Tree-based Methods

Sanjay Arora
AI Center of Excellence
Red Hat

Ulrich Drepper
AI Center of Excellence
Red Hat

Why Decision Trees

- One of the most widely used ML algorithms
- Easy and fast to train
- Minimal data preprocessing required
- Can handle both real-valued and categorical variables
- Variants (random forests and boosted decision trees) can be extremely performant

Problem Definition: Classification

$$\text{Given features} : x_1, x_2, \ldots, x_n$$

$$\text{Predict class or label, y} = 0 \text{ or } 1$$

Problem Definition: Classification

$$\text{Given features} : x_1, x_2, \ldots, x_n$$

$$\text{Predict class or label, y} = 0 \text{ or } 1$$

Will discuss regression later

Decision Tree

Series of if-else statements on features that distinguish between two classes

Decision Tree

Series of if-else statements on features
that distinguish between two classes

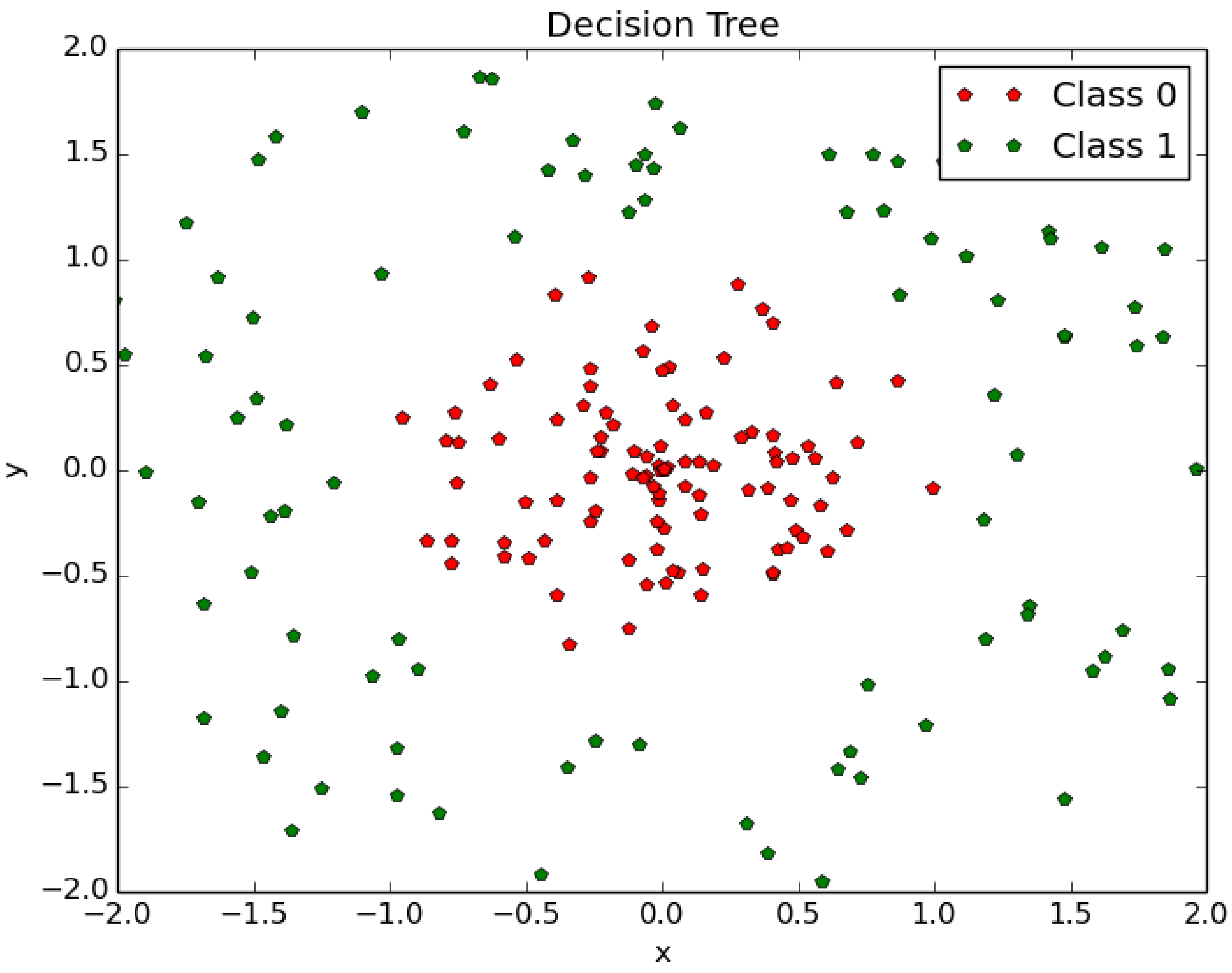
Some we code up all the time!

Decision Tree

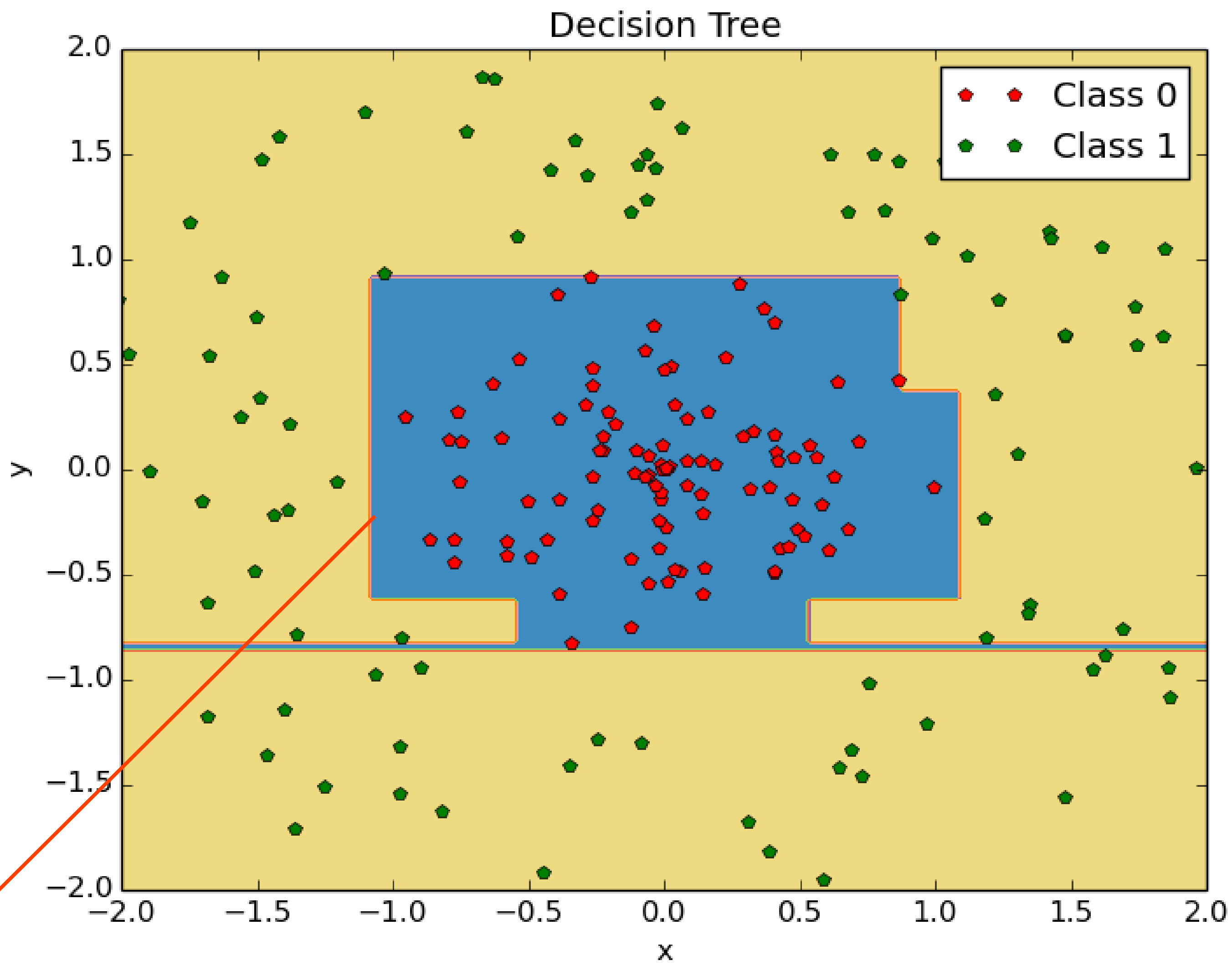
Series of if-else statements on features
that distinguish between two classes

Something we code up all the time!

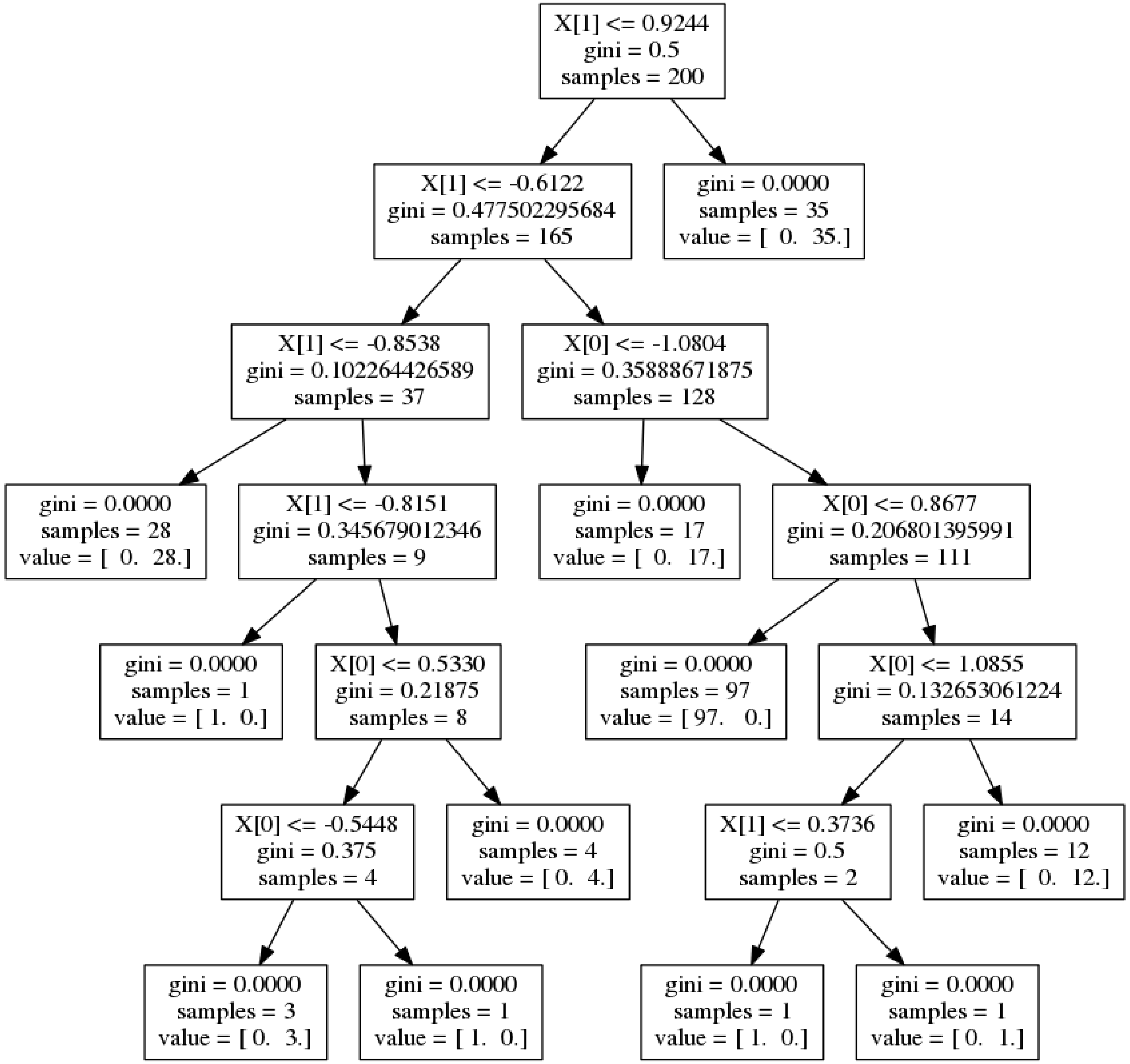
But now the expressions are learned!

