




THE FRANZ EDELMAN AWARD
Achievement in Operations Research

A Multiobjective Optimization for Clearance in Walmart Brick-and-Mortar Stores

Yixian Chen,^a Prakhar Mehrotra,^a Nitin Kishore Sai Samala,^a Kamilia Ahmadi,^a Viresh Jivane,^a Linsey Pang,^a Monika Shrivastav,^a Nate Lyman,^b Scott Pleiman^b

^a Walmart Labs, Sunnyvale, California 94086; ^b Walmart, Bentonville, Arkansas 72716

Contact: yixian.chen@walmartlabs.com,  <https://orcid.org/0000-0001-9660-9176> (YC); prakhar.mehrotra@walmart.com (PM); nitinkishoresai.samala@walmartlabs.com (NKSS); kahmadi@walmart.com (KA); viresh.jivane@walmartlabs.com (VJ); linsey.pang@walmartlabs.com (LP); monika.shrivastav@walmartlabs.com (MS); nate.lyman@walmart.com (NL); scott.pleiman@walmart.com (SP)

Accepted: November 3, 2020

<https://doi.org/10.1287/inte.2020.1065>

Copyright: © 2021 INFORMS

Abstract. We developed a novel multiobjective markdown system and deployed it across many merchandising units at Walmart. The objectives of this system are to (1) clear the stores' excess inventory by a specified date, (2) improve revenue by minimizing the discounts needed to clear shelves, and (3) reduce the substantial cost to relabel merchandise in the stores. The underlying mathematical approach uses techniques such as deep reinforcement learning, simulation, and optimization to determine the optimal (marked-down) price. Starting in 2019, after six months of extensive testing, we implemented the new approach across all Walmart stores in the United States. The result was a high-performance model with a price-adjustment policy tailored to each store. Walmart increased its sell-through rate (i.e., the number of units sold during the markdown period divided by its inventory at the beginning of the markdown) by 21% and reduced its costs by 7%. Benefits that Walmart accrues include demographics-based store personalization, reductions in operating costs with limited numbers of price adjustments, and a dynamic time window for markdowns.

Keywords: merchandising • price optimization • inventory optimization • deep reinforcement learning • Edelman Award

Introduction

Walmart Inc. has been the world's largest retailer since 1989, generating total revenue of \$514.4 billion in 2019. Walmart's U.S. business focuses on three strategic merchandising units: grocery (56% net sales), health and wellness (11% net sales), and general merchandise (33% net sales). General merchandise includes entertainment (e.g., electronics), home and seasonal products, hardlines (e.g., stationery), and apparel. In the United States, the company employs 1.5 million associates who serve customers in 4,769 brick-and-mortar stores spanning all 50 states. Walmart earns the trust of customers every day by providing a broad assortment of quality merchandise and services using everyday low price (EDLP) and everyday low cost (EDLC) strategies. EDLP is the pricing philosophy under which Walmart prices items at a low price every day so that customers trust that the prices are the lowest in the market and do not fluctuate. EDLC is the company's commitment to control expenses so that cost savings benefit its customers.

The retail landscape is an ever-changing phenomenon. In past decades, the convenience of shopping implied more expensive products; this expectation is no longer true because consumers now expect both convenience and low prices. Today's retail industry is at an inflection point, primarily driven by changes in consumer-behavior patterns. The competition among retailers has shifted from a race among a few massive players that sell nationally to several smaller players that cater to local regions. As a result, retailers must not only provide a range of new products, but also consistently find new ways to make their operations more efficient and nimbler, thus enabling them to pass on savings to their customers. History is rife with examples in which retailers have consistently employed disruptive innovations to meet their customers' needs and to understand their behavior; for example, they created supercenters, which became one-stop shops, and introduced online shopping with one-day delivery. However, because of the recent wave of massive digital transformation, the changes in consumer patterns

have become more intermittent and less predictable. To keep up with these sporadic changes in customer needs, the core of every retail organization (i.e., its merchandising function) must adapt to consumer trends.

Markdown is one of the core merchandising operations for retailers. It is a deliberate reduction in the price of retail merchandise to increase the sales velocity (i.e., rate of sales at a specific point in time) to clear shelves of an item by a specific date. It can happen for any of the following reasons:

1. To remove unsold merchandise and, thus, create display space for other products. In Walmart's retailing, *modular* refers to all products that a store sells within a given product category (e.g., ice cream). When the composition of products within a given product category changes (e.g., the replacement of one brand of ice cream with another), we refer to this change as *modular refresh* and the new composition of products as the *new modular*. For a product category, modular refresh occurs at least once a year for a given store.

2. To remove seasonal items (e.g., Christmas lights) after the season has ended.

3. To remove unsold perishable items (e.g., meat or milk) close to their sell-by-date expiration.

In a given year, roughly 80% of the categories—including obsolete merchandise from modular refresh, seasonal items, and perishable items—experience markdowns in a particular store. The number of categories needing markdown differs every year because the decision making inherent in each supply chain strategy varies. Unlike other price changes that a store can make, the price of an item undergoing markdown is reduced permanently, and the item will not be replenished—that is, Walmart will not refill the item on its shelves.

Walmart is a mass merchandiser and carries more than 100,000 items in any given store or supercenter across all spectrums of retail (e.g., grocery, general merchandising, and health and wellness) (Walmart 2005). Hence, markdown decisions must be made across all its merchandising units; that is, across multiple stores, different merchandising units are marked down at the same time. Determining an optimized markdown strategy for Walmart is more complicated than for other retailers, because the company's markdown strategy requires that its stores provide updated prices for markdown products across different categories each week. The effect of having diverse, but associated, items in the same stores makes finding an optimal markdown strategy even more challenging (e.g., marking down tortilla chips affects the sale of salsa because consumers often buy these products together).

Walmart's Previous Markdown Approach

For simplicity of execution across various organizations within the business (e.g., operations, merchandising), Walmart implemented a rule-based, tiered discounting process (e.g., 25%–50%–75%) for all items that need to be marked down. Using a 25%–50%–75% policy, all items in a given category are marked down by 25% across all its stores during the first week of the markdown period; after a week, all unsold inventory in this category is then discounted to 50% of the original price. Finally, after several weeks, if any products in the category are still left, they will be marked down to 75% of the original price. Any inventory left unsold after this period will be removed from the shelves.

Why Do We Need an Algorithmic Approach to Markdowns?

Although the previous rule-based approach provides easy interpretability and implementation, it does not leverage patterns in the data (e.g., it does not consider the demand pattern for different items); it is not optimal for achieving revenue; and it is not personalized to each specific store and item. For example, each item in a store can have different price elasticities and a unique optimal price, at which Walmart can sell everything by the desired date. The objectives of making the markdown process more efficient and more data-driven inspired us to develop an algorithmic approach to markdowns.

Markdowns in brick-and-mortar stores face more constraints than they do in e-commerce environments. Items in stores must be sold out by a fixed target date to provide space for new products (i.e., new modulares). Meanwhile, sufficient inventory must remain on the shelves during the markdown period (i.e., clearance period) to fulfill customer demands and avoid out-of-stock situations. In an e-commerce environment, products are simply removed from catalogues if they are sold out. Moreover, because of the operating cost of making price changes (i.e., labeling and labor costs) in a store, updates to marked-down prices are only allowed a limited number of times; however, in an e-commerce environment, price updates can be executed many times. From an algorithmic point of view, brick-and-mortar stores are a more challenging environment to train and learn when compared with the e-commerce environment. For example, the e-commerce digital environment allows the logging of data streams (e.g., item clicks or items added to a cart), which can provide the algorithms with more data about customers. Conversely, stores have mostly observational data due to the high cost of running experiments within the stores.

