CS57300
PURDUE UNIVERSITY

FEBRUARY 7, 2019

DATA MINING

DECISION TREE

BUILDING TREE RECURSIVELY

Buildtree(examples, attributes)

/*examples: a list of training examples at the current node
attributes: a set of candidate attributes to place question on*/

If examples={} **then** return

If examples have the same label *y* **then** return a leaf node with label *y*

If attributes={} **then** return a leaf node with the majority label in examples

A = Best_attribute(examples, attributes) /*Suppose attribute *A* has *n* possible values*/

Create an internal node, *node(A)*, with *n* children

For attribute *A*'s *i*-th possible value *A(i)*:

The i-th child of node(A) = **Buildtree**({examples with its value on *A* being *A(i)*}, attributes-{*A*})

SELECTING THE BEST ATTRIBUTE

- Information gain $Gain(S, A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$
- Gini gain $Gain(S, A) = Gini(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} Gini(S_v)$
- Chi-square score $\chi^2 = \sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i}$

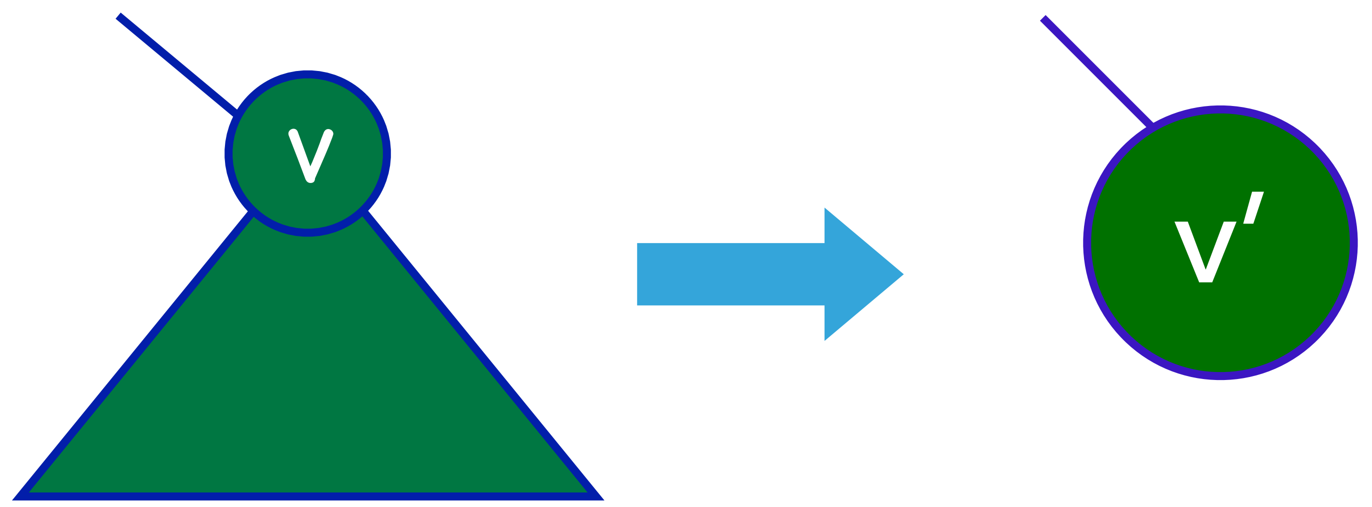
WHEN TO STOP GROWING

- Full growth methods
  - There are no examples left
  - All examples at a node belong to the same class
  - There are no attributes left for further splits
- What impact does this have on the quality of the learned trees?
  - Trees **overfit** the training data and accuracy on testing data suffers

HOW TO AVOID OVERFITTING IN DECISION TREES

- Post-pruning
  - Separate the training data into a training set and a validation set (i.e., a pruning set).
  - Fully grow a tree
  - Use the pruning set to evaluate the utility of pruning (i.e. deleting) nodes from the tree
- Pre-pruning
  - Apply a statistical test to decide whether to expand a node
  - Add penalty terms in scoring functions to prefer trees with smaller sizes

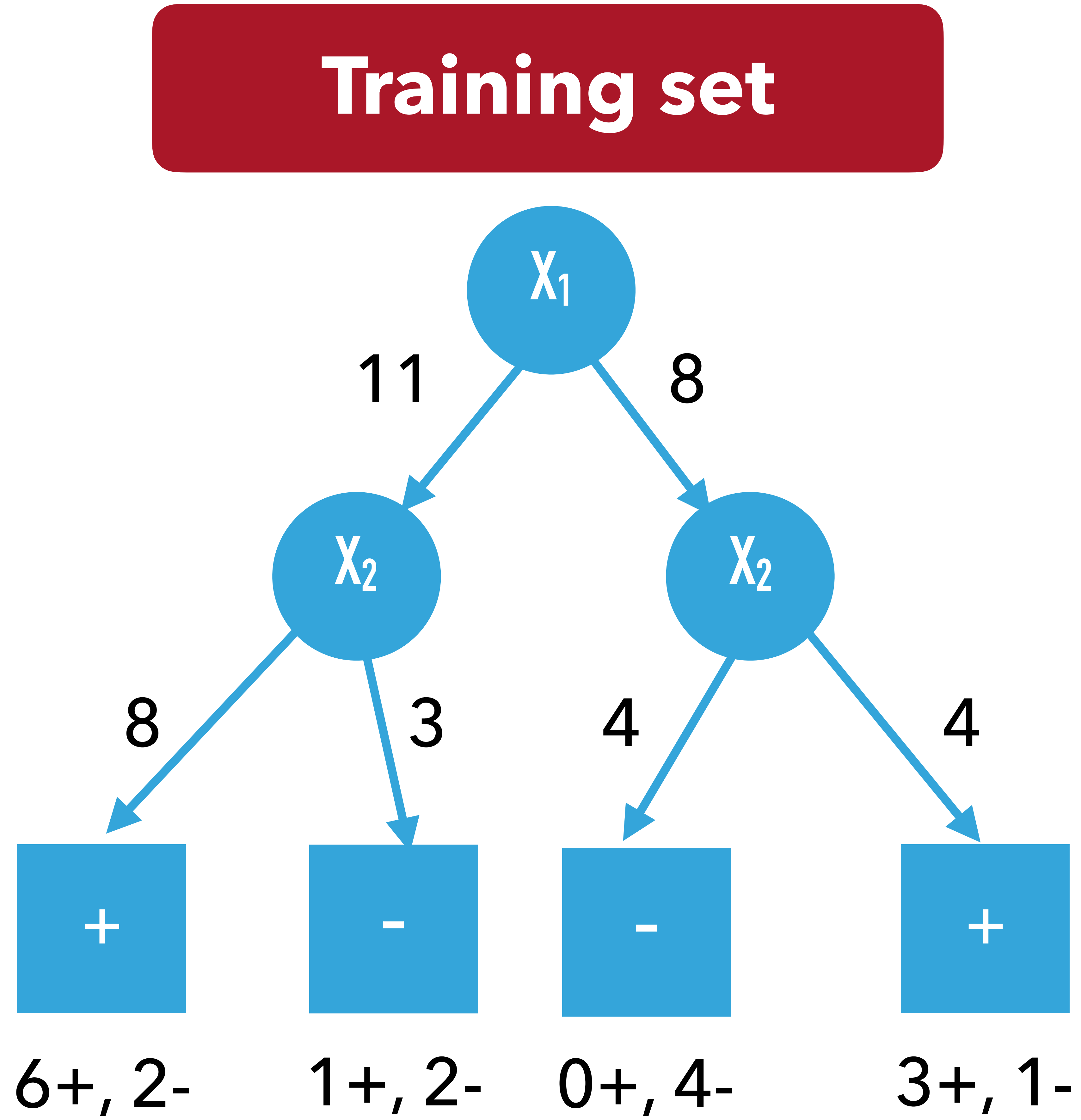
POST-PRUNING METHOD: REDUCED ERROR PRUNING



