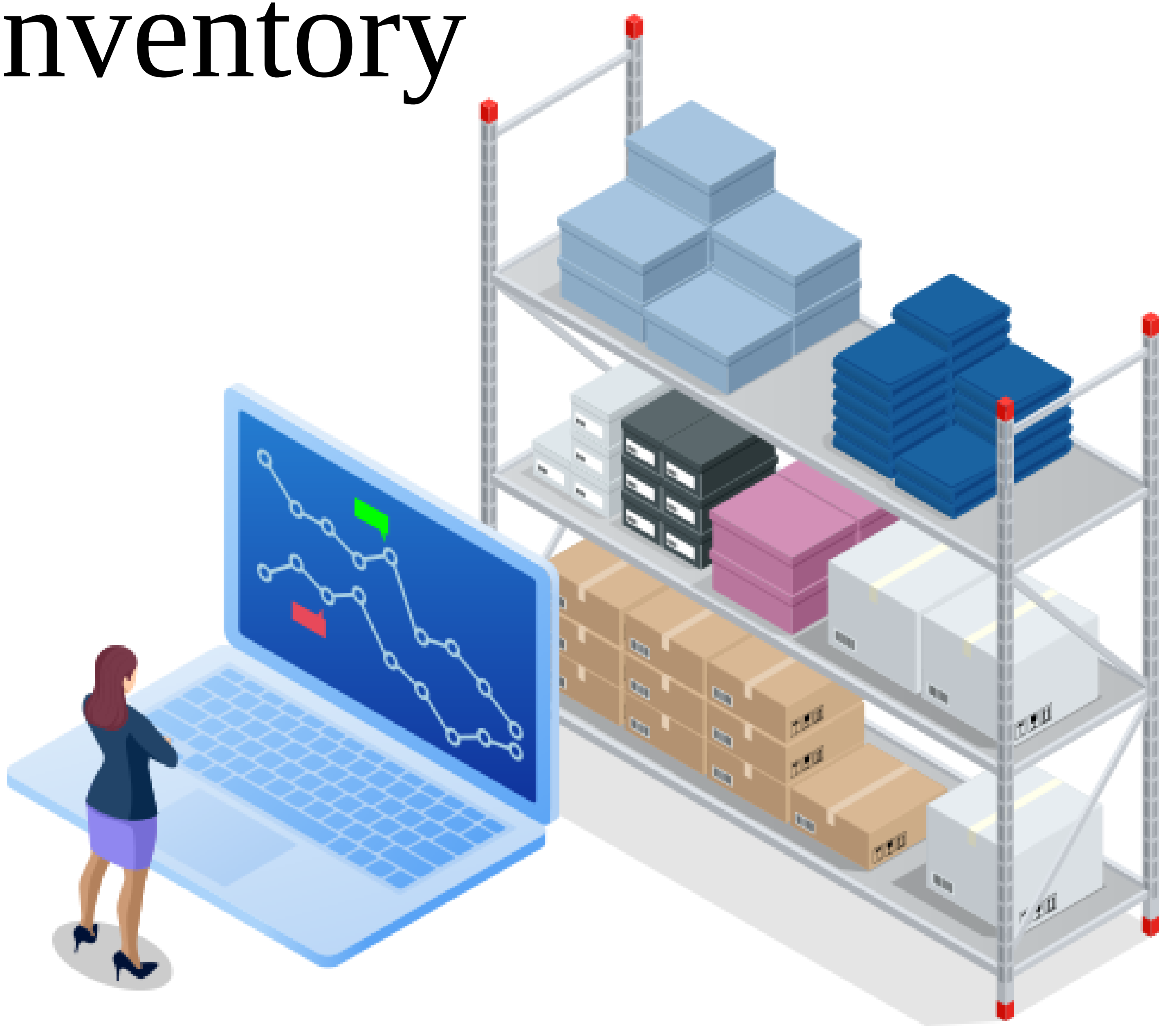
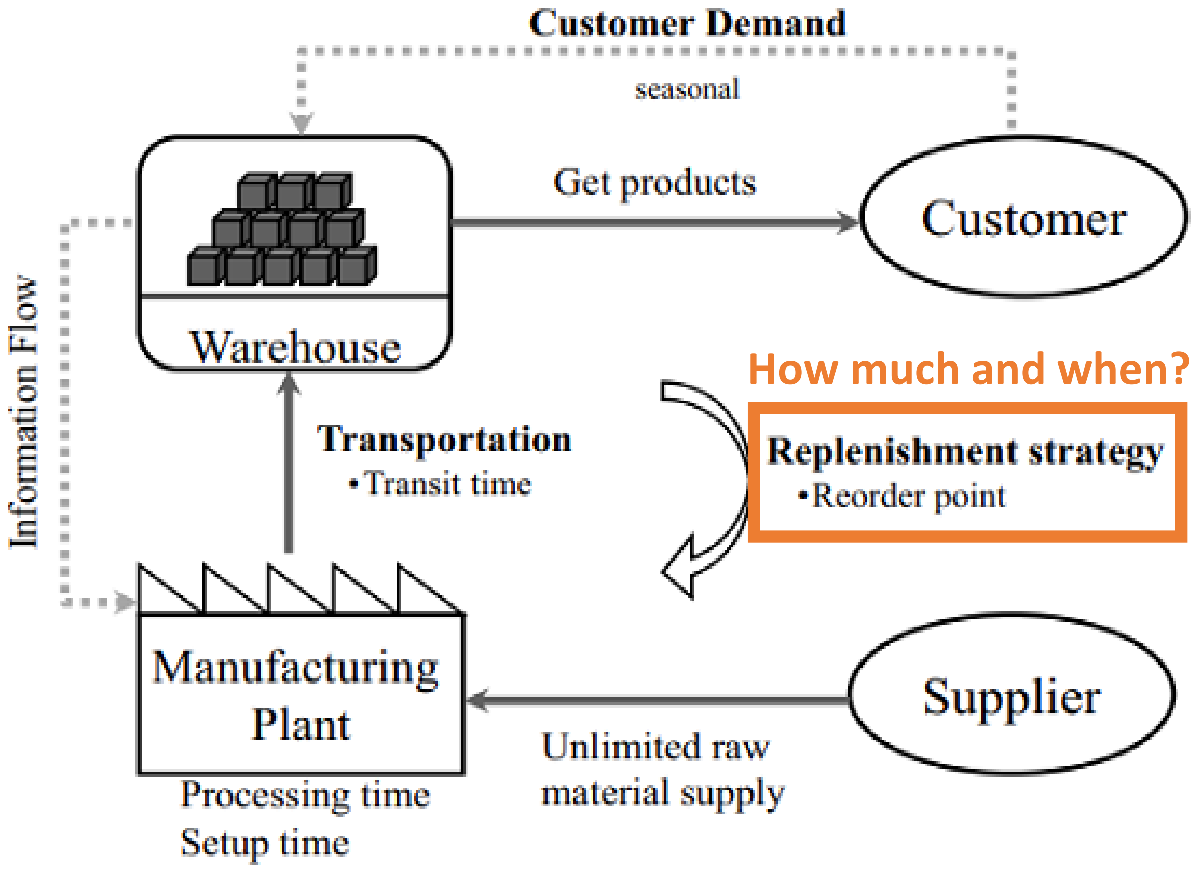
