Dynamic Pricing for Perishable Goods

Norman Rzepka and Johannes Filter

August 24, 2016

Abstract

We present an interactive system that allows the user to develop pricing strategies for selling perishable goods in a competitive marketplace. The system provides a configurable demand estimation model, a calculation for optimal pricing strategies and simulations to evaluate the strategies.

1 Introduction

Today's marketplaces are highly dynamic. With the introduction of online marketplaces and digitalization it has become very easy to update prices multiple times a day. Also, for many goods it is very common to have a large number of sellers. It is a major competitive advantage to have a well-optimized pricing strategy when operating in such markets. A special case of these verticals are markets for perishable goods. In the following we will illustrate two examples of perishable products that are sold in highly dynamic marketplaces:

Airline tickets can only be sold before the departure of the flight, obviously. Therefore they have a very specific date when their sale ends. The demand for the tickets varies greatly throughout the sale. For this report, we assume that the demand is highest shortly before the airplane departs. Especially, for popular routes, such as Frankfurt to London, there are many airline carriers a passenger can choose from. Both established airlines such as Lufthansa and British Airways as well as low-cost airlines like Ryanair serve that route. When choosing a particular flight, some passengers might be looking for the cheapest flight. Others may value the service quality or the proximity of the airports that more expensive airlines offer. A good demand estimation model has to account for a variety of such aspects.

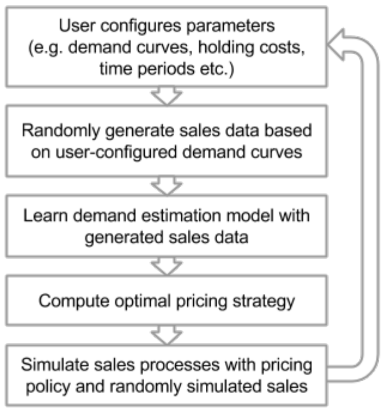
User configures parameters (e.g. demand curves, holding costs, time periods etc.)
Randomly generate sales data based on user-configured demand curves
Learn demand estimation model with generated sales data
Compute optimal pricing strategy
Simulate sales processes with pricing policy and randomly simulated sales

Figure 1: Developing and evaluating dynamic pricing strategies is an iterative process with multiple steps that benefits from fast software tooling.

Fruits such as bananas also have a time-bound selling period because they will eventually rot. When bananas are first presented on the shopping floor of a supermarket they may still be green. After some time they will be fully ripe for a couple of days. Finally, the life-cycle ends when the bananas get brown spots and begin to rot. In this example a consumer is most likely to buy the banana when it is fully ripe. Moreover, bananas are a basic commodity which makes the price very important for the demand estimation. Consumers will very likely choose the fruits from the seller with the lowest price. This is in contrast to the airline tickets where also several other factors play a major role in costumer demand.

As shown in figure 1, developing pricing strate-

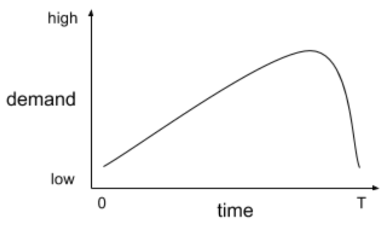

Figure 2: Simplified relationship between demand and time for airline ticket sales

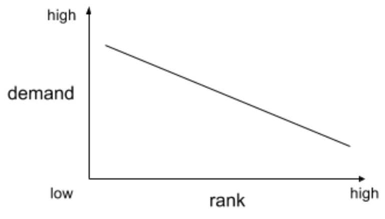

Figure 3: Simplified relationship between demand and price rank for banana sales

gies is an iterative process that benefits from fast software tooling that enables quick cycles. In this report we present a high-performing system for developing dynamic pricing strategies for selling perishable goods. The system provides a configurable demand estimation model, a calculation for optimal strategies and simulations to evaluate these strategies. In the following sections we describe all three components. Section 5 shows the dashboard that allows users to design strategies for their specific needs and evaluate them with a set of simulation statistics. In Section 6 we report how the system has been implemented.

