Artificial Intelligence

Lecture 14 – Decision Trees

Outline

- Classification and regression
- Inducing decision trees
- Example: waiting for a table in a restaurant
- Choosing informative attributes
- Expressiveness of decision trees
- Exercise: spam filtering

Inductive Learning

- A form of supervised learning, where we are given the correct (or approximately correct) value of the function for particular inputs
- Each example is of the form (x, $f(x)$), where x is the input(s) and $f(x)$ is the value of the function for x
- Given a set of examples (x, $f(x)$), generate a function $h(x)$ which approximates f
- h is called a hypothesis
- A hypothesis should be
  - consistent (i.e., should fit the data)
  - generalise well (predict unseen examples correctly)

Decision Trees

- Decision tree learning is one of the simplest forms of inductive learning
- Decision trees often represent boolean functions - input is an object or situation described by a set of attributes; value is a yes/no decision
- Aim is to find a compact representation of the example set that also correctly predicts on unseen cases (the test set)
- Proceed by repeatedly choosing the most informative property to split the samples on the value

Classification vs Regression

- In a decision tree, each example is of the form (x, $f(x)$), where the input x is described by a set of attributes and $f(x)$ is the decision for the input x
- The attributes and the output (decision) can be *discrete* or *continuous*
- Learning a discrete valued function is called *classification learning*, as each input is associated with one of finitely many classes
- Learning a continuous function is called *regression*
- Focus on *boolean classification*, where each input is classified as being a member (or not) of a single class, or the output is a yes/no decision

How Decision Trees Work

- Examples are described by a set of attributes
- A decision tree classifies an example by performing a number of tests on these attributes
- Each internal node in the tree corresponds to a test of the value of one of the attributes
- Arcs from the node are labelled with the possible results of the test
- A test of an attribute appears at most once on any given branch of the tree
- Each leaf node specifies a value to be returned (a classification of the input) if that leaf is reached

Example: Waiting for a Table

- Problem is whether to wait for a table if a restaurant currently has no free tables
- Aim to learn a definition for the goal predicate (decision) *WillWait*
- In a given situation, if *WillWait* is true, we will wait for a table; otherwise we will try another restaurant
- We assume that we have lsome examples of previous situations where we had to decide whether to wait for a table, and what out decision was in that situation
- We assume that our previous decisions are correct - aim is just to learn the function to decide whether to wait for a table

Example: Waiting for a Table

- We choose the following attributes to describe the examples

  Alternate: whether there is another restaurant nearby

  Bar: whether the restaurant has a bar to wait in

  Fri/Sat: true on Fridays and Saturdays

  Hungry: whether we are hungry

  Patrons: how many people are in the restaurant (*None*, *Some*, *Full*)

  Price: restaurant's price range ($, $$, $$$)

