Supplementary Material for "On Statistical Efficiency in Learning"

Jie Ding, Enmao Diao, Jiawei Zhou, and Vahid Tarokh

NUMERICAL EXPERIMENTS

The model classes under consideration are logistic regression and feed-forward neural networks. We also released an open-source python package 'gtic' at https://pypi.python.org/pypi/gtic, in which we build a tensor graph of GTIC upon the 'theano' platform. Users can simply provide their tensor variables of loss and parameters and obtain the GTIC instantly. We choose the logistic regression as a representative statistical model whose log-likelihood function is known to be convex (under mild conditions), and its minimization is relatively well understood. We consider the neural networks with one hidden layer and sigmoid activation functions. The convergence of optimizing more complicated neural networks remains a challenging problem. But we observed from various experiments that the computed second-order matrix $\hat{V}_n$, once symmetrized using $(\hat{V}_n + \hat{V}_n^{\mathrm{T}})/2$, produces numerically stable results of GTIC and model selection.

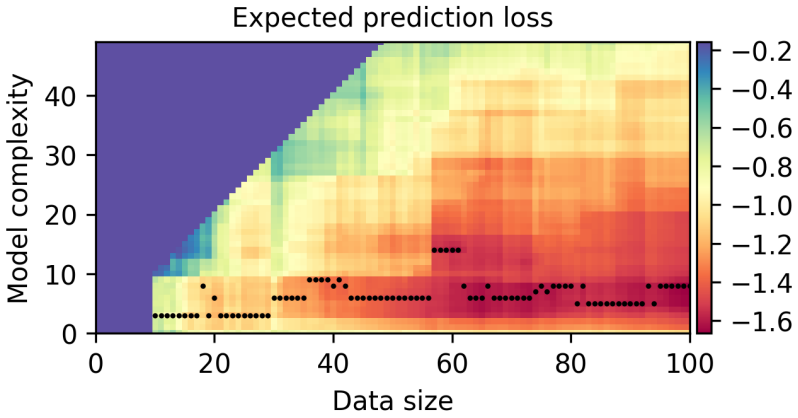
A. Logistic Regression Models

We generate data from a logistic regression model, where the coefficient vector is $\boldsymbol{\beta} = 10 \times [1^{-1.5}, \ldots, 100^{-1.5}]^{\mathrm{T}}$, and covariates $x_1, \ldots, x_{100}$ are independent standard Gaussian. Suppose that we sequentially obtain and learn the data, starting from $t = 10$, and then $t = 11, \ldots, 100$. We consider the model class that consists of logistic regression models of dimensions $1, \ldots, \lfloor \sqrt{t} \rfloor$ at each time step t. A model of dimension d means that the coefficients of the first d variables $x_1, \ldots, x_d$ are unknown parameters and the coefficients of the remaining variables are restricted to be zero. The model class is nested because a small model is a particular case of a large model. Since the maximum dimension of candidate models is restricted to be no larger than $\sqrt{100} = 10$, the model class is considered misspecified. We summarize the results in Fig. 1 and 2.

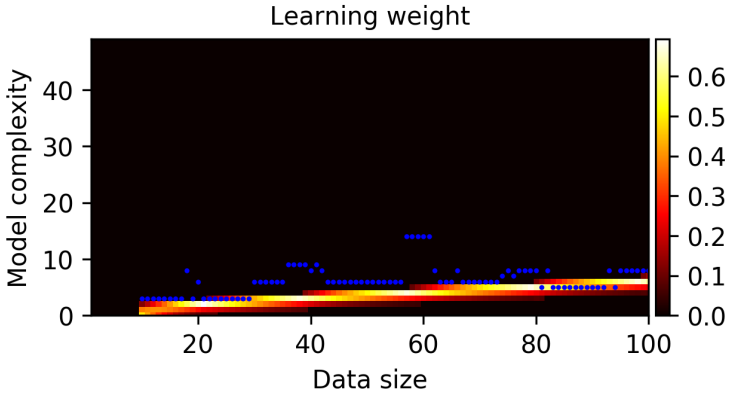
To illustrate the efficiency of GTIC, we first simulate model selection results with batch data. We numerically compute each trained model's prediction loss (obtained by testing on a large dataset) and then identify the optimal model (with the least loss). In Fig. 2a, we compare the performance of GTIC to different types of CV. Holdout takes 70% of the 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 with all the other approaches. As shown in Fig. 1a and 1b, although the optimal model of each sample 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 (Fig. 2a). This result is consistent with our definition of efficient learning.

