Notes: Alternating Least Squares

Signal Data Science

This document is an exposition of the method of alternating least squares (ALS) for imputation of missing values. The reference is Hastie *et al.* (2014), Matrix Completion and Low-Rank SVD via Fast Alternating Least Squares; ALS is implemented in the `softImpute` package in R.

Theory

The operation of matrix multiplication allows us to *multiply* two matrices and form a new matrix. It is illustrated below:

$$\overset{4\times 2 \text{ matrix}}{\begin{bmatrix} a_{11} & a_{12} \\ \cdot & \cdot \\ a_{31} & a_{32} \\ \cdot & \cdot \end{bmatrix}} \overset{2\times 3 \text{ matrix}}{\begin{bmatrix} \cdot & b_{12} & b_{13} \\ \cdot & b_{22} & b_{23} \end{bmatrix}} = \overset{4\times 3 \text{ matrix}}{\begin{bmatrix} \cdot & x_{12} & x_{13} \\ \cdot & \cdot & \cdot \\ \cdot & x_{32} & x_{33} \\ \cdot & \cdot & \cdot \end{bmatrix}}$$

Figure 1: An illustration of matrix multiplication, where $x_{12} = a_{11}b_{12} + a_{12}b_{22}$ and $x_{33} = a_{31}b_{13} + a_{32}b_{23}$.

Note specifically the dimensions of the resulting matrix. If $\mathbf{A}$ is an $n \times p$ matrix and $\mathbf{B}$ is a $p \times m$ matrix, the product $\mathbf{AB}$ will have dimensions $n \times m$.[1] What if p is very small compared to n and m? We will be able to obtain quite a large matrix just from multiplying together two very narrow matrices (one tall and one wide).

In general, we find that it is possible to decompose large matrices into the *product* of multiple *smaller* matrices. This is the key behind the method of alternating least squares.

The task at hand is that given a matrix $\mathbf{X}$ with many missing entries, we want to construct a filled-in matrix $\mathbf{Z}$ which *minimizes some loss function*. It turns out that we can write a *regularized* cost function which makes this task straightforward, and also that we can write the solution which minimizes the cost function as

[1] If the matrices all have real values, we can write $\mathbf{A} \in \mathbb{R}^{n\times p}$, etc.

$$\mathbf{Z} \approx \mathbf{A}\mathbf{B}^\intercal$$

for an appropriate choice of a tall matrix $\mathbf{A}$ and a wide matrix $\mathbf{B}^\intercal$, where the operator $\intercal$ denotes the *transpose* of a matrix, (flipping a $n \times m$ matrix so that its dimensions become $m \times n$). Note that for the existing data in $\mathbf{X}$ we simply use that rating data directly in the filled-in matrix $\mathbf{Z}$ instead of the approximated values in $\mathbf{A}\mathbf{B}^\intercal$ (hence the $\approx$ symbol).

Our imputation method is an indirect one in the sense that instead of *directly* trying to calculate missing values from existing ones, we ask what the optimal filled-in matrix $\mathbf{Z}$ would look like and infer the missing values based on an analysis of $\mathbf{Z}$. Precisely, we are trying to minimize the differences between the filled-in entries of $\mathbf{X}$ and the corresponding entries of $\mathbf{A}\mathbf{B}^\intercal$ along with a regularization term controlled by a parameter λ.[2] Our cost function only considers the matrix entries which correspond to existing data (the filled-in values of $\mathbf{X}$), but the fashion in which we estimate $\mathbf{A}$ and $\mathbf{B}$ operate on the *entirety* of each matrix. Consequently, the entries of $\mathbf{A}\mathbf{B}^\intercal$ corresponding to *missing* data in $\mathbf{X}$ serve as rating estimates.

Our task is now simply to estimate the matrices $\mathbf{A}$ and $\mathbf{B}$. It turns out that the optimal estimates are related via the equation

$$\mathbf{B} = \left(\mathbf{A}^\intercal\mathbf{A} + \lambda\mathbf{I}\right)^{-1}\mathbf{A}^\intercal\mathbf{Z}$$

and vice versa with $\mathbf{A}$ and $\mathbf{B}$ switched, where λ is the regularization parameter and $\mathbf{I}$ is the identity matrix.[3] For mathematical reasons, this is actually equivalent to running a regularized[4] least squares regression for each column of $\mathbf{Z}$ with the columns of $\mathbf{A}$ as predictors, with the coefficient estimates corresponding to entries of $\mathbf{B}$![5]

As such, this suggests a strategy for estimating $\mathbf{A}$ and $\mathbf{B}$. First, we start by initializing $\mathbf{A}$. Next, we use the regression strategy described above both to

[2] One can ask the question of why we don't simply estimate $\mathbf{A} = \mathbf{B} = \mathbf{0}$ in the degenerate case of $\mathbf{X}$ having *no* missing values. If that were the result, it would contradict the fact that $\mathbf{A}\mathbf{B}^\intercal$ should be equal to the soft-thresholded SVD of $\mathbf{Z}$ (presented later in this exposition)! The reason is that although $\mathbf{A}\mathbf{B}^\intercal$ contains information about the *differences* between $\mathbf{X}$ and $\mathbf{Z}$, we don't try to impute those differences directly (in which case we might stop immediately if there were no differences whatsoever) but rather *infer* them by trying to bring $\mathbf{X}$ and $\mathbf{A}\mathbf{B}^\intercal$ closer together.