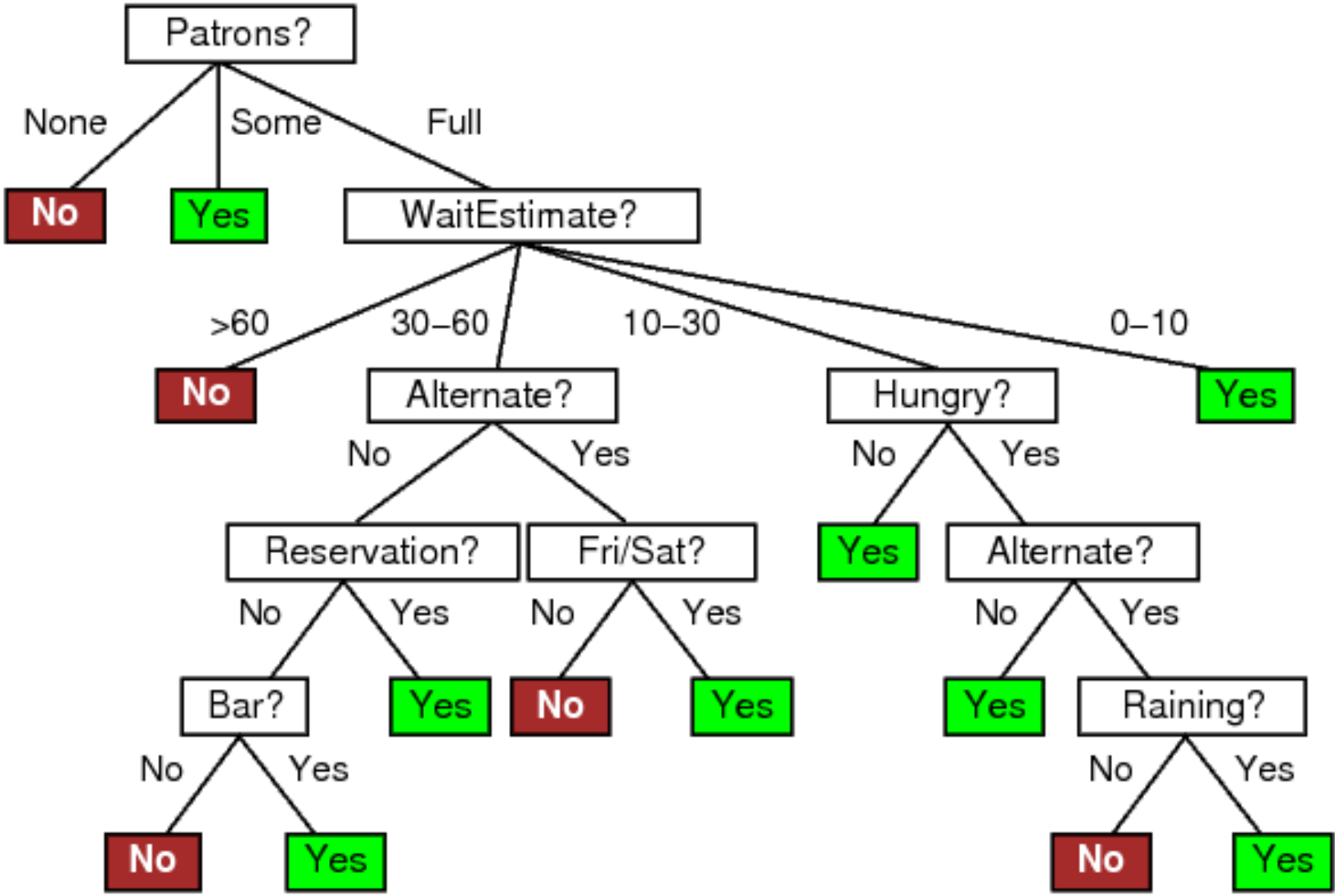
  Raining: whether it's raining outside

  Reservation: whether we made a reservation

  Type: the kind of restaurant (French, Italian, Thai, burger)

  WaitEstimate: likely time to we have to wait (0-10 mins, 10-30, 30-60, > 60)

Example Decision Tree



Training Examples for *WillWait*

Example	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est	Will Wait
1	Yes	No	No	Yes	Some	$$$	No	Yes	French	0-10	Yes
2	Yes	No	No	Yes	Full	$	No	No	Thai	30-60	No
3	No	Yes	No	No	Some	$	No	No	Burger	0-10	Yes
4	Yes	No	Yes	Yes	Full	$	Yes	No	Thai	10-30	Yes
5	Yes	No	Yes	No	Full	$$$	No	Yes	French	> 60	No
6	No	Yes	No	Yes	Some	$$	Yes	Yes	Italian	0-10	Yes
7	No	Yes	No	No	None	$	Yes	No	Burger	0-10	No
8	No	No	No	Yes	Some	$$	Yes	Yes	Thai	0-10	Yes
9	No	Yes	Yes	No	Full	$	Yes	No	Burger	> 60	No
10	Yes	Yes	Yes	Yes	Full	$$$	No	Yes	Italian	10-30	No
11	No	No	No	No	None	$	No	No	Thai	0-10	No
12	Yes	Yes	Yes	Yes	Full	$	No	No	Burger	30-60	Yes

