Boosting

Ron
credit: Flora

Outline

Objectives

1. Differentiate boosting from bagging
2. Name the tuning parameters for boosting (e.g., gradient boosting)

What is boosting?

- **Bagging** is a simple ensemble technique involves creating multiple copies of the original training data set using bootstrap, fitting a separate decision tree to each copy, and then combining all of the trees in order to create a single predictive model

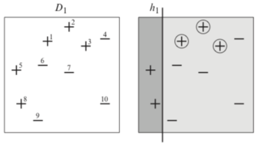
- **Boosting** is an ensemble technique in which the predictors are not made independently, but *sequentially*: each tree is grown using information from previously grown trees

AdaBoost (Adaptive boosting)

- AdaBoost retrains the algorithm iteratively by choosing the training set based on accuracy of previous training.
  - After training a classifier at any level, AdaBoost assigns weight to each training item. Misclassified item is assigned higher weight so that it appears in the training subset of next classifier with higher probability.
- The weight of each trained classifier at any iteration depends on the accuracy achieved.
  - After each classifier is trained, the weight is assigned to the classifier as well based on accuracy. More accurate classifier is assigned higher weight so that it will have more impact in final outcome.

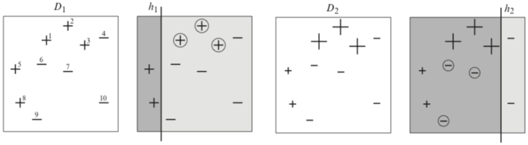
Example: Ensemble of 3 Decision Stumps

Source: [Schapire and Freund, 2012]



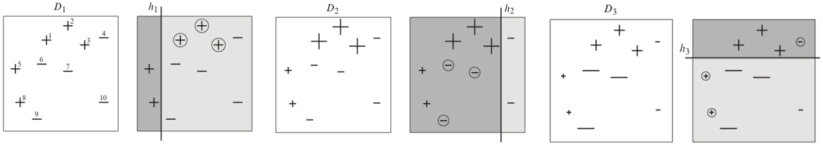
Example: Ensemble of 3 Decision Stumps

Source: [Schapire and Freund, 2012]



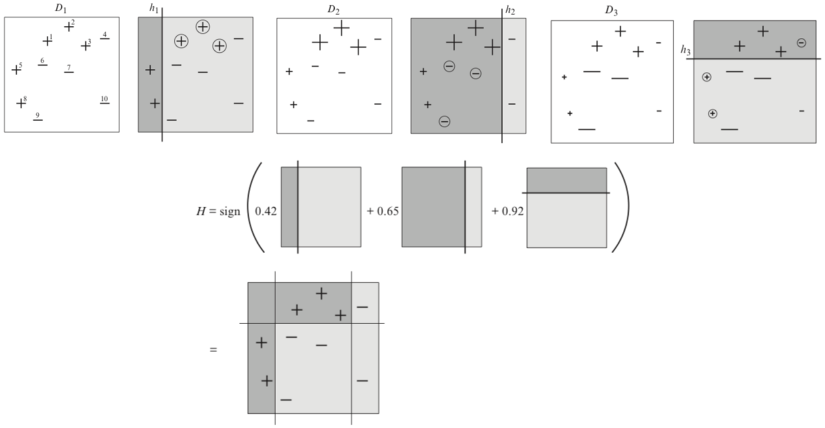
Example: Ensemble of 3 Decision Stumps

Source: [Schapire and Freund, 2012]



Example: Ensemble of 3 Decision Stumps

Source: [Schapire and Freund, 2012]



Algorithm

Given $(X_1, y_1), \ldots, (X_n, y_n)$ where $X_i \in \mathbb{R}^p$ and $y_i \in \{-1, +1\}$

1. Set weights $D_1 = 1/n$ all for i in the training set
2. For $t = 1, 2, \ldots, T$, repeat
    2.1 Fit weak classifier $h_t : X \to \{-1, +1\}$ with lowest weighted error ϵ_t using D_t

    $$\epsilon_t = P_{i \sim D_t}[h_t(x_i) \neq y_i] = \sum_{i:h_t(x_i) \neq y_i} D_t(i)$$

