Shannon Inspired Approach to Limits of Learning

Contract No: W911NF-16-1-0561

Yearly Report for September 1, 2016 - August 31, 2017

PI: Vahid Tarokh

I. Introduction

We considered the general problem of learning from data, where the objective of the learner is to build a generative/discriminative model based a set of historical data. The learner then would like to measure the performance of the inferred function with some objective in mind. Let $x^n = \{x_i\}_{i=1}^n$, where $x_i \in \mathcal{X}$, denote the data samples at the learner's disposal that are assumed to be drawn i.i.d. from an unknown joint density function $p(x)$. We assumed that the learner can express her objective in terms of minimizing a parametric loss function $\ell(x;\theta)$, which is a smooth function of the parameter vector θ. The learner solves for the unknown parameter vector $\theta \in \Theta$ where $\Theta \subseteq \mathbb{R}^k$. Further, let $L(\theta) := E\{\ell(x;\theta)\}$ be the risk associated with the parameter vector θ, where the expectation is with respect to the density $p(\cdot)$. Ideally, the goal of the learner is to choose the parameter vector θ^λ such that

$$\theta^* = \arg\min_{\theta\in\Theta} L(\theta) = \arg\min_{\theta\in\Theta} E\{\ell(x;\theta)\}. \tag{1}$$

Since the density function $p(\cdot)$ is unknown, the learner cannot compute θ^* and hence cannot achieve $L(\theta^*)$. Instead, she usually solves the empirical version of the problem and chooses $\hat{\theta}^\lambda(x^n)$ as follows:

$$\hat{\theta}^\lambda(x^n) = \arg\min_{\theta\in\Theta} \left\{ \hat{L}(\theta) \right\} = \arg\min_{\theta\in\Theta} \left\{ \hat{E}\{\ell(x_i;\theta)\} \right\}, \tag{2}$$

where $\hat{E}$ is the empirical mean operator with respect to the training samples x^n, i.e.,

$$\hat{E}\{f(x_i)\} = \frac{1}{n}\sum_{i=1}^{n} f(x_i), \tag{3}$$

and

$$\hat{L}(\theta) = \hat{E}\{\ell(x_i;\theta)\} = \frac{1}{n}\sum_{i=1}^{n} \ell(x_i;\theta). \tag{4}$$

The question of whether a concept is asymptotically learnable in a hypothesis class (or a filtration of hypothesis classes) has been extensively studied and understood. On the other hand, the daunting task of machine learning is to understand how quickly this could be obtained and what is the best strategy of the learner given a finite data set. Most of the theoretical work on this front concerns worst-case and probabilistic upper bounds on the loss incurred when a new sample arrives. However, these bounds are usually pessimistic in nature and practitioners resort to methods such as cross validation for model selection and learning. We emphasize that the goal of this project is to devise a theoretical framework to obtain the limits of learning as new data arrives, which will in turn equip the learner with practical tools for finding the best learning machine within a given model class (limits of learning within each class) and also to be able to choose the optimal complexity within a filtration of model classes (limits of learning across classes)

In the previous year, we set out to answer this question and provided a framework inspired from information theory for estimating the out-of-sample prediction error from a given data set. In particular, our contributions are listed as follows.

- We defined the limit of learning (LoL) for vector-time series.
- We discovered a Shannon-like formula for LoL in well-specified parametric cases.
- We developed numerical algorithms and software for LoL computation.
- We numerically demonstrate the superior performance of our proposal even for small amount of data.
- We quantified its relationship with model selection, and developed LoL-inspired model selection penalties.
- We proved the consistency in well-specified cases and efficiency in mis-specified cases.
- We investigated surrogates of the LoL for arbitrary datasets.
- We proposed computation algorithms for fast online implementation, and released a python package specifically for neural networks.

- We studied the issue of non-identifiability in non-negative matrix factorization problems.
- We proposed a framework for LoL in cognitive settings.

II. Definition of the Limit of Learning for Vector-Times Series

In this section, we provide the definition of the limit of learning.

Definition 1 (Limit of learning). *We define the limit of learning as*

$$LOL = I_U(X_{n+1}; X_1, \ldots, X_n, \Theta) - I_U(X_{n+1}; X_1, \ldots, X_n),$$

where $U(\cdot)$ can be thought of as the **measure** *of learning.*

Let us first provide two information theoretic interpretations for this definition by considering $I_U(\cdot;\cdot)$ to be the mutual information. In this case, X_{n+1} is distinguishable when $X_1, \ldots, X_n$ is at the learner's disposal if it is in the set of $2^{I(X_{n+1};X_1,\ldots,X_n)}$ jointly typical values (asymptotic equipartition property). Thus, the limit of learning as defined above is the amount of exponential loss in the error probability that is lost when the learner (who does not know Θ) predicts X_{n+1} from $X_1, \ldots, X_n$.

In the case where U is Shannon's entropy we have

$$I_U(X_{n+1}; X_1, \ldots, X_n, \Theta) - I_U(X_{n+1}; X_1, \ldots, X_n) = U(\Theta | X_1, \ldots, X_n) - U(\Theta | X_1, \ldots, X_{n+1}).$$

In this case, it is clear that $I_U(X_{n+1}; X_1, \ldots, X_n, \Theta) - I_U(X_{n+1}; X_1, \ldots, X_n) \geq 0$. In Section III, we will discuss algorithms based sequential Monte Carlo methods to numerically compute the limit of learning.

Theorem 1. *Suppose that $X_1, \ldots, X_n$ are drawn i.i.d. from an unknown distribution p. Further assume that $\dim(\Theta) = d$. Then, under mild conditions, we have*

$$LOL = \frac{d}{2} \log\left(1 + \frac{1}{n}\right).$$

This result is reminiscent of Shannon's celebrated channel capacity formula, which is

$$C = \frac{K}{2} \log\left(1 + \frac{P}{\sigma^2}\right).$$

We believe that there is an analogous formula for the limit of learning for very general objectives of the learner when the data is drawn i.i.d. from an unknown distribution that does not necessarily belong to the model class. We also believe that this can be further extended to the dependent data under some assumptions. We are currently investigating these directions, and have derived some preliminary results.

III. Relationship with Model Selection

Next, we comment on the relation between the limit of learning (LOL) defined in the previous section and the task of model selection. Let us consider LOL again in the asymptotic limit as $n \to \infty$:

$$2\text{LOL} = d \log(1 + n) - d \log(n).$$

This resembles the Bayesian information criterion (BIC). Motivated by it, we propose to use the penalty function $-2U(\Theta | X_1, \ldots, X_n)$ instead of the BIC penalty function (which is reached in the asymptotic form of the LOL) [1], [2].

Lemma 1. *For a well-specified parametric case with finite dimensional parameters, the LOL-inspired model selection penalty is consistent, and achieves the same performance as BIC as $n \to \infty$.*

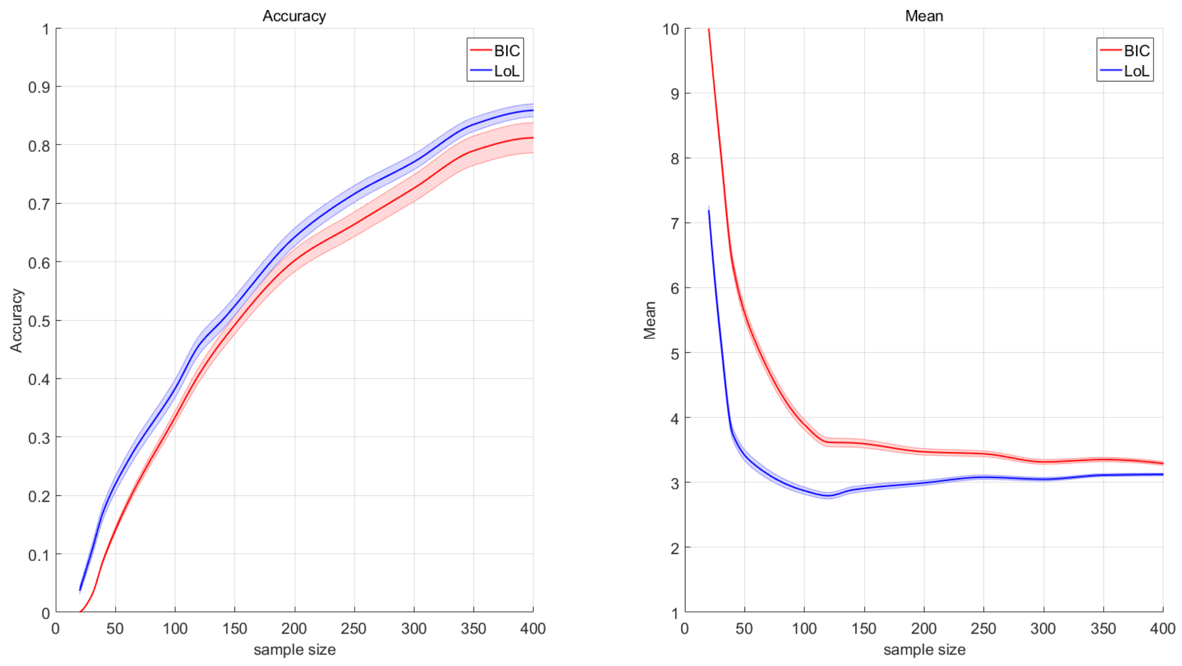


Fig. 1: The left plot shows the empirical accuracy of selecting the true model using LOL and BIC. The right plot shows the average model order selected by LOL and BIC as a function of the sequence length. The noise is assumed to be i.i.d. normal Gaussian.

This result is expected given the asymptotic relation between LOL and BIC. Further, while the LOL-inspired criterion is asymptotically good, the question is how can one compute this criterion in practice.

While the new penalty function is analogous to BIC as n grows, the question is whether it would perform better than BIC for finite n. To answer this question, we present a simple experiment on the selecting the autoregressive (AR) order from AR class of orders $1, \ldots, 10$. In the experiment, we use the conditional mutual information as the LOL inspired model selection criterion and directly compare with BIC. We remark that the LOL-inspired criterion is calculated using the SMC that is discussed in Section V. The true order of AR process used to generate the samples was 3, and the noise was generated as i.i.d. standard Gaussian. We sweep the experiment for varying sample sizes in the range from 20 to 400. The results for the error probability as well as the average selected model are plotted in Figure 1. As can be seen, the LOL-inspired model selection criterion provides superior performance to BIC for finite samples. We also observe that although both BIC and the conditional mutual information lead to consistent estimators for the order of the process, the latter converges faster and hence performs better in the finite sample case of interest in this example. In the second experiment, to show that this phenomenon is not just confined to the Gaussian noise process, we changed the generation noise to i.i.d. standard Laplacian process. As can be seen in Figure 2, similar trends as in the previous case could be observed in this case as well.

As a second well-studied example, we let $I_U(X_{n+1}; X_1, \ldots, X_n, \Theta)$ be the negative logarithm of the out-of-sample prediction error of X_{n+1} given $X_1, \ldots, X_n, \Theta$. In other words, $U(\Theta|X_1, \ldots, X_n)$ is supposed to be the squared error scaled by the Fisher information matrix. Under these assumptions the LOL-inspired model selection penalty gives back AIC (Akaike Information Criterion) and TIC (Takeuchi information criterion) in the well-specified and mis-specified setting, respectively. In practice, one would mostly be concerned with the mis-specified setting as the real data almost never conforms to a true model.