Constructing a (Trivial) Decision Tree

- Trivial solution - construct a decision tree that has one path to a leaf for each example
- Each path tests an attribute in turn and branches as in the input example until we reach the leaf which has the classification of the example
- This just memorises the cases and does not find any pattern in the examples - can't generalise to examples not in the training set
- Aim is to find the smallest decision tree that is consistent with the examples
- Finding the *smallest* tree is intractable - however with heuristics we can find a small decision tree

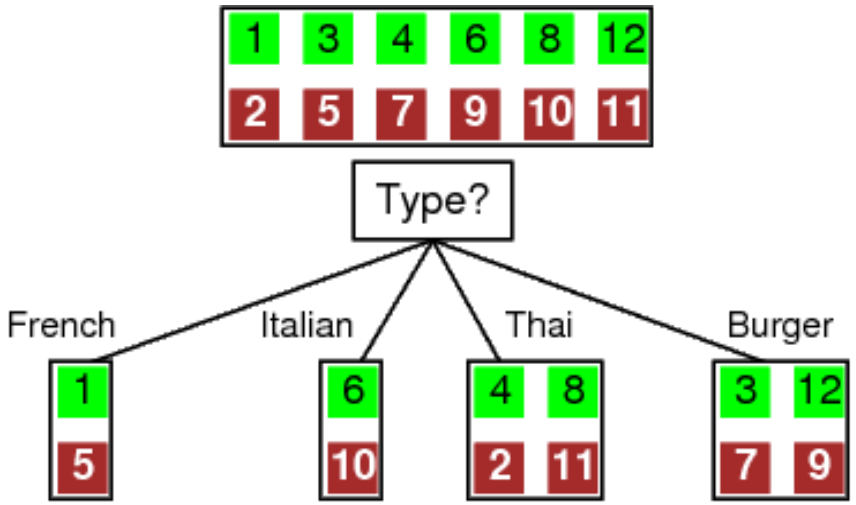
Constructing a Decision Tree

- Split the training set into positive examples (the ones for which *WillWait* is true) and negative examples (*WillWait* is false)
- Find the most important/informative attribute and make this the root test
- After the first attribute test splits the examples, each outcome is a new decision tree learning problem with fewer examples and one fewer attribute
- After splitting the examples on root attribute, choose the most informative attribute to split any remaining unclassified examples
- Repeat until all examples are classified

