CS150: Database & Datamining

Lecture 23: Analytics & Machine Learning V

Xuming He

Spring 2019

Acknowledgement: Slides are adopted from the Berkeley course CS186 by Joey Gonzalez and Joe Hellerstein, Toronto CSC411 by Rich Zemel, Caltech CS155 by Yisong Yue.

Taxonomy
of Machine Learning

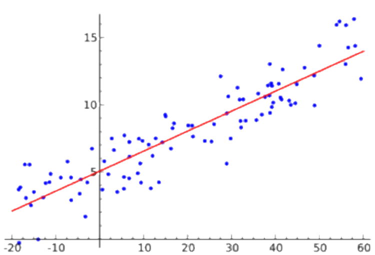
Labeled Data

Indirect (reward)

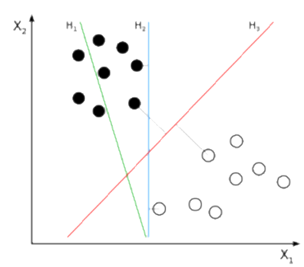
Unlabeled Data

Supervised Learning

Regression



Classification

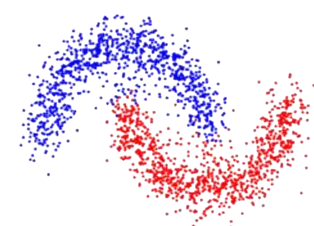


Reinforcement & Bandit Learning

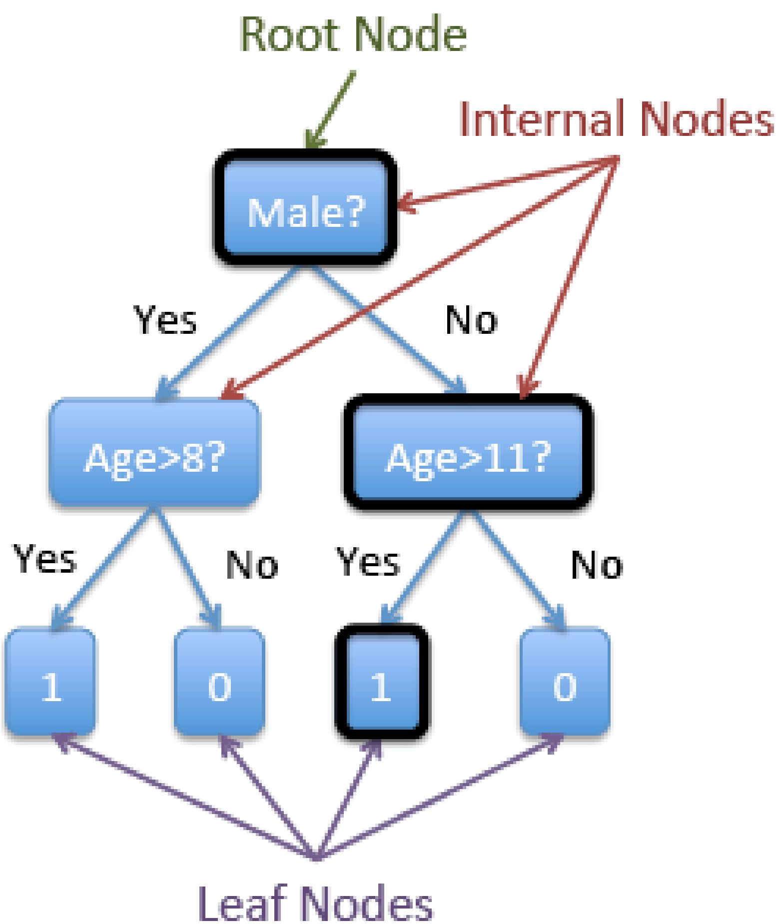
Unsupervised Learning

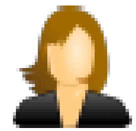
Dimensionality Reduction

Clustering



Decision tree



Input: 

Alice
Gender: Female
Age: 14

Prediction: Height > 55”

Every **internal node** has a **binary** query function q(x).

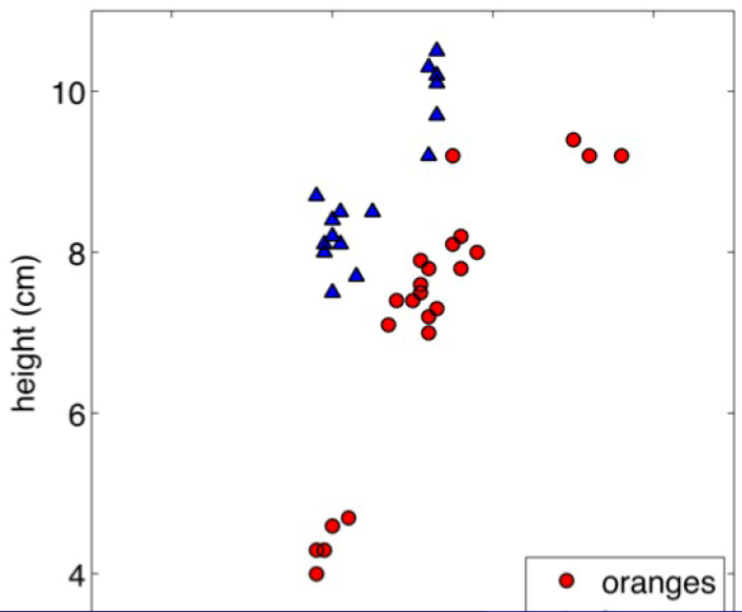
Every **leaf node** has a prediction, e.g., 0 or 1.

Prediction starts at **root node.**
Recursively calls query function.
Positive response → Left Child.
Negative response → Right Child.
Repeat until Leaf Node.

Decision tree

➢General idea

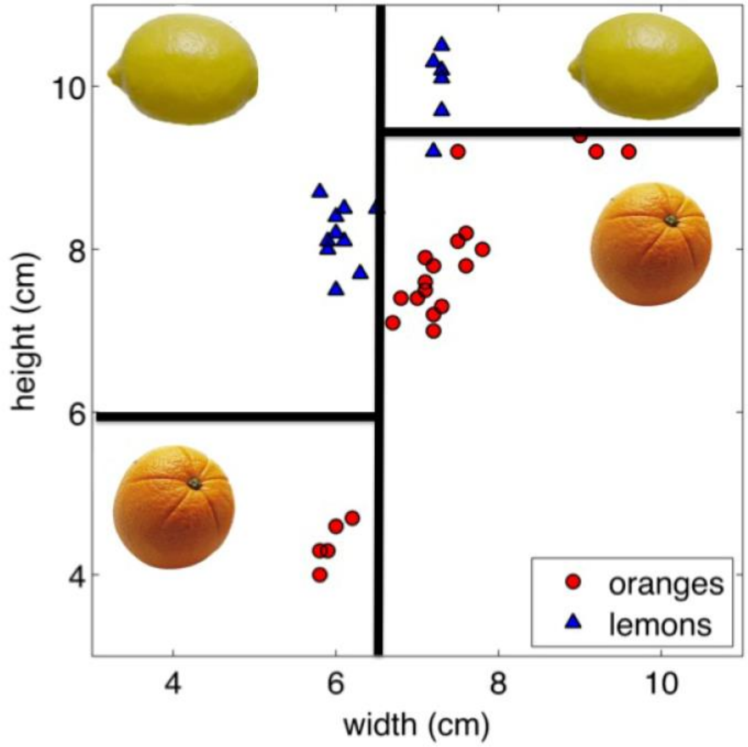
- Pick an attribute, do a simple test
- Conditioned on a choice, pick another attribute, do another test
- In the leaves, assign a class with majority vote
- Do other branches as well

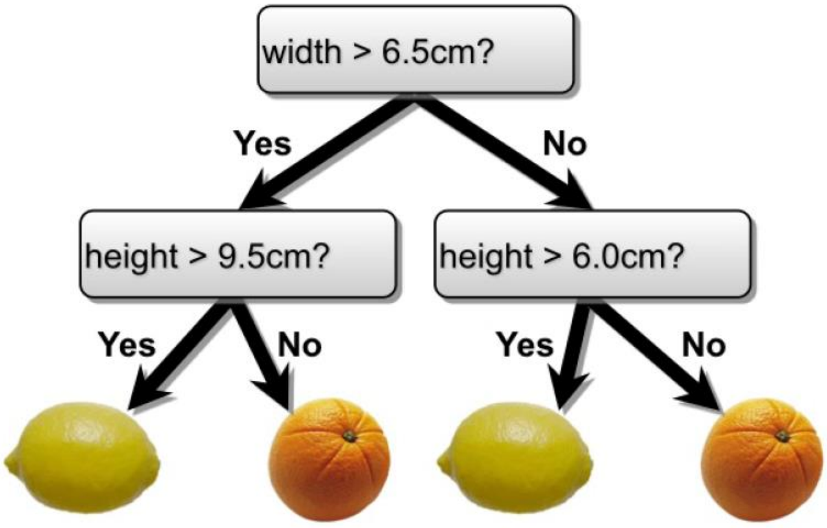


Another example

➢ Classify fruits

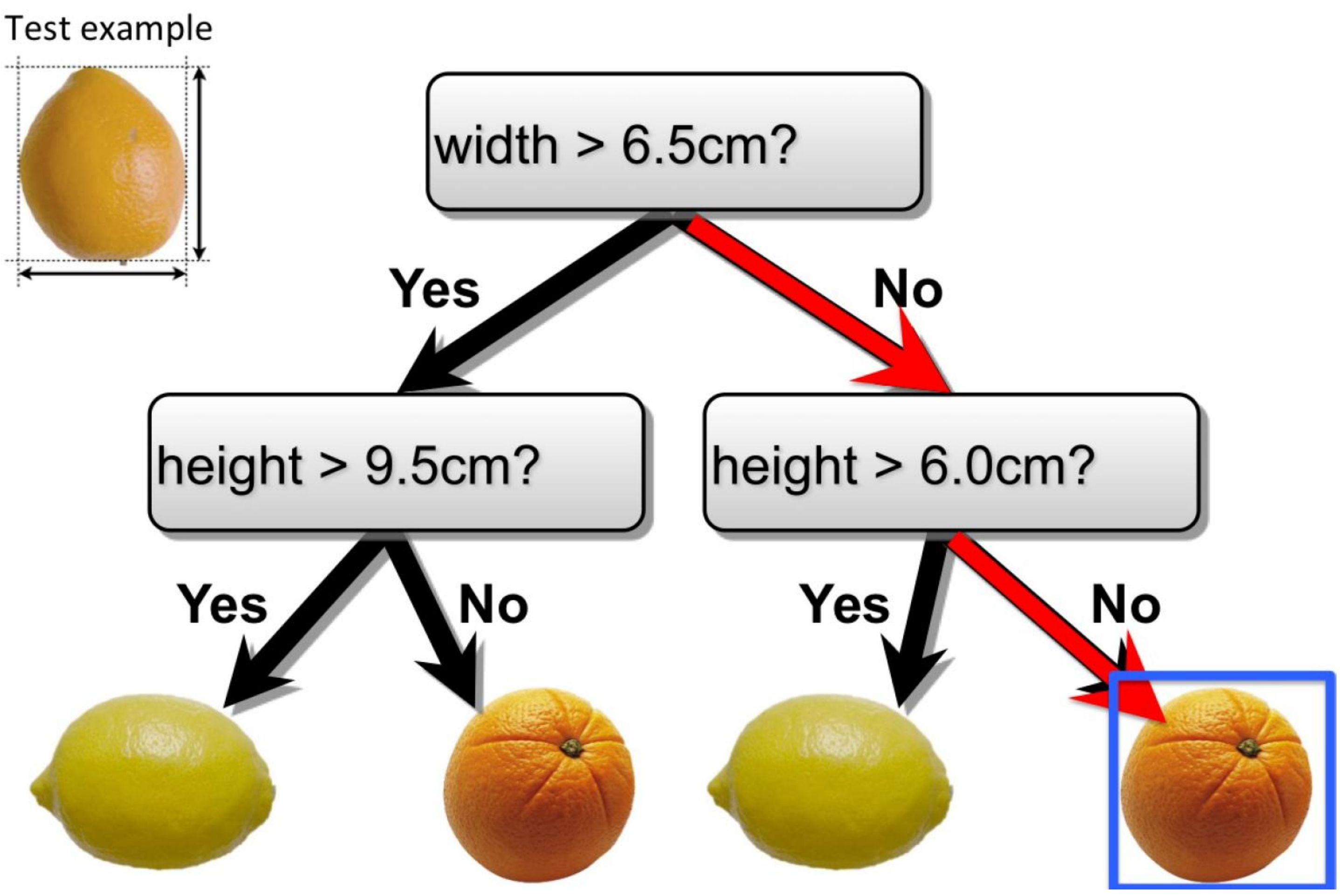
- Gives axes aligned decision boundaries



