Principal Component Analysis

Maciej Staniszewski and Adam Foster

Executive Summary

Principal Component Analysis (PCA) is a technique used for reducing the dimensionality of a dataset allowing for a more intuitive data exploration and analysis.

Table of Contents

Problem Description

In finance, similarly to other domains of science which are heavily reliant on concepts of data analysis and exploration, we are often facing large sets of data. These can be often difficult to analyse manually, even when plotted graphically. Noticing recurring patterns is highly impractical when the number of variables in the data exceeds reasonable bounds.

Interpreting data on aggregate level is especially difficult. Estimating joint distributions is an elementary step when determining the data generating process behind the data set. This in turn is a necessary step in inference-based thinking and causal discovery in data. Dimensionality curse is resticting us to three easily observable dimensions. When analysing complex data sets consisting of hundreds of variables, it is crucial that a dimensionality reducing operation is performed, enabling scientists to consider these joint distributions of every variable in a simplified form.

Theoretical Background

A very well known method of representing variables for such analysis is the Principal Component Analysis, commonly known as PCA[1]. The main idea behind it is to create new variables from existing ones, that maximize the ratio of original observed variance to the variance captured by them. To do that, n-dimensional dataset needs to be reduced to k-dimensional one, where k is smaller than n.

The coordinates of that new system are the namesake principal components. They represent directions in n-dimensional space which retain the most of the original variance when casting the data points onto them. The first vector created this way is said to explain the biggest portion of original variance and is thus called the first principal component. Data points projected on this newly created coordinate system can be explored manually and jointly due to the reduced dimensionality.

Mathematical Foundations

TODO :)

1 Principal Component Analysis - Wikipedia

Project Specification

In our project we decided to explore a well-known financial equity index – *Standard & Poor's 500* or *S&P500*[2]. Its price[3] is determined by the joint market performance of five hundred largest US-based companies traded publicly. We wanted to find the companies which drive the price variability of the index and determine if the performance of *S&P500* can be expressed as a linear combination of the underlying assets. We relied on PCA to decompose daily returns of the index into principal components and obtained main drivers from the coordinates of the first component. We arrived at a quantitative solution which helps analysts to understand price moves of complex financial products, such as an equity index.

Overview of the Data

We used a dataset which includes all *S&P500* single-name stock daily close prices in a time period between 2013 and 2018[4]. To obtain the index daily close prices in that period we used the aforementioned Python `yfinance` library. We combined these two datasets into one `pandas` dataframe and cleaned the data: transformed prices into returns, pivoted the data for compatiiblity with other libraries, ensured coherency between all starting dates and ending dates. We obtained the following data set:

date	AAL	AAPL	AAP	ABBV	ABC	ABT	ACN	ADBE	ADI	ADM	...
2013-02-08	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
2013-02-11	-0.019661	0.010422	-0.006464	-0.011034	-0.002772	-0.004359	-0.003274	-0.012270	0.008315	0.001985	...
2013-02-12	-0.013140	-0.025067	0.002679	-0.011994	0.004277	0.001168	0.004106	0.006470	0.004123	0.017503	...
2013-02-13	0.027330	-0.001903	0.004707	-0.004235	-0.006814	0.004665	0.002590	-0.002057	-0.000216	0.011360	...
2013-02-14	-0.045703	-0.000899	-0.001646	0.036859	0.002787	0.006965	-0.005846	-0.005153	0.006053	0.007702	...

5 rows × 505 columns

Figure 1: Five rows of preprocessed dataset including daily returns of all S&P500 single-name stocks and the index price betwen 2013 and 2018.

Methodology Used

We used Python programming language inside a Jupyter Notebook[5] to interactively display both code and its results. The following packages were included:

- `numpy`[6] for basic mathematical operations
- `pandas`[7] for dataset manipulations
- `ipykernel` Jupyter kernel for Python code execution

2 Standard and Poor's 500 - Wikipedia
3 S&P500 Index Price - S&P Global
4 S&P500 Single Name Prices - Kaggle Dataset
5 Jupyter Notebook - Main Page
6 NumPy - Python
7 Pandas - Python Data Analysis Library

- `matplotlib`[8] for data visualisation with plots
- `yfinance` for financial data from Yahoo Finance[9]

We were operating on the dataset described in detail in section Overview of the Data. We used `cov()` Python method to compute the covariance matrix which sample is presented below:

	AAL	AAPL	AAP	ABBV	ABC	ABT	ACN	ADBE	ADI	ADM	...
AAL	0.000504	0.000069	0.000082	0.000085	0.000072	0.000080	0.000076	0.000104	0.000105	0.000075	...
AAPL	0.000069	0.000213	0.000035	0.000045	0.000029	0.000048	0.000051	0.000065	0.000087	0.000040	...
AAP	0.000082	0.000035	0.000359	0.000068	0.000044	0.000065	0.000047	0.000050	0.000056	0.000063	...
ABBV	0.000085	0.000045	0.000068	0.000284	0.000079	0.000090	0.000055	0.000076	0.000073	0.000067	...
ABC	0.000072	0.000029	0.000044	0.000079	0.000194	0.000062	0.000044	0.000044	0.000041	0.000036	...

5 rows × 505 columns

Figure 2: Five rows of covariance matrix of daily returns of S&P500 single-name stocks and the index price betwen 2013 and 2018.

After such preprocessing we were ready to start considering eigenvectors and eigenvalues. The results of our analysis are present in the next section - Results.

Results

Summary and Recommendations

8 Matplotlib - Visualizing with Python
9 Yahoo Finance - Top ETFs