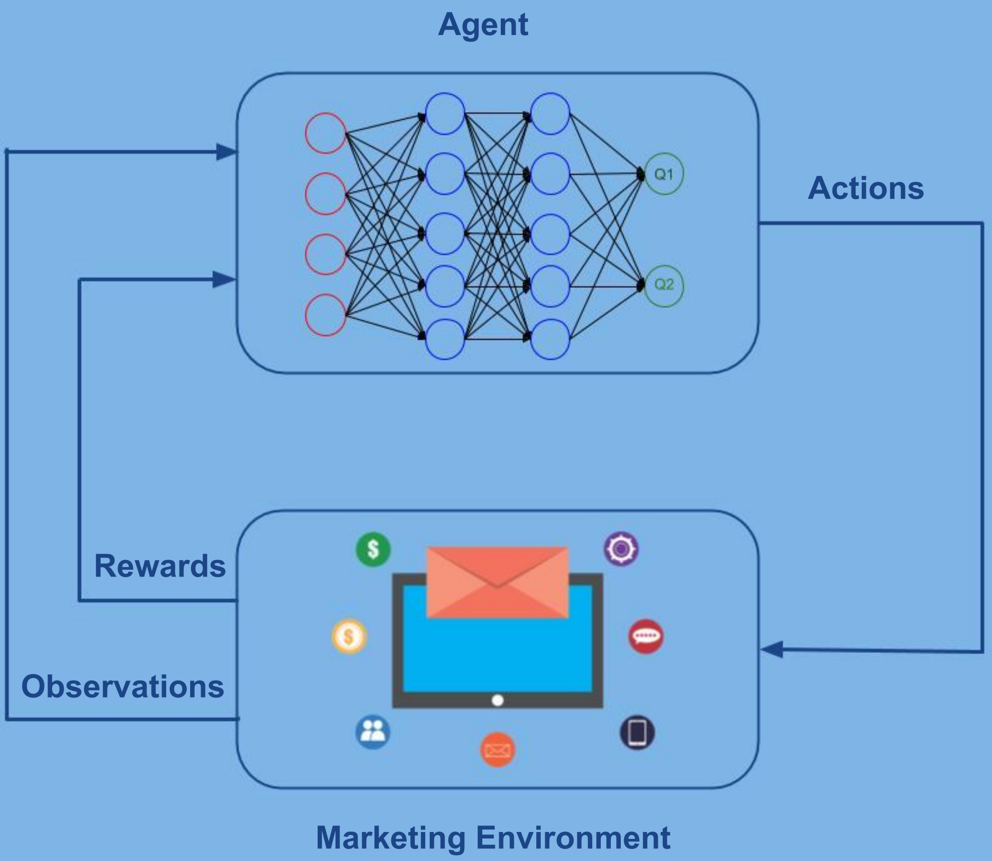
Artificial Intelligence CONFERENCE

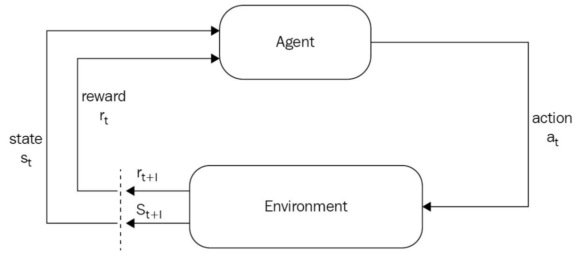
Evaluate Deep Q-Learning for Sequential Targeted Marketing with 10-fold Cross Validation

