Ensemble learning methods
Part II: Boosting

Outline

• Bagging vs. boosting

• Simple boosting trees – the regression case

• Adaptive boosting – classification with AdaBoost
Bagging and boosting — the difference

- **Bagging**: each predictor is trained independently

- **Boosting**: each predictor is built on the top of previous predictors trained
  - Like bagging, boosting is also a voting method. In contrast to bagging, boosting actively tries to generate complementary learners by training the next learner on the mistakes of the previous learners.
Basic idea

- Boosting is a method that produces a very accurate predictor by combining rough and moderately accurate predictors.
- It is based on the observation that finding many rough predictors (rules of thumb) can be easier than finding a single, highly accurate predictor.
- Boosting combines the outputs of many “weak” classifiers (“rules of thumb”) to produce a powerful “committee.”

Motivation

- How to extract rules of thumb that will be the most useful?
- How to combine moderately accurate rules of thumb into a single highly accurate prediction rule?
Simple boosting with regression trees

1. Initialization: Set $h(x) = 0$ and $r_i = y_i$ for all $i = 1, \ldots, n$ in the training set.

2. For $b = 1, \ldots, B$, repeat
   
   (a) Fit a tree $h^b$ with only $d$ splits to the training set $(X, r)$.
   (b) Update $h$ by adding the new tree
       \[
       h(x) \leftarrow h(x) + \lambda h^b(x)
       \]
   (c) Update the residuals
       \[
       r_i \leftarrow r_i - \lambda h^b(x_i)
       \]

3. Output the boosted model
   \[
   h(x) = \sum_{b=1}^{B} \lambda h^b(x)
   \]
Boosting with regression trees – tuning parameters

- The number of trees $B$

- The shrinkage parameter $\lambda$
  - 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 $\lambda$ can require using a very large value of $B$ in order to achieve good performance.

- The number $d$ of splits in each tree
  - Trees with just $d = 1$ split are called “stumps”.
AdaBoost is a boosting method that repeatedly calls a given weak learner, each time with different distribution over the training data. Then we combine these weak learners into a strong learner.

- originally proposed by Freund and Schapire (1996)

- great success
  - “AdaBoost with trees is the best off-the-shelf classifier in the world.” (Breiman 1998)
  - “Boosting is one of the most powerful learning ideas introduced in the last twenty years.” (Hastie et al, 2009)
Key questions

• How to choose the distribution?
• How to combine the weak predictors into a single predictor?
• How many weak predictors should be trained?

Schapire’s strategy: Change the distribution over the examples in each iteration, feed the resulting sample into the weak learner, and then combine the resulting hypotheses into a voting ensemble, which, in the end, would have a boosted prediction accuracy.
AdaBoost.M1 (Freund and Schapire, 1997) is the most popular boosting algorithm

- Consider a binary classification task with the training data

\[ Data = \{ \langle x_i, y_i \rangle : x_i \in X, y_i \in \{-1, +1\}, i = 1, \ldots, n \} \]

- We need to define distribution \( D \) over \( Data \) such that \( \sum_{i=1}^{n} D_i = 1 \).

- Assumption: a weak classifier \( h_t \) has the property

\[ \text{error}_{D}(h_t) < 1/2. \]
Adaboost (Adaptive Boosting) — key idea

Classifiers are trained on weighted versions of the original training data set, and then combined to produce a final prediction.

\[ h(x) = \text{sign} \sum_{t=1}^{M} \alpha_t h_t(x), \]

where \( \alpha_t \) are computed by the boosting algorithm, and weight the contribution of each respective \( h_t \).
AdaBoost – iterative algorithm

- Initialize the training distribution $D_1(i) = 1/n$ for $i = 1, \ldots, n$
- At each step $t$
  - Learn $h_t$ using $D_t$: find the weak classifier $h_t$ with the minimum weighted sample error $\operatorname{error}_{D_t}(h_t) = \sum_{i=1}^{n} D_t(i) \delta(h(x_i) \neq y_i)$
  - Set weight $\alpha_t$ of $h_t$ based on the sample error
    \[
    \alpha_t = \frac{1}{2} \ln \left( \frac{1 - \operatorname{error}_{D_t}(h_t)}{\operatorname{error}_{D_t}(h_t)} \right)
    \]
  - Update the training distribution
    \[
    D_{t+1} = D_t e^{-\alpha_t y_i h_t(x_i)} / Z_t \quad \text{where } Z_t \text{ is a normalization factor}
    \]
  - Stop when impossible to find a weak classifier being better than chance
- Output the final classifier $h_{final}(x) = \text{sign} \sum_{t=1}^{T} \alpha_t h_t(x)$
AdaBoost – training data weighting

**Constructing** $D_t$

- On each round, the weights of incorrectly classified instances are increased so that the algorithm is forced to focus on the hard training examples.

- $D_1(i) = 1/n$ for $i = 1, \ldots, n$

- given $D_t$ and $h_t$ (i.e. update $D_t$):

  $$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \cdot \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases} = \frac{D_t(i)}{Z_t} e^{-\alpha_t y_i h_t(x_i)},$$

  where $Z_t$ is normalization constant $Z_t = \sum_i D_t(i) e^{-\alpha_t y_i h_t(x_i)}$

- $\alpha_t$ measures the importance that is assigned to $h_t$

As the iterations proceed, examples that are difficult to classify correctly receive ever-increasing influence
Weights of the base learners $\alpha_t$

- $\text{error}_{\mathcal{D}_t}(h_t) < \frac{1}{2} \Rightarrow \alpha_t > 0$
- the smaller the error, the bigger the weight of the (weak) base learner
- the bigger the weight, the more impact on the (strong) resulting classifier

$$\text{error}_{\mathcal{D}_t}(h_1) < \text{error}_{\mathcal{D}_t}(h_2) \implies \alpha_1 > \alpha_2$$

- $\mathcal{D}_{t+1} = \frac{1}{Z_t} \mathcal{D}_t \, e^{-\alpha_t y_i h_t(x_i)}$

  The weights of correctly classified instances are reduced while weights of misclassified instances are increased.
Multiclass problem – generalization of the two-class case

- Assume classification task where $Y = \{y_1, \ldots, y_k\}$

  $$h_t : X \rightarrow Y,$$

  $$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \cdot \left\{ \begin{array}{ll}
e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\
e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{array} \right.$$  

  $$h_{\text{final}}(x) = \arg\max_{y \in Y} \sum_{\{t \mid h_t(x) = y\}} \alpha_t.$$
Summary of examination requirements

• Ensembles, bagging, boosting – general principles
• Simple bagging algorithm
• Random Forests
• Boosting with regression trees
• AdaBoost
• Practical use of bagging and boosting methods in R