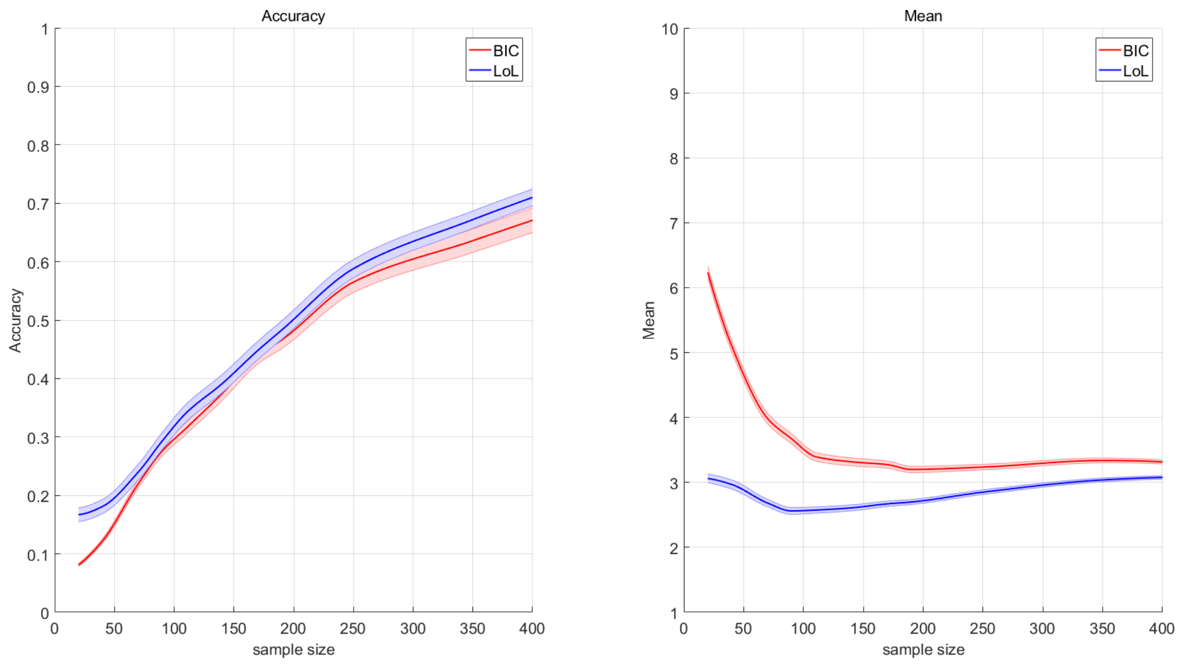


Fig. 2: The left plot shows the empirical accuracy of selecting the true model using LOL and BIC. The right plot shows the average model order selected by LOL and BIC as a function of the sequence length. The noise is assumed to be i.i.d. normal Laplace.

IV. Asymptotic Efficiency of TIC

Our next contribution concerns with the asymptotic efficiency of TIC under some regularity conditions [3]. Before we state the result, we introduce some notation.

Let $\mathcal{G}$ denote a set of candidate models. The loss function for each data size n and $\alpha \in \mathcal{G}$ (model class) is a map $l : \mathcal{D} \times \Theta[\alpha] \to \mathbb{R}$, usually written as $l(\boldsymbol{z}, \theta_t; \alpha)$, where $\mathcal{D}$ is the data domain, $\Theta[\alpha]$ is the parameter space associated with model α, and α is included to emphasize the model under consideration. Moreover, the expected prediction loss given by candidate model α, denoted by $\mathcal{L}_n(\alpha)$, is defined by

$$\mathcal{L}_n(\alpha) \triangleq \mathbb{E}_* l\big(\cdot, \hat{\theta}_t[\alpha]; \alpha\big) = \int_{\mathcal{D}} p(\boldsymbol{z}) l\big(\boldsymbol{z}, \hat{\theta}_t[\alpha]; \alpha\big) d\boldsymbol{z}. \tag{5}$$

Here, $\mathbb{E}_*$ denotes the expectation with respect to the distribution of a future (unseen) random variable $\boldsymbol{z}$.

We consider loss functions $l(\cdot)$ such that $\mathcal{L}_n[\alpha]$ is always nonnegative. A common choice is to use negative log-likelihood of model α minus that of the true data generating model. Table I lists some other loss functions widely used in machine learning.

TABLE I: Some common loss functions in addition to negative log-likelihood

Name	quadratic	exponential	hinge	perceptron	logistic
Formula	$(y - \theta_t^{\mathrm{T}} \boldsymbol{x})^2$	$e^{-y\theta_t^{\mathrm{T}} \boldsymbol{x}}$	$\max\{0, 1 - y\theta_t^{\mathrm{T}} \boldsymbol{x}\}$	$\max\{0, -y\theta_t^{\mathrm{T}} \boldsymbol{x}\}$	$\log(1 + e^{-y\theta_t^{\mathrm{T}} \boldsymbol{x}})$
Domain	$y \in \mathbb{R}$	$y \in \mathbb{R}$	$y \in \mathbb{R}$	$y \in \mathbb{R}$	$y \in \{0, 1\}$

For a measurable function $f(\cdot)$, we define $E_n f(\cdot) = n^{-1} \sum_{i=1}^{n} f(\boldsymbol{z}_i)$. We let $\boldsymbol{\psi}_n(\boldsymbol{z}, \theta_t; \alpha) \triangleq \nabla_{\theta_t} l(\boldsymbol{z}, \theta_t; \alpha)$, and $\nabla_{\theta_t} \boldsymbol{\psi}_n(\boldsymbol{z}, \theta_t; \alpha) \triangleq \nabla_{\theta_t}^2 l(\boldsymbol{z}, \theta_t; \alpha)$, which are respectively measurable vector-valued and matrix-valued functions of

θ_t. We define the matrices

$$V_n(\theta_t;\alpha) \triangleq \mathbb{E}_* \nabla_{\theta_t} \boldsymbol{\psi}_n(\cdot,\theta_t;\alpha)$$
$$J_n(\theta_t;\alpha) \triangleq \mathbb{E}_* \big\{ \boldsymbol{\psi}_n(\cdot,\theta_t;\alpha) \times \boldsymbol{\psi}_n(\cdot,\theta_t;\alpha)^{\mathrm{T}} \big\}$$

Recall the definition of $\mathcal{L}_n[\alpha]$. Its sample analog (also referred to as the *in-sample loss*) is defined by $\hat{\mathcal{L}}_n[\alpha] \triangleq E_n l\big(\cdot, \hat{\theta}_t[\alpha];\alpha\big)$. Similarly, we define

$$\hat{V}_n(\theta_t;\alpha) \triangleq E_n \nabla_{\theta_t} \boldsymbol{\psi}_n(\cdot,\theta_t;\alpha)$$
$$\hat{J}_n(\theta_t;\alpha) \triangleq E_n \big\{ \boldsymbol{\psi}_n(\cdot,\theta_t;\alpha) \times \boldsymbol{\psi}_n(\cdot,\theta_t;\alpha)^{\mathrm{T}} \big\}$$

Each candidate model α produces an estimator $\hat{\theta}_t[\alpha]$ (referred to as the minimum loss estimator) defined by

$$\hat{\theta}_t[\alpha] \triangleq \arg\min_{\theta \in \Theta[\alpha]} \frac{1}{n} \sum_{i=1}^{n} l(z_i,\theta_t;\alpha). \tag{6}$$

We propose to use the following penalized model selection procedure, which generalizes TIC from negative log-likelihood to general loss functions.

Generalized TIC (GTIC) procedure: Given data $\boldsymbol{z}_1,\ldots,\boldsymbol{z}_n$ and a specified model class $\mathcal{G}$. We select a model $\hat{\alpha} \in \mathcal{G}$ in the following way: 1) for each $\alpha \in \mathcal{G}$, find the minimal loss estimator $\hat{\theta}_t[\alpha]$ defined in (6), and record the minimum as $\hat{\mathcal{L}}_n[\alpha]$; 2) select $\hat{\alpha} = \arg\min_{\alpha \in \mathcal{G}} \underline{\mathcal{L}_n}[\alpha]$, where

$$\underline{\mathcal{L}_n}[\alpha] \triangleq \hat{\mathcal{L}}_n[\alpha] + n^{-1}\,\mathrm{tr}\big\{ \hat{V}_n(\hat{\theta}_t[\alpha];\alpha)^{-1} \hat{J}_n(\hat{\theta}_t[\alpha];\alpha) \big\}. \tag{7}$$

Theorem 2. *Under mild assumptions, then the $\hat{\alpha}_n$ selected by GTIC procedure is asymptotically efficient, in the sense that*

$$\frac{\mathcal{L}_n[\hat{\alpha}_n]}{\min_{\alpha \in \mathcal{G}} \mathcal{L}_n[\alpha]} \to_p 1 \tag{8}$$

as $n \to \infty$.

A more mathematically rigorous statement and proof are included in our manuscript [3]. There are clear paths for the extension of the framework to more practical settings, e.g., to drop the assumption that the data of interest is generated by an i.i.d. process.

V. Computation Algorithms

In this section, we focus on numerical methods for the calculation of the limits of learning. We discuss our methods for both well-specified and mis-specified cases.

A. Computation of LOL in the Well-Specified Case

In the well-specified case the posterior could be computed given the samples. In particular, we have used the SMC (sequential Monte Carlo) method to compute the posterior and consequently the limit of learning. This method was used to calculated the entropic LOL case was provided as an example in the previous section. We remark that the algorithm is general and could be used to calculate the limit of learning in general in the well-specified case. We have developed software for this purpose, which is posted along with this report.

B. Computation of LOL in the Mis-Specified Case

There are several applications where the learner receives the data points one sample at a time, and would like to start building a model for the learning process. In particular, the learner would like to start from simpler hypotheses and start expanding to more complicated hypotheses as more data arrives. If this is implemented naively, it is possible for the data arrival pattern to lead to fluctuations in the selected model in an arbitrary model. On the other hand, smooth model selection may be desired. In this case, the learner would increase the model complexity smoothly and monotonically as new data samples arrive.

Next, we will present our proposed sequential model expansion algorithm in the mis-specified case. It includes tracking several experts (candidate models), and maintaining an active set of experts at each time (to reduce computational cost). Then, the algorithm would perform switching of the active subset when new data samples arrive. The algorithm would output a set of weights associated with the active experts that will be used to predict the next data point at each time. The full description of an algorithm that performs based on the described method is presented next.

Algorithm 1 Sequential model expansion

```
input $\{(\boldsymbol{y}_t, \boldsymbol{x}_t : t = 1, \dots, T\}$, $\eta > 0$, $\zeta \in [0,1]$, $w_{0,1} = 1, w_{0,2} = \cdots = w_{0,K} = 0$, candidate models $\mathcal{G} = \{\alpha_1, \alpha_2, \dots\}$,
    $s = 0$ ($\alpha_{s+1}, \dots, \alpha_{s+K}$ are the maintained active subsets of models), $K \in \mathbb{N}$, threshold $\rho \in [0,1]$
output $\boldsymbol{p}_t = [p_{t,1}, \dots, p_{t,K}]^{\mathrm{T}}$ (predictive distribution over the active models) for each $t = 1, \dots, T$
1: for $t = 1 \to T$ do
2:     Obtain $y_t$ and compute $v_{t,k} = w_{t-1,k} \exp\{-\eta \underline{\mathcal{L}_n}[\alpha_{s+k}]\}$ for each $k = 1, \dots, K$, where $\underline{\mathcal{L}_n}[\alpha]$ was defined using all
       the data up to time step $t$
3:     Let
       $$w_{t,k} = \begin{cases} (1-\zeta)\ v_{t,k} & \text{if } k = 1 \\ (1-\zeta)\ v_{t,k} + \zeta\ v_{t,k-1} & \text{if } 1 < k < K \\ v_{t,k} + \zeta\ v_{t,k-1} & \text{if } k = K \end{cases}$$
4:     Let $p_{t,k} = (\sum_{k=1}^{K} w_{t,k})^{-1} w_{t,k}$, $k = 1, \dots, K$
5:     if $p_{t,1} \le \rho$ and $p_{t,K} \ge 1 - \rho$ then
6:         Let $s = s + 1$
7:         Let $w_{t,k} = w_{t,k'}$, where $k = 1, \dots, K$ and $k' = (k + 1 \bmod K)$ (relabeling the active models)
8:     end if
9: end for
```

Finally, we remark that we can use surrogates for the limits of learning for dependent data as well. For example, cross-validation is used as a surrogate for the prediction loss. These are the subjects of our current investigation.

