# Parallel ASHA for Hyperparameter Tuning

Group 9 - The Rain in Spain

Speaker 1: Matthew Frost
Speaker 2: Molly Farrant
Speaker 3: Abhinav Behal

# Project Overview

## Parallel Machine Learning

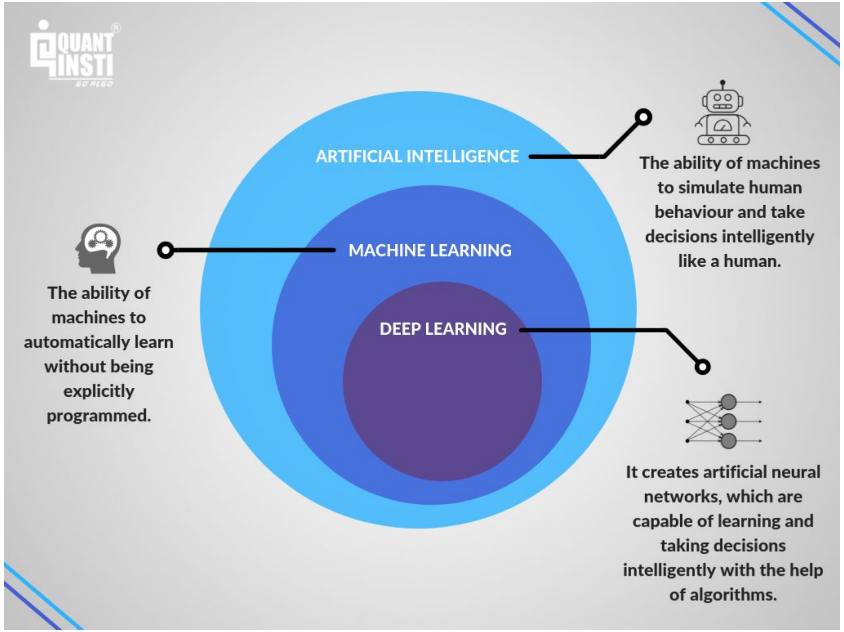
- Resurgence of Machine Learning research since interest died in the 70's
- Performance is a big issue
- Common applications of parallelisation
  - Within an algorithm
  - Training process
  - Cross evaluation
- A difficult problem is the generation of the best model quickly - ***hyperparameter tuning***

# Artificial Intelligence

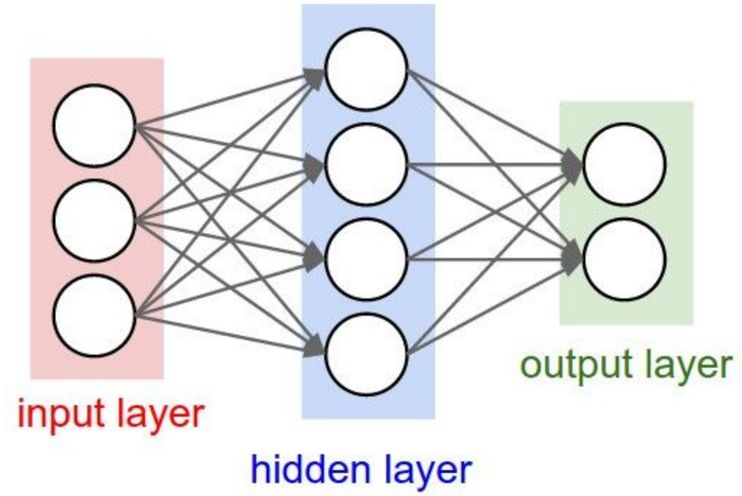
- **Artificial intelligence** is computer simulation of human intelligence
  - Rules, knowledge based systems, expert systems

- **Machine learning** is the ability for machines to learn without being explicitly programmed