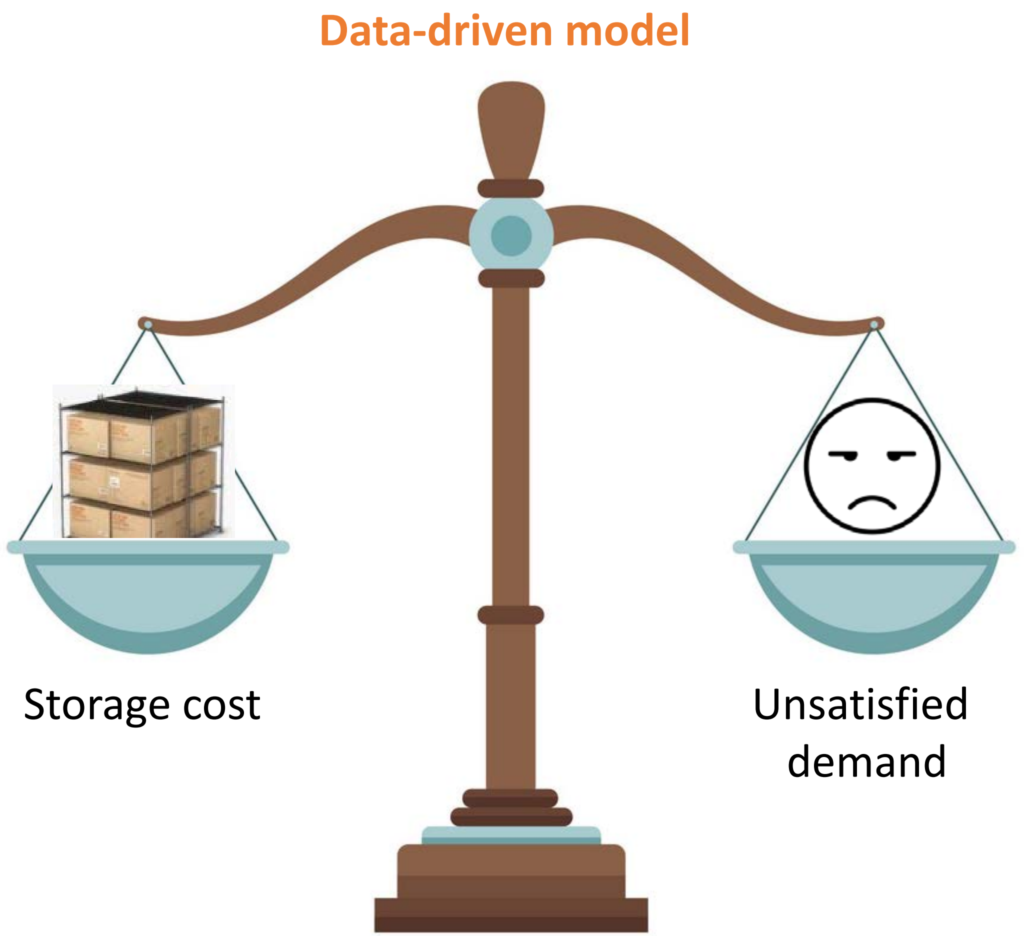
Data-Driven Inventory Management



