

A Successive Quadratic Approximation Approach for Tuning Parameters in a Previously Proposed Regression Algorithm

Patrick Hosein^a, Kris Manohar and Ken Manohar^b

Department of Computer Science, The University of the West Indies, St. Augustine, Trinidad

Keywords: Regression, Parameter Tuning, Convex Optimization, Machine Learning.

Abstract: We investigate a previously proposed regression algorithm that provides excellent performance but requires significant computing resources for parameter optimization. We summarize this previously proposed algorithm and introduce an efficient approach for parameter tuning. The speedup provided by this optimization approach is illustrated over a wide range of examples. This speedup in parameter tuning increases the practicality of the proposed regression algorithm.

1 INTRODUCTION

Parameter optimization or tuning is the process of evaluating potential parameter values to identify the optimal value for the related objective function. This optimal value results in an output value for the objective function that is either the minimum output if it is a cost function or the maximum output if it is a utility function. Parameter optimization can be a time-consuming and even a manual process. However, it is important to invest the time to complete this and achieve the best possible performance.

The proposed regression algorithm in (Hosein, 2023) addresses the bias-variance trade-off and determines the optimal trade-off between personalization and robustness and illustrates the approach using an automobile insurance claims dataset. This algorithm depends on a tuning parameter κ with an objective function of Mean Square Error (MSE). As MSE is a cost function, the optimal value of κ will result in the minimum MSE value.

The relationship between MSE and κ was observed to be a nonlinear function with a consistent shape. The function is initially convex but then becomes concave and approaches a limiting value from below. Therefore, the optimal value of κ that minimizes the MSE lies in the convex region but note that this is not a pure convex optimization problem.


Gradient descent is a popular optimization algorithm. It can iterate and move along the direction of


the steepest negative gradient until the minimum point is reached. However, it may get stuck at saddle points and it introduces additional hyper-parameter tuning for its learning rate. Newton's method can potentially converge faster than gradient descent but this requires an analytic function for which the second derivative exists and can be computed. For our problem this is not the case.

Quadratic approximation involves approximating a nonlinear function with a simpler quadratic function. Successive quadratic approximation (SQA) iteratively approximates an objective function with a quadratic function and then solves to find the minimum point of the quadratic function. On each iteration, the minimum point of the quadratic function move towards converging with the minimum point of the objective function. Given that a single parameter is to be optimized, i.e. κ , and the observed shape properties of the function, we use a successive quadratic approximation approach which we describe in detail later.

2 PREVIOUS WORK AND CONTRIBUTIONS

Since the previously proposed regression algorithm is relatively new there are no publications on the parameter optimization aspect which is our focus. There are papers on parameter optimization for regression as for example the work done in (Wang et al., 2021). However their focus is on developing a tuning-free

^a  <https://orcid.org/0000-0003-1729-559X>

^b  <https://orcid.org/0000-0001-9079-2475>

Huber Regression approach whereas we are looking at a completely different regression approach which we describe later.

In the paper (Stuke et al., 2020) the authors assess three different hyperparameter selection methods: grid search, random search and an efficient automated optimization technique based on Bayesian optimization. They found that for a large number of hyper-parameters, Bayesian Optimization and random search are more efficient than a grid search. We also illustrate that our proposed hyper-parameter tuning approach is more efficient than a grid search.

Our contribution is an improvement on the parameter optimization method used for the regression algorithm presented in (Hosein, 2023). We demonstrate, through examples, the potential speedup in computation thus increasing the usability of their regression algorithm.