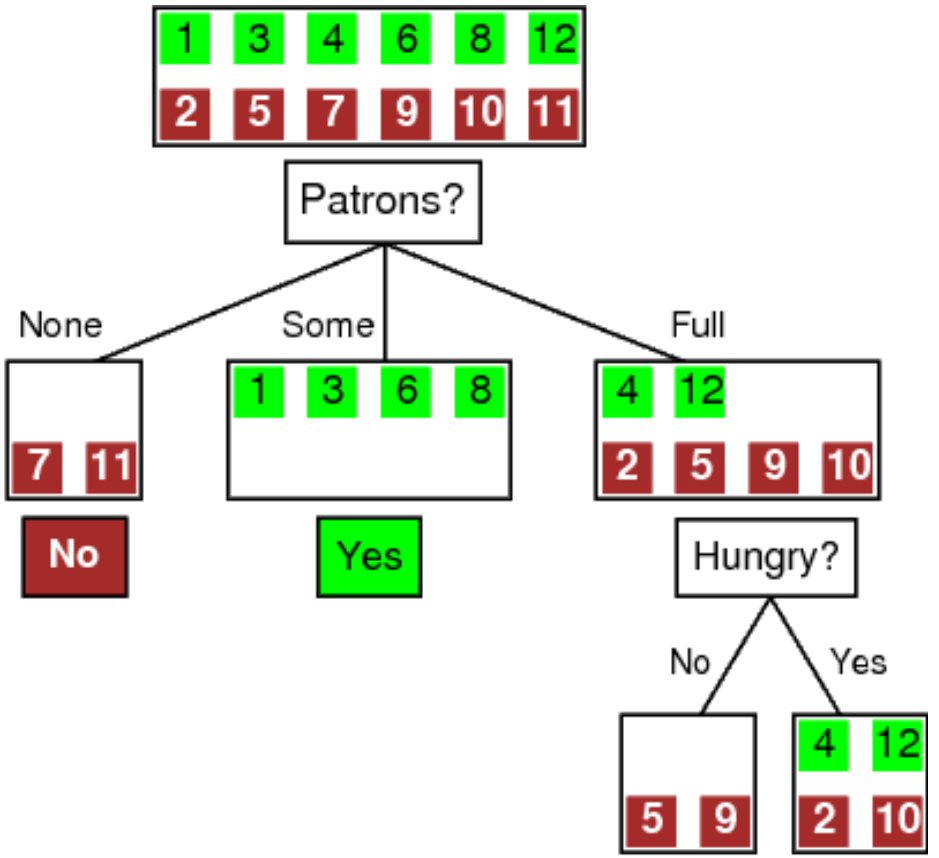
Choosing the Most Informative Attribute

- The most informative attribute is the one which makes the most difference to the classification of an example
- *Type* is an uninformative attribute as each possible outcome gives approximately the same proportion of positive and negative examples as the original set
- *Patrons* is an informative attribute: it allows us to correctly classify six examples
  - for all examples where *Patrons* = *None*, *WillWait* is false
  - for all example where *Patrons* = *Some*, *WillWait* is true
  - leaving only six examples still to classify (those where *Patrons* = *Full*)
- Split the remaining unclassified examples on the most informative attribute for those examples (*Hungry* in this case)

Choosing the Most Informative Attribute



(a)



(b)

