Singular Value Decomposition

Johan Astborg

Computational Programming with Python

Project Presentation, May 2015

Outline

1. Motivation
2. Theory
3. The project

Motivation

Matrix decomposition

Consider the matrix A

$$A = \begin{bmatrix} 4 & 11 & 14 \\ 8 & 7 & -2 \end{bmatrix}. \tag{1}$$

Because A is *singular*, the following eigendecomposition

$$A = VDV^T \tag{2}$$

is inapplicable[1].

[1] Clearly, since A is non-invertible

Motivation

Matrix decomposition cont.

Instead there is a non-trivial possibility, using the absolute values of the eigenvalues instead

$$A = UDV^*. \tag{3}$$

These are the *singular values* of A, where A^* is *conjugate transpose*.

Motivation

Geometric representation

Here will be a figure of the SVD.

Power Utility Portfolio Choice

- Portfolio Return

$$\begin{aligned} \underset{\mathbf{w}_t}{\text{maximise}} \quad & \boxed{\mathbf{w}_t \cdot \bar{\mathbf{r}}_t} + \frac{1-\gamma}{2} \boxed{\mathbf{w}_t \cdot (\Sigma_t \mathbf{w}_t)} \\ \text{subject to} \quad & \mathbf{w}_t \cdot \boldsymbol{\iota} = 1 \\ & \mathbf{w}_t \geq 0 \end{aligned}$$

- Portfolio Variance

Motivation

Geometric representation

$$\begin{aligned}
\sigma_1 &= \sqrt{360} = 6\sqrt{10} \\
\sigma_2 &= \sqrt{90} = 3\sqrt{10} \\
\sigma_3 &= 0
\end{aligned}$$

Motivation

Geometric interpretation of singular values

First two singular values of A are the length of the semiaxis of the ellipse (Fig 2).

Definition

Singular Value Decomposition

$$\Sigma = \begin{bmatrix} D & 0 \\ 0 & 0 \end{bmatrix}. \tag{4}$$

Theorem

Singular Value Decomposition

$$A = U \Sigma V^* \tag{5}$$

where U, V are orthogonal and its positive diagonal entries is called the *SVD* of A.

Proof

Singular Value Decomposition

... since V is orthogonal matrix, $U\Sigma V^* = AUV^T = A$

The Project

Singular Value Decomposition

Implement this in Python given constraints and a set of tasks.

The Project

Singular Value Decomposition

In theory we can follow the following steps

1. Find the orthogonal dianonalization of $A^T A$
2. Set up V and Σ
3. Construct U
4. Check singular values against eigenvalues ($||Av_i|| = \sigma_i$)

Reality check

Singular Value Decomposition

But numerical linear algebra is reality

...which means *IEEE-754* and 64-bit FPUs.

Reality check

Singular Value Decomposition

you've seen this *too many times* but here it is again

The Project

Requirements

Requirements for the project

1. 1
2. 2
3. 3
4. 4

The Project

Overview of numerical algorithms

Overview of algorithms for SVD implementation

1. 1
2. 2
3. 3
4. 4

The Project

The algorithm used

Motivation:
Suggested in the project description[2]

[2] checked some references anyhow

The Project
Implementation

How it was done

1. Paper and pen before thinking code
2. Just a few keystrokes, i.e. Python is a reduced *Lisp*
3. Emacs
4. MATLAB (as a reference)

The Project

Challenges

Yes, **software development** plus **numerical analysis** is

- A fruitful combination full of surprises
- Not that bad if you unplug and read some books first

The Project

Challenges

Not just one thing remains:

Actually appreciate and find use for the final software[3]

[3] The theory is clear enough

The Project

Checkout the source[4]

http://github.com/josatbg/python-svd

[4] The code will be there soon

The Project

Thanks and have a nice summer

Thanks!