Math 480 Final Project

Steiger, Taylor
tsteiger@uw.edu

Pak, James
jimmypak@uw.edu

Spring 2013

Contents

1 Preface

Taylor and James are current undergraduates in the mathematics department at the University of Washington. This paper is their final project for their Math 480 class. Upon reviewing the elements of SAGE dedicated to SVD's, they concluded there was not the functionality included in SAGE necessary to perform rank 1 modifications to the thin SVD. Taylor and James used an article published my Matthew Brand on the mathematics behind rank 1 modifications to the thin SVD titled, "Fast low-rank modications of the thin singular value decomposition," in order to implement this new funcitonality into SAGE. This paper outlines the steps Brand took and then concludes with a section of the included functionality of SAGE created by Taylor and James.

2 Introduction

Matthew Brand created an algorithm to approximate rank-1 additive modifications to the thin SVD. This paper uses Brand's algorithm and published paper to summarize the process and implement it within SAGE. It is important to note that this paper by James and Taylor only uses the first 3 sections of Brand's paper. Brand's paper is referenced at the end of this paper.

3 SVD

Consider the real matrix $\mathbb{X}\epsilon\mathbb{R}^{pxq}$. Then the singular value decomposition (SVD) diagnolizes $\mathbb{X}$ with orthonormal matrices $\mathbb{U}\epsilon\mathbb{R}^{pxp}$ and $\mathbb{V}\epsilon\mathbb{R}^{qxq}$. Then $\mathbb{S} = \mathbb{U}^T\mathbb{X}\mathbb{V}$ is diagonal and nonnegative. This is equivalent to saying that it decomposes $\mathbb{X}$ into sums of rank-1 matrices from singular value triplets: $\mathbb{X} = \mathbb{U}\text{diag}(s)\mathbb{V}^T = \sum_i \mathbf{u}_i s_i \mathbf{v}_i^T$ for singluar values of s_i on the diagonal of $\mathbb{S}$ and singular vectors $\mathbf{u}_i$ and $\mathbf{v}_i$ taken from the columns of $\mathbb{U}$ and $\mathbb{V}$.

4 Additive Modification

Let $\mathbb{A}\epsilon\mathbb{R}^{pxc}$ and $\mathbb{B}\epsilon\mathbb{R}^{pxc}$ be arbitrary matrices of rank c and $\mathbb{X}\epsilon\mathbb{R}^{pxq}$ have rank r and economy SVD $\mathbb{U}\mathbb{S}\mathbb{V}^T = \mathbb{X}$ with $\mathbb{S}\epsilon\mathbb{R}^{rxr}$. Then we are interested in the SVD of the sum:

$$\mathbb{X} + \mathbb{A}\mathbb{B}^T = \begin{bmatrix}\mathbb{U} & \mathbb{A}\end{bmatrix}\begin{bmatrix}\mathbb{S} & 0\\ 0 & \mathbb{I}\end{bmatrix}\begin{bmatrix}\mathbb{V} & \mathbb{B}\end{bmatrix}^T$$

Furthermore, let $\mathbb{P}$ be the orthogonal basis of the column space of $(\mathbb{I} - \mathbb{U}\mathbb{U}^T)\mathbb{A}$ and let $\mathbb{R}_\mathbb{A} = \mathbb{P}^T(\mathbb{I} - \mathbb{U}\mathbb{U}^T)\mathbb{A}$.

Then

$$\begin{bmatrix}\mathbb{U} & \mathbb{A}\end{bmatrix} = \begin{bmatrix}\mathbb{U} & \mathbb{P}\end{bmatrix}\begin{bmatrix}\mathbb{I} & \mathbb{U}^T\mathbb{A}\\ 0 & \mathbb{R}_\mathbb{A}\end{bmatrix}$$

.

