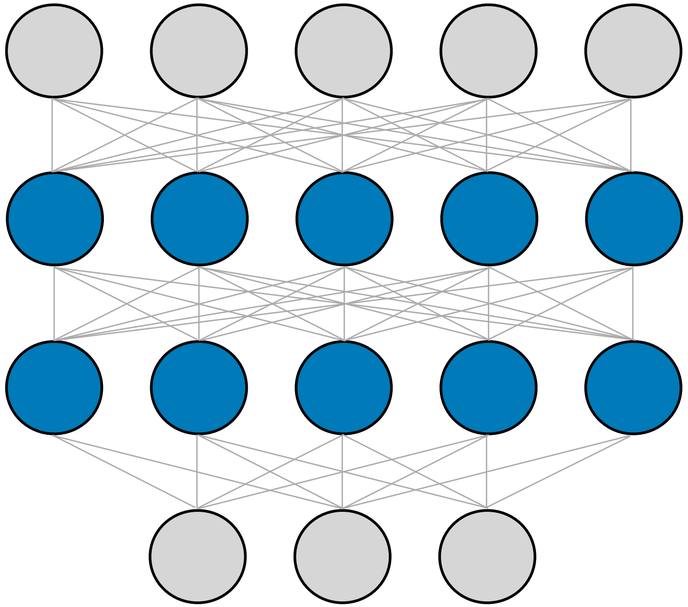
Machine Learning

Generated with: https://stablediffusionweb.com/#demo



Artificial neural networks
Decision trees
Support vector machines
Nearest Neighbour

Neural networks



Artificial neural networks are models which use data to learn and improve over time.

Why use machine learning/neural networks?

- It is very flexible and can be applied to many problems
- Can learn complex relationships between inputs/outputs
- Can offer large speed ups to existing analyses
- You have nothing Toulouse

Neurons

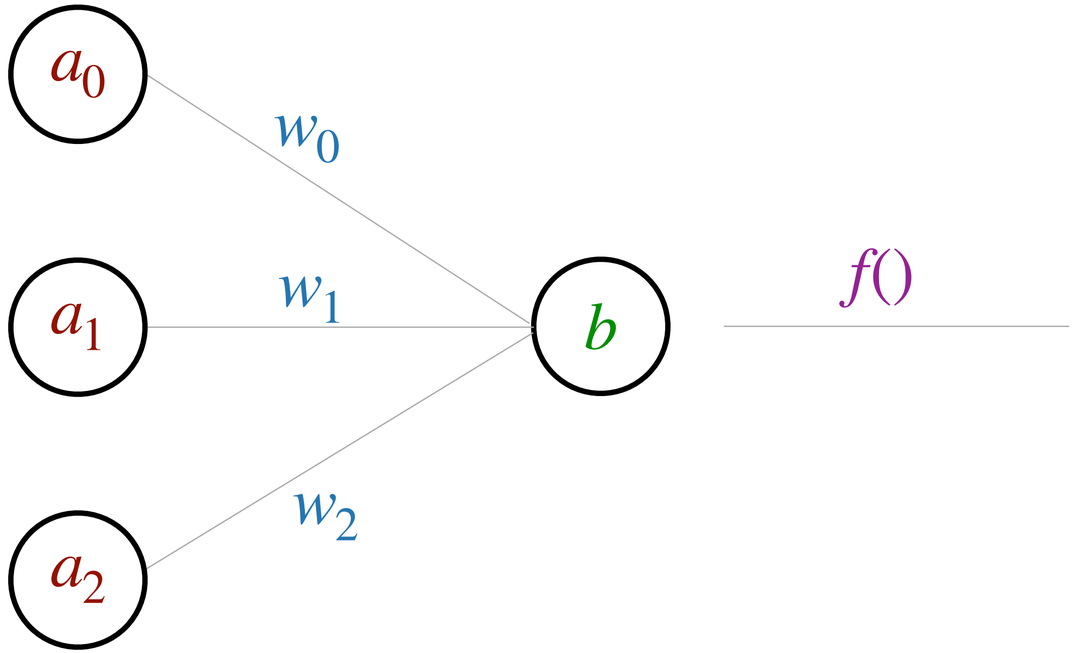
The building blocks of Neural networks are `Neurons'

Inputs - a_0, a_1, a_2

Weights - w_0, w_1, w_2

Bias - b

Activation function - $f()$



$$o = f\left(b + \sum_i a_i w_i\right)$$

Simple network

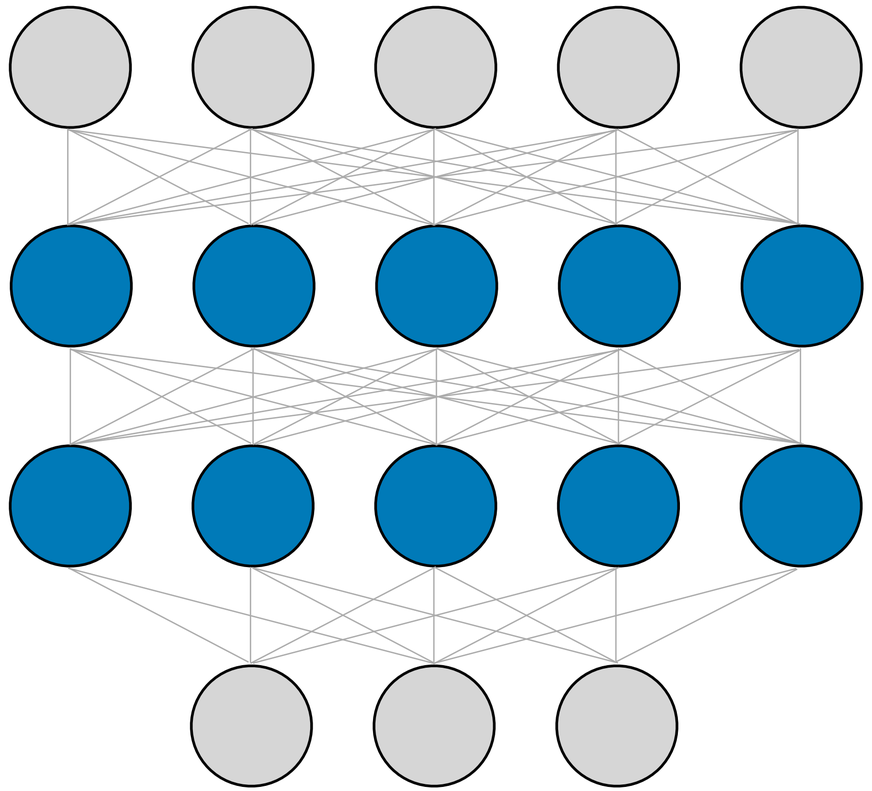
These can then be arranged into a Network or neurons

