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 κ 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 ϵ_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 δ_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, δ_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 δ (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 ϵ_0, i.e. $\max_{u_i \in D_n}\{|\kappa(\mathbf{x}_n), \mathbf{u}_i)|\} \leq \epsilon_0$, where ϵ_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.

5 Progress Report (Mid-term report)

There are few key things which we have deduced in the course of exploring POLK [1] and other online sparsification techniques available in the literature and they can be categorized as:

5.1 Attributes

For developing a better compression technique than KOMP in POLK framework, a few things have to be closely monitored:

1. What is the stopping criterion? Is it different from existing criteria, i.e., $\|f - \tilde{f}\|_{\mathcal{H}} \leq \epsilon_t$ and how changing this stopping criterion would affect the POLK proof. Like some other stopping criterion would surely change result of [1, Proposition 8].

2. If we use some different technique in place of projection step in KOMP, i.e., avoid "$j^* = \text{argmin}_{j \in \mathcal{I}} \gamma_j$" and "$\gamma_{j^*} > \epsilon_t$", then the model order proof will completely change

3. POLK does not fix the model order in advance, any existing dictionary element can get removed and it does not differentiate between older dictionary elements and newer ones. The weights of the dictionary also get revised providing the best approximate to the function.

5.2 Improving the compression algorithm staying in the framework of POLK

A few things can be tried out which allows the POLK proof to stay valid like:

1. Removal of more than one element at a time from the dictionary

2. Removal of all the elements at a single go which upon removal have error less than the error threshold, i.e., we remove all dictionary elements having indices $j \in \mathcal{I}$ that satisfy $\gamma_j \leq \epsilon_t$

3. Remove the dictionary element, whose removal causes the maximum error but still below the error threshold.

4. Calculating similarity measure, i.e., coherence [**?**] between new data point and existing dictionary elements.

5. Can the idea of Forgetron [2] be extended to POLK.

6. Can ϵ_t be adaptive like it changes with the algorithm behavior.

The first three ideas for compressing the dictionary mentioned above, are addressed in Section 5.2.1 and Algorithm 1.

Algorithm 1 Kernel Orthogonal Matching Pursuit (KOMP) with Multiple Removals

Require: function $\tilde{f}$ defined by dict. $\tilde{\mathbf{D}} \in \mathbb{R}^{p\times\tilde{M}}$, coeffs. $\tilde{\mathbf{w}} \in \mathbb{R}^{\tilde{M}}$, approx. budget $\epsilon_t > 0$
initialize $f = \tilde{f}$, dictionary $\mathbf{D} = \tilde{\mathbf{D}}$ with indices $\mathcal{I}$, model order $M = \tilde{M}$, coeffs. $\mathbf{w} = \tilde{\mathbf{w}}$.
for $j = 1, \ldots, \tilde{M}$ **do**
Find approximation error corresponding to each dictionary element $\mathbf{d}_j$ removed

$$\gamma_j = \min_{\mathbf{w}_{\mathcal{I}\setminus\{j\}}\in\mathbb{R}^{M-1}} \|\tilde{f}(\cdot) - \sum_{k\in\mathcal{I}\setminus\{j\}} w_k\kappa(\mathbf{d}_k,\cdot)\|_{\mathcal{H}} \ .$$

end for
Find dictionary index whose error is below threshold, ϵ_t: $\mathcal{Q} = \{j : \gamma_j < \epsilon_t\}$
if $\mathcal{Q} = \emptyset$
stop
else

$$\textbf{Step1}: \quad \gamma_{\mathcal{Q}} = \min_{\mathbf{w}_{\mathcal{I}\setminus\mathcal{Q}}\in\mathbb{R}^{M-|\mathcal{Q}|}} \|\tilde{f}(\cdot) - \sum_{k\in\mathcal{I}\setminus\mathcal{Q}} w_k\kappa(\mathbf{d}_k,\cdot)\|_{\mathcal{H}} \ .$$

if $\gamma_{\mathcal{Q}} > \epsilon_t$
Randomly choose $k \in \mathcal{Q}$ and update $Q = Q \setminus k$ and go to **Step 1**
else
Prune dictionary $\mathbf{D} \leftarrow \mathbf{D}_{\mathcal{I}\setminus Q}$
Revise set $\mathcal{I} \leftarrow \mathcal{I} \setminus Q$ and model order $M \leftarrow M - |Q|$.
Compute updated weights $\mathbf{w}$ defined by current dictionary $\mathbf{D}$

$$\mathbf{w} = \underset{\mathbf{w}\in\mathbb{R}^M}{\operatorname{argmin}} \|\tilde{f}(\cdot) - \mathbf{w}^T\boldsymbol{\kappa}_{\mathbf{D}}(\cdot)\|_{\mathcal{H}}$$

end
end
return $f, \mathbf{D}, \mathbf{w}$ of model order $M \leq \tilde{M}$ such that $\|f - \tilde{f}\|_{\mathcal{H}} \leq \epsilon_t$

