Boosting

Ryan Miller

Grinnell College
Statistics

Introduction

- Random Forests use *bagging* to build an ensemble of decision tree models
  - Many trees trained on slightly different data contribute to the model's predictions
  - This is an *aggregation* approach, as each base model is *trained separately* and the predictions are aggregated

Introduction

- Random Forests use *bagging* to build an ensemble of decision tree models
  - Many trees trained on slightly different data contribute to the model's predictions
  - This is an *aggregation* approach, as each base model is *trained separately* and the predictions are aggregated
- Today we will discuss *boosting* approaches, where the base models in the ensemble are *trained sequentially*
  - Boosting was originally developed as a classifier aimed at combining many "weak" classifiers into a more powerful "committee" by Freund and Schapire (1997) as "Adaptive Boosting" or "AdaBoost.M1"

Bagging vs. Boosting

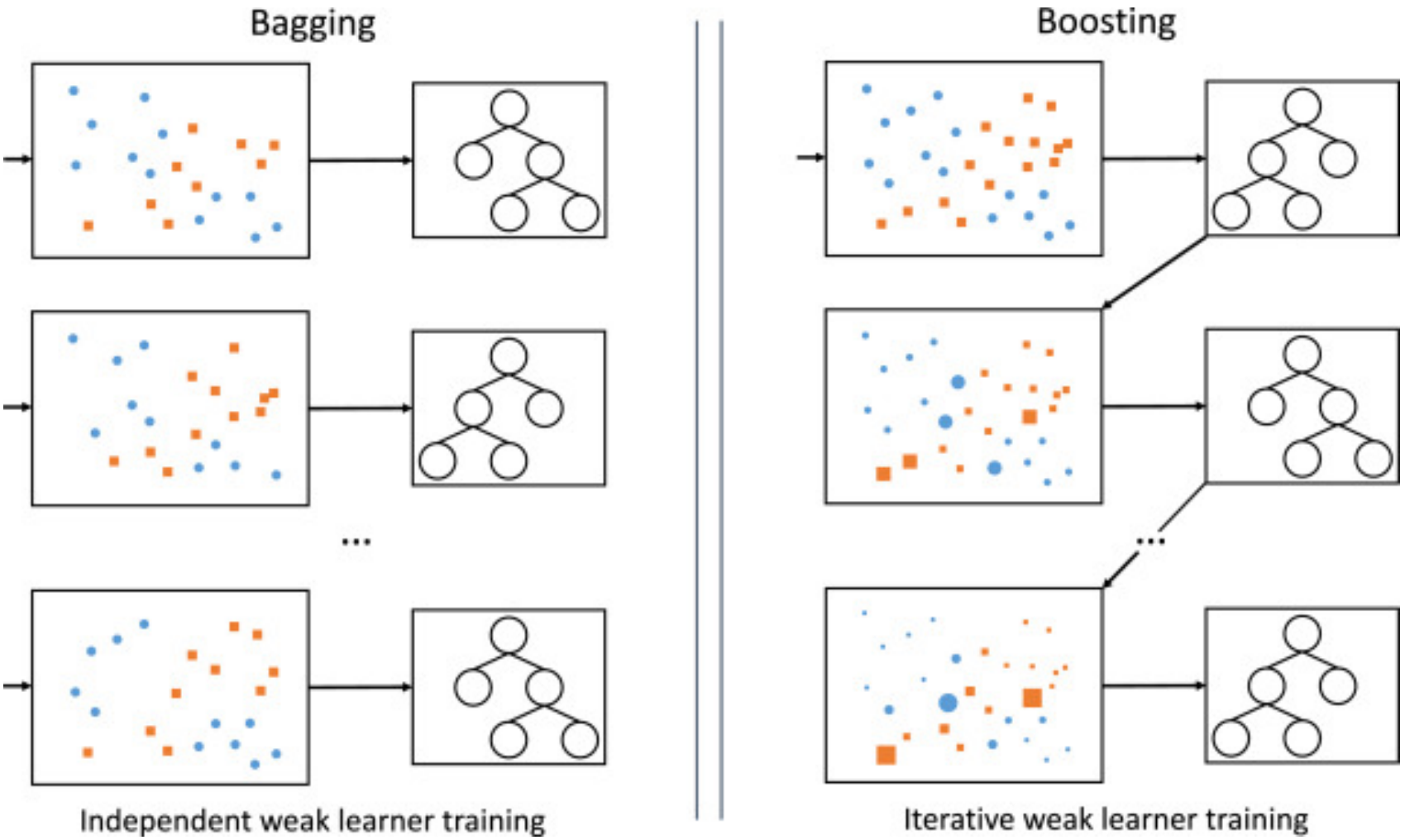

Image Credit: https://www.sciencedirect.com/science/article/pii/S1566253520303195

