IBM 3rd Problem

Xiangyu Zhang

December 22 2017

1 Introduction

We are concerned with how to price highly valued IT service contracts in this project for IBM, and in particular how to model the feature of market penetration.

For some clients that have a good prospect to be longer term business partners, the provider IBM might consider giving them a lower price (that could result in low or no profitability) on an initial deal since this may attract the client to do business in the future. On the other hand, if the client is one that the provider has often done business with before, then the provider can typically charge a premium, since he has a high chance of winning the deal with that client (since it is expensive for the client to switch to a different IT provider once it has purchased significant software and hardware from the first provider).

We want to capture the value of market penetration in our model, creating pricing strategies that attract clients at the beginning, and then charge a premium in the future.

2 Prediction Model

A highly accurate classifier has already been built that predicts whether the deal would be won or not given the initial price and other deal attributes. It is interesting that price is not the only nor the main factor in determining winnability.

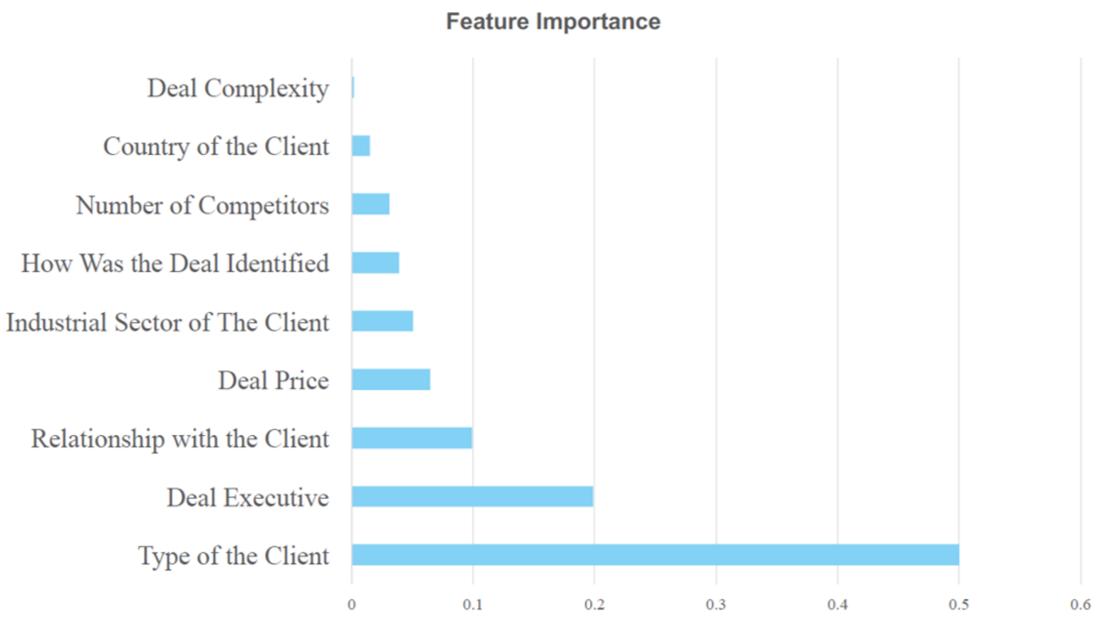
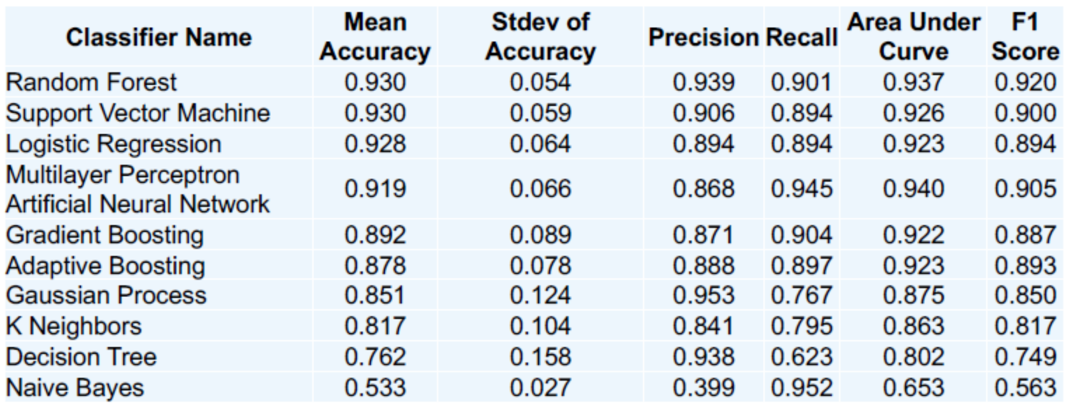


Figure 1: Feature Difference

The prediction model has high accuracy. For example, the Random Forest classifier has 0.930 mean accuracy with 0.0504 standard deviation.