5.2.1 Removing all the dictionary elements for which $\gamma_j \leq \epsilon_t$

Instead of removing a single dictionary element, we can remove multiple dictionary elements and check whether the resultant function obtained the reduced dictionary satisfies the error threshold or not. Else we can take any removed element and add it back to the original dictionary and then repeat the process. The detailed algorithm is given in Algorithm 1. The important thing to note here is that, now with mutiple removals we do not have to run the "while" loop which was run multiple times in KOMP till we have empty dictionary or when we have error more than the error threshold. Thus we end up in calculating less number of projections as compared to KOMP algorithm, thereby improving the speed of KOMP. The proof for convergence of objective function error sequence is still valid as we have the same stopping criterion $\|f - \tilde{f}\|_{\mathcal{H}} \leq \epsilon_t$.

We now discuss the main idea linking Algorithm 1 to the proof of finite model order growth. If we select as $\epsilon = K\eta^{3/2}$, then

$$\operatorname{dist}(\kappa(\mathbf{x}_t,\cdot), \mathcal{H}_{\mathbf{D}_t}) \leq \frac{K\sqrt{\eta}}{|\ell'(f_t(\mathbf{x}_t), \mathbf{y}_t)|} \ , \tag{9}$$

which implies $\gamma_{M_{t+1}} \leq \epsilon$ holds and since $\gamma_{M_{t+1}} \geq \min_j \gamma_j$, we may conclude that $\min_j \gamma_j < \epsilon_t$ is satisfied. Consequently the model order at the subsequent step does not grow $M_{t+1} \leq M_t$ whenever (9) is valid. In comparison to the KOMP algorithm in POLK, here also the same equivalence relation holds but with multiple removals. Now taking negation of the above

Algorithm 2 Sparsity using Dissimilarity Measure

Require: function $\tilde{f}$ defined by dict. $\tilde{\mathbf{D}} \in \mathbb{R}^{p\times\tilde{M}}$, coeffs. $\tilde{\mathbf{w}} \in \mathbb{R}^{\tilde{M}}$, $\mathbf{x}_t$, approx. budget $\epsilon_t > 0$

initialize $f = \tilde{f}$, dictionary $\mathbf{D} = \tilde{\mathbf{D}}$ with indices $\mathcal{I}$, model order $M = \tilde{M}$, coeffs. $\mathbf{w} = \tilde{\mathbf{w}}$.

while candidate dictionary is non-empty $\mathcal{I} \neq \emptyset$ **do**

for $j = 1, \ldots, \tilde{M}$ **do**

Find the dissimilarity measure with each dictionary element $\mathbf{d}_j$

$$\gamma_j = 1/\kappa(\mathbf{d}_j, \mathbf{x}_t) \; .$$

end for

Find dictionary index corresponding to minimum dissimilarity measure: $j^* = \text{argmin}_{j\in\mathcal{I}}\, \gamma_j$

if minimal approximation error exceeds threshold $\gamma_{j^*} > \epsilon_t$

Add $\mathbf{x}_t$ to the dictionary $\mathbf{D}$, i.e., $\mathbf{D} = [\mathbf{D}, \quad \mathbf{x}_t]$

Compute updated weights $\mathbf{w}$ defined by current dictionary $\mathbf{D}$

$$\mathbf{w} = \underset{\mathbf{w}\in\mathbb{R}^M}{\text{argmin}} \|\tilde{f}(\cdot) - \mathbf{w}^T \boldsymbol{\kappa}_{\mathbf{D}}(\cdot)\|_{\mathcal{H}}$$

else

Keep the dictionary $\mathbf{D}$ same, i.e., $\mathbf{D} \leftarrow \mathbf{D}$

end

end while

return $f, \mathbf{D}, \mathbf{w}$ of model order $M \leq \tilde{M}$ such that $\|f - \tilde{f}\|_{\mathcal{H}} \leq \epsilon_t$

relation, we have when $M_{t+1} = M_t + 1$, i.e., the growth in model order implies that the condition

$$\text{dist}(\kappa(\mathbf{x}_t, \cdot), \mathcal{H}_{\mathbf{D}_t}) > \frac{K\sqrt{\eta}}{|\ell'(f_t(\mathbf{x}_t), \mathbf{y}_t)|} \tag{10}$$

holds. Thus, resulting in addition of a new data point ot the dictionary and there by increasing the size of dictionary.