AdaBoost

To begin, suppose Y is a binary variable encoded by $\{-1, 1\}$, and define the error rate as:

$$\text{error} = \frac{1}{n} \sum_{i=1}^{n} I(y_i \neq G(\mathbf{x}_i))$$

- Here, $G(\mathbf{x}_i)$ represents the predicted class for an observation with predictors $\mathbf{x}_i$
  - Notice that if y_i does not match the predicted class the summation increases by 1
  - Thus, we see that error $= 1 -$ classification accuracy

AdaBoost

In AdaBoost, $G()$ is a sum of M sequentially built classifiers trained on *differently weighted versions* of the data:

$$G(x) = \text{Sign}\left(\sum_{m=1}^{M} \alpha_m G_m(x) \right)$$

- $\alpha_1, \ldots, \alpha_M$ let each base learner to contribute differently to the prediction (some models to contribute more)
  - Typically, $\alpha_m = log((1 - \text{error}_m)/\text{error}_m)$
- Each training data-point, $(\mathbf{x}_i, y_i)$ is also given a different weight, w_{im}, at each iteration
  - At the first step of the algorithm, these weights are set to $\frac{1}{n}$, so that all observations contribute equally
  - At step m, the weights of observations misclassified by $G_{m-1}(x)$ are increased by a factor of $exp(\alpha_{m-1})$

AdaBoost (algorithm)

Pseudocode for the original AdaBoost algorithm:

```
w = 1/n  # initialize weights
for i in 1:M:
  G_m = G_m + model.fit(X, w, y) # next fit
  err_m = error(G_m)         # calculate error
  alpha_m = log((1-err_m)/err_m)
  w_i = w_i*exp(a_m*(y_i != G_m(x_i)))  # reweight
```

- Notice that $log((1-0.5)/0.5) = 0$, so a model that is randomly guessing won't contribute to the ensemble
- If an observation was misclassified, its weight in the next model is increased by a factor of $exp(\alpha_m)$

AdaBoost (diagram)

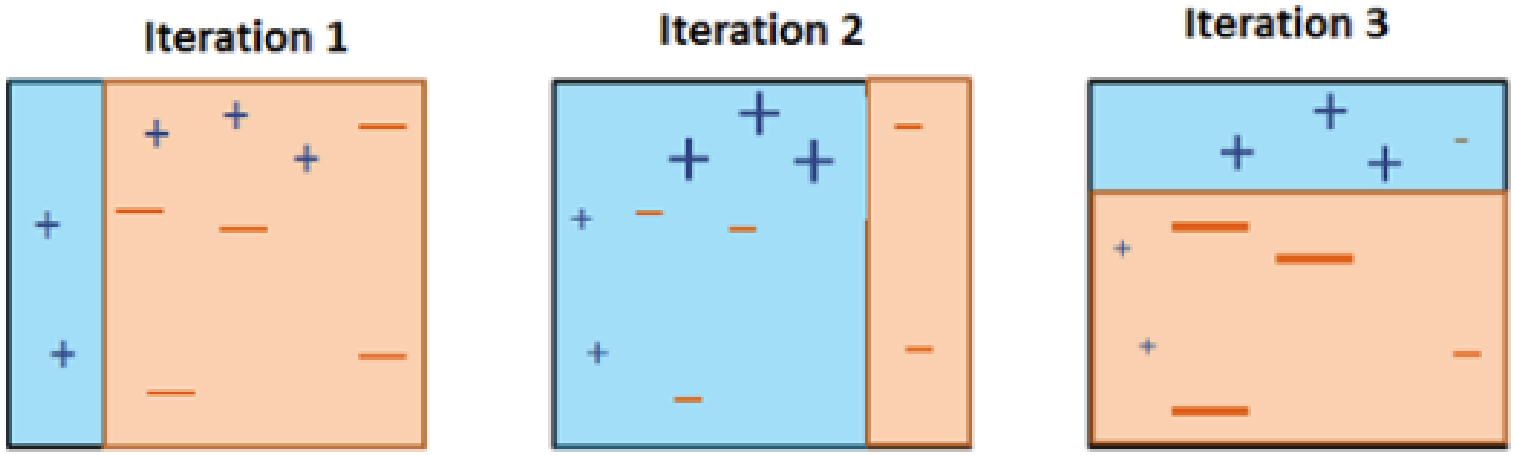

Image Credit: Packt Big data and Business Intelligence

Gradient Boosting

- AdaBoost re-weights observations that are misclassified before training the next model
  - Gradient boosting takes a more general approach: train subsequent models to the *residuals* (or *pseudoresiduals* for cost functions other than squared error)
- At each iteration, t, of gradient boosting, the algorithm finds a base model (estimates $\hat{f}_t$):

$$\hat{f}_t(\mathbf{X}) = \arg\min_{f_t}\left(L(y, \hat{y}_{t-1} + \eta f_t(\mathbf{X}))\right)$$

- For the squared error cost function, this is amounts to fitting the base model to the residuals (we'll see more on this in a future homework assignment)