Angel Wei Huang
STOR 892 Final Project
November 5, 2020

The problem





Grewal et al. (2015)

Data preprocessing

ORDERNU	QUANTITY	PRICEEACI	ORDERLIN	SALES	ORDERDATE	STATUS	QTR_ID	MONTH_II	YEAR_ID	PRODUCTLINE
10112	29	100	1	7209.11	3/24/2003 0:00	Shipped	1	3	2003	Classic Cars
10126	38	100	11	7329.06	5/28/2003 0:00	Shipped	2	5	2003	Classic Cars
10140	37	100	11	7374.1	7/24/2003 0:00	Shipped	3	7	2003	Classic Cars
10150	45	100	8	10993.5	9/19/2003 0:00	Shipped	3	9	2003	Classic Cars
10163	21	100	1	4860.24	10/20/2003 0:00	Shipped	4	10	2003	Classic Cars

Data from https://www.kaggle.com/kyanyoga/sample-sales-data7

- Pick one product: classic cars
- Combine different sub-models (orderline) of classic cars
- Ignore price differences between sub-models
- Aggregate quantity sold by month

Year	*Month*	*Sale*
2003	1	334
2003	2	120
2003	3	929
2003	4	465

Sales data

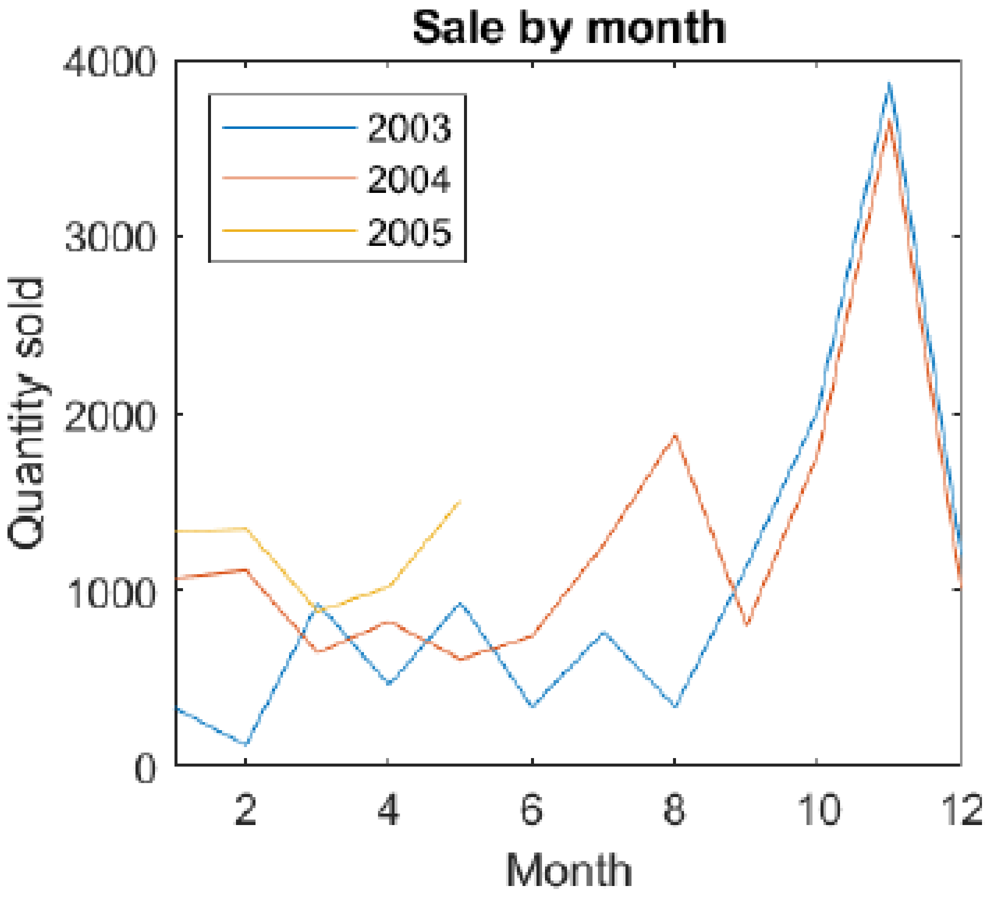


FIGURE 1. *Toy classic car sales data*

Models

- 1. Linear programming based on all time periods
- 2. Linear programming with m-policy
- 3. Neural network demand forecasting

Model 1. Linear programming based on all time periods

- At the beginning of each time period i ($1 \leq i \leq n$ total number of time periods)
- We have X_i quantity of inventory on-hand
- There will be D_i amount of demand during this period
- We do not know the real demand, so use the sales data to simulate the demand
- Goal: to decide A_i, the amount to order at the end of period i.

