Hyper Parameter Optimization using RBF Surrogate Models

Tommaso Aldinucci
tommaso.aldinucci@stud.unifi.it

Abstract

In this submission it is presented a simple application of surrogate models related to hyper parameter tuning. We developed and trained a simple Neural Network on a binary classification problem, then we implemented a surrogate model based on RBF interpolation which was used to iteratively find new aspirant hyper parameters.

1 Introduction

During the last 30 years Machine Learning has been increasing its popularity as one of the most effective field in Computer Science, improving the complexity of its tasks, thanks to the growing of the GPUs power. However the NN models, given their expressiveness, require the tuning of some (or several) paramenters which can optimize the learning strategy, achieving better results using the same model architecture.

These hyper parameters can make quite a difference during the train process as the same model, trained with different choices of learning rate, momentum etc, can lead to dissimilarities in accuracy.

Therefore it is clearly required a method such that it is able to find new appropriate aspirants for the hyper parameters. As the complexity of the model grows, the cost for the evaluation of a single training session can be very expensive. For this reason this method should be smart enough to suggest new parameters that potentially could improve the accuracy of the NN model. In literature there are several methods for the optimization of hyper parameters such as *grid-search, random-search* or *Bayesian optimization*. Unlike *Bayesian optimization* which uses a probabilistic surrogate model to maximize the probability that the NN model achieves a known score given the hyper parameters, we use a different strategy to explore the space of the parameters.

2 Problem Definition

Let:

- $T \in \mathbb{N}$ a fixed number of epochs for the training;
- $w \in \mathbb{R}^n$ weights of the NN
- $f(w)_T$ the value of the objective with w weights and T epochs (validation loss for example)

We hope our weights will be as near as possible to:

$$\hat{w} \in \arg\min_{w} f(w)_T \tag{1}$$

Since at the end of the learning the model hopefully achieves a good $w^* \approx \hat{w}$, we can treat w^* as a costant and define the hyper parameter optimization problem as:

$$\min_{h} f(h)_{w^*,T} \qquad\qquad h \in \mathbb{R}^n \tag{2}$$

In practice we constrain h to be $\in X \subset R^n$ as we know meaningful ranges for the hyper parameters, getting a *box-constraints* problem.

3 Building the surrogate

Given few points $h_i,\ i = 1, ...k$, hyper parameters already tested, we want to define a surrogate function s such that $s(h_i) = f(h_i)\ \ \forall\ i = 1, ...k$. In this implementation we use Radial Basis Function i.e. we choose $s(h)$ in the form:

$$s(h) := \sum_{j=1}^{k} \lambda_j \phi(||h - h_j||) \tag{3}$$

So using the set of hyper parameters as nodes for the interpolation.

There are several possible choices for ϕ such as *cubic*, *thin splate spline* or *gaussian.* We know that choosing ϕ as gaussian the system:

$$\begin{gathered} \Phi_i^T \lambda = f_i \qquad\qquad \forall\ i = 1, ...k \\ \\ \Phi_{i,j} := \exp(-\gamma||h_i - h_j||^2) \end{gathered} \tag{4}$$

admits a unique solution because of the positive definiteness of Φ, avoiding the use of a polynomial as guarantee for the interpolation. Using RBF we iteratively search for new hyper parameters increasing the accuracy of the surrogate related to the expensive function.

4 Choosing new hyper parameters

Aware about the expensive cost for a call to the objective, we need to define a merit function which can suggest new points potentially better than the previous ones. We exploit the concept of ”bumpiness” as a way to define new aspirants by parsimony i.e, given an aspiration level for the objective $\hat{f}$ that we choose as the minimum of the surrogate minus ϵ, we choose as new point $h_{k+1} = \hat{h}$ the one which minimizes the bumpiness function :

$$\lambda^T \Phi \lambda \tag{5}$$

where Φ and λ are related to the surrogate built on the previous set of hyper parameters adding $(\hat{h}, \hat{f})$. For the implementation we use an equivalent formulation of the problem, defining the bumpiness as:

$$Bump(\hat{h}) = (\hat{f} - s(\hat{h}))^2\ \ \det(\Phi)/\det \begin{pmatrix} \Phi & \phi_{k+1} \\ \phi_{k+1}^T & 1 \end{pmatrix} \tag{6}$$

So we want to solve the global optimization problem:

$$\min_{\hat{h}} Bump(\hat{h}) \qquad\qquad \hat{h} \in X \subset \mathbb{R}^n \tag{7}$$

with X compact. The solution of this optimization problem gives a new point from which the objective should be called.

5 Minimizing the bumpiness

Solving the bumpiness problem requires the choice of a suitable global optimization algorithm. Actually we are satisfied with a rough approximation of the global optimum so we make a few iterations of the algorithm.

This implementation used Differential Evolution with a memetic strategy including a phase of local search with L-BFGS-B, a limited memory version of BFGS for *box-constraints* problems.

