**Sequential Model Expansion**

We propose a novel algorithm for sequential model expansion. It achieves low prediction loss with a small computational cost. Suppose that are the observed data, where we use to denote respectively the responses and covariates. In the context of statistical inference or machine learning, we usually postulate a set of candidate models and select one of them by some method such as cross validation, denoted by . Our goal is to minimize the expected predictive loss (given some loss function ), in the sense that the selected is as good as the best offered by the candidate set:

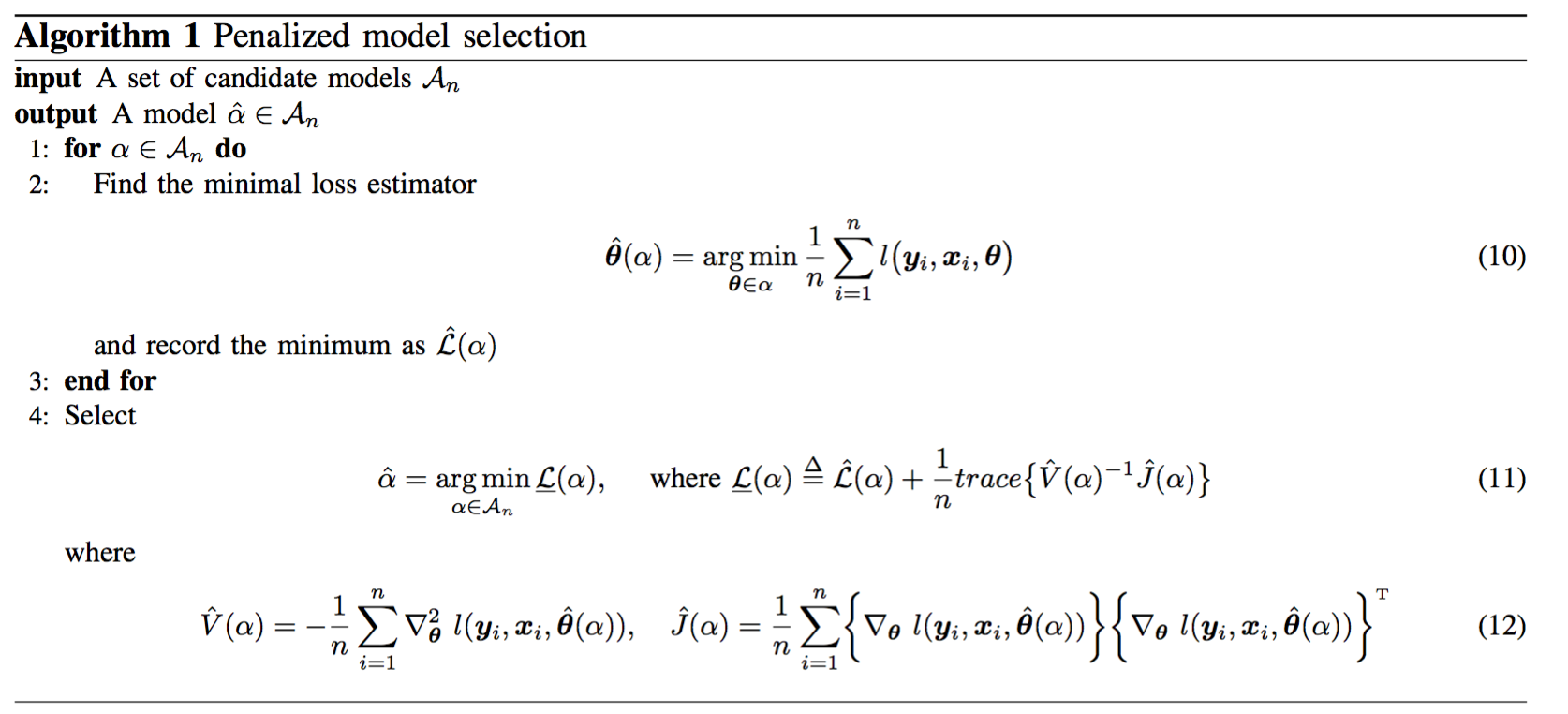
in probability, as ,

where is an estimated parameter under model , and denotes the expectation with respect to the true data generating process. The above desired is called an “efficient” model.