Model 1. Linear programming based on all time periods

Policy: if the inventory on hand falls below a certain threshold x, we will order A_i to make up the difference, otherwise we will not order.

$$A_i = \begin{cases} x - (X_i - D_i), & \text{if } X_i - D_i < x \\ 0, & \text{if } X_i - D_i \geq x \end{cases} \tag{1}$$

Thus, we have inventory X_{i+1} at the beginning of period $i + 1$:

$$X_{i+1} = max(X_i - D_i, 0) + A_i \tag{2}$$

Then the unsatisfied demand at the end of period i is:

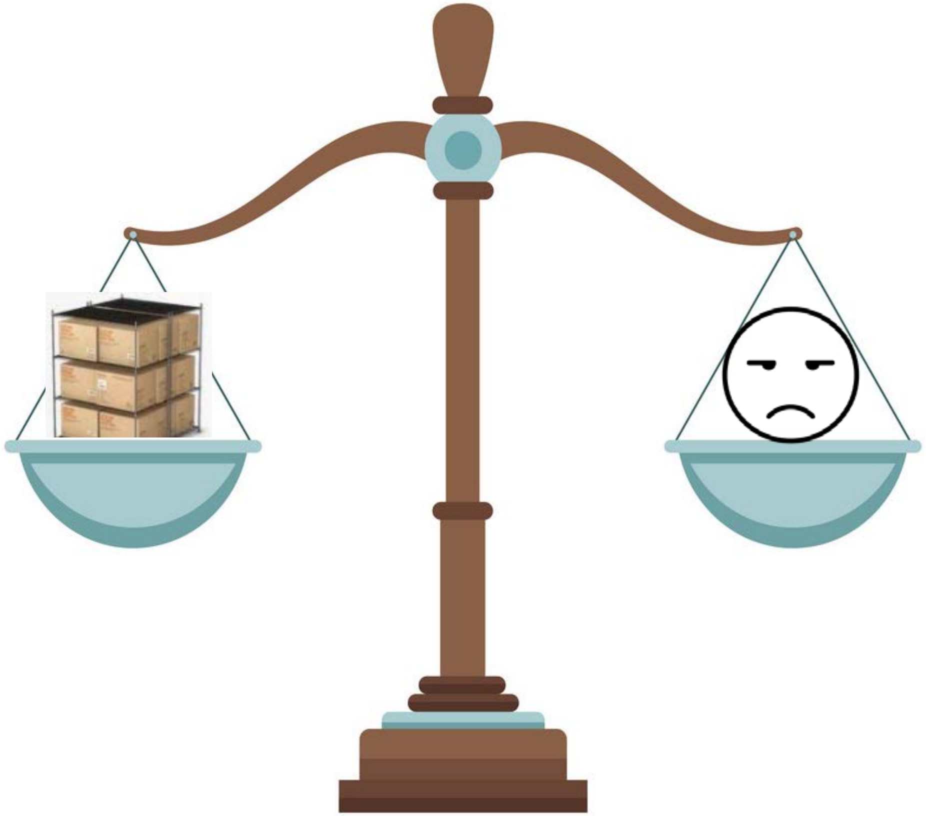
$$L_i = max(D_i - X_i, 0) \tag{3}$$

Model 1. Target function

Optimizing --

Minimizing inventory

$$G = \sum_{i=1}^{n} X_i$$



Satisficing –

No more than 5% unfulfilled demands

$$\sum_{i=1}^{n} L_i \le .05(\sum_{i=1}^{n} D_i)$$

Model 1. Optimal policy

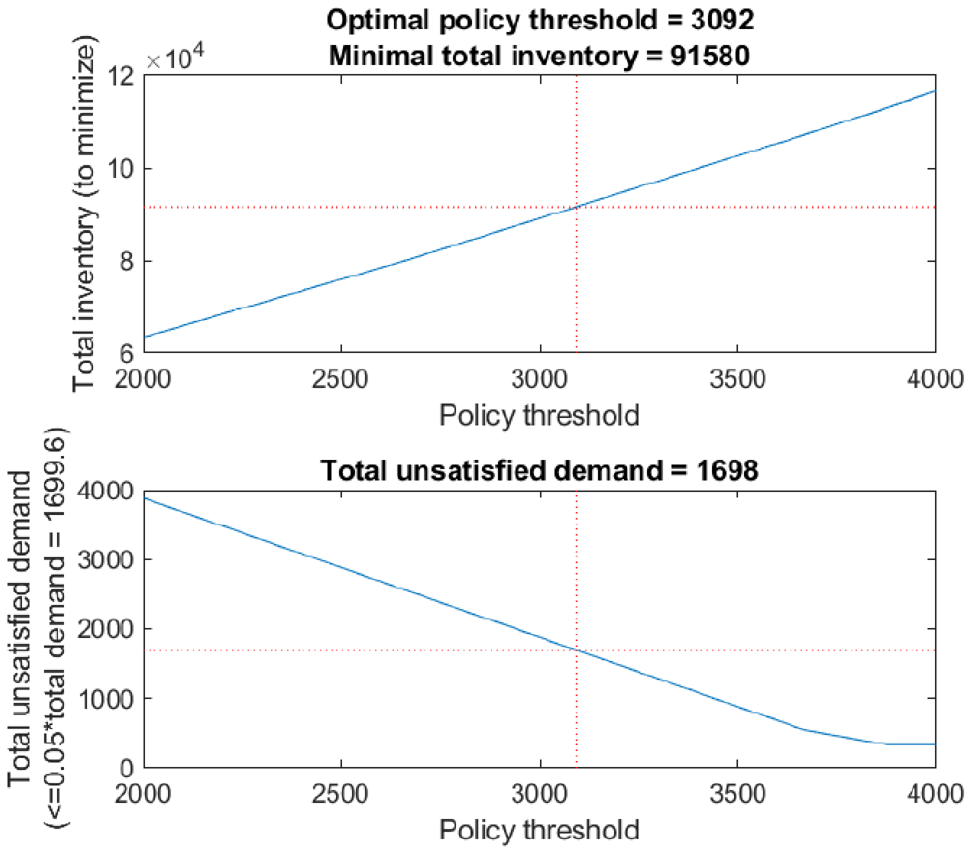