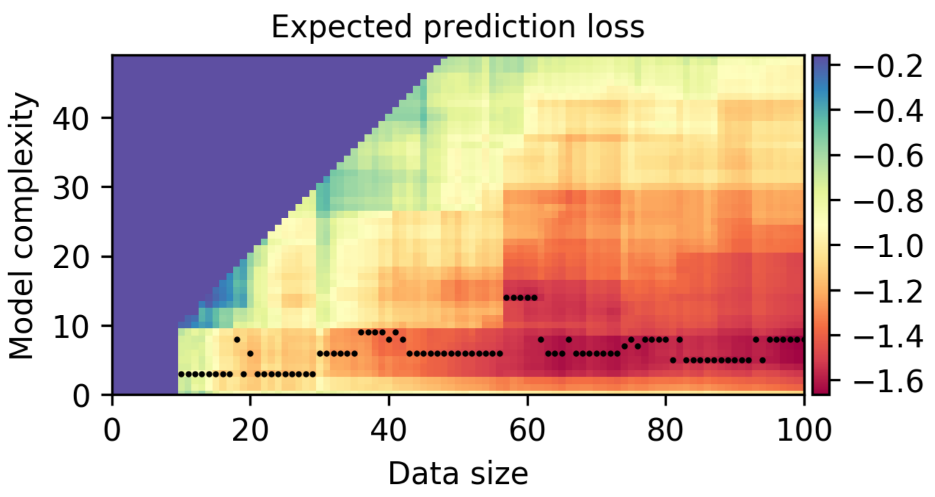
C. Numerical Experiments

We experiment on two model classes: logistic regression and single-layer feed-forward neural networks.

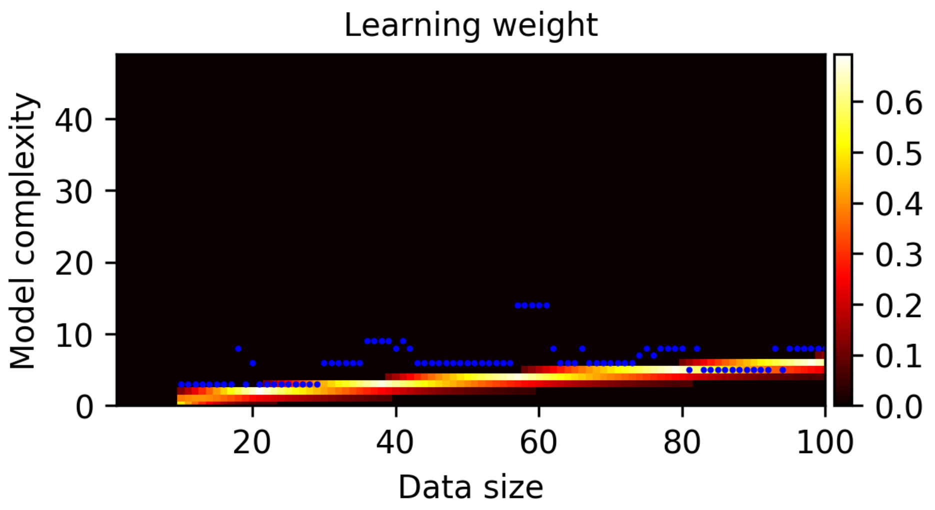
We implemented and released a python package "gtic" at *https://pypi.python.org/pypi/gtic*, in which we build a tensor graph of GTIC upon the *theano* platform. Users can simply input in their tensor variables of data and parameters, and obtain the GTIC instantly.

Logistic Regression Models

We consider the model class to be logistic regression. We generate data from a logistic regression model, where the coefficient vector is $\boldsymbol{\beta} = 10 \times [1^{-1.5}, \dots, 100^{-1.5}]^{\mathrm{T}}$, and covariates $x_1, \dots, x_{100}$ are independent standard



(a) Heat-map showing the true 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.

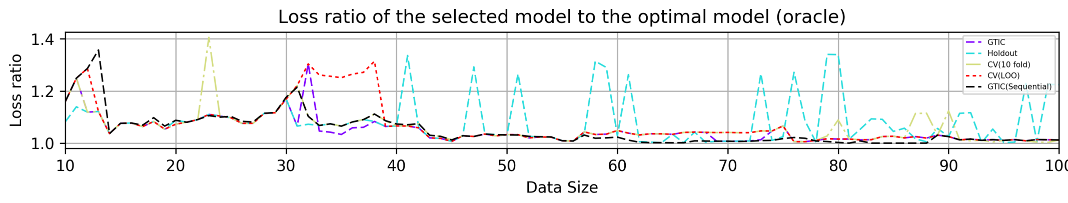


(b) Heat-map showing our predictive weights over the candidate models (y-axis) at each data size (x-axis), using sequential model expansion.

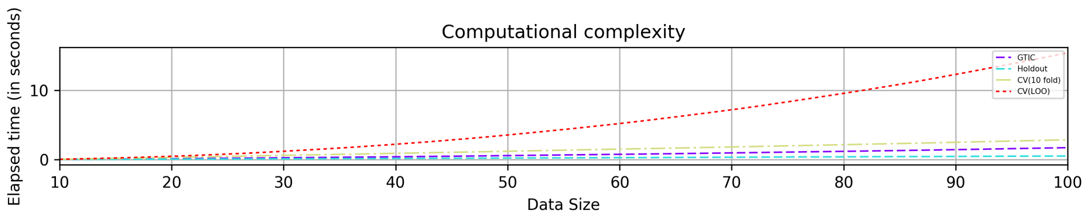
Fig. 3: Experiment 1: logistic regression models

Gaussian. Suppose that we sequentially obtain and learn the data, starting from $t = 10$, and then $t = 11, \dots, 100$. We restrict the maximum dimension of candidate models to be $\lfloor \sqrt{t} \rfloor$ at time t (see our theoretical assumptions). Here, a model of dimension d means that the first d covariates are nonzero. The model class is nested because a small model is a special case of a large model. We summarize the results in Fig. 3 and 4.

To illustrate the efficiency of GTIC, we first simulate model selection results with batch data. We numerically compute the true prediction loss of each trained model (obtained by testing on a large dataset), and then identify the optimal model (with the least loss). In Fig. 4a, 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 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 bit. Our proposed sequential model expansion algorithm smoothly expands the model

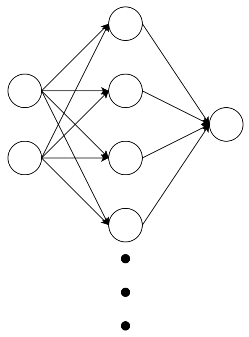


(a) Plot showing the loss of our predictor (GTIC) and cross validations at each data size

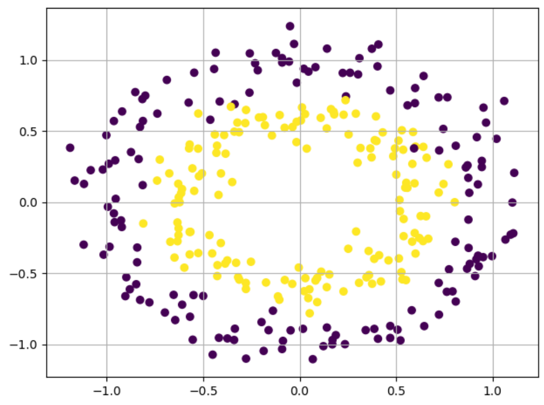


(b) Plot showing the computational costs.

Fig. 4: Experiment 1: logistic regression models



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

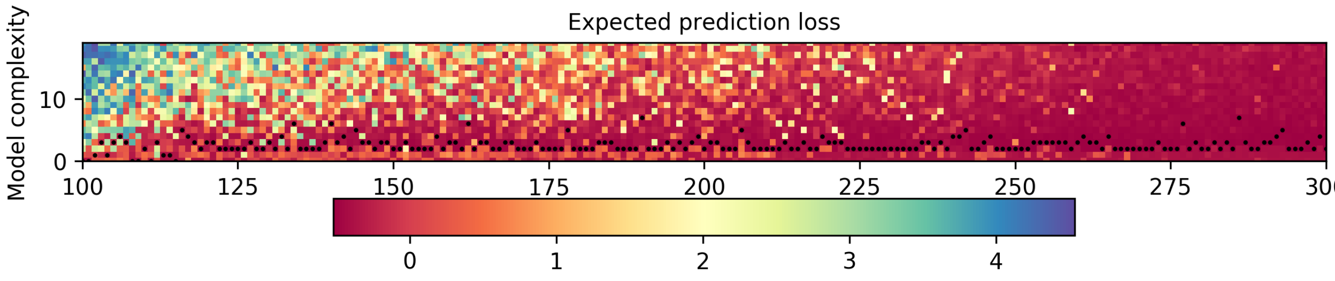


(b) A set of 300 data uniformly sampled from two circles corrupted by Gaussian noise ($\mu = 0$, $\sigma^2 = 0.1$, radius ratio $= 0.6$)

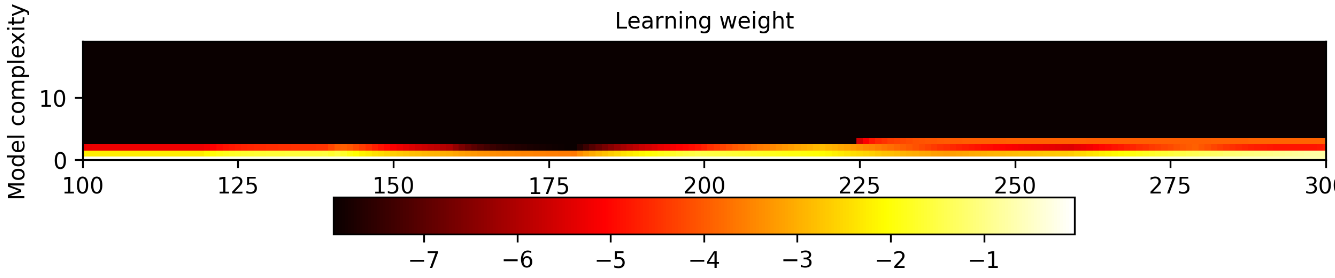
Fig. 5: Experiment 2: neural networks

and yields the best performance compared to all the other approaches. As shown in Fig. 3a and 3b, 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. This result is consistent with our definition of efficient Learning.

The computation cost of all approaches is provided in Fig. 4b. 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. However, depending on the problem and data, we may need 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 the optimal number of fold for CV. With GTIC, we do not need to sacrifice much on computation cost, but can still achieve theoretically justifiable result which is as good as LOO.

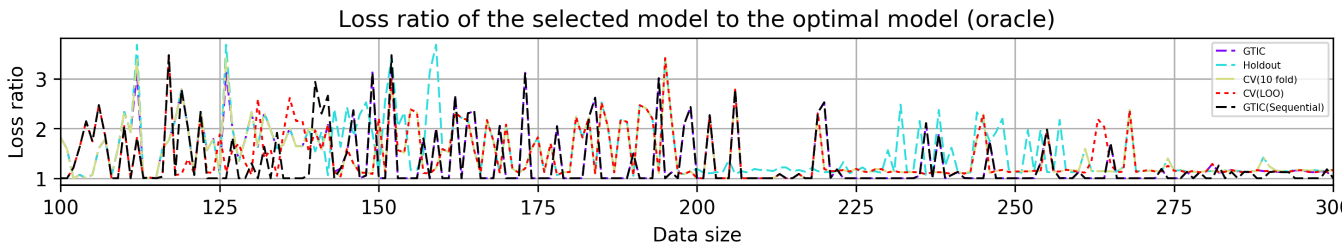


(a) Heat-map showing the prediction loss of estimated candidate models of each dimension (y-axis) at each data size (x-axis), where the black dots indicate the model of optimal loss at each data size.

