Tensorial Methods in Optimization

J. W. Kennington[1]

[1]HBK Capital Management

TX-LA SIAM Conference, Nov 2019

Contents

Contents

Preliminaries

Some terminology and conventions:

- *Vector Space* (denoted V) of dimension n over a field C
- C is typically either $\mathbb{R}$ or $\mathbb{C}$, elements called *scalars*
- Vector $v \in V$ represented by sequence of n scalars, called coefficients
- Vector indices "upper", e.g. $v = c_1 e^1 + \ cdot \cdot \cdot + c_n e^n = c_i e^i$ (omit $\sum$, "Einstein notation")
- Dual vector indices "lower", e.g. $f \in V^*$ s.t. $f = f_j e^j$

Brief Survey of Tensors

A *Tensor* T is essentially a collection of r vectors and s dual vectors. Two different, but equivalent definitions.

Expansion coefficients

- Programmers, practitioners
- Indexed collections of coefficients (n-dimensional arrays)
- $T = T^{i_1 \dots i_r}{}_{j_1 \dots j_s} e_{i_1} \otimes \cdots \otimes e^{j_s}$

Multilinear maps

- Mathematicians, theorists
- Functions linear in each argument (functional programming)
- $T^{i_1 \dots i_r}{}_{j_1 \dots j_s} = T(e_{i_1}, \dots, e^{j_s})$

We adopt the multilinear map convention in this talk. We can redefine vectors and dual vectors as maps, $v : V^* \to C$ and $f : V \to C$.

Tensor Operators

Two important operators: *contraction* and *product*. Most others can be reduced to compositions of these.

Tensor contraction: given a tensor ${F_{ij}}^{kl}$, we can contract a pair of indices to remove them by ${T_i}^l = {F_{ia}}^{al}$. Essentially a *sum-product* along two dims of the tensor, distributing the result across remaining dims.

Tensor product: given two tensors F_{ij} and G^{kl}, the product of the two is represented by ${T_{ij}}^{kl} = F_{ij} \otimes G^{kl}$. In map terms, given $v, w \in V$ and $f, g \in V^*$, $(v \otimes f)(g, w) \equiv v(g)f(w)$; functional currying!

Contents

Ordinary Least Squares Equation

General setup is $X\beta = y$, where

$$X = \begin{bmatrix} X_{11} & X_{12} & \dots & X_{1p} \\ X_{21} & X_{22} & \dots & X_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n1} & X_{n2} & \dots & X_{np} \end{bmatrix} \quad \beta = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix} \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \end{bmatrix}$$

Choosing a quadratic objective function $S(\beta) = |y - X\beta|^2$ and minimize

$$\underset{\beta}{\operatorname{argmin}}\, S(\beta) \implies \beta = (X^T X)^{-1} X^T y$$

Several benefits

- Pedagogically cleaner, matrices $\sim$ written equations
- Operators like "transpose" have geometric meaning (flip the array)

Longitudinal Generalization

Let's attempt a longitudinal implementation of OLS (add new dimension for time).

Simplicity of matrix equations comes at a cost:

- Generalizations are difficult, e.g. "how to add new dimension?"
- Results in clunky, suboptimal code (looping over new dim)
- Operator definitions are less clear (3D matrix inverse?)

We are left with an unfortunate choice: add an index to a matrix equation

$$\beta \implies \beta_t$$

Contents

Ordinary Least Squares Equation

Recasting the OLS equation as a tensorial expression yields

$$\beta^i = \phi({X_j}^l {X^j}_k)^i{}_k {X_l}^k y^l$$

Where $\phi \in \mathcal{L}\left(S^1_1\right)$ is a linear operator on the type $(1,1)$ tensors.

Some notes:

- ϕ usually a pairwise inversion, but there are special cases
- Despite increased index notation, still precise
- Contractions eliminate need for transposing of matrices

Longitudinal Generalization

Using tensorial expressions for OLS yields several benefits[1]:

- Adding new dimensions is easy
- Mathematical clarity maintained
- Underlying code implementation is fast[2]

Adding a new index for *time*, we obtain the following

$$y^i \to y^{ti}, X^k{}_l \to X^{tk}{}_l \quad \implies \beta^{ti} = \phi(X^t{}_j{}^i X^{tj}{}_k)^i{}_k X^t{}_l{}^k y^{tl}$$

Also be represented as a tensor product with a time tensor T^t,

$$y^{ti} = T^t \otimes y^i, \quad X^{tk}{}_l = T^t \otimes X^k{}_l$$

[1] Benefits extend to all uses of matrices in optimization

[2] Important for practitioners

Contents

Special Cases

Pseudo inverse techniques for nearly-singular matrices

- Moore-Penrose
- Hermitian matrix $\implies \phi(X) \equiv X$
- Unitary matrix $\phi(X) \equiv X^T$

Note: we assume a common basis $\mathcal{B} = \{e_i\}$ such that $e_i = (0, ..., 1, ..., 0)$ where the 1 occurs at the i^{th} coordinate.

Computational Advantages

- Multilinear-map definition:
  - allows code to use functional programming paradigms
  - minimizing memory consumption of object instances
- N-Dimensional Arrays:
  - vectorization of operations
  - boosting using GPU / TPU parallelism
  - only possible using N-Dim arrays

Contents

Risk Model Calculation

Example taken from quantitative finance, general context of model

- A set of F *risk factors*, want to be linearly neutral to each one
- A set of U *instruments*, potential portfolio constituents
- Each instrument has a *loading*, or first-order exposure, to each factor

Compute the total risk R^t of a portfolio P^{tu} by using the factor-covariance tensor C^t_{ff} and the factor loadings tensor L^{tut}

$$R^t \equiv P^t{}_u L^{tu}{}_f C^{tf}{}_f L^{tf}{}_u P^{tu}$$

References

N. Jeevanjee.
An Introduction to Tensors and Group Theory for Physicists.
Springer Science+Business Media, 2015.

S. Lang.
Algebra.
Springer, 2002.