Bayesian optimization for selecting training and validation data for supervised machine learning

David Bergström, Mattias Tiger, and Fredrik Heintz

Linköping University, 581 83 Linköping, Sweden
davbe125@student.liu.se, {mattias.tiger, fredrik.heintz}@liu.se

Abstract. Validation and verification of supervised machine learning models is becoming increasingly important as their complexity and range of applications grows. This paper describes an extension to Bayesian optimization which allows for selecting both training and validation data, in cases where data can be generated or calculated as a function of a spatial location.

Keywords: Bayesian optimization · AutoML · supervised learning

1 Introduction

An important problem in supervised machine learning is selecting what data to use for training. More precisely, there are cases where data has to be manually collected and labeled, preprocessed or simulated and thus come at a cost of time and resources. Another important problem is how to evaluate the models once trained. Both these problems are difficult to solve on their own, as the choice of training data affects what evaluation data should be used and vice versa.

In some of these cases data can be generated as a function of a spatial location. Our contribution is a method for using these functions to generate sets of training and evaluation data sets. We apply and adapt Bayesian optimization to jointly select both a training and an evaluation data set.

The selection of a fixed-size training data set is formulated as an optimization problem, where the aim is to find a training data set which yields good overall performance, according to some performance metric, once used as training data for a supervised machine learning model. The evaluation data set is chosen iteratively such that the overall uncertainty of the model's performance over a spatial region is minimized, resulting in an approximation of the model's performance over said region. Finally, the selection of an evaluation data set is used as a performance metric for when choosing the training data set.

Previous work related to applying Bayesian optimization to supervised machine learning have focused on using it for hyperparameter optimization.

2 Bayesian optimization

Bayesian optimization is a state of the art black-box optimization method, which aims to minimize the total number of function evaluations [4]. This makes it suit-

able for functions which are expensive to evaluate, e.g. training supervised machine learning models. A crucial part of the algorithm is the surrogate function, often a Gaussian process, used to model the function which is being optimized. A Gaussian process being a highly flexible non-parametric model, useful for regression [3]. The method is summarized in algorithm 1.

Algorithm 1 Bayesian optimization

Input: Objective function f; iterations N; input space X; acquisition function α
Output: Best estimate x^* of the highest function value

1: **for** $i \leftarrow 1$, N **do**
2: $\quad \mathcal{GP} \leftarrow$ Gaussian process regression with data $\langle x_j, y_j \rangle_{j=1}^{i-1}$
3: $\quad$ Select $x_i \in \arg\max_{x \in X} \alpha(x, \mathcal{GP})$
4: $\quad y_i \leftarrow f(x_i)$
5: **return** $x^* \leftarrow \arg\max_{x_i \in \{x_1, \ldots, x_N\}} y_i$

The aim of supervised machine learning is to model the relationship between a input space X and a output space Y, with the intention of predicting the output when presented with a previously unseen inputs. This is done by analyzing sets of previously collected data points, each point $p \in \mathcal{D}_{\text{train}} \subseteq \mathcal{D} \subset X \times Y$.

A data generating function f_{gen} is defined as a function which takes a spatial position p and generates one or more data points $(x, y) \in X \times Y$. One example of a data generating function is simulators such as CARLA [1], where both color and semantically segmented images can be generated from a position.

In order to adapt Bayesian optimization to make it suitable for selecting data for a supervised machine learning model, a few modifications are needed. First, instead of selecting a single vector x_i to evaluate at step 3, we need to select a fixed-size set of positions, a configuration. Secondly, instead of step 4 we want to generate data by evaluating the data generating function with the configuration, train the model on the resulting data and finally evaluate the resulting model.

2.1 Extending Gaussian processes to sets of points

A Gaussian process can be modified to model functions over fixed-size sets by first concatenating the points into one large vector and then constructing a kernel which is invariant to the order of elements which corresponds to points in the original set. A permutation invariant-kernel is constructed as follows, as described in [2]:

$$k_{\text{PI}}(x, x') = \sum_{g \in G} \sum_{g' \in G} k(g(x), g'(x')), \tag{1}$$

where G is a set of functions and k is some regular kernel, e.g. the squared exponential. Each element of G is a function which permutes the vector in one of the ways in which the kernel should be invariant to.

2.2 Estimating loss

In an ideal scenario the trained model would be evaluated by generating all possible data pairs and calculating the total loss. However, this is not always

feasible as generating data can be both costly and time-consuming. Instead the requirement on the model can be rephrased, using the data generating function f_{gen}, as having low loss over a closed space. The closed space can be chosen to describe the use-case of the model, e.g. where the model will be deployed.