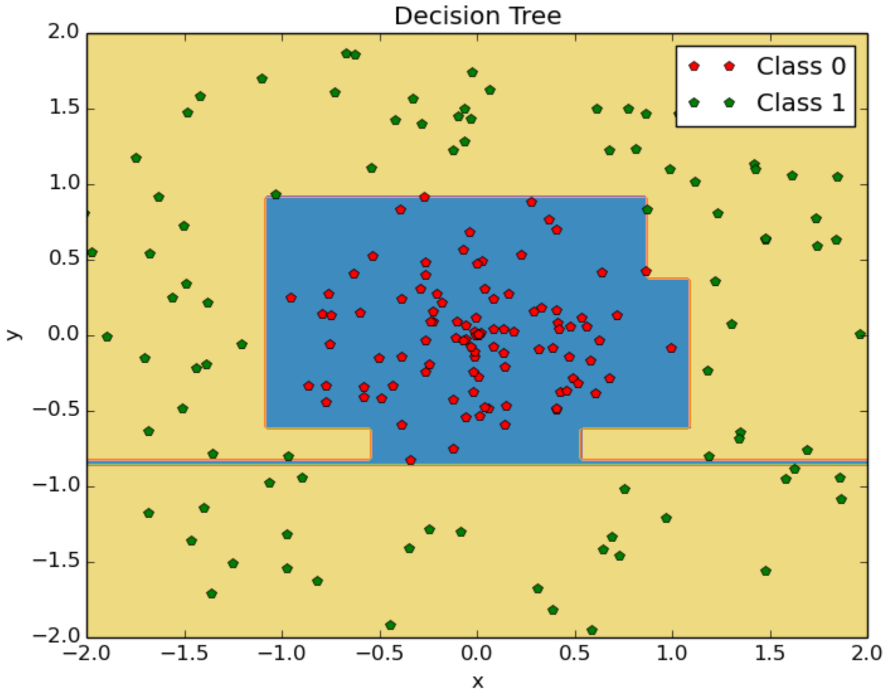


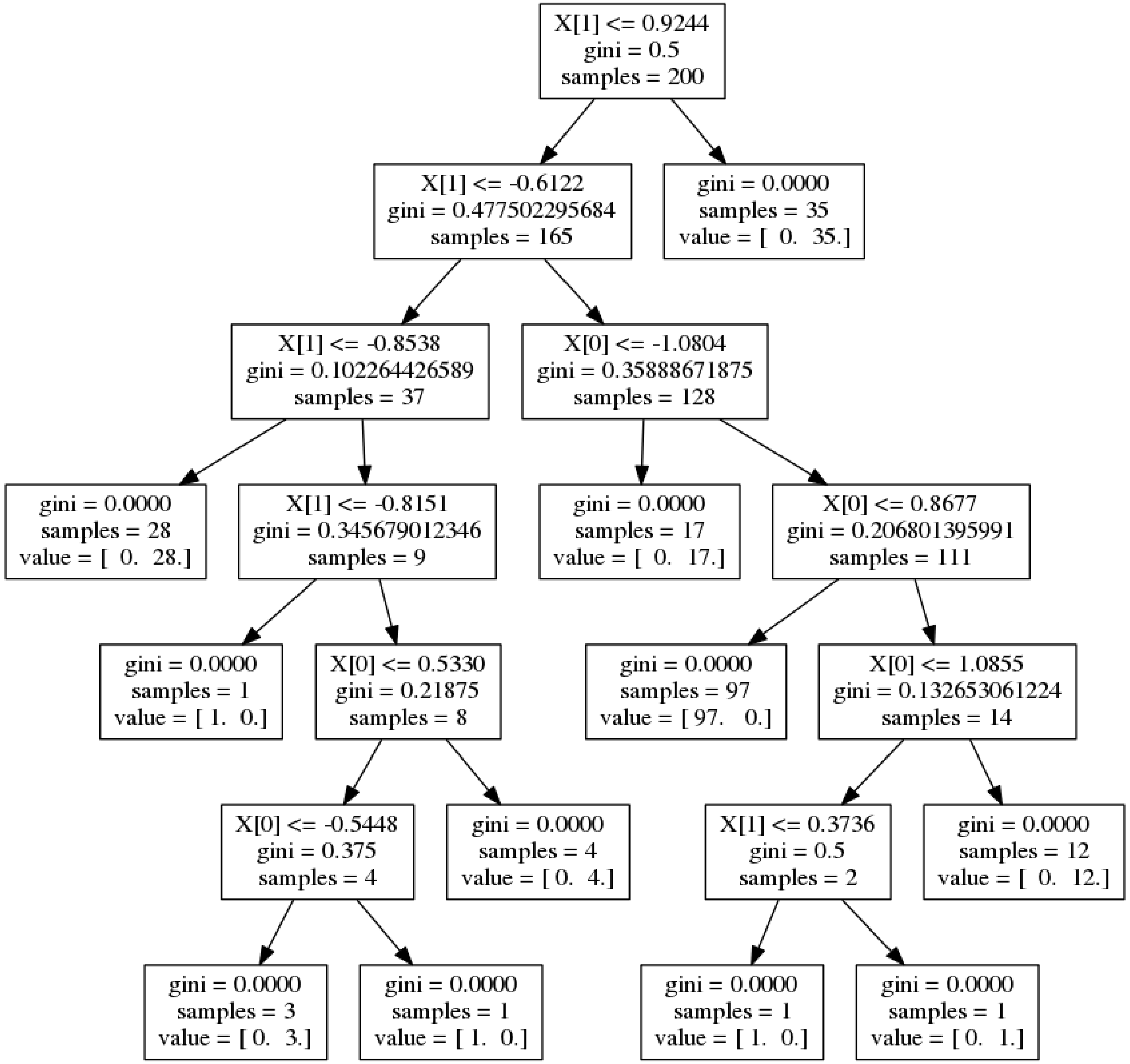
Is it possible to separate red and green dots with a sequence of if-else statements on x and y?



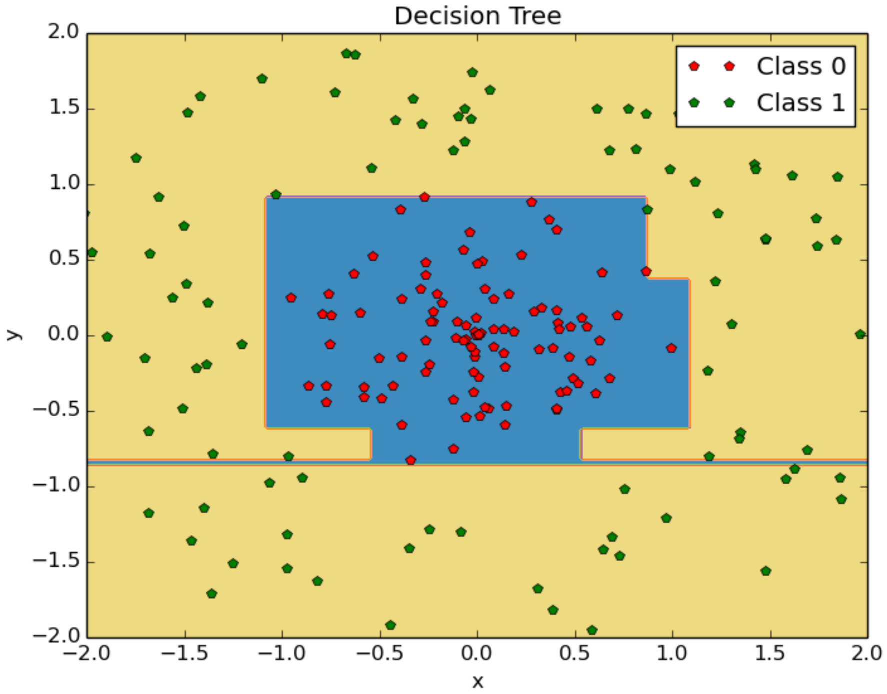
Any if-else statement on x or y ⟷ Horizontal or vertical line

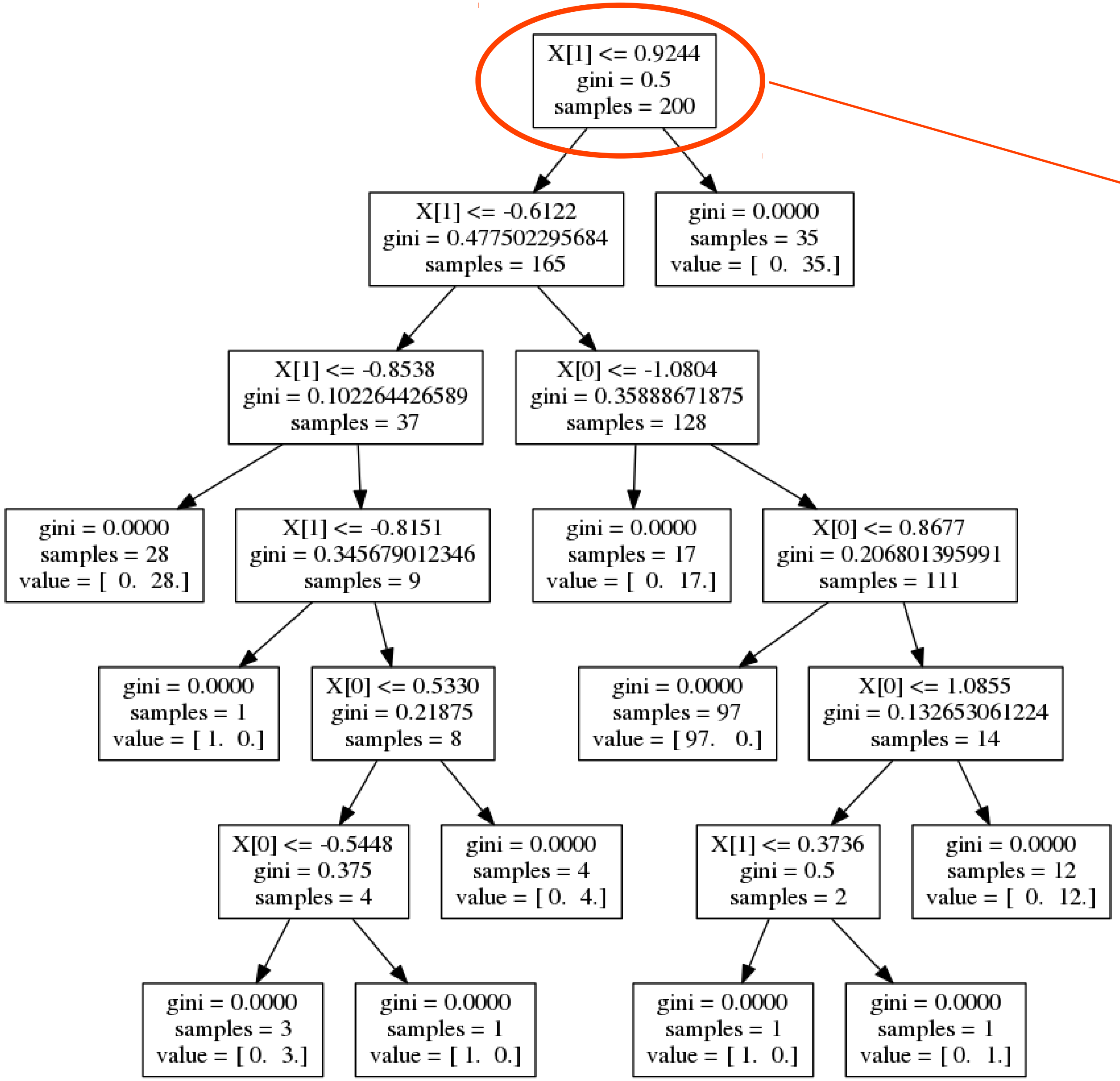

X[1] <= 0.9244
gini = 0.5
samples = 200
X[1] <= -0.6122
gini = 0.477502295684
samples = 165
gini = 0.0000
samples = 35
value = [0. 35.]
X[1] <= -0.8538
gini = 0.102264426589
samples = 37
X[0] <= -1.0804
gini = 0.35888671875
samples = 128
gini = 0.0000
samples = 28
value = [0. 28.]
X[1] <= -0.8151
gini = 0.345679012346
samples = 9
gini = 0.0000
samples = 17
value = [0. 17.]
X[0] <= 0.8677
gini = 0.206801395991
samples = 111
gini = 0.0000
samples = 1
value = [1. 0.]
X[0] <= 0.5330
gini = 0.21875
samples = 8
gini = 0.0000
samples = 97
value = [97. 0.]
X[0] <= 1.0855
gini = 0.132653061224
samples = 14
X[0] <= -0.5448
gini = 0.375
samples = 4
gini = 0.0000
samples = 4
value = [0. 4.]
X[1] <= 0.3736
gini = 0.5
samples = 2
gini = 0.0000
samples = 12
value = [0. 12.]
gini = 0.0000
samples = 3
value = [0. 3.]
gini = 0.0000
samples = 1
value = [1. 0.]
gini = 0.0000
samples = 1
value = [1. 0.]
gini = 0.0000
samples = 1
value = [0. 1.]


