

Machine Learning & Data Mining

CS/CNS/EE 155

Lecture 9:

Decision Trees, Bagging & Random Forests

Announcements

- Homework 2 harder than anticipated
 - Will not happen again
 - Lenient grading
- Homework 3 Released Tonight
 - Due 2 weeks later
 - (Much easier than HW2)
- Homework 1 will be graded soon.

Announcements

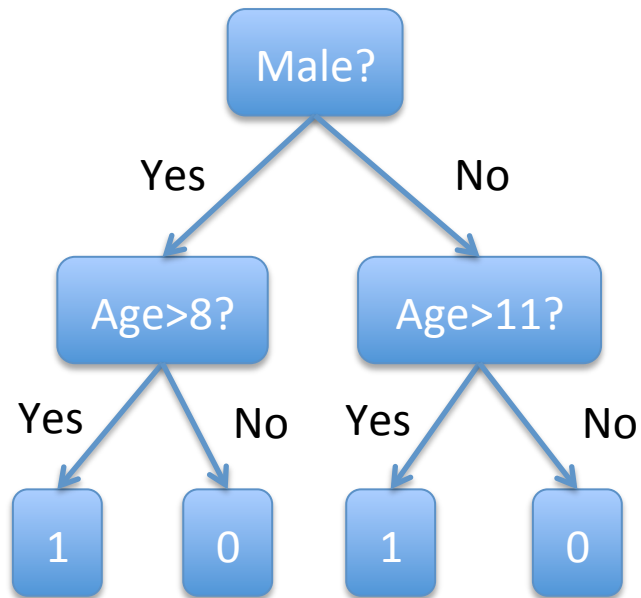
- Kaggle Mini-Project Released
 - <https://kaggle.com/join/csee155> (also linked to from course website)
 - Dataset of Financial Reports to SEC
 - Predict if Financial Report was during Great Recession
 - Training set of ~5000, with ~500 features
 - Submit predictions on test set
 - Competition ends in ~3 weeks
 - Report due 2 days after competition ends
 - **Grading: 80% on cross validation & model selection, 20% on accuracy**
 - If you do everything properly, you should get at least 90%
 - **Don't panic! With proper planning, shouldn't take that much time.**
 - Recitation tomorrow (short Kaggle & Decision Tree Tutorial)

This Week

- Back to standard classification & regression
 - No more structured models
- Focus to achieve highest possible accuracy
 - Decision Trees
 - Bagging
 - Random Forests
 - Boosting
 - Ensemble Selection

Decision Trees

(Binary) Decision Tree

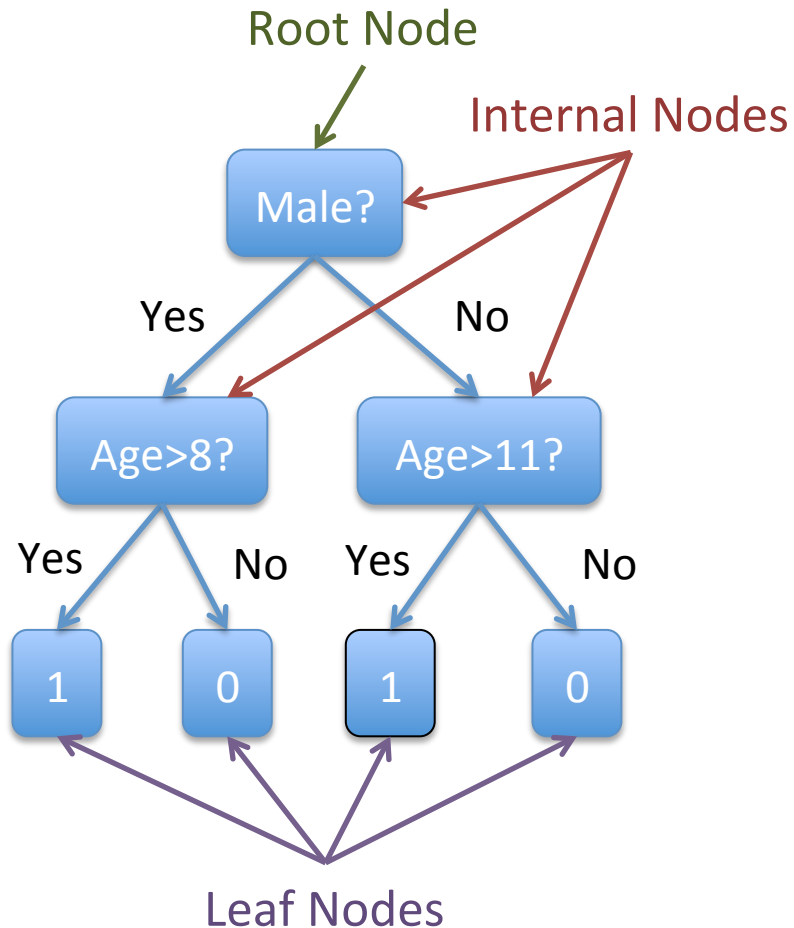


Don't overthink this, it is literally what it looks like.

Person	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

(Binary) Decision Tree



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.

(Binary) Decision Tree

Input:

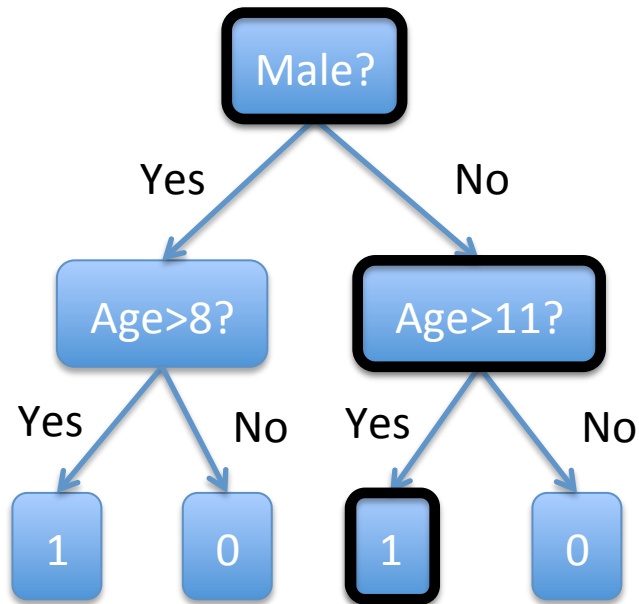


Alice

Gender: Female

Age: 14

Prediction: Height > 55"



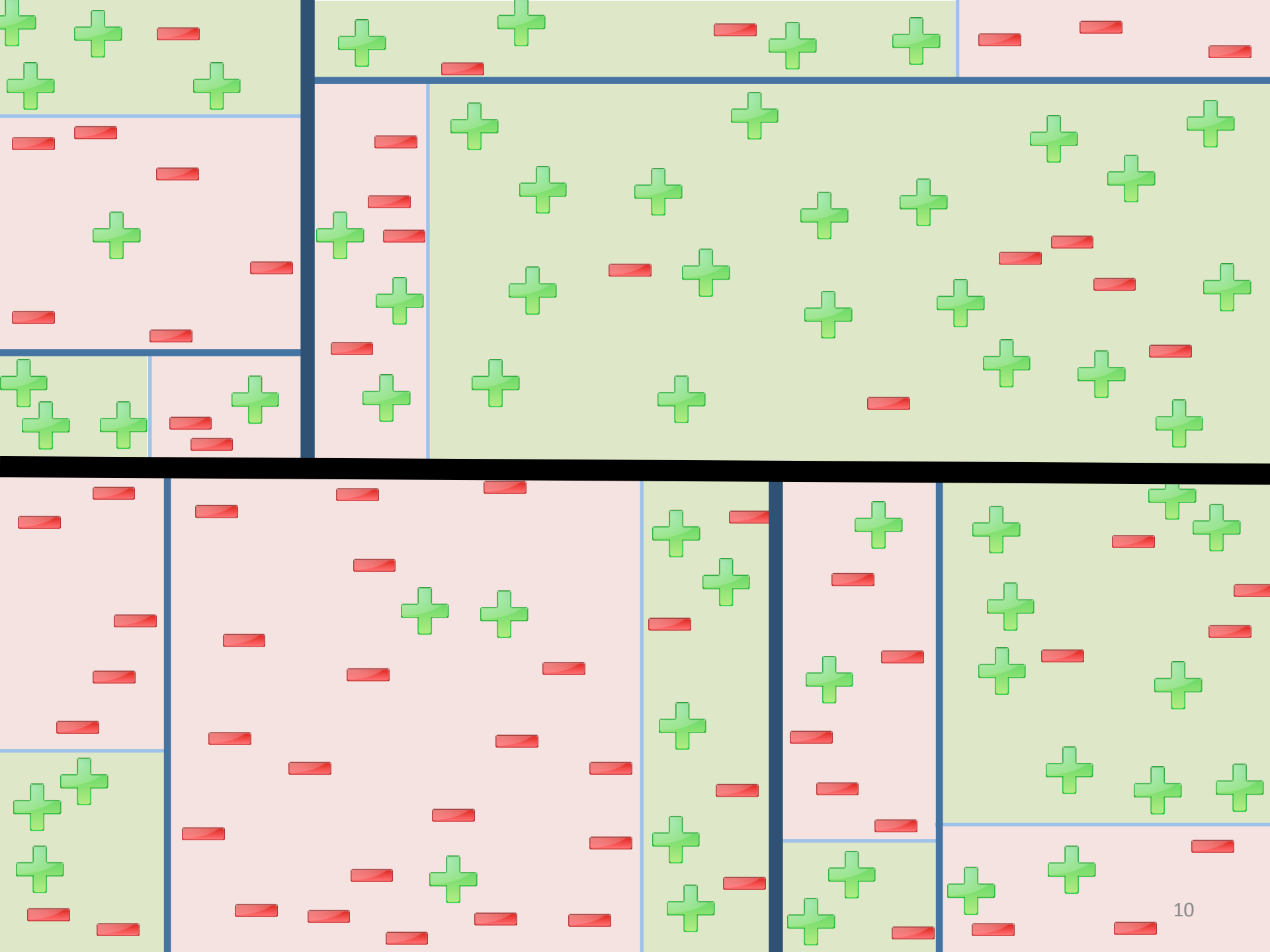
Every **internal node** has a **binary** query function $q(x)$.

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.

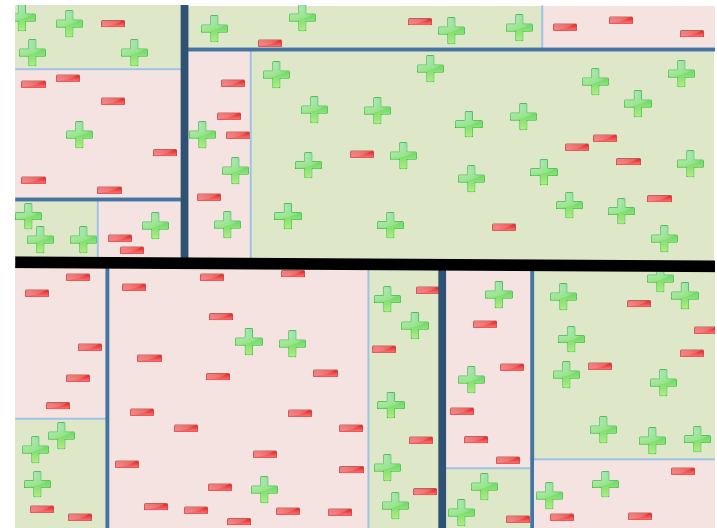
Queries

- Decision Tree defined by Tree of Queries
- Binary query $q(x)$ maps features to 0 or 1
- Basic form: $q(x) = \mathbf{1}[x^d > c]$
 - $\mathbf{1}[x^3 > 5]$
 - $\mathbf{1}[x^1 > 0]$
 - $\mathbf{1}[x^{55} > 1.2]$
- Axis aligned partitioning of input space



Basic Decision Tree Function Class

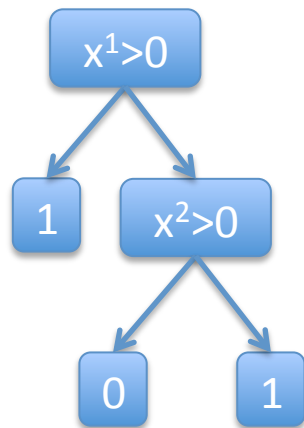
- “Piece-wise Static” Function Class
 - All possible partitionings over feature space.
 - Each partition has a static prediction.
- Partitions axis-aligned
 - E.g., No Diagonals
- (Extensions next week)



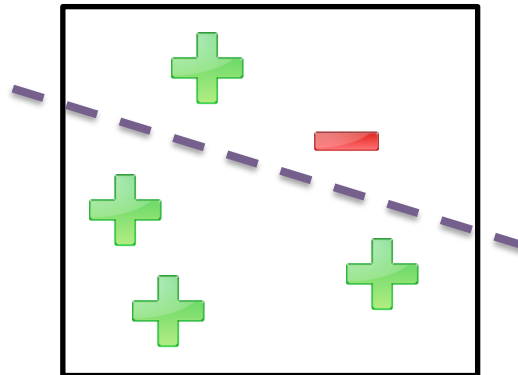
Decision Trees vs Linear Models

- Decision Trees are NON-LINEAR Models!

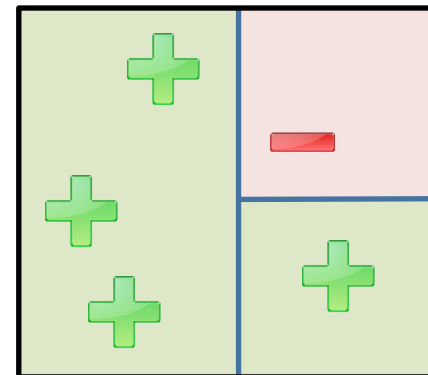
- Example:



No Linear Model
Can Achieve 0 Error



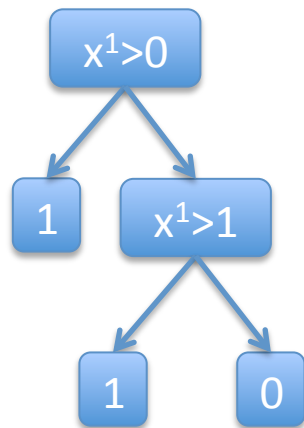
Simple Decision Tree
Can Achieve 0 Error



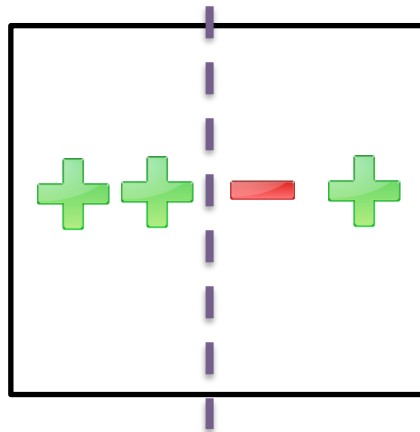
Decision Trees vs Linear Models

- Decision Trees are NON-LINEAR Models!

