COSC343: Artificial Intelligence

Lecture 6: Classification and Decision Trees

Lech Szymanski

Dept. of Computer Science, University of Otago

In today's lecture

- Classification
- Optimal/Naive Bayes classifiers
- Decision Trees
- Information Theory

Classification

A classification function $f(x_1, ..., x_M)$, or $f(\mathbf{x})$ where $\mathbf{x} = [x_1 \quad \dots \quad x_M]^T$, takes a set of attributes $x_1, \dots, x_M$ and returns a **class label** from the set $\{c_1, ..., c_K\}$.

- Attributes can be discrete or continuous

We're concerned with methods that learn a classification function from a set of *labelled* examples:

- That is, the training data consist of N sample inputs - each a vector of dimension M that has been pre-labelled with a label from the set of K possible labels.

Optimal Bayes Classifier

If we knew the probability distribution of the label given observations (model input), then we could make the optimal classification decision like so:

$$y = f(x) = c_k \mid \arg \max_{c_k \in c_1, \dots, c_K} p(c_k|\mathbf{x})$$

Recall that, for the purpose of the above probability comparison, we can derive the same result using Baye's rule :

$$p(c_k|\mathbf{x}) = p(\mathbf{x}|c_k)p(c_k)$$

Hence, optimal classification is given by the following rule:

$$y = f(x) = c_k \mid \arg \max_{c_k \in c_1, \dots, c_K} p(\mathbf{x}|c_k)p(c_k)$$

An example: Optimal Bayes Classifier

Joint probability distributions
over attributes given the label

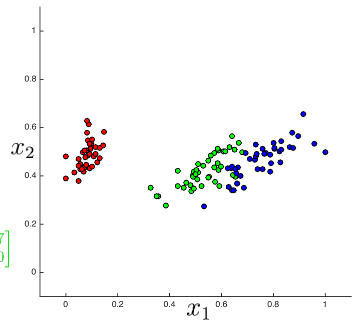
$$p(red) = p(green) = p(blue) = \frac{1}{3}$$

$$p(\mathbf{x}|red) = \frac{1}{2\pi|\Sigma_r|} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_r)\Sigma_r^{-1}(\mathbf{x}-\boldsymbol{\mu}_r)^T} \quad \boldsymbol{\mu}_r = \begin{bmatrix}0.0825\\0.4854\end{bmatrix} \quad \Sigma_r = \begin{bmatrix}0.0011 & 0.0011\\0.0011 & 0.0036\end{bmatrix}$$

$$p(\mathbf{x}|green) = \frac{1}{2\pi|\Sigma_g|} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_g)\Sigma_g^{-1}(\mathbf{x}-\boldsymbol{\mu}_g)^T} \quad \boldsymbol{\mu}_g = \begin{bmatrix}0.5261\\0.4134\end{bmatrix} \quad \Sigma_g = \begin{bmatrix}0.0069 & 0.0037\\0.0037 & 0.0040\end{bmatrix}$$

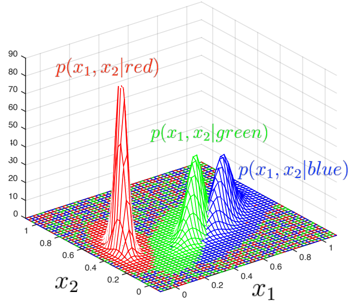
$$p(\mathbf{x}|blue) = \frac{1}{2\pi|\Sigma_b|} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_b)\Sigma_b^{-1}(\mathbf{x}-\boldsymbol{\mu}_b)^T} \quad \boldsymbol{\mu}_b = \begin{bmatrix}0.7699\\0.4741\end{bmatrix} \quad \Sigma_b = \begin{bmatrix}0.0109 & 0.0063\\0.0063 & 0.0058\end{bmatrix}$$

