Advanced Machine Learning for Personalization
Homework-2: Contextual bandit algorithm using disjoint LinUCB

Author: Apoorv Purwar (ap3644)

Columbia University

Overview

This report details the steps used to implement the disjoint LinUCB Algorithm, based on the paper 'A contextual-bandit Approach to Personalized News Article Recommendation by Li, et al. and shows the results achieved under various scenarios.

Dataset

In order to implement a contextual bandit algorithm, I used the a personalized dataset constituting of 10,000 time steps by Criteo, with each time step containing the information of current user action, reward and a vector of size 100 denoting the context. Link to dataset - http://www.cs.columbia.edu/~jebara/6998/dataset.txt

Implementation & Evaluation

In order to implement the LinUCB Algorithm for the given dataset, we first parse each line of the input text file in the following manner -

- Strip every line of new line characters.
- Iterate over each line of input, which acts as individual time steps and split the line based on single space. This gives us a list of 102 integers.
- Pop the head of the list and assign it as the arm for the current step.
- From the updated list with 101 elements, pop the head again and assign it as the reward for the current step.
- Take the remaining 100 elements and assign them to the context array for the current step.

This gives us all the parameter required to perform the online prediction of the arms.

Now with all the required parameters in hand, we calculate the coefficient, payout and standard deviation for each arm at every step and select the arm with the highest payout (i.e. Upper Confidence Bound) as our prediction. This prediction is followed by an update of matrices 'A' and 'b' for the predicted arm. This is repeated for all the time steps.

The approach outlined in the Algorithm 1 is used to implement the same.

Algorithm 1 LinUCB with disjoint linear models.

Inputs: $c_t \in \mathbb{R}_+$
for $t = 1, 2, 3, \ldots, T$ **do**
  Observe features of all arms $a \in \mathcal{A}_t$: $\mathbf{x}_{t,a} \in \mathbb{R}^d$
  for all $a \in \mathcal{A}_t$ **do**
    if a is new **then**
      $\mathbf{A}_a \leftarrow \mathbf{I}_d$ (d-dimensional identity matrix)
      $\mathbf{b}_a \leftarrow \mathbf{0}_{d\times 1}$ (d-dimensional zero vector)
    end if
    $\hat{\boldsymbol{\theta}}_a \leftarrow \mathbf{A}_a^{-1}\mathbf{b}_a$
    $p_{t,a} \leftarrow \hat{\boldsymbol{\theta}}_a^\top \mathbf{x}_{t,a} + c_t\sqrt{\mathbf{x}_{t,a}^T\mathbf{A}_a^{-1}\mathbf{x}_{t,a}}$
  end for
  Choose arm $a_t = \arg\max_{a\in\mathcal{A}_t} p_{t,a}$ with ties broken arbitrarily, and observe a real-valued payoff r_t
  $\mathbf{A}_{a_t} \leftarrow \mathbf{A}_{a_t} + \mathbf{x}_{t,a_t}\mathbf{x}_{t,a_t}^\top$
  $\mathbf{b}_{a_t} \leftarrow \mathbf{b}_{a_t} + r_t\mathbf{x}_{t,a_t}$
end for

In order to evaluate the accuracy of our algorithm, we used the technique of Cumulative Take-Rate replay. The cumulative take-rate replay at time T is defined as -

$$C(T) = \frac{\sum_{t=1}^{T} y_t \times \mathbf{1}[\pi_{t-1}(\mathbf{x}_t) = a_t]}{\sum_{t=1}^{T} \mathbf{1}[\pi_{t-1}(\mathbf{x}_t) = a_t]}.$$

Whenever the predicted arm is equal to the current arm, the identity function evaluates to one and CTR is updated for that time stamp.

Results

The explore-exploit mechanism is incorporated in our algorithm, by tuning the value of 'alpha'. I tried several different mechanism to identify the values of alpha which work the best for our algorithm, and the results for the following alpha values are displayed -

- For Alpha = 1 (Figure 1):
- For Alpha = 0.001 (Figure 2)
- For Alpha = $1/\sqrt{t}$ (Figure 3)
- For Alpha = 0.001*(Correct-Predictions/10) (Figure 4)

