Ollscoil na Gaillimhe
University of Galway

Deep Learning

Topic 1: Overview of Machine Learning Concepts

Prof. Michael Madden
Chair of Computer Science
Head of Machine Learning Group
University of Galway

Learning Objectives

After successfully completing this topic, you will be able to…

- Define Artificial Intelligence and Machine Learning
- Identify some key ML tasks and techniques that can be applied to tackle them
- Explain the basic ideas of some classic ML algorithms
- Explain the motivation for learning with ensembles
- Describe some ensemble learning approaches such as bagging, boosting and stacking
- Explain the principles of Random Forests and Gradient Boosting

What is AI?

- Computer systems that can **perform tasks that require intelligence** when performed by people
- OR: Computer systems that **act rationally**

- Artificial **General** Intelligence is a goal, but:
  - Usually focus on limited domains
  - Build specialised machines that may out-perform humans at specific tasks

- Sub-disciplines of AI:
  - **Machine Learning**, natural language processing, perception, planning, knowledge representation, reasoning under uncertainty, …
  - **Deep Learning** is a sub-discipline of Machine Learning

What is Machine Learning?

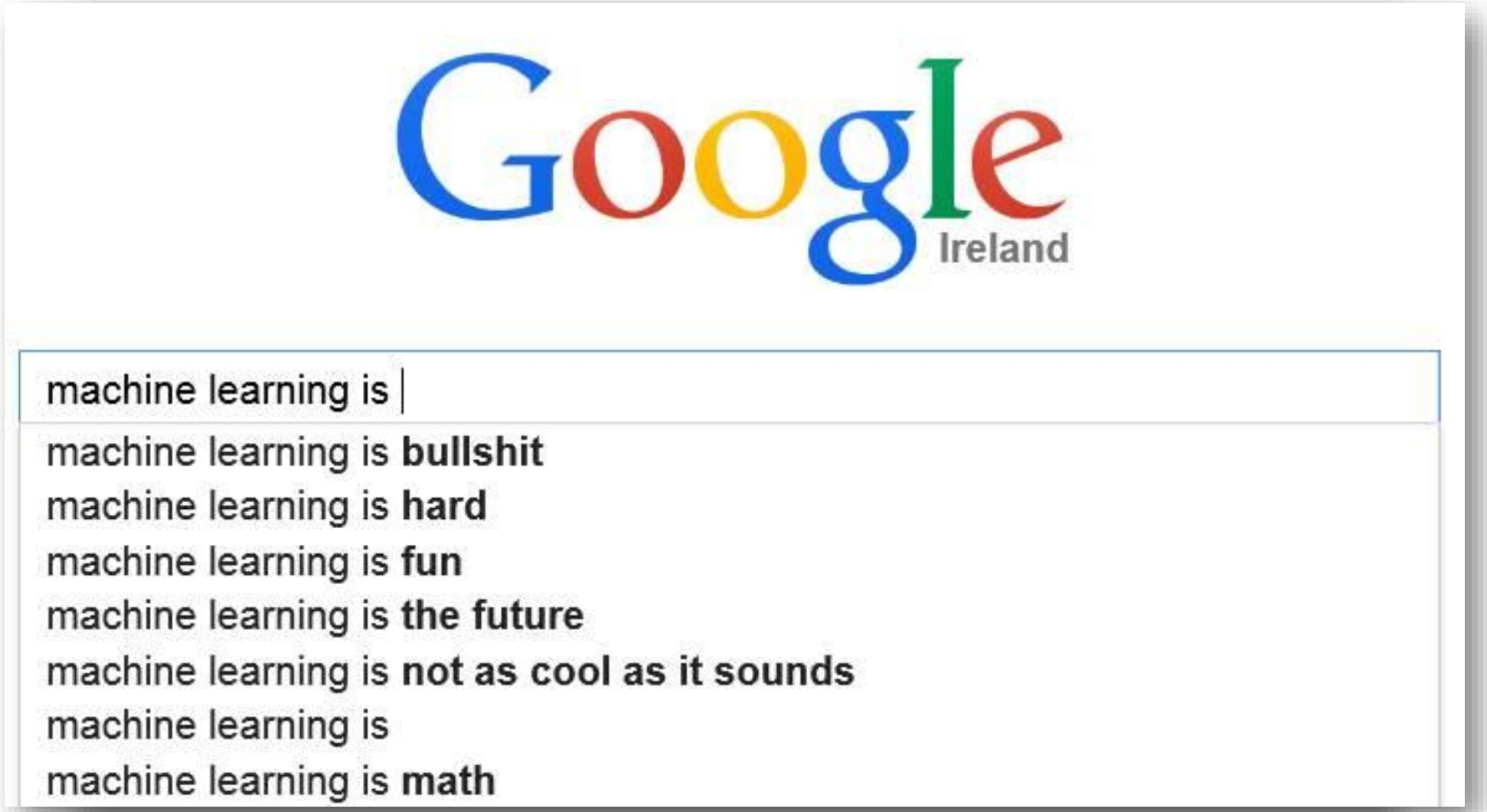
- Samuel, 1959:
  - "Field of study that gives computers the ability to learn without being explicitly programmed"

- Mitchell, 1997:
  - Improvement with experience at some task
  - A well-defined ML problem:
    - Improve over task *T* (e.g. play chess)
    - wrt performance measure *P* (e.g. % wins)
    - based on experience *E* (e.g. games against opponent)

Arthur Samuel, 1901-1990

Image source:
http://www.computer.org/portal/web/awards/cp-samuel

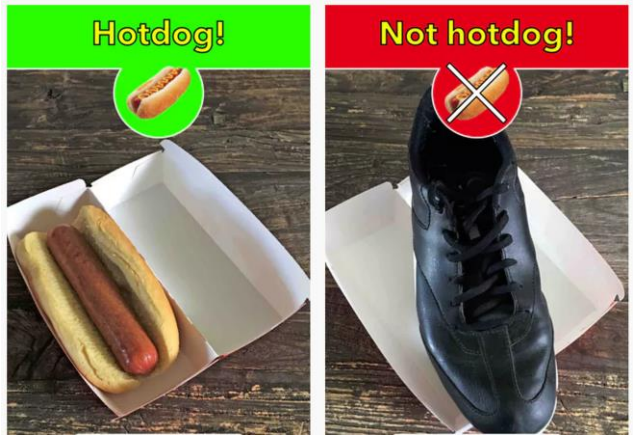
Or We Could Ask Google …

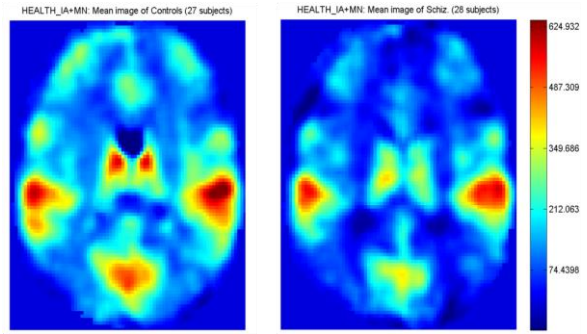


Some Key Ideas in Machine Learning

Major Types of Classic ML Task [1]

