# Notes on 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.[1] We develop the theory in the context of filling in the missing values of a sparse matrix $\mathbf{X}$ representing different users' ratings of various movies, with users as rows and movies as columns.

## 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 $n \times m$.[2] 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.

[1] The details presented here are also covered in the `softImpute` vignette.

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

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 our regularized cost function.[3] It turns out that we can decompose $\mathbf{Z}$ the product of two matrices, *i.e.*,

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

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 $Z$ would look like and infer the missing values based on our analysis. Precisely, we are trying to minimize a cost function consisting of (1) the differences between the filled-in entries of $\mathbf{X}$ and the corresponding entries of $\mathbf{A}\mathbf{B}^\intercal$ along with (2) a regularization term controlled by a parameter $\lambda$.[4] 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

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

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

[3] Specifically, we minimize the expression $\frac{1}{2}\left\|P_\Omega\left(\mathbf{X}\right) - P_\Omega\left(\mathbf{A}\mathbf{B}^\intercal\right)\right\| + \frac{\lambda}{2}\left(\|\mathbf{A}\|_F^2 + \|\mathbf{B}\|_F^2\right)$, where $\Omega$ is the set of positions of $\mathbf{X}$ which do not correspond to missing values, $P_\Omega(\mathbf{X})$ denotes $\mathbf{X}$ with the positions *not* in $\Omega$ set to 0, and $\|\mathbf{X}\|_F$ denotes the Frobenius norm of $\mathbf{X}$ (the square root of the sum of squares of entries of $\mathbf{X}$).

[4] 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.

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

entries of $\hat{\mathbf{B}}$![6]

As such, this suggests a strategy for estimating $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$. First, we start by initializing $\mathbf{A}$. Next, we use the regression strategy described above both to 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 same process and use it to update $\mathbf{Z}$ and $\mathbf{A}$. We repeat in this *alternating* fashion until we achieve convergence ($\mathbf{A} \to \hat{\mathbf{A}}$ and $\mathbf{B} \to \hat{\mathbf{B}}$).

## Soft-thresholded SVD

After running the algorithm described above, `softImpute()` returns the imputed matrix $\mathbf{Z}$ in a *special form*. It turns out that when we have convergence of $\mathbf{A}$ and $\mathbf{B}$ to their optimal estimates, the product $\hat{\mathbf{A}}\hat{\mathbf{B}}^\intercal$ 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 $\lambda$, setting an entry $d_i$ equal to 0 if $|d_i| \leq \lambda$.[7] 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

$$\hat{\mathbf{A}}\hat{\mathbf{B}}^\intercal = S_\lambda(Z).$$

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 $\hat{\mathbf{A}}\hat{\mathbf{B}}^\intercal$, and so the final imputed matrix $Z \approx \hat{\mathbf{A}}\hat{\mathbf{B}}^\intercal$ can be calculated.

[6] 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 $\Gamma$ 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.

In addition, we can check that the dimensions match up. Suppose that $\mathbf{Z} \in \mathbb{R}^{n\times p}$, $\hat{\mathbf{A}} \in \mathbb{R}^{n\times f}$, and $\hat{\mathbf{B}} \in \mathbb{R}^{p\times f}$. Then the columns of $\mathbf{Z}$ and $\hat{\mathbf{A}}$ 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 $\hat{\mathbf{B}}$.

[7] 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.

## Dimensionality reduction

From the definition of the soft-thresholded SVD, we see that increasing $\lambda$ 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 $\lambda$ 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 $\lambda$ 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 of (1) $\mathbf{U}$ (a tall $n \times f$ matrix), (2) $\mathbf{D}^\star$ (a small square $f \times f$ matrix), and (3) $\mathbf{V}^\mathsf{T}$ (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 $\mathbf{V}^\mathsf{T}$ 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

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

## A simpler algorithm?

Suppose that we write $\mathbf{M} = \mathbf{A}\mathbf{B}^\mathsf{T}$, giving us the relation $\hat{\mathbf{M}} = \hat{\mathbf{A}}\hat{\mathbf{B}}^\mathsf{T}$. It's not wrong to wonder if we can simply eliminate the (somewhat unintuitive) alternating least squares part of the algorithm by simply doing the following:

1. Initialize $\mathbf{M}$.
2. Use $\mathbf{M}$ to calculate $\mathbf{Z}$.
3. Calculate the soft-thresholded SVD of $\mathbf{Z}$ and call that the new estimate of $\mathbf{M}$.
4. Repeat steps 2–3 until we have convergence of $\mathbf{M} \to \hat{\mathbf{M}}$.

The above algorithm is much more straightforward than what was described in the previous sections. However… it is *very slow* for larger matrices! The entire *raison d'être* for representing $\mathbf{M}$ as the product of two matrices and iteratively estimating those two matrices with alternating least squares is to solve the same problem in a much faster way.

The algorithm above was developed in an older paper, Mazumder and Tibshirani (2010).[8] We present it here for illustrative purposes in the hopes that it sheds more light on what ALS is trying to accomplish, because it is simpler and better illustrates the underlying connection between the singular value decomposition of $\mathbf{Z}$ and the imputation of missing values. In practice, however, it is completely superseded by ALS.

[8] Titled Spectral Regularization Algorithms for Learning Large Incomplete Matrices.