- Grow a full tree *T* using the **training set**
- Let *v* be an internal node of the current tree *T*
- If we prune at *v* to create a new tree *T′*, in *T′*, the subtree rooted at *v* will be replaced by a leaf node *v′*, whose label is the majority label for all **training examples** fall under *v*
- Define the gain of pruning at *v* as, in the **pruning set**, # of misclassified examples under *v* (in *T*) - # of misclassified examples that in *v′* (in *T′*)
- Repeat: Prune at node with largest gain until only negative gain nodes remain

Source: www.ailab.si/blaz/predavanja/uisp/slides/uisp05-PostPruning.ppt

REDUCED ERROR PRUNING EXAMPLE

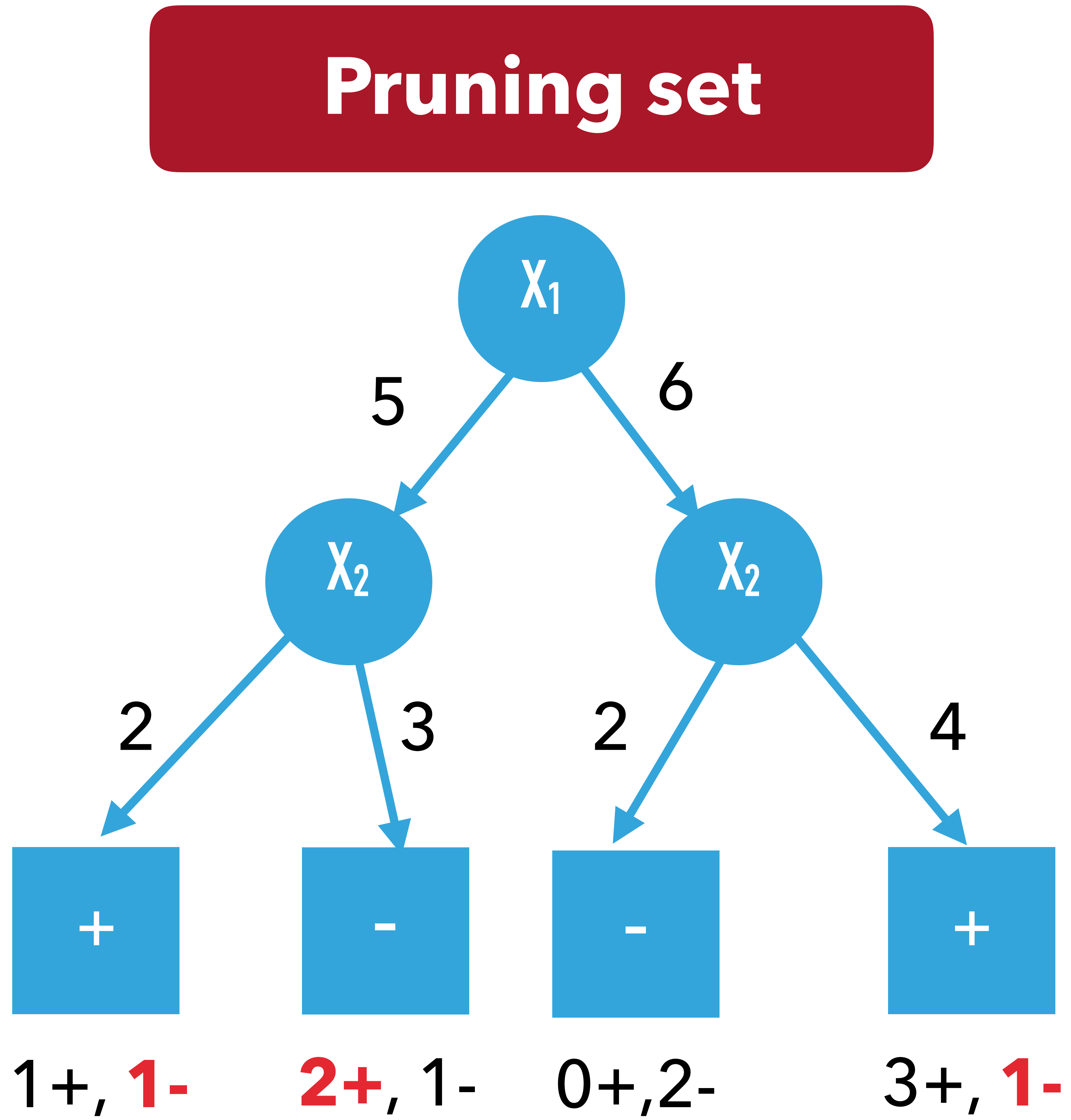


Gain at node X_2 (left): 3-2=1

Gain at node X_2 (right): 1-3=-2

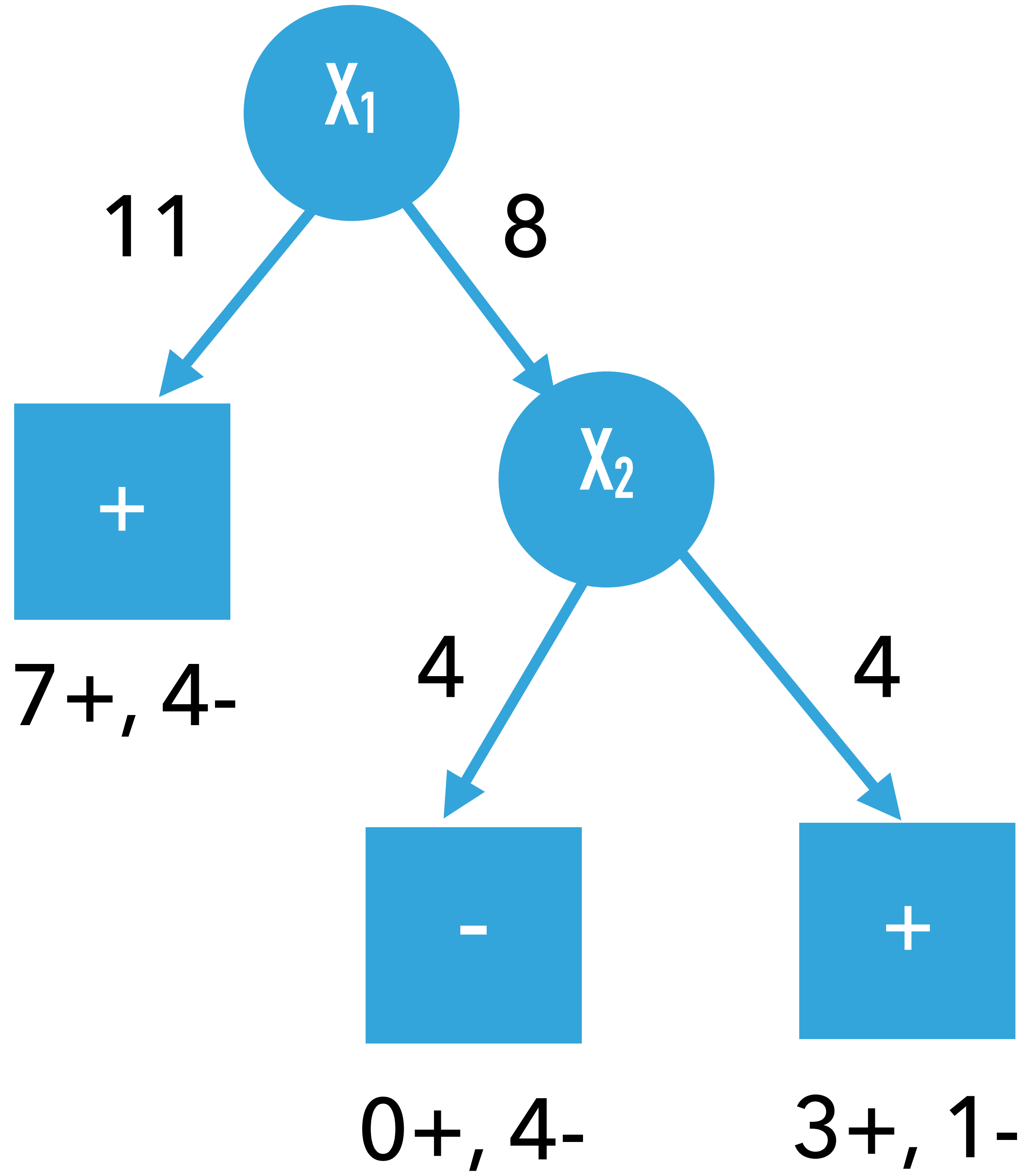
Gain at node X_1: 4-5=-1

Prune at node X_2 (left)!