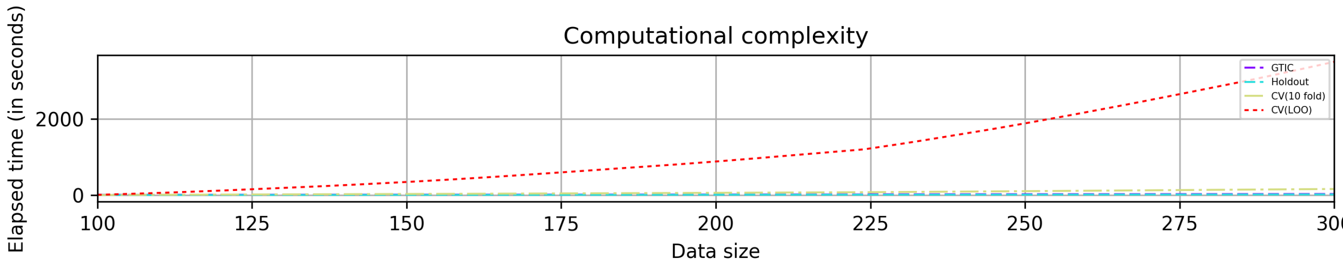


(b) Heat-map showing our predictive weights over the candidate models (y-axis) at each data size (x-axis).

Fig. 6: Experiment 2: neural networks



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



(b) Plot showing the computational costs.

Fig. 7: Experiment 2: neural networks

Neural Networks

We consider the model class to be single-layer feed-forward neural networks (see Fig. 5a). Neural networks are inherently miss-specified 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. And because we want to classify into two groups, the output dimension is one. In this experiment, the model complexity of our model is the number of hidden nodes in the single hidden layer.

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 small sample size. 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 than one hidden node to accurately classify the data. 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 data size is shown in Fig. 6a. 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. In Fig. 7a, the loss ratio varies quite a lot when the sample size is small, but gradually converges. This is partially because the influence of overfitting on the predictive power decreases as sample size increases. In other words, even if we choose a model that is slightly overfitting, the loss ratio is still close to one. Our proposed sequential algorithm is superior to other approaches as shown in Fig. 6b, 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 when their loss is relatively small.

The computational cost is shown in Fig. 7b. 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 a benefit that CV cannot enjoy in naive sequential model selection framework. Therefore, we encourage the use of GTIC in sequential model expansion scheme.

VI. Non-Parametric Models

Thus far, our focus was on learning using parametric models. The second area for extension is to consider non-parametric models, which is the subject of this section. In fact, we have derived e a preliminary algorithm for the computation of the limit of learning in a non-parametric model setting as well [4]. The algorithm is based on the ϵ-covering of the function space, and then designing appropriately vanishing rate of ϵ as the sample size increases. This includes learning from different experts and spaces.

An algorithmic description of our prediction procedure is summarized in Algorithm 2. We then discuss the related works, examples, and rigorous theoretical analysis.

We note that when $\eta = 1$, $\alpha = 0$, and the scoring rule is the negative log likelihood of the data, Algorithm 2 becomes the standard Bayesian posterior update. Smaller η and properly chosen nonzero α give a “tempering” effect on the weight updating, offering more flexibility/tolerance for potential underlying changes, while producing different rates of convergence.

Algorithm 2 Kinetic Prediction with Discretized Function Space

```
input Discretization parameter $\varepsilon$ and a model class $\mathcal{G} \subset \mathscr{G}$, data $\{y_t : t = 1, \ldots, T\}$ observed sequentially, learning parameter
    $\eta > 0$, mixing parameter $\alpha \in (0, 1)$.
output $\{\boldsymbol{p}_t \in \mathcal{R}_n^N : t = 1, \ldots, T\}$ (predictive distributions over the function bases), and $\{\hat{g}_t : t = 1, \ldots, T\}$ (predicted data
    generating densities).
1: Choose an $\varepsilon$-net for $\mathcal{G}$ (preferably with minimal cardinality), and obtain function base $\{g_1^{(\varepsilon)}, \ldots, g_N^{(\varepsilon)}\}$ (with cardinality
    $N$ as described above)
2: Initialization: $w_{0,1} = \cdots = w_{0,N} = 1/N$;
3: for $t = 1 \to T$ do
4:     Predict $\hat{g}_t$ according to the distribution $\{\boldsymbol{p}_t\}$ (using one of the methods outlined in Theorem ?? below) with $p_{t,i} =$
       $(\sum_{j=1}^N w_{t-1,j})^{-1} w_{t-1,i}$, $i = 1, \ldots, N$;
5:     Observe (incoming data) $y_t$ and compute $v_{t,i} = w_{t-1,i} \exp\{-\eta \; s(g_i^{(\varepsilon)}, y_t)\}$ for each $i = 1, \ldots, N$
6:     Let $w_{t,i} = (1 - \alpha) v_{t,i} + \alpha N^{-1} \sum_{j=1}^N v_{t,j}$ for each $i = 1, \ldots, N$.
7: end for
```

A. Synthetic Data Experiment: Abrupt Changes in Nonparametric Densities

Example 1 (Unknown distributions with unknown abrupt changes). *Suppose that Y_t at different t's are independent random variables on $[0, 1]$, with true density functions in the form of*

$$g_t^*(y) = \begin{cases} 3.25 - (c_t - y) & \textit{if } c_t - 0.25 \le y < c_t \\ 3.25 - (y - c_t) & \textit{if } c_t \le y < c_t + 0.25 \\ 0.25 & \textit{otherwise} \end{cases} \tag{9}$$

where $c_t \in [0.25, 0.75]$. In other words, the true densities are in the form of a constant plus a triangle centered at c_t. Suppose that $c_1, \ldots, c_T$ are piecewise constant, with abrupt changes at $\lfloor T/2^k \rfloor, k = 1, \ldots, \lfloor \log_2(T) \rfloor$, and with values in each segment alternating between 0.25 and 0.75.

If we are to use logarithmic scoring rule, can we come up with $\hat{g}_t$'s that achieve the oracle bound in the sense that

$$\frac{1}{T} \sum_{t=1}^{T} \{-\log \hat{g}_t(Y_t) + \log g_t^*(Y_t)\} \to_{a.s.} 0. \tag{10}$$

It is a challenging prediction task, since the locations and number of abrupt changes, as well as the form of densities are completely unknown. Our proposed methodology will solve this challenge under some mild assumptions on g_t^*'s e.g. we will require these to belong to various function spaces and to satisfy some mathematical properties such as Lipschitz condition. We will also assume that the number of change points is sub-linear in T.

The purpose of this experiment is to demonstrate the application of kinetic prediction to nonparametric model classes, where densities are not necessarily differentiable and are only known to belong to the Lipschitz class. We generate synthetic data from Example 1. By implementing Algorithm 2 and the strategy discussed in [4], we obtain Fig. 8 which shows the sequential predictive weights and snapshots of the predictors at time $t = 250, 500$. As we can see from the plot, the distributional changes in predictive weights mostly capture the true change points (especially around $t = 250$ when there are sufficient data before and after that change). To show the convergence, we also repeat the experiments for different T's ($T = 50, 100, 150, 250, 500$) and estimate the difference between the average loss of our predictor and the oracle (i.e. the left term in (10)). The estimates and their standard errors are shown in Fig. 9. As our theory expects, the differences converge to zero.

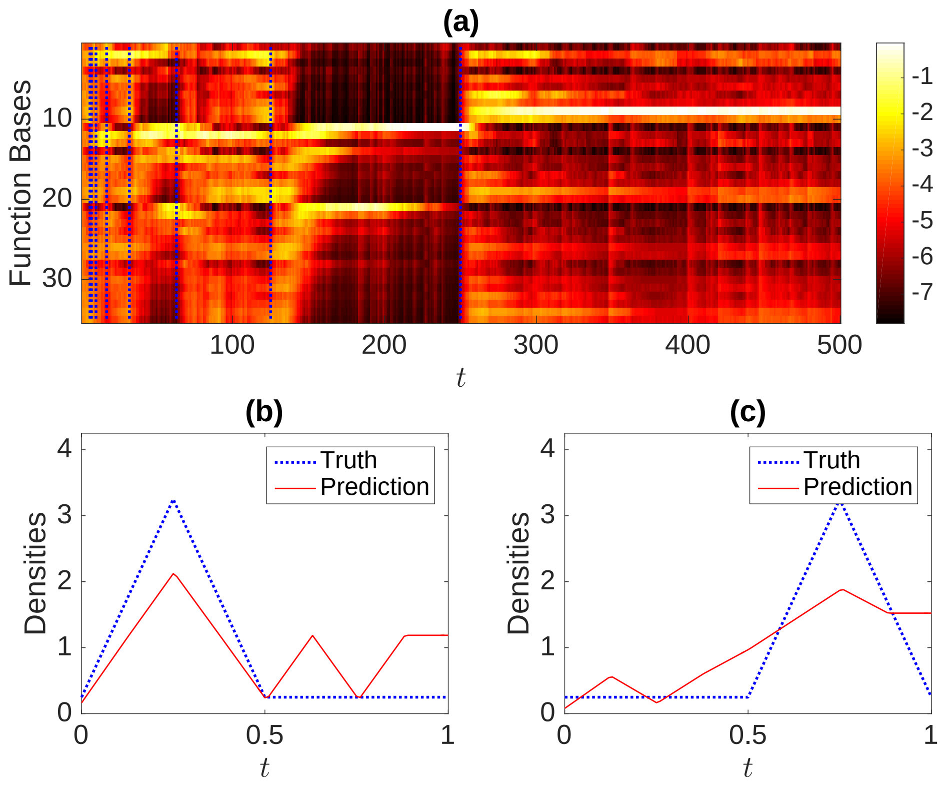


Fig. 8: Synthetic data experiment in Subsection VI-A: (a) the heatmap showing the sequential predictive weights (in log scale) over the function bases, along with the true locations of abrupt changes marked in green dashes, and (b) true and predicted density functions at $t = 250$ and $t = 500$.

B. Synthetic Data Experiment: Abrupt Changes and Smooth Variations in Mean

The purpose of this experiment is to demonstrate the application of kinetic prediction to parametric models with both abrupt changes and smooth variations. We consider the independent Gaussian model $Y_t \sim \mathcal{N}(\mu_t, 1)$, with parameter space $\mathcal{Y} = [-10, 10]$. The time-varying means μ_t's consist of four segments, first a quadratic trend, then two linear trends with different slopes, followed by a cosine pattern. Each switch from one segment to another is abrupt. The mathematical formula is

$$\mu_t = \begin{cases} 5 - T^{-2}(t - T_1/2)^2 & \text{if } t \le T_1 \\ -7 + 60(t - T_1)/T & \text{if } T_1 < t \le T_2 \\ -7 + 20(t - T_2)/T & \text{if } T_2 < t \le T_3 \\ 2 + 5\sin\{6\pi(t - T_3)/(T - T_3)\} & \text{otherwise} \end{cases}$$

where $T_k \triangleq \lfloor kT/4 \rfloor$ for $k = 1, 2, 3$.

