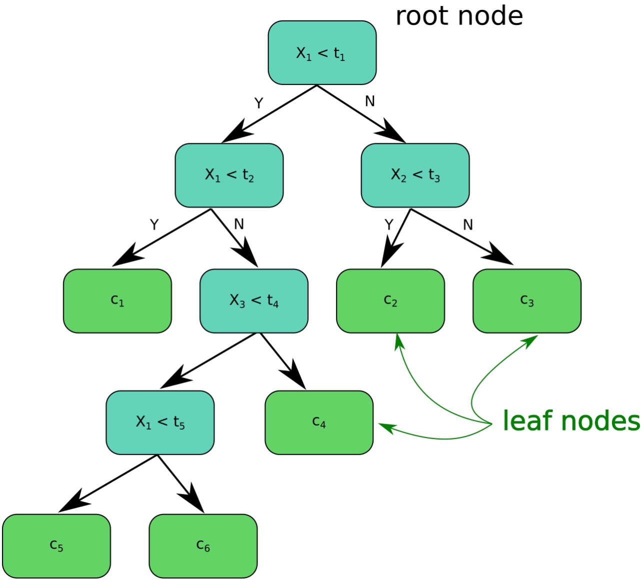
Decision Tree and Boosting

David Ivan

June 18, 2018

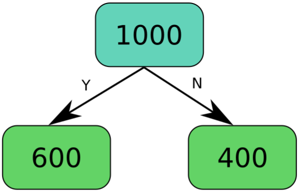
Decision Tree



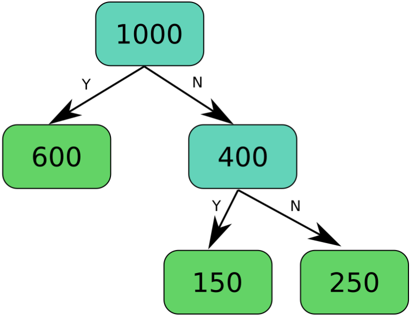
Train a decision tree

1000

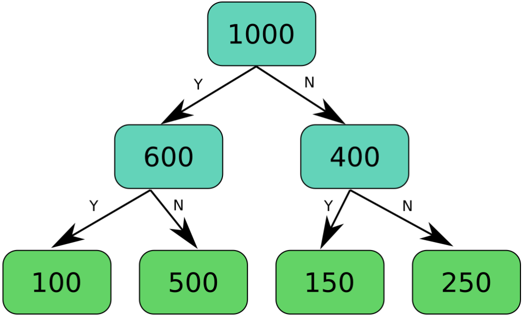
Train a decision tree



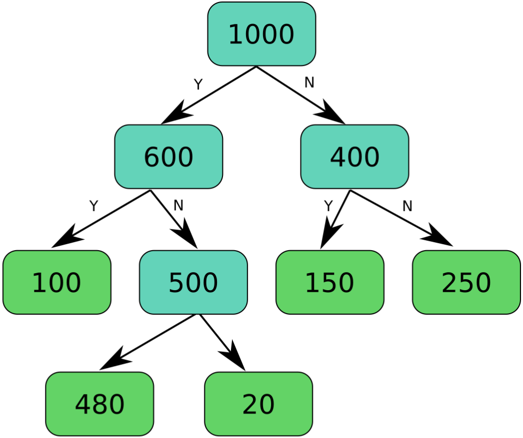
Train a decision tree



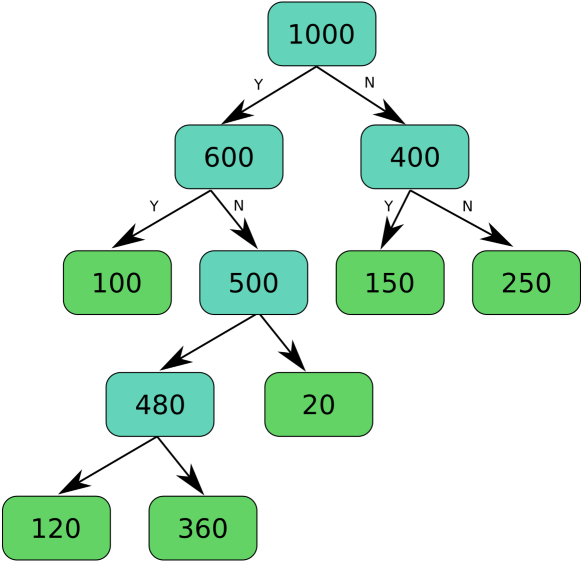
Train a decision tree



Train a decision tree



Train a decision tree

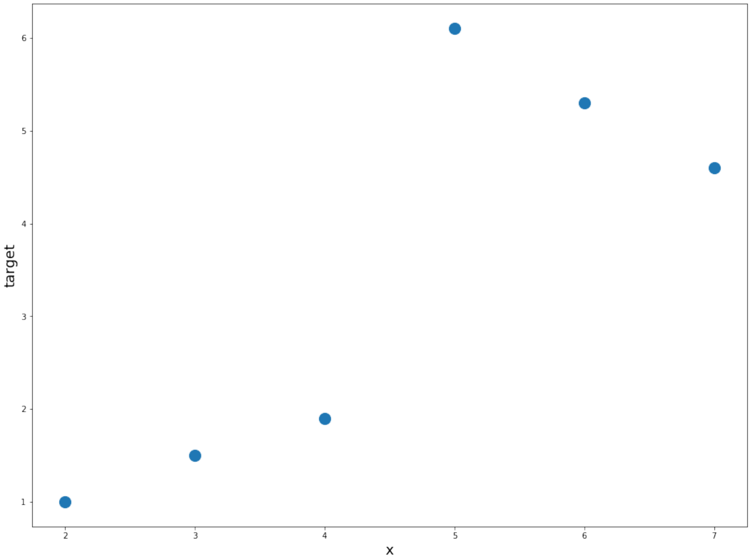


Split

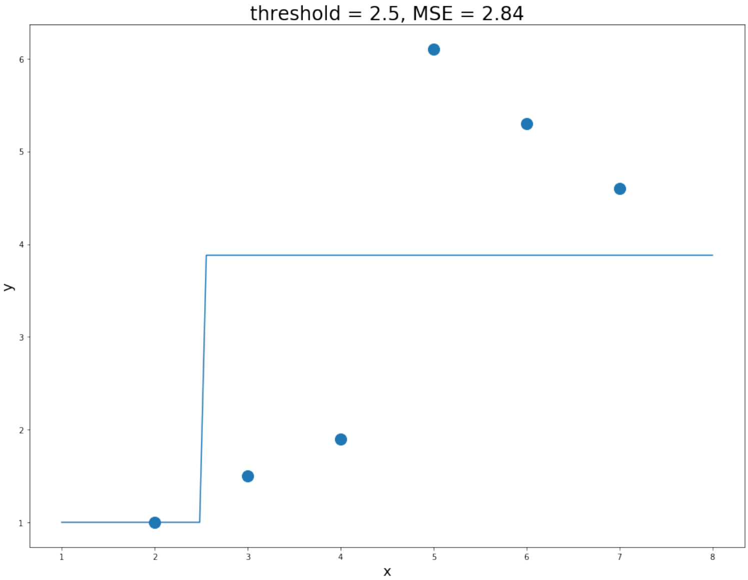
How to split a node?

```
max_split_quality = 0
for f in features:
  x_sorted = sort(x[:,f])
  for x_0, x_1 in zip(x_sorted[:-1], x_sorted[1:]):
    midpoint = 0.5*(x_0 + x_1)
    split_quality = get_split_quality(x, f, midpoint)
    if split_quality > max_split_quality:
      max_split_quality = split_quality
      feature_split = f
      threshold = midpoint
return feature_split, threshold
```