Figure 1: Accuracy of LinUCB for alpha=1; CTR = 0.2

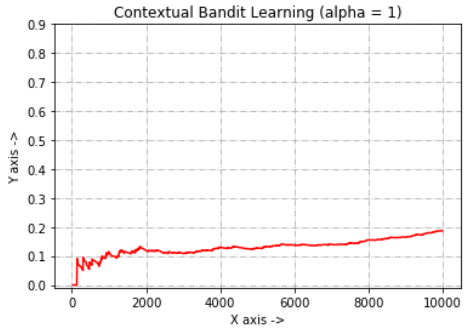


Figure 2: Accuracy of LinUCB for alpha=0.001; CTR = 0.94

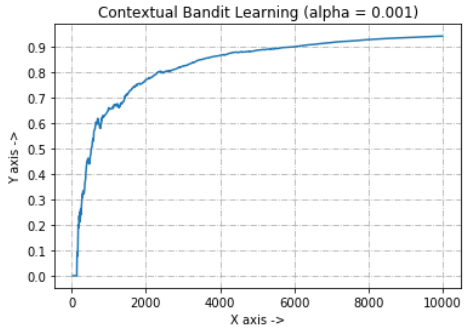
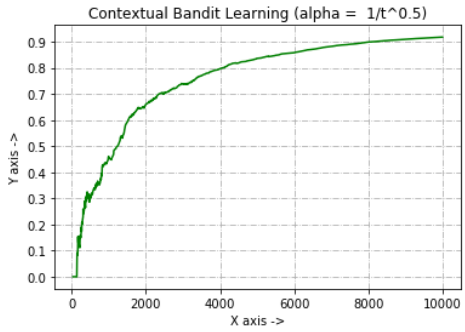


Figure 3: Accuracy of LinUCB for alpha=$1/\sqrt{t}$; CTR = 0.91



A comparative analysis of the various alpha values shows us the how accuracy of our algorithm varies based on different alpha values in Figure 5.

As it is evident from the plots, the best CTR value was achieved with alpha as '0.001*(Number of Correct Predictions/10)' when the CTR value is 0.95
This is followed by the CTR values of 0.94, 0.91 and 0.20 for alpha values of 0.001, $1/\sqrt{t}$ and 1 respectively.

The different alpha values were chosen to balance the explore-exploit trade off.

To understand this trade off better, for each alpha value, I plotted the mean UCB value for each arm and the corresponding number of correct prediction for each arm in comparison of the total count of the arms. Figures 6-9, detail the same.

Figure 4: Accuracy of LinUCB for alpha=0.001/Correct-Prediction*0.1; CTR = 0.95

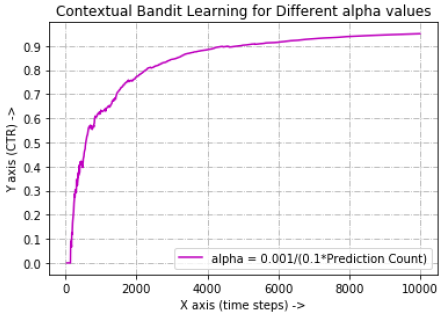
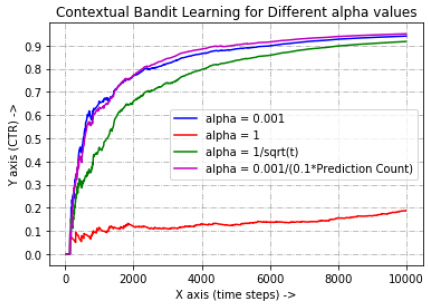


Figure 5: Accuracy for varying Alphas



As it is evident from the figure, when the alpha value is 1, there is the least amount of exploration vs exploitation, as represented by the bars which are almost of the same length throughout. Due to this reason we can also observe that the count of predictions for all the arms is almost the same, thereby giving us extremely poor CTR value of 0.2.

Now when we change our alpha value as a function of square root of time steps, we can see a significant improvement in our results. This is primarily because we are regulating our exploration as the time passes and are limiting our predictions so as to exploit the most out of our trained algorithm with the passing time.

When we use the alpha value of 0.001, we get even better results. The reason being that we are limiting the exploration to a very small value in this case, and exploiting the most. This works well particularly in this dataset, because as we can see from the total count of each arm, the distribution in this dataset is quite uniform. This assures that the pay off which we receive by exploiting more than exploring, has a positive outcome for us.

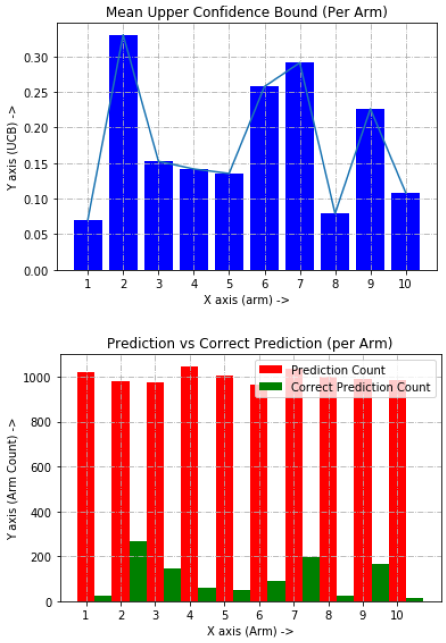
In order to improve the CTR even further and achieve even better results on this dataset, I experimented by

selecting the alpha value of 0.001, which has given us the best results as of now and tweaked it even further by dividing it by the count of the correct prediction for each arm. What this does is that it increases the exploitation particularly for the arms which are giving us better results and increased the exploration of the arms which have not been the best predictions, so far. This approach raised the CTR value of the LinUCB algorithm even further to 0.9508, which has been the best CTR rate of all the alpha values I experimented with.

