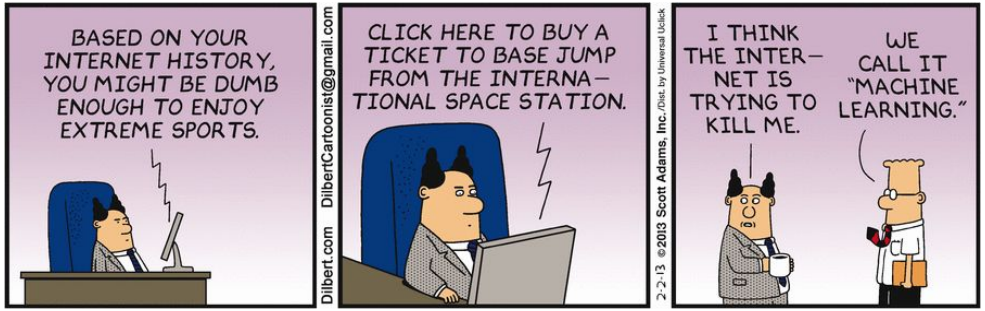
SB ML Meetup

Intro to Machine Learning and Decision Trees

Kevin Malta, Graduate Student at UCSB

Overview

- Introduction to Machine Learning
  - Motivation
  - Types of Learning
  - No Free Lunch Theorem
- Decision Trees
  - ID3
  - C4.5
- Ensembles of Decision Trees (if time permits)
  - Bagging
  - Boosting



Machine Learning and Data Science

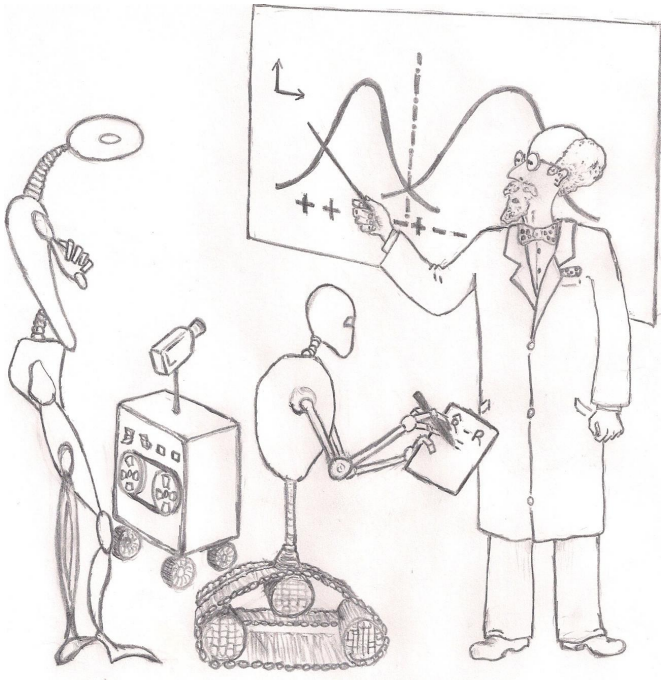
- Machine Learning:
  - The study of algorithms that
    - improve their performance
    - at some task
    - with experience
  - Machine Learning = Statistics
  - Regression - vs - Classification
- Data Science: the discipline/art of extracting information from data.
  - Data comes in all forms and its acquisition is accelerating
  - Estimated Google index 45 billion pages
  - Clickstream data: 10-100 TB/day
  - Transaction data: 5-50 TB/day
  - “We are drowning in data but starving for knowledge” - John Naisbitt (futurist)
- Computer science is to software engineering as machine learning is to data science

Popularity of Machine Learning

- Machine learning is the preferred approach to
  - Speech recognition, Natural language processing
  - Computer vision
  - Medical outcomes analysis
  - Robot control
- This trend is accelerating
  - Improved machine learning algorithms
  - Improved data capture, networking, faster computers
  - Software too complex to write by hand
  - New sensors / IO devices
  - Demand for self-customization to user, environment

