CS230

Personalized Ad Recommendation Systems with Deep Reinforcement Learning

Andrei Cheremukhin
Department of Computer Science
Stanford University
cheremuk@stanford.edu

Abstract

In this project, we explore deep learning (DL) techniques and reinforcement learning (RL) algorithms to learn good policies for personalized ad recommendation systems. While most systems maximize the immediate gain, a better approach would be the life time value (LTV) of the user-system interaction. RL algorithms take into account the long-term effect of an action, and thus, are more suitable choice for this domain. The LTV approach has been explored before with the hand-tuned features. With recent exciting achievements of deep RL it is a natural question if we can apply DL techniques to obviate the feature selection procedure.

1 Introduction

Recommendation systems are widely used by almost every commercial site, proposing users goods, news articles and other items which are likely relevant to their interest. A policy can recommend content considered to be the best for each particular used based on a profile of that user's interests and preferences inferred from their history of online activity. Machine learning, and in particular, reinforcement learning is suitable choice to learn the policy. Good policy can help to increase users' satisfaction with a website or the yield of a marketing campaign.

2 Related work

Prior this has been formalized as a contextual bandit problem with the objective of maximizing the total number of user clicks by Li, Chu, Langford, and Schapire (2010). Their objective was to maximize the click-through rate (CTR), which the ratio of the total number of clicks all users make on a web-page to the total number of visits to the page.

$$CTR = \frac{Total\#ofClicks}{Total\#ofVisits} * 100$$

Theocharous, Thomas, and Ghavamzadeh (2015) argued that better results are possible by formulating personalized recommendation as a Markov decision problem (MDP) with the objective of maximizing the total number of clicks users make over repeated visits to a website. They applied a random forest (RF) algorithm and a batch-mode reinforcement learning algorithm called fitted Q iteration (FQI) aiming to improve the number clicks users made over multiple visits to a website. They called this algorithm life-time value (LTV) optimization. To measure the performance of the model Theocharous et al. used the LTV metric.

$$LTV = \frac{Total\#ofClicks}{Total\#ofVisitors} * 100$$

There is still one major obstacle hindering of the widespread application of the RL technology to PAR: how to evaluate the performance of a policy returned by a RL algorithm without deploying it using only the historical data that has been generated by one or more other policies). This problem, also known as off-policy evaluation evaluation, is of extreme importance not only in ad recommendation systems, but in many other domains such as health care and finance. Thomas et al. (2015) recently proposed recently proposed model-free approach that computes a lower-bound on the expected return of a policy using a concentration inequality to tackle the off-policy evaluation problem.
We use the same metric to compare the performance of our DL approach.

3 Dataset and Features

For our project we use the following data set the "Coupon Purchase Prediction" challenge from the Kaggle web site. Recruit Ponpare is Japan's leading joint coupon site, offering huge discounts on everything from hot yoga, to gourmet sushi, to a summer concert bonanza. On the coupon site when customers visit, they are shown one of a finite number of offers. The goal of the competition was to improve the recommendations given in the Ponpare coupon site which offers discounts in various markets. The reward is 1 when a user buys on the offer and 0, otherwise. The data set includes categories shown in Table 1.

Features	Description
Customer information	registered date, gender, age, unregistered date, residential prefecture
Coupon information	category, discount rate, list price, discount price, sales release etc.
Coupon browsing log	purchased flag, view date, customer id, coupon id etc.
Coupon area	small area name, listed prefecture name, coupon id

The samples were gathered over a time interval of roughly one year and include 23K anonymized users and 33K items. Before handling the data, we have removed users that had less then 20 interactions with the system or have zero clicks, and we were left with only 13K users and an average success rate of 0,042.
In sequential recommender systems the state features are often composed of engineered parameters such as the time steps since the last click, the cumulative reward or the user's demographics (Pednault et al., 2002; Theocharous et al., 2015). These suffer from scaling and tuning issues, redundancy. Here, we will not proceed with feature engineering besides adding very basic features describing our state space e.g. time since last visit, sum of previous reward, the number of visits so far, the last time there was positive reward. Including these features in our data set allows us to formulate our problem as a RL problem. Hand-crafted feature set will be fixed during all experiments.