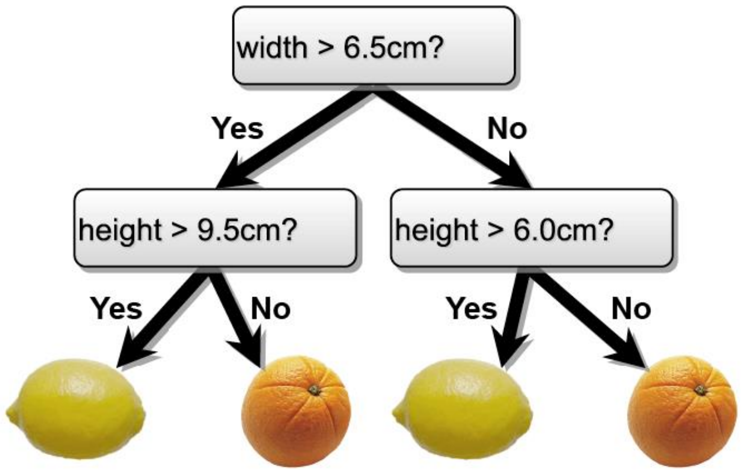


Another example

- ➢ Classify fruits



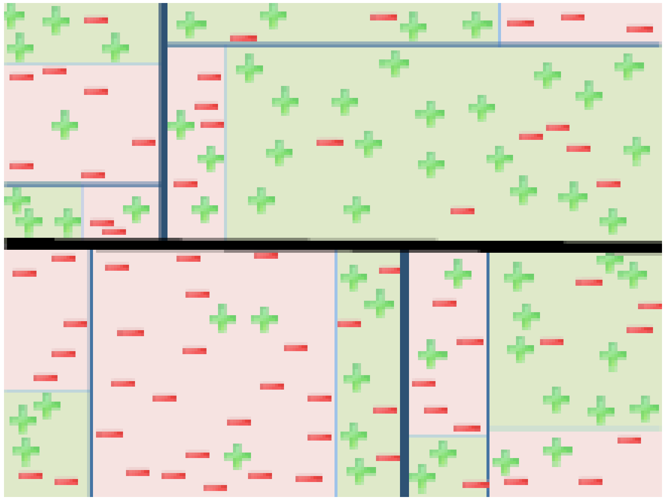
Decision tree



- Internal nodes test attributes
- Branching is determined by attribute value
- Leaf nodes are outputs (class assignments)

Decision tree function class

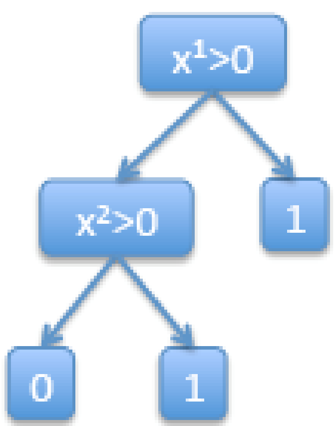
- "Piece-wise Static" Function Class
  - All possible partitionings over feature space.
  - Each partition has a static prediction.
- Partitions axis-aligned
  - E.g., No Diagonals
- (Extensions next week)



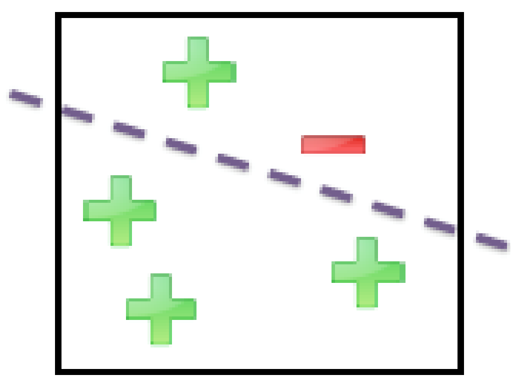
Decision tree vs Linear model

- Decision Trees are NON-LINEAR Models!

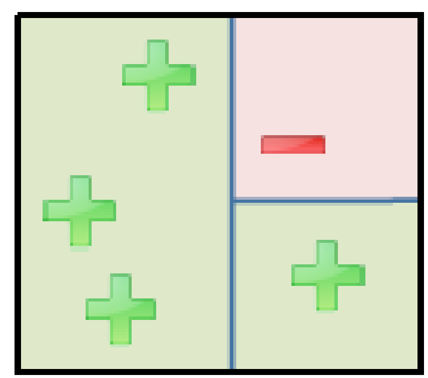
- Example:



No Linear Model
Can Achieve 0 Error



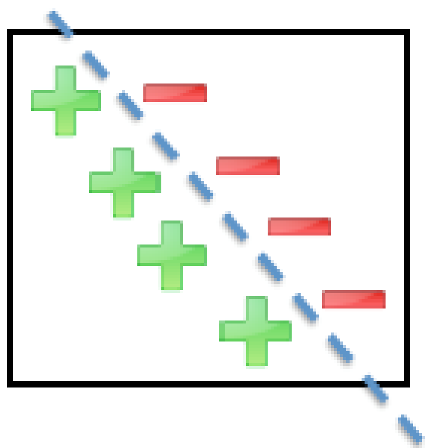
Simple Decision Tree
Can Achieve 0 Error



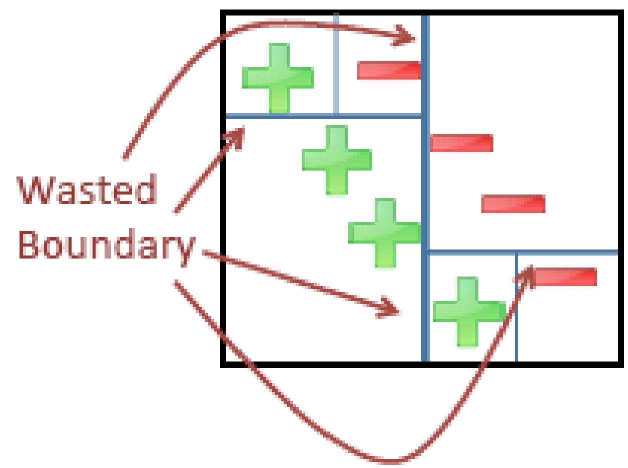
Decision tree vs Linear model

- Decision Trees are AXIS-ALIGNED!
  - Cannot easily model diagonal boundaries
- Example:

Simple Linear SVM can Easily Find Max Margin



Decision Trees Require Complex Axis-Aligned Partitioning



Decision tree: inference

➢ Classification and Regression

- Each path from root to a leaf defines a region R_m of input space
- Let $\{(x^{(m_1)}, t^{(m_1)}), \dots, (x^{(m_k)}, t^{(m_k)})\}$ be the training examples that fall into R_m
- **Classification tree**:
  - discrete output
  - leaf value y^m typically set to the most common value in $\{t^{(m_1)}, \dots, t^{(m_k)}\}$
- **Regression tree**:
  - continuous output
  - leaf value y^m typically set to the mean value in $\{t^{(m_1)}, \dots, t^{(m_k)}\}$

Note: We will only talk about classification

Decision tree: Learning

➢ How do we construct a useful decision tree?

Learning the simplest (smallest) decision tree is an NP complete problem [if you are interested, check: Hyafil & Rivest'76]

- Resort to a greedy heuristic:
  - Start from an empty decision tree
  - Split on next best attribute
  - Recurse
- What is best attribute?

Decision tree: training example

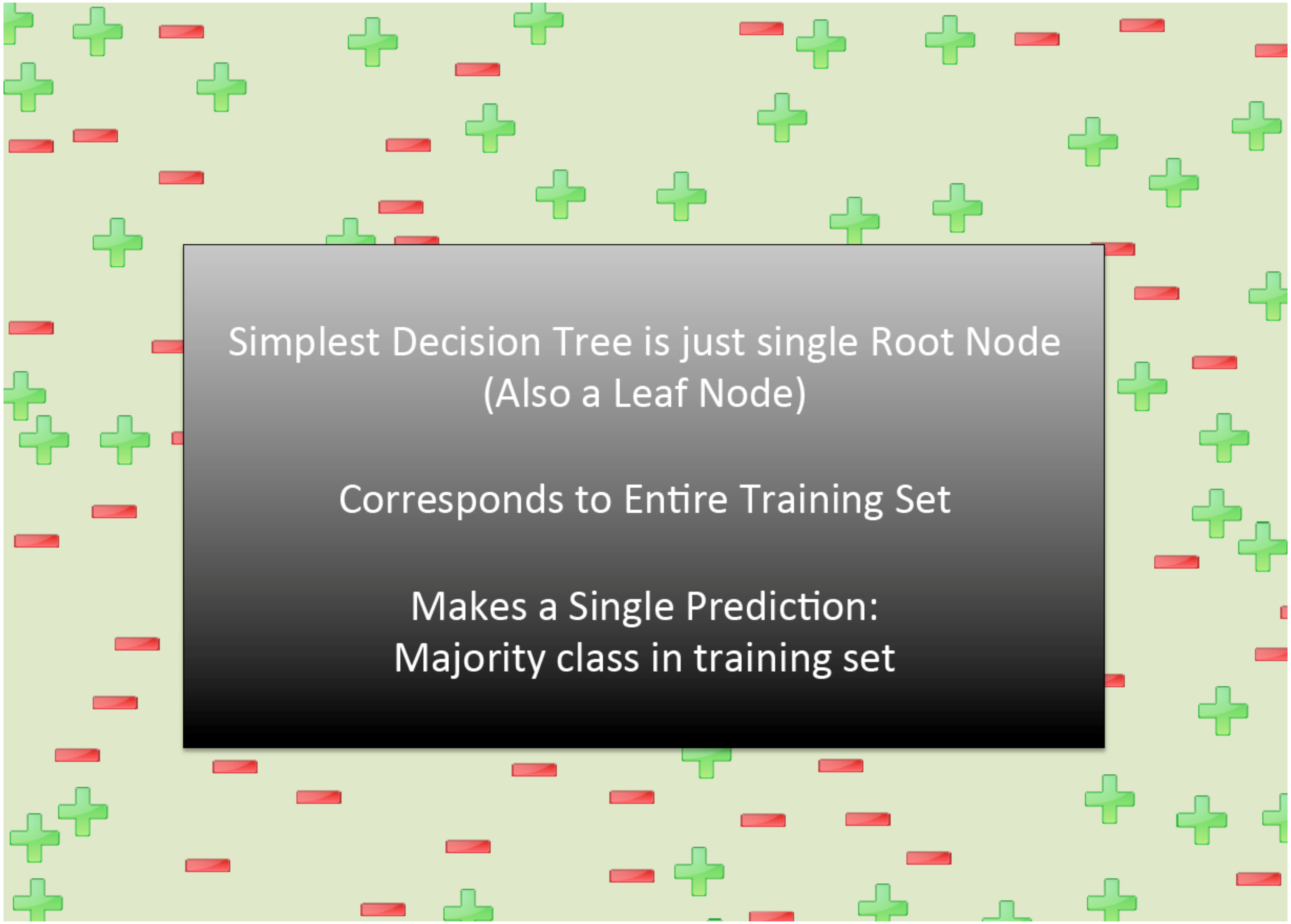
- What if just one node?
  - (I.e., just root node)
  - No queries
  - Single prediction for all data

Name	Age	Male?	Height > 55"
Alice	14	0	1
Bob	10	1	1
Carol	13	0	1
Dave	8	1	0
Erin	11	0	0
Frank	9	1	1
Gena	10	0	0