2 Demand estimation model

The demand estimation model provides information about the expected sales of a particular good in a specific time period. So, our dependent variable y is the number of expected sales which may be a real number greater or equal to zero. Our explanatory variables $\vec{x}$ include time and price rank among other features. We create our demand estimation model in two steps: Generate training data and train a regression model. This approach makes it easy to work with real-world training data once historical data becomes available.

First, we generate training data. For that we use a linear combination of a time-based model and a model based on price ranks. The time-based model could express scenarios where the sales probability varies throughout the selling period. For airline tickets that could be a function as shown in Figure 2 that has its peak close to the end. The price rank model represents the influence of the relative ordering of competitors in the marketplace based on price. Specifically, it describes the sales probability if the current seller has the lowest price, second lowest price, third lowest price etc. In our fruit example this would be a function as shown in Figure 3 that decreases steadily. Mathematically, we use configurable cubic polynomials for both models. Therefore, this data generation approach can be easily adapted for a large range of market situations. In order to emulate real-world sales, we add some randomness to each data point.

In our implementation, we generate training data with the help of a random number generator and user-supplied parameters. Users configure the following parameters: the number of time periods T, a set of possible prices A and the coefficients for the time-based polynomial models $c_{t,i}$ and rank-based polynomial models $c_{r,i}$. Because we use cubic polynomial i ranges from 1 to 4. Now, we generate 1000 training data points for every time period t. For each data point we uniformly draw four prices from the set of possible prices A: One for our own offer price a and three for the competitor prices $\vec{p}$. Based on these prices, we calculate our price rank r. As shown in figure 4 we derive explanatory variables $\vec{x}$ from these parameters and then the dependent variable y.

In the second step, we train a regression model on the training data we just created. In our implementation a linear regression model suffices. Other regression models such as support vector regression or neural networks could also be used. This regression model will later be used to estimate demand intensities, i.e. expected sales for a specific time period.

$$\vec{x} = \begin{pmatrix} r \\ a - min(\vec{p}) \\ a \\ t' \\ t'^2 \\ t' \cdot r \\ \sqrt{t'} \\ (1-t')^3 \\ t' \cdot (1-t')^2 \\ t'^2 \cdot (1-t') \\ t'^3 \\ (1-r)^3 \\ r \cdot (1-r)^2 \\ r^2 \cdot (1-r) \\ r^3 \end{pmatrix} \text{ with } t' = \frac{t}{T} \quad (1)$$

$$y = \begin{pmatrix} c_{t,1} \\ c_{t,2} \\ c_{t,3} \\ c_{t,4} \\ c_{r,1} \\ c_{r,2} \\ c_{r,3} \\ c_{r,4} \end{pmatrix} \cdot \begin{pmatrix} x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{15} \end{pmatrix} \cdot Uniform(0,1) \quad (2)$$

Figure 4: Explanatory variables $\vec{x}$ and dependent variable y

3 Optimal pricing strategies

Pricing strategies suggest prices for a specific market situation. A very basic strategy would be to use constant prices through-out the sales period. In many cases such a basic strategy will be outperformed by a more sophisticated strategy that takes the dynamic market situation into account. The market situation can be defined by a set of state parameters. In our system the state parameters include the number of items in stock n, competitor prices $\vec{p}$ and time t.

An optimal strategy suggests prices that will lead to the highest expected profits. We implemented a calculation for optimal strategies based on recursive Bellman equations. In the following we explain the calculation of optimal strategies in more detail (see Figure 5 for a precise definition).

Given a discrete price range, the Bellman function V computes the maximum expected profits and the according price a via recursion. The total expected profits are computed by summing up the expected profits of time period t and all future profits. $\pi(i, x(a,s))$ represents the probability mass function of a *Poisson* distribution. The distribution has been parameterized by $x(a,s)$ which is the prediction function of the previously mentioned demand estimation model. In our implementation, the function iterates over all reasonably probable number of items sold, so that it makes up for 99.99% of the probability mass.

Other variables and parameters:

- z represents the salvage profits that can be gained by selling goods in a secondary market after the end of the sales period. For example, half-rotten bananas might not be sellable to humans but could still be used for animal food.