3 PROBLEM DESCRIPTION

Let us first consider the case of a single feature with ordinal independent values (e.g., age). Denote the dependent value of sample i by y_i and the independent value by x_i . Suppose we need to predict the dependent value for a test sample with independent value \hat{x} . One predictor is the average of the dependent variable over all samples with independent value \hat{x} . However there may be none, or few samples, to obtain a robust prediction. We can instead include nearby samples in the prediction (i.e., aggregation) but how “big” should the neighbourhood be and how much weight should we assign to our neighbours. We use the following approach. We take a weighted average of the dependent values over all samples but weight the average based on the distance between the independent values of the test sample and each training sample. In particular we use the following predictor.

$$\hat{y}(\kappa) \equiv \frac{\sum_{s \in \mathbf{S}} \frac{y_s}{(1+d_s)^\kappa}}{\sum_{s \in \mathbf{S}} \frac{1}{(1+d_s)^\kappa}} \quad (1)$$

where $d_s = |\hat{x} - x_s|$, \mathbf{S} is the set of training samples and κ is a hyper-parameter. Note that, if $\kappa = 0$ then the predictor is simply the average taken over dependent values of all samples. If there are one or more samples such that $x_s = \hat{x}$ then as κ goes to infinity then the predictor tends to the average of these samples. The optimal κ typically lies somewhere between these extremes. One can find the optimal value of κ by doing a linear search but that requires a significant amount of computing. We introduce an efficient approach to finding the optimal value of κ .

Note that one can perform a similar computation in the case of categorical data. In this case the distance between two samples is defined as the absolute difference between the average dependent values of the categories of the two samples. In addition, this approach can be extended to multiple features. In this case the distance between two samples is the Euclidean distance based on the single feature distances (with some normalization). However, we again need to optimize over κ . We provide examples for this case as well. More details on the regression algorithm can be found in (Hosein, 2023).

4 AN ILLUSTRATIVE EXAMPLE

We determine the optimal κ as follows. For a given κ we use K -Fold validation to determine the resulting Normalized Mean Square Error. This is the MSE divided by the MSE obtained if the predictor was just the average over all samples of the dependent variables (in the training set). Hence we obtain a value of 1 at $\kappa = 0$. We then find the value of κ that minimizes this NMSE.

Let $E(\kappa)$ represent the Normalized Mean Square Error given a parameter value κ . In practice, we have found that this function has the shape illustrated in Figure 1. The function is convex to the left of the dashed line and concave and increasing to the right. The minimum point lies within the convex region. We can summarise the proposed optimization approach as follows. Starting with any three points on the curve we determine the quadratic function passing through these points. If this quadratic function is convex then we find the minimum point and replace the maximum of the previous three points with this new point. If, however, this quadratic function is concave then we know that the minimum lies to the left of the point with the smallest κ value. In this case we replace the three points with (1) the point with the lowest κ value, (2) the point at $\kappa = 0$ and (3) the point midway between these two. The quadratic function through these three points is guaranteed to be convex and so we can continue the process. This ensures that we gradually move to the convex region and, once there, we converge to the minimum value. Pseudo-code for this algorithm is provided in Algorithm 1.

Algorithm 1: Pseudo-code for Quadratic approximation Algorithm.

```

1:  $E(\kappa) \equiv$  normalized mean square error for  $\kappa$ 
2: If  $E(0.5) \geq 1$  then  $\kappa^* < 0.5$  so EXIT (feature is not useful)
3: Set  $\kappa_1 = 0, \kappa_2 = 10, \kappa_3 = 20, \kappa^* = \kappa_{\arg \min_i \{E(\kappa_i)\}}$ 
4: Set  $\varepsilon = 1 + \nu$  ( $\nu$  determines the level of accuracy obtained)
5: while ( $\varepsilon > \nu$ ) do
6:    $(a, b, c) \leftarrow$  coefficients of quadratic through points at  $\kappa_1, \kappa_2$  and  $\kappa_3$ 
7:   if ( $a \leq 0$ ) then
8:      $\kappa_3 = \min_i \{\kappa_i\}, \kappa_2 = \frac{\kappa_3}{2}, \kappa_1 = 0$ 
9:   else
10:     $\kappa_{\arg \max_i \{E(\kappa_i)\}} \leftarrow \frac{-2a}{b}$ 
11:   end if
12:    $\varepsilon = |\kappa^* - \kappa_{\arg \min_i \{E(\kappa_i)\}}|$ 
13:    $\kappa^* \leftarrow \kappa_{\arg \min_i \{E(\kappa_i)\}}$ 
14: end while
15: Return  $\kappa^*$ 

