Recommendation using Reinforcement Learning

Kaitao Yang

Kaitao Yang

Senior Data Engineer
Meta · Full-time
Mar 2022 - Present · 9 mos
Boston, Massachusetts, United States

Senior Data Scientist, Senior Data Science Manager
Philips · Full-time
Dec 2019 - Nov 2021 · 2 yrs
Amsterdam, North Holland, Netherlands

Data Scientist (Deep Learning)
eBay · Full-time
Sep 2017 - Dec 2019 · 2 yrs 4 mos
Amsterdam Area, Netherlands

Data Engineer (Deep Learning)
Jheronimus Academy of Data Science · Full-time
May 2016 - Aug 2017 · 1 yr 4 mos
's-Hertogenbosch

Education

Eindhoven University of Technology
Doctorate in Engineering, Electrical Engineering
2012 - 2016

Xiamen University
Master of Science, Computer Science
2009 - 2012

Minzu University of China
Bachelor of Engineering, Electrical Information Engineering
2005 - 2009

Comparison of policies

ID	Policy	Number of success
1	Always select `version1`	12,772
2	Always select `version2`	12,779
3	Choose an action (version1 or version2) which had the higher success rate yesterday	13,436
4	The same as policy #3, except that has ε=0.01 probability to uniform-randomly select an action.	mean: 13,427, std: 36
5	Use a model to predict action (version1 or version2) based on new_movers, length_of_residence, model at time t is trained using data at time t-1. Model is a decision tree.	19,496
6	The same as policy #5 except that model is a logistic regression model.	**19,596**
7	Choose version1 if new_comer<=0.5 else version2	19,451

Outline

- Data Exploration
- Reinforcement learning introduction
- Multi-armed Bandit
- Contextual Multi-armed Bandit
- Future work

Data Exploration

Dataset statistics	
Number of variables	20
Number of observations	513863
Missing cells	240184
Missing cells (%)	2.3%
Duplicate rows	0
Duplicate rows (%)	0.0%

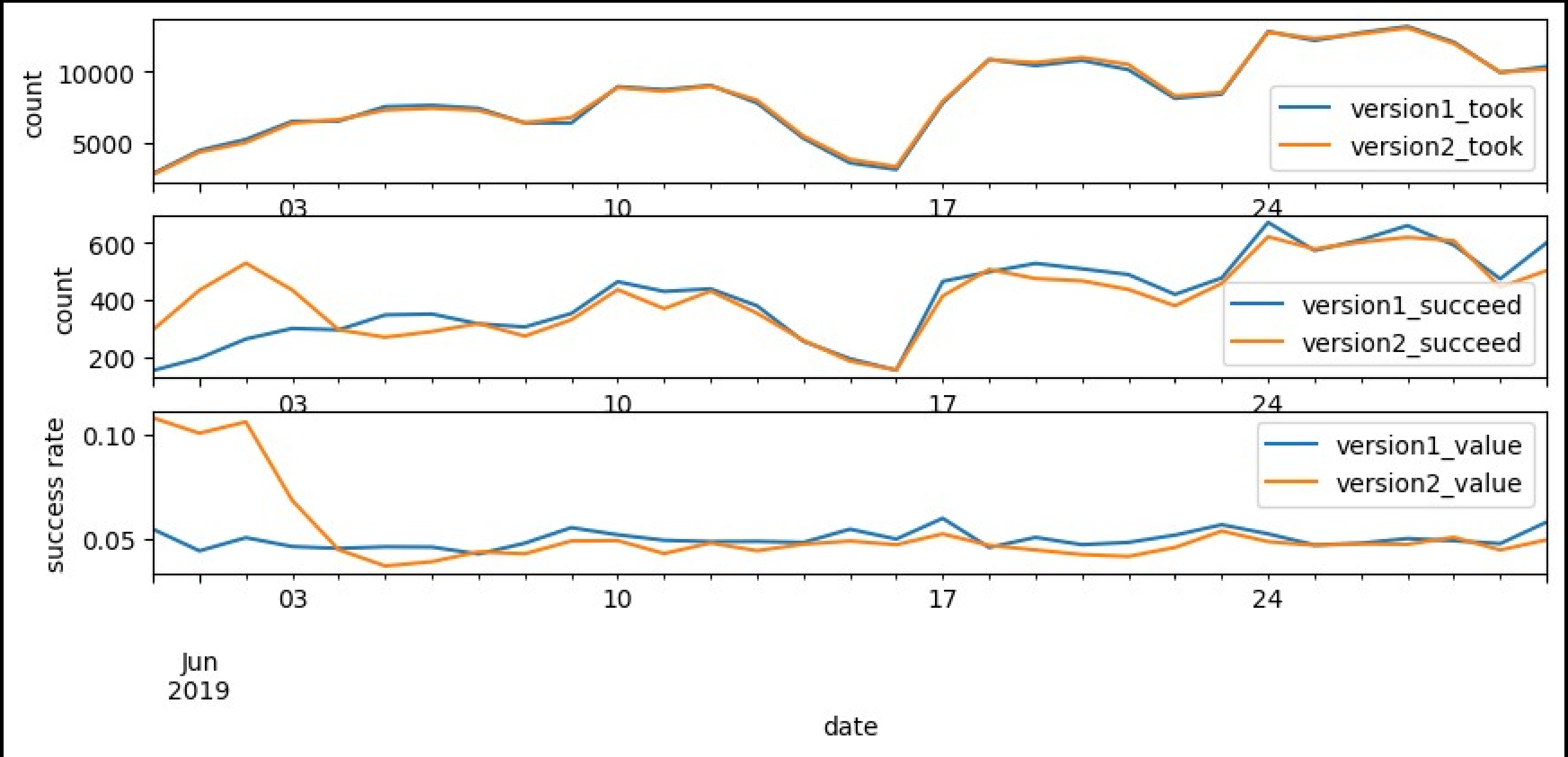
Variable types	
Numeric	12
Categorical	8