REDUCED ERROR PRUNING EXAMPLE

Training set

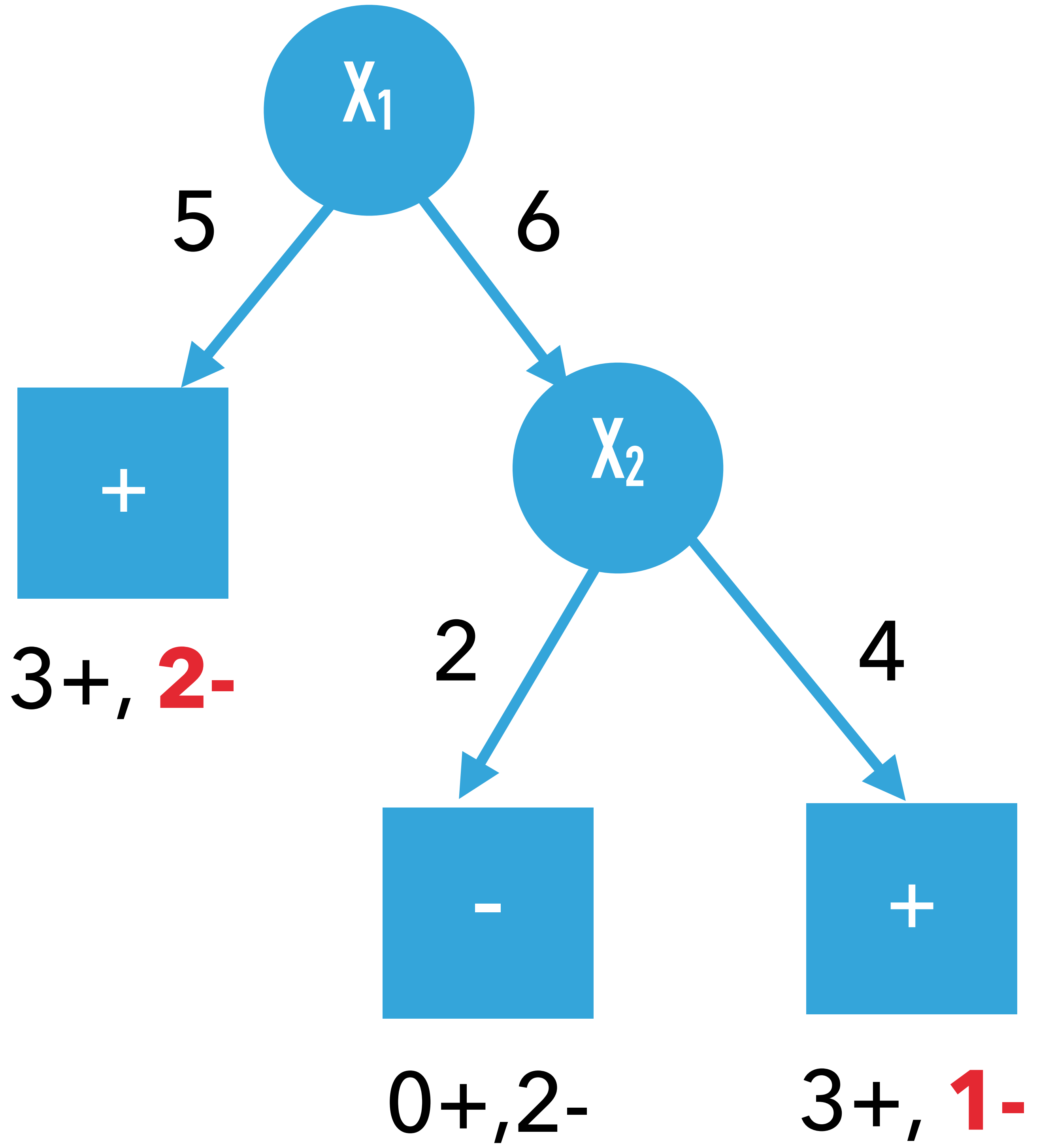


Gain at node X_2: 1-3=-2

Gain at node X_1: 3-5=-2

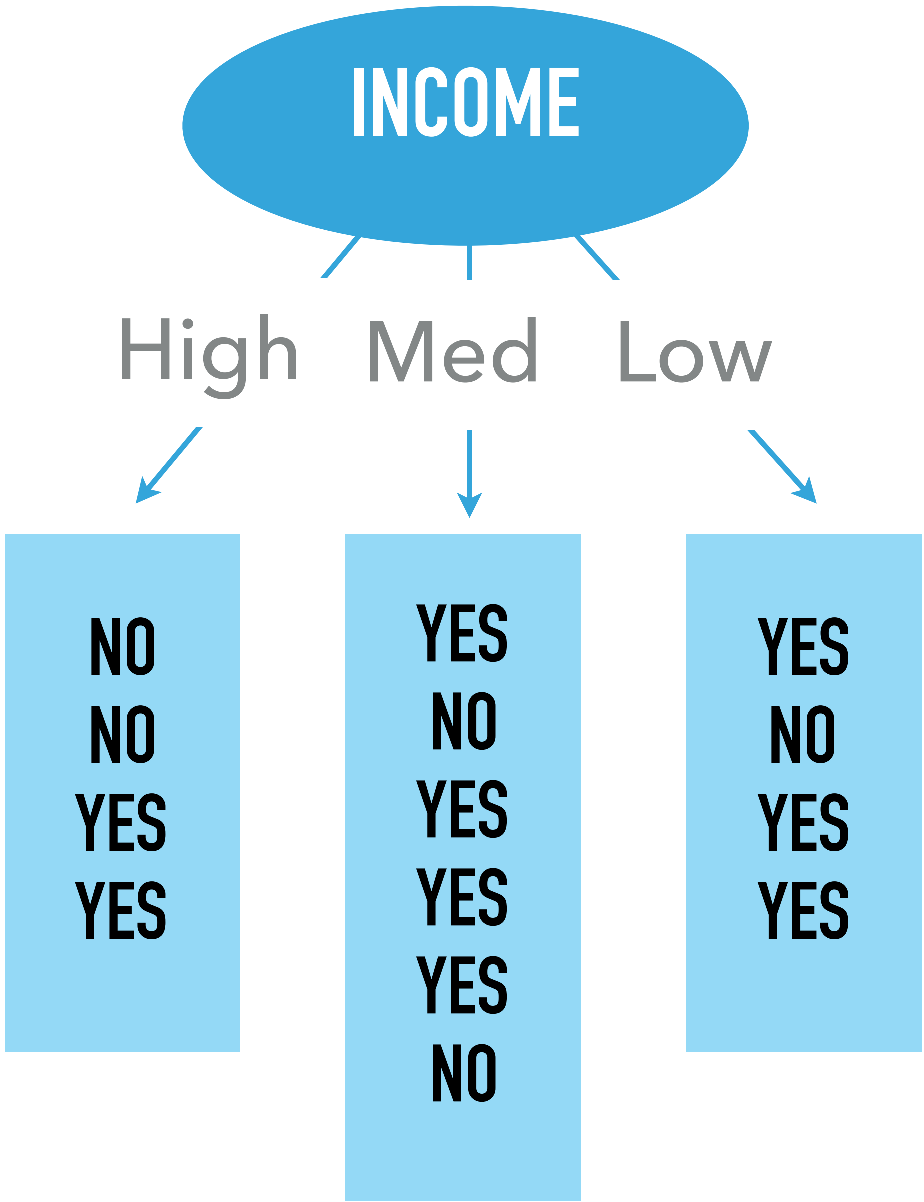
Done!

Pruning set



PRE-PRUNING METHODS

- Stop growing tree at some point during top-down construction when there is no longer sufficient data to make reliable decisions



	Buy	No buy	
High	2	2	4
Med	4	2	6
Low	3	1	4
	9	5	14

Gain(S,Income)= 0.029

Gini-Gain (S,Income)= 0.020

$$\chi^2 = 0.57$$

IS THIS SPLIT REALLY MEANINGFUL?