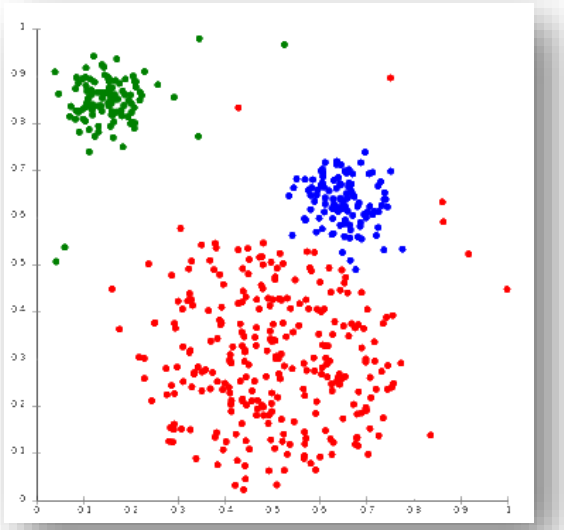
1. **Classification**





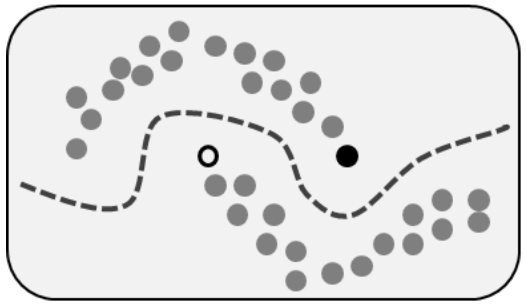
2. **Regression**

3. **Clustering**



Major Types of Classic ML Task [2]

4. Co-Training



5. Relationship Discovery

6. Reinforcement Learning

Techniques for these Tasks

- Classification
  Decision trees, SVMs
- Regression
  Linear Regression, k-NN; Neural Nets
  (good for Classification too)
- Clustering
  k-Means, EM-Clustering; Neural Nets
- Relationship Discovery
  Association Rules; Bayesian nets
- Learning From Part-Labelled Data
  Co-Training; Transductive Learning
  [Combines ideas from clustering & classification]
- Reinforcement Learning
  Q-Learning, SARSA

Supervised

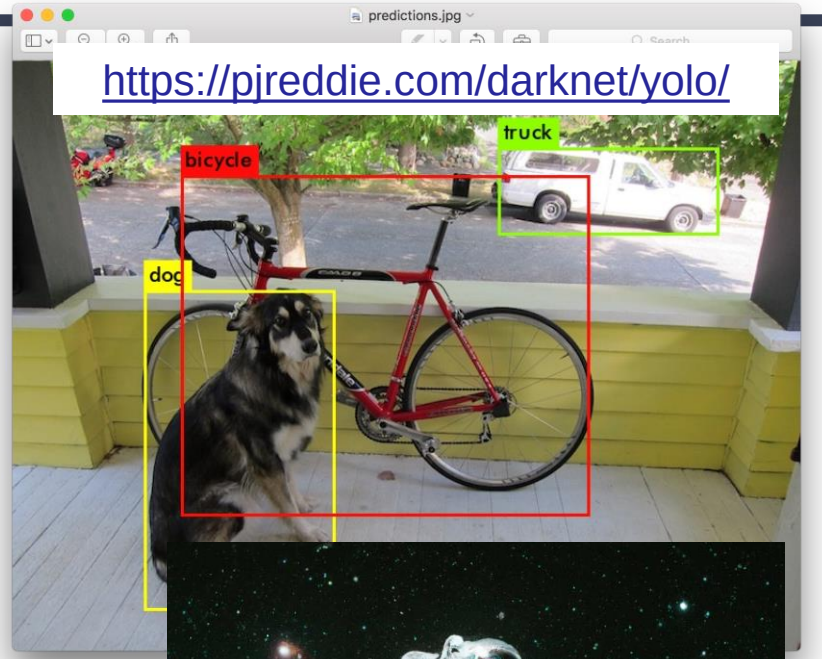
Unsupervised

Semi-supervised

Reward-Based

More Recent ML Tasks & Approaches Using DL

- Computer Vision and Robotic Control
  - Object detection: objects with bounding boxes
  - Instance segmentation: pixel masks for each
  - Robot movement planning
- Mappings and Embeddings
  - Mapping text to text (e.g. translation)
  - Mapping images to text (e.g. caption generation)
  - Mapping text to images (e.g. DALL-E 2)
  - Mapping images to images (e.g. semantic segmentation)
- Reducing Need for Labelled Data
  - Self-supervised learning



What Do All ML Techniques Have in Common?

- In all cases, machine searches for a hypothesis that best describes the data presented to it
- Choices to be made:
  - How is hypothesis expressed?
    mathematical equation, logic rules, graph format, table, weights …
  - How is search carried out?
    systematic (breadth-first or depth-first), heuristic (most promising first), …
  - How do we measure quality of hypothesis?
  - What is appropriate format for data?
  - How much data is required?

Reminder –
Definition of ML Problem

- Improve over task *T*
- wrt performance measure *P*
- based on experience *E*

What Else to We Need to Know About?

- To apply ML:
  - How to formulate a problem
  - How to prepare the data
  - How to select an appropriate algorithm
  - How to interpret the results
- To evaluate results and compare methods:
  - Separation between training, testing & validation
  - Performance measures:
    simple metrics, statistical tests, graphical methods
- To improve performance:
  - Use ensemble methods
  - Apply feature engineering
  - Understand theoretical limits on performance

Review of some Classic Algorithms for Supervised Learning

Supervised Learning: Motivating Examples

1. Estimate sale price of future houses, given past data of house sizes, locations and their prices
2. Before unlocking tablet, determine whether you or somebody else is looking at the webcam
3. Decide whether a chemical spectrum of a mixture has evidence of containing cocaine, based on other spectra with & without cocaine
4. Predict concentration of cocaine in mixture

Key feature:

given "right answers" / "ground truth" as start point.