Classifier Name	Mean Accuracy	Stdev of Accuracy	Precision	Recall	Area Under Curve	F1 Score
Random Forest	0.930	0.054	0.939	0.901	0.937	0.920
Support Vector Machine	0.930	0.059	0.906	0.894	0.926	0.900
Logistic Regression	0.928	0.064	0.894	0.894	0.923	0.894
Multilayer Perceptron Artificial Neural Network	0.919	0.066	0.868	0.945	0.940	0.905
Gradient Boosting	0.892	0.089	0.871	0.904	0.922	0.887
Adaptive Boosting	0.878	0.078	0.888	0.897	0.923	0.893
Gaussian Process	0.851	0.124	0.953	0.767	0.875	0.850
K Neighbors	0.817	0.104	0.841	0.795	0.863	0.817
Decision Tree	0.762	0.158	0.938	0.623	0.802	0.749
Naive Bayes	0.533	0.027	0.399	0.952	0.653	0.563

Figure 2: Classifier Comparison

3 MDP model

We formulate the pricing problem as a Markov Decision Process(MDP), in which the state variable contains attributes of the deal, and the control variable is the price we charge.

To understand the cone difficulty of the pricing problem, we formulate the basic model first and then introduce some extensions to the basic model. If we can find the optimal strategy for the basic model, it should not be hard to extend our results to a more general one.

3.1 Basic Model

We assume the client comes to IBM at each time period $t = 1, 2, 3, ...$ and tells IBM their requirements. The deal attributes vector x_t consists of some features in the prediction model known at the beginning at each time period t such as the type of client, deal executive, relationship with the client, etc. After IBM receives the request from client, we first calculate the cost for providing such service c_t based on the deal attributes x_t and then charge service price p_t. The client then will decide whether IBM wins the deal based on the deal attributes x_t and price p_t. Whether IBM wins the deal is described by a Bernoulli random variable w_t, whose distribution is determined by x_t and p_t. As time goes by, some deal attributes will change. For example, the relationship with the client will increase or decrease based on the client's deal history.

Mathematically, we want to maximize

$$\mathbb{E}\sum_{t=1}^{\infty}\gamma^t(p_t(x_t) - c_t(x_t))w_t$$

which γ is the discount factor. To find the optimal pricing policy for the above MDP, we need to estimate two kinds of conditional distributions: whether w_t equals 1(IBM wins the deal) conditioned on p_t and x_t,

$$w_t|x_t, p_t$$

and the next stage deal attributes x_{t+1} based on x_t, p_t and w_t

$$x_{t+1}|x_t, p_t, w_t.$$

Using the prediction model, we can estimate $w_t|x_t, p_t$ with high accuracy so we only need to estimate the conditional distribution $x_{t+1}|x_t, p_t, w_t$ based on historical data. Some attributes of the deal do not change through time, for example the country of client, so the attributes whose conditional distribution we need to estimate are of low dimension.

Remark. The dimensionality of the state space determines whether it is practically feasible to solve this MDP computationally using standard methods such as value iteration. As described, the state space has dimension of at most $6 + K$: 6 is the number of features in the prediction model, leaving out price (which is a control variable), and country of the client and industrial sector (which do not change over time); and K is the number of variables we need additionally to calculate the cost of the deal c_t. Solving a dynamic program with 6 or more dimensions is typically not possible without approximate methods, but if the 6 features described above arise independently over time, or according to some lower-dimensional sufficient statistic, then it may be possible to reduce the dimensionality further making a computational solution feasible.

3.2 Model Extension

There are several extensions to the basic MDP model. We introduce two of them in this subsection.