Training data

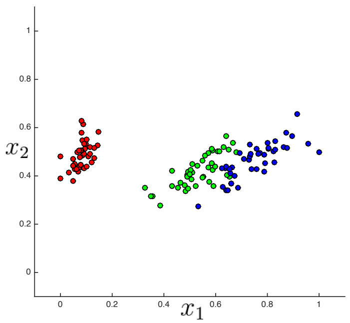


An example: Optimal Bayes Classifier

Joint probability distributions
over attributes given the label

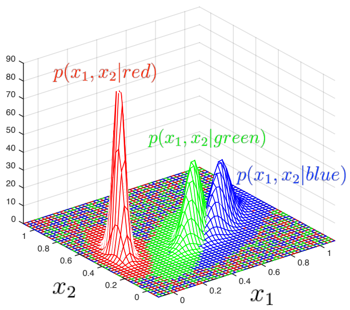


Training data

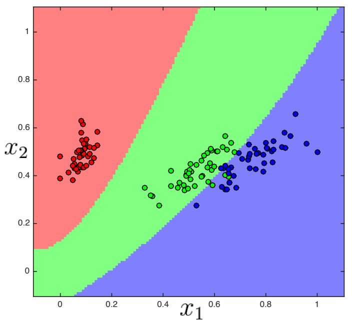


An example: Optimal Bayes Classifier

Joint probability distributions
over attributes given the label

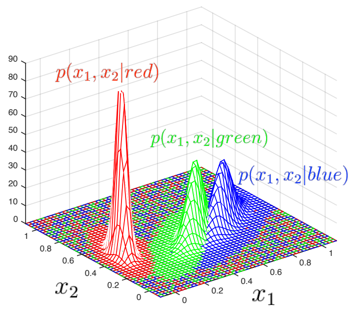


Classification and training data

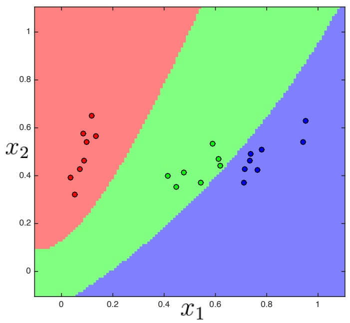


An example: Optimal Bayes Classifier

Joint probability distributions
over attributes given the label



Classification and test data



But...

- It's **very rare** that the joint probability distribution of attributes given the label is known
- It's **extremely hard** to estimate the joint probability distribution of attributes given the label, especially when there is a large number of attributes

Naive Bayes Classifier

Make a classification following the optimal rule:

$$y = f(x) = c_k \mid \arg \max_{c_k \in c_1,\ldots,c_K} p(\mathbf{x}|c_k)p(c_k) \text{ ,}$$

but simplify estimation of the probability distributions with a *naive* assumption that input attributes are independent of each other

$$y = f(x) = c_k \mid \arg \max_{c_k \in c_1,\ldots,c_K} \prod_i p(x_i|c_k)p(c_k) \text{ .}$$

An example: Naive Bayes classifier

$$p(red) = p(green) = p(blue) = \frac{1}{3}$$

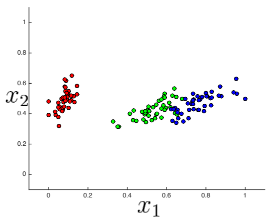
Assuming normal distribution of each attribute given the label, we need the following distributions:

$$p(x_1|red) = \mathcal{N}(\mu_{1r}, \sigma^2_{1r})$$
$$p(x_2|red) = \mathcal{N}(\mu_{2r}, \sigma^2_{2r})$$

$$p(x_1|green) = \mathcal{N}(\mu_{1g}, \sigma^2_{1g})$$
$$p(x_2|green) = \mathcal{N}(\mu_{2g}, \sigma^2_{2g})$$

$$p(x_1|blue) = \mathcal{N}(\mu_{1b}, \sigma^2_{1b})$$
$$p(x_2|blue) = \mathcal{N}(\mu_{2b}, \sigma^2_{2b})$$