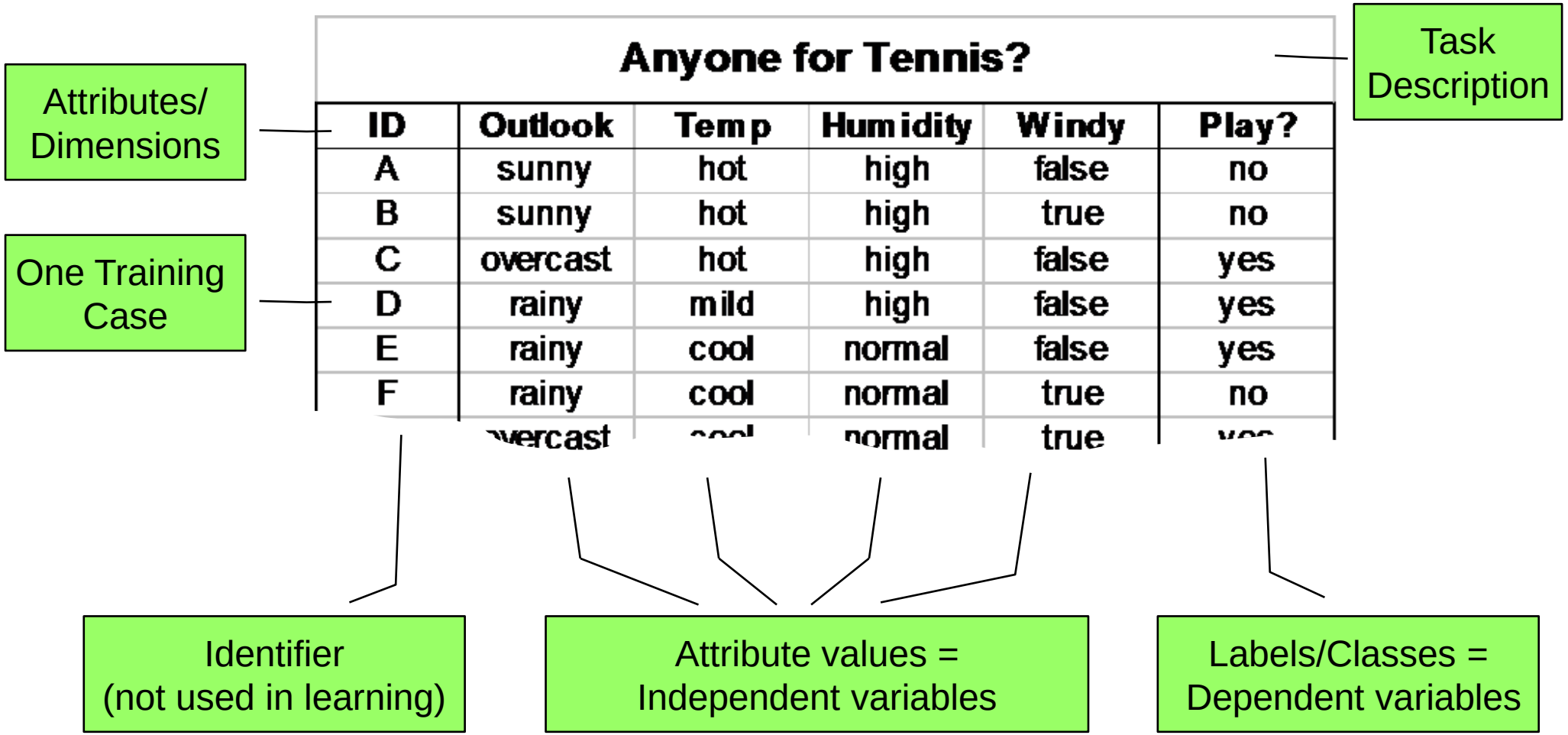
Which tasks are classification and which are regression?

Supervised Learning: Task Definition

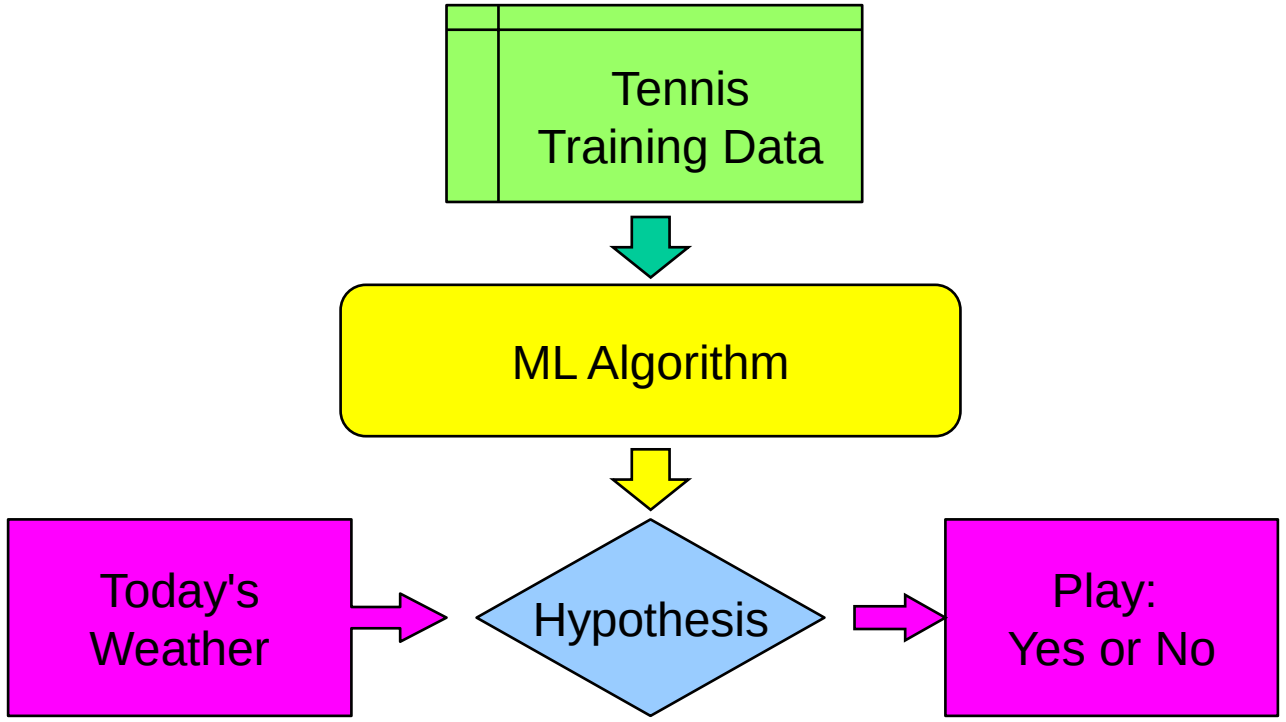
- Given examples, return function *h* (*hypothesis*) that approximates some 'true' function *f* that (hypothetically) generated the labels for the examples
  - Have set of examples, the **training data**:
    each has a **label** and a set of **attributes** that have known **values**
  - Consider *labels* (classes) to be *outputs* of some function *f*; the observed *attributes* are its *inputs*
  - Denote the attribute value inputs ***x,***
    labels are their corresponding outputs ***f(x)***
  - Function ***f*** is *unknown*; want to discover an approximation of it, ***h***
  - Can use ***h*** to **predict** labels of new data: generalisation