Decision Tree
Class 0
Class 1
y
x
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

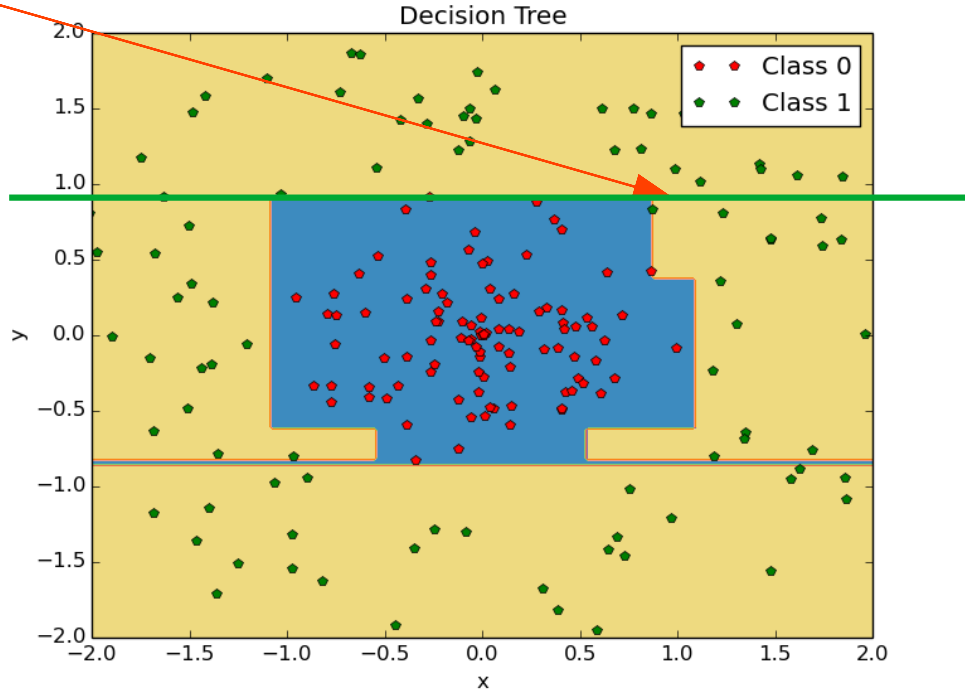

X[1] <= 0.9244
gini = 0.5
samples = 200
X[1] <= -0.6122
gini = 0.477502295684
samples = 165
gini = 0.0000
samples = 35
value = [0. 35.]
X[1] <= -0.8538
gini = 0.102264426589
samples = 37
X[0] <= -1.0804
gini = 0.35888671875
samples = 128
gini = 0.0000
samples = 28
value = [0. 28.]
X[1] <= -0.8151
gini = 0.345679012346
samples = 9
gini = 0.0000
samples = 17
value = [0. 17.]
X[0] <= 0.8677
gini = 0.206801395991
samples = 111
gini = 0.0000
samples = 1
value = [1. 0.]
X[0] <= 0.5330
gini = 0.21875
samples = 8
gini = 0.0000
samples = 97
value = [97. 0.]
X[0] <= 1.0855
gini = 0.132653061224
samples = 14
X[0] <= -0.5448
gini = 0.375
samples = 4
gini = 0.0000
samples = 4
value = [0. 4.]
X[1] <= 0.3736
gini = 0.5
samples = 2
gini = 0.0000
samples = 12
value = [0. 12.]
gini = 0.0000
samples = 3
value = [0. 3.]
gini = 0.0000
samples = 1
value = [1. 0.]
gini = 0.0000
samples = 1
value = [1. 0.]
gini = 0.0000
samples = 1
value = [0. 1.]

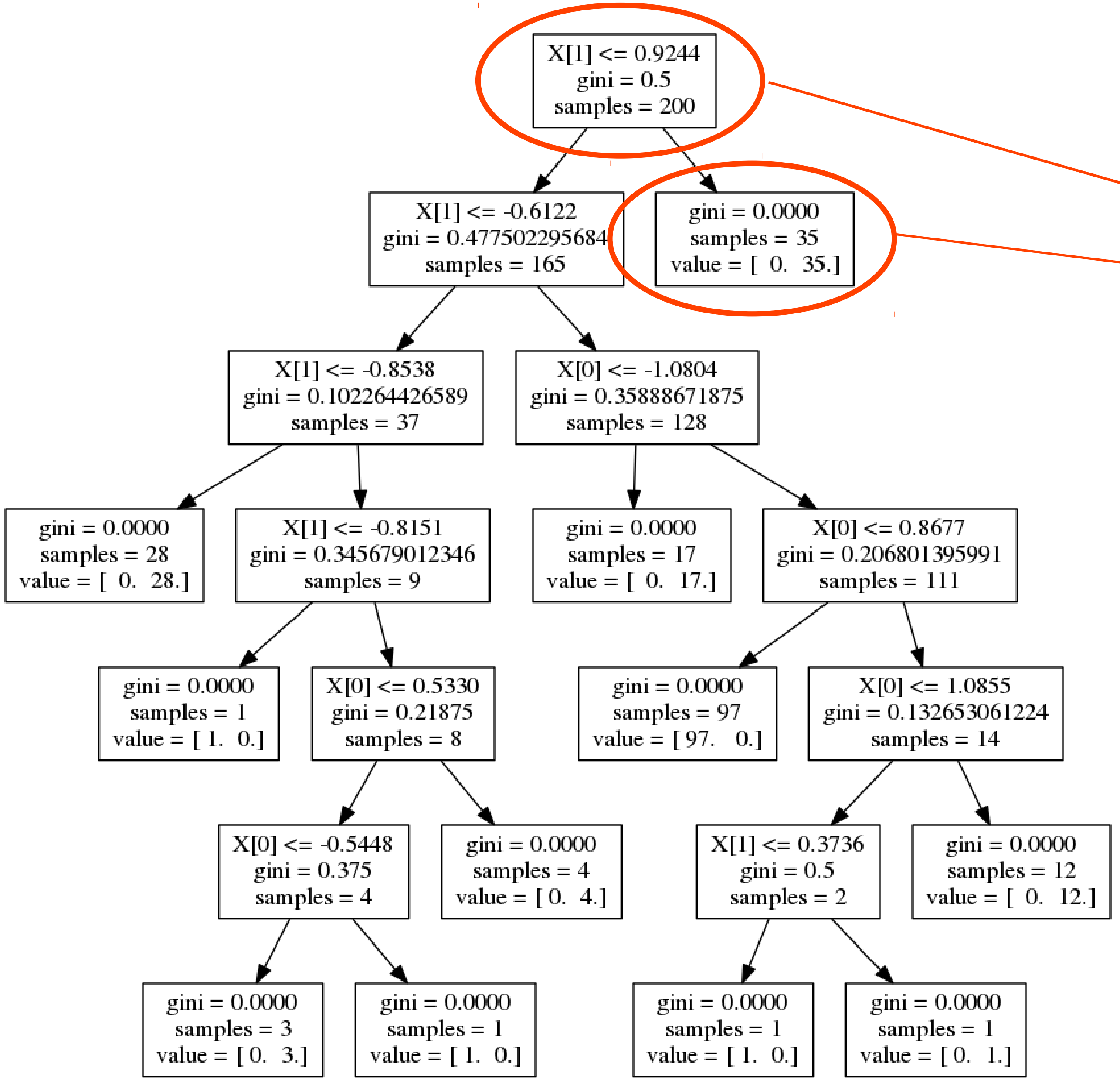
X[0] = x
X[1] = y


Decision Tree
Class 0
Class 1
y
x
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

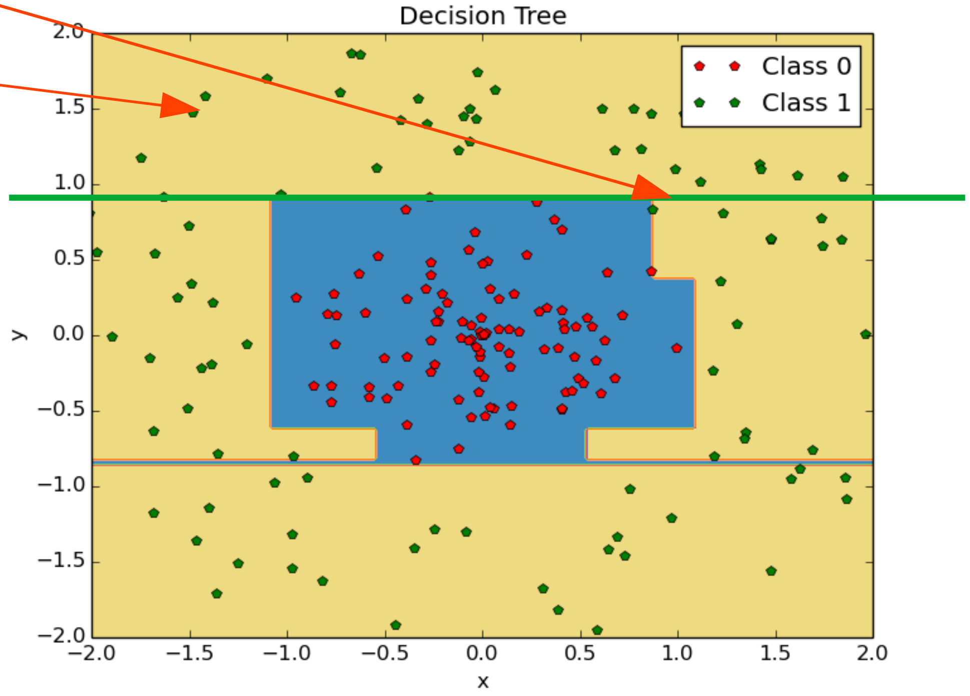

X[1] <= 0.9244
gini = 0.5
samples = 200
X[1] <= -0.6122
gini = 0.477502295684
samples = 165
gini = 0.0000
samples = 35
value = [0. 35.]
X[1] <= -0.8538
gini = 0.102264426589
samples = 37
X[0] <= -1.0804
gini = 0.35888671875
samples = 128
gini = 0.0000
samples = 28
value = [0. 28.]
X[1] <= -0.8151
gini = 0.345679012346
samples = 9
gini = 0.0000
samples = 17
value = [0. 17.]
X[0] <= 0.8677
gini = 0.206801395991
samples = 111
gini = 0.0000
samples = 1
value = [1. 0.]
X[0] <= 0.5330
gini = 0.21875
samples = 8
gini = 0.0000
samples = 97
value = [97. 0.]
X[0] <= 1.0855
gini = 0.132653061224
samples = 14
X[0] <= -0.5448
gini = 0.375
samples = 4
gini = 0.0000
samples = 4
value = [0. 4.]
X[1] <= 0.3736
gini = 0.5
samples = 2
gini = 0.0000
samples = 12
value = [0. 12.]
gini = 0.0000
samples = 3
value = [0. 3.]
gini = 0.0000
samples = 1
value = [1. 0.]
gini = 0.0000
samples = 1
value = [1. 0.]
gini = 0.0000
samples = 1
value = [0. 1.]

