BAYESIAN OPTIMIZATION: HYPERPARAMETER TUNING FOR GBM

Christoffer Riis (153147)

Technical University of Denmark

1. INTRODUCTION

In this project, Bayesian Optimization (BO) is used to find the optimal hyper-parameters for the gradient boosting machine (GBM) LightGBM trained to predict the popularity of a post on Instagram. I will first explain the concepts of a GBM and BO, and thereafter I carry out experiments on different hyper-parameters of LightGBM with different BO approaches alongside investigating BO for finding a single or multiple hyper-parameters.

2. THEORY

The theory behind this project is two-fold. The first section is on Gradient Boosting Machines (GBMs), which are the models I want to tune, i.e. finding the optimal hyper-parameters. The second section is on Bayesian Optimization (BO), which are the method that I will use to find the optimal hyper-parameters for the GBMs.

2.1. Gradient Boosting Machines

A Gradient Boosting Machine (GBM) is an ensemble of weak predictors used for regression and classification in machine learning [1] and is formulated as:

$$F(x; \{\beta_m, a_m\}_{m=1}^{M}) = \sum_{m=1}^{M} \beta_m h_m(x; a_m), \tag{1}$$

where h_m is typically modeled by regression trees, x is the input variable and β_m is the weight (or learning rate) of regression tree h_m with parameters a_m. A GBM is a model built in a stage-wise fashion similar to other ensemble methods, e.g. AdaBoost or Random Forest, and is optimized by a best greedy-step such that an update of the GBM is given by:

$$F_m(x) = F_{m-1}(x) + \beta_m h(x; a_m). \tag{2}$$

So far, I have introduced the very important hyper-parameter *learning rate* (β_m), which weight the impact of each regression tree. One could argue that the learning rate is used to regularize the model, since smaller learning rates corresponds to take smaller steps in the direction of the gradient. This form of regularization is often referred to as *regularization by shrinkage*. In the following, I will lay out three other regularization parameters, which all are connected to the base learners (regression trees).

- **Maximum number of leaves**: The maximum number of leaves is used to penalize the complexity of the trees. Specifying the number of leaves can also be seen as a post-pruning of the tree.
- **Feature fraction**: By only using a fraction of the features, the individual regression trees will be trained on different feature subsets and thus, each tree will explore different connections between the features.
- **L2 regularization**: Applying L2 regularization is a common approach in machine learning, however how it is applied varies for different algorithms. Without going into details, the L2 regularization in GBMs is applied when calculating the pseudo-residuals, i.e. a regularization term is added when the pseudo-residuals are computed for each terminal region [1].

As for all regularization parameters, the optimal value is a trade-off between underfitting and overfitting, e.g. constraining the maximum number of leaves to low a value might result in underfitting, whereas a high value or no constrain is prone to overfit the model to the training data. Finding the optimal values for the four hyper-parameters laid out above can be troublesome. In the following section, I will shortly explain how Bayesian Optimization can be used to automate the process of finding the optimal hyper-parameters.

2.2. Bayesian Optimization

Bayesian Optimization can be motivated by how a group of people will optimize a function. Imagine that a group of people is given a few data points and they should estimate where the maximum of the underlying function is. Some people will choose a point close to the current maximum, others will interpolate or extrapolate between points, and some will guess on unexplored areas of the space. Together this group of people act as an heuristic for finding the place of the maximum value by combining *exploration*, *exploitation*, and *uncertainty*. These three factors are central in Bayesian Optimization (BO). BO uses Bayes' Theorem:

$$P(\text{model}|\text{data}) \propto P(\text{data}|\text{model})P(\text{model}), \tag{3}$$

and can be used to sequentially optimize a noisy, black-box function $f : \mathcal{X} \rightarrow \mathbb{R}$ such that $x^* = \arg\max_{x \in \mathcal{X}} f(x)$, where $\mathcal{X}$ is the space of interest. BO is sequential since for each iteration, a new data point x_i is sampled, the objective function $f(x_i)$ is queried, and then the model posterior is updated. To update the model posterior, the objective function f is modeled as a probabilistic model (which also gives uncertainty estimations) such that the posterior is given as the product of the likelihood and the prior of f:

$$P(f|\text{data}) \propto P(\text{data}|f)P(f), \tag{4}$$

where the data consists of the samples from the objective function $(x_i, f(x_i))$ with $x_i \in \mathcal{X}$, i.e. the posterior corresponds to the approximation of the true unknown objective function. In this project, f is approximated with a Gaussian Process as the probabilistic model. For the Gaussian Processes in this project, the *squared-exponential* kernel is used to measure the similarity between samples, i.e. the different values for the parameters. The kernel is given as:

