Using Machine Learning, Business Rules, and Optimization for Flash Sale Pricing

Igor Elbert, Distinguished Data Scientist, Gilt.com
Dr. Jacob Feldman, CTO, OpenRules, Inc.
Building a Pricing System for an Online Retailer

• GILT:
  – Online retailer selling curated collections of fashion products via flash sales

• Expected Functionality:
  – Utilize sales history to predict demand for ever-changing assortments of thousands of products
  – Collaborate with business domain experts to quickly generate optimal prices that can immediately go live on site
• A combination of Machine Learning, Business Rules, and Multi-Objective Optimization:
  – Predictive Analytics
    • R, xgboost
  – Business Rules
    • OpenRules
  – Optimization
    • OpenRules/JSR-331 with various linear solvers
Before Gilt – sample sales
Gilt pioneered online “flash sales” in US
Gilt is a members-only lifestyle destination and ecommerce site that provides insider access to today’s top designer brands as well as exclusive local experiences.
9.7M+ active members

7K+ packages shipped daily

1M+ active mobile app users*

1B+ highest press impressions from a single partnership**

400 sales launch weekly

100 countries shipped to

50% of revenue is generated via mobile purchases

1.5M+ social media followers

*Gilt for iPhone, iPad & Android
**Michael Bastian x Hewlett-Packard
How to price thousands of items every day?
• Predict demand for every product in a given sale for all possible prices

• Find the best combination of prices to satisfy business objectives (weighted mix of revenue, margin, sell-through, etc)

• Present price recommendations to business
How it’s done

1. Data Prep
2. Demand Prediction
3. Price Optimization
4. Result validation
Data Prep

Operational DB

Data Warehouse

Training Data
(known demand)

Future Data
(unknown demand)

Aster,
SQL,
Map/Reduce

Operational DB

Data Prep

Data Warehouse

Training Data
(known demand)

Future Data
(unknown demand)

Aster,
SQL,
Map/Reduce

Build Model

R, xgboost

ML

Predict Demand

R, xgboost

ML

Train/Transform

Model

Generate All Possible Prices and Totals

Optimize
(Find the ‘best’ set of prices)

OpenRules

Opt

Result validation

flash sale

test/control analysis

Result validation

Price Optimization

Generate All Possible Prices and Totals

Optimize
(Find the ‘best’ set of prices)

OpenRules

Opt

Result validation

Price Optimization

Generate All Possible Prices and Totals

Optimize
(Find the ‘best’ set of prices)

OpenRules

Opt

Result validation

Price Optimization
1. Data Preparation

Data Set For Predictions

- Product Attributes
- Product Performance
- Sale Attributes

- Category
- Brand
- Color
- Material
- MSRP
- Discount
- Exposures
- Price Changes
- Initial
- Single/Multi Branded
- Duration
- Season
- Day
- Holidays
- Initial
2. Demand Prediction

Example: Predicted Demand and Revenue at different Prices
3. Price Optimization

• Goals:
  - optimize per product and per sale
  - allow business user to set goals (revenue, sell-through, margin, or combination)

