Intro to Machine Learning (CS436/CS580L)

Lecture 13: Decision Trees

Xi Peng, Fall 2018

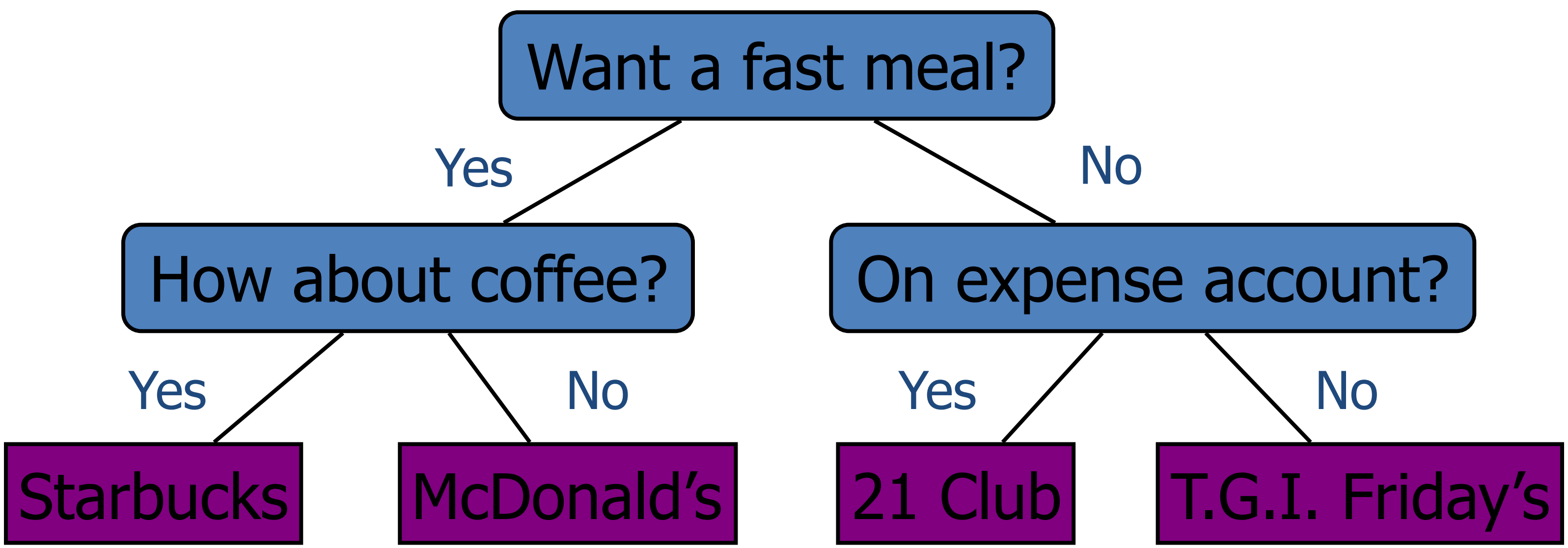
Thanks to Tom Mitchell, Andrew Ng, Ben Taskar, Carlos Guestrin, Eric Xing, Hal Daume III, David Sontag, Jerry Zhu, Tina Eliassi-Rad, and Chao Chen for some slides & teaching material.

Today

- Decision Trees
  - Entropy and Information Theory
  - Information Gain
  - “Purity” or “Classification Accuracy”

Decision Trees

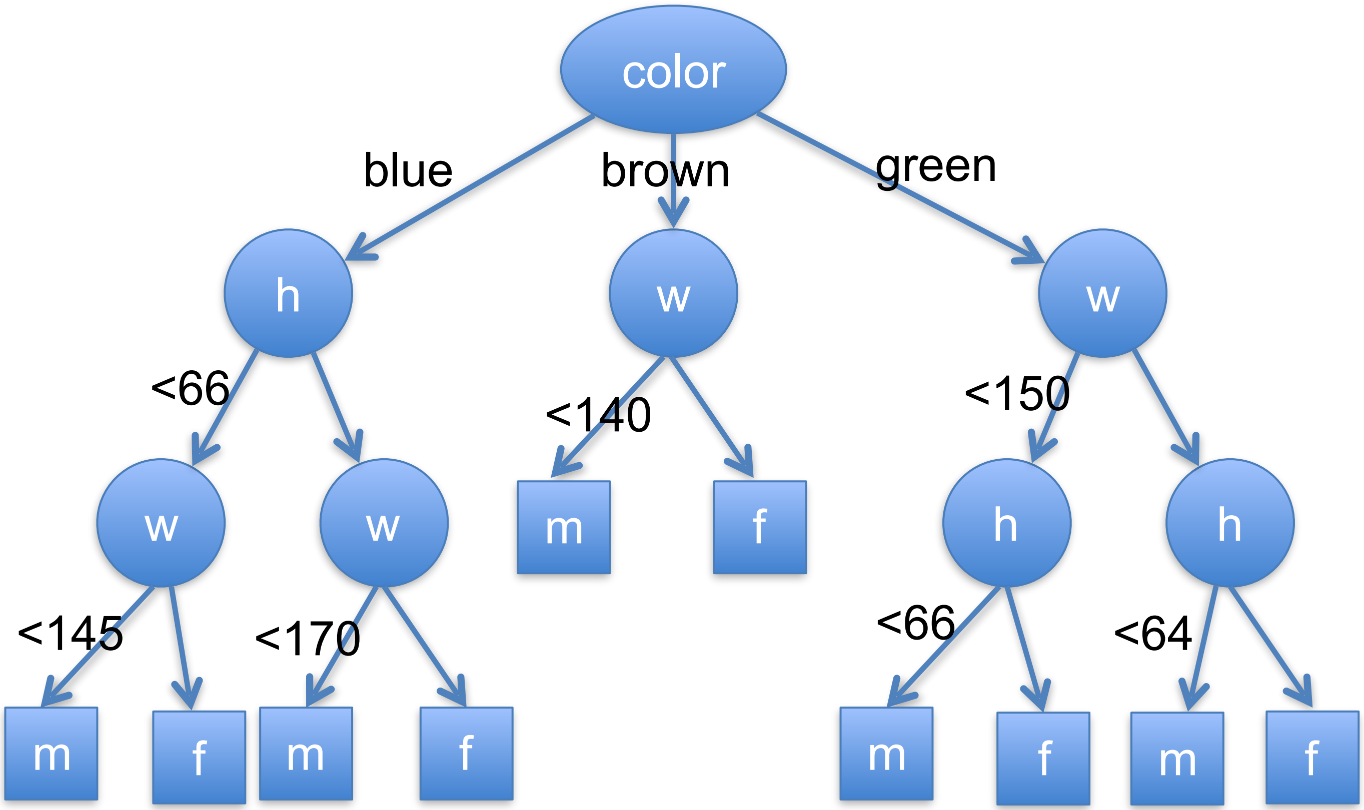
- Trees that define a decision process
  - Internal nodes: questions associated with a specific feature
  - Leaves: Decisions



Example Data Set

Height	Weight	Eye Color	Gender
66	170	Blue	Male
73	210	Brown	Male
72	165	Green	Male
70	180	Blue	Male
74	185	Brown	Male
68	155	Green	Male
65	150	Blue	Female
64	120	Brown	Female
63	125	Green	Female
67	140	Blue	Female
68	165	Brown	Female
66	130	Green	Female

Decision Trees



- Very easy to evaluate.
- Nested if statements

More formal Definition of a Decision Tree

- A **Tree** data structure
- Each **internal node** corresponds to a feature
- **Leaves** are associated with target values.
- Nodes with **nominal/categorical features** have *N* children, where *N* is the number of nominal values
- Nodes with **continuous features** have two children for values less than and greater than or equal to a **break point**.

Training a Decision Tree

- How do you decide what feature to use?
- For continuous features how do you decide what break point to use?

- Goal: Optimize Classification Accuracy.

Example Data Set

Height	Weight	Eye Color	Gender
66	170	Blue	Male
73	210	Brown	Male
72	165	Green	Male
70	180	Blue	Male
74	185	Brown	Male
68	155	Green	Male
65	150	Blue	Female
64	120	Brown	Female
63	125	Green	Female
67	140	Blue	Female
68	165	Brown	Female
66	130	Green	Female

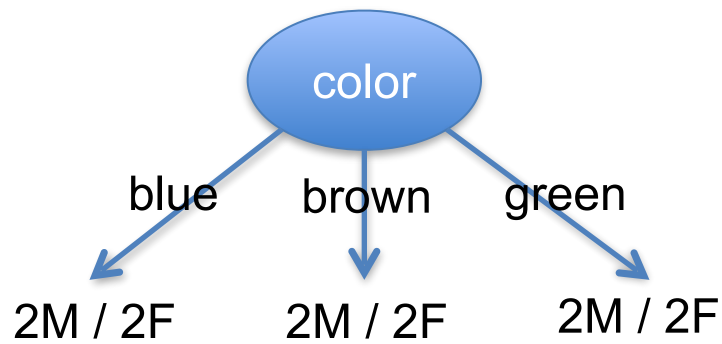
Baseline Classification Accuracy

- Select the majority class.
  - Here 6/12 Male, 6/12 Female.
  - Baseline Accuracy: 50%

- How good is each branch?
  - The improvement to classification accuracy

Training Example

- Possible branches



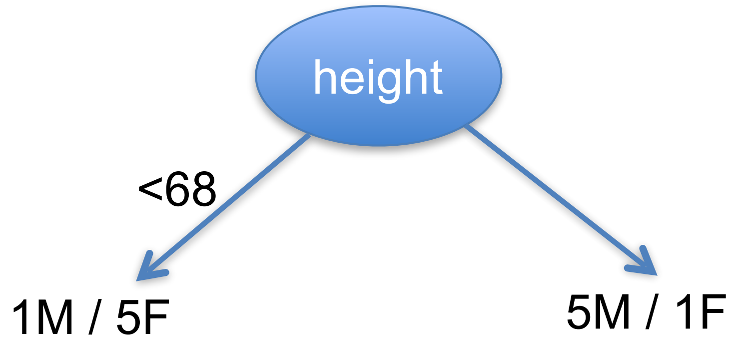
- For each node, take majority class
  - If equal numbers, random
- 50% error before split and 50% error after

Height

Height	Weight	Eye Color	Gender
63	125	Green	Female
64	120	Brown	Female
65	150	Blue	Female
66	170	Blue	Male
66	130	Green	Female
67	140	Blue	Female
68	165	Brown	Female
68	155	Green	Male
70	180	Blue	Male
72	165	Green	Male
73	210	Brown	Male
74	185	Brown	Male

Training Example

- Possible branches



50% Accuracy before Branch

83.3% Accuracy after Branch

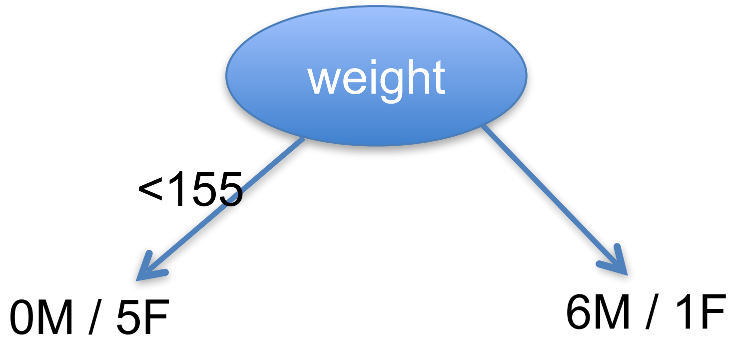
33.3% Accuracy Improvement

Weight

Height	Weight	Eye Color	Gender
64	120	Brown	Female
63	125	Green	Female
66	130	Green	Female
67	140	Blue	Female
65	150	Blue	Female
68	155	Green	Male
68	165	Brown	Female
72	165	Green	Male
66	170	Blue	Male
70	180	Blue	Male
74	185	Brown	Male
73	210	Brown	Male

