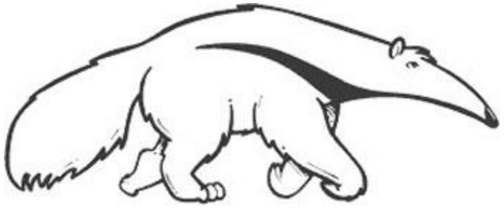
+

Machine Learning and Data Mining

Decision Trees

Kalev Kask



BREN:ICS
INFORMATION AND COMPUTER SCIENCES

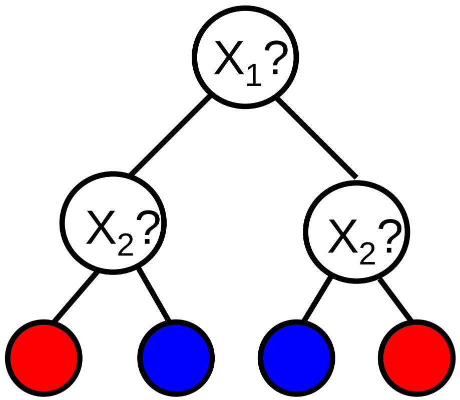
UNIVERSITY of CALIFORNIA IRVINE

Decision trees

- Functional form f(x;µ): nested "if-then-else" statements
  - Discrete features: fully expressive (any function)
- Structure:
  - Internal nodes: check feature, branch on value
  - Leaf nodes: output prediction

"XOR"

x_1	x_2	y
0	0	1
0	1	-1
1	0	-1
1	1	1

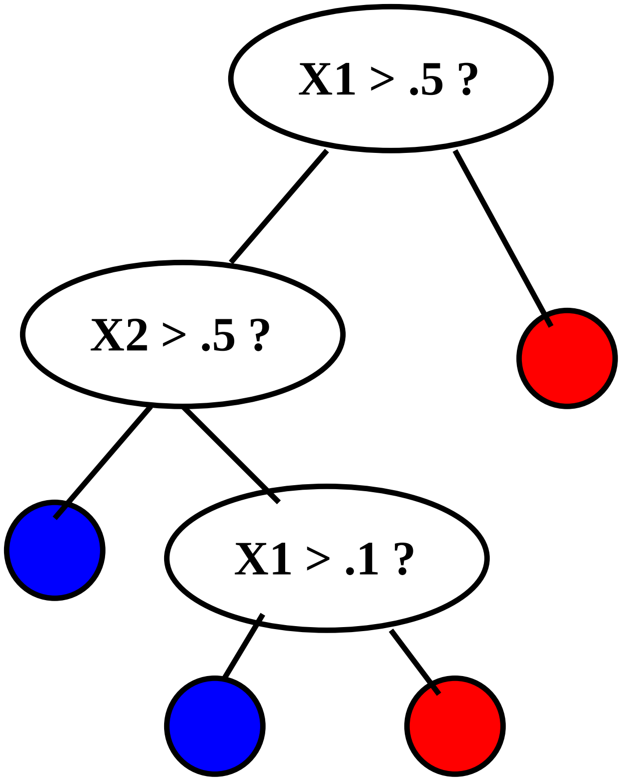


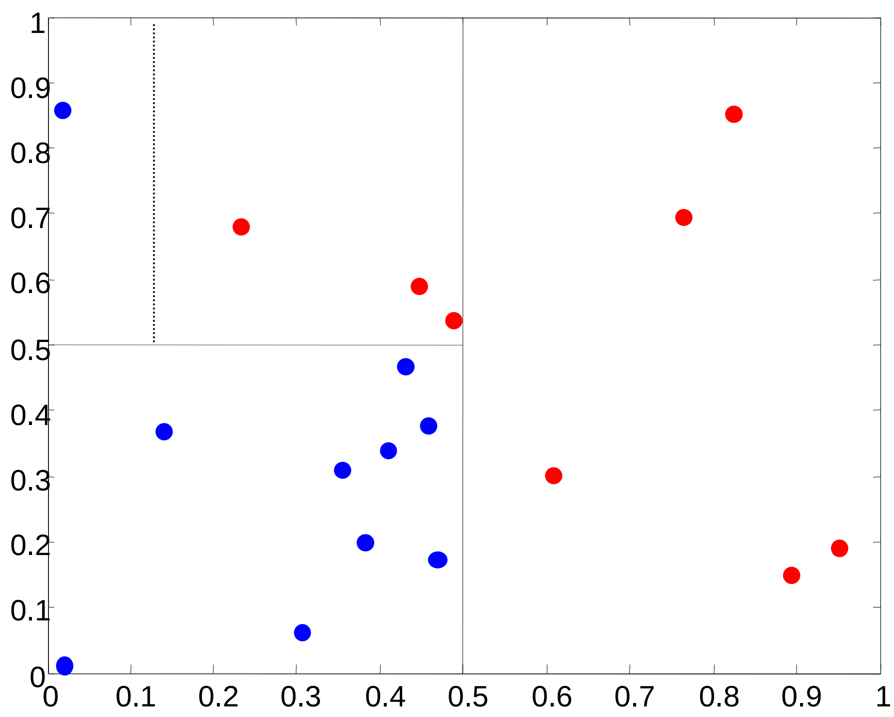

```
if X1:                      # branch on feature at root
    if X2:  return +1       # if true, branch on right child feature
    else:   return -1       # & return leaf value
else:  # left branch:
    if X2:  return -1       # branch on left child feature
    else:   return +1       # & return leaf value
```

Parameters?
Tree structure, features, and leaf outputs

Decision trees

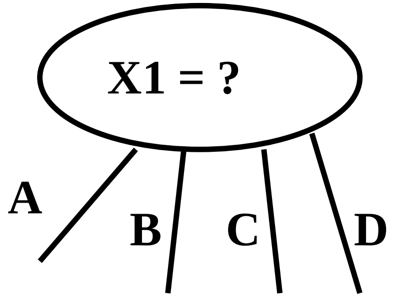
- Real-valued features
  - Compare feature value to some threshold

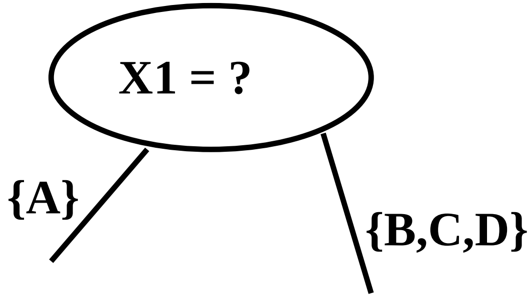


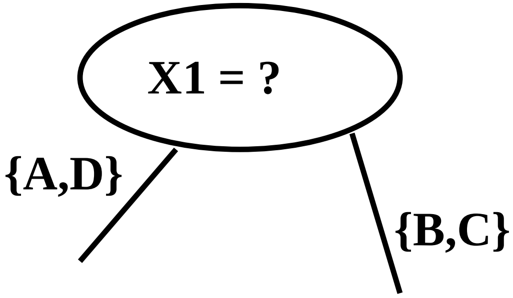


Decision trees

- Categorical variables
  - Could have one child per value
  - Binary splits: single values, or by subsets