There are many different ways to arrange these

Deep Learning is just many hidden layers

For classification:

Output 1 Neuron - probability of signal being present

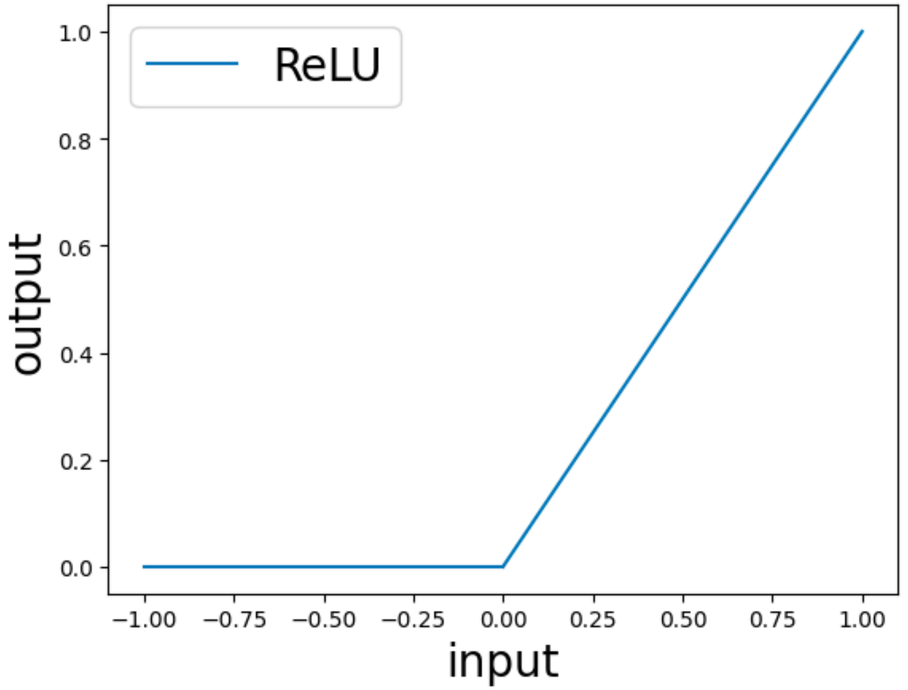


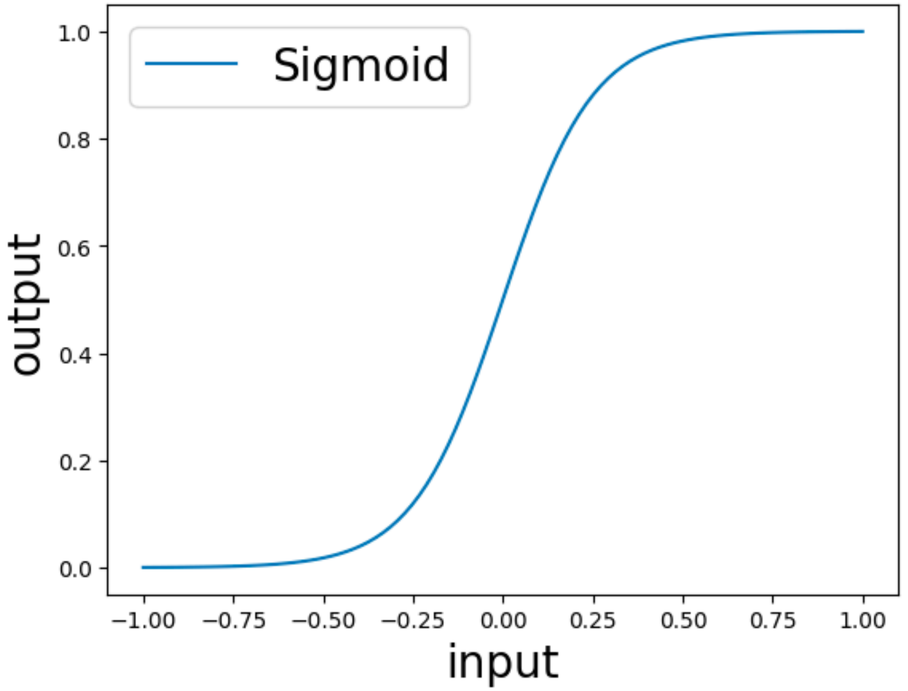
Activation function

The activation function adds non linearities into the network

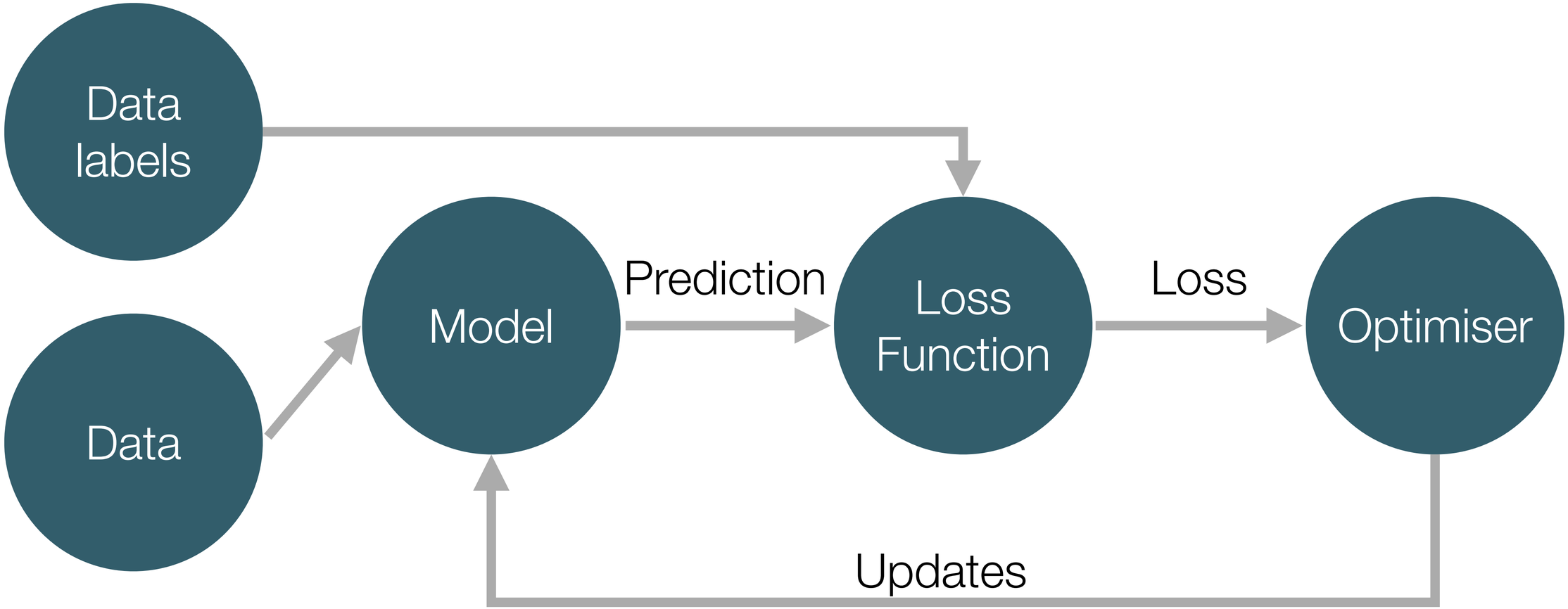
It is key to making neural networks learn effectively.