We used quadratic loss $(Y_t - \hat{\theta}_t)^2/2$ to evaluate predictive performance in the algorithm. The multiplier $1/2$ is not theoretically essential, but it has the natural interpretation of negative log-likelihood of Gaussian observations with unit variance. By implementing our proposed algorithm, we obtain Fig. 10(a) which shows the sequential predictive weights. As we can see from the plot, our predictive weights capture the true trends, and switch promptly after abrupt changes occur. Fig. 10(b) plots our predictor versus true mean at each time step, along with the realization of data. The prediction well matches the truth in general. We also repeat the experiments for different

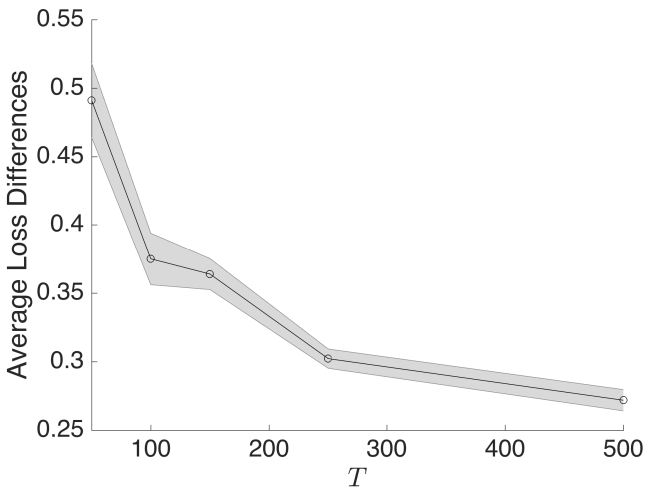


Fig. 9: Synthetic data experiment in Subsection VI-A: the average loss of our predictor minus the oracle for different T's, where each point was computed from 10 repeated experiments and the shaded region describes the $+1/-1$ standard errors.

T's ($T = 50, 100, 150, 250, 500$) and estimate the difference between the average loss of our predictor and the oracle. The estimates and their standard errors are plotted in Fig. 11, showing the convergence to zero.

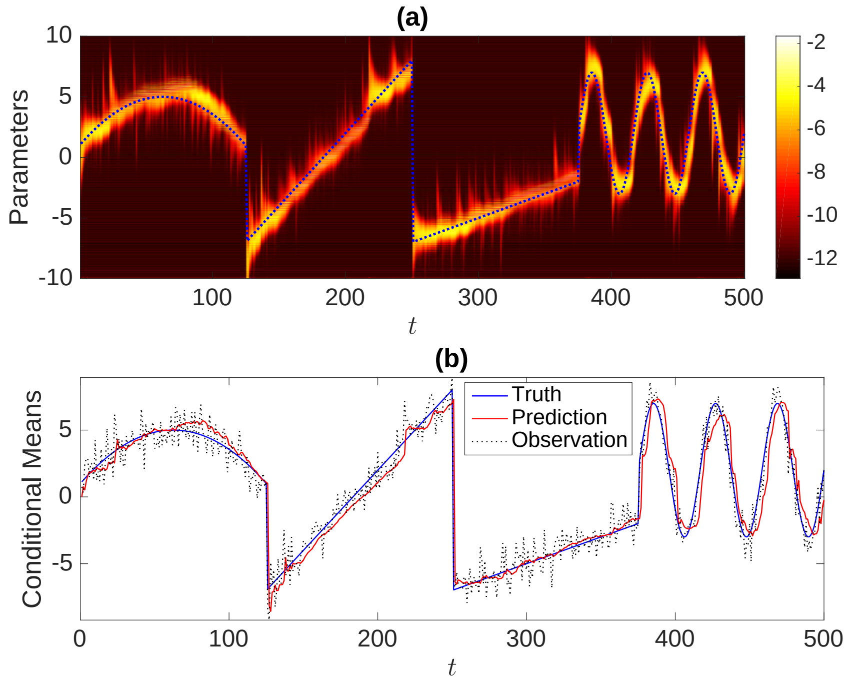


Fig. 10: Synthetic data experiment in Subsection VI-B: (a) the heatmap showing the sequential predictive weights (in log scale) over the parameter bases, along with the true parameters at each time step marked in blue dashes, and (b) the true and predicted mean at each time step, along with the observed data.

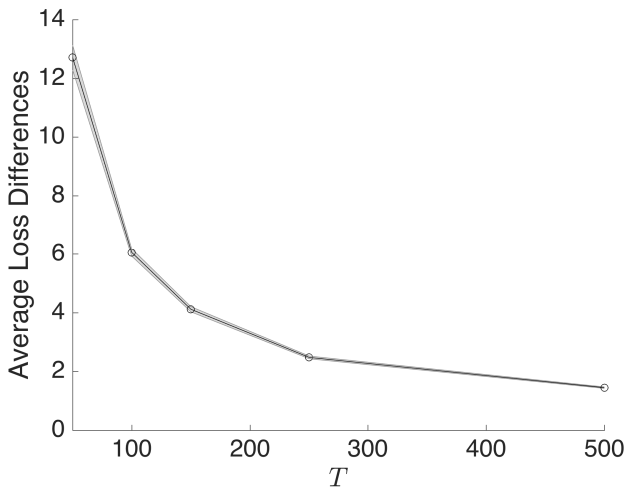


Fig. 11: Synthetic data experiment in Subsection VI-B: the average loss of our predictor minus the oracle for different T's, where each point was computed from 10 repeated experiments and the shaded region describes the $+1/-1$ standard errors.

C. Extensions

In fact, for parametric models, instead of discretizing the parameter space, we propose a novel algorithm to directly update over the continuous parameter space. The algorithm can be regarded as an extension of the previous ϵ-covering techniques. An algorithmic description of our prediction procedure is summarized in Algorithm 3. A Monte Carlo based approach for efficient implementation is summarized in Algorithm 4.

```
**Algorithm 3** Kinetic Prediction with Continuous Parameter Space
Input: Compact parameter space $\Theta \subseteq \mathcal{R}_n^n$, data $\{y_t, t = 1, \ldots, T\}$ observed sequentially, learning parameter $\eta > 0$,
    and mixing parameter $\alpha \in [0, 1)$.
Output: Predictive distribution of the unknown parameter $\{p_t(\theta), t = 1, \ldots, T\}$ and predicted sequence $\{\hat{\theta}_t\}_{t=1}^T$.
1: Initialization: $f_0(\theta) = 1$, $\forall \theta \in \Theta$;
2: for $t = 1 \to T$ do
3:     Predict parameter $\hat{\theta}_t$ according to the predictive distribution $p_t(\theta) = \{\int_\Theta f_{t-1}(\tilde{\theta}) d\tilde{\theta}\}^{-1} f_{t-1}(\theta)$
4:     After receiving $y_t$, compute the score function $s(\theta, y_t)$ for all $\theta \in \Theta$;
5:     Update:
           $\tilde{f}_t(\theta) = f_{t-1}(\theta) \cdot e^{-\eta s(\theta, y_t)}$        (11)
6:     Mix:
           $f_t(\theta) = (1-\alpha)\tilde{f}_t(\theta) + \alpha \frac{\int_\Theta \tilde{f}_t(\tilde{\theta}) d\tilde{\theta}}{|\Theta|}$        (12)
7: end for
```

Algorithm 4 Sequential Monte Carlo for Kinetic Prediction (SMC-KP) in Compact Parameter Spaces

```
Input: Compact parameter space $\Theta \subseteq \mathcal{R}_n^n$, data $\{y_t, t = 1, \ldots, T\}$ observed sequentially, learning parameter $\eta > 0$,
    mixing parameter $\alpha \in (0, 1)$, number of particles $N$, and ESS threshold $c \in [0, 1]$.
Output: $(W_t^i, \theta_t^i)_{i=1}^N$ as weighted samples from the predictive distribution $\{p_t(\theta), t = 1, \ldots, T\}$ defined in
    Algorithm 3.
1: Initialization: sample $\theta_1^i$ independently from $\mathrm{Unif}(\Theta)$, and let $W_1^i = N^{-1}$, for all $i = 1, \ldots, N$.
2: for $t = 1 \to T$ do
3:   Use the weighted samples $(W_t^i, \theta_t^i)_{i=1}^N$ to approximate the predictive distribution $p_t(\theta)$.
4:   Receive/Read $y_t$.
5:   Update $w_t^i = W_t^i \cdot e^{-\eta s(\theta_t^i, y_t)}$ for all $i = 1, \ldots, N$;
6:   Calculate $\tilde{R}_t = \sum_{i=1}^N w_t^i$, $\hat{R}_t = \prod_{\tau=1}^t \tilde{R}_\tau$, (to be used in Step (11));
7:   Normalize $W_{t+1}^i = w_t^i / \sum_{i=1}^N w_t^i$. Use $(W_{t+1}^i, \theta_t^i)_{i=1}^N$ to approximate $\tilde{f}_t(\theta)$;
8:   Calculate $ESS = 1/\big(\sum_{i=1}^N (W_t^i)^2\big)$;
9:   if $ESS < cN$ then
10:    Resample:
       Resample $(W_{t+1}^i, \theta_t^i)_{i=1}^N$ to obtain equally weighted samples $(W_{t+1}^i = N^{-1}, \theta_t^i)_{i=1}^N$;
11:    Rejuvenate/Move:
       Draw $\theta_{t+1}^i \sim K'(\cdot|\theta_t^i)$ where $K'$ is an MCMC kernel targeting at $\tilde{f}_t(\theta)$;
12:  else
13:    $W_{t+1}^i = W_{t+1}^i$, $\theta_{t+1}^i = \theta_t^i$;
14:  end if
15:  Move $\theta_{t+1}^i$ according to the transition kernel $K(\cdot|\theta_{t+1}^i)$ in the following way: with probability $1 - \alpha$, let
     $\theta_{t+1}^i = \theta_{t+1}^i$, and with probability $\alpha$, let $\theta_{t+1}^i \sim \mathrm{Unif}(\Theta)$, for all $i = 1, \ldots, N$.
16: end for
```

In the sequel, we provide two real-world applications of our proposed method.

D. Real Data Experiment: Discovering Time-Varying Cointegration

The purpose of this experiment is to apply kinetic prediction to discover time-varying cointegration [5], [6] in financial applications. The panel data we used (denoted by $\boldsymbol{Y}$) consist of 390 stock prices of Apple Inc. (denoted by $Y_{1,t}$) and Alphabet Inc. Class A (denoted by $Y_{2,t}$), collected every half hour starting from November 17, 2015. The data were standardized using a mean and standard deviation that were calculated using historical data collected before that date. We note that a linear transform of the raw data does not cause essential difference in Algorithms 3 and 4, but it makes easier to plot. The pre-processed data are shown in Fig. 12(a). Each of $Y_{1,t}, Y_{2,t}$ is a process integrated of order 1, according to the augmented Dickey-Fuller test under 0.01 significance level.

Suppose that there exist two deterministic series $a_{0,t}, a_{1,t}$ with a few unknown abrupt changes, such that $Y_{2,t} - (a_{0,t} + a_{1,t} Y_{1,t})$ is stationary. This time-varying cointegration may be more reasonable than a fixed-parameter cointegration, considering potential regime changes in the financial market. We are primarily interested in a sequential context, where the estimates of $a_{0,t}, a_{1,t}$ depends only on $\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_{t-1}$, and we hope that "the predictive residues"

$$R_t \triangleq Y_{2,t} - (\hat{a}_{0,t} + \hat{a}_{1,t} Y_{1,t}) \tag{13}$$

become stationary. We first apply Algorithm 4 with the quadratic score $s : (\theta, y_t) \mapsto (y_{2,t} - a_0 - a_1 y_{1,t})^2/2$, parameter space $a_{0,t}, a_{1,t} \in [-2, 2]$, and default tuning parameters as described in [4]. The means and standard deviations of parameters computed from our predictive distribution at each time step are plotted in Fig. 12(c)&(d). We use those means as $\hat{a}_{0,t}, \hat{a}_{1,t}$ and compute the predictive residues in (13), and plot the sequence in Fig. 12(b) (abbreviated as "Kinetic"). For comparison, we also apply classical recursive least squares (RLS) method to estimate $\hat{a}_{0,t}, \hat{a}_{1,t}$, and plot the corresponding predictive residues in Fig. 12(b) (abbreviated as "RLS"). For numerical stability, all the results are computed starting from $t = 6$. From Fig. 12, we can see that the parameters do have significant regime changes over time, especially for the intercept term. The residue sequence given by "Kinetic" is stationary, while that given by "RLS" is not, according to the Dickey-Fuller test under 0.01 significance level. Visually, the former sequence does look more stationary as they fluctuate more frequently around zero (marked by a black dash).