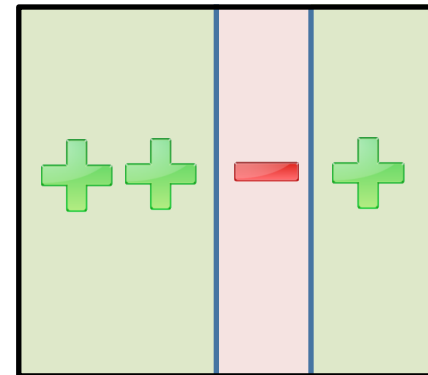
- Example:



No Linear Model
Can Achieve 0 Error



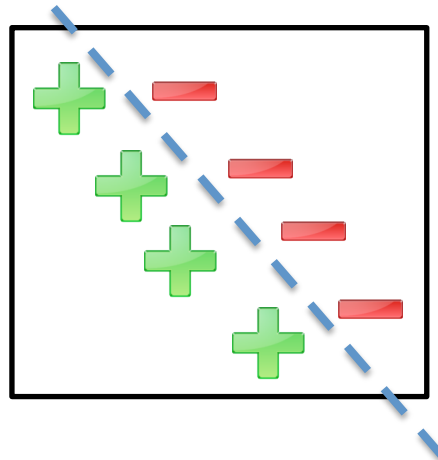
Simple Decision Tree
Can Achieve 0 Error



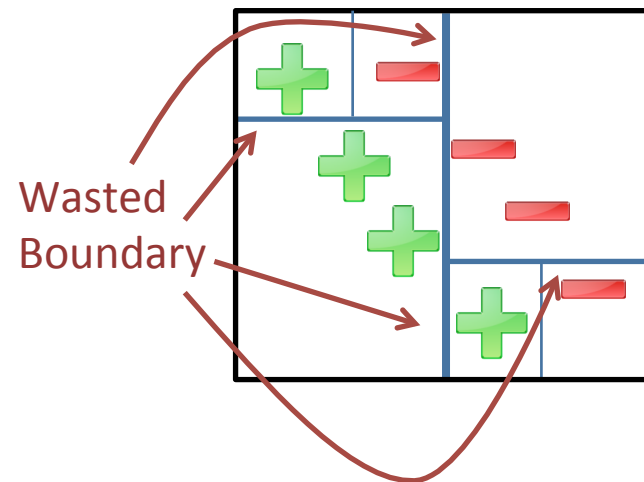
Decision Trees vs Linear Models

- 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



More Extreme Example



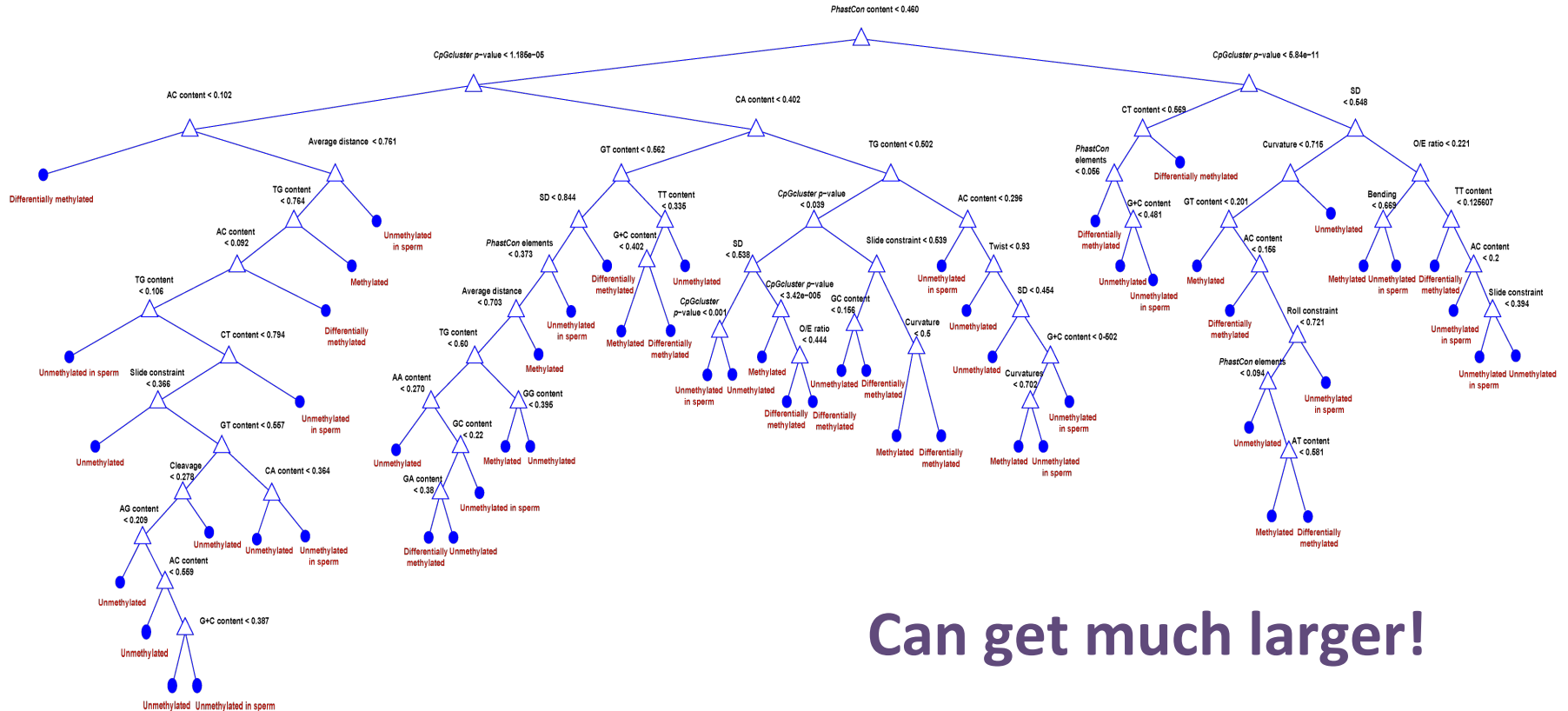
Decision Tree wastes most of model capacity on useless boundaries.

(Depicting useful boundaries)

Decision Trees vs Linear Models

- Decision Trees are often more accurate!
- Non-linearity is often more important
 - Just use many axis-aligned boundaries to approximate diagonal boundaries
 - (It's OK to waste model capacity.)
- **Catch:** requires sufficient training data
 - Will become clear later in lecture

Real Decision Trees

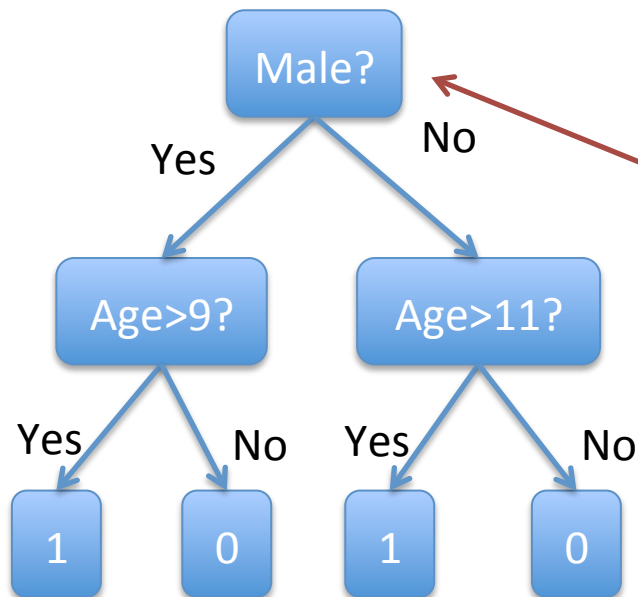


Can get much larger!

Image Source: <http://www.biomedcentral.com/1471-2105/10/116>

Decision Tree Training

Every node = partition/subset of S
Every Layer = complete partitioning of S
Children = complete partitioning of parent

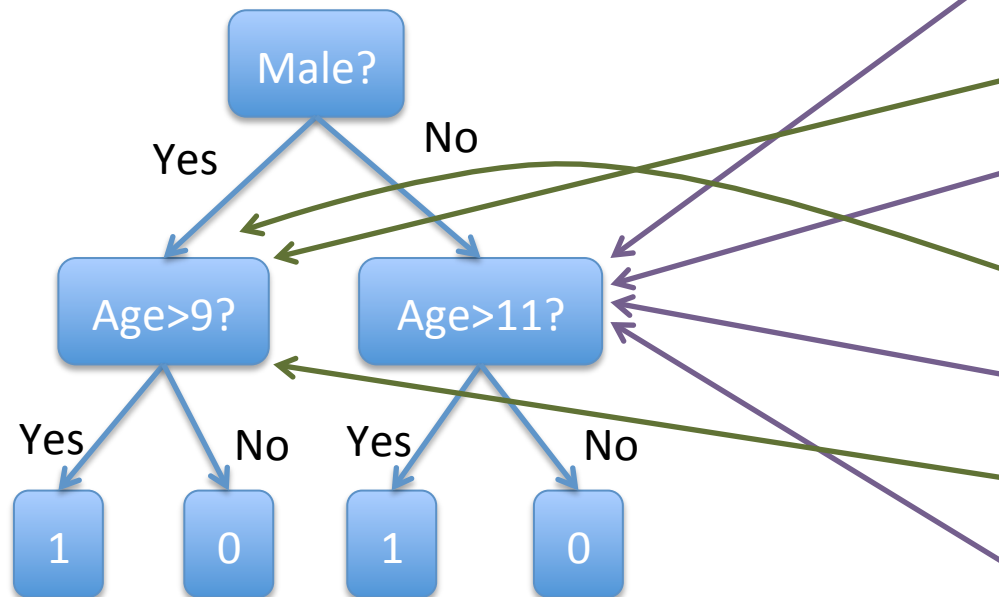


Name	Age	Male?	Height > 55"
Alice	14	0	1
Bob	10	1	1
Carol	13	0	1
Dave	8	1	0
Erin	11	0	0
Frank	9	1	1
Gena	10	0	0

A red bracket labeled 'S' spans the entire table. A red bracket labeled 'x' spans the 'Age' and 'Male?' columns. A red bracket labeled 'y' spans the 'Height > 55"' column. A red arrow points from the 'S' bracket to the 'Male?' node in the decision tree.

Decision Tree Training

Every node = partition/subset of S
Every Layer = complete partitioning of S
Children = complete partitioning of parent



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

xy

Thought Experiment

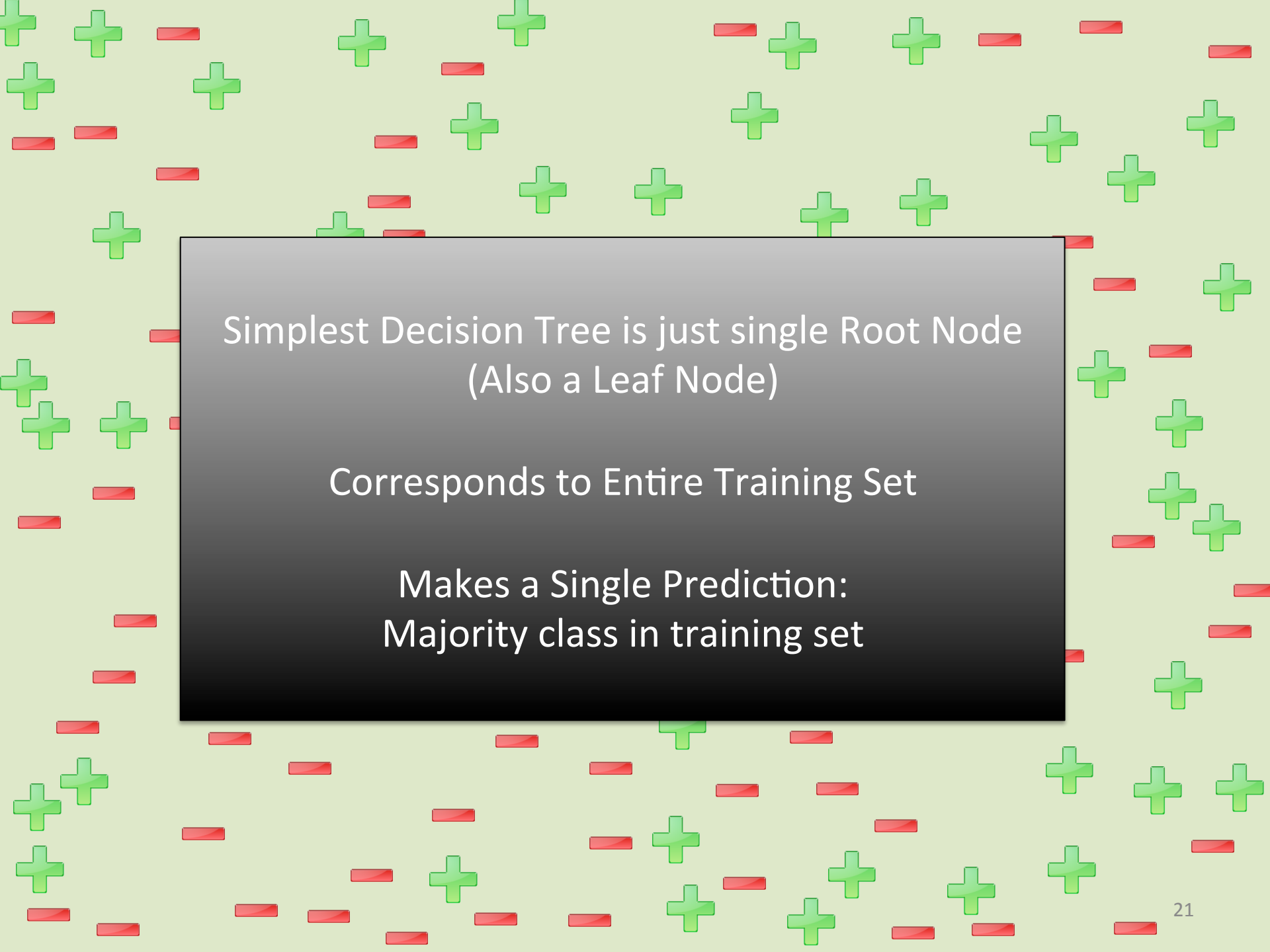
- What if just one node?
 - (I.e., just root node)
 - No queries
 - Single prediction for all data



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



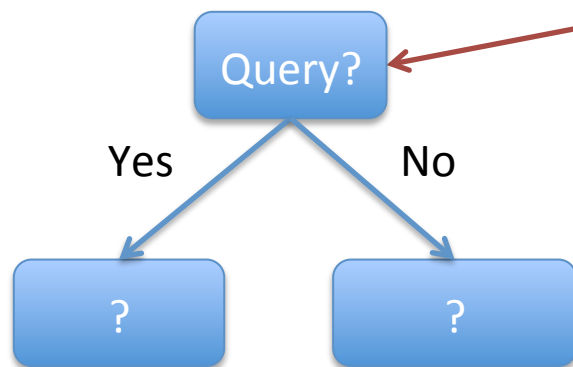
Simplest Decision Tree is just single Root Node
(Also a Leaf Node)