Training data

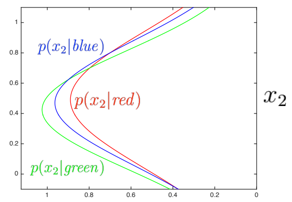


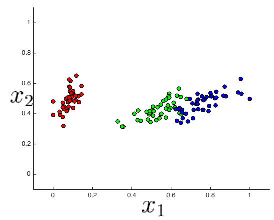
, which can be computed using estimators for univariate Gaussian mean and variance

$$\mu_i = \frac{1}{N}\sum_i x_i$$
$$\sigma_i^2 = \frac{N}{N-1}\sum_i (x_i - \mu_i)^2$$

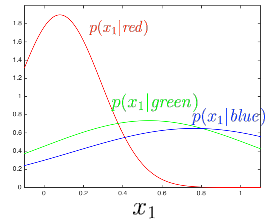
An example: Naive Bayes classifier

Training data and the classifier





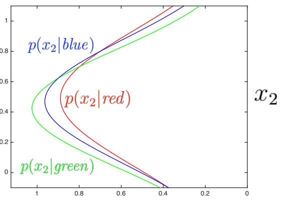
Conditional distributions over the second attribute

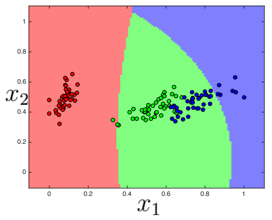


Conditional distributions over the first attribute

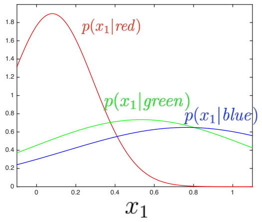
An example: Naive Bayes classifier

Classification and the training data





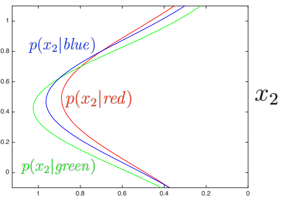
Conditional distributions
over the second attribute

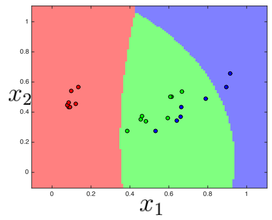


Conditional distributions
over the first attribute

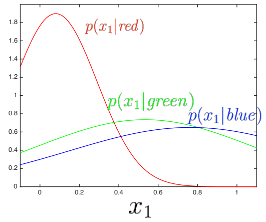
An example: Naive Bayes classifier

Classification and the test data





Conditional distributions
over the second attribute

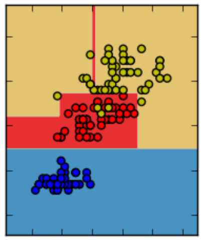


Conditional distributions
over the first attribute

Decision-tree learning

A simple classifier-learning algorithm, which involves the construction of a **decision tree**.

A decision tree makes a sequence of partitions of the training data, one attribute at a time.



An example scenario

The example dataset in the book involves learning about when Stuart Russell is prepared to wait for a table in a restaurant.

The input variables are:

- Alt (Boolean): is there an alternative restaurant nearby?
- Bar (Boolean): does the restaurant have a bar?
- Fri (Boolean): true if Friday or Saturday
- Hun (Discrete): whether he is hungry
- Pat (Discrete): how many patrons are there (*none*, *some*, *full*)
- Price (Discrete): the restaurant's price range (*$*, *$$*, *$$$*)
- Rain (Boolean): is it raining outside?
- Res (Boolean): does he have a reservation?
- Type (Discrete): *French*, *Italian*, *Thai*, *Burger*
- Est (Discrete): wait time estimation given by the host (*0-10*min, *10-30*, *30-60*, or *>60*)

We want a method for learning a hypothesis function that delivers a sensible value for *WillWait* for each possible combination of input variable values.

