Glaze Approximations

This software produces a glaze recipe from a given list of ingredients whose Unity Molecular Formula (UMF) approximates a given target UMF.

To read this document, you will need to be familiar with basic linear algebra notation. A primer can be found here: http://www.cs.mcgill.ca/~dprecup/courses/ML/Materials/linalg-review.pdf

Let M be a matrix with n rows and p columns, where n is the number of oxides in the UMF, and p is the number of ingredients in the recipe. The (i, j)th entry of M gives, for ingredient j, the number of moles of the ith oxide there is in 100 g of that ingredient. (The columns of this matrix can be computed by taking the percentage analysis for each ingredient and dividing each percentage by the appropriate molecular weight.) Let $\mathbf{r}$ be a column vector of length p, representing a recipe where the amount of each ingredient is given in hectograms (multiples of 100 g). Then the number of moles of each oxide in recipe $\mathbf{r}$ is given by $M\mathbf{r}$.

Let $\boldsymbol{\tau}$ be a column vector of length n, where each value represents a number of moles of each corresponding oxide. We call this the *target* vector; it represents the molar amounts that we would like our recipe to have.

The *sum of squared errors* between the number of moles of different oxides contained in a recipe $\mathbf{r}$ and the number of moles of corresponding oxides in a target vector $\boldsymbol{\tau}$ is given by

$$(M\mathbf{r} - \boldsymbol{\tau})^\mathsf{T}(M\mathbf{r} - \boldsymbol{\tau}) = \mathbf{r}^\mathsf{T}M^\mathsf{T}M\mathbf{r} - 2\boldsymbol{\tau}^\mathsf{T}M\mathbf{r} + \boldsymbol{\tau}^\mathsf{T}\boldsymbol{\tau}$$

The smaller the sum of squared errors, the "closer" the molar formula for the recipe is to the target. Given this definition, we can find the recipe $\mathbf{r}$ that has the minimum sum of squared errors by solving the following optimization problem:

$$\begin{aligned} \min_{\mathbf{r}} \quad & \mathbf{r}^\mathsf{T}M^\mathsf{T}M\mathbf{r} - 2\boldsymbol{\tau}^\mathsf{T}M\mathbf{r} + \boldsymbol{\tau}^\mathsf{T}\boldsymbol{\tau} \\ \text{subject to } & \mathbf{r} \geq 0. \end{aligned}$$

The solution to this problem is a vector $\mathbf{r}^*$ that produces the smallest sum of squared errors, subject to the constraint that the amounts in the recipe are all nonnegative.

Glaze analyses are often expressed as a Unity Molecular Formula, where the molar amounts are normalized so that the total moles of the oxides that are considered fluxes sum to 1. Consider a column vector $\mathbf{f}$ of length n that has an entry for each oxide that is 1 if the oxide is considered a flux, and 0 otherwise. Then the sum of the moles of flux for a recipe $\mathbf{r}$ is given by $\mathbf{f}^\mathsf{T}M\mathbf{r}$. If we want to find the recipe whose UMF has the lowest sum of squared errors with respect to a target $\boldsymbol{\tau}$, we can solve the following modified optimization problem:

$$\begin{aligned} \min_{\mathbf{r}} \quad & \mathbf{r}^\mathsf{T}M^\mathsf{T}M\mathbf{r} - 2\boldsymbol{\tau}^\mathsf{T}M\mathbf{r} + \boldsymbol{\tau}^\mathsf{T}\boldsymbol{\tau} \\ \text{subject to } & \mathbf{r} \geq 0 \\ & \mathbf{f}^\mathsf{T}M\mathbf{r} = 1. \end{aligned}$$

This problem finds the recipe with the smallest sum of squared errors, subject to an additional constraint that the fluxes in the recipe sum to 1. This is equivalent to finding a recipe whose UMF is as close to the target as possible in terms of squared error. Both of the optimization problems above are *convex quadratic programs* that are easily solved with modern math libraries.

If $M^\mathsf{T}M$ is positive definite, then the objective above is strictly convex, and there is a unique optimal $\mathbf{r}$. If they are not, there may be multiple equally good solutions (recipes). As of December 2018, the glaze formulator uses the popular Goldfarb and Idnani algorithm for optimization, which is widely implemented. However, it requires a positive definite $M^\mathsf{T}M$ to work (and hence can only find a single unique solution) so if there is redudancy, we have to manage it outside this algorithm.

The matrix $M^\mathsf{T}M$ is positive definite only if M has full column rank (that is, the columns of M are linearly independent). Finding the column rank of M is straightfoward by using rank-revealing QR decomposition.

If M is found to not be full rank, we proceed as follows. First, we identify columns of M that are "redundant" in the sense that they are linear combinations of other columns of M. The right null space of M is the space of all recipes, allowing negative amounts, that result in zero moles of every oxide. If any such recipes exist, this means that one or more columns of M can be written as a linear combination of other columns of M. Columns of M (ingredients) that have non-zero coefficients in any recipe in the null space may be written as a linear combination of other columns, and are potentially redundant, meaning we might be able to remove that column (ingredient) and find an equally good recipe.

We use QR decomposition of M^{T} to find an orthogonal basis for the right null space of M. Then, by looking at the non-zero coefficients of the recipes in the null space, we identify which columns are potentially redundant. We remove these columns one at a time and check the rank of the reduced matrix. If it has full rank, we compute the optimal recipe and record its squared error. If it doesn't, we apply our approach recursively to the smaller matrix. Sometimes, when a "redundant" column is eliminated, the resulting solutions are not optimal because of the constraint that the final computed recipes must have nonnegative amounts. So, as a final step, we eliminate any recipes produced in this process whose squared errors are not equal to the minimum squared error found.