4 Methods

4.1 Reinforcement learning

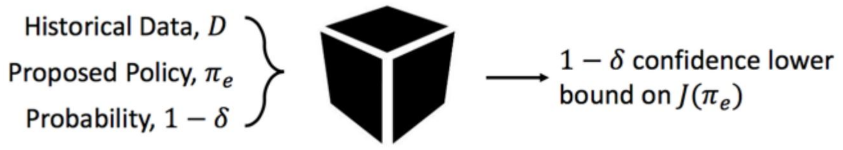
We model the system as a Markov decision process (MDP). We denote by s_t the feature vector describing a user's t^{th} visit to the website and by a_t the t^{th} ad shown to the user and refer to them as a state and an action. We call r_t the reward, which is 1 if the user clicks on the ad a_t and 0, otherwise. We assume that the users visit at most Ttimes and set $T = 20$ according to the users in our data set. We write $\tau = \{s_1, a_1, r_1, s_2, a_2, r_2, ..., s_T, a_T, r_T\}$ to denote the history of interactions with one user, and we call τ a trajectory. The return of a trajectory is the discounted sum of rewards, $R(\tau) = \sum_{t=1}^{T} \gamma^{t-1}$, where is γ a discount factor.
A policy π is used to determine the probability of showing each ad. Let $\pi(a|s)$ denote the probability of taking action a in state s, regardless of the time step t. The goal is to find a policy that maximizes the expected total number of clicks per user $p(\pi) = E[R(\tau)|\pi]$. Our historical data is a set of trajectories, one per user. Formally, D is the historical data containing n trajectories $\{\tau_i\}_{i=1}^{n}$, each labeled with the behavior policy π_i that produced it. We are also given an evaluation policy π_e that was produced by a RL algorithm, the performance of which we would like to evaluate.

4.2 Batch RL and High-confidence off-policy evaluation (HCOPE)

Thomas et al. (2015) recently proposed High-confidence off-policy evaluation (HCOPE), a family of methods that use the historical data D in order to find a lower-bound of the performance of the evaluation policy π_e with confidence $1-\beta$. We use the following HCOPE approach which is based on importance sampling estimator: concentration inequality (CI).
The importance sampling estimator:

$$\hat{p}(\pi_e|\tau_i, \pi_i) = R(\tau_i) * \prod_{t=1}^{T} \frac{\pi_e(a_t^{\tau_i}|s_t^{\tau_i})}{\pi_i(a_t^{\tau_i}|s_t^{\tau_i})}$$

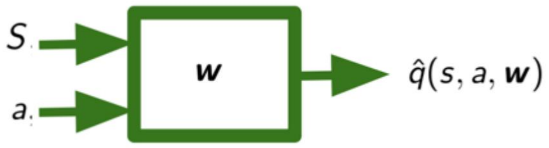
We write $\underline{\rho}^{CI}(X,\beta)$ to denote the $1-\beta$ confidence lower-bound produced by our algorithm to be consistent with Theocharous et al. (2015).



We assume that our data set contains user trajectories produced by a myopic agent with a random strategy with $\gamma = 0$. We split the random strategy data into a training, a validation and a test set in the following proportion 90%:5%:5%. The training data set is used to optimize an ϵ-greedy with $\epsilon = 0.1$ and LTV ad recommendation with $\gamma = 0.9$ strategies.

4.3 Ad recommendation algorithm

We use deep Q network (DQN) for a state-action value function approximation instead of a table which allows us to handle high-dimensional continues and discrete variables: in particular, coupon and customer information. DQN has H hidden fully connected layers with N neurons each and RELU as an activation function similar to V. Mnih et. al. (2015). The output layer uses sigmoid σ as an activation function to produce probability of showing the particular ad. H and N are our hyperparameters.