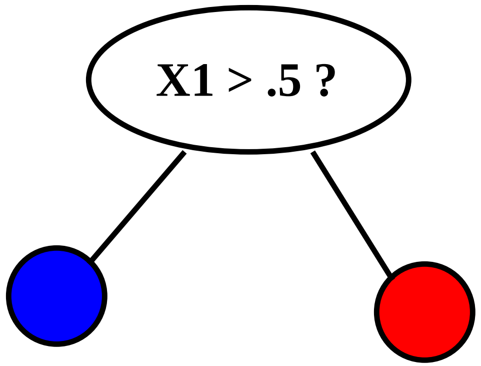
The discrete variable will not appear again below here…

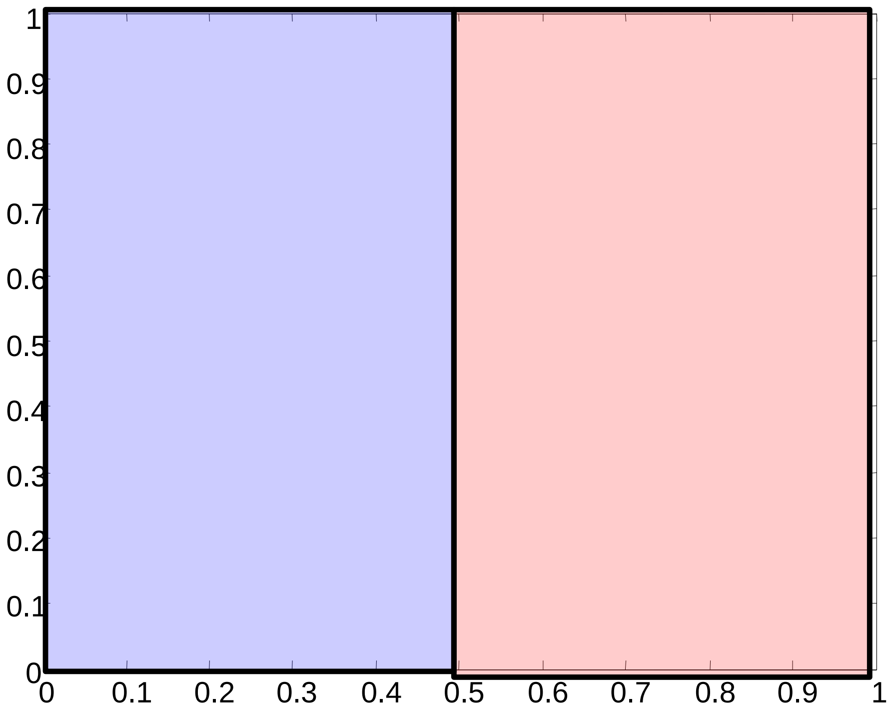
Could appear again multiple times…

(This ^^^ is easy to implement using a 1-of-K representation…)

Decision trees

- "Complexity" of function depends on the depth
- A depth-1 decision tree is called a decision "stump"
  - Simpler than a linear classifier!

