Decision Tree, Random Forest, and XGBoost

Jindal K. Shah (jindal.shah@okstate.edu)
School of Chemical Engineering
Oklahoma State University

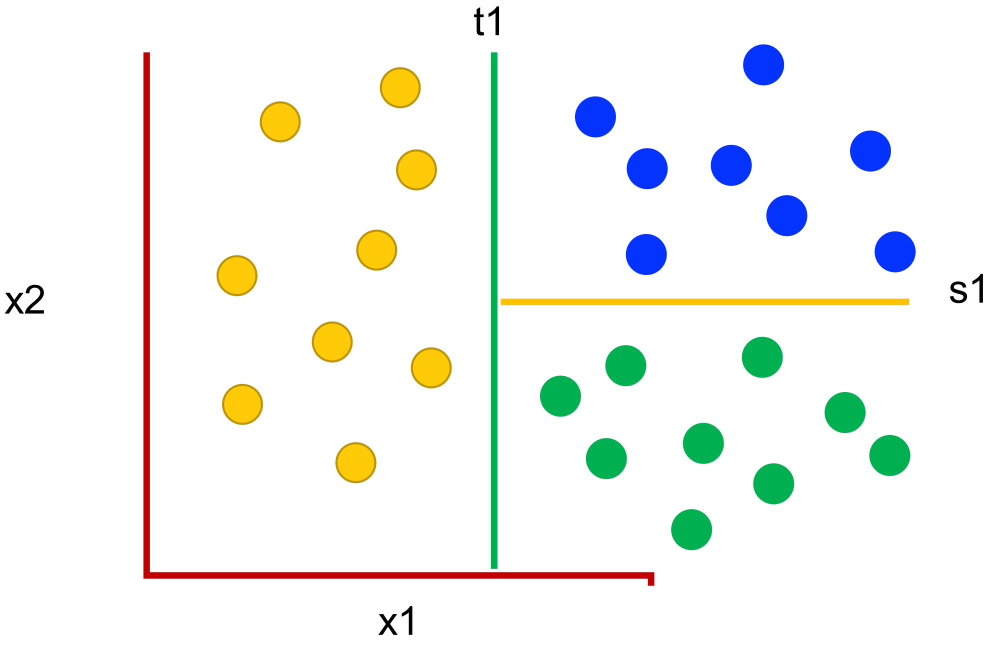
9th iCoMSE Workshop: Machine Learning for Molecules

Topics

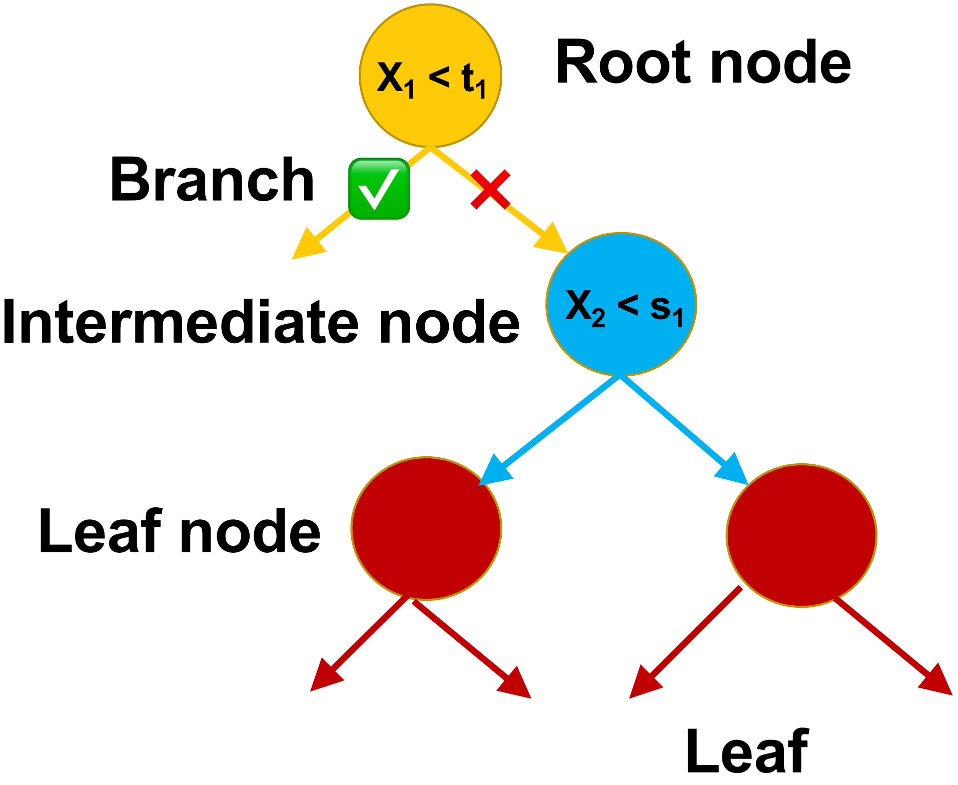
- Decision Tree
- Bagging (bootstrap + aggregating)
- Random forests
- Hands-on Exercise
- Gradient Boosting and extreme Gradient Boosting (XGBoost)

