A Contest fORged by Amazon Web Services and cORe

Shuotao Diao, sdiao@usc.edu

University of Southern California

1 Introduction

A company can only make replenishment at the start of the month. We assume that the cost of replenishment is 0. If there is some order not satisfied, one can make backorder at cost \$3. At the end of the month, if the inventory is below 90 pieces, the holding cost is \$1 for each piece. If the inventory is above 90 pieces, one needs to pay an extra dollar for each of the additional units. Accordingly, the company should be interested in determining how many pieces that need to be replenished so that the total cost is the lowest. In this report, a two-stage predictive stochastic programming model is created to model the inventory management problem.

The organization of this report is as follows. At first, we will introduce the mathematical formulation of the two-stage inventory management problem. It is worth noting that the objective function in the form of conditional expectation. We refer the two-stage inventory management problem in section 2 as true problem. Second, two methods, including parametric method and non-parametric method, are built to approximate the true problem in section 2. Next, several data analysis methodologies are used to study the nature of demand data provided by Amazon Web Services (AWS). Finally, we provide numerical results and instructions for using our approach.

2 Mathematical Formulation

Decision Variables

y_1	Inventory at the end of the month
x	Number of pieces replenished at the start of the month
s	Number of pieces backordered

We will use upper-case letter to denote random variable while using lower-case letter to denote one realization of the random variable. For instance, Z is

a random variable and z is a realization of the Z.
Parameters and Random Variables

y_0	Inventory at the start of the month
W	Response
Z	Predictor
$d(W)$	Demand in the month

Accordingly, we are interested in determining how many pieces replenished at the start of the month so that the total cost the lowest. Two-stage inventory management problem is formulated below:

$$\min_{x \geq 0} \ \mathbb{E}[F(y_0, x, W)|Z = z], \tag{1}$$

where $F(y_0, x, W)$ is the minimum of the following second-stage problem:

$$F(y_0, x, W) = \min_{y_1, s} \ y_1 + \max(0, y_1 - 90) + 3s \tag{2a}$$

$$\text{s.t. } y_1 = y_0 + x + s - d(W) \tag{2b}$$

$$y_1 \geq 0, s \geq 0 \tag{2c}$$

The reader who is interested in the theory behind problem (1) can refer to Bertsimas and Kallus [2019].

In this paragraph, we will explain the model setup. (2a) means

$$y_1 + \max(0, y_1 - 90) + 3s = \begin{cases} y_1 + 3s & \text{if } y_1 \leq 90 \\ y_1 + (y_1 - 90) + 3s & \text{if } y_1 > 90 \end{cases} \tag{3}$$

That is, if the inventory at the end of the month is not greater than 90, the total cost is the sum of regular holding cost and the backorder cost. If the inventory at the end of the month is the sum of the regular holding cost, extra holding cost for the additional units and the backorder cost. (2b) constrains that the inventory at the end of the month must equal to the sum of inventory at the beginning of the month, replenishment and backorder minus the demand. It is worth noting that in the practical implementation, problem (2) is replaced by the linear programming problem below.

$$F(y_0, x, W) = \min_{y_1, s, u} \ y_1 + u + 3s \tag{4a}$$

$$\text{s.t. } y_1 = y_0 + x + s - d(W) \tag{4b}$$

$$u \geq y_1 - 90 \tag{4c}$$

$$y_1 \geq 0, s \geq 0, u \geq 0 \tag{4d}$$

3 Parametric Model

Deng and Sen [2018] develops parametric model for predictive stochastic programming problem. Let Z_t denote the demand in month t. In this section, an

autoregressive model, which is denoted by AR(p), is used to predict the demand in month t.

$$Z_t = c + \sum_{i=1}^{p} \phi_i Z_{t-i} + \epsilon_t \tag{5}$$

p is the number of order in the model. In this report, we consider that $p = 1, 2, 3$. Let $\hat{Z}_i$ is one copy of Z_t from the AR(p) model. The following model is used to approximate (1).

$$\min_{x \geq 0} \ \frac{1}{N} \sum_{i=1}^{N} F(y_0, x, \hat{Z}_i) \tag{6}$$

At the end of this section, we summarize key steps for running the parametric method in Algorithm 1.