$$k_{SE}(x_i, x_j) = \sigma^2 \exp\left(-\frac{\|x_i - x_j\|^2}{2l^2}\right) + \sigma_\varepsilon, \tag{5}$$

where the length scale $l > 0$ determines the length of extrapolation, the output variance $\sigma^2 > 0$ is a scaling factor that determines the average distance from the mean of the function we are modeling, and σ_ε^2 is the variance of the noise on the sampled data points. These three parameter are chosen by maximum likelihood estimation [2]. However, I have still not described how to compute the prior $P(f)$. According to the *no free lunch* theorem, nothing is free. In this case, we introduce the two new parameters μ_f and σ_f, which are the prior's mean and standard deviation. In practice, these will be estimated from a small batch of samples, e.g. five initial data points, from the objective f.

The next core component of BO, is how to choose which point to query in an iteration and the answer lies in the *acquisition functions*. Formally, an acquisition functions is defined as an auxiliary function $a : \mathcal{X} \rightarrow \mathbb{R}^+$, which uses the previous acquired data points and the hyper-parameters of the probabilistic model to define the next point to acquire x_{i+1}. In this project, I will consider four different acquisition functions given in the following. Let Φ be the cumulative distribution function and ϕ the probability density function. Let $mu(\cdot)$ and $\sigma(\cdot)$ denote the mean and standard deviation for all values in $\mathcal{X}$, x^+ be the current best samples, i.e. $x^+ = \arg\max_{x_n \in x_{1:i}} f(x_n)$, and let ξ be the trade-off parameter between exploitation and exploration. Finally, let Z denote $\frac{\mu(x) - f(x^+) - \xi}{\sigma(x)}$. Then the four functions are given as:

1. **Probability of improvement**

$$\text{PI}(X) = \Phi(Z),$$

2. **Expected Improvement**

$$\text{EI}(x) = (\mu(x) - f(x^+))\Phi(Z) + \sigma(x)\phi(Z),$$

if $\sigma(x) > 0$ and otherwise equal to 0.

3. **Gaussian Process Upper Confidence Bound** Here, I will both consider an exact version leading to no-regret as well as an approximation.

$$\text{GP-UCB} = \mu(x) + \sigma(x)\sqrt{2\log\left(\frac{n^{\frac{d}{2}+2}\pi^2}{3\delta}\right)},$$

where n is the iteration, d is the dimensionality of $\mathcal{X}$, and $\delta \in (0, 1)$. The approximation is given as

$$\text{GP-UCB}_{approx}(x) = \mu(x) + \epsilon \log(n)\sigma(x),$$

where ϵ is to be determined by the user.

3. METHOD

In this project, I use BO to optimize four parameters of a GBM: *learning rate*, *maximum number of leaves*, *feature fraction*, and *L2 regularization*. I use the implementation of LightGBM [3] to predict the popularity of a post on Instagram, where the accuracy measure is given by the Spearman Ranking Correlation (SRC). The Instagram data set consists of 16 different social features and has 50k observations. During the optimization, the data set is split with 45k samples for training and 5k samples for testing. To ensure generalization, the data split is happening at random at each iteration of BO.

4. EXPERIMENTS

4.1. Comparison of acquisition functions

In the first experiment, we will compare the four acquisition functions by estimating the optimal learning rate given the default values of the LightGBM implementation for the other values. At first, the search space of the learning rate is defined as a parameterizing of the discrete space $\hat{\mathcal{X}}_\beta = \{1, 2, ..., 100\}$ into $\mathcal{X}_\beta$ by $\mathcal{F}_\beta(x) = 0.1^{x/20}$. Additionally, the discrete space is standardized before used in BO to avoid numerical issues. We sample five data points and then takes 30 steps with Bayesian Optimization using all four acquisition functions. After the 30 iterations, the optimal learning rate for each acquisition function is then defined as $\beta^* = \arg\max_x \mu(x)$. The results are seen in Table 1 together with the highest SRC for each $\mu(x)$, the variance of the noise, and the value of the free parameter. Since the noise variance is relatively high, the corresponding SRC for each of the learning rates are estimated with an average over ten trials. The results are displayed in Table 2, where it is seen that the learning rate at 0.018 yield the best SRC with lowest standard deviation as well. Hence, in this case the probability of improvement

Acquisition function	$\max \mu(x)$	β^*	σ_ε^2	Free param.
PI	**.429**	.040	.0256	$xi = 0.01$
EP	.426	.141	.0251	$xi = 0.01$
GP-UCB	.428	.316	.0239	$\delta = 0.01$
GP-UCB$_{approx}$	.418	.141	.0247	$\epsilon = 0.01$

Table 1: The optimal learning rate with different acquisition functions.

yielded the best result. However, it should be noted that the free parameter has not been tuned for any of the four acquisition functions.

