Lecture 8: Decision trees

EMAT31530/Feb 2018/Raul Santos-Rodriguez

Have a look at ...

... Russell and Norvig (Ch. 18.3)

... Hastie, Tibshirani, Friedman. The elements of statistical learning, (Ch. 9.2)

... Criminisi, Shotton, Konukoglu. Decision Forests: A Unified Framework for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning. Microsoft research.

... Python: `http://scikit-learn.org/`

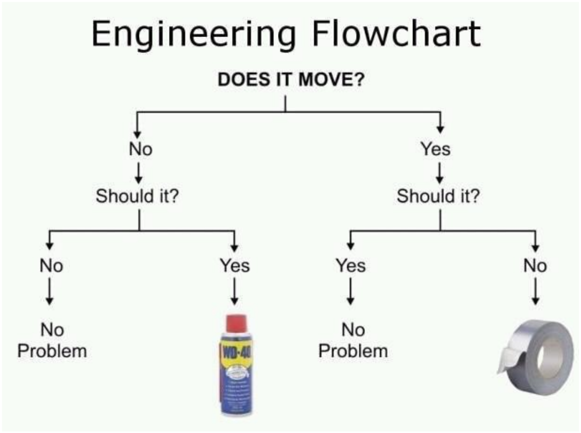
... Java: `http://www.cs.waikato.ac.nz/ml/weka/`

Outline

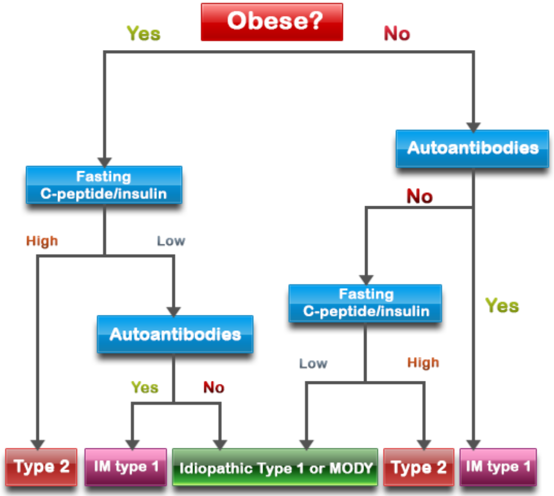
We will discuss:

- Decision tree structure
- Learning with decision trees
- Random forests

Motivation: trees and decisions



Motivation: trees and decisions



Motivation: object recognition

?

?

?