x

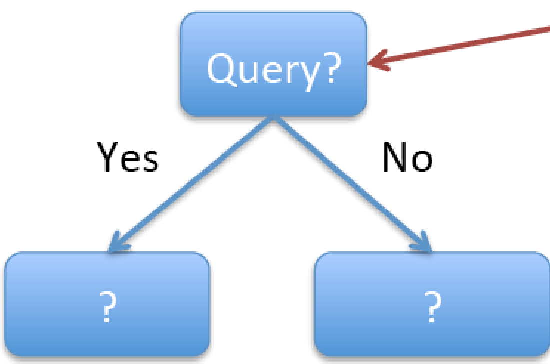
y

Decision tree: training example



Decision tree: training example

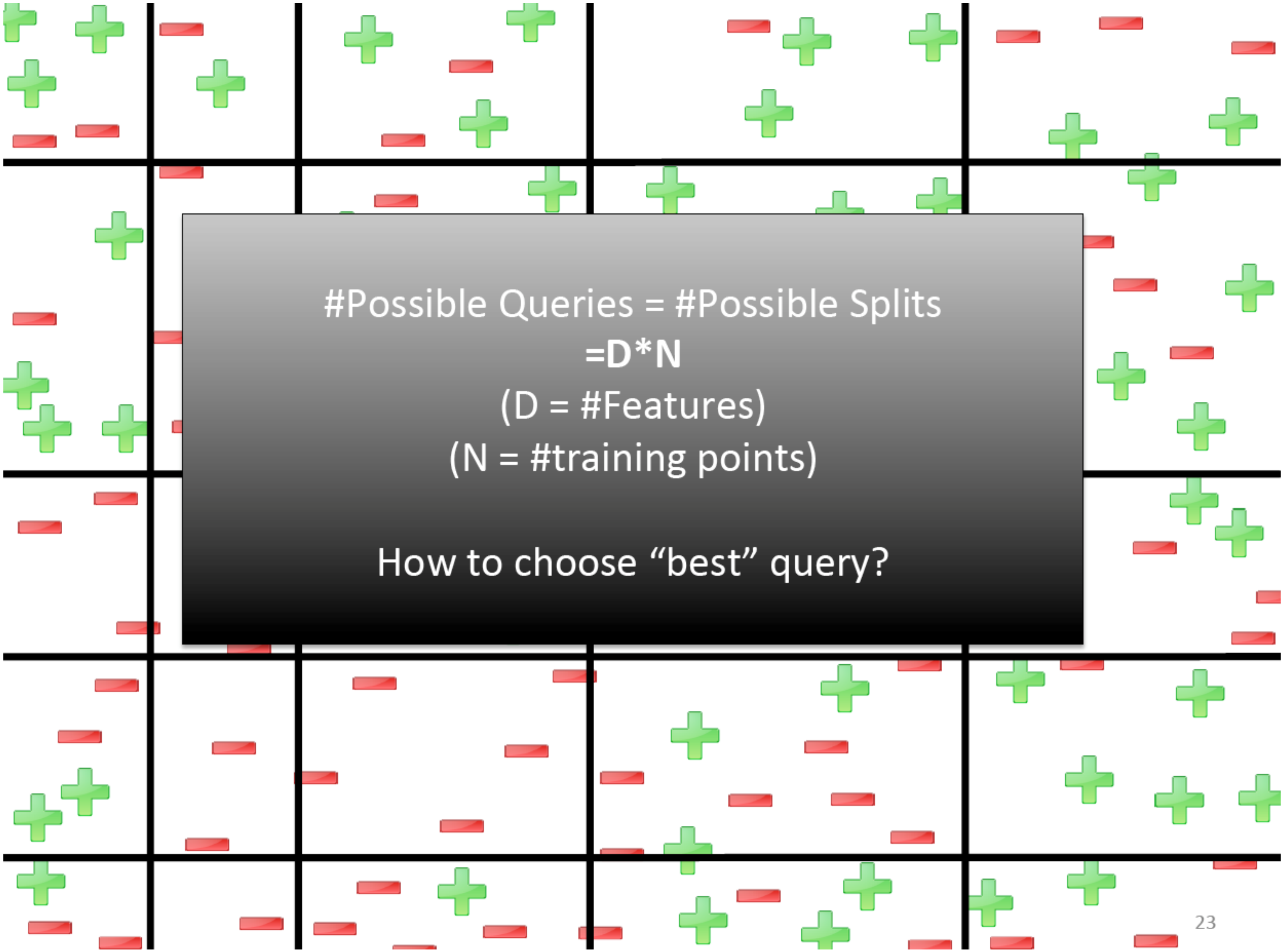
- What if 2 Levels?
  - (I.e., root node + 2 children)
  - Single query (which one?)
  - 2 predictions
  - **How many possible queries?**



Name	Age	Male?	Height > 55”
Alice	14	0	1
Bob	10	1	1
Carol	13	0	1
Dave	8	1	0
Erin	11	0	0
Frank	9	1	1
Gena	10	0	0

$\underbrace{\text{Age, Male?}}_{x}$ $\underbrace{\text{Height > 55”}}_{y}$

Decision tree: training example



#Possible Queries = #Possible Splits

=D*N

(D = #Features)

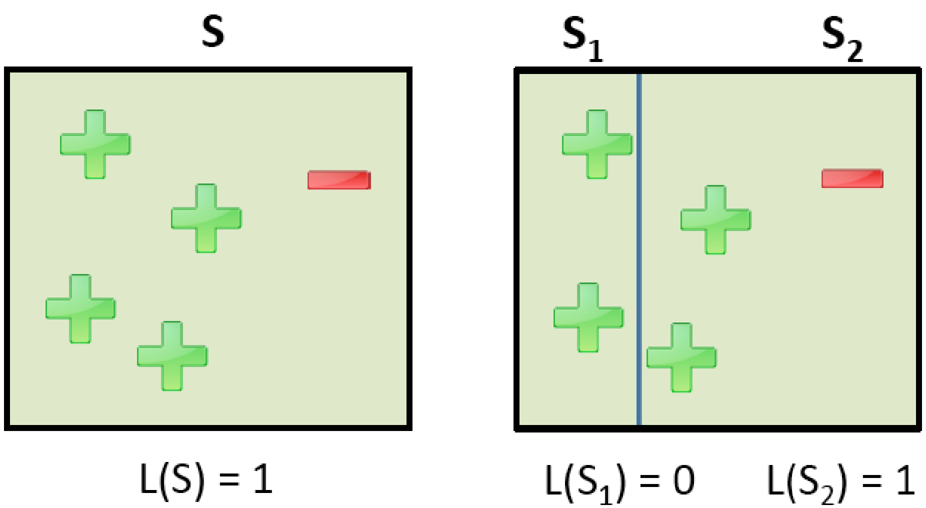
(N = #training points)

How to choose “best” query?

Decision tree: Impurity

Classification Error
of best single prediction

- Define impurity function:
  - E.g., 0/1 Loss: $L(S') = \min_{\hat{y} \in \{0,1\}} \sum_{(x,y) \in S'} 1_{[\hat{y} \neq y]}$



Impurity Reduction = 0

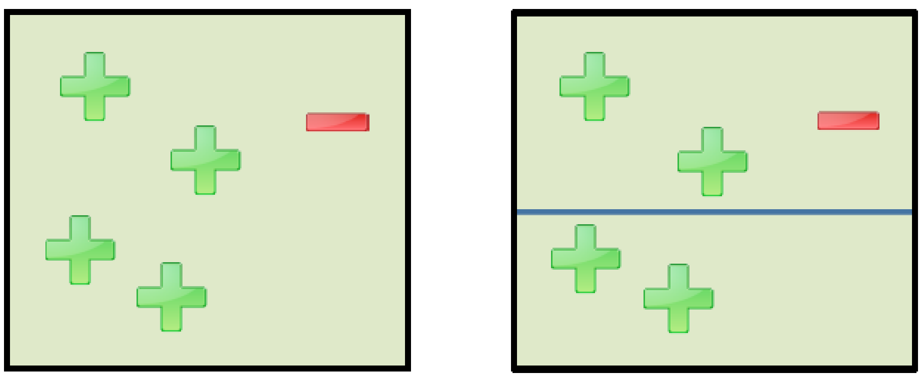
No Benefit From This Split!

Decision tree: Impurity

Classification Error
of best single prediction

- Define impurity function:
  - E.g., 0/1 Loss: $L(S') = \min_{\hat{y} \in \{0,1\}} \sum_{(x,y) \in S'} 1_{[\hat{y} \neq y]}$

S



S_2

S_1

L(S) = 1

$L(S_1) = 0$ $L(S_2) = 1$

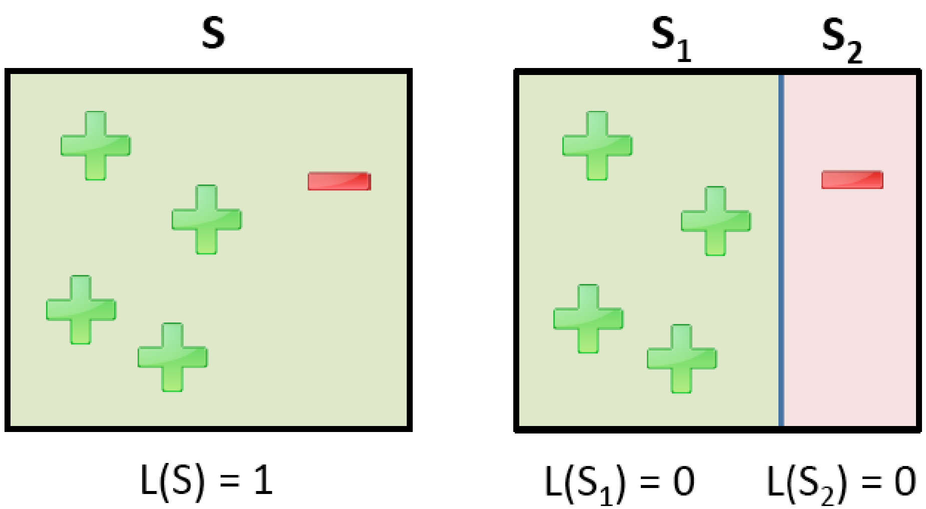
Impurity Reduction = 0

No Benefit From This Split!

Decision tree: Impurity

Classification Error
of best single prediction

- Define impurity function:
  – E.g., 0/1 Loss: $L(S') = \min_{\hat{y} \in \{0,1\}} \sum_{(x,y) \in S'} 1_{[\hat{y} \neq y]}$



Impurity Reduction = 1

Choose Split with largest impurity reduction!

Decision tree: Impurity = Loss

- **Training Goal:**
  - Find decision tree with low impurity.

- **Impurity Over Leaf Nodes = Training Loss**

$$L(S) = \sum_{S'} L(S')$$

S' iterates over leaf nodes
Union of S' = S
(Leaf Nodes = partitioning of S)

$$L(S') = \min_{\hat{y} \in \{0,1\}} \sum_{(x,y) \in S'} 1_{[\hat{y} \neq y]}$$

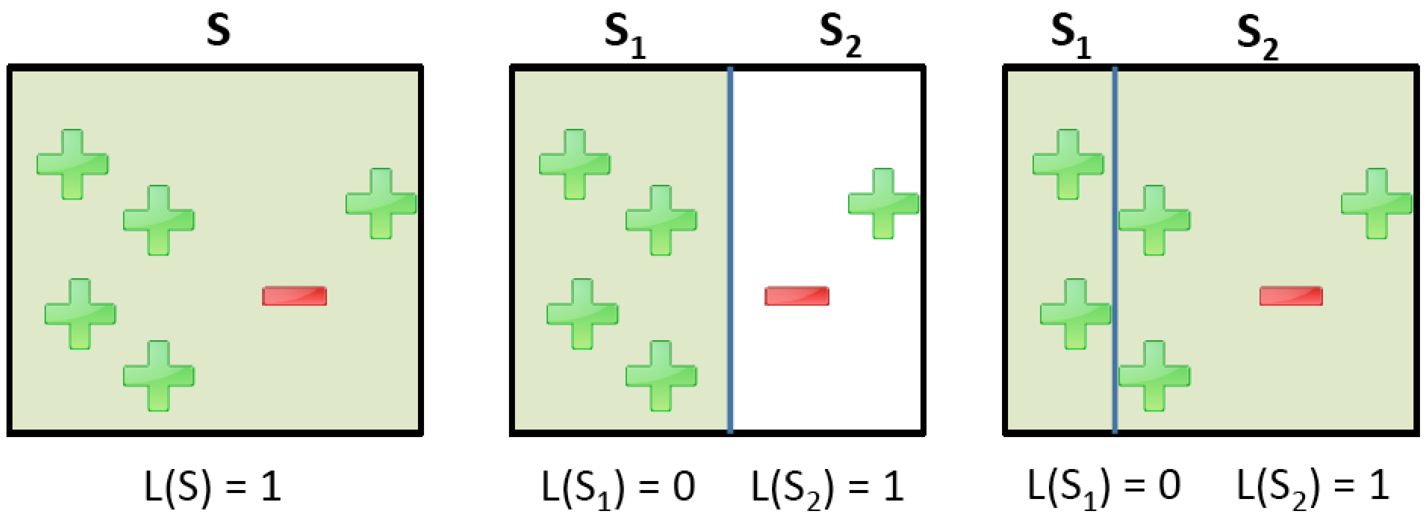
Classification Error on S'

Decision tree: 0/1 Loss ?

- What split best reduces impurity?

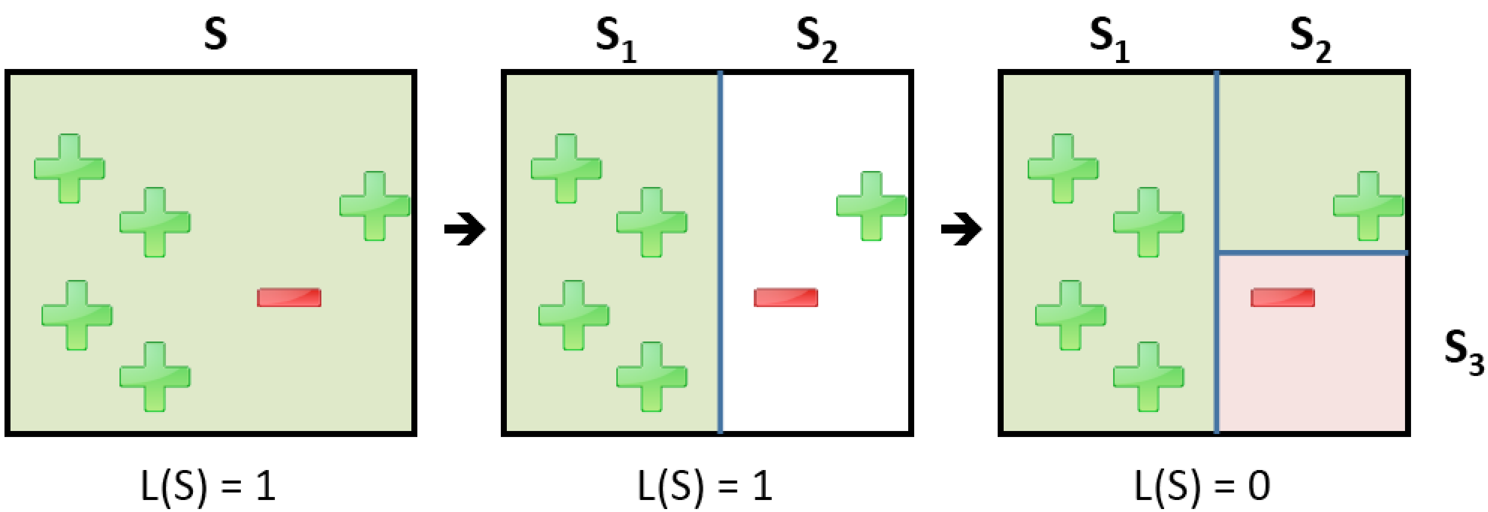
$$L(S') = \min_{\hat{y} \in \{0,1\}} \sum_{(x,y) \in S'} 1_{[\hat{y} \neq y]}$$

All Partitionings Give Same Impurity Reduction!