• Iterate quickly
## Sample Rules

<table>
<thead>
<tr>
<th>Minimal Number of Previous Exposures</th>
<th>Variable</th>
<th>&lt;oper&gt;</th>
<th>Value</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is</td>
<td>0</td>
<td></td>
<td>20</td>
<td>Initial Sales</td>
</tr>
<tr>
<td>Is</td>
<td>Minimum Discount from MSRP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is</td>
<td>0</td>
<td></td>
<td>40</td>
<td>Repeat Sales</td>
</tr>
<tr>
<td>Is</td>
<td>Percent Difference from Original Price</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is</td>
<td>0</td>
<td></td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>Is</td>
<td>Minimal Margin Percent</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is</td>
<td>0</td>
<td></td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Is</td>
<td>Minimal Sell Through Percent</td>
<td></td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Is</td>
<td>1</td>
<td></td>
<td>20</td>
<td>Repeat Sales</td>
</tr>
<tr>
<td>Is</td>
<td>Minimum Discount from MSRP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is</td>
<td>1</td>
<td></td>
<td>40</td>
<td>Repeat Sales</td>
</tr>
<tr>
<td>Is</td>
<td>Percent Difference from Original Price</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is</td>
<td>1</td>
<td></td>
<td>30</td>
<td>Repeat Sales</td>
</tr>
<tr>
<td>Is</td>
<td>Minimal Margin Percent</td>
<td></td>
<td></td>
<td>Repeat Sales</td>
</tr>
<tr>
<td>Is</td>
<td>1</td>
<td></td>
<td>20</td>
<td>Repeat Sales</td>
</tr>
<tr>
<td>Is</td>
<td>Minimal Sell Through Percent</td>
<td></td>
<td>20</td>
<td>Repeat Sales</td>
</tr>
<tr>
<td>Is</td>
<td>10</td>
<td></td>
<td>60</td>
<td>Exit Sales</td>
</tr>
<tr>
<td>Is</td>
<td>Minimum Discount from MSRP</td>
<td></td>
<td></td>
<td>Exit Sales</td>
</tr>
<tr>
<td>Is</td>
<td>10</td>
<td></td>
<td>40</td>
<td>Exit Sales</td>
</tr>
<tr>
<td>Is</td>
<td>Percent Difference from Original Price</td>
<td></td>
<td></td>
<td>Exit Sales</td>
</tr>
<tr>
<td>Is</td>
<td>10</td>
<td></td>
<td>5</td>
<td>Exit Sales</td>
</tr>
<tr>
<td>Is</td>
<td>Minimal Margin Percent</td>
<td></td>
<td></td>
<td>Exit Sales</td>
</tr>
<tr>
<td>Is</td>
<td>10</td>
<td></td>
<td>5</td>
<td>Exit Sales</td>
</tr>
<tr>
<td>Is</td>
<td>Minimal Sell Through Percent</td>
<td></td>
<td>40</td>
<td>Exit Sales</td>
</tr>
</tbody>
</table>
# Optimization Weights

<table>
<thead>
<tr>
<th>Variable</th>
<th>&lt;oper&gt;</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gross Revenue Weight</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Gross Margin Weight</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>Gross Sell Through Weight</td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

Sample Results For A Sale:

<table>
<thead>
<tr>
<th>Target</th>
<th>Revenue</th>
<th>Margin</th>
<th>Sell-through</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max Revenue</td>
<td>$6,606</td>
<td>58%</td>
<td>23%</td>
</tr>
<tr>
<td>Max Margin</td>
<td>$4,289</td>
<td>67%</td>
<td>16%</td>
</tr>
<tr>
<td>Max Sell-through</td>
<td>$5,628</td>
<td>48%</td>
<td>24%</td>
</tr>
</tbody>
</table>
Per sale optimization

Best predictors of demand (number of units sold):

• Number of units available

• Price, Discount, MSRP

• Item price relative to the prices of other items in the sale

• Product attributes, etc

Prediction changes:
Before: predict demand for all acceptable prices
Now: same as before but for all possible totals
### Example

**Prices:** $2 or $4

<table>
<thead>
<tr>
<th>Item</th>
<th>Price</th>
<th>Total</th>
<th>Demand</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ball</td>
<td>$2</td>
<td>$3</td>
<td>4</td>
</tr>
<tr>
<td>Ball</td>
<td>$2</td>
<td>$4</td>
<td>4</td>
</tr>
<tr>
<td>Ball</td>
<td>$2</td>
<td>$5</td>
<td>4</td>
</tr>
<tr>
<td>Ball</td>
<td>$2</td>
<td>$6</td>
<td>3</td>
</tr>
</tbody>
</table>