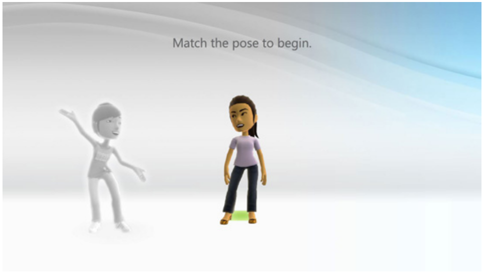
Motivation: object recognition

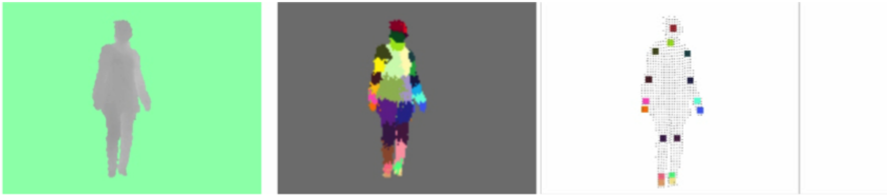
CAR

SCOOTER

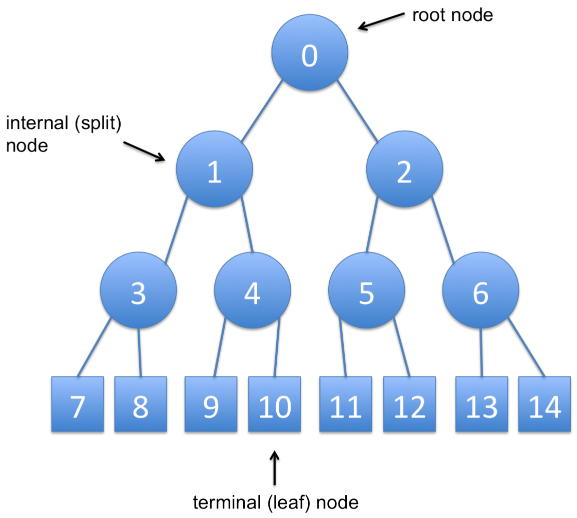
PEDESTRIAN

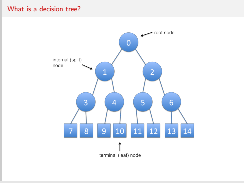
Motivation: pose recognition





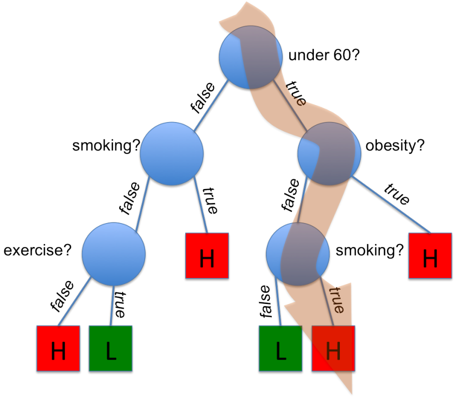
What is a decision tree?




What is a decision tree?

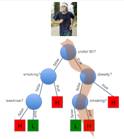
A tree is a special type of graph. It is a data structure made of a collection of nodes and edges organized in a hierarchical fashion. Nodes are divided into internal (or split) nodes and terminal (or leaf) nodes. We denote internal nodes with circles and terminal ones with squares. All nodes have exactly one incoming edge. In contrast to general graphs a tree cannot contain loops.
A decision tree represents a function that takes as input a vector of attribute values and returns a decision (a single output value). The input and output values can be discrete or continuous. For now we will concentrate on problems where the inputs have discrete values and the output has exactly two possible values; this is binary classification, where each positive example input will be classified as true (a positive example) or false (a negative example). Each internal node has exactly two or more outgoing edges.

Insurance risk assessment example



Insurance risk assessment example

Insurance risk assessment example



A decision tree reaches its decision by performing a sequence of tests. Each internal node in the tree corresponds to a test of the value of one of the input attributes, and the branches from the node are labeled with the possible values of the attribute. Each leaf node in the tree specifies a value to be returned by the function.

From data to trees

Examples described by attribute values (Boolean, categorical, continuous, etc.)
E.g., situations where I will assess the insurance risk as high/low:

Example	Attributes					Target
	Age < 60	*Smoking*	*Exercise*	*Obesity*	*City*	*Risk*
X_1	*F*	*F*	*F*	*T*	*Manchester*	*H*
X_2	*F*	*F*	*F*	*T*	*Liverpool*	*H*
X_3	*F*	*T*	*F*	*F*	*Bristol*	*H*
X_4	*F*	*F*	*T*	*F*	*Liverpool*	*L*
X_5	*T*	*F*	*T*	*F*	*Manchester*	*L*
X_6	*T*	*T*	*F*	*T*	*London*	*H*
X_7	*T*	*F*	*F*	*F*	*London*	*L*
X_8	*F*	*F*	*T*	*T*	*Bristol*	*L*

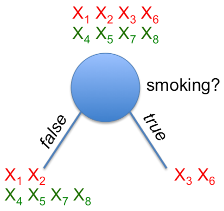
Classification of examples is low (L) or high (H)

From data to trees

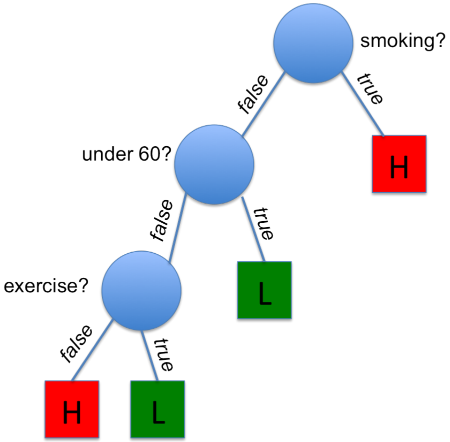
Examples described by attribute values (Boolean, categorical, continuous, etc.)
E.g., situations where I will assess the insurance risk as high/low:

Example	Attributes					Target
	Age < 60	*Smoking*	*Exercise*	*Obesity*	*City*	*Risk*
X_1	*F*	*F*	*F*	*T*	*Manchester*	*H*
X_2	*F*	*F*	*F*	*T*	*Liverpool*	*H*
X_3	*F*	*T*	*F*	*F*	*Bristol*	*H*
X_4	*F*	*F*	*T*	*F*	*Liverpool*	*L*
X_5	*T*	*F*	*T*	*F*	*Manchester*	*L*
X_6	*T*	*T*	*F*	*T*	*London*	*H*
X_7	*T*	*F*	*F*	*F*	*London*	*L*
X_8	*F*	*F*	*T*	*T*	*Bristol*	*L*

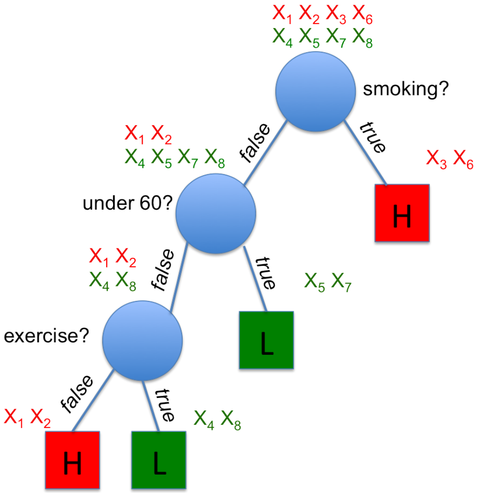
Classification of examples is low (L) or high (H)



A learned decision tree



A learned decision tree



Question

To achieve good generalisation, how to construct the tree?

$\boxed{\text{Few nodes}}$

$\boxed{\text{As many nodes as possible}}$

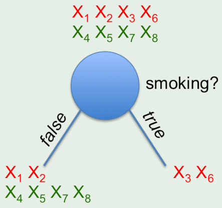
$\boxed{\text{A random number of nodes}}$

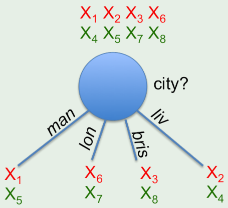
How to construct a decision tree?

Aim: find a small tree consistent with the training examples
Idea: (recursively) choose "most significant" attribute as root of (sub)tree

A good attribute splits the examples into subsets that are (ideally) "all red" or "all green"

How to pick a question?



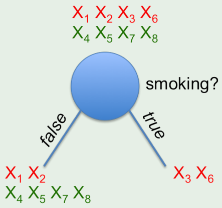


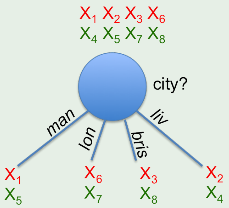
How to construct a decision tree?

Aim: find a small tree consistent with the training examples
Idea: (recursively) choose "most significant" attribute as root of (sub)tree

A good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"

How to pick a question?





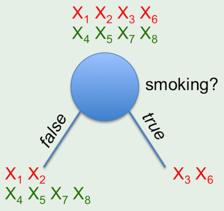
smoking? is a better choice → gives *information* about the classification

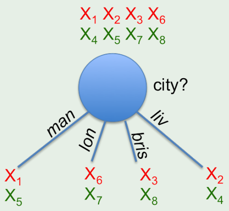
How to pick a question: entropy

Aim: find a small tree consistent with the training examples
Idea: (recursively) choose "most significant" attribute as root of (sub)tree

A good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"

How to pick a question?

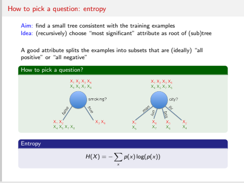




Entropy

$$H(X) = -\sum_x p(x)\log(p(x))$$

How to pick a question: entropy



How to pick a question: entropy

Aim: find a small tree consistent with the training examples
Idea: (recursively) choose "most significant" attribute as root of (sub)tree

A good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"

How to pick a question?