Influencing the sale of an item by changing its price is a classic problem that is studied extensively in the literature (Gallego and Wang 2014, Ferreira et al. 2016). One way to address the problem is to ascertain the price–volume relationship and use it to determine the new price point. However, as we mention above, markdowns in stores not only have a rigid constraint that they must be cleared from the shelves by a specified date, but the number of price changes must be limited due to operations costs. Our approach adds a time dimension to this price–volume relationship and treats it as an optimization problem. Another challenge we faced was to design a category-agnostic algorithm that could be applied to a wide range of items in Walmart’s stores and would be simple to execute. In analyzing the problem, we determined that a successful algorithmic approach to markdowns would provide the following benefits:

1. Reduced operations and labor costs by reducing the number of physical price updates (e.g., limiting the number to one or two updates).
2. Reduced unsold inventory at the end of the modular life cycle and allocation of the freed space to higher-selling items (i.e., opportunity cost).
3. Increased revenue by finding the lowest markdown rates that would achieve a product sell-through rate (STR) of 100% by the end of the markdown period (i.e., the time at which new products would be available to stock the shelves). STR is the number of units sold during the markdown period divided by the units in inventory at the beginning of the period. It is important to maintain items not sold out at the beginning of the markdown and sell them gradually until the end of the markdown period.

In the remainder of this paper, we discuss the details of the algorithm, the development and operational aspects of executing it within stores, the engineering architecture and its implementation, and key results and benefits to the end users.

Algorithmic Framework

Theoretically, the optimal markdown strategy may be a single-markdown price point that results in clearing the shelves by the specified date, but not before and not later than this date. This strategy is prompted by the knowledge that fewer price changes are easier to implement and avoid the cost of relabeling merchandise. Knowing whether the single-pricing strategy is substantially suboptimal is important, especially when mean reservation prices (i.e., the maximum prices that buyers are willing to pay for products) decline over time (Wedad and Keskinocak 2003, Gupta et al. 2006, Caro and Gallien 2012). In such cases, the next best alternative is to use a multiple-pricing strategy that results in clearing the shelves by a specified date. Hence, we proposed a multiobjective optimization

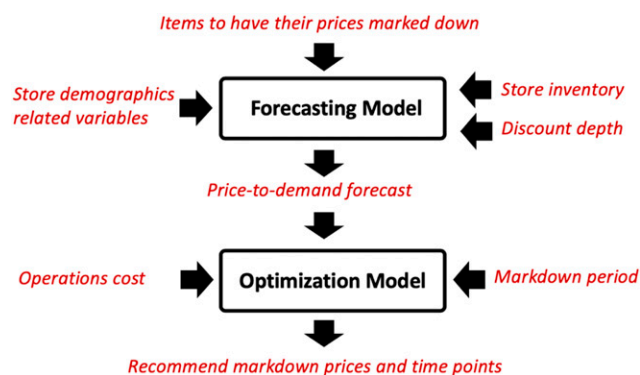
in a constrained environment that aims to maximize revenue by addressing the three benefits we list in the Why Do We Need an Algorithmic Approach to Markdowns? section.

The core building blocks of this framework are the forecasting engine and the optimization engine. The forecasting engine estimates the expected demand at various price points, including a scenario with no price change. The optimization engine recommends the final strategy (i.e., single-price markdown or multiple-price markdown) based on which of these strategies is optimal, given the objective. Figure 1 shows the high-level architecture of our framework. As we illustrate in this figure, the decision variables of the optimization model are the prices and the time points at which the price is updated. In the objective function, relabeling cost is considered as a penalty.

The Forecasting Engine

The objective of the forecasting engine is to estimate the price–demand relationship for every item. Gupta et al. (2006) tackled the problem of markdowns in a classical revenue management (RM) model by assuming that demand is either deterministic or stochastic. In a deterministic setting, the demand of an item is an exponential function of price only. In a stochastic setting, the demand of an item is an exponential function of price with some degrees of uncertainty. However, we made three improvements to apply it to our domain. (1) The demand in an RM model is a function of price; in our model, it is a function of multiple predictor variables. (2) We impose a constraint that we must keep inventory on the shelves during the markdown period. (3) The price-sensitivity parameter, the sensitivity of demand quantity to changes in price, in an RM model is market-based; we determine the price-sensitivity parameter through historical data.

Figure 1. (Color online) The Markdown Process Consists of Forecasting to Determine the Price-to-Demand Relationship and Optimization to Determine the Most Favorable Markdown Prices and Markdown Period



In our approach, we assume that demand is stochastic and a function of several variables (e.g., holidays, time of year, and store footfall), including price. To accurately estimate the price–demand relationship, we employ an ensemble approach of forecasting models that uses two models:

1. Gradient-boosting regression (Boehmke and Greenwall 2019) learns the relationship between various independent variables, including price, to estimate demand (i.e., demand is a function of price and other predictor variables where the function is learned from historical data). This model produces a prediction model in the form of an ensemble of weak prediction models (i.e., decision trees); see Spedicato et al. (2018) for more details. Its predictor variables are (a) aggregated time-series data of sales for the merchandise, including both current sales and lagged (i.e., previous period) time-series sales; (b) price-based features, such as price fluctuation from the previous week and baseline price, which include absolute differences and margins; (c) calendar-specific binary features, such as holidays, paydays, and special events; (d) categorical features, such as type of holiday or calendar event; (e) numerical features, including available inventory, current inventory level compared with inventory at the start of the markdown period; (f) taxonomy of items in the Walmart ecosystem, such as subcategory, category, department, and division of the item; and (g) transformations of price-based features, such as the logarithmic transformation of price.

2. A partial differential equation (PDE) based on the Heat equation (Cole et al. 2010) encapsulates all possible independent variables, except price and time, into a single variable k and explicitly models the relationship between inventory decreases and prices; that is, inventory is a function of price k and time, function is solution to a second-order PDE, and k is learned from historical data.

The first demand-forecasting model (using gradient boosting) generates weekly forecasts based on multiple predictor variables. This can be applicable in determining multiple price updates during the markdown period. The second model (based on the Heat equation) forecasts the demand based on a single price during the entire markdown period and looks at how on-hand inventory decreases. We use different use cases for each forecasting model.

We also considered the efficacy of forecasting the price-to-demand relationship by explicitly calculating the price elasticity. However, using elasticity has some drawbacks. For example, calculating an effective elasticity curve requires isolating the effects on demand by exclusively using pricing policies and not considering other external factors. Demand is a function of many variables, and modelling it using only price elasticity does not provide a full view of the

environment, thus, resulting in missing some factors that are important in a realistic model.

Optimization

As we illustrate in Figure 1, the decision variables of the optimization model are the prices and the price-updating time points. In the objective function, relabeling cost is considered as a penalty. Specifically, we want to determine the cases where a single-pricing markdown strategy outperforms a two-price strategy. We address these questions by introducing two methods: a single-pricing strategy and a multiple-pricing strategy.

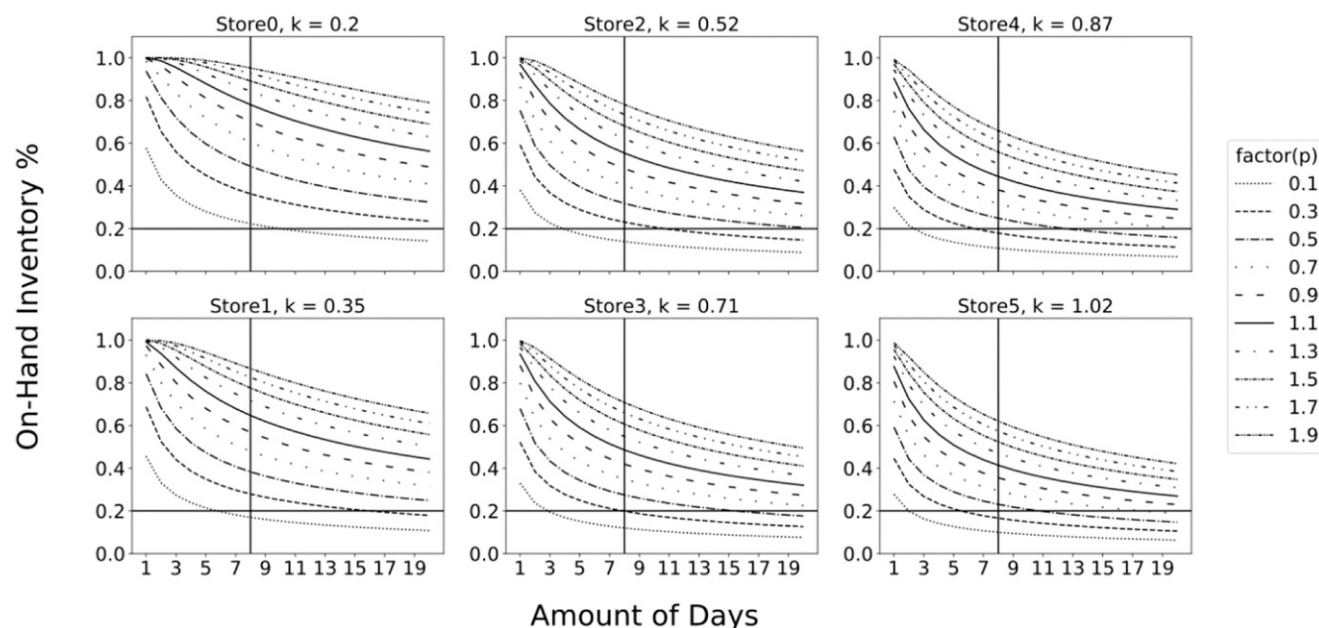
In the development phase, we considered dynamic programming (DP) (Li et al. 2020) to determine the impact of the markdown-optimization module on operating costs and revenue generation. However, we found that DP is not practical in our case because of the large amount of data involved.

