Winter 2021
BUS 37904
Professor Jean-Pierre Dubé
Professor Günter J. Hitsch

Advanced Quantitative Marketing: Chebyshev Approximation

Code (in your language of choice) a set of functions for multidimensional Chebyshev approximation. Below are some hints and suggestions. Some of these hints are specific to `R`. The hints are organized by four functions that initialize the data needed for Chebyshev approximation, calculate Chebyshev polynomials, estimate the Chebyshev regression coefficients, and predict approximated function values.

Goal: Approximate some function $f(\boldsymbol{x})$, $\boldsymbol{x} \in \mathbb{R}^D$, on the rectangle $[a_1, \ldots, b_1] \times \cdots \times [a_D, \ldots, b_D]$.

`initializeChebyshevApproximator`

Create the main data needed for Chebyshev approximation. Supply this function with the dimension of the problem, D, the highest degree of the polynomials used for approximation, N, the number of nodes in each dimension, M, and the bounds of the rectangle on which the function should be approximated.

1. Calculate the original M Chebyshev nodes in one dimension on the interval $[-1, 1]$, $\nu_1, \ldots, \nu_M$. See the lecture notes for the formula. Then create a $M^D \times D$ matrix $\boldsymbol{X}$ where each row represents one of the $D-$dimensional nodes in the form $(\nu_{k_1}, \ldots, \nu_{k_D})$, where each k_i is an index between 1 and M.

   Hint: in `R` there's a convenient function called `expand.grid` that creates this grid from a list of vectors. You can create such as list by first declaring `l = list()` and then using `for (k in 1:D) l[[k]] =` *`name of vector`* . Similar functions exist in other languages, too.

2. Calculate the $N+1$ Chebyshev polynomials $T_k(\nu)$ for $\nu_1, \ldots, \nu_M$. It's best to have a separate routine for this (see `calculateChebyshevPolynomials` below). Then build a $M^D \times (N+1)^D$ matrix $\boldsymbol{T}$, where each row represents all $(N+1)^T$ tensor product basis polynomials (see lecture notes). To efficiently calculate $\boldsymbol{T}$ first calculate a matrix $\boldsymbol{B}$ with element $T_k(\nu_i)$ in row i and column k ($\boldsymbol{B}$ will have M rows and $N+1$ columns). Then recursively apply the Kronecker product. Be careful about the order, i.e. whether you calculate $\boldsymbol{T} \leftarrow \boldsymbol{B} \otimes \boldsymbol{T}$ or $\boldsymbol{T} \leftarrow \boldsymbol{T} \otimes \boldsymbol{B}$. This requires some thought!

3. Package all the created data, including $\boldsymbol{X}$ and $\boldsymbol{T}$, in an appropriate container. In `R` you can use a list, in Matlab a structure, in Julia a type, and in Python or C++ you would most efficiently create a class. Also include information on the approximation problem, such as D, and include the vector of Chebyshev regression coefficients $\boldsymbol{\theta}$, initialized to some arbitrary values.

`calculateChebyshevPolynomials`

For some vector $\boldsymbol{x}$ calculate all $N+1$ Chebyshev polynomials separately for each element in $\boldsymbol{x}$.

`calculateChebyshevCoefficients`

Allow for one argument f, the function that you want to approximate. Then find the Chebyshev coefficients from the regression $f(\boldsymbol{Z}) = \boldsymbol{T}\boldsymbol{\theta}$. Hint: You can simplify and pre-compute parts of the OLS estimator. In particular, verify that the columns of $\boldsymbol{T}$ are mutually orthogonal.

`evaluateChebyshev`

Input: A vector or matrix with points $\boldsymbol{x}$ for which to evaluate the Chebyshev approximation. This will require creating the full set of Chebyshev polynomials for $\boldsymbol{x}$, $T(\boldsymbol{x})$, and then predict $T(\boldsymbol{x})^T\boldsymbol{\theta}$. To calculate $T(\boldsymbol{x})$ you may want to use the Kronecker product again. And as discussed before, the order is once again crucial.

Once you have created these approximation routines test them on the following three functions on the rectangle with lower bounds $\boldsymbol{a} = [-5, -2, -3]$ and upper bounds $\boldsymbol{b} = [2, 4, 3]$:

$$
\begin{aligned}
f(\boldsymbol{x}) &= x_1 x_3^3 + x_2 x_3 + x_1^2 x_2 x_3^2, \\
g(\boldsymbol{x}) &= x_1 \log(5 + x_2) \cdot x_3, \\
h(\boldsymbol{x}) &= x_1^2 \cos(x_2) \exp(x_3).
\end{aligned}
$$

To measure the approximation quality choose 10,000 uniform random draws on the rectangle and compare the true and approximated value functions both using the mean and max of the differences in absolute value. Compare the approximation quality for different polynomial degrees N and nodes M. $N = 3$ and $M = 7$ provide good starting points.