Training Example

- Possible branches



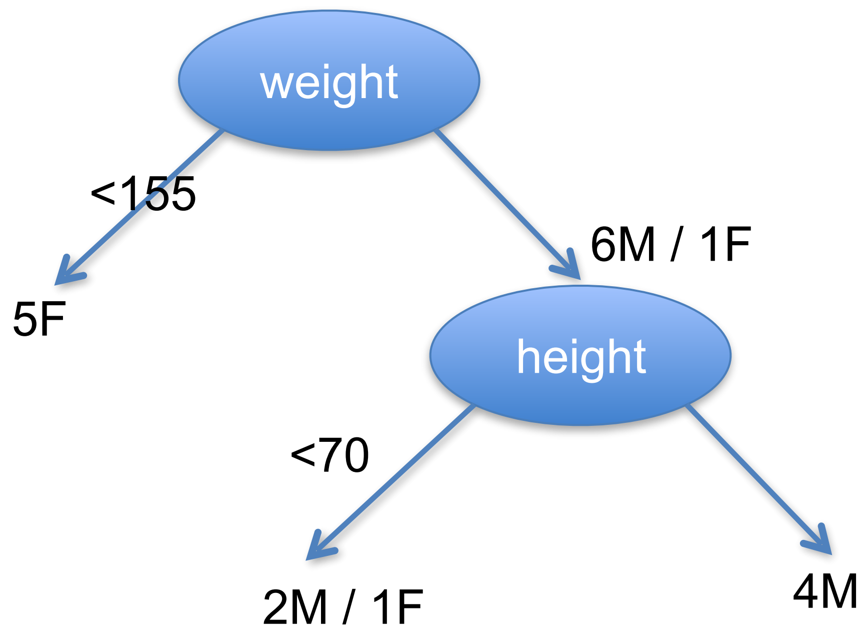
50% Accuracy before Branch

91.7% Accuracy after Branch

41.7% Accuracy Improvement

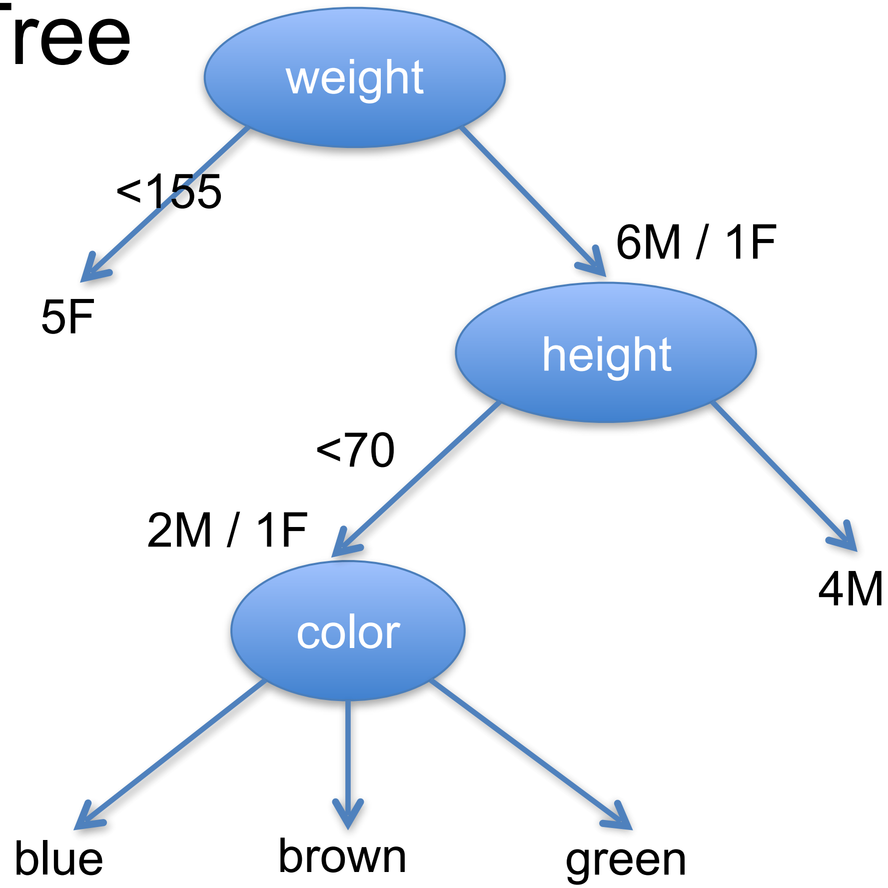
Training Example

- Recursively train child nodes.



Training Example

- Finished Tree



Generalization

- What is the performance of the tree on the training data?
  - Is there any way we could get less than 100% accuracy?

- What performance can we expect on unseen data?

Evaluation

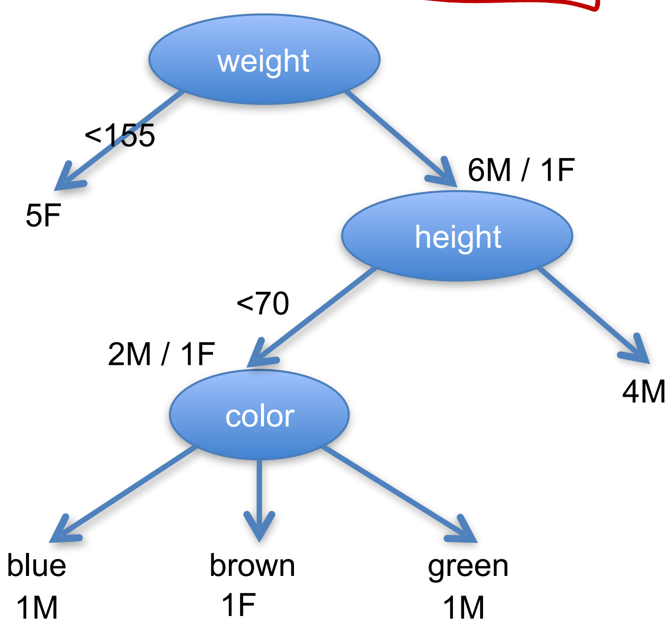
- Evaluate performance on data that was not used in training.
- Isolate a subset of data points to be used for evaluation.
- Evaluate generalization performance.

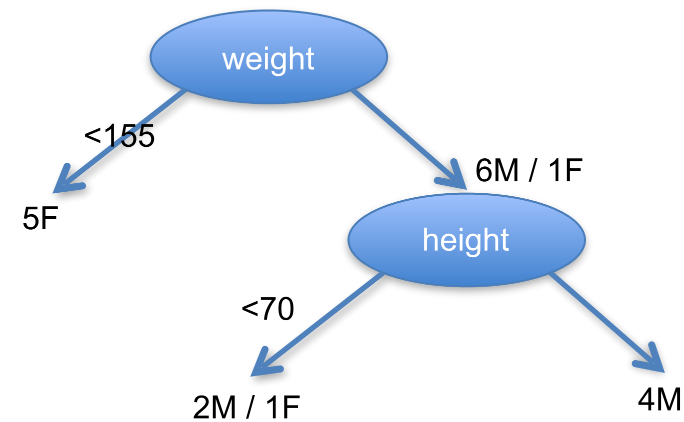
Evaluation of our Decision Tree

- What is the Training performance?
- What is the Evaluation performance?
  - Never classify male under 155
  - Never classify female over 155, and over 70.
  - The middle section is trickier.
- What are some ways to make these similar?

Pruning

- There are many pruning techniques.
- A simple approach is to have a minimum membership size in each node.





Decision Trees

- Training via **Recursive Partitioning.**
- Simple, interpretable models.
- Different node selection criteria can be used.
  - Information theory is a common choice.
- Pruning techniques can be used to make the model more robust to unseen data.

Entropy and Information Theory

- Entropy is a measure of how homogenous a data set is.
  - Also how even a probability distribution or a random variable is.
  - Smaller it is, less homogenous it is ([N,0] → 0)
- The unit of Entropy is the **bit**.
- Under an Information Theory perspective entropy represents the fewest bits it would take on average to transmit information in a signal (i.e. a random variable)

Entropy

- Say I have a vocabulary of 4 items.
  - A, B, C, D.
- A standard encoding of these might be
  - 00, 01, 10, 11.
- 2 bits per vocabulary item.
- However, if A is much more common, it might be more efficient to use this coding
  - 0, 10, 111, 110
- Exercise: What is the average bit length if there are 150 As, 40 Bs, 5 Cs, and 5Ds?
  - 1.3

Calculating Entropy

$$H(X) = -\sum_{i \in X} p_i \log p_i$$

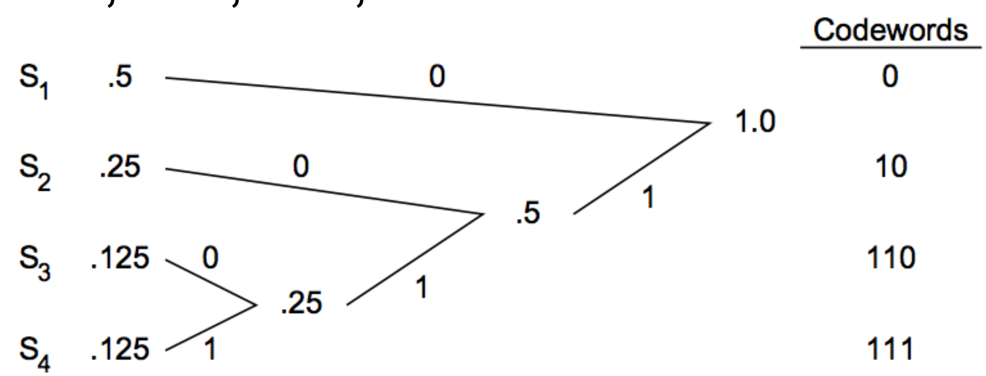
- p_i is the probability of selecting the ith value.
- For example, say X = {A A A B B B B B}
- In the calculation of entropy 0 log 0 = 0

$$H(X) = -\left(\frac{3}{8}\log\frac{3}{8} + \frac{5}{8}\log\frac{5}{8}\right)$$

- Previous slide example 1.0418
- Coding scheme reaching (arbitrarily close to) the entropy limit:
  - Huffman coding / Block coding
  - https://www.princeton.edu/~cuff/ele201/kulkarni_text/information.pdf

Huffman/block coding

- Average coding length arbitrarily close to the entropy
- Example 1: p_i = 1/2, 1/4, 1/8, 1/8



$$\begin{aligned} \text{average length} &= (1)\left(\frac{1}{2}\right) + (2)\left(\frac{1}{4}\right) + (3)\left(\frac{1}{8}\right) + (3)\left(\frac{1}{8}\right) \\ &= 1.75 \quad \text{bits/symbol} \end{aligned}$$