We plot the average score $\sum_{i=1}^{t} R_i^2/t$ given by two methods at each time step t given that $T = 390$ in Fig. 13(a). We also compute the average score at the last time step, namely $\sum_{t=1}^{T} R_t^2/T$ for $T = 50, 100, 150, 250, 390$ (with Algorithm 4 running on corresponding tuning parameters), and plot the score of "RLS" minus that of "Kinetic" in Fig. 13(b). The results show that "Kinetic" is dominantly better than "RLS" method also in terms of scores.

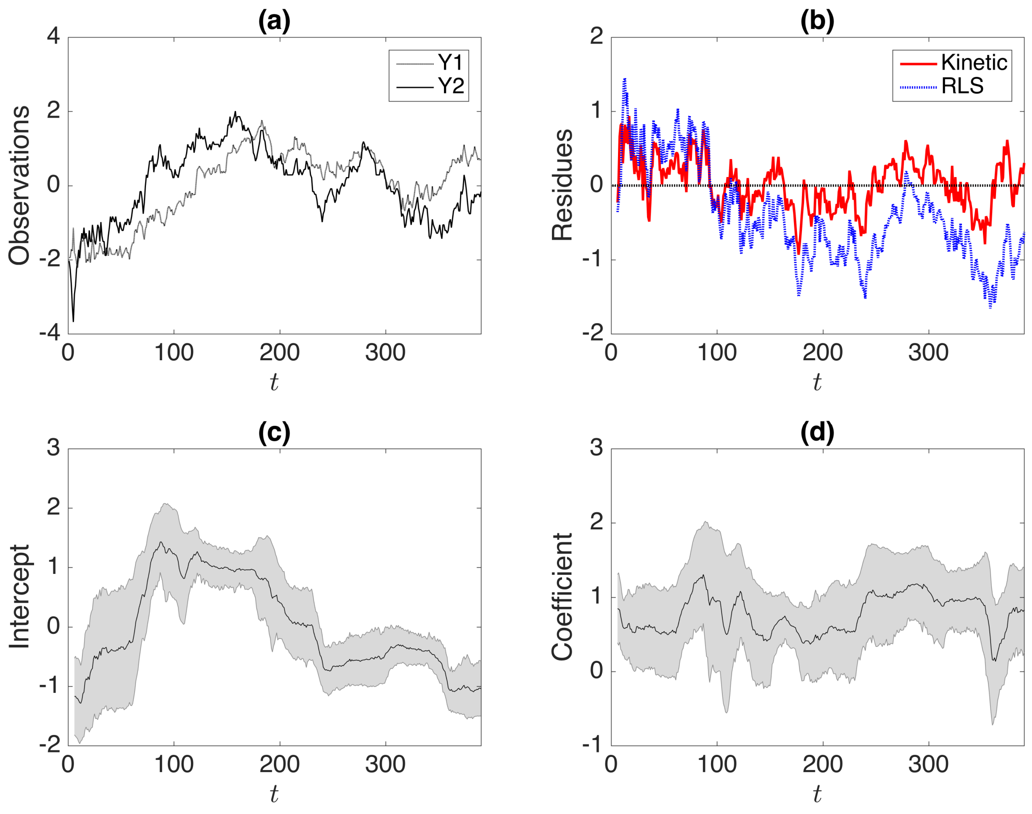


Fig. 12: Real data experiment in Subsection VI-D: (a) the preprocessed observations, (b) the cointegration residues of two prediction methods, (c)&(d) the sequential predictive means of the intercept a_0 and coefficient a_1 in cointegration, along with their +1/-1 standard deviations.

E. *Real Data Experiment: Predicting Stochastic Volatilities*

The purpose of this experiment is to apply kinetic prediction to predict stochastic volatilities in a financial market. Forecasting volatility plays an important role in risk management and asset allocation. To model financial time series with time-varying volatility, the autoregressive conditional heteroskedasticity (ARCH) models [7] and

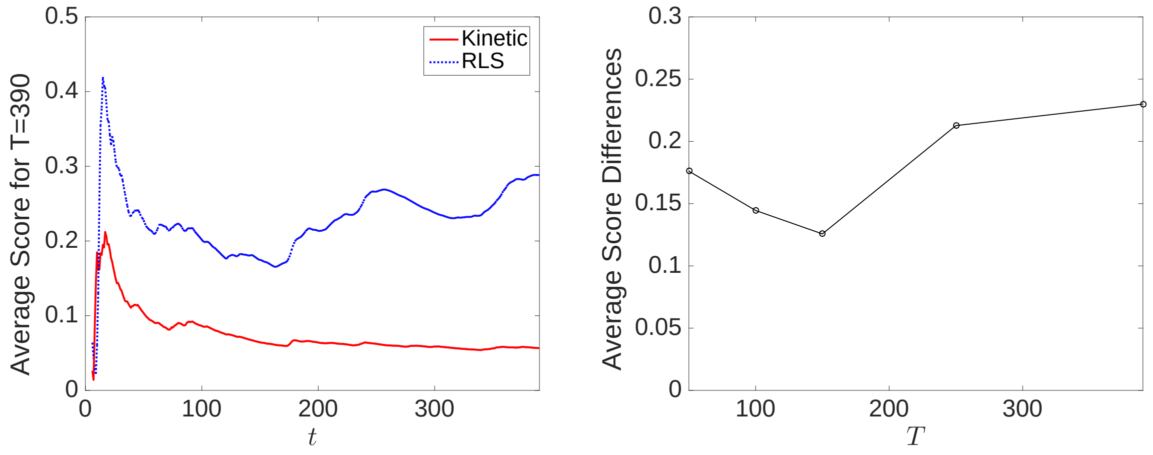


Fig. 13: Real data experiment in Subsection VI-D: (a) the average scores of two methods at each time step for $T = 390$, (b) the average score of our kinetic predictor minus that of the recursive least squares method for different T's.

the generalized ARCH (GARCH) [8] have been commonly adopted. We refer to [9] for an extensive literature on stochastic volatility models.

We collected 500 daily stock prices of SPDR S&P 500 ETF (SPDR), denoted by s_t, from Jan 6, 2007 to 31 Dec 08. Let $Y_t = 10^2 \log(s_t/s_{t1})$ be the so-called log-returns (where the scaling is done for numerical convenience). The squared log-returns are shown in Fig. 14(a) in black dash. We note that the squared return on an asset at one time step (assuming a zero mean return) can be interpreted as a conditionally unbiased estimator of the true but unobserved conditional variance of the asset. After some exploratory studies, we adopt the following GARCH(1,1) as the parametric model class,

$$v_t^2 = a_0 + a_1 Y_{t-1}^2 + b_1 v_{t-1}^2, \quad Y_t = v_t \varepsilon_t, \tag{14}$$

where ε_t's are i.i.d. $\mathcal{N}(0,1)$ noises. In other words, the true (but unobservable) volatility v_t at each time step t only depends on the squared observation Y_{t-1} and the true volatility in the last time step. We used negative log-likelihood as the score function. In other words, $s : (\theta, y_t) \mapsto (\log h_t + h_t^{-1} y_t^2)/2$ is used, where h_t is the predicted volatility at time step t corresponding to $\theta \triangleq [a_0, a_1, b_1]$. Here, h_t can be recursively computed using (14), θ, and $v_1 \triangleq Y_1^2$. Rigorously speaking, the above score is predictive log-likelihood, since h_t is computed using all the data before time step t in a sequential setting [10]. It is worth noting that in the volatility forecasting literature, it is also common to write a score function in the form of $L(v_t^2, h_t)$, and use volatility proxies such as the squared return Y_t^2 in place of v_t^2. Our scoring function is equivalent to the so called "QLIKE" loss functions, which is proved to be robust in the sense of [11, Definition 1]. Squared score in the form of $(Y_t^2 - h_t)^2$ is also commonly used, but it seems more sensitive to extreme observations and the level of volatility of returns. For other widely-used score functions used to evaluate conditional variance forecasting, we refer to [11] and references therein.

We suppose that the true parameters $a_{0,t}, a_{1,t}, b_{1,t}$ at each time step t may not be all the same (e.g. during a financial crisis). We first apply Algorithm 4 with parameter space $a_{0,t}, a_{1,t}, b_{1,t} \in [0,1]$ (which is standard for GARCH), and default tuning parameters as described in [4]. The means and standard deviations of parameters computed from our predictive distribution at each time step are plotted in Fig. 14(b)(c)(d). We use those means to compute the predictive volatilities, and plot the sequence in Fig. 14(a) (abbreviated as "Kinetic"). For comparison, we also apply a fixed-parameter GARCH(1,1) to estimate the parameters at each time t, using $y_1, \ldots, y_{t-1}$. We then plot the corresponding predictive volatilities in Fig. 14(a) (abbreviated as "GARCH").

We plot the average scores of two methods at each time step t given $T = 500$ in Fig. 15(a). We also compute the average score at the last time step $t = T$, for $T = 50, 100, 150, 250, 390$ (with Algorithm 4 running on corresponding tuning parameters), and plot the scores of "GARCH" minus those of "Kinetic" in Fig. 15(b). The results show that "Kinetic" is dominantly better than "GARCH" method also in terms of scores.

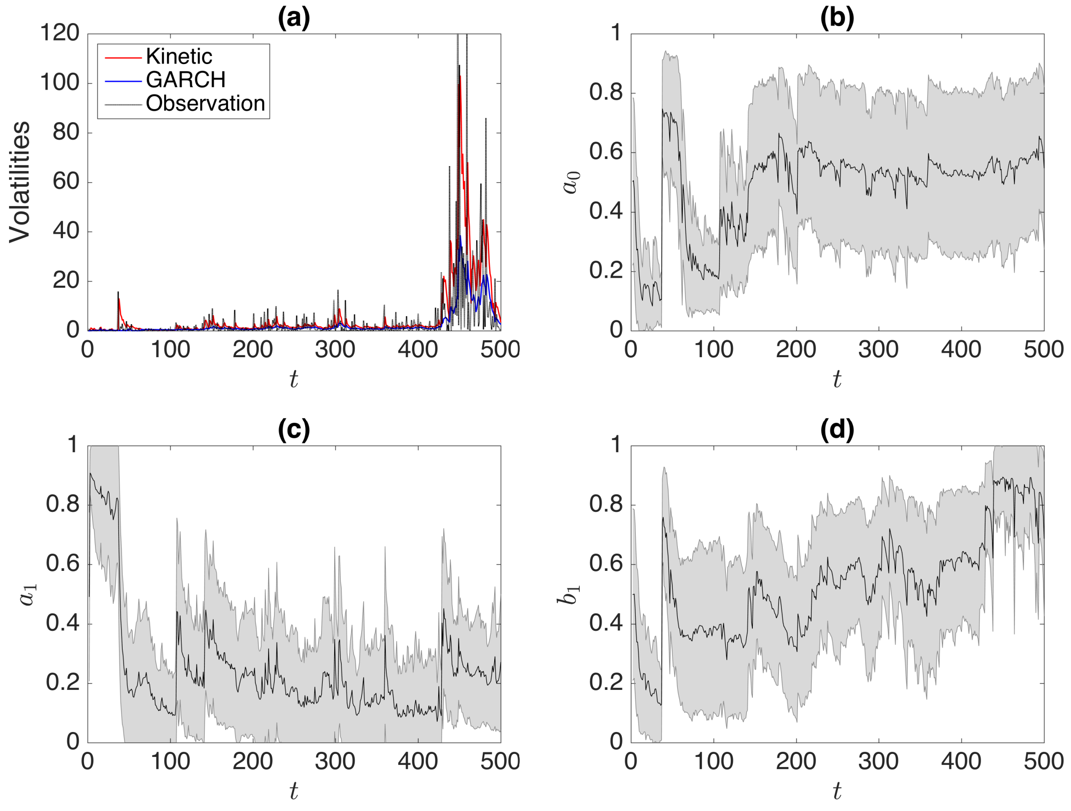


Fig. 14: Real data experiment in Subsection VI-E: (a) the predicted volatilities by kinetic and recursive GARCH methods, along with the squared log returns, (b)-(d) the sequential predictive means of the three coefficients a_0, a_1, b_1, along with their +1/-1 standard deviations.