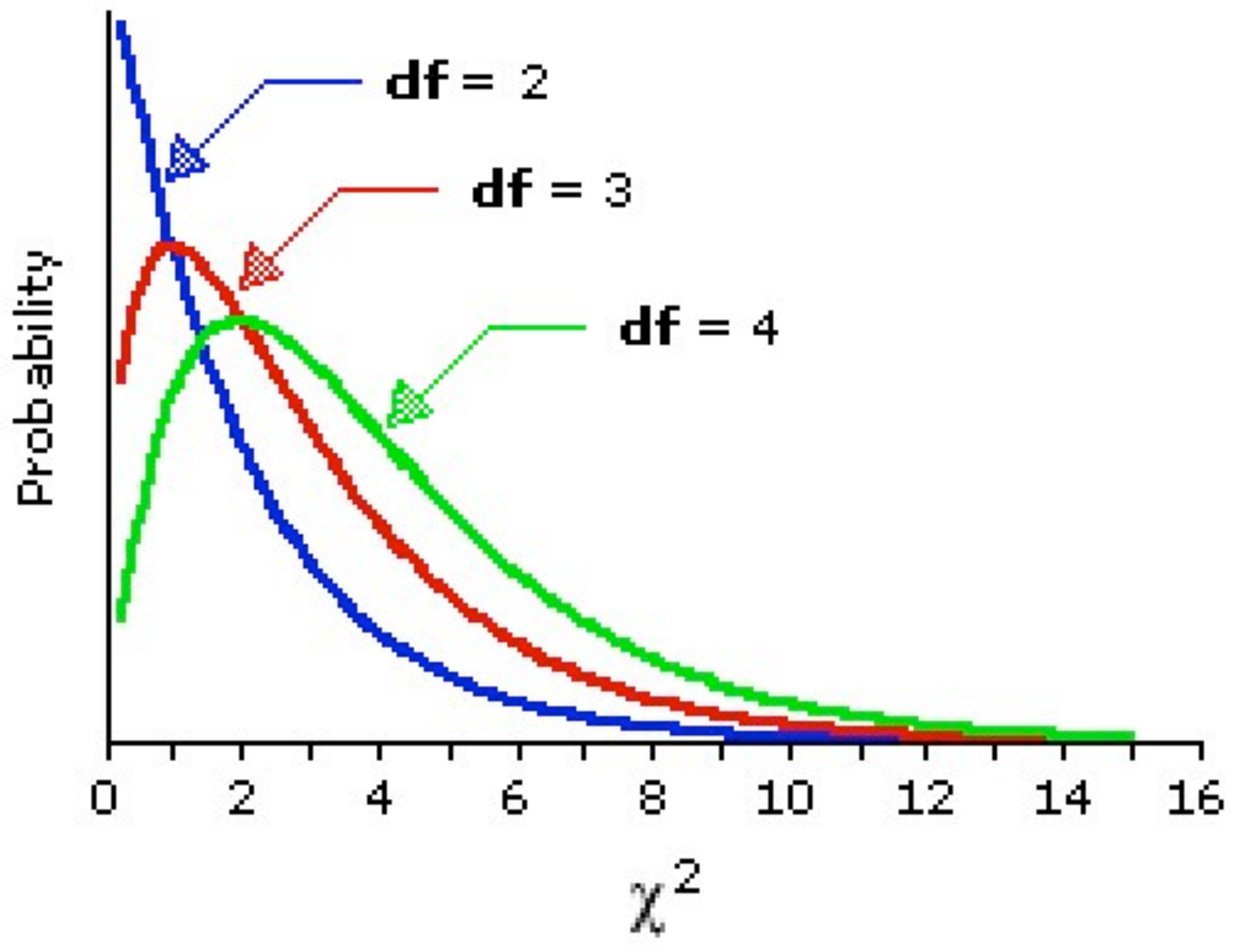
PRE-PRUNING METHODS

- Approach:
  - Choose threshold on feature score (e.g., information gain, gini gain)
  - Stop splitting if the best feature score is below threshold
  - Threshold can be decided through significance in statistical test or cross validation

EXAMPLE: DETERMINE CHI-SQUARE THRESHOLD ANALYTICALLY

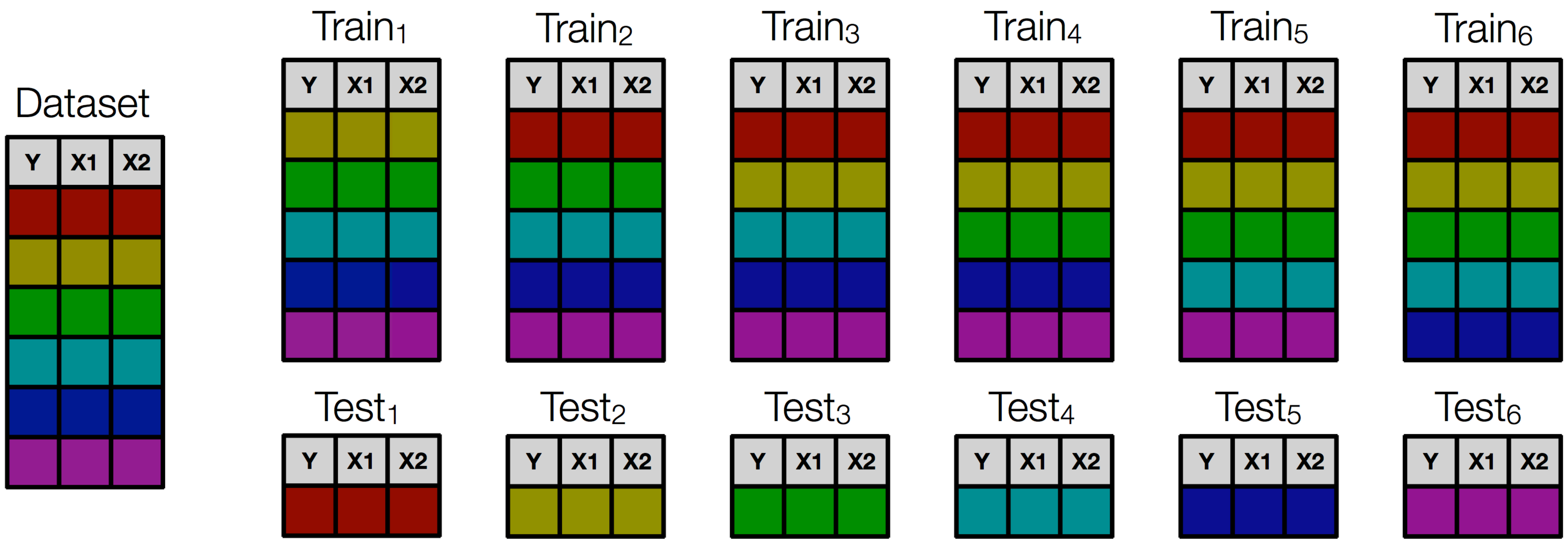
- Chi-square has known sampling distribution, can look up significance threshold
    - Degrees of freedom= (#rows-1)(#cols-1)
    - 3*2 table:
      5.99 is 95% critical value
- Stop growing when chi-square feature score is not **statistically significant**

$$\chi^2 = \sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i}$$



K-FOLD CROSS VALIDATION

- Randomly **partition** training data into k folds
- For i=1 to k
  - Learn model on D - ith fold; evaluate model on ith fold
- Average results from all k trials



