# Efficient Online Sparse Kernel Learning

Hrusikesha Pradhan (16104272), Ritesh Kumar (160575)

Electrical Engineering, IIT Kanpur

## 1 Introduction

Reproducing kernel Hilbert spaces provide a nice framework to learn the nonparametric function representation. It is well proven that in case of empirical risk minimizer function, the function can be represented as a linear combination of kernels evaluated at the training point by the help of Representer Theorem. Thus using Representer theorem allows us to transform the search from an infinite dimensional space to a search over parameters and in case of online settings the number of parameters grows with each incoming sample. This property allows to learn a complicated function with these help of kernel representation but on the other hand this expressive power also needs large amount of memory for large data sets and infinite memory for streaming applications. This causes a huge training cost and there have been a lot of work in the literature to control this memory growth. We plan to propose an efficient sparsification technique in tandem with POLK framework and better than the existing KOMP technique used in POLK.

## 2 Problem Formulation

In the case of supervised kernel learning, we consider a Hilbert space denoted as $\mathcal{H}$, where the elements are functions, $f : \mathcal{X} \to \mathcal{Y}$. The functions can be represented in terms of elements of $\mathcal{X}$, owing to these two properties:

$$(i)\ \langle f, \kappa(\mathbf{x}, \cdot)) \rangle_{\mathcal{H}} = f(\mathbf{x}) \quad \text{for all } \mathbf{x} \in \mathcal{X}\ , \qquad (ii)\ \mathcal{H} = \overline{\text{span}\{\kappa(\mathbf{x}, \cdot)\}} \quad \text{for all } \mathbf{x} \in \mathcal{X}\ . \quad (1)$$

where $\kappa$ is a kernel function, $\kappa : \mathcal{X} \times \mathcal{X} \to \mathcal{R}$ and $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ denotes the Hilbert inner product for $\mathcal{H}$. We further assume that the kernel is positive semidefinite, i.e. $\kappa(\mathbf{x}, \mathbf{x}') \geq 0$ for all $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$. Function spaces with this structure are called reproducing kernel Hilbert spaces (RKHS).

From, property (1) (ii) we can write any function $f \in \mathcal{H}$ as a linear combination of kernel evaluations. For kernelized and regularized empirical risk minimization, the Representer Theorem establishes that the optimal $f$ in the hypothesis function class $\mathcal{H}$ may be written as an expansion of kernel evaluations *only* at elements of the training set as

$$f(\mathbf{x}) = \sum_{n=1}^{N} w_n \kappa(\mathbf{x}_n, \mathbf{x})\ . \quad (2)$$

where $\mathbf{w} = [w_1, \cdots, w_N]^T \in \mathbb{R}^N$ denotes a set of weights. The upper summand index $N$ in (2) is henceforth referred to as the model order. Now, with the basics of RKHS stated above,

we move to frame the main objective function when the data samples arrive sequentially. The observed incoming data sample $(\mathbf{x}_n, \mathbf{y}_n)$ are independent realization from a stationary joint distribution of the random pair $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$. Thus, we write the objective function as,

$$f^* = \underset{f\in\mathcal{H}}{\operatorname{argmin}}\, R(f) := \underset{f\in\mathcal{H}}{\operatorname{argmin}}\, \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ell(f(\mathbf{x}), y)] + \frac{\lambda}{2}\|f\|_{\mathcal{H}}^2 \tag{3}$$
$$= \underset{\mathbf{w}\in\mathbb{R}^{\mathcal{I}}}{\operatorname{argmin}}\, \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ell(\sum_{n\in\mathcal{I}} w_n\kappa(\mathbf{x}_n, \mathbf{x}), y)] + \frac{\lambda}{2}\|\sum_{n,m\in\mathcal{I}} w_n w_m \kappa(\mathbf{x}_m, \mathbf{x}_n)\|_{\mathcal{H}}^2 \quad .$$

where in the last inequality we have expanded the function using representer theorem given in (2).