3.2.1 Random Arrival Extension

In this model we assume that the inter-arrival time A_n of the client is i.i.d integer random variable sampled from distribution F. The n^{th} time T_n that the client comes to IBM is given by

$$T_n = \sum_{t=1}^{n} A_t$$

Similarly the n^{th} deal's attributes, cost, price and whether IBM wins the deal are given by x_n, c_n, p_n and w_n. Mathematically we want to maximize

$$\mathbb{E} \sum_{n=1}^{\infty} \gamma^{T_n} (p_n(x_n) - c_n(x_n)) w_n$$

If we can estimate $x_{t+1}|x_t, p_t, w_t$ accurately in the basic model, we additionally only need to estimate the distribution of F to solve the random arrival model.

3.2.2 Learning Customer Style Extension

Perhaps the clients coming to IBM are of different types. For example some clients are the "buying-lots-of-stuff" type and some are the "buying-as-little-as-I-can" type. We can use an additional parameter θ to capture these differences. Extending the basic model, we want to maximize

$$\mathbb{E} \sum_{t=1}^{\infty} \gamma^{t} (p_t(x_t, \theta) - c_t(x_t, \theta)) w_t$$

This type θ may be unknown initially, and drawn from a prior probability distribution. By observing the customer's behavior over time we may learn θ.

4 Discussion Review

In this section we formulate the MDP model with the random inter-arrival time along with some other assumptions which make our MDP model computationally tractable.

4.1 Framework

We assume that the inter-arrival time A_n of the client is i.i.d integer random variable sampled from some distribution F. The n^{th} time T_n when the client comes to IBM is given by

$$T_n = \sum_{t=1}^{n} A_t$$

When the customer comes to IBM at the n^{th} time, it will tell IBM its IT service requirement r_n. Based on the requirement r_n, IBM can calculate the cost $c(r_n)$ for providing such service. At the same time, the deal attribute vector d_n is also known to IBM, which will affect whether IBM wins the deal, and d_n consists of 6 components, type of client, deal executive, relationship with the client, deal identification, number of competitors and deal complexity. We use $x_n = (r_n, d_n)$ to denote the state variable of our MDP model. Based on x_n IBM charges price $p_n(x_n)$ for the IT service and whether IBM wins the deal is described by a Bernoulli random variable w_n, whose distribution is determined by x_n and p_n. Suppose we already know the conditional distributions

$$w_n | x_n, p_n$$

$$x_{n+1} | x_n, p_n, w_n.$$

and we want to maximize

$$\mathbb{E} \sum_{n=1}^{\infty} \gamma^{T_n} (p_n(x_n) - c_n(r_n)) w_n$$

with the discount factor γ.

4.2 Assumption

To make our MDP computationally tractable, we propose the following assumptions.

1. The inter-arrival time A_n is i.i.d and independent with all requirements r_m and deal attributes d_m for $m = 1, 2, 3, ...$

2. The distribution of $(n+1)^{th}$ requirement r_{n+1} only depends on the past requirement r_m for $m = 1, 2, 3, ..., n$ and is irrelevant of the past deal attributes d_m and prices p_m.

3. The components of n^{th} deal attribute, except the relationship with the client, are i.i.d random variables and independent with all earlier requirements r_m and deal attributes d_m. For the n^{th} deal the distribution of the relationship with the client only depends on the relationship of the $(n-1)^{th}$ deal and whether IBM wins the contract w_{n-1}.

Shortcoming: requirement r_n is not Markovian but depends on history r_m for $m = 1, 2, ..., n-1$.

5 New Model

In this section a new model is presented to handle the non-Markovian property of customer requirement. A basic observation is that the cost c_n rather than the requirement r_n is of interest. And it is a reasonable assumption that the cost c_n is markovian even i.i.d. We make this only change so the new model is very similar with the original one. The differences between these two model are written in red in the following context.

For implementation convenience, we assume that the cost c_n is i.i.d sampled in the following context. But how to model c_n as a Markov Chain is definitely a good investigation direction in the future.