- s is the market state. In our system, we only

$$
\begin{aligned}
V(t,n,\vec{s}) &= \max_{a\geq 0}\left\{\sum_i \pi(i,\vec{x}(a,\vec{s}))\cdot\big(\min(n,i)\cdot a - n\cdot l + \delta\cdot V(t+1,(n-i)^+,\vec{s})\big)\right\} && (3)\\
V(T,n,\vec{s}) &= z\cdot n && (4)\\
V(t,0,\vec{s}) &= 0 && (5)
\end{aligned}
$$

Figure 5: Adapted Bellman equation for selling perishable goods [7]

encode the competitor prices $\vec{p}$ in s. For other applications this could contain additional relevant parameters that influence customer demand, e.g. time of year.

- l is the holding cost of one item in one time period. In the real-world this could represent costs to store an item in a warehouse or on the shopping floor.
- t represents a time period.
- n represents the number of remaining items in stock.

The Bellman function is implemented with dynamic programming [1]. Our system will compute the optimal prices for all t and n in a specified discrete range. Whenever the state space changes, i.e. the competitor prices $\vec{p}$ change, new optimal prices will be computed. This makes up a fully specified pricing strategy that can be applied in a competitive market. Figure 6 shows a visualization of such a pricing strategy.

4 Simulation

In their abstract representation, pricing strategies are hard to compare against other strategies and difficult to evaluate how they will perform. Therefore, we implemented functionality to simulate sales using a well-defined pricing strategy. Our system runs a series of simulations and computes summary statistics which provide an intuitive overview of a strategy's performance.

Each simulation consists of a parameterized number of time periods. The pricing strategy is consulted in every time period to provide the current price. Based on that price a number of sales is drawn from a *Poisson* distribution that has been parameterized with the respective demand estimation. The competitor prices $\vec{p}$ are the same at the start of each simulation. However, the competitors will randomly adjust their prices during the simulations.

In the current implementation, our system executes 100 simulations which we found to be enough to provide a good understanding of a pricing strategy.

After the simulations are run, a selection of summary statistics are computed:

- Mean price for each time period
- Mean number of items in inventory for each time period
- Mean accumulated profit for each time period
- Out-of-stock probability for each time period
- Histogram of profits after each simulation

5 Dashboard

All of the system's functionality is exposed through an interactive dashboard. As shown in Figure 7 the dashboard allows the user to define the demand estimation models with drag and drop interactivity. Other parameters, such as initial competitor prices $\vec{p}$, salvage profits z and holding costs l, can be set in the dashboard as well. These configurable parameters make the system easily adaptable to a wide range of market situations.

Once the pricing strategy is computed, it can be explored in an interactive graph. This graph shows the optimal prices for specific time periods t and number of remaining items in stock n.

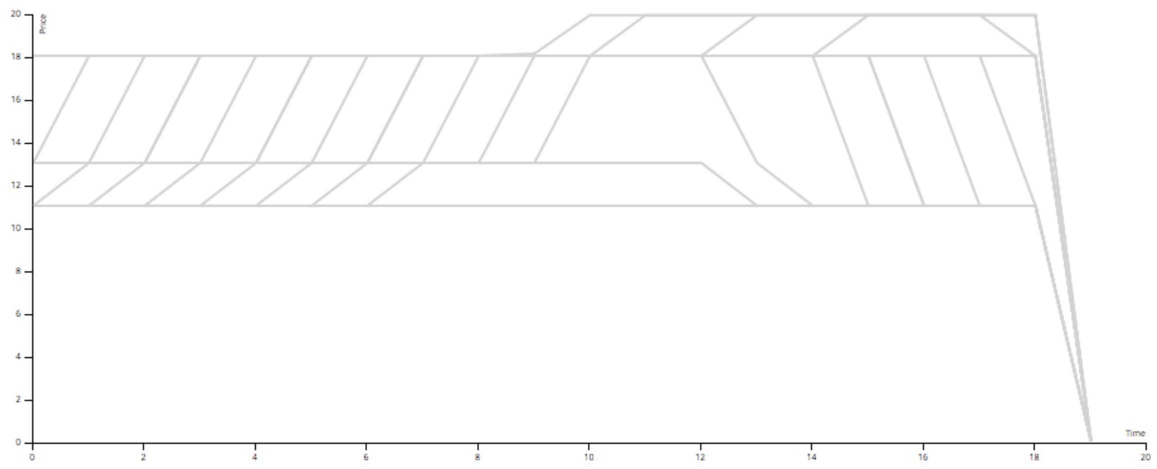

Figure 6: Visualization of an example pricing strategy for selling airline tickets