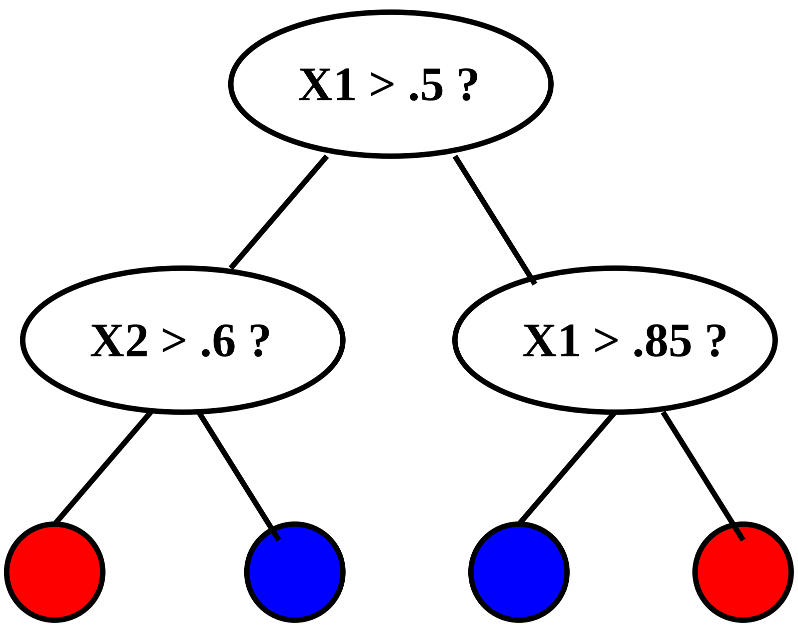


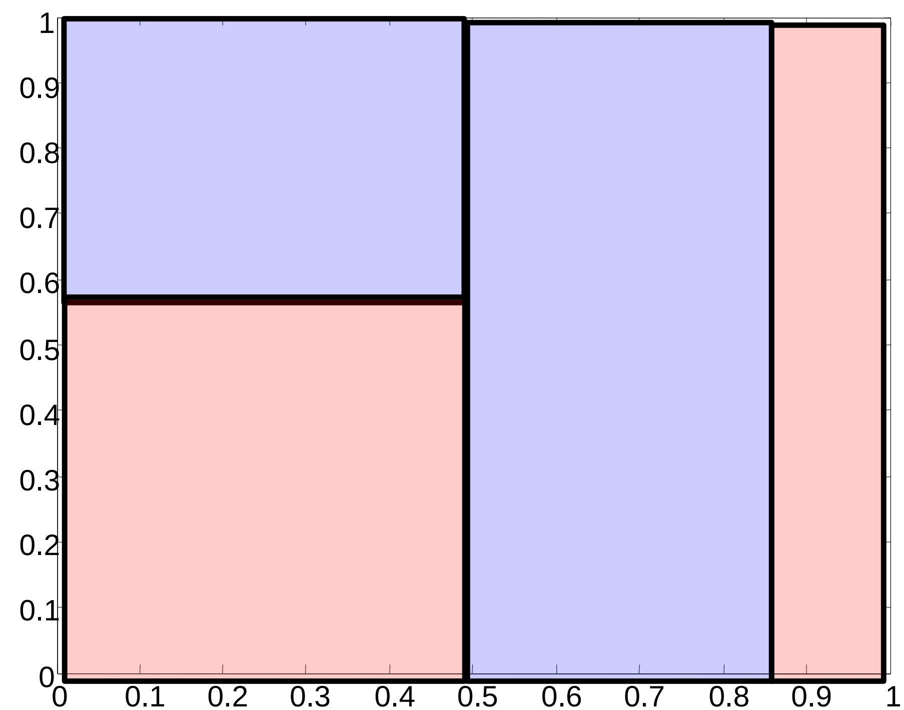


Decision trees

- "Complexity" of function depends on the depth

- More splits provide a finer-grained partitioning





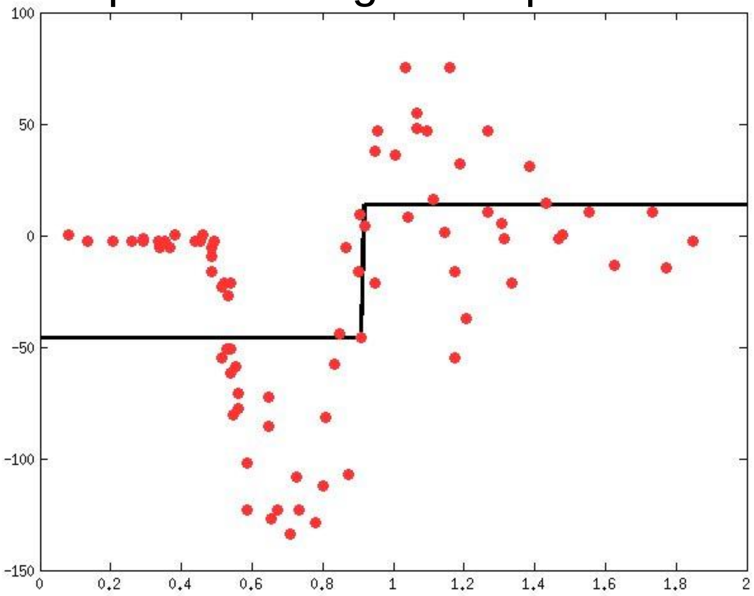
Depth d = up to 2^d regions & predictions

Decision trees for regression

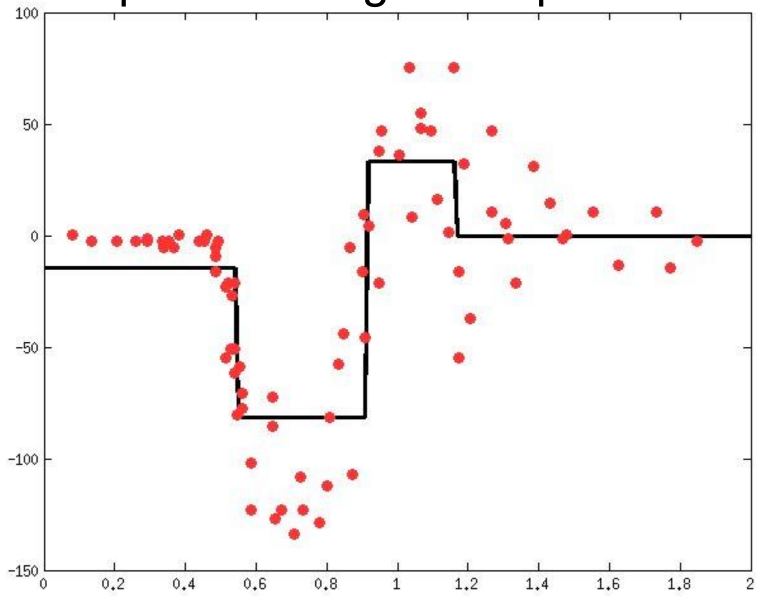
- Exactly the same
- Predict real valued numbers at leaf nodes

- Examples on a single scalar feature:

Depth 1 = 2 regions & predictions



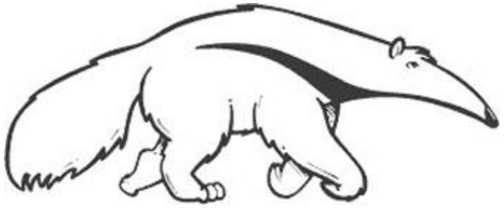
Depth 2 = 4 regions & predictions …



Machine Learning and Data Mining

Learning Decision Trees

Kalev Kask



Learning decision trees

- Break into two parts
  - Should this be a leaf node?
  - If so: what should we predict?
  - If not: how should we further split the data?

Example algorithms: ID3, C4.5
See e.g. wikipedia, "Classification and regression tree"

- Leaf nodes: best prediction given this data subset
  - Classify: pick majority class; Regress: predict average value
- Non-leaf nodes: pick a feature and a split
  - Greedy: "score" all possible features and splits
  - Score function measures "purity" of data after split
    - How much easier is our prediction task after we divide the data?
- When to make a leaf node?
  - All training examples the same class (correct), or indistinguishable
  - Fixed depth (fixed complexity decision boundary)
  - Others …

Learning decision trees

Algorithm 1 BuildTree(D): Greedy training of a decision tree

Input: A data set $D = (X, Y)$.

Output: A decision tree.

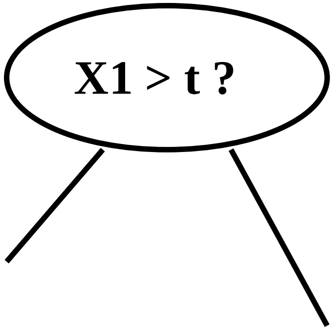
if LeafCondition(D) **then**
  $f_n = \text{FindBestPrediction}(D)$
else
  $j_n, t_n = \text{FindBestSplit}(D)$

