Statistical Learning and Data Mining

Module 15: Boosting

Semester 2, 2017

Discipline of Business Analytics, The University of Sydney Business School

Module 15: Boosting

1. Boosting
2. Forward Stagewise Additive Modelling
3. AdaBoost
4. Gradient boosting

Boosting

Boosting

Boosting is one of the most powerful methods in statistical learning. The method is based on an additive model of the form

$$\widehat{f}(\boldsymbol{x}) = \sum_{m=1}^{M} \beta_m b(\boldsymbol{x}, \theta_m),$$

where β_m, $m = 1, \ldots, M$ are expansion coefficients, and $b(\boldsymbol{x}, \cdot)$ is a simple fixed function of the input $\boldsymbol{x}$ characterised by parameters $\boldsymbol{\theta}_m$.

Boosting

Additive basis expansion model:

$$\widehat{f}(\boldsymbol{x}) = \sum_{m=1}^{M} \beta_m b(\boldsymbol{x};\, \theta_m)$$

- In boosting, each function $b(\boldsymbol{x}, \theta_m)$ is a weak learner, in the sense that it is only weakly correlated with the response (regression) or slightly better than random guessing (classification).

- Boosting sequentially and adaptively combines a large number of weak learners towards a final model $\widehat{f}(\boldsymbol{x})$, which a strong learner.

Boosting regression trees

In the regression setting, one option is to use regression trees as basis functions (the weak learners), leading to the model

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} T(\boldsymbol{x}; \boldsymbol{\theta}_m),$$

where $T(\cdot; \boldsymbol{\theta}_m)$ are regression trees with a fixed number of splits and $\boldsymbol{\theta}_m$ the parameters of the tree (the split variables, split points, and predictions at the terminal nodes).

How can we fit this model for a large number of trees M?

Boosting regression trees (key concept)

Algorithm Boosting for regression trees

1: Set the number of trees M, the maximum depth d of each tree, and the shrinkage parameter ν.

2: Initialise $\widehat{f}(\boldsymbol{x}) = 0$ and $r_i = y_i$ for all the training set.

3: **for** $m = 1$ to M **do**

4: Fit a tree T_m of depth d to the training data $\{r_i, \boldsymbol{x}_i\}_{i=1}^N$.

5: Update the regression function $\widehat{f}(\cdot)$ by adding in a shrunken version of the new tree,

$$\widehat{f}(\boldsymbol{x}) \leftarrow \widehat{f}(\boldsymbol{x}) + \nu T_m(\boldsymbol{x})$$

6: Update the residuals,

$$r_i \leftarrow r_i - \nu\, T_m(\boldsymbol{x}).$$

7: **end for**

8: Output the boosted model,

$$\widehat{f}(\boldsymbol{x}) = \sum_{m=1}^{M} \nu\, T_m(\boldsymbol{x}).$$

Tuning parameters

- Number of trees (M): more trees lead to higher complexity and better fit to the training data. However, boosting is typically slow to overfit the data as we increase M.

- Shrinkage (ν): a lower ν shrinks each tree and slows down the learning process. Smaller values of λ lead to large values of M for the same training error, so that there is a trade-off. Typically $\nu < 0.1$ leads to the best test performance.

- Depth (d): a higher d leads a higher interaction order, and therefore controls the complexity of the model.

Forward Stagewise Additive Modelling

Forward stagewise additive modelling (key concept)

Boosting is based on **forward stagewise additive modelling**, which we now consider in a more general seting.

- In the example of boosting regression trees, we sequentially added new small trees without adjusting the split variables and parameters of the trees that had already been added.
- At each step, we simply fit the residuals of the current model, improving the overall fit.

Forward stagewise additive modelling

Starting from the additive expansion model

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} \beta_m \, b(\boldsymbol{x};\, \theta_m),$$

we want solve the empirical risk minimisation problem

$$\min_{\{\beta_m, \boldsymbol{\theta}_m\}_{m=1}^{M}} \sum_{i=1}^{N} L\left(y_i, \sum_{m=1}^{M} \beta_m b(\boldsymbol{x};\, \theta_m)\right).$$

The optimisation involves a large number of parameters and is in general computationally infeasible.

Forward stagewise additive modelling

Empirical risk minimisation for the additive expansion model:

$$\min_{\boldsymbol{\beta},\boldsymbol{\theta}} \sum_{i=1}^{N} L\left(y_i, \sum_{m=1}^{M} \beta_m\, b(\boldsymbol{x};\, \theta_m)\right).$$

Forward stagewise additive modelling approximates the solution by adding one new term at time, and solving the subproblem of fitting only a single basis function $b(\boldsymbol{x};\, \theta_m)$.

Forward stagewise additive modelling

Algorithm Forward stagewise additive modelling

1: Initialise $f(\boldsymbol{x}) = 0$.