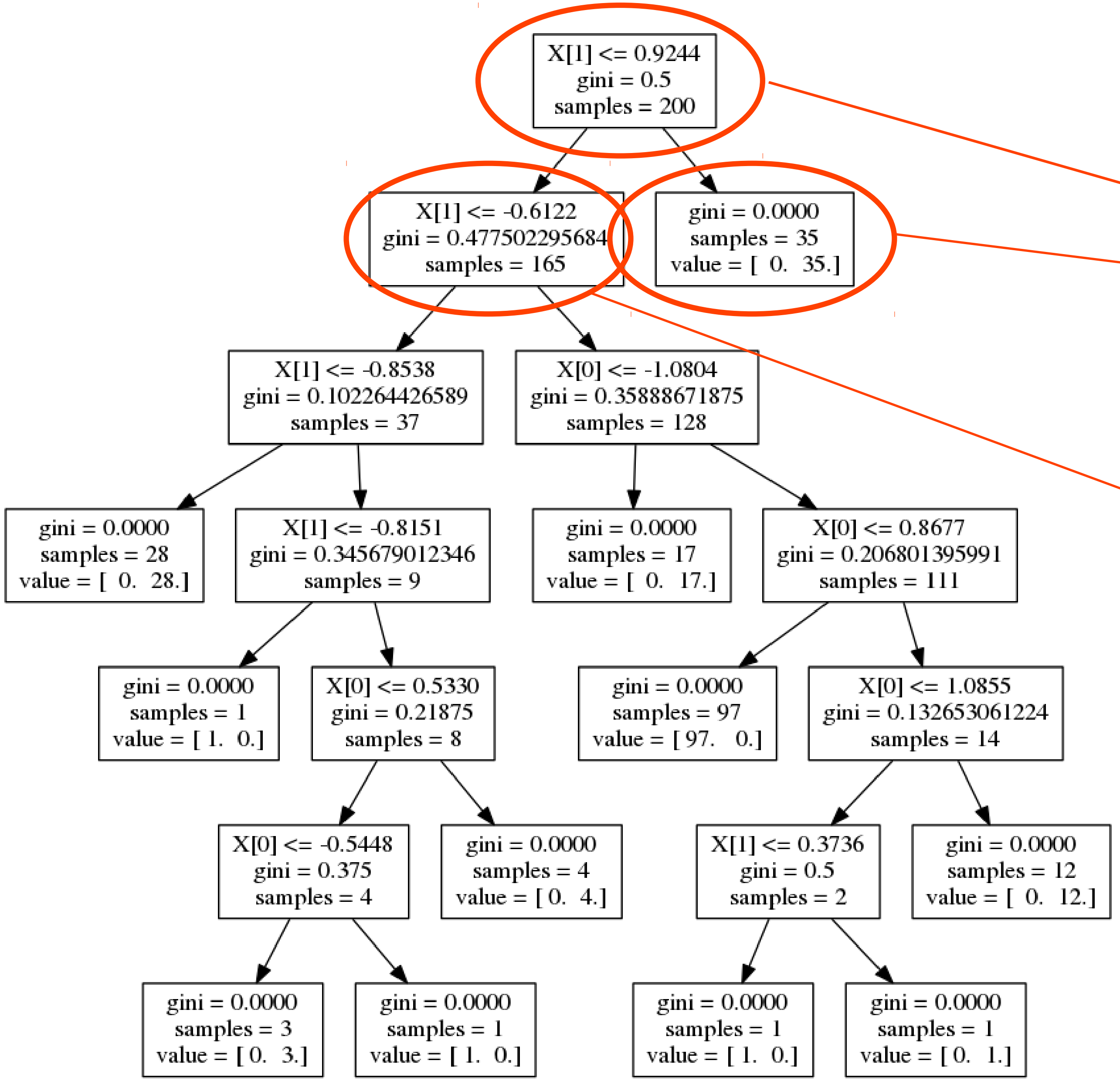
X[0] = x
X[1] = y


Decision Tree
Class 0
Class 1
y
x
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

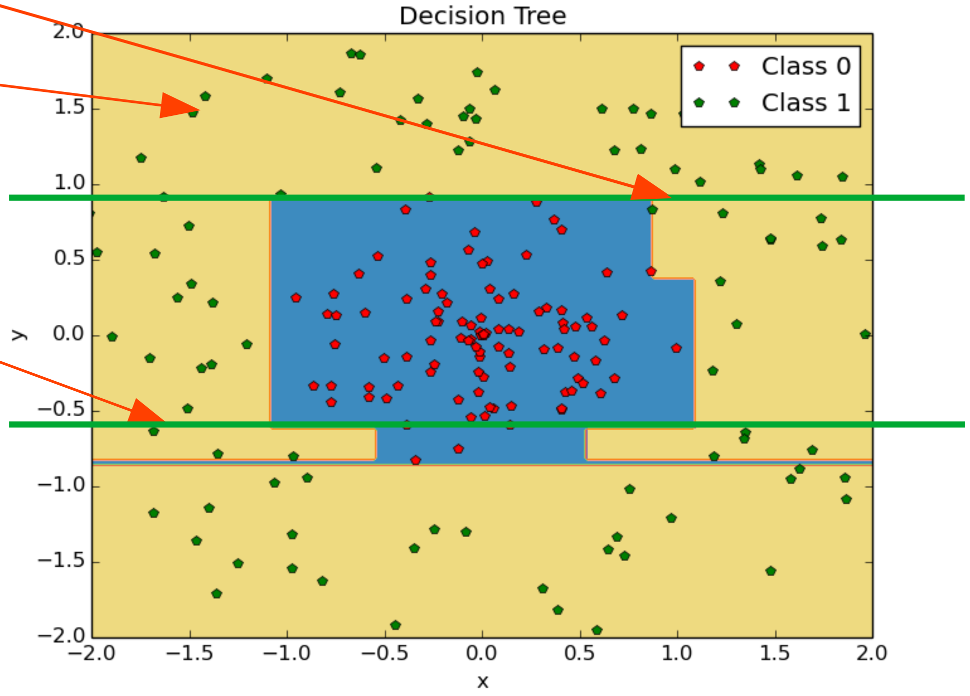

X[1] <= 0.9244
gini = 0.5
samples = 200
X[1] <= -0.6122
gini = 0.477502295684
samples = 165
gini = 0.0000
samples = 35
value = [0. 35.]
X[1] <= -0.8538
gini = 0.102264426589
samples = 37
X[0] <= -1.0804
gini = 0.35888671875
samples = 128
gini = 0.0000
samples = 28
value = [0. 28.]
X[1] <= -0.8151
gini = 0.345679012346
samples = 9
gini = 0.0000
samples = 17
value = [0. 17.]
X[0] <= 0.8677
gini = 0.206801395991
samples = 111
gini = 0.0000
samples = 1
value = [1. 0.]
X[0] <= 0.5330
gini = 0.21875
samples = 8
gini = 0.0000
samples = 97
value = [97. 0.]
X[0] <= 1.0855
gini = 0.132653061224
samples = 14
X[0] <= -0.5448
gini = 0.375
samples = 4
gini = 0.0000
samples = 4
value = [0. 4.]
X[1] <= 0.3736
gini = 0.5
samples = 2
gini = 0.0000
samples = 12
value = [0. 12.]
gini = 0.0000
samples = 3
value = [0. 3.]
gini = 0.0000
samples = 1
value = [1. 0.]
gini = 0.0000
samples = 1
value = [1. 0.]
gini = 0.0000
samples = 1
value = [0. 1.]

X[0] = x
X[1] = y

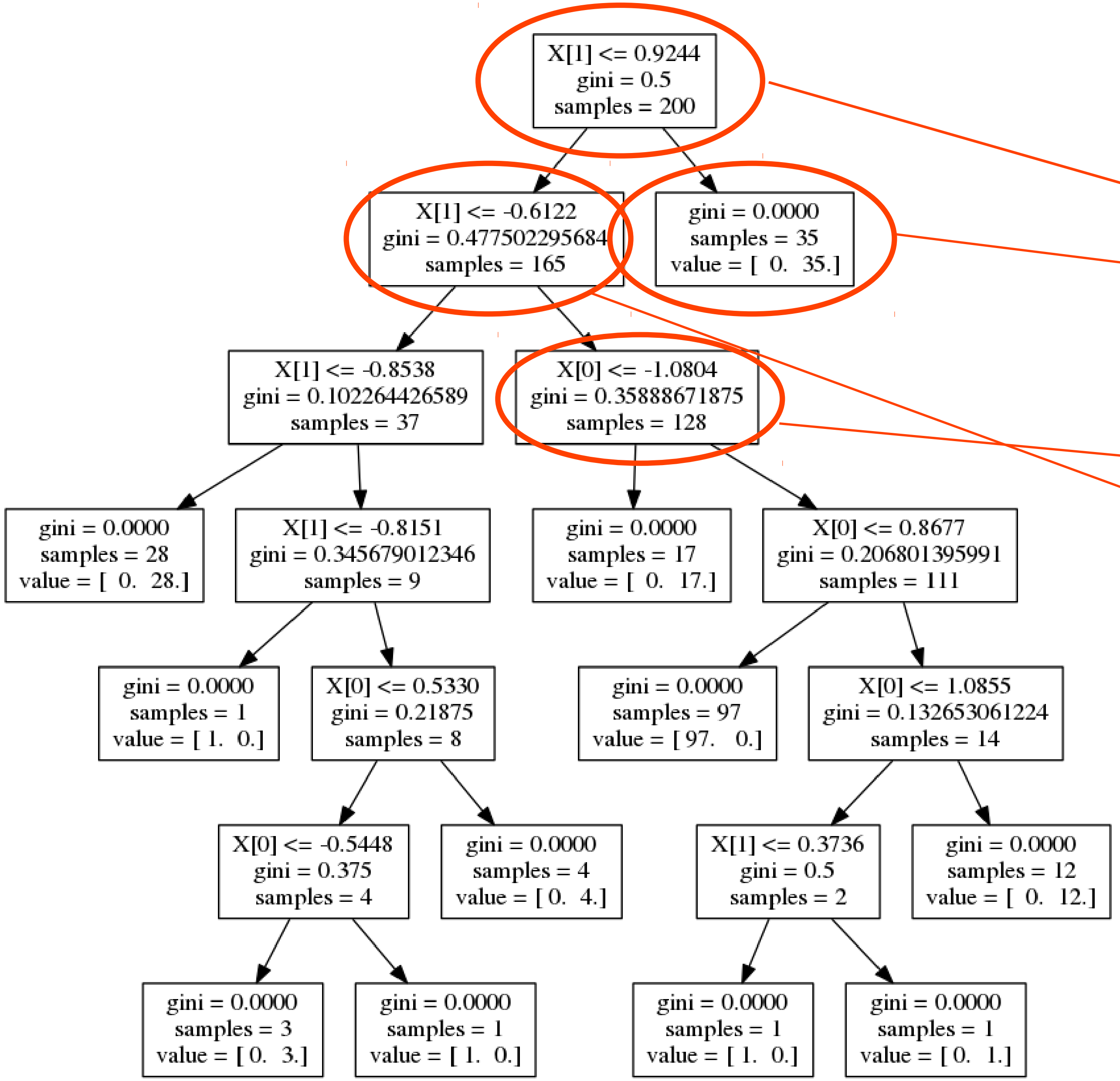

Decision Tree
Class 0
Class 1
y
x
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