	success=0	success=1
experience		
version2	244505	12779
version1	243807	12772

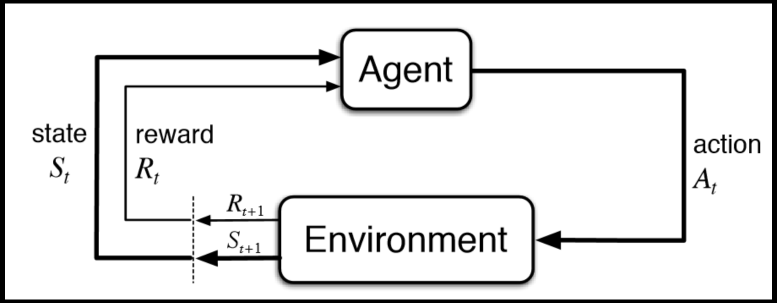
Missing value ratio:
('20.99%', 'days_since_last_visit'),
('2.33%', 'year_home_built'),
('2.33%', 'net_worth'),
('2.33%', 'montrd_home_security_sys_own_value'),
('2.33%', 'mkt_trend_env_focused_hh_value'),
('2.33%', 'mkt_organic_product_purchasers_value'),
('2.33%', 'mkt_green_product_purchasers_value'),
('2.33%', 'length_of_residence'),
('2.33%', 'income'),
('2.33%', 'home_market_value'),
('2.33%', 'high_end_shoppers_value'),
('2.33%', 'do_it_yourselfer_value'),
('0.15%', 'zipcode'),
('0.0%', 'visit_id'),
('0.0%', 'success'),
('0.0%', 'repeat_visit'),
('0.0%', 'pro'),
('0.0%', 'new_movers'),
('0.0%', 'experience'),
('0.0%', 'date_time')

Click here for data_profiling_report.html

Data Exploration (continued)



Reinforcement learning introduction



Markov decision process

A Markov decision process is a 4-tuple (S, A, P_a, R_a), where:

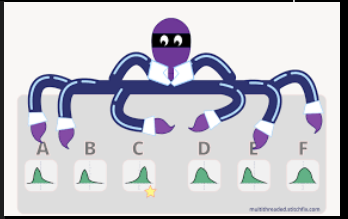
S is a set of states called the state space,

A is a set of actions called the action space (alternatively, A_s is the set of actions available from state s),

$P_a(s, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$ is the probability that action a in state s at time t will lead to state s' at time $t+1$,

$R_a(s, s') = R(s_{t+1} = s' \mid s_t = s, a_t = a)$ is the immediate reward received after transitioning from state s to state s', due to action a.

Multi-armed bandit



A multi-armed bandit is a 2-tuple (A, R_a), where:

A is a set of actions called the action space (alternatively, A_t is the set of actions available at time step t),

$R_a = R(a_t = a)$ is the immediate reward received after taking action a at time step t.