5.1 Framework

We assume that the inter-arrival time A_n of the client is i.i.d integer random variable sampled from some distribution F. The n^{th} time T_n when the client comes to IBM is given by

$$T_n = \sum_{t=1}^{n} A_t$$

When the customer comes to IBM at the n^{th} time , it will tell IBM its IT service requirement r_n. Based on the requirement r_n, IBM can calculate the cost $c(r_n)$ for providing such service. At the same time, deal attribute vector d_n is also known to IBM, which will affect whether IBM wins the deal, and d_n consists of 6 components, type of client, deal executive, relationship with the client, deal identification, number of competitors and deal complexity. We use $x_n = (c_n, d_n)$ to denote the state variable of our MDP model. Based on x_n IBM charges price $p_n(x_n)$ for the IT service and whether IBM wins the deal is described by a Bernoulli random variable w_n, whose distribution is determined by x_n and p_n. Suppose we already know the conditional distributions

$$w_n | x_n, p_n$$

$$x_{n+1} | x_n, p_n, w_n.$$

and we want to maximize

$$\mathbb{E} \sum_{n=1}^{\infty} \gamma^{T_n} (p_n(x_n) - c_n) w_n$$

with the discount factor γ.

5.2 Assumption

To make our MDP computationally tractable, we propose the following assumptions.

1. The inter-arrival time A_n is i.i.d and independent with all costs c_m and deal attributes d_m for $m = 1, 2, 3, ...$

2. The cost c_n is i.i.d r.v and independent with deal attributes d_m for $m = 1, 2, 3, ...$

3. The components of n^{th} deal attribute, except the relationship with the client, are i.i.d random variables and independent with all earlier costs c_m and deal attributes d_m. For the n^{th} deal the distribution of the relationship with the client only depends on the relationship on $(n-1)^{th}$ deal and whether IBM wins the contract w_{n-1}.

5.3 Bellman Equation

For implementation convenience, we introduce some new notations for the MDP model. The state variable x_n has 6 components, which are rc(Relationship with the Client), de(Deal Executive), di(How Was the Deal Identified), nc(Number of Competitors), dc(Deal Complexity) and co(Deal Cost). All other components are i.i.d r.v. except rc(Relationship with the Client), so we use an additional notation θ to denote the vector of i.i.d component. Saying

$$x_n = (r_n, \theta_n) = (r_n, de_n, di_n, nc_n, dc_n, co_n)$$

In the following context we use r_n for the client relationship on the n^{th} deal and $C(x_n, p_n, w_n)$ for utility function

$$C(x_n, p_n, w_n) = (p_n(x_n) - co_n)w_n$$

The discount factor for this MDP is not γ anymore, but $\mathbb{E}\gamma^{A_0}$ because of i.i.d property of A_n. To avoid introducing a new notation, we will still use γ to denote the discount factor.

We can easily write down the Bellman equation for the MDP as

$$V(x) = max_p\mathbb{E}[C(x, p, w) + \gamma\mathbb{E}[V(x_{new})|x, p]]$$

Because $x = (r, \theta)$ and θ is i.i.d vector, we can define value function for r as

$$V(r) = \mathbb{E}_\theta V(r, \theta)$$

and then we can rewrite the Bellman equation as

$$V(r) = \mathbb{E}_\theta[max_p(c(r, \theta, p) + \gamma\mathbb{E}[V(r_{new})|r, \theta, p])]$$

which

$$c(r, \theta, p) = \mathbb{E}_w C(r, \theta, p, w).$$

We can also define Bellman mapping for value function $V(r)$ $T : V \to TV$

$$TV(r) = \mathbb{E}_\theta[max_p(c(r, \theta, p) + \gamma\mathbb{E}[V(r_{new})|r, \theta, p])].$$

The Bellman mapping is very useful and we will implement a toy model using contraction property of Bellman mapping.

6 Toy Model

In this section we will introduce the toy model implemented in detail. To describe the MDP model, we need three things, distributions of i.i.d components, prediction model and state transition matrix. We will introduce each one thing in one subsection.

6.1 The i.i.d Components

There are 5 i.i.d components, de(Deal Executive), di(How Was the Deal Identified), nc(Number of Competitors), dc(Deal Complexity) and co(Deal Cost). We assume $de, di, ncanddc$ can only take discrete value, saying $0, 1, 2, ..., U$, and they are distributed uniformly on these values. The higher value they take, the higher probability IBM wins the deal. The variable co(Deal Cost) is log-uniformly distributed on interval $[a, b]$. Mathematically speaking, we assume

$$de, di, nc, dc \sim \text{Discrete Uniform}(U)$$

$$log(co) \sim \text{Uniform}(a, b)$$

We set parameters $(U, a, b) = (4, 3, 7)$ to do implementation.