EXAMPLE: CHOOSING A GINI THRESHOLD WITH CROSS VALIDATION

Smaller threshold means the tree would be complex as we would keep growing the tree until we reach below that threshold value

- For i in 1.. k
  - For t in threshold set (e.g, [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8])
    - Learn decision tree on $Train_i$ with Gini gain threshold t (i.e. stop growing when max Gini gain is less than t)
    - Evaluate learned tree on $Test_i$ (e.g., with accuracy)
  - Set $t_{max,i}$ to be the t with best performance on $Test_i$
- Set t_{max} to the average of $t_{max,i}$ over the k trials
- Relearn the tree on all the data using t_{max} as Gini gain threshold

ALGORITHM COMPARISON

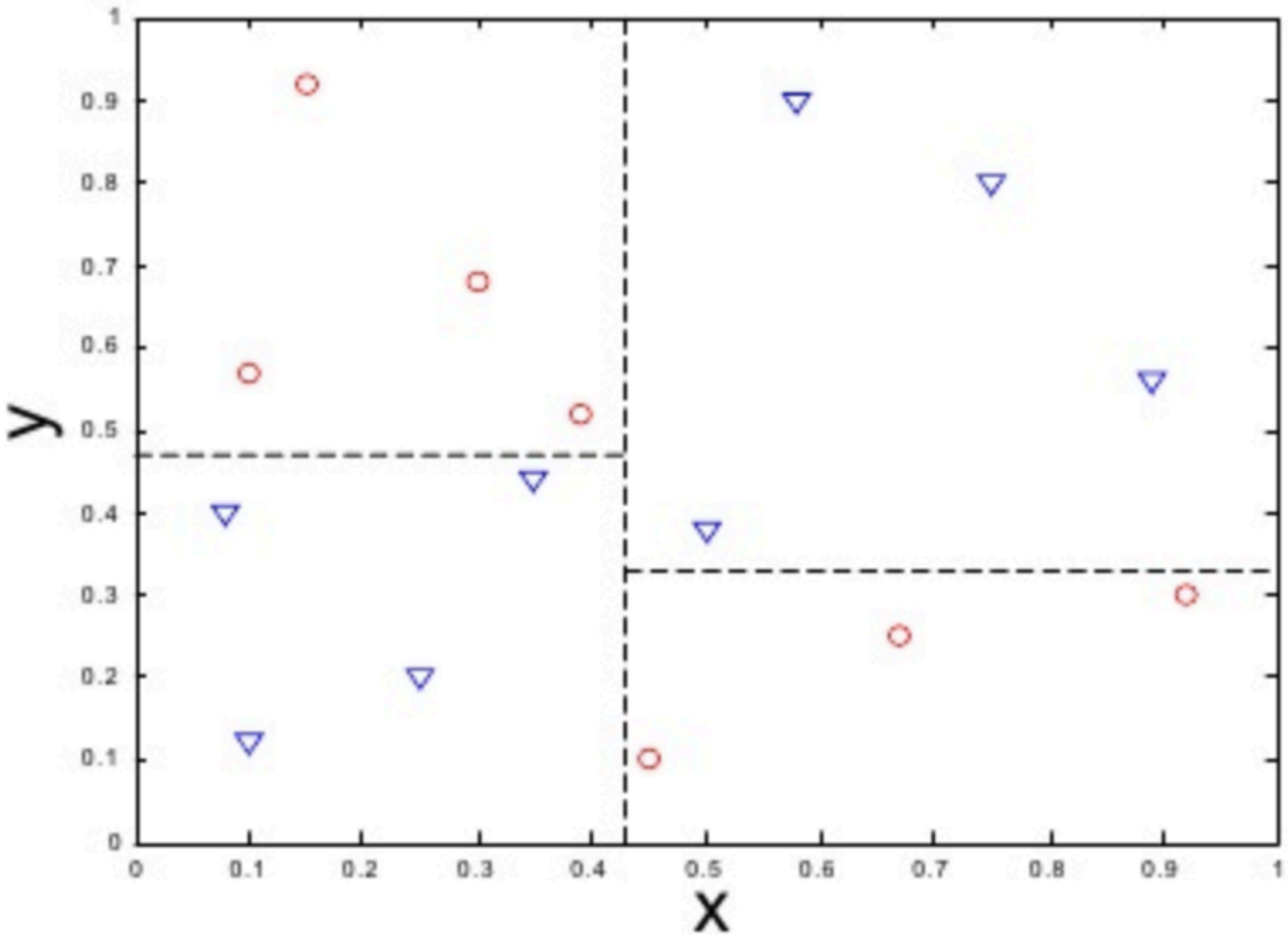
- CART
  - Evaluation criterion: **Gini gain**
  - Search algorithm: Heuristic, greedy search
  - Pruning mechanism: **Cross-validation to select gini threshold**
- C4.5
  - Evaluation criterion: **Information gain**
  - Search algorithm: Heuristic, greedy search
  - Pruning mechanism: **Reduce error pruning**