2: **for** $m = 1$ to M **do**

3: Compute

$$\widehat{\beta}, \widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\beta}, \boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{N} L\big(y_i, f_{m-1}(\boldsymbol{x}_i) + \beta b(\boldsymbol{x}_i; \boldsymbol{\theta})\big).$$

4: Set

$$f_m(\boldsymbol{x}) = f_{m-1}(\boldsymbol{x}) + \nu b_m(\boldsymbol{x}).$$

5: **end for**

6: The estimated model is $f_M(\boldsymbol{x})$.

AdaBoost

AdaBoost

Consider the two class classification problem with the response coded as $Y \in \{-1, 1\}$.

The **AdaBoost** method combines a sequence of weak classifiers $G_m(\boldsymbol{x})$, $m = 1, \ldots, M$, by weighted majority voting

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

where the weights α_m are determined by the fitting algorithm.

AdaBoost

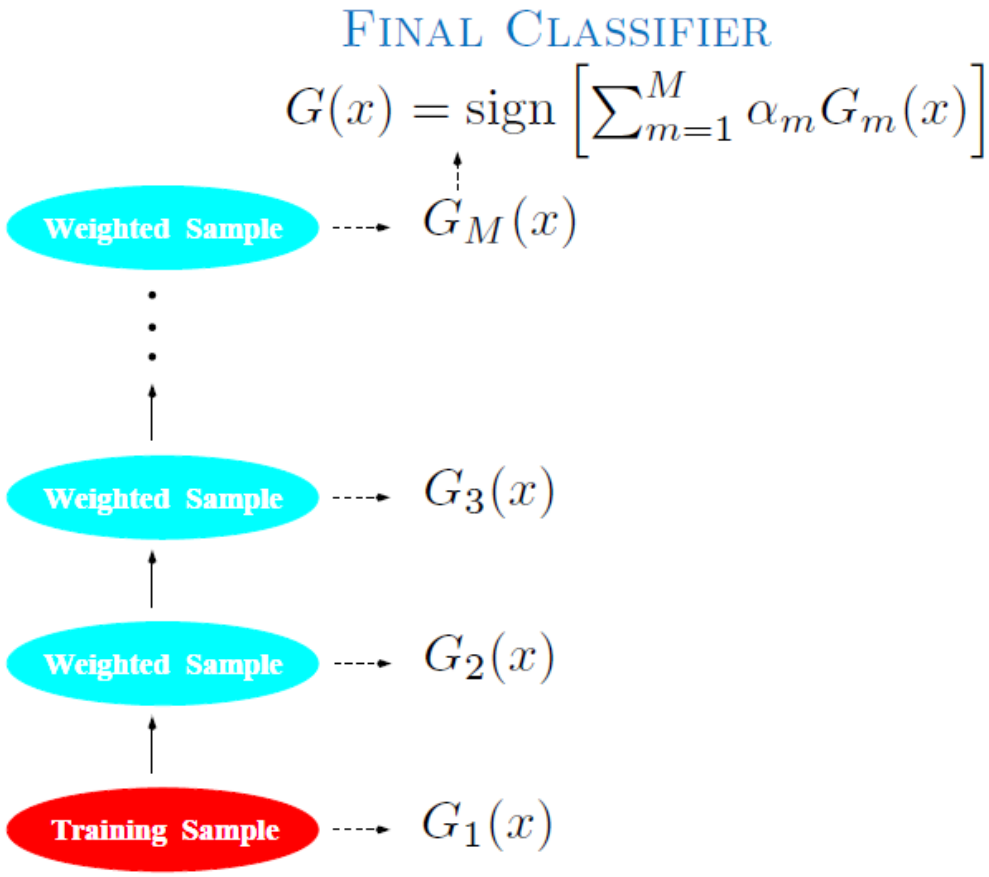


FIGURE 10.1. *Schematic of AdaBoost. Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.*

AdaBoost

Algorithm Adaboost.M1

1: Initialise the observation weights $w_i = 1/N$, $i = 1, \ldots, N$.

2: **for** $m = 1$ to M **do**

3: (a) Fit a classifier $G_m(\boldsymbol{x})$ to the training data using weights w_i.

4: (b) Compute the weighted error rate

$$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I\left(y_i \neq G_m(\boldsymbol{x}_i)\right)}{\sum_{i=1}^{N} w_i}$$

5: (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.

6: (d) Update the weights,

$$w_i \leftarrow w_i \exp\left[\alpha_m I\left(y_i \neq G_m(\boldsymbol{x}_i)\right)\right].$$

7: **end for**

8: Output the classification $G(\boldsymbol{x}) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(\boldsymbol{x})\right]$.