Many choices of activation functions - useful for different purposes





Training networks



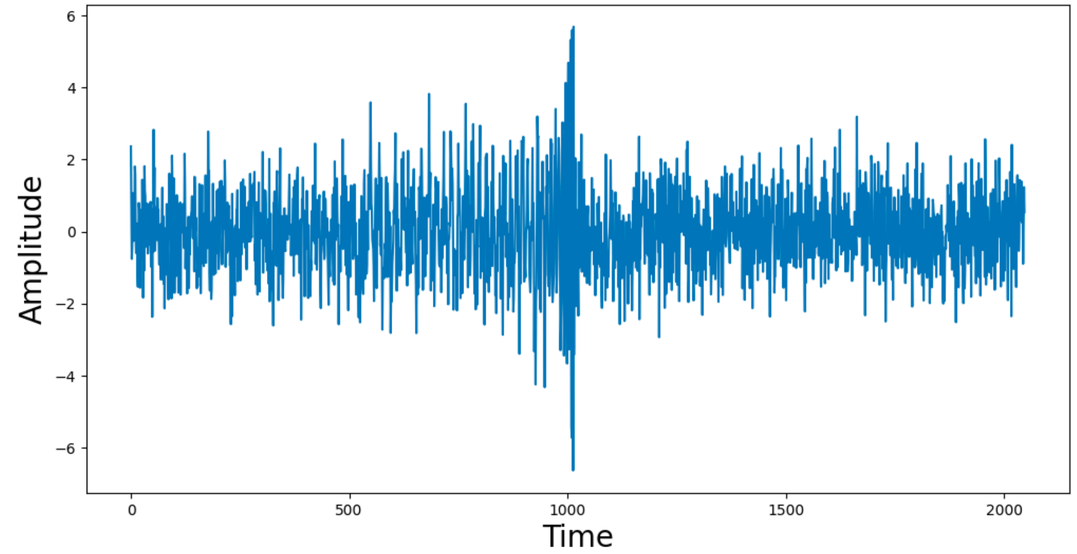
Data preparation/preprocessing

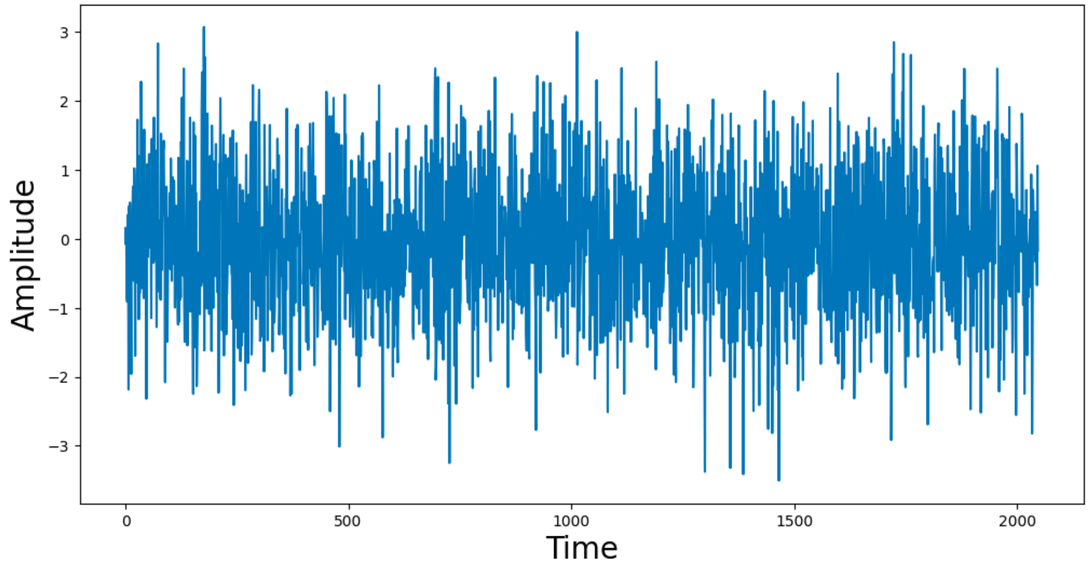
Tomorrow we will use a whitened time series such that the noise is Gaussian with a Variance of 1.

Each of these have a label

For detection:
- 0 for noise
- 1 for signal + noise

For parameter estimation:
- nothing for noise
- signal parameters for signal + noise