Next, we discuss the computational complexity of each method. We write $a_n = \Omega(b_n)$ if there exists a positive constant c such that $c^{-1} < a_n/b_n < c$ for all n. Suppose that we use the Newton-Raphson method for logistic regression with sample size n and model dimension d. In one iteration of update, the complexity of computing the first and second derivatives is $\Omega(nd + nd^2)$, and the complexity of computing the matrix inverse is no more than $\Omega(d^3)$. Suppose that we converge in a fixed number of iterations and $d < n$, the overall complexity in estimating a logistic regression model is $\Omega(nd^2)$. The complexity of computing the GTIC penalty is similarly calculated to be $\Omega(nd^2)$. In our implementation, the candidate models have dimensions $1, \ldots, d$. The overall computation complexity of performing m-fold CV is $\Omega(mn \cdot 1^2) + \cdots + \Omega(mn \cdot d^2) = \Omega(mnd^3)$, and that of LOO is $\Omega(n^2 d^3)$. The overall

J. Ding is with the School of Statistics, University of Minnesota, Minneapolis, Minnesota 55455, USA. E. Diao and V. Tarokh are with the Department of Electrical and Computer Engineering, Duke University, Durham, North Carolina 27708, USA. J. Zhou is with the John A. Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, 02138 USA.



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



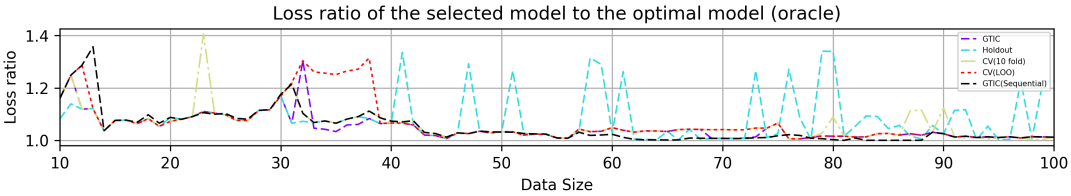
(b) Heat-map showing our predictive weights over the candidate models (y-axis) at each sample size (x-axis), using sequential model expansion.

Fig. 1: Experiment 1: logistic regression models

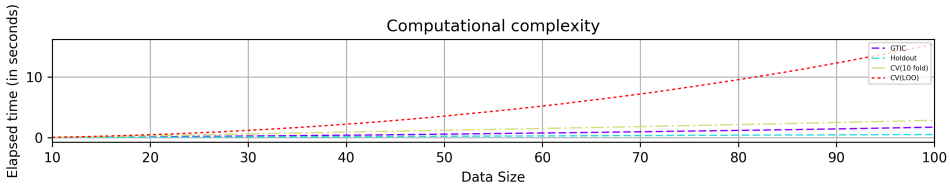
complexity of GTIC is $\Omega(nd^3)$. We note that the gradient and hessian involved in the GTIC penalty term can be analytically computed using symbolic expression computation saved on the disk in advance (e.g., using 'theano').

In our simulation, the computation cost of each approach is provided in Fig. 2b. As shown in the figures, under logistic regression, GTIC is slightly better than 10-fold CV but worse than Holdout. GTIC is no more than ten times faster than 10-fold CV. This is because we need to compute the penalty term in GTIC plus the evaluation of n data, while each of the ten folds uses less than n data. However, depending on the problem and data, we may need a different number of folds for CV in order to have a satisfactory result. Since GTIC performs almost as well as LOO and 10-fold CV, we suggest using GTIC instead of guessing or searching the optimal number of folds for CV. With GTIC, we do not need to sacrifice much on computation cost but can still achieve a theoretically justifiable result as good as LOO.