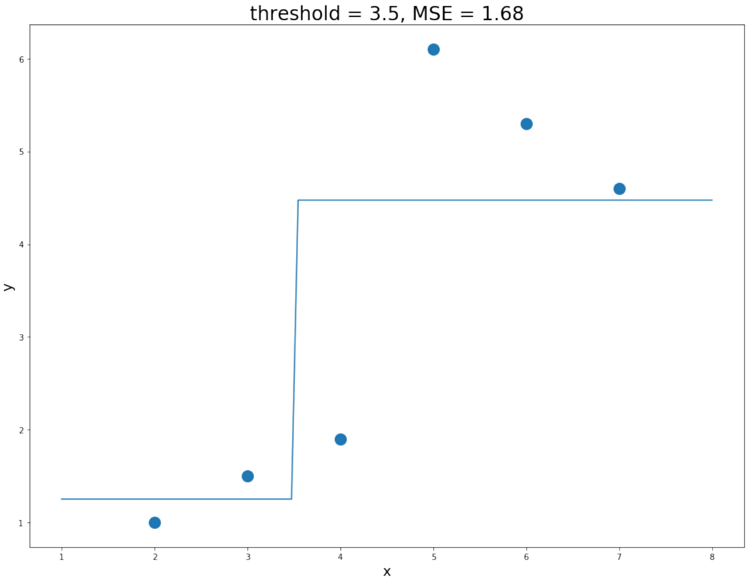
Split



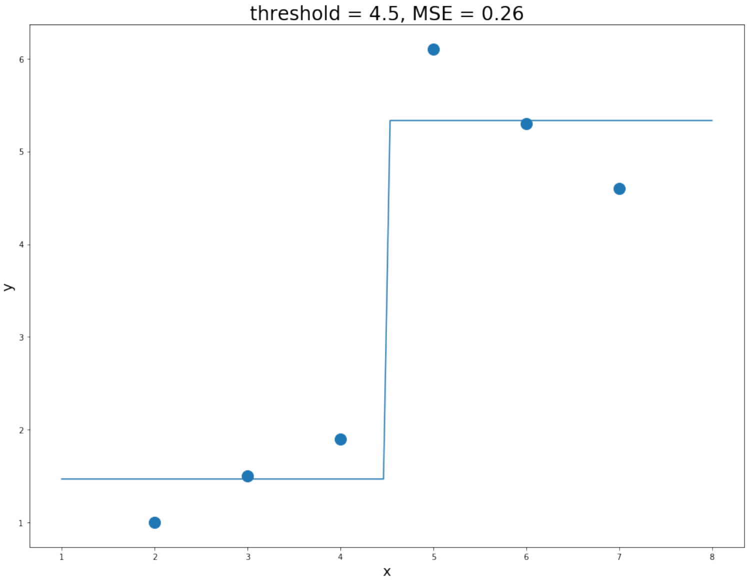
Split



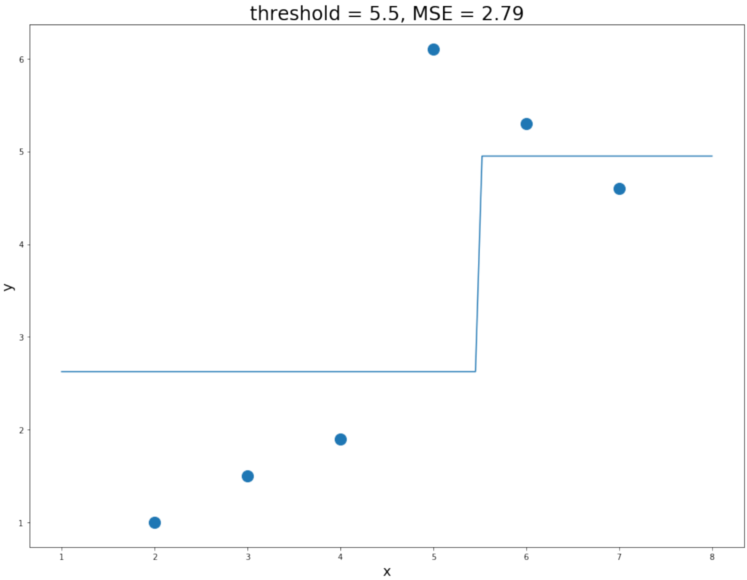
example



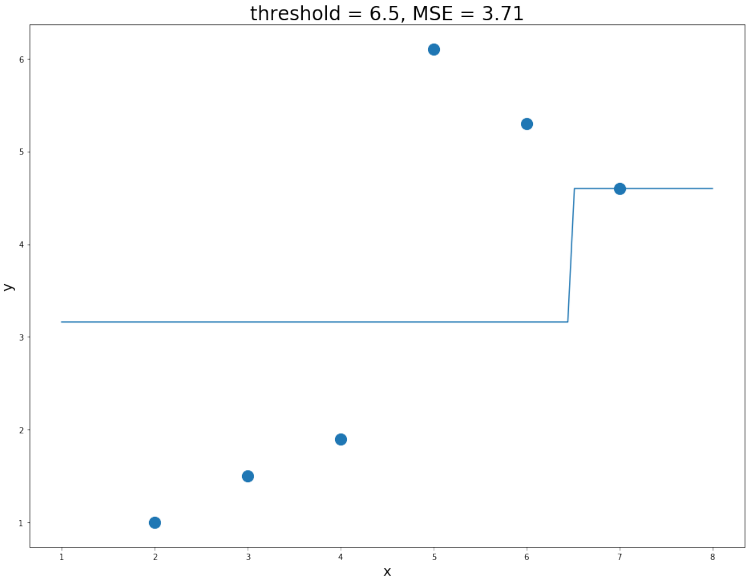
example



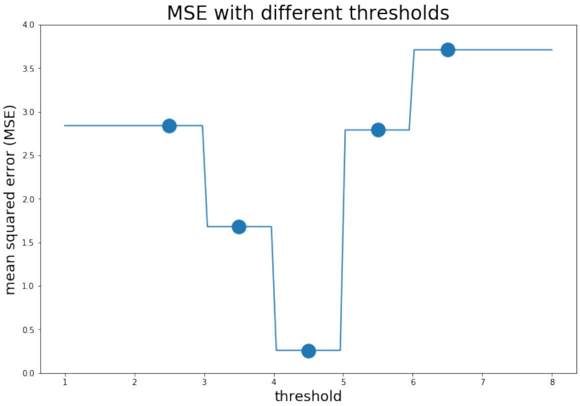
example



example

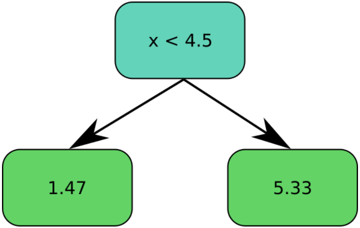


example

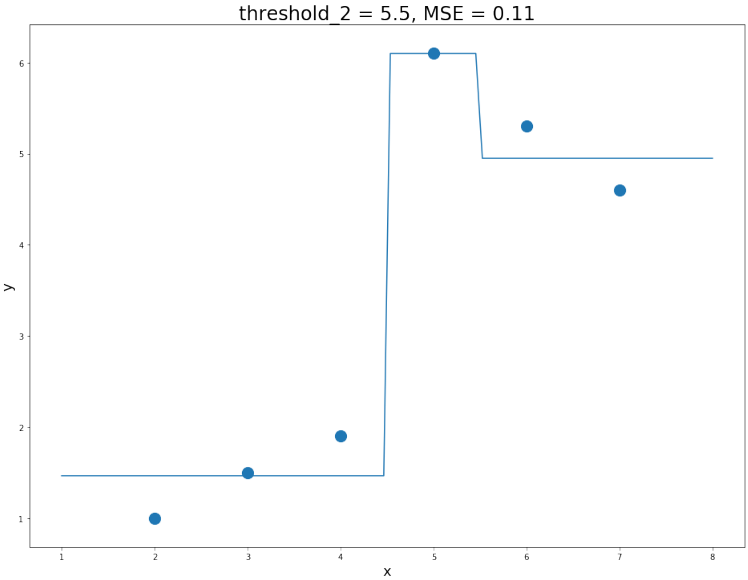


MSE with different thresholds
mean squared error (MSE)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
1
2
3
4
5
6
7
8
threshold