We used the generalization of stochastic gradient descent to the functional settings to solve the above unconstrained optimization problem in (3). For a given realization $(\mathbf{x}_t, y_t)$, we compute the functional stochastic gradient of $\mathbb{E}_{\mathbf{x},\mathbf{y}}[\ell(f(\mathbf{x}), y)]$ as,

$$\nabla_f \ell(f(\mathbf{x}_t), y_t)(\cdot) = \frac{\partial \ell(f(\mathbf{x}_t), y_t)}{\partial f(\mathbf{x}_t)} \frac{\partial f(\mathbf{x}_t)}{\partial f}(\cdot) \tag{4}$$

We denote the first term as $\ell'(f(\mathbf{x}_t), y_t) := \partial \ell(f(\mathbf{x}_t), y_t)/\partial f(\mathbf{x}_t)$. To evaluate the second term on the right-hand side of (4), differentiate both sides of the expression defining the reproducing property of the kernel [cf. (1)(i)] with respect to $f$ to obtain

$$\frac{\partial f(\mathbf{x}_t)}{\partial f} = \frac{\partial \langle f, \kappa(\mathbf{x}_t, \cdot)\rangle_{\mathcal{H}}}{\partial f} = \kappa(\mathbf{x}_t, \cdot) \tag{5}$$

The same analogy can be applied to the functional derivative of the second term in (3). Thus using those above computations we now compute the functional stochastic gradient step for the problem in (3) as

$$f_{t+1} = (1-\eta_t\lambda) f_t - \eta_t \nabla_f \ell(f_t(\mathbf{x}_t), y_t) = (1-\eta_t\lambda) f_t - \eta_t \ell'(f_t(\mathbf{x}_t), y_t)\kappa(\mathbf{x}_t, \cdot) \;, \tag{6}$$

where $\eta_t > 0$ is an algorithm step-size either chosen as diminishing with $\mathcal{O}(1/t)$ or a small constant. We further require that, given $\lambda > 0$, the step-size satisfies $\eta_t < 1/\lambda$ and the sequence is initialized as $f_0 = 0 \in \mathcal{H}$. Given this initialization, we make use of the Representer Theorem (2), at time $t$, the function $f_t$ may be expressed as an expansion in terms of feature vectors $\mathbf{x}_t$ observed thus far as

$$f_t(\mathbf{x}) = \sum_{n=1}^{t-1} w_n \kappa(\mathbf{x}_n, \mathbf{x}) = \mathbf{w}_t^T \boldsymbol{\kappa}_{\mathbf{X}_t}(\mathbf{x}) \;. \tag{7}$$

We use the notation $\mathbf{X}_t = [\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}] \in \mathbb{R}^{p\times(t-1)}$ and $\boldsymbol{\kappa}_{\mathbf{X}_t}(\cdot) = [\kappa(\mathbf{x}_1, \cdot),\ \ldots\ , \kappa(\mathbf{x}_{t-1}, \cdot)]^T$ on the right-hand side of (7). Moreover, observe that the kernel expansion in (7), taken together with the functional update (6), yields the fact that performing the stochastic gradient method in $\mathcal{H}$ amounts to the following parametric updates on the kernel dictionary $\mathbf{X}$ and coefficient vector $\mathbf{w}$:

$$\mathbf{X}_{t+1} = [\mathbf{X}_t,\ \ \mathbf{x}_t], \quad \mathbf{w}_{t+1} = [(1-\eta_t\lambda)\mathbf{w}_t,\ \ -\eta_t \ell'(f_t(\mathbf{x}_t), y_t)] \;, \tag{8}$$

It can be observed that this update causes $\mathbf{X}_{t+1}$ to have one more column than $\mathbf{X}_t$. We define the *model order* as number of data points $M_t$ in the dictionary at time $t$ (the number

of columns of $\mathbf{X}_t$). FSGD is such that $M_t = t - 1$, and hence grows unbounded with iteration index $t$. Thus in this term project, we plan to propose an efficient compression technique to curb the model order growth problem such that we have an efficient function representation using a compressed kernel dictionary. In the next section, we mention some compression techniques and the challenges associated with them when implemented in our non-parametric functional settings.

# 3 Compression Techniques and the challenges

We have a very efficient compression technique in POLK which uses the orthogonal matching pursuit technique known as KOMP (Kernel orthogonal matching pursuit). We plan to make the compression algorithm more efficient such that it takes less time and give us a dictionary which gives a better sparse representation of the function $f$. Among the plethora of sparsification techniques available in the literature, we present a few techniques below:

- Subset Selection method: KOMP removes one dictionary element at a particular instant but we can remove a certain number of dictionary elements, i.e., a small subset of dictionary elements at a time and check the compression criterion. This subset can be removed randomly or any specific criterion can be used. And the study can be done on efficient selection of these subsets such that the function representation is not poor.