Optimizing Revenue and Cost Using Only a Single-Markdown Pricing Strategy

The search for an optimal single-markdown price strategy must satisfy two constraints, as we mention above: (1) Shelves should be clear (i.e., achieving a 100% sell through of on-hand inventory by the end of markdown period is necessary); and (2) the 100% sell through should not happen prior to a prespecified duration that precedes the end of markdown period. That prespecified duration is a user-controlled parameter that can vary by product, but is typically within one week.

To obtain the price that achieves maximum expected revenue, we map the price–time inventory. We derive an equation that describes how the distribution of inventory evolves over time at a store, governing the inventory-based adoption of a markdown price, as we describe below. In this equation, the stochastic variable of price sensitivity is an important variable, which is unique to a particular store and item. Instead of assuming the price-sensitivity value, we use historical data to find the price-sensitivity k value. This value is computed by testing a full range of markdown prices for several weeks using a subset of sampled stores prior to the markdown period. In Figure 2, we simulate inventory reduction by changing the price of one product in different stores (i.e., we use different price-sensitivity markets). To find a solution that satisfies the PDE, the boundary conditions, and the initial conditions, we use the Fourier method (i.e., separation of variables). The findings from actual inventory over time with prices validate our assumption of approximating inventory using this PDE. Figure 2 shows an example using one product.

Figure 2. The Graphs Illustrate Simulations of Inventory Reductions over Time with Prices (0.1, 0.3, ... 1.9) for One Item at Store #0, ..., #5



Note. In the graphs, where $k = 0.2, 0.35, \dots, 1.02$, k is a stochastic variable that determines price sensitivity.

Appendix A includes the solution to the PDE equation and the single-markdown price-optimization formulation.

Multiple-Markdown Price Optimization

In some situations, the single-pricing strategy might not be optimal and may not guarantee that shelves are cleared by a specified date. For example, for a non-perishable, expensive item with a long markdown window, multiple price changes may be more efficient. It is increasingly difficult to forecast how demand changes during the selling season. When a markdown time window is long, other factors (e.g., holidays, weather changes, or news releases) may impact the underlying stochastic price-sensitivity value at the beginning of the markdown period. Longer markdown periods introduce additional uncertainties, and using a single price results in high error rates in estimating the final inventory. The supposed suboptimality of the first price change can be accounted for in the subsequent price drops. For specific items, whose demand varies significantly due to confounding factors that could, for example, stem from competitors and substitute products (Şen 2016), it would be more efficient to identify the amount of sales lift we can achieve at various prices and select a multiple-repricing policy. This would help us to accomplish two counterintuitive objectives at the same time—clearing inventory and maximizing sales revenue. In the academic literature, several studies have concluded that single pricing may not always be optimal

(Willemain et al. 2004, Cachon and Kök 2007, Zhang and Cooper 2009), primarily due to the decline of reservation prices over time.

By solving for the optimal single price, retailers are able to see the potential benefits from adopting more flexible pricing strategies. Retailers also receive feedback on the benefits of introducing new price points and identify redundant ones (e.g., a 25% and 30% markdown both result in the same customer response). Having a multiple-markdown pricing strategy also serves as a policy to meet the objective in case the single-pricing strategy falls short of clearing shelves by a specified date.

To solve the multiple-pricing optimization problem, we consider modeling as a Markov decision process, as studied in the literature (van Otterlo and Wiering 2012). The demand for the products can change according to the fluctuations in the intensity with which customers arrive to browse the store, the level of inventory available on the shelves, and the previous prices customers observed. The demand arises from an underlying consumer-choice model, in which the prices at which different customers decide to purchase are independent and identically distributed random variables and are not time-invariant. Gupta et al. (2006) demonstrate the approach of N -pricing opportunities with an assumption of independent demand in different periods and also provide a heuristic procedure that considers demand correlation. Wang et al. (2016) develop a markdown policy by studying the perception of customer price fairness.

Some consumers may compare their experience with others who purchase the same items. Price unfairness or inequity is perceived when different prices are quoted without reasonable explanations. Jagabathula and Rusmevichientong (2016) assume that the customers follow a two-stage choice process; customers consider the set of products with prices less than a threshold (i.e., reservation price) and choose the most preferred product from the set considered. They develop a tractable nonparametric expectation-maximization algorithm to fit the model to the aggregate transaction data and design an efficient algorithm to determine the profit-maximizing of price. Smith and Achabal (1998) and Sethi and Cheng (1997) suggest the use of a set of states of relevant information about the demand, and the transition between states is governed by a Markov chain.

With N (usually less than four) opportunities to iteratively reduce prices, we consider a finite-horizon Markov decision process (MDP) (Puterman 2014) because it is a sequential decision-making problem, which fits a clear MDP structure. An MDP is described by a finite set of states, actions, rewards, and transition probabilities between states. At each decision epoch (i.e., time), which is each week in our case, a decision maker observes the state of the system. The system state provides the decision maker with all necessary information for choosing an action (action is a new markdown price, in our case) from the set of available actions in that state (state includes all the information about the product, store, and time, in our case). Selecting an action in a state has two results: (1) The decision maker receives a reward or incurs a cost (in our case, the reward is the sales minus cost), and (2) the system evolves to a possibly different state at the next decision epoch. Both the reward and transition probabilities depend on the state and the choice of an action. As this process evolves over time, the decision maker receives a sequence of rewards. A policy provides the decision maker with a prescription for choosing this action in any possible future state; in our case, a policy is about a sequence of choosing markdown prices in any possible future state. A decision rule specifies the action to be chosen at a particular time; in our case, a decision rule specifies the markdown price to be chosen at a particular time. It may depend on the present state alone or may include all previous states and actions. A policy is a sequence of decision rules. Implementing a policy generates a sequence of rewards. The sequential decision problem is to choose a policy to maximize a function of this reward sequence. MDPs are useful for studying optimization problems solved via dynamic programming (Bertsekas 2005) and reinforcement learning (Sutton and Barto 2018). Dynamic-programming methods are well developed

mathematically; however, they require a complete and accurate model of the environment (Sutton and Barto 2018). Q-learning (Watkins 1989) is one of the most popular reinforcement-learning algorithms to solve an MDP using an action-value function. The action-value function returns the expected true value (Q-value) of an action in a state under a given policy. A true value of an action is the mean reward when that action is selected. Averaging the rewards actually received is one method used to estimate the value of an action.

We built our environment using OpenAI Gym—a toolkit for developing and comparing reinforcement-learning algorithms (Sutton and Barto 2018). It employs an underlying demand-prediction model to predict the expected difference in sell-through rate, given a markdown price and period. Under this environment, we simulate millions of episodes by initially exploring random actions and eventually following a specific policy of actions to exploit good markdown prices that yield a high cumulative reward. This allows us to avoid the cost of time and money to collect data by conducting live experiments in stores and helps us to augment the data.