Corresponds to Entire Training Set

Makes a Single Prediction:
Majority class in training set

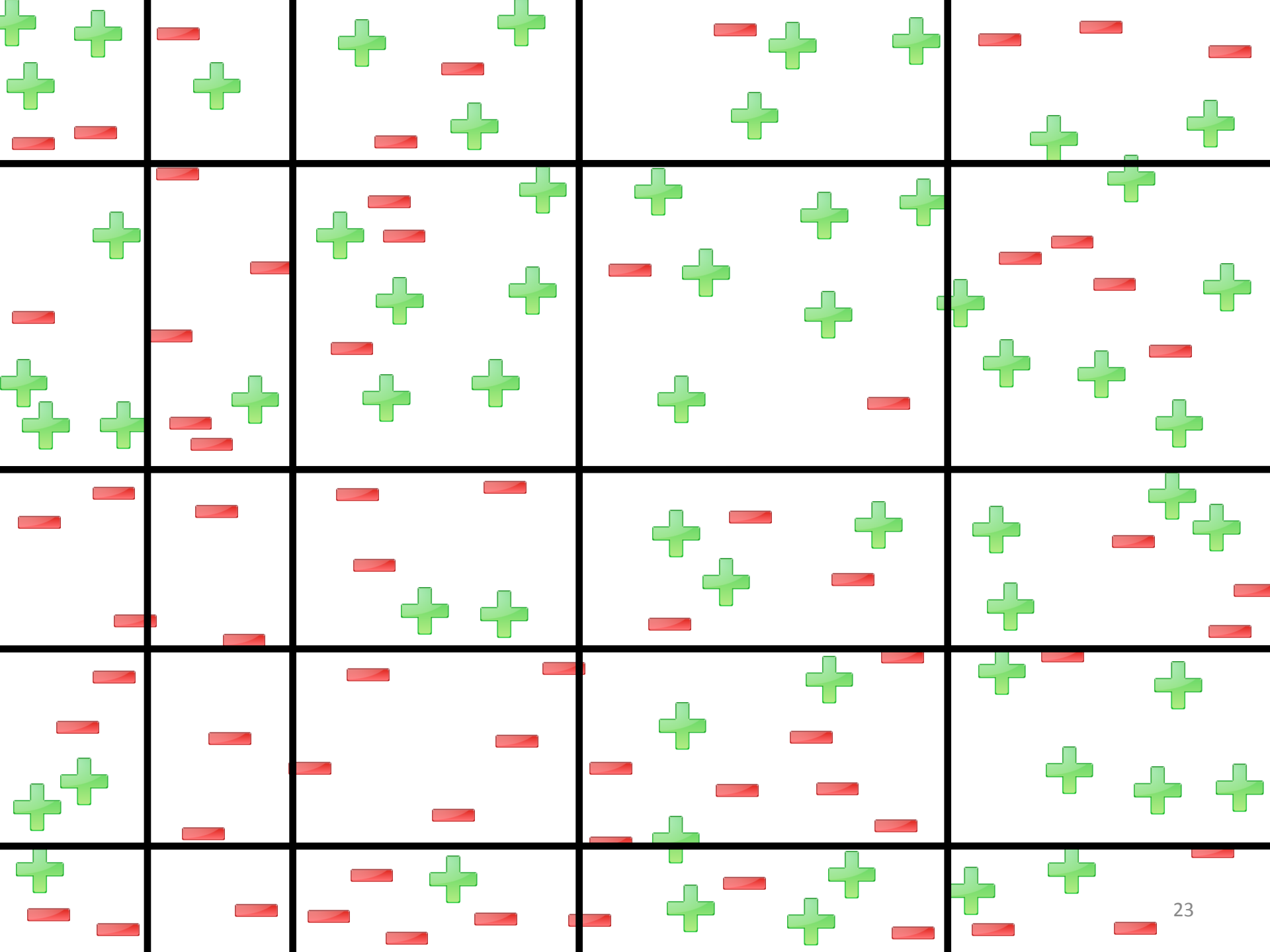
Thought Experiment Continued

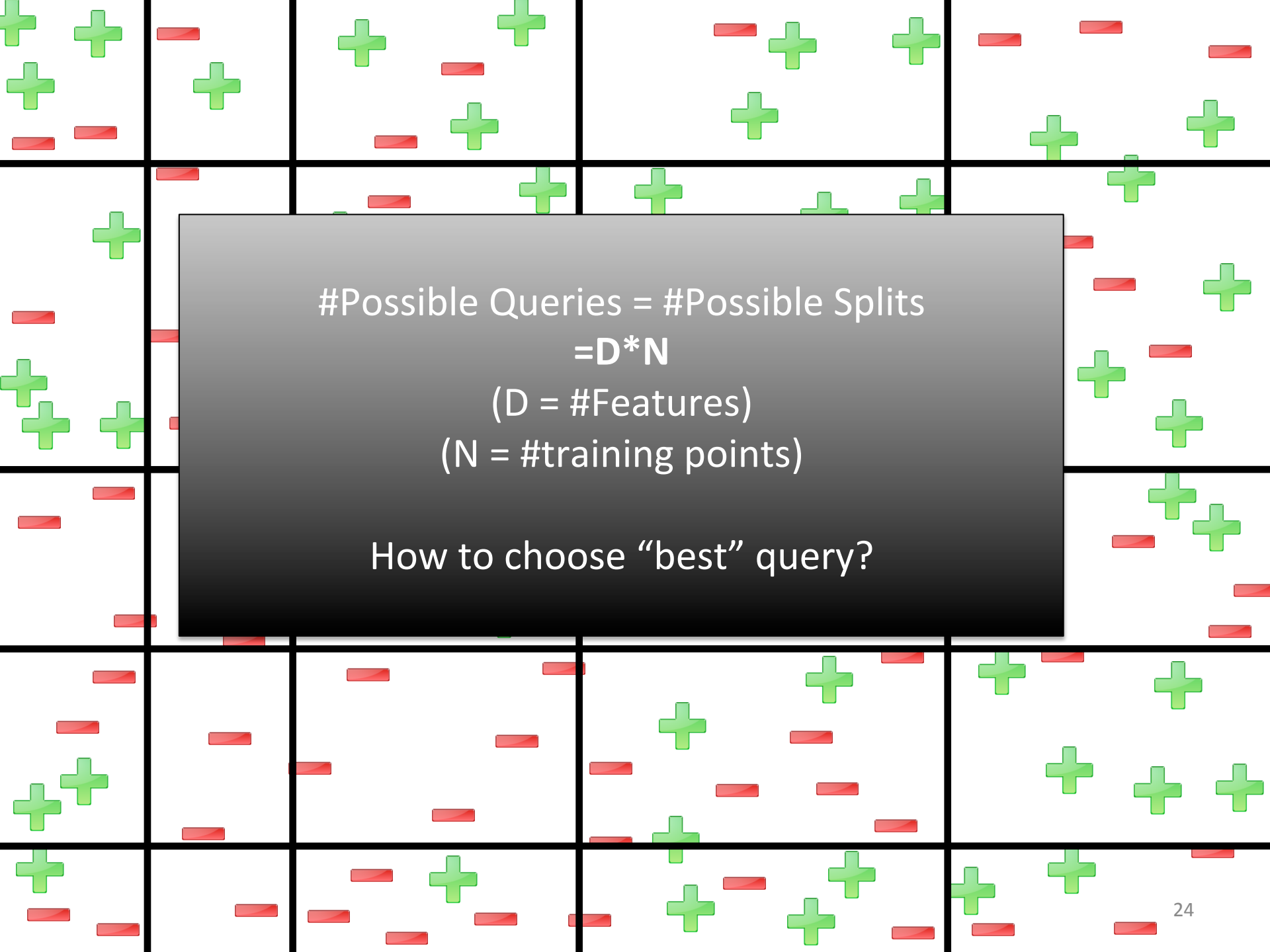
- What if 2 Levels?
 - (I.e., root node + 2 children)
 - Single query (which one?)
 - 2 predictions
 - **How many possible queries?**



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

A red bracket labeled 'S' spans the entire table. Below the table, a red bracket labeled 'x' spans the 'Age' and 'Male?' columns, and another red bracket labeled 'y' spans the 'Height > 55"' column.





#Possible Queries = #Possible Splits
 $= D * N$

(D = #Features)
(N = #training points)

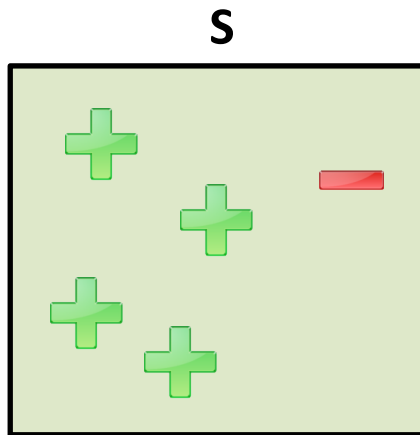
How to choose “best” query?

Impurity

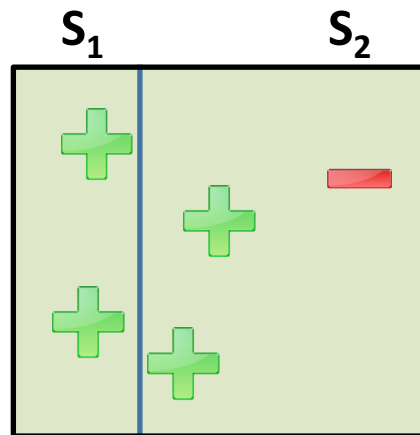
- Define impurity function:

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

Classification Error
of best single prediction



$$L(S) = 1$$



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

Impurity Reduction = 0

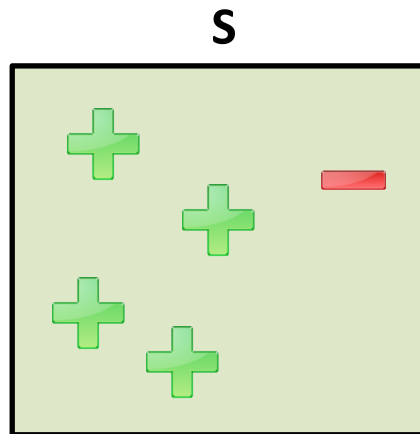
**No Benefit From
This Split!**

Impurity

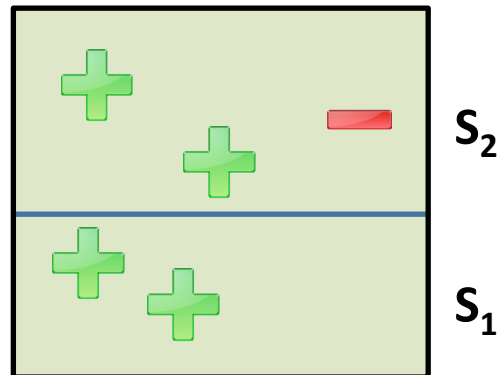
- Define impurity function:

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

Classification Error
of best single prediction



$$L(S) = 1$$



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

Impurity Reduction = 0

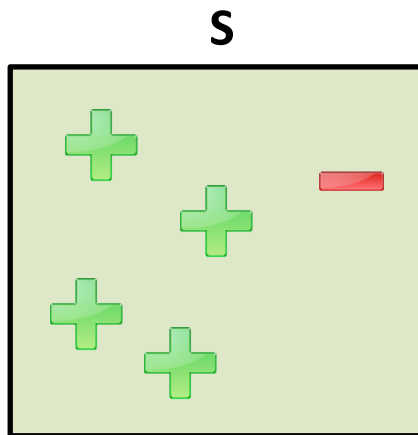
No Benefit From This Split!

Impurity

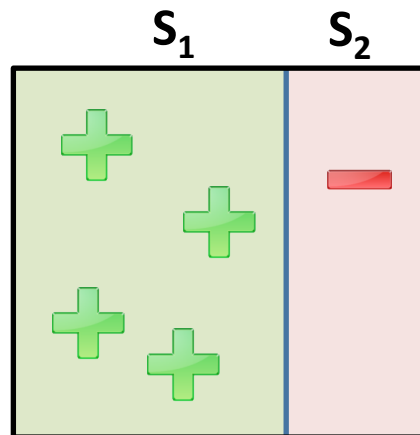
- Define impurity function:

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

Classification Error
of best single prediction



$$L(S) = 1$$



$$L(S_1) = 0$$

$$L(S_2) = 0$$

Impurity
Reduction = 1

**Choose Split with
largest impurity
reduction!**

Impurity = Loss Function

- **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)

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

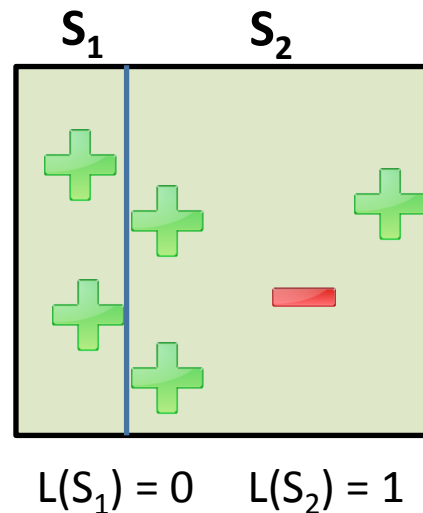
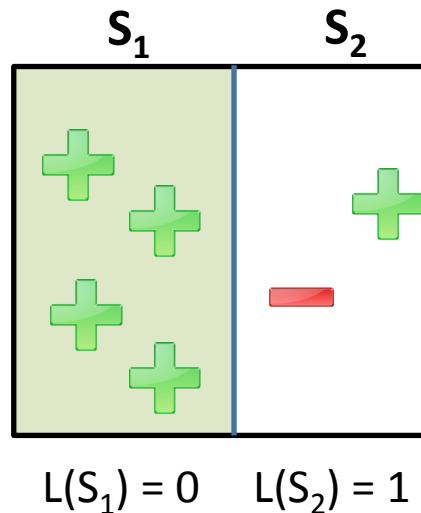
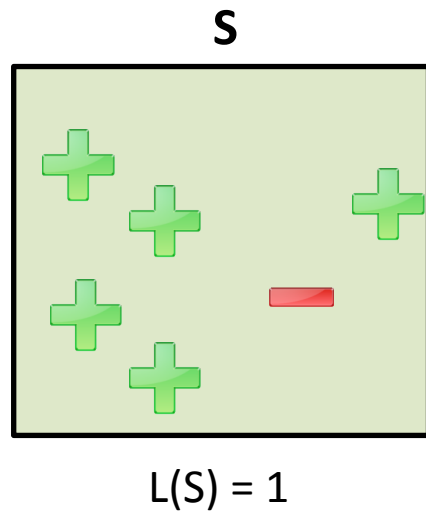
Classification Error on S'

Problems with 0/1 Loss

- What split best reduces impurity?

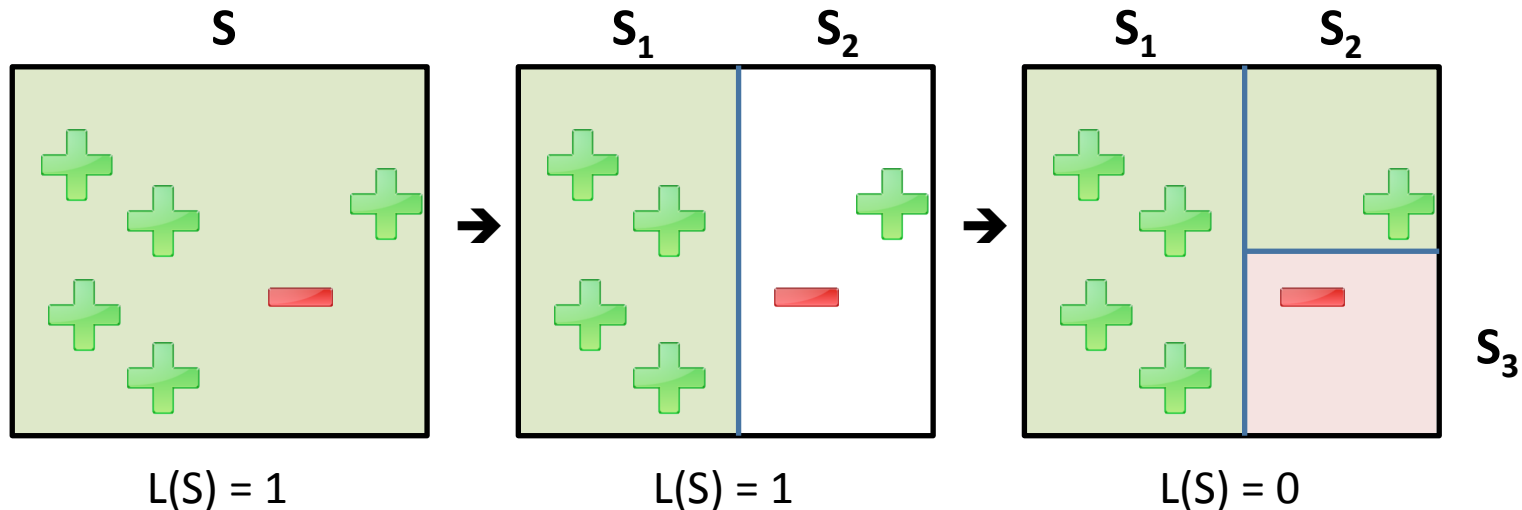
$$L(S') = \min_{\hat{y} \in \{0,1\}} \sum_{(x,y) \in S'} 1_{[\hat{y} \neq y]}$$

**All Partitionings Give Same
Impurity Reduction!**



Problems with 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...

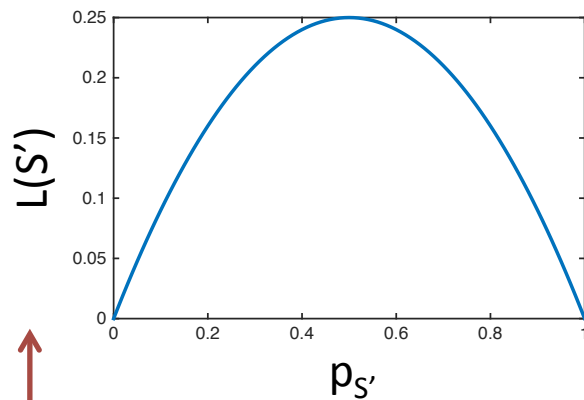


Surrogate Impurity Measures

- 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



Assuming $|S'|=1$

Worst Purity

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

$P = 1, L(S') = |S'| * 0$

$P = 0, L(S') = |S'| * 0$

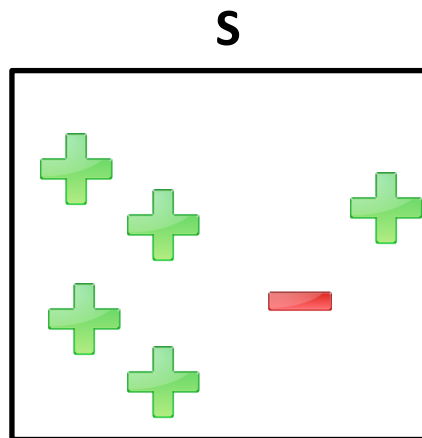
Perfect Purity

Bernoulli Variance as Impurity

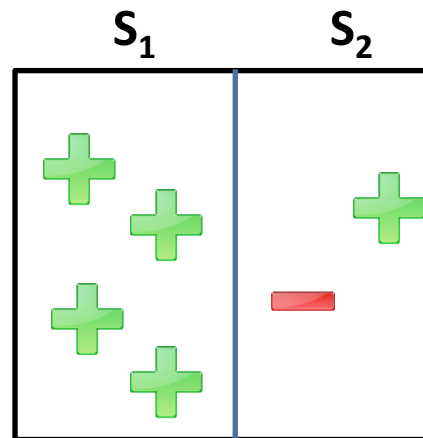
- What split best reduces impurity?

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

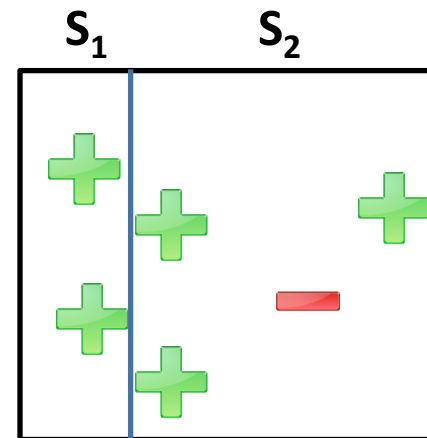
$p_{S'}$ = fraction of S' that are positive examples



$$L(S) = 5/6$$



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

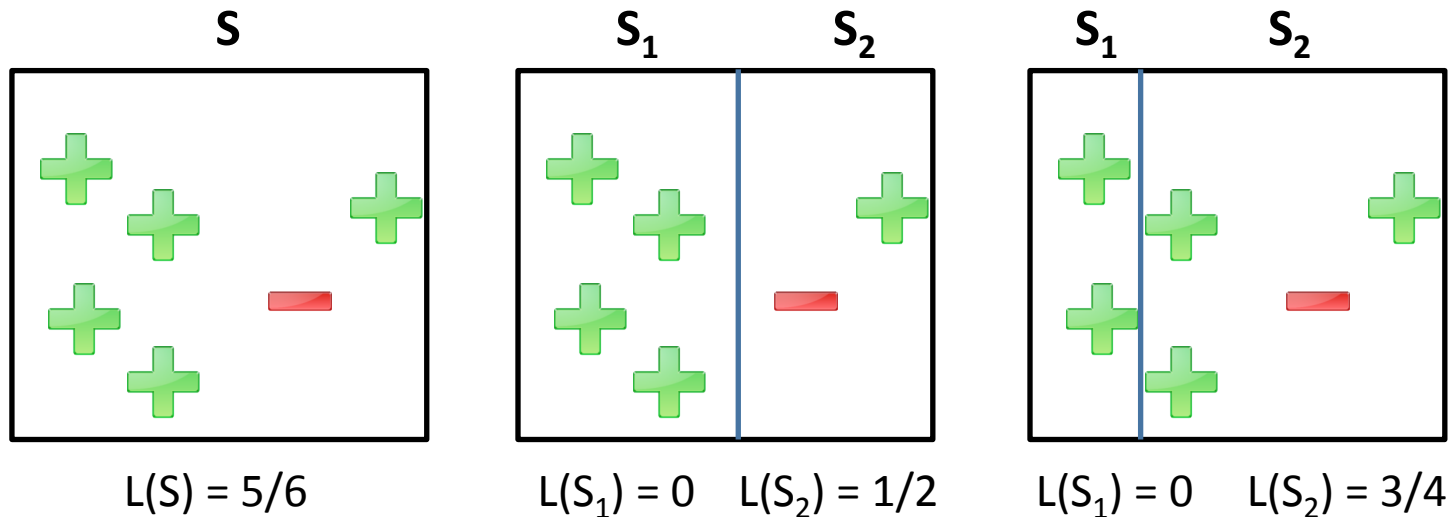


$$L(S_1) = 0 \quad L(S_2) = 3/4$$

Best!