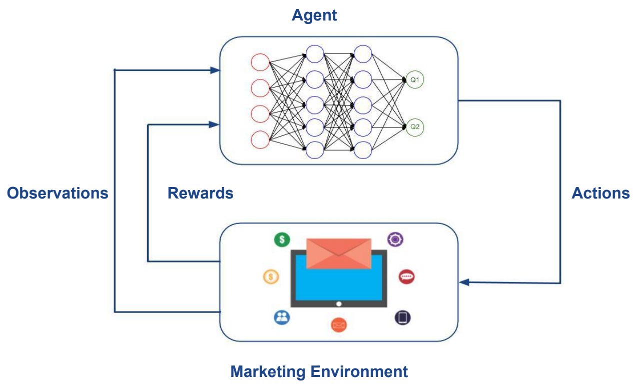
Jian Wu
Full Stack Engineer

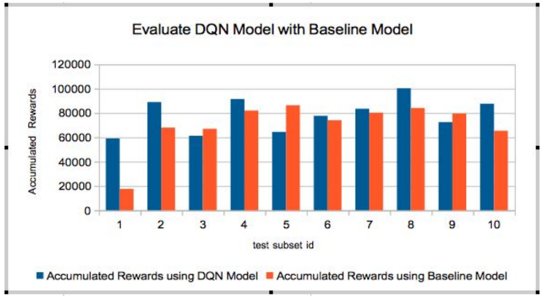


The idea of this talk was developed at Samsung SDSRA AI Lab among Kannan Parthasarathy, Luis Quintela, Alex Beloi, Abhishek Mishra, and myself when working on AI and Machine Learning Framework Project.

Special Thanks go to Kannan Parthasarathy, Luis Quintela, Alex Beloi, Abhishek Mishra, and Samsung SDSRA AI Lab to make this talk possible.







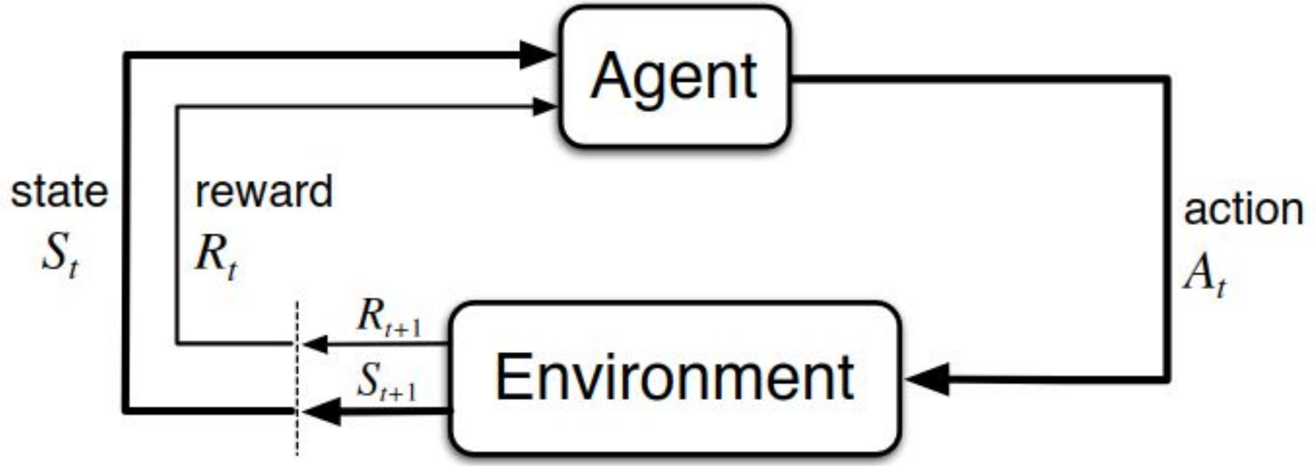
1. **Introduction to Deep Q-Learning: Training and Evaluation**

2. Applying Deep Q-Learning to Enterprise Applications

3. Evaluate Deep Q-Learning for Sequential Targeted Marketing

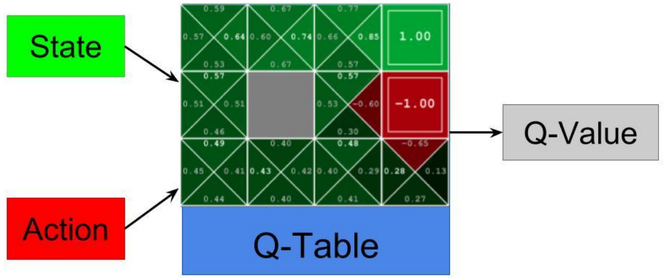
Reinforcement Learning: Overview

In a Markov Decision Process (MDP), an agent learns an optimal rewarding policy through interaction with environment, taking Actions and observing the received Rewards ...