By structuring the search to explore and exploit the sequence of actions given an initial state, where state includes factors such as inventory units, store, and original prices that lead to maximum reward, we estimate the optimal action-value function for all state and action (e.g., a new markdown discount considered as an action) combinations. Action-value function is the function of state-action pairs that estimate the goodness of performing a given action in a given state. The notion “goodness” here is defined in terms of future rewards that can be expected or, more precisely, in terms of expected return. As a result, we obtain the optimal policy of actions and learn this policy for each store; the output is a sequence of price changes, their markdown percentages, and the duration of the price changes.

The model starts with a randomly initialized action-value function, which yields the expected discounted reward. It starts from a state, chooses a discounted price, uses policies for all time steps, recursively iterates to determine the optimal action-value-based function using the Bellman Equation (Dixit 1990), and reaches the optimal policy. To solve the optimization problem using an MDP, a dynamic-programming technique is usually used (Sutton and Barto 2018). We note that the construction of an action-value function suffers from the well-known curse of dimensionality of dynamic programming (Bertsekas 2005). For example, suppose there are m units in the starting inventory and T periods. Then, the size of the state space is exponential in m^T . The more information related to demand that we add into the

state space, the larger the size of the state space is. To resolve this issue of high dimensionality of state space, we use a deep-learning algorithm (Double-DQN) (van Hasselt et al. 2015) to map values from the state space. Double-DQN combines Q-learning with two identical deep neural networks; the first learns to update the value function during the experience replay (Lin 1992), and the second is a copy of the last episode of the first model. The Q-value is computed by using the second model. For the experience replay, the observed state transitions are stored for some time and sampled uniformly from this memory bank to update the network. We use this deep-learning algorithm, which utilizes a convolutional neural network architecture trained with a variant of state space and action-value functions, to combine simulation and function approximations to alleviate the curse of dimensionality associated with the traditional Monte Carlo approach and to match the scale of the data available. Another advantage of using this algorithm is that it overcomes the overoptimism of Q-learning in large-scale problems, which results from the inherent estimation errors.

In summary, we use both single-pricing and multiple-pricing optimization to determine markdown prices. Before the start of the markdown period, we run both the single-pricing and multiple-pricing models and compare their estimated objective values, including revenue, label cost, and labor cost of disposal. We then select the model that provides a higher objective value (i.e., revenue minus multiple costs), which is usually the single-pricing model. During the markdown period, we monitor the actual sales automatically. If, for the reasons we explain above, the single-pricing strategy does not perform as estimated, we run the multiple-pricing model again and provide an updated price. Appendix B includes the formulation for multiple-markdown pricing optimization.

Development and Implementation

The design, implementation, and validation of this work were led by Walmart's merchandising data-science team. The application-development team working with the data-science team demonstrated to the management of both the merchandising and operations teams that using store-specific, data-driven algorithms provides results that consistently outperform the legacy rule-based policy.

A key component to winning user support was that we ran extensive experiments rather than going directly to production. Each time, we executed the prices generated by our algorithm; we compared the results with those obtained by using the rule-based policy. Our work also served as a proof of concept of applying deep reinforcement learning (deep-RL) within large organizations, such as Walmart, worldwide. We have outlined

the challenges and identified corresponding methods to overcome them in applying our end-to-end optimization framework on a large scale. Furthermore, in a complicated and dynamic market, a system with the ability to learn and adapt simultaneously provides a competitive advantage.

Developing the concept and implementing a model of the markdown process required time and effort to ensure a successful deployment and user acceptance. As we tested the algorithm in Walmart stores, we added practical constraints (e.g., a markdown-percentage threshold). We built a dashboard to help us interactively provide prices based on the inventory units on the shelf. Appendix A illustrates the practical constraint on the clearing timing period. By breaking down our analyses, we found that the performance associated with price changes differed in a few stores. Therefore, we had to determine the characteristics of each store by experimenting. As a result of these analyses, we determined that incorporating a store-clustering algorithm, while splitting stores for experimentation purposes, provided us with the capabilities to compare the results.

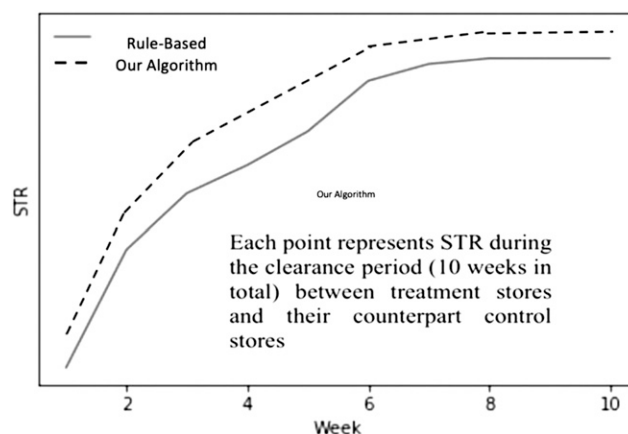
The markdown system we developed consists of four modules: store clustering, experimentation, price-demand prediction, and optimization modeling. One advantage of the system is that each module can be updated independently of the other modules. For example, price-demand prediction can be improved independently of optimization modeling. Experiments can be run both offline and online, depending on the solution options. Our online experiments allowed us to execute different prices in brick-and-mortar stores, while offline experiments used simulations that are based on historical transaction data. One of the biggest challenges to implementing the full system across all product categories was to provide a seamless end-user experience for such a complex distributed backend system. This implies that supporting some of the rule-based requirements, such as determining the stores and products to be included in a markdown decision and selecting the markdown date, had to be incorporated into the new algorithms. Many decisions that are critical to Walmart's operations had to be made under stringent time constraints. We entered these inputs (e.g., store, product to be marked down, and end date of markdown period) into our application as Apache Kafka messages. From the user interface of the application, merchants (i.e., employees responsible for making merchandising decisions) can choose to use our recommendations, the rule-based policy, or their manually created prices. To measure our results, we compared the performance of each markdown plan (i.e., a plan for the products in all product categories, including thousands of stock-keeping units (SKUs) across all associated stores, to be deleted

before a new modular refresh date) using our algorithms against those using the rule-based system. We measured the percentage of U.S. markdown plans that used our recommendations since we launched the system and found that it started with 30% of grocery products and grew to 100% of all products, except apparel.

Measurable results from our experiments and simulations helped to validate our methodology. In 2018, to determine the value of our optimization algorithm, we used deep-RL (i.e., explore-exploit) to optimize markdown strategies for food and consumable products by comparing the results of our strategies with those that used the rule-based system. Data scientists and engineers from the Walmart machine-learning group and the pricing-software-engineering group collaborated to develop a prototype experimental system. Executing the experiments related to pricing within stores presented many challenges (i.e., business constraints), which the data scientists incorporated into the experimental design. Designing the experiments to compare different pricing strategies is an example of a constraint. Customers in a brick-and-mortar store see only one price for all units of a given SKU; however, online retailers can show different customers differing prices for the same product. To resolve this challenge, we segmented the stores based on demographics and sales, as we explain in the Algorithmic Framework section, then applied the rule-based strategy to stores within each segment. We selected the salad-dressing product line in about 1,200 stores to test our algorithm; the remaining stores used the rule-based system. Our business analysis team worked to develop metrics to compare the proposed and legacy systems on a weekly basis. Initially, although the new system resulted in clearing out more inventory, we did not achieve the expected revenue results because our system provided more discounts than the rule-based system. To address this, we collected more data and corrected the Walmart internal inventory data, validating and enhancing the sales information, simulating the store inventory and the pricing system, making necessary corrections, and dividing stores into experimental and control groups based on geographical data. Our efforts in making these improvements showed that the explore-exploit framework provided significant benefits in the product departments, such as furniture, as we show in Figures 3–7. Based on the success of our results, additional merchants wanted to include their products as part of our experiments, and we included additional stores.

