Constructing a PCA Index | Replicating S&P 500

aiquants Research | © Dayton Nyamai

[9]:

```
import os
import time
import datetime
import numpy as np
import pandas as pd
import scipy.stats as scs
from pylab import plt, mpl
from sklearn.decomposition import KernelPCA, PCA
from sklearn.preprocessing import StandardScaler


plt.style.use('seaborn-v0_8')
mpl.rcParams['savefig.dpi'] = 300
mpl.rcParams['font.family'] = 'serif'
pd.set_option('mode.chained_assignment', None)
pd.set_option('display.float_format', '{:.4f}'.format)
np.set_printoptions(suppress=True, precision=4)
os.environ['PYTHONHASHSEED'] = '0'
```

In this section, we present an illustrative example showcasing the application of Principal Component Analysis (PCA) in a specific context. We gather data from S&P 500 Index, all the individual stocks comprising the index. Subsequently, we employ PCA to extract the principal components, which we utilize to construct a composite index.

Data

Pull S&P 500 data for 2020

[10]:

```
dataFrame = pd.read_csv('sp500Index.csv',
                        index_col=0, parse_dates=True).dropna()

dataFrame[dataFrame.columns[:8]].head()
```

[10]:

```
                 A     AAL    AAPL    ABBV     ABT    ACGL      ACN     ADBE
Date
2020-01-02 83.9481 28.9829 73.2490 75.6563 81.7302 43.4000 199.2254 334.4300
```

```
2020-01-03 82.6002 27.5482 72.5369 74.9382 80.7338 43.3500 198.8936 331.8100
2020-01-06 82.8444 27.2194 73.1149 75.5296 81.1568 43.5200 197.5948 333.7100
2020-01-07 83.0983 27.1198 72.7710 75.0987 80.7056 43.1600 193.3287 333.3900
2020-01-08 83.9188 27.7375 73.9416 75.6310 81.0346 42.7300 193.7080 337.8700
```

Implementing Principal Component Analysis (PCA)

Typically, Principal Component Analysis (PCA) is applied to standardized normalized datasets.

[11]:

```python
# Convert the dataframe to a numpy array
X = dataFrame.to_numpy()

# Standardize the data by subtracting the mean and dividing by the standard␣
 ↪deviation
X = StandardScaler().fit_transform(X)

# Initialize PCA with 10 components
pca = PCA(n_components=10)

# Perform PCA on the standardized data
X_pca = pca.fit_transform(X)

# Print the explained variance of each component
print("Explained Variance of Each Component:")
print(pca.explained_variance_ratio_)
```

```
Explained Variance of Each Component:
[0.5454 0.2713 0.0544 0.0325 0.0192 0.0125 0.0089 0.0083 0.0053 0.0042]
```

Create Index Fund of S&P500 using principle component weights

[12]:

```python
# Get sum of weights for first 3 principle components
stock_weights_pca = pca.components_[0,:] + pca.components_[1,:] + pca.
 ↪components_[2,:]

# Define number of stocks to include in index fund
top_n = 100

# Get boolean array of top n elements
bool_top_stocks = abs(stock_weights_pca) > np.
 ↪sort(abs(stock_weights_pca))[len(stock_weights_pca)-top_n]

# Create data structures with weights and ticker names that define index fund
index_fund_weights = (stock_weights_pca[bool_top_stocks])
index_fund_tickers = dataFrame.columns[bool_top_stocks]
```

Bar plot visualizing index fund holdings

```
[13]: plt.figure(num=None, figsize=(24, 6))
      plt.bar(np.arange(len(index_fund_weights)), index_fund_weights/np.
       ↪max((index_fund_weights)), color = 'g', edgecolor = 'k')
      plt.title("Relative Stock Weights in Index Fund", fontsize=16)
      plt.xticks(np.arange(len(index_fund_weights)), index_fund_tickers, rotation=90)
      plt.xlabel('Stock Ticker Names', fontsize=12)
      plt.ylabel('Relative Weight', fontsize=12)
      plt.figtext(0.5, -0.1, 'Fig 1-1. Bar plot visualizing index fund holdings',␣
       ↪style='italic', ha='center')
      plt.show()
```