Also known as Pure Inductive Learning

Training Data Example

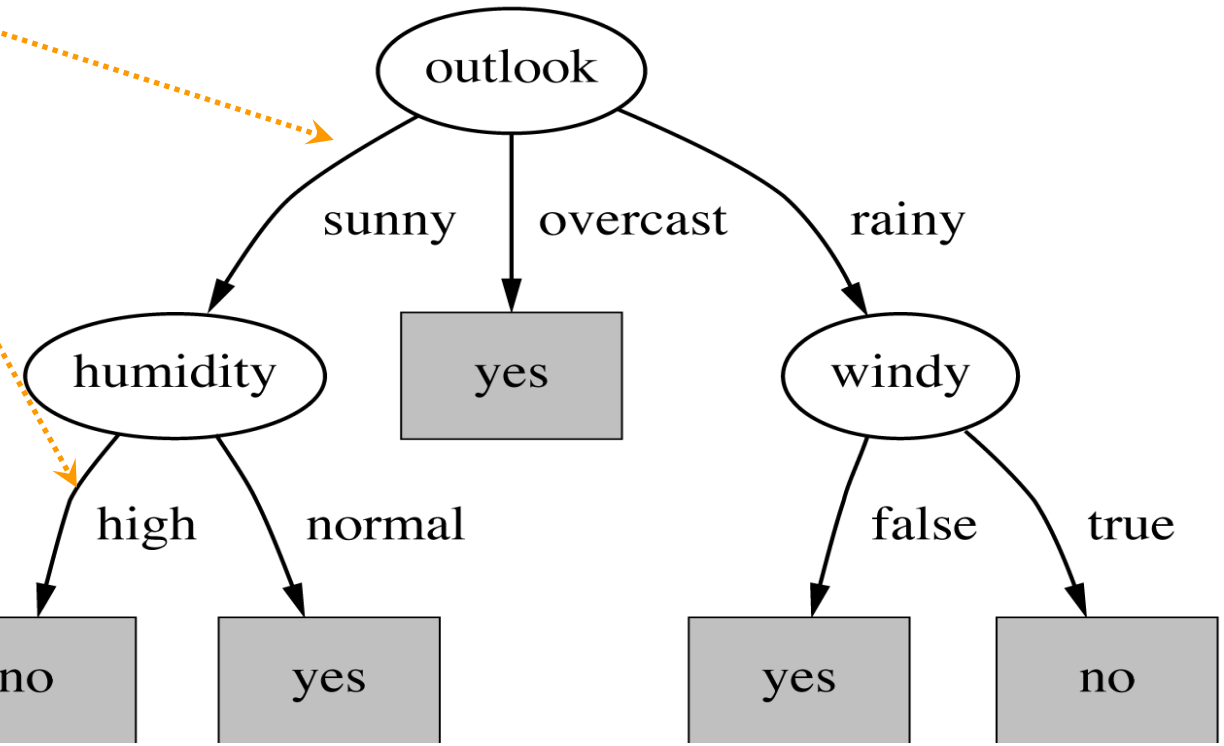


Supervised Learning



Example of a Decision Tree

Outlook	sunny
Temp	hot
Humidity	high
Windy	false
Play?	no



Inductive Learning of a Decision Tree

Step 1
- For all attributes that have not yet been used in the tree, calculate their **entropy** and **information gain** values for the training samples

Step 2
- Select the attribute that has the highest information gain

Step 3
- Make a tree node containing that attribute

Repeat
- This node **partitions** the data:
  apply the algorithm **recursively** to each partition

k Nearest Neighbour Algorithm

- Operation is relatively easy to appreciate
- Key insight:
  - Each sample can be considered to be a point in sample space
  - If two samples are close to each other in space,
    they should be close to each other in their target values
- How it works:
  - The training data samples are simply stored
  - When you want to classify a new **query case**:

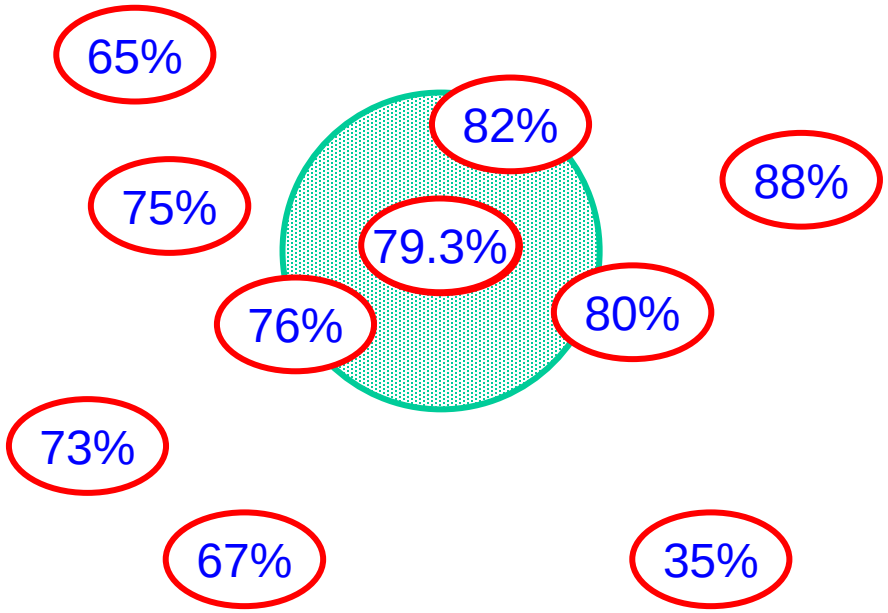
    Compare it to the stored set and retrieve the most similar one(s)

    Give the query case a label based on the similar one(s)

Overview of kNN Algorithm

- *k*-Nearest Neighbour algorithm:
  - No real training phase: just store the training cases
  - Given a query case with value to be predicted, base prediction on several (*k*) nearest neighbours
  - Compute distance from query case to all stored cases, and pick the nearest *k* neighbours (e.g. sort the distances and pick the lowest *k*)
  - Need appropriate measure of distance (or similarity): choice of distance metric will affect results
- Classification with kNN:
  - Neighbours vote on classification of query case
- Regression:
  - Average the value of the neighbours

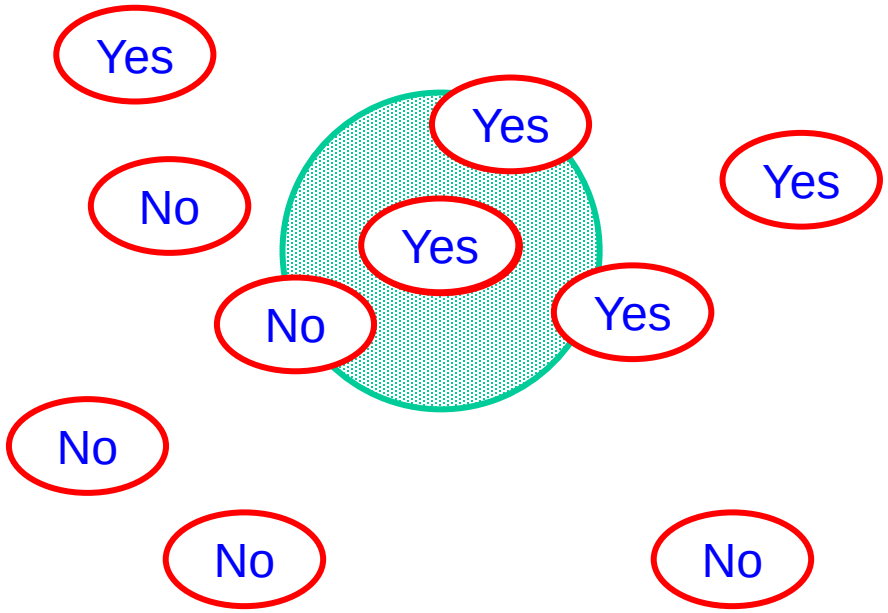
k Nearest Neighbours: Visualisation of Regression Example



