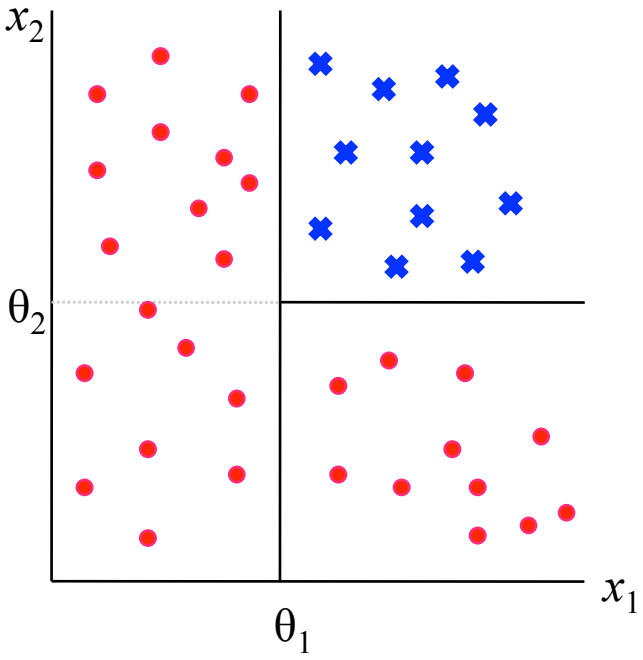
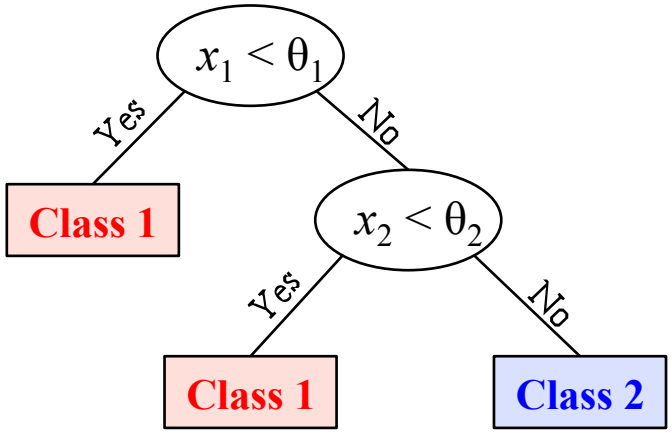
Decision Trees

CS 550: Machine Learning

Decision Trees

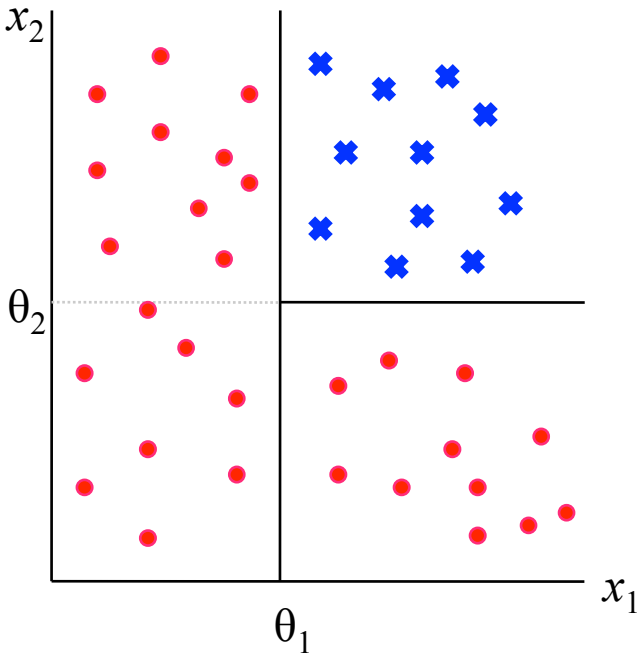
- A decision tree provides a classification or regression model built in the form of a tree structure
- It corresponds to partitioning the input space into localized regions, each of which can make different decision
- Decision tree learning aims to find these partitions