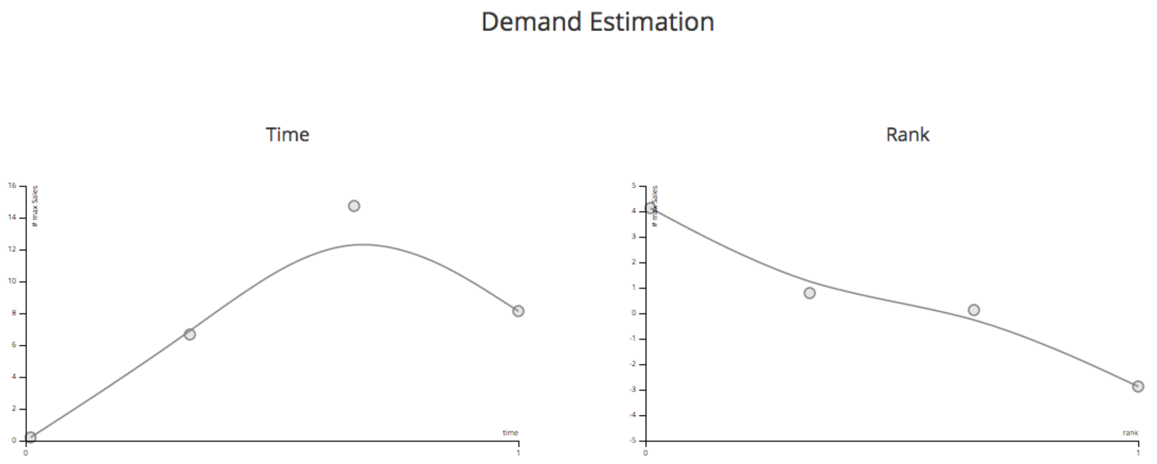

Figure 7: Screenshot of demand estimation model configuration. Both of the models have four knobs to control the cubic polynomials.

