Generalization of Thompson Sampling for Multiple Categorical and Numerical Variables with Application for Fraud Detection

Alex Zolotovitski
Bellevue, WA 98005, www.linkedin.com/in/alexzolot

Abstract
Thompson Sampling is a well-known effective algorithm of reinforcement learning in cases when the probability of reward depends on one categorical variable. Using a combination of unsupervised and supervised learning methods, we generalized the algorithm for the case when the reward depends on multiple categorical and numerical variables, tuned it with a simulation, and applied it to a fraud detection audit.

The method demonstrated good cumulative gain: checking 50% of candidate cases selected by the algorithm we could detect 96% of fraud cases (96% true positive rate) having 99% of related monetary loss (maximum possible reward).

Key Words: reinforcement learning, Thompson Sampling, fraud detection

1. Business task

The task of fraud detection is to predict either a customer is a fraudster or not. And specific of the application is that labeling variable y∈{null – not tested, 0 – not fraud, 1 – fraud, NA - skip} is done by humans – auditors or investigators – and is very time consuming and expensive in comparison to marketing tasks.
The objective is to test minimal number of subjects (ii, id, AN -Account Number) , to find maximum number or dollar amount of fraud.

1.1 Data

The data are similar to other supervised learning applications:

Table 1: Typical data (top of a table)

id(AN)	class	x1	x2	cost(AR,$)	y
002	A	7.10	2.10	649.90	1
003	A	7.70	2.30	850.50	
006	A	5.80	1.90	725.90	
008	A	6.40	2.30	386.60	
009	A	6.50	2.00	357.80	
001	B	4.80	0.20	494.00	0

We have an *Id* which is an account number (AN), a categorical variable "*class*", two numerical variables *x1* and *x2*, characterizing a customer, a dollar amount, e.g. balance of account receivable (*AR*), related to a lost if this customer is a fraudster, and variable *y* – the label, a result of the investigation. At this stage of the audit we investigated two customers with *Id= 002* and *001*, who happened to be a fraudster (*002*) and a not-fraudster (*001*).

Based on the data we need to figure out which Id should be tested next. This is a typical task of reinforcement learning.

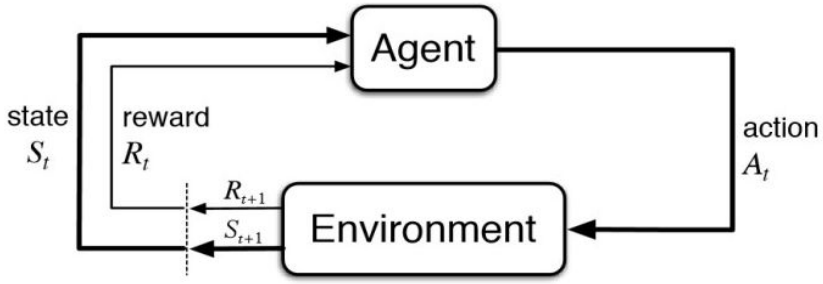
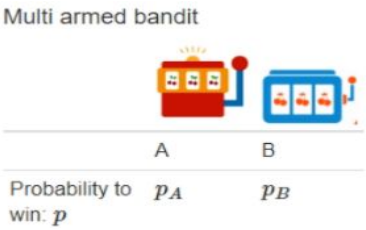


Figure 1: Scheme of reinforcement learning.

In this situation *Agent* is our application, *Environment* is our investigator, *state* is what pool is available and results of previous testings, *action* is which Id should be tested the next, and *reward* depends on results of investigation – is it fraud or not.

The simplest task of this type is a "multi armed bandit". Let we have two slot machines, A (red) and B (blue)



If we pull handle of machine A, we get reward $r_A = \$10$ with probability p_A, and if we pull handle of machine B, we get reward $r_B = \$20$ with probability p_B. We don't know the probabilities, and need to estimate them as a result of the testing, as well as which handle to pull next. The Tompson sampling is a popular algorithm for this task.