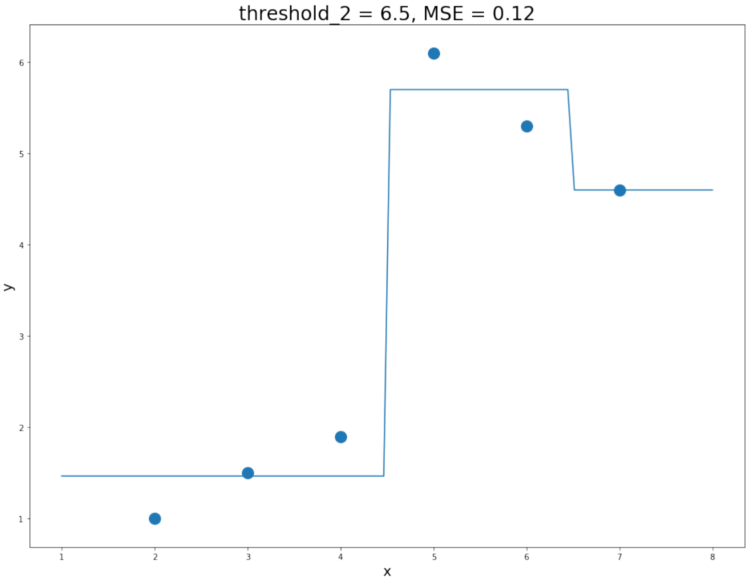
example



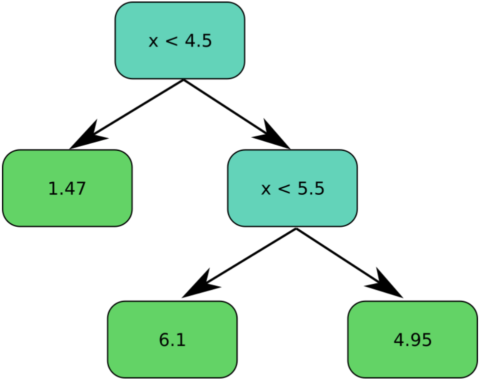
example



example

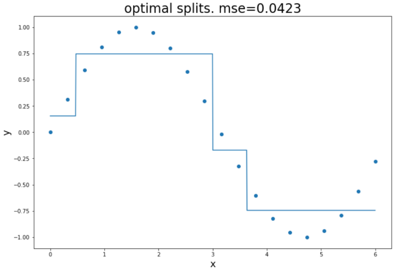


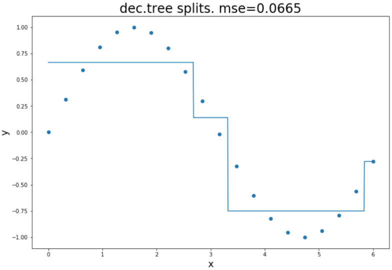
example



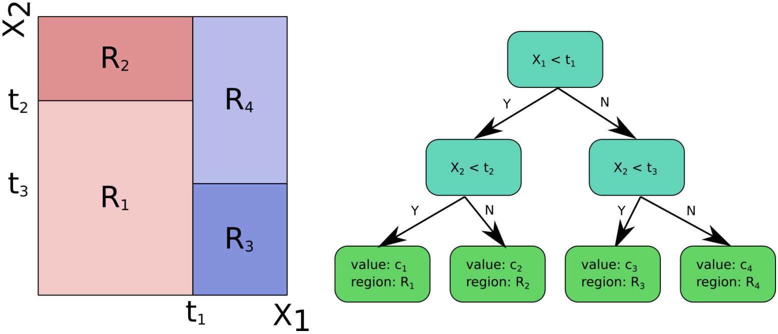
greedy splits vs optimal splits

greedy split is not optimal in long-term

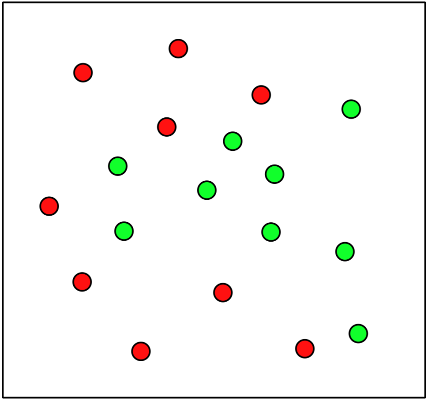




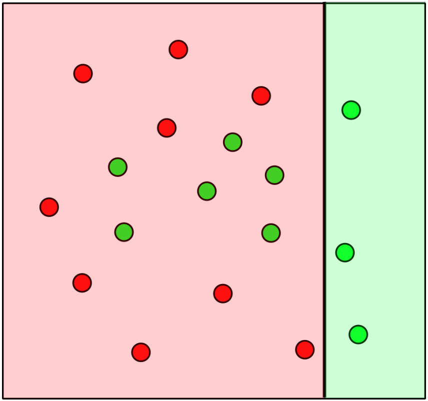
partitioning the feature space



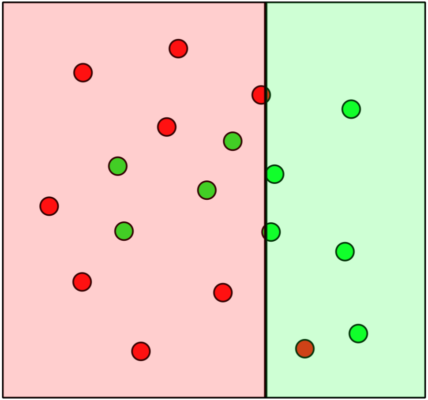
Classification with decision trees



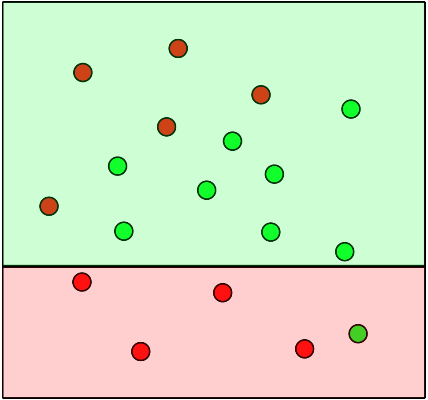
Classification with decision trees



Classification with decision trees



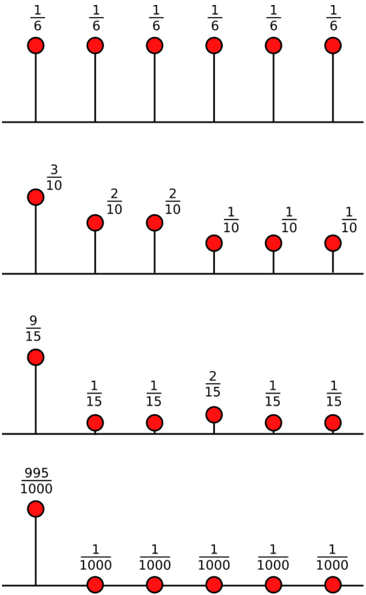
Classification with decision trees



Impurity measure: entropy

$$H(X) = H(p_1, p_2, ..., p_n) = -\Sigma_i^n p_i \cdot log_2(p_i)$$

Information Theory



H(X)=2.58

H(Y)=2.45

H(Z)=1.87

H(V)=0.057

Information Theory

Conditional entropy:

$$H(X|Y = y) = -\Sigma_i p(x_i|y) \cdot log_2(p(x_i|y))$$

$$H(X|Y) = \Sigma_y P(Y = y) \cdot H(X|Y = y)$$

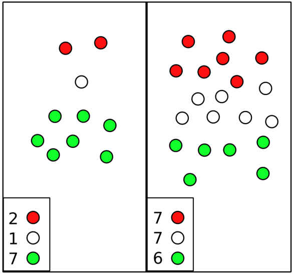
Information gain (information conveyed about X by Y):

$$I(X|Y) = H(X) - H(X|Y)$$

Information Theory

$$X = \begin{cases} \text{"red"} & (9/30) \\ \text{"white"} & (8/30) \\ \text{"green"} & (13/30) \end{cases}$$

$$Y = \begin{cases} \text{"left"} & (10/30) \\ \text{"right"} & (20/30) \end{cases}$$