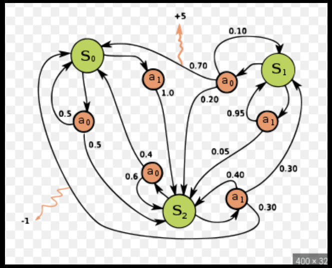
Contextual multi-armed bandit

A contextual multi-armed bandit is a 3-tuple (S, A, R_a), where:

S is a set of states called the state space (namely, the context for actions),

A is a set of actions called the action space (alternatively, A_t is the set of actions available at time step t),

$R_a(s) = R(s_t = s, a_t = a)$ is the immediate reward received after taking action a at time step t when the state is s.



How to evaluate a policy?

- Deterministic policy
- Maximize the total number of success of n days
- e.g., policy1: always choosing version1, leads to 12,772 success in given the dataset.
- e.g., policy2: always choosing version2 leads to 12,779 success.

- Stochastic policy
- Maximize the expectation of total number of success of n days (and minimize the variance of it)

Multi armed bandit (deterministic, greedy action selection)

- Time step: daily (we are free to choose other time granularity: e.g., weekly, hourly, etc.).
- Policy3: choose an action (version1 or version2) which had the higher success rate yesterday.

$$Q_t(a) = \frac{\text{number of success when `a` taken at t-1}}{\text{number of times `a` taken at t-1}}$$

$$A_t = \underset{a}{argmax}\, Q_t(a)$$

- Result: 13,436 success in given the dataset (5.1% better than policy2).

Multi armed bandit (stochastic, ε-greedy action selection)

- Policy4: the same as policy3, except that:

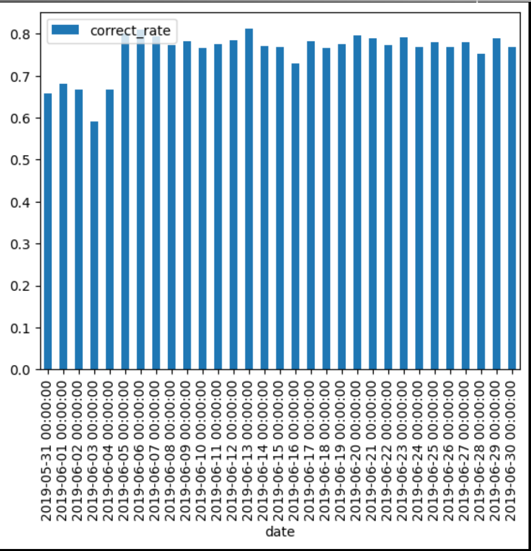
$$A_t = \begin{cases} \underset{a}{argmax}\, Q_t(a) & \text{with probability } 1-\epsilon \\ \text{a random action} & \text{with probability } \epsilon \end{cases}$$

- Result: mean: 13,427, std: 36, ε=0.01.