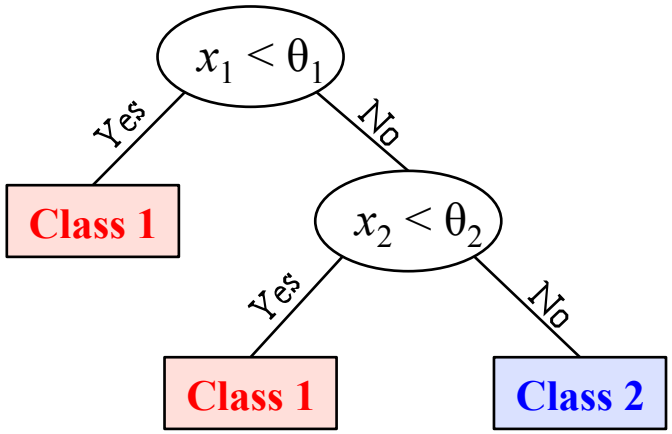




Decision Trees

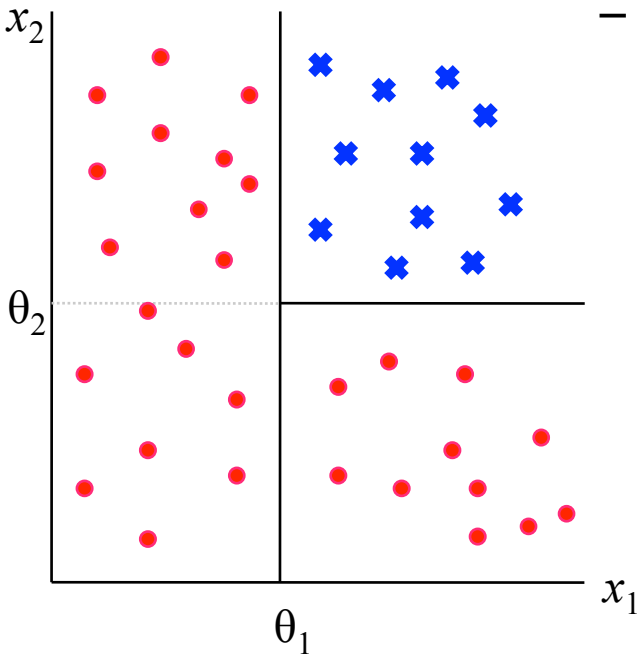
- It is composed of internal decision nodes and leaves
  - An internal node corresponds to a test function whose discrete outcomes label the branches
  - A leaf defines a localized region (and a class for classification and a numerical value for regression)

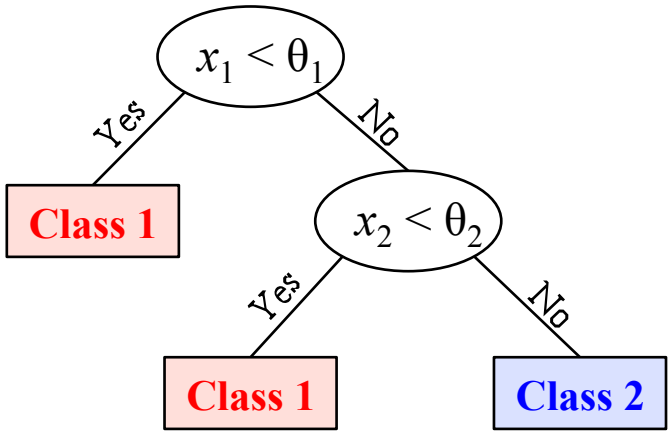




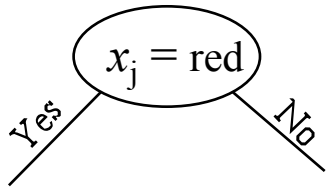
Decision Trees

- In training, the goal is to construct a tree yielding the minimum error
  - At each step, the "best" split is selected among all possible ones
  - Tree construction iteratively continues until all leaves are pure
  - This is the basis of CART, ID3, and C4.5 algorithms
- For an unseen instance, start at the root, take branches according to the test outcomes until a leaf is reached
  - The value in the leaf is the output

