Online Prototypes and Criticisms

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

In our work we develop an algorithm for finding an example-based summary of our dataset. We combine two already existing algorithms to obtain a new one, providing online updates of the subset of prototypes and the subset of criticisms. We propose two approaches to this problem and present results regarding both of them.

1 Introduction and related work

In order to make complex distributions understandable for humans, a part of machine learning research was devoted to creating methods of summarising datasets using sets of *prototypes*. The idea is to choose such subsample that minimizes some distance between the empirical distribution of the subset and the distribution of the whole dataset (the distances used for this puprose vary depending on further specifications of the method). New approaches of machine learning researchers [**?**] involve presenting a set of *criticisms* in addition to prototypes, to make the summary even more interpretable for humans. The results of human pilot study descried in [**?**] show that indeed criticisms increase interpretability of the method – in comparison to the method based on prototypes only, subjects were able to predict results of the method more accurately. However, there was no evidence that the critisms inrease efficiency of the task, since the subjects performed faster when only prototypes were shown to them.

Another extension of the explanation-based methods was proposed by [**?**]. Their method is designed to choose a subsample of size M of prototypes from an 'online' stream of observations of length N. It is assumed additionally that N is not known in advance, therefore, the algorithms stores a set of M prototypes at all times. As each new observation from the stream arrives, the subsample of prototypes is updated – the new observation may be swapped (with some probability) with one of the already existing prototypes. The 'online' method prevents us from storing all data for later consideration whether it should be classified as a prototype or not

Our goal was to combine the two approaches and create an algorithm that makes an 'online' summary of a given dataset, consisting both of prototypes and criticisms.

2 Theoretical background

2.1 The Greedy algorithm: combining prototypes with criticisms

The Maximum Mean Discrepancy (MMD) is a measure of distance between two distributions P and Q and is defined as follows:

$$MMD(\mathcal{F}, P, Q) = \sup_{f \in \mathcal{F}} \left(E_{X \sim P}[f(X)] - E_{X \sim Q}[f(Y)] \right), \tag{1}$$

where $\mathcal{F}$ is a function space. In the following discussion, we will focus on the case when $\mathcal{F}$ is a reproducing kernel Hilbert space (RKHS) with kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. It can be shown that

$$MMD^2(\mathcal{F}, P, Q) = 0 \quad \iff P \text{ is indistinguishable from } Q \text{ on } \mathcal{F}.$$

[**?**] propose a method for prototypes and criticisms selection based on the minimization of the MMD^2 statistic mentioned above. Let N be the number of available observations, M be the desired number of prototypes representing the dataset and $[N]$ be the set $\{1, 2, \ldots, N\}$. The algorithm aims at finding a subset $S \subset [N]$ of indices such that $|S| = M$ and S minimizes $MMD^2(\mathcal{F}, X, X_S)$, where X is the set of observations and X_S is a subsample corresponding to indices S. The empirical distribution of both X and X_S are used in the computation of the MMD statistic. This implies that the empirical means will substitute the expected values in equation (1). It can be shown that minimizing the MMD^2 is equivalent to maximizing the following function $J(S)$ over S such that $|S| = M$:

$$J(S) = \frac{2}{n|S|} \sum_{i \in [N], j \in S} k(x_i, x_j) - \frac{1}{|S|^2} \sum_{i,j \in S} k(x_i, x_j). \tag{2}$$

An approximate solution of the above optimization problem is given by Algorithm 4 .

Algorithm 1 The Greedy algorithm

```
1: Input: $M, S = \emptyset$
2: while $|S| < M$ do
3:     for $i \in [N] \setminus S$ do
4:         $f_i = F(S \cup i) - F(S)$
5:         $S = S \cup \{\text{argmax } f_i\}$
6:     end for
7: end while
8: return $S$.
```

Theoretical results [**?**] ensure that, under certain conditions on the kernel matrix, the subset obtained by Algorithm 4 achieves at least a fraction $(1 - \frac{1}{e})$ of the objective value achieved by the optimal subset. The conditions are satisfied by the radial basis function kernel which will be used in the following analysis.

In addition to selecting prototypes, the Greedy algorithm allows for the selection of observations that are not well explained by prototypes. These data points will be called prototypes. Let c denote the number of desired criticisms. In order to find data points that deviate the most from prototypes, the following cost function is considered in [**?**]:

$$L(C) = \sum_{l \in C} \left| \frac{1}{n} \sum_{i \in [n]} k(x_i, x_l) - \frac{1}{m} \sum_{i \in S^*} k(x_i, x_l), \right| \tag{3}$$

where S^* is the set of prototypes. A straightforward approach would be to maximize the above function over sets $C \subset [N] \setminus S^*$ such that $|C| = c$. However, the authors suggest that more diverse criticisms are obtained if a regularising function is added to $L(C)$. Namely, they propose to solve the following optimization problem:

$$\max_{C \subset [N] \setminus S^*, |C| = c} L(C) + r(C).$$

The regularising function used both in the paper [**?**] and in our work is given by:

$$r(K, C) = \log \det K_{C,C},$$

where $K_{C,C}$ is the part of the kernel matrix K corresponding to the set of indices C.