- **Deep learning** is ML using neural networks



# Deep Learning

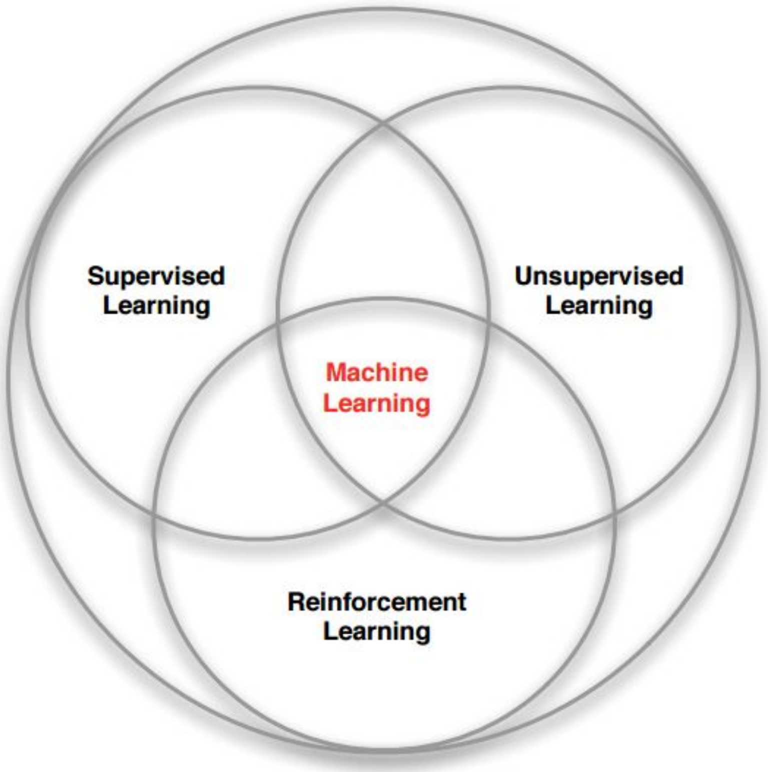



- An incredibly powerful machine learning technique
- Involves the application of Artificial Neural Networks
- Can approximate any nonlinear function
- Often for image recognition, speech recognition

# Machine Learning

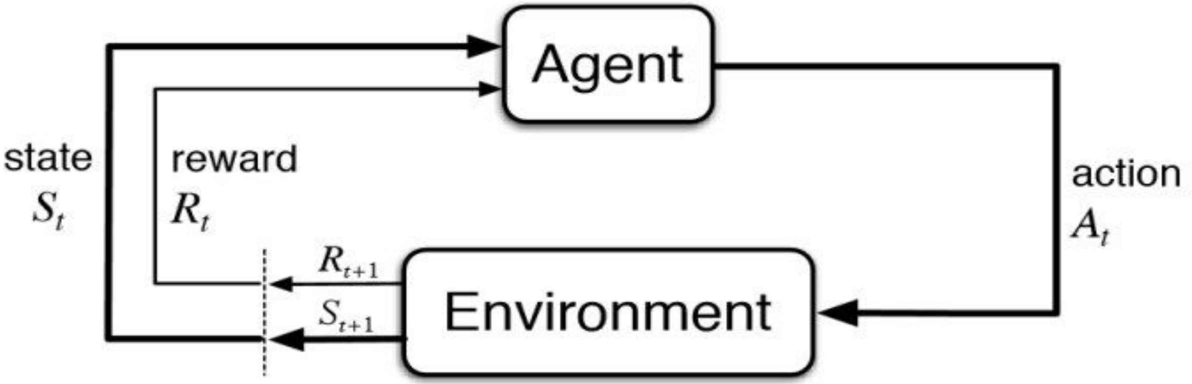
The ability of machines to learn without being specifically programmed.

- Unsupervised
- Reinforcement
- **Supervised**
  - Regression
  - **Classification**



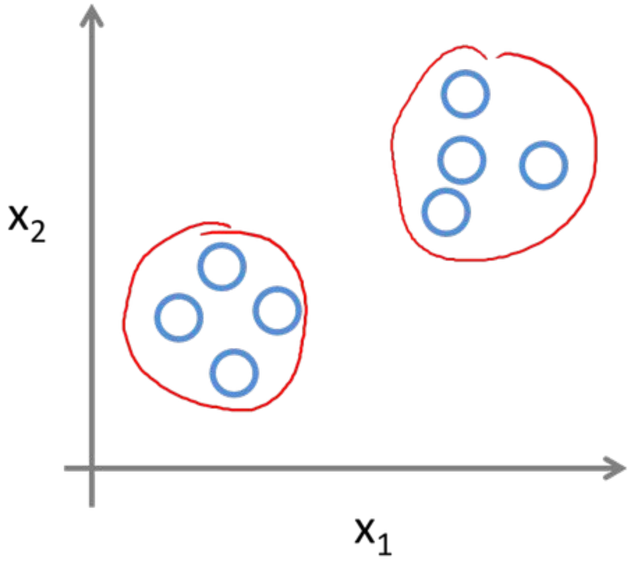
# ML - Reinforcement Learning

- The learner receives rewards and punishments for its actions
- Algorithm must determine the ideal behaviour within the specified context to maximise its rewards / minimise its punishments



# ML - Unsupervised Learning

- No definite output
- Training data is **unlabelled**
- Aims to find structures or patterns in the data