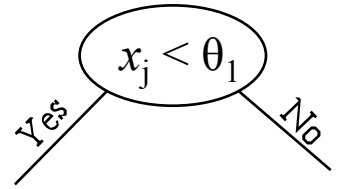




Decision Trees

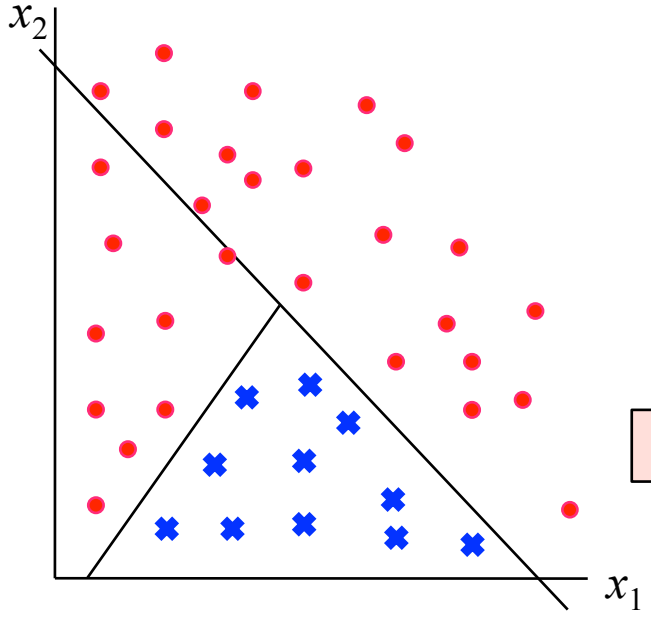


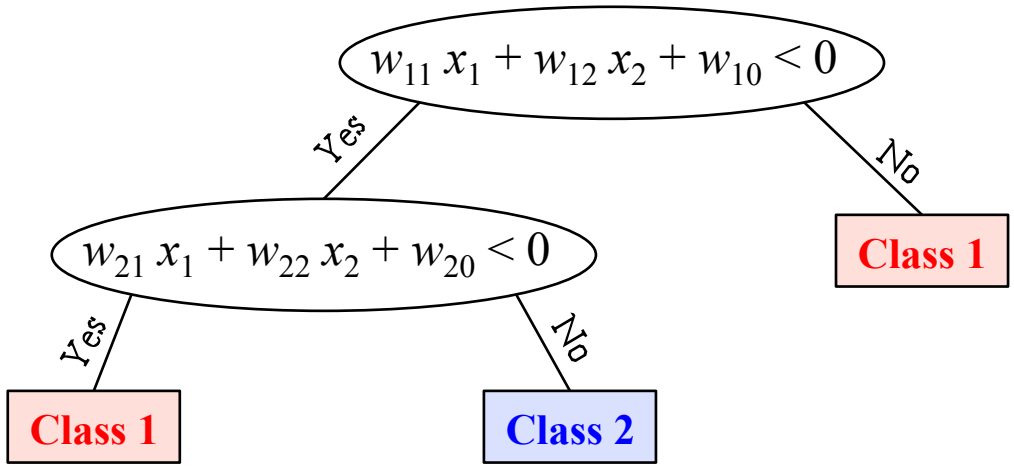
Discrete features



Continuous features

- Univariate trees
  - Test functions use one feature at a time
  - Define splits orthogonal to the coordinate axes
- Multivariate trees
  - Test functions use more than one feature at a time