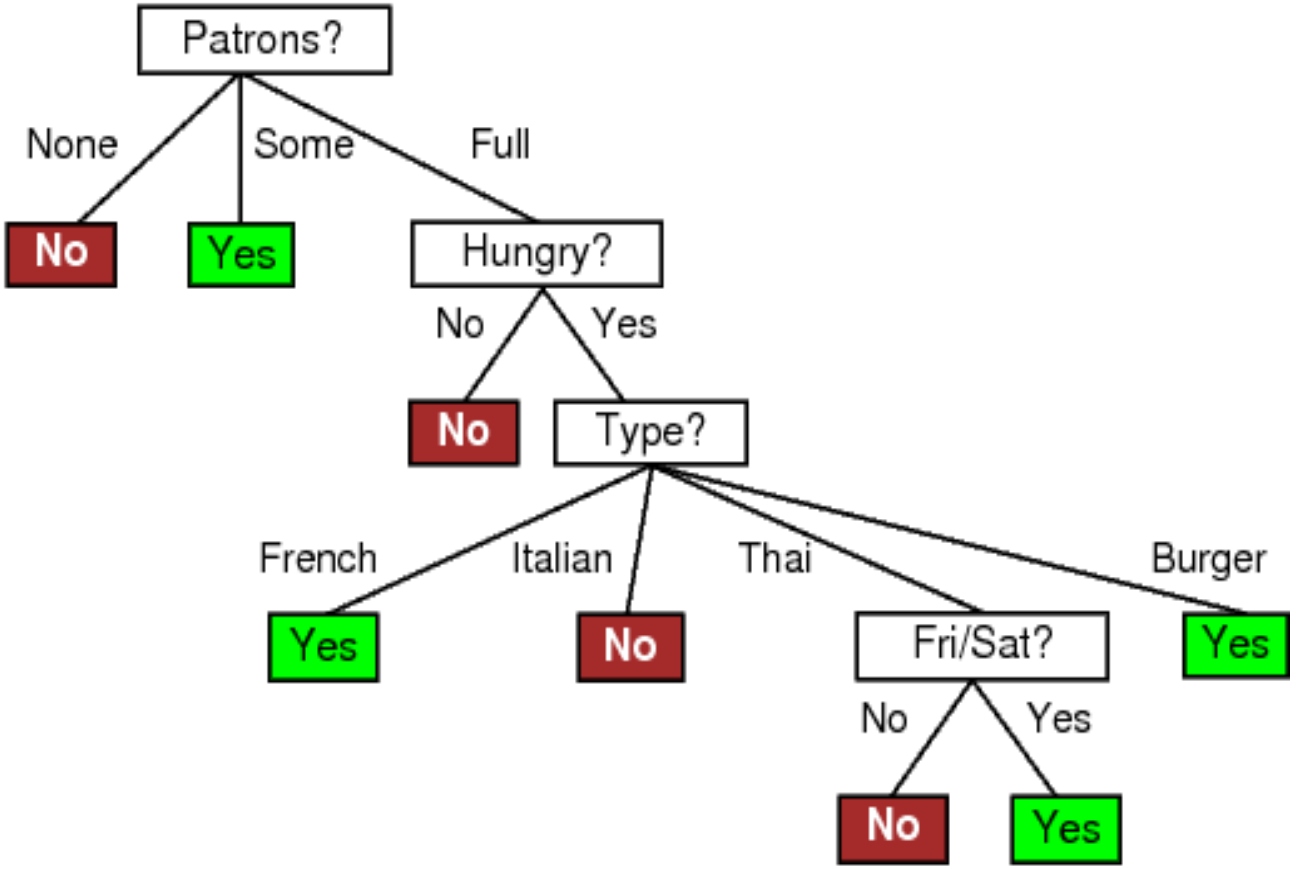
Decision Tree Learning Algorithm

- At each stage of the recursion, there are four cases to consider:
  - if all the remaining examples are positive (or negative) we are done - we can answer *Yes* or *No*
  - if there are both positive and negative examples, choose the best attribute to split them
  - if none of the examples are true or false for any of the remaining attribute values, it means that no such example has been observed and we return a default value calculated from the majority classification at the node's parent

Decision Tree Learning Algorithm

- If there are both positive and negative examples but no attributes left, we have a problem
- The examples have the same description (attribute values along the path to the current node), but different classifications
  - some of the data are incorrect (noise)
  - the attributes don't give enough information to completely describe the situation
  - the domain is nondeterministic
- One solution is to return the "majority vote": if there are more positive than negative examples remaining, make this a Yes leaf node, and vice versa

Example: Induced Tree



Example: Induced Tree

- Hypothesis is consistent and considerably simpler than the original tree used to generate the examples
- Induced tree does not contain tests for *Alternative*, *Bar*, *Price*, *Raining*, *Reservation* and *WaitEstimate* as it's possible to classify all the examples without them
- Learning algorithm looks at examples, not at the true function, so the resulting decision tree will probably make mistakes
- For example, the training set did not include a case where the wait is 0-10 minutes and the restaurant is full, and many other combinations

Informative Attributes

- We can make the notion of an "informative" attribute precise using information theory
- The amount of information required for a correct classification is given by

$$I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

where *p* is the number of positive examples in the training set and *n* is the number of negative examples

Informative Attributes

- An attribute *A* divides the training set into *v* subsets, where *v* is the number of possible values of the attribute
- If each subset *i* has p_i positive examples and n_i negative examples, after testing by attribute *A* we need an additional

$$Remainder(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right)$$

bits of information to classify the example

Informative Attributes

- The *information gain* from the attribute test *A* is the difference between the original information requirement and the new requirement