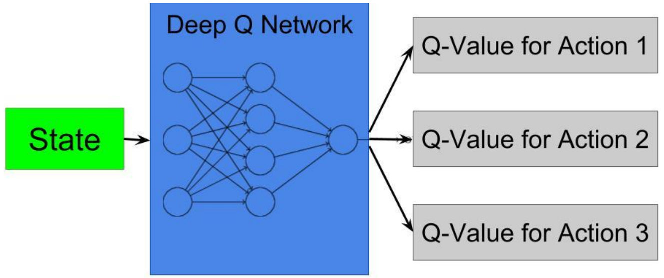
[Drawing from Sutton and Barto, Reinforcement Learning: An Introduction]

Q-Learning & Deep Q-Learning



(Tabular) Q-Learning

Q-Learning is one of the value based RL algorithms, it learns an action value function Q(s,a), which produces the expected future rewards when taking action “a” at state “s”, you can think that Q function is a measure of the “good quality” of the action “a” at state “s”. In small discrete state-action space, Q function can be implemented with a Q-Table



Deep Q-Learning

Deep Q-Learning is trying to use a deep neural network to approximate Q function, hence called “Deep” Q-Learning,

For large or continuous state-action space, Deep Q Network is an effective way to approximate Q function

Training Deep Q-Learning Model: Tips & Tricks

From “Lecture 3 DQN + Variants” by Volodymyr Mnih at 2017 Deep RL Bootcamp at Berkeley

1. Experience Replay using Replay Buffer

   -- Apply Q-updates on batches of past experience instead of online

   -- Make the data distribution more stationary

2. Using less frequently updated network to compute targets (Target Network)

   -- Keeps the target function from changing too quickly

3. Using Huber loss Function instead of squared loss

   -- Make Q-Network more stable

4. Using RMSprop or Adam Optimizer instead of vanilla SGD

   -- Optimization in RL really matters

Evaluate Deep Q-Learning Model(s)

As Deep Q-Learning is to find a policy/model optimized for better long term rewards, we need to verify that the trained model does consistently produce higher accumulated rewards over certain period of time with enough sequential steps/episodes.

Also, recent studies(*) found that “Deep Reinforcement Learning could overfit in various ways. … In particular, the same agents and learning algorithms could have drastically different test performance, even when all of them achieve optimal rewards during training”.

To make sure that Deep Q-Learning does produce a generalized model, “cross validation” is one practical way to check/certify the generalization performance. We divide all the sample data into several sub-datasets, and shuffle these sub-datasets into separate training and test sets. During training, the agents have access to all the data from the training datasets; then trained model will be tested/validated with the initial states from the test dataset.

(*) A Study on Overfitting in Deep Reinforcement Learning
Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents

Evaluate Deep Q-Learning effectively using simulator(s)

When evaluating Deep Q-Learning model with test data, it’ll likely lead to different state transition and episodic trajectories, the simulators are often used to evaluate trained model(s) in a cost-effective and timely way as it might be taking long time and/or very expensive to evaluate trained model in real production environment.

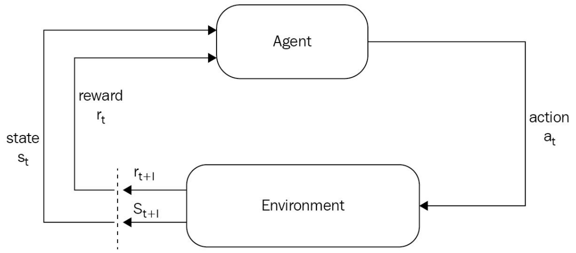
Simulators or simulation environments have been widely used in some industries like aviation and gaming, but building simulators to evaluate RL models in different business domains is still very hard, several research papers in the past have discussed some techniques and tools to build such simulators:

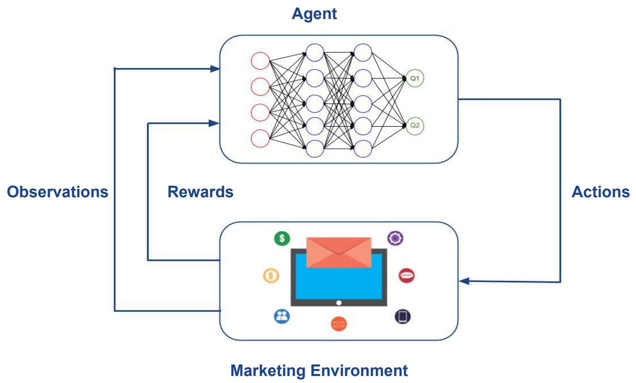
-- Tree and Random Forest based simulator from Sequential Cost-Sensitive Decision Making with Reinforcement Learning