Classification Trees

- For tree construction, iteratively select the "best" split until all leaves are pure
- What is the "best" split?
  - The goodness of a split is quantified by an impurity measure
  - Entropy is one of the most commonly used measures

$$\underbrace{I(m)}_{\textbf{Entropy at node m}} = -\sum_{i=1}^{C} \underbrace{P_m(C_i)}_{\text{Probability of having i-th class at node m}} \log P_m(C_i)$$

number of classes (pointing to C)

$$\underbrace{I(S)}_{\textbf{Entropy of a binary split S}} = P_{left}\, I(left) + P_{right}\, I(right)$$

Classification Trees

Construct a tree for the training instances below

	x_1	x_2	class
S_1	red	0.5	**1**
S_2	red	0.2	**2**
S_3	green	0.5	**2**
S_4	blue	0.1	**1**
S_5	red	-0.5	**2**
S_6	green	0.1	**1**
S_7	green	0.4	**2**
S_8	blue	0.0	**2**

- At every step
  1. List all possible splits
  2. Calculate the entropy for every split
  3. Select the one with the minimum entropy

Classification Trees

Construct a tree for the training instances below

	x_1	x_2	class
S_1	red	0.5	**1**
S_2	red	0.2	**2**
S_3	green	0.5	**2**
S_4	blue	0.1	**1**
S_5	red	-0.5	**2**
S_6	green	0.1	**1**
S_7	green	0.4	**2**
S_8	blue	0.0	**2**

- For classification
  - Each different value of a **discrete feature** will define a split
  - Halfway between **continuous feature** values of the samples belonging to different classes will be split points
  - Possible splits:

    x_1 = red x_1 = green x_1 = blue

    $x_2 \leq 0.05$ $x_2 \leq 0.15$ $x_2 \leq 0.45$

Classification Trees

Construct a tree for the training instances below

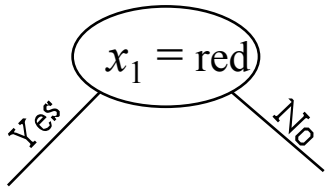
	x_1	x_2	class
S_1	red	0.5	**1**
S_2	red	0.2	**2**
S_3	green	0.5	**2**
S_4	blue	0.1	**1**
S_5	red	-0.5	**2**
S_6	green	0.1	**1**
S_7	green	0.4	**2**
S_8	blue	0.0	**2**

- Calculate the entropy for all possible splits

$$I(x_1 = red) = P_{Yes}\, I(Yes) + P_{No}\, I(No)$$

$$= \frac{3}{8}\left(-\frac{1}{3}\log\frac{1}{3} - \frac{2}{3}\log\frac{2}{3}\right) + \frac{5}{8}\left(-\frac{2}{5}\log\frac{2}{5} - \frac{3}{5}\log\frac{3}{5}\right)$$

$$= 0.9512$$



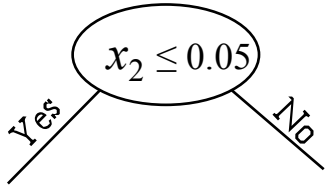
S_1 – Class 1
S_2 – Class 2
S_5 – Class 2

S_3 – Class 2
S_4 – Class 1
S_6 – Class 1
S_7 – Class 2
S_8 – Class 2

$$I(x_2 \le 0.05) = P_{Yes}\, I(Yes) + P_{No}\, I(No)$$

$$= \frac{2}{8}\left(-0\log 0 - 1\log 1\right) + \frac{6}{8}\left(-\frac{3}{6}\log\frac{3}{6} - \frac{3}{6}\log\frac{3}{6}\right)$$

