Lecture 16 - 29/10/24

Recall:

- Bagging
- Random Forest
- Boosting
- Stacking

→ One group of ensemble mainly aims to reduce variance!

Another one aims to reduce bias!

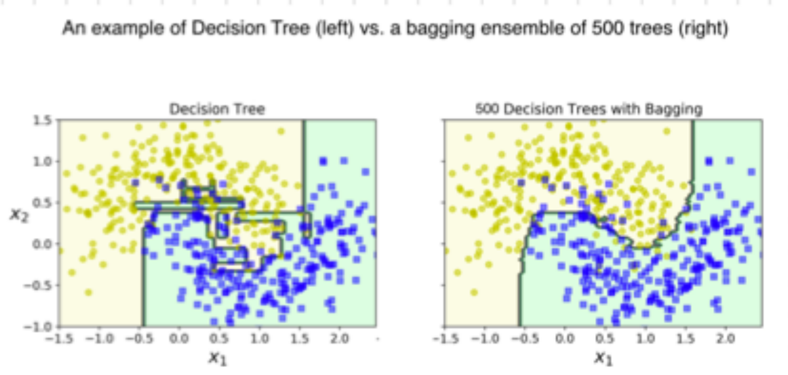
Bagging

- meaning → Bootstrap Aggregation
- Training:
  - Make bootstrapped subsets of the training data
  - Train a separate model on each subset.
  - Given that samples come from same dataset, not truly indip.

Note: Creating diversity in the data, introduces diversity in the models

- Prediction:
  - combination of predictions of all models
- Final Step:
  - typically use hard voting to combine predictions for classifications.
  - typically use averaging for regression.
- Scenario:
  - I have a classifier which has high variance prob.
  - It has low bias.
  - I have computational power available

try Bagging!!!



We can see how we provide a great way to avoid overfitting

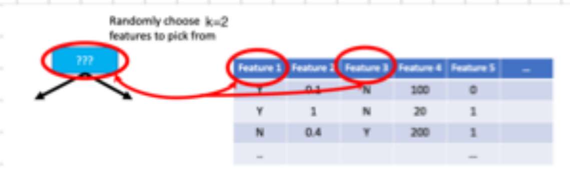
- Validation:
  - Out-of-Bag evaluation
  - We know that in bagging we use bootstrapped data → Sampling with Replacement
    - Not all datapoints will be in a given bag!
    - ≠ datapoint are incl./exclud. in ≠ bags.
  - On avg 37% are out-of-bag samples if bags are the same size as dataset
  - Out of Bag samples will be used for validation!
  - ∀ data point in original set we combine pred where it wasn't sampled

- Validation error is the avg error across these combined preds.
- The OOB prediction error is an unbiased estimation of the test error.
- This method can replace cross-validation.

- Advantages:
  - Easy to implement
  - Training in parallel
  - Pred is avg or vote
  - Reduces variance
  - Out-of-bag validation

Random Forest

- Essentially a bagging decision tree with modified splitting criterion
- Process:
  - Make bootstrapped data sets from training set
  - $\forall$ dataset, train a full decision tree with modified split strategy:
    - Before finding each split:
      - * Randomly choose k features
      - * Only consider those feature or split
  - Output of random forest is the aggregated output of all trees



m=2

Random Forest are nice because:

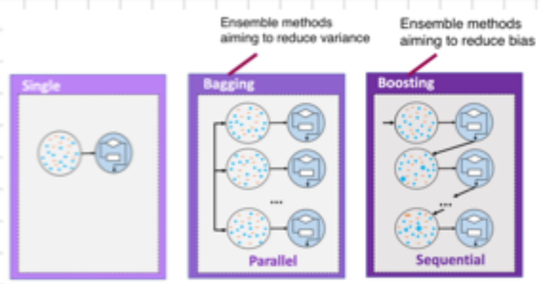
- Only few hyper-parameters:
  - m = # of bootstrapped datasets.
  - k = # of randomly selected features.
- Based on decision trees which avoid hard process.
- Average depth estimates feature importance!

Extra Trees

- Meaning → Extremely Randomized Trees
- Unlike bagging & random forest, they fit each D.T. on whole dataset. ~~Bootstrap~~
- Unlike random forest, their algo picks a random cut-point at each of the k features.
- Like random forest, their algo randomly samples k features to find the good split.

* We know that Bagging & associated methods → reduce variance
* What about reducing bias though?

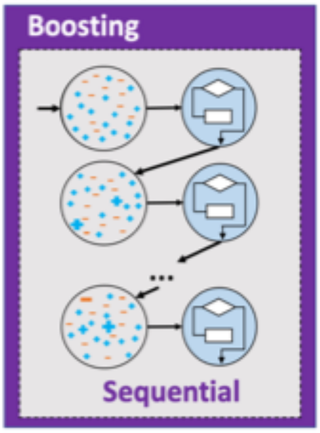
Use Boosting!!!

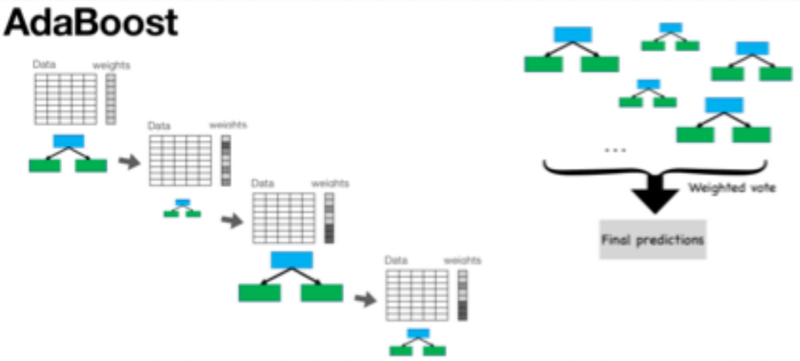


Boosting

A bit better than random guess

- Sequence of weak learners combined into a strong learner
- Steps:
  - Train a weak learner
  - Get training error
  - Train next weak learner based on that error
  - ⋮
  - Final predictions: a combination of all trained weak learners.
- Various Boosting methods exist → AdaBoost, GradientBoosting...