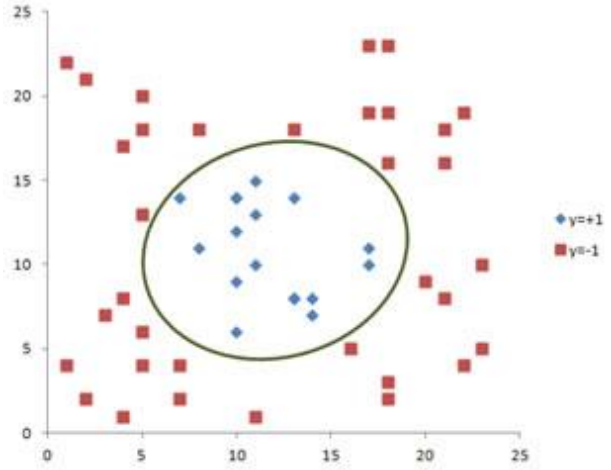
Different Learning Styles

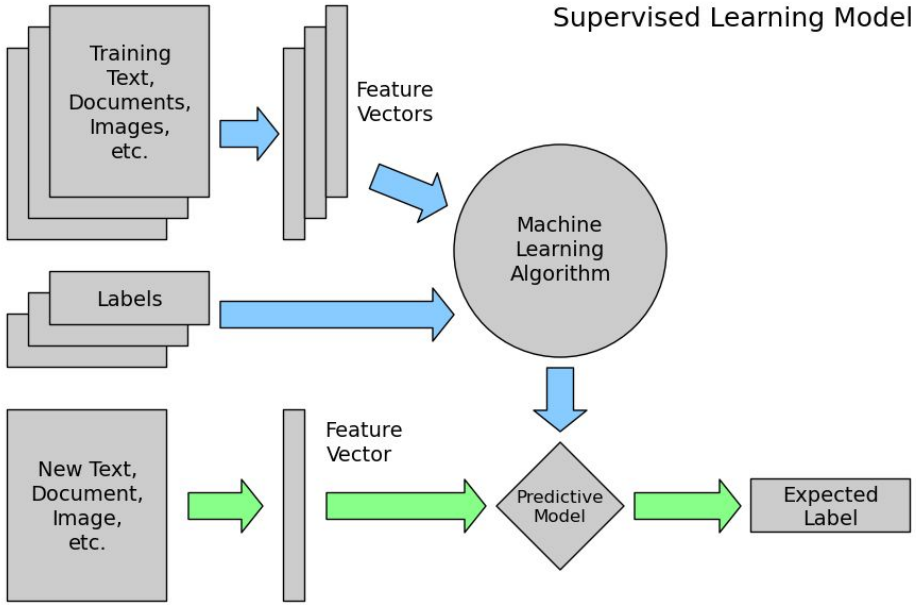
- Supervised Learning
- Unsupervised Learning
- Semi-supervised Learning
  - Active Learning



Supervised Learning

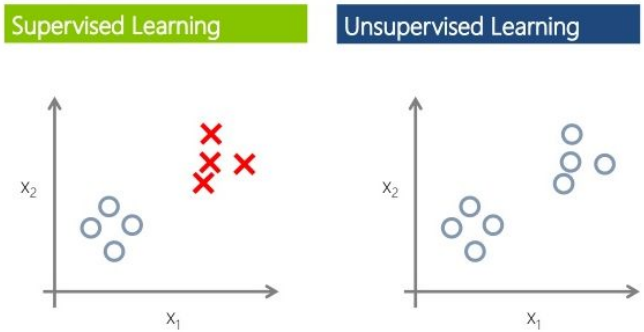
- Given: a collection of data points each with a label
- Teach the model by example
  - Extrapolation and generalization
  - E.g. the classification problem below