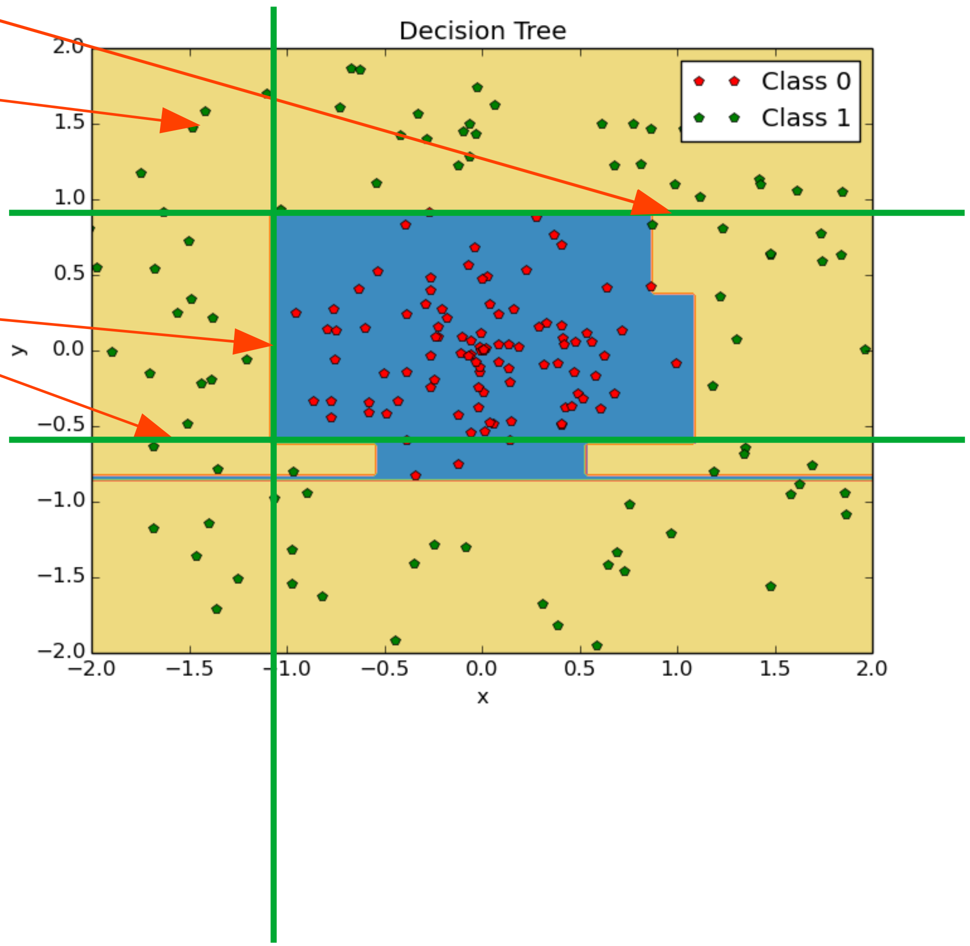

X[1] <= 0.9244
gini = 0.5
samples = 200
X[1] <= -0.6122
gini = 0.477502295684
samples = 165
gini = 0.0000
samples = 35
value = [0. 35.]
X[1] <= -0.8538
gini = 0.102264426589
samples = 37
X[0] <= -1.0804
gini = 0.35888671875
samples = 128
gini = 0.0000
samples = 28
value = [0. 28.]
X[1] <= -0.8151
gini = 0.345679012346
samples = 9
gini = 0.0000
samples = 17
value = [0. 17.]
X[0] <= 0.8677
gini = 0.206801395991
samples = 111
gini = 0.0000
samples = 1
value = [1. 0.]
X[0] <= 0.5330
gini = 0.21875
samples = 8
gini = 0.0000
samples = 97
value = [97. 0.]
X[0] <= 1.0855
gini = 0.132653061224
samples = 14
X[0] <= -0.5448
gini = 0.375
samples = 4
gini = 0.0000
samples = 4
value = [0. 4.]
X[1] <= 0.3736
gini = 0.5
samples = 2
gini = 0.0000
samples = 12
value = [0. 12.]
gini = 0.0000
samples = 3
value = [0. 3.]
gini = 0.0000
samples = 1
value = [1. 0.]
gini = 0.0000
samples = 1
value = [1. 0.]
gini = 0.0000
samples = 1
value = [0. 1.]

X[0] = x
X[1] = y


Decision Tree
Class 0
Class 1
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
y
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x

X[0] = x
X[1] = y





Red Hat

Training

Starting Point:

$$\text{Given n features: } x_1, \ldots, x_n$$

$$\text{Given labels: } y$$

Question:

$$\text{What feature } x_i \text{ should be choose to make a split on?}$$

$$\text{For given feature, what value should we make the split on?}$$

Training

Starting Point:

Given n features: $x_1, \ldots, x_n$

Given labels: y

Question:

What feature x_i should be choose to make a split on?

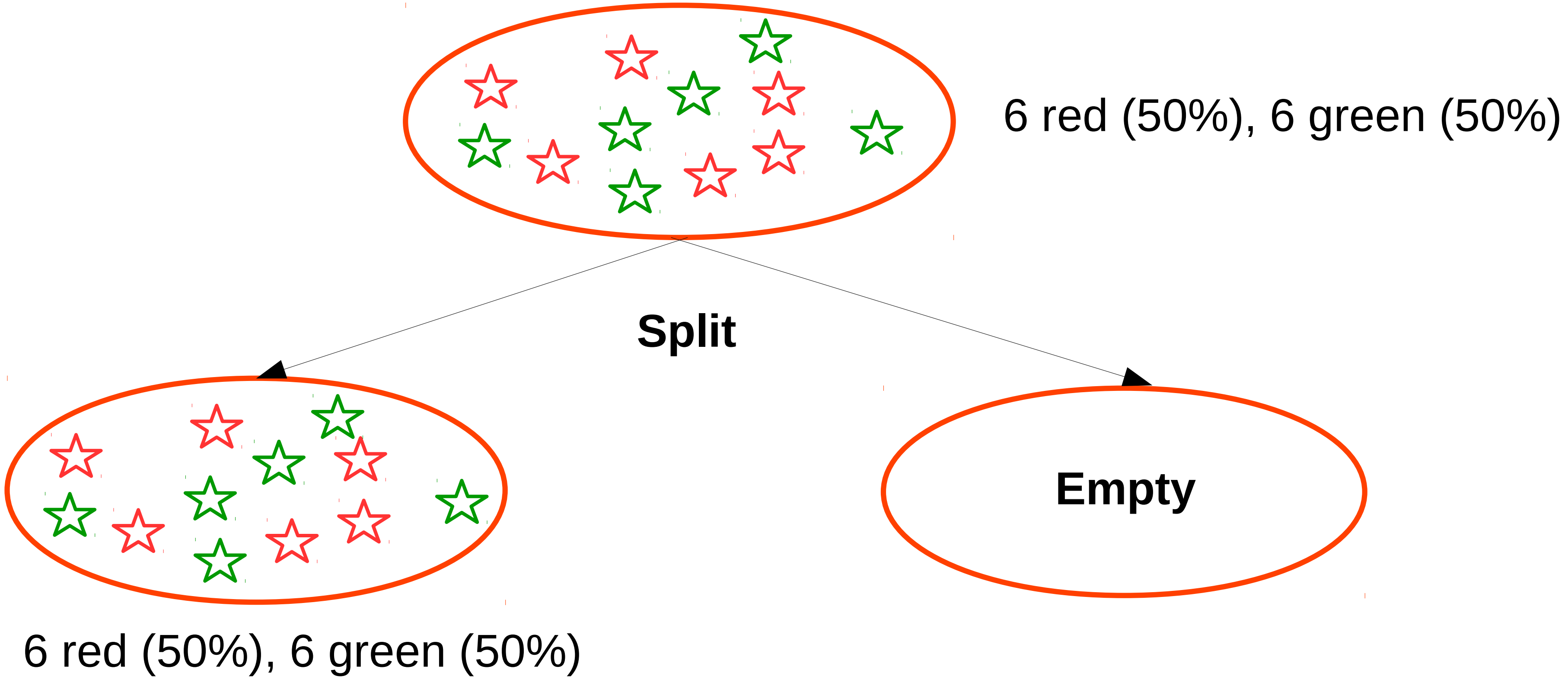
For given feature, what value should we make the split on?

How about?:

Loop over features and all possible values to split on

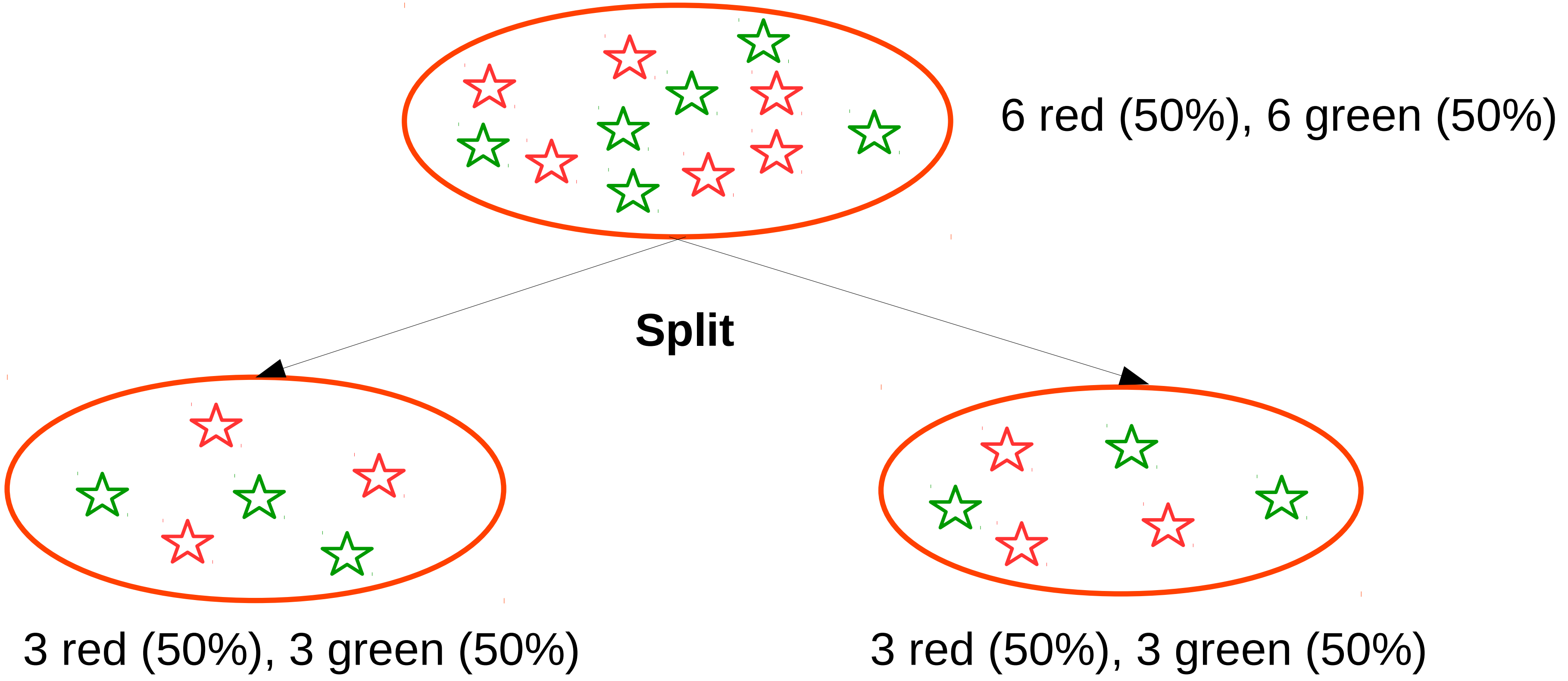
Pick one that maximizes "purity" of categories after the split

Training



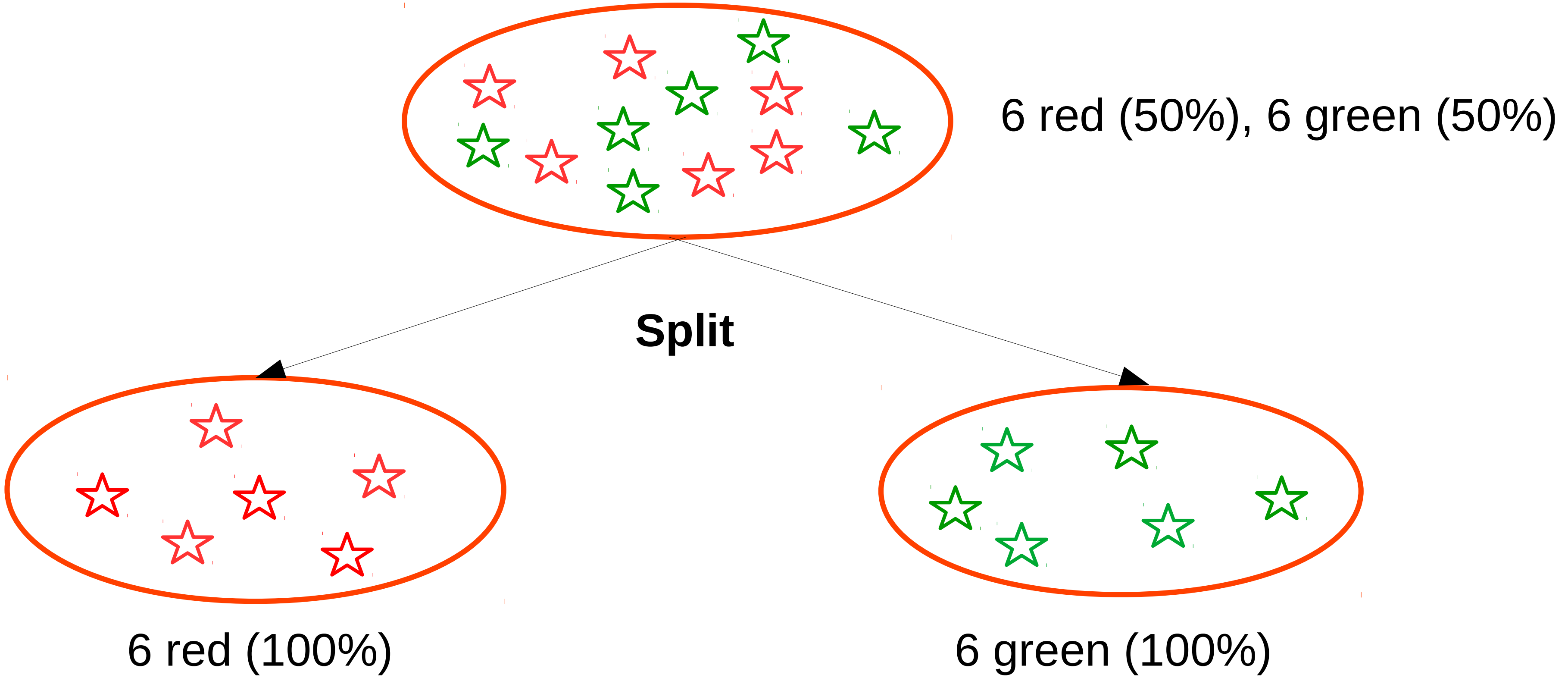
Not useful! Still stuck with original data