In another experiment, we considered two underlying data generating models. One model (called $\mathcal{M}_1$) is generated using a logistic regression model with coefficients $\boldsymbol{\beta} = [1, 2^{-0.1}, \ldots, p^{-0.1}]$ and standard Gaussian covariates. The other model (called $\mathcal{M}_2$) is generated using a logistic regression model with coefficients $\boldsymbol{\beta} = [0.999, 0.999^2, \ldots, 0.999^p]$ and standard Gaussian covariates. We numerically compare the performance of AIC, BIC, slope heuristics (denoted by SH), and TIC under various choices of sample size n and the number of covariates p. The SH method is based on the AIC shape (linear in dimension) and the 'dimension jump' approach to determine the multiplicative constant. The performance is evaluated using out-sample prediction loss, prediction accuracy, and prediction efficiency, summarized in Tables I, II, III, respectively. The best performing method is highlighted with bold. The results show that TIC performs the best in all cases, and BIC is always the worst for the misspecified

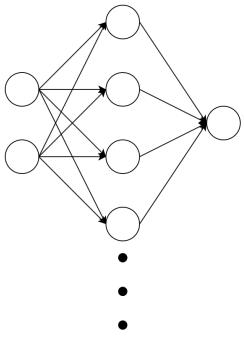


(a) Plot showing the loss of our predictor (GTIC) and cross-validations at each sample size

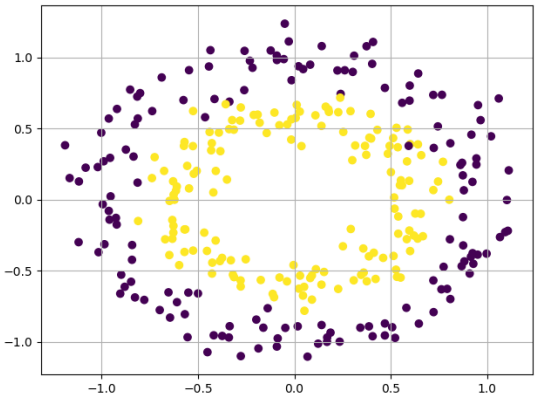


(b) Plot showing the computational costs.

Fig. 2: Experiment 1: logistic regression models



(a) An illustration of the single-layer feed-forward neural network



(b) A set of 300 data uniformly sampled from two circles corrupted by Gaussian noise ($\mu = 0$, $\sigma^2 = 0.1$, radius ratio $= 0.6$)

Fig. 3: Experiment 2: neural networks

model class. The results are summarized with three evaluation metrics, mainly because they are all used in practice, and their interpretations are different. The out-sample prediction loss of Tables I, also referred to as the "testing loss", shows the KL divergence from the true data generating model to the learned model (up to an additive constant). The prediction accuracy of Table II, also referred to as the "classification accuracy", shows the accuracy of threshold decisions, and it is not a proper loss for training. The prediction efficiency of Table III shows the relative deviation from the underlying truth compared with the best candidate model in hindsight. To complement the tabulated results, we also visualize the out-sample prediction loss for $\mathcal{M}_1$ in a boxplot (Figure 6).

B. Neural Networks

We consider the model class to be single-layer feed-forward neural networks with the sigmoid activation function (see Fig. 3a). Neural networks are inherently misspecified models.

Data are generated from the following way. A set of two-dimensional data are uniformly sampled from two circles (with radius ratio 0.6), corrupted by independent Gaussian noise with mean 0 and variance 0.1 (generated from python package 'sklearn' dataset "make_circle"). The goal is to correctly classify the data into two groups, the larger and smaller rings. Since we have two-dimensional data, our input dimension for the model is two. Since we want to classify into two groups, the output dimension is two. Our model's complexity is the number of hidden nodes in the single hidden layer in this experiment.

TABLE I: Performance comparison in terms of the out-sample prediction loss, for data generated from two different logistic regression models. The values are averaged from 50 independent replications, with standard errors (i.e., standard deviations divided by $\sqrt{50}$) in the parentheses.

