Lecture 20 -- Boosting

Thursday, November 14, 2024 3:27 PM

What is boosting doing?

Additive model:

basis function

$$h(x) = \sum_{m=1}^{M} \beta_m \, b(x ; \gamma_m)$$

weight

parameter for basis

How do we fit these?

For some loss L

$$\{\hat{\beta}_m, \hat{\gamma}_m\}_{m=1}^{M} = \underset{\beta_m, \gamma_m}{\operatorname{argmin}} \sum_{n=1}^{N} L\left(y_n, \sum_m \beta_m b(x_n, \gamma_m)\right)$$

Problem: can be difficult

→ lots of params

→ depending on L may be difficult

Soln: Greedy approach

Forward Stagewise Additive Modeling

Forward Stagewise Additive Modeling

Do one value of m at a time.

$\hat{G}_0(x) = 0$

For $m = 1, \dots, M$

ⓐ $\hat{\beta}_m, \hat{\gamma}_m = \boxed{\underset{\beta,\gamma}{\operatorname{argmin}} \sum_{n=1}^{N} L(y_n, \hat{G}_{m-1}(x) + \underline{\beta b(x,\gamma)})}$

ⓑ $\hat{G}_m(x) = \hat{G}_{m-1}(x) + \underline{\hat{\beta}_m b(x, \hat{\gamma}_m)}$

Return $\hat{G}_M(x) = \sum_{m=1}^{M} \hat{\beta}_m b(x, \hat{\gamma}_m)$.

Ex. Regression w/ squared error

$$L(y, z) = (y - z)^2$$

then

$$L(y_n, \hat{G}_{m-1}(x) + \beta b(x;\gamma))$$

$$= (\underbrace{y_n - \hat{G}_{m-1}(x)}_{r_{nm}} - \beta b(x;\gamma))^2$$

r_{nm} = residual of predicting y_n

using $\hat{G}_{m-1}$

r_{nm} = residual of pre...
using $\hat{G}_{m-1}$

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

So SAM basically fits sequentially to residuals of prev. fit.

What does SAM have to do with boosting?

AdaBoost $\approx$ SAM w/ Exponential Loss

$$\boxed{L(y,h) = e^{-yh}}$$

If I do SAM w/ exp loss then at each step I solve

$$\hat{\beta}_m, \hat{f}_m = \underset{\beta, f_m}{\operatorname{argmin}} \underbrace{\sum_{n=1}^{N} L(\overset{y}{y_n}, \overbrace{\hat{G}_{m-1}(x) + \beta f_m(x)}^{h})}$$

using L = exp loss

$$\sum \exp(-y_n [\hat{G}_{m-1}(x) + \beta f_m(x)])$$

$$\sum_n \exp\left(-y_n\left[\hat{G}_{m-1}(x) + \beta f_m(x)\right]\right)$$

$$= \sum_n \underbrace{\exp\left(-y_n \hat{G}_{m-1}(x)\right)}_{\text{no } f_m,\ \beta} \exp\left(-y_n \beta f_m(x)\right)$$

I can regard as a weight w_{nm}

$$= \sum_n w_{nm} L(y_n, \beta f_m(x))$$

weights depend on accuracy of $\hat{G}_{m-1}$.

overall = weighted exp. loss

For cts y, can do this where we fit each new additive part to residuals of prev. iteration.

Practically:

Tuning params:

- loss function (*)

- loss function (*)
- number of trees M
- Shrinkage factor

$$\hat{G}_m(x) = \hat{G}_{m-1}(x) + \nu \alpha_m \hat{f}_m(x)$$

learning rate $\nu \in [0,1]$

- number of splits in my trees $\hat{f}_m$

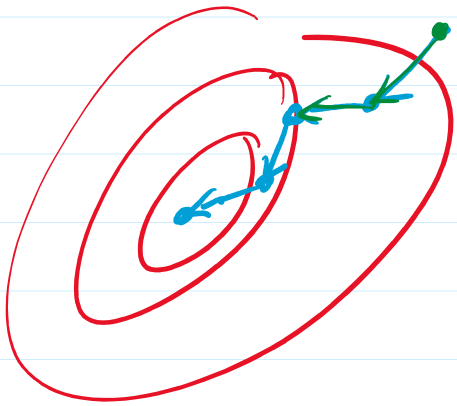
Can generalize this to any differentiable loss function L

Called: Gradient Boosting

Analogy:

Gradient Descent:

Want to optimize function J



For $m = 1, \dots, M$

$$x_m = x_{m-1} - \alpha_m \nabla J \Big|_{x_{m-1}}$$

<u>Boosting:</u> For $m = 1, \dots, M$

$$\hat{G}_m(x) = \hat{G}_{m-1}(x) + \alpha_m \hat{f}_m(x)$$

Looks like gradient descent if

$\hat{f} \approx -$ grad. of loss fn.

<u>Neural Networks</u>

NNs are predictive functions $\hat{f}$ that are built via <u>composition</u> of simpler functions: $\hat{f}_\ell$

$$\hat{f}(x) = \hat{f}_L(\hat{f}_{L-1}(\hat{f}_{L-2}(\cdots \hat{f}_2(\hat{f}_1(x))))).$$

Here L is called the number of <u>layers</u>

Here L is called the number of <u>layers</u>

$\hat{f}_\ell$ is called a layer.

The simplest NNs are called <u>feed forward</u> networks b/c the calc. from layer ℓ is fed forward into layer $\ell+1$.

For FNNs the $\hat{f}_\ell$ are comprised of two very simple operations:

(1) a linear transformation

$$W^{(\ell)} h + b^{(\ell)}$$

biases

generic input to layer

weights

(2) a non-linear transf $g^{(\ell)}$

activation function

[typically applied element-wise]

$\mathbb{R}^M$

$$\hat{f}_\ell(h) = g^{(\ell)}(W^{(\ell)} h + b^{(\ell)})$$

So $\hat{f}_\ell(h) = g^{(\ell)}(W^{(\ell)} h + b^{(\ell)})$

If $h \in \mathbb{R}^K$ then

$W^{(\ell)} \in \mathbb{R}^{K \times M}$, $b^{(\ell)} \in \mathbb{R}^M$

and we apply $g^{(\ell)}$ element-wise

<u>$L=1$</u>

$$\hat{f}(x) = \hat{f}_1(x) = g^{(1)}(W^{(1)} x + b^{(1)})$$

<u>$L=2$</u>

$$\hat{f}(x) = \hat{f}_2(\hat{f}_1(x))$$

$$= \hat{f}_2(g^{(1)}(W^{(1)} x + b^{(1)}))$$

$$= g^{(2)}(W^{(2)}(g^{(1)}(W^{(1)} x + b^{(1)})) + b^{(2)})$$

$$h^{(1)} = \hat{f}_1(x)$$

$$= g^{(2)}\left(W^{(2)} h^{(1)} + b^{(2)}\right)$$

Why called NN?

They can be described as a network (directed acyclic graph – DAG)

e.g.