[3] The identity matrix is a matrix with 1 on the diagonal and 0 elsewhere. Multiplying it by a different matrix leaves that matrix unchanged.

[4] Specifically, this is equivalent to using Tihkonov regularized linear regression with Tikhonov matrix $\Gamma = (\lambda\mathbf{I})^{1/2}$. This is also called *ridge regression* and reduces to L^2 regularization in the case where Γ is the identity matrix. We are essentially running a linear regression of each column of $\mathbf{Z}$ with the columns of $\mathbf{A}$ as predictors and getting $\mathbf{B}$ back as the coefficient estimates.

[5] We can check that the dimensions match up. Suppose that $\mathbf{Z} \in \mathbb{R}^{n\times p}$, $\mathbf{A} \in \mathbb{R}^{n\times f}$, and $\mathbf{B} \in \mathbb{R}^{p\times f}$. Then the columns of $\mathbf{Z}$ and $\mathbf{A}$ all have n entries each, and so we can run p different linear regressions (one for each column of $\mathbf{Z}$) and get out f coefficient estimates each time. We therefore estimate $p \times f$ different coefficient estimates in total, which matches up with the dimensions of $\mathbf{B}$.

generate predictions for $\mathbf{Z}$ and to generate an estimate for $\mathbf{B}$. Next, we can switch the places of $\mathbf{A}$ and $\mathbf{B}$ in the above equation and use the same process to update $\mathbf{Z}$ and $\mathbf{A}$. We repeat in this *alternating* fashion until we achieve convergence.

Soft-thresholded SVD

After running the algorithm described above, `softImpute()` returns the imputed matrix $\mathbf{Z}$ in a *special form*. It turns out that the product $\mathbf{AB}^\intercal$ is related to $\mathbf{Z}$ in yet another fashion!

Taking a step back: in general, *all* matrices can be decomposed into a product of the form $\mathbf{UDV}^\intercal$ called the singular value decomposition (SVD) where $\mathbf{D}$ is a diagonal matrix (the only nonzero entries are on the diagonal). We can compute a *modified* version of the SVD for $\mathbf{Z}$ called the *soft-thresholded SVD* formed by taking $\mathbf{D}$ and shrinking the entries on its diagonal toward 0 by a value λ, setting an entry d_i equal to 0 if $|d_i| \leq \lambda$.[6] With the modified matrix $\mathbf{D}^\star$, we can compute the soft-thresholded SVD as $S_\lambda(\mathbf{Z}) = \mathbf{UD}^\star\mathbf{V}^\intercal$.

The connection between $\mathbf{AB}^\intercal$ and $\mathbf{Z}$ lies in the somewhat remarkable relation

$$\mathbf{AB}^\intercal = S_\lambda(\mathbf{Z})$$

for the optimal estimates of $\mathbf{A}$ and $\mathbf{B}$.

Indeed, `softImpute()` will return three matrices as `$u`, `$d`, and `$v`, corresponding to the matrices in $S_\lambda(\mathbf{Z}) = \mathbf{UD}^\star\mathbf{V}^\intercal$. From those, we also know $\mathbf{AB}^\intercal$, and so the imputed matrix $\mathbf{Z} \approx \mathbf{AB}^\intercal$ can be calculated.

Dimensionality reduction

From the definition of the soft-thresholded SVD, we see that increasing λ sufficiently high will make every value in $\mathbf{D}^\star$ equal to 0. The immediate takeaway is that by calculating the maximum value in $\mathbf{D}$, we can establish an *upper bound* for the values of λ to test. However, there is a more important and subtler interpretation of the results of ALS in connection with the regularization parameter.

It is likely that the optimal value of λ is one which drives some *but not all* of the values in $\mathbf{D}$ to 0. An $n \times n$ diagonal matrix with a *rank* of k, *i.e.*, k nonzero values on the diagonal can simply be rewritten as a $k \times k$ diagonal matrix without any nonzero values on the diagonal. Our decomposition then becomes the product

[6] Soft-thresholding is basically solving a L^1 regularized cost function very rapidly by looking at the first derivative. Refer back to the theoretical discussion in *Linear Regression: Regularization* for some related details. We can therefore think of soft-thresholded SVD as a sort of L^1 regularized version of SVD which shrinks the singular values closer to 0.

of (1) $\mathbf{U}$ (a tall $n \times f$ matrix), (2) $\mathbf{D}^\star$ (a small square $f \times f$ matrix), and (3) $\mathbf{V}^\intercal$ (a wide $f \times m$ matrix) for some small value of f. We can interpret this as being able to *summarize* both users and movies in terms of f factors, with the columns of $\mathbf{U}$ being factor scores for users and the rows of $\mathbf{V}^\intercal$ being factor scores for movies.

If a user has factor scores $\mathbf{u} = (u_1, u_2, \ldots, u_f)$, a movie has factor scores $\mathbf{m} = (m_1, m_2, \ldots, m_f)$, and the diagonal entries of $\mathbf{D}^\star$ are given by $\{d_1, d_2, \ldots, d_f\}$, then the predicted rating for that user–movie pair is simply given a weighted inner product of $\mathbf{u}$ and $\mathbf{m}$ equal to

$$\langle \mathbf{u}, \mathbf{m} \rangle = \mathbf{u}^\intercal \mathbf{D} \mathbf{m} = \sum_{i=1}^{f} u_i d_i m_i.$$