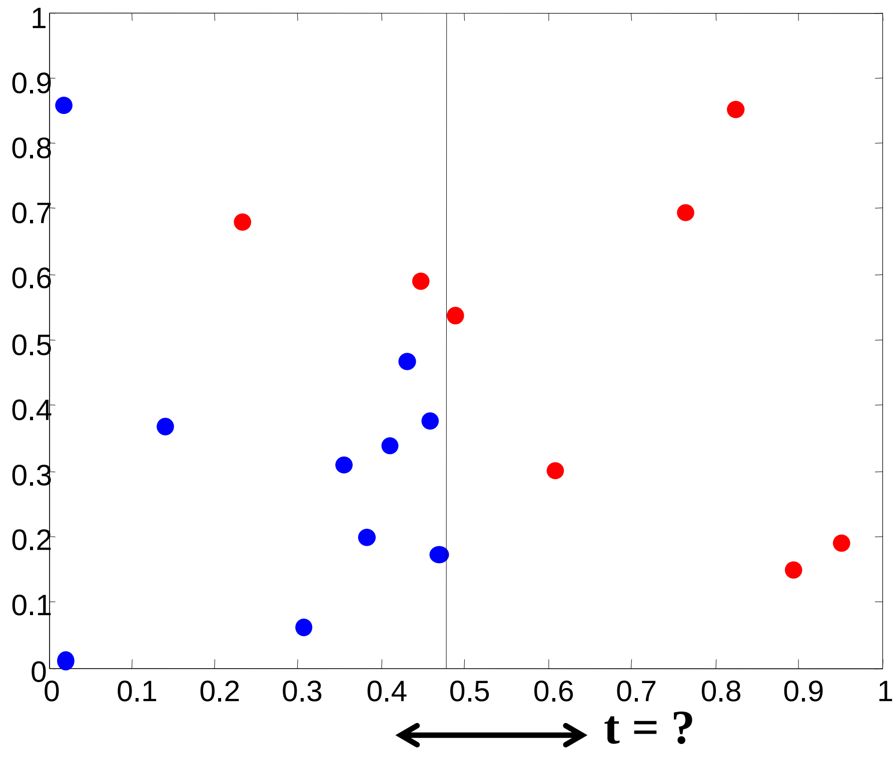
  $D_L = \{(x^{(i)}, y^{(i)}) \,:\, x^{(i)}_{j_n} < t_n\}$ and
  $D_R = \{(x^{(i)}, y^{(i)}) \,:\, x^{(i)}_{j_n} \geq t_n\}$

  leftChild = BuildTree(D_L)
  rightChild = BuildTree(D_R)
end if

Scoring decision tree splits

- Suppose we are considering splitting feature 1
  - How can we score any particular split?
  - “Impurity” – how easy is the prediction problem in the leaves?
- “Greedy” – could choose split with the best accuracy
  - Assume we have to predict a value next
  - MSE (regression)
  - 0/1 loss (classification)
- But: “soft” score can work better





Entropy and information

- "Entropy" is a measure of randomness
  - How hard is it to communicate a result to you?
  - Depends on the probability of the outcomes

- Communicating fair coin tosses
  - Output: H H T H T T T H H H H T …
  - Sequence takes n bits – each outcome totally unpredictable

- Communicating my daily lottery results
  - Output: 0 0 0 0 0 0 …
  - Most likely to take one bit – I lost every day.
  - Small chance I'll have to send more bits (won & when)

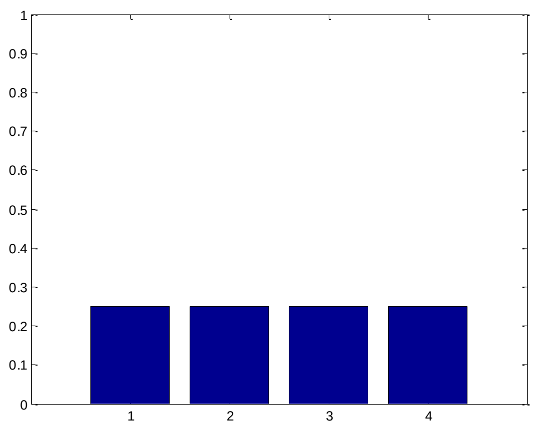
Lost: 0
Won 1: 1(…)0
Won 2: 1(…)1(…)0

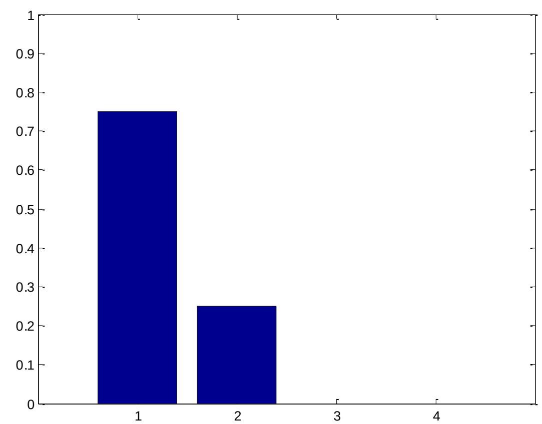
- Takes less work to communicate because it's less random
  - Use a few bits for the most likely outcome, more for less likely ones

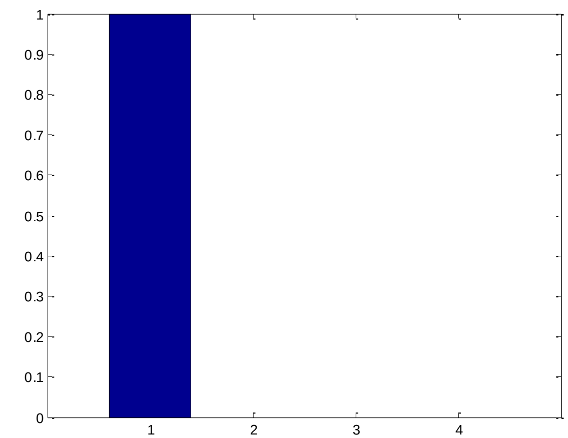
Entropy and information

- Entropy H(x) ´ E[log 1/p(x)] = ∑ p(x) log 1/p(x)
  - Log base two, units of entropy are "bits"
  - Two outcomes: H = - p log(p) - (1-p) log(1-p)

- Examples:







H(x) = .25 log 4 + .25 log 4 + .25 log 4 + .25 log 4 = log 4 = 2 bits

H(x) = .75 log 4/3 + .25 log 4 ¼ .8133 bits

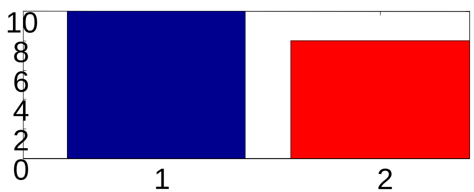
H(x) = 1 log 1 = 0 bits

Max entropy for 4 outcomes

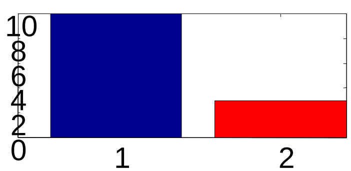
Min entropy

Entropy and information

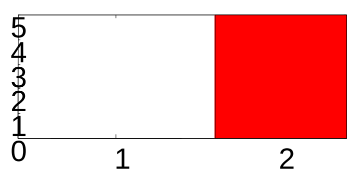
- Information gain
  - How much is entropy reduced by measurement?
- Information: expected information gain