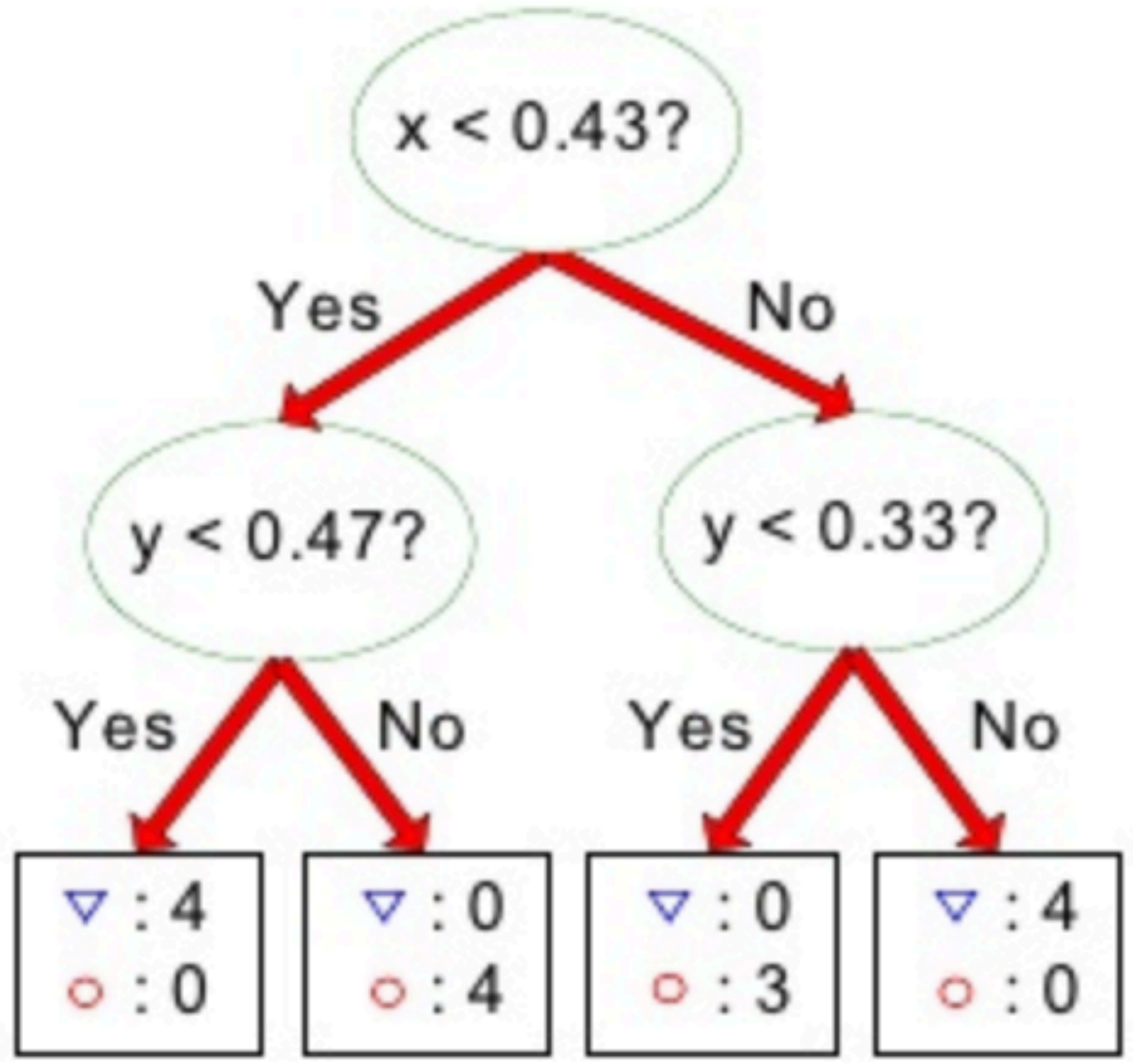
NAIVE BAYES VS. DECISION TREES

- Naive Bayes
  - **Probabilistic classification**: output posterior class distribution p(y|**x**), and model the underlying probability distributions
  - Parametric model
  - Model space: parameters in prior distributions p(y) and conditional distributions p(**x**|y)
  - Scoring function: likelihood function / posterior probability of observing the data
  - Search: Convex optimization

- Decision trees
  - **Discriminative classification**: output class labels and model the decision boundary directly
  - Non-parametric model
  - Model space: all possible trees that can be generated from the set of attributes: different attribute to use on each node, different ways to split continuous variables into intervals, different depth of the tree, etc.
  - Scoring function: misclassification rate
  - Search: Greedy, heuristic search

DECISION TREES MODEL DECISION BOUNDARIES





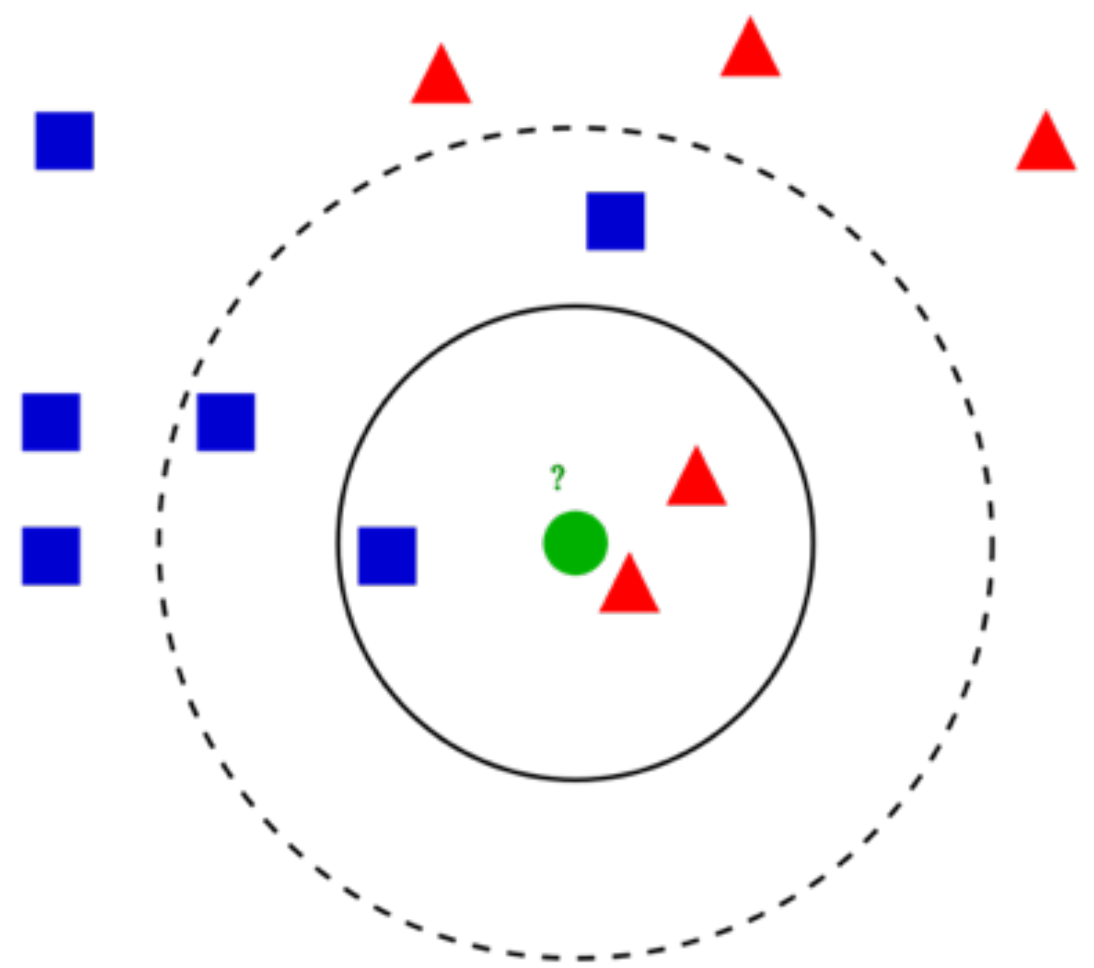
NEAREST NEIGHBOR

NEAREST NEIGHBOR

- Discriminative classification, non-parametric, instance-based method
- Assumes that all points are represented in *p*-dimensional space
- Learning
  - Stores (i.e., memorizes) all the training data
- Prediction
  - Look for "nearby" training examples
  - Classification is made based on class labels of neighbors