**1. Efficient Selection and Estimation of the Predictive Loss**

In many real applications, data is sequentially observed, and the candidate model class should adapt to the size of data. Thus, we are interested in selecting the efficient model from that may depend on . The classical way of model selection is cross validation. However, it has been shown that under linear regression models, only leave-one-out cross validation (LOOCV) is efficient in a mis-specified model class (presumably “all models are wrong”). However, LOOCV is computationally intensive.

In fact, we can prove under reasonable assumptions that the Algo.1 (referred to as TIC) below selects an efficient , with very low computational cost when the size of is much smaller than and the derivatives of the loss are easy to calculate.

**

In other words, the value in Eq.(11) is also an accurate estimation of the true predictive loss . Consider, for example, linear Gaussian regression models. The predictive loss of a model can be well estimated by its mean squared loss plus a penalty proportional to the dimension.

**2. Directional Expert Tracking**

We have resolved the goal of selecting the efficient model given a fixed number of observations. However, in a sequential setting, the following issues naturally arise:

*Issue 1: Suppose that at time step , we can accurately estimate the predictive loss of each candidate model, thereby selecting an efficient model. However, the efficient model at time step is not necessarily efficient at time step , when an extra (batch of) data samples arrive. Which model shall we use for time step when the extra data have not been revealed?*

*Issue 2: Suppose that we select one model and use it at each time step. The path of the historically selected models may fluctuate drastically (which will be illustrated in Fig. 2). Instead, it is more appealing (either philosophically or computationally) to enforce the selected models to evolve along a continuous path: from a small dimension to a large one.*

In the sequel, we shall use *t* to emphasize the dependence on time. Inspired by learning theory, we will provide an algorithm in next subsection that is “efficient” not only in learning a model but also in computational cost. Specifically, we propose a predictor that has cumulative loss (over time) close to the following optimum benchmark:

(\*)

where the minimization is over all tuples that have at most non-repetitive points and that are restricted to the chain 1-2-3-…, and is a given number. For example, .

Before we proceed, we first propose a new expert tracking technique that is more effective than the state-of-the-art. Recall that in the last monthly report, we borrowed the technique from the fixed-share algorithm which aims to minimize

(\*\*)

where is an expert producing loss at time , is our predictor, , and is the number of ’s such that . However, we hope to minimize

(\*\*\*)