Interpretation of 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



Best!

Other Impurity Measures

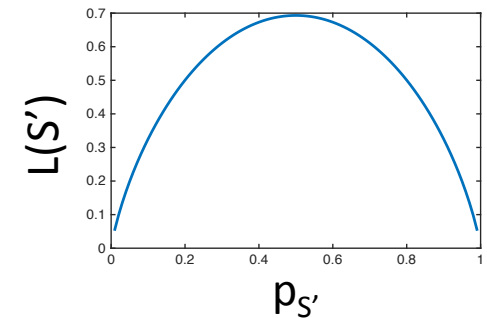
Define: $0 \cdot \log(0) = 0$

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

- aka: Information Gain:

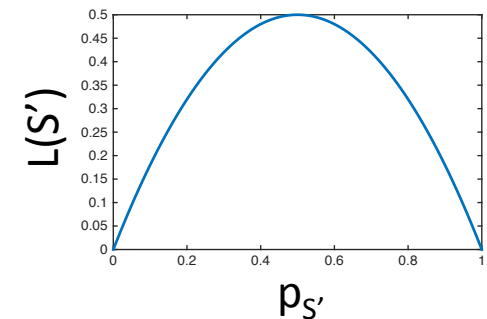
$$IG(A, B | S') = L(S') - L(A) - L(B)$$

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



- Gini Index:

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



See also: <http://www.ise.bgu.ac.il/faculty/liorr/hbchap9.pdf>
(Terminology is slightly different.)

Other Impurity Measures

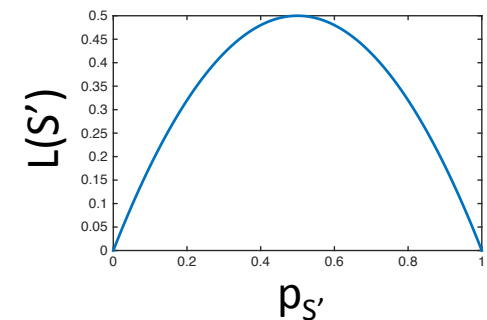
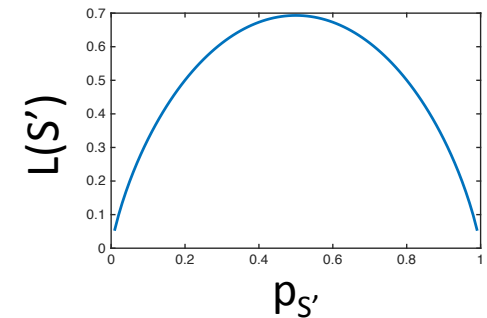
Define: $0 \cdot \log(0) = 0$

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

– aka: Information Gain:

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

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



Most Good Impurity Measures
Look Qualitatively The Same!

See also: <http://www.ise.bgu.ac.il/faculty/liorr/hbchap9.pdf>
(Terminology is slightly different.)

Top-Down Training

- Define impurity measure $L(S')$
 - E.g., $L(S') = \text{Bernoulli Variance}$

Loop: Choose split with greatest impurity reduction (over all leaf nodes).

Repeat: until stopping condition.

Step 1:
 $L(S) = 12/7$

1

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

Top-Down Training

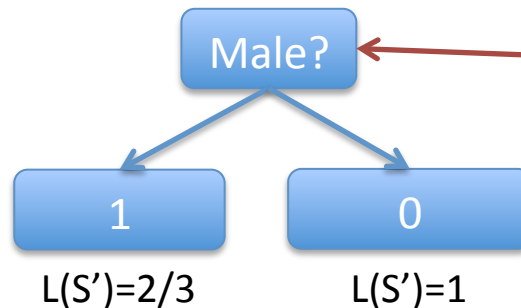
- Define impurity measure $L(S')$
 - E.g., $L(S') = \text{Bernoulli Variance}$

Loop: Choose split with greatest impurity reduction (over all leaf nodes).

Repeat: until stopping condition.

Step 1:
 $L(S) = 12/7$

Step 2:
 $L(S) = 5/3$



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

A red bracket on the left side of the table groups all rows. Below the table, a red bracket labeled "X" spans the "Male?" and "Height > 55\"" columns, and another red bracket labeled "Y" spans the "Height > 55\"" column.

Top-Down Training

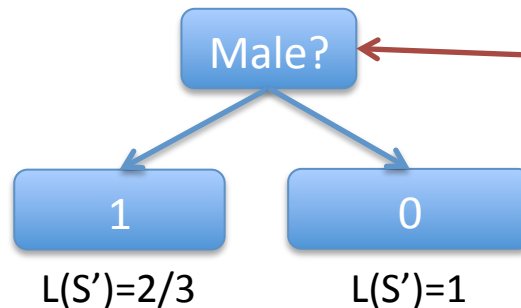
- Define impurity measure $L(S')$
 - E.g., $L(S') = \text{Bernoulli Variance}$

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.

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

A red bracket labeled "X" is under the "Male?" column, and a red bracket labeled "Y" is under the "Height > 55\"" column.

Top-Down Training

- Define impurity measure $L(S')$
 - E.g., $L(S') = \text{Bernoulli Variance}$

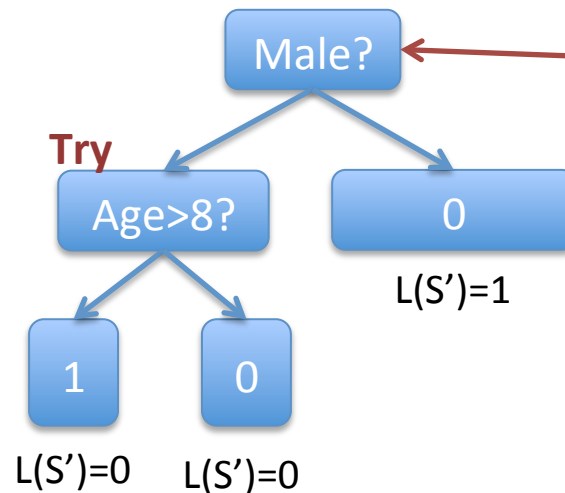
Loop: Choose split with greatest impurity reduction (over all leaf nodes).

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

Top-Down Training

- Define impurity measure $L(S')$
 - E.g., $L(S') = \text{Bernoulli Variance}$

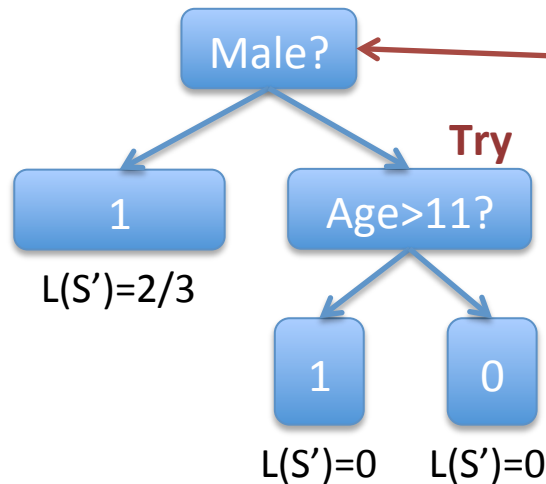
Loop: Choose split with greatest impurity reduction (over all leaf nodes).

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

Below the table, a red bracket labeled "x" spans the "Age" and "Male?" columns. Another red bracket labeled "y" spans the "Height > 55\"" column.

Top-Down Training

- Define impurity measure $L(S')$
 - E.g., $L(S') = \text{Bernoulli Variance}$

Loop: Choose split with greatest impurity reduction (over all leaf nodes).

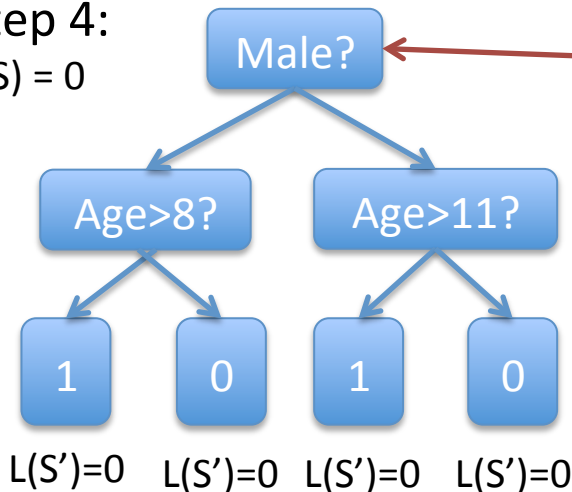
Repeat: until stopping condition.