$$= 0.7500$$



S_5 – Class 2
S_8 – Class 2

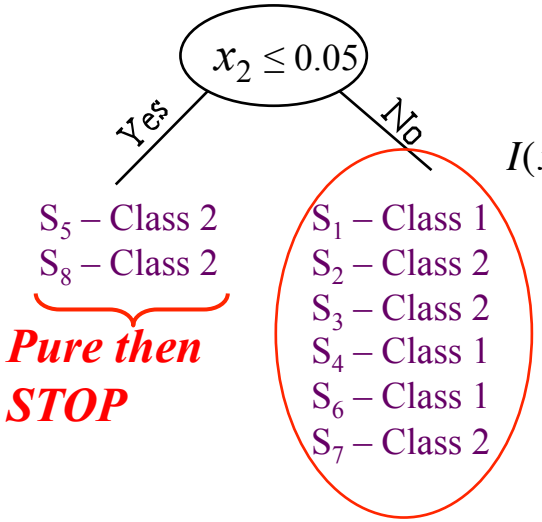
S_1 – Class 1
S_2 – Class 2
S_3 – Class 2
S_4 – Class 1
S_6 – Class 1
S_7 – Class 2

Classification Trees

Construct a tree for the training instances below

	x_1	x_2	class
S_1	red	0.5	**1**
S_2	red	0.2	**2**
S_3	green	0.5	**2**
S_4	blue	0.1	**1**
S_5	red	-0.5	**2**
S_6	green	0.1	**1**
S_7	green	0.4	**2**
S_8	blue	0.0	**2**

- Select the split with the minimum entropy and continue



$$I(x_1 = red) = P_{Yes}\, I(Yes) + P_{No}\, I(No)$$

$$= \frac{2}{6}\left(-\frac{1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2}\right) + \frac{4}{6}\left(-\frac{2}{4}\log\frac{2}{4} - \frac{2}{4}\log\frac{2}{4}\right)$$

...

Continue for this branch

List all possible splits for this branch, calculate the entropy for each, and select the one with the min entropy

Alternative Splitting Criteria

Entropy at node m

$$I(m) = -\sum_{i=1}^{C} \underbrace{P_m(C_i)}_{\text{Probability of having i-th class at node m}} \log P_m(C_i)$$

Gini impurity at node m

$$I(m) = \sum_{i \neq j} P_m(C_i) \;\; P_m(C_j) = \frac{1}{2}\left[1 - \sum_i \left(P_m(C_i)\right)^2\right]$$

Misclassification impurity at node m

$$I(m) = 1 - \max_i \; P_m(C_i)$$

When to Stop Splitting

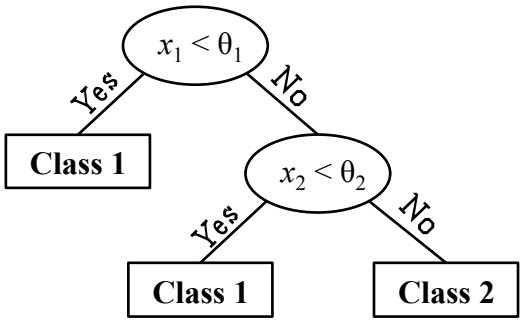
- Until all leaves are pure → **Overfitting**
- To prevent overfitting
  - Set a small threshold value in the reduction in impurity
  - Use cross validation techniques (e.g., continue splitting if the cross validation error is decreasing)
  - Use an explicit measure of the complexity to encode the training samples and the tree, stop growing when the encoding size is minimized (*minimum description length principle*)
  - Use statistical tests (e.g., use chi-squared statistic to understand if a split differs significantly from a random one)
- Then, it might be useful to keep the classes existing in a leaf together with their class probabilities

Pruning

- **Prepruning:** Stop growing the tree earlier before it overfits the training samples
- **Postpruning:** Grow the tree until it overfits the training samples (all leaves are pure) then prune the grown tree
  - *Reduced error pruning:* Remove nodes (or subtrees) only if the pruned tree performs no worse than the unpruned one over the validation set
  - *Rule post pruning:* Convert a tree into a set of rules and simplify (prune) each rule by removing any preconditions that result in no-worse-than validation performance

$$\text{if } \underbrace{(x_1 \geq \theta_1)}_{\mathbf{A}} \text{ and } \underbrace{(x_2 \geq \theta_2)}_{\mathbf{B}} \text{ then Class 2}$$