5.2.2 Compression through Dissimilarity Measure

The main difference between KOMP and using dissimilarity measure compression is KOMP *removes* dictionary element based on certain criteria and dissimilarity measure compression *adds* element to the dictionary depending upon the dissimilarity measure. We have summarized the compression process in Algorithm 2. The convergence analysis of objective function error sequence stays the same as we have the same criterion as KOMP $\|f - \tilde{f}\|_{\mathcal{H}} \leq \epsilon_t$ but the proof for finite model order growth will change completely as we don't carry out projections now, so we wont have the relation (9) which is required for the proof. But we can provide proof for the finite mode order growth to be finite in this case from [**?**]. Algotihm 2 does not remove old dictionary points it can only add new data points to the dictionary.

5.3 Few more compression techniques

Forgetron: A perceptron with *shrinking* and *removal* steps. Hypothesis definition:

$$f_t = \sum_{i \in I_t} \sigma_{i,t} y_i K(\mathbf{x}_i, \cdot) \tag{11}$$

Steps:

- The Forgetron initializes I_1 to be the empty set, which implicitly sets f_1 to be the zero function.
- On round t, if a prediction mistake occurs, a tree step update is performed:
  1. Standard Perceptron update: index t is inserted into the active set and the weight $\sigma_{t,t}$ is set to be 1.
  2. Shrinking step: Let I'_t denote the active set which results from this update and $f'_t(\mathbf{x}) = f_t(\mathbf{x}) + y_t K(\mathbf{x}_t, \mathbf{x})$. We scale f' by a coefficient $\phi_t \in (0, 1]$ and define $f''_t = \phi_t f'_t$ and setting $\sigma_{i,t+1} = \phi_t \sigma_{i,t}$ for all $i \in I'_t$ we get
  $$f''_t(\mathbf{x}) = \sum_{i \in I'_t} \sigma_{i,t+1} y_i K(\mathbf{x}_i, \mathbf{x}) \tag{12}$$
  3. If the budget constraint is violated and $|I'_t| > B$ then I_{t+1} is set to be $I'_t \setminus \{r_t\}$ where $r_t = \min I'_t$. Otherwise, I_{t+1} simply equals I'_t.

BOGD: Bounded online gradient descent for scalable kernel-based online learning. The function calculations are similar to POLK paper but the compression technique is different. Steps:

- If $\ell'(y_t f_t(\mathbf{x})) \neq 0$, ie need to include this data point, then
  - if $|I_t| < B$ then
  $$f_{t+1}(\cdot) = (1 - \eta) f_t(\cdot) - \eta \ell'(y_t f_t(\mathbf{x}_t)) y_t \kappa(\mathbf{x}_x, \cdot)$$
  $$\text{and } I_{t+1} = I_t \cup \{t\}$$
  - else, compute the sampling distribution $\mathbf{p}_t = (p_1^t, ..., p_B^t)$. Sample an index i_k from $\{1, ..., B\}$ according to distribution $\mathbf{p}_t$. Define $I_{t+1} = I_k \cup \{t\} \setminus \{i_k\}$

BPA: Budgeted Passive Aggresive algorithm for classification

- If loss is non-zero and $|I_t| < B$ then update weights $\mathbf{w}_t$ and $I_{t+1} = I_t \cup \{t\}$.
- Else, find updated weight including t data point. Find the point to remove r. New weights are calculated by removing the point r and $|I_{t+1} = I_t \cup \{t\} \setminus \{r\}$

BPA-Simple: Calculate the weight of the new point and remove point r. Weights of rest of the points remain unchanged. **BPA-Projecting**: Original algorithm. More time and space complexity. **BPA-NN**: Fix k. For k nearest neighbours of the removed point r, update their weights. In the paper, $k = 1$.

5.4 Plan of Action

We will explore more algorithms available in the literature and see if we can build upon the exiting POLK framework. We will also try to explore other algorithms and see if we can prove convergence analysis for them.

References

[1] A. Koppel, G. Warnell, E. Stump, and A. Ribeiro, "Parsimonious online learning with kernels via sparse projections in function space," *arXiv preprint arXiv:1612.04111*, 2016.

[2] O. Dekel, S. Shalev-Shwartz, and Y. Singer, "The forgetron: A kernel-based perceptron on a fixed budget," in *Advances in neural information processing systems*, 2006, pp. 259–266.