Entropy

$$H(X) = -\sum_{x} p(x)\log(p(x))$$

The idea is to pick the attribute that goes as far as possible toward providing an exact classification of the examples. A perfect attribute divides the examples into sets, each of which are all positive or all negative and thus will be leaves of the tree. We will use the notion of **information gain**, which is defined in terms of entropy.

How to pick a question: example

Define $B(q)$ as the entropy of a Boolean random variable that is true with probability q:

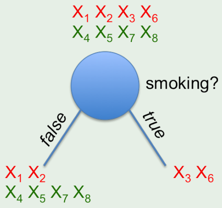
$$B(q) = -(q \log_2 q + (1-q) \log_2 (1-q))$$

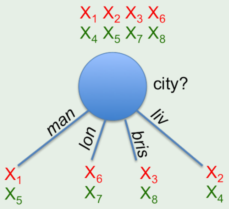
If a training set contains p positive examples and n negative examples, then the entropy of the goal attribute on the whole set is

$$H(Goal) = B\left(\frac{p}{p+n}\right)$$

In our example $p = n = 4 \rightarrow B(0.5) = 1$ bit.

How to pick a question?





How to pick a question: information gain

An attribute A with d values splits the training set E into subsets $E_1, \ldots, E_d$. Each subset E_k has p_k positive examples and n_k negative examples.

Remainder

The expected entropy remaining after testing attribute A is

$$Remainder(A) = \sum_{k=1}^{d} \frac{p_k + n_k}{p + n} B\left(\frac{p_k}{p_k + n_k}\right)$$

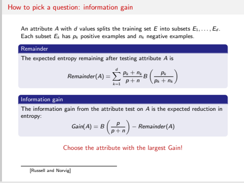
Information gain

The information gain from the attribute test on A is the expected reduction in entropy:

$$Gain(A) = B\left(\frac{p}{p + n}\right) - Remainder(A)$$

Choose the attribute with the largest Gain!

[Russell and Norvig]



How to pick a question: information gain

An attribute A with d values splits the training set E into subsets $E_1, \ldots, E_d$. Each subset E_k has p_k positive examples and n_k negative examples.

Remainder

The expected entropy remaining after testing attribute A is

$$Remainder(A) = \sum_{k=1}^{d} \frac{p_k + n_k}{p + n} B\left(\frac{p_k}{p_k + n_k}\right)$$

Information gain

The information gain from the attribute test on A is the expected reduction in entropy:

$$Gain(A) = B\left(\frac{p}{p+n}\right) - Remainder(A)$$

Choose the attribute with the largest Gain!

[Russell and Norvig]

An attribute A with d values splits the training set E into subsets $E_1, \ldots, E_d$. Each subset E_k has p_k positive examples and n_k negative examples. If we go along that branch, we will need and additional $B(p_k/(p_k + n_k))$ bits of information to answer the question. A randomly chosen example from the training set has the kth value from the attribute with probability $(p_k + n_k)/(p + n)$, so the expected entropy remaining after testing attribute A is

$$Remainder(A) = \sum_{k=1}^{d} \frac{p_k + n_k}{p + n} B\left(\frac{p_k}{p_k + n_k}\right)$$

The information gain from the attribute test on A is the expected reduction in entropy:

$$Gain(A) = B\left(\frac{p}{p+n}\right) - Remainder(A)$$

In fact $Gain(A)$ is just what we need as a measure to pick a question.

How to pick the nodes: example

A test on a single attribute might give us only part of the 1 bit: entropy remaining after the attribute test.

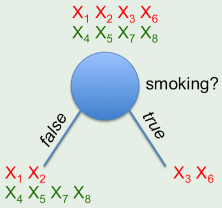
$$Remainder(Smoking) = \frac{2}{8}B(1) + \frac{6}{8}B\left(\frac{2}{6}\right) \approx 0.689 bits$$

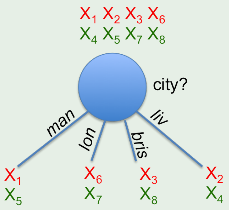
$$Remainder(City) = \frac{2}{8}B(0.5) + \frac{2}{8}B(0.5) + \frac{2}{8}B(0.5) + \frac{2}{8}B(0.5) = 1 bits$$

$$Gain(Smoking) = 1 - 0.689 = 0.311 bits$$

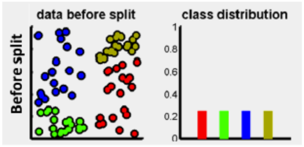
$$Gain(City) = 1 - 1 = 0 bits$$

How to pick a question?