This visualisation assumes Euclidean distance

Estimate: 79.3%

k Nearest Neighbours: Visualisation of Classification Example



3 nearest neighbours vote:
Decision is Yes

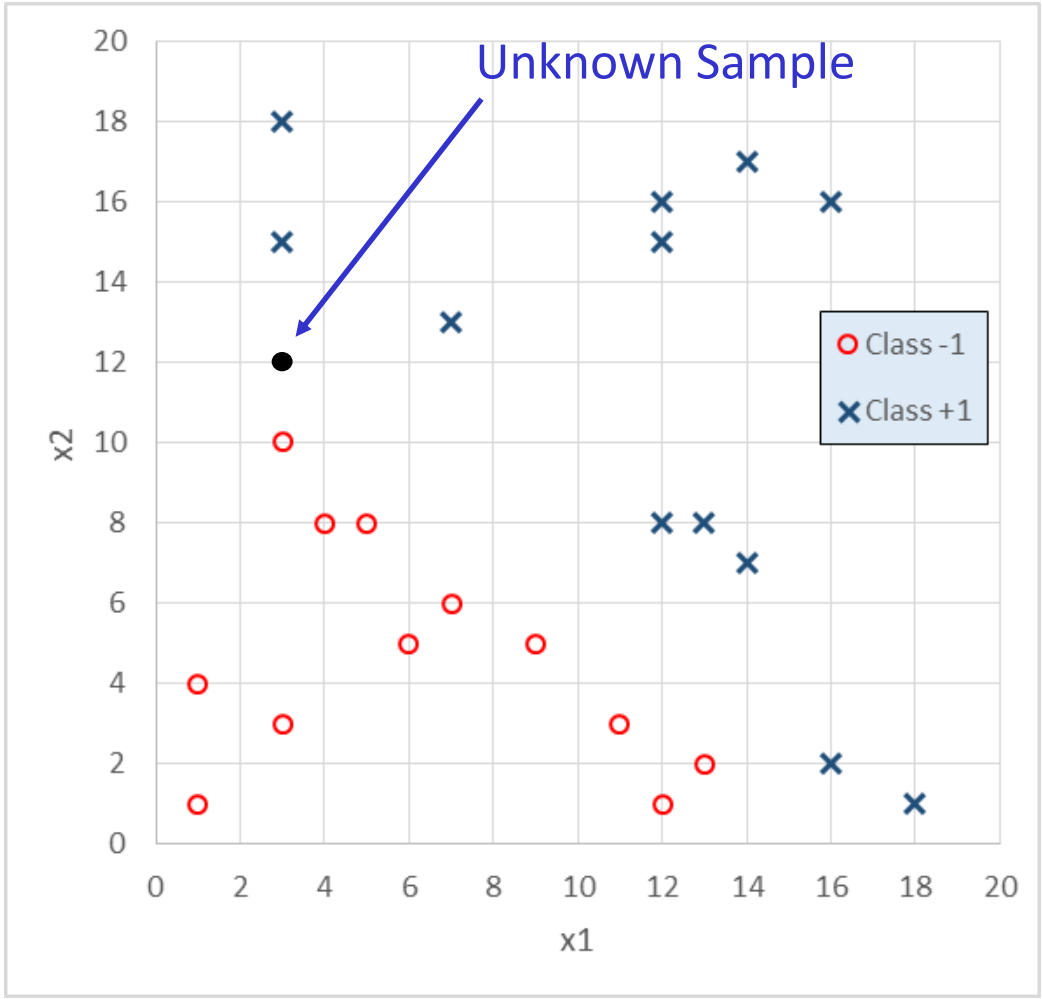
Classification with Support Vector Machine

- Is unknown sample X or O ?
  Goal: Find model that correctly classifies a new sample, $\mathbf{x}^{(i)}$
  Note that the inputs $\mathbf{x}^{(i)}$ are numeric.

- X's are assigned : +1
- O's are assigned : -1
- These will be our values for $y^{(i)}$

- Training Data ($\mathbf{x}^{(i)}$: $y^{(i)}$)
  $\mathbf{s}_1$: (11, 3: -1) **O**
  $\mathbf{s}_2$: (3, 10: -1) **O**
  $\mathbf{s}_3$: (14, 17: +1) **X**
  $\mathbf{s}_4$: (16, 16: +) **X** …

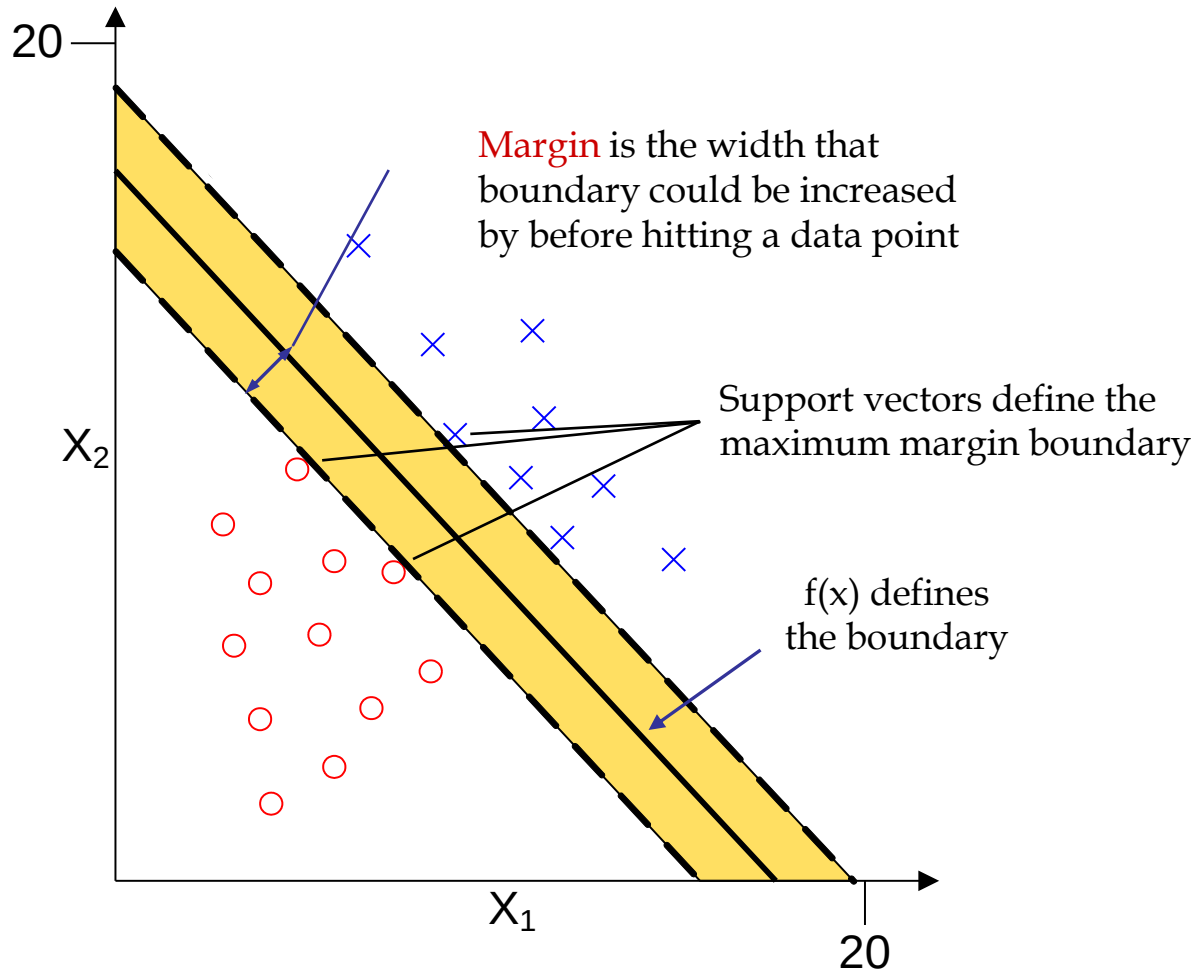
Unknown Sample s = (3, 12: ?)
  – What class does this belong to?



