Optimization of Weekly Meals

Martina De Luca

I. Introduction

The objective of this project was to find daily recipes for users throughout the week by taking into account multiple dietary restrictions and user preferences, while minimizing the cost.

This program enables users to try a variety of new recipes that match their personal constraints without investing a lot of time, therefore simplifying the process.

The result of this project is a program that returns a recipe for each meal of the day of the week, in this case: breakfast, lunch, snack, dinner and dessert. These meals need to comply to the restrictions entered by the user, which for this project are minimum and maximum of: calories, fat, protein, sodium, vegetable portions, fruit portions and recipe rating. In addition, all meals selected need to satisfy the type of diet entered by the user. Possible types of diets the users can select from are: healthy, pescatarian, paleo, vegan, vegetarian and wheat/gluten-free, only one can be selected.

The number of constraints make the problem non-trivial, specially having to satisfy having multiple features within a specific range according to the user selection. In addition, it is important to make sure that all constraints imposed are reasonable and result in a feasible problem.

The main challenges around this project involve the data collection and cleaning processes. Since prices are values that change over time and depend on multiple factors, such as location and store, this task required merging multiple datasets to get a final list of ingredient prices. Furthermore, the list recipes obtained was not complete in all its features and some were inaccurate, which required review and improvement.

II. Formulation

A. Mathematical model

For this project the target was to minimize the overall cost of the daily recipes, therefore the objective function is the sum of the cost of each of the recipes selected by the program. For this purpose, the optimization variable x is a column vector of size n (number of recipes in the dataset) that indicates if the recipe is selected or not with a value between 0 and 1, which represents the probability. Then, the column vector c of size n contains the price for each of the recipes. Finally, the objective function will be the dot product of these two vectors, which results in the sum of the recipe prices. This can be mathematically formulated as:

$$
\begin{aligned}
&x \in \mathbb{R}^n \\
&x_i \in [0,1] \\
&c \in \mathbb{R}^n \\
&f_0(x) = c^T x
\end{aligned}
$$

For the constraints, the following variables and functions are defined:

1) Quantity of meals in a day: This constraint enforces that the program returns five meals for each day. Which mathematically translates to the following, indicating that the sum of the values of x is five. Note that since the value of $x \in [0,1]$ instead of $x \in \{0,1\}$, the result of the program are the five recipes with the highest values.

$$1^T x = 5$$

2) Diet type: This constraint enforces that all recipes chosen match the requested diet type. For this, the vector $f \in \mathbb{R}^n$ was defined, where $f_i \in \{0,1\}$, which indicates for each recipe if it matches the diet type requested (value of 1) or not (value of 0). This mathematically translates to the following, indicating that the sum of the values is five, therefore the five recipes selected are of the diet type requested.

$$f^T x = 5$$

3) Meal type: This constraint enforces that the program returns one meal of each type (breakfast, lunch, snack, dinner, dessert). For this, the matrix $T \in \mathbb{R}^{5xn}$ was defined, where $t_{ij} \in \{0,1\}$. Each row of this matrix represents one of each of the meal types and the columns the recipes. Each of the cells indicates if the recipe is of that type of meal (value of 1) or not (value of 0). The multiplication of T and x should result in a column vector of ones of size 5, indicating that there is one (and only one) recipe for each meal.

$$Tx = 1$$

4) Dietary restrictions and user preferences: This constraint enforces that the sum of the values for the features are between the lower and upper bounds requested. The features considered for this constraint are: calories, fat, protein, sodium, vegetable portions, fruit portions and user rating. For these constraints the matrix $A \in \mathbb{R}^{7xn}$ contains one row for each of the mentioned features, and the columns represent the recipes. In addition, the vector $b \in \mathbb{R}^7$ includes the upper bounds for each of

the features and similarly $d \in \mathbb{R}^7$ includes the lower bounds.

$$Ax \leq b$$
$$Ax \geq d$$

5) Non-negative values: This constraint enforces that all values returned for each recipe are non-negative.