Let $\mathbb{Q}\mathbb{R}_{\mathbb{B}} = (\mathbb{I} - \mathbb{V}\mathbb{V}^T)\mathbb{B}$.
Then

$$\mathbb{X} + \mathbb{A}\mathbb{B}^T = \begin{bmatrix}\mathbb{U} & \mathbb{P}\end{bmatrix} \mathbb{K} \begin{bmatrix}\mathbb{V} & \mathbb{Q}\end{bmatrix}^T$$

and

$$\mathbb{K} = \begin{bmatrix}\mathbb{I} & \mathbb{U}^T\mathbb{A} \\ 0 & \mathbb{R}_{\mathbb{A}}\end{bmatrix} \begin{bmatrix}\mathbb{S} & 0 \\ 0 & \mathbb{I}\end{bmatrix} \begin{bmatrix}\mathbb{I} & \mathbb{V}^T\mathbb{B} \\ 0 & \mathbb{R}_{\mathbb{B}}\end{bmatrix}^T = \begin{bmatrix}\mathbb{S} & 0 \\ 0 & 0\end{bmatrix} + \begin{bmatrix}\mathbb{U}^T\mathbb{A} \\ \mathbb{R}_{\mathbb{A}}\end{bmatrix} \begin{bmatrix}\mathbb{V}^T\mathbb{B} \\ \mathbb{R}_{\mathbb{B}}\end{bmatrix}^T$$

.

Diagonalizing $\mathbb{K}$ as $\mathbb{U}'^T\mathbb{K}\mathbb{V}' = \mathbb{S}'$ gives the rotations $\mathbb{U}'$ and $\mathbb{V}'$ of the extended subspaces $\begin{bmatrix}\mathbb{U} & \mathbb{P}\end{bmatrix}$ and $\begin{bmatrix}\mathbb{V} & \mathbb{Q}\end{bmatrix}$ such that

$$\mathbb{X} + \mathbb{A}\mathbb{B}^T = \left(\begin{bmatrix}\mathbb{U} & \mathbb{P}\end{bmatrix} \mathbb{U}'\right) \mathbb{S}' \left(\begin{bmatrix}\mathbb{V} & \mathbb{Q}\end{bmatrix} \mathbb{V}'\right)^T$$

is the desired SVD.

5 Rank-1 Modification

For the SVD of $\mathbb{U}\$\mathbb{V}^T + \mathbb{A}^T$ with vectors $\mathbf{a}\epsilon\mathbb{R}^{\mathbf{p}}$ and $\mathbf{b}\epsilon\mathbb{R}^{\mathbf{q}}$, the equation:

$$\begin{bmatrix}\mathbb{U} & \mathbb{A}\end{bmatrix} = \begin{bmatrix}\mathbb{U} & \mathbb{P}\end{bmatrix} \begin{bmatrix}\mathbb{I} & \mathbb{U}^T\mathbb{A} \\ 0 & \mathbb{R}_{\mathbb{A}}\end{bmatrix}$$

can be changed in a partial step by the modified Gram-Schmidt algorithm. These alterations are expressed as:

$$\mathbf{m} \doteq \mathbb{U}^{\mathbf{T}}\mathbf{a}; \quad \mathbf{p} \doteq \mathbf{a} - \mathbb{U}\mathbf{m}; \quad R_a = \|\mathbf{p}\|; \quad \mathbb{P} = {R_a}^{-1} \cdot \mathbf{p}$$

.

Similarly,

$$\mathbf{m} \doteq \mathbb{V}^{\mathbf{T}}\mathbf{b}; \quad \mathbf{q} \doteq \mathbf{b} - \mathbb{V}\mathbf{n}; \quad R_b = \|\mathbf{q}\|; \quad \mathbb{Q} = {R_b}^{-1} \cdot \mathbf{q}$$

Expresses the common operations on the last column or on all columns expressed as rank-1 modifications of an SVD $\mathbb{U}\$\mathbb{V}^T = \mathbb{X}$ to give $\mathbb{U}'\$'\mathbb{V}'^T = \mathbb{X} + \mathbf{a}\mathbf{b}^{\mathbf{T}}$