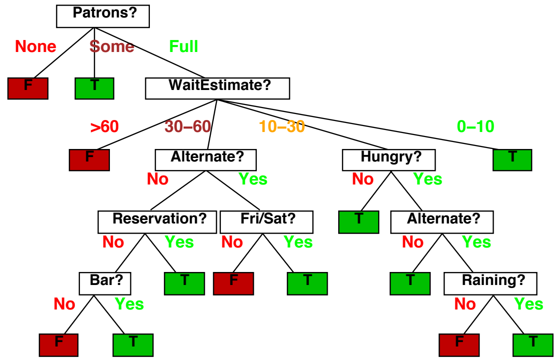
A toy training set

Item	Attributes										Class
	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est	WillWait
x_1	T	F	F	T	some	$$$	F	T	french	0-10	T
x_2	T	F	F	T	full	$	F	F	thai	30-60	F
x_3	F	T	F	F	some	$	F	F	burger	0-10	T
x_4	T	F	T	T	full	$	F	F	thai	10-30	T
x_5	T	F	T	F	full	$$$	F	T	french	>60	F
x_6	F	T	F	T	some	$$	T	T	italian	0-10	T
x_7	F	T	F	F	none	$	T	F	burger	0-10	F
x_8	F	F	F	T	some	$$	T	T	thai	0-10	T
x_9	F	T	T	F	full	$	T	F	burger	>60	F
x_{10}	T	T	T	T	full	$$$	F	T	italian	10-30	F
x_{11}	F	F	F	F	none	$	F	F	thai	0-10	F
x_{12}	T	T	T	T	full	$	F	F	burger	30-60	T

Decision trees

A decision tree is basically a big nested conditional statement.

E.g. here is the tree used to generate the training data:



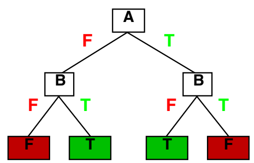
The leaves of the tree identify the values of the **class** variable *WillWait*.

The expressiveness of decision trees

Decision trees can express any function of the input attributes.

- E.g., for Boolean functions, if there are M input variables, we can trivially build a tree which has one ‘level’ for each variable:

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



So if need be, we can build a tree which has one path to a leaf node for each training example.

The need for compact decision trees

If we build a new path for each training example, we’re really just using the tree to *memorise the training examples*.

- The tree is likely to *overfit* the training examples

We should build the *simplest decision tress which is consistent with the data.*

Algorithm for building a compact decision tree

In this algorithm, we (recursively) find the input variable which does most work in separating the training examples according to their label, and create a node for this variable.

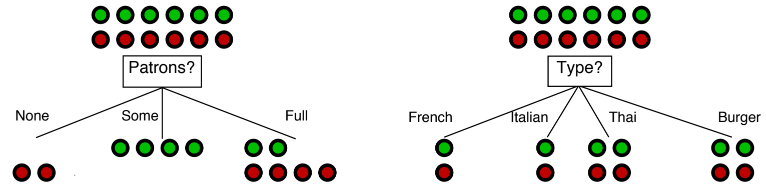
```
function DTL(examples, attributes, default) returns a decision tree

  if examples is empty then return default
  else if all examples have the same classification then return the classification
  else if attributes is empty then return MODE(examples)
  else
      best ← CHOOSE-ATTRIBUTE(attributes, examples)
      tree ← a new decision tree with root test best
      for each value v_i of best do
          examples_i ← {elements of examples with best = v_i}
          subtree ← DTL(examples_i, attributes − best, MODE(examples))
          add a branch to tree with label v_i and subtree subtree
      return tree
```


Choosing a good variable

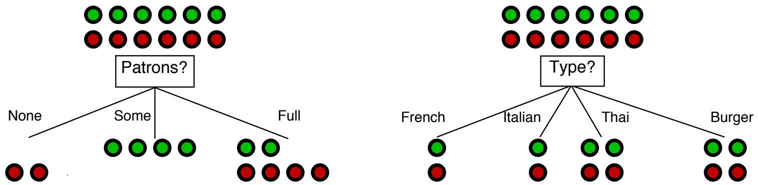
Which variable should we choose

- Ideally, we want one which allows us to predict the output variable with 100% accuracy
- But even if there isn't one, some variables are still better than others.