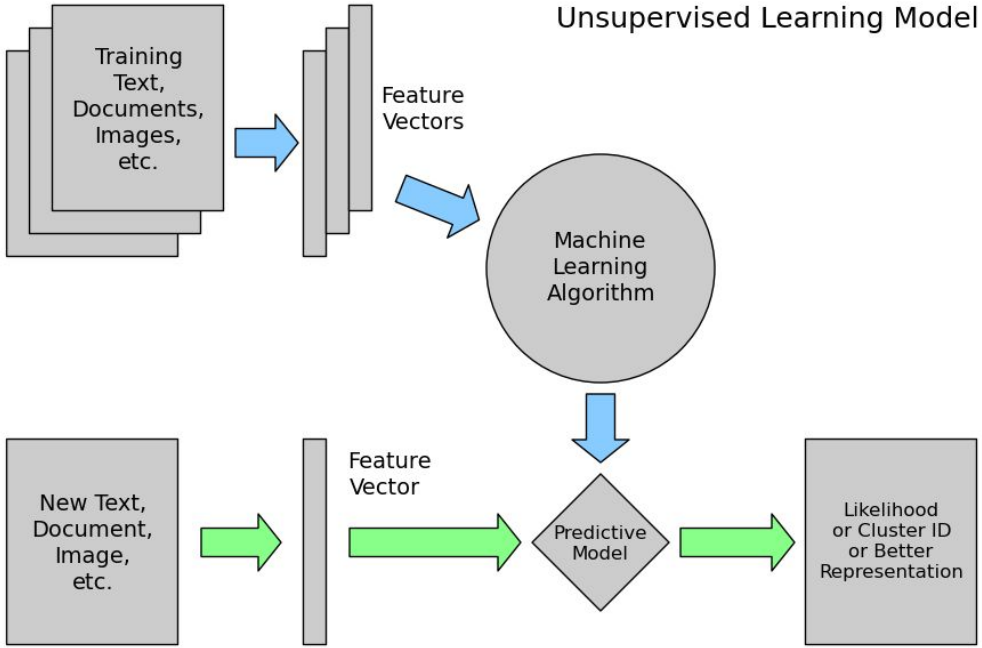




Unsupervised Learning

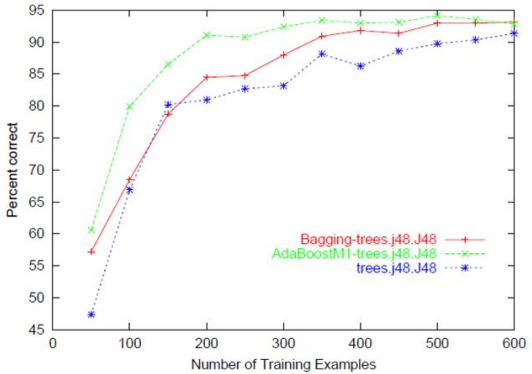
- Collection of unlabelled data
- Useful for:
  - Feature Extraction
  - When labels are not available
  - When features are not obvious
- E.g. k-means clustering
  - A form of feature extraction

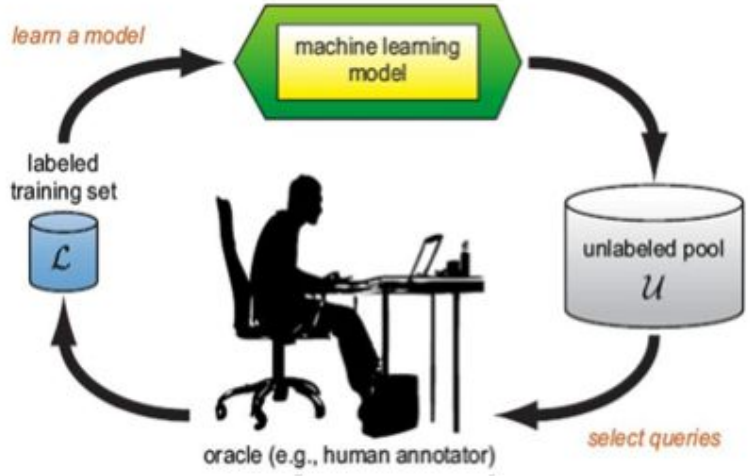




Semi-supervised Learning

- Some of the data is labelled, some is unlabelled
- E.g. Active Learning
  - If you have the opportunity to label data at an expense, which data points would you choose?
  - “Oracle” provides correct labels
    - e.g. Mechanical Turk
  - Learning Curve
    - trade off: labels acquired and accuracy

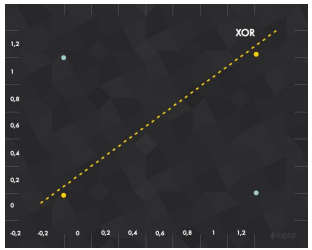




No Free Lunch Theorem

- “The computational cost of finding a solution, averaged over all problems in the class, is the same for any solution method.”
  - i.e. no solution offers a “short cut”
- Some are better at specific problems

x	y	x XOR y
0	0	0
0	1	1
1	0	1
1	1	0



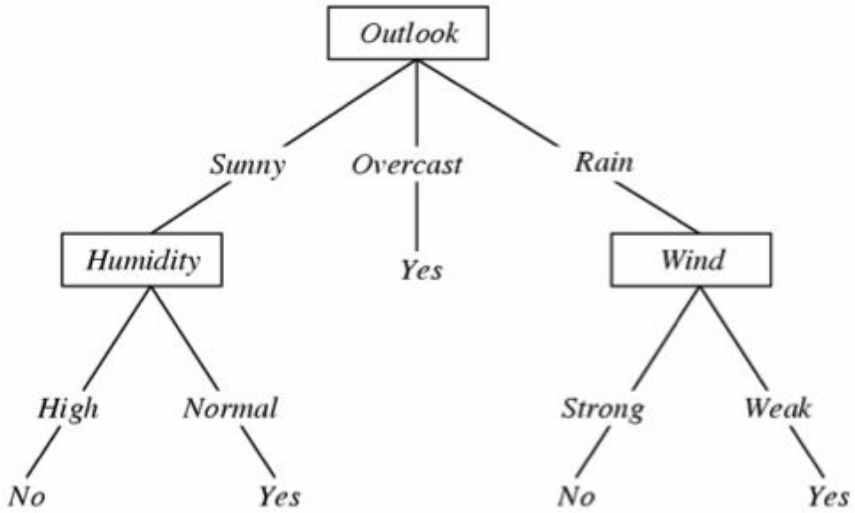
- The Folkloric version: not one algorithm can be viewed as the sledgehammer of Optimization/Machine Learning
  - I’m looking at you deep learning!

