Online Speaker Diarization with Interactive Learning

Anirudh Garg
anirudhg@iitk.ac.in
170117

Anubhav Satpathy
asatpath@iitk.ac.in
170136

Nitish Vikas Deshpande
nitishvd@iitk.ac.in
17807450

Abstract

In this project, we formulate the problem of speaker diarization as an online learning problem with labels provided by a human agent. To simulate a real-life scenario, we consider a setup where the labels are revealed episodically by the human annotator. We have implemented five methods: a) Linear Upper Confidence Bound Algorithm (LinUCB) b) Background Episodically Rewarded Linear Upper Confidence Bound Algorithm (BerlinUCB) c) BerlinUCB with self supervision using K Nearest Neighbours (KNN) d) BerlinUCB with self supervision using Gaussian Mixture Models (GMM) e) BerlinUCB with self supervision using K Means clustering. We perform experiments on nine setups where each setup involves either 5, 10 or 15 speakers and the label revealing probability as 0.01, 0.1 or 0.5. We observe that LinUCB provides the highest accuracy compared to other methods. We conclude that self-supervision does not help in improving the performance of online learning. As a future direction, we propose to incorporate a prior knowledge of speakers by appropriately initializing the model.

1 Introduction

Speaker diarization is the task to partition an audio stream into homogeneous segments according to the speaker identity. A layman's way to put it would be "Who spoke when". It is observed that some state-of-the art speaker diarization systems require really large datasets to train the clustering modules which might not be easily available everywhere. Here, the method of learning continually can be employed i.e., online learning. Online learning is a problem where data becomes available in a sequential order and later used to update the best predictor for future data or reward associated with the data features. The only way sometimes, in which the online learning agent can learn from the past experience is the feedback in terms of rewards approach. This online learning problem is particularly important in the field of sequential decision making. In sequential decision making, the best possible action to perform at each step to maximize the cumulative reward over time is chosen by the agent. It is important to obtain an optimal balance between the exploration of new actions and the exploitation of the possible rewards generated from known previous actions.

However, in real-world online learning problems, the human annotator may not be available always to provide feedback to the AI agent. In other words, the reward feedback can come in different episodes or frequencies. An effective system should adapt to the nonstationarity in these observed rewards. Reference [2] proposed a novel problem setting called *Online Learning with Episodic Reward*. We consider the problem setting in [2] for our online speaker diarization problem.

2 Problem Formulation

We formulate the online speaker diarization as a contextual-bandit problem similar to the online semi-supervised learning method in [2]. In a bandit problem, each arm of the bandit corresponds to a certain distribution of probability for the rewards and in each round, a particular arm is chosen by the agent. Reward and updates are received by the agent based on this move. A more refined version of this is known as the contextual bandit [1], where the results are obtained given the context based on a relationship between the feature vectors and the context.

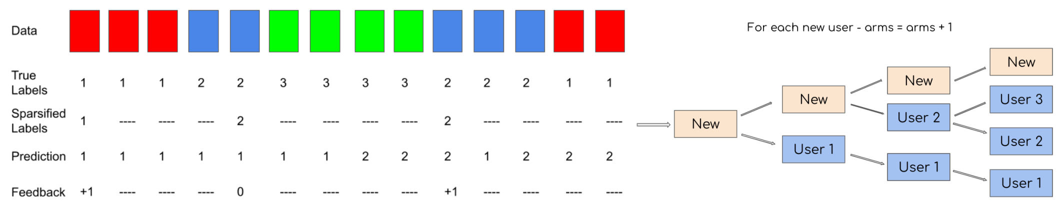


Figure 1: Left: Data stream with sparse labels (revealing probability = $\frac{3}{14}$); Right: The arm expansion process of the bandit

Following feedback strategy is used for updating the model. The reward at a timestep is +1 if the predicted arm is same as the label. In all other cases, reward is 0. Model receives a feedback only in the timesteps where label has been revealed. In practical settings, labels are revealed episodically.

2.1 A Multi-Armed Bandit Formulation

Formally, a contextual-bandit algorithm proceeds in discrete trials (t). In a trial t:

- A current user u_t and a set $\mathcal{A}_t$ of arms and actions is considered by the algorithm along with the corresponding feature vectors $x_{t,a}$, where $a \in \mathcal{A}_t$. Through the vector $x_{t,a}$, we obtain the information about both the user u_t and the chosen arm a. This vector will be referred to as the *context* in future.
- Analysing the payoffs in previous trials, an arm $a_t \in \mathcal{A}_t$ is chosen by A, which receives a payoff of r_{t,a_t}. The expectation of this payoff is dependant on both the user u_t and the arm a_t chosen by the user.
- The strategy to select the arms is improved by the new observation, ($x_{t,a}$, u_t, $r_{t,a}$). An important thing to note is that for the unchosen arms, $a \neq a_t$, we will receive no feedback i.e., no payoff $r_{t,a}$.