Next Topic: will tackle this with Logistic Regression.
For now, a brief overview of Support Vector Machine approach.

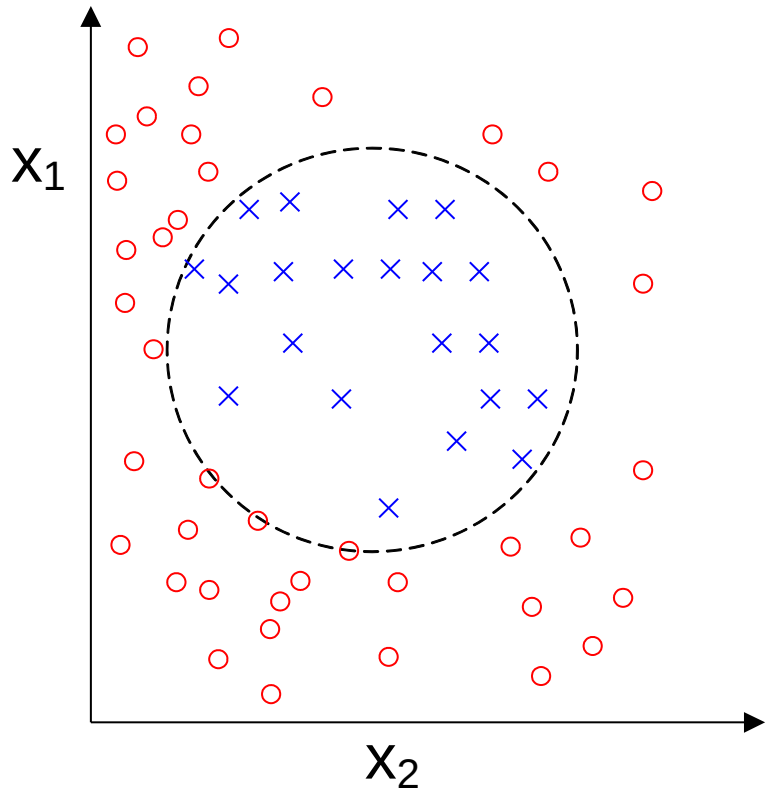
Linear Support Vector Machine (1)

- Finds maximum margin function, f(x), that linearly separates the two classes
- A wider margin implies a better generalisation
- Intuition is that we are placing line/hyperplane as “far” as possible from known data of both classes
- Called a Linear SVM
- Note: data vectors that ‘prop up’ plane are the “support vectors”

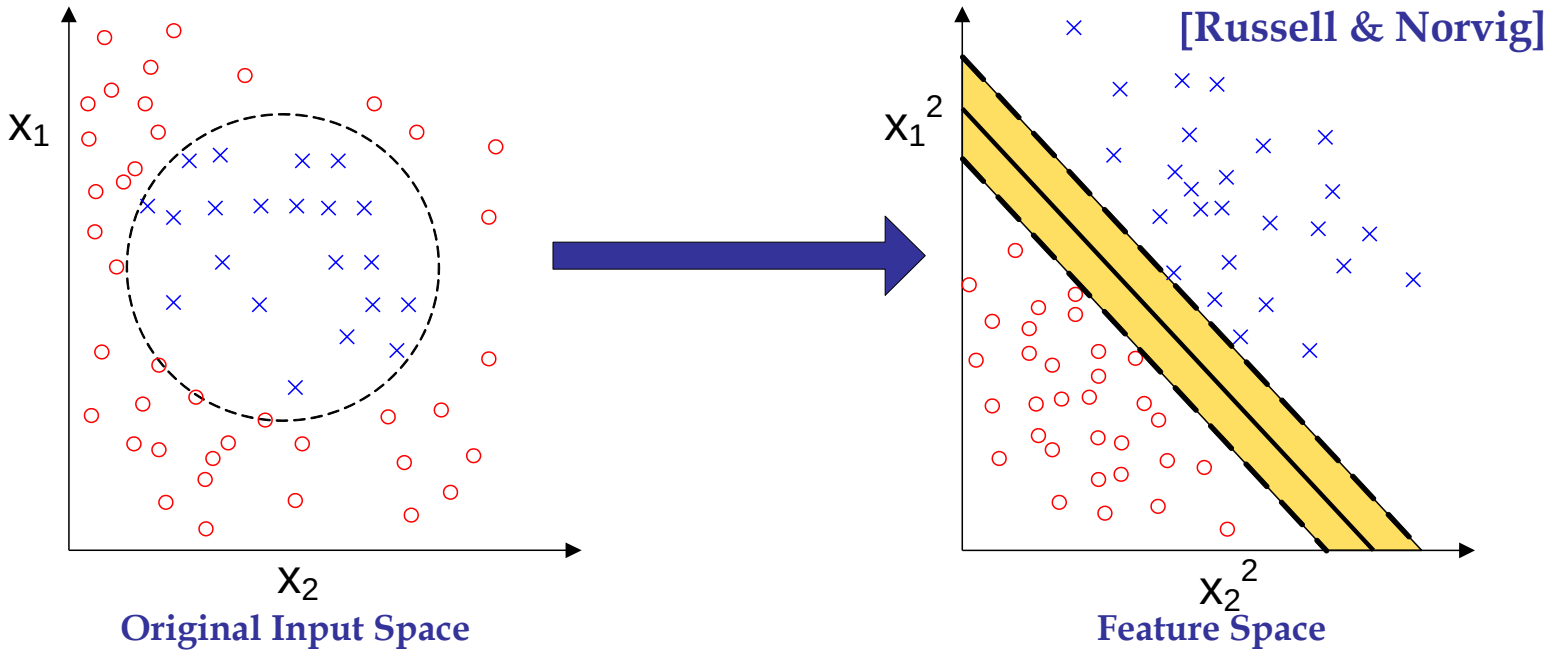


SVMs and Non-Linear Data (1)

- Problem: a linear classifier cannot be used on all data
- An obvious pattern exists in this example
- Solution: transform the data into a new feature space



SVMs and Non-Linear Data (2)



- When data is mapped using these two new features, a separating hyperplane can be found
- Mapping is performed by a Kernel Function