Example: improving a very weak classifier

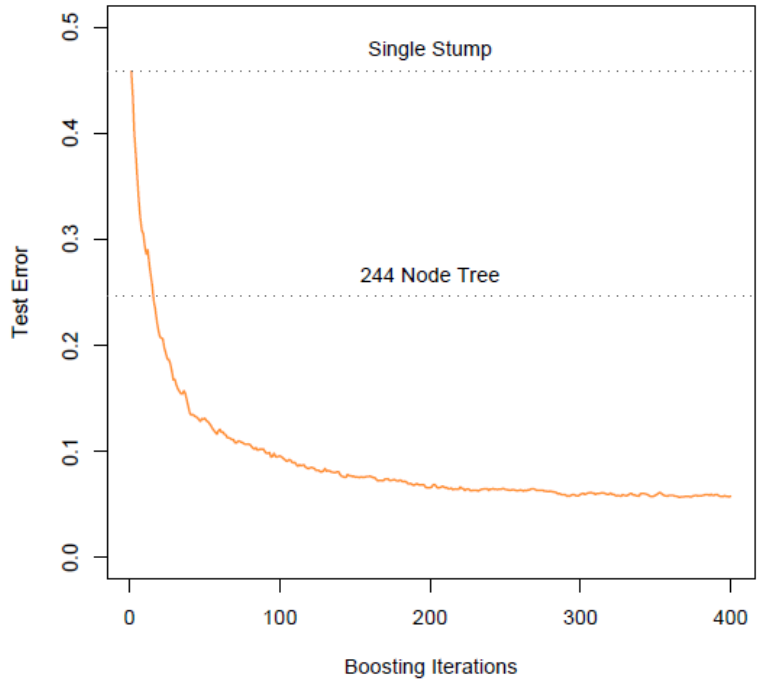


FIGURE 10.2. *Simulated data (10.2): test error rate for boosting with stumps, as a function of the number of iterations. Also shown are the test error rate for a single stump, and a 244-node classification tree.*

Example: Customer Churn

Test results

	Error rate	Sensitivity	Specificity	AUC	Precision
Logistic regression	0.224	0.795	0.759	0.884	0.744
ℓ_2 regularised	0.212	0.795	0.782	0.885	0.762
Decision Tree	0.236	0.752	0.774	0.836	0.746
Bagged trees	0.192	0.778	0.835	0.890	0.805
Random forest	0.180	0.803	0.835	0.903	0.810
Adaboost	**0.160**	**0.786**	**0.887**	**0.903**	**0.860**