Training



No information gained. Still 50-50% split

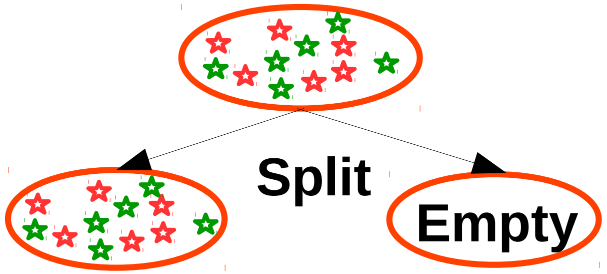
Training



Is this any good? Yes!! Learned a rule to separate green and red

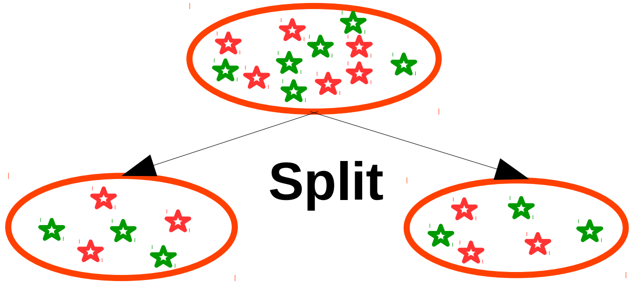
Red Hat

Training



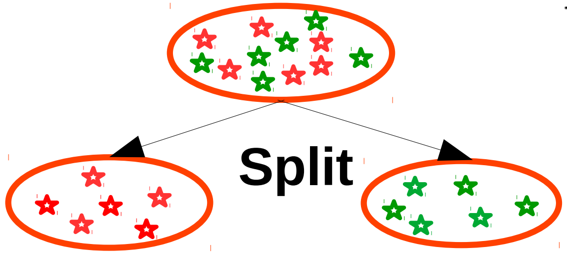
$p_{red} = 0.5$ $\quad X$

$p_{green} = 0.5$



$p_{red} = 0.5$ $\quad p_{red} = 0.5$

$p_{green} = 0.5$ $\quad p_{green} = 0.5$



$p_{red} = 1$ $\quad p_{red} = 0$

$p_{green} = 0$ $\quad p_{green} = 1$

Split: Some function of p_{red} and p_{green} to maximize

An Aside on Entropy

Toss a coin N times

Result: Get N_1 heads and N_2 tails

How do we describe the **state** of the system?

An Aside on Entropy

Toss a coin N times

Result: Get N_1 heads and N_2 tails

How do we describe the **state** of the system?

Macrostate: N_1 heads and N_2 tails

An Aside on Entropy

Toss a coin N times

Result: Get N_1 heads and N_2 tails

How do we describe the **state** of the system?

Macrostate: N_1 heads and N_2 tails

Microstate: Exact sequence of heads and tails

H T H H . . .

An Aside on Entropy

Toss a coin N times

Result: Get N_1 heads and N_2 tails

How do we describe the **state** of the system?

Relevant details

Macrostate: N_1 heads and N_2 tails

Microstate: Exact sequence of heads and tails

H T H H . . .

“Microscopic” description: too much detail

Red Hat

An Aside on Entropy

Define **Multiplicity**: Ω

Number of microstates corresponding to macrostate

How many detailed states correspond to one less detailed state

Coin Toss Experiment: N_1 heads and N_2 tails

$$\Omega = \binom{N_1 + N_2}{N_1} = \frac{(N_1 + N_2)!}{N_1!N_2!}$$

An Aside on Entropy

$$\Omega = \binom{N_1 + N_2}{N_1} = \frac{(N_1 + N_2)!}{N_1!N_2!}$$

Stirling's approximation: $\boxed{N! \approx N^N e^{-N}}$ for large N

$$\Omega \approx \frac{(N_1 + N_2)^{N_1+N_2} e^{-N_1-N_2}}{N_1^{N_1} e^{-N1} N_2^{N_2} e^{-N_2}}$$

$$\Omega \approx \frac{(N_1 + N_2)^{N_1}}{N_1^{N_1}} \frac{(N_1 + N_2)^{N_2}}{N_2^{N_2}}$$

An Aside on Entropy

$$\Omega \approx \frac{(N_1 + N_2)^{N_1}}{N_1^{N_1}} \frac{(N_1 + N_2)^{N_2}}{N_2^{N_2}}$$

Define: $\boxed{p_1 = \frac{N_1}{N_1 + N_2} \qquad p_2 = \frac{N_2}{N_1 + N_2}}$

$$\Omega = p_1^{-N_1} p_2^{-N_2} = p_1^{-p_1 N} p_2^{-p_2 N} = \boxed{(p_1^{-p_1} p_2^{-p_2})^N}$$

Red Hat

An Aside on Entropy

$$\Omega = (p_1^{-p_1} p_2^{-p_2})^N$$

Define **Entropy**:

$$\begin{aligned} S &= \log \Omega = N \log (p_1^{-p_1} p_2^{-p_2}) \\ &= -N(p_1 \log p_1 + p_2 \log p_2) \end{aligned}$$

Entropy per coin

$$\boxed{S = -(p_1 \log p_1 + p_2 \log p_2)} \qquad p_1 + p_2 = 1$$

An Aside on Entropy

Generalize to any **discrete** distribution:

Random variables takes N values - $a_1, \ldots, a_N$
with probabilities - $p_1, \ldots, p_N$

$$p_1 + \ldots + p_N = 1$$

$$\boxed{S \equiv -\Sigma_{i=1}^{N} p_i \log p_i} \quad p_1 + \ldots + p_N = 1$$

or **continuous** distribution:

$$\boxed{S \equiv -\int p(x) \log p(x) dx} \quad \int p(x) dx = 1$$

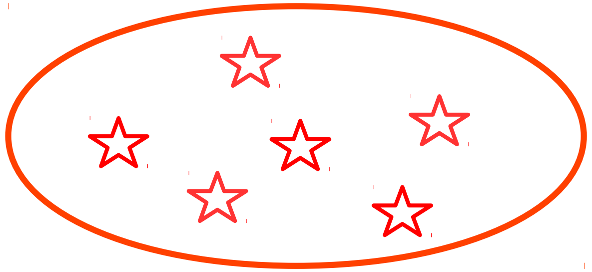
An Aside on Entropy

$$S = -(p_1 \log p_1 + p_2 \log p_2)$$

$$p_1 = \frac{6}{12} \qquad p_2 = \frac{6}{12}$$

$$S = -\frac{1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2} = \log 2 \boxed{\neq 0}$$

An Aside on Entropy



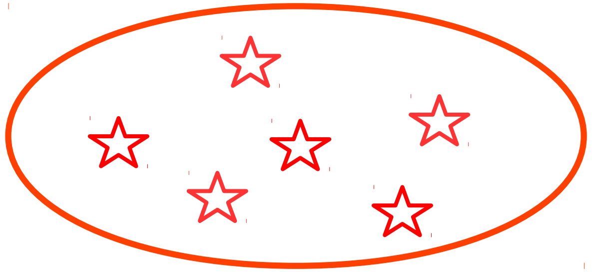
$$S = -(p_1 \log p_1 + p_2 \log p_2)$$

$$p_1 = \frac{6}{6} \qquad p_2 = \frac{0}{6}$$

$$S = -1 \log 1 - 0 \log 0 = \boxed{0}$$

Pure sample

An Aside on Entropy



$$S = -(p_1 \log p_1 + p_2 \log p_2)$$

$$p_1 = \frac{6}{6} \qquad p_2 = \frac{0}{6}$$

$$S = -1 \log 1 - 0 \log 0 = \boxed{0}$$

Pure sample

$$\text{Technical Point: } \lim_{x \to 0} x \log x \to 0$$

Training

Starting Point:

Given n features: $x_1, \ldots, x_n$

Given labels: y

Question:

What feature x_i should be choose to make a split on?

For given feature, what value should we make the split on?

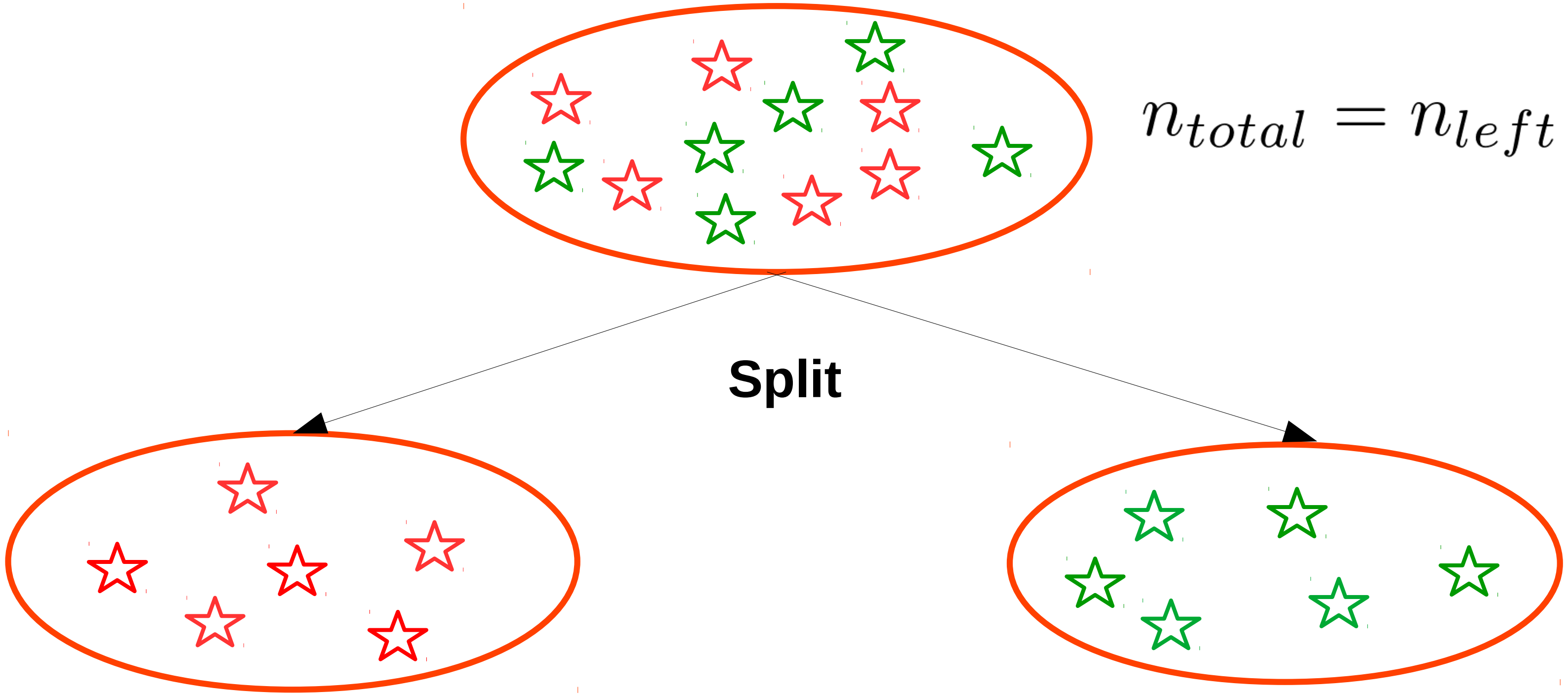
How about?:

Loop over features and all possible values to split on

$\boxed{\text{Pick one that maximizes "purity" of categories after the split}}$

Training

One possible choice for "purity": Entropy



$$n_{total} = n_{left} + n_{right}$$

n_{left} n_{right}

Smaller entropy = purer

Minimize: $$\boxed{\frac{n_{left}}{n_{total}} S_{left} + \frac{n_{right}}{n_{total}} S_{right}}$$

Red Hat

Training

Minimize: $$\boxed{\frac{n_{left}}{n_{total}} S_{left} + \frac{n_{right}}{n_{total}} S_{right}}$$

Repeat until:

End up with pure samples in each node

OR

Reach some maximum depth/number of levels of tree

OR

Reduction in cost is small enough

OR

Have too few samples in node

Training

Alternatively, minimize:

$$\boxed{\frac{n_{left}}{n_{total}} G_{left} + \frac{n_{right}}{n_{total}} G_{right}}$$

Gini $G_{node} = p_1(1 - p_1) + p_2(1 - p_2)$

$\rightarrow 0$ for pure nodes

$\rightarrow \frac{1}{2}$ for even split

$$p_1 = \frac{1}{2} \quad p_2 = \frac{1}{2}$$

Repeat until:

End up with pure samples in each node

OR

Reach some maximum depth/number of levels of tree etc.