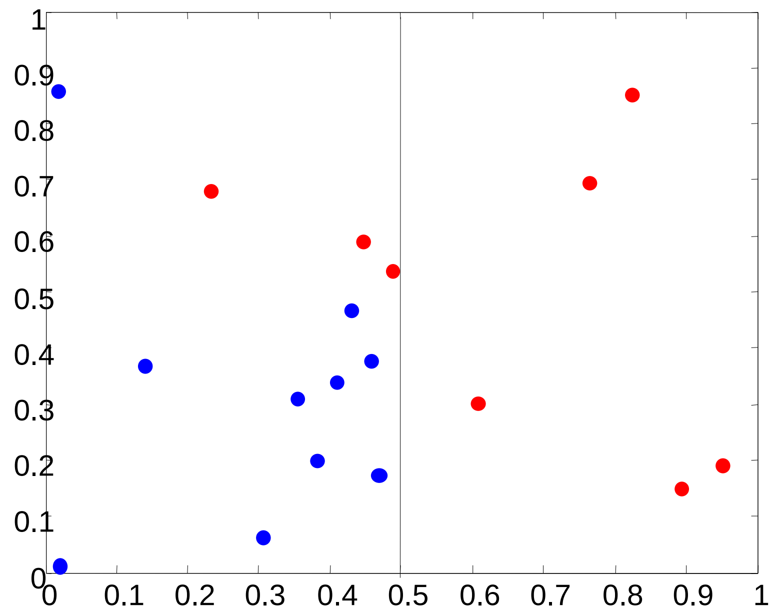
H = . 99 bits



H = .77 bits
Prob = 13/18



H=0
Prob = 5/18

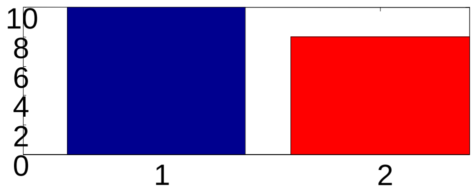


Information = 13/18 * (.99-.77) + 5/18 * (.99 – 0)

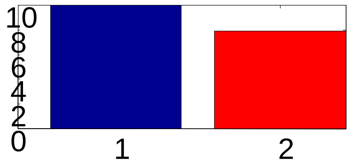
Equivalent: $\sum$ p(s,c) log [p(s,c) / p(s) p(c)]
= 10/18 log[(10/18) / (13/18) (10/18)] + 3/18 log[(3/18)/(13/18)(8/18) + …

Entropy and information

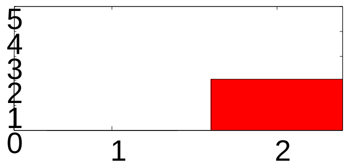
- Information gain
  - How much is entropy reduced by measurement?
- Information: expected information gain



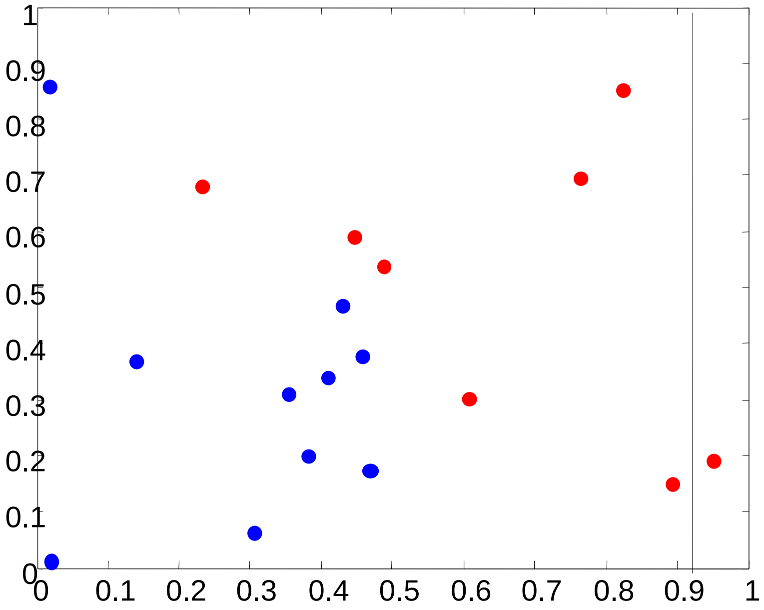
H = . 99 bits



H = .97 bits
Prob = 17/18



H=0
Prob = 1/18

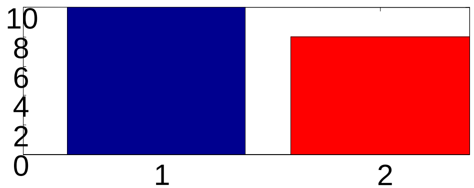


Information = 17/18 * (.99-.97) + 1/18 * (.99 – 0)

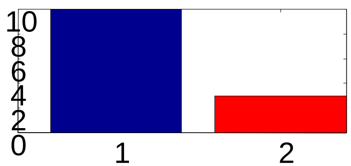
Less information reduction – a less desirable split of the data

Gini index & impurity

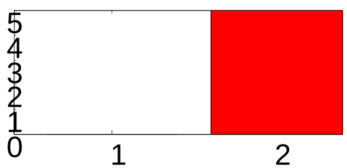
- An alternative to information gain
  - Measures variance in the allocation (instead of entropy)
- $H_{gini} = \sum_c p(c)\,(1-p(c))$ vs. $H_{ent} = -\sum_c p(c) \log p(c)$



