# Dynamic Pricing on Commercial Websites: A Computationally Intensive Approach

**Patrick Mullen, Kevin Seppi and Sean Warnick**

Computer Science Department, Brigham Young University, Provo, UT 84602

## Abstract

With commercial websites now selling anything that can be obtained in a brick and mortar store it would be helpful to automate the monitoring and the adjustment of prices to maximize pro t. We propose a computationally intensive, simulation-based, approach to dynamic pricing in this context. In this approach we simulate the e ect of various pricing strategies based on what we know about demand and choose the pricing strategy that obtains the highest revenue under this assumption. Using this method we explore the tension that exists between exploring prices to better learn demand behavior and exploiting what we have already learned.

## 1 Introduction

It is a tedious task to maintain prices on commercial websites selling hundreds or thousands of items. It would be better to automate the process of monitoring prices, monitoring quantities sold, and adjusting prices as needed to increase pro ts. We propose an optimal sampling approach to solve this problem. We estimate the long run value of each potential pricing policy to determine whether we should take advantage of what we know about demand or if we should use di erent prices to explore the demand curve. If we choose to explore, we will choose a price that may yield lower initial revenue but will allow us to pick better prices in the future.

An important part of any type of pricing algorithm is modeling demand. According to Kalyanam [3] a log linear model may be used to model demand (we also use the simulation based approach taken in Kalyanam's work). In this case the quantity demanded, $d$, is modeled using two parameters, and $\beta$, and the equation

$$d = e^{(\alpha + p\beta + \epsilon)}, \tag{1}$$

where $p$ is the chosen price, $\beta$ is strictly negative, and $\epsilon$ is noise drawn from a normal distribution with a mean of zero and variance $\sigma^2$. We assume that and $\beta$ change with time. At each time step we draw a new and $\beta$ from independent normal distributions with the old values as the mean and a variance of $\sigma^2_{\alpha\beta}$ (a random walk). We use a Kalman lter to track these variables as suggested by Harvey [2], however we use the notation from Russell [4]:

$$\begin{aligned}
\mu_{t+1} &= F\mu_t + K_{t+1}(z_{t+1} \quad HF\mu_t) \\
\Sigma_{t+1} &= (I \quad K_{t+1}H)(F\Sigma_t F^T + \Sigma_x) \\
K_{t+1} &= (F\Sigma_t F^\top + \Sigma_x)H^T(H(F\Sigma_t F^T + \Sigma_x)H^T + \Sigma_z)^{-1} \\
x_t &= [\ _t, \beta_t]^T \\
H &= [1, p]^T \\
F &= I
\end{aligned}$$

where $\mu$ is the mean of and $\beta$, $\Sigma_0$ is the prior covariance of $\mu$, $\mu_0$ is the prior mean of $\mu$, $z$ is the log of the observed quantity demanded, $\Sigma_z$ is the variance of $z|\ ,\beta$, $F$ is the system transition matrix and $\Sigma_x$ is the covariance of $x_{t+1}|x_t$. The Kalman lter attempts to estimate a hidden state of a system based on noisy observations over a period of time.

Using the ltered value of $\beta$, $\hat{\beta}$, we can attempt to maximize revenue, $pd$, by setting $p = \quad 1/\hat{\beta}$. This approach tends to select prices in a narrow range. If the initial estimate of $\beta$ is incorrect we will not see much information that would allow us to discover our error (slopes, such as $\beta$, are di cult to estimate when all of the data points are close to each other). Selecting a price away from $1/\hat{\beta}$ would help us better estimate the slope, but potentially at the cost of selecting a sub-optimal price in the current time step.

The Kalman lter provides a covariance matrix ($\Sigma_t$ which we will refer to from now on as $\Sigma_{\alpha\beta}$) in addition to means for and $\beta$ ($\mu_t$ which we will break into its individual components, $\hat{}$ and $\hat{\beta}$). These values model our understanding of the true and $\beta$. We can use these values to simulate what we expect to occur in the next time step. We can likewise simulate what

will happen in all time steps through the end of the time period we are interested in.