	$\mathcal{M}_1$				$\mathcal{M}_2$			
	AIC	BIC	SH	TIC	AIC	BIC	SH	TIC
$n = 100, p = 20$	**0.41 (0.01)**	0.53 (0.03)	**0.41 (0.01)**	**0.41 (0.01)**	0.38 (0.02)	0.54 (0.03)	0.38 (0.02)	**0.37 (0.01)**
$n = 100, p = 50$	0.66 (0.02)	0.68 (0.01)	0.61 (0.01)	**0.51 (0.02)**	0.68 (0.03)	0.71 (0.01)	0.69 (0.02)	**0.47 (0.02)**
$n = 300, p = 60$	**0.34 (0.01)**	0.64 (0.02)	**0.34 (0.02)**	**0.34 (0.01)**	**0.30 (0.01)**	0.44 (0.05)	0.31 (0.01)	**0.30 (0.01)**
$n = 300, p = 150$	0.81 (0.05)	0.69 (0.00)	0.67 (0.02)	**0.53 (0.02)**	0.81 (0.03)	0.70 (0.00)	0.65 (0.01)	**0.50 (0.01)**
$n = 500, p = 100$	**0.34 (0.01)**	0.68 (0.01)	0.36 (0.01)	**0.34 (0.01)**	0.30 (0.01)	0.44 (0.05)	**0.29 (0.01)**	**0.29 (0.01)**
$n = 500, p = 250$	0.95 (0.04)	0.69 (0.00)	0.65 (0.02)	**0.54 (0.02)**	0.96 (0.03)	0.69 (0.00)	0.55 (0.01)	**0.51 (0.01)**

TABLE II: Performance comparison in terms of the out-sample prediction accuracy, for data generated from two different logistic regression models. The values are averaged from 50 independent replications, with standard errors in the parentheses.

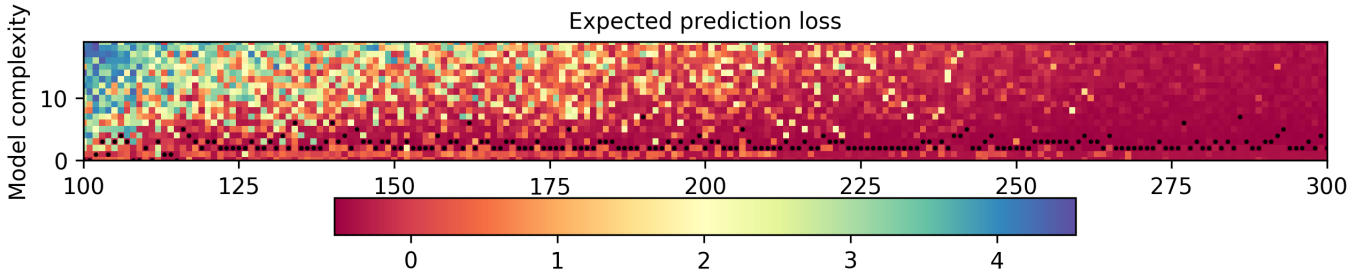
	$\mathcal{M}_1$				$\mathcal{M}_2$			
	AIC	BIC	SH	TIC	AIC	BIC	SH	TIC
$n = 100, p = 20$	**0.80 (0.01)**	0.74 (0.02)	**0.80 (0.01)**	**0.80 (0.01)**	0.82 (0.02)	0.75 (0.05)	0.82 (0.02)	**0.84 (0.01)**
$n = 100, p = 50$	0.68 (0.03)	0.54 (0.07)	0.73 (0.02)	**0.77 (0.02)**	0.73 (0.03)	0.63 (0.06)	0.73 (0.03)	**0.80 (0.02)**
$n = 300, p = 60$	**0.82 (0.01)**	0.60 (0.03)	**0.82 (0.01)**	**0.82 (0.01)**	**0.86 (0.01)**	0.80 (0.03)	**0.86 (0.02)**	**0.86 (0.01)**
$n = 300, p = 150$	0.73 (0.02)	0.57 (0.09)	0.74 (0.02)	**0.78 (0.01)**	0.76 (0.02)	0.67 (0.08)	0.79 (0.03)	**0.81 (0.02)**
$n = 500, p = 100$	**0.82 (0.01)**	0.57 (0.04)	0.81 (0.01)	**0.82 (0.01)**	**0.87 (0.01)**	0.77 (0.06)	**0.87 (0.01)**	**0.87 (0.01)**
$n = 500, p = 250$	0.72 (0.01)	0.50 (0.08)	0.76 (0.01)	**0.80 (0.01)**	0.72 (0.01)	0.40 (0.05)	0.75(0.02)	**0.79 (0.01)**