# ML - Supervised Learning

- Dataset with training examples
- Each example has an associated **label** which identifies it

- Regression
  - Used to determine the mathematical relationship between two or more variables and the level of dependency between them

- **Classification**
  - Used to classify data into predefined categories
  - **Binary classification** – only two categories

# Ensembles

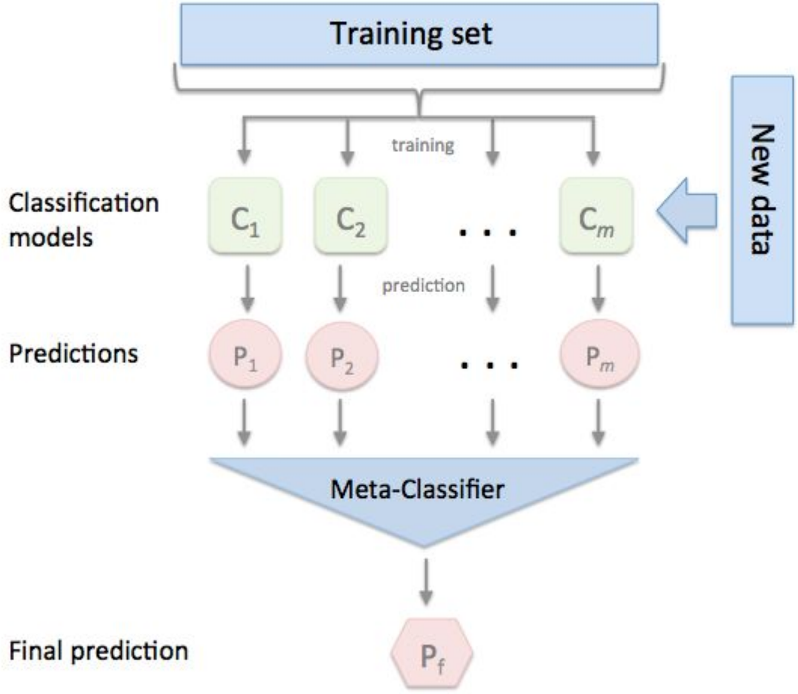
- Combine several base models in order to produce one aggregate predictive model

- **Weak learner**: individual rules that are not powerful enough to classify data
- **Strong learner**: combining the predictions of weak learners using average, weighted average or majority voting system

- Can be used for regression or **classification** algorithms

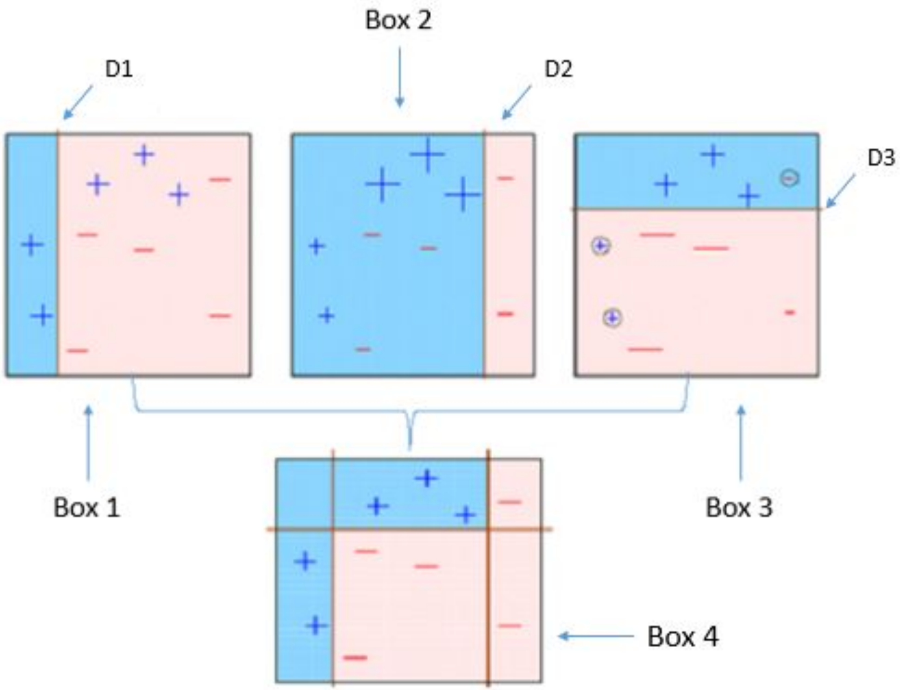
# Bagging

- **Parallel** approach
- Each model is exposed to a different **subset** of the training data
- Individual models are built separately
- Models combined via **averaging** or **majority voting**