In any moment of game the state is characterized by number of times we pulled handle of each machine and number of wins n_1 and losts n_0 with each machine. Our objective is to get maximum reward. In the beginning of game we have no information about the probabilities:

		A	B
Count lost (no fraud)	N_0	0	0
Count win (fraud)	N_1	0	0

Let after 7 trials, 3 times A and 4 times B, we came to the state

		A	B
Count lost (no fraud)	N_0	2	1
Count win (fraud)	N_1	1	3

Thompson sampling is a popular way to solve exploration vs exploitation trade-off.

Step of the algorithms

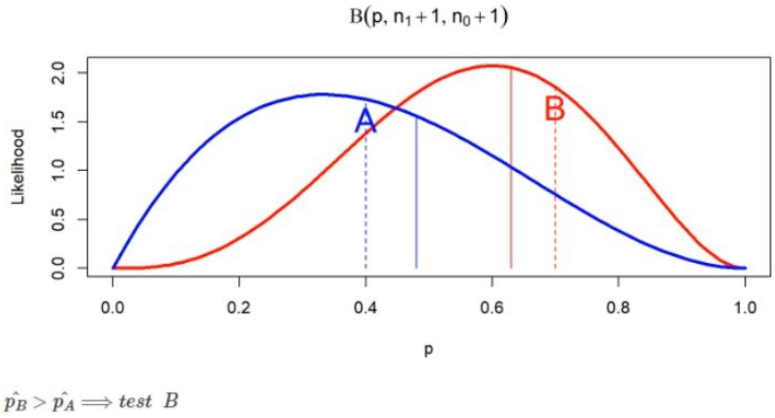
test B

Exploration vs. Expoitation trade-off

test 50:50

arm	n0	n1	p	p_hat ~ B(n1+1, n0+1)
A	2	1	0.40	0.48
B	1	3	0.70	0.63

Pure exploration means that we continue to pull equally handles of machines A and B. Pure exploitation means that we continue to pull only handle of more successful machine B. In Thompson sampling we pull two random numbers x_A , x_B from posterior distribution of probabilities to win on machines A and B, that are Beta functions with parameters $\alpha = n_1 + 1$, $\beta = n_0 + 1$, and choose the machine providing the larger reward:



$\hat{p_B} > \hat{p_A} \Longrightarrow test\ B$

Dashed lines on these plots show unknown actual probabilities to win on machines A and B, solid lines indicate x_A , x_B .

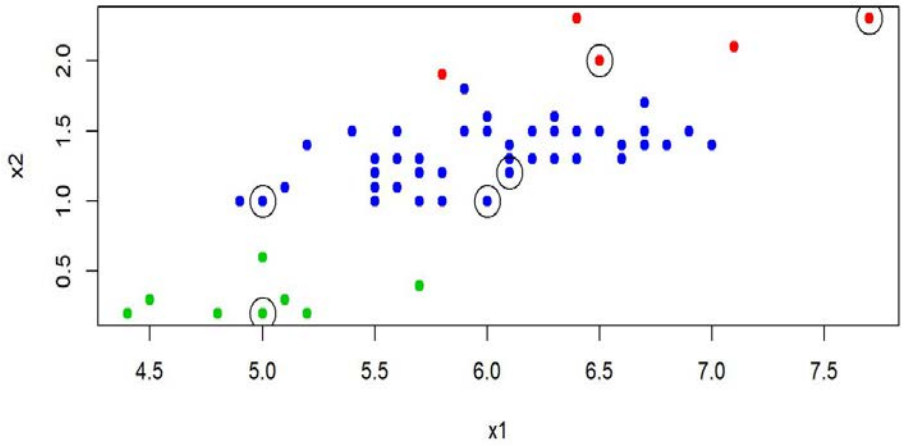
2. Generalization

The simplest more general case is if we have a number n_A of identical machines A, and a number n_B of identical machines B

In this case instead of the simple Thompson sampling we need to have two stages: we need 1) Draw a **presample** from classes - take a random representative one machine of type A and one machine of type B, and after that 2) to repeat the standard Thompson sampling between the two representatives in the presample. Instead of the first step we can take a presample of two from all machines, taking each machine with probability $\sim 1 / n_A$, $1 / n_B$ that would be asymptotically equivalent to taking one machine of each type.