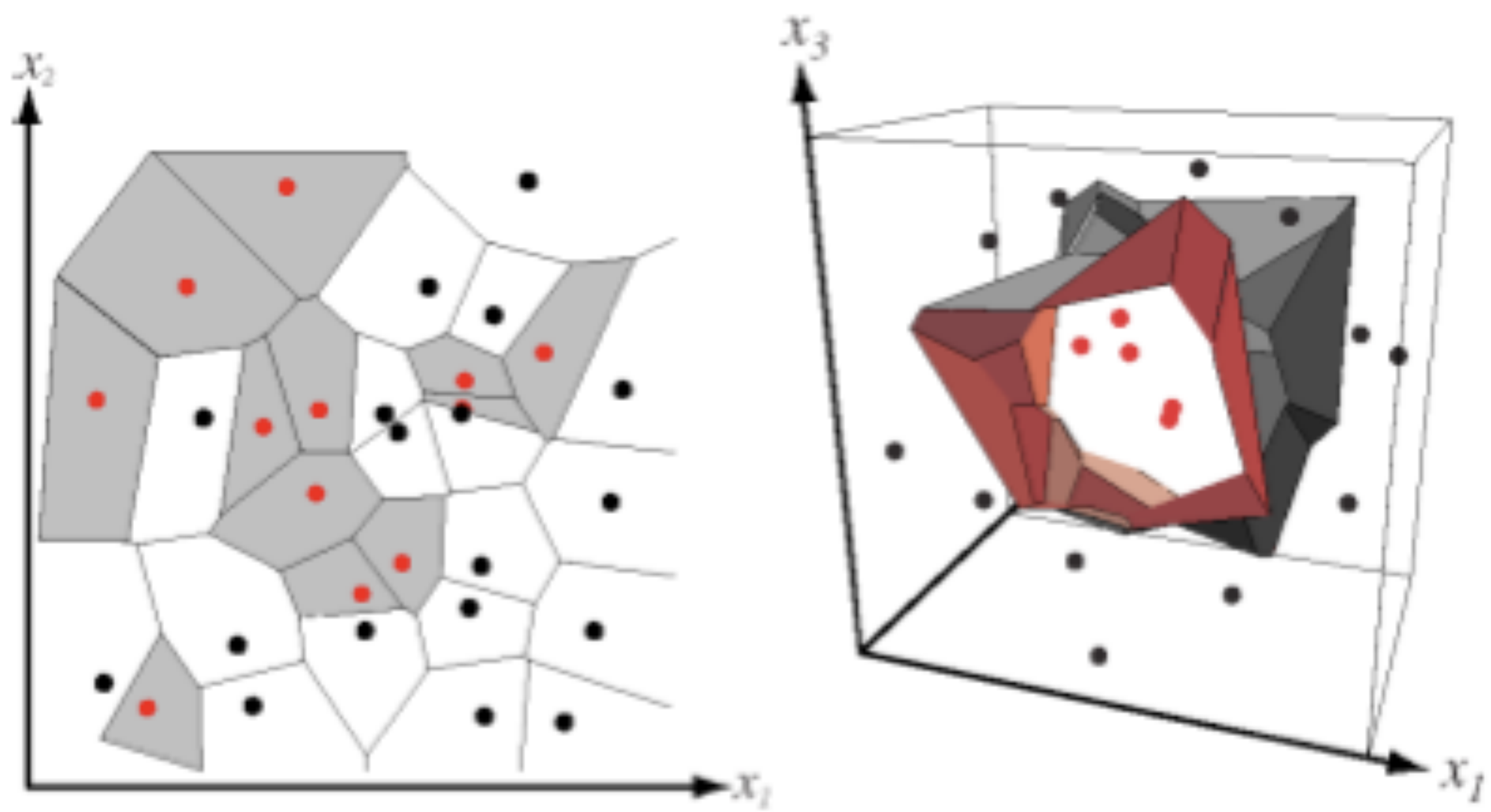
FROM 1NN TO KNN

- Training set: $(\mathbf{x}_1,y_1)$, $(\mathbf{x}_2,y_2)$, …, $(\mathbf{x}_n,y_n)$ where $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{ip}]$ is a feature vector of *p* attributes and y_i is a discrete class label
- To predict a class label for new instance *j*: Find the training instance point $\mathbf{x}_i$ such that $d(\mathbf{x}_i, \mathbf{x}_j)$ is minimized; Let $f(\mathbf{x}_j)= y_i$
- Key idea: Find instances that are "similar" to the new instance and use their class labels to make prediction for the new instance
  - 1NN generalizes to KNN when labels of the *K* nearest neighbors are considered; Let $f(\mathbf{x}_j)=g(\{y_i\})$, g() is an aggregation function such as majority vote.



1NN DECISION BOUNDARY

- For each training example *i*, we can calculate its **Voronoi cell**, which corresponds to the space of points for which i is their nearest neighbor
- All points in such a Voronoi cell are labeled by the class of the training point, forming a Voronoi tessellation of the feature space



Source: http://www.cs.bilkent.edu.tr/~saksoy/courses/cs551-Spring2008/slides/cs551_nonbayesian1.pdf

NEAREST NEIGHBOR: MODEL SPACE

- How many neighbors to consider (i.e., choice of *K*)?
  … Usually a small value is used, e.g. $K<10$
- What distance measure *d*() to use?
  … Euclidean L_2 distance is often used
- What function *g*() to combine the neighbors' labels into a prediction?
  … Majority vote is often used