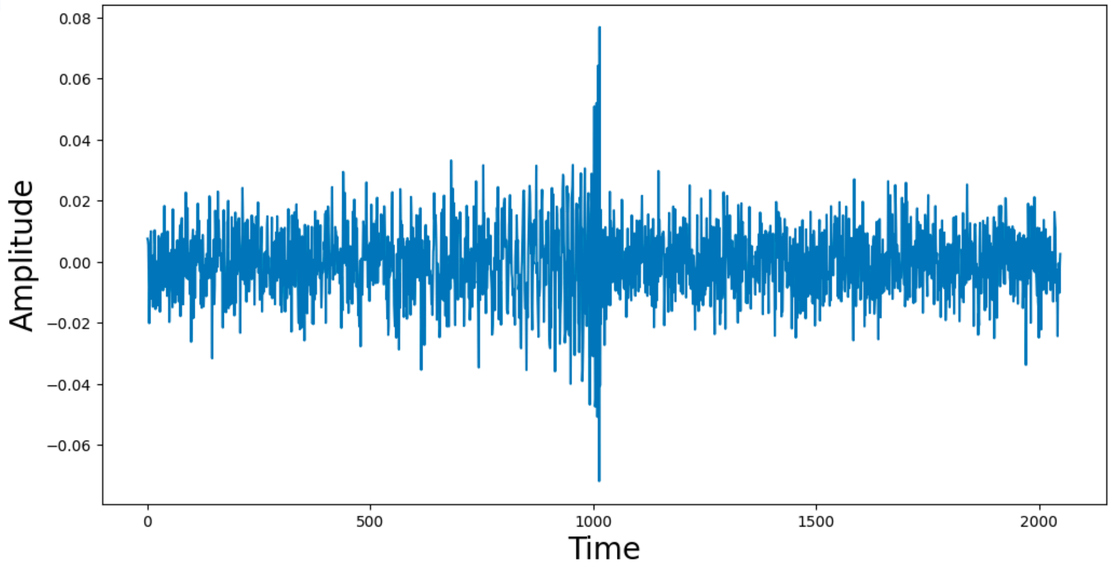
Data preparation/preprocessing

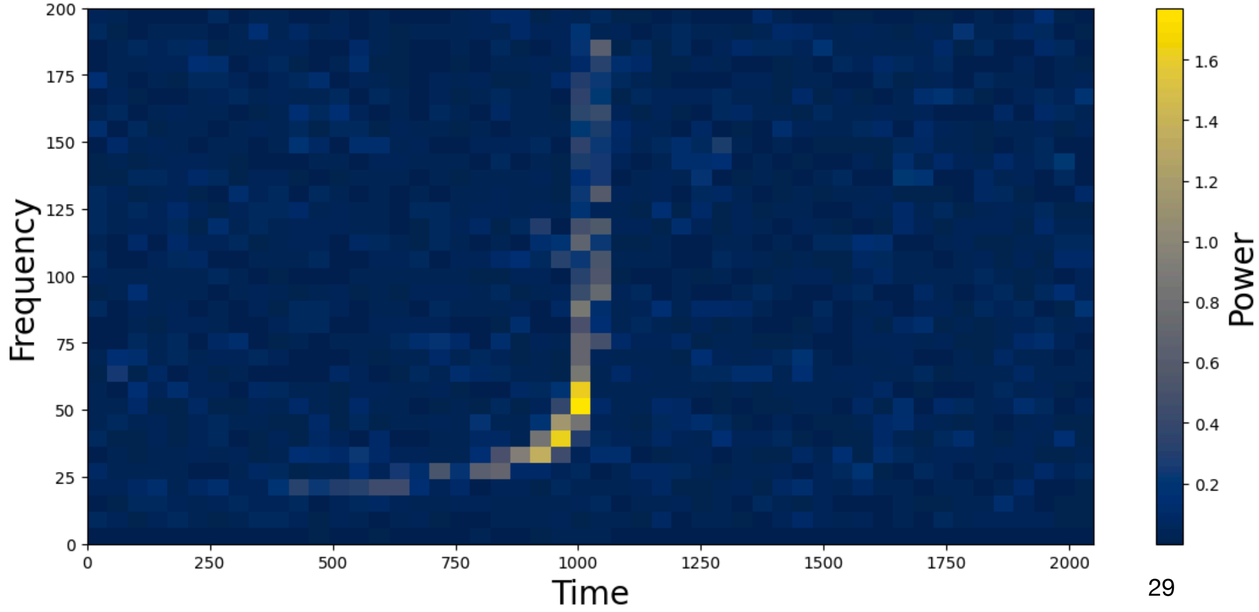
Data preparation is key for networks to learn efficiently

Scaling data - should always try to have values close to 1(both for inputs and for labels)

- Whiten the data

Make sure training data is as close as possible to expected real data





Loss functions

There are many different loss functions for many different purposes

The simplest starting point might be to minimise the absolute difference between the truth and the network estimate

$$L = \sum (p_{\text{true}} - p_{\text{pred}})^2$$

This is the mean squared error

Loss functions

For the detection case we want to compute the likelihood that a signal is present within the data.

This (log) likelihood is the binary cross entropy

$$L = \sum p_{\text{true}} \log(1 - p_{\text{pred}}) + (1 - p_{\text{true}}) \log p_{\text{pred}}$$

Here our p_{true} is our data label (either 0 or 1) and p_{pred} is out network output.

You have to be aware of the output activations of the neural network, in this case they need to be scaled between 0 and 1, e.g. using a sigmoid function

Optimisers

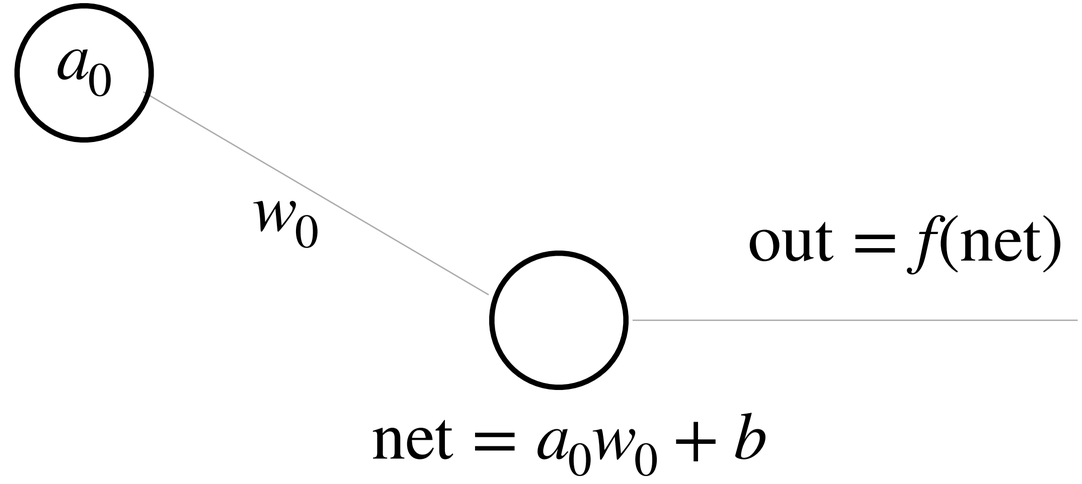
Optimisers are what compute the gradient for each parameter based on the loss