FIGURE 2. *Optimal policy to minimize inventory*

Models

- 1. Linear programming based on all time periods
- 2. Linear programming with m-policy
- 3. Neural network demand forecasting

Model 2. Linear programming with m-policy

1. Choose a time window *m* ($2 \leq m \leq 15$ months)
2. Use data from previous *m* time periods (periods k-$m+1$, k-$m+2,\ldots, k$, $k \geq m$) as training data
3. Use data from the next time period ($k+1$) as test data
4. Find *x* for each time window *m* as described in model 1 with small modification (6) with constraint (7)

$$\min_{x}(G) = \frac{1}{m} \sum_{i=k-m+1}^{k} X_i \tag{6}$$

$$\frac{1}{m} \sum_{i=k-m+1}^{k} L_i \leq \frac{.05}{m} \sum_{i=k-m+1}^{k} D_i \tag{7}$$

Model 2. Linear programming with m-policy

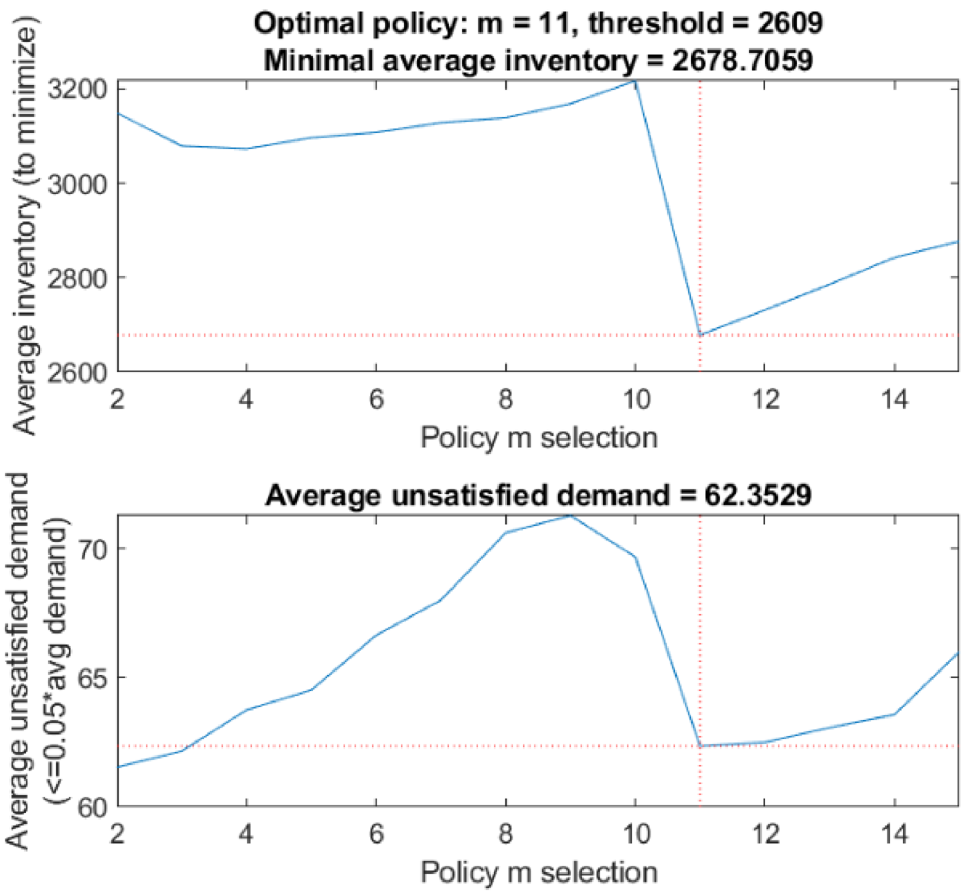
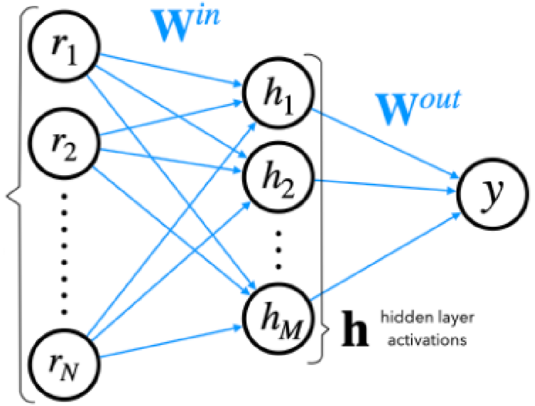