In our dynamic pricing algorithm we run this simulation for each price under consideration. Initially our algorithm will simulate the expected results under three possible scenarios:

1. Pick the optimal price ($p = 1/\hat{\beta}$) at all time steps
2. Pick a low price for the next time step and then use the optimal price for all of the subsequent time steps
3. Pick a high price for the next time step and then use the optimal price for all of the subsequent time steps

Once these simulations are completed, the algorithm simply selects the pricing choice that yields the largest long term revenue. Since we explore the e ect of high or low prices on the quantity demanded until there is little or no value in doing so, we consider this an "Optimal Sampling" approach.

We acknowledge that this direct simulation approach is computationally intensive. We assume that commercial websites are constructed to meet peak or near peak demand and that there is ample o -peak computational capacity to re-assess pricing.

Through this approach we quantify the tension between setting the price in a way we believe will maximize the current time period's revenue and setting the price to some other value in order to learn more about our target market and thereby increase the revenue in future time periods. Another solution to the dynamic pricing problem is to analytically estimate the future value by looking one step into the future and scaling the e ect to estimate the e ect on future time periods [1]. This approach relies on the correct estimation of the scaling and other parameters.

# 2 Algorithm

For purposes of explanation we present the algorithm in two parts. Algorithm 1 contains the main logic of the algorithm. This algorithm uses Algorithm 2 ($UsePthenExploit(p)$) to run a Kalman lter and return the expected revenue using the argument $p$ for the price in the rst time step and the estimated optimal price for all subsequent time steps.

Algorithm 1 begins by obtaining the actual quantity demanded from the previous time step. This value is passed through a Kalman lter to produce ltered estimate of and $\beta$, $\hat{}$ and $\hat{\beta}$, and a corresponding covariance matrix $\Sigma_{\alpha\beta}$. These values encode all that can be inferred about the true values of and $\beta$ given the current observation of the quantity demanded, all previous observations of quantity demanded, and any prior knowledge of and $\beta$. Since Algorithm 1 will simulate expected revenues, we must run the simulation multiple times to produce acceptably accurate estimates of the expected revenue (see the loop starting at line 4). This loop simulates new values of and $\beta$ and then accumulates expected revenue under three polices. We calculate $rev_{optimal}$ by assuming that the estimated optimal price is used for all time periods, including the rst one. We calculate $rev_{pmin}$ by assuming that the estimated optimal price is used for all time periods, *except* during the rst time period when a low price is used ($p_{min}$, set by the administrator as a lower bound on the price). We calculate $rev_{pmax}$ by assuming that the estimated optimal price is used for all time periods, *except* during the rst time period when a high price is used ($p_{max}$, set by the administrator as a upper bound on the price). Algorithm 1 concludes by returning the price, $1/\hat{\beta}$ (the estimated optimal price), $p_{min}$, or $p_{max}$ which yielded the maximum simulated long-term revenue.

**Algorithm 1** Simulating to choose the best price

```
1:  Get quantity demanded from previous time period
2:  Use the Kalman filter to compute Σ_αβ, α̂ and β̂
3:  rev_pmax, rev_pmin, rev_optimal ← 0
4:  for j = 0 to j < NumberOfDraws do
5:      // Simulate the next α and β
6:      α_temp ~ N(α̂, Σ_αβ)
7:      β_temp ~ N(β̂, Σ_αβ)
8:      // Calculate Simulated Revenue using Algorithm 2
9:      rev_optimal ← rev_optimal + UsePthenExploit(−1/β̂)
10:     rev_pmax ← rev_pmax + UsePthenExploit(p_max)
11:     rev_pmin ← rev_pmin + UsePthenExploit(p_min)
12: end for
13: Set p as the argument (rev_optimal, rev_pmax or
    rev_pmin) with the highest total revenue
```

Algorithm 2 begins by creating a new instance of the Kalman lter (called KF). This lter will be used to simulate the computation of the optimal price in future time periods. The loop starting at line 2 of Algorithm 2 simulates the behavior of the market and pricing system for all time periods from the current time up to the end of the period of interest. Note that the price passed in as an argument is used as the price for the rst time period. After the rst time period the demand is simulated and used to update KF. We then use KF to produce an estimate of $\beta$ which is in turn used to compute an expected optimal price (see lines 10-14).