$$x \geq 0$$

B. Optimization formulation

Based on all the variables and functions defined in the previous section, the optimization problem is formulated as the following linear function:

$$\begin{aligned} &\text{minimize } c^T x \\ &\text{subject to } 1^T x = 5 \\ &\qquad f^T x = 5 \\ &\qquad Tx = 1 \\ &\qquad Ax \leq b \\ &\qquad Ax \geq d \\ &\qquad x \geq 0 \end{aligned}$$

Transforming it into the standard form:

$$\begin{aligned} &\text{minimize } c^T x \\ &\text{subject to } 1^T x - 5 = 0 \\ &\qquad f^T x - 5 = 0 \\ &\qquad Tx - 1 = 0 \\ &\qquad Ax - b \leq 0 \\ &\qquad -Ax + d \leq 0 \\ &\qquad -x \leq 0 \end{aligned}$$

It is useful to note that this is a convex optimization problem, which we will use in the next section.

III. Numerical Studies

The datasets used were a list of recipes from Kaggle [1], which was extracted from Epicurious. This dataset contains a list of 20.052 recipes with 680 columns, which include information about nutritional facts, ingredients, event type, meal type, among others. This dataset was cleaned to keep only those recipes that had enough information for this project and in some cases information for the columns was added to increase the number of recipes in final dataset used.

Columns that were not relevant for this problem or that were empty were removed. Furthermore, some recipes (rows) were removed under the following criteria: duplicates by title, recipes missing information for the selected features, recipes with values that were higher that the daily individual recommendations, recipes without any ingredients, recipes with no meal type or with multiple meal types. As a result of the data cleaning, the data set remained with 7.274 recipes and 289 columns, which include the features mentioned above and the ingredients contained. To this data set, an additional column was added that contained the recipe estimated price based on the prices of the ingredients. These prices were obtained from multiple sources as previously mentioned [2] [3].

From this final dataset, the information was split to create the matrices and vectors corresponding to the objective and constraint functions defined in the previous sections.

For solving this problem, the method chosen was to use the package CVXPY, which is a simplified version of CVXOPT. CVXPY is a Python package to solve convex optimization problems [4]. The advantages it provides is that it doesn't require to provide the constraints in standard form and it chooses the solver that is more appropriate for the problem [5]. In this case, the solver used was ECOS (Embedded Conic Solver), which could continue to support this problem even if a larger dataset was used (up to 20k variables) [6].

Multiple combinations of inputs were tested, below a sample of the results obtained with the input detailed below. In order to use realistic values for testing the tool, this example was based on a calculator [7], which provides recommended dietary values given personal specifications, such as weight and height.

- Calories: from 1800 kcal to 2100 kcal.
- Fat: from 46g to 80g.
- Protein: from 40g to 50g.
- Sodium: from 0mg to 1500mg.
- Vegetable portions: from 2 to 5.
- Fruit portions: from 2 to 5.
- Minimum rating: 3 (maximum of 5 predefined).
- Diet choice: wheat/gluten-free.

Using these values as input, the program was run and resulted in the recipes listed in the tables below. The way the program is set up is to construct an optimization problem with the full data set and solve for the first day. After this, the recipes selected are removed and the variables adjusted to re-run the program and get the recipes for the next day. The same process is followed for the recipes for the remaining days of the week.

A. Tables of Results

Day 1:

Recipe	**x**	**Meal Type**
Yogurt with Granola	0.61	Breakfast
Potato-Parsnip Salad with Dressing	0.52	Lunch
Crispy Baby Yukon Gold Potatoes	0.87	Snack
Fresh Vegetable Pickles	0.74	Dinner
Dried Cranberry Compote	1.00	Dessert

Day 2:

Recipe	x	Meal Type
Banana Smoothie	0.97	Breakfast
Jalapeño-Glazed Chicken Breasts	0.51	Lunch
Roasted-Beet and Apple Relish	1.00	Snack
Chinese Black Rice and Salad	0.90	Dinner
Raspberry Sundaes with Sauce	0.90	Dessert

Day 3:

Recipe	x	Meal Type
Strawberry Rhubarb Smoothie	1.00	Breakfast
Sugar Snap Pea and Cabbage Slaw	0.76	Lunch
Cornbread Stuffing with Fruit	1.00	Snack
Cooked Rice	0.90	Dinner
Coconut and Lychee Sorbet	0.55	Dessert