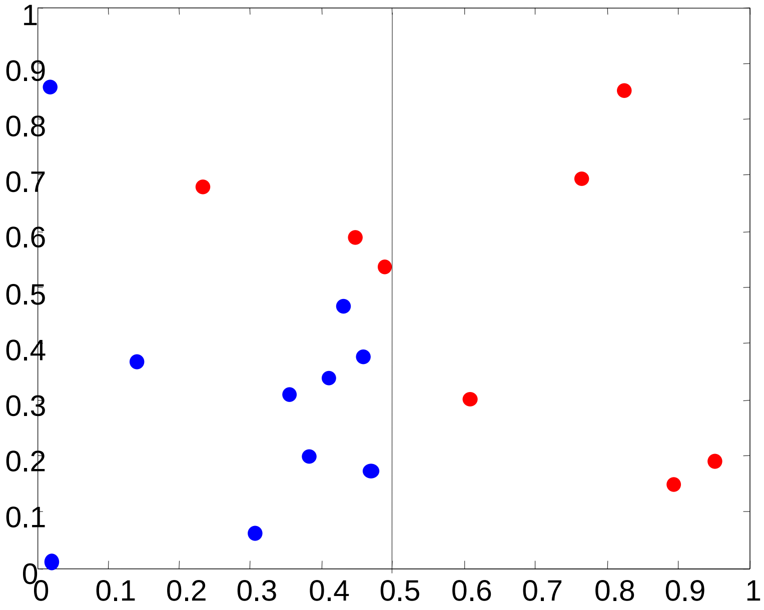
Hg = . 494



Hg = .355
Prob = 13/18



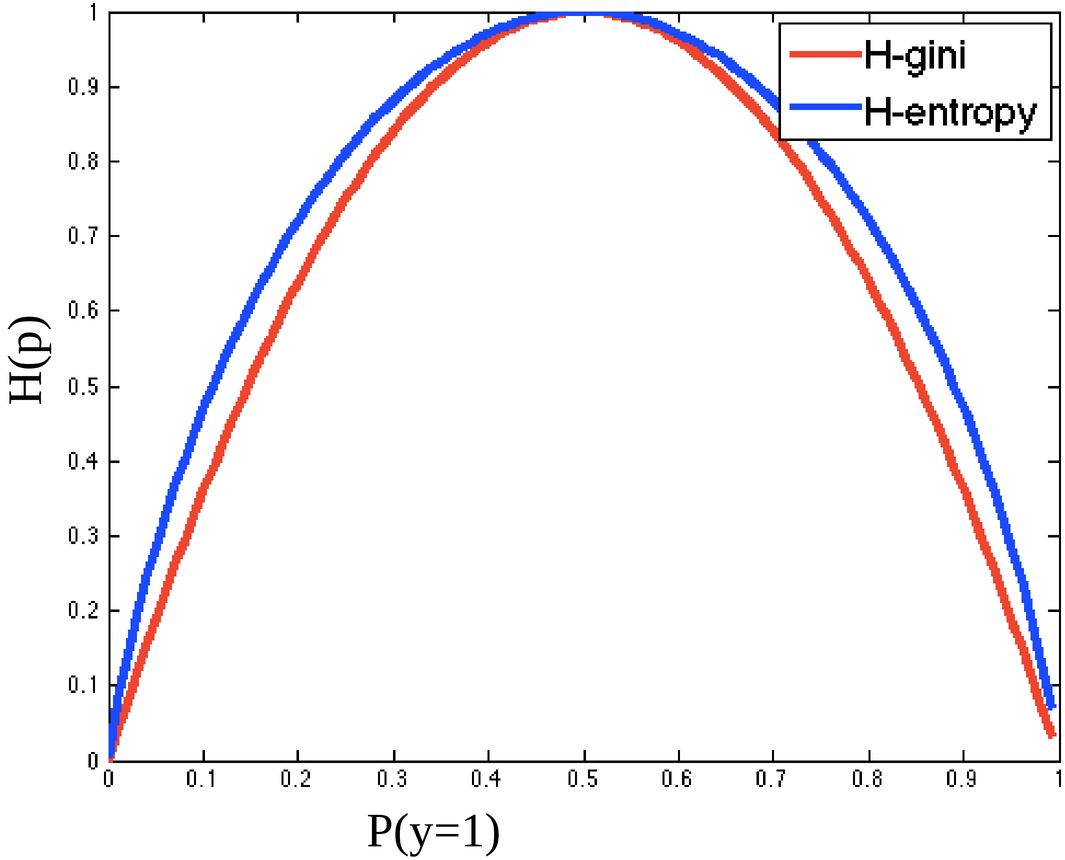
Hg = 0
Prob = 5/18



Gini Index = 13/18 * (.494-.355) + 5/18 * (.494 – 0)

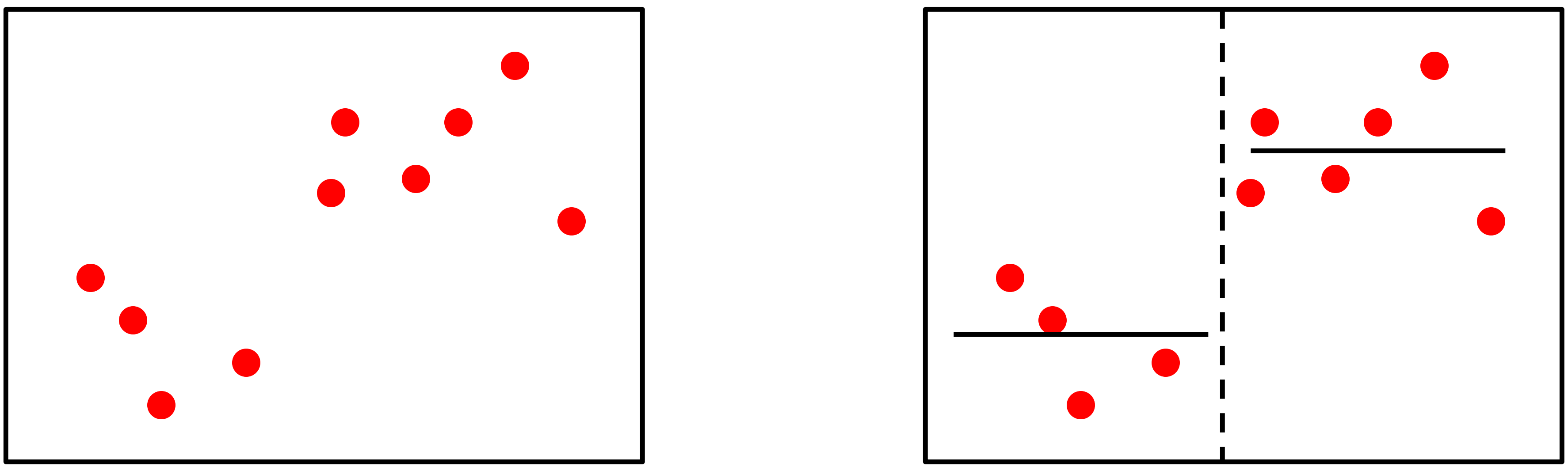
Entropy vs Gini impurity

- The two are nearly the same…
  - Pick whichever one you like



For regression

- Most common is to measure variance reduction
  - Equivalent to “information gain” in a Gaussian model…



Var = .25

Var = .1
Prob = 4/10

Var = .2
Prob = 6/10

Var reduction = 4/10 * (.25-.1) + 6/10 * (.25 – .2)

Scoring decision tree splits

Algorithm 1 FindBestSplit(D)

Input: A data set $D = (X, Y)$ of size m;
impurity function $H(\cdot)$.

Output: A split j^*, t^* minimizing impurity H

Initialize $H^* = 0$
for each feature j **do**
  Sort $\{x_j^{(i)}\}$ in order of increasing value
  for each i such that $x^{(i)} < x^{(i+1)}$ **do**
    Compute $p_c^L = \frac{1}{i} \sum_{k \leq i} \mathbb{1}[y^{(k)} = c]$
      and $p_c^R = \frac{1}{k-i} \sum_{k > i} \mathbb{1}[y^{(k)} = c]$
    Set $H' = \frac{i}{m} H(p^L) + \frac{m-i}{m} H(p^R)$
    if $H' < H^*$ **then**
      Set $j^* = j$, $t^* = (x^{(i)} - x^{(i+1)})/2$, $H^* = H'$
    end if
  end for
end for
Return j^*, t^*

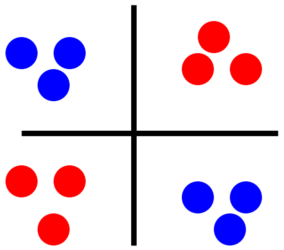
Building a decision tree

Algorithm 1 BuildTree(D): Greedy training of a decision tree

Input: A data set $D = (X, Y)$.