TABLE III: Performance comparison in terms of the out-sample prediction efficiency for data generated from two different logistic regression models. The values are averaged from 50 independent replications, with standard errors in the parentheses.

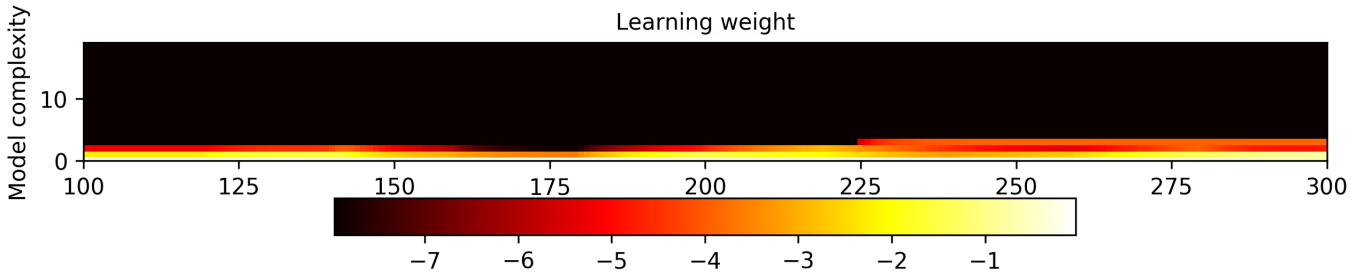
	$\mathcal{M}_1$				$\mathcal{M}_2$			
	AIC	BIC	SH	TIC	AIC	BIC	SH	TIC
$n = 100, p = 20$	**0.97 (0.01)**	0.58 (0.08)	**0.97 (0.02)**	**0.97 (0.01)**	0.97 (0.02)	0.58 (0.09)	0.97 (0.02)	**0.99 (0.02)**
$n = 100, p = 50$	0.67 (0.04)	0.60 (0.04)	0.80 (0.03)	**0.98 (0.02)**	0.60 (0.04)	0.54 (0.03)	0.61 (0.03)	**0.98 (0.01)**
$n = 300, p = 60$	**0.98 (0.01)**	0.32 (0.06)	**0.98 (0.01)**	**0.98 (0.01)**	0.96 (0.02)	0.64 (0.10)	0.96 (0.02)	**0.98 (0.01)**
$n = 300, p = 150$	0.58 (0.03)	0.70 (0.04)	0.76 (0.02)	**0.99 (0.01)**	0.56 (0.03)	0.66 (0.04)	0.79 (0.03)	**1.00 (0.00)**
$n = 500, p = 100$	**0.98 (0.01)**	0.32 (0.02)	0.95 (0.01)	**0.98 (0.01)**	0.96 (0.02)	0.62 (0.08)	0.98 (0.02)	**1.00 (0.00)**
$n = 500, p = 250$	0.50 (0.02)	0.73 (0.04)	0.81 (0.02)	**0.99 (0.01)**	0.48 (0.02)	0.68 (0.02)	0.92 (0.02)	**0.99 (0.01)**

TABLE IV: Performance comparison in terms of the average efficiency in a sequential setting, for data and models described in Subsection -B. The values are averaged from 20 independent replications (standard errors within 0.3).

