Introduction to Statistical Machine Learning

Dr Rafael de Andrade Moral
Associate Professor of Statistics, Maynooth University

rafael.deandrademoral@mu.ie
https://rafamoral.github.io

Outline

- Bayesian additive regression trees (BART)
- Boruta
- Neural networks
- Generalized additive models for location, scale and shape (GAMLSS)

BART

BART

- Bayesian Additive Regression Trees (BART; Chipman et al., 2010) is a Bayesian model based on a sum of trees
- Very good prediction properties
- Deals with interactions and non-linear relationships
- Based on a probability distribution so ‘easily’ extendable
- A Bayesian model, so have probabilistic uncertainty intervals for any required quantities
- A focus of research, with many extended versions already proposed in the literature

BART

$$y = \sum_{j=1}^{M} g(X, T_j, \Theta_j, \mu_j) + \varepsilon; \quad \varepsilon \sim \mathsf{N}(0, \sigma^2)$$

- y is the response and X are the covariates
- T is the tree structure (parameter)
- Θ is a set of split variables and values (parameters)
- μ is the set of terminal node values (parameters)
- The algorithm works by guessing initial values of all these (usually a stump), then proposing and accepting/rejecting new trees
- The big problem that occurs here is that the number of parameters changes when the tree structure changes
- The main reason that BART works is that we can collapse over the tree structure when we use a normally distributed prior on the μ terminal node parameters

BART

- Standard BART MCMC uses a back-fitting approach where each tree is fixed and the others updated in turn
- Requires us to propose and accept/reject a new tree at each iteration for M trees
- We put extra prior distributions on the size and shape of the trees to keep them small, and to keep the μ values from dominating the predictions
- Default BART has four moves to generate new trees, all of which are reversible:
  - *Grow*: picks a random terminal node and splits it in two by choosing a random split variable and split value
  - *Prune*: picks a pair of adjacent terminal nodes and merges them
  - *Swap*: picks two random internal nodes and swaps their split variables and values
  - *Change*: picks a random internal node and changes the split variable and value

BART

- Tree prior can be used to favour shallow/deep trees:

$$\text{P(node } d \text{ is non-terminal)} = \frac{\alpha}{(1+d)^{\beta}}$$

- Default settings of $\alpha = 2$ and $\beta = 0.95$ give
  - depth of 1: 0.05
  - depth of 2: 0.55
  - depth of 3: 0.28
  - depth of 4: 0.09
  - depth of 5: 0.03
- . . . but trees will grow deeper if the data demands it!

BART

- Extensions to BART:
  - BART-BMA (Hernandez et al., 2018, Statistics and Computing)
  - Soft-BART (Linero and Yang, 2018, JRSSB)
  - MOTR-BART (Prado et al., 2021, Statistics and Computing)
  - HEBART (Wundervald et al., 2022, ArXiv)
  - AMBARTI (Prado, Sarti et al., 2023, AOAS)
  - GP-BART (Maia et al., 2024, CSDA)
  - CSP-BART (Prado et al., *under review*)
  - ZaNI-Multinomial BART (Menezes et al., *in progress*)
  - ...

Boruta

Boruta

- Boruta is an algorithm used for feature selection in machine learning, helping to identify the most important variables in a dataset.
- It is a *wrapper method*, which means it relies on a machine learning model to evaluate the importance of features.
- Boruta specifically uses the random forest algorithm for assessing feature importance.
- It creates *shadow features* by shuffling the original features, maintaining their distribution but breaking any relationship with the target variable.
- The importance of original features is compared to the maximum importance of shadow features.
- This process is repeated multiple times to ensure robust selection. Each feature is given a score in each iteration.

Boruta

- Decision Making:
    - Important/Confirmed: A feature is considered important if its importance is significantly higher than the shadow features across many iterations.
    - Unimportant/Rejected: A feature is deemed unimportant if its importance is consistently lower.
    - Tentative: Features that do not clearly stand out as important or unimportant are marked as tentative.
- Computationally Intensive: While robust, Boruta can be computationally intensive due to the multiple iterations and comparisons.