In the event of total N trials, the payoff is described by $\sum_{t=1}^{N} r_{t,a_t}$, thus the expected payoff will be given by $\mathbb{E}\left[\sum_{t=1}^{N} r_{t,a_t^*}\right]$, where a_t^* denotes the arm which has the maximum payoff at the trial 't'. Our goal is to maximize the expected payoff that is described above.

2.2 The K-armed Bandit

K-armed bandit is a specific case of the general contextual bandit problem in which the user u_t is the same for all the trials and also the corresponding set of arms $\mathcal{A}_t$ remains unchanged and contains K arms for all the trials. As the arm set and context are invaried in each trial, they make no difference whatsoever to the bandit algorithm, thus it can be called as the context free bandit algorithm.

2.3 The LinUCB Algorithm

If we are given a parametric form of payoff function, an efficient method to compute the confidence interval, from the data, of the parameters, with which we can compute a UCB (Upper Confidence Bound) of the estimated arm payoff, is the LinUCB algorithm. The LinUCB algorithm results in a closed form solution when the payoff model is linear. Assuming that the expected payoff of an arm a is linear in its d-dimensional feature $x_{t,a}$ and employing an unknown vector θ_a^* acting as the coefficient, we get,

$$\mathbb{E}[r_{t,a_t}|\mathbf{x}_{t,a}] = \mathbf{x}_{t,a}^T \theta_a^* \tag{1}$$

At the t^{th} trial, let $\mathbf{D}_a$ be a matrix of dimension m × d. In this matrix, the rows correspond to the m contexts which were observed previously for the arm a. $c_a \in \mathbb{R}^m$ is the corresponding response vector. On implementing ridge regression to the data ($\mathbf{D}_a$, c_a) we get an estimate of the coefficients:

$$\hat{\theta} = (\mathbf{D}_a^T \mathbf{D}_a + \mathbf{I}_d)^{-1} \mathbf{D}_a^T c_a \tag{2}$$

where $\mathbf{I}_d$ is the dxd identity matrix. It can be shown that the payoff for arm *a* can have a reasonable UCB such that:

$$\mid \mathbf{x}_{t,a}^T \hat{\theta}_a - \mathbb{E}\big[r_{t,a_t} | \mathbf{x}_{t,a}\big] \mid \leq \alpha \sqrt{\mathbf{x}_{t,a}^T (\mathbf{D}_a^T \mathbf{D}_a + \mathbf{I}_d)^{-1} \mathbf{x}_{t,a}} \tag{3}$$

where $\mathbf{x}_{t,a} \in \mathbb{R}^d$ and $\alpha = 1 + \sqrt{\ln{(2/\delta)}/2}$. Thus, a_t can be reasonably estimated as:

$$a_t = \text{argmax}_{a \in \mathcal{A}_t} (\mathbf{x}_{t,a}^T \theta_a + \alpha \sqrt{\mathbf{x}_{t,a}^T \mathbf{A}_a^{-1} \mathbf{x}_{t,a}}) \tag{4}$$

where $\mathbf{A}_a = \mathbf{D}_a^T \mathbf{D}_a + \mathbf{I}_d$. A comprehensive description of the LinUCB algorithm is given by Algorithm 1. It is useful to keep in mind that LinUCB always chooses the arm with the highest UCB. We see that the computational complexity of this algorithm is at most cubic in number of features and is linear in the number of arms. Computation complexity can be decreased further by updating $\mathbf{A}_{a_t}$ in every step (which takes $O(d^2)$ time). The algorithm can be employed for a dynamic arm set too and gives efficient results as long as we make sure that the size of $\mathcal{A}_t$ is not too large.