Operation	*Known*	*Desired*	$\mathbf{a}$	$\mathbf{b}^{\mathbf{T}}$
Update	$\mathbb{U}\$\begin{bmatrix}\mathbb{V}^T & 0\end{bmatrix} = \begin{bmatrix}\mathbb{X} & 0\end{bmatrix}$	$\mathbb{U}'\$'\mathbb{V}'^T = \begin{bmatrix}\mathbb{X} & \mathbf{c}\end{bmatrix}$	$\mathbf{c}$	$\begin{bmatrix}0, \ldots, 0, 1\end{bmatrix}$
Downdate	$\mathbb{U}\$\mathbb{V}^T = \begin{bmatrix}\mathbb{X} & \mathbf{c}\end{bmatrix}$	$\mathbb{U}'\$'\mathbb{V}'^T = \mathbb{X}$	**-c**	$\begin{bmatrix}0, \ldots, 0, 1\end{bmatrix}$
Revise	$\mathbb{U}\$\mathbb{V}^T = \begin{bmatrix}\mathbb{X} & \mathbf{c}\end{bmatrix}$	$\mathbb{U}'\$'\mathbb{V}'^T = \begin{bmatrix}\mathbb{X} & \mathbf{d}\end{bmatrix}$	**d-c**	$\begin{bmatrix}0, \ldots, 0, 1\end{bmatrix}$
Recenter	$\mathbb{U}\$\mathbb{V}^T = \mathbb{X}$	$\mathbb{U}'\$'\mathbb{V}'^T = \mathbb{X}\,(\mathbb{I} - \frac{1}{q}\mathbf{1}\mathbf{1}^T)$	$-\frac{1}{q}\mathbb{X}\mathbf{1}$	$\mathbf{1}^T \doteq \begin{bmatrix}1, \ldots, 1\end{bmatrix}$

The equation:

$$\mathbb{K} = \begin{bmatrix}\mathbb{S} & \mathbf{0} \\ \mathbf{0} & 0\end{bmatrix} + \begin{bmatrix}\mathbf{m} \\ \|\mathbf{p}\|\end{bmatrix} \begin{bmatrix}\mathbf{n} \\ \|\mathbf{q}\|\end{bmatrix}^T$$

gets simplified to a diagonal + rank-1 matrix. Table 1 shows that specializations of the above equation are the various cases of updating, downdating, and revising individual columns of the SVD.

6 Reference

All work within the above sections is taken from Matthew Brand's article on "Fast low-rank modifications of the thin singular value decomposition."

7 Implementation in SAGE

```
r"""
Updates to the thin SVD using NumPy.

This function is a SAGE replication of Matthew Brand's article on "Fast low-rank modifications of the
thin singular value decomposition." <http://www.stat.osu.edu/~dmsl/thinSVDtracking.pdf>
This function is an approximation to the true thin SVD, therefore, no tests are provided.

AUTHORS:

- Taylor Steiger, James Pak (2013-06-10): initial version


EXAMPLES::

    Update

    sage: X = np.array([[1.0,2.0,3.0,4.0],[3.0,2.0,5.0,5.0],[5.0,3.0,1.0,1.0],[7.0,7.0,7.0,7.0]])
    sage: U, s, V = np.linalg.svd(X, full_matrices = False)
    sage: a = np.reshape(np.array([4.0,5.0,1.0,7.0]), (-1, 1))
    sage: U, S, V = svd_update(U, np.diag(s), V, X, a, update = True)

    Downdate

    sage: X = np.array([[1.0,2.0,3.0,4.0],[3.0,2.0,5.0,5.0],[5.0,3.0,1.0,1.0],[7.0,7.0,7.0,7.0]])
    sage: U, s, V = np.linalg.svd(X, full_matrices = False)
    sage: U, S, V = svd_update(U, np.diag(s), V, X, downdate = True)

    Revise

    sage: X = np.array([[1.0,2.0,3.0,4.0],[3.0,2.0,5.0,5.0],[5.0,3.0,1.0,1.0],[7.0,7.0,7.0,7.0]])
    sage: U, s, V = np.linalg.svd(X, full_matrices = False)
    sage: a = np.reshape(np.array([4.0,5.0,1.0,7.0]), (-1, 1))
    sage: U, S, V = svd_update(U, np.diag(s), V, X, a)

    Recenter

    sage: X = np.array([[1.0,2.0,3.0,4.0],[3.0,2.0,5.0,5.0],[5.0,3.0,1.0,1.0],[7.0,7.0,7.0,7.0]])
    sage: U, s, V = np.linalg.svd(X, full_matrices = False)
    sage: U, S, V = svd_update(U, np.diag(s), V, X)

"""

#*****************************************************************************
#       Copyright (C) 2013 Taylor Steiger <tsteiger@uw.edu>
#       Copyright (C) 2013 James Pak <jimmypak@uw.edu>
#
#  Distributed under the terms of the GNU General Public License (GPL)
#  as published by the Free Software Foundation; either version 2 of
#  the License, or (at your option) any later version.
#                  http://www.gnu.org/licenses/
#*****************************************************************************

import numpy as np

def svd_update(U, S, V, X, c = None, update = False, downdate = False):
    """
    INPUT:

    - U -- a (nxn) matrix containing singular vectors of X.

    - S -- a (nxn) diagonal matrix containing singular values. the ith diagonal entry is the singular
value corresponding to the ith column of U.

    - V -- a (nxn) matrix containing singular vectors of X.

    - X -- a (mxn or nxm) matrix such that U^T*X*V=S.

    - c -- (default: None) a column vector for revision or update of decomposition.

    - update -- (default: False) boolean whether to add c to the decomposition. If true, c must also
be provided.

    - downdate -- (default: False) boolean whether to downdate the decomposition.

    OUTPUT:

    A 3-tuple consisting of matrices in this order:

    1. Transformed U.
    2. Transformed S.
    3. Transformed V.
```