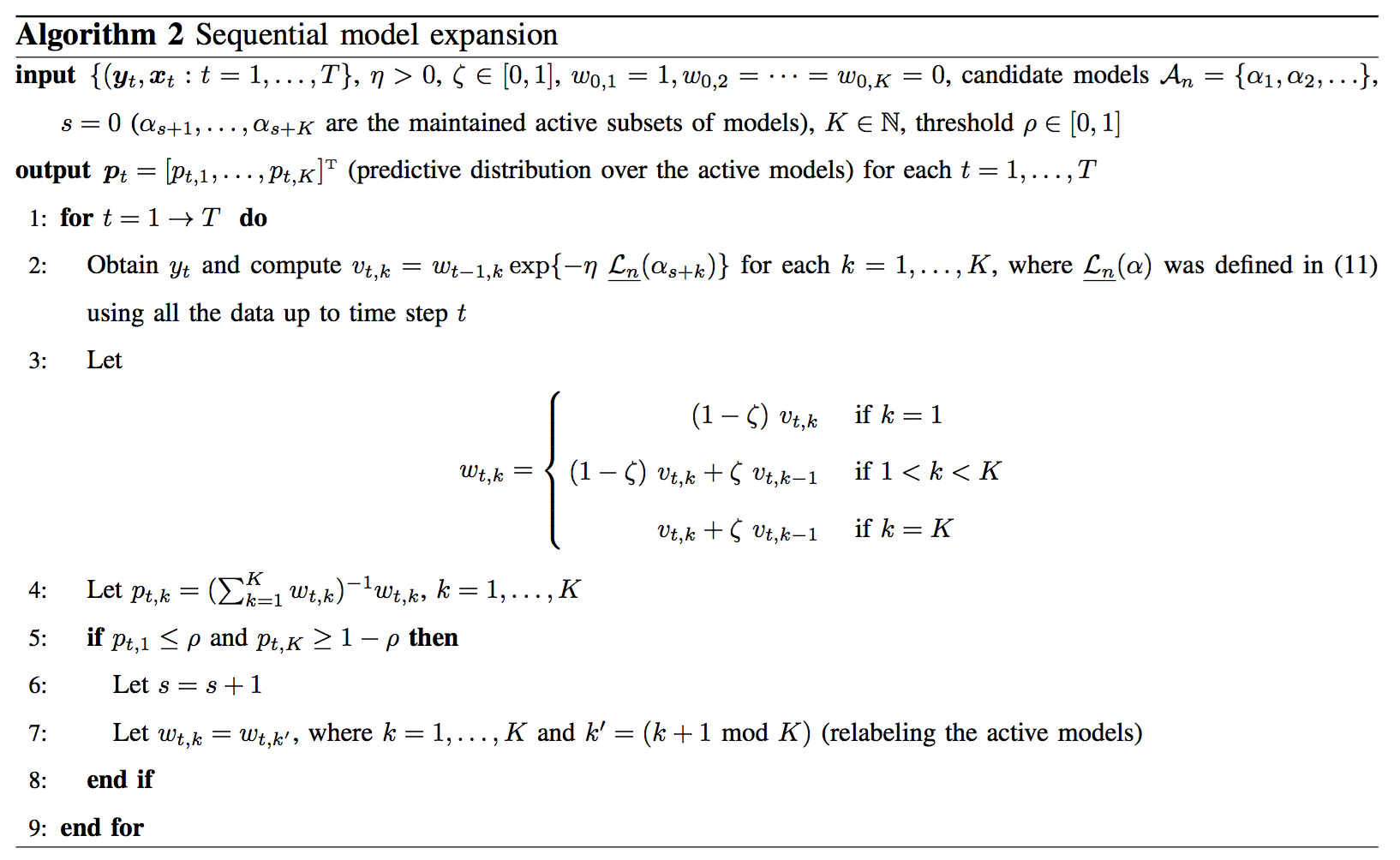
Recall that the state-of-the-art method uses the following weight updating schemes to bound (\*\*): suppose that is the weight of expert at time , then

Intuitively, instead of using the exponentially updated weights directly, a “tax” of rate is imposed on each expert, and then the total tax revenue is redistributed equally among experts. In doing that, the experts are encouraged to rejuvenate their past performance and “start a new life”, so that we can track the best expert in different time epochs. It then gives the natural idea that we redistribute the revenue in a directional way, thus encouraging the experts to switch in a directional path.

**3. Sequential Model Expansion: Algorithm**

Based on the previous discussions, we propose the following algorithm (referred to as Generalized TIC, or GTIC), where each candidate model and its estimated predictive loss are regarded respectively as an expert and loss at each time.