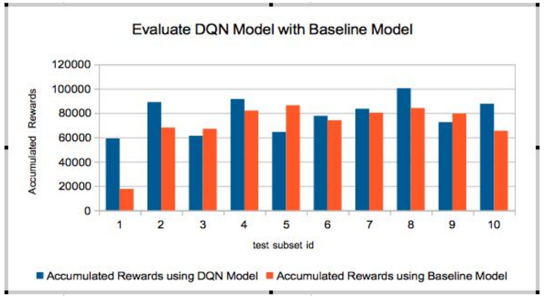
-- Simulator using nearest neighbour representation from Concurrent Reinforcement Learning from Customer Interaction

-- Bayesian network based simulator from Customer Simulation for Direct Marketing Experiments

For our Deep Q-Learning project, we built a K-means cluster based simulator using scikit-learn for testing and evaluation.







1. Introduction to Deep Q-Learning: Training and Evaluation

2. Applying Deep Q-Learning to Enterprise Applications

3. Evaluate Deep Q-Learning for Sequential Targeted Marketing

Apply Deep Q-Learning to Enterprise Applications

1. Apply Deep Q-Learning with offline batch training

   As Deep Q-Learning supports flexible off-policy sampling and minibatch processing to update Q-Network during training, we can extend Deep Q-Learning to offline batch training, leveraging "Lambda architecture" with large volume of existing/historical customer interaction data

2. Many technical challenges when applying Deep Q-Learning to enterprise applications

   -- Identify proper business use case for Deep Q-Learning

   -- Identify proper environment for Deep Q-Learning

   -- Coherent environment time >> Convergence time

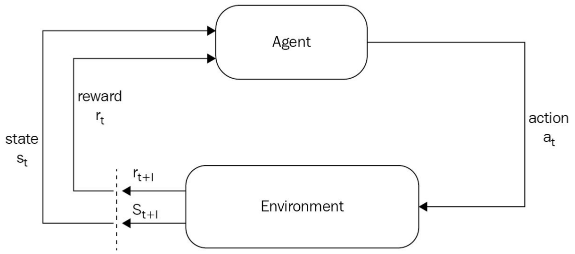
   -- Design proper business actions

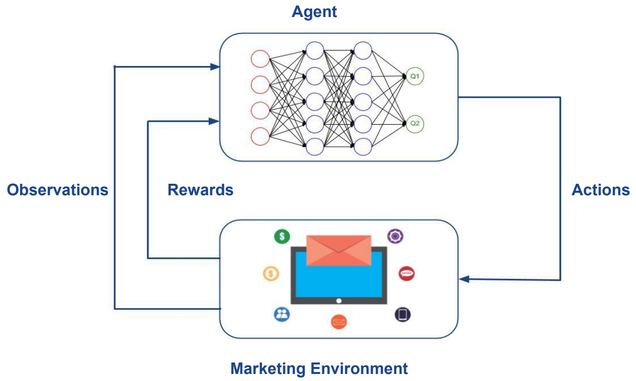
   -- Design proper reward function

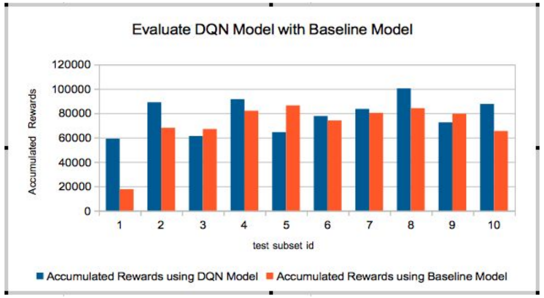
   -- be aware the unintended consequences with reward function

   -- Design proper strategy to explore state action space

From "Reinforcement Learning in the Wild & Lessons Learned" by Mohamad Charafeddine, Director, Technology Planning, Samsung SDSRA at AI World Forum San Francisco 2018







1. Introduction to Deep Q-Learning: Training and Evaluation

2. Applying Deep Q-Learning to Enterprise Applications

3. **Evaluate Deep Q-Learning for Sequential Targeted Marketing**

Deep Q-Learning for Sequential Target Marketing

We'll demonstrate applying Deep Q-Learning to Sequential Target Marketing with public dataset KDD Cup 1998 dataset, from data processing, building simulator, training and evaluation.