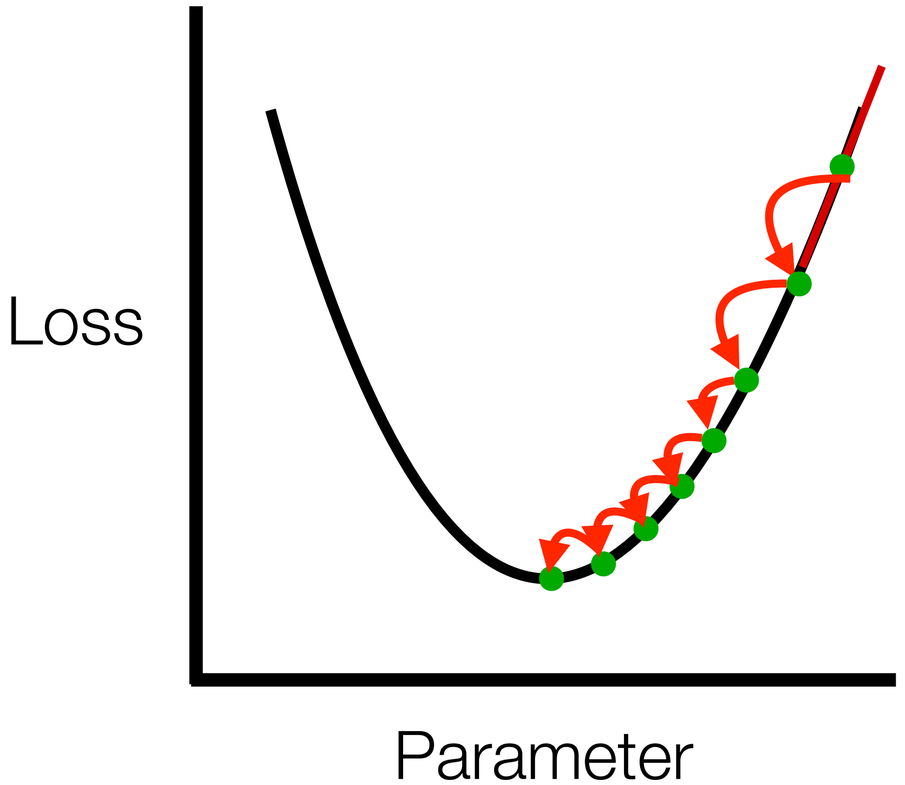
$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial \text{out}_i} \frac{\partial \text{out}_i}{\partial \text{net}_i} \frac{\partial \text{net}_i}{\partial w_i}$$

And update the parameters based on that gradient

$$w_{\text{new}} = w_i + \alpha \frac{dL}{dw_i}$$

α= learning rate



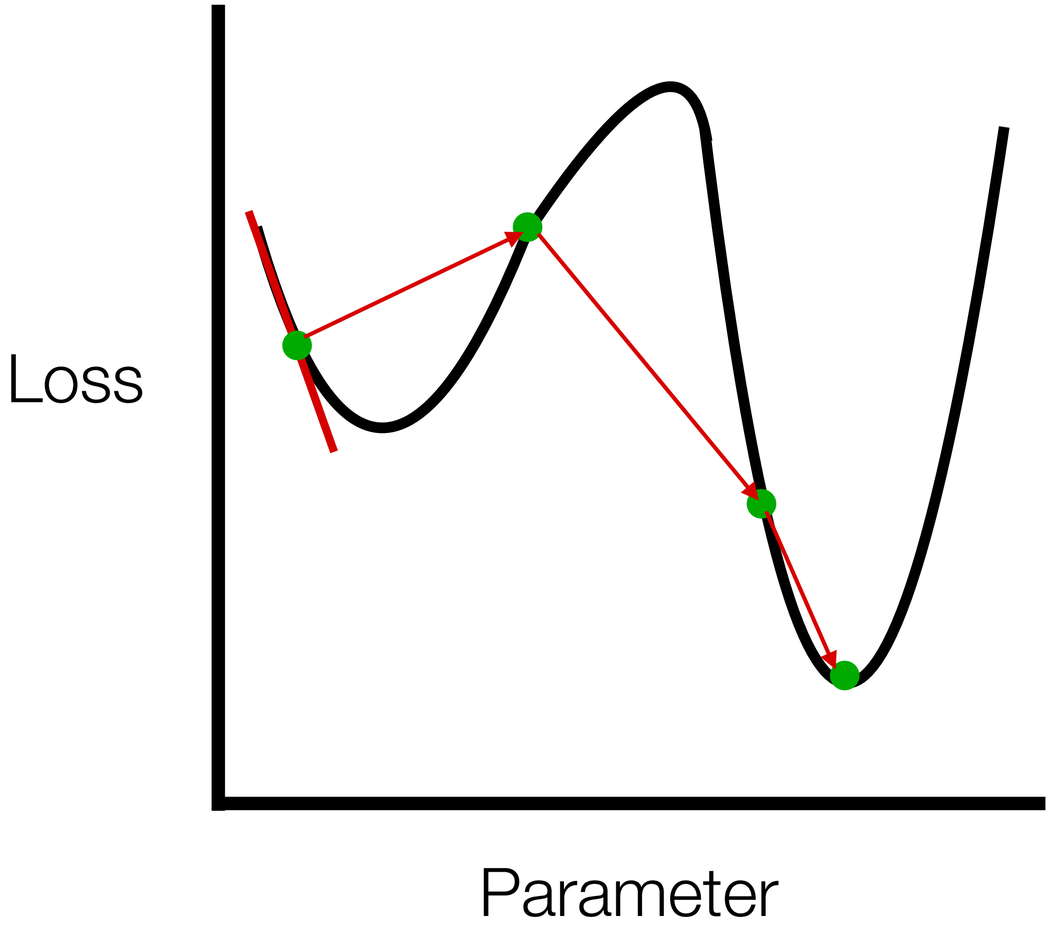


Optimisers

Stochastic gradient descent computes gradient over smaller batches

Many additions to this including momentum, weight decay etc

Tomorrow we will be using the ADAM optimiser which includes these additions

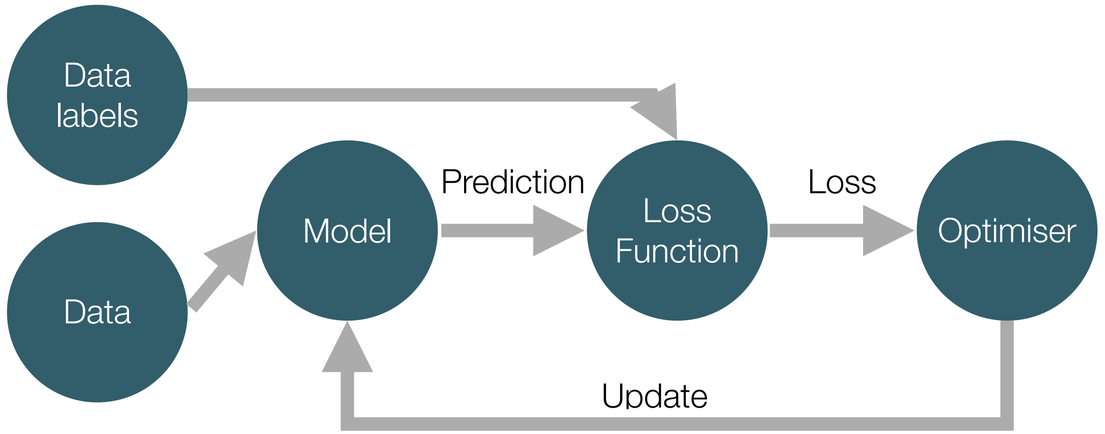


Training a network

The training process involves calculating the loss and updating the optimiser and network parameters for many examples of data.

