Summary: Convex Polynomial Price Adjustment Optimization

Peter Decrem

March 28, 2025

Objective

To develop and analyze a price adjustment model for trades based on customer tier and trade DV01. The model utilizes convex and monotonic Bernstein polynomials, allowing for flexible yet controlled adjustments. The goal is to optimize the scaling of these polynomial adjustments to maximize the Profit & Loss (P&L) generated from *winning* trades, subject to a user-defined constraint on the *Losing DV01 Ratio*.

Methodology

1. **Input Data:**
   - Trade data including `customerName`, `tier`, `firmAccount`, `cusip`, `amount`, `mid` price, execution `side` (BUY/SELL), `tradePrice`, and `dv01`.
   - A per-`cusip` sensitivity factor ϵ (fixed at 0.1 in this implementation).
2. **Polynomial Modeling:**
   - Two 1-dimensional Bernstein polynomials are used as the basis for adjustments:
     - $f_1(t)$: Based on normalized customer `tier` ($tier_{\text{norm}} \in [0, 1]$).
     - $f_2(d)$: Based on normalized `dv01` ($dv01_{\text{norm}} \in [0, 1]$).
   - The polynomials are defined by their control points (coefficients) $C_k^{(1)}$ and $C_k^{(2)}$, respectively.
   - Two rescaling factors, r_1 and r_2, are introduced to globally scale the output of the base polynomials.
3. **Polynomial Fitting (Base Coefficients):**
   - The user provides target points (4 initial Z-values for each polynomial) via sliders. These define target shapes.
   - The base polynomial coefficients ($C_k^{(1)}$, $C_k^{(2)}$) are determined by solving separate optimization problems for each polynomial:
     - **Objective:** Minimize the sum of squared errors between the polynomial output and the user-defined target Z-values at the corresponding normalized X-coordinates.
     - **Constraints:**
       * *Convexity:* Enforced for both polynomials ($\Delta^2 C \geq 0$). Ensures the second derivative is non-negative.
       * *Monotonicity (Increasing):* Enforced *only* for the DV01 polynomial (f_2) ($\Delta C \geq 0$). Ensures a larger DV01 results in a larger (or equal) multiplier basis. The same method is just for Tier
     - **Solver:** CVXPY with a suitable solver (e.g., SCS).

4. **Price Adjustment Simulation:**
   - For each trade i:
     - Normalize $tier_i$ and $dv01_i$.
     - Calculate base polynomial outputs: $v_{1,\text{base}} = f_1(tier_{\text{norm},i})$, $v_{2,\text{base}} = f_2(dv01_{\text{norm},i})$.
     - Apply rescaling: $value_1 = r_1 \cdot v_{1,\text{base}}$, $value_2 = r_2 \cdot v_{2,\text{base}}$.
     - Calculate adjustment: $adjustment_i = \epsilon_i \cdot value_1 \cdot value_2 \cdot \text{sign}(\text{side}_i)$.
     - Calculate adjusted price: $price_{\text{adj},i} = mid_i + adjustment_i$.

5. **Metrics Calculation:**
   - *Win Condition:* Trade i is 'winning' if $price_{\text{adj},i} > price_{\text{trade},i}$ (BUYs), or $price_{\text{adj},i} < price_{\text{trade},i}$ (SELLs).
   - *Losing DV01 Ratio:* $\frac{\sum_{i \in \text{Losing}} dv01_i}{\sum_{\text{all } i} dv01_i}$.
   - *Potential P&L (per trade vs mid):* $pnl_{\text{pot},i} = amount_i \cdot (mid_i - price_{\text{adj},i}) \cdot \text{sign}(\text{pnl}_i)$.
   - *Actual Winning P&L:* $\sum_{i \in \text{Winning}} pnl_{\text{pot},i}$.
   - *Potential P&L (All Trades):* $\sum_{\text{all } i} pnl_{\text{pot},i}$.
   - *P&L (Favorable Adjustment):* $\sum_{i \text{ where Adj Favors}} pnl_{\text{pot},i}$, where 'Adj Favors' means $price_{\text{adj}} \leq mid$ (BUYs) or $price_{\text{adj}} \geq mid$ (SELLs).
   - *Efficiency Ratio:* $\frac{\text{Actual Winning P\&L}}{\text{Potential P\&L (All Trades)}}$.

6. **Optimization:**
   - **Goal:** Find optimal rescaling factors r_1^*, r_2^*.
   - **Objective:** Maximize Actual Winning P&L.
   - **Constraint:** $|\text{Losing DV01 Ratio} - \text{Target Ratio}| \leq \text{Tolerance}$. The Target Ratio is user-defined.
   - **Bounds:** $0.5 \leq r_1, r_2 \leq 2.0$.
   - **Solver:** `scipy.optimize.minimize` using the `COBYLA` method.

7. **Visualization (Dashboard):**
   - Interactive sliders for initial polynomial Z-values and rescaling factors (r_1, r_2).
   - Input fields for polynomial `degree` and optimization Target Ratio.
   - Plots showing initial/current PWL points, base fitted polynomial, and rescaled polynomial (on original Tier/DV01 axes).
   - Display of global metrics (Losing DV01 Ratio, various P&Ls, Efficiency).
   - Heatmaps showing Losing DV01 Ratio and Efficiency Ratio across a grid of r_1, r_2 values.
   - Comparison tables showing initial vs. current metrics aggregated by CUSIP, Tier, and Customer, with sortable delta columns.
   - Download button for the results DataFrame.

Key Formulas

- **Bernstein Basis Polynomial:**

$$B_{k,n}(t) = \binom{n}{k} t^k (1-t)^{n-k}, \quad t \in [0,1]$$

- **Polynomial Evaluation (Base):**

$$f(t_{\text{norm}}) = \sum_{k=0}^{n} C_k B_{k,n}(t_{\text{norm}})$$

- **Rescaled Values:**

$$value_1 = r_1 \sum_{k=0}^{n_1} C_k^{(1)} B_{k,n_1}(tier_{\text{norm}})$$
$$value_2 = r_2 \sum_{k=0}^{n_2} C_k^{(2)} B_{k,n_2}(dv01_{\text{norm}})$$

- **Adjustment:**

$$adjustment = \epsilon \cdot value_1 \cdot value_2 \cdot \text{sign(side)} \quad \text{where sign(BUY)} = -1, \text{sign(SELL)} = +1$$

- **Adjusted Price:**

$$price_{\text{adj}} = mid + adjustment$$

- **Convexity Constraint:**

$$C_{k+2} - 2C_{k+1} + C_k \geq 0 \quad \forall k \in \{0, ..., n-2\}$$

- **Monotonicity Constraint (Increasing):**

$$C_{k+1} - C_k \geq 0 \quad \forall k \in \{0, ..., n-1\}$$

- **Optimization Problem:**

$$\max_{0.5 \leq r_1, r_2 \leq 2.0} \left(\sum_{i \in \text{Winning}(r_1, r_2)} pnl_{\text{pot},i}(r_1, r_2) \right)$$

subject to:

$$\left| \frac{\sum_{i \in \text{Losing}(r_1,r_2)} dv01_i}{\sum_{\text{all } i} dv01_i} - \text{Target Ratio} \right| \leq \text{Tolerance}$$

Conclusion

This approach provides a structured way to model price adjustments using constrained polynomials derived from simple user inputs. It allows for analysis of trade outcomes under different scaling assumptions and enables optimization to find parameters that best meet a defined objective (maximizing winning P&L) under a quantifiable risk constraint (Losing DV01 Ratio). The interactive dashboard facilitates exploration and understanding of the model's behavior.