FIGURE 3. *Optimal policy to minimize inventory*

Models

- 1. Linear programming based on all time periods
- 2. Linear programming with m-policy
- 3. Neural network demand forecasting

Model 3. Neural Network[1]



Let $\mathbf{r}^{(n)} = \begin{bmatrix} r_1^{(n)} & r_2^{(n)} & \dots & r_N^{(n)} \end{bmatrix}^T$ denote the vector of sales data from the previous N months in the nth training set. We built a neural network that is described by the following set of equations:

$$\mathbf{h}^{(n)} = \mathbf{W}^{in}\mathbf{r}^{(n)} + \mathbf{b}^{in}, \qquad [\mathbf{W}^{in} : M \times N], \tag{8}$$

$$y^{(n)} = \mathbf{W}^{out}\mathbf{h}^{(n)} + \mathbf{b}^{out}, \qquad [\mathbf{W}^{out} : 1 \times M], \tag{9}$$

where $y^{(n)}$ denotes the scalar output of the network: the next month's order quantity.

The M-dimensional vector $\mathbf{h}^{(n)}$ denotes the activation of the hidden layer of the network.

$$\mathbf{h}^{(n)} = \phi(\mathbf{W}^{in}\mathbf{r}^{(n)} + \mathbf{b}^{in}) \tag{10}$$

1. McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, *5*(4), 115–133.

Model 3. Neural Network – Gradient Descent[2]

1. Evaluate the loss on the training data.

$$Loss = \frac{1}{P}\sum_{n=1}^{P}\left(y^{(n)} - \tilde{y}^{(n)}\right)^2$$

2. Compute the gradient of the loss with respect to each of the network weights.

$$\frac{\partial L}{\partial \mathbf{W}^{in}}, \frac{\partial L}{\partial \mathbf{b}^{in}}, \frac{\partial L}{\partial \mathbf{W}^{out}}, \frac{\partial L}{\partial \mathbf{b}^{out}}$$

3. Update the network weights by descending the gradient that was calculated in step 2.

$$\mathbf{W}^{in} \leftarrow \mathbf{W}^{in} - \alpha \frac{\partial L}{\partial \mathbf{W}^{in}}$$