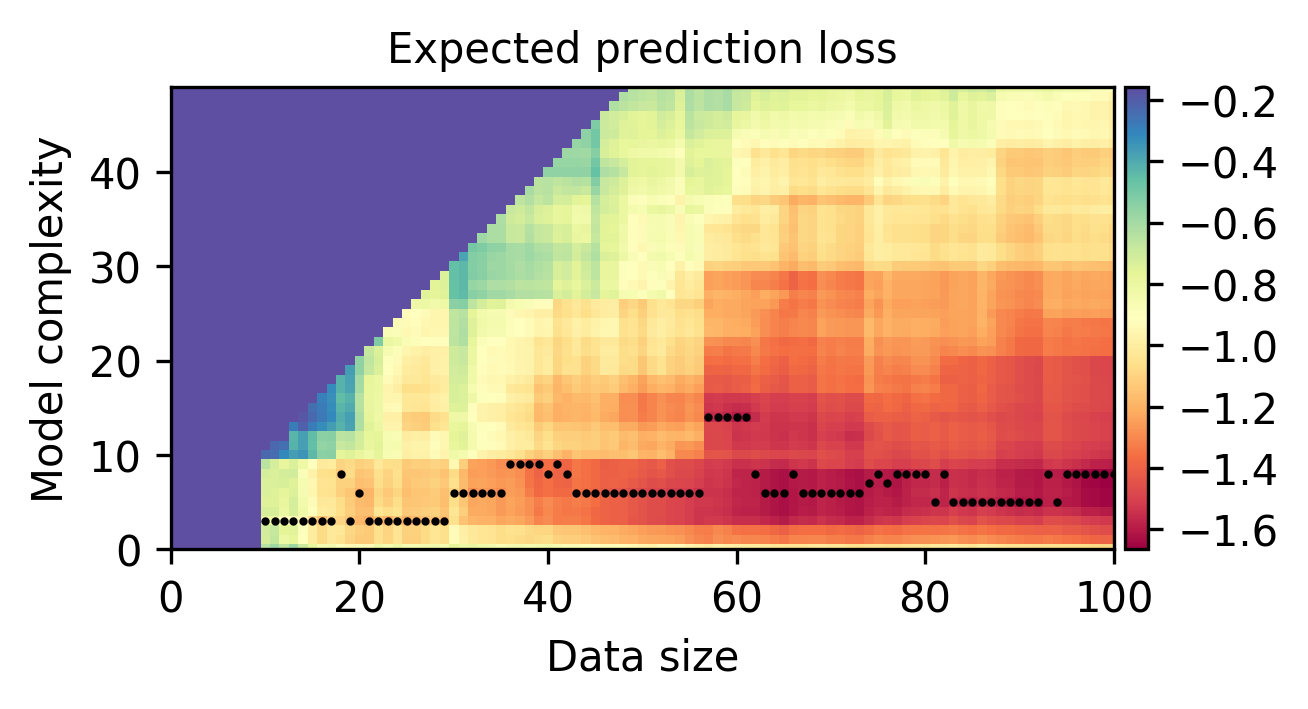
To further reduce the computational cost, we maintain only an active subset (of size K) instead of all the candidate models at each time. The active subset starts from ; it switches to when the weight of the smallest model becomes small and that of the largest model becomes large; it continues to switch upon the aggregation of data. The output from the algorithm is a set of weights associated with each active model, which can be understood as a “posterior distribution over the models” given the current observations. The weights can be further used to predict the next data point at each time.



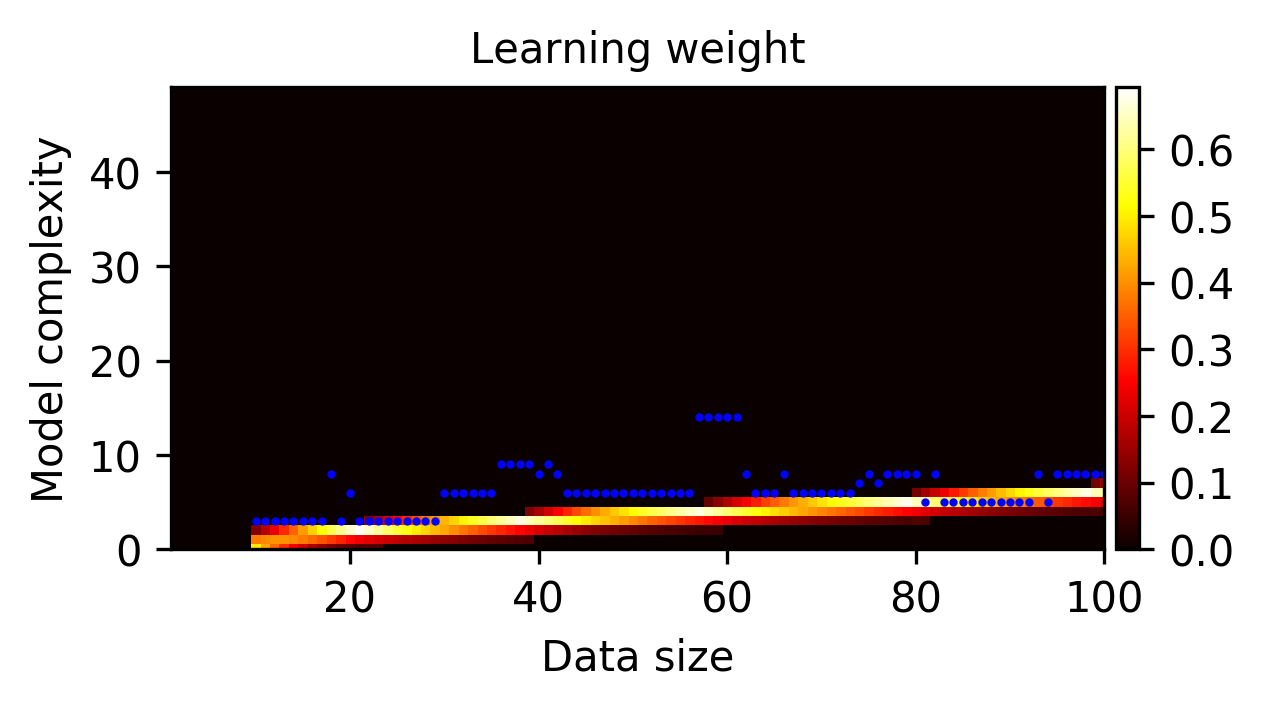
**4. Sequential Model Expansion: Experiments**