Decision Tree

- Supervised learning method
- Regression/classification
- Non-linear model
- If…then…else…
- Non-parametric model
- Piecewise continuous



Decision Tree



$X_1 < t_1$

Root node

Branch ✓ ✗

$X_2 < s_1$

Intermediate node

Leaf node

Leaf

Depth of the tree = maximum number of branches to reach a leaf

Objective function

- The objective is to minimize the residual sum of squares

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

- Here *J* represents the number of regions the feature space is partitioned into. The prediction in each of the regions is given by the average response.

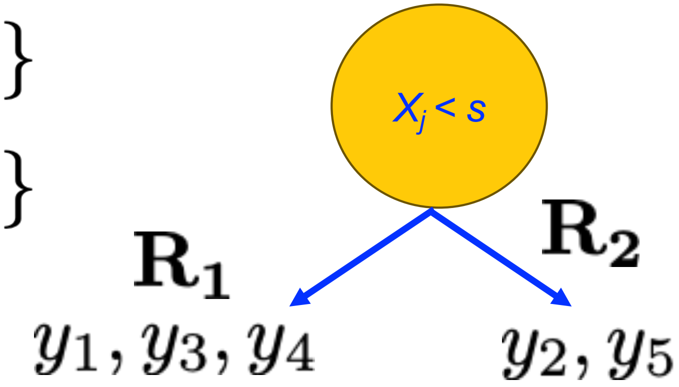
$$\hat{y}_{R_j} = \frac{\sum_{i \in R_j} y_i}{N_{R_j}}$$

Partitioning the Feature Space

- Top-down greedy approach known as *recursive binary splitting:*
  - Begins at the top of the tree and successively splits the feature space
  - Greedy because the split at a particular step minimizes the RSS at that step rather than splitting in such a way to achieve a better tree in a future step
- Consider a split over a feature *j* and the corresponding threshold value *s*, which divides the data such that

$$R_1(j, s) = \{X | X_j < s\}$$

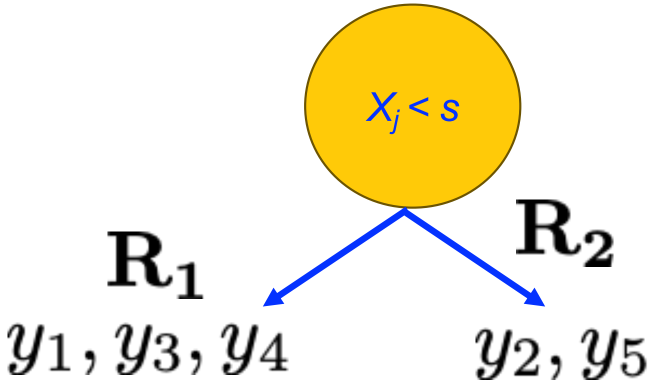
$$R_2(j, s) = \{X | X_j \geq s\}$$



Selection of feature and its threshold

- Minimize the RSS

$$\sum_{i\in R_1(j,s)} (\underset{\underset{i=1,3,4}{\uparrow}}{y_i} - \underset{\underset{\frac{y_1+y_3+y_4}{3}}{\uparrow}}{\hat{y}_{R_1}})^2 + \sum_{i\in R_2(j,s)} (\underset{\underset{i=2,5}{\uparrow}}{y_i} - \underset{\underset{\frac{y_2+y_5}{2}}{\uparrow}}{\hat{y}_{R_2}})^2$$