VII. Predict-then-Interpret Methodology for Multivariate Polynomial Regression

In this part of work, we propose a novel heuristic approach toward the multivariate polynomial regression. It employs a Random Forest predictor as its first step to generate data points that we want. Then it uses this predictor to analyze top degrees and interaction between variables. Finally a least square regression is conducted on terms that are analyzed as likely to exist. The top-degree and interaction analysis allow us to eliminate a large number of irrelevant terms, thereby significantly reducing the complexity. Comparison with LASSO are conducted, which shows that our method is much more feasible when the number of explanatory variable is large.

The key novel points are elaborated in the following subsections.

A. Random Forest as First Step

We use a Random Forest predictor as the first step of our methodology. This serves as a black box predictor, which allows us to customarily set input vectors and obtain (predicted) output values. In the ideal case, the Random

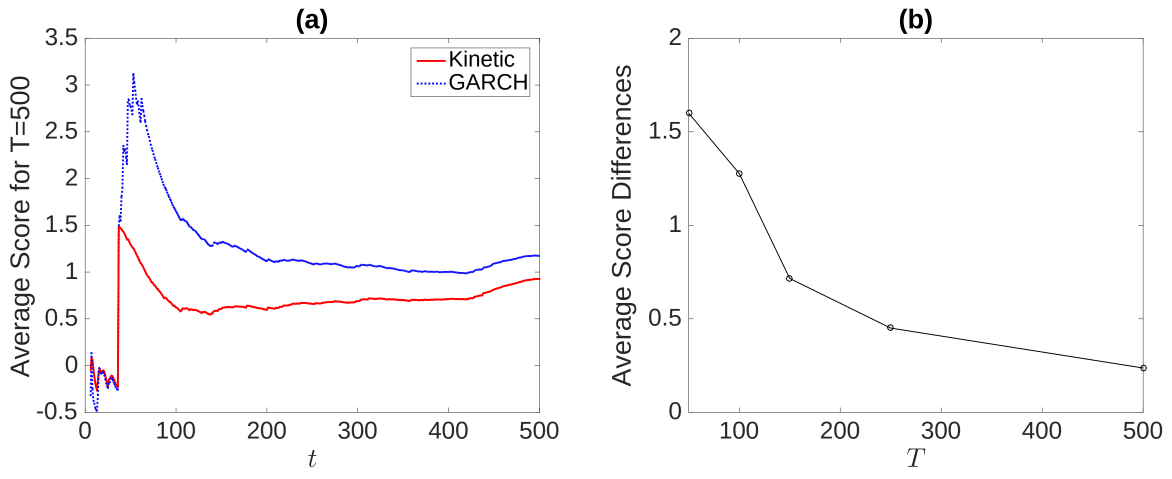


Fig. 15: Synthetic data experiment in Subsection VI-E: (a) the average scores of two methods at each time step for $T = 500$, (b) the average score of our kinetic predictor minus that of the recursive GARCH method for different T's.

Forest predictor will be an exact copy of the true underlying model, therefore we can gain much of the model from the black box predictor.

The reason we choose Random Forest is that this is one of the best-performing predictors we have right now. Its drawback is that a large training dataset is required to achieve satisfactory accuracy. This will be further explored in later sections.

Traditionally we have seen methods that conduct OLS or LASSO first and then fit a Random Forest to the residual. While that can be pretty predictive, it lacks interpretability because Random Forest is known to be a rather un-interpretable model. We here use Random Forest for a rough prediction, and that supplies information for our next regression steps.

B. *Estimate Top Degree of Each Variable*

We estimate the highest polynomial degree present in each variable. As an example, in $y = x_1^2 + x_1 x_2^2$, the highest degree of x_1 and x_2 are both 2. This step is based on our prior assumption that the true model is multivariate polynomial. In the case when it is not, it can still be well approximated by a polynomial expansion of certain degree. This step also has the effect of identifying variables that are irrelevant. They will be identified to have top degree 0. AIC is used as the criterion of choosing the optimal top degree value.

C. *Check Interactions between Variables*

We check whether interaction exists within a certain subset of variables, which is key to determining the actual model terms. By "interaction" we mean cross terms as polynomials. For example $x_1 \times x_2^2$ is an interaction between x_1 and x_2. We do the check by taking increments in input variable values and then comparing the corresponding increments in the predicted output. For example we wish to check whether interaction exists within 2 variables x_1 and x_2. We take a independent random increment on each variable, which changes x_i to $\tilde{x}_i = x_i + \Delta x_i, \quad i = 1, 2$. All other variables at hold are some constant value, denoted here by x', which will be explained later. Then we check whether the following inequality holds:

$$f(\tilde{x}_1, \tilde{x}_2, x') - f(x_1, x_2, x') = \left[f(\tilde{x}_1, x_2, x') - f(x_1, x_2, x')\right] + \left[f(x_1, \tilde{x}_2, x') - f(x_1, x_2, x')\right].$$

The intuition is that when there are no cross terms containing x_1 and x_2, the absence of interaction implies that increments on each variable should sum up to be equal to be the increment on all variables. This can be justified by considering derivatives of f. When cross terms are indeed nonexistent, the above equation will always hold. The above equation can be easily generalized to any subset of k variables, with k individual increments taking summation on the right hand side. Chi-square test is used for testing whether the equation holds.

Now we turn to the question of what values to set other variables. Let S be the subset of variables we are studying, and T be its complementary set. There are two choices for setting variables in T to: 0 or nonzero random values. Of course setting to 0 requires that 0 is in the range of variables, and we make that possible by first normalizing each variable. Both choices are useful and have different implications. When they are set to 0, the polynomial structure of model implies that any term that contains a variable in T would vanish, leaving only terms that are "purely" in S. This approach is useful when we wish to know exactly which terms are present in the model. On the other hand, setting variables in T to random nonzero values means detection only on the interaction level, not on the term level.

Here we provide two examples to illustrate this. Suppose we are detecting interaction between variables x_1, x_2, x_3. By setting all other variables to zero and checking whether the increment equation holds for the three variables, we can know whether any term in the form of $x_1^\alpha x_2^\beta x_3^\gamma$ exists, where at least 2 in α, β, γ are nonzero. On the other hand, suppose we are detecting interaction between two variables x_1 and x_2 and set other variables to random nonzero values, then we are actually detecting whether there exist any term that contains both x_1 and x_2, and possibly other variables.

VIII. Algorithm Description

The algorithm is described in the following table:

TABLE II: Algorithm for Multivariate Polynomial Regression

```
Input: Explanatory variables $\boldsymbol{X} \in \mathbb{R}^{N \times P}$, response variable $\boldsymbol{y} \in \mathbb{R}^N$, upper-bound of degree of each variable $D$.
Output: Regression results: Terms and Coefficients.

1  Build Random Forest predictor from data $\boldsymbol{X}$ and $\boldsymbol{y}$.
2  Estimate the top degree of each explanatory variable, capped at $D$.
3  Discard all explanatory variables that have top degree = 0.
4  For each pair of variables, check if they have any interaction by setting other variables to nonzero random values.
5  Group the variables into various "connected components", such that variables in different groups do not interact
   with each other.
6  Within each group, find all specific cross term interaction patterns by setting other variables to 0.
7  Expand these cross term interaction patterns into terms, according to their top degrees.
8  Conduct regression on the discovered terms.
```

IX. Experiments

Here we present some experiments that compare our method with the popular LASSO, in terms of running time cost and accuracy.

A. Time Costs

Below is the plot that compares the running time cost between our method and Matlab-integrated LASSO function. The real model is $y = x_1^2 + x_1x_2 - x_3 - 2x_3x_4^2 + x_4x_5 + 2x_5$, which contains 5 variables and with a marginal top degree of 2. All variables follow normal distribution $\mathcal{N}(0, 1)$. The expression "N_var q/p" in the legend indicates that there are p explanatory variables provided and q among them are actually relevant in real model. The horizontal axis is the upper bound of allowed degree, chosen as a prior. Since in this case the marginal top degree of true model is 2, ideally this value should be chosen no less than 2.

It should be noted that the "LASSO, 5/12" case has only the $D = 2$ point, because in both $D = 3$ and $D = 4$ situations the program simply couldn't run because too much memory space is required.

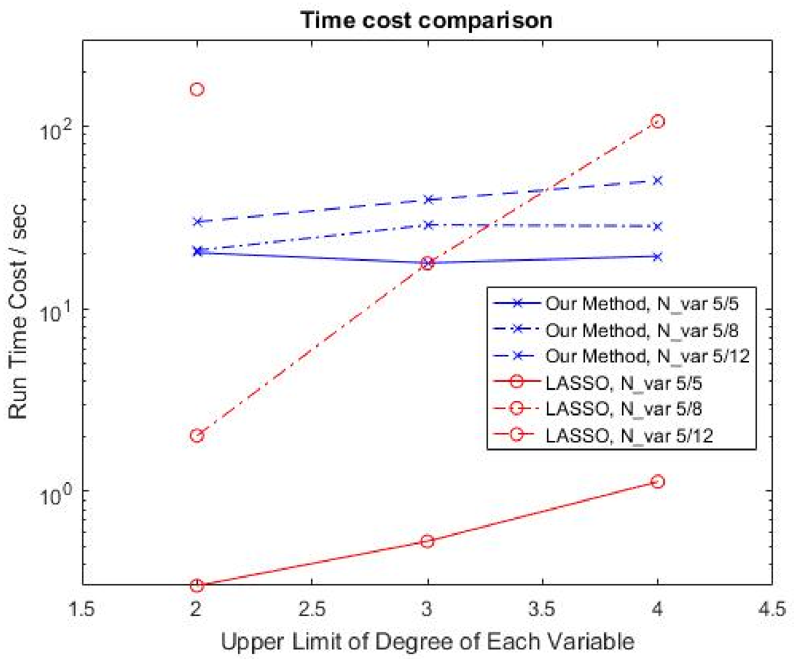


Fig. 16: The running time cost of our method and LASSO. N_var q/p indicates that there are p explanatory variables provided and q among them are actually relevant in real model.

It can be seen from the figure that while LASSO does perform extremely well in small variable size cases, it becomes very poor when the number of variables and the degree of the model grows, even becoming infeasible. By comparison, the time cost our proposed method is much more acceptable when the number of variables is very large, and it only increases mildly as the number of variables grows.

The result above is expectable, since the LASSO model doesn't even filter out irrelevant variables as its first step, which seems a really necessary step. However, when interaction is present between variables, unlike linear cases, it is no longer straightforward to see whether a variable is relevant to the model or not. As far as I see, a proper method is still to make use of a predictor such as Random Forest. So far I haven't seen any work in that form (and I inquired Jie about this as well), so it could potentially still be our innovation.

B. Accuracy

Now we test the prediction accuracy of the two methods. The test focuses on the following aspects:

- n_o – Number of terms overfitted in the prediction model.
- n_u – Number of terms underfitted in the prediction model.

- L – Loss of the prediction model, tested on independently generated test data.