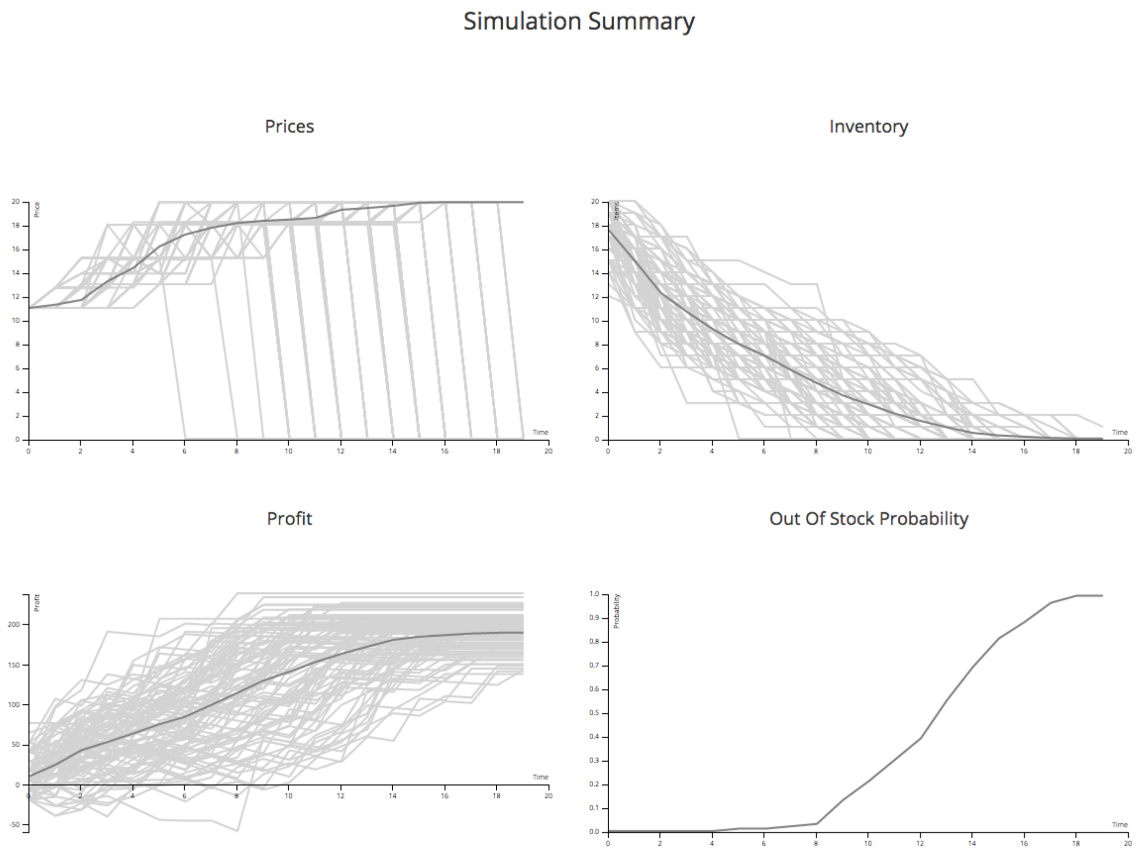

Figure 8: Screenshot of simulation summary statistics

All of the summary statistics outlined in Section 4 are also displayed in interactive charts, as shown in Figure 8. In addition to the mean values, these charts also show the individual values. This allows the user to examine individual simulations including potential outliers. By using this interactive tool the user can get a good sense of the pricing strategy and how it will perform in a real market situation.

6 Implementation

One of our design goals for our dynamic pricing system was to enable high interactivity for a user. This requires the underlying computations to be performed in a highly efficient manner.

We implemented the system in `Python` 3. The `numpy` [5] library was very useful because of its vector and matrix operators. Using vectorized operations instead of `Python` loops improved the performance greatly. Executing the Bellman function to compute an optimal pricing strategy requires dynamic programming. We found that our `Python` implementation of the Bellman function was prohibitively slow, due to a large number of function invocations and loops. Also, vectorized computation was not applicable, due to the recursive nature of dynamic programming. Therefore, we implemented this part of the system as a `Python` extension written in `C++14`. We used the `Python` bindings of the `Boost` library [2] to achieve interoperability between the `Python` and `C++` code. Our system runs inside a `Flask` [4] server which exposes the system's functionality via a `HTTP` interface. The dashboard is implemented as a single-page web application in `HTML5` and `JavaScript`. We used the `D3` [3] library for the interactive graphs and `React` [6] for the forms.

7 Summary

We presented an interactive system for developing and evaluating pricing strategies for selling perishable goods in a competitive environment. Users can adapt the system to a wide range of market situations by tweaking several parameters. The system calculates optimal pricing strategies for specified market situations. With simulations, users can get a better understanding of the pricing strategies and how they will perform in the marketplace.

8 Acknowledgments

The presented system and this report are the result of a group project in the seminar "Dynamic Pricing" at the Hasso Plattner Institute in Potsdam. The seminar was taught by Dr. Rainer Schlosser, Martin Boissier and Dr. Matthias Uflacker.

References

[1] Richard Bellman. "Dynamic programming and Lagrange multipliers". In: *Proceedings of the National Academy of Sciences* 42.10 (1956), pp. 767–769.

[2] *Boost.Python.* `http://www.boost.org/doc/libs/1_61_0/libs/python/doc/html/`. Accessed: 2016-07-21.

[3] *D3.* `https://d3js.org/`. Accessed: 2016-07-20.

[4] *Flask.* `http://flask.pocoo.org/`. Accessed: 2016-07-20.

[5] *NumPy.* `http://www.numpy.org/`. Accessed: 2016-07-20.

[6] *React.* `https://facebook.github.io/react/`. Accessed: 2016-07-20.

[7] Rainer Schlosser. "Dynamic Pricing on Online Marketplaces: Strategies and Applications". Description of Projects. May 2016.