Reducing the alpha value even further from 0.001 wasn't very helpful, as it kept the CTR rate approximately the same. Which made me believe that the number of 0.001 was the sweet spot.

Another interesting pattern which can be observed in these plots is that the arm for which the UCB value is maximum has the most number of predictions, followed by the arm with second largest mean UCB and so on. This validates the fact that LinUCB selects the arm with the highest Upper Confidence Bound as the prediction. And, this is an indication of the fact that the implemented algorithm is correct.

Figure 6: Mean UCB and Correct vs Total Arms for Alpha = 0.001



Conclusion

From our experiments, it is safe to assume that LinUCB algorithm is indeed helpful in tackling the challenges in Contextual and Online Learning and in building effective Recommendation Engines.

Figure 7: Mean UCB and Correct vs Total Arms for Alpha = 1

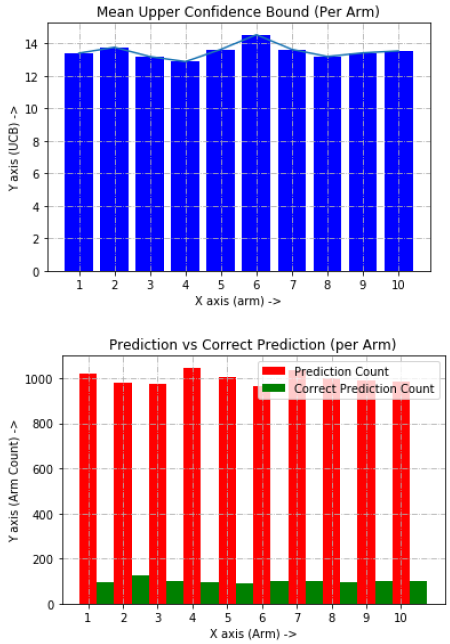
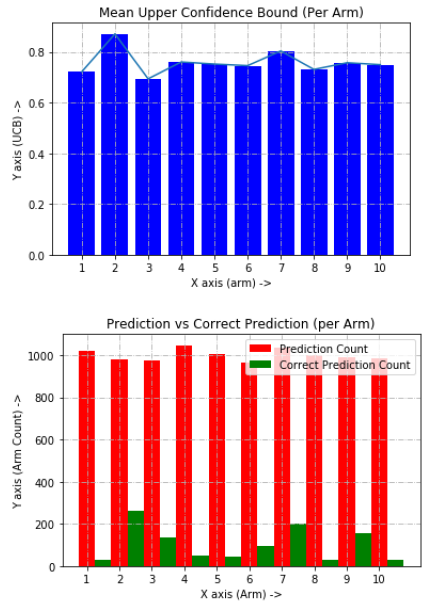
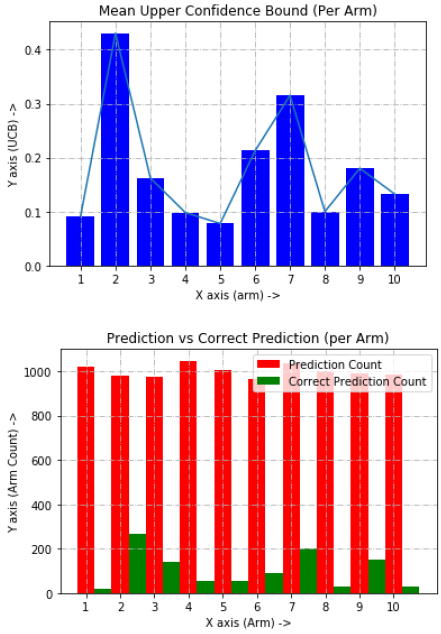


Figure 8: Mean UCB and Correct vs Total Arms for Alpha = $1/\sqrt{t}$



Moreover, it was apparent from the experimentation that the choice of 'alpha' is very important as it governs the exploitation vs exploration trade off and can drastically improve the results, if selected wisely. Understanding how the UCB value is distributed with reference to the correct answers for each arm was significantly helpful in this, as then I

Figure 9: Mean UCB and Correct vs Total Arms for Alpha = 0.001 * Correct-Prediction/10



was able to come up with an even more effective formula to control the exploration vs exploitation trade off, and receive my best result of 0.9508 as the CTR value.

References

Following papers and resources were referenced for this assignment -

1) Li, Lihong; Chu, Wei; Langford, John; Schapire, Robert.
A Contextual-Bandit Approach to Personalized News Article Recommendation.

2) Lihong Li, Wei Chu, John Langford, Xuanhui Wang
Unbiased Offline Evaluation of Contextual-bandit-based News Article Recommendation Algorithms

3) http://www.cs.columbia.edu/ jebara/6998/Notes6.pdf

4) http://www.cs.columbia.edu/˜jebara/6998/dataset.txt