Red Hat

Training

Pick rows

Loop over features:

  Loop over distinct values/cuts :

    Compute entropy/gini/variance if split on this feature and cut

Pick feature and cut minimizing metric

Repeat recursively for each daughter node till stopping condition

Decision Trees: Advantages

Minimal pre-processing required - no normalization for example

No assumptions about distribution of data

Simple to interpret - if not too deep

Can discover interactions between features automatically

Run fast

Decision Trees: Disadvantages

Use "horizontal" and "vertical" lines to learn decision boundaries
Result in over-complex trees

Can easily overfit data

Unstable: small changes in data can lead to very different trees

Cannot extrapolate

Easy to Use

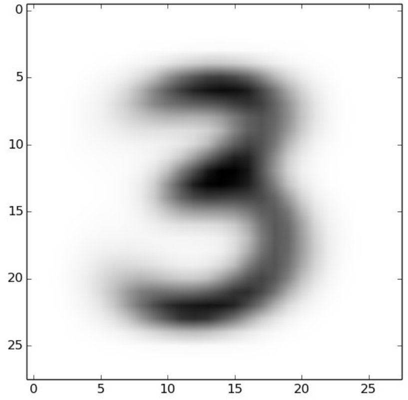
```
#Decision Tree Classifier
from sklearn import tree

model = tree.DecisionTreeClassifier()
model.fit(features, labels) #train the model

model.predict(test_features) #make predictions
#See documentation for other functions
```


Red Hat

Example



Recall digits dataset

Let's use decision trees to identify the digit 3 (label $= 1$) from other digits (label $= 0$)

Example

C_i	$pred = 0$	$pred = 1$
$label = 0$	$0 \to 0$	$0 \to 1$
$label = 1$	$1 \to 0$	$1 \to 1$

Recall confusion matrix

	pred = non-3	pred = 3
label = non-3	11052	251
label = 3	241	1056

$$\text{Accuracy} = \frac{(0 \to 0) + (1 \to 1)}{(0 \to 0) + (1 \to 1) + (0 \to 1) + (1 \to 0)} = 96.1\%$$

Ideally = 100%

Uniform Model (everything 0) = 89.7%

Example

C_i	$pred = 0$	$pred = 1$
$label = 0$	$0 \to 0$	$0 \to 1$
$label = 1$	$1 \to 0$	$1 \to 1$

Recall confusion matrix

	pred = non-3	pred = 3
label = non-3	11052	251
label = 3	241	1056

$$\text{Sensitivity/Recall} = \frac{(1 \to 1)}{(1 \to 1) + (1 \to 0)} = 81.4\%$$

Ideally = 100%

Uniform Model (everything 0) = 0%

Red Hat

Example

C_i	$pred = 0$	$pred = 1$
$label = 0$	$0 \rightarrow 0$	$0 \rightarrow 1$
$label = 1$	$1 \rightarrow 0$	$1 \rightarrow 1$

Recall confusion matrix

	pred = non-3	pred = 3
label = non-3	11052	251
label = 3	241	1056

$$\text{Specificity} = \frac{(0 \rightarrow 0)}{(0 \rightarrow 0) + (0 \rightarrow 1)} = 97.8\%$$

Ideally = 100%

Uniform Model (everything 0) = 100%

Example

Recall

Simple to interpret - if not too deep

Example

Decision Trees: Modifications

What if features are continuous not discrete?

Sort values for feature i

Try splitting at each value in training dataset

Decision Trees: Modifications

Regression problems:

Minimize: $$\frac{n_{left}}{n_{total}}\sigma^2_{left} + \frac{n_{right}}{n_{total}}\sigma^2_{right}$$

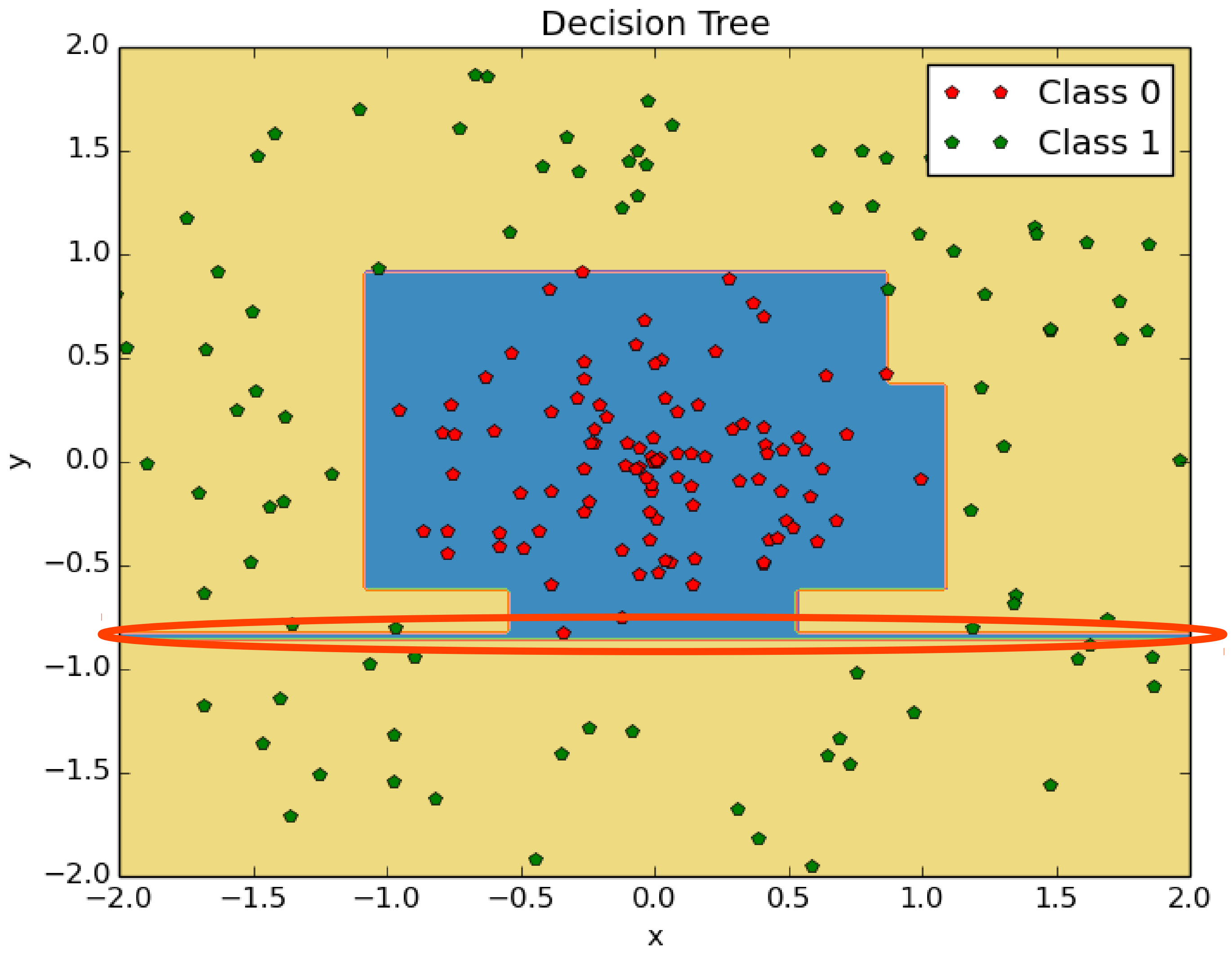
$$\sigma^2_{L/R} = \text{variance of numbers in node after split}$$
$$= \Sigma_{i=1}^{N_{L/R}} \frac{(y_i - c_{L/R})^2}{N_{L/R}}$$

Mean of values
in left/right node

Number of examples
in left/right node

Value of i^{th} example

Decision Trees: Pruning



Trees overfit easily!!!

Decision Trees: Pruning

Trees overfit easily!!!

One possible solution:

Grow a full tree

Starting from the leaves, remove splits that don't result in large increases in entropy

Ensembles of Trees

Decision Trees: Bagging

What if had many trees and average results?

This gives a huge improvement in practice

Decision Trees: Bagging

What if had many trees and average results?

This gives a huge improvement in practice

Intuition:

Each tree performs well on a subset of data

Trees vote or average to give overall prediction

Decision Trees: Bagging

Consider n random variables: $X_1, \ldots, X_n$

independent and identically distributed

$$\mathrm{mean}(X_i) \equiv \mu \qquad\qquad \mathrm{var}(X_i) \equiv \sigma^2$$

$$\mathrm{mean}\left(\frac{X_1 + \ldots + X_n}{n}\right) = \frac{\mathrm{mean}(X_1) + \ldots + \mathrm{mean}(X_n)}{n} = \frac{n\mu}{n} = \boxed{\mu}$$

$$\mathrm{var}\left(\frac{X_1 + \ldots + X_n}{n}\right) = \mathrm{var}\left(\frac{1}{n}\Sigma_i X_i\right) = \frac{1}{n^2}\Sigma_i \mathrm{var}(X_i) = \boxed{\frac{\sigma^2}{n}}$$

Red Hat

Decision Trees: Bagging

Run n decision trees on overlapping subsets of dataset

Bagging Average results from all n trees

If trees <u>independent</u> :

Averaging should give: $\boxed{\mu \pm \frac{\sigma}{\sqrt{n}}}$

$\to \mu$ as $n \to \infty$

Red Hat

Decision Trees: Bagging

Run n decision trees on overlapping subsets of dataset

Bagging Average results from all n trees

If trees independent :

Averaging should give: $\boxed{\mu \pm \frac{\sigma}{\sqrt{n}}}$

$\rightarrow \mu$ as $n \rightarrow \infty$

What if trees not independent?
(More precisely: errors from trees are uncorrelated)

Decision Trees: Bagging

Consider n random variables: $X_1, \ldots, X_n$

NOT independent but identically distributed

$$\text{mean}(X_i) \equiv \mu \qquad\qquad \text{var}(X_i) \equiv \sigma^2$$

Pairwise correlations: $\mathbb{E}(X_i X_j) - \mathbb{E}X_i \mathbb{E}X_j \equiv \rho\sigma^2, i \neq j$

$$\text{mean}\left(\frac{X_1 + \ldots + X_n}{n}\right) = \frac{\text{mean}(X_1) + \ldots + \text{mean}(X_n)}{n} = \frac{n\mu}{n} = \boxed{\mu}$$

$$\text{var}\left(\frac{X_1 + \ldots + X_n}{n}\right) = \boxed{\frac{(1-\rho)\sigma^2}{n} + \rho\sigma^2}$$

Red Hat

Decision Trees: Bagging

Run n decision trees on overlapping subsets of dataset

Bagging Average results from all n trees

If trees not independent :

Averaging should give: $\boxed{\mu \pm \left(\frac{(1-\rho)\sigma^2}{n} + \rho\sigma^2 \right)}$

$\rightarrow \boxed{\mu \pm \rho\sigma^2}$ as $n \rightarrow \infty$

Correlated trees will always have variance in the average results

Red Hat

Decision Trees: Bagging

$$\mathrm{var}(\frac{X_1 + \ldots + X_n}{n}) = \frac{(1-\rho)\sigma^2}{n} + \rho\sigma^2$$
$$\to \mu \pm \rho\sigma^2 \text{ as } n \to \infty$$

Even with many trees, have variance because of correlations

Solution: Reduce correlations between trees

Red Hat

Reducing Correlations: Options

Train M trees

Recall training loop for a decision tree :

Pick rows

Loop over features:

  Loop over distinct values/cuts :

    Compute entropy/gini/variance if split on this feature and cut

Pick feature and cut minimizing metric

Repeat recursively for each daughter node till stopping condition

Reducing Correlations: Options

Train M trees

Recall training loop for a decision tree :

Pick rows
Loop over features:
    Loop over distinct values/cuts :
        Compute entropy/gini/variance if split on this feature and cut
Pick feature and cut minimizing metric
Repeat recursively for each daughter node till stopping condition

Where can we insert some randomness?