A variation of this algorithm has been created where once we exploited (used the estimated optimal

**Algorithm 2** UsePthenExploit(p): Simulated revenue using price $p$ for the rst time period then use the estimated optimal price

```
Initialize a new Kalman filter (KF) for use inside of UsePthenExploit(p) using Σ_αβ, α̂ and β̂
for k = 1 to t <= TimePeriodsLeft do
    if k=1 then
        price ← p
    else
        price ← −1/β_KF
    end if
    //α_temp and β_temp come from Algorithm 1
    demand ← e^(α_temp + price * β_temp)
    revenue ← revenue + price * demand
    Update KF based on simulated demand
    β_KF ← getbeta(KF)
end for
return revenue
```

price rather than either the lower or upper prices) we would always use the expected optimal price. We refer to this variant as *Once Exploit Always Exploit*. This has the bene t of running faster, since we do not need to simulate at every time step. Since we allow the true and $\beta$ to vary, *Once Exploit Always Exploit* may diverge from the true and $\beta$. To overcome this we can add in some random exploration. With a certain probability we choose a random price (uniform between $p_{min}$ and $p_{max}$) instead of using our knowledge of $\beta$ to choose the price optimally. We call this variant *With Random Exploration*.

Another variation is created by adding more price points to the simulation. We call this variant *Five Prices*. Instead of just choosing between an upper price, a lower price and setting the price optimally, we can have some other xed prices in between (2 more, for a total of 5 choices in this case). This o ers a nice compromise between learning and maximizing our revenue for the immediate time period. These extra price points allow us to learn something about the market without having to lose as much pro t as if we used the boundary prices.

# 3 Simulation Setup

We have tested the behavior of our pricing algorithm using a simulated market. This simulation was run for 1000 times steps. The result presented below are averages over 4000 simulations of each of these 1000 time steps. We have constructed the simulated market using the parameters given in Table 1.

To initialize our priors for the Kalman Filter we assume that we have two quantities demanded, one for the highest allowed price and one for the lowest allowed price. We used these two data points to solve for and $\beta$ using the two instances of Equation 1 assuming no noise:

$$log(d_{pmin}) = + p_{min}\beta,$$
$$log(d_{pmax}) = + p_{max}\beta,$$

Table 1: Initial Values for Variables

| Variable | Value |
|---|---|
| | 9.0 |
| $\beta$ | 1.3 |
| $\sigma^2$ | 3.5 |
| $\sigma^2_{\alpha\beta}$ | .005 |
| $p_{max}$ | 3.0 |
| $p_{min}$ | 0.25 |

Our system covariance matrix ($\Sigma_x$) has the true variance ($\sigma_{\alpha\beta}$) in the diagonals, and $\sigma_{\alpha\beta}/2$ in the o diagonals. Changing this covariance matrix within a certain amount did not have a large e ect on the performance of our algorithm. The prior covariance matrix ($\Sigma_0$) was initialized as $(I\mu_0)/2$. The sensor covariance matrix ($\Sigma_z$) was initialized to $I\sigma^2$. The observation ($z_t$) is the natural log of the quantity demanded at each time step.

# 4 Results

Figure 1 shows the mean cumulative revenues for each of our variants. This is total revenue up to some time step, divided by this time step. *Optimal Pricing* refers to the revenue obtained if the true values of the parameters ( and $\beta$) were known at each time step. This is the best possible revenue. *Five Prices* uses the once exploit always exploit ideology with ve prices instead of three. The other two policies are based on three prices. *With Random Exploration* simulates until the exploit price is chosen at which point it exploits with 99% probablity and chooses the price randomly otherwise. From this graph we can see where the con ict between exploration and exploitation comes in. *Always Exploit* is the revenue obtained when we use the Kalman lter to estimate $\beta$ and exploit it at every time step, but never explore other prices nor simulate the potential revenue.

Note that if we are only setting prices for a short period of time it is best to use the *Exploit Always* policy, since this brings in more revenue short term. Note also that the *Always Simulate* policy is not shown on the graph because it never achieved an average revenue above 10000.