Decision tree: 0/1 Loss ?

- 0/1 Loss is discontinuous
- A good partitioning may not improve 0/1 Loss…
  - E.g., leads to an accurate model with subsequent split…

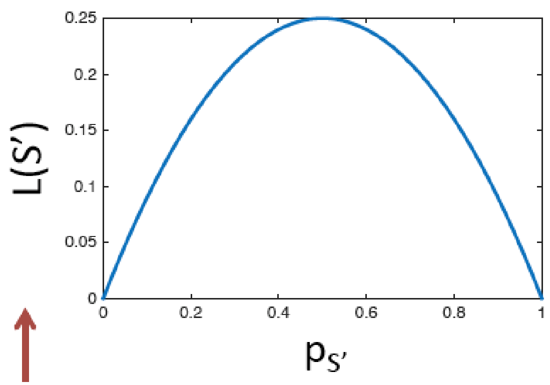


Decision tree: Surrogate impurity

- Want more continuous impurity measure
- First try: Bernoulli Variance:

$$L(S') = |S'| p_{S'}(1 - p_{S'}) = \frac{\#pos * \#neg}{|S'|}$$

$p_{S'}$ = fraction of S' that are positive examples



Worst Purity

P = 1/2, L(S') = |S'|*1/4
P = 1, L(S') = |S'|*0
P = 0, L(S') = |S'|*0

Perfect Purity

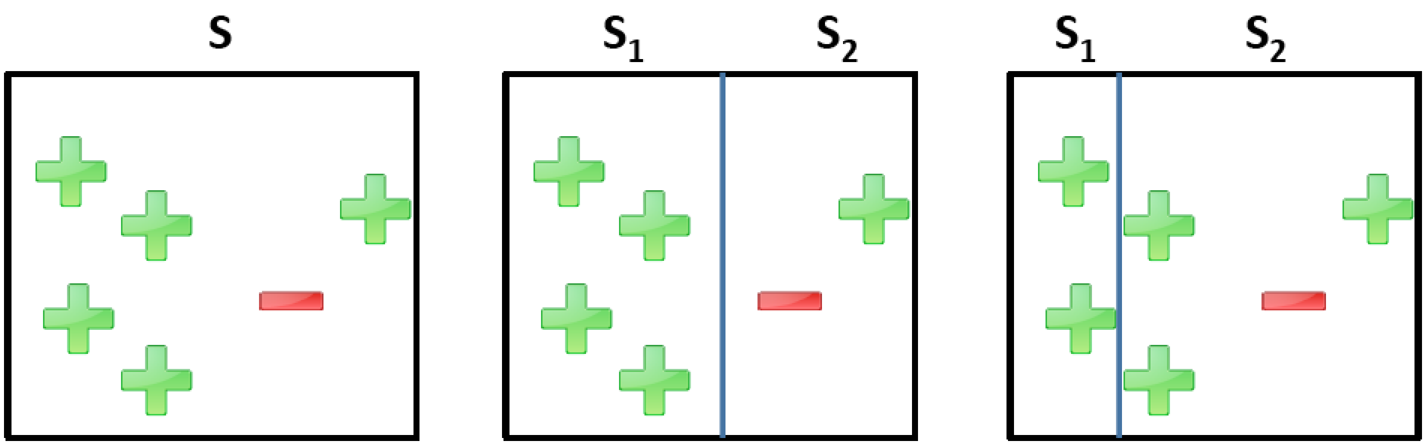
Assuming |S'|=1

Decision tree: Bernoulli variance

- What split best reduces impurity?

$$L(S') = |S'|\, p_{S'}(1 - p_{S'}) = \frac{\#pos * \#neg}{|S'|}$$

$p_{S'}$ = fraction of S' that are positive examples



L(S) = 5/6

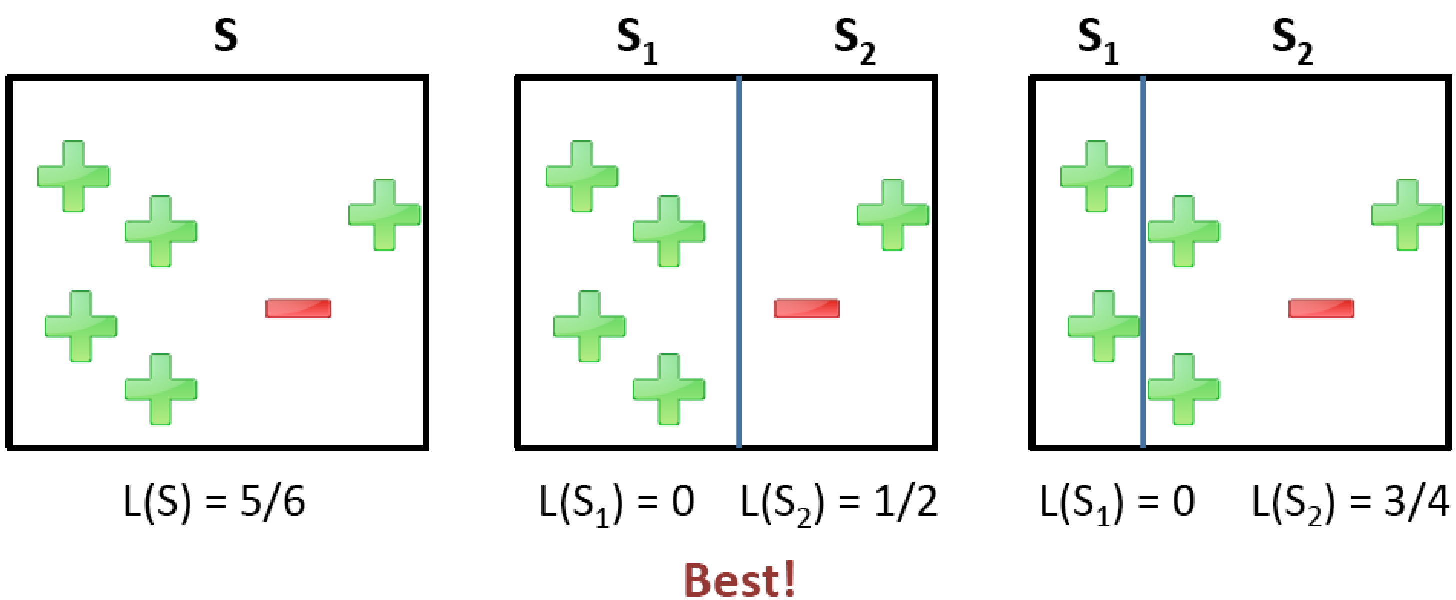
$L(S_1) = 0$ $L(S_2) = 1/2$

Best!

$L(S_1) = 0$ $L(S_2) = 3/4$

Decision tree: Bernoulli variance

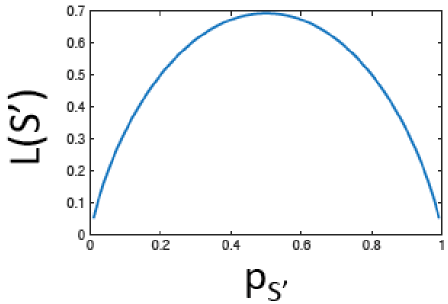
- Assume each partition = distribution over y
  - y is Bernoulli distributed with expected value $p_{S'}$
  - **Goal:** partitioning where each y has low variance



Decision tree: Surrogate impurity

Define: 0*log(0) = 0

- Entropy: $L(S') = -|S'|\left(p_{S'} \log p_{S'} + \left(1 - p_{S'}\right)\log\left(1 - p_{S'}\right)\right)$



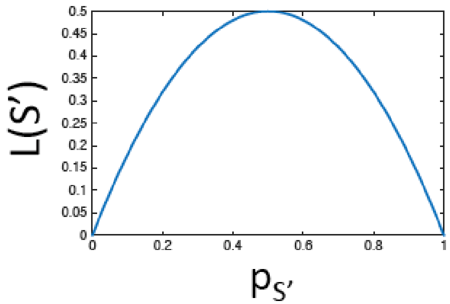
  - **aka: Information Gain:**

$$IG(A,B \mid S') = L(S') - L(A) - L(B)$$

  - (aka: Entropy Impurity Reduction)
  - Most popular.

- Gini Index:

$$L(S') = |S'|\left(1 - p_{S'}^2 - \left(1 - p_{S'}\right)^2\right)$$



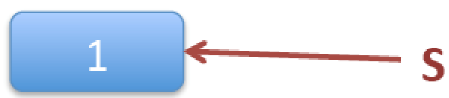
Decision tree: Learning

➢ How do we construct a useful decision tree?

- Define impurity measure L(S')
  - E.g., L(S') = Bernoulli Variance

Loop: Choose split with greatest impurity reduction (over all leaf nodes).
Repeat: until stopping condition.

Step 1:
L(S) = 12/7



Name	Age	Male?	Height > 55"
Alice	14	0	1
Bob	10	1	1
Carol	13	0	1
Dave	8	1	0
Erin	11	0	0
Frank	9	1	1
Gena	10	0	0

x y

Decision tree: Learning

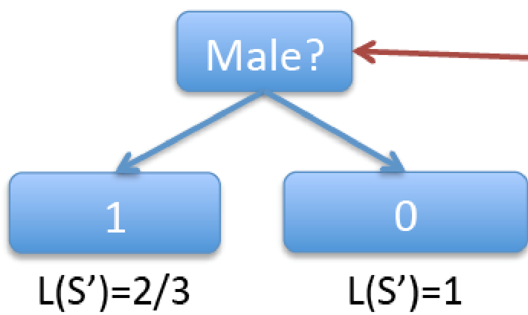
➢ How do we construct a useful decision tree?

- Define impurity measure L(S’)
  - E.g., L(S’) = Bernoulli Variance

Loop:	Choose split with greatest impurity reduction (over all leaf nodes).
Repeat:	until stopping condition.

Step 1:
L(S) = 12/7

Step 2:
L(S) = 5/3



S

Name	Age	Male?	Height > 55”
Alice	14	0	1
Bob	10	1	1
Carol	13	0	1
Dave	8	1	0
Erin	11	0	0
Frank	9	1	1
Gena	10	0	0

x y

Decision tree: Learning

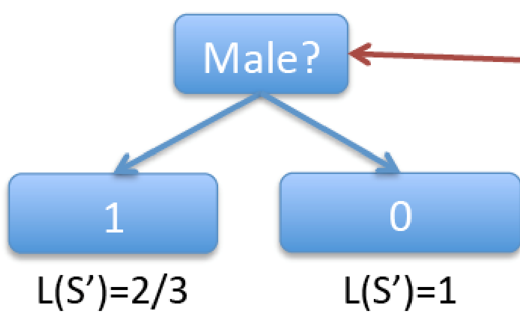
➤ How do we construct a useful decision tree?

- Define impurity measure L(S')
  - E.g., L(S') = Bernoulli Variance

Loop:	Choose split with greatest impurity reduction (over all leaf nodes).
Repeat:	until stopping condition.

Step 1:
L(S) = 12/7

Step 2:
L(S) = 5/3



Step 3: Loop over all leaves, find best split.

S

Name	Age	Male?	Height > 55"
Alice	14	0	1
Bob	10	1	1
Carol	13	0	1
Dave	8	1	0
Erin	11	0	0
Frank	9	1	1
Gena	10	0	0

x y

Decision tree: Learning

➤ How do we construct a useful decision tree?

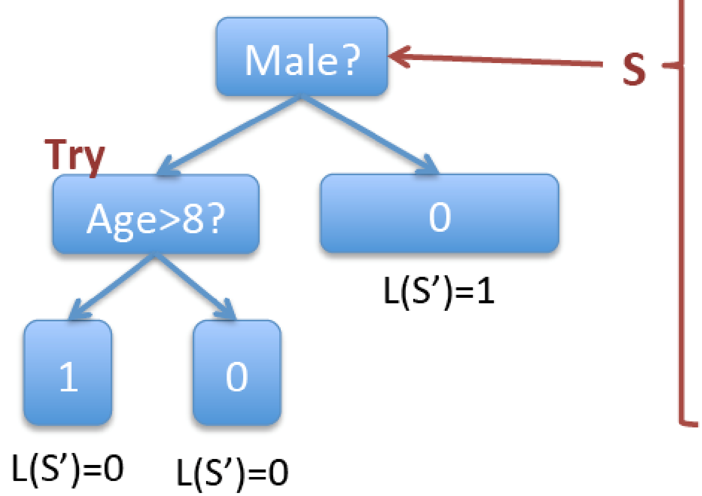
- Define impurity measure L(S')
  - E.g., L(S') = Bernoulli Variance

Loop: Choose split with greatest impurity reduction (over all leaf nodes).
Repeat: until stopping condition.

Step 1:
L(S) = 12/7

Step 2:
L(S) = 5/3

Step 3:
L(S) = 1



Name	Age	Male?	Height > 55"
Alice	14	0	1
Bob	10	1	1
Carol	13	0	1
Dave	8	1	0
Erin	11	0	0
Frank	9	1	1
Gena	10	0	0

x y

Decision tree: Learning

➢ How do we construct a useful decision tree?