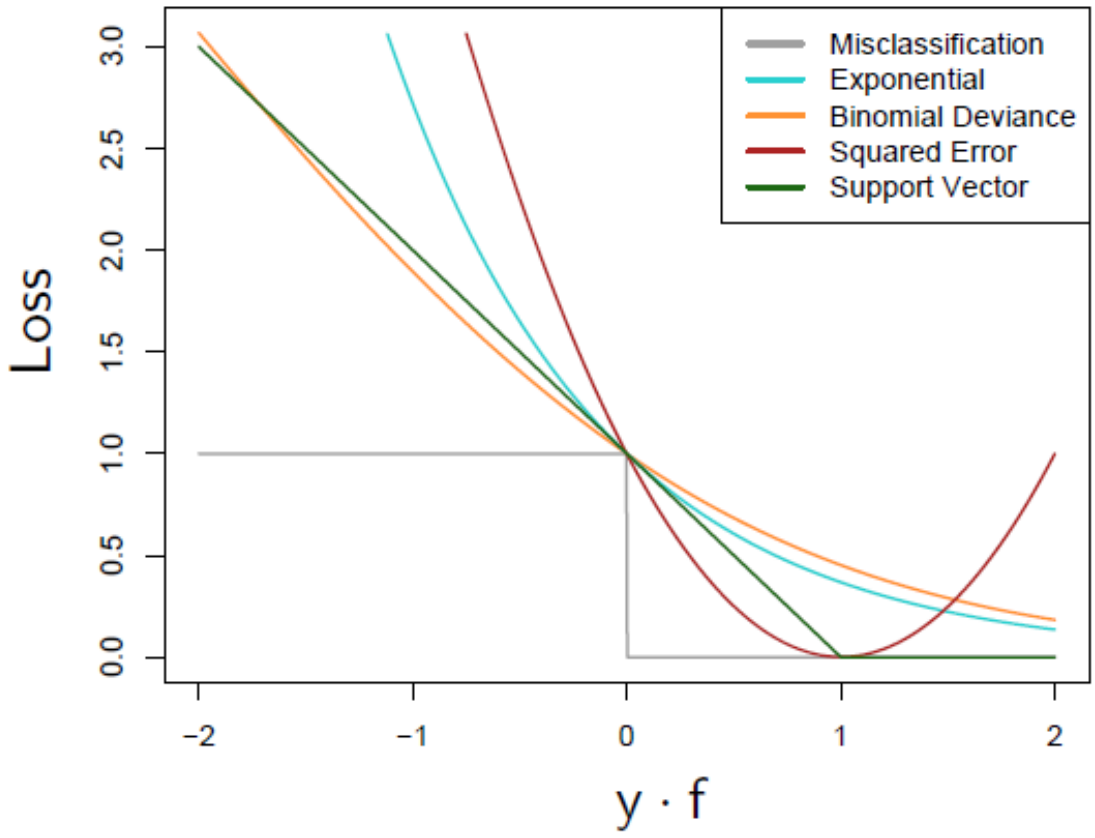
AdaBoost

- The AdaBoost was a major advance in supervised learning, making it a very popular boosting method.
- The Real AdaBoost algorithm extends the method for estimating probabilities.

AdaBoost

- We can show that AdaBoost corresponds to forward stagewise additive modelling based on the exponential loss function $L(y, f(\boldsymbol{x})) = \exp(-yf(\boldsymbol{x}))$.
- The advantage of the exponential loss is computational: it leads to the simple reweighing scheme of the algorithm.
- On the other, the exponential loss is far less robust than the cross-entropy loss in noisy settings, and the performance of AdaBoost degrades in such cases.

Loss functions



Gradient boosting

Gradient boosting

At each step in the forward stagewise procedure for boosting trees, we must solve

$$\min_{\boldsymbol{\theta}_m} \sum_{i=1}^{N} L\big(y_i, f_{m-1}(\boldsymbol{x}_i) + T(\boldsymbol{x}_i; \boldsymbol{\theta}_m)\big).$$

That requires us to find regions and constants $\{R_{jm}, \gamma_{jm}\}_1^{J_m}$ for the tree $T(\boldsymbol{x}_i; \boldsymbol{\theta}_m)$, given the current model $f_{m-1}(\boldsymbol{x}_i)$.

Gradient boosting

- Forward stagewise minimisation subproblem:

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} L(y_i, f_{m-1}(\boldsymbol{x}_i) + T(\boldsymbol{x}_i; \boldsymbol{\theta})).$$

- Given regions R_{jm} defined by the tree, finding optimal constants is typically straightforward by solving

$$\gamma_{jm} = \operatorname{argmin}_{\gamma} \sum_{\boldsymbol{x}_i \in R_{jm}} L(y_i,\ f_{m-1}(\boldsymbol{x}_i) + \gamma).$$

- However, finding the regions to solve the overall minimisation problem is computationally infeasible for general loss functions.

Gradient boosting

Forward stagewise minimisation subproblem:

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} L\big(y_i, f_{m-1}(\boldsymbol{x}_i) + T(\boldsymbol{x}_i; \boldsymbol{\theta})\big).$$

- Gradient boosting simplifies the minimisation problem, leading to an approximate solution.
- The next slide shows the generic gradient boosting algorithm for regression. The version for classification is similar.
- For the squared error loss, the resulting algorithm is equivalent to the least squares boosting algorithm from the beginning of the module.

Gradient tree boosting

Algorithm Gradient tree boosting

1: Initialise $\widehat{f}_0(\boldsymbol{x}) = \underset{\gamma}{\text{argmin}} \sum_{i=1}^{N} L(y_i, \gamma)$.

2: **for** $m = 1$ to M **do**

3: (a) For $i = 1, \ldots, N$ compute

$$r_{im} = - \left[\frac{\partial L(y_i, f(\boldsymbol{x}_i))}{\partial f(\boldsymbol{x}_i)} \right].$$

4: (b) Fit a regression tree to the targets r_{im}, giving terminal regions R_{jm}, $j = 1, 2, \ldots, J_m$.

5: (c) For $j = 1, \ldots, J_m$ compute

$$\gamma_{jm} = \underset{\gamma}{\text{argmin}} \sum_{\boldsymbol{x}_i \in R_{jm}} L(y_i,\ f_{m-1}(\boldsymbol{x}_i) + \gamma).$$

6: (d) Update $f_m(\boldsymbol{x}) = f_{m-1}(\boldsymbol{x}) + \nu \sum_{j=1}^{J_m} \gamma_{jm} I(\boldsymbol{x}_{\in} R_{jm})$, where ν is the shrinkage parameter.

7: **end for**

8: Output the boosted model $f(\boldsymbol{x}) = f_M(\boldsymbol{x})$.

Advantages of tree-based methods for data mining

- Natural handling of mixed data types.
- Handling of missing values.
- Robustness to outliers in the predictor space.
- Insensitive to monotone transformations of inputs.
- Computational scalability.
- Ability to handle irrelevant inputs.

Review questions

- What is the general idea of boosting?
- What are the key parameters in boosting?
- Why is boosting fundamentally different from other ensembles of trees such as random forests?