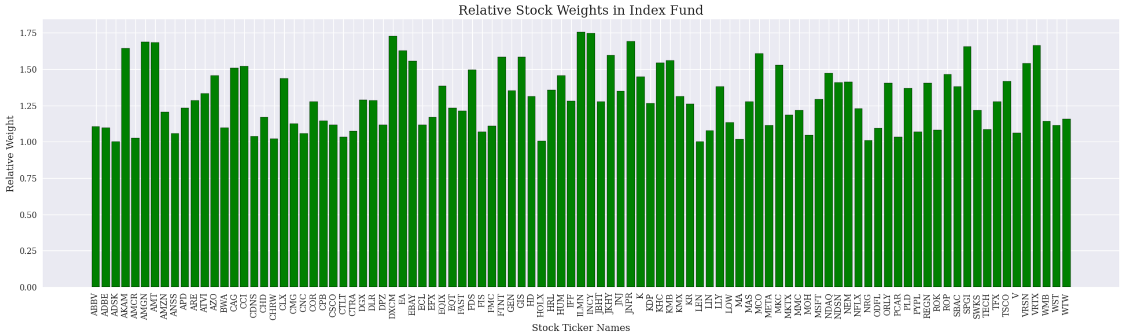

Fig 1-1. Bar plot visualizing index fund holdings

Compare S&P 500 to PCA Index Fund

Figure 1-2 illustrates the replication of the SP500 index by the PCA index.

```
[14]: # create figure
      plt.figure(num=None, figsize=(24, 6))

      # plot S&P 500 over time
      plt.subplot(2, 1, 1)
      plt.plot(np.sum(dataFrame, axis=1))
      plt.title("S&P 500 (2020)", fontsize=16)

      # plot index fund over time
      plt.subplot(2, 1, 2)
      plt.plot(-1*np.sum(index_fund_weights * dataFrame.iloc[:,bool_top_stocks],␣
       ↪axis=1))
      plt.title("PCA_Index Fund - 100 stocks  (2020)", fontsize=16)

      # add space between plots
      plt.subplots_adjust(hspace = 0.5 )
      plt.figtext(0.5, 0.01, 'Fig 1-2. Replicating  the S&P 500 Index using the PCA␣
      ↪Index', style='italic', ha='center')
      plt.show()
```

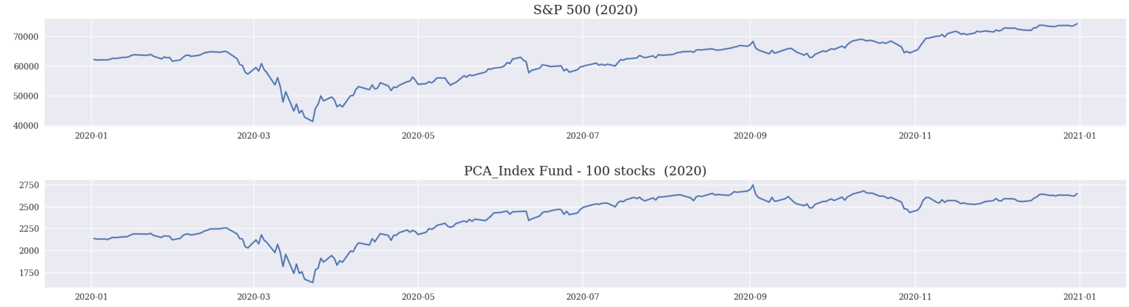

Fig 1-2. Replicating the S&P 500 Index using the PCA Index

Compare the returns of the two investment strategies over a one-year period

Actual percentage return of the S&P500

```
[15]: actual_percent_return = np.sum((dataFrame.iloc[len(dataFrame)-1,:] - dataFrame.
      ↪iloc[0,:]))/np.sum((dataFrame.iloc[0,:]))

      print('Percentage return of  S&P500')
      print(actual_percent_return)
```

```
Percentage return of  S&P500
0.193800234486313
```

Actual percentage return of the PCA Index

```
[16]: top_stocks_pc1_percent_return = np.sum(index_fund_weights * (dataFrame.
      ↪iloc[len(dataFrame)-1,bool_top_stocks] - dataFrame.iloc[0,bool_top_stocks]))/
      ↪np.sum(index_fund_weights * (dataFrame.iloc[0,bool_top_stocks]))

      print('Percentage return of PCA Index')
      print(top_stocks_pc1_percent_return)
```

```
Percentage return of PCA Index
0.23832183783101527
```

The outcome achieved through the implementation of the PCA index investment strategy is remarkably impressive, considering the straightforward nature of the approach.