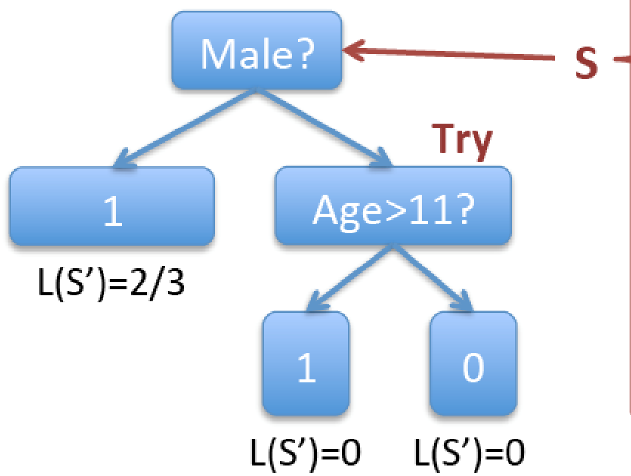
- Define impurity measure L(S')
  - E.g., L(S') = Bernoulli Variance

Loop:	Choose split with greatest impurity reduction (over all leaf nodes).
Repeat:	until stopping condition.

Step 1:
L(S) = 12/7

Step 2:
L(S) = 5/3

Step 3:
L(S) = 2/3



Name	Age	Male?	Height > 55"
Alice	14	0	1
Bob	10	1	1
Carol	13	0	1
Dave	8	1	0
Erin	11	0	0
Frank	9	1	1
Gena	10	0	0

x y

Decision tree: Learning

- How do we construct a useful decision tree?
  - Define impurity measure L(S')
    - E.g., L(S') = Bernoulli Variance

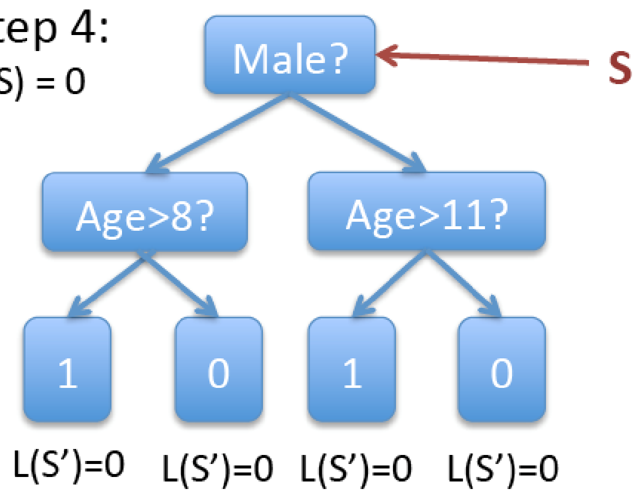
Loop: Choose split with greatest impurity reduction (over all leaf nodes).
Repeat: until stopping condition.

Step 1:
L(S) = 12/7

Step 2:
L(S) = 5/3

Step 3:
L(S) = 2/3

Step 4:
L(S) = 0



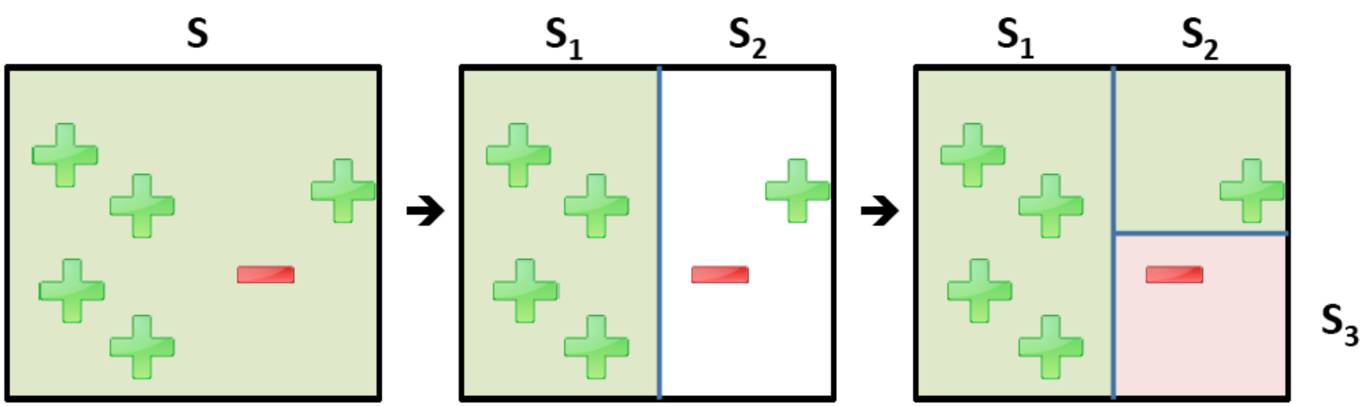
Name	Age	Male?	Height > 55"
Alice	14	0	1
Bob	10	1	1
Carol	13	0	1
Dave	8	1	0
Erin	11	0	0
Frank	9	1	1
Gena	10	0	0

x y

Decision tree: Learning

➢ Properties of top-down learning

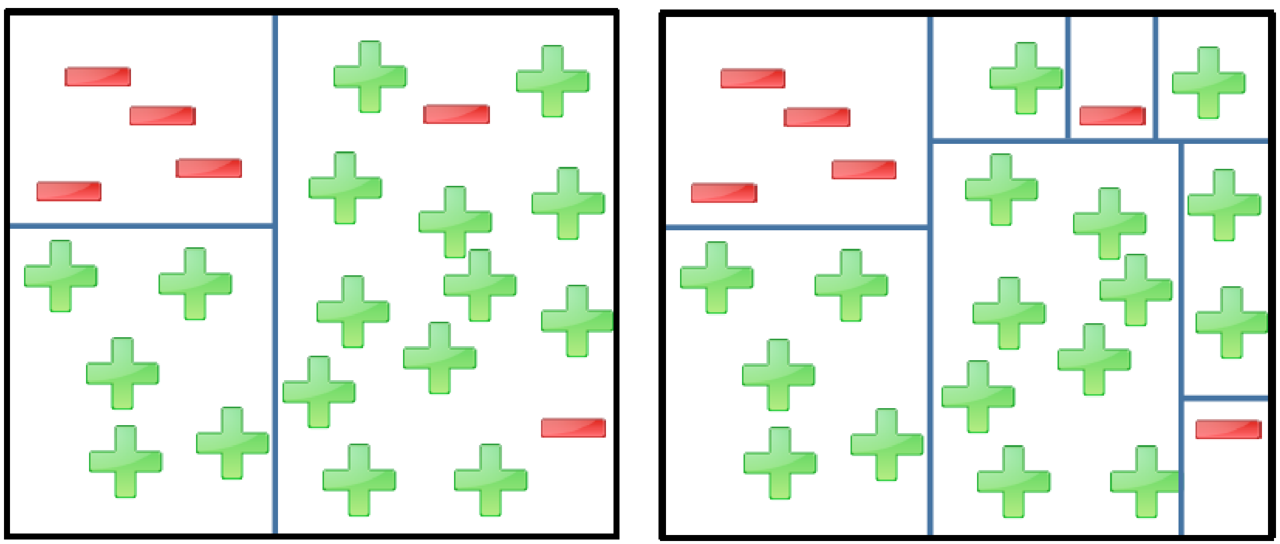
- Every intermediate step is a decision tree
  - You can stop any time and have a model
- Greedy algorithm
  - Doesn't backtrack
  - Cannot reconsider different higher-level splits.



Decision tree: Learning

- When to stop splitting?
  - If kept going, can learn tree with zero training error.
    - But such tree is probably overfitting to training set.
  - How to stop training tree earlier?
    - I.e., how to regularize?

Which one has better test error?



Decision tree: Learning

➢ When to stop splitting?

- **Minimum Size:** do not split if resulting children are smaller than a minimum size.
  - **Most common stopping condition.**

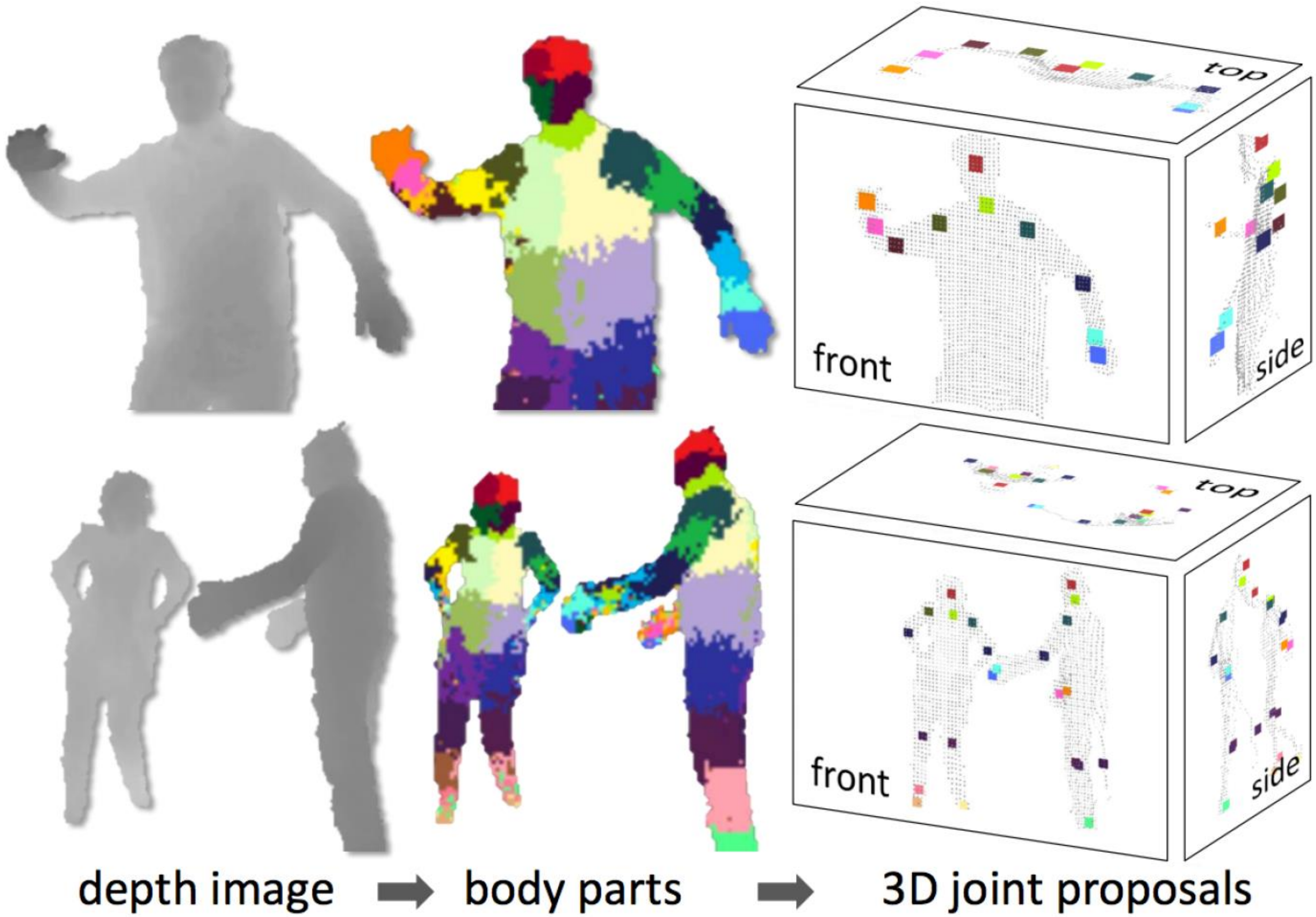
- **Maximum Depth:** do not split if the resulting children are beyond some maximum depth of tree.

- **Maximum #Nodes:** do not split if tree already has maximum number of allowable nodes.

- **Minimum Reduction in Impurity:** do not split if resulting children do not reduce impurity by at least δ%.