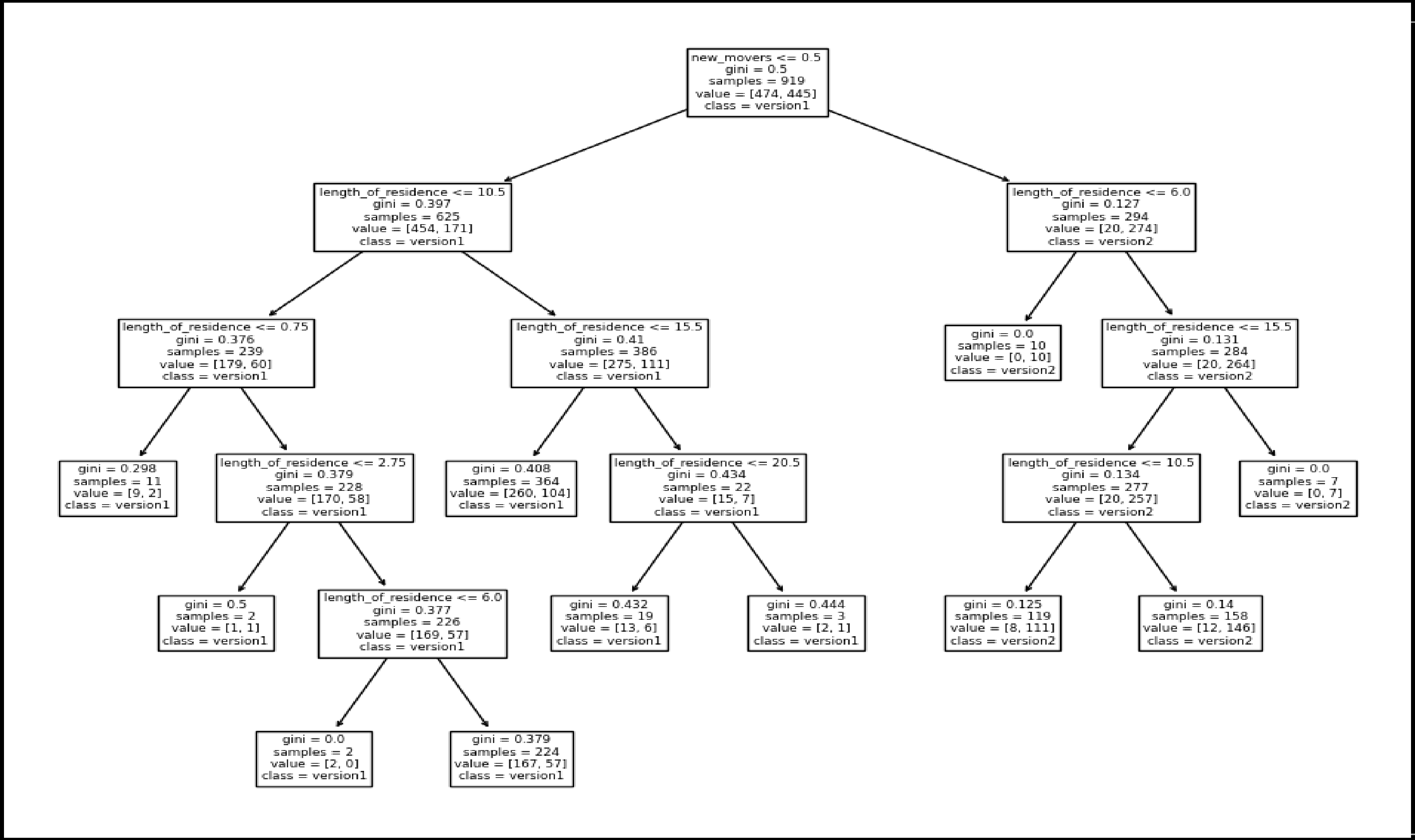
Contextual multi armed bandit (deterministic)

- Policy: a model (e.g., a decision tree, or a logistic regression model) is used to predict the action based on state. At each time step, the model is trained using data of previous time step. At the first time step, the actions are randomly chosen from action set with equal probability.
- Result: 19,496 success using decision tree, 19,596 success (53.3% better than policy2) using logistic regression.

- information gain ratio:
  - (0.3012, 'length_of_residence'),
  - (0.2896, 'new_movers'),
  - (0.0683, 'visit_id'),
  - (0.0584, 'date_time'),
  - (0.0541, 'zipcode'),
  - (0.0459, 'year_home_built'),
  - (0.038, 'montrd_home_security_sys_own_
  - (0.0347, 'mkt_green_product_purchasers_
  - (0.0294, 'days_since_last_visit'),
  - (0.0222, 'mkt_organic_product_purchaser
  - (0.0176, 'mkt_trend_env_focused_hh_valu
  - (0.0168, 'home_market_value'),
  - (0.0157, 'income'),
  - (0.0107, 'high_end_shoppers_value'),
  - (0.0071, 'net_worth'),
  - (0.0032, 'do_it_yourselfer_value'),
  - (0.0005, 'pro'),
  - (0.0002, 'repeat_visit')



An example of the decision tree

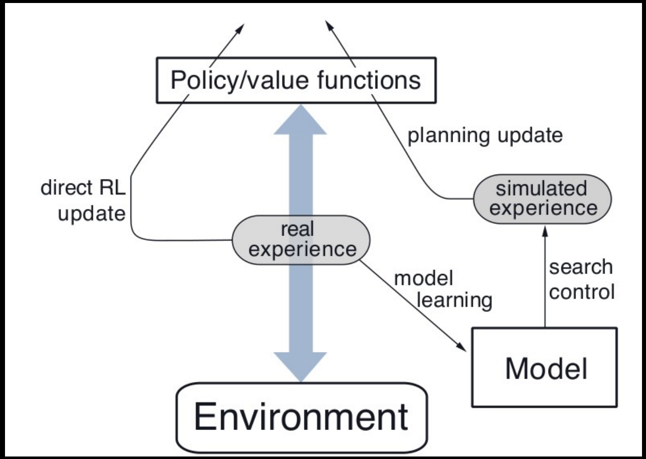


A policy derived from decision tree

- Policy: choose version1 if new_comer<=0.5 else version2
- Result: 19,451 success (compared to 19,496 for the decision tree based model)

Future work

- How to make use of unsuccessful data points?
  - Dyna-Q



The end

- Thanks for your attention, I am open to questions.