Algorithm 1 Parametric Predictive Stochastic Programming

Step 0 (Initialization): Set $l = 0$, $N_0 > 0$, $\Delta > 0$, choose weight function v, observe predictor z, choose $p \in \{1, 2, 3\}$
Step 1 (Parameter Estimation): Estimate parameters in AR(p) model.
Step 2 (Scenario Generation): Generate N_l scenarios based on AR(p) model. Let S_l denote the data set generated in this step.
Step 3 (Optimization): Solve (6) by feeding S_l. Let X_l denote the optimal solution in this step.
Step 4 (Stopping Criterion): If x_l satisfies the stopping criterion, then terminate. Otherwise, let $N_{l+1} = N_l + \Delta$ and $l = l + 1$, and go to Step 1.

Currently, the stopping criterion is set to "if l is greater than L, then stop." The better way is to introduce statistical tools to determine when the estimated solution is good enough.

4 Non-parametric Model

Without knowing the conditional distribution of W given z, (1) can be approximated by the following non-parametric model.

$$\min_{x \geq 0} \ \sum_{i=1}^{N} v_i(z_1) F(y_0, x, W_i) \tag{7}$$

$v_i(z_1)$ is the weight function which will be calculated by k nearest neighbor method or kernel method. As for the k nearest neighbor method, the weight function is calculated as follows:

$$v_i(z) = \mathbb{I}(Z_i \in S_{k,N}(z)) = \begin{cases} 1 & \text{if } Z_i \text{ is in the } k \text{ nearest neighbor of } z \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

As for the kernel method, the weight function is calculated as follows:

$$v_i(z) = \frac{K(\frac{z-Z_i}{\|h_N\|})}{\sum_{i=1}^{N} K(\frac{z-Z_i}{\|h_N\|})}. \tag{9}$$

We will use Epanechnikov kernel and quartic kernel for $K(\cdot)$.

1. Epanechnikov kernel: $K(z) = (1- \parallel z \parallel^2)\mathbb{I}(\parallel z \parallel \leq 1)$
2. Quartic kernel: $K(z) = (1- \parallel z \parallel^2)^2\mathbb{I}(\parallel z \parallel \leq 1)$

The key steps for running the non-parametric method are summarized in Algorithm 2. We let S_0 denote the original data set.

Algorithm 2 Non-parametric Predictive Stochastic Programming

Step 0 (Initialization): Set $l = 0$, $N_0 > 0$, $\Delta > 0$, choose weight function v, observe predictor z

Step 1 (Data Augmentation): Generate a data set with size N_l from S_0. Let S_l denote the data set generated in this step.

Step 2 (Non-parametric Estimation): Calculate the value of weight function for each sample in S_l

Step 3 (Optimization): Solve (7) by feeding S_l and the corresponding weight functions. Let x_l the denote the optimal solution obtained in this step.

Step 4 (Stopping Criterion): If x_l satisfies the stopping criterion, then terminate. Otherwise, let $N_{l+1} = N_l + \Delta$ and $l = l + 1$, and go to Step 1.

5 Data Analysis

According to figure 1, there may be a trend in the graph. On the other hand, figure 1 shows that there is peak at 60^{th} time period, which is Decemeber, 2000. Initially, polynomial regression with orders from 1 to 5 are used for the data fitting.

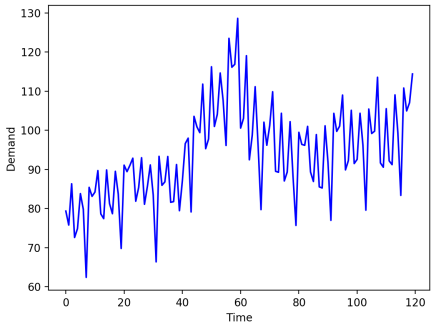


Figure 1: Monthly demand for a hardware device from 1996 to 2006

The first 80% of the data is used a the training set and the last 20% of the data is used as the test set. Figure 2 shows the plots of true demand versus the fitted demand by the polynomial model using training set. Table 1 shows the mean squared error of the polynomial model using test set increases as the order of polynomial model increases. Therefore, we will only consider removing linear trend for estimating AR(p) model.