2.2 Streaming MMD Minimization

The following section summarizes the main results of the paper *Super-Sampling with a Reservoir* [**?**] that we have used as an inspiration for our Faster Algorithm [ADD REFERENCE].

The Super-Sampling with a Reservoir is an updated version of the simplest reservoir sampling for unweighted data [ADD REFERENCE] which allows for processing streaming data online (the simplest algorithm results in a random sample without replacement). This is possible thanks to the properties of reproducing kernel Hilbert spaces and kernel mean embeddings of distributions. A useful representation of a kernel k is as an inner product over an explicit "feature space" mapping $\phi : \mathcal{X} \to \mathcal{H}$, with

$$k(x, x') = \langle \phi(x), \phi(x') \rangle$$

However, the feature map $\phi(x)$ is infinite dimensional (as in squared exponential kernel). So, to make it possible to construct an online algorithm, we explicitly instantiate an approximate feature space representation $\hat{\phi}(x) \in \mathbb{R}^D$. This is achieved in [**?**] by the usage of a finite vector of D random Fourier projections, where each feature is of the form:

$$\hat{\phi}(x) = \sqrt{\frac{2}{D}} \begin{bmatrix} \cos(\omega_1^T x + b_1) \\ \vdots \\ \cos(\omega_D^T x + b_D) \end{bmatrix}$$

where each ω_d is drawn from the distribution $p(\omega)$; $p(\omega)$ arises by taking the Fourier transform of the kernel, and each b_d is uniform on $[0, 2\pi]$. Then these random Fourier features $\hat{\phi}(x) \in \mathbb{R}^D$ approximate the true (potentially infinite dimensional) feature map $\phi(x) \in \mathbb{H}$. Then, an approximate kernel defined by the inner product of the approximate feature maps, i.e. $k(x, x') \approx \hat{\phi}(x)^T \hat{\phi}(x)$, provides an unbiased estimate of the evaluations of the kernel function, with

$$\mathbb{E}_{\omega,b}[\hat{\phi}(x)^T, \hat{\phi}(x)] = k(x, x').$$

Now, we can use random Fourier features evaluated at finite sample points to approximate the empirical estimate of the mean embedding, i.e.

$$\hat{\mu}_N = \frac{1}{N} \sum_{i=1}^{N} \hat{\phi}(x_i). \tag{4}$$

In the same manner, we can define a second empirical estimate of the mean embedding and approximate it by:

$$\nu_M = \frac{1}{M} \sum_{j=1}^{M} \phi(y_j), \tag{5}$$

where $y_j \in Y_M \subset X$; $M << N$, i.e. Y_M is a small subset of the full points in the set X. Having defined $\hat{\mu}_N$ and $\hat{\nu}_M$, we can rewrite the MMD [ADD REFERENCE] as an RKHS norm $||\mu_N \nu_M||_{\mathcal{H}}$.

Streaming subset selection [TO BE ADDED Petya]

3 Our Algorithms

We will now describe several methods which compute criticisms online. The idea here is that as a new data point arrives we would like to determine whether to add it to the set of criticisms or not. We want to maintain a fixed number of criticisms so if we do decide to include the new point then we need to exclude one of the existing criticisms. The first M values in the stream are considered as the starting set of *prototypes* while the second set of M points are treated as the initial set of *criticism*. Each time a new data point x_n is considered, the algorithm selects M prototypes among the $M + 1$ values available optimizing the objective function (refer to equation of the greedy algorithm). This is done by starting from an empty set of prototypes and then looping over the available points. If the new data point x_n is accepted as a prototype, the algorithm starts the evaluation of x_{n+1}. Conversely, if x_n is rejected as a prototype is then tested as a potential criticism. Once done for the identification of prototypes, the algorithm selects M criticisms among the $M + 1$ values available. The new set of criticism is picked optimizing the function (ref to $L(C)$ mentioned in the greedy algorithm). The regularization function (refer to greedy part) is considered. This algorithm is described in

Algorithm 2 Online Search

```
 1: Input: Stream of samples: $\mathbf{x}_1, \dots, \mathbf{x}_n$
 2: Output: A subset of prototypes: $P$ and a subset of criticisms: $C$.
 3: Set the initial set of prototypes to be: $P_0 = \{\mathbf{x}_1, \dots, \mathbf{x}_M\}$
 4: Set the initial set of criticisms to be: $C_0 = \{\mathbf{x}_{M+1}, \dots, \mathbf{x}_{2M}\}$
 5: for $n = M+1, \dots, N$ do
 6:     $P_n$ = OnlinePrototypes($P_{n-1}, x_n$)
 7:     if $x_n \notin P_n$ then
 8:         $C_n$ = OnlineCriticisms($C_{n-1}, x_n$)
 9:     end if
10: end for
11: return $P_N, C_N$
```

