Lecture November 5

Adaboost / Adaptive modeling / iterative modeling

Basic idea is to use a simple approximation and iterate based upon this.

- Regression

$$C(f) = \frac{1}{n}\sum_i (y_i - f(x_i))^2$$

Data set $\{(x_0, y_0), (x_1, y_1) \ldots (x_{n-1}, y_{n-1})\}$

- Define a function $f(x) = f_M(x)$

$$\boxed{f_M(x) = \sum_{m=0}^{M} \beta_m b(x; \gamma_m)}$$

often start with $f_0(x) = 0$ or some random values.

- want to minimize

$$\hat{\beta}, \hat{\gamma} = \underset{\beta\gamma}{\operatorname{argmin}}\, C(f_M)$$

Example

$$b(x; \gamma) = 1 + \gamma x$$

$M = 1$

$$f_1(x) = f_0(x) + \beta_1 b(x; \gamma_1)$$

$$f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$$

$$f_0(x) = 0$$

$$(\hat{\beta}_m, \hat{\gamma}_m) = \underset{\gamma\,\beta}{\arg\min} \sum_{i=0}^{n-1} \left(y_i - f_{m-1}(x) - \beta b(x; \gamma) \right)^2$$

Take derivatives wrt γ, β

$$\frac{\partial C}{\partial \beta} = -2 \sum_i (1+\gamma x_i)(y_i - \beta(1+\gamma x_i)) = 0$$

$$\frac{\partial C}{\partial \gamma} = -2 \sum_i \beta x_i (y_i - \beta(1+\gamma x_i)) = 0$$

Solve with stochastic gradient descent

algorithm Regression case

initialize $f_0(x; \gamma) = 0$

have a model for $b(x; \gamma)$

for $m = 1 : M$

a) optimize and compute

$$(\hat{\beta}_m, \hat{\gamma}_m) = \underset{\beta, \gamma}{\arg\min} \sum^{n-1} \mathcal{L}(y_i, f_{m-1}(x_i; \gamma_{m-1})$$

$i=0$ $+\beta b(x_i;\gamma))$

b) set $f_m(x;\gamma) =$

$f_{m-1}(x;\gamma_{m-1}) + \beta_m \underline{b(x;\gamma_m)}$

End for (continue till M)

This type of additive expansion is at the heart of many learning techniques.

Example: hidden layer in NN

$$b(x;\gamma) = \sigma(\gamma_0 + \gamma_1 x)$$

$$\sigma(t) = \frac{1}{1+e^{-t}}$$

Example: Decision trees

γ parametrizes split variable and split points of a node

Useful observation:

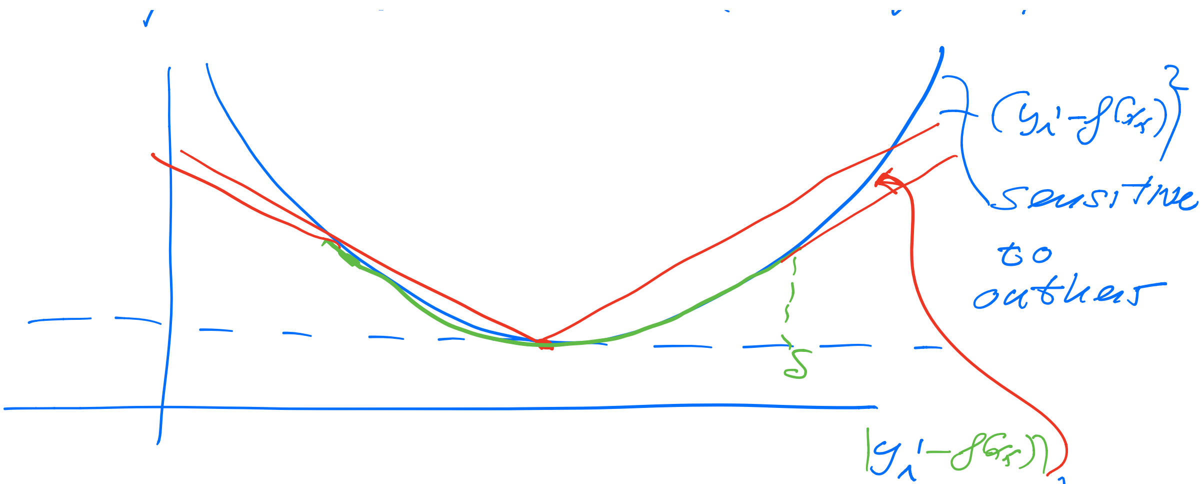
Individual L_i =

$$(\underbrace{y_i - f_{m-1}(x_i)}_{\text{Defines residual } r_{im}} - \beta b(x;\gamma))^2$$

$$= (r_{im} - \beta b(x;\gamma))^2$$

r_{im} = residual.

squared error: $(y_i - f(x_i))^2$



absolute error $= |y_i - f(x_i)|$

Huber Loss function

$$\mathcal{L}_i(y_i, f(x_i)) = \begin{cases} (y_i - f(x_i))^2 & \text{for } |y_i - f(x_i)| \le \delta \\ 2\delta |y_i - f(x_i)| - \delta^2 & \text{else} \end{cases}$$

often used with few data points and outliers.

Toward gradient boosting

$$C(f) = \sum_i \mathcal{L}(y_i, f(x_i))$$

$$\hat{f} = \underset{f \in \mathbb{R}^n}{\arg\min}\, C(f)$$

$$f = \{ f(x_0), f(x_1) \dots f(x_{n-1}) \}$$

M

$$f_M(x) = \sum_{m=0} f_m(x)$$

f_0 given by an initial guess.

Simplest way to solve the problem is steepest descent (before gradient descent)

Define gradient g_m

$$g_{im} = \left.\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right|_{f(x_i) = f_{m-1}(x_i)}$$

updated solution

$$\boxed{f_m = f_{m-1} - \rho_m g_m}$$

learning rate

$$\hat{\rho}_m = \arg\min_{\rho} L(f_{m-1} - \rho g_m)$$

ρ = scalar

Adaboost for classification

Hastie et al secs 10.1 – 10.10