We only test cases where both methods are viable. When the number of variables grows, LASSO becomes no longer a possible choice. Results are shown in the following two tables. Number of variables are respective 5/5 and 5/8. Additive Gaussian Noise is not present in our

TABLE III: Comparison of accuracy, N_var 5/5.

Degree Limit	Method	n_o	n_u	Loss L
2	Ours	0	1	0.1473
	LASSO	1	0	0.0122
3	Ours	0	0	0.0001
	LASSO	2	0	0.0131
4	Ours	0	1	0.5809
	LASSO	3	0	0.0057

TABLE IV: Comparison of accuracy, N_var 5/8.

Degree Limit	Method	n_o	n_u	Loss L
2	Ours	0	1	0.1145
	LASSO	1	0	0.0111
3	Ours	0	1	0.0661
	LASSO	3	0	0.0047
4	Ours	0	1	0.1308
	LASSO	10	0	0.0079

As can be seen, our method in most cases has one underfitted term, while LASSO generally has some overfitted terms. Having excessive terms does little harm to prediction, and that is why LASSO generally has a smaller loss. It should also be noted that in one case our method recovered the real terms perfectly and achieved almost zero error.

In all cases, the term that our method failed to identify is x_1x_2, and it fails with a reason. Since all variables are i.i.d. $\mathcal{N}(0,1)$, a cross-term such as x_1x_2 tends to be very close to 0, resulting in low SNR and low significance. In our particular example, the variable x_2 appears in this term only, so it is likely to be neglected when we are examining the top degree of each variables, and the mistake passes on to later steps.

Going further with this analysis, we can consider cases in real application, where variables are typically not zero-mean, and we need to normalize (shift) them first. During the shift, polynomial terms are expanded and new terms are generated. As an example suppose $x_i\ \mathcal{N}(\mu_i,\sigma_i^2),\quad i=1,2$, and the true model is $y=x_1x_2$. After normalization $\hat{x}_i=\frac{x_i-\mu_i}{\sigma_i}$, we get $y=\mu_1\mu_2+\mu_1\sigma_2\hat{x}_2+\mu_2\sigma_1\hat{x}_1+\sigma_1\sigma_2\hat{x}_1\hat{x}_2$. When a term is significant in the original true model, μ_1 and μ_2 are not likely to be very small, therefore the existence of terms$\mu_1\sigma_2\hat{x}_2+\mu_2\sigma_1\hat{x}_1$ would guarantee that we can get the correct top degrees, and therefore a more accurate prediction. In summary this is to say that, if our method missed a term (as what happened in our experiment), it is likely to be because the term is insignificant on its own. When the variables don't start with zero-mean, they are much less likely to be neglected.

C. Strengths and Weaknesses of Proposed Methodology

To sum things up, we list the strengths and weaknesses of our proposed methodology in the following:

Strengths:

1) Is able to work when the *total* number of explanatory variables is large but the number of *really relevant* variables is not very large. By comparison, LASSO and the common OLS methods will simply fail to operate.
2) Insensitive to the number of irrelevant variables provided.
3) Suits the multivariate polynomial hypothesis well, and offers insight into variable degrees and especially interaction patterns.

Weaknesses:

1) Is rather slow when the number of variables is small, compared with LASSO.
2) Requires a large dataset for training the predictor in the first step. This data size grows as the number of truly relevant variables grows.
3) Is Unstable when the real model is not polynomial. Currently looking for ways to improve this.

X. Non-identifiability

This year we spent exploring how to find multiple optima in the problem of non-negative matrix factorization (NMF). Formally, we start with a non-negative matrix of data X, and our goal is to factor it as $X = WA$, where W and A are both non-negative matrices of low rank.

This problem is relevant to the program goals because it discusses what to do in the case that a problem is non-identifiable, but we still wish to interpret the parameters that we find. In the case of non-negative matrix factorization, the discovered dictionaries – the rows of A – are commonly interpreted as the main directions of variation in the data set. It is one of the simplest models that has interesting forms of non-identifiability: unlike principal components analysis or factor analysis, where different solutions are connected by scalings or rotations, in NMF different solutions can be disconnected and live in different subspaces. It is also commonly used when we expect the features to add to the data – for example, we expect each disease a patient has to show up as billing codes in their record; codes do not generally cancel out or subtract from each other. However, if one is to engage in such interpretation, it is important that we can find multiple solutions, if they exist.

Over the course of the year, we explored the following questions:

1) In exact NMF, is it possible to have multiple optima? Under what circumstances? We found there are two ways in which multiple optima can exist: inner rotations, which keep the factorization in the same subspace, and outer rotations, which change the subspace. The space of outer rotations is relatively easy to search, but the space of inner rotations can be very disconnected (imagine rotating a pyramid inside an octant, it might hit the sides but then fit after some more rotation). This work is currently under review at JMLR.

2) Are there efficient ways to find these solutions? We explored using a combination of rapidly-exploring random trees and random restarts to help cover the space of possible solutions. The random restarts were meant to find solutions that were disconnected, and the rapidly-exploring random trees to find alternate solutions locally. We found that in practice, the random restarts dominated: some amount of local exploration (rapidly-exploring random trees, Markov chain Monte Carlo approaches) helped tune the solution, but most of the variation in solutions was coming from the random restarts. This observation was somewhat disappointing, and we are still determining whether we can do better exploration.

3) If we have multiple solutions for one NMF problem, can it help us find solutions to another problem? Interestingly, our early work on this question seems to be yes. While it may be hard to find multiple optima for a single NMF problem (requires many random restarts), once we have the set of rotations that give us alternate solutions for one problem, those end up being good candidates for other NMF problems. While somewhat surprising, the intuition seems to be that there a limited set of ways in which one can rotate a pyramid to fit in an orthant.

Thus, if you collect a set of ways for a set of (even synthetic) problems, that can give you a good starting point for solving a new problem. We expect to submit this work to AAAI in the fall.

XI. Limits of Learning in the Cognitive Setting

Our goal is to investigate neural circuits in the exacting setting that knowledge acquisition can occur from single interactions, the results of these acquisitions are rapidly evaluatable subcircuits, and recall in response to an external input can be in the form of a rapid evaluation of a composition of subcircuits that have been acquired at arbitrary different earlier times. The most basic computational task here is the cognitive function of association, which is defined here to mean that a subcircuit is to be set up so that the excitation of a set of neurons that represent A will in future cause the excitation of a set of neurons that represent B. We note that this is a distinct and more onerous notion than the traditionally considered "auto-associative" networks where a set of memorized strings have to be retrieved. In particular the composability constraint adds a dimension that has not been considered before. The analysis is in terms of the three parameters: n the number of neurons, d the number of other neurons each neuron is connected to, and k, the inverse of the maximum synaptic strength when neurons have a firing threshold of unity.

One specific goal is to determine the capacity C of the system, namely the number of associations that can be memorized in succession so that they all, even the earliest ones memorized, remain functional after all the memorizations. In particular one wants to understand how C depends on n, d,and k. Our preliminary results study a simple mechanism, which we call the Basic Mechanism, that can work, in principle even with weak synapses. We show that the composability constraint is severe and limits the capacity to $O((d/k)^2)$, even independent of n, while an information theoretic estimate would be a higher $O(nd)$, linear in the number of synapses. For example for the realistic case of $d = n^{1/2}$, the capacity is linear in n, the number of neurons, and not the number of synapses. We have both upper and lower bounds: With the composability constraint capacity about $O((d/k)^2)$ is achievable, and that is close to the limit. In contrast, in the absence of the composability constraint, the higher information theoretic bound of $O(nd)$ (equal to $O(n^{3/2})$ in the case in question) can be approached.

The work is included in the paper [12]. This paper may be the first in the area of computational neuroscience ever accepted for publication by this conference series, and may influence others to bring the techniques of theoretical computer science to bear on this area. The work performed is summarized by the following draft abstract from the paper, which is currently being finalized.

We investigate neural circuits in the exacting setting that (i) the acquisition of a piece of knowledge can occur from a single interaction, (ii) the result of each such interaction is a rapidly evaluatable subcircuit, (iii) hundreds of thousands of such subcircuits can be acquired in sequence without substantially degrading the earlier ones, and (iv) recall can be in the form of a rapid evaluation of a composition of subcircuits that have been so acquired at arbitrary different earlier times.

We develop a complexity theory, in terms of asymptotically matching upper and lower bounds, on the capacity of a neural network for executing, in this setting, the following action, which we call *association*: Each action sets up a subcircuit so that the excitation of a chosen set of neurons A will in future cause the excitation of another chosen set B.

A succession of experiences, possibly over a lifetime, results in the realization of a complex set of subcircuits, each corresponding to a different association learned from a different experience. The composability requirement constrains the model to ensure that, for each association as realized by a subcircuit, the excitation in the triggering set of neurons A is quantitatively similar to that in the triggered set B, and also that the unintended excitation in the rest of the system is negligible. These requirements ensure that chains of associations can be triggered

We first analyze what we call the Basic Mechanism, which assumes direct connections between neurons in the triggering set A and the target set B. We consider networks of n neurons with expected number d of connections to and from each. We show that in the composable context capacity growth is limited by d^2, a severe limitation if the network is sparse, as it is in cortex. We go on to study the Expansive Mechanism, that additionally uses relay neurons which have high synaptic weights. For this mechanism we show that the capacity can grow as dn, to within logarithmic factors. From these two results it follows that in the composable regime, for the realistic cortical estimate of $d = n^{\frac{1}{2}}$, superlinear capacity of order $n^{\frac{3}{2}}$ in terms of the neuron numbers can be realized by the Expansive Mechanism, instead of the linear order n to which the Basic Mechanism is limited. More generally, for both mechanisms, we establish matching upper and lower bounds on capacity in terms of the parameters n, d, and the inverse maximum synaptic strength k.

The results as stated above assume that in a set of associations, a target B can be triggered by at most one set A. We go on to show that the capacities are similar if the number m of As that can trigger a B is greater than one but small, but become severely constrained if m exceeds a certain threshold.

References

[1] J. Ding, V. Tarokh, and Y. Yang, "Bridging AIC and BIC: a new criterion for autoregression," *IEEE Trans. Inf. Theory*, 2017.

[2] ——, "Optimal variable selection in regression models," *submitted*, 2016.

[3] J. Ding, E. Diao, J. Zhou, and V. Tarokh, "Approaching the predictive limit of learning," *in preparation*, 2017.

[4] J. Ding, J. Zhou, and V. Tarokh, "Asymptotically optimal prediction for time-varying data generating processes," *in preparation*, 2017.

[5] R. F. Engle and C. W. Granger, "Co-integration and error correction: representation, estimation, and testing," *Econometrica*, pp. 251–276, 1987.

[6] C. W. Granger and H. S. Lee, "An introduction to time-varying parameter cointegration," in *Economic Structural Change*. Springer, 1991, pp. 139–157.

[7] T. Bollerslev, "Generalized autoregressive conditional heteroskedasticity," *J. Econom.*, vol. 31, no. 3, pp. 307–327, 1986.

[8] R. F. Engle, "Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation," *Econometrica*, pp. 987–1007, 1982.

[9] O. E. Barndorff-Nielsen and N. Shephard, "Econometric analysis of realized volatility and its use in estimating stochastic volatility models," *J. Roy. Statist. Soc. Ser. B*, vol. 64, no. 2, pp. 253–280, 2002.

[10] A. P. Dawid, "Present position and potential developments: Some personal views: Statistical theory: The prequential approach," *J. Roy. Statist. Soc. Ser. A*, pp. 278–292, 1984.

[11] A. J. Patton, "Volatility forecast comparison using imperfect volatility proxies," *J. Econom.*, vol. 160, no. 1, pp. 246–256, 2011.

[12] L. G. Valiant, "Capacity of neural networks for lifelong learning of composable tasks," *to appear in Proc. 58th Annual IEEE Symposium on Foundations of Computer Science, Berkeley*, 2017.