    2.2 Set $\alpha_t = \dfrac{1}{2} \ln \dfrac{1 - \epsilon_t}{\epsilon_t}$

    2.3 Update

    $$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

    where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution)
3. Output the final classifier

$$H = \text{sign} \sum_{t=1}^{T} \alpha_t h_t(x)$$

Gradient Boosting

- Gradient Boosting doesn't modify the sample distribution
- Instead of on a new sample distribution, the weak learner trains on the remaining errors of the strong learner
- The contribution of the weak leaner to the strong one is computed using a gradient descent optimization process

https://bit.ly/35mred9

Algorithm for regression trees

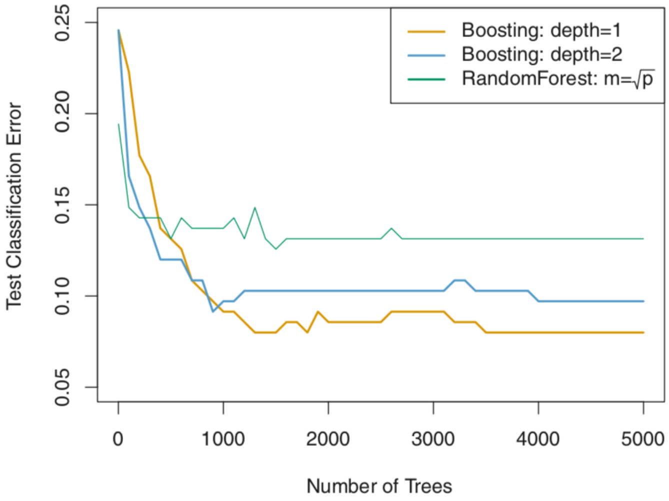
1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set
2. For $b = 1, 2, \ldots, B$, repeat
   - 2.1 Fit a tree $\hat{f}^b$ with d splits ($d+1$ terminal nodes) to the training data (X, r_i)
   - 2.2 Update $\hat{f}$ by adding in a shrunken version of the new tree
   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$$
   - 2.3 Update the residuals
   $$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i)$$
3. Output the boosted model
$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$$

Rationale

- Unlike fitting a single large decision tree to the data, the boosting approach instead learns slowly.
- Given the current model, we fit a decision tree to the residuals from the model. We then add this new decision tree into the fitted function in order to update the residuals.
- Each of these trees can be rather small, with just a few terminal nodes, determined by the parameter d in the algorithm
- By fitting small trees to the residuals, we slowly improve $\hat{f}$ in areas where it does not perform well. The shrinkage parameter λ slows the process down even further, allowing more and different shaped trees to attack the residuals.

Example: gene expression data

predict cancer vs normal using expression measurements of 500 genes from 349 patients



Tuning parameters for boosting

Tuning parameters for boosting

- The number of trees B. Unlike bagging and random forests, boosting can overfit if B is too large, although this overfitting tends to occur slowly if at all. We use cross-validation to select B.

Tuning parameters for boosting

- The number of trees B. Unlike bagging and random forests, boosting can overfit if B is too large, although this overfitting tends to occur slowly if at all. We use cross-validation to select B.
- The shrinkage parameter λ, a small positive number. This controls the rate at which boosting learns. Typical values are 0.01 or 0.001, and the right choice can depend on the problem. Very small λ can require using a very large value of B in order to achieve good performance.

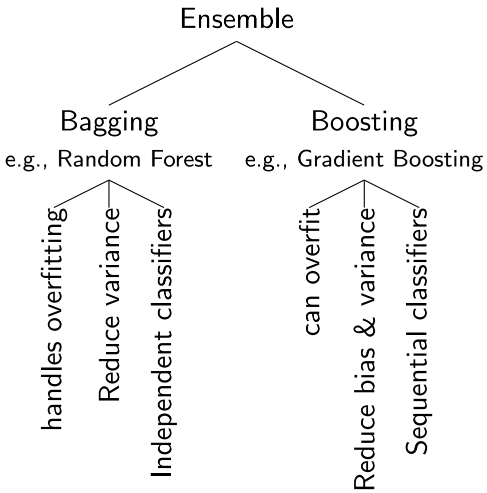
Tuning parameters for boosting

- The number of trees B. Unlike bagging and random forests, boosting can overfit if B is too large, although this overfitting tends to occur slowly if at all. We use cross-validation to select B.
- The shrinkage parameter λ, a small positive number. This controls the rate at which boosting learns. Typical values are 0.01 or 0.001, and the right choice can depend on the problem. Very small λ can require using a very large value of B in order to achieve good performance.
- The number of splits d in each tree, which controls the complexity of the boosted ensemble. Often $d = 1$ works well, in which case each tree is a stump, consisting of a single split and resulting in an additive model. More generally d is the interaction depth, and controls the interaction order of the boosted model, since d splits can involve at most d variables.

AdaBoost vs Gradient Boosting

AdaBoost	**Gradient Boosting**
Use a base weak learner and they try to boost the performance of a weak learner by iteratively shifting the focus towards problematic observations that were difficult to predict. At the end, a strong learner is formed by combining all the weak learners	
Shortcomings are identified by high-weight data points	Shortcomings are identified by gradients
Exponential loss gives more weighted for those samples fitted worse	Dissect error components to bring in more explanation

Bagging vs Boosting


Ensemble
Bagging
e.g., Random Forest
handles overfitting
Reduce variance
Independent classifiers
Boosting
e.g., Gradient Boosting
can overfit
Reduce bias & variance
Sequential classifiers