Given that a space Ω has been chosen, the total loss in that area is:

$$\mathcal{L}(\Omega) = \int_{\Omega} [\ell(f(x, \mathcal{M}), y)\, dp, (x, y) \leftarrow f_{\text{gen}}(p)]\, dp = \int_{\Omega} \ell(p, f(., \mathcal{M}))\, dp, \quad (2)$$

where ℓ is a point-wise loss function, e.g. the L^2 norm of the squared difference.

The inner loss function $\ell(p, f(., \mathcal{M}))$ can be approximated by placing a Gaussian process prior on it. Here Bayesian optimization is used for approximating a function, rather than optimizing it. This is achieved by using a pure exploration acquisition function $\alpha_{\text{PE}} = \sigma(x)$, where $\sigma(.)$ is the predicted standard deviation. The intent of α_{PE} is to have the algorithm select points with high uncertainty and thus reduce the overall uncertainty of f, ultimately leading to a good approximation. The total loss can be estimated by estimating the integral over the approximated surface. The approximated loss surface can also be used during run-time, e.g. to provide a certainty measure of the model's predictions.

3 Preliminary results

The aim of this section is to present some initial results of applying the two parts of the proposed method. The two parts are applied to simplified problems with similar properties to selecting data for supervised learning. The simplified problem consists of placing a set of wireless access points such that the Wi-Fi signal strength is high in a small simulated 2D building. The loss for a given configuration of wireless access points is defined as the square of the inverted signal, i.e. $\ell(p, c) = ((s(p, c) + 1)^{-2}$, where $s(p, c)$ is the signal strength for a configuration c evaluated at point p.

The first experiment uses Bayesian optimization with the permutation invariant kernel to place 6 wireless access points, finding the configuration which results in the lowest total signal strength. The result is shown in figure 1.

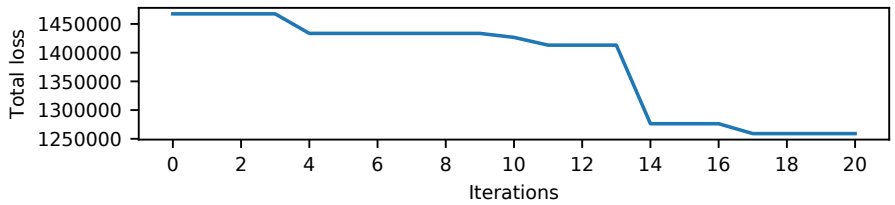

Fig. 1. The loss of the best configuration is shown for each iteration.

The second experiment takes the final configuration from the first experiment and estimates the loss surface. Bayesian optimization is run for 200 iterations using the pure exploration acquisition function. The result is shown in figure 2.

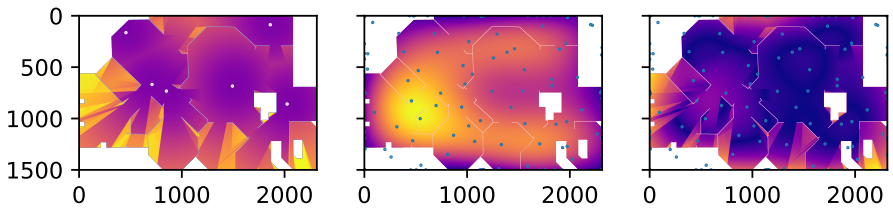

Fig. 2. Estimated loss surface. The left-most figure shows the ground truth loss surface, where white dots mark where access points have been placed. The middle figure shows the estimated loss surface and the right-most shows the element-wise absolute error between the two, both showing blue dots wherever the function have been evaluated.

4 Conclusion

One issue which becomes apparent is that current frameworks for Gaussian processes and Bayesian optimization struggle with cases where there are holes in the domain from which points are chosen. These holes are present whenever there is a locations where it is not possible to collect data, e.g. the location being inside a wall or not having any data for that particular area. This issue is present in both the training data selection and the validation data selection.

In the case of validation data selection, it causes several points to be wasted since they are placed inside walls, which in turn decreases the quality of the approximation. Points outside of the valid map are given a loss of 0, which causes low predicted mean to "leak in" from the walls.

The method assumes that the data generating function gives two similar output values given two similar input values. This means that it might generalize to any function with this property. However, this has not been verified.

Future work will focus on using the described method to select data for training and evaluating deep neural networks as well as look into alternative optimization methods to avoid placing points inside walls.

References

1. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V.: CARLA: An Open Urban Driving Simulator. In: Proceedings of the 1st Annual Conference on Robot Learning, pp. 1–16 (2017)
2. Duvenaud, D.: Automatic model construction with Gaussian processes. University of Cambridge (2014).
3. Rasmussen, C.E., and Williams, C.K.: Gaussian processes for machine learning. The MIT Press (2006)
4. Snoek, J., Larochelle, H., and Adams, R.P.: Practical Bayesian Optimization of Machine Learning Algorithms. In: Advances in Neural Information Processing Systems 25, pp. 2951–2959. Curran Associates, Inc.(2012)