Output: A decision tree.

if LeafCondition(D) **then**
  f_n = FindBestPrediction(D)
else
  j_n, t_n = FindBestSplit(D)
  $D_L = \{(x^{(i)}, y^{(i)}) : x^{(i)}_{j_n} < t_n\}$ and
  $D_R = \{(x^{(i)}, y^{(i)}) : x^{(i)}_{j_n} \geq t_n\}$
  leftChild = BuildTree(D_L)
  rightChild = BuildTree(D_R)
end if



Stopping conditions:
* # of data < K
* Depth > D
* All data indistinguishable (discrete features)
* Prediction sufficiently accurate

* Information gain threshold?
  Often not a good idea!
  No single split improves,
    but, two splits do.
  Better: build full tree, then prune

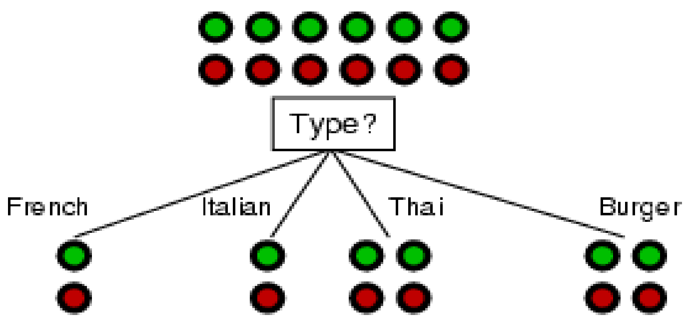
Example

[Russell & Norvig 2010]

- Restaurant data:

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
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

- Split on:



Root entropy: 0.5 * log(2) + 0.5 * log(2) = 1 bit

Leaf entropies: 2/12 * 1 + 2/12 * 1 + … = 1 bit

No reduction!

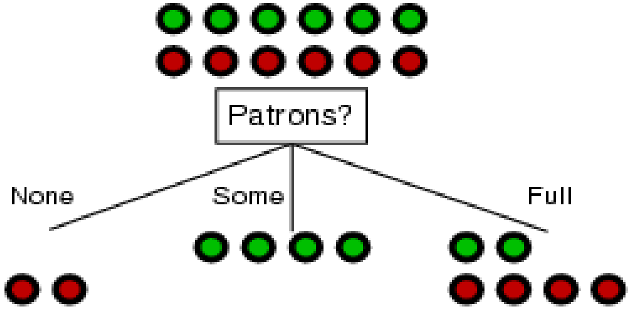
Example

[Russell & Norvig 2010]

- Restaurant data:

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
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

- Split on:



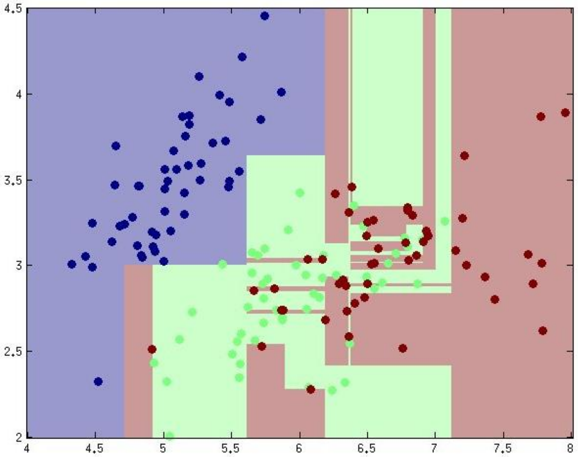
Root entropy: 0.5 * log(2) + 0.5 * log(2) = 1 bit

Leaf entropies: 2/12 * 0 + 4/12 * 0 + 6/12 * 0.9

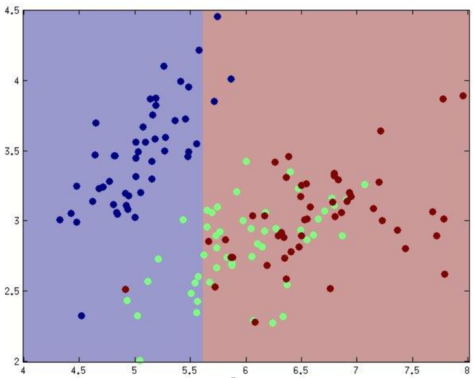
Lower entropy after split!