In an experiment, we generate a sequence of data from the true model: , where are independent Gaussian random variables of zero mean and unit variance, is a vector. Suppose that we sequentially obtain and learn the data, starting from *,* then. We choose the candidate models to be logistic regression models of dimension =10 for each. Here, a model of dimension means that the first elements of are nonzero.

To illustrate the efficiency of our method (TIC and GTIC), we first simulate model selection results with batch data. We obtain the optimal model (oracle) by testing the trained model on a large dataset and the optimal model is the one with the least loss. In Fig. 3, we compare the performance of GTIC to different types of CV. Holdout takes 70% data for training and tests on 30% data. It fluctuates throughout the experiment, and most of the time it yields the worst performance. GTIC, 10-fold CV and LOO perform well in this experiment. However, both GTIC and 10-fold CV fluctuate a little bit. Our proposed sequential model expansion algorithm smoothly expands the model and yields the best performance compared to all the other approaches. As shown in Fig. 1 and Fig. 2, although the optimal model of each data size is not always identical to the selected model from our model expansion algorithm. The loss of our selected model is almost the same as the optimal model which is exactly what we expected from the efficiency definition. The computation cost of all approaches is provided in Fig. 4. As shown in the figures, under logistic regression, GTIC is slightly better than 10-fold CV but worse than Holdout. Indeed, we need to compute the penalty term in GTIC. Depending on the problem and data, we may need different number of fold for CV in order to have a satisfactory result. However, since GTIC performs almost as well as LOO and 10-fold CV, we suggest using GTIC instead of guessing the optimal number of fold for CV. With GTIC, we do not need to sacrifice much on computation but still have a theoretically justifiable result which is as good as LOO.