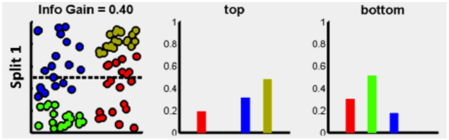




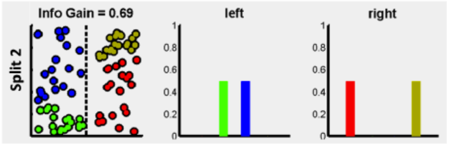
How to pick the nodes: example



(a)



(b)



(c)

Fig. 2.5 Information gain for discrete, non-parametric distributions. (a) Dataset $\mathcal{S}$ before a split. (b) After a horizontal split. (c) After a vertical split. In this example the vertical split produces purer class distributions in the child nodes. Classes are colour coded.

[Criminisi, Shotton, Konukoglu]

Generalisation revisited

Goal: we want to optimize the expected information gain of an unseen test point (not in training data).

... since we cannot know what that test point will be (e.g., in the case of the insurance data, we cannot know whether the next data point will have as a City attribute, London, Manchester, Bristol or Liverpool) $\rightarrow$ we have to take an expectation over the distribution of data.

... since we do not know how the data is intrinsically distributed, we use the empirical distribution of the training data. Therefore when doing the split at a node, we need to weight the entropy of the leaf nodes according to the train population that goes in each leaf (what fraction of the training data went into which leaf).

If we exclude these weights:

WHITEBOARD

Decision tree algorithm

Algorithm

```
function DECISION-TREE-LEARNING(examples, attributes, parent examples) returns a tree

  if examples is empty then return PLURALITY-VALUE(parent examples)
  else if all examples have the same classification then return the classification
  else if attributes is empty then return PLURALITY-VALUE(examples)
  else
    $A \leftarrow \arg\max_{a \in attributes}$ IMPORTANCE($a$, examples)
    tree ← a new decision tree with root test $A$
    for each value $v_k$ of $A$ do
      $exs \leftarrow \{e : e \in examples \quad \text{and} \quad e.A = v_k\}$
      subtree ←DECISION-TREE-LEARNING($exs$, $attributes - A$, examples)
      add a branch to tree with label $(A = v_k)$ and subtree subtree
    return tree
```

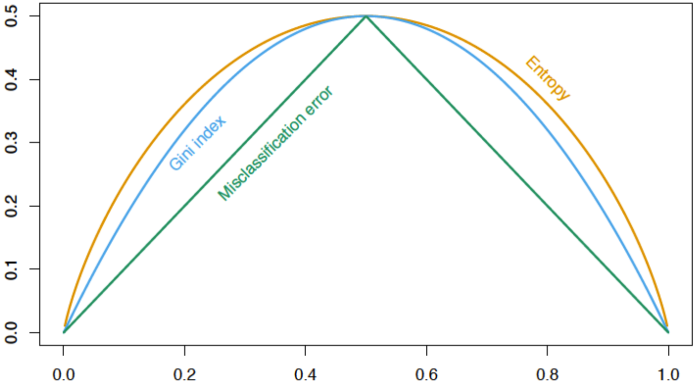
IMPORTANCE

The function IMPORTANCE(a, *examples*) returns the information gain of *atrribute a* for the examples in *examples*.

PLURALITY-VALUE

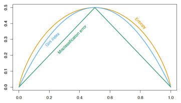
The function PLURALITY-VALUE(*examples*) selects the most common output value among a set of *examples*, breaking ties randomly.

Decision trees in practice: alternative quality measures



Entropy	$-q \log q - (1-q) \log(1-q)$
Gini index	$2q(1-q)$
Misclassification error	$1 - \max(q, 1-q)$

Decision trees in practice: alternative quality measures



Entropy	$-q \log q - (1-q)\log(1-q)$
Gini index	$2q(1-q)$
Misclassification error	$1 - \max(q, 1-q)$

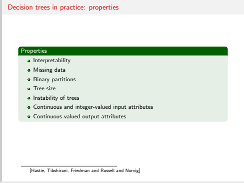
All three are similar, but entropy and the Gini index are differentiable, and hence more amenable to numerical optimization.