2.1 Numerical variables

In the case of numerical variables we can use clustering and split all observations to clusters. Suppose that the clusters are well defined, so probabilities of fraud p and rewards r are almost the same inside the clusters and could be different between the clusters . In this case instead of numerical variables $\{x\}$ we can use a categorical variable *cluster_Id* to describe the observations, then we get the previous case of a number of observations of each type, where instead of n_A and n_B we will have a number of observations in each cluster.



Again, instead of the first step we can take a presample of n_C *(number of clusters)* from all machines, taking each machine with probability ~ $1 / n_i$ ~ $1 / dens_i$ (n_i - number of observations in the corresponding cluster, $dens_i$ - density of observations in the vicinity of observation i) that would be asymptotically equivalent to taking one machine of each type. If instead of one observation per cluster we include in the preset more observations per cluster, we shift the exploration-exploitation balance in direction of more exploitation, if we take in the presample less than one observation per cluster, we shift the exploration-exploitation balance in direction of more exploration.

3. Implementation

Our implementation of the algorithm has following features:

1. It supports multiple users (auditors) working simultaneously ,
2. It it has small latency (~ 0.2 sec) and
3. It is stochastic, so any Id has a chance to be investigated

Initially all Id = i = Account Number (AN) are in the pool, that they can be tested and results y of the testing are reported.

3.1. Steps of the Modified Thompson Sampling Algorithm

Prepare Data

1. Evaluate density of observations in the pool using kernal density estimation

$$\text{density} \quad d_i = \sum_{j \in pool} K(\vec{x_i} - \vec{x_j}) \equiv \sum K(\vec{x_i} - \vec{x_j}) \;\; \theta(j \in pool)) \;\; , \; \forall i \in pool \;\; ,$$

$$\text{where K is a Kernel:} \quad K(\vec{r}) = exp(-\frac{|\vec{r}|}{r_0}) \;\; ,$$

Optimal value of radius r_0 should be about the average radius of the clusters.
Smaller value r_0 would increase exploitation, larger value r_0 would increase exploration.

2. Evaluate densities d_0 and d_1 for found not-fraud and fraud:

$$d_{0,i} = \sum_{j \in report, y_j = 0} K(\overrightarrow{x_i} - \overrightarrow{x_j}) \equiv \sum K(\overrightarrow{x_i} - \overrightarrow{x_j}) \cdot \theta(j \in report) \cdot \theta(y_j = 0)) \ , \ \forall i \in pool \ ,$$

$$d_{1,i} = \sum K(\overrightarrow{x_i} - \overrightarrow{x_j}) \cdot \theta(j \in report) \cdot \theta(y_j = 1)) \ , \ \forall i \in pool \ .$$

3. Create uniformly distributed random variable $u_i \sim U(0,1)$ for each observation in the pool and order observations in the `pool` by u_i / d_i .

At test on the *`nextAN`* tester's request

1. Take top n_{pre} observations from the `pool` into `presample`. Smaller value n_{pre} would increase exploration, larger value n_{pre} would increase exploitation.
2. Create variables n_0 and n_1 by normalizing densities d_0 and d_1 to total numbers of observed not-fraud and fraud:

$$n_0 = N_0 * d_0 / \sum d_0 \ , \ n_1 = N_1 * d_1 / \sum d_1$$

3. Sample AN from the `presample` using modified Thompson sampler, using table `report` – create stochastic "reward" variable *`rew`* from Beta distı $rew \sim AR * \mathrm{B}(n_1 + 1, n_0 + 1)$

and choose observation *i* with maximum reward i = argmax(*rew*).

