The Kohli-Sukumar Algorithm for Product-Line Design

1 Problem Formulation

1.1 Notation

Let:

- $\Omega = \{1, 2, \ldots, K\}$ denote the set of K attributes
- J_k denote the levels of attribute $k \in \Omega$
- $\Psi = \{1, 2, \ldots, M\}$ denote the set of M items (product profiles)
- $\Theta = \{1, 2, \ldots, I\}$ denote the set of I consumers
- w_{ijk} denote the part-worth utility of level $j \in J_k$ of attribute k for consumer i

1.2 Product Representation

Each product $m \in \Psi$ is represented by a binary matrix $X^m \in \{0, 1\}^{K \times J}$ where:

$$x_{jkm} = \begin{cases} 1 & \text{if product } m \text{ has level } j \text{ of attribute } k \\ 0 & \text{otherwise} \end{cases}$$

For the special case of binary attributes (each attribute is either present or absent), we have $|J_k| = 2$ for all k, and products can be represented as binary vectors $\boldsymbol{x}_m \in \{0, 1\}^K$.

1.3 Utility Function

The utility consumer i derives from product m is:

$$u_{im} = \sum_{k \in \Omega} \sum_{j \in J_k} w_{ijk} x_{jkm}$$

For binary attributes, this simplifies to:

$$u_{im} = \boldsymbol{w}_i^T \boldsymbol{x}_m$$

where $\boldsymbol{w}_i \in \mathbb{R}^K$ is consumer i's vector of part-worths.

2 The Kohli-Sukumar Algorithm

2.1 Single-Option Pricing

For each candidate product m, the **single-option price** p_m^* is the price that maximizes profit when product m is offered alone:

$$p_m^* = \arg\max_p \{(p - c_m) \cdot |\{i \in \Theta : u_{im} \geq p\}|\}$$

where c_m is the cost of producing product m.

Key insight: Kohli-Sukumar pre-compute these prices and keep them *fixed* throughout the algorithm.

2.2 Greedy Sequential Selection

Algorithm 1 Kohli-Sukumar Greedy Algorithm

Input: Consumer utilities $\{u_{im}\}$, costs $\{c_m\}$, max products L
Initialize: $\mathcal{S}^{(0)} = \emptyset$ (selected products)
Compute single-option prices: $p_m^* \leftarrow \arg\max_p (p - c_m) \sum_i \mathbb{I}[u_{im} \geq p]$
for $\ell = 1$ to L **do**
  Define candidate set: $\mathcal{C} = \Psi \setminus \mathcal{S}^{(\ell-1)}$
  for each candidate $m \in \mathcal{C}$ **do**
    Compute marginal profit gain:

$$\Delta\pi_m = \sum_{i \in \Theta} \mathbb{I}[\text{consumer } i \text{ switches to } m] \cdot (p_m^* - c_m)$$

    where consumer i switches if:

- $u_{im} - p_m^* > 0$ (positive utility from m)
- $u_{im} - p_m^* > \max_{n \in \mathcal{S}^{(\ell-1)}} \{u_{in} - p_n^*\}$ (better than current best)

  end for
  Select product with maximum marginal gain:

$$m^* = \arg\max_{m \in \mathcal{C}} \Delta\pi_m$$

  Update: $\mathcal{S}^{(\ell)} = \mathcal{S}^{(\ell-1)} \cup \{m^*\}$
  if $\Delta\pi_{m^*} \leq 0$ **then**
    break (no improvement possible)
  end if
end for
Return: Product line $\mathcal{S}^{(\ell)}$ and prices $\{p_m^*\}_{m \in \mathcal{S}^{(\ell)}}$

2.3 Dynamic Programming Formulation (Original Paper)

Kohli-Sukumar also describe their approach using dynamic programming notation. Let $S^*(k)$ denote the optimal partial product line after k selections:

$$S^*(k) = S^*(k-1) + [W_j(k)][1]'$$

where:

- $W_j(k)$ is the j-th column of the part-worth matrix
- $[W_j(k)][1]'$ represents the incremental contribution
- The selection criterion maximizes the weighted sum of positive elements

3 Computational Complexity

The algorithm has complexity $O(LMI)$ where:

- L = maximum number of products in the line
- M = number of candidate products ($2^K - 1$ for binary attributes)
- I = number of consumers

At each iteration ℓ:

1. Evaluate $(M - \ell + 1)$ candidates
2. For each candidate, compute utilities for I consumers
3. Select the candidate with maximum marginal profit

4 Key Properties

4.1 Fixed Pricing

Critical property: Prices are computed *once* using single-option analysis and remain **fixed** throughout the greedy selection. This differs from more sophisticated approaches that might re-optimize prices given the current menu.

4.2 Myopic Optimization

The algorithm is *myopic* - at each step, it selects the product that provides the maximum immediate profit gain without considering future interactions.

4.3 Monotonicity

The marginal profit is non-increasing: $\Delta\pi_{m^*}^{(\ell)} \geq \Delta\pi_{m^*}^{(\ell+1)}$, ensuring the algorithm terminates when no profitable addition exists.

5 Example: Binary Attributes

For products with K binary attributes:

1. **Library Generation**: Create $2^K - 1$ candidate products (excluding null product)
2. **Utility Computation**: $U = \Theta W^T$ where $\Theta \in \mathbb{R}^{I \times K}$ is the consumer part-worth matrix
3. **Single-Option Pricing**: For each product m:

$$p_m^* = \arg\max_p \left\{ (p - c_m) \cdot \frac{|\{i : \boldsymbol{w}_i^T \boldsymbol{x}_m \geq p\}|}{I} \right\}$$

4. **Greedy Selection**: Iteratively add products that maximize marginal per-capita profit