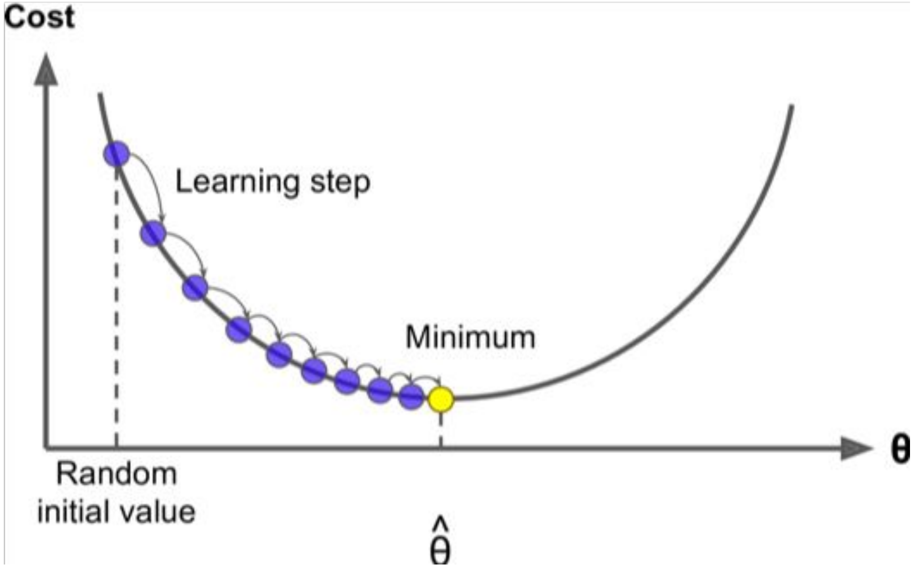
# Boosting

- **Sequential** approach
- Each model is exposed to the **entire** training dataset
- Each new model is influenced by the performance of those built previously



# Gradient Boosting

- Trains models sequentially
- Each new model gradually minimises the loss function
- Loss function is a measure of how well a given model fits the relevant training data



- An implementation of gradient boosting

- Designed for speed and performance
    - **Parallelization**
    - **Distributed Computing**
    - **Cache Optimization**

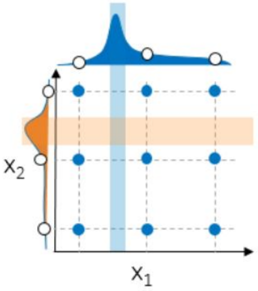
- Currently achieving the best performance on a range of difficult machine learning tasks

# Hyperparameter Tuning

Hyperparameter - ***A parameter whose value is set before the learning process starts***

- Most machine learning algorithms typically require a dozen or more hyperparameters, e.g. for XGBoost:
  - Maximum tree depth
  - Sample type
  - Rate drop
  - ...
- These parameters influence the overall performance of algorithm
- Due to the shear number of possible configurations, you need a way to pick the **best**

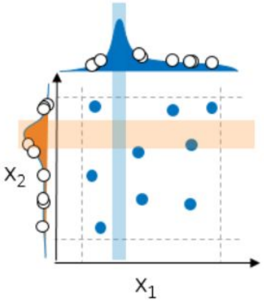
# Grid Search



(a) Standard Grid Search

- Exhaustive search of a subset of the hyperparameter space
- Guided by a performance metric
- Can be parallelised trivially
- Implemented in all popular machine learning packages
- Number of possible configurations grows immensely with more hyperparameters, hence the search must be quite coarse in order to have an acceptable runtime

# Random Search



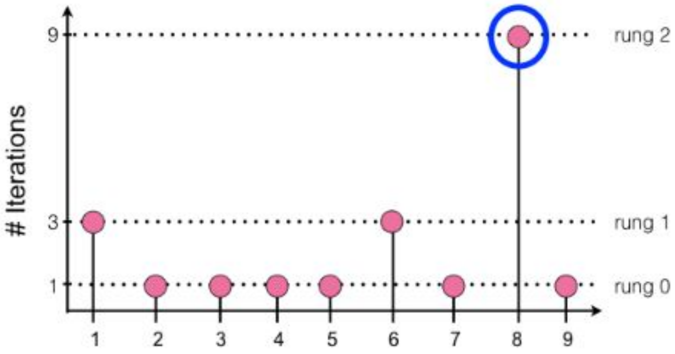

(b) Random Search

- Simpler than Grid Search, can also be parallelised trivally
- Starts off with a random hyperparameter configuration
- Proceeds by choosing a new random configuration around the current one
- Picks the best of the two (using similar metrics to Grid Search) and repeats
- Terminates upon reaching a pre-defined number of iterations or performance metric value
- Also implemented in all popular machine learning packages