Day 4:

Recipe	x	Meal Type
Cranberry-Orange Marmalade	1.00	Breakfast
Grilled Corn on the Cob	0.36	Lunch
Riesling-Braised Sauerkraut	0.92	Snack
Vegetables, Eggs, Anchovy Cream	0.73	Dinner
Sorbet in Grapefruit Cups	0.73	Dessert

Day 5:

Recipe	x	Meal Type
Melon,Nectarine,Grape,Plum Compote	1.00	Breakfast
Bitter Greens with Vegetables	0.59	Lunch
Cabbage and Apple Slaw with Pecans	0.55	Snack
Cauliflower Pizzas with Cheese, Kale	0.76	Dinner
Mango-Lime Ice	0.80	Dessert

Day 6:

Recipe	x	Meal Type
Spinach-Pineapple-Mint Juice	1.00	Breakfast
Shaved Asparagus with Vinaigrette	0.39	Lunch
Dried Fruits and Nuts	0.63	Snack
Quinoa Brown Rice Sushi	0.88	Dinner
Peach-Raspberry Milk Shake	0.49	Dessert

Day 7:

Recipe	x	Meal Type
Tapioca with Stewed Apples	0.52	Breakfast
Bigeye Tuna with Vinaigrette	0.67	Lunch
Cauliflower "Couscous" With Fruit	0.78	Snack
Seared Tuna with Sauce and Roast	0.88	Dinner
Orange-Spice Fruit Compote	1.00	Dessert

Summary of prices:

Day	Optimal Value	Actual Price
Day 1	11.52	6.45
Day 2	15.50	16.32
Day 3	23.79	20.48
Day 4	27.82	23.88
Day 5	30.01	35.66
Day 6	39.90	47.37
Day 7	57.04	58.35

As shown in the table above, the optimal values returned by the algorithm are not exactly the total price of the recipes. Since x is not exactly 0 and 1, the cost minimized by the solver is considering some fraction of each recipe, as we can see from the x column in each of the tables by day.

IV. Discussion and Future Directions

This project was a great way to put into practice all the material covered in the course, from start to finish. From being able to translate a problem into a formulation of an optimization problem, understand how to identify and incorporate all the necessary constraints, to then be able apply and test numerical solvers and review the solution against our initial objective. This way, it is clear to see how we can use optimization to solve multiple types of problems.

The program constructed offers a baseline solution for this problem, which can be improved in multiple ways, such as collecting more accurate values for prices and the features. Furthermore, some additional elements could be added such as possibility for the user to update ratings, prices and other values from the original dataset as the user tests the recipes and adjusts to their own experiences. Moreover, additional features could be added such as recipe difficulty or portion details. This last one would improve the program to be able to allow for batch cooking or cooking for more than one person, but would require a review of how the constraints are formulated to account for more portions and people.

References

[1] Hugo Darwood. Epicurious - Recipes with Rating and Nutrition. URL https://www.kaggle.com/datasets/hugodarwood/epirecipes.

[2] U.S. Bureau of Labor Statistics. Average Retail Food and Energy Prices, U.S. and Midwest Region. URL https://www.bls.gov/regions/mid-atlantic/data/averageretailfoodandenergyprices_usandmidwest_table.htm.

[3] Global Price Info. Food Prices in USA Grocery Stores. URL https://www.globalprice.info/en/?p=usa/food-prices-in-usa.

[4] CVXPY Documentation. URL https://www.cvxpy.org/.

[5] S. Diamond and S. Boyd. CVXPY: A Python-Embedded Modeling Language for Convex Optimization, 2016. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4927437/.

[6] A. Domahidi, E. Chu, and S. Boyd. ECOS: An SOCP solver for embedded systems. In *European Control Conference (ECC)*, pages 3071–3076, 2013.

[7] U.S. Department of Agriculture. DRI Calculator for Healthcare Professionals. URL https://www.nal.usda.gov/human-nutrition-and-food-safety/dri-calculator.