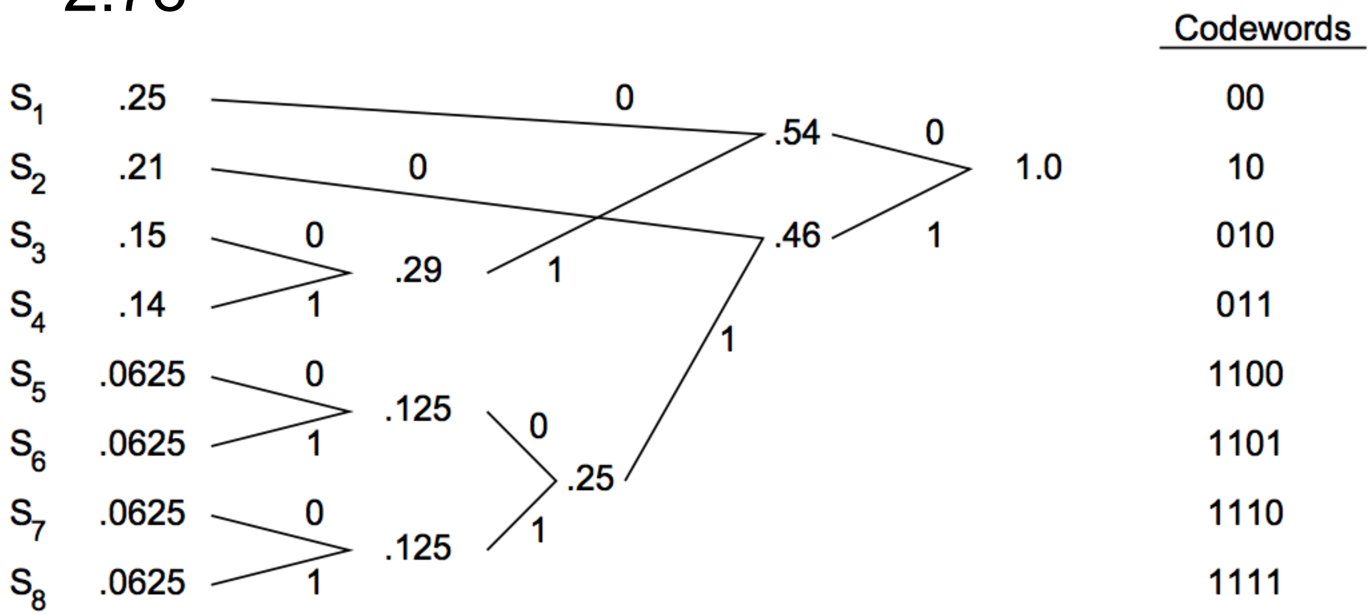
$$\begin{aligned} H &= \frac{1}{2}\log_2 2 + \frac{1}{4}\log_2 4 + \frac{1}{8}\log_2 8 + \frac{1}{8}\log_2 8 \\ &= \frac{1}{2} + \frac{1}{2} + \frac{3}{8} + \frac{3}{8} \\ &= 1.75 \end{aligned}$$

From Paul Cuff

Huffman/block coding

– Average coding length = 2.79 bits / symbol

– H = 2.78

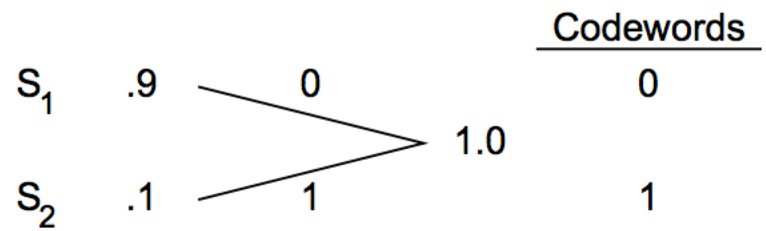


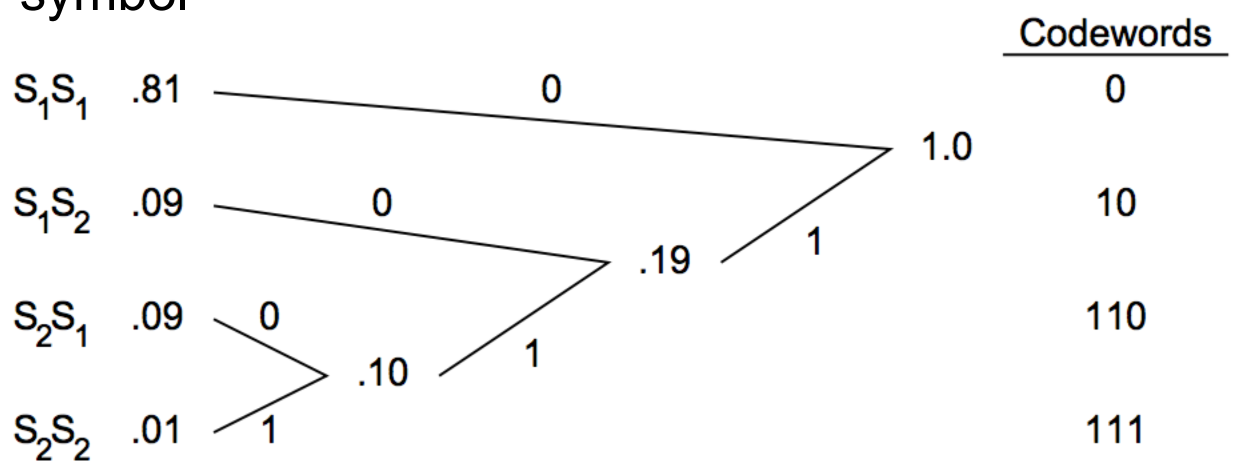
– Algorithm:
- Put every symbol into a heap
- Take smallest two, merge them, put their sum into the heap
- Repeat until only one left

From Paul Cuff

Huffman/block coding

- Issue: may never be close enough
- H = 0.47
- Avg coding length = 1
- Take blocks (of 2)
  - 1(0.81)+2(0.09) + 3(0.09 + 0.01) = 1.29 bits / block
  - 0.645 bits per symbol



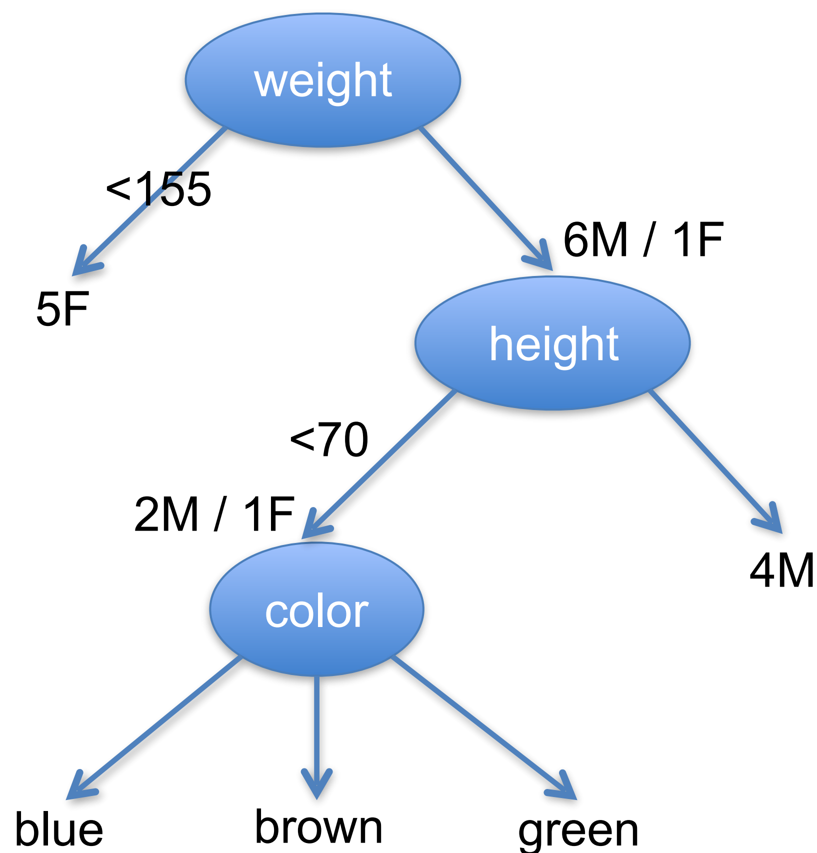


From Paul Cuff

- In general: bigger blocks, closer to the entropy

Splitting a Node

- Categorical values
- Continuous values:
  – find the threshold so the two split nodes have the smallest entropy

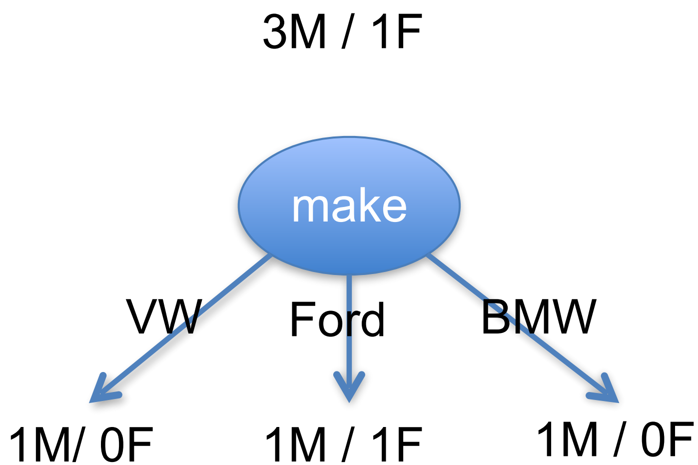


Information Gain

- In our previous example we examined the improvement to classification performance.
  – Error reduction or change to overall accuracy.
- Using entropy the measure that is optimized is Information Gain.
  – The difference in the entropy of the **label** or **class distribution** before or after a particular decision tree split.

Calculating Information Gain

$$Gain(S,F) = H(S) - \sum_{f \in values(F)} \frac{|S_f|}{|S|} H(S_f)$$



$H(S) =$

$H(S_{VW}) =$

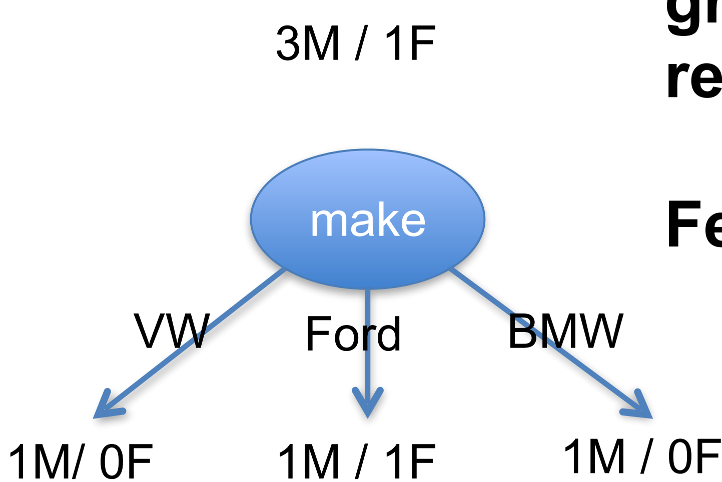
$H(S_{Ford}) =$

$H(S_{BMW}) =$

$Gain(S,F) =$

Calculating Information Gain

$$Gain(S,F) = H(S) - \sum_{f \in values(F)} \frac{|S_f|}{|S|} H(S_f)$$



Identify the feature with the greatest Information Gain and repeat this process recursively!

Feature can be reused

Other Measures

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k),$$

Misclassification error: $\frac{1}{N_m} \sum_{i \in R_m} I(y_i \neq k(m)) = 1 - \hat{p}_{mk(m)}$.

Gini index: $\sum_{k \neq k'} \hat{p}_{mk} \hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$.

Cross-entropy or deviance: $-\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$.

(9.17)

- Ignore m subscript
- k(m) the class with largest probability p_k (prediction if using majority vote)
- Gini: expected error

Section 9.2.3 of Hastie et al.

