# Class 4: Probabilistic Machine Learning

Andrew Parnell
andrew.parnell@mu.ie


PRESS RECORD

https://andrewcparnell.github.io/intermediate_ML

# Learning outcomes

- Understand the basics of Bayesian inference and how it relates to machine learning and uncertainty estimation
- Introduction to Bayesian Additive Regression Trees, Bayesian Gaussian Processes, and Bayesian Adaptive Smoothing Splines
- Code and discussion of all of these models

# Bayesian inference

- Rather than getting single 'best' estimates of the weights and biases in a neural network, imagine instead that you got a probability distribution of them instead
- These probability distributions arise as **samples** from the probability distribution
- For example, for a hidden layer with 3 inputs and 4 outputs you would have 12 weights (and 4 biases) so 16 parameters altogether structured in a matrix. The Bayesian version would have many thousand of these matrices, all of which are equally likely given the data presented to that layer
- You could then take the mean or any other sample statistic of these weights to push predictions through the NN
- Having uncertainty means you can make **probabilistic predictions** from your machine learning model

# Bayesian inference 2

- The same idea applies to any other machine learning model where a single best set of parameters is estimated
- The price you pay for getting the full probability distribution is a huge extra computational burden
- For this reason most Bayesian machine learning programs fit simpler models with far fewer parameters than a standard NN

# Software for fitting Bayesian models

There are lots of software packages for fitting Bayesian models though they mostly focus on statistical rather than complex machine learning models

- `rstan`, `rjags`, `nimble` are R packages that can flexibly fit lots of different types of Bayesian model
- `BART`, `dbarts`, `BartMachine` are R packages that fit a specialised type of Bayesian machine learning model (more on this next)
- There are very few good packages for fitting Bayesian neural network models in R, though the `tfprobability` package fits tensorflow models with a probabilistic layer (not clear if Bayesian or not). Currently very hard to use in R

# How is a Bayesian model different from the models we have already covered?

- In the keras models we have covered already we aim to minimise a loss function subject to the constraints imposed by the layers. We can have many more layers and parameters (weights) than data points
- In a Bayesian model we impose constraints (commonly called prior distributions) to ensure that the model can be fitted and matches our understanding of the data. This can be very application specific
- In keras we find the values of the weights that minimise the loss function. In Bayesian inference we find all the values of the weights that come close to minimising the loss function, keeping only those that are in a reasonable window. We take as many of these as we can (usually 1,000+)

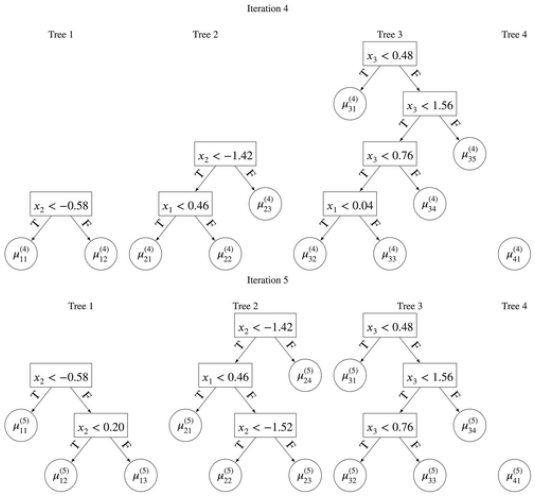
# Introduction to BART

- One of the most popular and flexibly Bayesian machine learning models is Bayesian Additive Regression Trees (BART)
- It works a lot like gradient boosted trees (also a very competitive machine learning technique)
- Very simple to implement:

```
library(BART)
bart_model <- wbart(x.train, y.train, x.test)
```

- Also works for lots of different types of data (count, classification, etc) but not multivariate response (yet)

# Structure of BART




From https://onlinelibrary.wiley.com/doi/abs/10.1002/sim.8347

# BART in maths

The mathematical equation behind BART is:

$$y = \sum_{t=1}^{T} g(X, T_t, M_t) + \epsilon$$

- where $T$ is the number of trees
- $T_t$ are the individual trees
- $M_t$ are the terminal node weights
- $g()$ is a function that looks up the weight for the covariate values $X$
- $\epsilon \sim N(0, \sigma^2)$ is a model error term

For a fixed number of trees the model estimates all the tree structures and all the terminal node weights