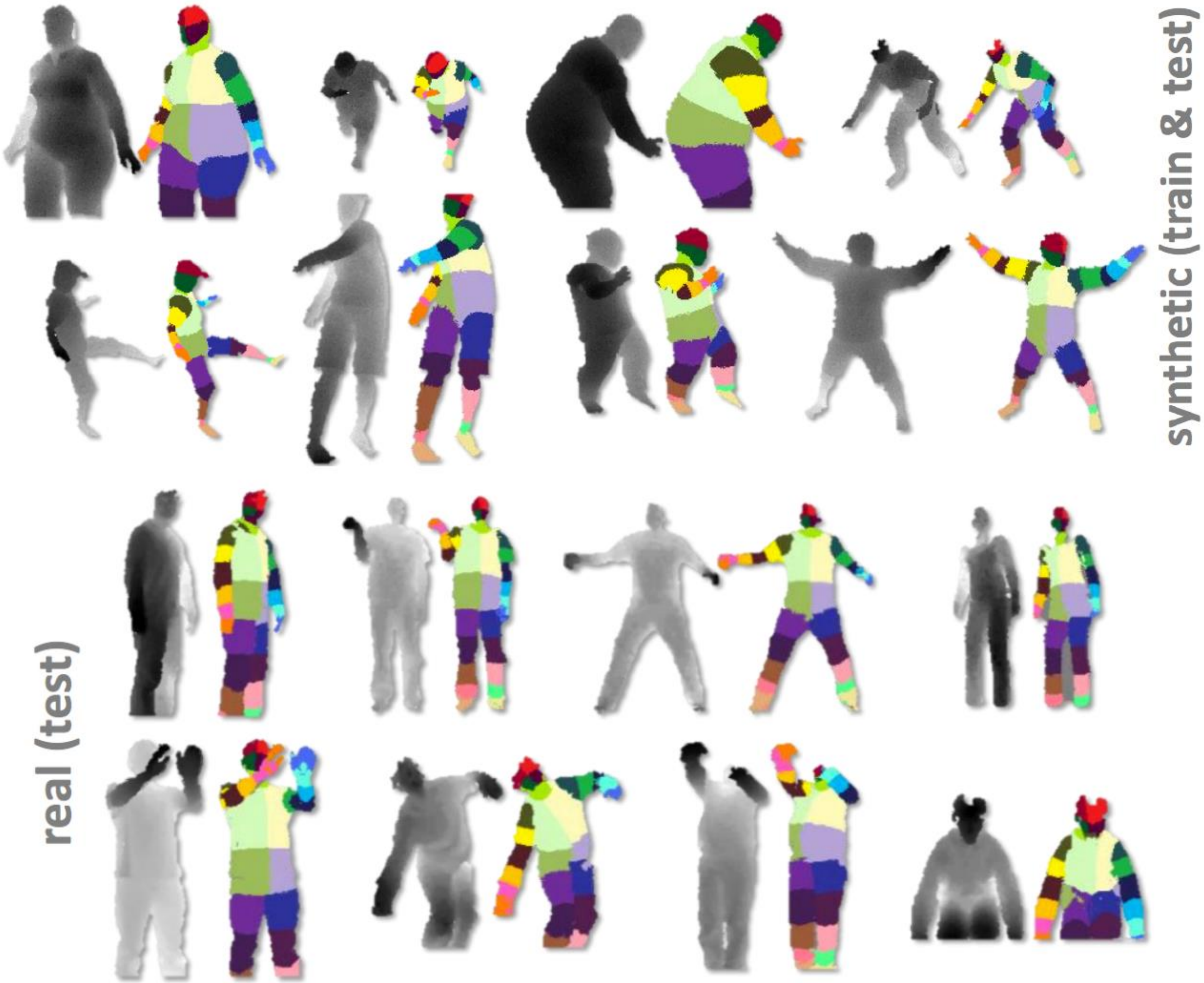
Decision tree: Applications

- Decision trees are in XBox: Classifying body parts



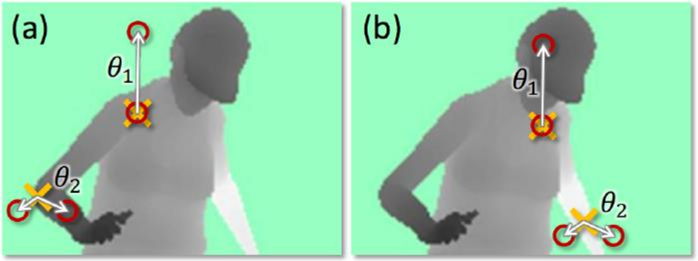
Decision tree: Applications

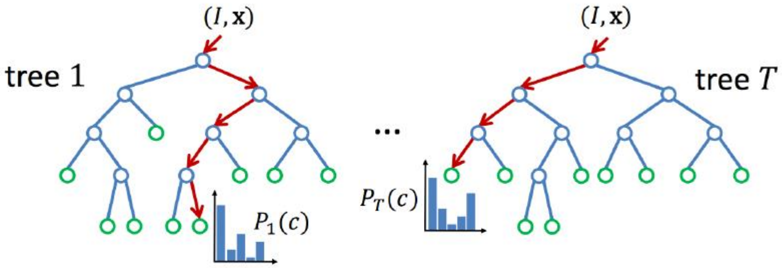
- Trained on million(s) of examples



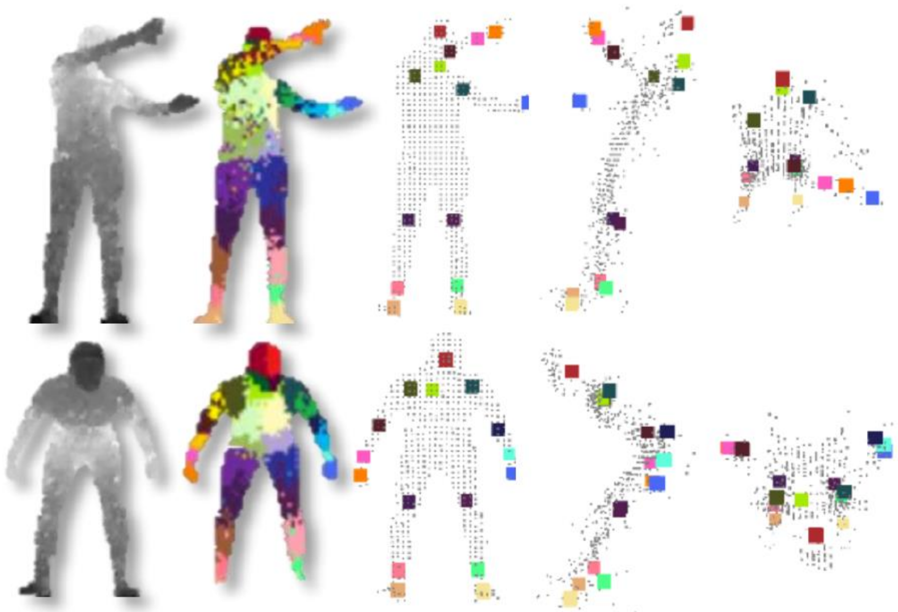
Decision tree: Applications

- Trained on million(s) of examples





- Results:



Decision tree: Summary

- Train Top-Down
  - Iteratively split existing leaf node into 2 leaf nodes
- Minimize Impurity (= Training Loss)
  - E.g., Entropy
- Until Stopping Condition (= Regularization)
  - E.g., Minimum Node Size
- Finding optimal tree is intractable
  - E.g., tree satisfying minimal leaf sizes with lowest impurity.

Decision tree: Summary

- Piecewise Constant Model Class
  - Non-linear!
  - Axis-aligned partitions of feature space
- Train to minimize impurity of training data in leaf partitions
  - Top-Down Greedy Training
- Often more accurate than linear models
  - If enough training data

Ensemble methods

- Typical application: classification
- Ensemble of classifiers is a set of classifiers whose individual decisions are combined in some way to classify new examples
- Simplest approach:
  1. Generate multiple classifiers
  2. Each votes on test instance
  3. Take majority as classification
- Classifiers are different due to different sampling of training data, or randomized parameters within the classification algorithm
- Aim: take simple mediocre algorithm and transform it into a super classifier without requiring any fancy new algorithm

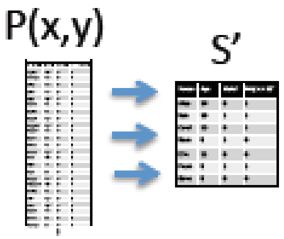
Ensemble methods

- Differ in training strategy, and combination method
  - Parallel training with different training sets
    1. Bagging (bootstrap aggregation) – train separate models on overlapping training sets, average their predictions
  - Sequential training, iteratively re-weighting training examples so current classifier focuses on hard examples: boosting

Why do Ensemble methods work?

- Based on one of two basic observations:
  1. Variance reduction: if the training sets are completely independent, it will always help to average an ensemble because this will reduce variance without affecting bias (e.g., bagging)
     - reduce sensitivity to individual data points
  2. Bias reduction: for simple models, average of models has much greater capacity than single model (e.g., hyperplane classifiers, Gaussian densities).
     - Averaging models can reduce bias substantially by increasing capacity, and control variance by fitting one component at a time (e.g., boosting)

Bagging



- **Goal:** reduce variance

sampled independently

- **Ideal setting:** many training sets S'
  - Train model using each S'
  - Average predictions

Variance reduces linearly
Bias unchanged

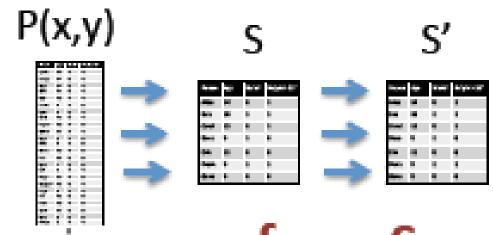
$$\boxed{\underbrace{E_S[(h_S(x) - y)^2]}_{\text{Expected Error On single (x,y)}} = \underbrace{E_S[(Z-\check{z})^2]}_{\text{Variance}} + \underbrace{\check{z}^2}_{\text{Bias}}}$$

Variance **Bias**

$Z = h_S(x) - y$
$\check{z} = E_S[Z]$

Bagging = Bootstrap Aggregation

"Bootstrapping"



from S

- **Goal:** reduce variance
- **In practice:** resample S' with replacement
  - Train model using each S'
  - Average predictions

Variance reduces sub-linearly
(Because S' are correlated)
Bias often increases slightly

$$\boxed{\underbrace{E_S[(h_S(x) - y)^2]}_{\text{Expected Error On single (x,y)}} = E_S[(Z-\check{z})^2] + \check{z}^2}$$

Variance Bias

$Z = h_S(x) - y$

$\check{z} = E_S[Z]$

Random Forests

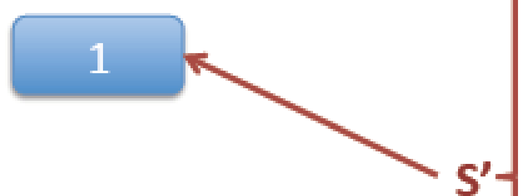
- **Goal:** reduce variance
  - Bagging can only do so much
  - Resampling training data asymptotes
- **Random Forests:** sample data & features!
  - Sample S'
  - Train DT
    - At each node, sample features
  - Average predictions

Further de-correlates trees

Random Forests

Loop: Sample T random splits at each Leaf. Choose split with greatest impurity reduction.
Repeat: until stopping condition.

Step 1:



Name	Age	Male?	Height > 55"
Alice	14	0	1
Bob	10	1	1
Carol	13	0	1
Dave	8	1	0
Erin	11	0	0
Frank	9	1	1
Gena	10	0	0

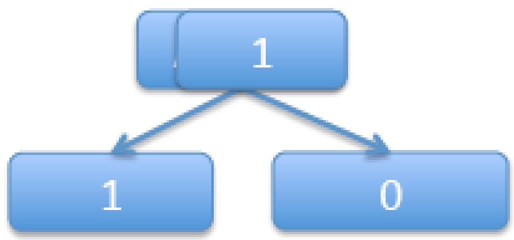
x y

Random Forests

Loop: Sample T random splits at each Leaf. Choose split with greatest impurity reduction.
Repeat: until stopping condition.

Step 1:

Step 2:



Randomly decide only look at age,
Not gender.

S'

Name	Age	Male?	Height > 55”
Alice	14	0	1
Bob	10	1	1
Carol	13	0	1
Dave	8	1	0
Erin	11	0	0
Frank	9	1	1
Gena	10	0	0

x y

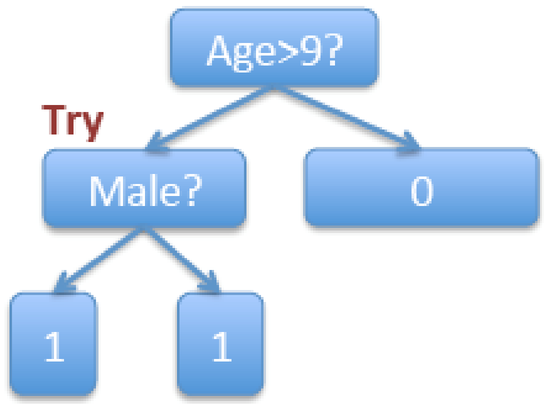
Random Forests

Loop: Sample T random splits at each Leaf. Choose split with greatest impurity reduction.
Repeat: until stopping condition.

Step 1:

Step 2:

Step 3:



Randomly decide only look at gender.

S'

Name	Age	Male?	Height > 55"
Alice	14	0	1
Bob	10	1	1
Carol	13	0	1
Dave	8	1	0
Erin	11	0	0
Frank	9	1	1
Gena	10	0	0

x y

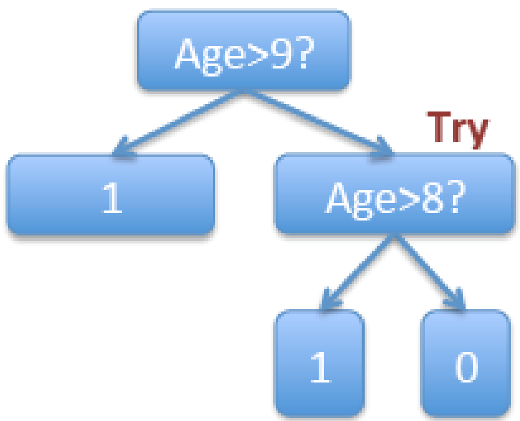
Random Forests

Loop: Sample T random splits at each Leaf. Choose split with greatest impurity reduction.
Repeat: until stopping condition.

