Column Sketching

September 9, 2020

```
[1]: import numpy as np
     import matplotlib.pyplot as plt
     %matplotlib inline
     np.random.seed(seed=777)
     #Fake some data for exhibition purpose
     X = np.random.uniform(0,1,500)
     X = X.reshape(-1,10)
     print(X)
     #This is a 50 by 10 dataset

     import pandas as pd
```

```
[[0.15266373 0.30235661 0.06203641 0.45986034 0.83525338 0.92699705
  0.72698898 0.76849622 0.26920507 0.64402929]
 [0.09337326 0.07968589 0.58961375 0.34334054 0.98887615 0.62647321
  0.68177928 0.55225681 0.26886006 0.37325939]
 [0.2229281  0.1864426  0.39064809 0.19316241 0.61091093 0.88280845
  0.62233882 0.25311894 0.17993031 0.81640447]
 [0.22537162 0.51685714 0.51849582 0.60037494 0.53262048 0.01331005
  0.52409726 0.89588471 0.76990129 0.1228517 ]
 [0.29587269 0.61202358 0.72613812 0.46349747 0.76911037 0.19163103
  0.55786672 0.55077816 0.47222549 0.79188496]
 [0.11524968 0.6813039  0.36233361 0.34420889 0.44951875 0.02694226
  0.41524769 0.9222317  0.09120557 0.31512178]
 [0.52802224 0.32806203 0.44891554 0.01633442 0.0970269  0.69258857
  0.83594341 0.42432199 0.8487743  0.54679121]
 [0.35410346 0.72724968 0.09385168 0.8928588  0.33625828 0.89183268
  0.296849   0.30164829 0.80624061 0.83760997]
 [0.63428133 0.3113273  0.02944858 0.39977732 0.51817346 0.00738845
  0.77494778 0.8544712  0.13153282 0.28767364]
 [0.32658881 0.90655956 0.99955954 0.77088429 0.04284752 0.96525111
  0.97521246 0.2025168  0.67985305 0.46534506]
 [0.92001748 0.72820735 0.24585653 0.01953996 0.70598881 0.77448287
  0.4729746  0.80146736 0.17539792 0.72016934]
 [0.3678759  0.53209295 0.29719397 0.37429151 0.72810013 0.39850784
  0.1058295  0.39858265 0.52196395 0.1060125 ]
 [0.85349239 0.51839737 0.61106603 0.82915051 0.01689229 0.770302
  0.80847474 0.8030573  0.11476295 0.88808986]
```

```
[0.10667592 0.23981016 0.80759403 0.10198876 0.07828535 0.34543636
 0.83598444 0.06554275 0.6360506  0.67026503]
[0.79456481 0.2994673  0.75812402 0.83553903 0.37388061 0.41591966
 0.7343975  0.70654535 0.7356914  0.57297293]
[0.94526627 0.28584853 0.65293073 0.81104054 0.40542651 0.51364871
 0.14014905 0.15721947 0.03541432 0.49180601]
[0.68388763 0.17620439 0.78650459 0.13184595 0.60086037 0.51830006
 0.00414951 0.08791205 0.52267979 0.00458374]
[0.02282895 0.93060174 0.06619459 0.12418814 0.82842641 0.36812763
 0.04768249 0.60247133 0.37106164 0.68651169]
[0.25305283 0.51846337 0.57930518 0.69496954 0.568014   0.57607863
 0.97195988 0.09615256 0.15056744 0.21890239]
[0.18433482 0.17429123 0.91739598 0.087738   0.79970306 0.83145554
 0.27993361 0.4459065  0.76829829 0.16288936]
[0.77254856 0.53478201 0.15692983 0.47321536 0.70398682 0.35157667
 0.07338461 0.48765463 0.20388045 0.78410032]
[0.15491643 0.29149967 0.40241112 0.07275669 0.37958101 0.51590022
 0.71629344 0.64766257 0.82290917 0.77098869]
[0.22900459 0.66012962 0.37327435 0.55183087 0.0804195  0.2109952
 0.65217012 0.02133481 0.17582667 0.19648443]
[0.84129635 0.41820739 0.31345148 0.68702801 0.89515608 0.36977154