4. Send this observation to the tester
5. Exclude the observation from the pool.`

At test on tester's request *`doneAN`*

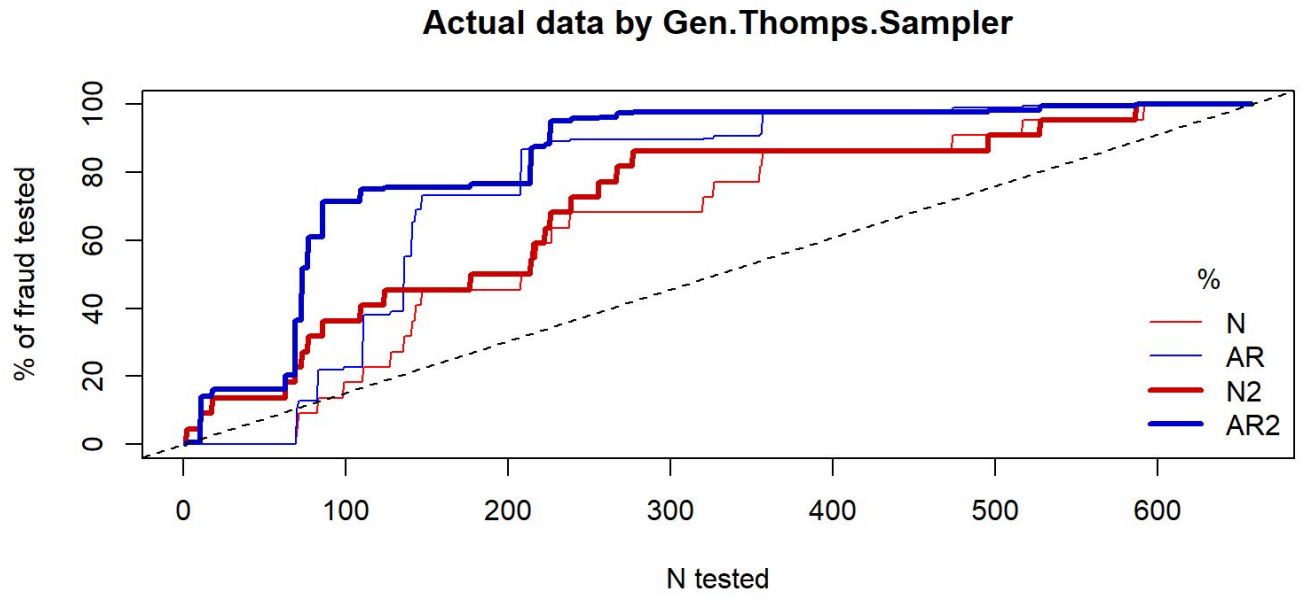
On the tester's request *`doneBAN`* providing $y_j = y \in \{0, 1, NA\}$