Step 1:

Step 2:

Step 3:



Randomly decide only look at age.

Name	Age	Male?	Height > 55”
Alice	14	0	1
Bob	10	1	1
Carol	13	0	1
Dave	8	1	0
Erin	11	0	0
Frank	9	1	1
Gena	10	0	0

S’

x

y

Random Forests: Summary

- Extension of Bagging to sampling Features
- Generate Bootstrap S' from S
  - Train DT Top-Down on S'
  - Each node, sample subset of features for splitting
    - Can also sample a subset of splits as well
- Average Predictions of all DTs

Boosting: Overview

- Boosting for **classification** works by combining *weak learners* into a more accurate *ensemble classifier*
  - A weak learner need only do better than chance

- Training consists of multiple *boosting rounds*
  - During each boosting round, we select a weak learner that does well on examples that were hard for the previous weak learners
  - “Hardness” is captured by weights attached to training examples

Boosting: Model

- Defines a classifier using an additive model:

$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \ldots$$

Strong classifier

Features vector

Weight

Weak classifier

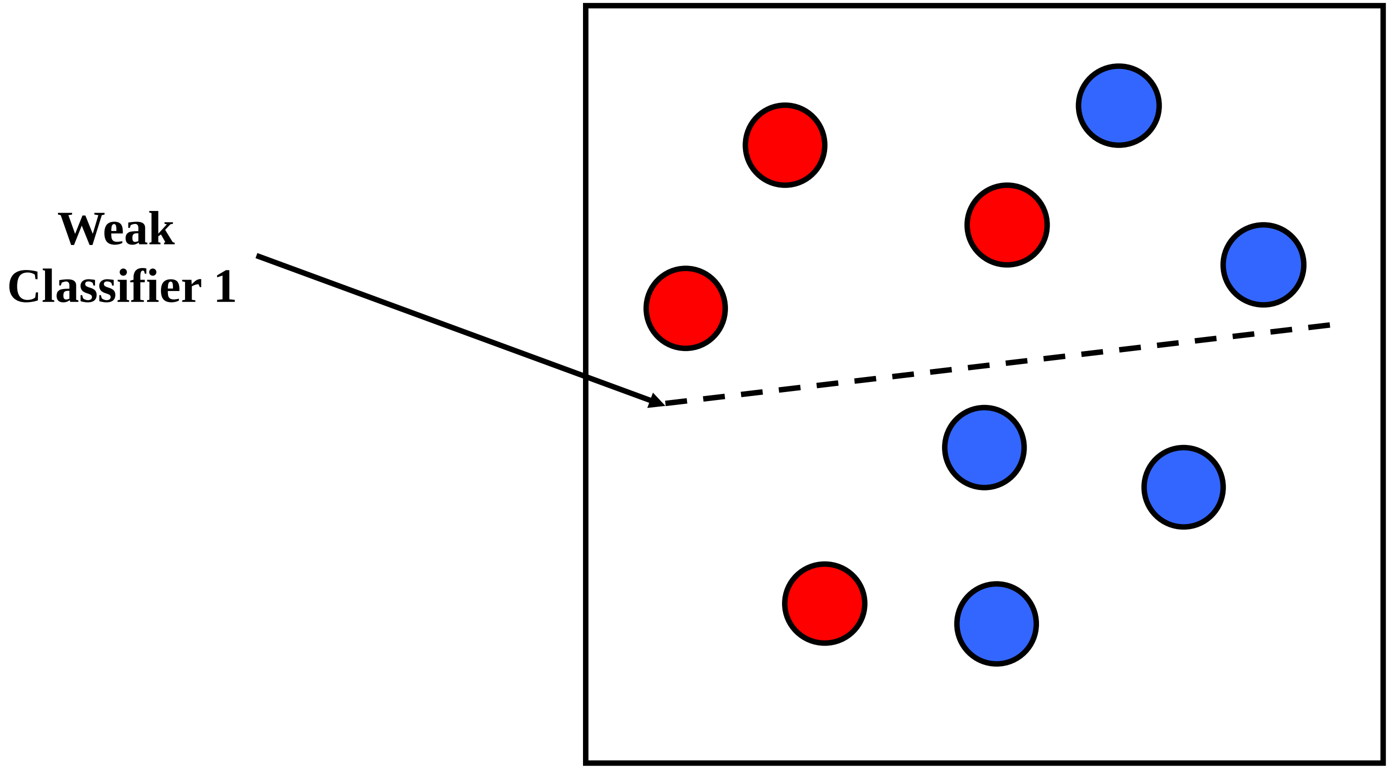
- We need to define a family of weak classifiers

$f_k(x)$ from a family of weak classifiers

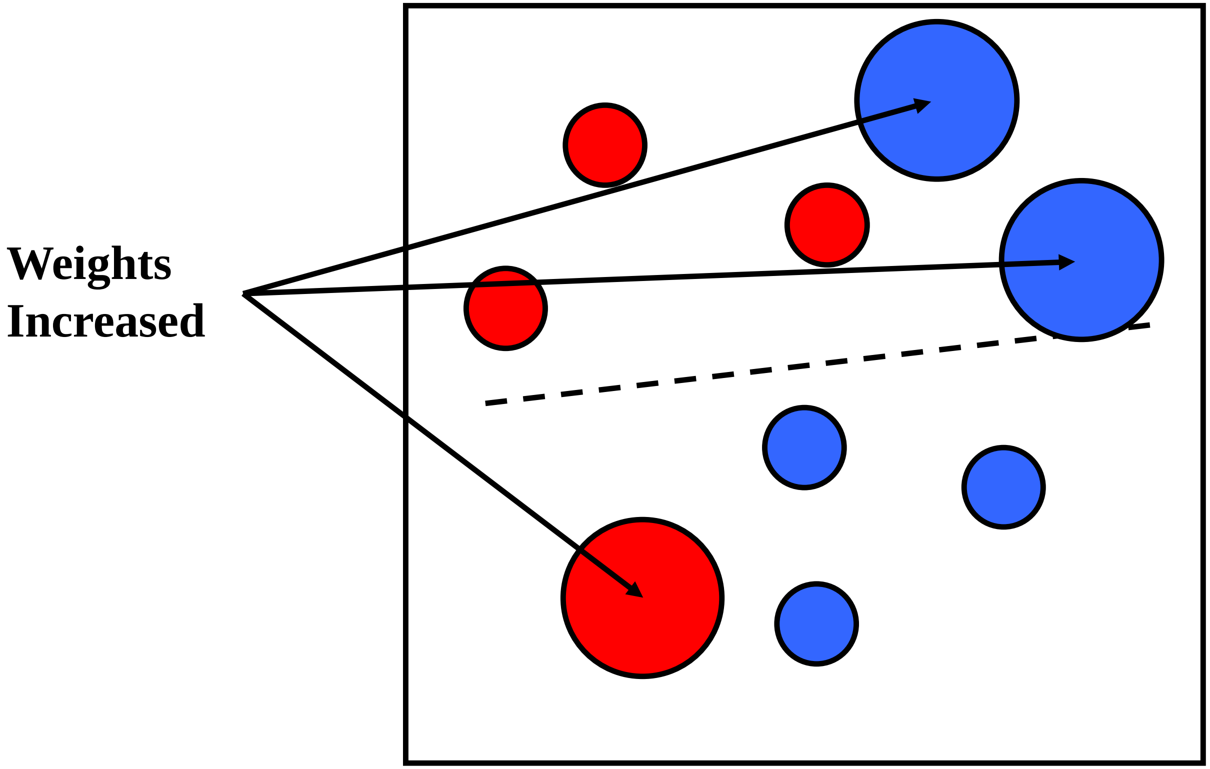
Boosting: Learning Procedure

- Initially, weight each training example equally, e.g., 1/N
- In each boosting round:
    - Find the weak learner that achieves the lowest *weighted* training error
    - Raise the weights of training examples misclassified by current weak learner
- Compute final classifier as linear combination of all weak learners (weight of each learner is directly proportional to its accuracy)