(Dis)Advantages of Decision-Tree Models

- Advantages
  - Ease of interpretation
  - Graphical representation and understanding by a non-expert
  - Scaling of features is not required

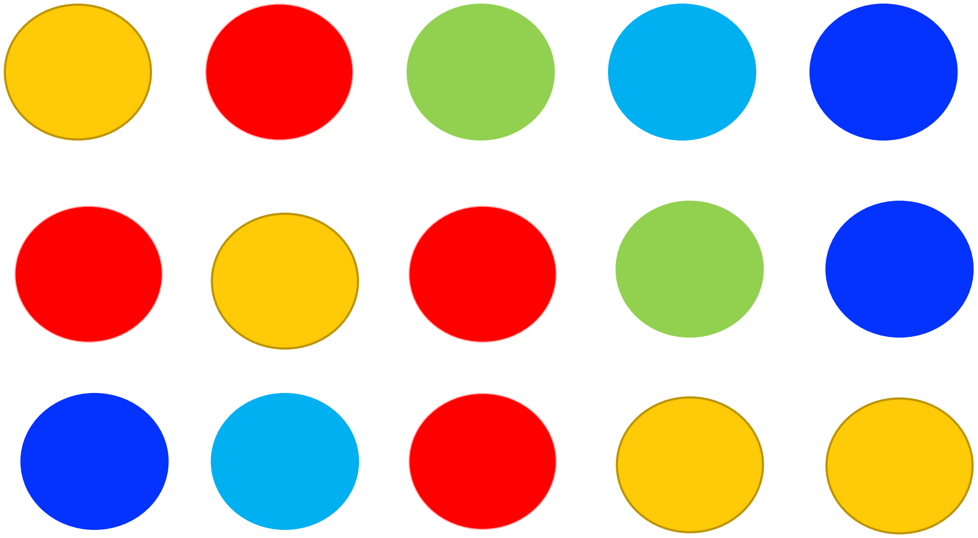
- Disadvantages
  - Accuracy is usually lower than other regression-based approaches
  - Small changes in the data can greatly impact the tree structure
  - As outputs are only piecewise continuous, multiple inputs can yield identical results.

Overcoming disadvantages

- Bagging (bootstrap + aggregating)
  - Using multiple decision-tree models
- Random forests

Bootstrap sampling

- Using the same data set, create multiple data sets by randomly drawing samples with replacement



Original Data

Bootstrapped Sample

Bootstrapped Sample

Aggregating

- For each of the bootstrapped data set *i*, develop a decision-tree model and predict a response $f_i(x)$

- Average each of the responses to obtain the response due to bagging.

$$f_{\mathrm{bag}}(x) = \frac{\sum_i f_i(x)}{B}$$

Random Forests

- Multiple decision-tree models
- Bootstrapped data set
- Randomly selected subset of features at every split
- Achieves decorrelation of trees
- Hyperparameters:
  - Number of trees
  - Number of features to select at every split
  - Minimum number of samples required at an internal node
  - Minimum number of samples required at a leaf node

Hands-on Exercise

- RandomForest.ipynb
- Dataset: Surface tension of deep eutectic solvents

Gradient Boosting

- Borrow concept from RF but build trees sequentially
- Idea is to fit to residuals from the previous prediction
- Consider the following dataset

$$\{x_1, y_1\}, \{x_1, y_1\}, \{x_1, y_1\}, \ldots, \{x_n, y_n\}$$

- In the first step, response for each step is predicted to be the average response

$$y_i^0 = \frac{\sum y_i}{N}$$

- Residual for each of the data point is computed as

$$r_i^0 = y_i - y_i^0$$

Gradient Boosting

- A decision-tree is obtained for the residuals, which provides an estimate of the residual for the *i*th datapoint, say $\hat{r}_i^1$

- New prediction = old prediction + learning parameter * residual prediction

$$\hat{y}_i^1 = y_i^0 + \nu * \hat{r}_i^1$$

- New residual = Output – New prediction $r_i^1 = y_i - \hat{y}_i^1$

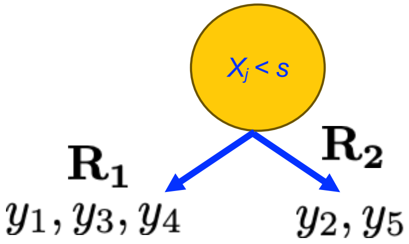
- Fit a decision-tree to r_i^1 and update predictions
- As one might imagine, the number of trees becomes a hyperparameter