1. Repeat steps 1-3 from "Prepare Data" = reevaluate densities d, d_0 and d_1 with new `report` and `pool`:

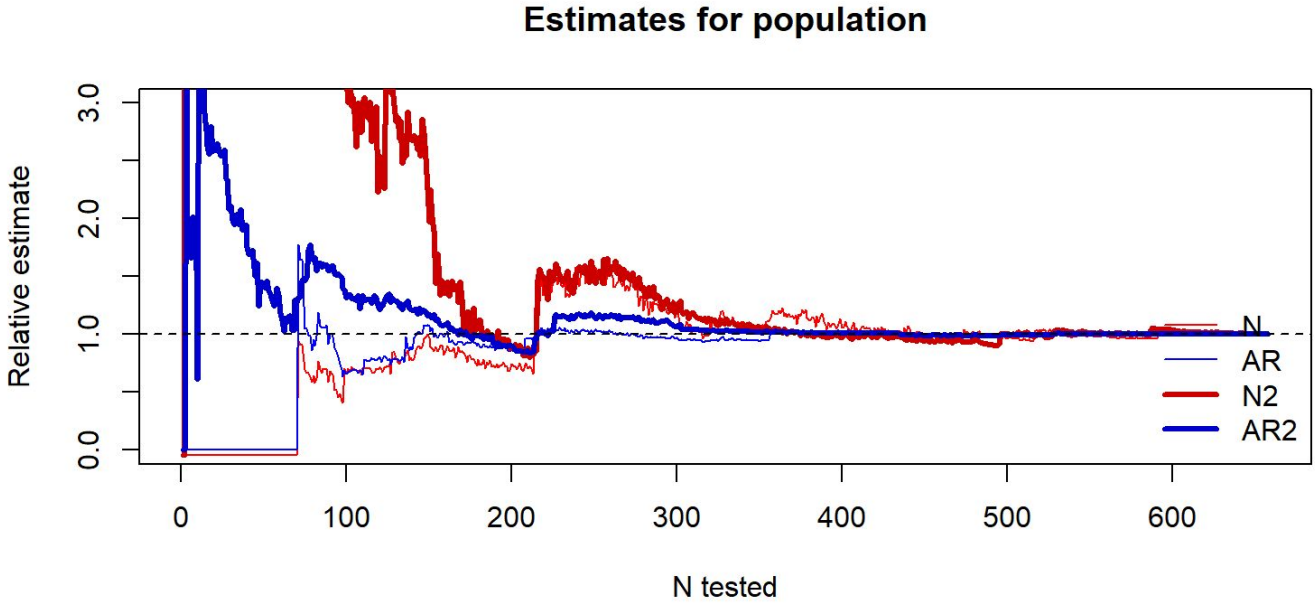
$$d_{y_j,i} \ += \ K(x_i - x_j) \ , \ y_j \in \{0,1\}$$

4. Retro Thompson on actual data

This is the result of using the generalized Thompson algorithm on actual testing data. 660 subjects were tested.