In the following year, 2019, we extended our experiments to all 4,769 U.S. stores and to all food and consumable-products categories to verify the robustness and reliability of the model. The feedback from the operations teams within the stores inspired the data

Figure 3. This Graph Shows the Evolution of STR as a Function of Time for Furniture Products



science team to develop the optimal single-price strategy, which would save significant label-printing costs. When a single-pricing model was ready for testing, the team had to address the business concern of whether an overdiscounted single price would result in revenue loss. To resolve this concern, we developed a program to simulate the algorithm. The simulation showed the discounted-price recommendation and the inventory that would still be on the shelves at the end of the markdown period. Beginning in September 2019, we tested and validated single-pricing optimization at a subset of the stores. Promising results convinced both the merchants and operations departments of its value. P. Mehrotra presented the results from both the single- and multiple-pricing approaches to Walmart’s executive president of U.S. operations and its U.S. chief executive officer to gain their acceptance, the backing of Walmart management, and the agreement (i.e., support) of all stakeholders. As a result, Walmart management began preparations for the development and deployment of a fully integrated optimization system. We had ongoing discussions with the merchandising department and data science teams on how to determine when to use single pricing. Our machine-learning engineering team designed the system with both the capability to rollout our optimal prices to all stores on specific categories and to run experiments for testing new features or constraints. A close partnership among the operations-research practitioners, the merchants, and the operations department was a key to our success.

Organizational Adoption of the Algorithm

Analytics can be transformative only if it is aptly adopted by the enterprise and is accepted by end users. Over the past few years, enterprises have invested millions of dollars in developing artificial intelligence and machine-learning capabilities,

but the adoption remains low (Gartner 2018). One of the fundamental reasons for this resistance is that challenges are inherent in explaining the complex algorithms (also referred to as black-box algorithms) to business experts. In a retail environment, merchants are the experts and play a key role in most of the merchandising decisions, ranging from determining the products that should be carried across channels (i.e., stores and e-commerce websites), the inventory level of each product that each store should carry, and the prices at which these products should be sold. At Walmart, merchants determine the markdown strategy, while store associates execute it (e.g., change price labels or remove unsold inventory) within the store.

As we embarked on our algorithmic journey, although we had the support of senior-level executives, we still wanted to empower the merchants by building a user interface (UI) that would provide them with full visibility to the system. Early in the project, during the experimentation phase, we worked with a few merchants to understand the business and to enhance the business constraints within the algorithm. Doing so gave us a clear idea of the metrics we would need to publish in the UI. Our software (called CMD internally) allows merchants to select and review the algorithmic recommendations for all the markdown plans before execution. Although we were confident of the performance of the personalized algorithmic recommendations, we still included an option to allow merchants to choose the rule-based policy. This flexibility and transparency reinforced the confidence of the merchants and led to an acceptance rate of around 96% for executing the markdowns our algorithm recommended in the production system; in the remaining 4% of cases, our recommended prices were not accepted for reasons such as the need to negotiate with some suppliers about specific thresholds related to markdown prices. This bottoms-up approach and the high degree of coordination among business stakeholders, data scientists, machine-learning engineers, and platform-software developers have been crucial in accelerating the early adoption of our approach at an organizational level.

Engineering Architecture

Although the system's UI is used by the corporate merchants, the pricing decisions are sent directly to each brick-and-mortar store's point-of-sale (POS) system. We developed an immutable hybrid infrastructure that integrates our optimization systems in the cloud with Walmart's merchant tools and backed up by on-premises solutions. This architecture enables the extraction and processing of data from multiple Big Data sources in the cloud and our own data centers. The overall system comprises components

developed to process markdown plans in a multiple-stage approach broadly classified as filter (i.e., sort by category to process), execute (process the data and run the algorithms), and publish (combine and send results to each store's POS system). This approach has enabled us to address the scalability needs of individual stages and the capability to dynamically tune underlying system-component resources. To accomplish efficient and timely processing, the components we implemented were loosely coupled for seamless horizontal scaling of data processing and algorithmic computation with increased resiliency to systemic disruptions. To support the growing volume of store and product combinations applicable for markdown computations, we leveraged and optimized open-source distributed computing and parallel execution frameworks, which are now seamlessly integrated with highly scalable storage and data-warehouse solutions in the cloud. We successfully tested the architecture using 68 million store and product combinations to confirm the effectiveness of the markdown process across departments.

The infrastructure alleviates the data-maintenance workload and provides flexible processing capacity. When a markdown plan is generated, the pricing strategies for all items and stores are available nationwide within five minutes. In addition, we instituted processes, such as inventory refreshes, that continue to improve the quality of the real-time data.

Benefits

Below, we list the three key business metrics that contribute to quantifying the impact on the business:

1. STR: An STR of 100% means that all items marked for clearance were sold by the end of markdown period.
2. Clean shelf: Clean shelf refers to the percentage of store-item combinations that reach 100% sell through (i.e., zero inventory) by the end of markdown period and that also have nonzero inventory on the shelf within one week of the end of the markdown period. A clean shelf of 100% means that across all store-item combinations, we were able to sell all items marked for clearance by the end of markdown period, but had inventory on the shelf at some time during the week preceding the end of the markdown period.
3. Cost: Cost is the amount we discount from the original retail price to achieve maximum STR.

Using all three metrics helps us to measure the success of our algorithms because using only one of them by itself cannot gauge success. For example, by just reducing price, we can increase the STR; however, the discount we are willing to give has an upper bound. Consider the following undesirable scenario: We can give a 99.9% discount on an item and, thus, almost certainly achieve an STR of 100%; for example,

most customers will buy an item with an original of price of \$100 when the clearance price has been reduced to 10 cents.

Figure 3 shows the STR results of using our algorithm and using the rule-based approach (25%–50%–75%) for furniture products in 2019. Figure 4 shows the average relative changes across stores in cost and STR for 20 product departments, which accounted for the majority of markdowns in 2019. In all cases, our algorithm achieved an STR that was higher or at least the same when compared with the rule-based approach; however, the cost was lower in 16 of 20 departments. We ran all our experiments in all Walmart stores in the United States, as we describe above in the Development and Implementation section. In conducting our experiments, we divided the stores such that half used our new system and half used the rule-based system. In the graphs in Figures 5, 6, and 7, we compare the algorithmic approach with Walmart's rule-based policy.

This approach, which we implemented in all 4,769 of Walmart's U.S. stores, has achieved outstanding cost-saving performance. Beginning in February 2019, we deployed our system in 60 of 70 departments. By the end of 2019, the results demonstrated that this new algorithmic approach increased the STR by a weighted average of about 21% and achieved cost reductions of about 7% on average. We also increased the average time by which we cleared the shelves on time by 15.8%. These metrics translate into millions of dollars of financial savings.

Moreover, Walmart accrued additional savings in labor, printing labels, and redeployment of labor (opportunity costs) as a result of using the single-pricing strategy. For example, printing a label and labeling a product takes an average of two minutes, and the rule-based system involves three pricing updates. Therefore, using the single-pricing strategy to mark down 25 items in a store would save a store associate ($2 \times 25 \times (3 - 1)$), or 100 minutes, which

Figure 4. The Graph Shows the Relative Difference in Cost and STR When Using Our Algorithm Instead of the Rules-Based Policy Across 20 Departments

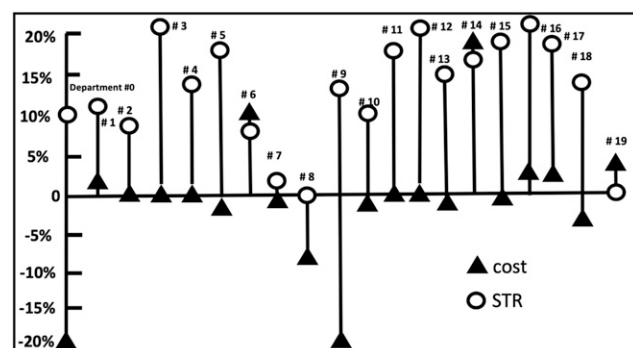
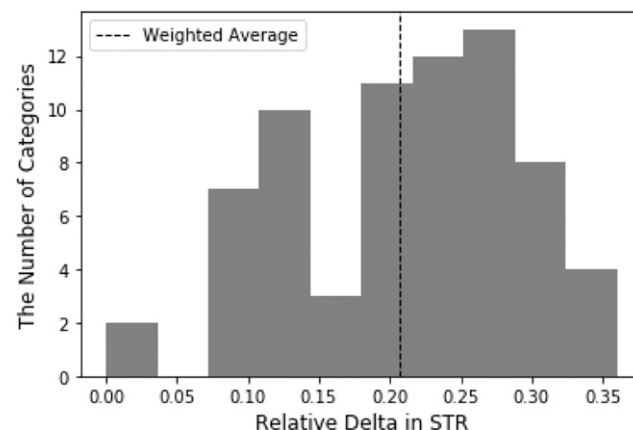