Holdout	CV (10 fold)	CV (LOO)	AIC	BIC	SH	GTIC	GTIC (Sequential)
0.62	0.67	0.87	0.71	0.58	0.69	0.75	0.92

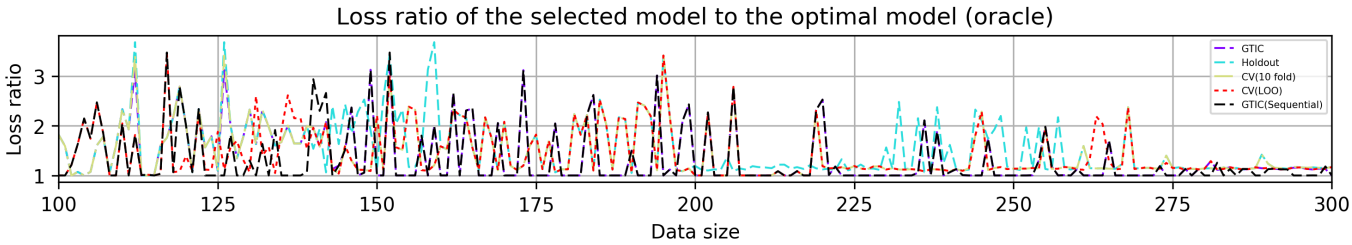


(a) Heat-map showing the prediction loss of estimated candidate models of each dimension (y-axis) at each sample size (x-axis), where the black dots indicate the model of optimal loss at each sample size.

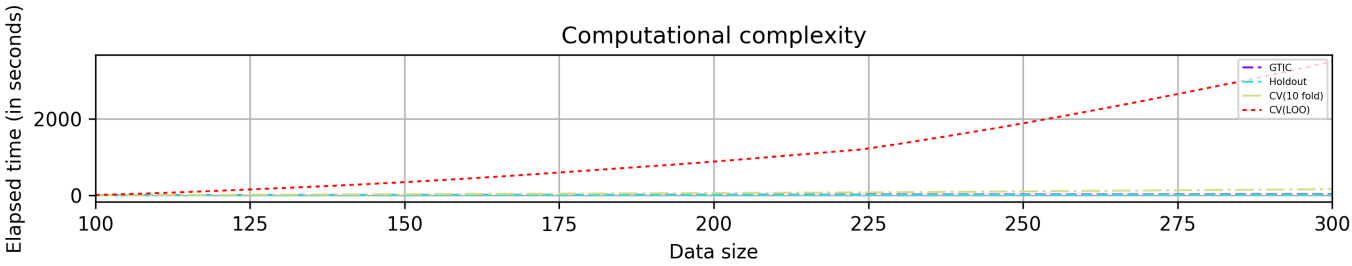


(b) Heat-map showing our predictive weights over the candidate models (y-axis) at each sample size (x-axis).

Fig. 4: Experiment 2: neural networks



(a) Plot showing the loss of our predictor (GTIC) and cross-validations at each sample size.



(b) Plot showing the computational costs.

Fig. 5: Experiment 2: neural networks

We sequentially obtain and learn the data, starting from $t = 100$, then $t = 101, \ldots, 300$. We start from 100 samples because Neural Network is likely to converge to a local optimal for a small sample size. The path of expansion is the number of hidden nodes in the single hidden layer. Since data are not linearly separable, we need at least one hidden layer to classify the data accurately. We restrict the maximum number of hidden nodes to be $\sqrt{t}$/(input dimension) due to our assumption. 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 sample sizes, as shown in Fig. 4a. With a small sample size, the cost of overfitting