Step 1:
 $L(S) = 12/7$

Step 2:
 $L(S) = 5/3$

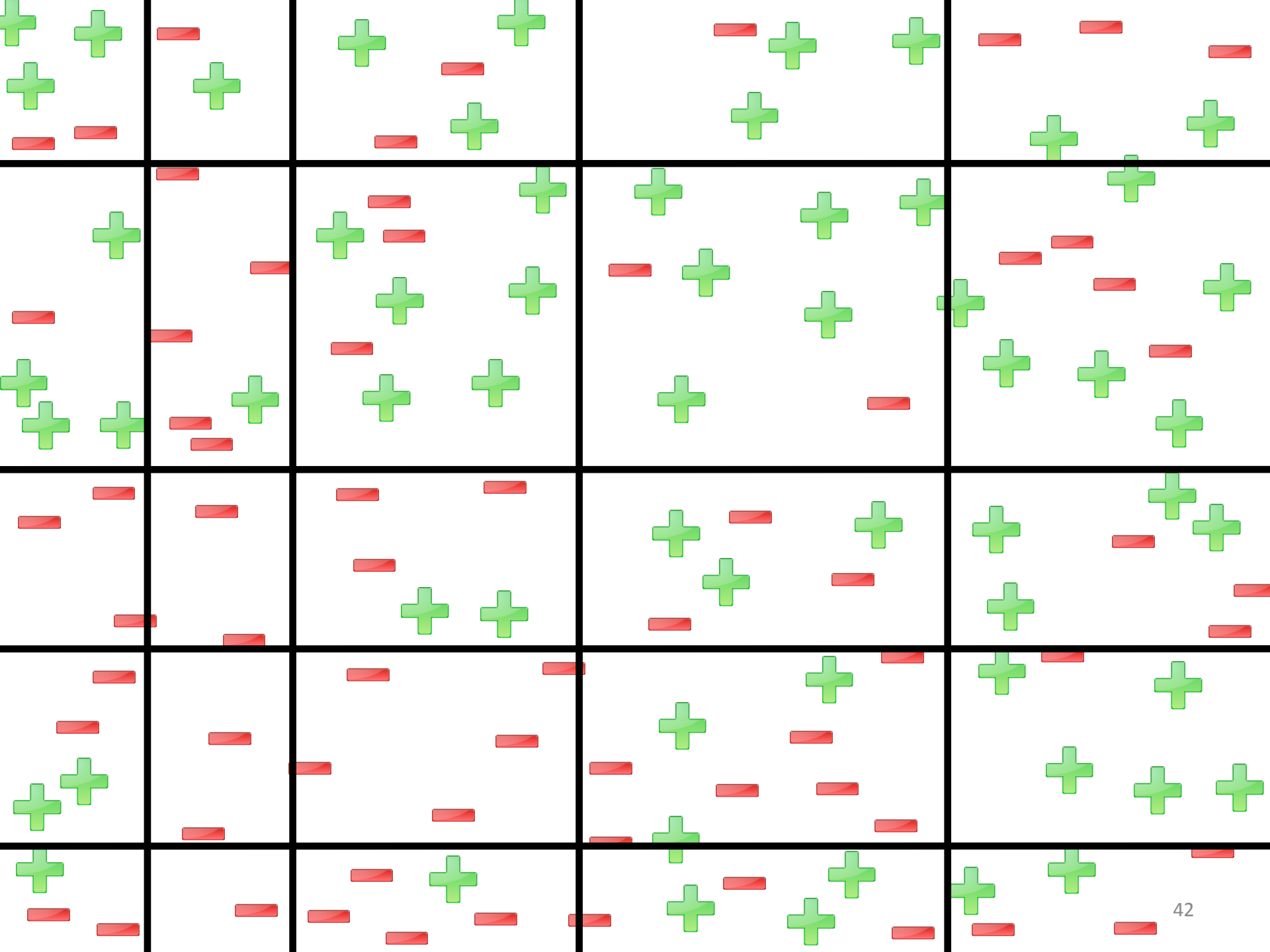
Step 3:
 $L(S) = 2/3$

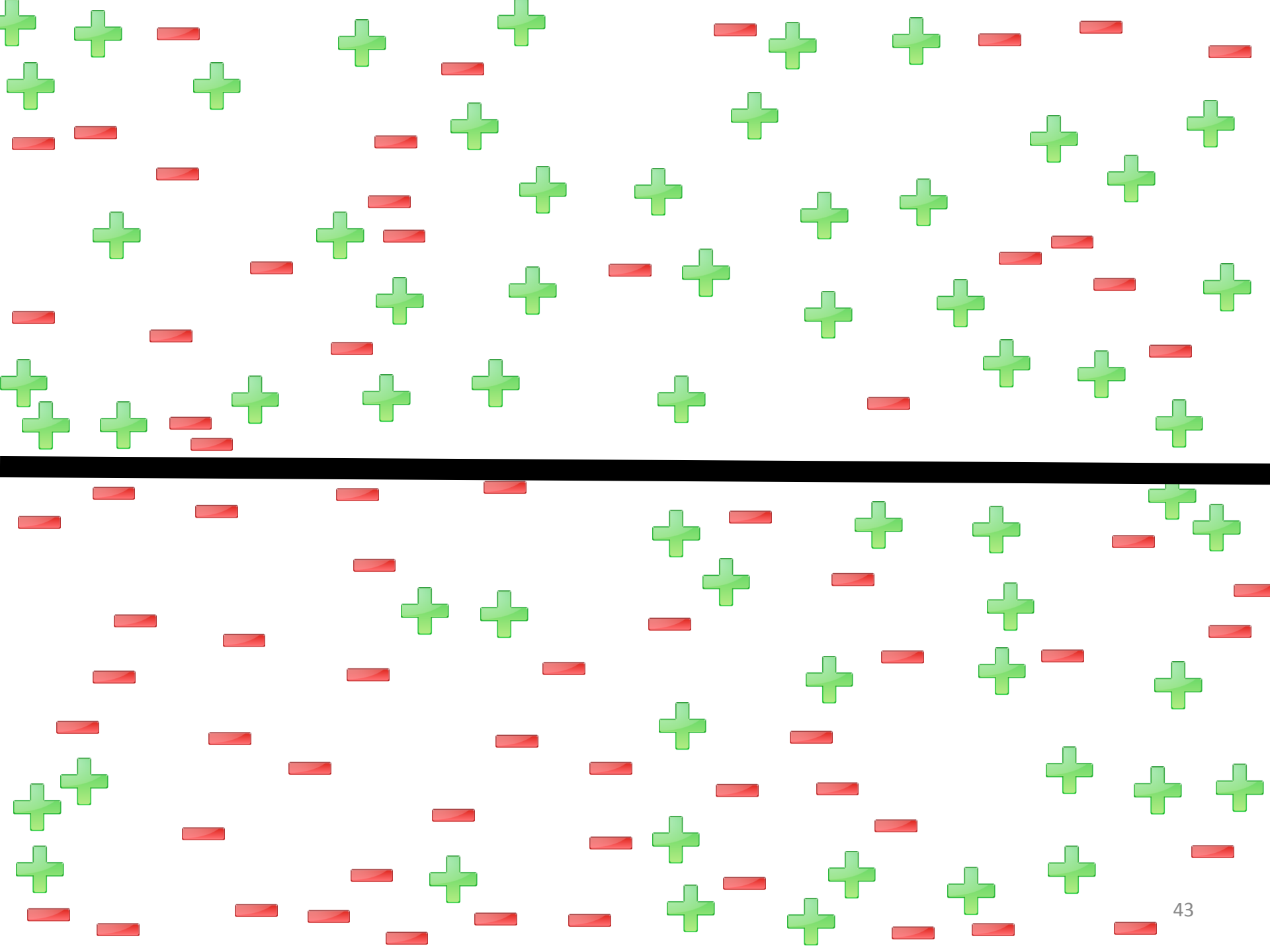
Step 4:
 $L(S) = 0$

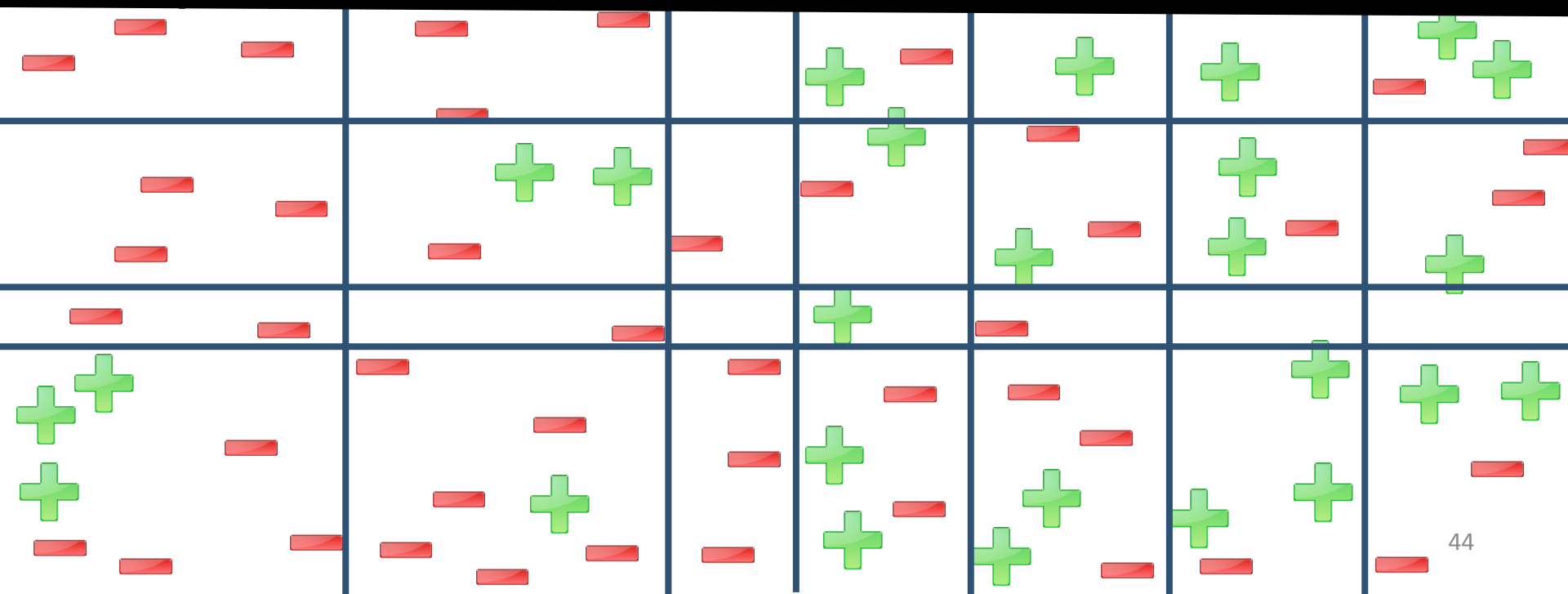
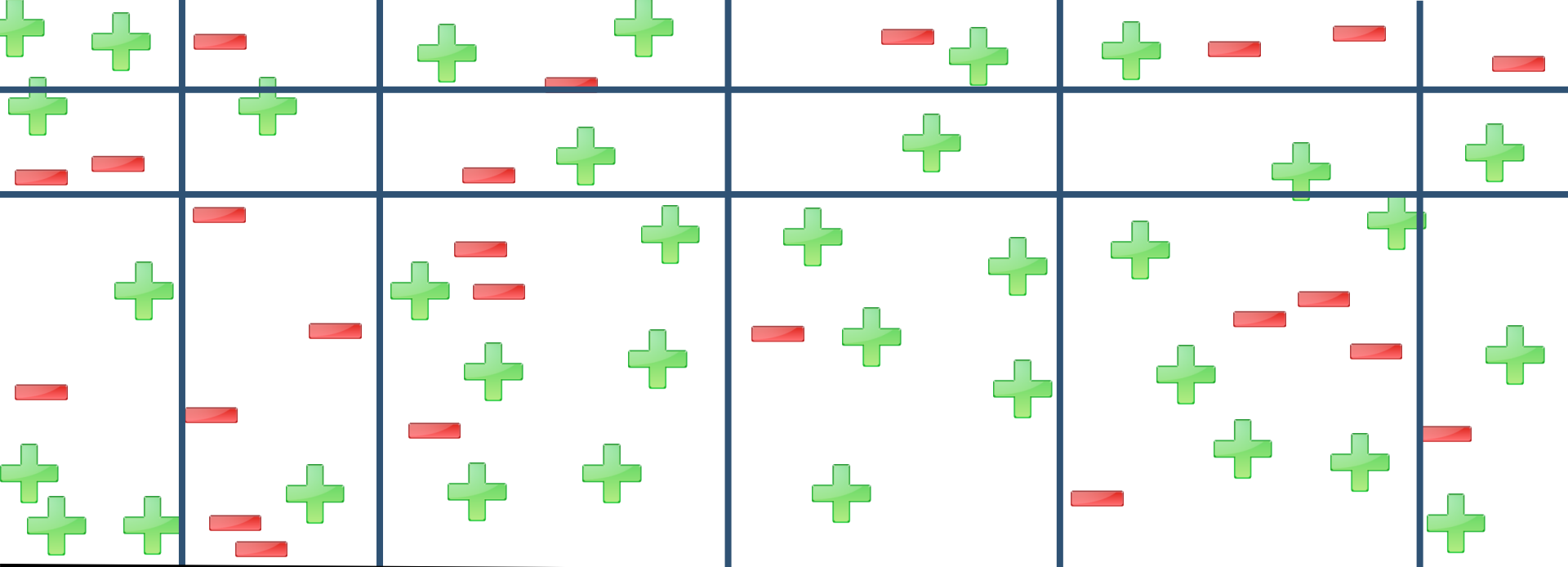


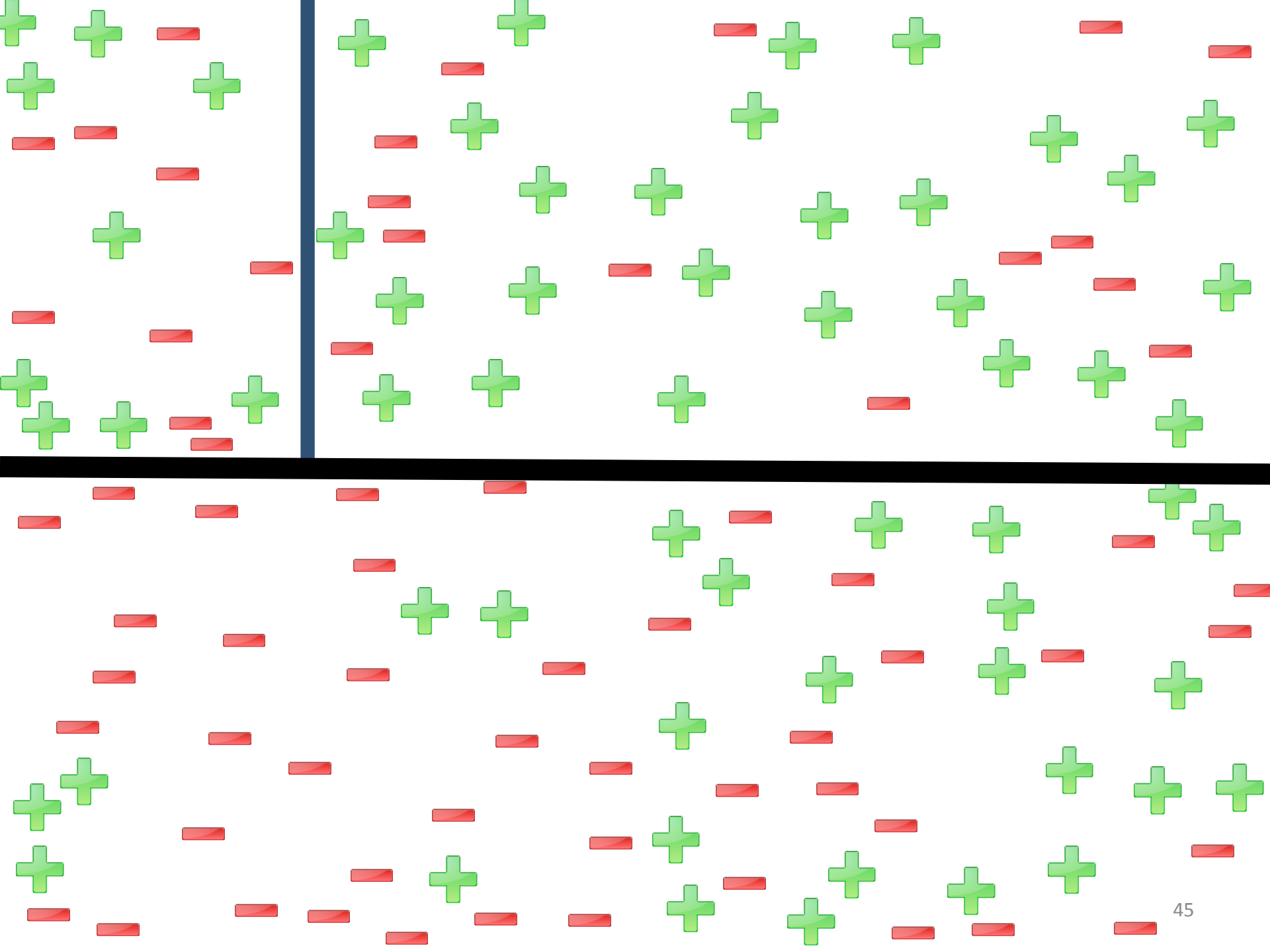
Name	Age	Male?	Height > 55"
Alice	14	0	1
Bob	10	1	1
Carol	13	0	1
Dave	8	1	0
Erin	11	0	0
Frank	9	1	1
Gena	10	0	0

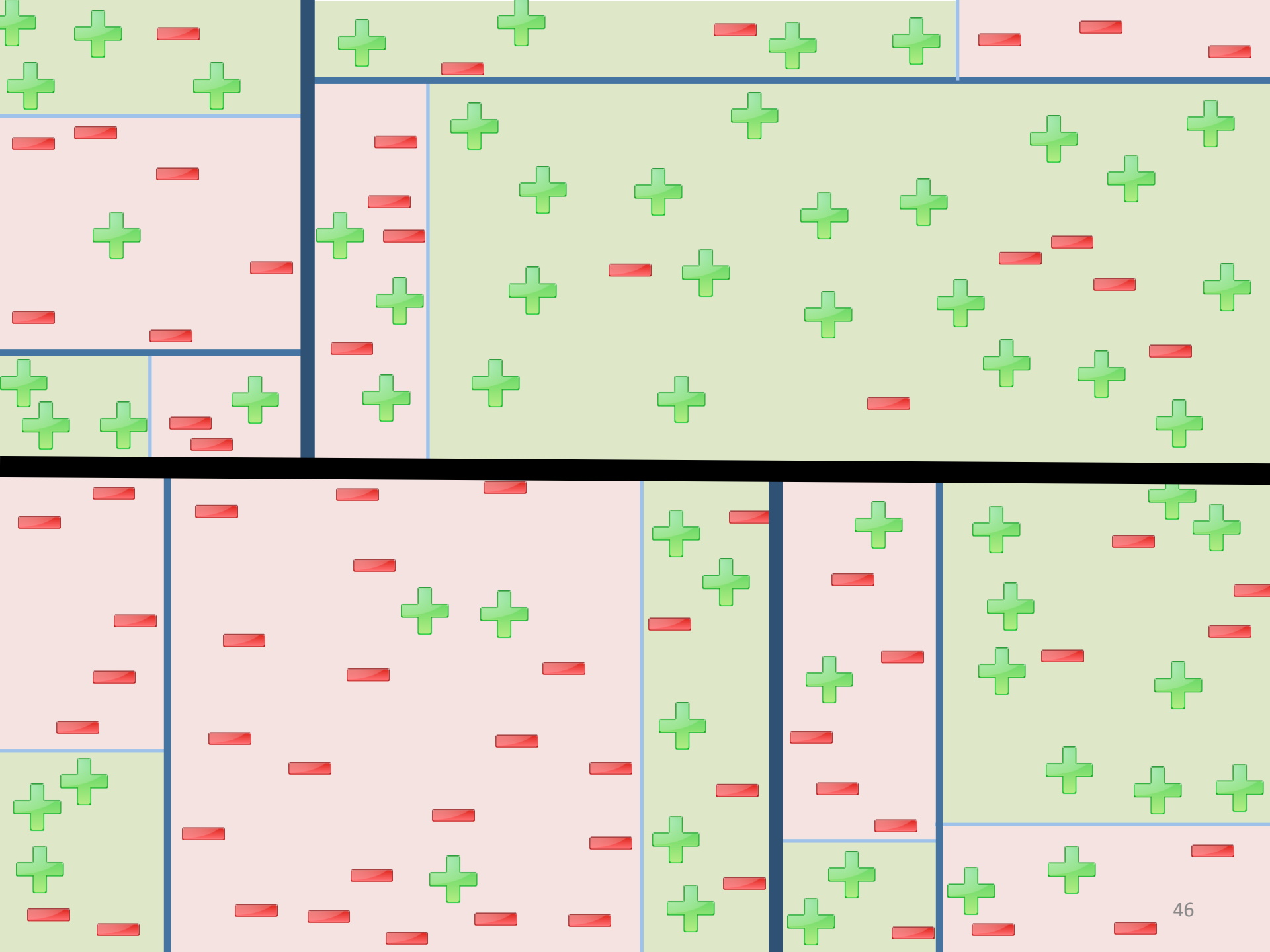
x
y





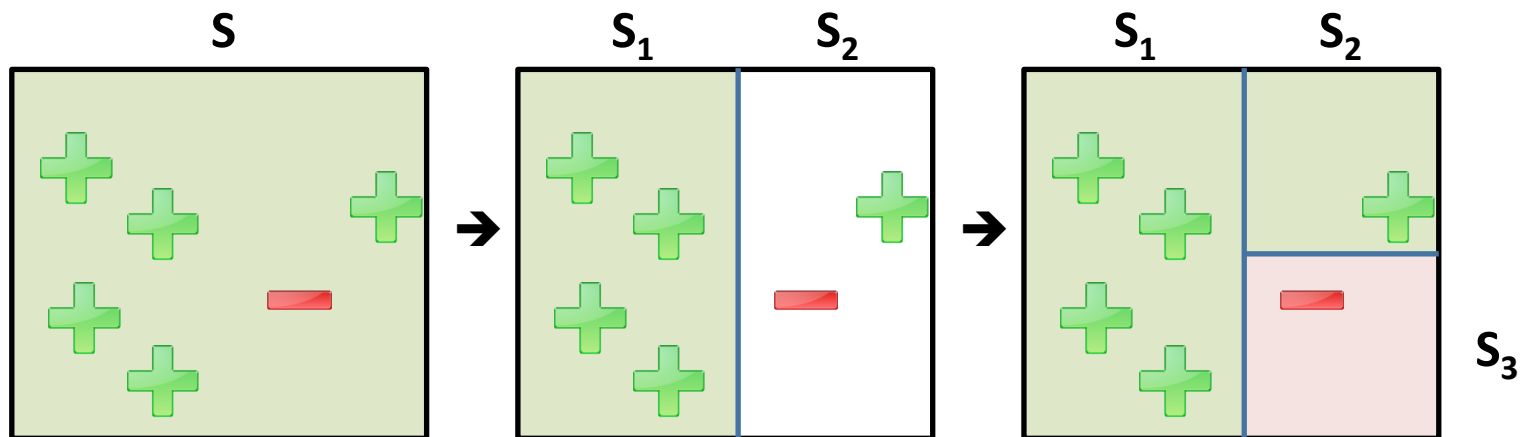






Properties of Top-Down Training

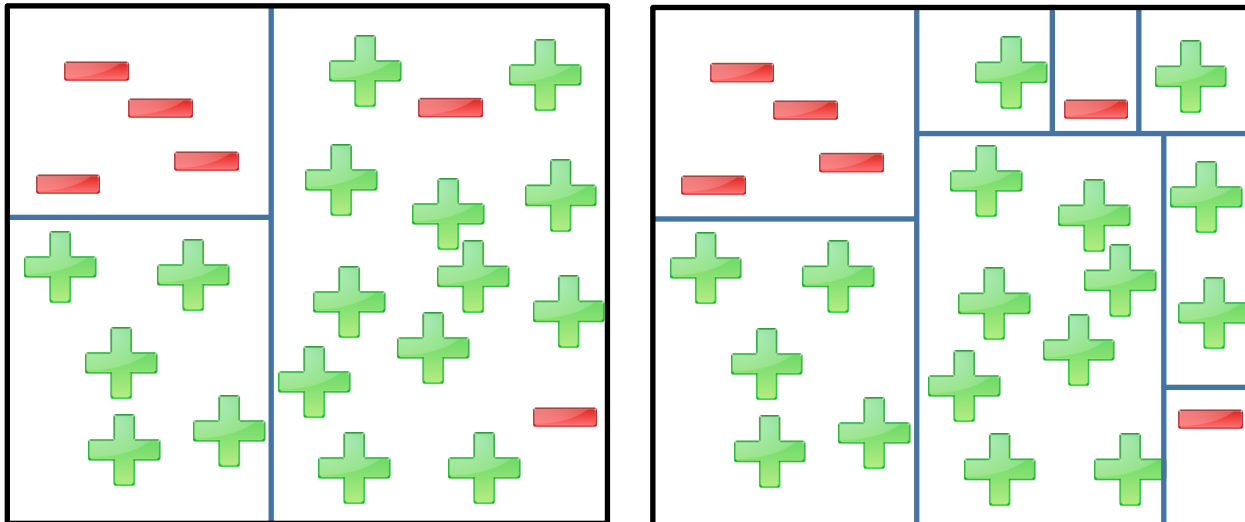
- Every intermediate step is a decision tree
 - You can stop any time and have a model
- Greedy algorithm
 - Doesn't backtrack
 - Cannot reconsider different higher-level splits.



