Foundations of Machine Learning

Brett Bernstein

August 22, 2018

Boosting: Concept Check

Boosting Learning Objectives

- Compare learning a linear model on a fixed set of basis functions on the input space, and an "adaptive basis function model" where the basis functions are learned.
- In particular, explain the "recipe" for an adaptive basis function model in terms of the base hypothesis space, and combined hypothesis space.
- Give psuedo-code for forward stagewise additive modeling (FSAM).
- Give the ingredients for gradient boosting machines; in particular, be able to explain why we need a [sub]differentiable loss function w.r.t. the prediction.
- Explain how gradient boosting uses "functional" gradient descent - i.e. learning the basis function (i.e. function in the base hypothesis space) that is closest to the negative gradient step direction given the current prediction function.
- Explain options for step sizes (line search and shrinkage parameter/learning rate).
- Explain variations on gradient boosting (stochastic gradient boosting, and column subsampling).

1. ($\star$) Show the exponential margin loss is a convex upper bound for the $0-1$ loss.

   Solution. Recall that the exponential margin loss is given by $\ell(y,a) = e^{-ya}$ where $y \in \{-1,1\}$ and $a \in \mathbb{R}$, and the $0-1$ loss is $\mathbf{1}(y \neq \operatorname{sgn}(a))$. If $\operatorname{sgn}(y) \neq a$ then $ya \leq 0$ and
   $$e^{-ya} \geq 1 - ya \geq 1 = \mathbf{1}(y \neq \operatorname{sgn}(a)).$$
   In general $e^{-ya} \geq 0$ so the we obtain the upper bound. To prove convexity, we compute the second derivative and note that it is positive:
   $$\frac{\partial^2}{\partial a^2} e^{-ya} = y^2 e^{-ya} > 0.$$

2. Show how to perform gradient boosting with the hinge loss.

   Solution. Recall that the hinge loss is given by $\ell(y,a) = \max(0, 1-ya)$. Define g by

   $$g(y,a) = \begin{cases} -y & \text{if } 1-ya > 0, \\ 0 & \text{else.} \end{cases}$$

   Then $g(y,a)$ is a subgradient of $\ell(y,a)$ with respect to a. At stage m of gradient boosting, we alredy have formed

   $$f_{m-1} = \sum_{i=1}^{m-1} \nu_i h_i.$$

   We then compute the pseudoresiduals r_m given by

   $$r_m = -\left(g(y_1, f_{m-1}(x_1)), \ldots, g(y_n, f_{m-1}(x_n))\right).$$

   After building the mock dataset $D^m = \{(x_1, (r_m)_1), \ldots, (x_n, (r_m)_n)\}$ we perform a least squares fit to obtain $h_m \in \mathbb{H}$. Then we can determine ν_m (usually a small fixed value). Finally we let $f_m = f_{m-1} + \nu_m h_m$.

3. Suppose we are using gradient boosting. On each step we can do a better job of fitting the pseudoresiduals if we allow for deeper trees. Why might deep trees be discouraged while gradient boosting?

   Solution. Deep trees can lead to overfitting the data.