Order	1	2	3	4
Mean Squared Errors	124.37	277.55	970.39	1284.35

Table 1: Mean squared errors of the polynomial models using test set

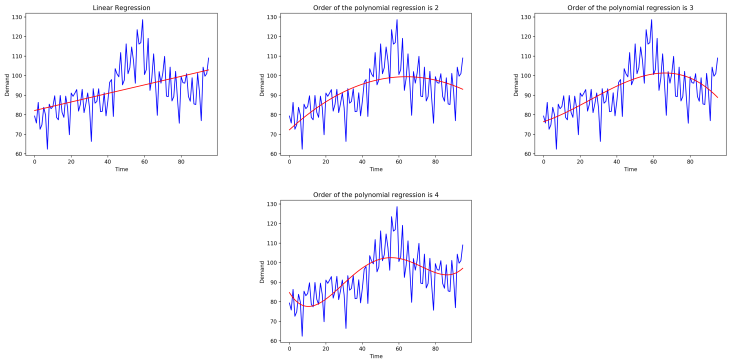
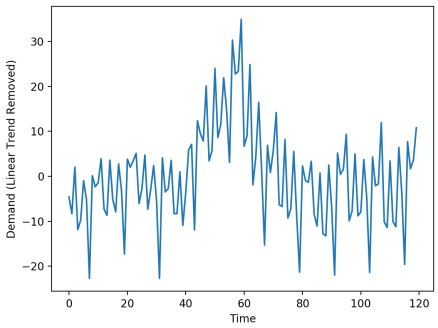


Figure 2: Plots of true demand versus the fitted demand by the polynomial model using training set

5.1 Autoregressive Model

According to table 2, AR(3) has the smallest AIC and BIC. Therefore AR(3) is preferable to used a base model for generating scenarios for the parametric model in Section 3.



:

Figure 3: Linear trend is removed from the raw data

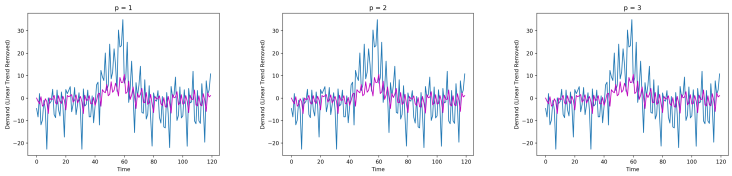


Figure 4: Autogressive Models with p from 1 to 3

p	1	2	3
AIC	905.040	904.626	848.915
BIC	913.402	915.776	862.852

Table 2: Caption

6 Numerical Results

July, 2004 is chosen as the test month. We require that we are only allowed to use the data before July, 2004 to make a decision about how many pieces that need to be replenished at the beginning of the month. As for the data augmentation on the test set, we add a noise, which is Normal random variable with mean 0 and standard deviation 9.29, to the actual demand in July, 2004, which is 96.21. The size of the test set is 10000. We illustrate the performance

of our program by using kNN method. As for kNN method, we let $k = \lfloor N^{\beta} \rfloor$, where $\beta \in (0, 1)$ and N is the size of the dataset.

β_k	N_0	Δ	L
0.6	1000	50	100

Table 3: Parameters for Non-parametric PSP based on kNN

The estimated solution of x, which is the replenishment, after 50 iterations is 85.8654. The results about evaluating the quality of the estimated solution via the test set are provided below.

	Cost	Holding Cost	Backorder Cost
Mean	12.728	10.4711	2.25694
95% CI	[12.5622,12.8937]	[10.3098,10.6323]	[2.11688,2.397]

Table 4: Caption

7 Tutorial

In this section, we will illustrate how to run the algorithm. In the higher level, the user needs to use python files to generate training database with the choice of models (non-parametric model or parametric model).
Database Generator:

As for the python packages, numpy, sklearn, pandas and matplotlib are required. In order to successfully make a decision, one needs to store all the data before the current month in one csv file which has the same format as the one provided by the contest.

parametric_database_generator.py is used to generate training database based on the parametric model. The user needs to specify the path of the original csv file and the path for exporting training database. We use XML-type format to store the training database.

nonparametric_database_generator.py is used to generate training database based on the parametric model. The user needs to specify the path of the original csv file and the path for exporting training database.

psp_parametric_model.py contains all the functions for generate training database based on the parametric model, while psp_nonparametric_model.py contains all the functions for generate training database based on the parametric model.

Solving PSP:

It is required that user needs to install CPLEX and link CPLEX to the program. Xcode is recommmened.

User needs to specify the the path of the training database in the main.cpp. If test database is not provided, user needs to comment out the declaration of the path for test database.

References

Dimitris Bertsimas and Nathan Kallus. From predictive to prescriptive analytics. *Management Science*, 2019.

Yunxiao Deng and Suvrajeet Sen. Learning enabled optimization: Towards a fusion of statistical learning and stochastic programming. *INFORMS Journal on Optimization (submitted)*, 2018.