When to Stop?

- In 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?



Stopping Conditions (Regularizers)

- **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 $\delta\%$.

Pseudocode for Training

Algorithm 1 TREE(): Initialize Decision (Sub-)Tree Data Structure

```

1: input:  $S$                                      //data partition
2: input:  $L$                                      //loss function
3: Initialize data structure  $\mathcal{T}$ :
4:    $\mathcal{T}.data \leftarrow S$                      // pointer to training data partition
5:    $\mathcal{T}.q \leftarrow \text{NULL}$                      // decision query
6:    $\mathcal{T}.left \leftarrow \text{NULL}$  // subtree for positive query response
7:    $\mathcal{T}.right \leftarrow \text{NULL}$  // subtree for negative query response
8:    $\mathcal{T}.\ell \leftarrow L(S)$  // impurity/loss on training data partition
9: return:  $\mathcal{T}$ 

```

Stopping condition is minimum
leaf node size: $N_{\min}$

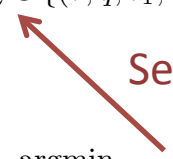
Algorithm 3 TRAIN(): Top-Down Decision Tree Training

```

1: input:  $S, \mathcal{Q}, N_{\min}, L$ 
2:  $\mathcal{T} \leftarrow \text{TREE}(S)$                                      // root node
3: repeat
4:    $Q \leftarrow \emptyset$ 
5:   for every leaf node  $\tau$  in  $\mathcal{T}$  do
6:     for every  $q \in \mathcal{Q}$  do
7:        $S_1 \leftarrow \{(\hat{\mathbf{x}}, \hat{\mathbf{y}}) \in \tau.data \mid q(\hat{\mathbf{x}}) = 1\}$ 
8:        $S_2 \leftarrow \{(\hat{\mathbf{x}}, \hat{\mathbf{y}}) \in \tau.data \mid q(\hat{\mathbf{x}}) = 0\}$ 
9:       if  $|S_1| \geq N_{\min} \wedge |S_2| \geq N_{\min}$  then
10:         $\tau_1 \leftarrow \text{TREE}(S_1, L)$ 
11:         $\tau_2 \leftarrow \text{TREE}(S_2, L)$ 
12:         $Q \leftarrow Q \cup \{(\tau, q, \tau_1, \tau_2)\}$ 
13:      end if
14:    end for
15:  end for
16:  if  $|Q| > 0$  then
17:     $(\tau, q, \tau_1, \tau_2) \leftarrow \underset{(\tau', q', \tau'_1, \tau'_2)}{\text{argmin}} \tau'.\ell - (\tau'_1.\ell + \tau'_2.\ell)$ 
18:     $\tau.q \leftarrow q$ 
19:     $\tau.left \leftarrow \tau_1$ 
20:     $\tau.right \leftarrow \tau_2$ 
21:  end if
22: until  $|Q| = 0$ 
23: return:  $\mathcal{T}$ 

```

Select from Q



Classification vs Regression

Classification	Regression
Labels are $\{0,1\}$	Labels are Real Valued
Predict Majority Class in Leaf Node	Predict Mean of Labels in Leaf Node
Piecewise Constant Function Class	Piecewise Constant Function Class
Goal: minimize 0/1 Loss	Goal: minimize squared loss
Impurity based on fraction of positives vs negatives	Impurity = Squared Loss

Recap: Decision Tree Training

- 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.

Recap: Decision Trees

- 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

Bagging

(Bootstrap Aggregation)

Outline

- Recap: Bias/Variance Tradeoff
- Bagging
 - Method for minimizing variance
 - Not specific to Decision Trees
- Random Forests
 - Extension of Bagging
 - Specific to Decision Trees

Outline

- **Recap: Bias/Variance Tradeoff**
- Bagging
 - Method for minimizing variance
 - Not specific to Decision Trees
- Random Forests
 - Extension of Bagging
 - Specific to Decision Trees

Test Error

- **“True” distribution:** $P(x,y)$
 - Unknown to us
- **Train:** $h_S(x) = y$
 - Using training data: $S = \{(x_i, y_i)\}_{i=1}^N$
 - Sampled from $P(x,y)$
- **Test Error:**
$$L_P(h_S) = E_{(x,y) \sim P(x,y)} [L(y, h_S(x))]$$
- **Overfitting:** Test Error >> Training Error

True Distribution P(x,y)

Person	Age	Male?	Height > 55"
James	11	1	1
Jessica	14	0	1
Alice	14	0	1
Amy	12	0	1
Bob	10	1	1
Xavier	9	1	0
Cathy	9	0	1
Carol	13	0	1
Eugene	13	1	0
Rafael	12	1	1
Dave	8	1	0
Peter	9	1	0
Henry	13	1	0
Erin	11	0	0
Rose	7	0	0
Iain	8	1	1
Paulo	12	1	0
Margaret	10	0	1
Frank	9	1	1
Jill	13	0	0
Leon	10	1	0
Sarah	12	0	0
Gena	8	0	0
Patrick	5	1	1

⋮

Training Set S

Person	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	8	0	0



y

h(x)

Test Error:

$$\mathcal{L}(h) = E_{(x,y) \sim P(x,y)} [L(h(x), y)]$$

Bias-Variance Decomposition

$$E_S [L_P(h_S)] = E_S [E_{(x,y) \sim P(x,y)} [L(y, h_S(x))]]$$

- For squared error:

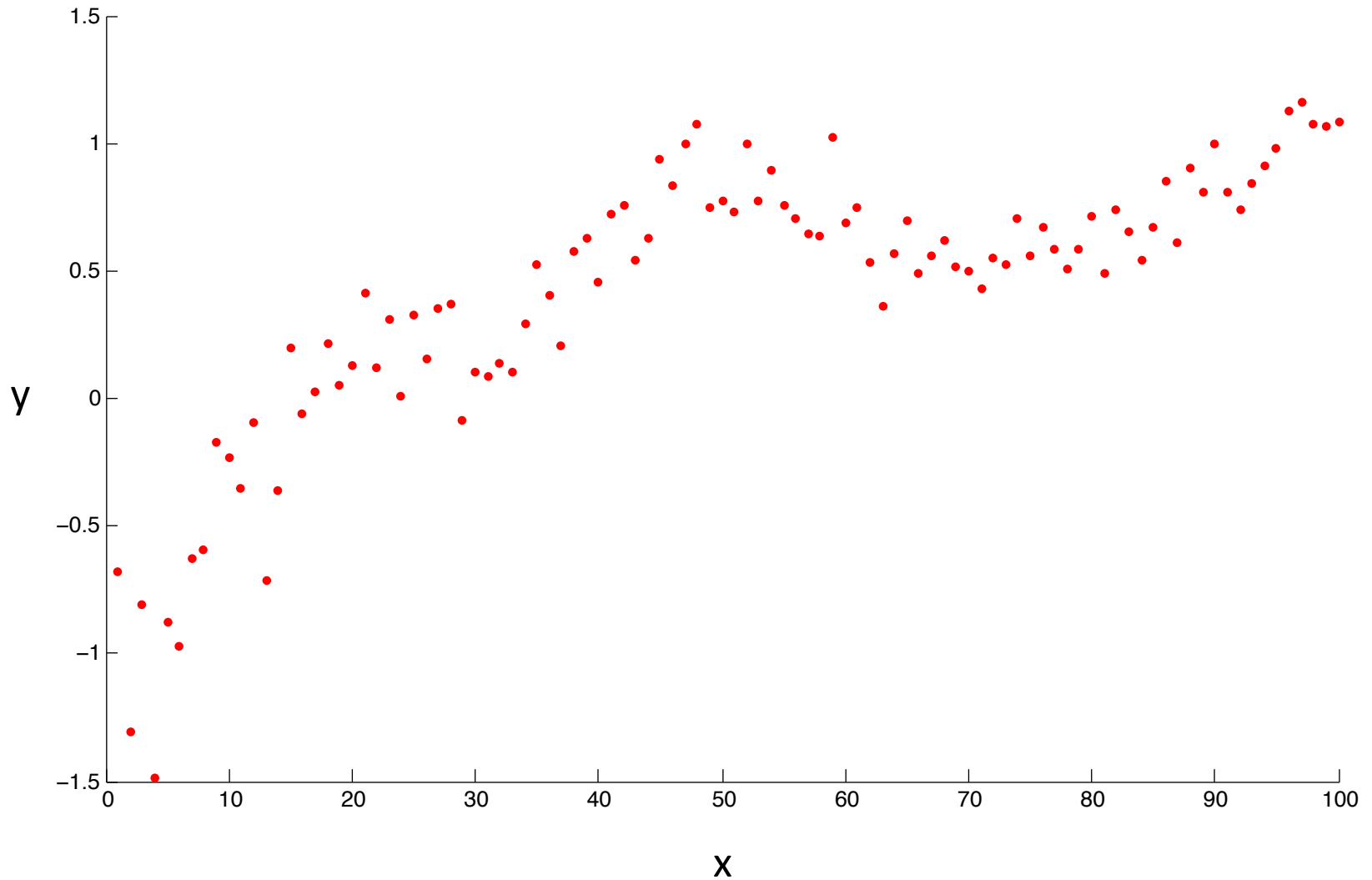
$$E_S [L_P(h_S)] = E_{(x,y) \sim P(x,y)} \left[\underbrace{E_S [(h_S(x) - H(x))^2]}_{\text{Variance Term}} + \underbrace{(H(x) - y)^2}_{\text{Bias Term}} \right]$$

$$H(x) = E_S [h_S(x)]$$

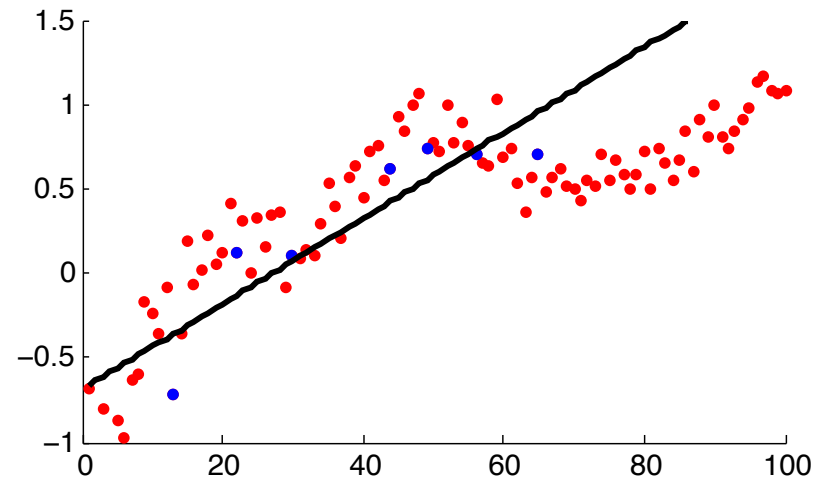
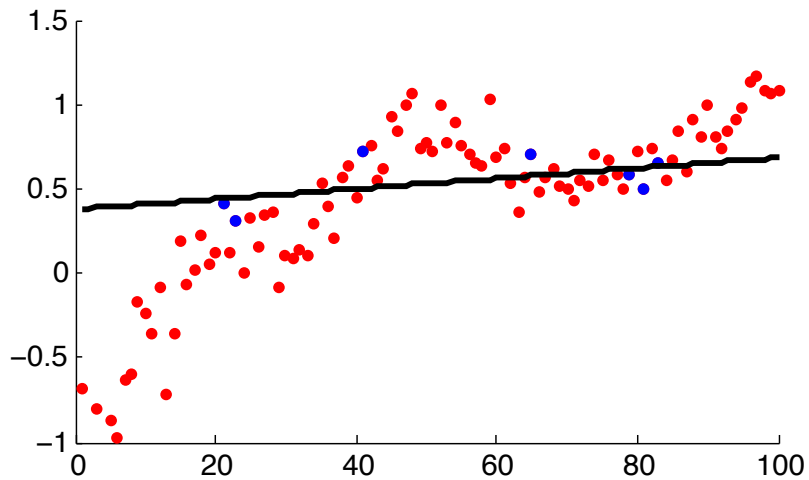
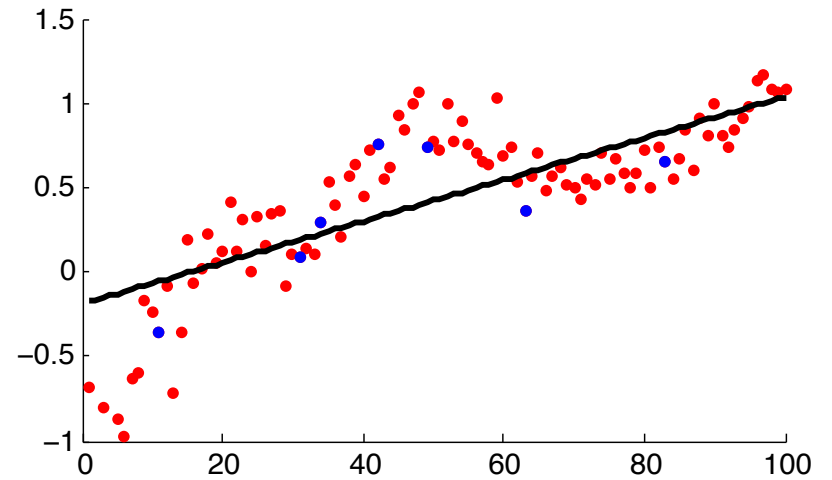
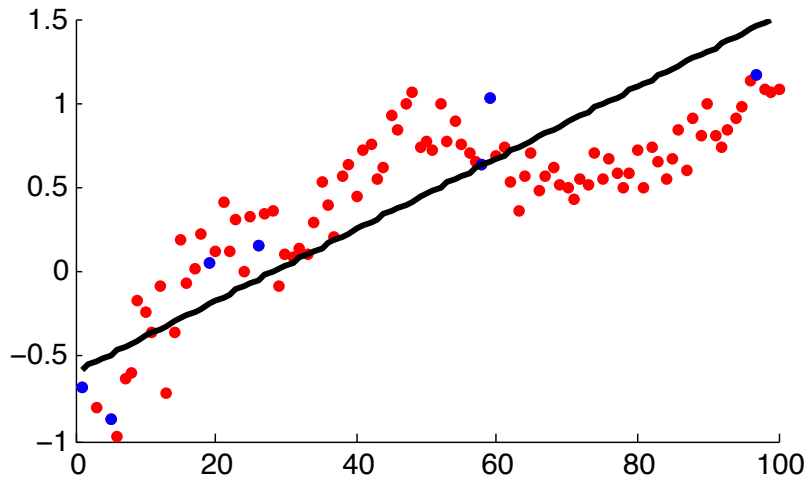


“Average prediction on x”