Purity (binary class)

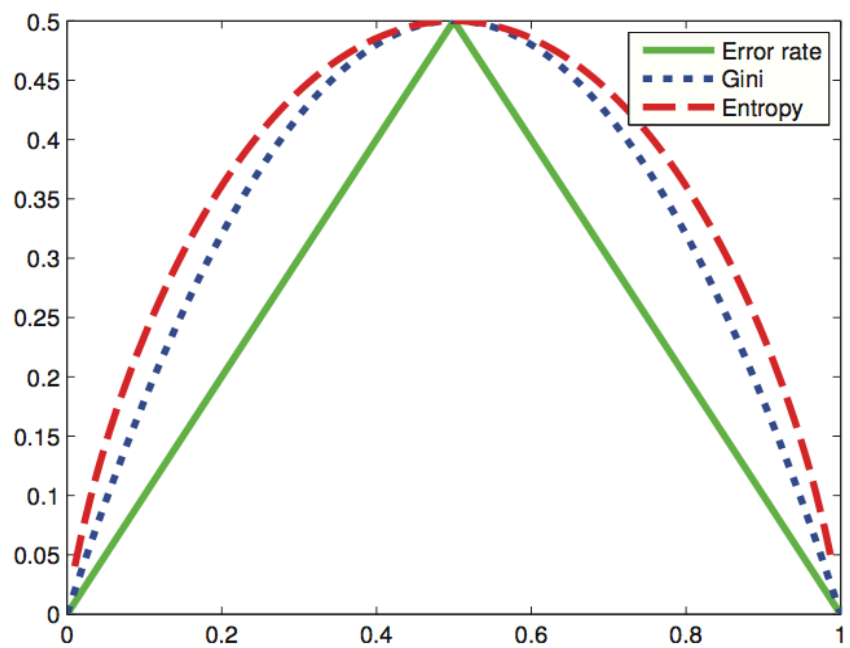
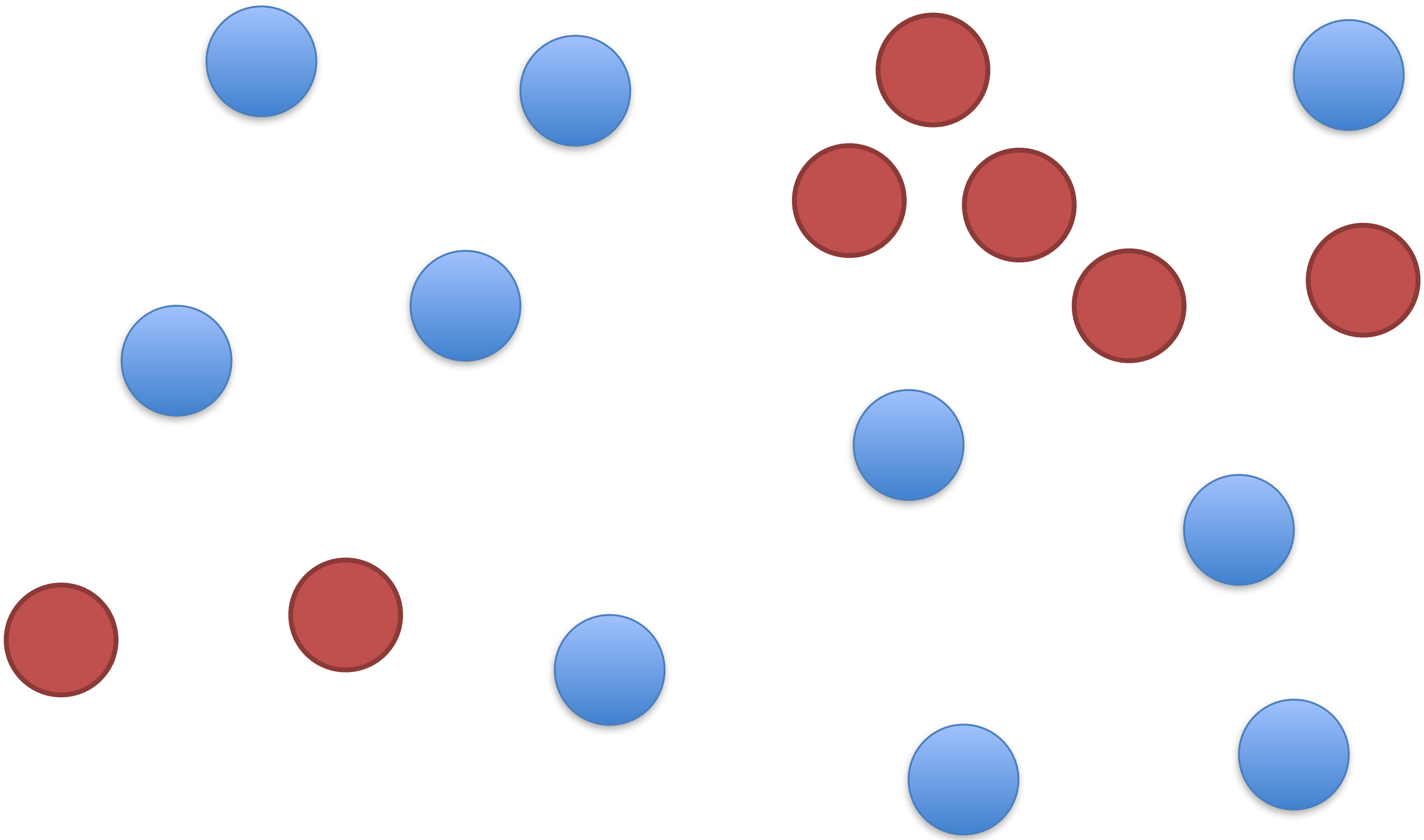
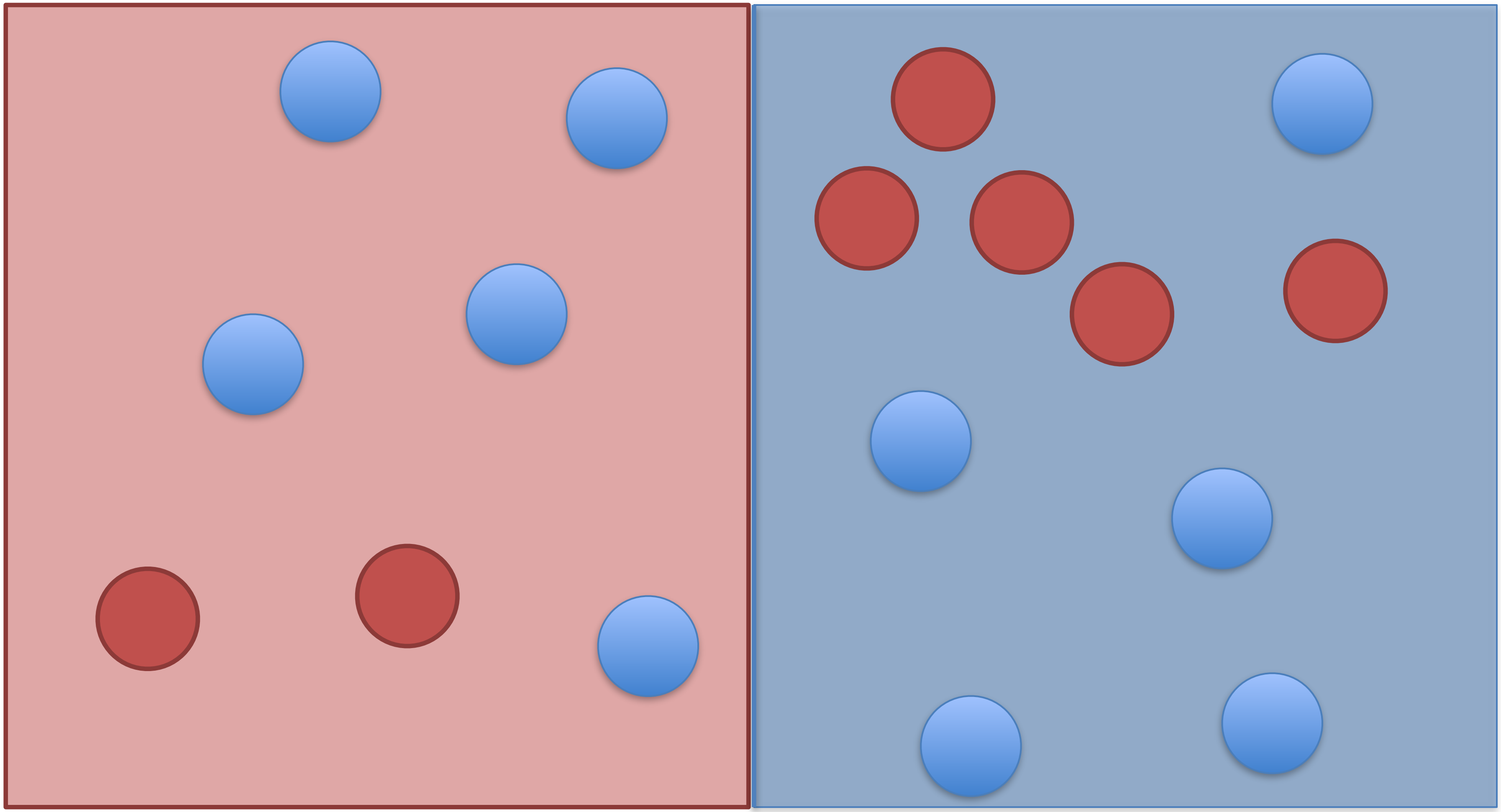


Figure 16.3 Node impurity measures for binary classification. The horizontal axis corresponds to p, the probability of class 1. The entropy measure has been rescaled to pass through (0.5,0.5). Based on Figure 9.3 of (Hastie et al. 2009). Figure generated by `giniDemo`.