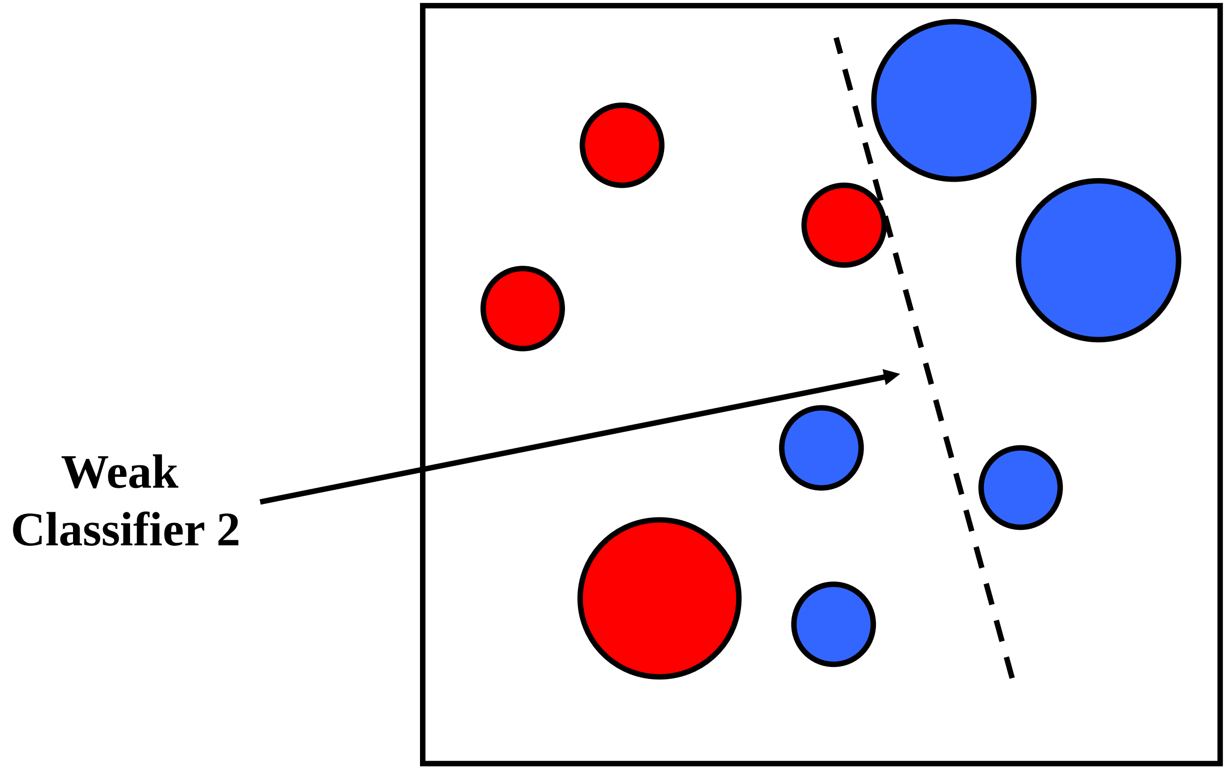
Boosting: Illustration



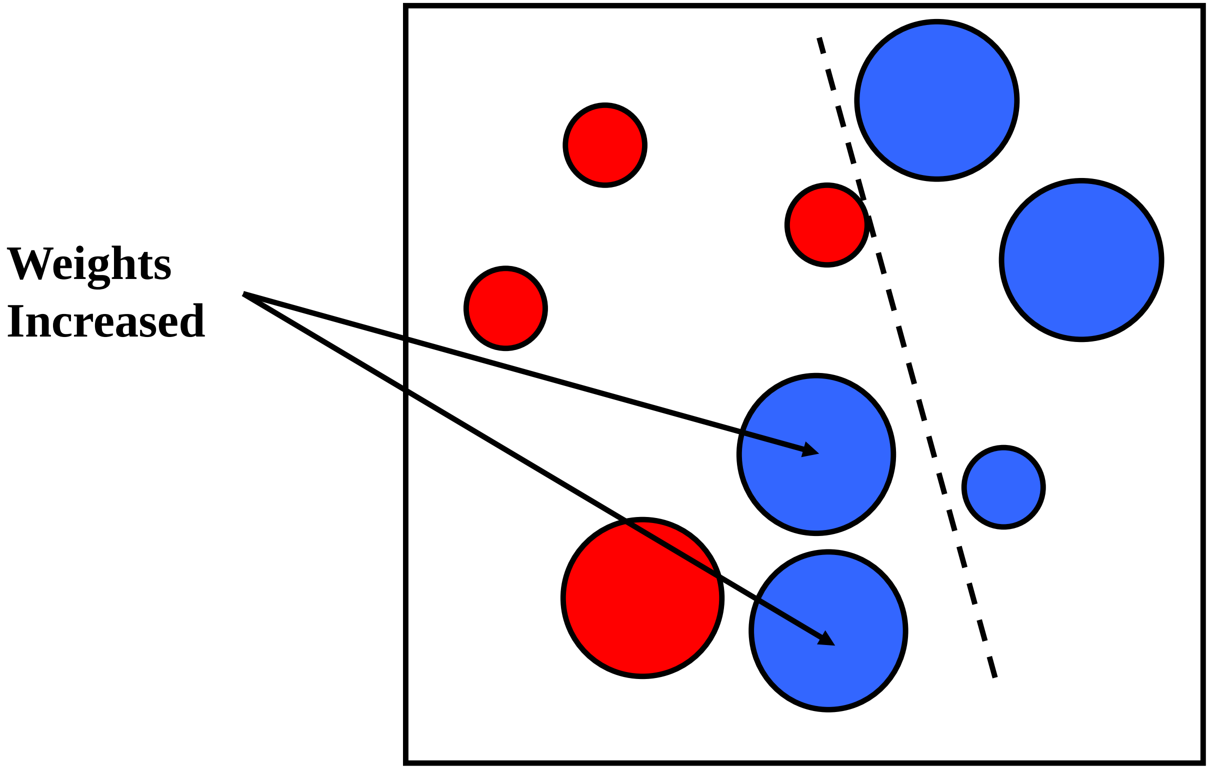
Boosting: Illustration



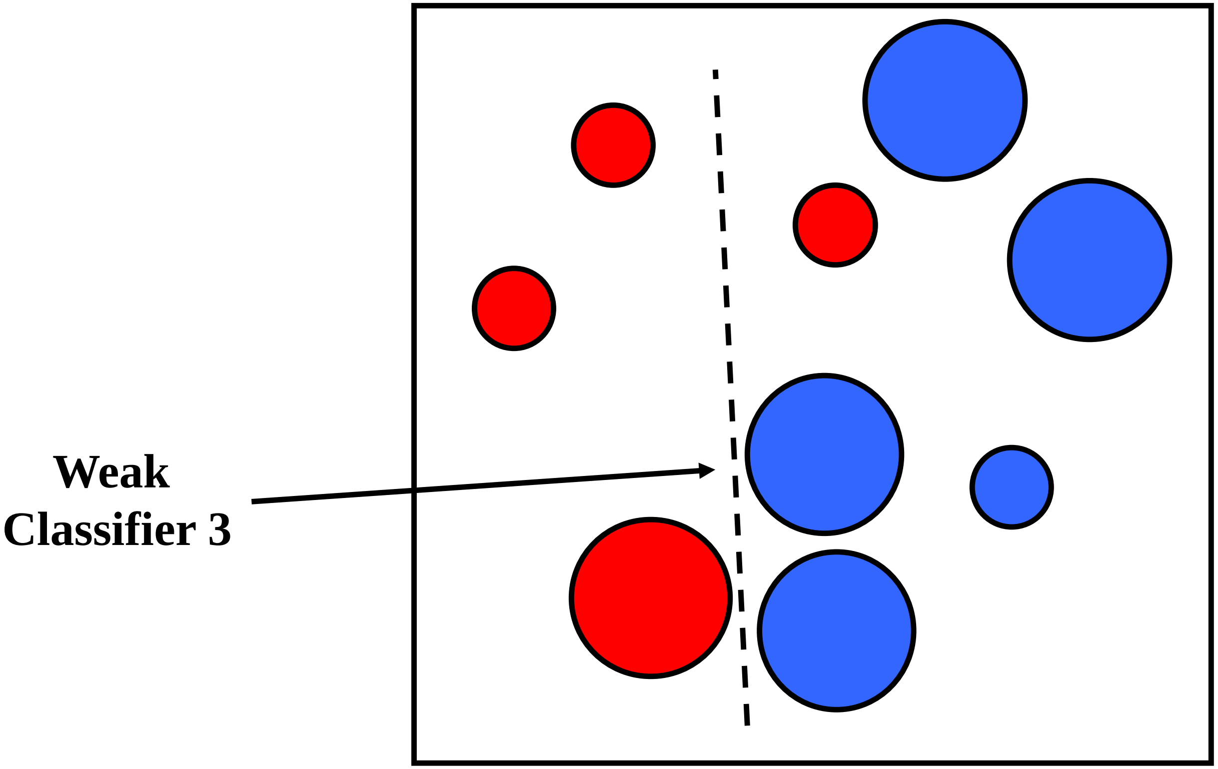
Boosting: Illustration



Boosting: Illustration

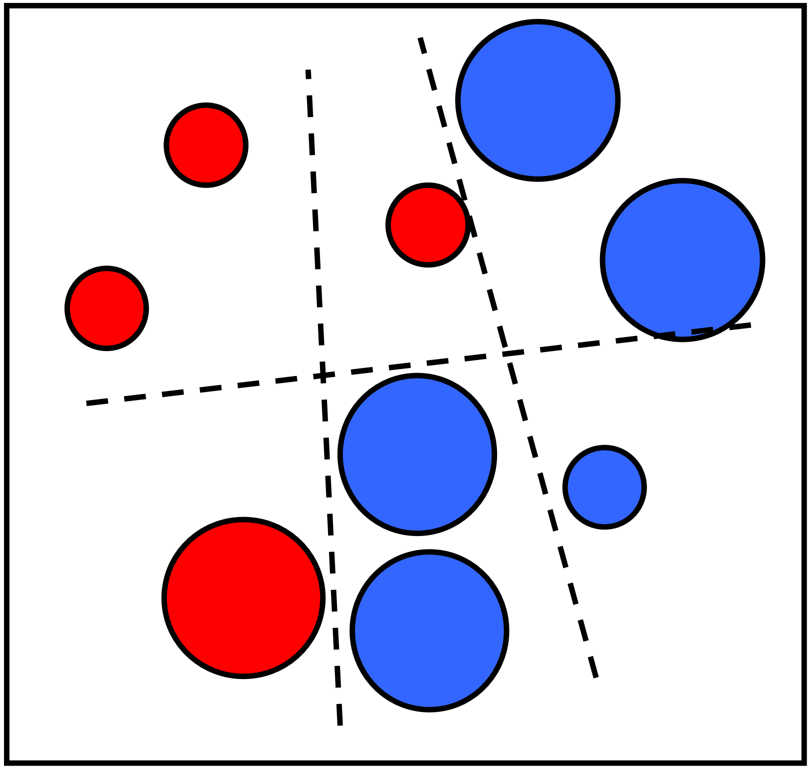


Boosting: Illustration



Boosting: Illustration

Final classifier is a combination of weak classifiers



Boosting: AdaBoost algorithm

$$S = \{(x_i, y_i)\}_{i=1}^N$$

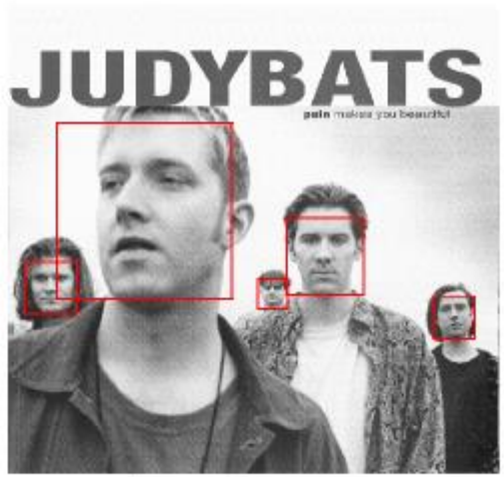
$$y_i \in \{-1, +1\}$$

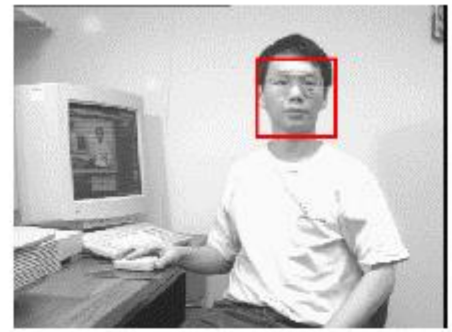
E.g., best decision stump

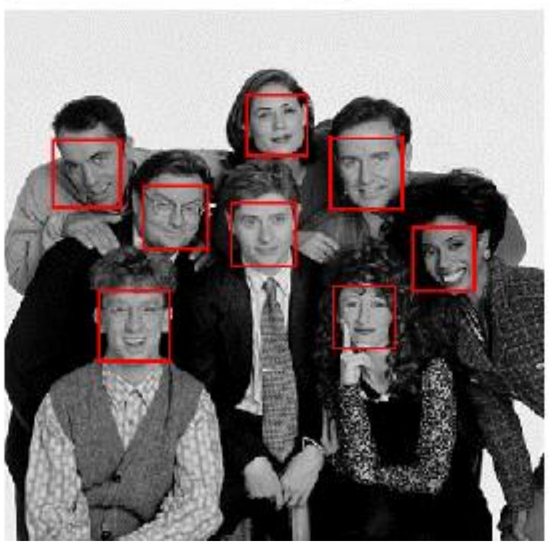
- Init $D_1(x) = 1/N$
- Loop $t = 1 \ldots n$:
  - Train classifier $h_t(x)$ using (S, D_t)
  - Compute error on (S, D_t): $\varepsilon_t \equiv L_{D_t}(h_t) = \sum_i D_t(i) L(y_i, h_t(x_i))$
  - Define step size a_t: $a_t = \frac{1}{2} \log \left\{ \frac{1 - \varepsilon_t}{\varepsilon_t} \right\}$
  - Update Weighting: $D_{t+1}(i) = \frac{D_t(i) \exp\{-a_t y_i h_t(x_i)\}}{Z_t}$
- **Return:** $h(x) = \text{sign}(a_1 h_1(x) + \ldots + a_n h_n(x))$

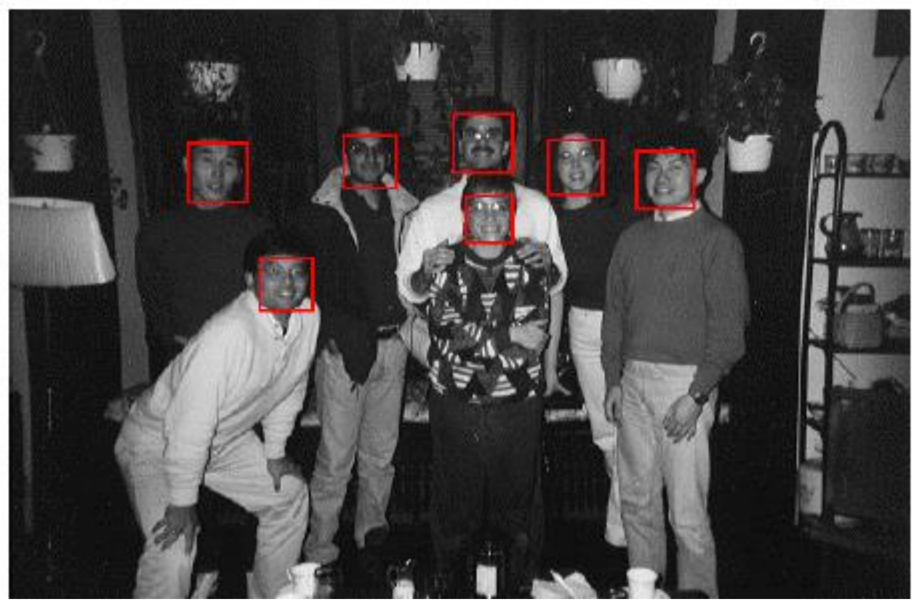
Normalization Factor s.t. D_{t+1} sums to 1.

Boosting: Applications

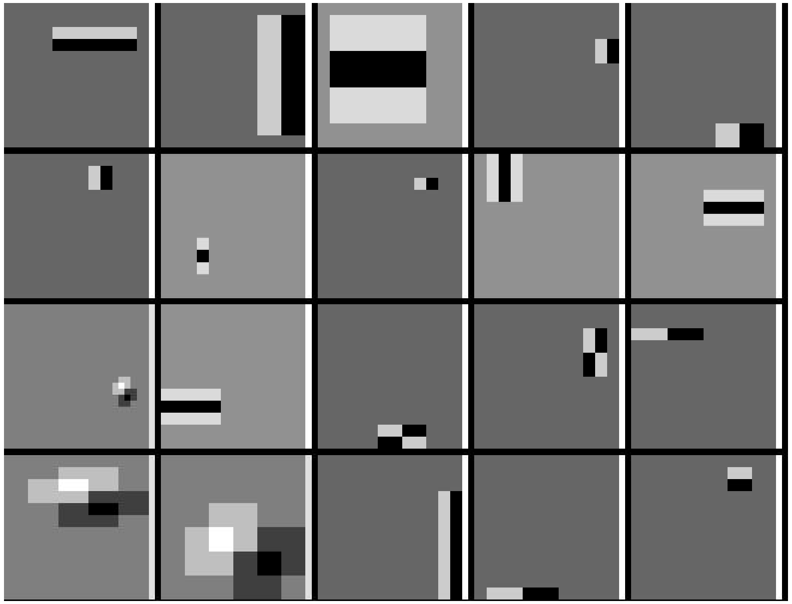








Boosting: Applications



Taxonomy
of Machine Learning

Labeled Data

Indirect (reward)

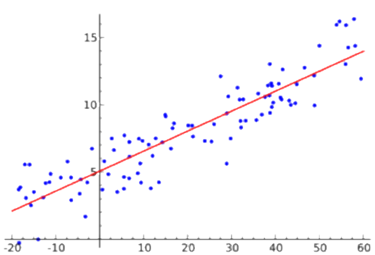
Unlabeled Data

Supervised Learning

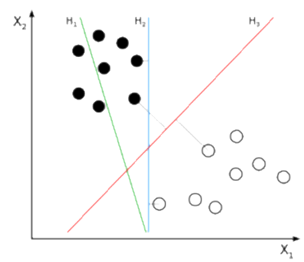
Reinforcement & Bandit Learning

Unsupervised Learning

Regression



Classification



Dimensionality Reduction

Clustering