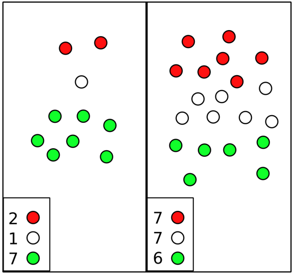
$$(X|Y=\text{"left"}) = \begin{cases} \text{"red"} & (2/10) \\ \text{"white"} & (1/10) \\ \text{"green"} & (7/10) \end{cases}$$

$$(X|Y=\text{"right"}) = \begin{cases} \text{"red"} & (7/20) \\ \text{"white"} & (7/20) \\ \text{"green"} & (6/20) \end{cases}$$

Information Theory

$$X = \begin{cases} \text{"red"} & (9/30) \\ \text{"white"} & (8/30) \\ \text{"green"} & (13/30) \end{cases}$$

$$Y = \begin{cases} \text{"left"} & (10/30) \\ \text{"right"} & (20/30) \end{cases}$$



$$(X|Y=\text{"left"}) = \begin{cases} \text{"red"} & (2/10) \\ \text{"white"} & (1/10) \\ \text{"green"} & (7/10) \end{cases}$$

$$(X|Y=\text{"right"}) = \begin{cases} \text{"red"} & (7/20) \\ \text{"white"} & (7/20) \\ \text{"green"} & (6/20) \end{cases}$$

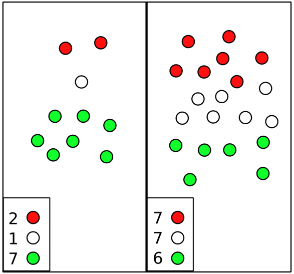
$$H(X) = H(\frac{9}{30}, \frac{8}{30}, \frac{13}{30})$$

$$= -\frac{9}{30} log_2(\frac{9}{30}) - \frac{8}{30} log_2(\frac{8}{30}) - \frac{13}{30} log_2(\frac{13}{30}) = 1.5524$$

Information Theory

$$X = \begin{cases} \text{"red"} & (9/30) \\ \text{"white"} & (8/30) \\ \text{"green"} & (13/30) \end{cases}$$

$$Y = \begin{cases} \text{"left"} & (10/30) \\ \text{"right"} & (20/30) \end{cases}$$



$$(X|Y=\text{"left"}) = \begin{cases} \text{"red"} & (2/10) \\ \text{"white"} & (1/10) \\ \text{"green"} & (7/10) \end{cases}$$

$$(X|Y=\text{"right"}) = \begin{cases} \text{"red"} & (7/20) \\ \text{"white"} & (7/20) \\ \text{"green"} & (6/20) \end{cases}$$

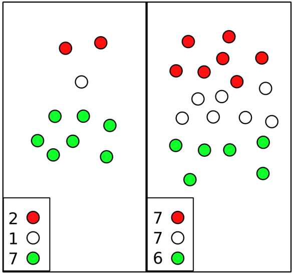
$$H(X|Y = \text{left}) = H(\frac{2}{10}, \frac{1}{10}, \frac{7}{10})$$

$$= -\frac{2}{10} log_2(\frac{2}{10}) - \frac{1}{10} log_2(\frac{1}{10}) - \frac{7}{10} log_2(\frac{7}{10}) = 1.1568$$

Information Theory

$$X = \begin{cases} \text{"red"} & (9/30) \\ \text{"white"} & (8/30) \\ \text{"green"} & (13/30) \end{cases}$$

$$Y = \begin{cases} \text{"left"} & (10/30) \\ \text{"right"} & (20/30) \end{cases}$$



$$(X|Y=\text{"left"}) = \begin{cases} \text{"red"} & (2/10) \\ \text{"white"} & (1/10) \\ \text{"green"} & (7/10) \end{cases}$$

$$(X|Y=\text{"right"}) = \begin{cases} \text{"red"} & (7/20) \\ \text{"white"} & (7/20) \\ \text{"green"} & (6/20) \end{cases}$$

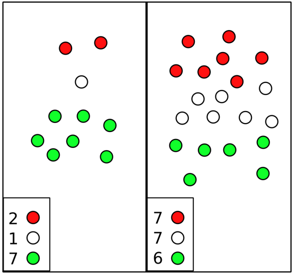
$$H(X|Y = \text{right}) = H(\frac{7}{20}, \frac{7}{20}, \frac{6}{20})$$

$$= -\frac{7}{20} log_2(\frac{7}{20}) - \frac{7}{20} log_2(\frac{7}{20}) - \frac{6}{20} log_2(\frac{6}{20}) = 1.5813$$

Entropy

$$X = \begin{cases} \text{"red"} & (9/30) \\ \text{"white"} & (8/30) \\ \text{"green"} & (13/30) \end{cases}$$

$$Y = \begin{cases} \text{"left"} & (10/30) \\ \text{"right"} & (20/30) \end{cases}$$



$$(X|Y=\text{"left"}) = \begin{cases} \text{"red"} & (2/10) \\ \text{"white"} & (1/10) \\ \text{"green"} & (7/10) \end{cases}$$

$$(X|Y=\text{"right"}) = \begin{cases} \text{"red"} & (7/20) \\ \text{"white"} & (7/20) \\ \text{"green"} & (6/20) \end{cases}$$

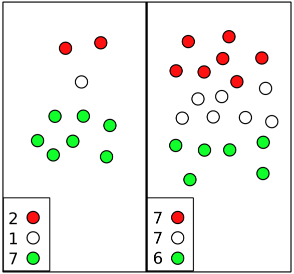
$$H(X|Y) = P(Y = \text{left}) \cdot H(X|Y = \text{left}) + P(Y = \text{right}) \cdot H(X|Y = \text{right}) =$$

$$= \frac{10}{30} \cdot 1.1568 + \frac{20}{30} \cdot 1.5813 = 1.4398$$

Information Theory

$$X = \begin{cases} \text{"red"} & (9/30) \\ \text{"white"} & (8/30) \\ \text{"green"} & (13/30) \end{cases}$$

$$Y = \begin{cases} \text{"left"} & (10/30) \\ \text{"right"} & (20/30) \end{cases}$$