Figure 5. The Graph Shows the Distribution of Relative Difference (i.e., Delta) in STR When Using Our Algorithm Instead of the Rules-Based Policy Across 70 Categories

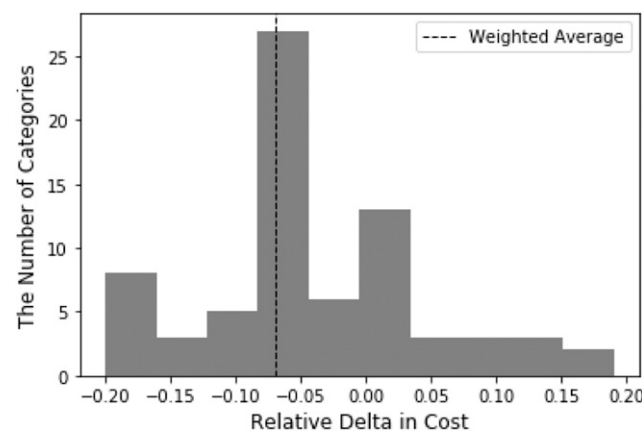


Note. The weighted average (i.e., weighted by total units of product) = 21%; that is, using our algorithm provided a STR increase of 21% over the rule-based policy.

equates to 1.6 people-hours for one store's single-markdown plan. In addition, the single-pricing strategy can save label costs for printing materials. One label costs approximately 20 cents; therefore, the printing cost savings would be $(\$0.20 \times 25 \times (3 - 1))$, or \$10 per store, for 25 marked-down items. Because Walmart marks down more than 1,000 items each month, the potential financial savings could be massive.

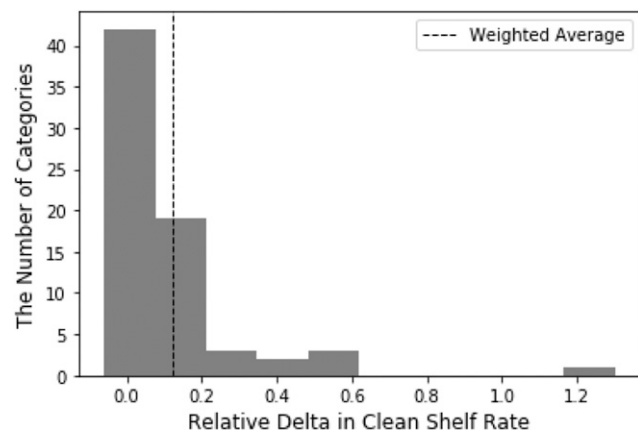
In addition to financial savings, other tangible benefits can accrue from the feedback loop resulting from markdown operations. Markdown results provide vital feedback to improve other upstream optimization models and, thus, prevent excessive stocking of inventory in stores. Figure 8 illustrates the saturation effect of sales based on static inventory.

Figure 6. The Graph Shows the Distribution of Relative Difference (i.e., Delta) in Cost When Using Our Algorithm Instead of the Rules-Based Policy Across 70 Categories



Note. The weighted average = -7%; that is, on average, using our algorithm reduced costs by 7% compared with the rules-based policy.

Figure 7. The Graph Shows the Distribution of Relative Difference (i.e., Delta) in Clean-Shelf Rates When Using Our Algorithm Instead of the Rules-Based Policy Across 70 Categories



Note. The weighted average = 15.8%; that is, on average, using our algorithm achieved a 15.8% higher clean-shelf rate compared with the rules-based policy.

This reflects the decrease of the marginal probability of a sale as limited by store inventory. As an example, see Department #6 in Figure 4. A 50% and 60% markdown result in the same sales due to limited inventory; higher cost does not result in a proportionally higher STR.

When forecasted demand is higher than actual sales because of the saturation effect, using forecasted demand to determine an optimal price is not as realistic as using an inventory-decreasing function, as we implemented in our system. Product #1 in Figure 9 illustrates a case in which assortment operations receive outlier feedback related to deviations from the markdown results; for example, a deviation might result because a product was not placed in the best position of the modular. Assortment operations in

Figure 8. The Graph Illustrates Inventory vs. Sold Units Associated with Forecasted Demand and the Saturation Effect

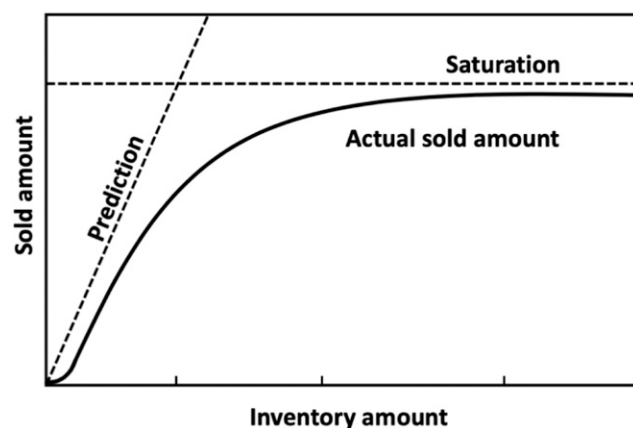
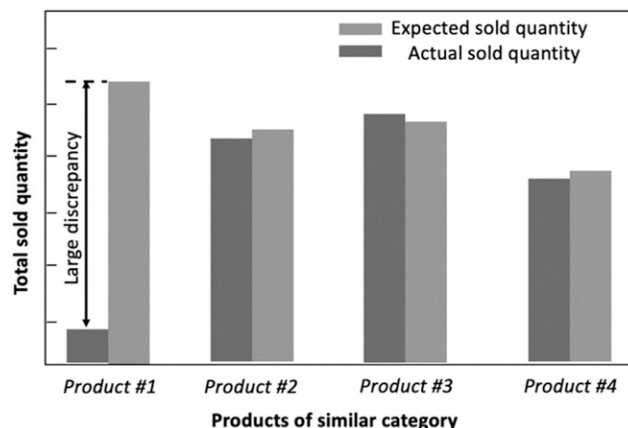


Figure 9. The Chart Illustrates a Situation in Which Our System Infers an Assortment Issue (Li et al. 2015) (i.e., Inadequate Exposure on the Shelf) from the Markdown Results



retailing involve determining the types of products that stores display in a modular for purchase by consumers, the quantity of the products available to customers, and the position of the product in a modular (Li et al. 2015). For example, consider the situation in which four products are similar, but three have similar sales patterns and one does not sell well, although its price has been deeply marked down. A potential cause could be that the product that did not sell well had been placed such that customers could not easily find it.

More predictability and reliability in markdown operations implies better planning of store labor. Store associates who are available to service customers, because they are utilizing the time saved from not having to label and remove marked-down items from the shelves, can provide better and more services for customers. Furthermore, the algorithmic approach has resulted in improving the jobs of the approximately 3,000 merchants who previously spent hours in calculating financial losses resulting from markdowns. Most merchants have become enthusiastic users of the new tool, gratefully seeing their responsibilities evolving from repetitive, manual data entry and making decisions based on their experience rather than based on data, to exceptions handling and scenario planning.

This approach has also had significant societal impact for low- and median-income families who shop at Walmart. The company uses any efficiency gains it achieves to lower the prices of its products for its customers. The efficiency gains include labor savings from removing marked-down items and labels and the salvage cost of unsold items. All these cost savings contribute to lowering the prices of household

essentials, thus enabling customers to purchase more items with the same budget.