 0.50159014 0.10483828 0.71975822 0.63448114]
[0.80783829 0.54800581 0.02194742 0.51894304 0.09897583 0.43511542
 0.85139725 0.44118134 0.89596741 0.12072653]
[0.33303632 0.66768574 0.37253643 0.90108999 0.46836287 0.10777418
 0.31245389 0.45278092 0.24713217 0.06938743]
[0.23291488 0.09418977 0.04359004 0.69394711 0.28812972 0.01575116
 0.08326803 0.24411837 0.10969615 0.72659145]
[0.42820558 0.07101384 0.06750895 0.02519358 0.42825842 0.93007952
 0.20008667 0.39672322 0.33959378 0.9823881 ]
[0.26554444 0.3623058  0.70875493 0.01155356 0.167133   0.75875134
 0.41508562 0.77236123 0.61117578 0.17911996]
[0.40636652 0.25461546 0.50239927 0.73145006 0.51088933 0.71745862
 0.76113276 0.00623583 0.70304607 0.1365286 ]
[0.95663804 0.20531562 0.61395393 0.0095305  0.06847696 0.97789675
 0.97365789 0.32974385 0.87039916 0.17444905]
[0.39047352 0.54510419 0.4985748  0.41435601 0.08113125 0.50904101
 0.66225782 0.75083042 0.54691749 0.75474874]
[0.08975803 0.80619849 0.40007355 0.16537872 0.69858091 0.52814575
 0.44028495 0.33694007 0.14911278 0.67519679]
[0.70946756 0.89642958 0.227859   0.17226834 0.81164764 0.39406533
 0.73010599 0.04674746 0.77670941 0.87342282]
[0.75026214 0.76977188 0.80826695 0.75775349 0.78125569 0.38302591
 0.53078614 0.12339639 0.27472316 0.55060072]
[0.23894661 0.50084472 0.19263048 0.60425696 0.35960111 0.36886074
 0.62917861 0.78660705 0.48590931 0.69192111]
[0.87554652 0.81877026 0.40281936 0.57234235 0.80657272 0.57380207
 0.50132095 0.1432569  0.40690395 0.51162535]
```

```
 [0.18927863 0.34074641 0.1228205  0.69749297 0.96862159 0.6271846
  0.31281466 0.53188689 0.89087163 0.59373093]
 [0.88013472 0.61303218 0.93233281 0.18296368 0.02774407 0.33227895
  0.01076536 0.44528887 0.93149479 0.201871  ]
 [0.54213979 0.92942292 0.84216478 0.078738   0.75564842 0.4047021
  0.53665957 0.04737449 0.68841986 0.55807444]
 [0.03669628 0.75678572 0.32432037 0.01432732 0.94748355 0.51850623
  0.68308831 0.55382561 0.05646545 0.51749431]
 [0.36943643 0.05924653 0.10640852 0.86190162 0.16830811 0.06004371
  0.21744526 0.20526956 0.5349881  0.18730338]
 [0.49325975 0.65963597 0.59545756 0.64787981 0.2240036  0.52404835
  0.93018037 0.62463147 0.21829653 0.70607196]
 [0.30926068 0.29041254 0.52918557 0.26915884 0.6525739  0.46158807
  0.08430998 0.73240877 0.37039744 0.59914316]
 [0.96407415 0.54916536 0.80262017 0.16681119 0.02511028 0.77530006
  0.31921348 0.79156353 0.21402362 0.80138723]
 [0.52418399 0.67885827 0.94021637 0.82493606 0.34068255 0.80114599
  0.07084921 0.17493708 0.42903926 0.52804127]
 [0.32702651 0.47491164 0.23533784 0.98811328 0.06705414 0.72075662
  0.45687396 0.11821332 0.13410829 0.45344758]
 [0.03750877 0.77659848 0.46021742 0.94319951 0.83279599 0.34511277
  0.96941911 0.98689797 0.1267323  0.09586372]
 [0.10259792 0.45599757 0.23395138 0.83984091 0.75213683 0.30652171
  0.55532686 0.78316191 0.07237307 0.61426566]
 [0.44044365 0.1243283  0.53103499 0.62764022 0.28191159 0.26811501
  0.9924446  0.54324407 0.69513388 0.71643219]]
```