This dataset was collected by a non-profit organization that helps US Veterans raising money via direct mailing campaigns, it is a well studied dataset for RL algorithms by several research papers, including:

-- Sequential Cost-Sensitive Decision Making with Reinforcement Learning

-- Recurrent Reinforcement Learning: A Hybrid Approach

-- Autonomous CRM Control via CLV Approximation with Deep Reinforcement Learning in Discrete and Continuous Action Space

-- Customer Simulation for Direct Marketing Experiments

In these research papers, KDD-Cup-98 were mainly used through sampling.

In our experiment, the full KDD-Cup-98 dataset are used for training and evaluation. Some basic statistics of the dataset are:

-- Total number of records: 1526592

-- Total number of persons: 95763

-- Max reward: 999.32 (donation minus mailing cost), Min reward: -0.68 (mailing cost)

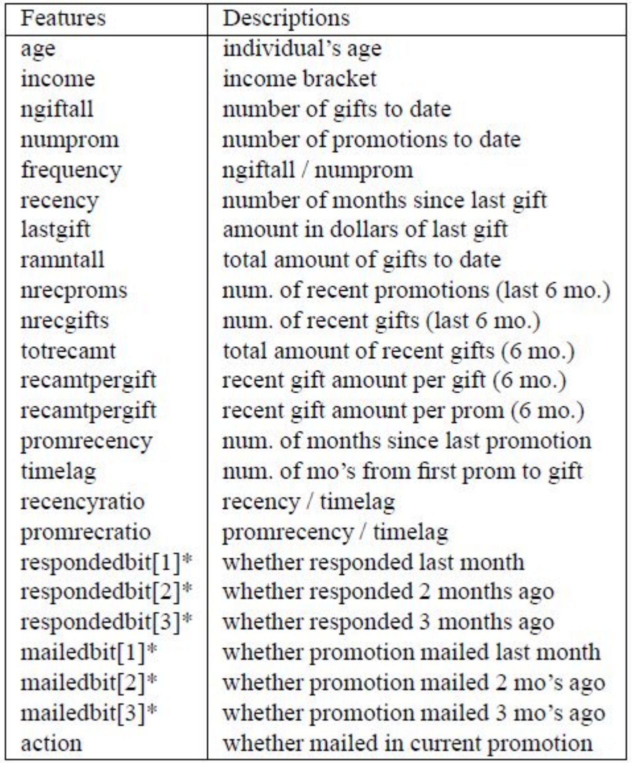
Processing Dataset + Building Simulators

1. Processing Dataset

The original dataset is first transformed to a dataset with 23 feature vector including several derived temporal features as right feature list table shown

2. Building K-means Cluster based Simulator

Using transformed dataset, we build a K-means Cluster based simulator for evaluating the trained DQN Model and Baseline Model. Our simulator implements OpenAI Gym Environment interface, and can be used as one gym environment.



Features	Descriptions
age	individual's age
income	income bracket
ngiftall	number of gifts to date
numprom	number of promotions to date
frequency	ngiftall / numprom
recency	number of months since last gift
lastgift	amount in dollars of last gift
ramntall	total amount of gifts to date
nrecproms	num. of recent promotions (last 6 mo.)
nrecgifts	num. of recent gifts (last 6 mo.)
totrecamt	total amount of recent gifts (6 mo.)
recamtpergift	recent gift amount per gift (6 mo.)
recamtpergift	recent gift amount per prom (6 mo.)
promrecency	num. of months since last promotion
timelag	num. of mo's from first prom to gift
recencyratio	recency / timelag
promrecratio	promrecency / timelag
respondedbit[1]*	whether responded last month
respondedbit[2]*	whether responded 2 months ago
respondedbit[3]*	whether responded 3 months ago
mailedbit[1]*	whether promotion mailed last month
mailedbit[2]*	whether promotion mailed 2 mo's ago
mailedbit[3]*	whether promotion mailed 3 mo's ago
action	whether mailed in current promotion

From "Empirical Comparison of Various Reinforcement Learning Strategies for Sequential Targeted Marketing"

Training DQN Models

1. The transformed dataset is splitted into 10 sub datasets by:

   -- Extract person id as key from feature data set

   -- Randomly split person id set to 10 subsets

   -- Partition the full dataset into 10 sub datasets using 10 person id subsets

2. For 10 sub datasets, each sub dataset is used as test dataset once, 9 remaining sub datasets are used as training dataset to train DQN model, total 10 DQN models are trained with same Neural Network Architecture, discount rate and learning rate.