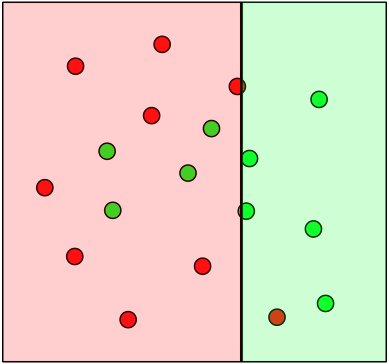


$$(X|Y=\text{"left"}) = \begin{cases} \text{"red"} & (2/10) \\ \text{"white"} & (1/10) \\ \text{"green"} & (7/10) \end{cases}$$

$$(X|Y=\text{"right"}) = \begin{cases} \text{"red"} & (7/20) \\ \text{"white"} & (7/20) \\ \text{"green"} & (6/20) \end{cases}$$

$$I(X|Y) = H(X) - H(X|Y) = 1.5524 - 1.4398 = 0.1126 \text{ bit}$$

Information gain - example

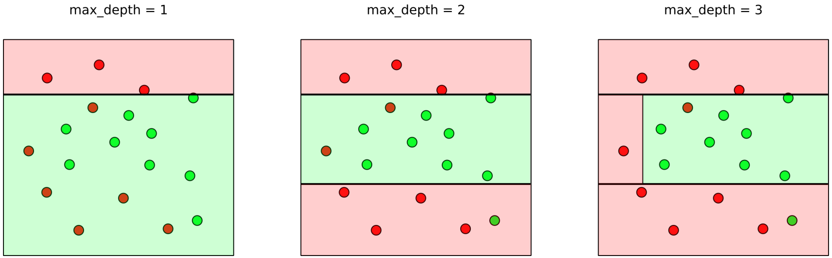


$$H(X) = H(\tfrac{9}{18}, \tfrac{9}{18}) = 1$$
$$H(X|Y = \text{left}) = H(\tfrac{8}{12}, \tfrac{4}{12}) = 0.918$$
$$H(X|Y = \text{right}) = H(\tfrac{5}{6}, \tfrac{1}{6}) = 0.650$$
$$H(X|Y) = \tfrac{12}{18} H(X|Y = \text{left}) + \tfrac{6}{18} H(X|Y = \text{right}) = 0.8287$$
$$I(X|Y) = H(X) - H(X|Y) = 1 - 0.8287 = 0.1713$$

Most common stopping criterias

- maximum depth
- maximum leaf node
- min samples leaf (only split a node if both children resulting have at least this many samples)
- min samples split (only split a node if it has at least this many samples)

classification example

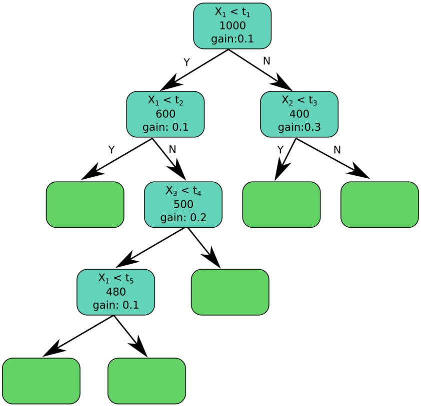


feature importances

Calculate the feature importances as follows:

```
feat_imp = [0 for f in features]
for branch in tree.branches:
   feature = branch.feature
   gain = branch.information_gain
   weight = branch.samples_ratio
   feat_imp[f] += weight * gain
feat_imp = normalize(feat_imp)
```

feature importances - example



f_i = [0, 0, 0]

f_i[X1] += gain*weight = 0.1*1=0.1
f_i = [0.1, 0, 0]

f_i[X1] += 0.1 * 0.6 = 0.06
f_i = [0.16, 0, 0]

f_i[X2] += 0.3 * 0.4 = 0.12
f_i = [0.16, 0.12, 0]

f_i[X3] += 0.2 * 0.5 = 0.1
f_i = [0.16, 0.12, 0.1]

f_i[X1] += 0.1 * 0.48 = 0.048
f_i = [0.208, 0.12, 0.1]

f_i = [0.486, 0.28, 0.234]

pros and cons

pros:

- no need to scale input features
- highly interpretable
- provides feature importance

cons:

- not generalize well
- not so good at picking linear relationships between features
- easy to overfit
- can be computationally expensive
- decision boundaries are parallel to the axes

Boosting

Boosting

Converting weak learners into one strong learner.
Boosting algorithms:

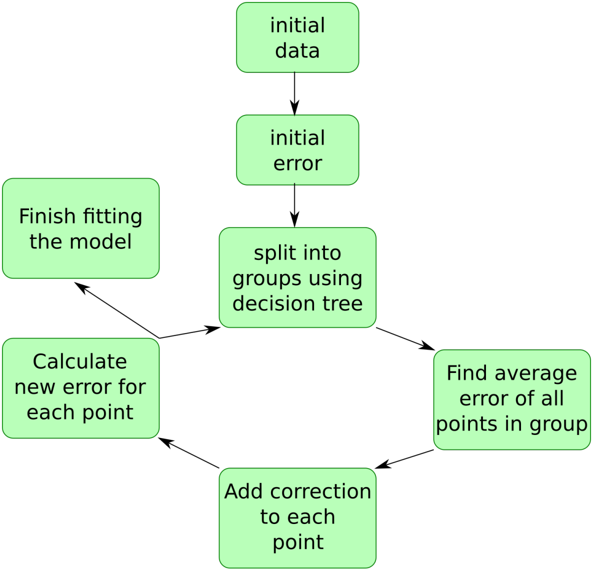
- AdaBoost
- Gradient Tree Boosting
- BrownBoost
- CoBoosting
- GentleBoost
- etc.

why boosting?