Future Work: Extending the Algorithmic Approach for Apparel

Since we implemented our model and its associated methodology at Walmart, we have enhanced both and transformed them into a more general inventory- and pricing-optimization framework for the company's markdown business. The products to which we extended our model include apparel, a category that has large variations in demand; goods that can be significantly impacted by seasonal factors; and those that can suffer from significant spoilage if the markdown strategy is not managed properly. Future enhancements to our system include (1) a gradient-boosting forecasting model for each product based on additional underlying predictor variables, such as size, color, brand, modular type, and season; and (2) a simulation-optimization approach to optimize inventory settings for each apparel product on each day of the upcoming week. This approach can accommodate changes in sales based on customer choices and marked-down products and could reduce costs by tens of millions of dollars annually for the apparel business division. Apparel is a category that is key to future clearance sales and their associated revenues.

Acknowledgments

As of this writing Jaisankar Kanumuri, Ankit Jajoo, Saahil Agrawal, and Chandra Madhumanchi, with support from a central engineering team, are working to deploy the single-pricing markdown strategy and deep-RL. Under the executive supervision and sponsorship of Walmart Executive Vice President Scott Pleiman, the data scientist team has developed and implemented the algorithms. The authors thank the following Walmart groups, which were instrumental in making this project successful:

- Data science
- Machine-learning engineers
- Pricing-software engineering
- U.S. operations management and U.S. merchandising management
- Merchants and store managers
- Product management.

The authors thank their coaches, Ken Fordyce and Harrison Schramm, for their enthusiastic support and invaluable guidance in preparing for the Edelman Award competition.

Appendix A. Single-Markdown Price-Model Formulation

In this appendix, we describe the solution to the partial differential equation and the math formulation of our optimization model to determine a single-markdown price (p) to be used throughout the markdown duration. The objective is to maximize expected revenue, given

an initial inventory level (I_0), regular price (P), duration of the markdown time (t), and labor cost (c), to remove unsold inventory at the end of the duration; meanwhile, some inventory must remain on the shelf during the markdown until a short time (T_{\min}) prior to the arrival of the new modular.

$D(p, t)$ = demand function, which represents the demand at time t if the price is p ; p_j

I = markdown price for product j ;

I_0 = inventory at the start of the markdown period;

t = time; $t = 0$ at the start of the markdown period;

c = per-unit labor cost of disposing of unsold inventory of a given item;

T_{\min} = time duration when nonzero inventory must exist on the shelf;

I = on-hand inventory quantity (we assume no inventory is on order because replenishment ceased prior to the markdown period);

π^I = revenue less cost generated in the expression form of inventory;

π^d = revenue less cost generated in the expression form of demand;

k = price-sensitivity parameter;

t_{outdate} = duration (weeks) from the markdown starting week until the time that new products arrive for placement on a shelf;

d_{off} = price-discount maximum limit (e.g., 90% is an example of a discount's maximum limit).

For demand function $D(p_j, t)$, the profit during the markdown period can be written as follows:

$$\pi^d(p_j, t_{\text{outdate}}) = \begin{cases} p_j D(p_j, t_{\text{outdate}}) - c(I_0 - D(p_j, t_{\text{outdate}})) & \text{if } D(p_j, t_{\text{outdate}}) < I_0. \\ p_j I_0 & \text{otherwise} \end{cases} \quad (\text{A.1})$$

For on-hand inventory function $I(p, t)$, the profit during markdown can be written as:

$$\pi^I(p_j, t_{\text{outdate}}) = p_j (I_0 - I(p_j, t_{\text{outdate}})) - c * I(p_j, t_{\text{outdate}}). \quad (\text{A.2})$$

Inspired by the Black-Scholes equation in diffusion form (Hull 2018), we propose that the on-hand inventory function (I) is twice differentiable with respect to price (p) and once with respect to time (t), as shown in Equation (A.3). The proportionality stochastic variable, (k), represents sensitivity of demand to the price at a store and is unique for each store and product. Intuitively, k changes over time and has a randomness of its own.

$$\frac{\partial I}{\partial t} = k \frac{\partial^2 I}{\partial p^2}. \quad (\text{A.3})$$

Recall that the PDE has boundary conditions. The first condition in Equation (A.4) guarantees that on-hand inventory will not drop if the price goes to infinity. Equation (A.5) states the initial inventory at the beginning of the markdown period and that no replenishment will occur

once the markdown period begins. The third condition indicates that inventory will be sold out if Walmart sells the items for free. The last condition indicates that inventory will eventually be zero if time proceeds to infinity without replenishment. Let T be the set of all the week durations of the markdown period $\{1, 2, 3, 4, \dots\}$ and P be the set of all the potential price points:

$$I(\infty, t) = I_0 \text{ as } p \rightarrow \infty, \forall t \in T, \quad (\text{A.4})$$

$$I(p, 0) = I_0 \forall p \in P, \quad (\text{A.5})$$

$$I(p, t) = 0 \text{ as } p \rightarrow 0, \quad (\text{A.6})$$

$$I(p, t) = 0 \text{ as } t \rightarrow \infty. \quad (\text{A.7})$$

From Equation (A.3) and the boundary conditions from Equation (A.4) to Equation (A.7), we can derive I as follows:

$$I(p, t) = I_0 \left(\operatorname{erf} \left(\frac{p}{\sqrt{4kt}} + \varepsilon \right) \right). \quad (\text{A.8})$$

erf denotes the error function, where $\operatorname{erf} \left(\frac{p}{\sqrt{4kt}} + \varepsilon \right) = \frac{2}{\pi} \int_0^{\frac{p}{\sqrt{4kt}} + \varepsilon} e^{-x^2} dx$.

ε is the random variable that represents the error term. For each SKU (j), we solve the following optimization model: Maximize

$$p_j \left(I_0 - I_0 \left(\operatorname{erf} \left(\frac{p_j}{\sqrt{4k * t_{\text{outdate}}} + \varepsilon} \right) \right) \right) - c_j * I_0 \left(\operatorname{erf} \left(\frac{p_j}{\sqrt{4k * t_{\text{outdate}}} + \varepsilon} \right) \right), \quad (\text{A.9})$$

subject to

$$I(p_j, t_{\min}) > 0, \quad (\text{A.10})$$

$$p_j < P_j, \quad (\text{A.11})$$

$$1 - \frac{p_j}{P_j} \leq d_{\text{off}}. \quad (\text{A.12})$$

Appendix B. Multiple-Markdown Prices Model Formulation

In this appendix, we describe the math formulation of our model to determine multiple-markdown prices to be used throughout the markdown duration. We use a Markov decision process (Bertsekas 2005) to model multiple-pricing processes. At each time step, the process is in some state (s), and the decision maker may choose any action (a) that is available in state (s). The process responds at the next time step by randomly moving into a new state (s') and giving the decision maker a corresponding reward (R). The action-value function (Sutton and Barto 2018), also called a Q function, $Q\{\pi_t(s_j, a_j)\}$, tells us the value (reward) of taking an action a_j in some state s_j when following a specific policy. It is the expected return given the state and action under π_t . It specifies the goodness of choosing a particular action in a state with a policy. By structuring the search to explore and exploit the sequence of actions given an initial state, we are optimizing

$$\text{Maximize } Q\{\pi_t(s_j, a_j)\} = E_{\pi}[R_{j,t} | s_{j,t} = s_j, a_{j,t} = a_j], \quad (\text{B.1})$$

subject to

$$\sum_{a_j} \pi(s_j, a_j) = 1. \quad (\text{B.2})$$

We define the action-value function as a recursive function (Bellman's Equation) to maximize the expected revenue and minimize the related cost (Nagare and Dutta 2018):

$$Q\{\pi_t(s_j, a_j)\} = \sum_{s'_j} P(s'_j | s, a) R_{s'_j} + \gamma \sum_{a'} \pi(s', a') Q\{\pi(s'_j, a'_j)\}. \quad (\text{B.3})$$

We use the following notations:

π_t = policy-mapping states to markdown prices chosen at time t ;

$a_{j,t}$ = an allowed markdown percentage chosen from a set ranging from a minimum (e.g., 5%) to a maximum (e.g., 85%) for product j at time t ;

$Q\{\pi(s_j, a_j)\}$ = action-value function, expected discounted reward if action a_j from state s_j is performed and then follows the optimal policy;

$R_{j,t}$ = expected discounted reward obtained after transitioning from state s to state s' , which we designed to ensure that the decision-making model chooses actions that result in achieving the main objective, while also following certain intrinsic behaviors at time t ;

$s_{j,t}$ = state $s_{j,t}$, a vector of a list of feature values (e.g., available inventory, previous time step's markdown percentage, revenue, sell-through rate, number of upcoming holidays);

$P(s', r | s, a)$ = the probability of transitioning from state s to state s' with one-step dynamics taking action a (i.e., markdown percentage).