```

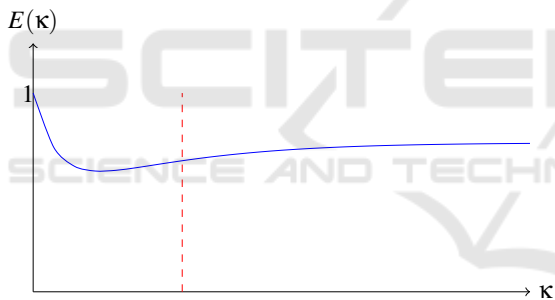


Figure 1: Typical Shape of the $E(\kappa)$ Function.

5 NUMERICAL EXAMPLES

5.1 Computation Comparison

In this section we provide some numerical examples to compare the number of evaluations required before we obtain an acceptable value for κ . We compare this with the number of iterations required for a linear search. Suppose our stopping criterion is when $\varepsilon \leq \nu$. If we were to do a grid search then, in order to obtain similar performance, the grid search spacing must be at least ν . We therefore must start at $\kappa = 0$ and increase κ by at most ν in each iteration. If the resulting optimal value is κ^* then once we reach this point the NMSE will start increasing and we can then stop. Therefore the number of iterations required is

approximately $\left\lceil \frac{\kappa^*}{\nu} \right\rceil$ and hence we can use this to determine the number of iterations required with a linear search.

5.2 Insurance Risk Assessment

In this section we use the dataset that was used in (Hosein, 2023) to illustrate the potential computation savings. We used six features from the Comprehensive policies of the dataset and a threshold of $\nu = 0.01$. In Table 1 we provide the values for each step of the process. In Step 0 we initialize the first 3 points. Note that we only need to evaluate the function for $\kappa = 10$ and $\kappa = 20$ since we know the value at $\kappa = 0$ is 1. The minimum point for the associated quadratic function occurs at $\kappa = 14.88$ and so we replace the point at zero (which has a value greater than the other two points) with this point. However, these three points now form a concave function and so in Step 2 we choose κ values of (0, 5, 10). In all remaining steps the three points form a convex quadratic and we repeat until there is very little change in the best (so far) value of κ .

After 9 iterations we converge for the chosen threshold of $\nu = 0.01$. A linear search would have taken approximately $9.20/0.01 = 920$ iterations. In other words the computation time of the proposed approach would be approximately 1% of the time for a linear search. If we had instead set $\nu = 0.1$ then the

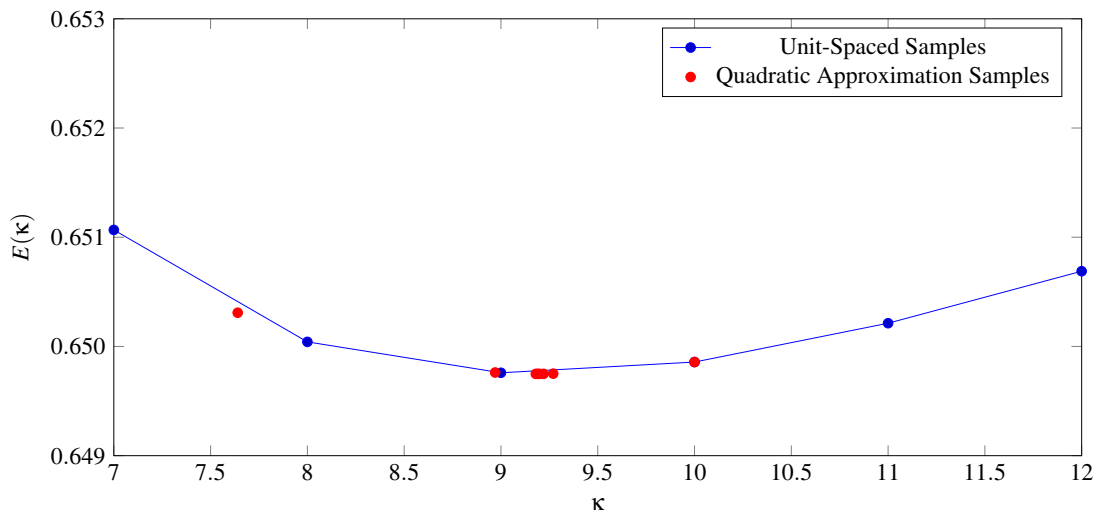


Figure 2: $E(\kappa)$ as a function of κ for Selected Features.