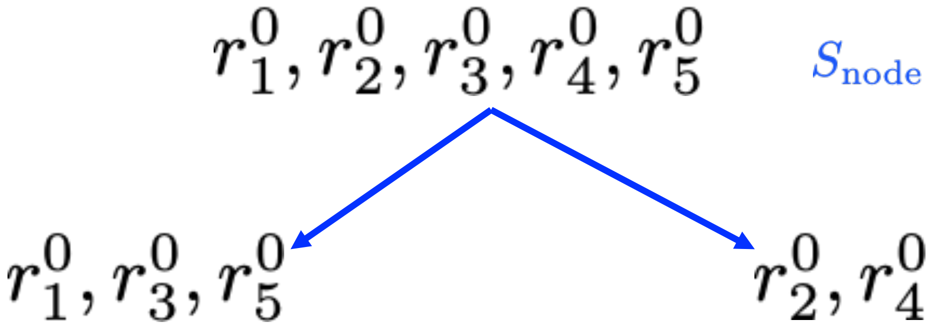
Extreme Gradient Boosting

- Very similar to the gradient boosting but the split is based on similarity score and gain
- As before, compute residuals: $r_i^0 = y_i - y_i^0$
- Compute similarity score

$$\frac{\sum_i^{N_r} r_i^0}{\text{No. of residuals} + \underset{\text{Regularization parameter}}{\lambda}}$$

Splitting a Node in XGBoost





$$S_{\text{node}} = \frac{\left(r^0_1 + r^0_2 + r^0_3 + r^0_4 + r^0_5\right)^2}{5 + \lambda}$$

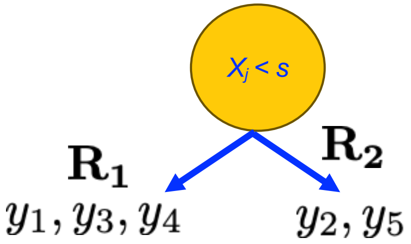
$$S_{\text{left}} = \frac{\left(r^0_1 + r^0_3 + r^0_5\right)^2}{3 + \lambda}$$

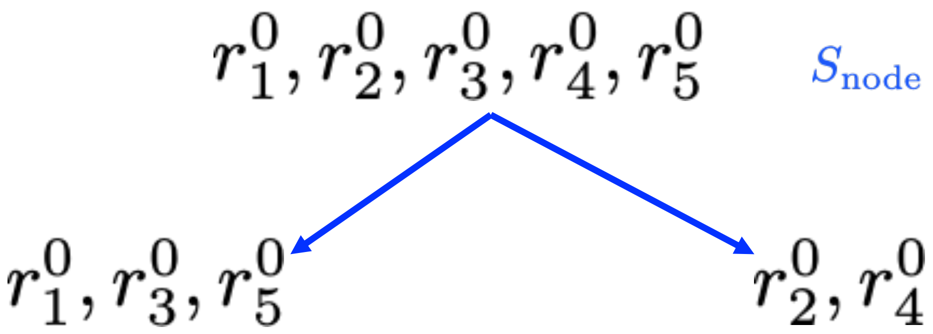
$$S_{\text{right}} = \frac{\left(r^0_2 + r^0_4\right)^2}{2 + \lambda}$$

$$\text{Gain} = S_{\text{left}} + S_{\text{right}} - S_{\text{node}}$$

Step through different values of the threshold and features; select the pair that maximizes Gain.

Output of a Leaf





$$r_1^0, r_2^0, r_3^0, r_4^0, r_5^0$$

$$r_1^0, r_3^0, r_5^0 \qquad r_2^0, r_4^0$$

$$S_{\text{node}} = \frac{\left(r_1^0 + r_2^0 + r_3^0 + r_4^0 + r_5^0\right)^2}{5+\lambda}$$

$$\text{Output}_{\text{left}} = \frac{\left(r_1^0 + r_3^0 + r_5^0\right)}{3+\lambda} \qquad \text{Output}_{\text{right}} = \frac{\left(r_2^0 + r_4^0\right)}{2+\lambda}$$

for $\lambda = 0$**, output is average of the residuals**

New predictions are obtained in a similar manner as that for the gradient boosting method – Slide 16

Thank you!