We generally batch data - to fit as much data on GPU as possible

```
for epoch in epochs:
    for batch, label in data:
        model_output = model(batch)
        loss = loss_fn(model_output, label)
        gradients = optimiser(loss)
        model.update(gradients)
```

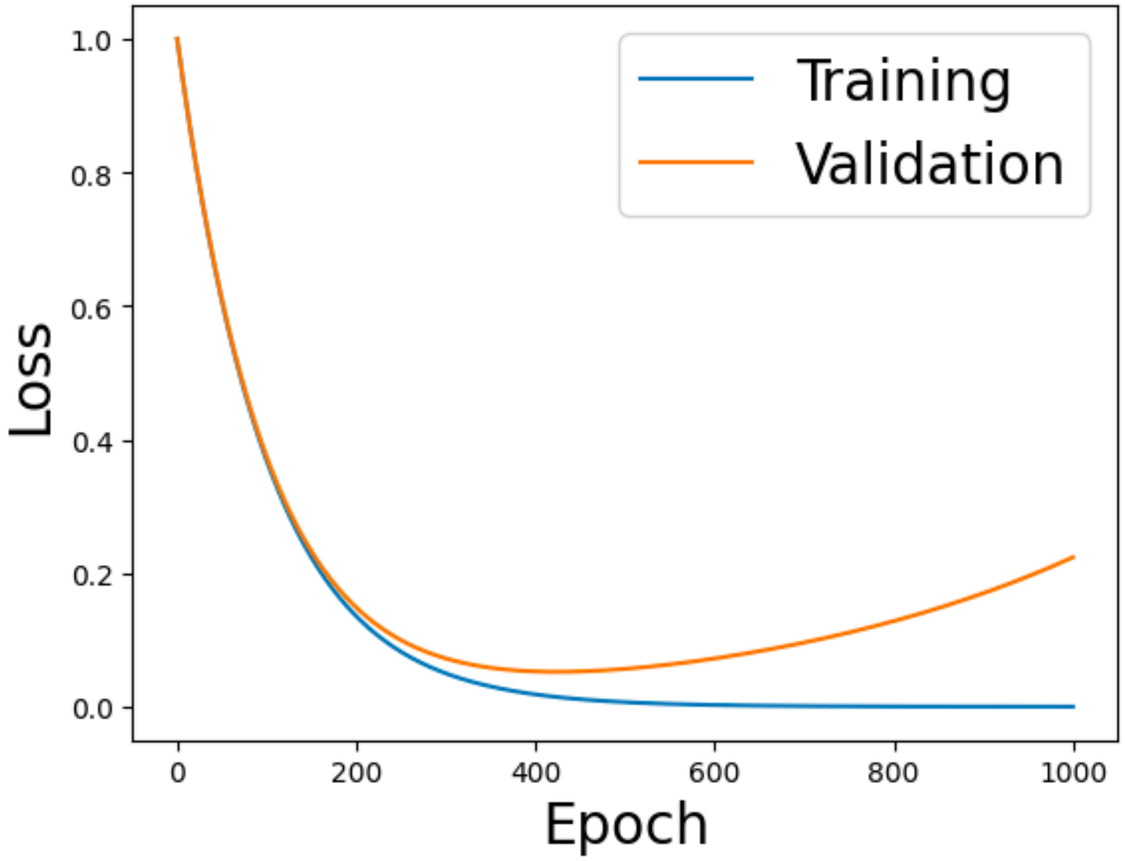


Validation

Validating the network is important to prevent overfitting to the training data

Overfitting is where the network learns the training data, not the underlying structure in the data.

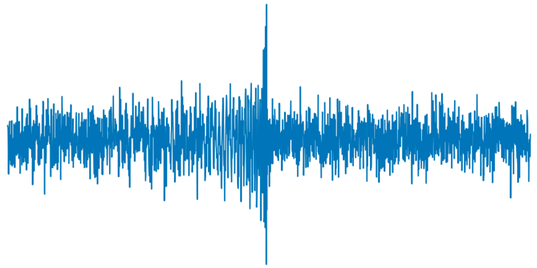
Compute loss during training without updating network.

