TAP - Matheuristic - Hyper-parameters Bayesian Optimization

Alexandre Chanson

2021

1 Relations between parameters and instance

We consider the case of a VPLS matheuristic for solving TAP which is based on the hamming distance.

1.1 Definitions

Let I be a TAP instance, it is tuple $< n, V, T, C, \epsilon_t, \epsilon_d >$ where n is the number of query, V a vector of length n storing the interestingness values of each query, T a vector of execution time for each query, C an n square matrix representing distance between queries. Finally ϵ_t and ϵ_d respectively the maximum time and distance budget.

Let h be the maximum hamming distance to the previous solution, m_{iter} the maximum number of iteration and t_{iter} the time limit for one iteration. Additionally a hard constraint η is given by the user on the total run time per instance, in our case we set $\eta = 10 minutes$.

1.2 Properties

- $1 < h < n$
- if h increases then the local search space grows
- if h increases then t_{iter} must increase
- $m_{iter} \times t_{iter} <= \eta$
- experiments have shown that if t_{iter} grows then m_{iter} must decrease

We are looking to introduce information about the problem instance when computing the ideal hyper-parameters for the Matheuristic. Intuitively the distribution of values in the instances such as distances and interestingness must affect properties of the solution and by extent the difficulty of solving it.

Further we assume that interest, distances and query execution times are bounded by constants, $0 < v_i < 1$, $0 \leq c_{i,j} <$ and $0 < t_i < \theta$. Thus it is possible to use similar equal width histograms to convey information about the instance regardless of the particular instance. Let β be the number of bins used for the histograms, and $\mu_{X,i}(I)$ the relative frequency for the i^{th} bin of property X for instance I, for example $\mu_{V,0}(I)$ represents the relative frequency of the queries which interestingness are comprised between 0 and $\frac{1}{\beta}$ in the instance I.

$\forall I \in \mathcal{I}$ there are 3 functions $f_1 : \mathcal{I} \to [\![1, n]\!]$, $f_2 : \mathcal{I} \times [\![1, n]\!] \to \mathbb{N}$, $f_3 : \mathbb{N} \times [\![1, n]\!] \to \mathbb{N}$ such that:

$$h = f_1(I) \tag{1}$$

$$t_{iter} = f_2(I, h) \tag{2}$$

$$m_{iter} = f_3(\eta, t_{iter}) = E(\frac{\eta}{t_{iter}}) \tag{3}$$

Whereas f_3 can be inferred from the properties of the problem however this is not the case for f_1 and f_2. In a first series of experiments we will assume a linear combination of their input is adequate.

$$\begin{aligned} f_1(I) =& a_1 \cdot n(I) + a_2 \cdot \epsilon_d(I) + a_3 \cdot \epsilon_t(I) \\ &+ \sum_{i=4}^{4+\beta} a_i \cdot \mu_{V,i}(I) + \sum_{i=4+\beta}^{4+2\beta} a_i \cdot \mu_{T,i}(I) + \sum_{i=4+2\beta}^{4+3\beta} a_i \cdot \mu_{C,i}(I) \end{aligned} \tag{4}$$

$$\begin{aligned} f_2(I, h) =& b_1 f_1(I) + b_2 I.n + b_3 I.\epsilon_d + b_4 I.\epsilon_t \\ &+ \sum_{i=5}^{5+\beta} b_i \cdot \mu_{V,i}(I) + \sum_{i=5+\beta}^{5+2\beta} b_i \cdot \mu_{T,i}(I) + \sum_{i=5+2\beta}^{5+3\beta} b_i \cdot \mu_{C,i}(I) \end{aligned} \tag{5}$$

With $a_1, a_2, a_3, ..., a_{4+3\beta}, b_1, ..., b_{5+3\beta} \in \mathbb{R}$

Given the previous formulation we define p the parameters of the matheuristic as the tuple $p =< a_1, a_2, a_3, ..., a_{4+3\beta}, b_1, ..., b_{5+3\beta} >$ valid under a given constant η .

2 Bayesian optimization

Typically Bayesian hyper-parameter optimization is used to find the values of a ML model such that the error on a validation set is minimal. The term "error" can vary depending on the task the model is trained to perform regression, classification, multi class classification. The only characteristics of the error function f_e is that it takes as parameter the trained model and a validation set and returns a real value. Furthermore it is a so called black box function that can only be evaluated (no gradient). As the evaluation of such error function requires heavy computation (model training as an example) an approximation of the error function is needed this is called the surrogate function f_s it is constructed from sampling values of f_e. This is commonly done using a Gaussian Process regression which is updated with each new sample of f_e

2.1 Definitions

We construct a set of instances $\mathcal{I}_e$ representative of real instances, for each instance $I \in \mathcal{I}_e$ we dispose of z_I^* the value of the objective function for an optimal solution to I. Let p be a tuple describing the parameters of the matheuristic $\mathcal{M}$, P is the set of all tuples p representing valid parameters. Let $\mathcal{M} : \mathcal{I} \times P \to R^+$ be the matheuristic, it maps a instance of TAP and a set of parameters to to an objective value. The error function for our problem is defined by Eq. 6 it quantifies the average relative deviation to the optimal solution's objective of our matheuristic.

$$f_e(p) = \frac{1}{|\mathcal{I}_e|} \sum_{I \in \mathcal{I}_e} \frac{z_I^* - \mathcal{M}(I, p)}{z_I^*} \tag{6}$$

2.2 Process

Initialization: a small set of point from the search space P is drawn randomly and sampled (We evaluate f_e for those points). We train a model f_s (ie : Gaussian process regressor or GPR) by maximizing the likelihood that it's parameters would have generated the sample).

We must select the next point to sample so that it should improve the value of f_e since f_e is not differentiable we look at f_s and find the point p_n in P such that the probability of improvement is maximal. We evaluate $f_e(p_n)$. This yields an additional point in our sample set, thus we retrain the GPR.

This greedy sampling phase repeats until a convergence criterion is reached.