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

- At each stage of the recursion, we choose the attribute with the largest information gain

Expressiveness of Decision Trees

- Decision tree describes a relationship between the goal predicate and a logical combination of attribute values
- Any propositional boolean function can be written as a decision tree
- For many boolean functions, decision trees provide a compact representation
- For some boolean functions, a decision tree will be exponentially large, e.g.,
  - parity function (returns 1 iff an even number of its inputs are 1)
  - majority function (returns 1 iff more than half its inputs are 1)

Exercise: Spam Filtering Decision Tree

- A decision tree is to be used to classify email messages into those that are spam and those that are not
- Aim is to learn a definition of the goal predicate *Spam* - if *Spam* is true, then a message is spam
- The examples are described using the following attributes:

  Attach: the message has attachments

  Image: the message contains images

  AddressBk: the sender is in the recipient's address book

  Subject: the subject of the message which can take the values *Prize* (the recipient has won a prize); *Goods* (the message offers goods for sale); and *Other* (any other subject)
- Induce a decision tree from the following set of examples

Exercise: Spam Filtering Examples

Example	Attach	Images	AddressBk	Subject	Spam
1	Yes	No	Yes	Prize	Yes
2	No	Yes	No	Goods	Yes
3	Yes	Yes	Yes	Prize	Yes
4	No	No	No	Other	No
5	Yes	No	No	Prize	Yes
6	No	Yes	Yes	Goods	No
7	No	No	No	Goods	Yes
8	No	Yes	No	Other	No

Exercise: Spam Filtering Attributes

- Consider each attribute in turn:
- When *Attach* is true, in 3 cases a message is spam and 0 cases it is not; when *Attach* is false, in 2 cases a message is spam and in 3 cases it is not
- When *Images* is true, in 2 cases a message is spam, and 2 cases it is not; when *Images* is false, in 3 cases a message is spam and 1 case it is not
- When *AddressBk* is true, in 2 cases a message is spam and in 1 case it is not; when *AddressBk* is false, in 3 cases a message is spam and in 2 cases it is not
- *Subject*:
  - for all 3 cases where *Subject* = *Prize*, *Spam* is true
  - for all 2 cases where *Subject* = *Other*, *Spam* is false
  - leaving 3 examples still to classify
- *Subject* is the most informative attribute

Exercise: Spam Filtering Attributes

Attribute	Value	Spam	¬Spam
Attach	true	3	0
	false	2	3
Images	true	2	2
	false	3	1
AddressBk	true	2	1
	false	3	2
Subject	Prize	3	0
	Goods	2	1
	Other	0	2

Exercise: Spam Filtering Examples

Example	Attach	Images	AddressBk	Subject	Spam
1	Yes	No	Yes	Prize	Yes
2	No	Yes	No	Goods	Yes
3	Yes	Yes	Yes	Prize	Yes
4	No	No	No	Other	No
5	Yes	No	No	Prize	Yes
6	No	Yes	Yes	Goods	No
7	No	No	No	Goods	Yes
8	No	Yes	No	Other	No

Exercise: Spam Filtering Attributes

- Repeat the process for the three remaining cases, for each of the remaining attributes (*Attach*, *Images*, and *AddressBk*)
- In none of the cases is *Attach* true; when *Attach* is false, in 2 cases a message is spam and in 1 case it is not
- When *Images* is true, in 1 case a message is spam and in 1 case it is not; when *Images* is false, in 1 case a message is spam
- When *AddressBk* is true, in all cases (1) a message is not spam; when *AddressBk* is false, in all cases (1) a message is spam
- *AddressBk* is therefore the most informative remaining attribute and is sufficient to complete the classification

Exercise: Spam Filtering Attributes

Attribute	Value	Spam	¬Spam
Attach	true	0	0
	false	2	1
Images	true	1	1
	false	1	0
AddressBk	true	0	1
	false	2	0