Winter Course Statistical Learning

Tree Ensembles: Gradient Boosting

Marjolein Fokkema

Leiden University

Tree ensemble methods

Bagging and random forests:

- Trees are independent, can be fitted in parallel
- Fit large trees (low bias)
- Average over predictions of many trees (lowers variance)

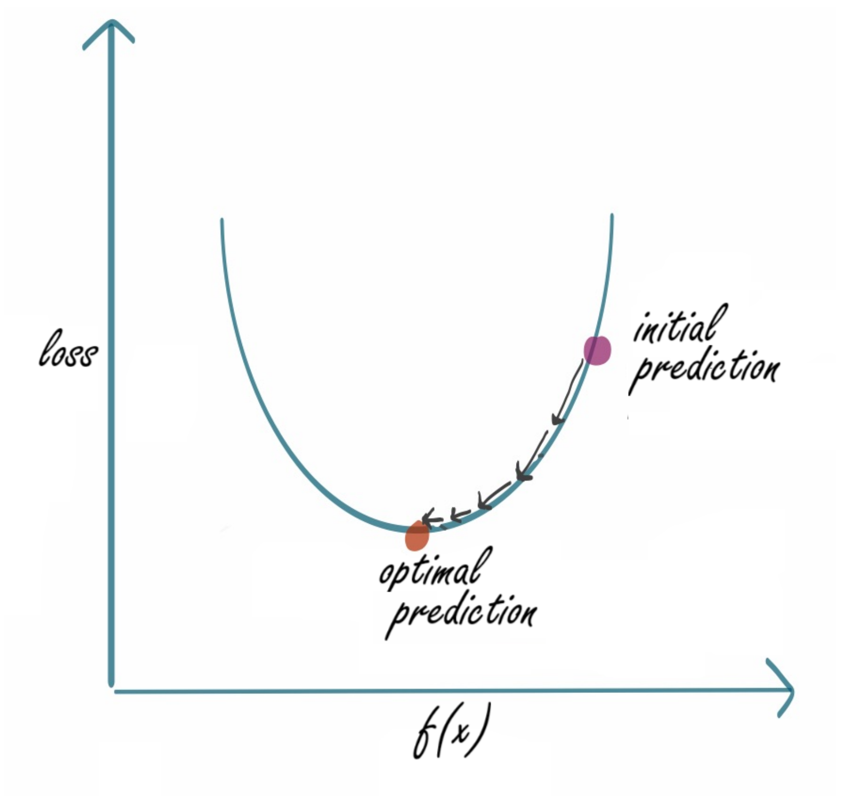
Gradient boosting:

- Trees are fitted sequentially
- Fit small trees (low variance)
- Average over predictions of many trees (lowers bias)

Ensembling can be done with any type of baselearner, but particularly effective with trees.

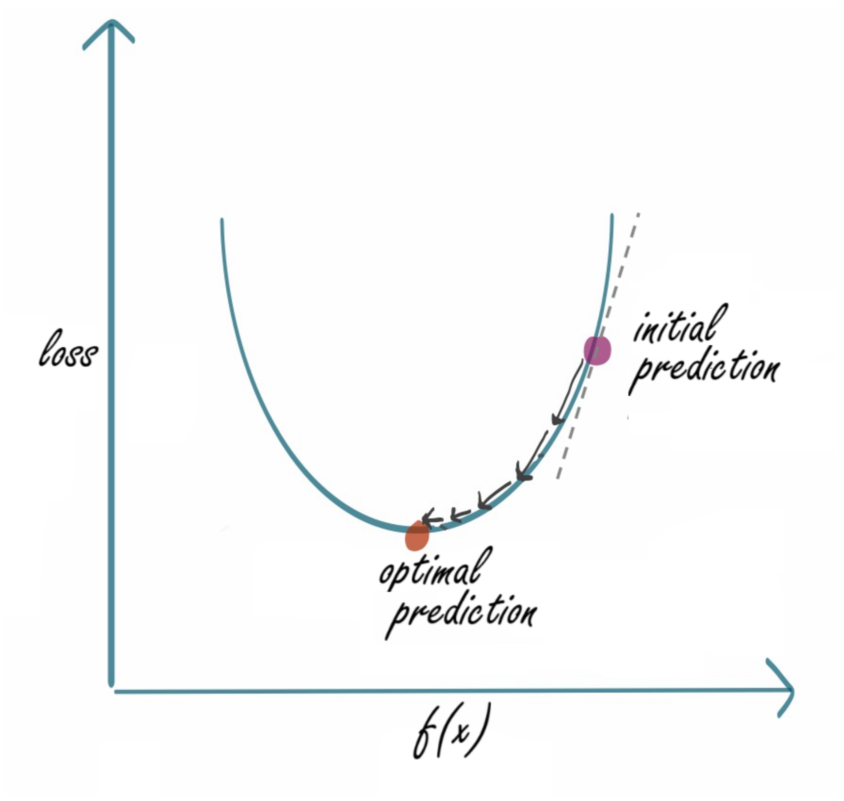
Gradient descent

Iterative procedure for finding the minimum of a function where no closed-form solution (formula) exists to compute it.