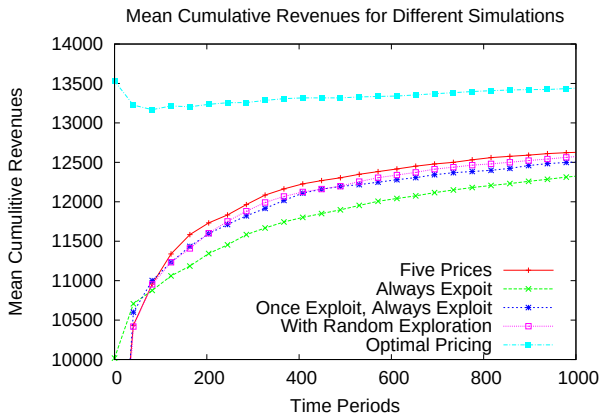


Figure 1: Optimal Pricing is what we reach for and Always Exploit is what we measure against.

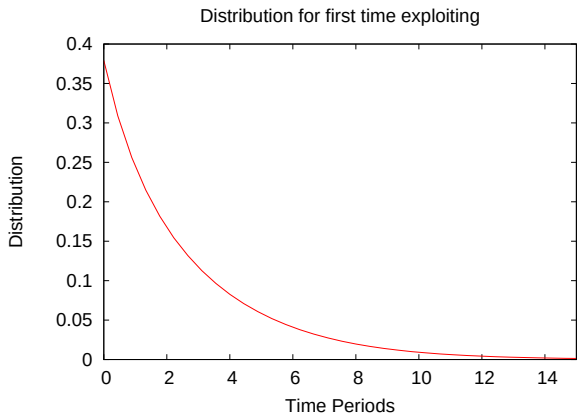


Figure 2: Distribution for rst time exploiting based on three prices and three draws

In the introduction we asserted that this computationally intensive approach to pricing was practical in the context of a commercial website. Table 2 shows the average time to run our pricing algorithm. It only took ten seconds to do a simulation for seven price points using 50 draws. Even though this is computationally intensive, it is very manageable on a server that is built to handle high volume websites.

Table 2: Seconds to complete one simulation (300 time steps) depending on number of prices and draws.

| | Number of Draws | | |
|---|---|---|---|
| Number of Prices | **3** | **10** | **50** |
| **3** | .27 | .97 | 4.74 |
| **5** | .48 | 1.61 | 7.40 |
| **7** | .68 | 2.26 | 10.38 |

Figure 2 shows a distribution of how long the simulation explored before exploiting the rst time. Surprisingly 37% of the time there was no exploration the rst time through the simulation. This is probably due to having good priors.

# 5 Conclusion

For our experimental setup the *Five Prices* simulation performed the best. We were very disappointed to see that *Always Simulate* performed so poorly. However our modi ed methods do o er some improvement over using just the Kalman based estimate of the market. The *Once Exploit Always Exploit* variant yielded good gains over *Always Exploit*. Small increases were added to this by adding some random exploration (*With Random Exploration*) or by adding more price points to the simulation (*Five Prices*). We note that if we have very short term goals we may want to just apply our prior knowledge rather than exploring other price points.

We have also demonstrated that the computational resources required by our algorithm are manageable in the context of commercial website.

There are several avenues that we would like to explore in the future. The rst is that our simulations assume that after the initial price exploitation we will exploit thereafter. This may not be the best policy and we would like to look at other options. In addition we may be able to quantify the number of time steps necessary to simulate. There should not be a need to run our simulation for the full 1000 time steps into the future as there is probably not much information gained after the rst few. We would also like to calculate the number of prices and draws that could be done on some typical servers with average workloads.

# References

[1] Alexandre X. Carvalho and Martin L. Puterman. Dynamic pricing and lerning over short time horizons. 2004.

[2] Andrew C. Harvey. *Forecasting, Structural Time Series Models and the Kalman Filter*. 1989.

[3] Kirthi Kalyanam. Pricing decisions under demand uncertainty: A Bayesian mixture model approach. *Marketing Science*, 15(3):207–221, 1996.

[4] Stuart J. Russell and Peter Norvig. *Arti cial Intelligence: A Modern Approach - Second Edition*. 2003.