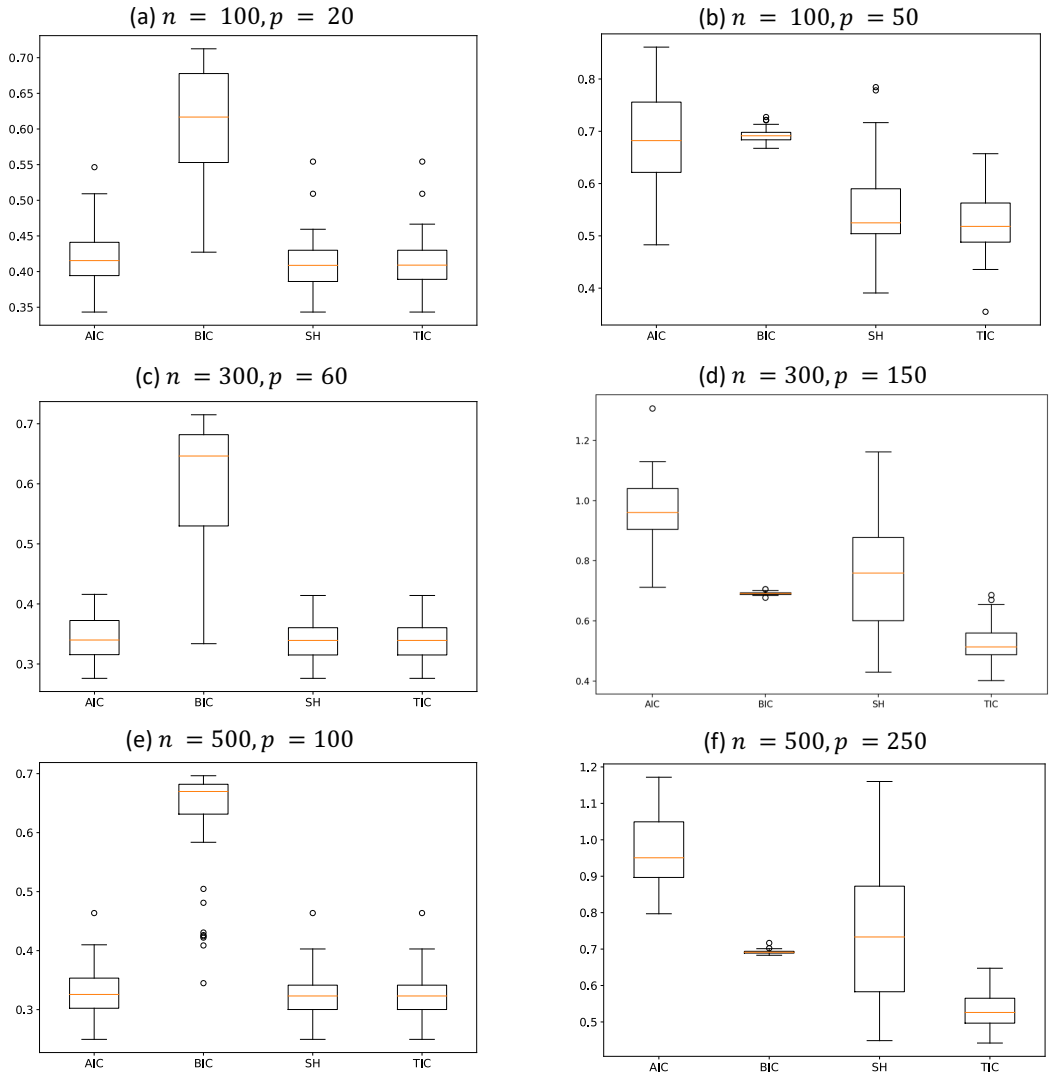


Fig. 6: Boxplot view of the prediction loss for $\mathcal{M}_1$

is considerably high. In Fig. 5a, the loss ratio varies quite a lot when the sample size is small, but gradually converges. It is mainly because the influence of overfitting on the predictive power decreases as the sample size increases. In other words, even if we choose a model that is slightly overfitting, the loss ratio is still close to one. The results in Fig. 4b show that the weights of smaller models in the active set are large enough to prevent the model from expanding. As a result, we alleviate the tendency to choose the overfitting models even when their loss is relatively small. The average efficiency over all the time steps is reported in Table IV, where we considered the use of various selection criteria in sequential settings (using Algo. 2).

The computational cost is shown in Fig. 5b. As expected, the computation of 10-fold CV and LOO increases significantly. However, since we can analytically compute the gradient and hessian involved in the GTIC penalty term, using symbolic expression computation software and saving them on the disk in advance, our computation cost is almost constant at each time step. Therefore, our overall computational cost is almost identical to Holdout. Furthermore, we can utilize warm-start in our implementation, which is an advantage that CV cannot enjoy in a sequential model selection framework. Therefore, we encourage the use of GTIC in the sequential model expansion scheme.