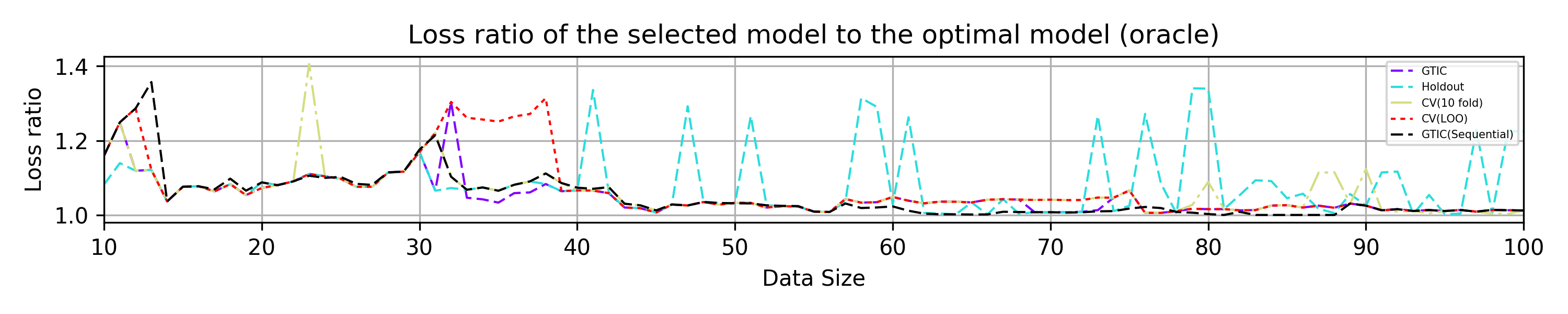


*Figure 1: Heat-map showing the prediction loss of estimated candidate models of each dimension (y-axis) at each data size (x-axis), where the black dots indicate the model of optimal loss at each data size. The true loss is numerically computed from independently generated test data.*

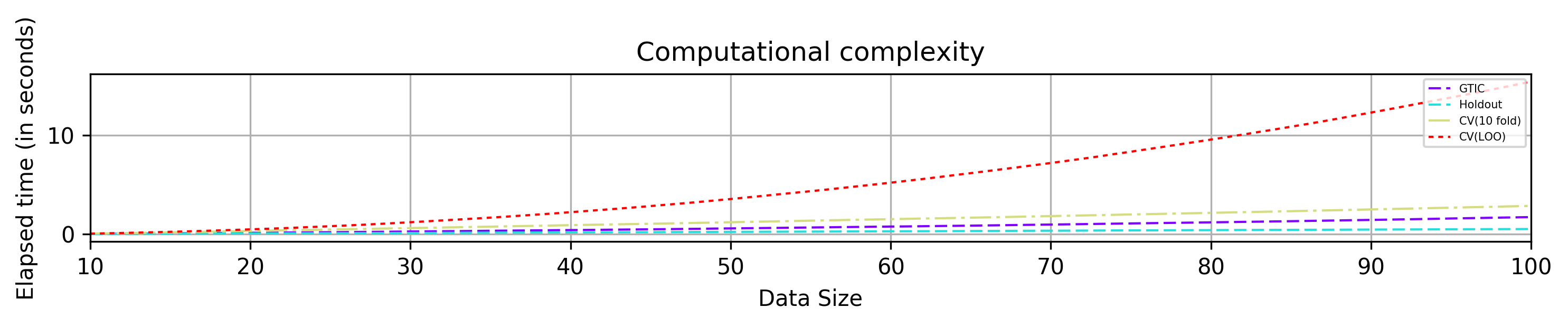


*Figure 2: Heat-map showing our predictive weights over the candidate models (y-axis) at each data*

*size (x-axis).*



*Figure 3: Plot showing the loss of our predictor (GTIC) and cross validations at each data size.*