ALGORITHM:

The SVD rank-1 modification algorithm is described by Matthew Brand
in the paper at, <http://www.stat.osu.edu/~dmsl/thinSVDtracking.pdf>.
The algorithm works as follows:

#. Extend V so that both its last column and row are the zero vector.
#. Compute a and b so that they perform the appropriate transformation.
#. If updating:
#. a = c, b^T = [0,...,0,1]
#. Else if downdating:
#. a = X[:,-1], b^T = [0,...,0,1]
#. Else if revising:
#. a = X[:,-1] - c, b^T = [0,...,0,1]
#. Else: #recentering
#. a = X * (I – (1/q) * (l * l^T)) # l = [1,...,1]
#. Compute m, p, Ra and P.
#. m = U^T * a
#. p = a - U * m
#. Ra = ||p||
#. P = Ra^(-1) * p
#. Compute n, q, Rb and Q, similarly to the previous step with substitution of b for a.
#. Compute K.
#. K = [S 0] * [m] *[n]^T
#. [0 0] [Ra] [Rb]
#. Diagonalize K.

WARNING::

During testing, this code showed strong deviations, since it is an approximation, from
the true thin SVD of the matrix X. However, testing was conducted with a small matrix X,
and may be working perfectly fine.
"""

```
V = np.vstack([V, np.zeros(V.shape[1])])
if down or type(c) == type(np.array([])):
    b = np.zeros(V.shape[0])
    b[-1] = 1
    b = np.reshape(b, (b.shape[0], 1))
    if down:
        a = np.reshape(np.multiply(X[:,-1], -1), (-1, 1))
    elif add:
        a = np.reshape(c, (-1, 1))
    else:
        a = np.reshape(X[:,-1] - c, (-1, 1))
else:
    ones = np.zeros(V.shape[0])
    ones = np.add(b, 1)
    b = np.reshape(ones, (-1, 1))
    a = np.reshape(np.multiply((-1/X.shape[1]), np.dot(X, b)), (-1, 1))

m = np.reshape(np.dot(np.transpose(U), a), (-1, 1))
p = np.reshape(a - np.dot(U, m), (-1, 1))
Ra = np.linalg.norm(p)
P = np.reshape(np.multiply((1 / Ra), p), (-1, 1))
n = np.reshape(np.dot(np.transpose(V), b), (-1, 1))
q = b - np.dot(V, n)
Rb = np.linalg.norm(q)
Q = np.reshape(np.multiply((1 / Rb), q), (-1, 1))

k = S
K = np.zeros((k.shape[0] + 1, k.shape[0] + 1))
K[:-1,:-1] = k
stack = np.vstack(np.append(m, Ra))
t = np.reshape(np.append(n, Rb), (1, -1))
dot = np.dot(stack, t)
K = np.add(K, dot)

D, P = np.linalg.eig(K)

return (np.transpose(np.linalg.inv(P)), np.diag(D), P)
```