Non-Linear SVM

- A **Non-Linear SVM** is one with a non-linear kernel
- SVM Decision Function now looks like:

$$f(x) = \sum_{i,j=1}^{N} \alpha_i y_i k(x, x_i) + b$$

- **Widely-used Kernels:**
  - Polynomial Kernel: $K(x, x_i) = (x \cdot x_i)^d$
  - Radial Basis Function Kernel (RBF): $K(x, x_i) = \exp\left(\frac{-\|x - x_i\|^2}{2\sigma^2}\right)$
  - Sigmoid Kernel: $K(x, x_i) = \tanh(x \cdot x_i - \theta)$

Practical Strategies to Improve Performance of Typical Classification and Regression Algorithms

Overview (1)

- Ensemble Models:
  - Combine decisions from multiple classifiers/regressors to improve overall classification performance
  - Techniques include boosting, bagging and stacking that can work with different underlying algorithms
- Random Forests:
  - A key early ensemble algorithm to create a “forest” of decision trees
- XGBoost (Extreme Gradient Boosting)
  - Implementation of Gradient Boosting algorithm to construct an ensemble of decision trees
  - One of the most popular algorithms for tabular data
  - https://www.kaggle.com/code/dansbecker/xgboost

Overview (2)

- Feature Engineering - broad term encompassing:
  - **Feature extraction:**
    e.g. extracting some high-level features from an image
  - **Feature transformation:**
    e.g. principal component analysis, FFTs, etc
  - **Feature subset selection:**
    keeping only a subset of the original features, discarding the rest

Will not spend further time on feature engineering here.

Ensembles: Basic Idea

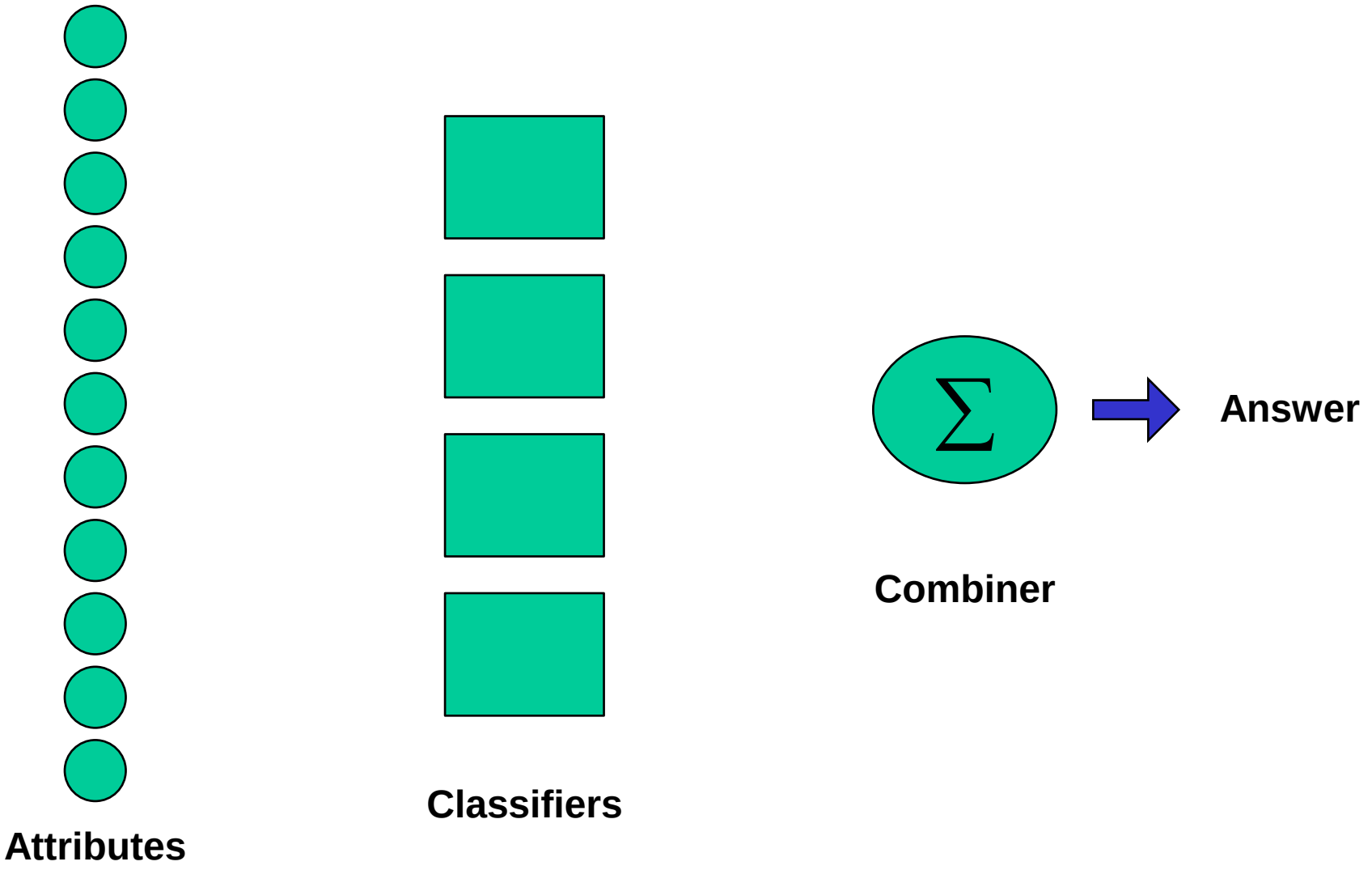
- Using your data, construct multiple classifiers
  - They form a final Ensemble (a.k.a. Committee or Jury)
  - Combine the decisions to arrive at final decision
  - As long as individual classifiers are **independent** and **better than random**, ensemble as a whole will be better than the any one of the individuals, on average (and never significantly worse)

- In ML projects, we often try multiple algorithms/ configurations and keep the single best classifier
  - May be counter-intuitive that we can do better by combining the best one with weaker ones

Combining Ensemble Members

- We can use techniques like those we used in kNN to combine neighbours' outputs
  - Except we are **not** just using the values of the nearest neighbours, we are using the values output by multiple models
- Classification:
  - Have all the classifiers vote: use the majority value
- Regression:
  - Compute an average
- Can combine these with weighting schemes