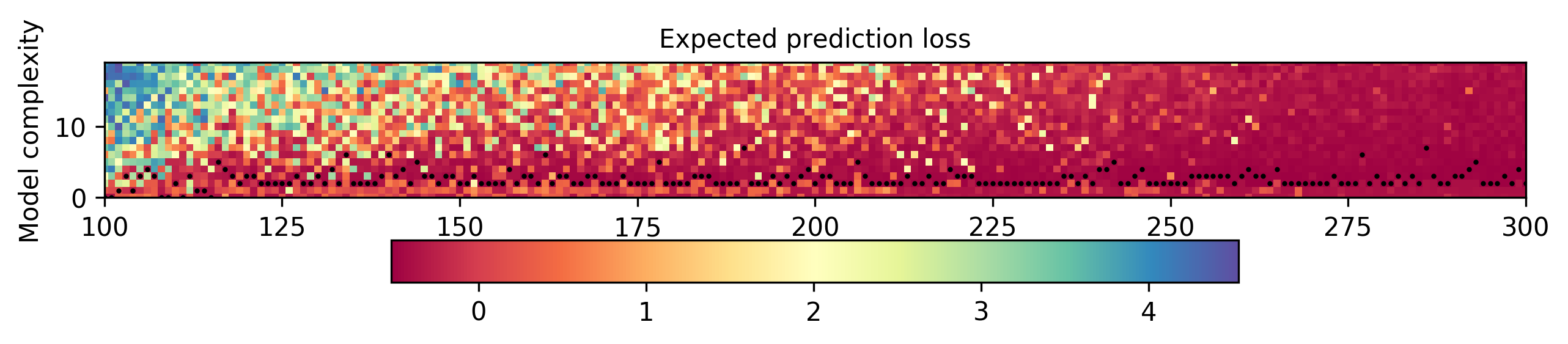


*Figure 4: Plot showing the computational costs.*

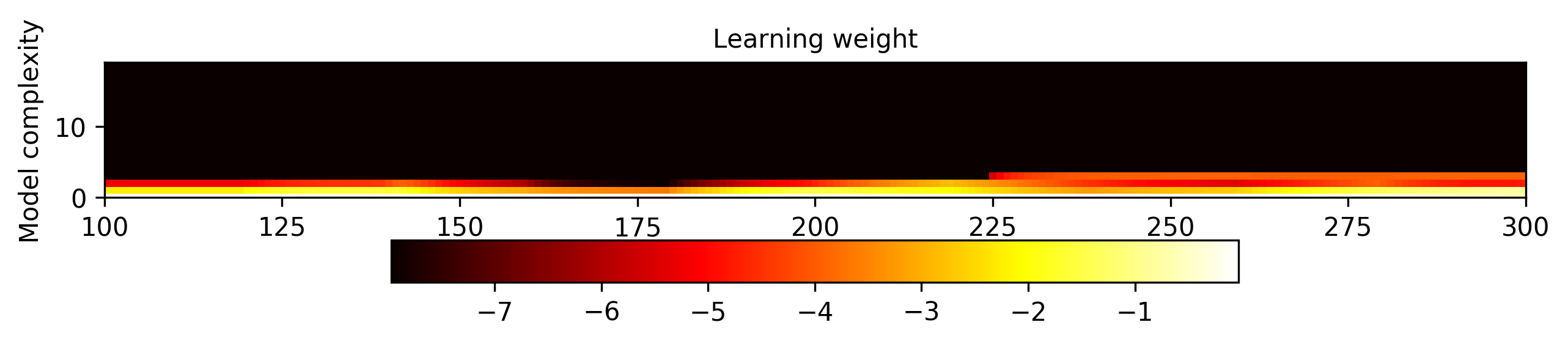
In another experiment, we generate a sequence of data from a single-layer Feed-Forward Neural Network. We sequentially obtain and learn the data, starting from 𝑡=100, then 𝑡=101,…,300. True model is a large circle containing a smaller circle in 2-D with Gaussian noise (mean 0 variance 0.2), and the scale factor between the inner and outer circle is 0.8. We start from 100 samples because Neural Network likely converges to a local optimal for small sample size. Neural Network is also inherently miss-specified for the true model. The path of expansion in this case is the number of hidden nodes in the single hidden layer. Since the true model is not linearly-separable, we do need more hidden nodes to accurately classify the data. We restrict the maximum number of hidden nodes to be divided by the input dimensions where n is the available data size. The path of expansion is in increasing order of the number of hidden nodes, since having a small number of hidden nodes is a special case of having more number of hidden nodes.

Similarly, the optimal model (oracle) is obtained by testing the trained model on a large dataset. The oracle loss of different models at different data size is shown in Fig. 5. With a small sample size, the cost of overfitting is considerably high. When we have enough samples for training, the cost of overfitting decreases. This effect may also depend on the dimension of input data and labels. We are interested in investigating this phenomenon in the future. In Fig. 7, the loss ratio varies quite a lot when sample size is small but gradually converges. This is partially because the cost of overfitting decreases. Even we choose a model which is slightly overfitting, the loss ratio is still low enough. However, this is not ideal. Our proposed sequential algorithm dominates other approaches as shown in Fig. 7, because the weight of smaller models in the active set is large enough to prevent the model to expand. As a result, we alleviate the tendency to choose the overfitting models even their loss is relatively small.