- Incremental and Decremental method: This method is quite similar to KOMP. This method consists of two steps: 1)adding only kernels that significantly reduce the approximation error (incremental step) and 2) removing kernels that no longer create significant error (decremental step). The threshold values is different for both the steps and this is how this algorithm is different from KOMP.

- Stopping criterion: Currently POLK uses the stopping criterion $\|f - \tilde{f}\|_{\mathcal{H}} \leq \epsilon_t$, where f is the sparse version of $\tilde{f}$ and $\epsilon_t$ is the approximation budget. In comparison to this different other criterion can be used for removal of dictionary elements.

- Sparse techniques from Gaussian process methods: Generally, in the Gaussian process literature the model order or the size of the sparse dictionary size is beforehand assumed to be some $M$ and then the problem is solved for the optimum dictionary elements either by solving some optimization problem where we use gradient search methods for the dictionary elements or some compression metric is used and then accordingly the search is carried out for optimum $M$ dictionary elements. But fixing $M$ beforehand is not straight forward in the context of learning a unknown function.

- Plats novelty criterion method: The distance of the new point, $\mathbf{x}_n$, from the current dictionary, $D_{n-1}$, is evaluated and if this distance is smaller than a given threshold $\delta_1$ (i.e., the new input vector is close to a point, which is already in the existing dictionary), then the newly arrived point is not added to $D_{n-1}$. Thus $D_n = D_{n-1}$. Otherwise, we compute the prediction error $e_n = y_n d_n$. If $|e_n|$ is smaller than a predefined threshold, $\delta_2$, then the new point is discarded and we set $D_n = D_{n-1}$. Only if $\mod e_n \geq \delta_2$, then $\mathbf{x}_n$ is inserted into $D_{n-1}$, forming the new dictionary $D_n = D_{n-1} \cup \{x_n\}$.

- Quantization of training data: If the distance of the point $x_n$ from the current dictionary $D_n$ is greater than or equal to the quantization size $\delta$ (i.e., $\mathbf{x}_n$ cannot be quantized to a point already contained in $D_n$) then $\mathbf{x}_n$ is classified as a new point and it is inserted into the dictionary $D_n = D_{n-1} \cup \{x_n\}$. Otherwise, $\mathbf{x}_n$ is classified as a

redundant point and the algorithm uses this information to update the coefficient of the closest center, i.e., the point $\mathbf{u}_n \in D_n$ closest to $\mathbf{x}_n$.

- Coherence based sparsification: The point $\mathbf{x}_n$ is inserted into the dictionary, if its coherence is below a given threshold $\epsilon_0$, i.e. $\max_{u_i \in D_n}\{|\kappa(\mathbf{x}_n), \mathbf{u}_i)|\} \leq \epsilon_0$, where $\epsilon_0$ is a parameter in [0,1] determining both the level of sparsity and the coherence of the dictionary.

- Sparsification under Classification settings: There are lot of works on logistic regression, SVM based on perceptron technique like Stoptron, Budget perceptron, Random perceptron, Tighter perceptron and Forgetron to mention a few. Currently its not very clear how to extend it for POLK and this needs a bit of study.

We can try the above methods and we can also combine two methods for better and efficient sparsification. These methods allows us to restrict the model order, i.e., the number of elements in the dictionary to a finite number but this also has to be proven theoretically. In some of the above mentioned methods, the selection criteria for including a new point into dictionary or the removing criteria from the dictionary is different from the criterion $\|f - \tilde{f}\|_{\mathcal{H}} \leq \epsilon_t$ and there by the proof of the POLK will not be valid in that case for that criterion. Thus for those methods the theoretical convergence also has to be proved with finite model order growth. But for those methods who have same criterion as POLK, the theoretical convergence analysis will stay valid and thus the performance of the new sparsification method in tandem with POLK will be shown via simulation results in comparison to the original sparsification of method of POLK, i.e., KOMP.

# 4 Plan of Action

Below we mention the detail plan for the efficient completion of the term project:

- By October 22nd: We plan to do more rigorous and exhaustive literature survey and find out more compression techniques and then finally deciding upon the compression techniques which are efficient than KOMP and as well which fit into POLK framework.

- By November 15th: If theoretical analysis is required then that will be done else if POLK theoretical analysis is valid then we will do extensive simulations to verify the compression algorithms efficiency in comparison to KOMP.

- By November 25th: The complete literature survey along with the theoretical results and simulation results will be drafted in to a report in a very detailed manner.