Algorithm 1 BerlinUCB

```
 1: Initialize $c_t \in \mathbb{R}_+, \mathbf{A}_a \leftarrow \mathbf{I}_d, \mathbf{b}_a \leftarrow \mathbf{0}_{dX1} \forall a \in \mathcal{A}_t$
 2: for $t = 1, 2, 3.....T$ do
 3:     Observe features $\mathbf{x}_t \in \mathbb{R}^d$
 4:     for all $a \in \mathcal{A}_t$ do
 5:         $\hat{\theta}_a \leftarrow \mathbf{A}_a^{-1} \mathbf{b}_a$
 6:         $p_{t,a} \leftarrow \hat{\theta}_a^T \mathbf{x}_t + c_t \sqrt{\mathbf{x}_t^T \mathbf{A}_a^{-1} \mathbf{x}_t}$
 7:     end for
 8:     Choose arm $a_t = \text{argmax}_{a \in \mathcal{A}_t} p_{t,a}$
 9:     if the background revealed the feedback then
10:         Observe feedback $r_{a_t,t}$
11:         $\mathbf{A}_{a_t} \leftarrow \mathbf{A}_{a_t} + \mathbf{x}_t \mathbf{x}_t^T$
12:         $\mathbf{b}_{a_t} \leftarrow \mathbf{b}_{a_t} + r_{a_t,t} \mathbf{x}_t$
13:     else
14:         if the background revealed NO feedbacks then
15:             if use self-supervision feedback then
16:                 $r' = [a_t == predict(\mathbf{x}_t)]$
17:                 $\mathbf{b}_{a_t} \leftarrow \mathbf{b}_{a_t} + r' \mathbf{x}_t$
18:             else
19:                 $\mathbf{A}_{a_t} \leftarrow \mathbf{A}_{a_t} + \mathbf{x}_t \mathbf{x}_t^T$
20:             end if
21:         end if
22:     end if
23: end for
```

The LinUCB algorithm corresponds to the first 12 lines of the BerlinUCB algorithm. In the LinUCB algorithm, we only observe the case in which the feedback is revealed.

2.4 The BerlinUCB Algorithm

The BerlinUCB algorithm is an extension of the LinUCB algorithm. It is extended in the case where the background reveals no feedbacks. We use different clustering algorithms to obtain the results in this case as shown in the pseudo-code. The clustering algorithms used are the K - nearest neighbours clustering, K-means clustering and the Gaussian Mixture Modeling. We briefly discuss them in the following sections.

2.4.1 The K-Nearest Neighbour Algorithm

The K-Nearest Neighbours [4] is a non-parametric algorithm where k nearest neighbours of a data point are observed. The data point is classified in the class whose data points are most abundant among the observed k-data points in its vicinity.

2.4.2 The K-Means Clustering Algorithm

The K-means clustering algorithm [5] is an iterative algorithm where the algorithm partitions the data into k different non-overlapping subgroups, each defined by a centroid. The iterations are initialized by randomly selecting k points acting as centroids and then assigning the data-points to their closest centroid based on their euclidean distance from it. Once the assignment is done, the centroid values are collected of the points belonging to the same centroid (cluster), thus updating the value of the centroid. Iteratively, assignment is done again and then the calculation of the centroids is done. If on re-assignment, no point is re-assigned to a different cluster the algorithm is stopped and the final assignments are the k-different non-overlapping subgroups.

2.4.3 The Gaussian Mixture Modeling Algorithm

The Gaussian Mixture Modelling Algorithm [6] is a probabilistic modeling algorithm where we generate the data points from a mixture of a finite number of Gaussian distributions with unknown parameters. The different Gaussian distributions are treated as the different clusters and data points generated from the same distribution belong to the same cluster.

3 Dataset and Implementation

We used VoxCeleb [3], which is a large scale speaker recognition dataset to generate 3 different kinds of data which correspond to 5, 10, 15 speakers to mimic the real world conversations . Three different kinds of reward streams were used with epiReward being 0.01, 0.1, 0.5. Thus a total of 9 learning environments were analysed. To evaluate the performance, we measured the accuracy, which is the ratio of the correct identification of speakers and the total number of time steps or more accurately the ratio of total reward and the total number of time steps. We used Mel-frequency spectral coefficients (MFCC) as the feature embedding. Using the spectrogram of the data, the MFCC creates a feature vector for the data that is further used in our computations.

4 Results

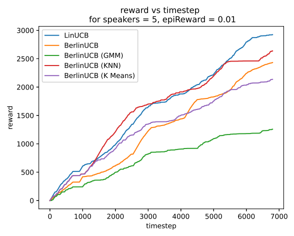
In figure 2, accuracy is calculated for different number of speakers (5, 10, 15) and different values of epiRewards (0.01, 0.1, 0.5). Sparsified reward streams with a revealing probability are known as the epiRewards. We can clearly observe from the table below that for higher epiReward, model achieves better accuracy.