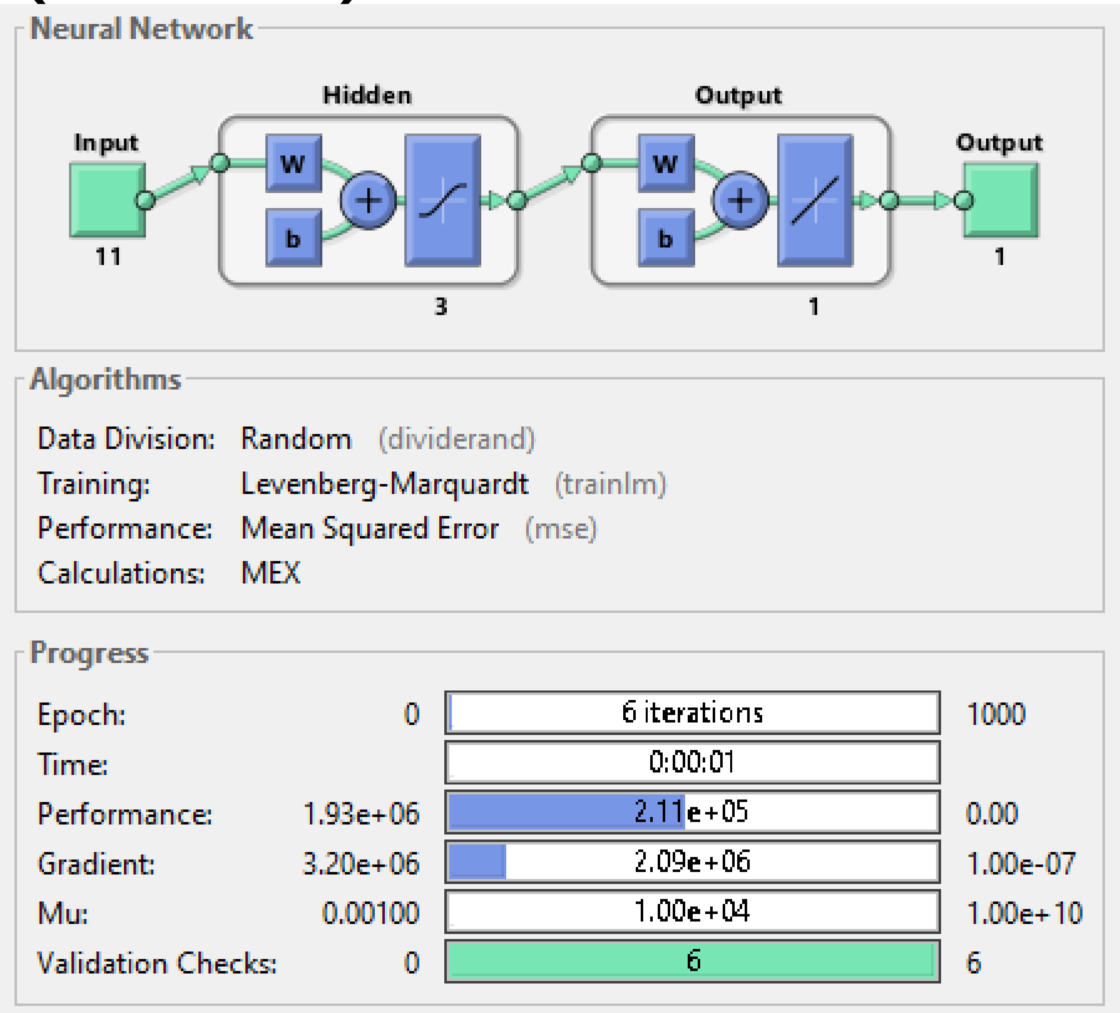
$$\mathbf{b}^{in} \leftarrow \mathbf{b}^{in} - \alpha \frac{\partial L}{\partial \mathbf{b}^{in}}$$

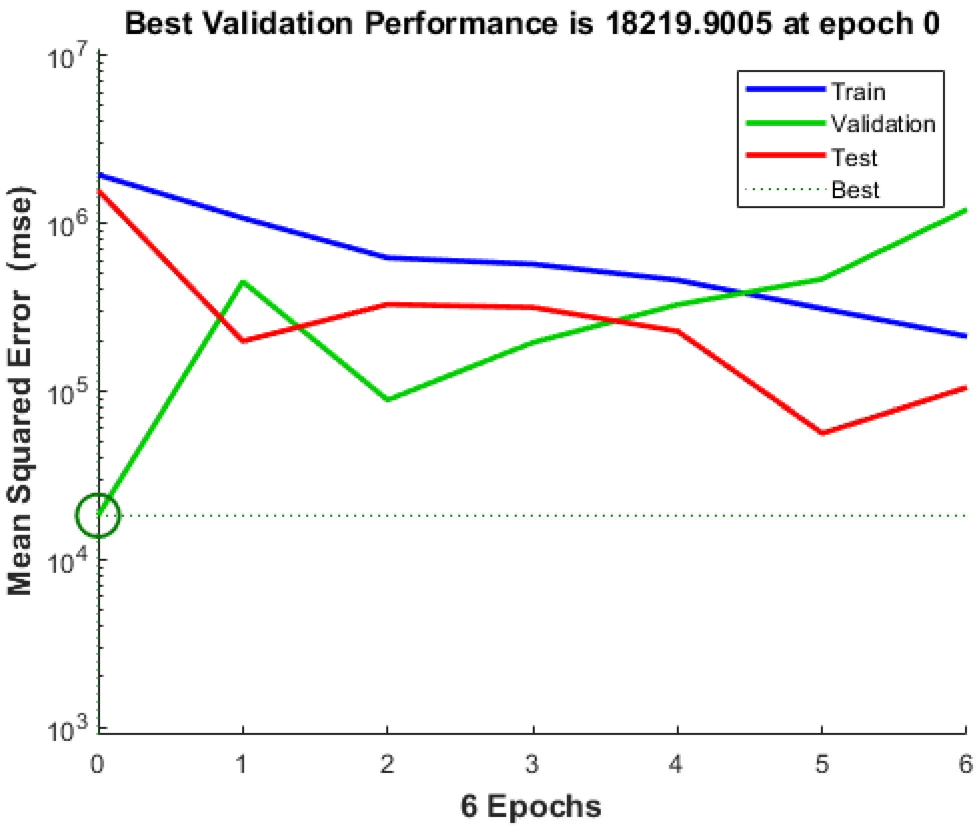
$$\mathbf{W}^{out} \leftarrow \mathbf{W}^{out} - \alpha \frac{\partial L}{\partial \mathbf{W}^{out}}$$

$$\mathbf{b}^{out} \leftarrow \mathbf{b}^{out} - \alpha \frac{\partial L}{\partial \mathbf{b}^{out}}$$