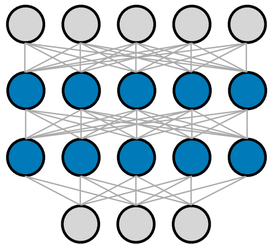


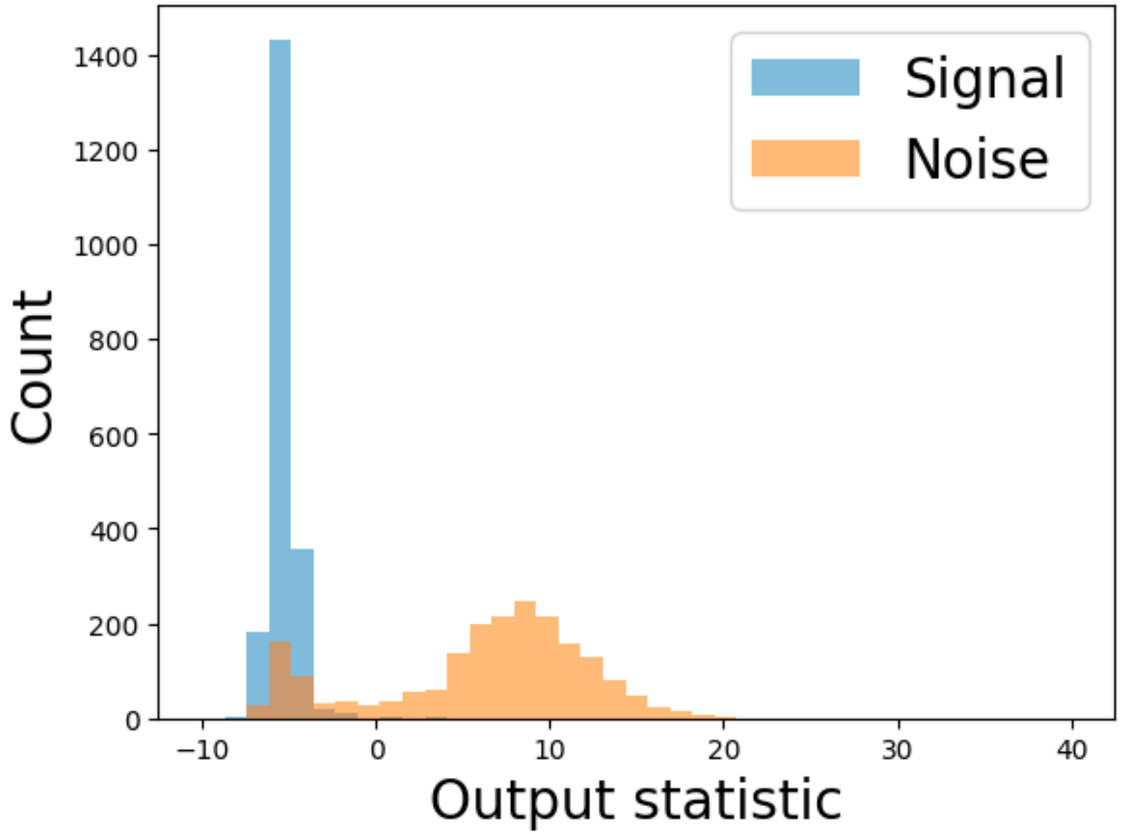
Testing data

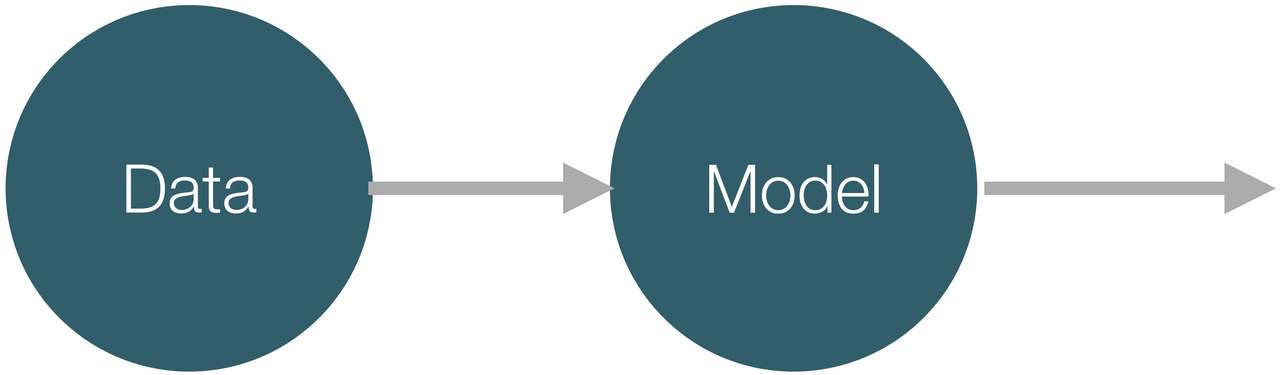
Testing data is to test the final output of the network

Why have testing and validation data?