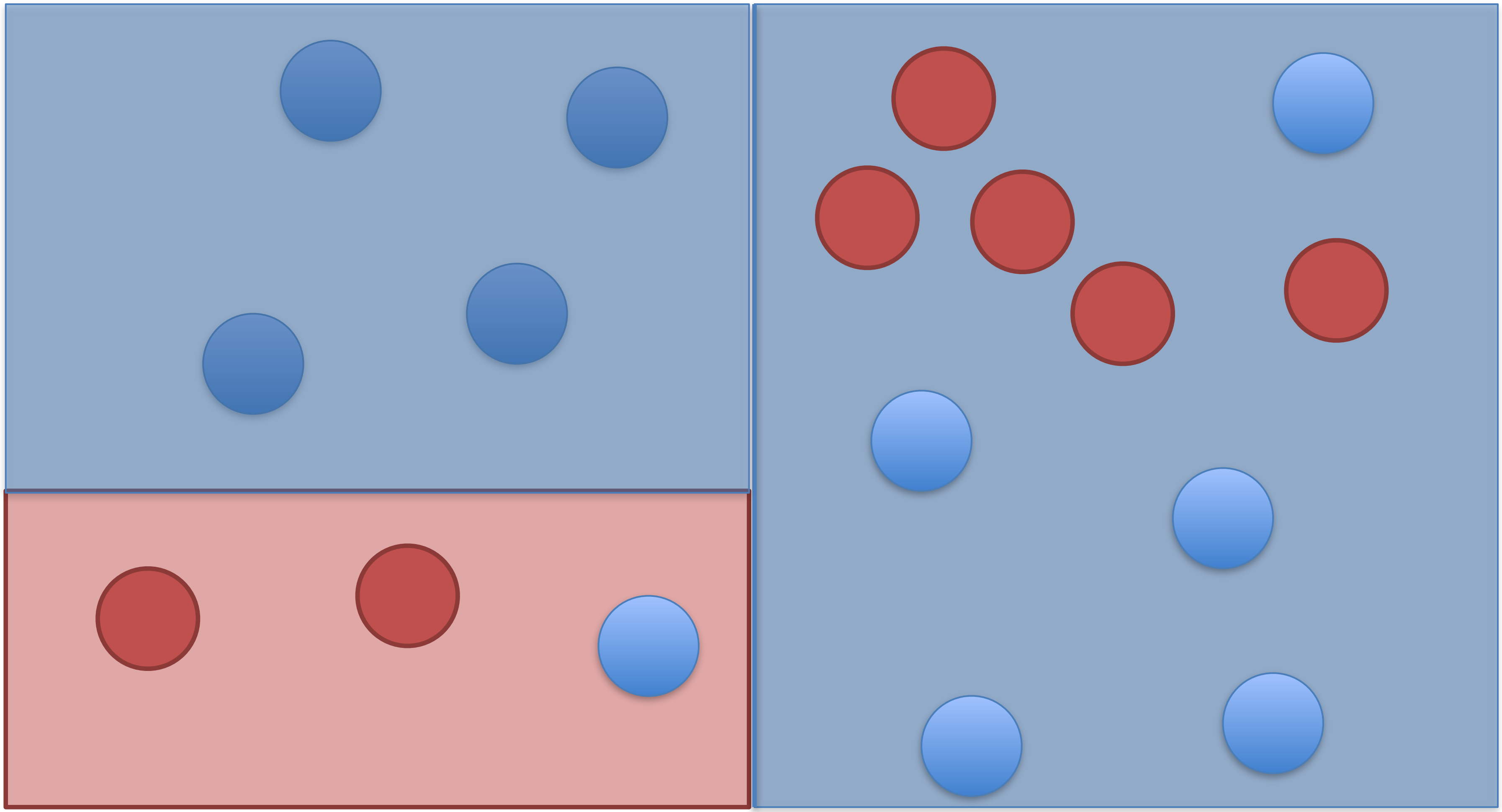
Visualization of Decision Tree Training



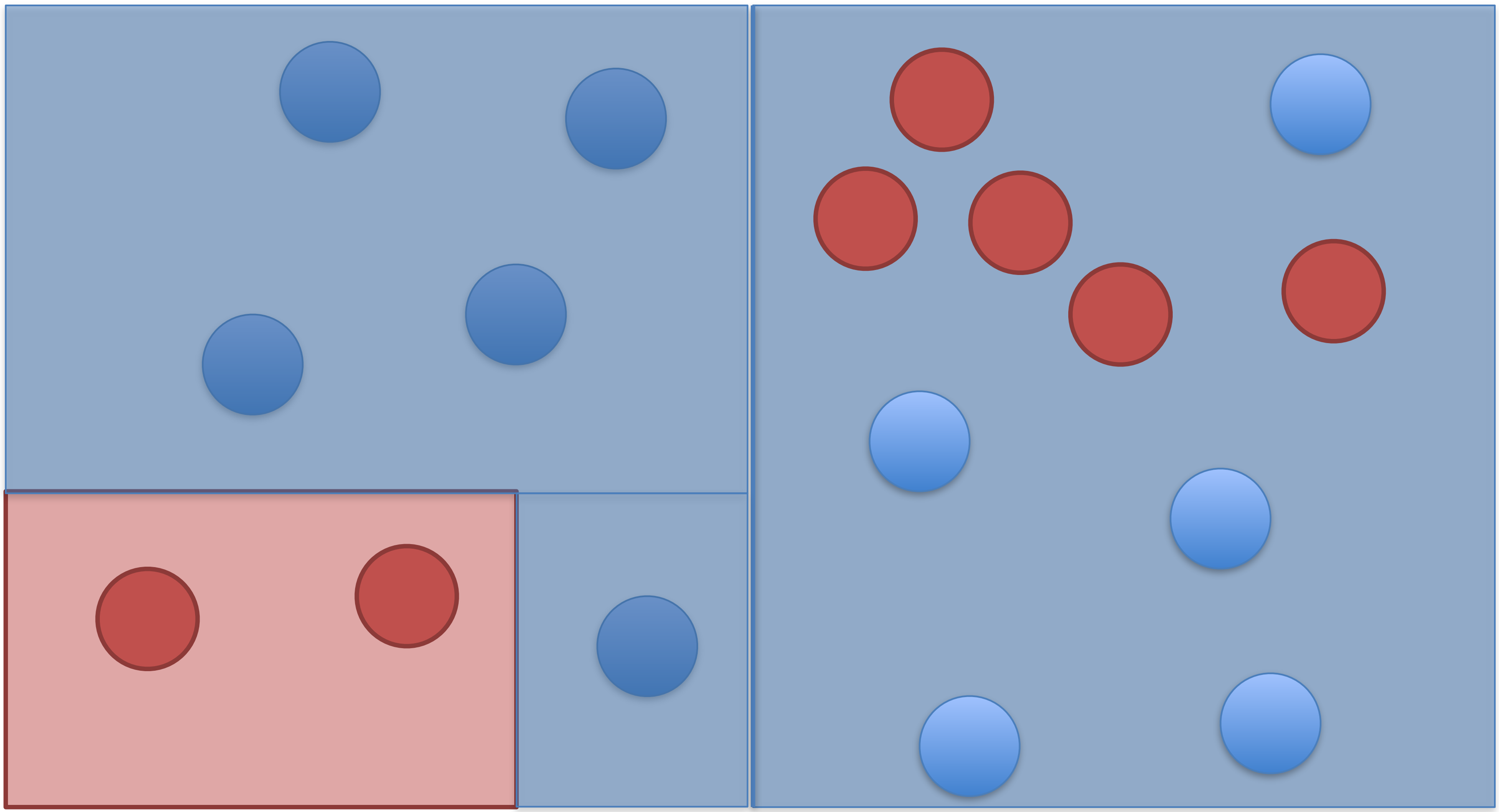
Visualization of Decision Tree Training



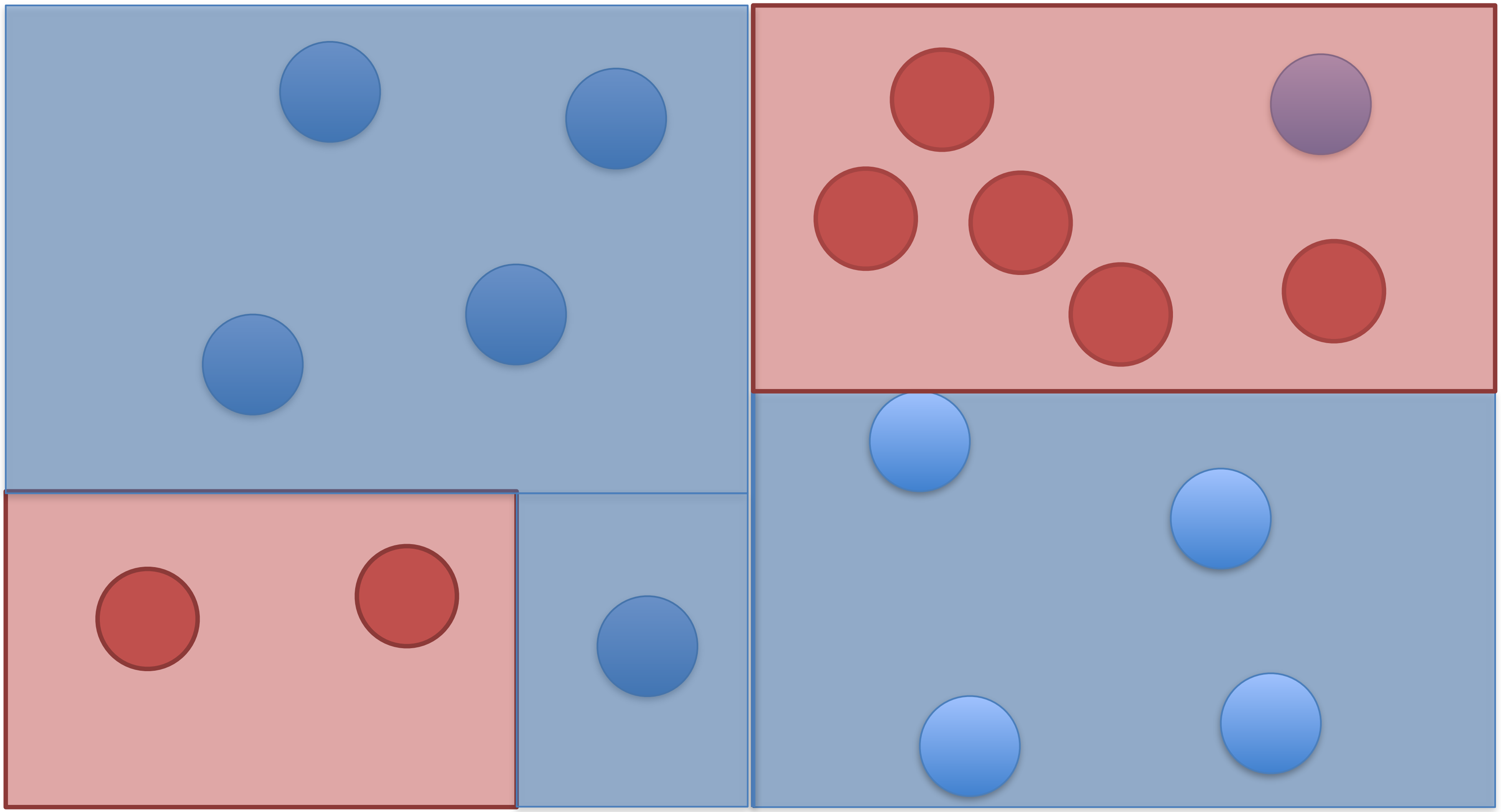
Visualization of Decision Tree Training



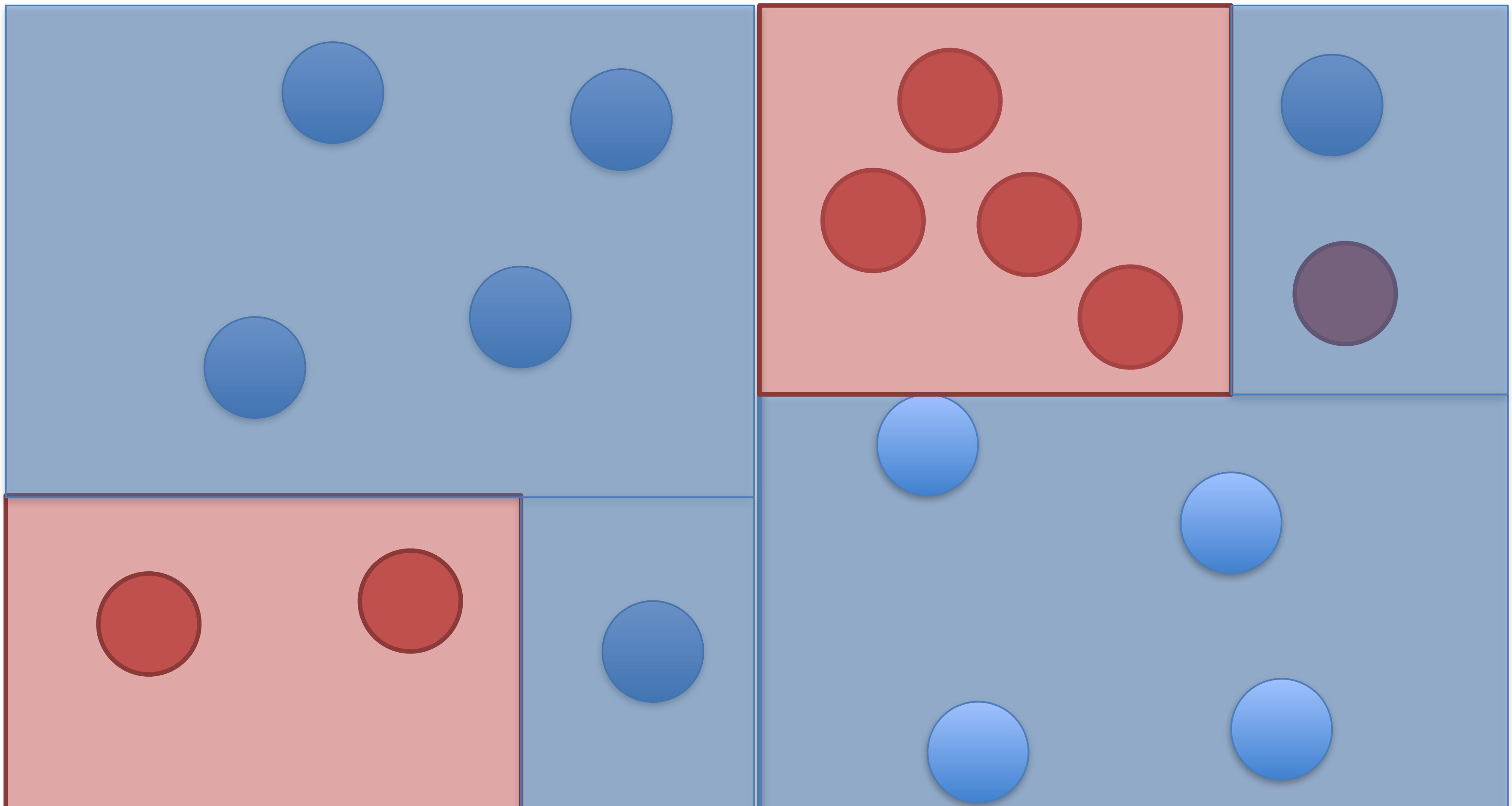
Visualization of Decision Tree Training



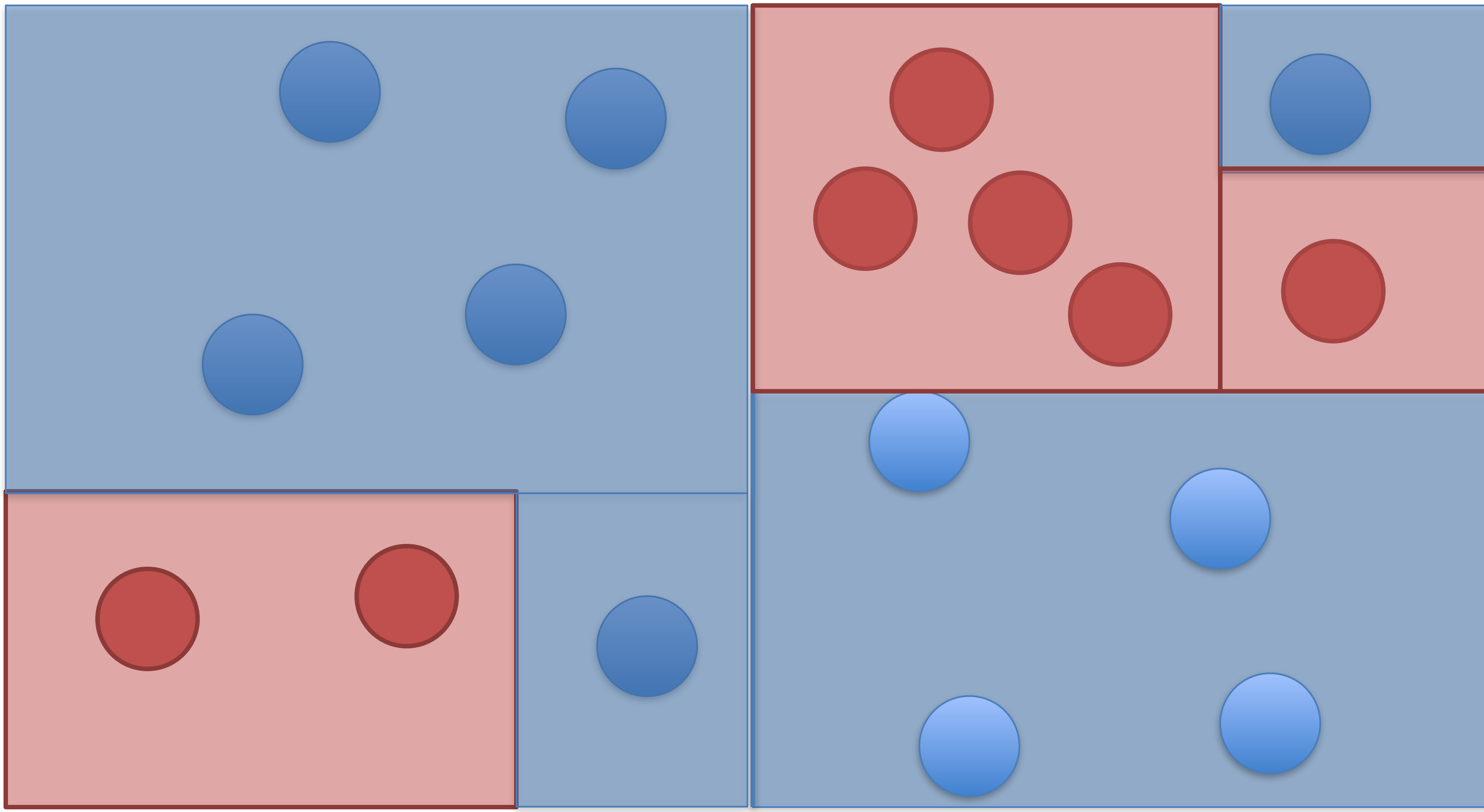
Visualization of Decision Tree Training



Visualization of Decision Tree Training



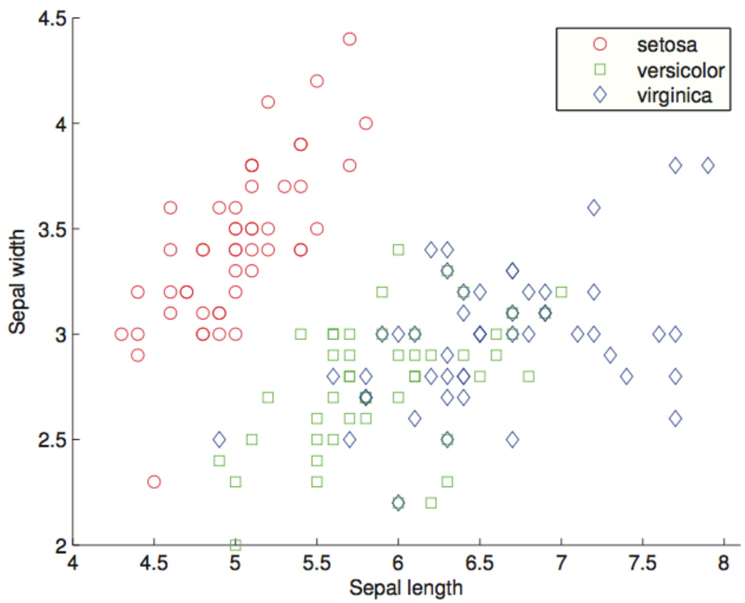
Visualization of Decision Tree Training



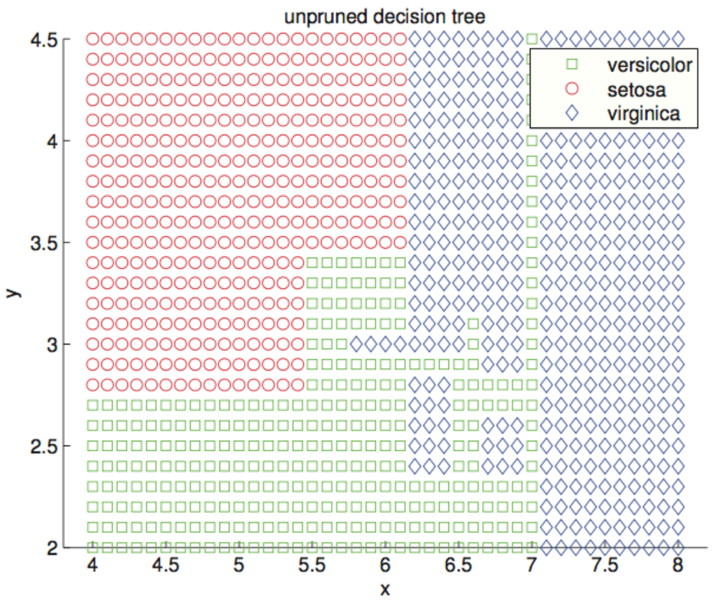
Evaluation of our Decision Tree