6.2 Prediction Model

Given the client relationship r, i.i.d components θ and the price p we charge p, the probability of IBM winning the deal can be calculated through the prediction model. For simplicity, we assume the price IBM charges satisfies $co \leq p \leq 2co$. For the client relationship we also assume it can only take discrete values $0, 1, 2, ..., U$ and the higher the value is, the better the relationship with the client is. We use a linear combination of features to model winning probability, saying

$$probability = \frac{0.001\frac{dc}{U} + 0.03\frac{nc}{U} + 0.04\frac{di}{U} + 0.06dp + 0.1\frac{rc}{U} + 0.2\frac{de}{U}}{0.431}$$

which $dp = 2 - \frac{p}{c}$ explains why we require $co \leq p \leq 2co$. The coefficient before each feature is given by feature plot.

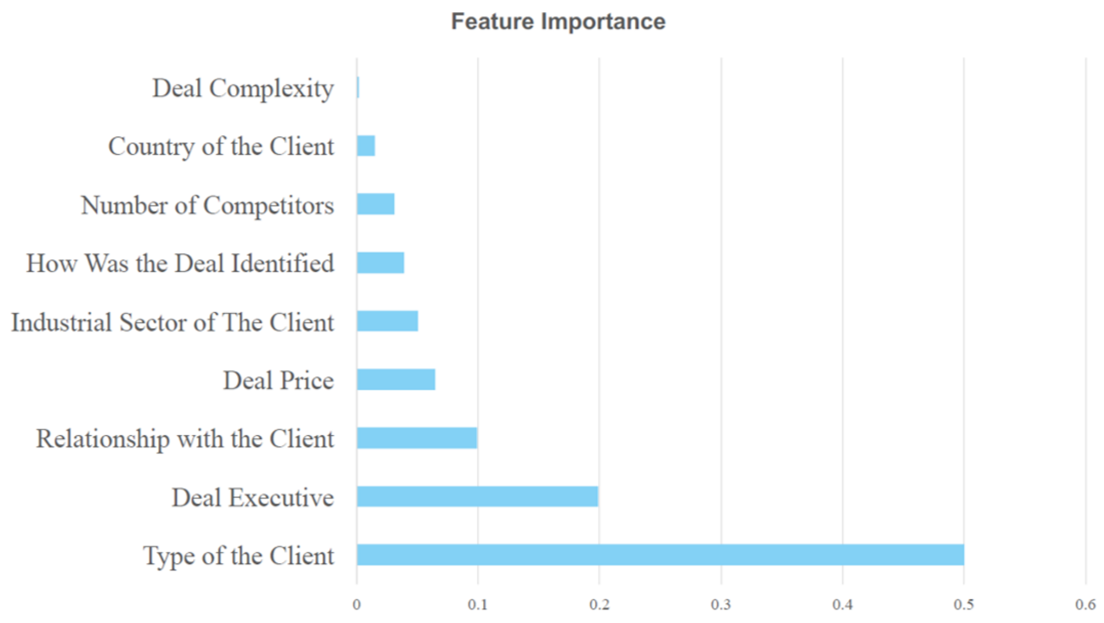


Figure 3: Feature Difference

6.3 State Transition Matrix

Given r_n, θ_n and p_n the client relationship evolves into a new state r_{n+1}. We assume the evolution is Markovian and only depends on the current client relationship r_n and whether IBM wins the deal w_n.

We can use the state transition function $p_w(i, j)$ to describe the evolution of client relationship, saying $p_w(i, j)$ is the probability of transition to j from i after observing the deal outcome w. Actually the transition function can be summarized as two $(U + 1) \times (U + 1)$ matrix corresponding to $w = 1$ and $w = 0$.

When $w = 1$ we set the transition matrix as

$$0.20.80.00.00.00.10.20.70.00.00.00.10.20.70.00.00.00.10.20.70.00.00.00.30.7,$$

otherwise we set the transition matrix as

$$1.00.00.00.00.00.50.50.00.00.00.00.50.50.00.00.00.00.50.50.00.00.00.00.50.5.$$

If IBM wins the deal, it has chance to improve the relationship with client so r_{n+1} becomes bigger than r_n in high probability. If IBM loses the deal and does not do business with the client for a long time, the client relationship will go down and r_{n+1} becomes smaller.

