Online Speaker Diarization with interactive learning

Anirudh Garg, Anubhav Satpathy, Nitish V. Deshpande,
Department of Electrical Engineering, Indian Institute of Technology Kanpur
Emails: anirudhg@iitk.ac.in, asatpath@iitk.ac.in, nitishvd@iitk.ac.in

EE698R Term Presentation

Motivation

Speaker diarization is the task to partition an audio stream into homogeneous segments according to the speaker identity.[1]

Limitations of traditional speaker diarization systems

(1) Usually require large datasets to train their audio extraction embeddings and clustering modules.

(2) In many real-world applications, the training set can be limited and hard to collect.

(3) Applying pretrained modules to out-of-distribution environments can be problematic.

What is online learning?

(1) Online learning is a common problem in many practical applications where the data become available in a sequential order and later used to update the best predictor for future data or reward associated with the data features.

Applying online learning framework to speaker diarization

(1) The AI speaker diarization agent learns to detect speaker identity on the fly through reward feedbacks provided by a human annotator.

Our goal: Speaker diarization + Online learning with human-in-the-loop

[1] Xavier Anguera et al. "Speaker diarization: A review of recent research". In: *IEEE Transactions on Audio, Speech, and Language Processing* 20.2 (2012), pp. 356–370.

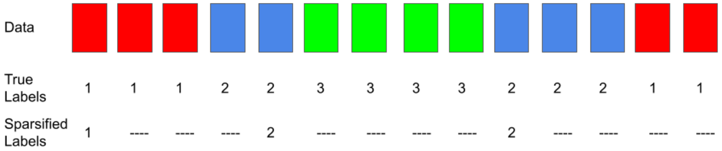
Problem Description

In many real-world applications, the reward feedbacks are NOT always revealed

(1) The reward feedback can come in different episodes or frequencies.

(2) An effective system should adapt to the nonstationarity in these observed rewards

(3) This work[2] proposed a novel problem setting called *Online Learning with Episodic Reward*.



Data	🟥	🟥	🟥	🟦	🟦	🟩	🟩	🟩	🟩	🟦	🟦	🟦	🟥	🟥
True Labels	1	1	1	2	2	3	3	3	3	2	2	2	1	1
Sparsified Labels	1	----	----	----	2	----	----	----	----	2	----	----	----	----

Figure: Sparsified labels with revealing probability $= 3/14$

Sparsified reward streams with a revealing probability are known as the epiRewards

[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.

Methodology

(1) The online speaker diarization problem is formulated as a contextual-bandit problem.

(2) Baseline Approach: Linear Upper Confidence Bound Algorithm (LinUCB)[3]

(3) Final Approach: Background Episodically Rewarded Linear Upper Confidence Bound Algorithm (BerlinUCB)[4]

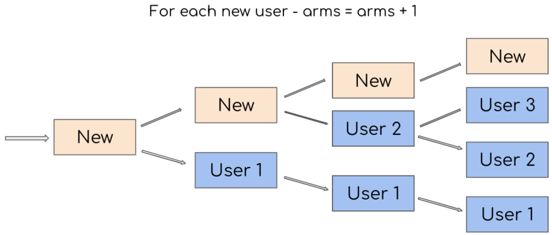


Figure: The arm expansion process of the bandit

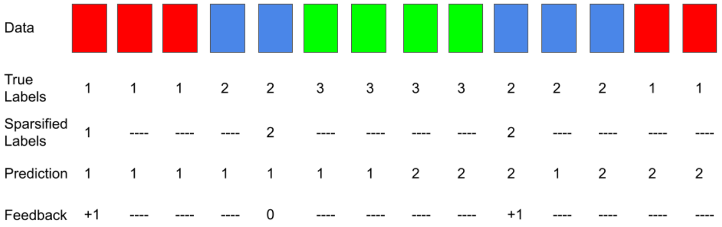
[3] Lihong Li et al. "A contextual-bandit approach to personalized news article recommendation". In: *Proceedings of the 19th international conference on World wide web*. 2010, pp. 661–670.