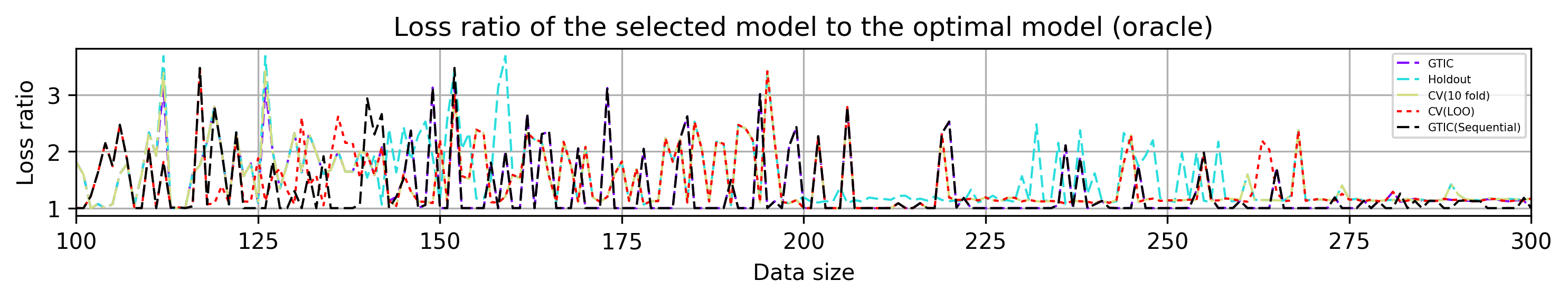
The computation cost is shown in Fig. 8. As expected, the computation of 10-fold CV and LOO increases significantly. However, since we *pre-compile the symbolic computation* for the penalty term (which involves second order derivatives), our computation cost is almost constant throughout the experiment. This is promising because when we want to sequentially expand Neural Networks, the path of expansion can be predetermined, and thus after pre-compilation, the computation cost is almost identical to Holdout. Furthermore, we can utilize *warm-start* in our implementation which is a benefit that CV cannot enjoy in naïve sequential model selection framework. Therefore, we encourage the use of GTIC in sequential model expansion scheme.



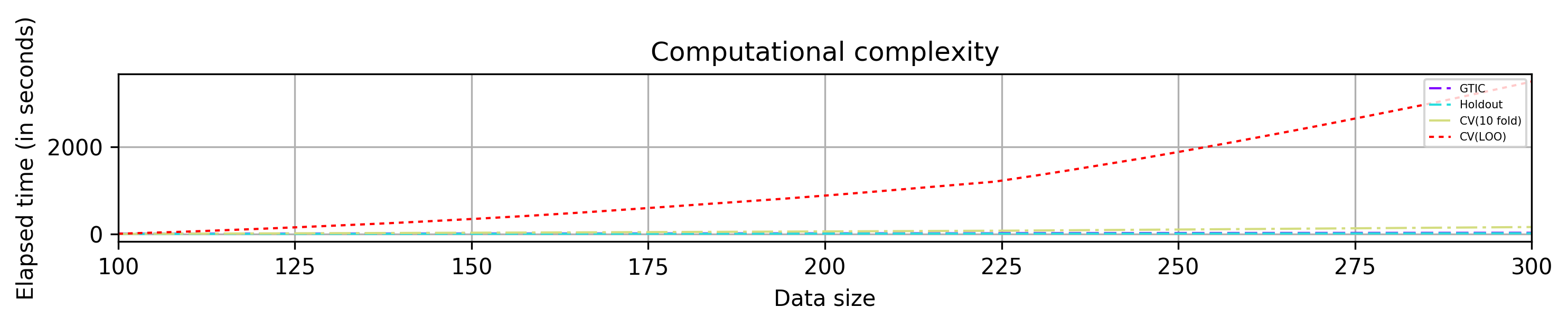
*Figure 5: Heat-map showing the prediction loss of estimated candidate models of each dimension (y-axis) at each data size (x-axis), where the black dots indicate the model of optimal loss at each data size.*



*Figure 6: Heat-map showing our predictive weights over the candidate models (y-axis) at each data size (x-axis).*



*Figure 7: Plot showing the loss of our predictor (GTIC) and cross validations at each data size.*



*Figure 8: Plot showing the computational costs.*