2. Ruder, S. (2016). An overview of gradient descent optimization algorithms. *ArXiv Preprint ArXiv:1609.04747*.

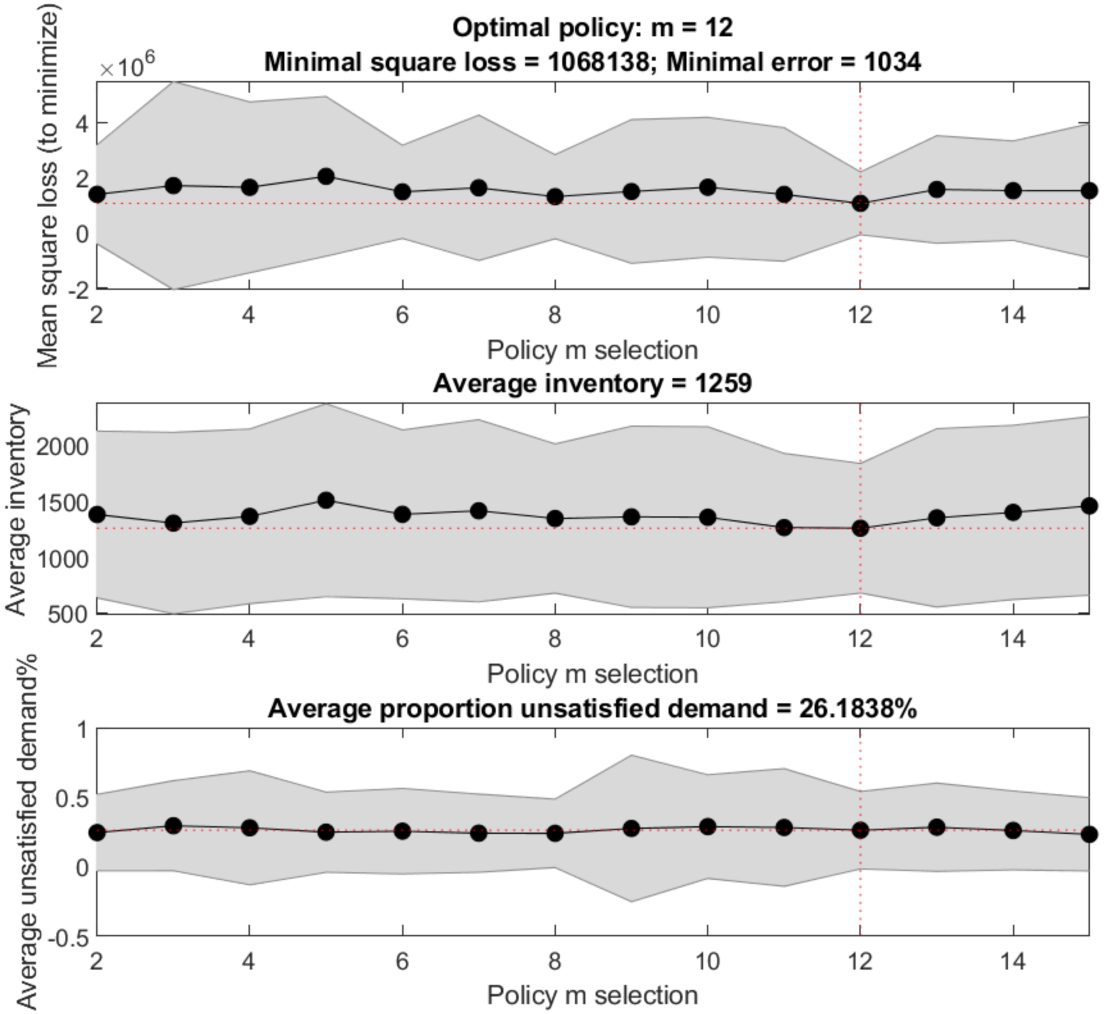
Model 3. Neural Network – training example (m=11)





Model 3. Neural Network – result

- 60:20:20 training:validation:test split
- 100 iteration to calculate performance on test set
- Much lower inventory, but with higher unsatisfied demand



Conclusion

- 1. Linear programming based on all time periods
  - Not realistic, demands more data
- 2. Linear programming with m-policy
  - More sensible and applicable, less demand for data storage (just moving window)
- 3. Neural network demand forecasting
  - Limited performance on small dataset
  - Add sequence information might help (eg. Long short-term memory RNN)

Future Direction

- Choose different target function or a combination of target functions

- Instead of weighing all *m* previous months equally in m-policy, one can choose different weights for the previous months (eg. weight recent months more heavily) when trying to predict the inventory.

- Other methods for time series forecasting given enough data: eg. auto-regression, recurrent neural network, to better predict the future inventory.

Thank you!

Questions?