Controlling complexity

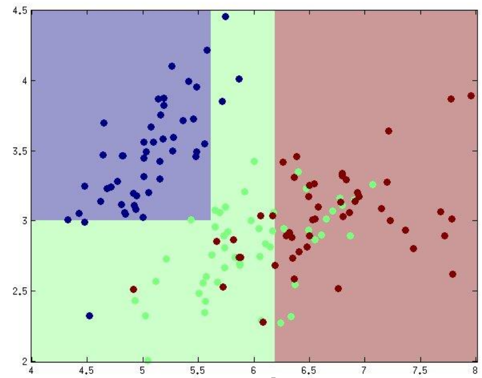
- Maximum depth cutoff



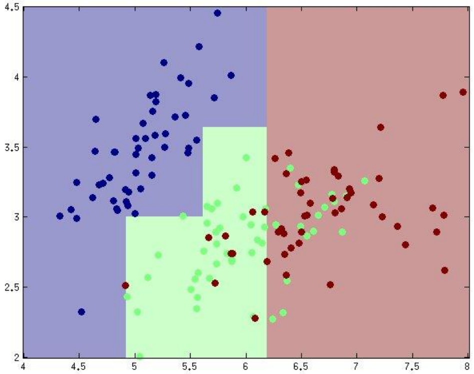
No limit



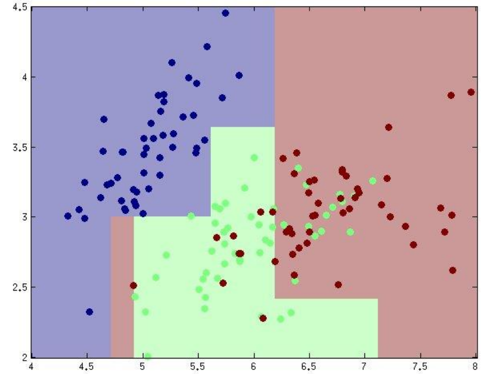
Depth 1



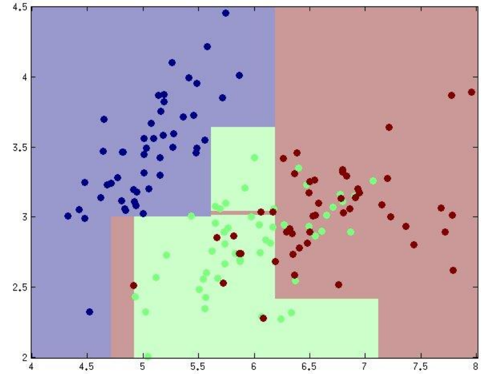
Depth 2



Depth 3



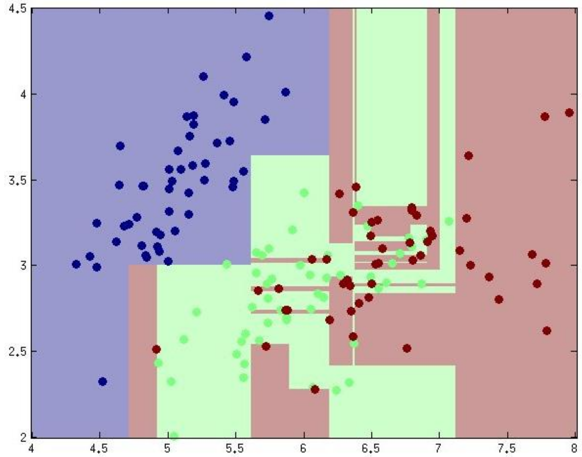
Depth 4



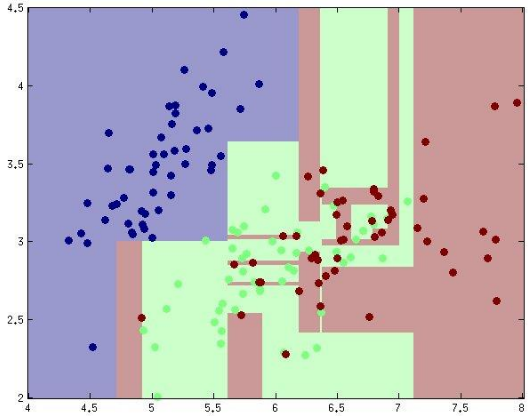
Depth 5

Controlling complexity

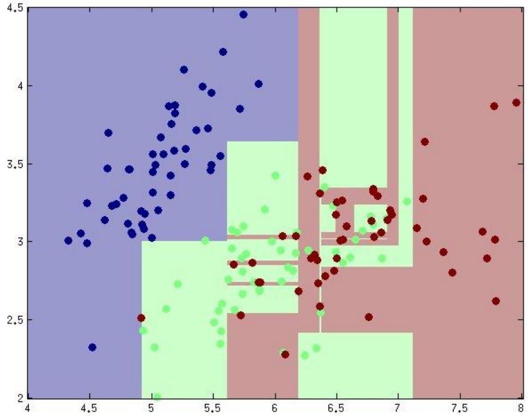
- Minimum # parent data



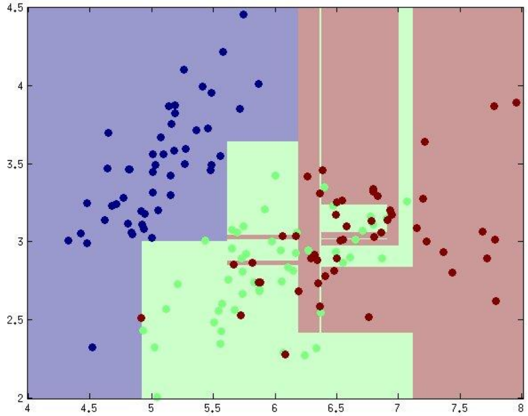
minParent 1



minParent 3



minParent 5



minParent 10

- Alternate (similar): min # of data per leaf

Computational complexity

- "FindBestSplit": on M' data
  - Try each feature: N features
  - Sort data: O(M' log M')
  - Try each split: update p, find H(p): O(M * C)
  - Total: O(N M' log M')
- "BuildTree":
  - Root has M data points: O(N M log M)
  - Next level has M *total* data points:

    $O(N\ M_L \log M_L) + O(N\ M_R \log M_R) < O(N\ M \log M)$
  - …

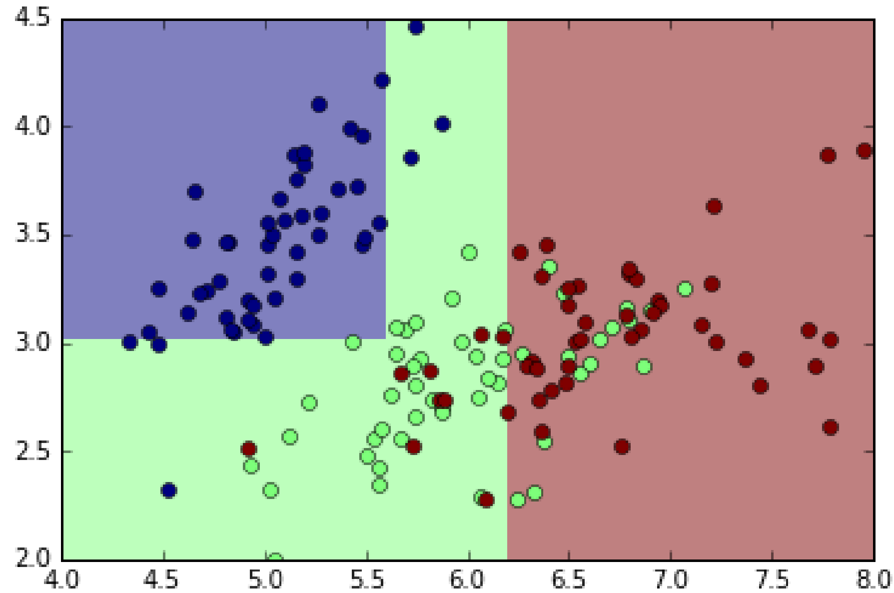
Decision trees in python

- Many implementations
- Class implementation:
  - real-valued features (can use 1-of-k for discrete)
  - Uses entropy (easy to extend)

```
T = dt.treeClassify()
T.train(X,Y,maxDepth=2)
print T

  if x[0] < 5.602476:
    if x[1] < 3.009747:
      Predict 1.0          # green
    else:
      Predict 0.0          # blue
  else:
    if x[0] < 6.186588:
      Predict 1.0          # green
    else:
      Predict 2.0          # red

ml.plotClassify2D(T, X,Y)
```



Summary

- Decision trees
  - Flexible functional form
  - At each level, pick a variable and split condition
  - At leaves, predict a value

- Learning decision trees
  - Score all splits & pick best
    - Classification: Information gain, Gini index
    - Regression: Expected variance reduction
  - Stopping criteria

- Complexity depends on depth
  - Decision stumps: very simple classifiers