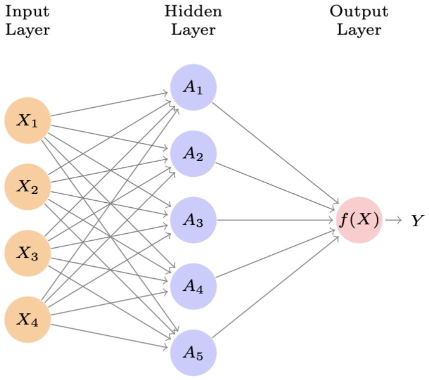
Neural Networks

Neural Networks

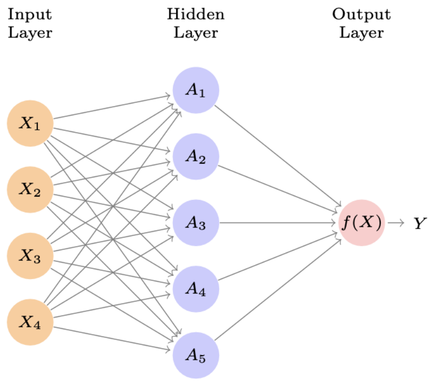
- Neural networks are computational models inspired by the human brain's architecture
- They consist of layers of interconnected nodes (neurons), where each connection has a weight that adjusts during training
- The network learns by adjusting these weights based on the error of its predictions compared to the actual outcomes
- The training involves iterative algorithms (e.g. backpropagation, stochastic gradient descent)
- Neural networks are widely used in fields such as computer vision, natural language processing, and predictive analytics
- Current research focusses on improving their interpretability, robustness, and efficiency, as well as exploring novel architectures like convolutional neural networks (CNNs) and recurrent neural networks (RNNs) for specialized applications

Neural Networks

- A simple neural network:



Neural Networks



$$Y = f(X) = \beta_0 + \beta_1 A_1 + \cdots + \beta_5 A_5$$

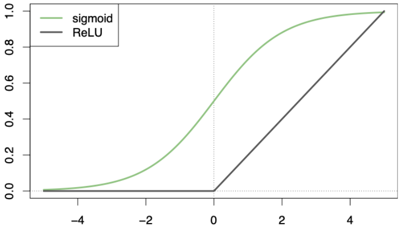
Neural Networks

$$\begin{aligned} Y = f(X) &= \beta_0 + \beta_1 A_1 + \cdots + \beta_5 A_5 \\ &= \beta_0 + \sum_{k=1}^{K} \beta_k A_k \\ &= \beta_0 + \sum_{k=1}^{K} \beta_k h_k(X) \\ &= \beta_0 + \sum_{k=1}^{K} \beta_k g\left(w_{k0} + \sum_{j=1}^{p} w_{kj} X_j \right) \end{aligned}$$

- All parameters $\beta_0, \ldots, \beta_K, w_{10}, \ldots, w_{Kp}$ are estimated from the data

Neural Networks

- $g(\cdot)$ are called *activation functions*
- Different options can be used, such as the sigmoid $g(x) = \frac{1}{1+e^{-x}}$ and ReLU (Rectified Linear Unit) $g(x) = x, x \geq 0$ and $g(x) = 0, x < 0$.



Neural Networks

- The nonlinearity in g is essential to capture complex nonlinearities and interaction effects between the predictors
- Example: take $X = (X_1, X_2)$ and $g(x) = x^2$, with

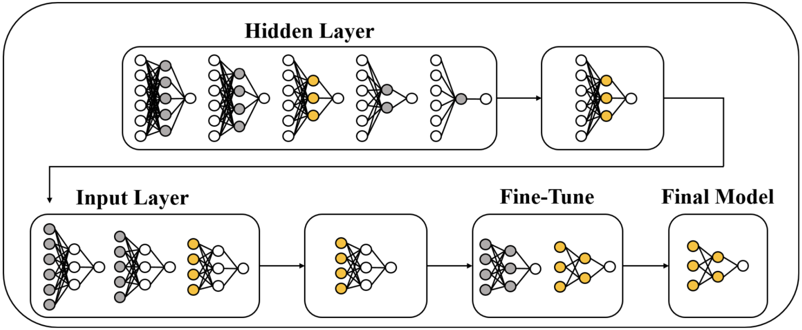
$$\begin{array}{rrr} \beta_0 = 0, & \beta_1 = \frac{1}{4}, & \beta_2 = -\frac{1}{4}, \\ w_{10} = 0, & w_{11} = 1, & w_{12} = 1, \\ w_{20} = 0, & w_{21} = 1, & w_{22} = -1. \end{array}$$

Neural Networks

- Statistical perspective: thinking of the neural network as a statistical model

$$y_i = \mathsf{NN}(x_i) + \varepsilon_i$$

- Allows for computation of likelihood and information criteria
- R package `selectnn` (McInerney and Burke, 2024)



GAMLSS

GAMLSS

- Generalized Additive Models for Location, Scale, and Shape (GAMLSS; Rigby and Stasinopoulos, 2005) are a very flexible semi-parametric modelling framework

$$Y \sim f(\mu, \sigma, \nu, \tau)$$

- Includes many distributions with up to 4 parameters
- Allows for distributional regression, i.e. all parameters can be modelled with covariates
- Spline and loess smoothing, as well as random effects allowed
- Unified framework for model diagnostics

GAMLSS

- The `gamlss` package is the main implementation
- Extra features available through companion packages
- The package `gamlss.add` includes extra features, such as the inclusion of regression trees and neural networks in the linear predictor for any parameter of interest
- e.g. $\mu = \text{NN}(x)$ or $\mu = \text{CART}(x)$