Decision Tree Learning

A collection of important algorithms for the data scientist's tool belt

Decision Tree for Playing Tennis

Day	Outlook	Temperature	Humidity	Wind	Play
D1	Sun	Hot	High	Low	No
D2	Sun	Hot	High	High	No
D3	Overcast	Hot	High	Low	Yes
D4	Rain	Sweet	High	Low	Yes
D5	Rain	Cold	Normal	Low	Yes
D6	Rain	Cold	Normal	High	No
D7	Overcast	Cold	Normal	High	Yes
D8	Sun	Sweet	High	Low	No
D9	Sun	Cold	Normal	Low	Yes
D10	Rain	Sweet	Normal	Low	Yes
D11	Sun	Sweet	Normal	High	Yes
D12	Overcast	Sweet	High	High	Yes
D13	Overcast	Hot	Normal	Low	Yes
D14	Rain	Sweet	High	High	No



Decision Trees

- Types of decision Trees
  - Iterative Dichotomizer 3 (ID3)
  - C4.5/C5
  - Classification and Regression Trees (CART)
  - CHAID
  - MARS
- Terms:
  - Training set
  - Data point
  - Attributes
  - Values
  - Target Attribute

Why Decision Trees?

- Benefits:
  - Simple to understand and interpret
  - Requires little data preparation
  - Able to handle both numerical and categorical data
  - Uses a white box model
  - Possible to validate a model using statistical tests
  - Robust
  - Performs well with large datasets
- Drawbacks:
  - Learning an optimal DT is NP-complete (greedy methods employed)
  - Overfitting
  - XOR and Parity
  - The “Information Gain” function is biased

Algorithmic Overview

- Recursively classify the examples based on one of the attributes until all examples have been used
- Branching is based on choosing the “best” attribute to classify on
  - We will talk about this in a little bit
- Recursion stops when:
  - All examples have the same value (1)
  - There are no more attributes (2)
  - There are no more examples (3)
- (3) needs no explanation, (1) and (2) will be discussed in the next slide
- Result: A model!
  - Prediction is achieved by traversing a tree with a data point
  - The leaf node is a decision

When does recursion stop?

- All examples have the same value (1)
  - when ancestor attributes and corresponding branch values, as well as the target attribute and value, are the same across examples

- There are no more attributes (2)
  - This happens when the training set is inconsistent
  - E.g., there are 2 or more examples having the same values for all but the target attribute
  - This is disambiguated by choosing the most popular target attribute value
  - This is a convention based on implementation

ID3

(Iterative Dichotomizer 3)

What is ID3?

- Developed by Ross Quinlan in the 1980s
- Prequel to C4.5 and C5
  - also developed by Ross Quinlan
- Very easy to use, but not perfect
- Picks attributes that favors the best reduction in information heterogeneity (entropy)
- Drawbacks:
  - Does not guarantee smallest possible decision tree
    - information gain does not guarantee optimal attribute selection
  - Works best with categorical data
    - It handles continuous variables by creating very “bushy” trees (this isn’t good
  - Some of these drawbacks are addressed in C4.5

Information Measure

- For example, flipping a fair coin once will give us events h and t each with probability ½ which means,
- A single flip of a coin gives us $-\log_2(1/2) = 1$ bit of information (h or t)
- Flipping a fair coin n times or n coins gives

$$-\log_2((1/2)^n) = \log_2(2^n) = n * \log_2(2) = n \text{ bits of information.}$$

- We could enumerate a sequence of 25 flips as, for example:

hthhtthhhhthttthhhhthtt = 10110010111010001011110100

- We thus get the nice fact that n flips of a fair coin gives us n bits of information

Entropy

- From information theory
  - The expected value of information of the distribution
- Given a random variable X, the (binary) entropy, H(X) is:

$$H(X) = -\sum_{i=1}^{n} P(X = i) \log_2 P(X = i)$$

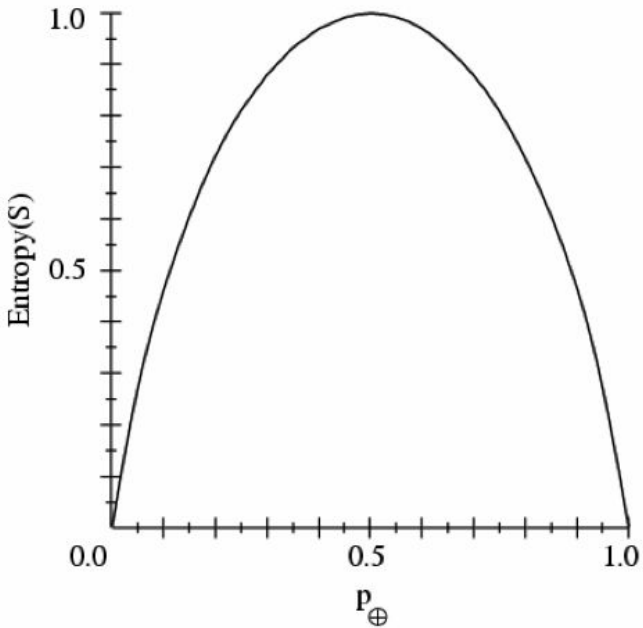
  - H(X) is the expected number of bits needed to encode a randomly drawn value of X (under most efficient code)
  - Why?
    - Most efficient code assigns log(1/P(X = i)) (base 2) bits to encode the message X = i
    - So, expected number of bits is:

$$\sum_{i=1}^{n} P(X = i)(-\log_2 P(X = i))$$

Sample Entropy

- S is a sample of training examples
  - Drawn from the universe
- The Probabilities can be replaced with proportions
  - p+ is the proportion of positive examples
  - p- is the proportion of negative examples
- Entropy measure the “homogeneity” or “purity” of the sample

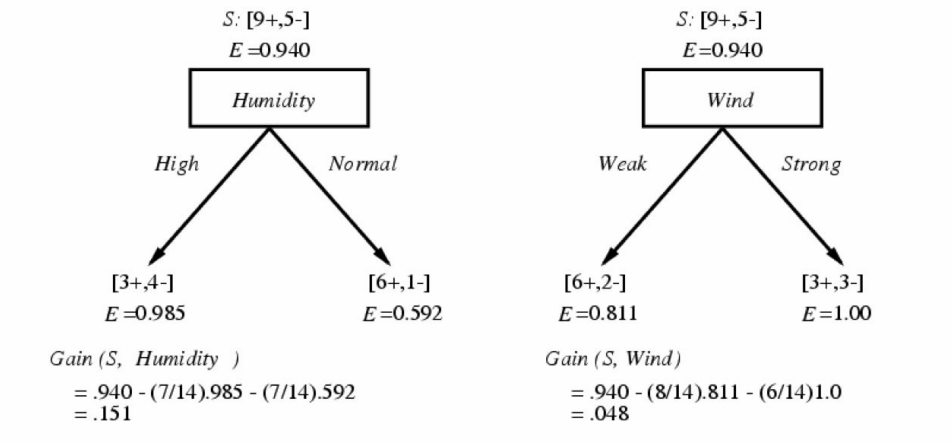
$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$



Information Gain

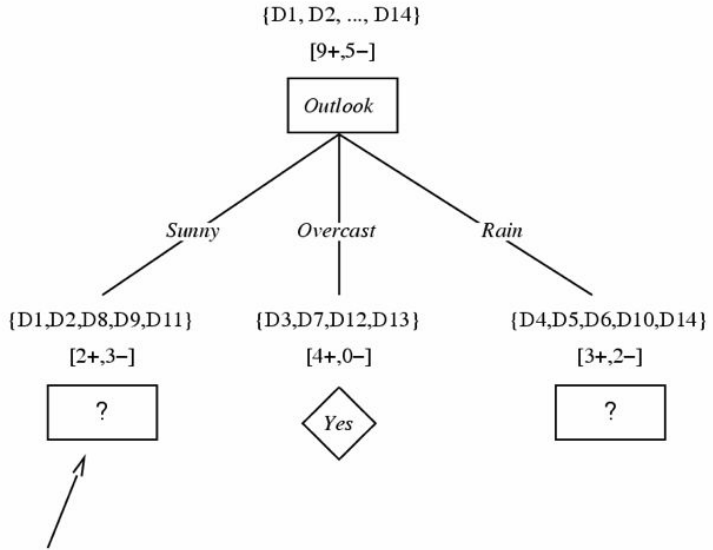
- Gain(S, A) = expected reduction in entropy due to sorting on an attribute A

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$



Selecting the Next Attribute

Day	Outlook	Temperature	Humidity	Wind	Play
D1	Sun	Hot	High	Low	No
D2	Sun	Hot	High	High	No
D3	Overcast	Hot	High	Low	Yes
D4	Rain	Sweet	High	Low	Yes
D5	Rain	Cold	Normal	Low	Yes
D6	Rain	Cold	Normal	High	No
D7	Overcast	Cold	Normal	High	Yes
D8	Sun	Sweet	High	Low	No
D9	Sun	Cold	Normal	Low	Yes
D10	Rain	Sweet	Normal	Low	Yes
D11	Sun	Sweet	Normal	High	Yes
D12	Overcast	Sweet	High	High	Yes
D13	Overcast	Hot	Normal	Low	Yes
D14	Rain	Sweet	High	High	No



S_{sunny} = {D1,D2,D8,D9,D11}

Gain (S_{sunny}, *Humidity*) = .970 − (3/5) 0.0 − (2/5) 0.0 = .970

Gain (S_{sunny}, *Temperature*) = .970 − (2/5) 0.0 − (2/5) 1.0 − (1/5) 0.0 = .570