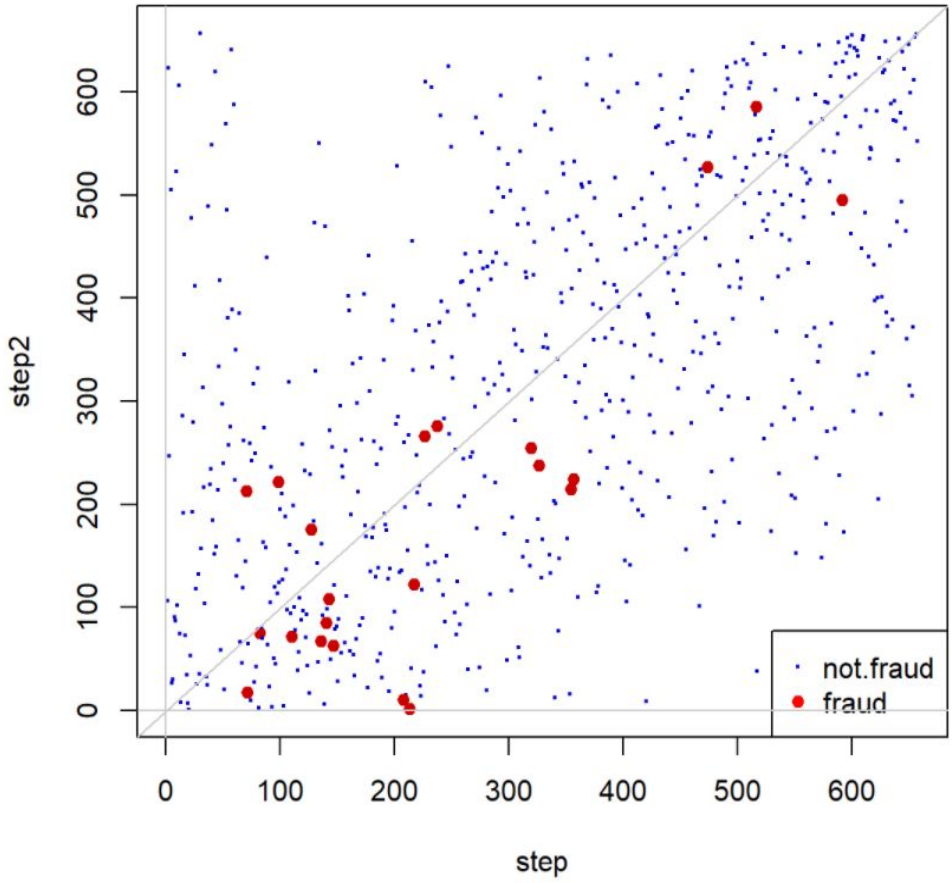


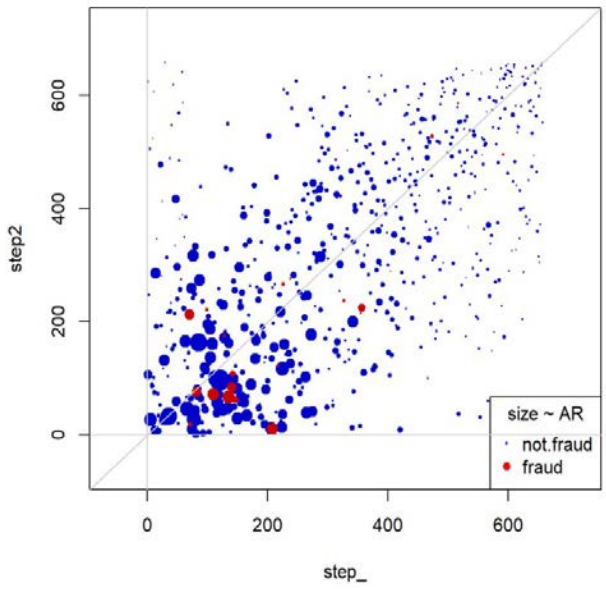
In the above plot the red lines show the number of found fraud using two variants of the algorithm in selecting the observations before we found the first fraud case, and blue lines – related dollar amount revenue/loss. We see that after testing of half of subjects we identify about 90% of all fraud cases and even higher percent of the related dollar amount.



This chart shows estimation of fraud rate of overall population. Again, we see that after testing half of subjects we are able provide a good estimate of the fraud rates for the population.

The following plots show correspondence between steps on which observations were tested in the two modifications of the algorithm.





We used two variants of warming up: from larger clusters (thin lines) and from the last clusterized - outliers (thick lines, N2 and AR2). In both variants we found almost all fraud after testing half of subjects.

Conclusion

The method demonstrated good cumulative gain: checking 50% of candidate cases selected by the algorithm we could detect 96% of fraud cases (96% true positive rate) having 99% of related monetary loss (maximum possible reward).

We see that the generalized Thompson Sampling after testing half of all observations selects for testing almost all fraud cases and has smaller absolute error of estimation of fraud in not-tested cases, but higher absolute error of the estimates.

References

1. 1. Multi-Armed Bandit With Thompson Sampling. George Pipis, https://predictivehacks.com/multi-armed-bandit-with-thompson-sampling
2. RFCDE: Random Forests for Conditional Density Estimation, Taylor Pospisil and Ann B. Lee, arXiv:1804.05753v2 /stat.ML/May 2018, https://github.com/tpospisi/RFCDE/tree/master/r/vignettes
3. Decision Forests for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning, A. Criminisi1 , J. Shotton2 and E. Konukoglu3, Microsoft Research technical report TR-2011-114
4. Bayesian Control Rule, Pedro A. Ortega http://www.adaptiveagents.org/_media/bayesiancontrolrule.pdf