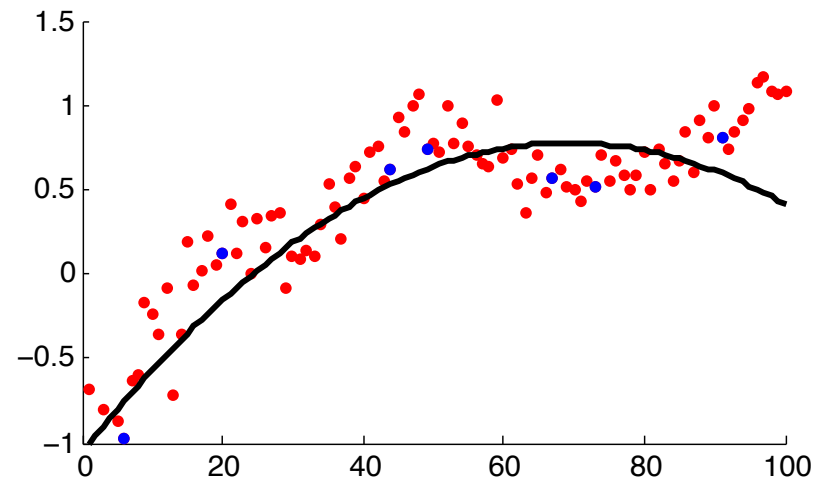
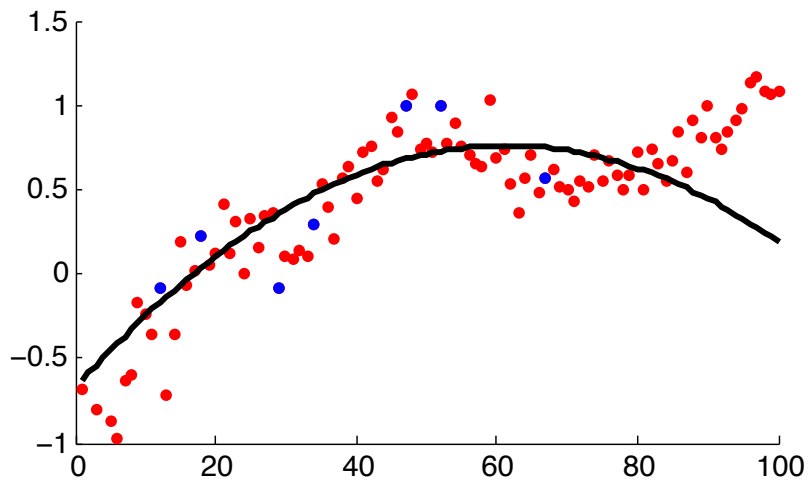
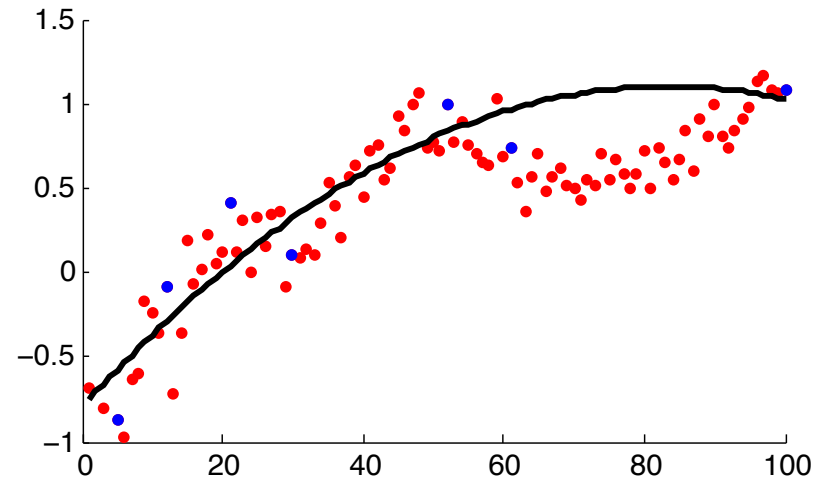
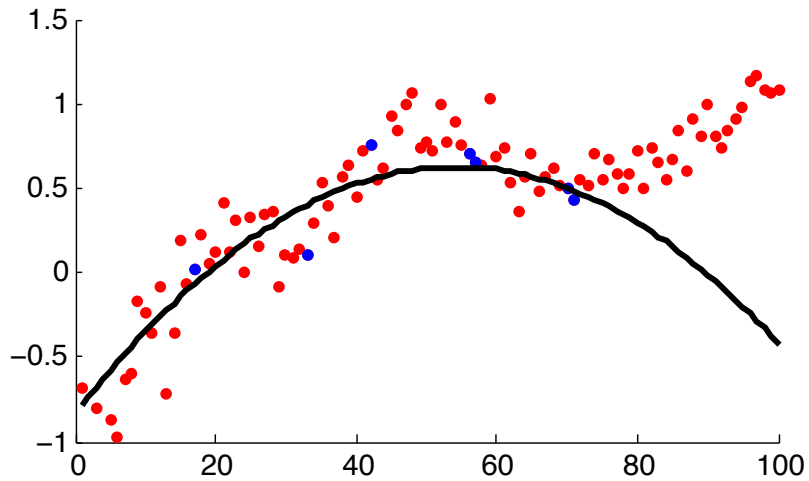
Example $P(x,y)$



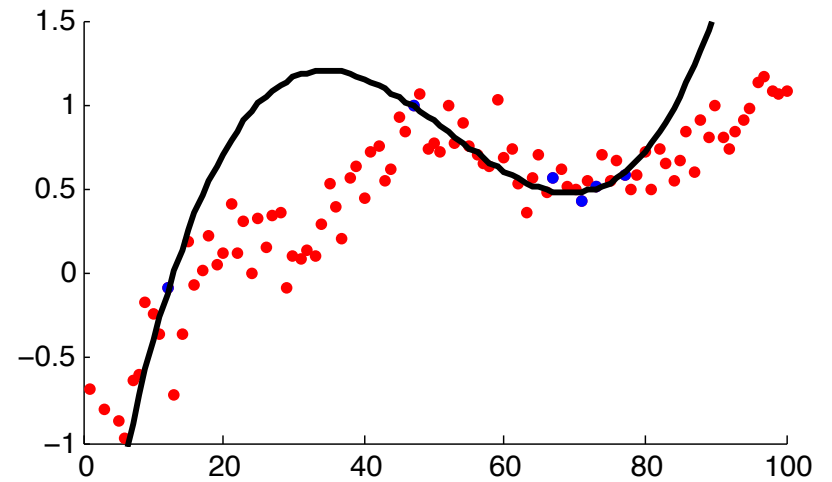
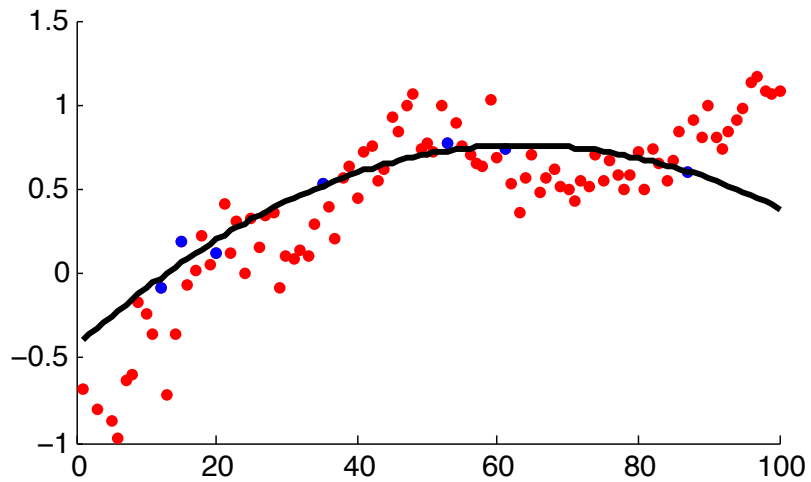
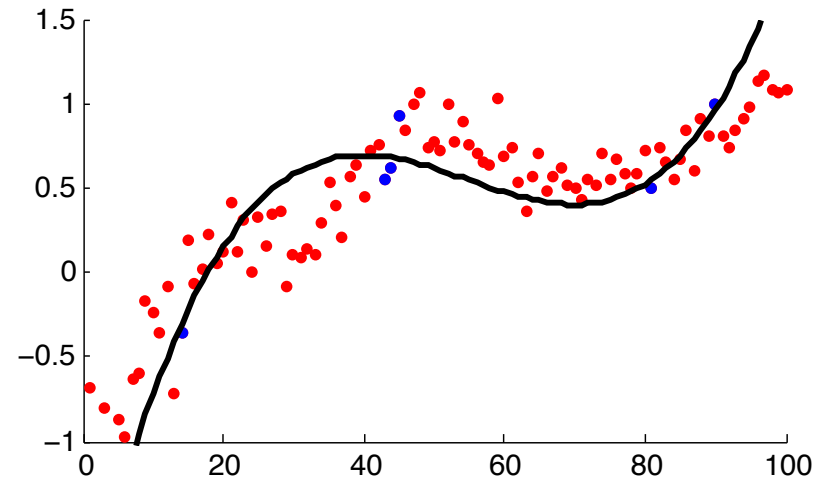
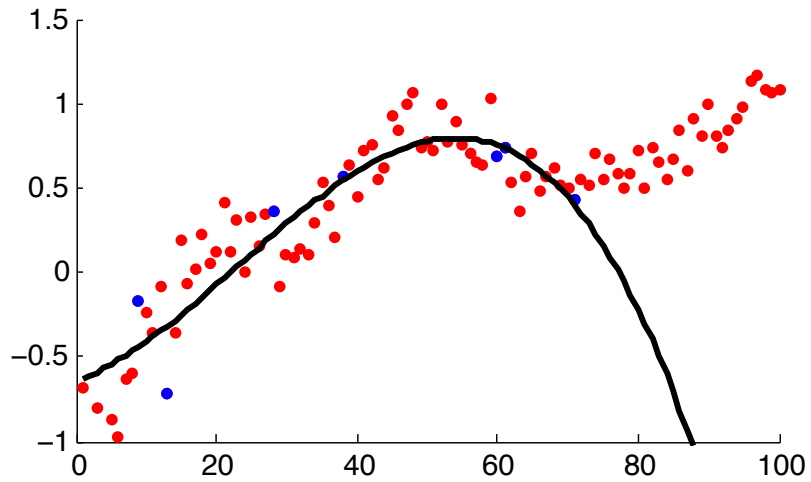
$h_s(x)$ Linear



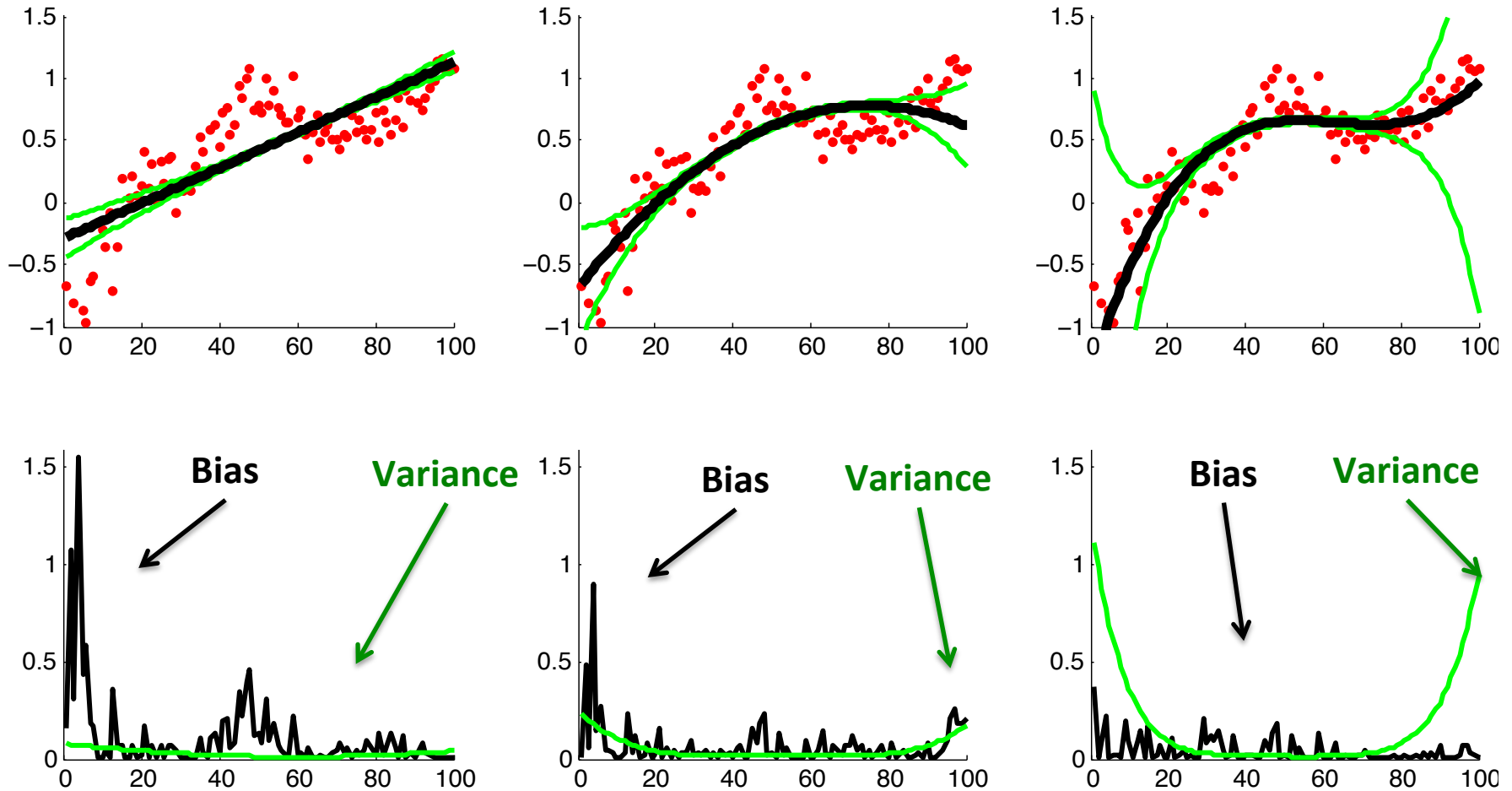
$h_s(x)$ Quadratic



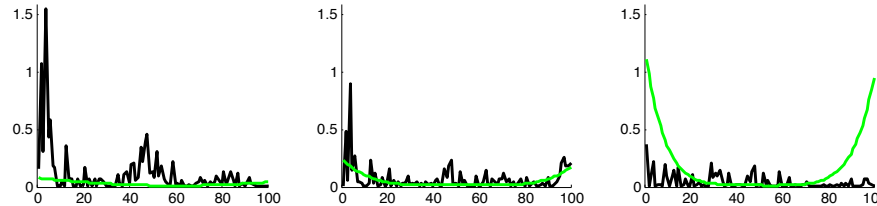
$h_s(x)$ Cubic



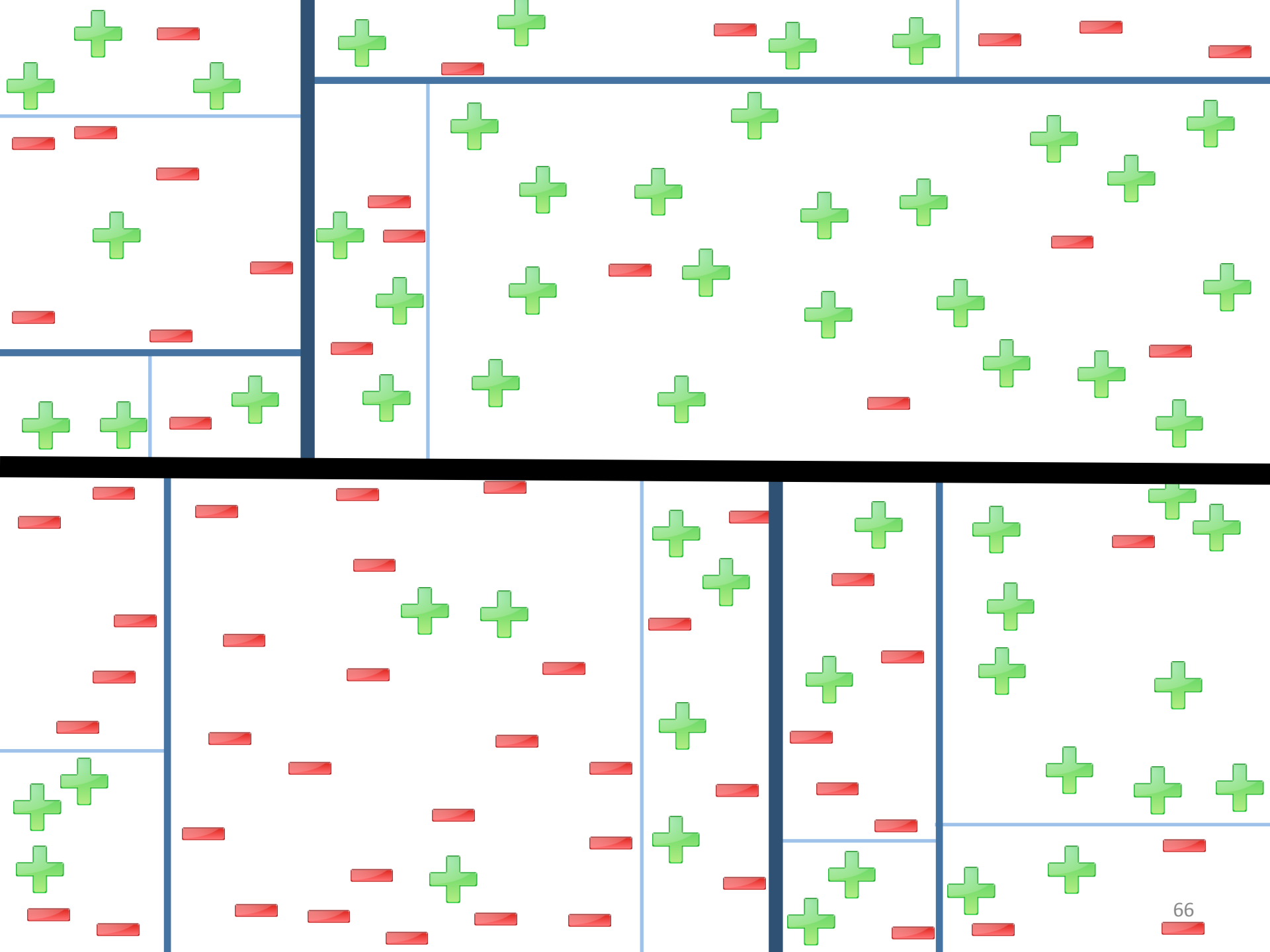
Bias-Variance Trade-off



Overfitting vs Underfitting



- High variance implies **overfitting**
 - Model class unstable
 - Variance increases with model complexity
 - Variance reduces with more training data.
- High bias implies **underfitting**
 - Even with no variance, model class has high error
 - Bias decreases with model complexity
 - Independent of training data size