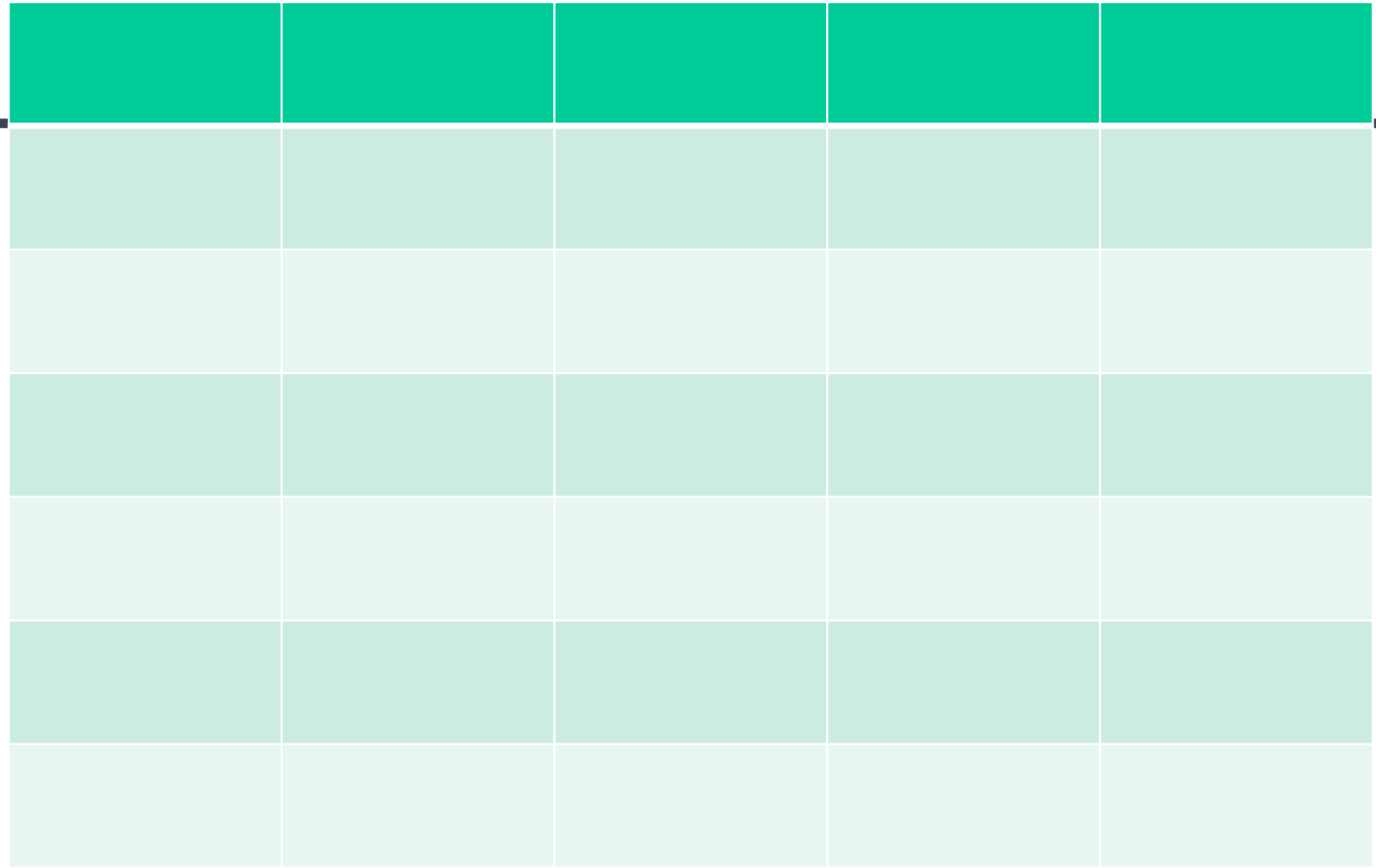
Combining Ensembles



Theoretical Basis: Condorcet Jury Theorem

- Marquis de Condorcet: "Essai sur l'application de l'analyse á la probabilité des décisions rendues á la pluralité des voix“, **1785**

- “Essay on the Application of Analysis to the Probability of Majority Decisions”
  - *Provided each voter is better than random and voters are not correlated with each other, then adding more voters increases the probability that the majority decision is correct*
  - *Could show this with a simulation …*



Ensembles & Diversity

- Easy to ensure all voters are better than random
  - **How would you do this?**

- Independence is more of a challenge
  - Will not work if our individual voters are correlated
- Strategies to create independent classifiers:
  - **Any ideas?**

Ensembles & Diversity

- Easy to ensure all voters are better than random
  - **How would you do this?**

- Independence is more of a challenge
  - Will not work if our individual voters are correlated
- Strategies to create (hopefully) independent classifiers:
  - Use different learning algorithms
  - Use different data sets:
    Entirely different or carve up the existing dataset
    by rows or by columns

Ensembles & Diversity

- Use different classifiers
  => **Stacking**

- Use entirely different datasets
  => Not sure this has a name, but sometimes done with deep neural nets that were trained separately

- Divide a single dataset into subsets by instances
  => **Bagging and Boosting**

- Divide a single dataset into subsets by attributes
  => **Done in Random Forests**

Ensembles: Bagging

- Simulates the existence of multiple data sets
- Bagging uses sampling with replacement to generate multiple datasets that are *partly independent* of each other, and applies the same algorithm to all datasets, to construct an ensemble of classifiers
- Works well if the classification algorithm is *unstable*
  - C4.5: YES; Naïve Bayes: NO
- Have discussed **bias-variance decomposition** in the ML module:

  Bias = expected error of combined classifier on new data

  Variance = expected error due to specific training set used
- Bagging works by reducing the variance through averaging/voting over multiple classifiers created with different training subsets

Ensembles: Boosting

- Also uses voting/averaging
- Weights models according to performance
- Iterative: new models are influenced by performance of previously built ones
  - Encourage new model to become an “expert” for instances misclassified by earlier models
  - Rationale: models should be experts that complement each other
- Boosting generates a series of classifiers
- Their output is combined to produce a weighted vote
  - The weights of all classifiers that vote for a particular class are summed
  - The class with the greatest total is chosen

Random Forests

- Proposed by Breiman as an extension to CART
  - CART is a Decision Tree classifier, quite like C4.5
  - Create a ‘forest’ of trees as an ensemble

- Works like bagging, *except*
  - Decision tree is base learner
  - As well as selecting a random subset of the instances to train each tree, **select a [small] random subset of the attributes,** often N/3 or sqrt(N)

Leo Breiman, 1928-2005
(Source: his Wikipedia page)

Stacking

- Stacking, a.k.a Stacked Generalisation, is a family of methods for *heterogenous classifiers*
  - Classifiers are built using different classification algorithms
- Because of their different characteristics, stacking seeks to combine them in a way that is more sophisticated than a simple vote
  - Use the outputs of the individual classifiers as the input into a **meta-learner**
  - Individual classifiers: **level 0 models**
  - Meta-learner: **level 1 model** (usually a simple linear alg.)
  - Specifically, the training data for the meta-learner comes from classifying held-out data on the level 0 models

Bagging, Boosting & Stacking: Summary

- Bagging
  - Bootstrap Aggregation, decrease the variance by generating multiple data sets of the same size as the original data set
- Boosting
  - Uses subsets of the original data set to produce averagely performing models weighted towards “harder” problems. It then “boosts” the results through a vote.
- Stacking
  - Combines the results from multiple heterogeneous classifiers to improve classification. A voting scheme is not applied, instead a linear alg is applied to make the classification

Learning Objectives - Review

Having completed this topic, you should now be able to …

- Define Artificial Intelligence and Machine Learning
- Identify some key ML tasks and techniques that can be applied to tackle them
- Explain the basic ideas of some classic ML algorithms
- Explain the motivation for learning with ensembles
- Describe some ensemble learning approaches such as bagging, boosting and stacking
- Explain the principles of Random Forests and Gradient Boosting