Decision trees in practice: properties

Properties

- Interpretability
- Missing data
- Binary partitions
- Tree size
- Instability of trees
- Continuous and integer-valued input attributes
- Continuous-valued output attributes

[Hastie, Tibshirani, Friedman and Russell and Norvig]



Decision trees in practice: properties

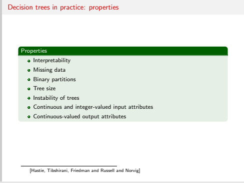
Properties

- Interpretability
- Missing data
- Binary partitions
- Tree size
- Instability of trees
- Continuous and integer-valued input attributes
- Continuous-valued output attributes

[Hastie, Tibshirani, Friedman and Russell and Norvig]

- Interpretability: In many areas of industry and commerce, decision trees are usually the first method tried when a classification method is to be extracted from a data set. One important property of decision trees is that it is possible for a human to understand the reason for the output of the learning algorithm. This representation is also popular among medical scientists, perhaps because it mimics the way that a doctor thinks. The tree stratifies the population into strata of high and low outcome, on the basis of patient characteristics.
- Missing data: In many domains, not all the attribute values will be known for every example. The values might have gone unrecorded, or they might be too expensive to obtain. This gives rise to two problems: First, given a complete decision tree, how should one classify an example that is missing one of the test attributes? Second, how should one modify the information-gain formula when some examples have unknown values for the attribute?
- Binary partitions: Rather than splitting each node into just two groups at each stage, we might consider multiway splits into more than two groups. While this can sometimes be useful, it is not a good general strategy. The problem is that multiway splits fragment the data too quickly, leaving insufficient data at the next level down. Hence we would want to use such splits only when needed. Since multiway splits can be achieved by a series of binary splits, the latter are preferred.

Decision trees in practice: properties

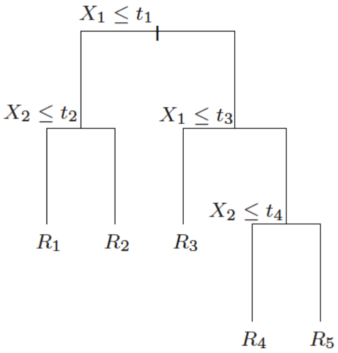

Decision trees in practice: properties

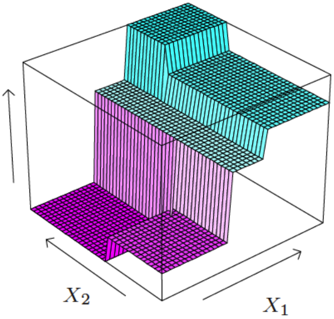
Properties
- Interpretability
- Missing data
- Binary partitions
- Tree size
- Instability of trees
- Continuous and integer-valued input attributes
- Continuous-valued output attributes

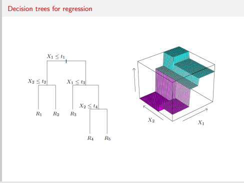
[Hastie, Tibshirani, Friedman and Russell and Norvig]

- Tree size: A very large tree might overfit the data, while a small tree might not capture the important structure. Tree size is a tuning parameter governing the model's complexity, and the optimal tree size should be adaptively chosen from the data. The preferred strategy is to grow a large tree, stopping the splitting process only when some minimum node size (say 5) is reached. Then this large tree is **pruned** using different strategies.
- Instability of trees: one major problem with trees is their high variance. Often a small change in the data can result in a very different series of splits, making interpretation somewhat precarious. The major reason for this instability is the hierarchical nature of the process: the effect of an error in the top split is propagated down to all of the splits below it. It is the price to be paid for estimating a simple, tree-based structure from the data.
- Continuous and integer-valued input attributes: Continuous or integer-valued attributes such as Height and Weight, have an infinite set of possible values. Rather than generate infinitely many branches, decision-tree learning algorithms typically find the split point that gives the highest information gain. For example, at a given node in the tree, it might be the case that testing on Weight ¿ 160 gives the most information. Efficient methods exist for finding good split points: start by sorting the values of the attribute, and then consider only split points that are between two examples in sorted order that have different classifications, while keeping track of the running totals of positive and negative examples on each side of the split point. Splitting is the most expensive part of real-world decision tree learning applications.