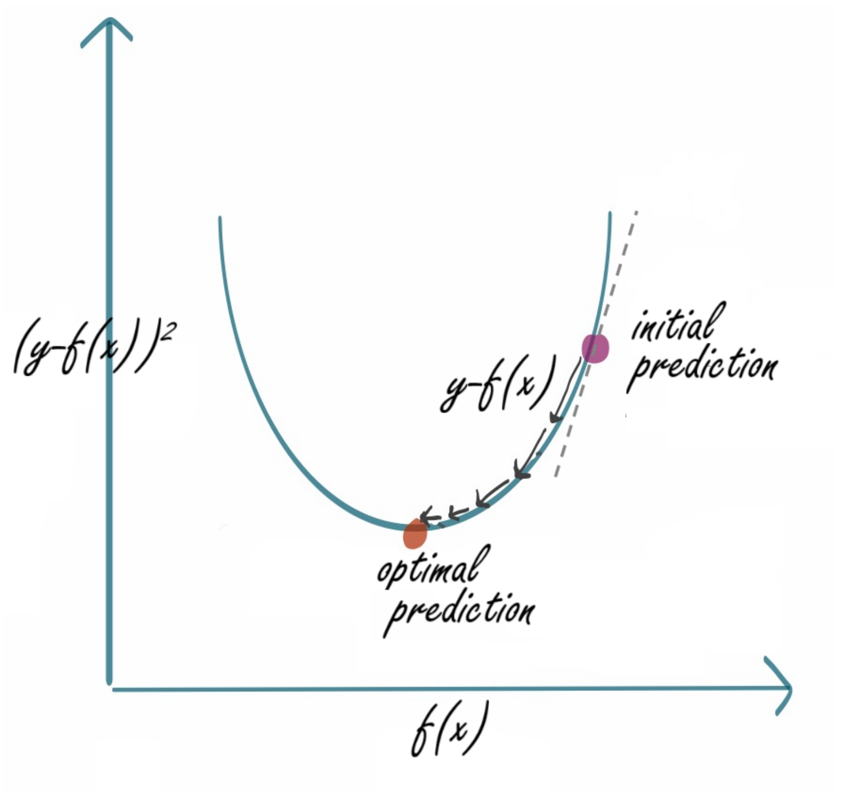
Gradient descent

Iterative procedure for finding the minimum of a function where no closed-form solution (formula) exists to compute it.



Gradient descent

Iterative procedure for finding the minimum of a function where no closed-form solution (formula) exists to compute it.



Gradient boosting

Given training data $\{x_i, y_i\}$ with $(i = 1, \ldots, n)$ and learning rate λ:

- Initialize with a constant, e.g.: $F_0(x) = \bar{y}$
- For $b = 1, \ldots, B$:
  - Compute *pseudo-residuals* (a.k.a. *negative gradient*):

$$r_{i,b} = y_i - F_{b-1}(x_i) = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x_i)=F_{b-1}(x_i)}$$

  - Fit a *regression* tree $\hat{f}_b(x)$ to predictors and pseudo-residuals $\{x_i, r_i\}$
  - Update the model: $F_b(x) = F_{b-1}(x) + \lambda \hat{f}_b(x)$
- Output final ensemble: $\hat{f}(x) = F_B(x)$

Stochastic gradient boosting

Given training data $\{x_i, y_i\}$ with $(i = 1, \ldots, n)$ and learning rate λ:

- Initialize with a constant, e.g.: $F_0(x) = \bar{y}$
- For $b = 1, \ldots, B$:
  - Compute *pseudo-residuals* (a.k.a. *negative gradient*):

$$r_{i,b} = y_i - F_{b-1}(x_i) = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x_i)=F_{b-1}(x_i)}$$

  - Fit a *regression* tree $\hat{f}_b(x)$ to a **sub sample of observations** from $\{x_i, r_i\}$
  - Update the model: $F_b(x) = F_{b-1}(x) + \lambda \hat{f}_b(x)$
- Output final ensemble: $\hat{f}(x) = F_B(x)$

Stochastic gradient boosting with binary outcome

Given training data $\{x_i, y_i\}$ with $(i = 1, \ldots, n)$ and learning rate λ:

- Initialize with a constant, e.g.: $F_0(x) = \bar{y}$ and $\eta_0 = \log\left(\frac{\bar{y}}{1-\bar{y}}\right)$
- For $b = 1, \ldots, B$:
  - Compute *pseudo-residuals* (a.k.a. *negative gradient*):

$$r_{i,b} = y_i - F_{b-1}(x_i) = -\left[\frac{\partial L(y_i, F(x_i))}{\partial \eta(x_i)}\right]_{\eta(x_i)=\eta_{b-1}(x_i)}$$

  - Fit a *regression* tree $\hat{f}_b(x)$ to subsample of observations from $\{x_i, r_i\}$
  - Update the **linear predictor of the** model: $\eta_b(x) = \eta_{b-1}(x) + \lambda \hat{f}_b(x)$
  - Note that $F_b(x) = \frac{e^{\eta(b)}}{1+e^{\eta(b)}}$. Thus, η_b is the *linear predictor* (can range from $-\infty$ to ∞), F_b is the predicted probability (can range from 0 to 1).
- Output final ensemble: $\hat{f}(x) = F_B(x)$.

Tuning parameters: Boosting

- `shrinkage` (λ, learning rate): Small, non-zero values; `gbm` default of 0.1 is quite high, values of .01 or .001 often better.

- `n.trees`: Number of random samples (trees) to generate. Generally requires values $> (1/\lambda)$.

- `distribution`: Specifies which loss function should be minimized, and thereby how the gradient is computed. For continuous responses specify ”gaussian” (or eg., ”laplace”, to minimize absolute error loss). For binary responses, use ”bernoulli”. See `?gbm` for more options.

- `interaction.depth` (tree depth; number of splits or nodes). Specifies the level of variable interactions allowed; default is 1, yielding trees with only a single split, i.e., an additive model.

- `bag.fraction`: Fraction of training observations randomly selected to fit each tree. A value of 1 will result in no subsampling, and all training observations used for inducing each tree. Default is 0.5, yielding a subsample comprising 50% of the training observations.