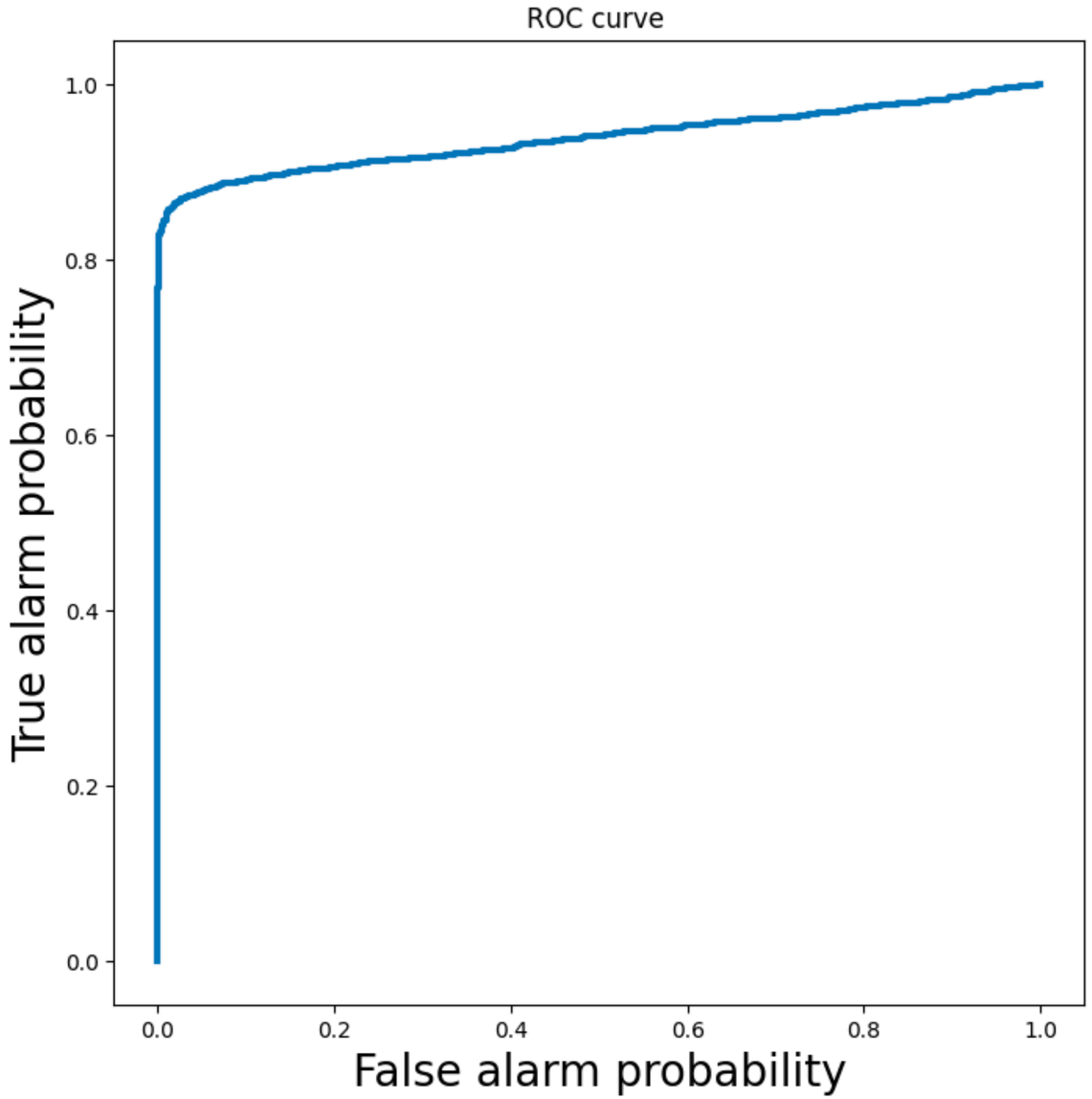


Evaluating performance

There are many metrics to look at to evaluate the performance on testing data

- Loss
- Accuracy
- True positive rate

For detection using the loss and ROC curves are good metrics



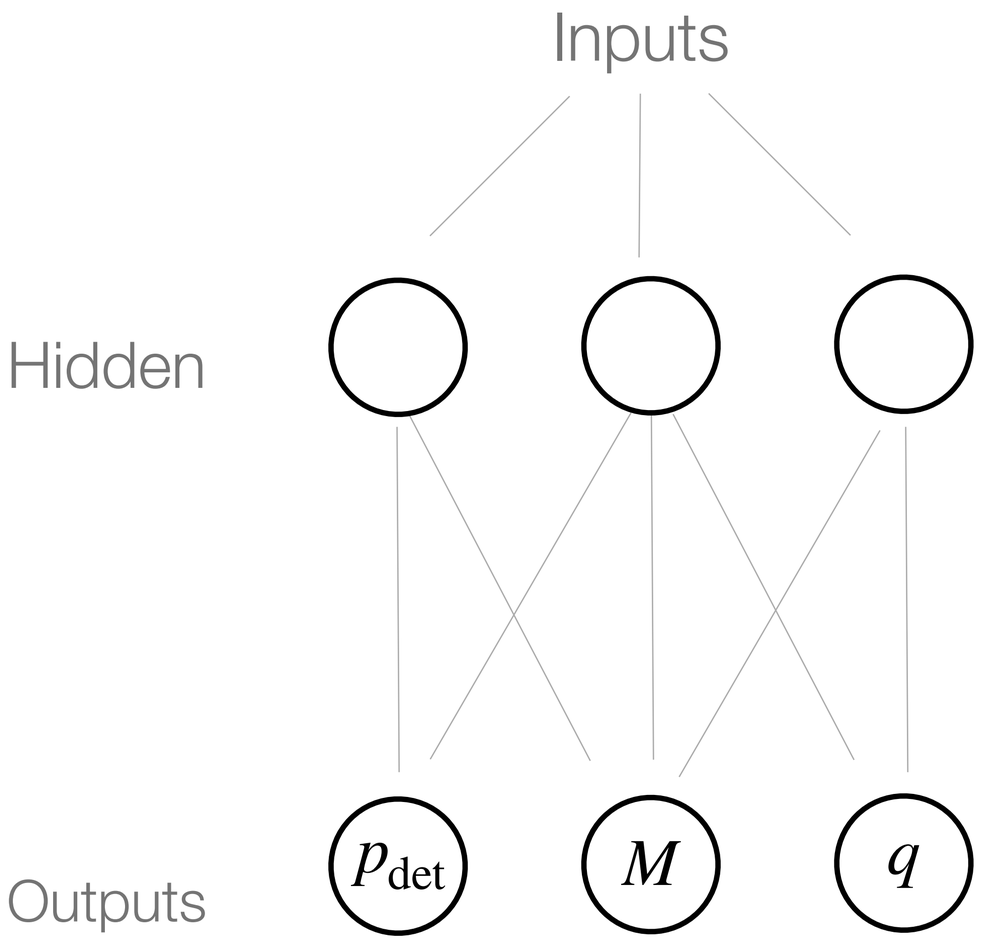
Parameter prediction

The outputs of the above network are

Signal or no signal

However, whilst useful information we really want more to give a followup search

We can also return estimates of parameter signals



Other networks

Convolutional layers (CNNs)

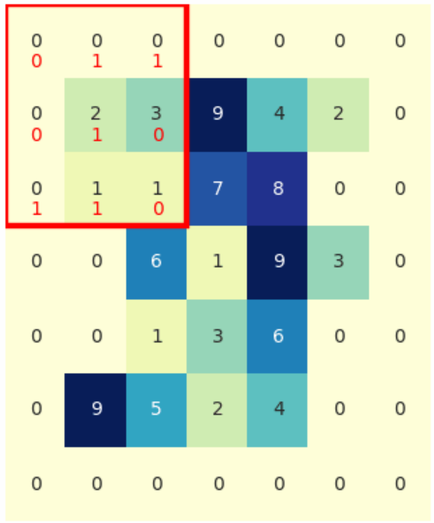
ResNets

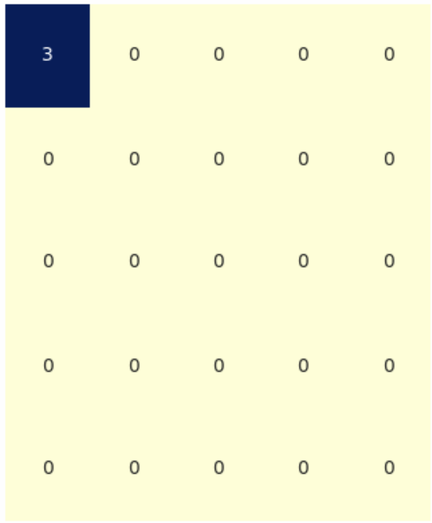
Auto-encoders

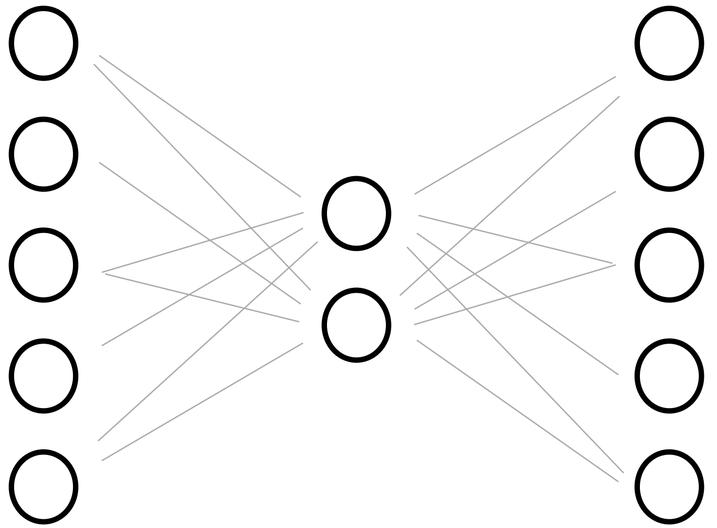
Flows

Transformers

Recurrent Neural Networks







Summary

Seen an overview of some of the traditional methods used to detect gravitational wave signals

Introduced some background of machine learning (specifically Neural Networks)

Showed how they can be used for detection and basic parameter estimation

Tomorrow we will put some of these things into practice!