Gain (S_{sunny}, *Wind*) = .970 − (2/5) 1.0 − (3/5) .918 = .019

More Issues with ID3

- Major limitation: overly sensitive to attributes with a large number of values
- E.g. if data points are people, their social security numbers are unique
- Unique valued attributes maximize the information gain equation, recall:

$$H(X) = -\sum_{i=1}^{n} P(X = i) \log_2 P(X = i)$$

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

C4.5

The Sequel

C4.5 Overview

- Nominated the top algorithm for data science (ICDM 2008 survey)
- Nodes no longer represent attributes, but tests on attributes
  - Three tests
    - Discrete Attributes:
      - Trivial Test, partition along each value
      - Partition on collections of the values
    - Continuous Attributes:
      - Thresholding
        - For n values, n-1 potential tests
- Features added in C4.5 that aren't included in ID3
  - continuous and discrete attributes using thresholding
  - Handles training data with missing attributes
  - different attribute weighting schemes
  - pruning the tree after creation

ID3 Unique Value Problem Revisited

- Solution to the ID3 unique value problem in C4.5:
- The Gain Ratio:

$$GainRatio(S, A) \equiv \frac{Gain(S, A)}{SplitInformation(S, A)}$$

$$SplitInformation(S, A) \equiv -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

where S_i is subset of S for which A has value v_i

Missing Attribute Values

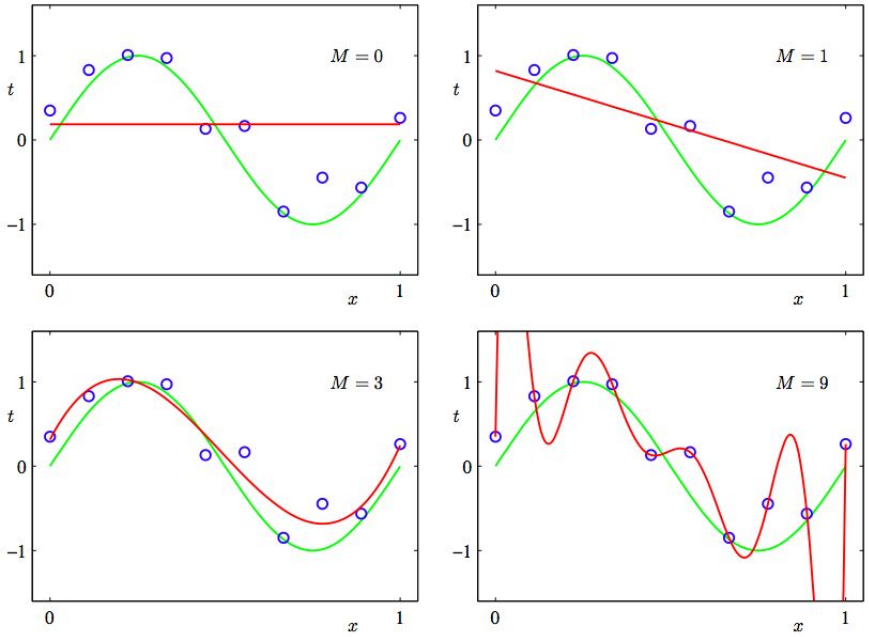
- What if some data fields of your data are missing values?
- Use the training example anyway
- Options:
  - If node n tests A, assign most common value of A among other example sorted to node n
  - Assign most common value of A among other examples with same target value
  - Assign probability p to each possible value v of A (this one is used in C4.5)
    - Propagate through tree by assigning fraction p of example to each descendent in tree
- New Equations:

$$G(S,B) = \frac{|S - S_0|}{|S|} G(S - S_0, B)$$

$$P(S,B) = -\frac{|S_0|}{|S|} \log\left(\frac{|S_0|}{|S|}\right) - \sum_{i=1}^{t} \frac{|S_i|}{|S|} \log\left(\frac{|S_i|}{|S|}\right)$$

Occam's Razor: The Law of Parsimony

- Heuristic technique to guide scientists in development of theoretical models
- In the context of machine learning: The simplest model is the one to choose
  - A complex hypothesis that fits the data is might be coincidence
  - A simple hypothesis that fits data is unlikely to be a coincidence
- In decision trees, this means smaller trees
  - Prevents "overfitting" and bolsters generalization
  - For example, "all trees with a prime number of nodes that use attributes beginning with 'Z'" is probably not correct