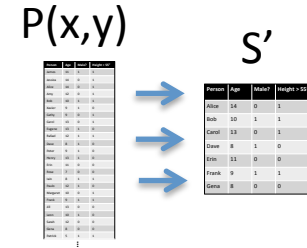


Decision Trees are Low Bias,
High Variance Models
Unless you Regularize a lot...
...but then often worse than Linear Models

Highly Non-Linear, Can Easily Overfit
Different Training Samples Can Lead to
Very Different Trees

Bagging

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



sampled independently

Variance reduces linearly
Bias unchanged

$$E_S[(h_S(x) - y)^2] = E_S[(Z - \check{z})^2] + \check{z}^2$$

⏟
↑
↑
 Expected Error Variance Bias
 On single (x,y)

$$Z = h_S(x) - y$$

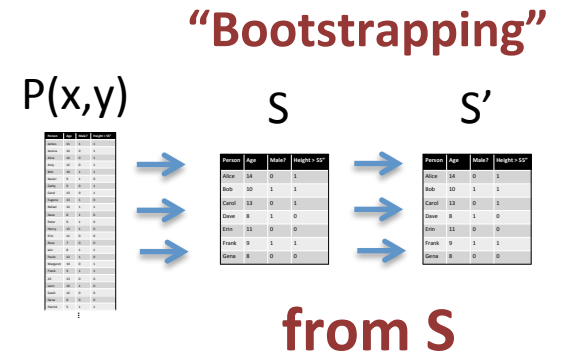
$$\check{z} = E_S[Z]$$

“Bagging Predictors” [Leo Breiman, 1994]

<http://statistics.berkeley.edu/sites/default/files/tech-reports/421.pdf>

Bagging

- **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

$$E_S[(h_S(x) - y)^2] = E_S[(Z - \check{z})^2] + \check{z}^2$$

Expected Error
On single (x,y)

Variance

Bias

$$Z = h_S(x) - y$$

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

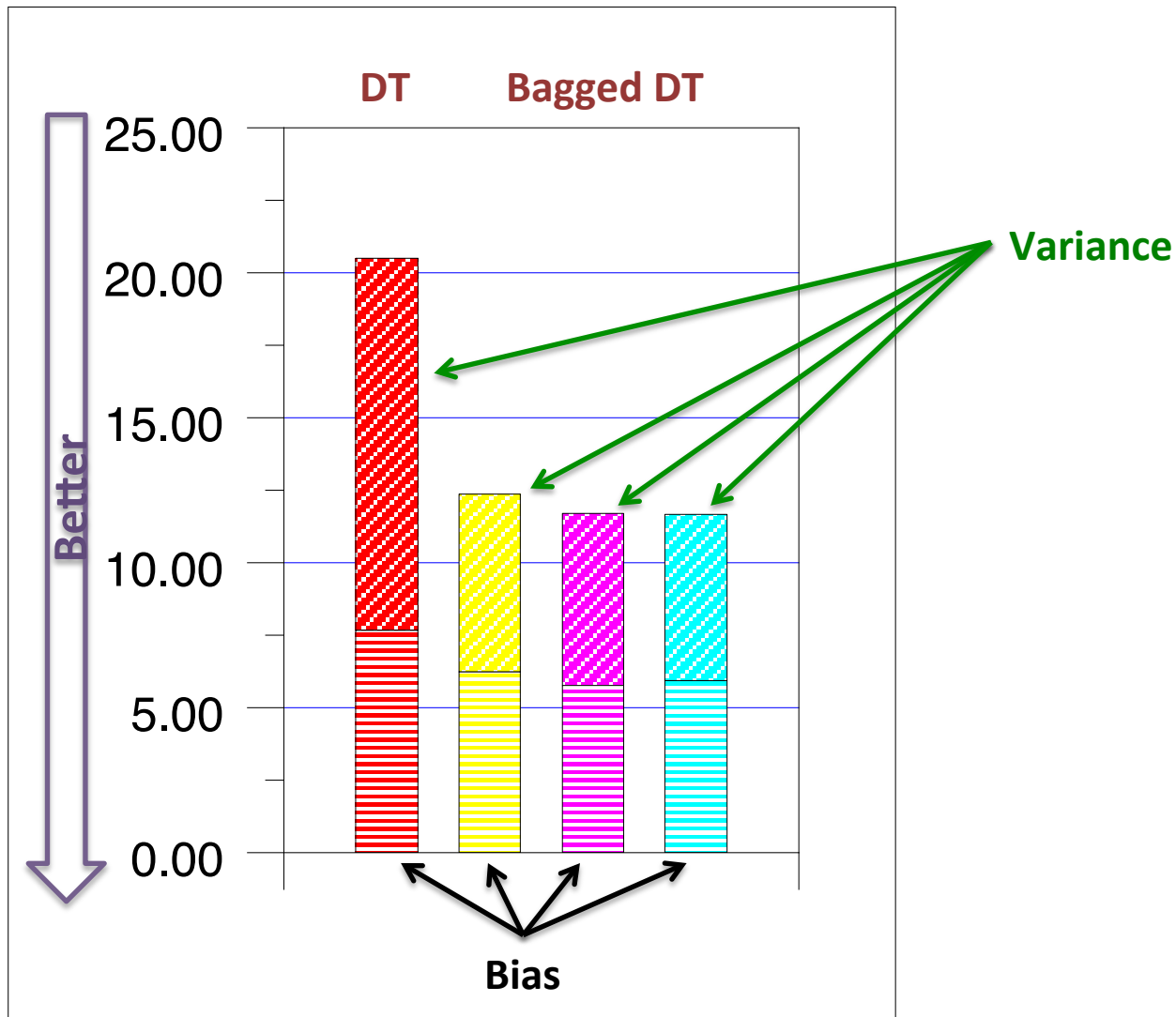
Bagging = Bootstrap Aggregation

“Bagging Predictors” [Leo Breiman, 1994]

<http://statistics.berkeley.edu/sites/default/files/tech-reports/421.pdf>

Recap: Bagging for DTs

- **Given:** Training Set S
- **Bagging:** Generate Many Bootstrap Samples S'
 - Sampled with replacement from S
 - $|S'| = |S|$
 - Train Minimally Regularized DT on S'
 - High Variance, Low Bias
- **Final Predictor:** Average of all DTs
 - Averaging reduces variance



“An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants”

Eric Bauer & Ron Kohavi, Machine Learning 36, 105–139 (1999)

<http://ai.stanford.edu/~ronnyk/vote.pdf>

Why Bagging Works

- Define Ideal Aggregation Predictor $h_A(x)$:

- Each S' drawn from true distribution P

$$h_A(x) = E_{S \sim P(x,y)} [h_S(x)]$$



Decision Tree Trained on S

- We will first compare the error of $h_A(x)$ vs $h_S(x)$
- Then show how to adapt comparison to Bagging

Analysis of Ideal Aggregate Predictor (Squared Loss)

$$h_A(x) = E_{S \sim P(x,y)} [h_S(x)]$$

Decision Tree Trained on S

$$\underbrace{E_S [L(y, h_S(x))]}_{\text{Expected Loss of } h_S \text{ on single } (x,y)} = E_S [(y - h_S(x))^2]$$

Linearity of Expectation

Expected Loss of h_S
on single (x,y)

$$= E_S [y^2] - 2E_S [yh_S(x)] + E_S [h_S(x)^2]$$

$$= y^2 - 2yE_S [h_S(x)] + E_S [h_S(x)^2]$$

$$E[Z^2] \geq E[Z]^2$$

($Z=h_S(x)$)

$$\geq y^2 - 2yE_S [h_S(x)] + E_S [h_S(x)]^2$$

$$= y^2 - 2yh_A(x) + h_A(x)^2$$

Definition of h_A

$$= (y - h_A(x))^2$$

$$= L(y, h_A(x))$$

Loss of h_A

“Bagging Predictors” [Leo Breiman, 1994]

<http://statistics.berkeley.edu/sites/default/files/tech-reports/421.pdf>

Key Insight

- Ideal Aggregate Predictor Improves if:

$$E_S [h_S(x)^2] > E_S [h_S(x)]^2 = h_A(x)^2$$

Large improvement if $h_S(x)$ is “unstable” (high variance)
 $h_A(x)$ is guaranteed to be at least as good as $h_S(x)$.

- Bagging Predictor Improves if:

$$E_S [h_S(x)^2] > E_S [E_{S' \sim S} [h_{S'}(x)]^2] = E_S [h_B(x)^2]$$

Improves if $h_B(x)$ is much more stable than $h_S(x)$
 $h_B(x)$ can sometimes be more unstable than $h_S(x)$
Bias of $h_B(x)$ can be worse than $h_S(x)$.

“Bagging Predictors” [Leo Breiman, 1994]

<http://statistics.berkeley.edu/sites/default/files/tech-reports/421.pdf>

Random Forests

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 – Random Features” [Leo Breiman, 1997]

<http://oz.berkeley.edu/~breiman/random-forests.pdf>

Top-Down Random Forest Training

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

Repeat: until stopping condition.

Step 1:

1

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 – Random Features” [Leo Breiman, 1997]

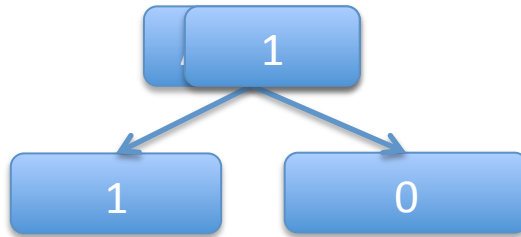
<http://oz.berkeley.edu/~breiman/random-forests.pdf>

Top-Down Random Forest Training

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 – Random Features” [Leo Breiman, 1997]

<http://oz.berkeley.edu/~breiman/random-forests.pdf>

Top-Down Random Forest Training

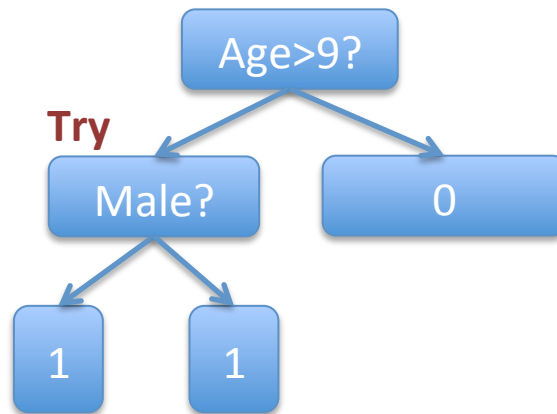
Loop: Sample T random splits at each Leaf.
Choose split with greatest impurity reduction.

Repeat: until stopping condition.

Step 1:

Step 2:

Step 3:



Randomly decide only look at gender.

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 – Random Features” [Leo Breiman, 1997]

<http://oz.berkeley.edu/~breiman/random-forests.pdf>

Top-Down Random Forest Training

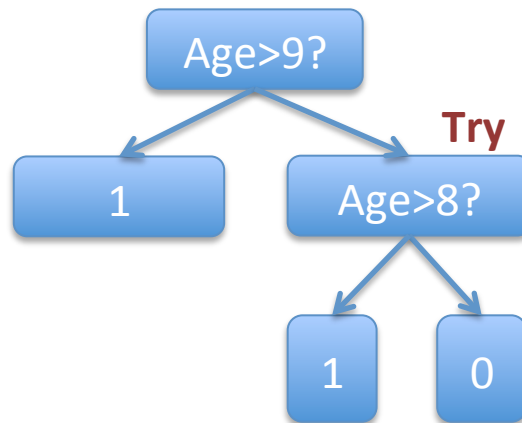
Loop: Sample T random splits at each Leaf.
Choose split with greatest impurity reduction.

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 – Random Features” [Leo Breiman, 1997]

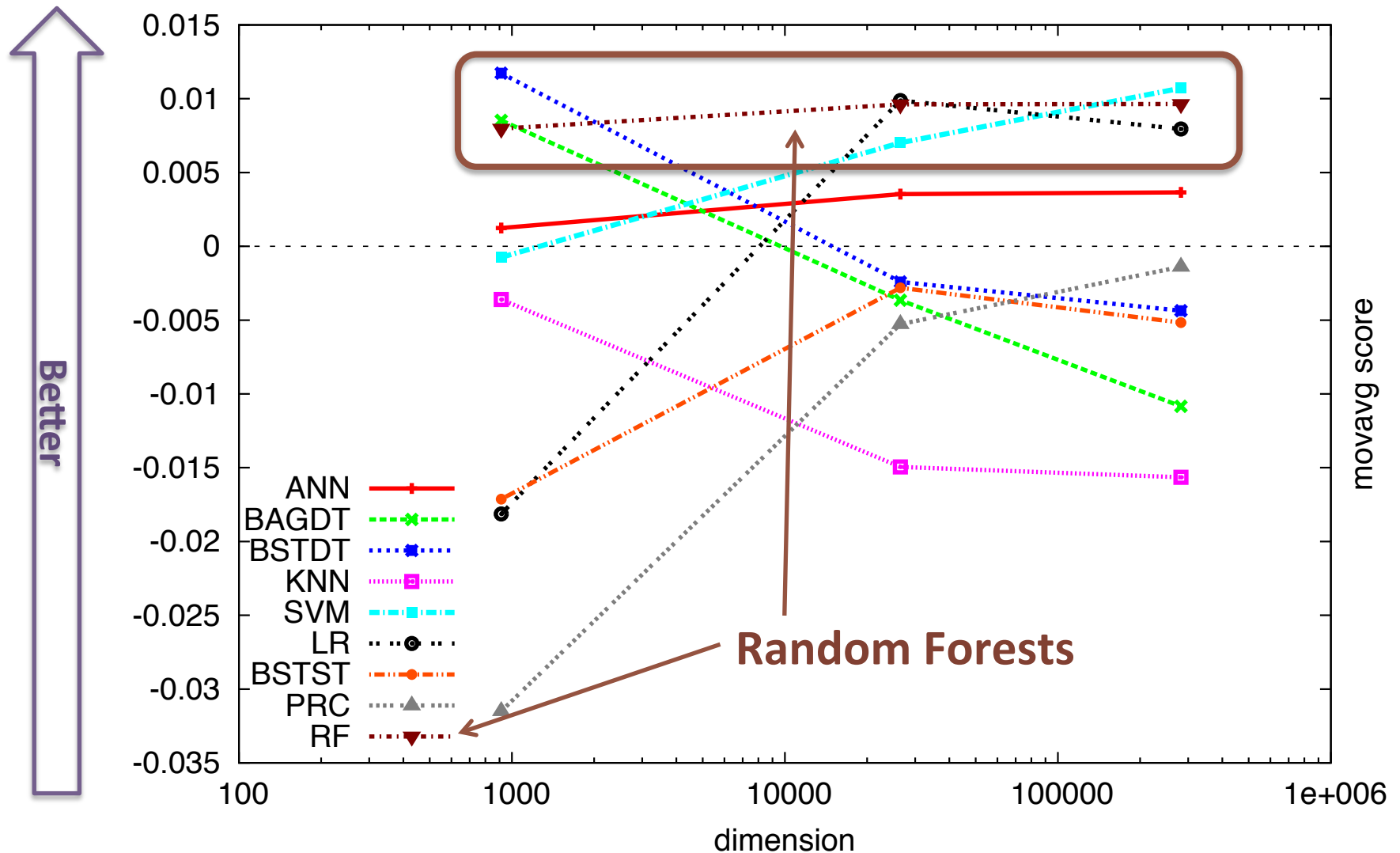
<http://oz.berkeley.edu/~breiman/random-forests.pdf>

Recap: Random Forests

- 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

“Random Forests – Random Features” [Leo Breiman, 1997]

<http://oz.berkeley.edu/~breiman/random-forests.pdf>



Average performance over many datasets
Random Forests perform the best

“An Empirical Evaluation of Supervised Learning in High Dimensions”
Caruana, Karampatziakis & Yessenalina, ICML 2008

Next Lecture

- **Boosting**
 - Method for reducing bias
- **Ensemble Selection**
 - Very general method for combining classifiers
 - Multiple-time winner of ML competitions
- **Recitation Tomorrow:**
 - Brief Tutorial of Kaggle
 - Brief Tutorial of Decision Tree Scikit-Learn Package
 - Although many decision tree packages available online.