Boosting

Nathanaël Carraz Rakotonirina

Mathématiques Informatique et Statistique Appliquées (MISA)
Université d'Antananarivo

- **Boosting** is an algorithm that sequentially fits additive models of the form

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

- Each model improves on the errors made by the previous model.
- The base model F_m is a weak learner (with accuracy is only slightly better than 50%).
- It reduces bias without introducing too much variance (stacking and random forests do the opposite).

Forward stagewise additive modeling

We sequentially optimize the objective. At iteration m:

$$(\beta_m, \boldsymbol{\theta}_m) = \arg\min_{\beta,\boldsymbol{\theta}} \sum_{i=1}^{N} l(y_i, f_{m-1}(\boldsymbol{x}_i) + \beta F(\boldsymbol{x}_i; \boldsymbol{\theta}))$$

then, we set

$$f_m(\boldsymbol{x}) = f_{m-1}(\boldsymbol{x}) + \beta_m F(\boldsymbol{x}; \boldsymbol{\theta}_m))$$

This depends on the loss function l and the weak learner F.

Different loss functions

- **Least squares boosting** using the squared error loss

$$l(y, F(\boldsymbol{x})) = (y - F(\boldsymbol{x}))^2$$

- **AdaBoost** using the exponential loss

$$l(y, F(\boldsymbol{x})) = \exp(-yF(\boldsymbol{x}))$$

- **LogitBoost** using the negative log likelihood

$$l(y, F(\boldsymbol{x})) = \log(1 + e^{-2yF(\boldsymbol{x})})$$

Gradient boosting

Instead of deriving versions of boosting for every different loss functions,we can derive a generic version called **gradient boosting**.

Algorithm 1: Gradient boosting

Initialize $f_0(x) = \arg\min_F \sum_{i=1}^{N} l(y_i, F(\boldsymbol{x}_i))$

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

Compute the gradient (pseudo residuals) $r_{im} = -\left[\frac{\partial(L(y_i, f(\boldsymbol{x}_i)))}{\partial(f(\boldsymbol{x}_i))}\right]_{f(\boldsymbol{x}_i)=f_{m-1}(\boldsymbol{x}_i)}$

Fit a model $F_m = \arg\min_F \sum_{i=1}^{N} (r_{im}, F(\boldsymbol{x}_i))^2$

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

end

return $f_M(\boldsymbol{x})$

The **shrinkage factor** $0 \leq \nu \leq 1$ controls the size of the updates as a form of regularization.

Loss functions for gradient boosting

Some commonly used loss functions:

- Squared error
- Absolute error
- Huber loss
- Logloss
- Exponential loss
- Hinge loss

More

Explore

- Extreme gradient boosting (XGboost)
- LightGBM
- Catboost