Reducing Correlations: Options

Train M trees

Recall training loop for a decision tree :

Pick $\boxed{\text{rows}}$

Loop over $\boxed{\text{features}}$:

Loop over distinct $\boxed{\text{values/cuts}}$:

Compute entropy/gini/variance if split on this feature and cut

Pick feature and cut minimizing metric

Repeat recursively for each daughter node till stopping condition

Where can we insert some randomness?

Reducing Correlations: Options

Train M trees

Option 1: Each tree trains on fraction p_s of rows

Option 2: Each tree trains on fraction p_f of features/columns

Option 3: Each **node** only looks at a fraction p_n of features

Option 4: Each node only looks at a fraction p_e of features **and** cuts

Reducing Correlations: Options

Train M trees

Option 1: Each tree trains on fraction p_s of rows

$\boxed{\text{Pasting}}$

Reducing Correlations: Options

Train M trees

Option 2: Each tree trains on fraction p_f of features/columns

$$\boxed{\text{Random subspaces}}$$

Reducing Correlations: Options

Train M trees

Option 1: Each tree trains on fraction p_s of rows

Option 2: Each tree trains on fraction p_f of features/columns

$\boxed{\text{Random patches}}$

Reducing Correlations: Options

Train M trees

Option 1: Each tree trains on fraction p_s of rows
bootstrapping (coming later)

Option 3: Each **node** only looks at a fraction p_n of features

$\boxed{\text{Random Forest}}$

(used heavily)

Reducing Correlations: Options

Train M trees

Option 4: Each node only looks at a fraction p_e of features **and** cuts

$\boxed{\text{Extra (Extremely Randomized) Tree}}$

Sampling Data

Make each tree look at a subset of the data

Boostrapping

N rows in data

Each tree draws N rows by random sampling with replacement

Some rows won't get picked

Some rows will get picked multiple times

Sampling Data

Boostrapping

N rows in data

Each tree draws N rows by random sampling with replacement

Each tree still trains on N rows

Rows not seen by tree form a validation dataset

These are called **Out-of-bag or OOB** samples

Performance of tree can be evaluated on the OOB data

Sampling Data

Boostrapping

Generally used in statistics to estimate variance in measurements

Bagging

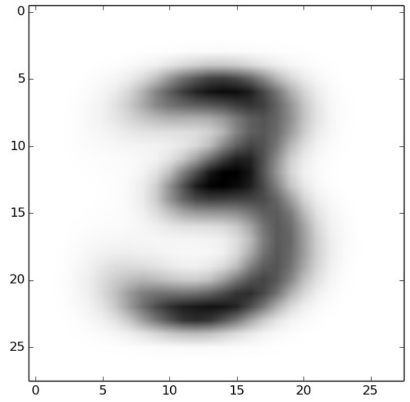
sklearn.ensemble.BaggingClassifier

class sklearn.ensemble.BaggingClassifier(**base_estimator**=None, **n_estimators**=10, **max_samples**=1.0, **max_features**=1.0, **bootstrap**=True, bootstrap_features=False, **oob_score**=False, warm_start=False, **n_jobs**=None, random_state=None, verbose=0)

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html

Random Forest Example

Random Forest: Digits Example



Recall digits dataset

Let's use a random forest to identify the digit 3 (label $= 1$) from other digits (label $= 0$)

Using 100 trees in forest - no fine tuning of parameters

Random Forest: Digits Example

Decision Tree

	pred = non-3	pred = 3
label = non-3	11052	251
label = 3	241	1056

$$\text{Accuracy} = 96.1\%$$

$$\text{Sensitivity/Recall} = 81.4\%$$

$$\text{Specificity} = 97.8\%$$

$$\text{type I error} = 2.2\%$$

$$\text{type II error} = 18.6\%$$

Random Forest

	pred = non-3	pred = 3
label = non-3	11320	5
label = 3	241	1034

$$\text{Accuracy} = 98.0\%$$

$$\text{Sensitivity/Recall} = 81.1\%$$

$$\text{Specificity} = 99.9\%$$

$$\text{type I error} = 0.0\%$$

$$\text{type II error} = 18.9\%$$

Further Topics

What we didn’t cover

Boosting

Interpretation (but see practical session)

Variable Importance

Treeinterpreter - Deltas

Partial Dependence Plots

Questions?

Thank you

Red Hat is the world’s leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHat

Boosting: AdaBoost

Problem Definition: Classification

$$\text{Given features} : x_1, x_2, \ldots, x_n$$

$$\text{Predict class or label, y} = 0 \text{ or } 1$$

Problem Definition: Classification

$$\text{Given features} : x_1, x_2, \ldots, x_n$$

$$\text{Predict class or label, y} = \cancel{0} \text{ or } 1$$

$$-1$$

$$\text{for convenience}$$

Red Hat

Boosting

Another way of combining various trees (or any estimators)

to build powerful models

Different from bagging!

AdaBoost: First Look at Boosting

What if want predictor to be of form:

$$F(x) = \text{sgn}(\underbrace{\Sigma_{m=1}^{M} \beta_m F_m(x)}_{\text{Weighted sum of m predictors}})$$

Red Hat

AdaBoost: First Look at Boosting

For any supervised learning algorithm, we have a cost function:

Sum over input examples

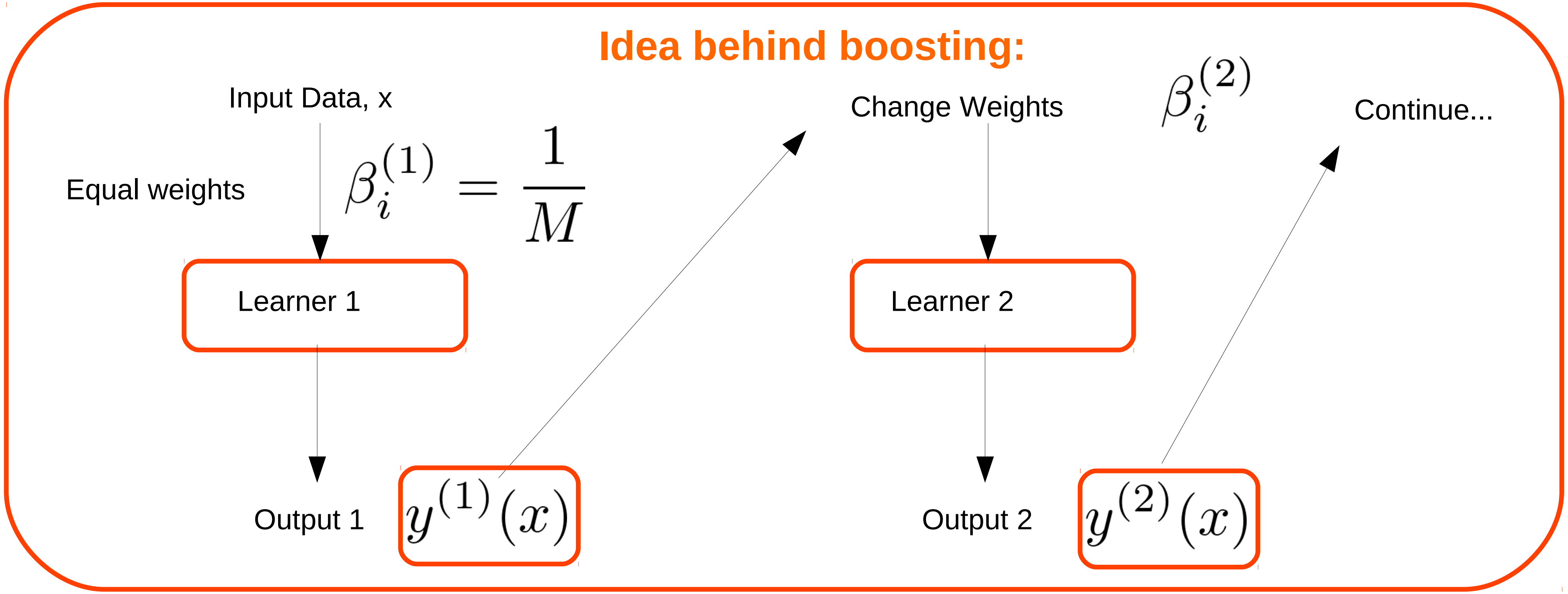
$$C(w_0, w_1, \ldots, w_n) = \underbrace{\Sigma_{i=1}^{M} C_i(w_0, w_1, \ldots, w_n)}$$

Assumption: every example **independently** and **equally** contributes to cost

$$C(w_0, w_1, \ldots, w_n) = \Sigma_{i=1}^{M} \beta_i C_i(w_0, w_1, \ldots, w_n)$$

Can emphasize some input examples more than others by introducing **multipliers**

Boosting: AdaBoost



Weigh examples wrongly classified by Learner 1 more and feed to Learner 2

$$\text{Output} = \Sigma_{i=1}^{N_{iter}} \alpha^{(i)} y^{(i)}(x)$$

Weighing over all classifiers depending on each classifier's accuracy = **trust factor**

Boosting: AdaBoost

Start:

$$C^{(1)}(w_0, w_1, \ldots, w_n) = \Sigma_{i=1}^{M} \beta_i^{(1)} C_i(w_0, w_1, \ldots, w_n)$$

$$\beta_i^{(1)} = \frac{1}{M}$$

First classifier - each example with equal weight

Boosting: AdaBoost

Error with weights:

$$\epsilon^{(1)} = \frac{\Sigma_{i \in \text{Incorrect}} \beta_i^{(1)}}{\Sigma_{i=1}^{M} \beta_i^{(1)}}$$

Sum over in**correctly predicted** examples

Sum over **all** examples

% of total weight incorrectly predicted

Boosting: AdaBoost

$$\text{Define: } e^{\alpha^{(1)}} \equiv \frac{1 - \epsilon^{(1)}}{\epsilon^{(1)}}$$

% of weights predicted **correctly**

% of weights predicted **incorrectly**

$$100\% \text{ accurate model} \longrightarrow \epsilon^{(1)} = 0 \longrightarrow e^{\alpha^{(1)}} = \infty \longrightarrow \alpha^{(1)} = \infty$$

$$50\% \text{ accurate model} \longrightarrow \epsilon^{(1)} = 0.50 \longrightarrow e^{\alpha^{(1)}} = 1 \longrightarrow \alpha^{(1)} = 0$$

$$0\% \text{ accurate model} \longrightarrow \epsilon^{(1)} = 1 \longrightarrow e^{\alpha^{(1)}} = 0 \longrightarrow \alpha^{(1)} = -\infty$$

Red Hat

Boosting: AdaBoost

Update weights:

$$\beta_i^{(2)} = \beta_i^{(1)} \text{ if example } i \text{ correctly predicted}$$

$$\beta_i^{(2)} = \beta_i^{(1)} e^{\underbrace{\alpha^{(1)}}_{\frac{1-\epsilon^{(1)}}{\epsilon^{(1)}}}} \text{ if example } i \text{ incorrectly predicted}$$

Red Hat

Boosting: AdaBoost

General algorithm: Learner m $\rightarrow$ Learner m+1

Minimize: $C^{(m)}(w_0, w_1, \ldots, w_n) = \Sigma_{i=1}^{M} \beta_i^{(m)} C_i(w_0, w_1, \ldots, w_n)$

Compute: $\epsilon^{(m)} = \dfrac{\Sigma_{i \in \mathrm{Incorrect}} \beta_i^{(m)}}{\Sigma_{i=1}^{M} \beta_i^{(m)}}$ $\qquad e^{\alpha^{(m)}} \equiv \dfrac{1 - \epsilon^{(m)}}{\epsilon^{(m)}}$

Update: $\beta_i^{(m+1)} = \beta_i^{(m)}$ if example i correctly classified

$\beta_i^{(m+1)} = \beta_i^{(m)} e^{\alpha^{(m)}}$ if example i incorrectly classified