β^*	.040	.141	.316
Avg. SRC	**.430** (±.010)	.429 (±.012)	.425 (±.010)

Table 2: Average SRC over ten trials for each of the optimal learning rates given by the four acquisition functions.

4.2. Optimal parameters

We will now estimate the optimal values for the other parameters by estimating the optimal parameter one by one using the probability of improvement. The search spaces are defined as the set $\{2, 10, 20, 30, ..., 1000\}$ for the number of leaves, $\{0.50, 0.51, ..., 1.00\}$ for the feature fraction, and the L2 regularization is defined as a parameterizing of the discrete space $\hat{\mathcal{X}}_\beta = \{1, 2, ..., 100\}$ into $\mathcal{X}_\beta$ by $\mathcal{F}_\beta(x) = 0.1^{6x/100}$. Table 3 shows the estimated optimal values and that the average SRC first decreases to 0.427 and then increases to 0.432. The average SRC should be lower bounded by the previous average SRC, since I am updating the parameters in a sequential order. However, the decrease in the SRC indicate uncertainty and noise in the estimates.

Acquisition function	Avg. SRC	β^*	σ_ε
#Leaves	.427 (±.011)	150	.0251
Fraction	.430 (±.011)	.59	.0251
L2	**.432** (±.014)	.024	.0253

Table 3: The optimal number of leaves, feature fraction, and L2 regularization found with probability of improvement.

4.3. 4D parameter search space

In the previous experiment with sequential parameter-tuning, I have assumed that the parameters are decoupled. However, that might not be the case and thus, it is preferred to search the for the best combination of parameters in parallel. However, this has some computationally drawbacks. If we simply combine the four search spaces into a 4-dimensional search space, the number of combinations will be above 10^7. The resulting distance matrix calculated using the squared-exponential kernel will when be infeasible to calculate and invert on most laptops. Instead, I will use the knowledge of the previous experiments as well as create a more coarse-grained search space. Thus, the 4D search space is spanned by the learning rate space parameterized by the discrete space $\hat{\mathcal{X}}_\beta = \{10, 11, ..., 50\}$ into $\mathcal{X}_\beta$ by $\mathcal{F}_\beta(x) = 0.1^{x/20}$, for the number of leaves I set $\mathcal{X}_{\#leaves} = \{20, 40, ..., 240\}$, for the feature fraction I set $\mathcal{X}_{frac} = \{.5, .6, ..., .9\}$, and for the L2 regularization I set $\mathcal{X}_{l2} = \{10^{-1}, 10^{-2}, ..., 10^{-5}\}$. This give around 10^4 combinations. Again, I use the probability of improvement to acquire the next data point. I take 300 steps with BO and when evaluate the average SRC. The results are seen in Table 4. The parameter-values are quit different from the values found through the sequential optimization. The avg. SRC is slightly higher with a value at 0.434. This suggests that the parameters are coupled and then you change one, you should adjust the others.

Avg. SRC	β^*	#leaves	fraction	l2
.434 (±.012)	.0063	200	.8	.1

Table 4: Average SRC and optimal parameters.

5. CONCLUSION

Firstly, I have optimized the learning rate for the LightGBM and achieved a SRC at 0.430 using default values for the other parameters. It is shown that the acquisition function *probability of improvement* gives the best result. Secondly, the sequential update of three regularization parameters increase the performance to 0.432. Lastly, I show that the parallel update of all four hyper-parameters yield the highest SRC at 0.434. In conclusion, this indicates that the hyper-parameters are coupled. Thus, in this case, it was better to use a parallel approach with a high-dimensional, coarse-grained search space rather than a sequential approach with a low-dimensional, fine-grained search spaces. The code can be found at `https://github.com/CoRiis/bayesian-optimization-lightgbm`.

6. REFERENCES

[1] Jerome H. Friedman, “Greedy Function Approximation: a Gradient Boosting Machine,” *Annals of Statistics*, , no. 2, 2001.

[2] Federico Bergamin and Kristoffer H. Madsen, “Bayesian optimization,” *Technical University of Denmark*, 2020.

[3] Guolin Ke et. al., “LightGBM: A highly efficient gradient boosting decision tree,” *Advances in Neural Information Processing Systems*, vol. 2017-Decem, no. Nips, pp. 3147–3155, 2017.