7 Implementation Results

We set the discount factor $\gamma = 0.95$ and implement value iteration method in Python. Suppose now we have an approximate value function $V_n(r)$, to update it we calculate the Bellman mapping $TV_n(r)$ and set $V_{n+1}(r) = TV_n(r)$. As $n \to \infty$, we can prove

$$V_n \to V \quad as \quad n \to \infty.$$

which V is the optimal value function.

Given the client relationship r, for every possible θ we maximize

$$c(r, \theta, p) + \gamma \mathbb{E}[V_n(r_{new})|r, \theta, p]$$

with respect to p. And then we do integration over all θ. For the variable co(Deal Cost), we use Euler Method to calculate this one dimension integration. As for other i.i.d components, we calculate every possible combination and sum them all.

The computational complexity for each iteration is $O((U+1)^5 \times N)$, which N is the number of points we select in interval $[a, b]$. Start from one upper bound and one lower bound for the optimal value function, we do iteration for $M = 200$ times and plot $V_n(r)$ as below. The total $2M = 400$ iterations take Thinkpad X240 1431s to finish all computation(Approximately 3 seconds for 1 iteration).

The lower bound we select is $V_0^L = (0, 0, ..., 0)$ and the upper bound is $V_0^U = (\frac{10^b}{1-\gamma}, \frac{10^b}{1-\gamma}, ..., \frac{10^b}{1-\gamma})$. The final iteration results are given in the following table. As we can see, the upper bound and lower bound are very close to each other.

Value Function Estimation					
	$V(0)$	$V(1)$	$V(2)$	$V(3)$	$V(4)$
Upper Bound	7712550	7992747	8425231	8820368	9028783
Lower Bound	7705539	7985736	8418221	8813357	9021772

8 Structure Result Of Optimal Policy

After we get some accurate estimation for the optimal value function, we can investigate some structure of the optimal policy. In the following context we will use $V(r)$ to denote the optimal value function.

For given client relationship r and i.i.d components θ, we can solve the following optimization problem to get the optimal price p

$$max_p \quad c(r, \theta, p) + \gamma \mathbb{E}[V(r_{new})|r, \theta, p]$$

The object function is actually quadratic with respect to p because $c(r, \theta, p)$ is quadratic w.r.t p and $\gamma \mathbb{E}[V(r_{new})|r, \theta, p]$ is linear w.r.t p. So we can solve optimal p in an exact form. Recall $\theta = (de, di, nc, dc, co)$, numerical results show that

1. The larger r(Client Relationship) is, the higher price p IBM will charge.

This is indeed what we want from this MDP model! For a new client, IBM will charge for lower price to improve the relationship with that client.

2. The larger de(Deal Executive) is, the higher price p IBM will charge.

This phenomena is easy to understand because if IBM knows that it has great chance to win the deal, it will charge for some premium.

3. Components di(Deal Identification), nc(Number of Competitor), dc(Deal Complexity) share the same property with de(Deal Executive).

4. The higher co(Deal Cost) is, the higher ratio p/c IBM charges.

This is not what we want. We want to charge for lower price when r(Client Relationship) is small, no matter how much co(Deal Cost) is.

The reason why this phenomenon happens is that in this MDP model every time IBM wins a deal, customer relationship will transit into larger value in high probability no matter the cost of the deal is small or big. So IBM tends to charge lower ratio p/c for small deals to improve customer relationship and charge higher ration for big deals to make profit. One possible way to fix this problem is to make the client relationship transition depend also on the co(Deal Cost).

9 Future Direction

There are several problems needing further investigations.

1. How to solve the MDP efficiently? The model formulation is very elegant and simple. Even if the state variable x is not of low dimension only 1 dimension, r(Client Relationship), evolves according to a Markov Chain. All the other variables are i.i.d sampled in the beginning of every time period.

The method we use now reduces the storage we need than standard method, but can we design faster algorithm to solve this MDP? Also can we get some non-trivial bound from this elegant structure?

2. How to learn client demand? How to learn the distribution of co(Deal Cost)? In the toy model we assume co(Deal Cost) is sampled from a log-uniform distribution, but we do not know the client's true distribution in reality. On the other hand we can learn(perhaps by Naive Bayesian Method) this distribution as we have done more and more deals with the client.

3. How to model dependence between r(Client Relationship) and co(Deal Cost)? The basic intuition should be that the larger co(Deal Cost) is, if IBM wins the deal, the better client relationship will be in the next period.