It is very powerful. Some competitions that have been won (in Kaggle) using boosting, or boosting in conjunction with other techniques:

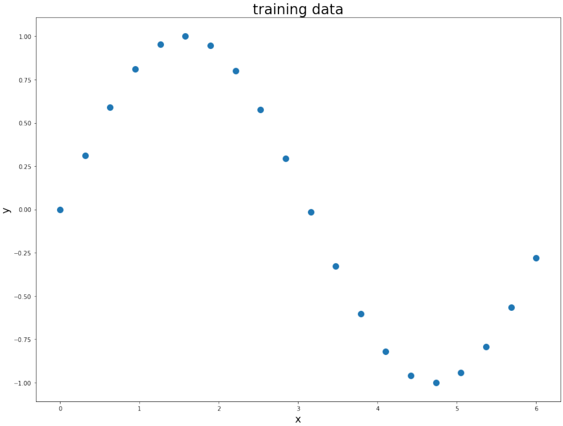
- Liberty Mutual Property Inspection
- Caterpillar Tube Pricing
- Avito Duplicate Ads Detection
- Facebook Robot Detection
- Otto Product Classification

Gradient Boosted Trees Regression



Gradient Boosted Trees Regression

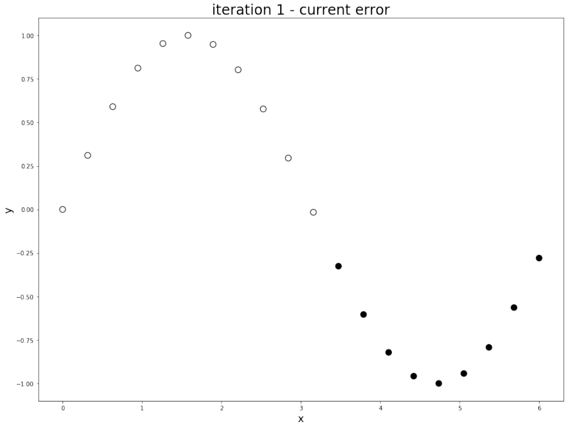
- initial prediction can be 0 (or average)
- use weak learners (e.g. decision tree with max_depth = 2)
- correction = learning_rate * error



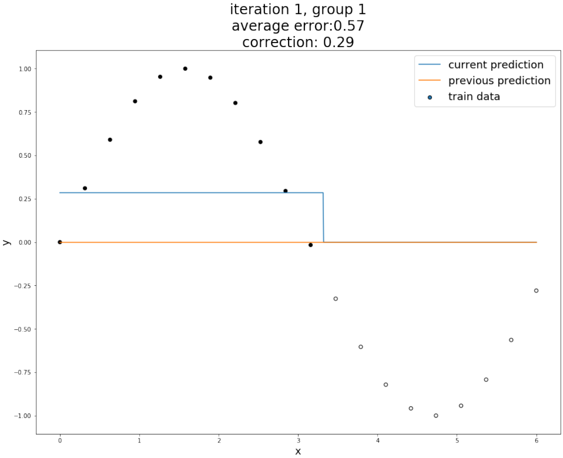
Apply the boosting algorithm with parameters:

- learning_rate = 0.5
- max_depth = 1

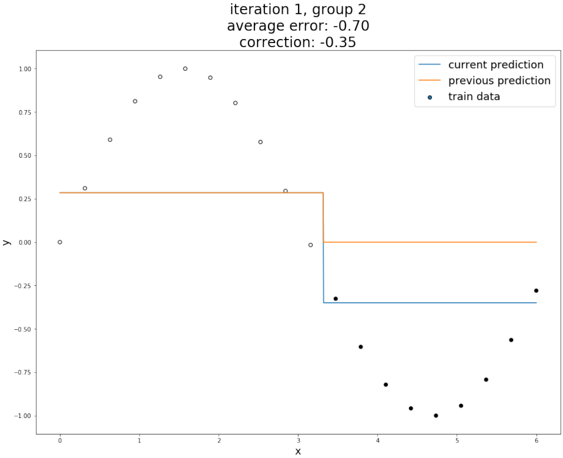
initial error and groups



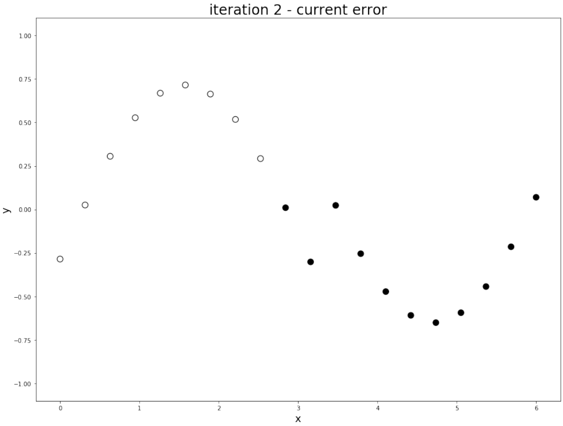
iteration 1 - group 1 - corrections



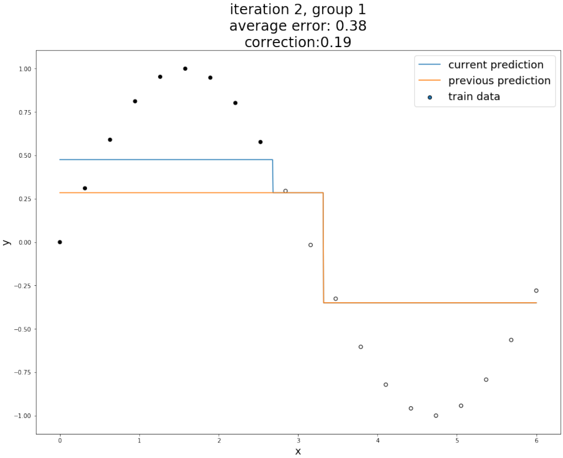
iteration 1 - group 2 - corrections



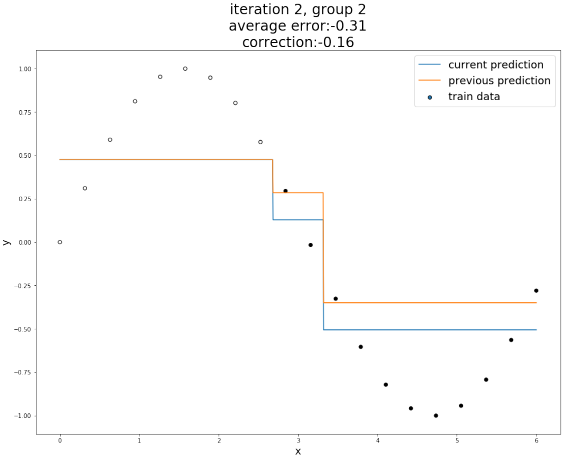
iteration 2 - current error



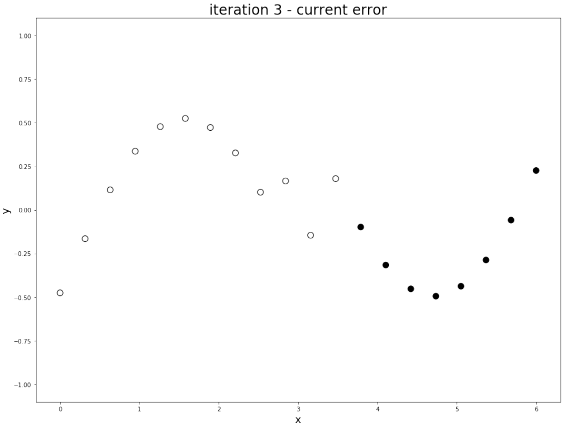
iteration 2 - group 1



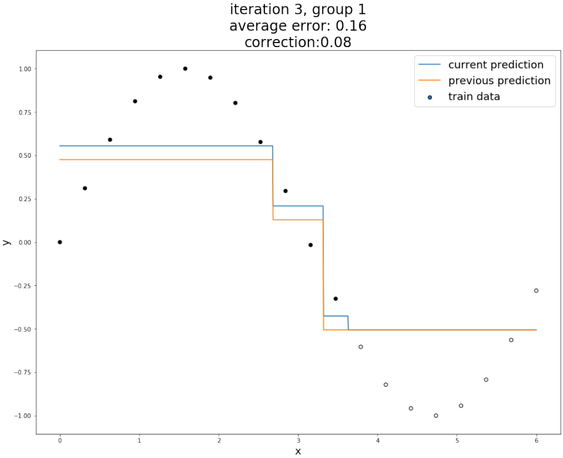
iteration 2 - group 2



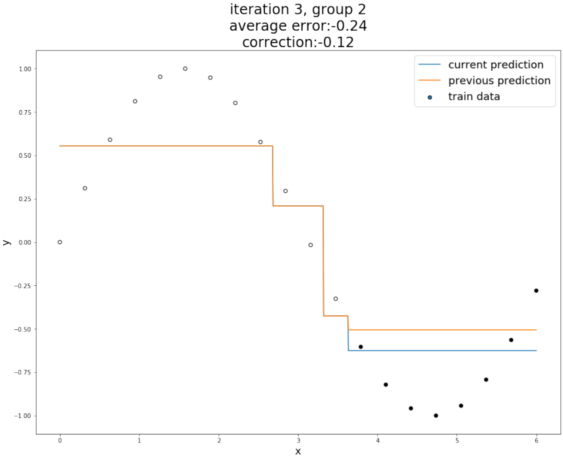
iteration 3


iteration 3 - current error
y
x
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
1
2
3
4
5
6

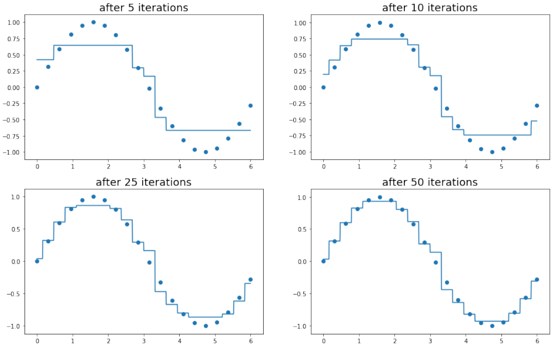
iteration 3 - group 1

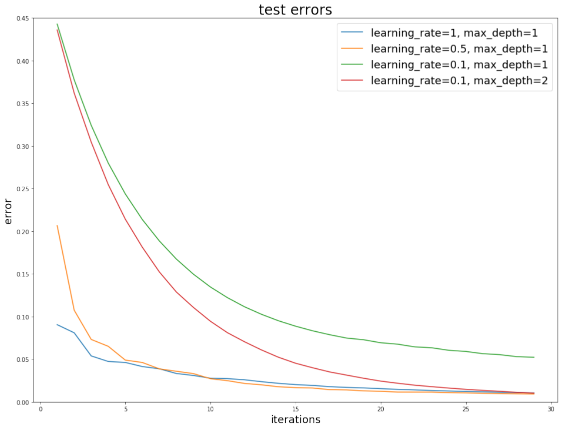


iteration 3 - group 2



test errors


after 5 iterations
after 10 iterations
after 25 iterations
after 50 iterations



test errors
learning_rate=1, max_depth=1
learning_rate=0.5, max_depth=1
learning_rate=0.1, max_depth=1
learning_rate=0.1, max_depth=2
error
iterations
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
5
10
15
20
25
30

pros and cons

pros:

- powerful at prediction
- harder to overfit

cons:

- slower training (sequential train)
- hard to interpret the predicted value

software

sklearn.ensemble.GradientBoostingClassifier
sklearn.ensemble.GradientBoostingRegressor
sklearn.ensemble.AdaBoostClassifier
sklearn.ensemble.AdaBoostRegressor
xgboost: Scalable and Flexible Gradient Boosting