3. We also trained 10 MLP based regression models using the same training dataset with same Neural Network Architecture as baseline models for evaluation.

Evaluate Trained DQN Models by Simulation (1)

Each trained DQN model and baseline model(MLP regression model) are evaluated with the matching test dataset.

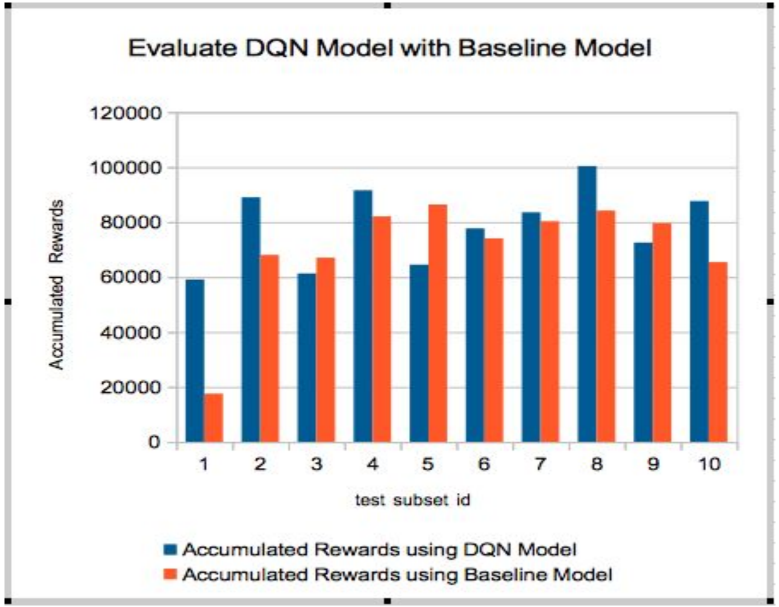
For each person in test subset, his/her initial state is used as start point, each person will run through 20 campaigns with trained DQN model against the simulator we built with K-means clusters. The associated baseline model is also evaluated in the same way for comparison.

For all the simulated campaigns through evaluation, DQN models outperformed baseline models with more accumulated rewards(*) in 7 of 10 test sub datasets.

DQN models produce higher total rewards with less mailing ratio; DQN models often produce higher rewards at later campaigns for higher total accumulated rewards.

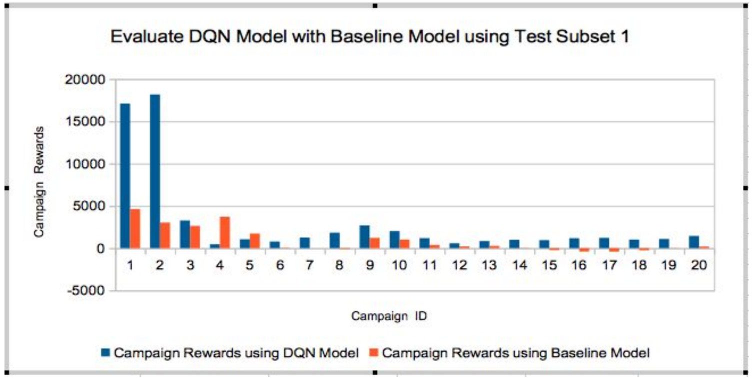
These results indicate that DQN models have the better mailing strategies towards the better long term rewards.

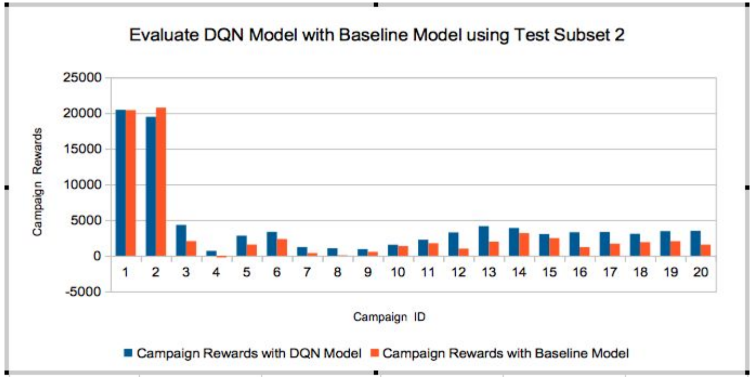
(*) These rewards as shown in the chart are average values of 3 test run for each DQN model and baseline model.