Decision trees for regression

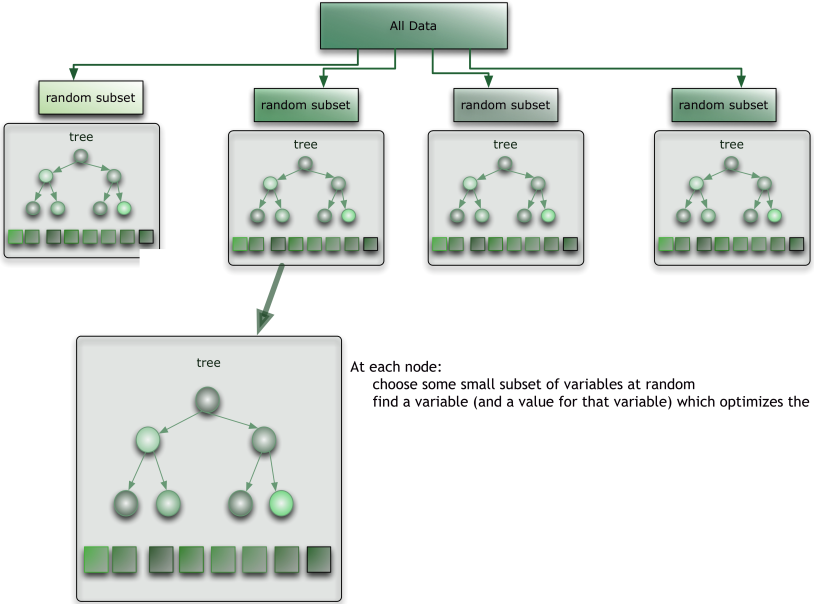




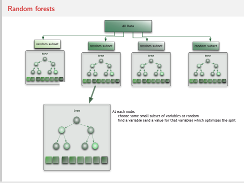

Decision trees for regression

Continuous-valued output attributes: If we are trying to predict a numerical output value, such as the price of an apartment, then we need a regression tree rather than a classification tree. A regression tree has at each leaf a linear function of some subset of numerical attributes, rather than a single value.

Random forests



At each node:
- choose some small subset of variables at random
- find a variable (and a value for that variable) which optimizes the split



Random forests

At each node:
choose some small subset of variables at random
find a variable (and a value for that variable) which optimizes the split

A random decision forest is an ensemble of randomly trained decision trees. The key aspect of the forest model is the fact that its component trees are all randomly different from one another. This leads to decorrelation between the individual tree predictions and, in turn, results in improved generalisation and robustness. The forest model is characterised by the same components as the decision trees.
Each classifier, individually, is a weak learner, while all the classifiers taken together are a strong learner. Thus, in ensemble terms, the trees are weak learners and the random forest is a strong learner.

Random forests algorithm

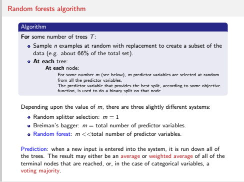
Algorithm

```
For some number of trees $T$:
    • Sample $n$ examples at random with replacement to create a subset of the data (e.g. about 66% of the total set).
    • At each tree:
        At each node:
            For some number $m$ (see below), $m$ predictor variables are selected at random from all the predictor variables.
            The predictor variable that provides the best split, according to some objective function, is used to do a binary split on that node.
```

Depending upon the value of m, there are three slightly different systems:

- Random splitter selection: $m = 1$
- Breiman's bagger: $m =$ total number of predictor variables.
- Random forest: $m <<$total number of predictor variables.

Prediction: when a new input is entered into the system, it is run down all of the trees. The result may either be an average or weighted average of all of the terminal nodes that are reached, or, in the case of categorical variables, a voting majority.



