# Singular Value Decomposition

Johan Astborg

Computational Programming with Python

Project Presentation, May 2015

## Outline

Motivation

2 Theory

3 The project

#### 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<sup>1</sup>.

<sup>&</sup>lt;sup>1</sup>Clearly, since A is non-invertible

Matrix decomposition cont.

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

$$A = UDV^*. (3)$$

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

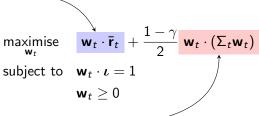


Geometric representation

Here will be a figure of the SVD.

# Power Utility Portfolio Choice

Portfolio Return



Portfolio Variance

#### Geometric representation

$$\sigma_1 = \sqrt{360} = 6\sqrt{10}$$

$$\sigma_2 = \sqrt{90} = 3\sqrt{10}$$

$$\sigma_3 = 0$$

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, 
$$U\Sigma V^* = AUV^T = A$$

Singular Value Decomposition

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

#### Singular Value Decomposition

In theory we can follow the following steps

- Find the orthogonal dianonalization of  $A^TA$
- ② Set up V and  $\Sigma$
- Construct U
- lacktriangle 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



#### Requirements

#### Requirements for the project

- **1**
- **2** 2
- **3** 3
- 4 4

Overview of numerical algorithms

### Overview of algorithms for SVD implementation

- **1**
- **2** 2
- **3**
- 4

The algorithm used

#### **Motivation**:

Suggested in the project description<sup>2</sup>

Implementation

#### How it was done

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

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

Challenges

### Not just one thing remains:

Actually appreciate and find use for the final software<sup>3</sup>

<sup>&</sup>lt;sup>3</sup>The theory is clear enough

Checkout the source<sup>4</sup>

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

<sup>&</sup>lt;sup>4</sup>The code will be there soon

Thanks and have a nice summer

Thanks!