speed up would have been $(9.2/0.1)/6 = 15$. In Figure 2 we plot the minimum values for each iteration but only for points near the optimum. We find that, after 4 iterations, we already have a reasonable result (at $\kappa = 8.97$) and for most practical purposes this result is sufficient.

Table 1: Summary of Steps in the Algorithm.

Step	$(\kappa_1, \kappa_2, \kappa_3)$	κ^*	$E(\kappa^*)$
0	(0, 10, 20)	10.0	0.6498572510787386
1	(10, 14.88, 20)	10.0	0.6498572510787386
2	(0, 5, 10)	10.0	0.6498572510787386
3	(5, 7.64, 10)	10.0	0.6498572510787386
4	(7.64, 8.97, 10)	8.97	0.6497613579708842
5	(8.97, 9.27, 10)	9.27	0.6497511801524423
6	(8.97, 9.22, 9.27)	9.22	0.6497490840222766
7	(9.18, 9.22, 9.27)	9.18	0.6497489707751954
8	(9.18, 9.22, 9.20)	9.20	0.6497489687581092
9	(9.18, 9.19, 9.20)	9.19	0.6497489655206691
10	(9.19, 9.19, 9.20)	9.19	0.6497489655206691

6 ADDITIONAL NUMERICAL EXAMPLES

We ran several additional examples and determined the speedup provided for each of them for different levels of κ accuracy. The datasets for these examples were taken from the UCI Machine Learning Repository (Dua and Graff, 2017). The details of these examples are as follows:

1. **Single Ordinal Feature.** We used the *age* feature of the student-math dataset (Cortez, 2014).
2. **Single Categorical Feature.** We used the *internet* feature of the student-math dataset (Cortez, 2014).
3. **Mix of Categorical and Ordinal Features.** We used the *internet*, *age*, *health*, and *absences* features of the student-math dataset (Cortez, 2014).
4. **Multiple Ordinal Features.** We used the cylinders, displacement, hp, weight, acceleration and year features of the auto-mpg dataset (Quinlan, 1993).

The speedup is the ratio of the number of iterations required for a linear search to the number required for the proposed approach. These are provided in Table 2 for different values of v . We note that as we increase the level of accuracy the proposed approach provides increased speedup. Also, as the number of features increases, the speedup also increases. In practice a problem may have tens of features and hence we expect even better performance for such cases.

Table 2: Speed up values for different levels of accuracy.

Example	$v = 0.1$	$v = 0.01$	$v = 0.001$
1	17	177	1776
2	12	123	1233
3	42	426	4260
4	91	918	9188

7 CONCLUSION AND FUTURE WORK

We described a regression approach that was previously proposed in the literature but which required significant computational resources. We introduce an approach to speed up parameter optimization and showed that it could result in as much as two orders of magnitude decrease in computation. In the future we plan to further improve this optimization approach, provide additional examples and provide a proof of convergence.

REFERENCES

- Cortez, P. (2014). Student Performance. UCI Machine Learning Repository.
- Dua, D. and Graff, C. (2017). UCI machine learning repository.
- Hosein, P. (2023). A data science approach to risk assessment for automobile insurance policies. *International Journal of Data Science and Analytics*, pages 1–12.
- Quinlan, R. (1993). Auto MPG. UCI Machine Learning Repository.
- Stuke, A., Rinke, P., and Todorović, M. (2020). Efficient hyperparameter tuning for kernel ridge regression with bayesian optimization. *Machine Learning: Science and Technology*, 2.
- Wang, L., Zheng, C., Zhou, W.-X., and Zhou, W.-X. (2021). A new principle for tuning-free huber regression. *Statistica Sinica*.