3.1 Online Greedy

We now introduce a sequential algorithm for selecting prototypes and criticism in an online fashion. The algorithm, which we will refer to as "Online Greedy" (OG), considers again the MMD as a measure of discrepancy between the samples and any selected subset. However, differently from 4, OG processes streaming data online selecting a subsets of size M for both prototypes and criticism. Notice how, the main difference between the OG algorithm and the G algorithm, lies in the computation of the kernel matrix. Indeed, in the original greedy algorithm, the Gram matrix $K \in R^{NxN}$ where $K_{ij} = k(x_i, x_j)$ is computed for the overall set of N observations which is available when running each step of the algorithm. In the online version of the algorithm, only $n \leq N$ observations are observed at each step. This implies that the subset of the Gram matrix, say K_{subset}, used at the first step is of dimension $2Mx2M$. Indeed, the first $2M$ observations are initially split in a set of M prototypes and M criticisms. We thus have a subset of K_{subset} of dimension $Mx2M$ both for the prototypes, say $K_{prototypes}$,and the criticism, say $K_{criticism}$. As soon as a new observation becomes available, the dimension of c and $K_{criticism}$ increases to $(M+1)x(2M+1)$. The point is then tested to be a prototype or a criticism. Depending on the type of point the corresponding matrices $K_{prototypes}$ and $K_{criticism}$ will be modified so as to have on the rows the new set of prototypes and on the columns the observations collected so far. The number of columns for both $K_{prototypes}$ and $K_{criticism}$ will increase overtime incorporating the observations collected. The number of columns will stay constant to the M. At each step, a standard greedy algorithm will be performed using the relevant kernel matrices.

Algorithm 3 The OG Algorithm

```
 1: Input: Stream of samples: $\mathbf{x}_1, \dots, \mathbf{x}_n$
 2: Output: A subset of prototypes: $P$ and a subset of criticisms: $C$.
 3: Set the initial set of prototypes to be: $P_0 = \{\mathbf{x}_1, \dots, \mathbf{x}_M\}$
 4: Set the initial set of criticisms to be: $C_0 = \{\mathbf{x}_{M+1}, \dots, \mathbf{x}_{2M}\}$
 5: for $n = M+1, \dots, N$ do
 6:     $P_n$ = Greedy($P_{n-1}, x_n$)
 7:     if $x_n \notin P_n$ then
 8:         $C_n$ = Greedy($C_{n-1}, x_n$)
 9:     end if
10: end for
11: return $P_N, C_N$
```

3.2 Faster Algorithm

Ideally we would like to use the same loss function that was used in the greedy criticism algorithm. If we were able to compute it then we could run a greedy-type algorithm on just the existing criticisms and the new data point.

However, we cannot compute this easily. This is because it would require us to calculate $k(x_i, x_n)$ for all $i \in [n]$ which will prove to be very computationally intensive for large datasets. The other point is that we may not want to have to store all the data up to this point so it may not even be possible to make this computation.

In order to get around this we can make use of the random fourier feature approximation. This allows us to write

$$L_n(C) = \sum_{l \in C} \left| \frac{1}{n} \sum_{i \in [n]} k(x_i, x_l) - \frac{1}{m} \sum_{i \in P_n} k(x_i, x_l) \right| \tag{6}$$

$$\approx \sum_{l \in C} \left| \frac{1}{n} \sum_{i \in [n]} \langle \hat{\phi}(x_i), \hat{\phi}(x_l) \rangle - \frac{1}{m} \sum_{i \in P_n} \hat{\phi}(x_i), \hat{\phi}(x_l) \right| \tag{7}$$

$$= \sum_{l \in C} \left| \frac{1}{n} \left\langle \sum_{i \in [n]} \hat{\phi}(x_i), \hat{\phi}(x_l) \right\rangle - \frac{1}{m} \left\langle \sum_{i \in P_n} \hat{\phi}(x_i), \hat{\phi}(x_l) \right\rangle \right| := \tilde{L}(C) \tag{8}$$

Now we can try and maximize this function online in order to identify the criticisms.

Instead of computing $k(x_i, x_n)$ for all $i \in [n]$, at each step we store $\sum_{i \in [n]} \hat{\phi}(x_i)$. Updating this at each stage only takes $O(D)$ operations where D is the number of random fourier features. An additional bonus is that we don't have to store all of the previous data points in order to make this update.

Of course we may still need to use a regularization term however this only requires us to calculate the kernel for the criticisms and so is not as computationally intensive. Therefore, the function that we will look to maximize at each step is:

$$\tilde{F}(C) := \tilde{L}(C) + \log \det K_{C,C} \tag{9}$$

Here at the nth step when we are considering the data point x_n if we have rejected and we want to update the set of criticisms to the subset of $C_{n-1} \cup x_n$ of size M^* which maximizes $\tilde{F}$. Thus as with the streaming prototype procedure we need to decide which point to drop. This procedure is

Algorithm 4 Online Criticisms

```
1: Input: $C_{n-1}, x_n$
2: $C_n = \text{argmax} \left\{ \tilde{F}(C) : C \subset C_{n-1} \cup x_n : |C| = M^* \right\}$
3: return $C_n$
```

References