DQN allows us to use the state-of-the-art RL algorithm, called neural fitted Q-iteration algorithm. Despite M. Riedmiller (2005) recommended to use resilient backpropagation algorithm Rprop, we use vanilla Stochastic Gradient Descent with mean squared error as loss.

$$L(w) = \sum_{\pi}[(q_\pi(s,a) - \hat{q}_\pi(s,a,w))^2]$$

and the following update rule for Q-learning

$$\Delta w = \alpha(r + \gamma * max_{\grave{a}}\hat{q}(\grave{s}, \grave{a}, w) - \hat{q}(s,a,w))\nabla_w \hat{q}(s,a,w)$$

We start with three data sets an X_train, X_val, X_test. Each one is made of complete user trajectories. A user only appears in one of those sets. We provide the pseudo code below for LTV optimization algorithm:

```
NFQ() {
  Q = init_DQN()
  generate pattern set P = {(inputt, targett), t = 1, ..., #D}
     where:
    inputt = st, at,
    targett = r(st, at, st+1) + gamma * maxQ(st+1, a)
  Qmax = SGD_training(P)
  return Qmax
}

LTV_optimization() {
  b = randomPolicy()
  for i = 1 to K do {
    Qmax = NFQ()
    e = epsilonGreedy(Qmax, Xvalidation)
    W = importanceWeigtedReturns(e, b, Xvalidation)
  }
  e = epsilonGreedy(Qbest, Xtest)
  return  importanceWeigtedReturns(e, b, Xtest)
}
```

We use a myopic agent with $\gamma = 1$ as a baseline model.

5 Experiments/Results/Discussion

For our experiments we had ϵ and γ fixed:

$$\epsilon = 0.1$$

$$\gamma = 0.9$$

We splitted the random strategy data into a training set, a test set and a validation set. We used the training data for training to optimize the greedy and LTV strategies. When a deep neural network is used as an arbitrary function appropriator in the NFQ algorithm, it does not converge monotonically, but rather osculates during training iterations. To alleviate the oscillation problem of NFQ, we used high confidence off-policy evaluation (HCOPE) framework within training loop. The loop keeps track of the best NFQ result according to a validation data set. Our final DQN has 3 fully connected layers with 16 neurons each: $H = 16, N = 16$.

With our algorithm we produce performance results both for the CTR and LTV metrics. To produce results for CTR we assumed that each visit is a unique visitor. In our experiment we compared LTV and CTR metrics. In general the LTV metric gives higher numbers than the CTR metric. Estimating the LTV metric however gets harder as the trajectories get longer and as the mismatch with the behavior policy gets larger. In this experiment the policy we evaluated was the random policy which is the same as the behavior policy.

6 Conclusion/Future Work

In this project, we used a framework for training and evaluating personal ad recommendation (PAR) strategies. This framework is mainly based on a family of high confidence off-policy evaluation (HCOPE) techniques recently developed Thomas et al., 2015. We showed that using these HCOPE techniques together with RL algorithms and deep Q network it is possible to learn and evaluate PAR strategies that optimize for customers' life-time value (LTV) without any hand-crafted feature selection. Also, these HCOPE techniques can also be used to evaluate the performance of a myopic strategy that optimizes for click through rate (CTR), and to provide high confidence bounds for it. We experimented with data set generated from real-world PAR campaigns to show the effectiveness of deep learning and reinforcement learning algorithms.

If we had more time we would like to do more experiments with other data sets e.g. with MovieLens's 20M data set. Also, it would be interesting to try other lower-bound estimation methods e.g. Student's t-test and Bias Corrected and Accelerated Bootstrap (BCa) and compare them with each other. Other useful experiment is exploring the effect of ϵ.

References

[1] D. Silver, L. Newnham, D. Barker, S. Weller, and J. McFall. Concurrent reinforcement learning from customer interactions. In In 30th International Conference on Machine Learning, 2013.

[2] R. Sutton and A. Barto. Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA, 1998.

[3] P. S. Thomas, G. Theocharous, and M. Ghavamzadeh. High confidence policy improvement. In Proceedings of the International Conference on Machine Learning, 2015.

[4] G. Theocharous, P. S. Thomas, M. Ghavamzadeh. Personalized Ad Recommendation Systems for Life-Time Value Optimization with Guarantees. In Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.

[5] L. Li, W. Chu, J. Langford. A Contextual-Bandit Approach to Personalized News Article Recommendation. 2010.

[6] A. Hallak, E. Yom-Tov, Y. Mansour. Automatic Representation for Life-Time Value Recommender Systems. 2016.