- What is the Training performance?
  - Training error: error on the training data set
- What is the Evaluation performance?
  - On unseen data (test data)
  - In training, we reserve a portion of training data, use them to evaluate the performance (called test error)
- Overfitting:
  - training error is much lower than test error
- What are some ways to make these similar?

Overfitting



(a)

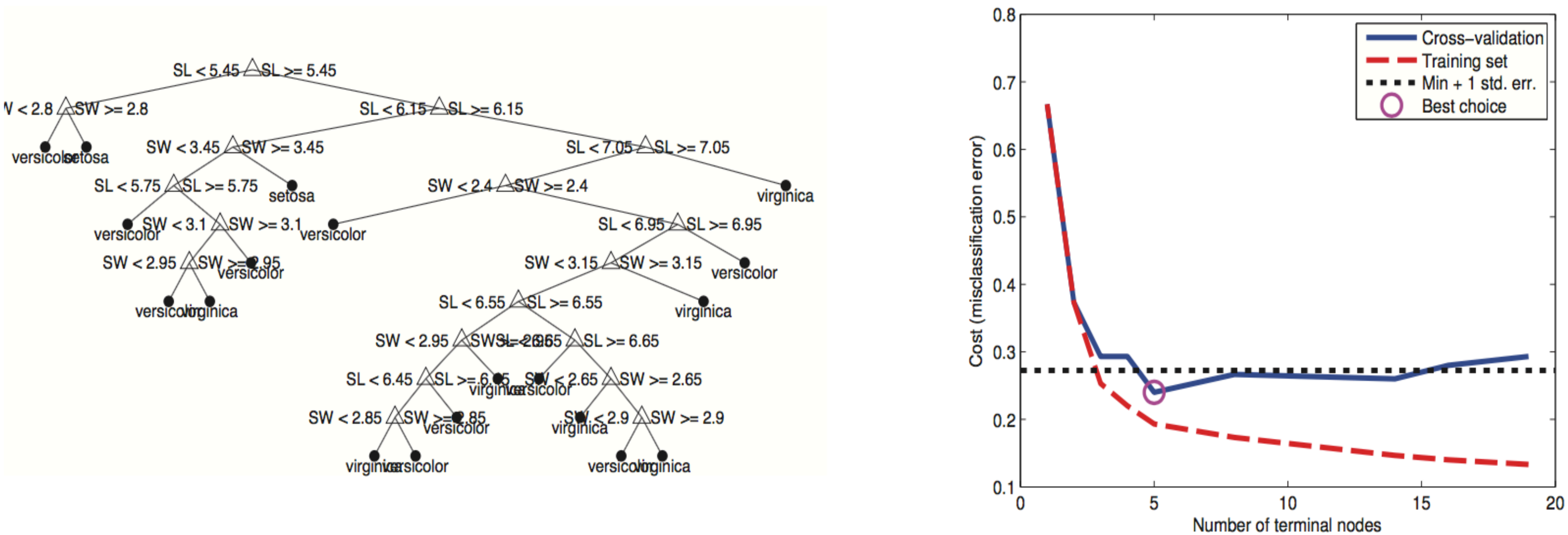


(b)

Figure 16.4 (a) Iris data. We only show the first two features, sepal length and sepal width, and ignore petal length and petal width. (b) Decision boundaries induced by the decision tree in Figure 16.5(a).