Gradient Boosting (pseudocode)

```
r = y ## Initialize residuals
f = 0 ## Initialize model to 'zero'

for i in 1:M:
    f_m = model.fit(X, r) ## Fit model to residuals
    f = f + eta*f_m  ## Add new model to ensemble
    r = r - eta*f_m  ## Update residuals
```

- The output is f, the ensemble model consisting of M different base models.
- The learning rate, η (eta), is a small positive number that controls how quickly boosting learns (by limiting how much a model can contribute to the ensemble)

Gradient Boosting vs AdaBoost (diagram)

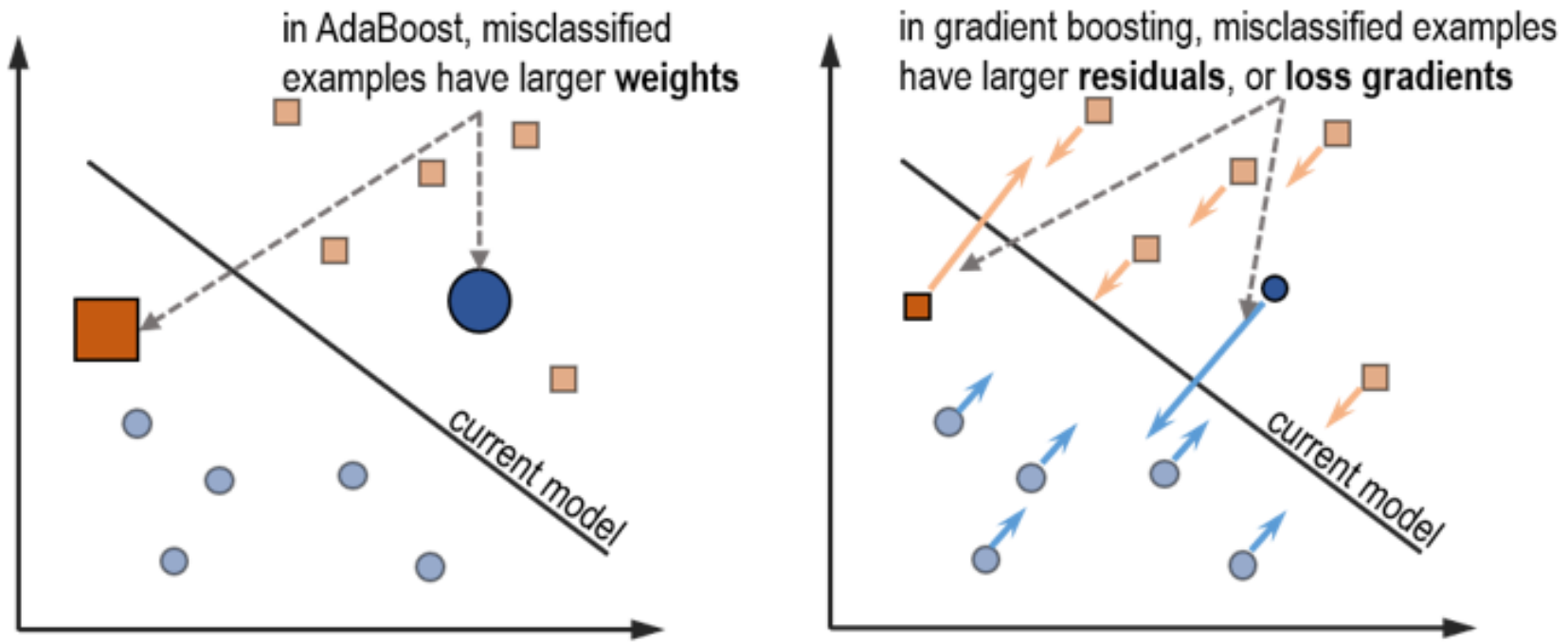

Image credit: Ensemble Methods for Machine Learning (textbook)

Comments on Gradient Boosting

- Like bagging, gradient boosting is a general way to build an ensemble model that doesn't necessarily need to use decision trees as base learners
  - That said, the `xgboost` (extreme gradient boosting) library is the most widely used boosting implementation by a large margin and it uses decision trees as its default
- Similar to bagging, there are hyperparameters of a boosting ensemble at two levels:
  - The level of the base learners, such as maximum depth, minimum samples to split etc.
  - The ensemble level, such as the learning rate, number of boosting iterations, feature subsampling, etc.

What to Know for the Next Quiz

1. Differences between boosting and bagging ensembles:
   - Base learners are trained independently in bagging, but sequentially in boosting
   - More base learners leads to overfitting in boosting, but not in bagging
   - Base learners are typically treated equally in bagging, but differentially in boosting
2. Basic concepts of gradient boosting:
   - New models are fit to the residuals (or pseudoresidual), thereby correcting the errors made previously
   - The learning rate controls how much each new base model contributes to the ensemble
   - The learning rate and number of boosting iterations work together to influence the bias-variance trade-off