Accuracy									
	e=0.01 n=5	e=0.01 n=10	e=0.01 n=15	e=0.1 n=5	e=0.1 n=10	e=0.1 n=15	e=0.5 n=5	e=0.5 n=10	e=0.5 n=15
LinUCB	0.43	0.459	0.355	0.494	0.528	0.454	0.524	0.571	0.477
BerlinUCB	0.357	0.534	0.417	0.402	0.469	0.416	0.453	0.500	0.445
BerlinUCB(KNN)	0.387	0.324	0.224	0.313	0.288	0.233	0.308	0.296	0.293
BerlinUCB(KMeans)	0.314	0.349	0.221	0.243	0.273	0.231	0.250	0.273	0.208
BerlinUCB(GMM)	0.184	0.159	0.178	0.258	0.385	0.199	0.382	0.405	0.227

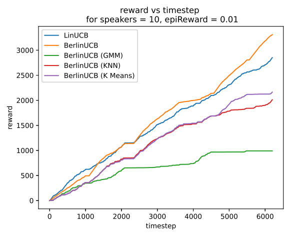
*** e = epiReward probability, n = total number of speakers in audio

5 Observations and Conclusion

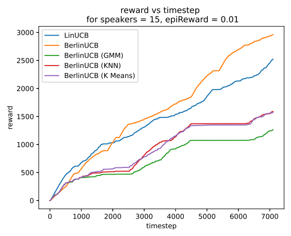
It was observed that, irrespective of epiReward, the case with 10 speakers gave the best results for the LinUCB as well as the BerlinUCB. Generally speaking, the case with 5 speakers gave better accuracy than the case with 15 speakers. For BerlinUCB with the Gaussian Mixture Modelling



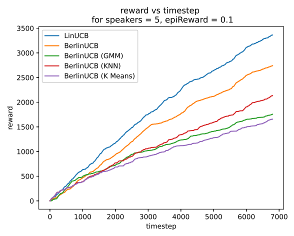
(a) 5 speakers, 0.01 epiReward



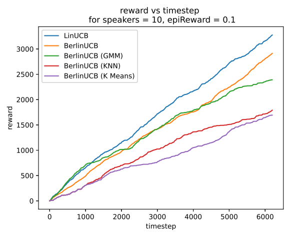
(b) 10 speakers, 0.01 epiReward



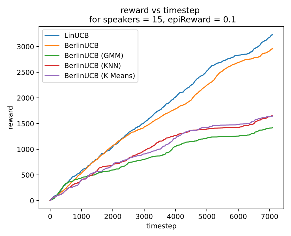
(c) 15 speakers, 0.01 epiReward



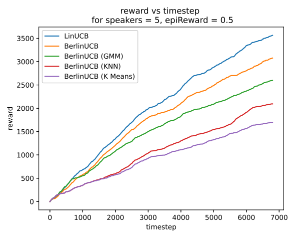
(d) 5 speakers, 0.1 epiReward



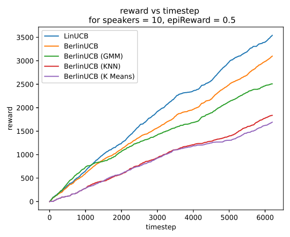
(e) 10 speakers, 0.1 epiReward



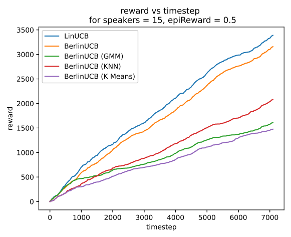
(f) 15 speakers, 0.1 epiReward



(g) 5 speakers, 0.5 epiReward



(h) 10 speakers, 0.5 epiReward



(i) 15 speakers, 0.5 epiReward

Figure 2: Plots for reward vs timestep for different number of speakers and epiRewards

algorithm, it was observed that the accuracy improved as we increased the epiReward. Subsequently, we can conclude that higher the epiReward, the greater the accuracy.

6 Future directions

Currently LinUCB performs better than all variants of BerlinUCB. So as of now we haven't yet utilized the advantage of incorporating self-supervision modules. In future we wish to explore data with oracle i.e., the prior knowledge of no. of speakers and appropriate model initialization. We expect that self-supervision modules will boost the accuracy performance above LinUCB after incorporating the oracle.

References

[1] Lihong Li et al. "A contextual-bandit approach to personalized news article recommendation". In: *Proceedings of the 19th international conference on World wide web*. 2010, pp. 661–670.

[2] Baihan Lin. "Online semi-supervised learning in contextual bandits with episodic reward". In: *Australasian Joint Conference on Artificial Intelligence*. Springer. 2020, pp. 407–419.

[3] Arsha Nagrani, Joon Son Chung, and Andrew Zisserman. "Voxceleb: a large-scale speaker identification dataset". In: *arXiv preprint arXiv:1706.08612* (2017).

[4] Simon Rogers and Mark Girolami. "A First Course in Machine Learning". In: second. Chapman-Hall, 2017, p. 205.

[5] Simon Rogers and Mark Girolami. "A First Course in Machine Learning". In: second. Chapman-Hall, 2017, pp. 206–212.

[6] Simon Rogers and Mark Girolami. “A First Course in Machine Learning”. In: second. Chapman-Hall, 2017, pp. 213–223.