# Evolutionary Approaches

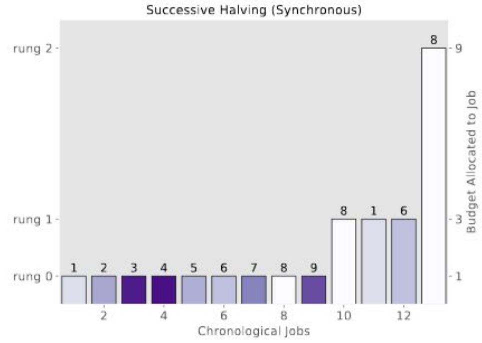
- Utilise evolutionary algorithms to search the hyperparameter space
- Basic approach:
  - Start off with a population of randomly selected configurations
  - Evaluate each configuration using a fitness function
  - Rank each configuration according to their fitness values
  - Use crossover and mutation functions to generate a new population
  - Repeat until a certain performance metric is reached, or when no improvements are being made
- Some open source libraries exist for evolutionary hyperparameter tuning, and have parallelisation built-in

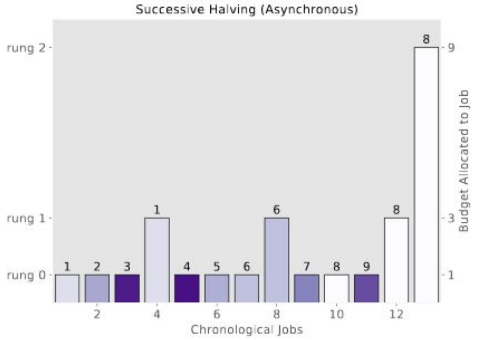
# Successive Halving Algorithm (SHA)




- More complex than the previous techniques, but still relatively simple:
  - **N** different parameter configurations randomly generated and allocated a number of epochs (time to run/iterations)
  - Select the top **1/K %** of the configurations, and give them each **K** times more epochs (promoted to next “rung”)
  - Repeat until only one configuration is left
- The synchronous nature of SHA makes any parallelisation attempts ineffective and in some cases can increase the total execution time

# Asynchronous SHA (ASHA)




- Leverages asynchronous subroutines to effectively parallelise SHA
- Promotes configurations to the next rung whenever possible
  - Unlike SHA which waits for all configurations in the current rung to finish
- The best performing configurations are promoted to the next rung once the number of configurations in that rung is a multiple of K
- Utilises workers that evaluate available configurations in parallel
- Configurations are assigned to workers as they become available

# Our Approach

- Distributed serverless parallelization
- ASHA is used on central machine, evaluation is in parallel
- Each serverless instance is given a single model to evaluate
- ASHA maximises parallelization - no blocking to wait entire bottom rung to finish

# Implementation

- XGBoost
- AWS Lambda for serverless training and evaluating individual models
- Python, Pandas, numpy, etc.

# Dataset - The Rain in Australia

- Binary classification
- Will it rain tomorrow?
- 24 columns x 142k rows
- Weather data from throughout current day (pressure, temperature)
- Predict next day

# Evaluation of Performance

- Metrics
  - Error rate on test partition of data set
  - Confusion Matrix
  - Accuracy, Recall, Precision
  - Time taken to produce final model

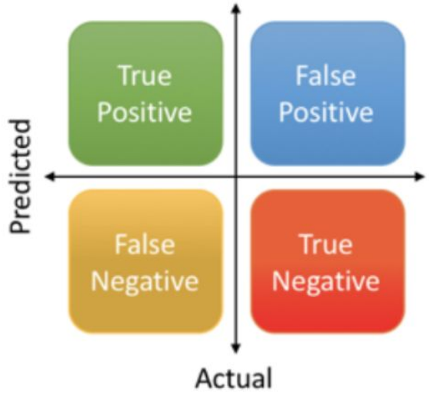
- Benchmark against
  - Grid Search
  - Random Search
  - Serial SHA
  - Default XGBoost tuning

# Evaluation of Performance

$$\textbf{Precision} = \frac{\text{True Positive}}{\text{Actual Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive + False Positive}}$$

$$\textbf{Recall} = \frac{\text{True Positive}}{\text{Predicted Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive + False Negative}}$$

$$\textbf{Accuracy} = \frac{\text{True Positive + True Negative}}{\text{Total}}$$



Kahoot!

# Q&A