[4] Lin, "Online semi-supervised learning in contextual bandits with episodic reward".

Feedback model for LinUCB

(1) If predicted arm is same as the label, reward is $+1$. Else reward is 0.

(2) LinUCB only provides feedback when the background provides the label. Here, revealing probability is $3/14$.



Data	🟥	🟥	🟥	🟦	🟦	🟩	🟩	🟩	🟩	🟦	🟦	🟦	🟥	🟥
True Labels	1	1	1	2	2	3	3	3	3	2	2	2	1	1
Sparsified Labels	1	----	----	----	2	----	----	----	----	2	----	----	----	----
Prediction	1	1	1	1	1	1	1	2	2	2	1	2	2	2
Feedback	+1	----	----	----	0	----	----	----	----	+1	----	----	----	----

Figure: Feedback model

The Multi-Armed Bandit

(1) 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 $a \in \mathcal{A}_t$.

(2) 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.

(3) 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.

(4) 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 no payoff $r_{t,a}$.

(5) K-armed bandit is a specific case of the general contextual bandit problem.

(6) In it 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.

The LinUCB Algorithm

(1) Given a parametric form of payoff function, we can compute a UCB (Upper Confidence Bound) of the estimated arm payoff using the LinUCB algorithm.

(2) The LinUCB algorithm results in a closed form solution when the payoff model is linear.

(3) We define the payoff expectation for an arm *a* as:

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

Here we assume 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

(4) At the t^{th} trial, let $\mathbf{D}_a$ be a matrix of dimension m $\times$ 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:

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

where $\mathbf{I}_d$ is the dxd identity matrix.

(5) It can be shown that the payoff for arm *a* can have a reasonable UCB such that:

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

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

where $\mathbf{x}_{t,a} \in \mathbb{R}^d$ and $\mathbf{A}_a = \mathbf{D}_a^T\mathbf{D}_a + \mathbf{I}_d$.

The BerlinUCB Algorithm

(1) The BerlinUCB algorithm is an extension of the LinUCB algorithm.

(2) It is extended in the case where the background reveals no feedbacks.

(3) We use different clustering algorithms to obtain the results in this case as shown in the pseudo-code.

(4) The clustering algorithms used are the K - nearest neighbours clustering, K-means clustering and the Gaussian Mixture Modeling.

BerlinUCB Pseudo Code

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 Dataset

(1) We used VoxCeleb,[5] which is a large scale speaker recognition dataset to generate 3 different kinds of data.

(2) The 3 kinds of data correspond to 5, 10, 15 speakers to mimic the real world conversations .

(3) Three different kinds of reward streams were used with epiReward being 0.01, 0.1, 0.5.

(4) Thus a total of 9 learning environments were analysed.

(5) 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.

(6) We used Mel-frequency cepstral 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.

[5] **nagrani2017voxceleb**.

Results

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	0.430	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

(*) Clearly, for particular value of n, model performs better for data with higher epiReward probability

(*) Currently LinUCB shows best results out of all the models in 7 out of 9 training environments.

(*) Irrespective of epiReward, the case with 10 speakers gives best results for LinUCB as well as BerlinUCB

(*) Inclusion of self supervision modules into BerlinUCB doesn't seem to improve performance

Plots : Reward obtained vs timestep

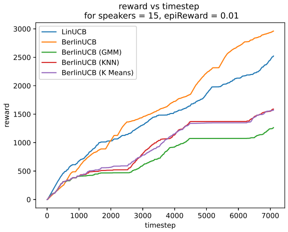


Figure: 15 speakers, epiReward = 0.01

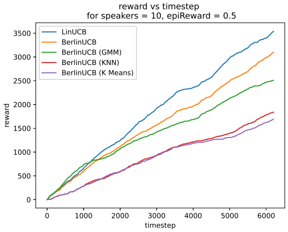


Figure: 10 speakers, epiReward = 0.50

Thank You