Say we're starting off with all 12 training examples, and trying to decide between *Patrons* and *Type*. Which variable provides more information about *WillWait*?



Choosing a good variable



Here's what we do:

- Calculate the expected **entropy*** of *WillWait* distributions conditional on values of the candidate attribute
- Entropy tells us how much uncertainty is left about data (with respect to its labels) after it's been split into subsets based on values of the attribute.
- Pick the attribute that gives a split with the lowest expected entropy

* Concept from information theory - not the same as, but somewhat related to, the concept of entropy in physics

Information theory

Entropy is the expectation of the number of bits required to encode data, given its optimal* encoding

- The more bits (higher entropy) the higher uncertainty in the data
- The fewer bits (lower entropy) the lower uncertainty in the data

$$H(X) = E\big[-log_b\big(P(X)\big)\big]$$

Discrete distribution

$$H(X) = -\sum_i p(x_i) log_b\big(p(x_i)\big)$$

Continuous distribution

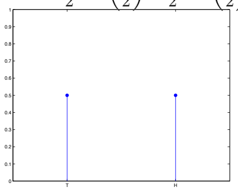
$$H(X) = -\int p(x) log_b\big(p(x)\big) dx$$

,where $0 log_b(0)$ is always taken to be 0.

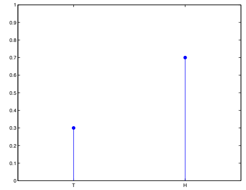
* In optimal encoding, message length is directly proportional the probability of its occurrence

Entropy as measure of uncertainty

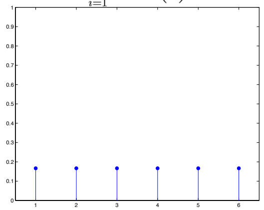
$$H(X) = -\frac{1}{2}log_2\left(\frac{1}{2}\right) - \frac{1}{2}log_2\left(\frac{1}{2}\right) = 1$$



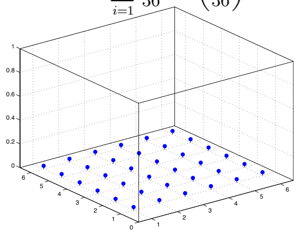
$$H(X) = -0.3log_2(0.3) - 0.7log_2(0.7) = 0.8813$$



$$H(X) = -\sum_{i=1}^{6}\frac{1}{6}log_2\left(\frac{1}{6}\right) = 2.5850$$



$$H(X) = -\sum_{i=1}^{36}\frac{1}{36}log_2\left(\frac{1}{36}\right) = 5.1699$$

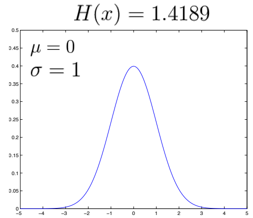


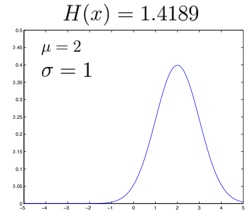
Entropy as measure of uncertainty

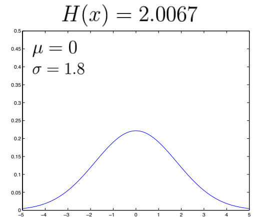
$$p(x) = \frac{1}{\sqrt{2\pi\sigma}}e^{-\frac{(x-\mu)}{2\sigma^2}}$$