After making each tree, the errors:

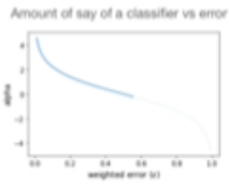
- influence weights of next training.
- determine weight of tree in final.

AdaBoost
Training

Start with equal weights for all the training samples, e.g., $w_i^{(1)} = \frac{1}{N}$

For $c = 1, \ldots, M$ iterations (trees):

- Train a model $h_c(\mathrm{x})$ based on the weighted training samples
- Calculate the total error on the training set: $\epsilon_c = \sum_{i, h_c(x_i) \neq y_i} w_i^{(c)}$
- Based on the total error, calculate the model's amount of say: $\alpha_c = ln(\frac{1-\epsilon_c}{\epsilon_c})$
- Update the samples' weights:
  - Increase for misclassified samples: $w_i^{(c')} = w_i^{(c)} e^{\alpha_c}$
  - Unchanged for correct classifications: $w_i^{(c')} = w_i^{(c)}$
- Normalize all weights so they sum up to 1: $w_i^{(c+1)} = \frac{w_i^{(c')}}{\sum_{j=1}^{n} w_j^{(c')}}$



Some implementations add a hyper-parameter (learning rate) here to avoid overfitting

Some implementations also decrease the weights of correctly classified samples and then normalize

Two types of weights:

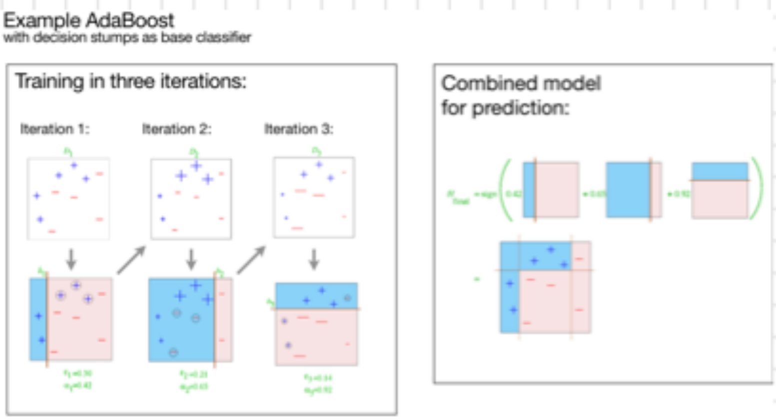
- w_i = training sample
- α_c = ∀ model

Prediction:

$$\text{sign}\left(\sum_{c=1}^{M} \alpha_c h_c(x)\right)$$

+1 or -1

Weak learners = stumps



We can incorporate the weights into the assessment of splits.

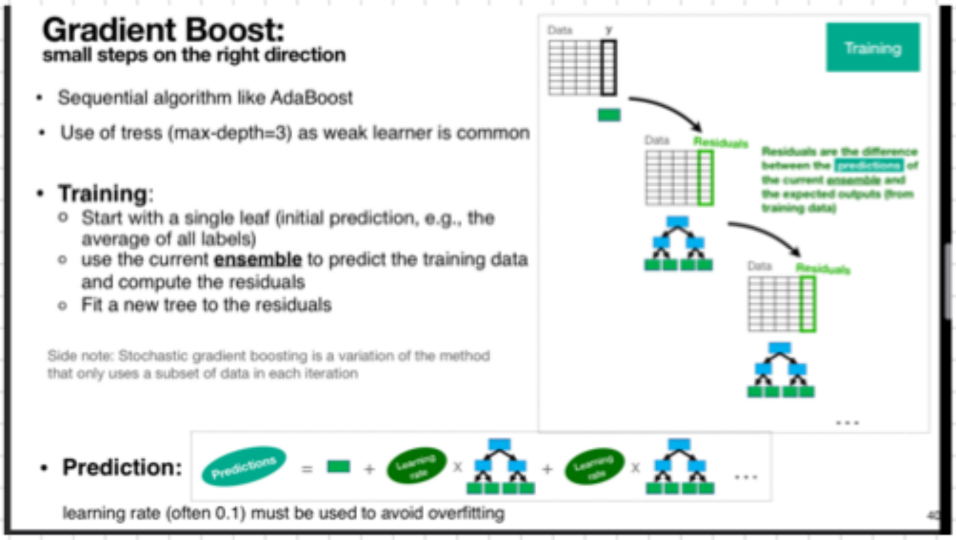
A datapoint is no more worth 1 unit, it counts as its weight w_i in impurity calculation.

↳ their weighted avg = $P_L \phi(I_L) + P_R \phi(I_R)$

$$p_k = \sum_{i, y_i = k} w_i$$

Gradient Boosting

- Commonly used for both regression & classification.



Bagging vs Boosting

Bagging	Boosting
Usually helps decreasing variance	Helps decreasing bias
Not much effect with stable models	Doesn't solve overfitting
Easy to parallelize	Sometimes hurts performance
	Cannot be fully parallelized

Bottom line: It depends!!

Stacking

- Instead of voting for aggr. → train meta-learner.

one of our choice

Should get preds of individual models as input & expected labels for data samples as output.