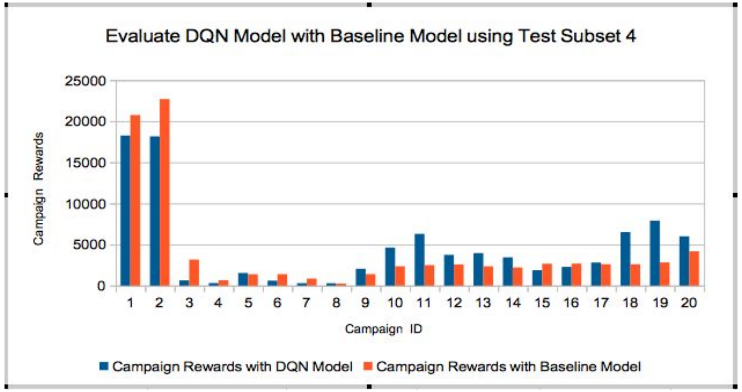


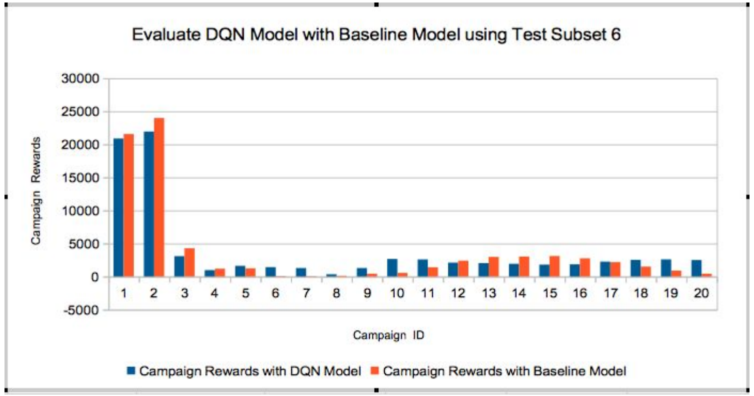
Evaluate Trained DQN Models by Simulation (2)

