Vignette Title

Vignette Author

2018-04-03

Methodology

- Table mit notation

Learning Theory Reminder

Loss Function

The aim of machine learning is to find a model (function) $\hat{f}$ which approximate the real but unknown f. which best suits our data. But finding this model requires a mapping from the training data $\mathcal{D}_{\text{train}} = \left\{(x^{(i)}, y^{(i)}) \mid i \in \{1, \ldots, n\}\right\}$ to a model $\hat{f}$. This mapping is called inducer. \

To quantify the goodness of a prediction $y = f(x)$ we need a function to measure the loss of this prediction. Basically, the loss function can also be seen as a metric between the true value y and its prediction y:

$$\begin{aligned} L : \mathcal{Y} \times \mathcal{X} &\to \mathbb{R}_+ \\ y, x &\mapsto L\left(y, f(x)\right) \end{aligned}$$

The loss function is used within the inducer to fit a function (model) $\hat{f}$ using training data $\mathcal{D}_{\text{train}}$ (see section). It is worth mentioning that different loss functions transfer their properties to the inducer. For instance measuring the absolute difference between y and $f(x)$ (absolute loss) is more robust in terms of outliers then measuring the qudratic differences (quadratic loss). \

The properties of the loss function is also used to tackle different tasks. Doing classification requires other loss functions than regression tasks. To get an overview about different losses and their use see section **??** about the implemented loss classes of `compboost`.

Empirical Risk

It would be desirable to have the loss for every possible combination of $x \in \mathcal{X}$ and the corresponding true value $y \in \mathcal{Y}$. Therefore, the natural thing would be to measure the expectation of the loss with respect to the joint distribution $\mathbb{P}_{xy}$. This expectation is defined as the risk $\mathcal{R}(f)$:

$$\mathcal{R}(f) = \mathbb{E}\left[L(y, f(x))\right] = \int L(y, f(x)) \; d\mathbb{P}_{xy}$$

Since $\mathbb{P}_{xy}$ is unknown it is not possible to exactly calculate $\mathcal{R}(f)$. The most common way to approximate the risk is to use its empirical analogon the mean using the observation of the training data $(y, x) \in \mathcal{D}_{\text{train}}$. This is called the empirical risk $\mathcal{R}_{\text{emp}}(f)$:

$$\mathcal{R}_{\text{emp}}(f) = \frac{1}{n} \sum_{i=1}^{n} L\left(y^{(i)}, f(x^{(i)})\right)$$

It is also common to use the empirical risk as a summed version:

$$\mathcal{R}_{\text{emp}}(f) = \sum_{i=1}^{n} L\left(y^{(i)}, f(x^{(i)})\right)$$

In `compboost` we are using the average version of the empirical risk.\

Loss Minimization

An obvious aim is now to minimize the empirical risk which is also known as loss minimization and use the function $\hat{f}$ which minimizes $\mathcal{R}_{\text{emp}}(f)$:

$$\hat{f} = \arg\min_{f \in H} \mathcal{R}_{\text{emp}}(f)$$

In component-wise boosting we assume that f is a function which can be parameterized by $\theta \in \Theta$ since we want to have interpretable learner (as we will see later). Hence, we can parameterize the empirical risk:

$$\mathcal{R}_{\text{emp}}(\theta) = \frac{1}{n} \sum_{i=1}^{n} L\left(y^{(i)}, f(x^{(i)}|\theta)\right)$$

Therefore, the loss minimization yields in finding a parameter setting $\hat{\theta}$ which minimizes $\mathcal{R}_{\text{emp}}(\theta)$:

$$\hat{\theta} = \arg\min_{\theta \in \Theta} \mathcal{R}_{\text{emp}}(\theta)$$

Gradient Boosting Reminder

Forward Stagewise Additive Modelling

Generally, boosting fits an additive model [**?**, p. 341]. This means, that we can fit f in an additive fashion

$$f(x) = \sum_{m=1}^{M} \beta[m] b\left(x, \theta^{[m]}\right)$$

where $\beta^{[m]}$, $m = 1, \ldots, M$ are the expansion coefficients and $b(x, \theta) \in \mathbb{R}$ are so-called basis functions of the input x specified by the parameters θ. \

Having $f(x)$ our goal is now to minimize $\mathcal{R}_{\text{emp}}$ using this additive structure:

$$\mathcal{R}_{\text{emp}}(f) = \frac{1}{n} \sum_{i=1}^{n} L\left(y^{(i)}, f(x^{(i)})\right) = \frac{1}{n} \sum_{i=1}^{n} L\left(y^{(i)}, \sum_{m=1}^{M} \beta^{[m]} b\left(x, \theta^{[m]}\right)\right)$$

We notice, that we can parameterize f by introducing a parameter vector θ_0 containing all parameter:

$$\theta_0 = \left(\left(\beta^{[1]}, \theta^{[1]}\right), \ldots, \left(\beta^{[M]}, \theta^{[M]}\right)\right)$$

Hence, we want to minimize $\mathcal{R}_{\text{emp}}(\theta_0) = \mathcal{R}_{\text{emp}}(f)$ with respect to $\theta 0$. However, the dimension of θ_0 can be very big which makes it difficult to find $\hat{\theta}_0$.\

As [**?**, p. 342] have pointed out, " *forward stagewise additive modeling approximate the solution by sequentially adding new basis functions to the expansion without adjusting the parameters and coefficients of those that have already been added.*" This procedure is shown in Algorithm 1.

Initialize $\hat{f}^{[0]} = 0$;

for $m \in \{1, \ldots, M\}$ **do**

  // Fit m-th baselearner:

  $\left(\hat{\beta}^{[m]},\ \hat{\theta}^{[m]}\right) = \arg\min_{\beta,\theta} \frac{1}{n} \sum_{i=1}^{n} L\left(y^{(i)},\ \hat{f}^{[m-1]}(x^{(i)}) + \beta^{[m]} b\left(x, \theta^{[m]}\right)\right)$;

  // Update $\hat{f}$:

  $\hat{f}^{[m]}(x) = \hat{f}^{[m-1]}(x) + \hat{\beta}^{[m]} b(x, \hat{\theta}^{[m]})$;

end

Algorithm 1: Forward stagewise additive modeling.

- find parameter in a greedy fashion (makes optimization very simple)

Gradient Boosting

- Ganz kurz forward stagewise additive modelling (eher zitieren)
- Ganz kurz boosting (eher zitieren)

Component-wise Boosting

- Bisschen genauer model based boosting
- Verweis auf implementierte baselearner in classes