Lecture 10 Complementary: More about Decision Trees

1. Overfitting in decision trees
   - Two approaches to pick simpler trees: ① early stopping ② pruning
     1) Early Stopping (stop the learning algorithm before tree becomes too complex)
        - Stopping conditions:
          ① All samples have the same target value
          ② No more features to split on
        - Early Stopping conditions: (hard to know exactly when to stop)
          ① Limit tree depth (choose max_depth using validation set)
          ② Do not consider splits that do not cause a sufficient decrease in classification error
          ③ Do not split an intermediate node which contains too few data points

Pros and Cons:

⊕ A reasonable heuristic for early stopping to avoid useless splits

⊖ Too short-sighted: We may miss out on "good" splits may occur right after "useless" splits.

2) Pruning (Simplify tree after the learning algorithm terminates)

Balance: measure of fit & measure of complexity

$L(T)$ = # of leaf nodes (the final groups)

Total Cost $C(T) = \text{Error}(T) + \lambda L(T)$

- Tree Pruning Algorithm:

  Repeat {

    Consider a split and compute $C(T)$ of the split;

    Undo the splits on Tsmaller, prune if total cost is lower:

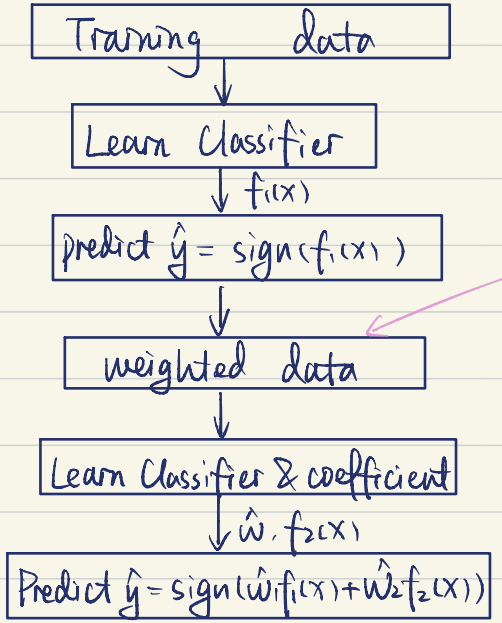
    $C(T_{smaller}) \leq C(T)$

  }

2. Boosting: Focus learning on "hard" points

⇒ put more weight on "hard"/misclassified or more important points

= Greedy learning ensembles from data



3. AdaBoost algorithm

- Start with same weight for all points: $\alpha_i = \frac{1}{N}$
- For $t = 1 \cdots T$ (T weak classifiers - single layer DTs) → weighted # of misclassified points
  - learn $f_t(x)$ with data weights α_i
  - compute coefficient $\hat{w}_t$: $\hat{w}_t = \frac{1}{2}\ln\left(\frac{1 - \text{weighted\_error}(f_t)}{\text{weighted\_error}(f_t)}\right)$
  - recompute weights α_i: $\alpha_i = \begin{cases} \alpha_i e^{-\hat{w}_t}, & \text{if } f_t(x_i) = y_i \\ \alpha_i e^{\hat{w}_t}, & \text{if } f_t(x_i) \neq y_i \end{cases}$
  - Normalize weights α_i: $\alpha_i = \alpha_i / \sum_{j=1}^{N} \alpha_j$
- Final model predicts by: $\hat{y} = \text{sign}\left(\sum_{t=1}^{T} \hat{w}_t f_t(x)\right)$

4. Boosting convergence & overfitting

Boosting tend to be robust to overfitting, but eventually will overfit.

⇒ must choose max number of components T