DQN models have the better results on test subsets 1, 2, 4, and 6



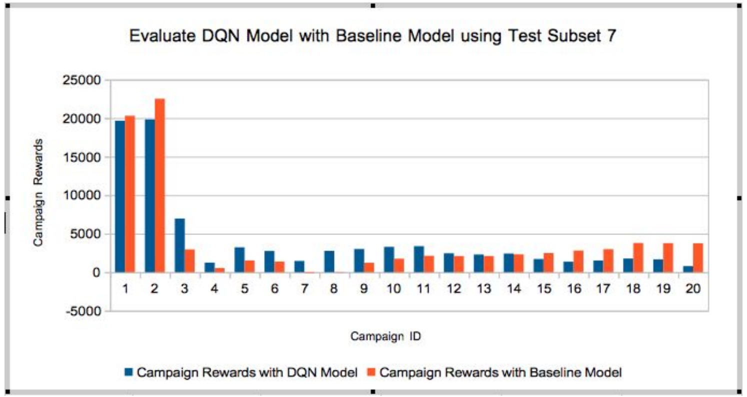


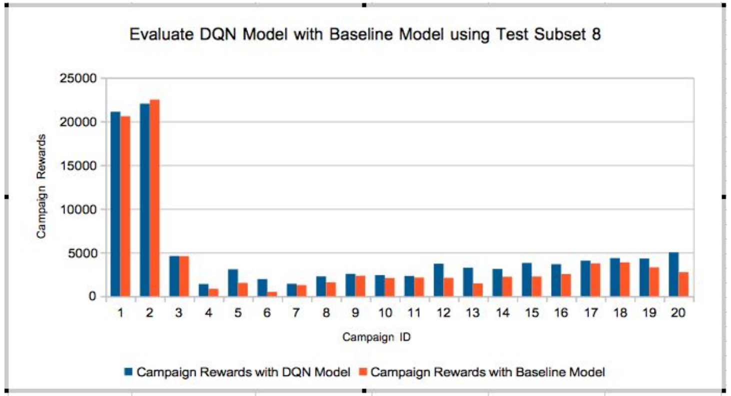


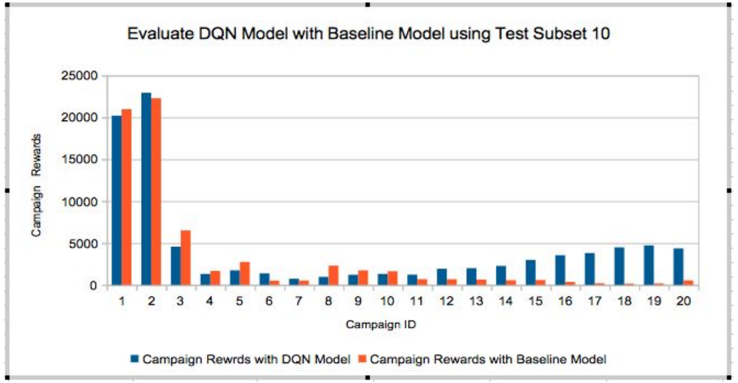


Evaluate Trained DQN Models by Simulation (3)

DQN models also have the better results on test subsets 7, 8 and 10

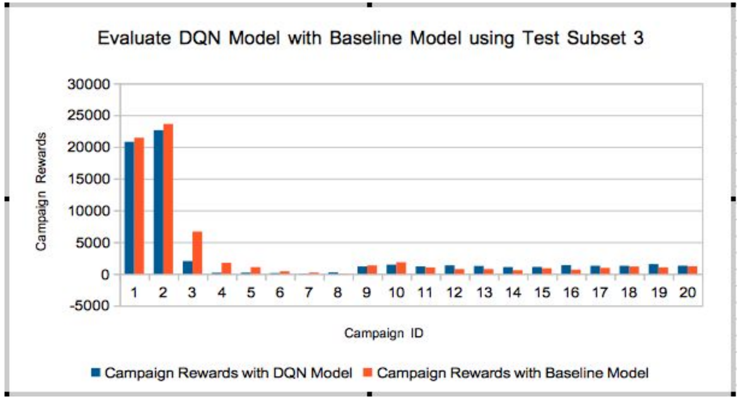


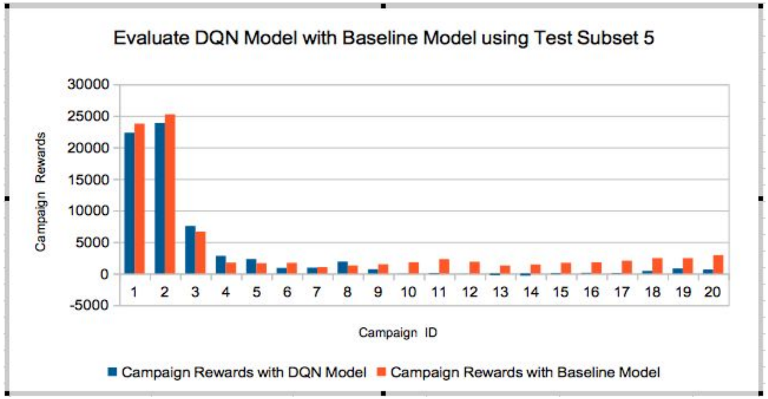


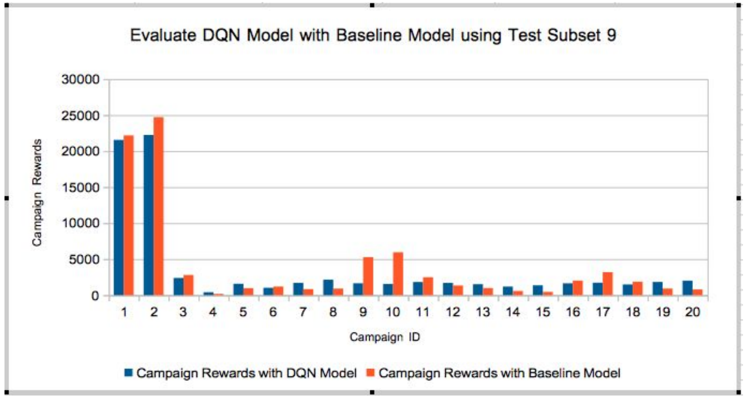


Evaluate Trained DQN Models by Simulation (4)

Baseline models outperformed DQN Models on test subsets 3, 5, and 9







Future Improvements

1. Running and automating the whole evaluation process on Kubernetes cluster

2. Finding better neural network architecture for Deep Q-Learning through hyperparameter search on neural network structure, including number of layers and number of nodes per layer.

3. Building better data-driven and domain-specific simulator for evaluations