Pruning

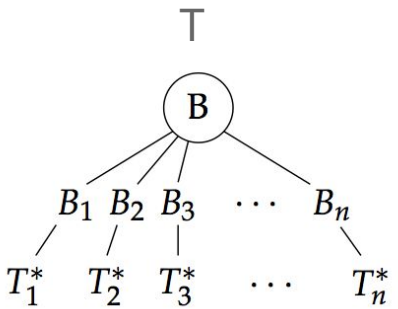
- Technique for decision trees to combat overfitting
- Types of pruning:
  - Minimum Description Length Principle
  - Post-Pruning
  - Reduced-Error Pruning
- Reduced-Error Pruning
  - Based on calculation of a confidence interval for the error rate
  - One wishes to estimate the *true* error rate E
    - This is determined over all data in the universe
    - Not at all feasible
  - Observe error rate F measured over the set of N training instances with M errors
  - True error rate is E, and U is an approximate upper bound for E

$$F = \begin{cases} (1-U)^N & \text{for } M = 0 \\ \sum_{i=0}^{M} \binom{N}{i} U^i (1-U)^{n-i} & \text{for } M > 0 \end{cases}$$

Reduced-Error Pruning

- Given:
  - T: Non-leaf decision tree
  - D: Training set
  - Pruned subtrees
  - B: a test
  - L: Leaf labelled with most frequent class in D
  - T*_f: Subtree of most frequent outcomes of B
- Compute:
  - Misclassifications of T, T*_f, and L
  - Find the lowest U amongst each of these
- If lowest is T*_f => Replace T by Leaf L
- If lowest is L => Replace T by T*_f
- Otherwise leave T unchanged

$$F = \begin{cases} (1-U)^N & \text{for } M = 0 \\ \sum_{i=0}^{M} \binom{N}{i} U^i (1-U)^{n-i} & \text{for } M > 0 \end{cases}$$



Ensembles Of Trees

Methods to Combine Decision Trees

Ensemble Methods

- Leverage a collection of models to achieve a prediction
  - Trade off between accuracy and diversity
    - Accuracy: how well the models perform
    - Diversity: do the models predict different data points correctly
- Different types of ensembles of decision trees:
  - Bagging
  - Boosted Trees
  - Rotation forest
- These methods commonly show up in data science competitions
  - E.g. Kaggle

Bagging (Random Forests)

- Bootstrap Aggregation = “Bagging”
- Proposed by Leo Breiman in 1994
- Bagging leads to “improvements for unstable features” - Breiman 1996
- Bootstrap Sampling:
  - Repeatedly sample a data set of size n with replacement, n times
- Given a data set Z of size n, bootstrap sample L times and construct L trees with each of the samples
- Note, each tree should have on average 63.2% of the unique examples from L (duplicates making up the rest)

Bagging Continued

BAGGING

Training phase

1. Initialize the parameters
   - $\mathcal{D} = \emptyset$, the ensemble.
   - L, the number of classifiers to train.
2. For $k = 1, \ldots, L$
   - Take a bootstrap sample S_k from $\mathbf{Z}$.
   - Build a classifier D_k using S_k as the training set.
   - Add the classifier to the current ensemble, $\mathcal{D} = \mathcal{D} \cup D_k$.
3. Return $\mathcal{D}$.

Classification phase

4. Run $D_1, \ldots, D_L$ on the input $\mathbf{x}$.
5. The class with the maximum number of votes is chosen as the label for $\mathbf{x}$.

- Classification is handled by plurality voting (majority wins)
- Regression is handled by averaging
- Why does this work?
  - Ensembles reduce variance
- When the classifier is a decision tree, we get a random forest!

Boosting (Trees)

- Goal: combine multiple base classifiers whose combined performance is significantly better than that of any of the base classifiers
- Sequentially trains weak learners
  - Each base classifier is trained on data that is weighted based on the performance of the previous classifier
- There are multiple boosting algorithms:
  - Hedge
  - AdaBoost
- AdaBoost
  - “Adaptive Boosting” = AdaBoost
  - Won Gödel prize in 2003
  - Sensitive to noisy data and outliers

AdaBoost

1. Start with equally weighted data
2. Apply first classifier
3. Repeat
    a. Increase weights for misclassified data
    b. Apply subsequent classifier

- Optimization of a cost function
    - Other algorithms use gradients to update weights
- Classification is achieved through voting
- In practice, overfitting rarely occurs (Bishop)

ADABOOST.M1

Training phase

1. Initialize the parameters
   - Set the weights $\mathbf{w}^1 = [w_1, \ldots, w_N]$, $w_j^1 \in [0, 1]$, $\sum_{j=1}^{N} w_j^1 = 1$. (Usually $w_j^1 = \frac{1}{N}$).
   - Initialize the ensemble $\mathcal{D} = \emptyset$.
   - Pick L, the number of classifiers to train.
2. For $k = 1, \ldots, L$
   - Take a sample S_k from $\mathbf{Z}$ using distribution $\mathbf{w}^k$.
   - Build a classifier D_k using S_k as the training set.
   - Calculate the weighted ensemble error at step k by

$$\epsilon_k = \sum_{j=1}^{N} w_j^k l_k^j, \tag{7.11}$$

   ($l_k^j = 1$ if D_k misclassifies $\mathbf{z}_j$ and $l_k^j = 0$ otherwise.)
   - If $\epsilon_k = 0$ or $\epsilon_k \geq 0.5$, ignore D_k, reinitialize the weights w_j^k to $\frac{1}{N}$ and continue.
   - Else, calculate

$$\beta_k = \frac{\epsilon_k}{1 - \epsilon_k}, \quad \text{where} \quad \epsilon_k \in (0, 0.5), \tag{7.12}$$