# BART output

- When we fit a BART model we get a large list of samples, each sample contains $T$ trees and their associated terminal node weights
- We can add up the trees in each sample to get a probability distribution of the predicted values for each sample
- We can then produce predictions with e.g. confidence intervals on the output

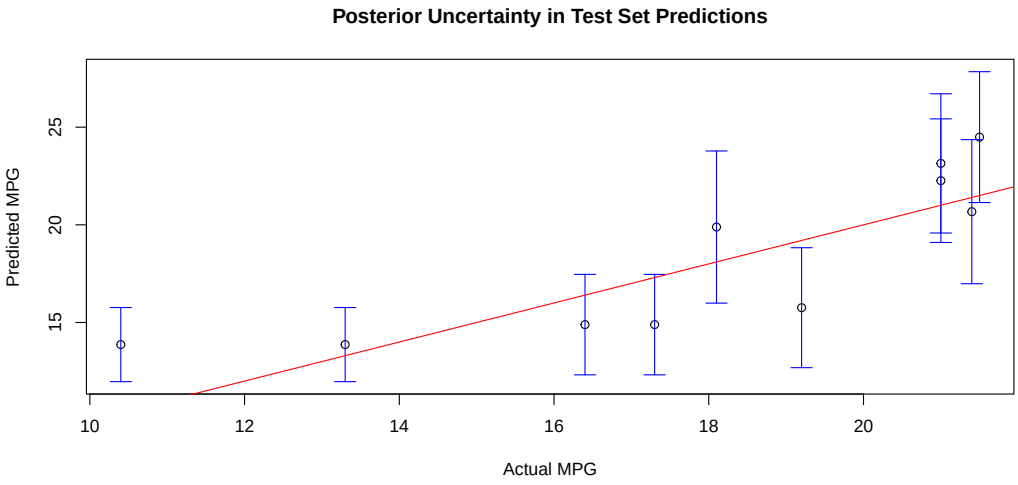
# BART modelling in practice

```r
bart_model <- wbart(x.train, y.train, x.test)

posterior_predictions <- bart_model$yhat.test

# Take what you want from the posterior distribution
posterior_mean <- colMeans(posterior_predictions)
posterior_sd <- apply(posterior_predictions, 2, sd)
posterior_low <- posterior_mean - 1.96 * posterior_sd
posterior_high <- posterior_mean + 1.96 * posterior_sd
```

# BART probabilistic output



# Bayesian Gaussian Processes

- Another common Bayesian machine learning model is Bayesian Gaussian Processes (BGPs)
- These work by fitting correlation values to the response variables. If an observation is similar in its feature space then it should be similar (i.e. highly correlated) in its response variable
- The BGP estimates a functional shape for the decay behaviour of this correlation
- Just like in other Bayesian models the parameters that control the shape of the decay are estimate with samples

# Fitting BGPs

There are many packages for fitting BGPs. One good example is the `BayesGPfit` package:

```
gp_model <- GP.Bayes.fit(matrix(y_train, ncol = 1), x_train,
                         poly_degree = 5L,
                         a = 0.001,
                         b = 2)
```

The values of `a` and `b` control the way the correlation decays (these can also be learnt or tuned). The `poly_degree` sets an underlying mean structure over which the correlation is estimated (like fitting a regression line and modelling the leftover residuals)

# Bayesian adaptive smoothing splines

- A final alternative model for fitting Bayesian machine learning models is that of **Bayesian adaptive smoothing splines** (BASS)
- A smoothing spline is a way of fitting a curve through data similar to a non-linear regression model
- The `BASS` R package is highly optimised for estimating the parameters of these smoothing splines and working on large data sets
- It doesn't (yet) work with categorical responses (e.g. classification models)

# Code for fitting all these models

- See `mtcars_bart.R` for the BART version
- `mtcars_bayesgpfit.R` for the BGP version
- `mtcars_bass.R` for the BASS version
- or compare with `mtcars_keras_nn.R` for a standard neural network version

# Extensions of these models

- The area of Bayesian machine learning is a very active research area
- Important problems require predictions with (probabilistic) uncertainty!
- Many extensions of these models (particularly BART) have been proposed over the last few years (e.g. combining BART with BASS, GPs, etc)
- Compuational challenges are the main hurdle to widespread adoption of Bayesian ML techniques

# Summary

- Bayesian machine learning is a really useful tool for smaller to medium data sets that would otherwise be very quick to fit using e.g. `keras`
- By producing probability distributions of weights (parameters) the amount of output increases by an order of magnitude
- Lots of exciting methods here