References

- Bertsekas DP (2005) *Dynamic Programming and Optimal Control*, vol. 1 (Athena Scientific, Belmont, MA).
- Boehmke B, Greenwell BM (2019) Gradient boosting. *Hands-On Machine Learning with R* (Chapman & Hall, London), 221–245.
- Cachon GP, Kök AG (2007) Implementation of the newsvendor model with clearance pricing: How to (and how not to) estimate a salvage value. *Manufacturing Service Oper. Management* 9(3):276–290.
- Caro F, Gallien J (2012) Clearance pricing optimization for a fast-fashion retailer. *Oper. Res.* 60(6):1404–1422.
- Cole K, Beck J, Haji-Sheikh A, Litkouhi B (2010) *Heat Conduction Using Green's Functions* (Taylor & Francis, Boca Raton, FL).
- Dixit AK (1990) *Optimization in Economic Theory* (Oxford University Press, Oxford, UK).
- Ferreira KJ, Lee BHA, Simchi-Levi D (2016) Analytics for an online retailer: Demand forecasting and price optimization. *Manufacturing Service Oper. Management* 18(1):69–88.
- Gallego G, Wang R (2014) Multiproduct price optimization and competition under the nested logit model with product-differentiated price sensitivities. *Oper. Res.* 62(2):450–461.
- Gartner (2018) Gartner survey shows organizations are slow to advance in data and analytics. Accessed January 30, 2020, <https://www.gartner.com/en/newsroom/press-releases/2018-02-05-gartner-survey-shows-organizations-are-slow-to-advance-in-data-and-analytics>.
- Gupta D, Hill AV, Bouzdine-Chameeva T (2006) A pricing model for clearing end-of-season retail inventory. *Eur. J. Oper. Res.* 170(2):518–540.
- Hull JC (2018) *Student Solution Manual for Options, Futures and Other Derivatives* (Pearson, New York).
- Jagabathula S, Rusmevichientong P (2016) A nonparametric joint assortment and price choice model. *Management Sci.* 63(9):3128–3145.

- Li G, Rusmevichientong P, Topaloglu H (2015) The d-level nested logit model: Assortment and price optimization problems. *Oper. Res.* 63(2):325–342.
- Li VC, Wan YW, Lin Y (2020) Dynamic programming-based heuristics for markdown pricing and inventory allocation of a seasonal product in a retail chain. *Asia-Pacific J. Oper. Res.* 37(1):1–30.
- Lin LJ (1992) Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learn.* 8(3):293–321.
- Nagare M, Dutta P (2018) Single-period ordering and pricing policies with markdown, multivariate demand and customer price sensitivity. *Comput. Indust. Engrg.* 125(November):451–466.
- Puterman ML (2014) *Markov Decision Processes: Discrete Stochastic Dynamic Programming* (John Wiley & Sons, Hoboken, NJ).
- Sen A (2016) Competitive markdown timing for perishable and substitutable products. *Omega* 64(October):24–41.
- Sethi SP, Cheng F (1997) Optimality of (s,S) policies in inventory models with Markovian demand. *Oper. Res.* 45(6):931–939.
- Smith SA, Achabal DD (1998) Clearance pricing and inventory policies for retail chains. *Management Sci.* 44(3):285–300.
- Spedicato GA, Dutang C, Petrini L (2018) Machine learning methods to perform pricing optimization. A comparison with standard GLMs. *Variance* 12(1):69–89.
- Sutton RS, Barto G (2018) *Reinforcement Learning: An Introduction* (MIT Press, Cambridge, MA).
- van Otterlo M, Wiering M (2012) Reinforcement learning and Markov decision processes. Weiring M, van Otterlo M, eds. *Reinforcement Learning: State-of-the-Art* (Springer, Berlin), 3–44.
- van Hasselt H, Guez A, Silver D (2015) Deep reinforcement learning with double Q-learning. *Proc. 30th AAAI Conf. Artificial Intelligence* (Association for the Advancement of Artificial Intelligence, Menlo Park, CA), 2094–2100.
- Walmart (2005) Our retail divisions. Accessed June 16, 2020, <https://corporate.walmart.com/newsroom/2005/01/06/our-retail-divisions>.
- Wang X, Fan ZP, Liu Z (2016) Optimal markdown policy of perishable food under the consumer price fairness perception. *Internat. J. Production Res.* 54(19):5811–5828.
- Watkins JCH (1989) Learning from delayed rewards. Doctoral dissertation, University of Cambridge, Cambridge, UK.
- Wedad E, Keskinocak P (2003) Dynamic pricing in the presence of inventory considerations: Research overview, current practices, and future directions. *Management Sci.* 49(10):1287–1309.
- Willemain TR, Smart CN, Schwarz HF (2004) A new approach to forecasting intermittent demand for service parts inventories. *Internat. J. Forecasting* 20(3):375–387.
- Zhang D, Cooper WL (2009) Managing clearance sales in the presence of strategic customers. *Production Oper. Management* 17(4):416–431.

Yixian Chen is a data scientist at Walmart Labs. She leads the clearance markdown data scientist focused on optimizing inventory and pricing. Prior to joining Walmart, she worked for the OR group at Micron Technology. She graduated from Georgia Institute of Technology, H. Milton Stewart School of Industrial Systems Engineering, specializing in OR and supply chain.

Prakhar Mehrotra is vice president of Machine Learning at Walmart, where he helps merchants take data-driven decisions. Prior to Walmart, he was head of Data Science at Uber.

He has an advanced engineer's degree in aeronautics from California Institute of Technology and a double master's in aerospace and applied math from Ecole Polytechnique and Caltech. He did undergraduate work in mechanical engineering at the National Institute of Technology, India.

Nitin Kishore Sai Samala is a senior machine learning engineer at Walmart Labs. He graduated from University of Massachusetts Amherst with a master's in computer science, specializing in NLP, ML, and deep reinforcement learning. Before grad school, he worked at Oracle as a full stack applications developer in India. He did his undergraduate work in electronics and communication engineering from BITS Pilani.

Kamilia Ahmadi is a data scientist at Walmart, where she has been involved with various projects such as assortment optimization using hyperlocal modeling, sale causal inference, and personalized push for notifying customers about clearance items in stores. Before joining Walmart, she was a PhD candidate in artificial intelligence at Utah State University.

Viresh Jivane is a machine-learning engineer at Walmart. Before joining Walmart, he worked with Intuit for three years and has completed his master's in software engineering from San Jose State University.

Linsey Pang is working as a staff data scientist in Walmart Laboratory at Sunnyvale. She has been working on multiple price-related research projects by applying various data science models. She completed her PhD from the University of Sydney in data mining and master by research from the National University of Singapore. Her research interests include data mining, outlier detection, and parallel computing.

Monika Shrivastav is strategy and operations lead at Walmart Labs. Prior to Walmart, she worked in Henry Labs. Prior to that, she was part of Nomura Investment Bank. She earned a master's in engineering management from International Technological University and a dual master's of business administration in finance and marketing from Prestige Institute of Management and Research.

Nate Lyman started his career with Walmart as a merchandise planner. Since then, he has held a variety of roles within planning prior to moving to business analytics. The roles offered development in merchandising fundamentals and then opportunities to lead associates supporting both the private brands and pricing roles. Prior to joining Walmart, he worked for the University of Arkansas following completion of his master's in applied economics.

Scott Pleiman is an executive vice president of merchandising operations at Walmart US. He has responsibility across the United States for both business-facing and centralized merchandising operations activities. He holds a master's of management from the Kellogg School of Management at Northwestern University, and he has a bachelor of science degree in accounting from Miami University.