Week4 Learning Reflection

Name: Xinyu Chang
Time: 2022.04.18-24
Class: CSE 416 Intro. Machine Learning
Lectures: 7 and 8

Summary

1. Decision tree is interpretable, having low bias, and easy to train. However, it has high variance and prones to overfitting. But we can solve it by using Random Forest.
2. Since the decision tree gets larger, with more nodes and branches, the classfication will eventually closer to zero -> prones to overfitting.
3. For numerical data, threshold split would be a good choice. But, the same feature can be used multiple times -> computationally expensive.
4. The depth of decision trees matter. If too small, then it's weak.(high bias) If too big, then prone to overfitting. (high variance)

Random Forest

Bootstrapping: Randomly create many new datasets from original
Aggregation: Train a classifier on each new dataset, Majority voting
Bagging: Bootstrap Aggregation - a technique to reduce variance at expense of computational time

Concepts

(Lecture 7)
Naive Bayes: Select the class that is the most likely (highest probability).

- **Bayes Rule:** $P(y =+ 1|x) = \frac{P(x|y=+1)P(y=+1)}{P(x)}$
- **Naive Bayes Assumption:** Words are independent from each other.
- **Naive Bayes Model:** $P(y|x_1, x_2,..., x_d) = \prod_{j=1}^{d} P(x_j|y)P(y)$

Generative Models: defines a distribution for generating x (e.g. Naive Bayes).

Discriminative Models: only cares about defining an doptimizing a decision boundary (e.g. Logistic Regression).

XOR: Exclusive or. A line migt not always support our decisions.

Decision Trees

- **Branch/Internal node:** splits into possible values of a feature.
- **Leaf Node:** final decision (the class value).
- **Majority Prediction**

Selecting Best split: select the split with lowest classification error.

- Classification Error = $\frac{\#\ mistakes}{\#\ datapoints}$

Recursive Build Tree Algorithm: Greedy

- BuildTree(node)
    - If the number of datapoints at the current node or the classification error is within a certain threshold
        - stop
    - Else
        - Split(node)
        - For child in node:
            - BuildTree(child)

Trees having classification error = 0 -> overfit.

Decision Stump: Tree with 1 internal node.

Selecting Real-valued Features

- Step 1: Sort the values of a feature $h_j(x)$:

    Let $\{v_1, v_2, ..., v_n\}$ denote sorted values with n datapoints
- Step 2:

    For i=1….N-1

        Consider split $t_i = (v_i + v_{i+1})/2$

        Compute classification error

    Chose the t* with the lowest classification error

Tree Decision Boundaries: Threshold boundaries for real-valued features.
Overfitting and Trees

Solution:

1. Set min. # data points in a node to split
2. Early stopping:

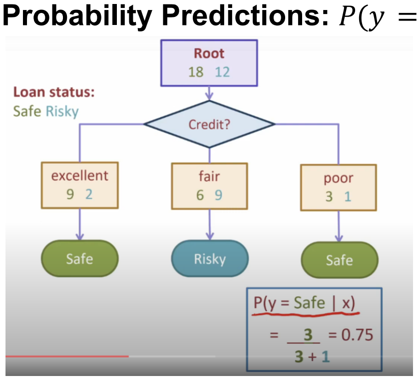
    Fixed length depth

    Maximum number of nodes

    Stop if error does not considerably decrease
3. Pruning
4. Fine-tune hyperparameters using a validation set

(Lecture 8)
Decision Trees

- **Probability Predictions:** $P(y = Safe \mid x)$



- **Overfitting:** Increasing the height of your tree will make the prediction more complex.
- **Early Stopping:** stop if each node reaches zero classification error/ no features to split
    - only grow up to a max depth hyperparameter (choose via validation)
    - Don't split if there is not a sufficient decrease in error
    - Require a minimum number of examples in a leaf node
- **Pruning**

Ensembles

- **Ensemble Model:** a collection of (generally weak) models that are combined in such a way to create a more powerful model
    - Random Forest (Bagging)
    - AdaBoost (Boosting)
- **Bootstrap:** Create many similar datasets by randomly sampling with replacement.
- **Random Forest:** A specific type of ensemble model uses bagging.
  versatile: works pretty well in a lot of cases and can serve many different purposes (classification, regression, clustering, feature importance)
  low maintenance: tends to require less hyper-parameter tuning.
  efficient: trees can be learned in parallel.
    - **How to train:** Random Sampling. Use bootstrapping. Also randomly select features, too.
        - Make T random samples of the training data that are the same size as the training data but are sampled with replacement
        - Train a really tall tree on each sampled dataset (overfit.)
    - **How to predict**
        - For a given example, ask each tree to predict what it thinks the label should be.
        - Take a majority vote over all trees.
- **Bagging:** Bootstrapped aggregation.

AdaBoost - Sequential

- **Weak Learner:** stump. A model that only does slightly better than random guessing.
- **AdaBoost model:** A model similar to Random Forest with two notable differences that impact how to train it quite severely.
  Decision stumps instead of high-depth trees.
  Each model in the ensemble gets a weight associated with it, and we take a weighted majority vote.
    - **Model coefficients:** $\hat{w}_t$. use as the end result to weight each model.
        - $\hat{w}_t = \frac{1}{2} ln(\frac{1-WeightedError(\hat{f}_t)}{WeightedError(\hat{f}_t)})$
    - **Data coefficients:** α_i. for each example in the dataset, update each time we train a new model.
        - $$\alpha_i \leftarrow \begin{cases} \alpha_i e^{-\hat{w}_t}, & if\ \hat{f}_t(x_i) = y_i \\ \alpha_i e^{\hat{w}_t}, & if\ \hat{f}_t(x_i) \neq y_i \end{cases}$$
    - **Weighted classification error:**
        - We want to minimize weighted classification error
    - **How to train:**
      For t in [1,...,T]:
      learn $\hat{f}_t(x)$ based on weights α_i

compute model weight $\hat{w}_t$

recompute weights α_i, assign same weight for each datapoint.

normalize α_i

- **How to predict:**

$$\hat{y} = \hat{F}(x) = sign(\sum_{t=1}^{T} \hat{w}_t \hat{f}_t(x))$$

- **AdaBoost Theorem:** As training error of boosted classifier goes to zero, T goes to infinity. The weak learner can do at least slightly better than complete random guessing.

Comparing AdaBoost and Random Forests

- AdaBoost: Powerful, High Maintennace, Expensive.

Uncertainties

1. So, the decision tree is based on Naive Bayes?