Try removing A or B and see what happens on the validation set

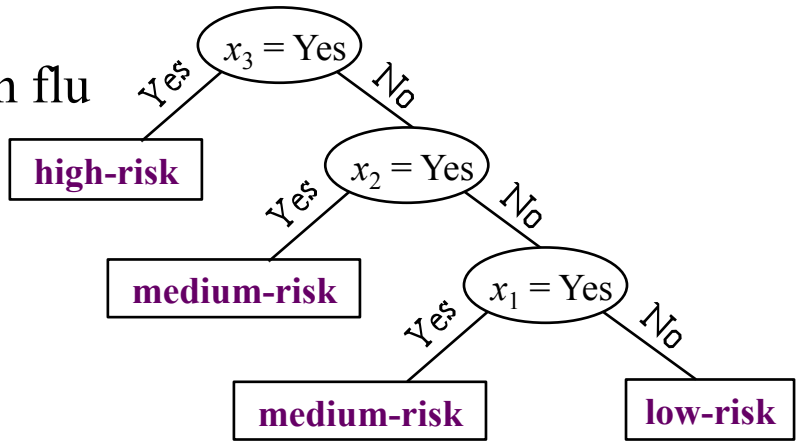


Rule Extraction

- One advantage of using a decision tree classifier is its ability to extract human interpretable rules

Consider the following problem setting in which we estimate the risk of getting avian flu

x_1: living close to migration routes (Yes / No)
x_2: feeding poultry (Yes / No)
x_3: contact with sick poultry (Yes / No)
x_4: gender (Male / Female)



Rule 1: if (contact = yes) then high-risk
Rule 2: if (contact = no) and (feeding = yes) then medium-risk
Rule 3: if (contact = no) and (feeding = no) and (close-living = yes) then medium-risk
Rule 4: if (contact = no) and (feeding = no) and (close-living = no) then low-risk

Rule support is the percentage of the training samples covered by the rule

Attributes with Differing Costs

- There is always trade-off between the classification accuracy and the cost of features used by the classification algorithm
  - More expensive features usually yield more accurate results

- One can build decision trees that are also sensitive to the cost of feature extraction by defining the splitting criterion accordingly

$$\text{Splitting criterion} = \frac{Gain(S, F)^2}{Cost(F)}$$ by Tan 93

$$\text{Splitting criterion} = \frac{2^{Gain(S, F)} - 1}{(Cost(F) + 1)^w}$$ by Nunez 98

$$Gain(S, F) = entropy(before) - entropy(after)$$

Missing Values

If a feature value is missing in a training instance

- Assign a value to this feature in entropy calculation
  - The most common value among training instances at the current node
  - The most common value among training instances at the current node that belong to the same class
- Assign a probability to every possible value of that feature and consider it as a set of fractional instances
  - Probabilities can be estimated based on the frequencies of that feature's values among training instances at the current node

If it is missing in an unseen instance

- A set of fractional instances can be used
  - The final decision is a weighted sum of the decisions of every reached leaf
- Estimate the missing value using the existing features

Regression Trees

- Continuous outputs at leaves (instead of class labels)
- Error measure is used for the goodness of a split (instead of an impurity measure)
- Iteratively grow the tree until the error measure falls below a certain threshold

$$\underbrace{E(m)}_{\textbf{Mean square error at node m}} = \frac{1}{|D_m|} \sum_{x_i \in D_m} (y_i - f_m)^2$$

training samples at node m

estimated output at node m

$$\underbrace{E(S)}_{\textbf{Mean square error of a binary split S}} = P_{left}\, E(left) + P_{right}\, E(right)$$

To estimate f_m

The mean (median) over the outputs of the training samples at node m could be used (piecewise constant approx.)

A linear function is fit over the outputs of the training samples at node m and its output value could be used (piecewise linear approx.)

Regression Trees

Exercise: Show how to construct a regression tree for the training samples given below (show how the selection of the stopping threshold affects the constructed tree)