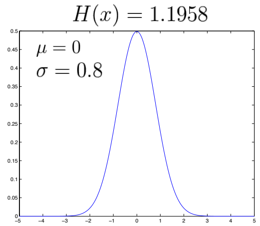
For the univariate normal distribution the entropy integral has the following closed form solution:

$$H(x) = \frac{1}{2}ln(2\pi e\sigma^2)$$









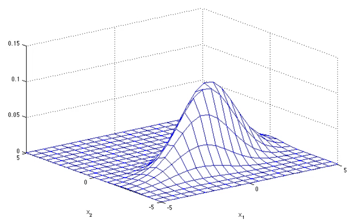
Entropy as measure of uncertainty

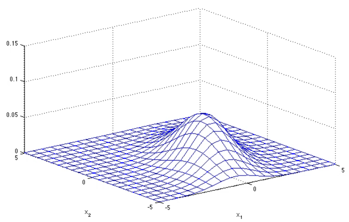
$$p(x_1, x_2) = \frac{1}{\sqrt{(2\pi)^2|\Sigma|}}e^{\left(-\frac{1}{2}([\mathbf{x}-\mu]^T\Sigma^{-1}[\mathbf{x}-\mu])\right)}$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad \mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \qquad \Sigma = \begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix}$$

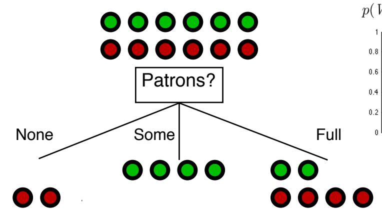
For the bivariate normal distribution the entropy integral has the following closed form solution:

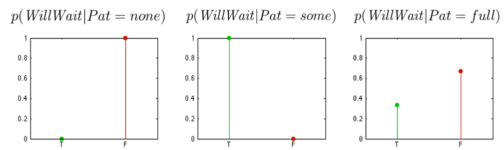
$$H(x_1, x_2) = 1 + \ln(2\pi) + \frac{1}{2}\ln|\Sigma|$$



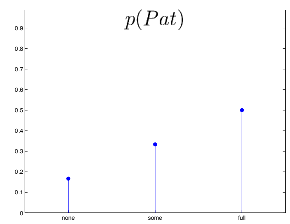


Uncertainty in a decision tree



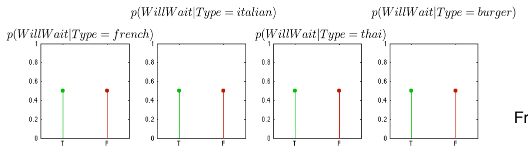


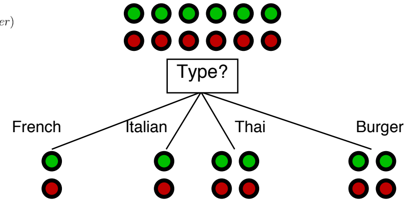
$$H(WillWait|Pat = none) = -0\log_2(0) - 1\log_2(1) = 0$$
$$H(WillWait|Pat = some) = -1\log_2(1) - 0\log_2(0) = 0$$
$$H(WillWait|Pat = full) = -\frac{1}{3}\log_2\left(\frac{1}{3}\right) - \frac{2}{3}\log_2\left(\frac{2}{3}\right) = 0.92$$

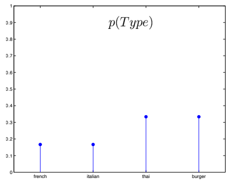


$$E[H(WillWait|Pat)] = \frac{1}{6}H(WillWait|Pat = none) + \frac{1}{3}H(WillWait|Pat = some) + \frac{1}{2}H(WillWait|Pat = full)$$
$$= 0.46$$