   - Update the individual weights

$$w_j^{k+1} = \frac{w_j^k \beta_k^{(1-l_k^j)}}{\sum_{i=1}^{N} w_i^k \beta_k^{(1-l_k^i)}}, \quad j = 1, \ldots, N. \tag{7.13}$$

3. Return $\mathcal{D}$ and $\beta_1, \ldots, \beta_L$.

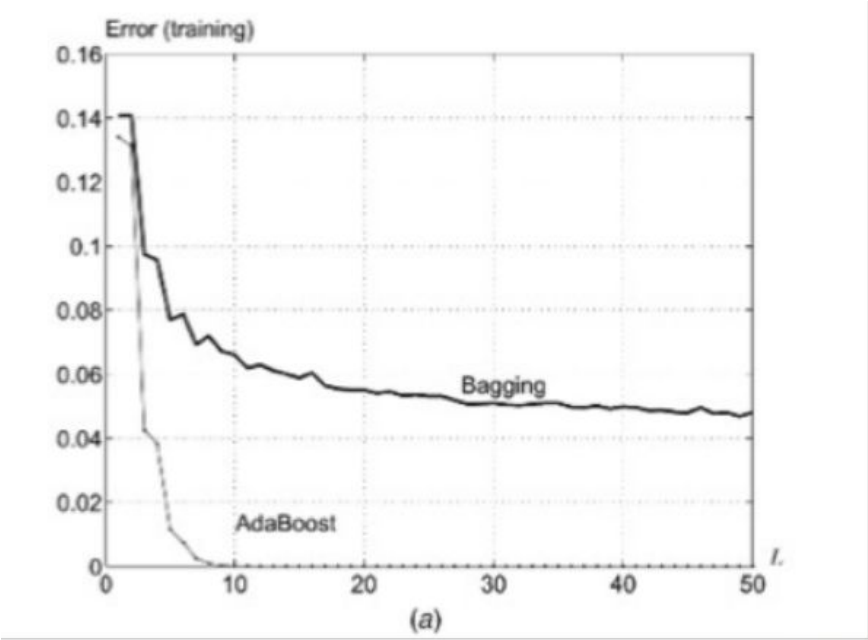
Classification phase

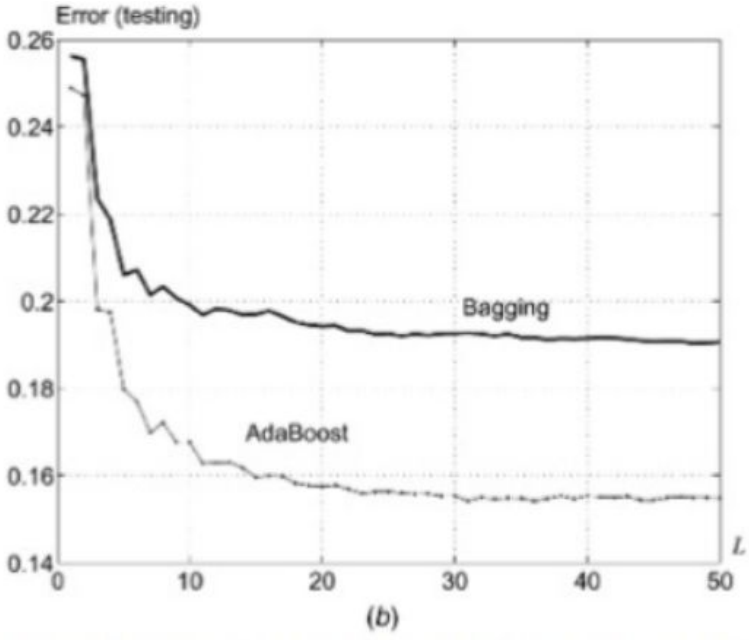
4. Calculate the support for class ω_t by

$$\mu_t(\mathbf{x}) = \sum_{D_k(\mathbf{x}) = \omega_t} \ln\left(\frac{1}{\beta_k}\right). \tag{7.14}$$

5. The class with the maximum support is chosen as the label for $\mathbf{x}$.

Bagging -vs- Boosting





Takeaways

- Machine Learning
  - What is it?
  - Why is it important?
- Data Science
  - What is it?
  - How does it relate to Machine Learning/employ machine learning?
- Decision Tree Learning
  - Greedy top-down learning of decision tree (ID3, C4.5, ...)
  - Overfitting and pruning
  - Extensions
- Ensembles of Decision Trees
  - How to create an ensemble
  - How to leverage the ensemble for a more accurate prediction

Thank You