Differential Evolution is a population based algorithm which samples a number of points p from the feasible set and evolves each of them through a linear combination of other individuals. Once we generated the trial point, we project it on the feasible set then we proceed with the local search using L-BFGS-B. We report the algorithm for clarity:

```
Data: $F \in (0, 2)$; $CR \in [0, 1]$ probability threshold, $X \subset \mathbb{R}^n$ feasible set
Sample uniformly $x_i$,  $i = 1, ...p$  from $X$ all different.
foreach $i \in \{1, ..., p\}$ do
    $\bar{i} := \mathcal{U}(1, ..., n)$;
    Sample $k_1, k_2, k_3 \in \{1, ..., p\} \setminus \{i\}$, all different;
    $\mathbf{trial} := \mathbf{x}_{k_1} + F(\mathbf{x}_{k_2} - \mathbf{x}_{k_3})$;
    for $j \in \{1, ..., n\}$ , $j \neq \bar{i}$ do
        if $\mathcal{U}(0, 1) < CR$ then
            $\mathbf{trial}^{(j)} = \mathbf{x}_i^{(j)}$;
        end
    end
    $\mathbf{trial} =$ project($\mathbf{trial}, X$);
    $\mathbf{trial} =$ L-BFGS-B($f, X, \mathbf{trial}$);
    if $f(\boldsymbol{trial}) < f(\boldsymbol{x}_i)$ then
        $\mathbf{x}_i = \mathbf{trial}$ ;
    end
end
```

Algorithm 1: Memetic Differential Evolution

6 Implementation and Results

For a practical analysis of RBF surrogate models, we trained a simple NN on a binary classification problem[1]. It has been useful to rescale the value of hyper parameters for the surrogate model to avoid numerical issues such as bad conditioning of the Φ matrix. We have optimized respect to the learning rate, momentum, decay, l_1-regularizer and a dropout (trying to avoid overfitting because of the few data).

[1] https://www.kaggle.com/adx891/sonar-data-set

We report the main routine of the whole algorithm:

```
Data: $H$ the set of already know hyper parameters $\{(h_i, f(h_i)),\ \ i = 1, ...k\}$ , $f(h)$
      objective function

while A stopping rule is false do
    Build the surrogate model related to $H$;
    $\hat{f}_j$ = Get the minimum of the surrogate problem $-\epsilon$;
    if $\mathcal{U}(0,1) < 0.25$ then
        $\hat{f}_j = -\infty$
    end
    $\hat{h}_j$ = Solve the bumpiness problem;
    $H = H \cup (\hat{h}_j, f(\hat{h}_j))$ ;
end
```

Algorithm 2: General Routine

For the experiment we used the following parameters:

- 20 iterations of the main algorithm (testing 20 new hyper parameters)
- $\gamma = 1$ for the RBF fitting
- 100 epochs for the training of the NN
- In MDE we used population of 10 points evolved with 20 generations
- $\epsilon = 1e - 03$ and with probability 0.25 we set $\hat{f}$ to $-\infty$ for a better exploration

Using the parameters illustrated above, we were able to discover new hyper parameters which turned out to be better than the initial ones.

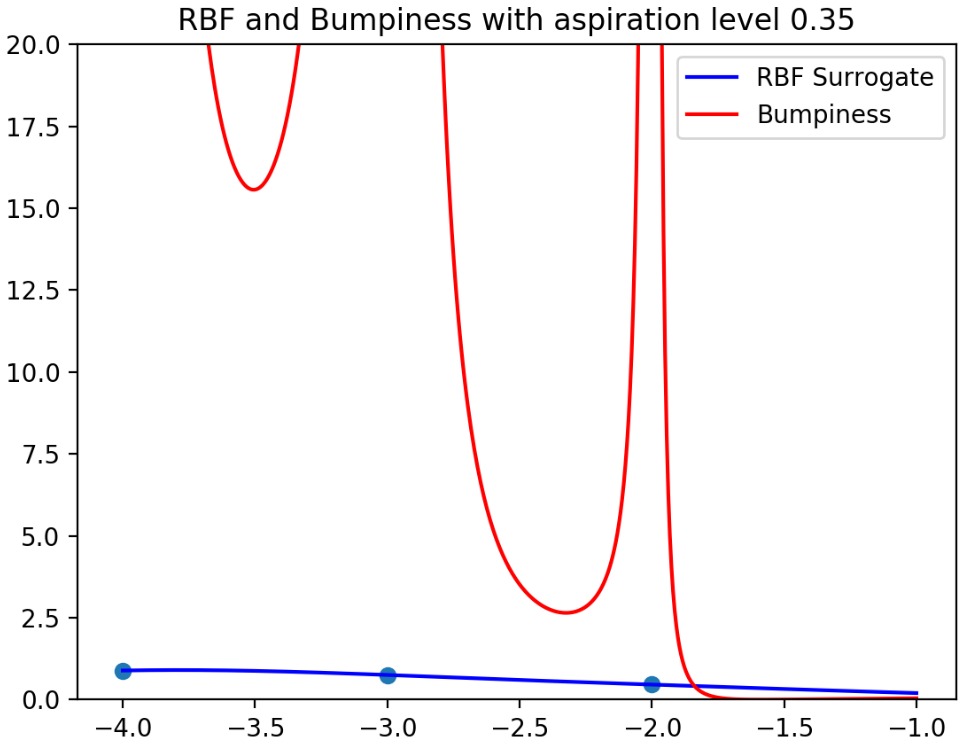

Figure 1: A visualization optimizing only the learning rate

In particular we improved the validation loss from 0.55 to 0.45 proving the effectiveness of this method.

7 Conclusion

RBF surrogate models are a valid choice for hyper parameter optimization. We were able to implement a simple version of the method and testing it on a binary classification problem verifying its effectiveness.