Uncertainty in a decision tree







$$H(WillWait|Type=french) = -\frac{1}{2}\log_2\left(\frac{1}{2}\right) - \frac{1}{2}\log_2\left(\frac{1}{2}\right) = 1$$

$$H(WillWait|Type=itailan) = -\frac{1}{2}\log_2\left(\frac{1}{2}\right) - \frac{1}{2}\log_2\left(\frac{1}{2}\right) = 1$$

$$H(WillWait|Type=thai) = -\frac{1}{2}\log_2\left(\frac{1}{2}\right) - \frac{1}{2}\log_2\left(\frac{1}{2}\right) = 1$$

$$H(WillWait|Type=burger) = -\frac{1}{2}\log_2\left(\frac{1}{2}\right) - \frac{1}{2}\log_2\left(\frac{1}{2}\right) = 1$$

$$E\big[H(WillWait|Type)\big] = \frac{1}{6}H(WillWait|Type=french) + \frac{1}{6}H(WillWait|Type=italian) + \frac{1}{3}H(WillWait|Type=thai) + \frac{1}{3}H(WillWait|Type=burger)$$

$$=1$$

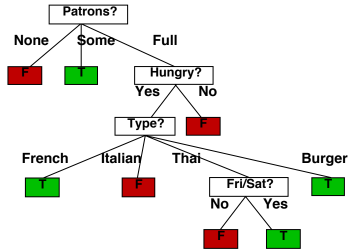
Uncertainty in a decision tree

$$H(WillWait) = 1$$

$$E\big[H(WillWait|Pat)\big] = 0.46$$

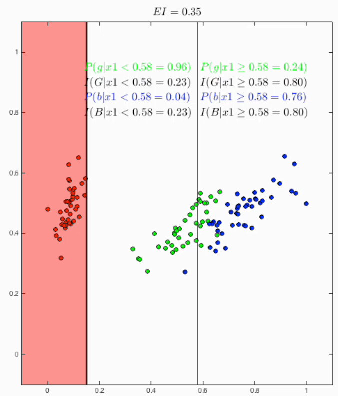
$$E\big[H(WillWait|Type)\big] = 1$$

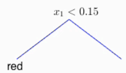
- So, learning *Type* doesn't reduce uncertainty about *WillWait* at all
- We should begin building the decision tree by creating a node for *Patrons*.
- The remainder of the decision tree learned from the 12 examples looks like this:



Note this is much simpler than the *data generating* tree! A more complex hypothesis isn't justified from the data.

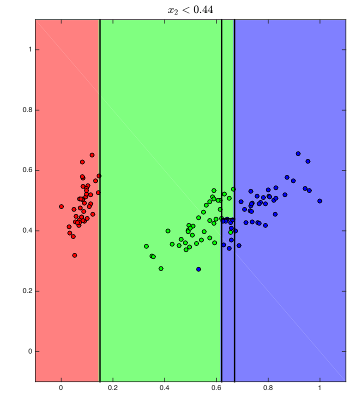
An example: Decision Tree Classifier

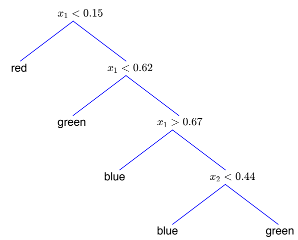




An example: Decision Tree Classifier

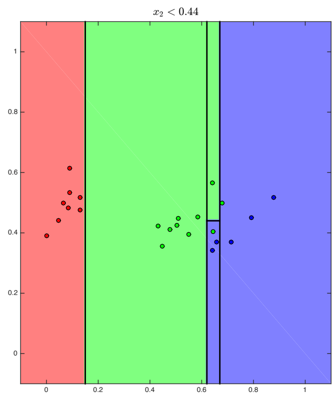
Classification on training data

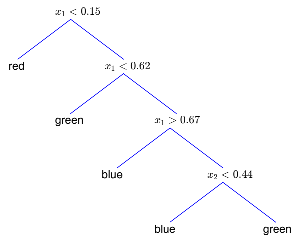




An example: Decision Tree Classifier

Classification on test data





Summary and reading

- Classification = model output is discrete
- Optimal Bayes Classifier – when probability distributions are known
- Naive Bayes Classifier – when attribute independence assumption is reasonable
- Decision Tree Classifier – categorisation based on attributes and reduction of uncertainty

Reading for the lecture: AIMA Chapter 18 Sections 3

Reading for next lecture: AIMA Chapter 18 Section 6