4. From operation management point of view, can we compare the price IBM charges by the myopic policy with our optimal policy? The simple intuition is that the price under myopic policy is higher than optimal policy, can we prove that? Furthermore, can we prove some structure results about the price gap under different policies, such as how the gap depends on co(Deal Cost), r(Client relationship), γ(Discount factor),etc.

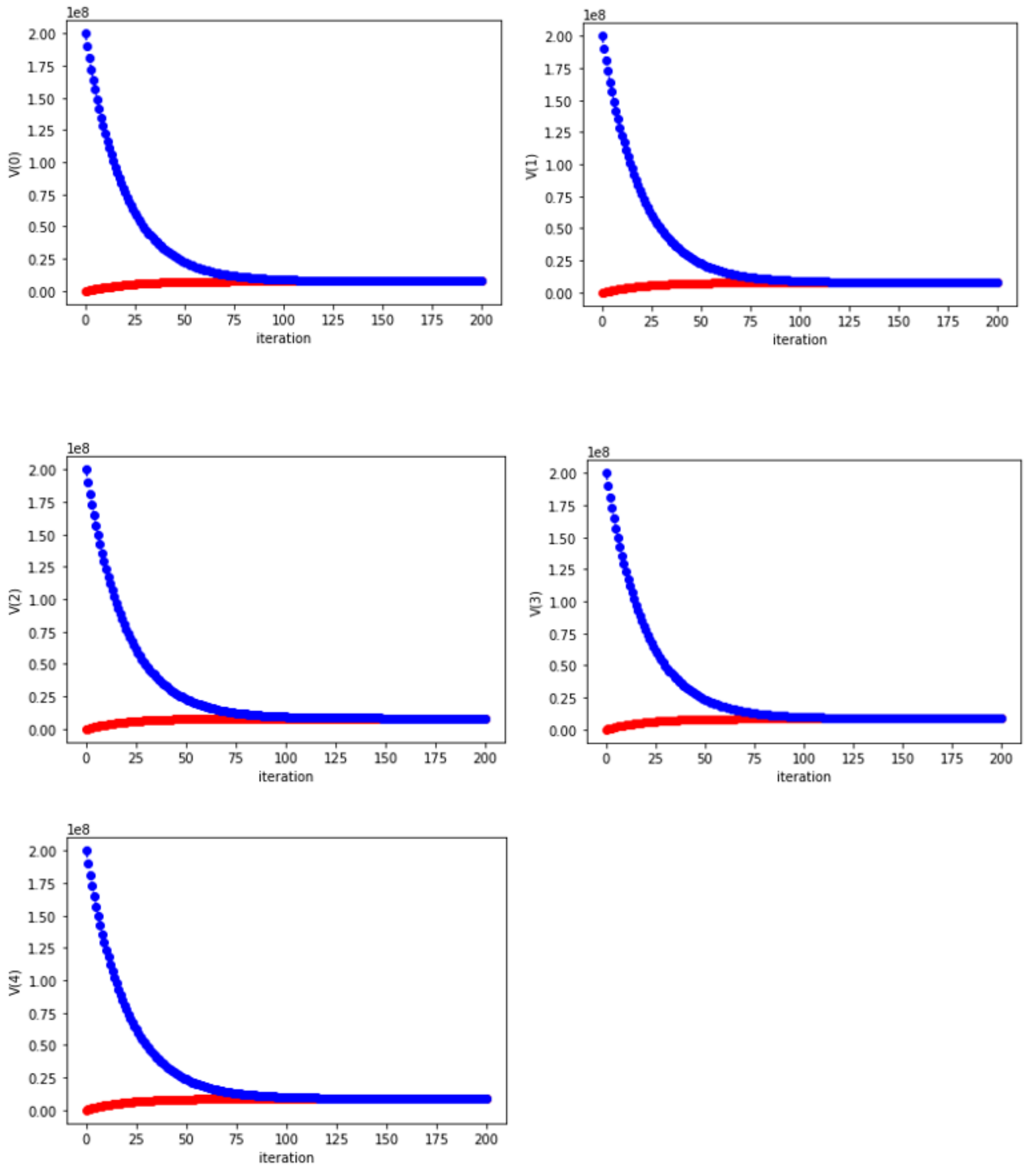


Figure 4: Value Function