Boosting

Peter N. Bakker

08-06-2023

Boosting

- Boosting er en metode til at kombinere flere "weak classifiers" for at opnå en stærkere og mere præcis classifiers (typsik i form af en skov af træer).
- Boosting opbygger gradvist en additiv model ved at tilføje nye classifiers i hver iteration.

Forward Stagewise Additive Modelling

Forward Stagewise Additive Modelling er et generel framework for at opbygge additive modeller.

- Initialiser modelen: $f_0(x) = 0$
- For $m = 1$ til M:
  1. Udregn

$$(\hat{\beta}_m, \hat{\gamma}_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma))$$

  2. Sæt $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$
- L er den pågældende loss funktion såsom MSE eller en likelihood baseret loss function. β er koefficienter i hvert led af "ekspansionen" og $b(x; \gamma) \in R$ er typisk simple funktioner (f.eks. en træstub) baseret på de givner x'er og parametrer i funktionen γ. For træmodeller er γ en split variable, split nodes og prædiktioner i leaf nodes.
- Til sidst bestemmes prædiktionerne for et nyt (x_i, y_i) par af et ensemble af de additive modeller.

AdaBoost (Adaptive Boosting)

- Tildel initialt vægte $w_i = 1/n$ til alle træningsdata, hvor n er antallet af observationer.
- For hver iteration $m = 1$ til M:
  1. Træn en "weak classifier" (typisk en træstub) $G_m(x)$ på træningsdataene med vægtede observationer.
  2. Beregn den vægtede fejlrate:

$$\text{Fejl}_m = \frac{\sum_{i=1}^{n} w_i \cdot \mathbb{I}(y_i \neq G_m(x_i))}{\sum_{i=1}^{n} w_i}$$

  3. Beregn klassifikatorens vægt i den endelige model (*amount of say*):

$$\alpha_m = \log\left(\frac{1 - \text{Fejl}_m}{\text{Fejl}_m}\right)$$

  4. Opdater vægtene for de næste iterationer:

$$w_i \leftarrow w_i \cdot \exp(\alpha_m \cdot y_i \cdot G_m(x_i)), \quad \text{for } i = 1, 2, \ldots, n$$

- Den endelige model er en vægtet sum af svage klassifikatorer:

$$G(x) = \sum_{m=1}^{M} \alpha_m \cdot G_m(x)$$

AdaBoost: Additiv modelfit

- AdaBoost kan ses som en metode til at fitte en additiv model.
- I hver iteration tilføjes en ny classifier til den eksisterende model ("skoven" af træstubber bliver større).
- Modellen er dog ikke lineær fordi vi fastholder tidligere $\alpha_1, \ldots, \alpha_{m-1}$.
- Modellen fittes ved at minimere en tabsfunktion, f.eks. eksponentiel tabsfunktion.

AdaBoost: Eksponentiel tabsfunktion

- AdaBoost bruger ofte en eksponentiel tabsfunktion, der resulterer i større vægtning observationer modellen ikke tidligere kunne prædiktere.
- Den eksponentielle tabsfunktion er defineret som:

$$L(y, f(x)) = \exp(-yf(x))$$

hvor y er den faktiske klasse og $f(x)$ er den prædikterede klasse.

- I AdaBoost skal man i hvert trin minimere

$$(\hat{\beta}_m, \hat{G}_m) = \arg\min_{\beta, G} \sum_{i=1}^{N} \exp\left[-y_i \left(f_{m-1}(x_i) + \beta G(x_i)\right)\right]$$

- Ved lang omskriving fås:
$w_i^{(m+1)} = w_i^{(m)} \cdot e^{\alpha_m I(y_i \neq G_m(x_i))} \cdot e^{-\beta_m}$
- Ligesom i forward stagewise algoritmen.

Gradient Boosting og Steepest Descent

- Gradient boosting er en metode til at bygge en additiv model ved at fitte på gradienten af en loss funktion i hver iteration.
- Steepest descent, eller steepest gradient descent, er en optimeringsmetode, der anvendes til at finde minimumspunktet for en funktion ved at bevæge sig i retningen af den største negative gradient.
- I gradient boosting bruges steepest descent til at finde værdierne af β og G, der minimerer tabsfunktionen.

Steepest Descent i Gradient Boosting

- Gradienten af tabsfunktionen er den retning, hvor funktionen vokser hurtigst.
- Steepest descent bruger den negative gradient af tabsfunktionen til at justere værdierne af β og G i hver iteration.
- Ved at bevæge sig i retningen af den negative gradient reduceres tabsfunktionen og modelens prædiktive evne forbedres.
- $\mathbf{h}_m = -\rho_m \mathbf{g}_m$ hvor ρ_m er en skalar og $\mathbf{g}_m \in R^N$.

$$\rho_m = \arg\min_{\rho} L(\mathbf{f}_{m-1} - \rho \mathbf{g}_m)$$

- Idé fit et træ til $\mathbf{g}_m$ og udvikl modellen heraf (vi kan godt lide at optimere vha. gradienter fra matematik).

Gradient Tree Boosting Algoritme

1. Initialisér den additive model: $\hat{f}_0(x) = \arg\min_{\gamma} \sum_{i=1}^{N} L(y_i, \gamma)$
2. For $m = 1$ til M:
   - 2.1 Beregn pseudo-residualer: $r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=\hat{f}_{m-1}(x)}$
   - 2.2 Fit et træ til pseudo-residualerne: $T_m(x)$
   - 2.3 Opdater den additive model: $\hat{f}_m(x) = \hat{f}_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$
3. Output for den endelige model: $\hat{f}(x) = f_M(x)$

- De endelige prædiktioner bestemmes ved ensemble.

Shrinkage og Regularisering ved Gradient Boosting

Shrinkage:

- Hvert træ har lige meget at skulle have sagt tilføj parameter ν til hvert træ ift. hvor de skal have at sige til prædiktion.
- ν kan betragtes som læringsraten i boosting algoritmen.
- Naturlig har antallet af træer M også betydning for om der kan være tale om overfit eller ej.

Regularisering:

- Brug af reguleringsparametre til at begrænse modellens kompleksitet.
- L1-regularisering (Lasso) og L2-regularisering (Ridge) anvendes.
- Gør koefficienterne mindre (og dimensionen lavere) ligesom kendt fra andre modeller.

Gradient Tree Boosting vs. AdaBoost

- Gradient tree boosting giver mulighed for forskellige tabsfunktioner og mere fleksibilitet i valg af træer. Et træ behøver ikke længere kun at være en træstub, men kan godt have mange flere split og leaf nodes.
- AdaBoost bruger en eksponentiel tabsfunktion, mens gradient tree boosting kan tilpasses forskellige tabsfunktioner afhængigt af problemet.

Dataeksempel phoneme

- Bestemme kategorien af ordlyde som "aa" eller "ao" bestemt ved frekvenser.
- Der 1717 observationer jeg sætter 1300 i train resten i test. Der er 50 træer i ada og 100 i gbm, som vist skulle være standard i deres pakker.
- Fejlrate ADA = 0.167
- Fejlrate GBM = 0.161

```
##
## pred_ada  aa  ao
##       ao  38 228
##       aa 119  32

## [1] "pred_gbm"

##
##           aa  ao
##   FALSE  118  28
##   TRUE    39 232
```