**Prices:** $1 or $3

<table>
<thead>
<tr>
<th>Item</th>
<th>Price</th>
<th>Total</th>
<th>Demand</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pen</td>
<td>$1</td>
<td>$3</td>
<td>7</td>
</tr>
<tr>
<td>Pen</td>
<td>$1</td>
<td>$4</td>
<td>7</td>
</tr>
<tr>
<td>Pen</td>
<td>$1</td>
<td>$5</td>
<td>8</td>
</tr>
<tr>
<td>Pen</td>
<td>$1</td>
<td>$6</td>
<td>8</td>
</tr>
<tr>
<td>Pen</td>
<td>$1</td>
<td>$7</td>
<td>8</td>
</tr>
</tbody>
</table>

**Price total:** $3 - $7
- Apply constraints early
- Calculate all the totals
Multiple Knapsack Problem / Bin-packing problem

• All items must be priced

• Each item must have only one price

• Sum of all prices should equal to one and only one total
Problem definition (MathProg)

set Look;
set Price := 1..10000;
set Total := 1..100000;

set Look_Price_Total within {l in Look, p in Price, t in Total};
param price { (l,p,t) in Look_Price_Total}, >= 0, integer := p;
param demand {Look_Price_Total}, >= 0, integer;
param revenue{ (l,p,t) in Look_Price_Total} := price[l,p,t] * demand[l,p,t];

param orig_price {Look_Price_Total}, >= 0, integer, default 0;
param base_price {Look_Price_Total}, >= 0, integer, default 0;
param msrp_price {Look_Price_Total}, >= 0, integer, default 0;
param num_units_available {Look_Price_Total}, >= 0, integer, default 0;

set Unique_Total := setof{ (l,p,t) in Look_Price_Total} t;

var Use{Look_Price_Total} binary;
var Use_Total{Unique_Total} binary;

maximize Revenue: \[ \sum_{(l,p,t) \in \text{Look_Price_Total}} \text{revenue}[l,p,t] \times \text{Use}[l,p,t] \];

s.t. one_of_each{ l \in \text{Look} }: \sum_{(l,p,t) \in \text{Look_Price_Total}} \text{Use}[l,p,t] = 1;

s.t. single_total: \sum_{t \in \text{Unique_Total}} \text{Use_Total}[t] = 1;

s.t. price_sum_is_total{ t \in \text{Unique_Total} }:
\[
\sum_{(l,p,t) \in \text{Look_Price_Total}} \text{price}[l,p,t] \times \text{Use}[l,p,t] = t \times \text{Use_Total}[t];
\]

---

<table>
<thead>
<tr>
<th>Look</th>
<th>Price</th>
<th>Demand</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ball</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Ball</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Ball</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>Ball</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>Ball</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>Ball</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Ball</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Ball</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Ball</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>Ball</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>Pen</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Pen</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Pen</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Pen</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>Pen</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>Pen</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Pen</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Pen</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Pen</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>Pen</td>
<td>3</td>
<td>7</td>
</tr>
</tbody>
</table>
Modeling and Solving Real-world Problems

- We modeled the problem using OpenRules and JSR-331 Standard

- Real optimization problems consist of hundreds of thousands records:
  - We used JSR-331 Constraint Solvers to validate the problem correctness. But actual problems were too large for constraint solvers
  - We tried various JSR-331 Linear Solvers (GLPK, LP-Solve, COIN suite, SCIP, and others)
  - None was able to solve large problems in a reasonable time or at all
How We Solved the Production Problem

• OpenRules was able to create a rules-based decision model that automatically splits one large problem into a set of smaller sub-problems (one for every individual total cost)

• While there may be thousands of sub-problems, JSR-331 Linear Solvers are able to quickly solve them

• Then OpenRules decision model analyzes all found solutions to come up with the optimal solution that satisfy a configurable combined objective – a maximal combination of Revenue, Margin, and Sell-Through

• Big advantage of this approach: it can be parallelized to solve even much larger problems!
Conclusion

• We applied a combination of Machine Learning, Business Rules, and Multi-Objective Optimization to solve a real-world operational problem – flash sale price optimization.

• The pricing methodology and tools that support each of these 3 decision management techniques were readily available and quite powerful.

• However, the production-level problems required a special ingenious approach to actually solve these problems.
Questions?

Igor Elbert
ielbert@gilt.com

Jacob Feldman
jacobfeldman@openrules.com