From K. Murphy book

Overfitting (cont'd)



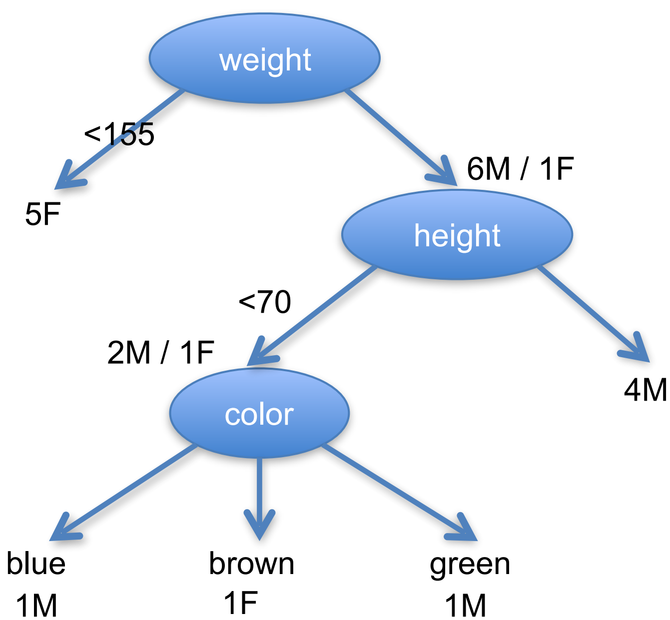
(a) (b)

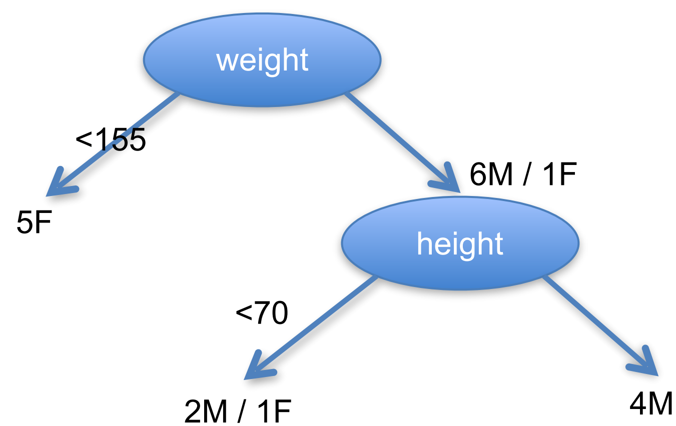
Figure 16.5 (a) Unpruned decision tree for Iris data. (b) Plot of misclassification error rate vs depth of tree. Figure generated by `dtreeDemoIris`.

From K. Murphy book
Code demo (plot_iris.py, iris dataset).

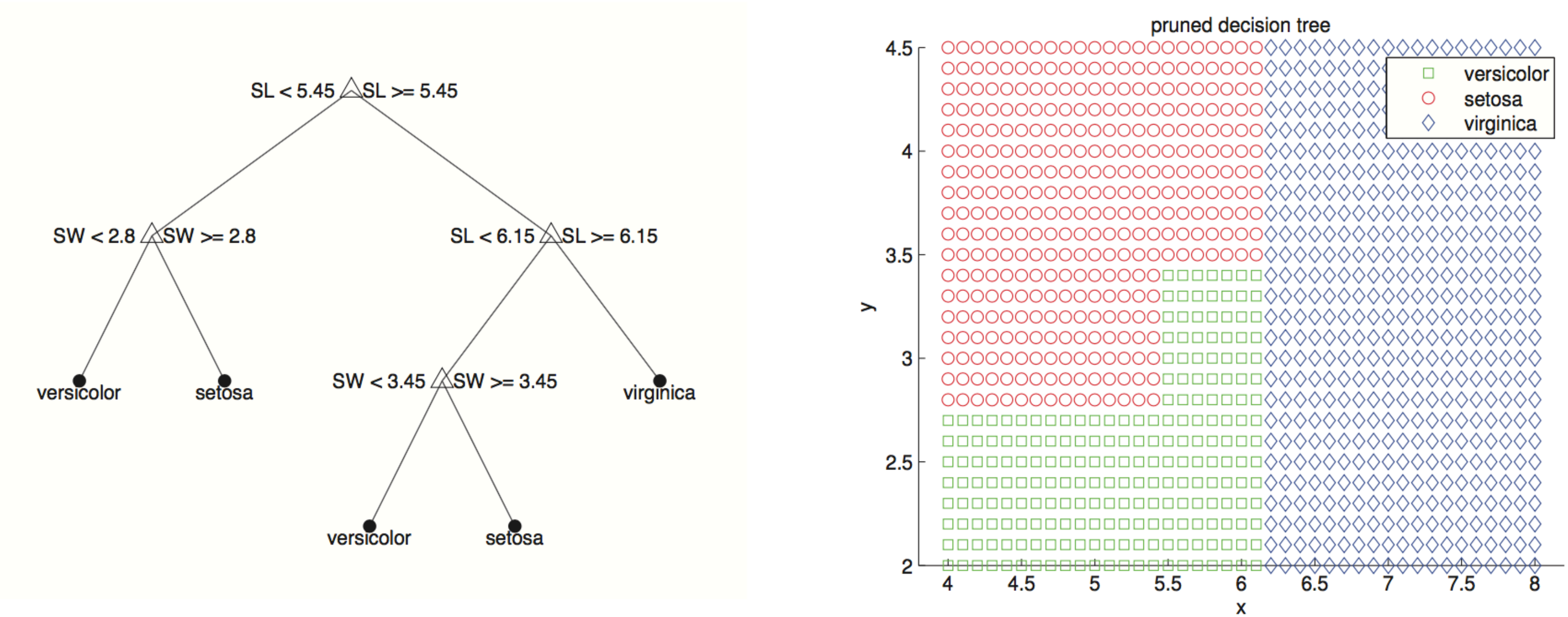
Pruning

- There are many pruning techniques.
  - minimum membership size in each node.
  - # of leaf nodes.   - Depth
- Start from the full tree, iteratively remove the split with the lest improvement





Pruned Tree

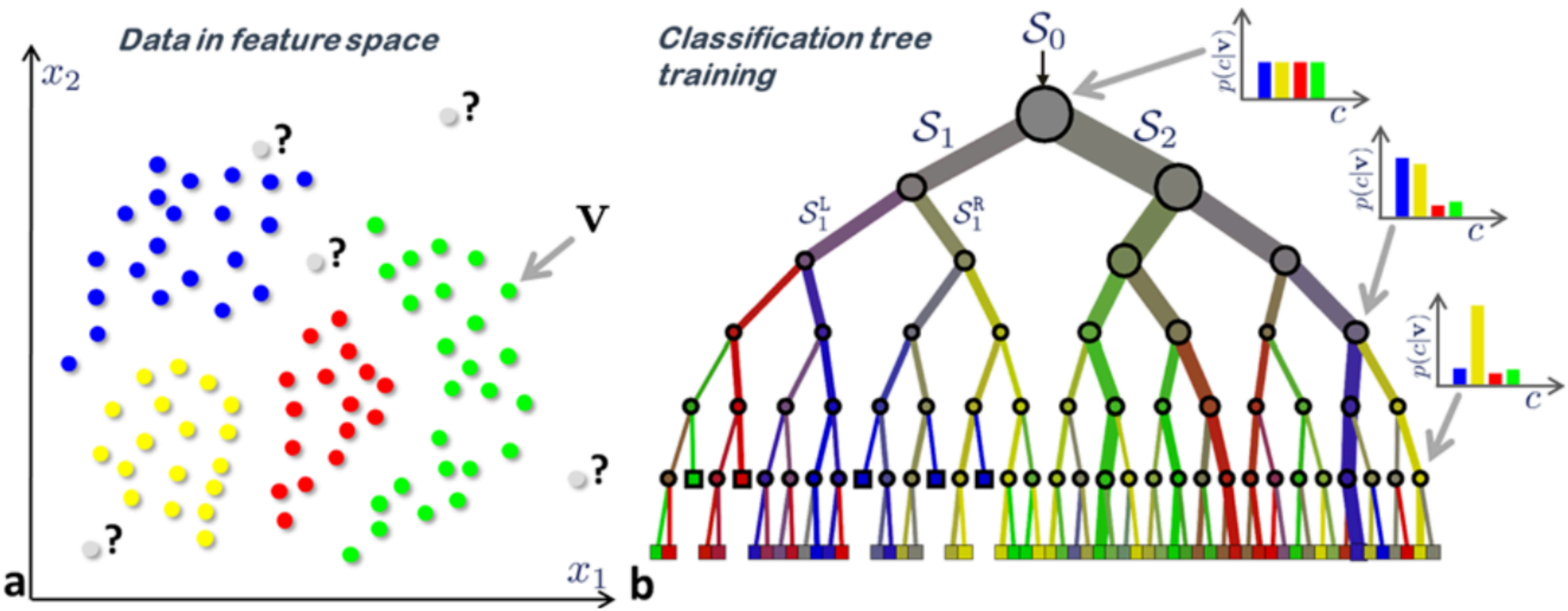


(a) (b)

Figure 16.6 Pruned decision tree for Iris data. Figure generated by `dtreeDemoIris`.

From K. Murphy book
Code demo (my_plot_forest_iris.py).

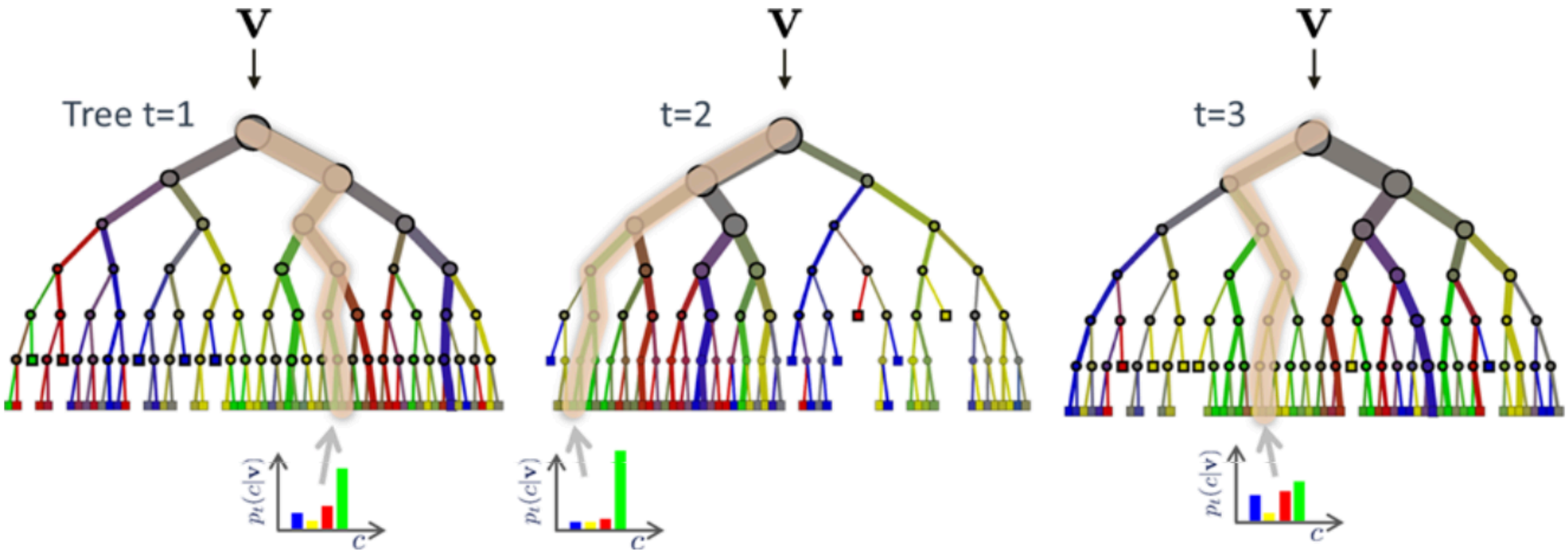
Pruned Tree



$$c^* = \arg\max_c p(c|\mathbf{v})$$

Figure from Criminisi & Shotton

Multiple Trees = Forest



Random forest:

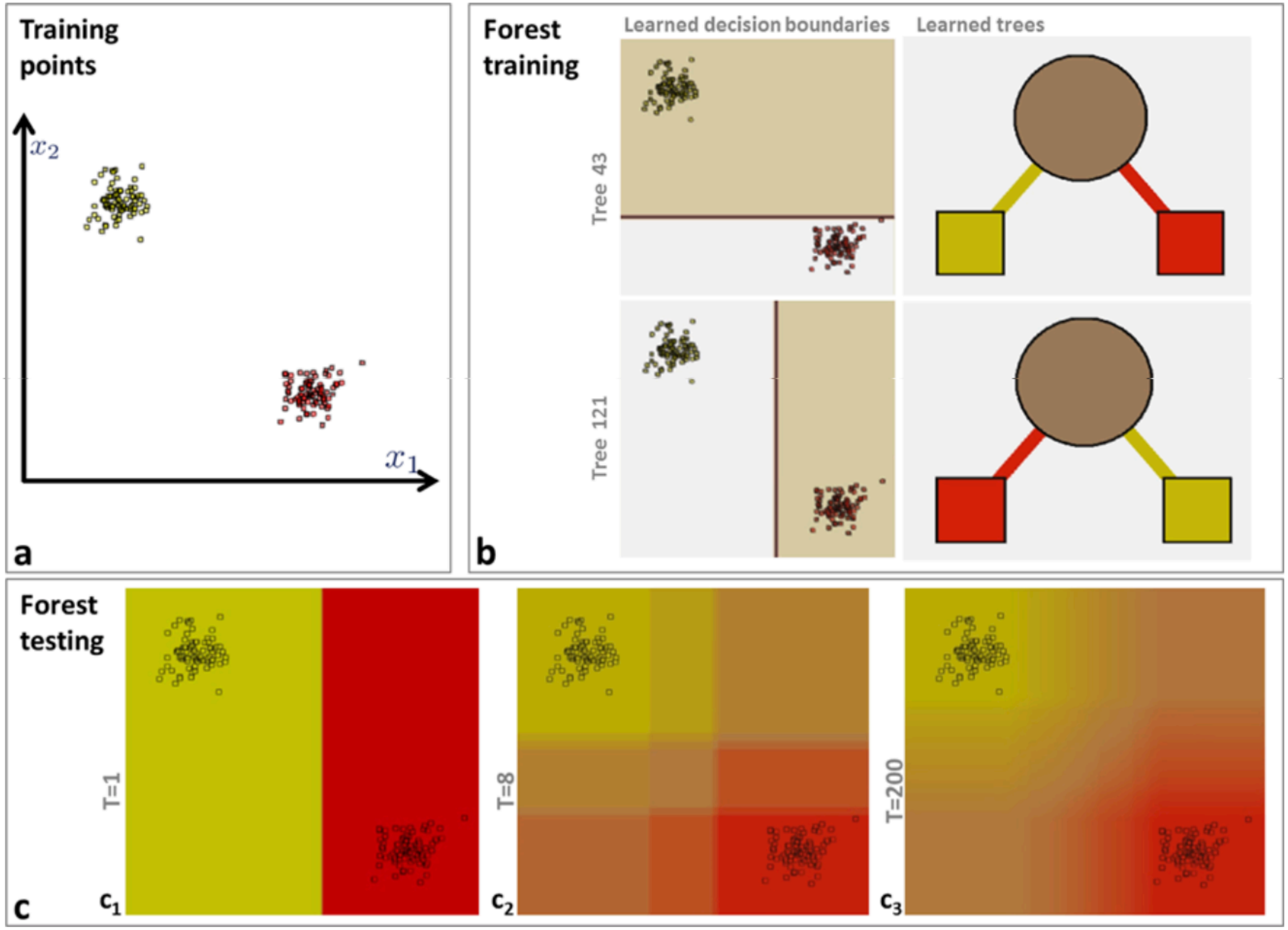
Generate many trees

Take the average of their decisions

$$c^* = \arg\max_c p(c|\mathbf{v}) \qquad p(c|\mathbf{v}) = \frac{1}{T}\sum_{t=1}^{T} p_t(c|\mathbf{v}),$$

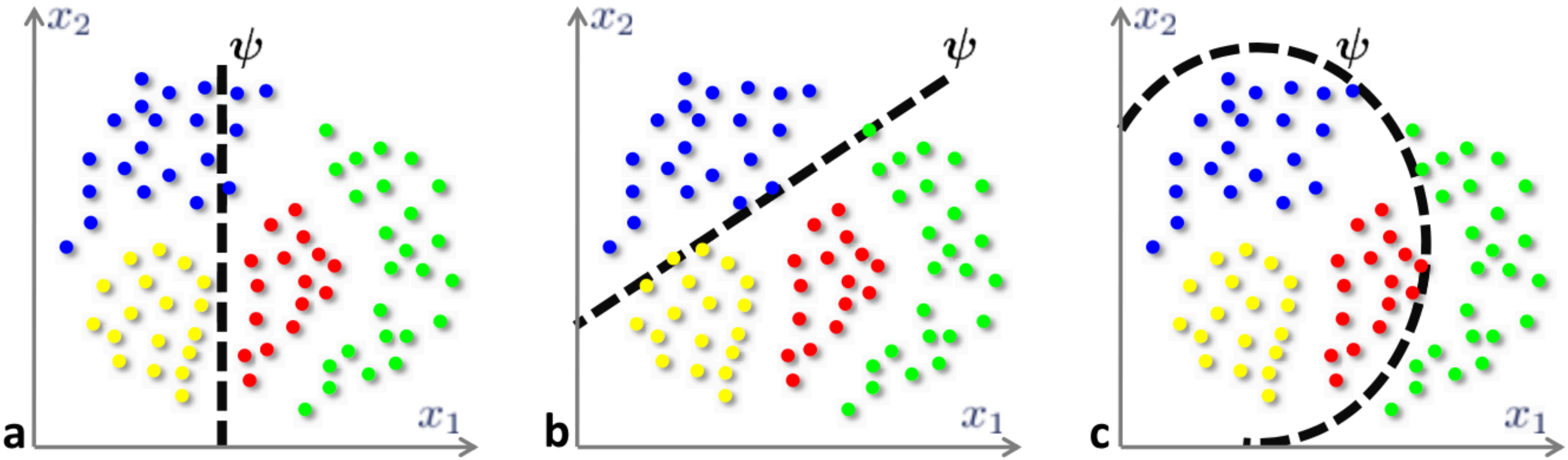
Random Forest

– Trees should make different decisions, but they all should be somehow correct!
– How?
  - Use partial information randomness
  - Random subset of data for each tree (bagging)
  - Random subset of features for each tree
  - Random splitting thresholds

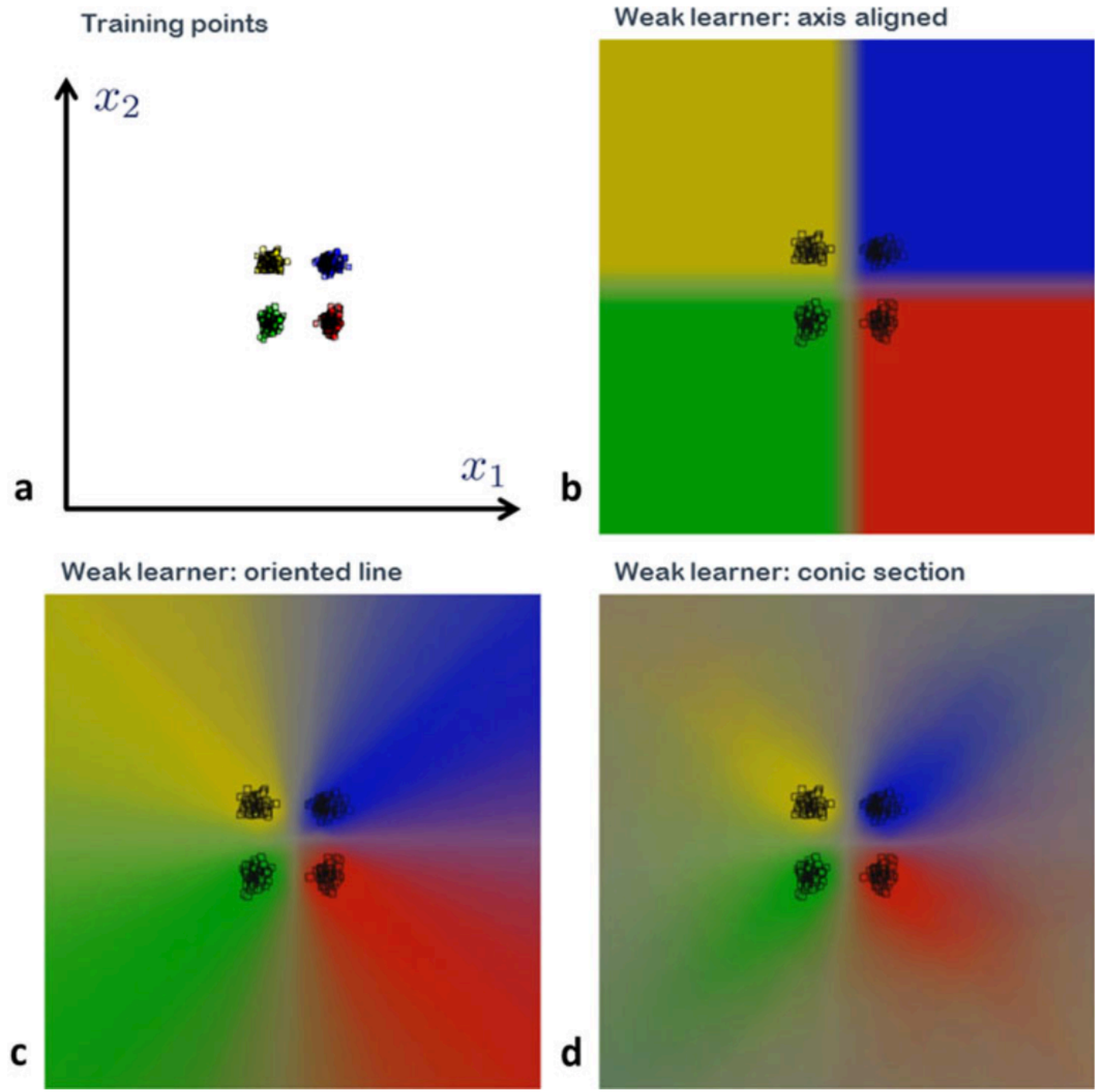


- Individual trees: overconfident predictions
- Average:
  - confident near data, less so in the middle

Random Forest



- Fancier weak learners
  - Linear: $ax_1 + bx_2 + c > t$
  - When is it equal to the original one?



Training points
x_2
x_1
a
Weak learner: axis aligned
b
Weak learner: oriented line
c
Weak learner: conic section
d

Generating Random Trees

- Demo in python
- Example in application
- References: book, slides (very big 500Mb)
  - http://research.microsoft.com/apps/pubs/default.aspx?id=158806
  - http://research.microsoft.com/en-us/um/people/antcrim/ACriminisi_DecisionForestsTutorial.pptx
- Ensemble methods:
  - Systematically generate a set of models and combine their results for prediction
  - Examples: random forest, AdaBoost, etc.