Random forests algorithm

Algorithm

```
For some number of trees $T$:
  • Sample $n$ examples at random with replacement to create a subset of the
    data (e.g. about 66% of the total set).
  • At each tree:
        At each node:
            For some number $m$ (see below), $m$ predictor variables are selected at random
            from all the predictor variables.
            The predictor variable that provides the best split, according to some objective
            function, is used to do a binary split on that node.
```

Depending upon the value of m, there are three slightly different systems:

- Random splitter selection: $m = 1$
- Breiman's bagger: $m =$ total number of predictor variables.
- Random forest: $m <<$total number of predictor variables.

Prediction: when a new input is entered into the system, it is run down all of the trees. The result may either be an average or weighted average of all of the terminal nodes that are reached, or, in the case of categorical variables, a voting majority.

With a large number of predictors, the eligible predictor set will be quite different from node to node. The greater the inter-tree correlation, the greater the random forest error rate, so one pressure on the model is to have the trees as uncorrelated as possible. As m goes down, both inter-tree correlation and the strength of individual trees go down. So some optimal value of m must be discovered.

Random forests in practice

Strengths: Random forest runtimes are quite fast, and they are able to deal with unbalanced and missing data.

Weaknesses: When used for regression they cannot predict beyond the range in the training data, and that they may over-fit datasets that are particularly noisy.

Parameters

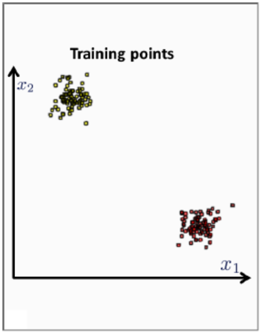
The parameters that most influence the behavior of a decision forest are:

- The maximum allowed tree depth D;
- The amount of randomness and its type;
- The forest size (number of trees) T;
- The choice of weak learner model;
- The training objective function;
- The choice of features in practical applications.

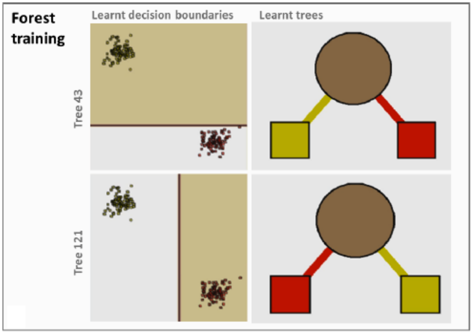
[Criminisi, Shotton, Konukoglu]

`https://www.theguardian.com/science/the-lay-scientist/2016/feb/18/has-a-rampaging-ai-algorithm-really-killed-thousands-in-pakistan`

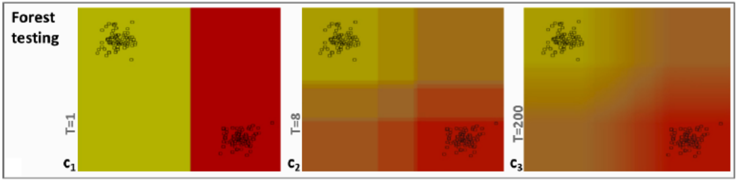
Random forests in practice: effect of T



(a)



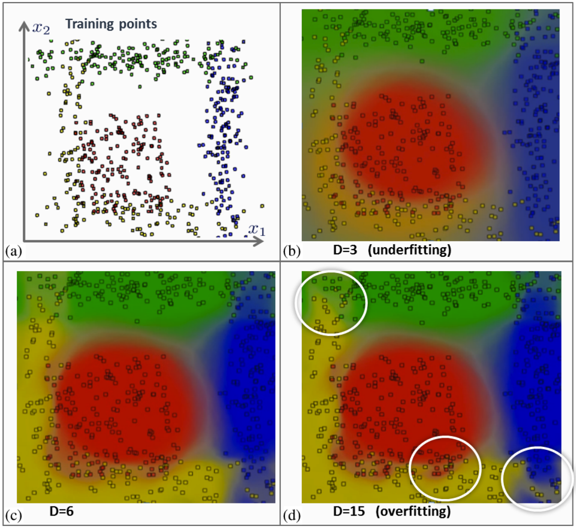
(b)



(c)

[Criminisi, Shotton, Konukoglu]

Random forests in practice: effect of D



[Criminisi, Shotton, Konukoglu]

Next lecture

We will start with a new topic: search!