[2]:
```
m = np.shape(X)[0]
p = np.shape(X)[1]
mm = int(m*(m-1)/2)
maxCorrelation = 0.95
maxCols = p
maxIteration = p+1
print(m,p,maxCorrelation,maxIteration)
```

```
50 10 0.95 11
```

[3]:
```
def matProd(mat1,mat2):
    #print(np.shape(mat1),np.shape(mat2))
    d = np.matmul(mat1.T,mat2)
    d1 = np.trace(d)
    return (d1)
matProd(X,X)
```

[3]: 151.89852200438813

[4]:
```python
#Col algorithm
from scipy.spatial import distance_matrix
#print(previousColDist)
#print(allColDist)

def myRange(start,stop,step):
    if start == stop:
        return ([stop])
    else:
        return (np.arange(start,stop,step))

def dist(matrix):
    d = distance_matrix(matrix,matrix)
    return (d**2)

correlation = 0.0
selectedCols = []
mm = int( m*(m-1)/2 - 1 )
previousColDist = np.zeros((m,m))
previousColDist = np.asarray(previousColDist)
allColDist = dist(X) #m*(m-1)/2

#VERBOSE = True for debugging
VERBOSE = False

iterations = 0
bestColumn = 0
bestCorrelation = 0.0

print('Maximal select-able number of dimensions',maxCols)
while (correlation<maxCorrelation and len(selectedCols)<maxCols) and␣
 ↪iterations<50:
    print('Selected columns: ',selectedCols)
    print('Cumulative correlation =',bestCorrelation)
    print('Iteration :',iterations)
    print('Coefficient threshold =',maxCorrelation)
    #print('previousColDist',previousColDist)
    bestColumn = 0
    bestCorrelation = 0.0
    previousBestCorrelation = 0.0
    iterations = iterations + 1

    for j in myRange(1,p,1):
        #print(j)
        if j not in selectedCols:
            X_j = X[:,j-1].reshape(-1,1)
            jColDist = dist(X_j)
```

```
            cumColDist = np.add(jColDist,previousColDist)
            #print(np.shape(jColDist))
            #Frobenius matrix coefficient
            #corr(A,B) = trace(A^{T}B)/sqrt(trace(A^{T}A)*trace(B^{T}B))
            correlation = matProd(cumColDist,allColDist)/np.sqrt(␣
↪(matProd(cumColDist,cumColDist)*matProd(allColDist,allColDist) ) )
            if VERBOSE:print('Correlation =',correlation,' if we include␣
↪column',j,'(represented as ',j-1,'in numpy array)')
            if correlation > bestCorrelation:
                bestColumn = j
                bestCorrelation = np.copy(correlation)
    if VERBOSE:print('***Best column to include is the column',bestColumn,'␣
↪with correlation', bestCorrelation)
    if previousBestCorrelation > bestCorrelation or bestCorrelation > 1:
        bestCorrelation = np.copy(previousBestCorrelation)
        break
    X_bestColumn = X[:,bestColumn-1]
    X_bestColumn = X_bestColumn.reshape(-1,1)
    bestColDist = dist(X_bestColumn)
    #print('bestColDist',bestColDist)
    previousColDist = np.add(bestColDist,previousColDist)
    if bestColumn not in selectedCols and bestColumn != 0:
        selectedCols.append(bestColumn)
    previousBestCorrelation = np.copy(bestCorrelation)

print('Output selected columns: ',selectedCols)
print('Final cumulative coefficient (coefficient between full distance matrix␣
↪and selected column distance matrix): ',bestCorrelation)
```

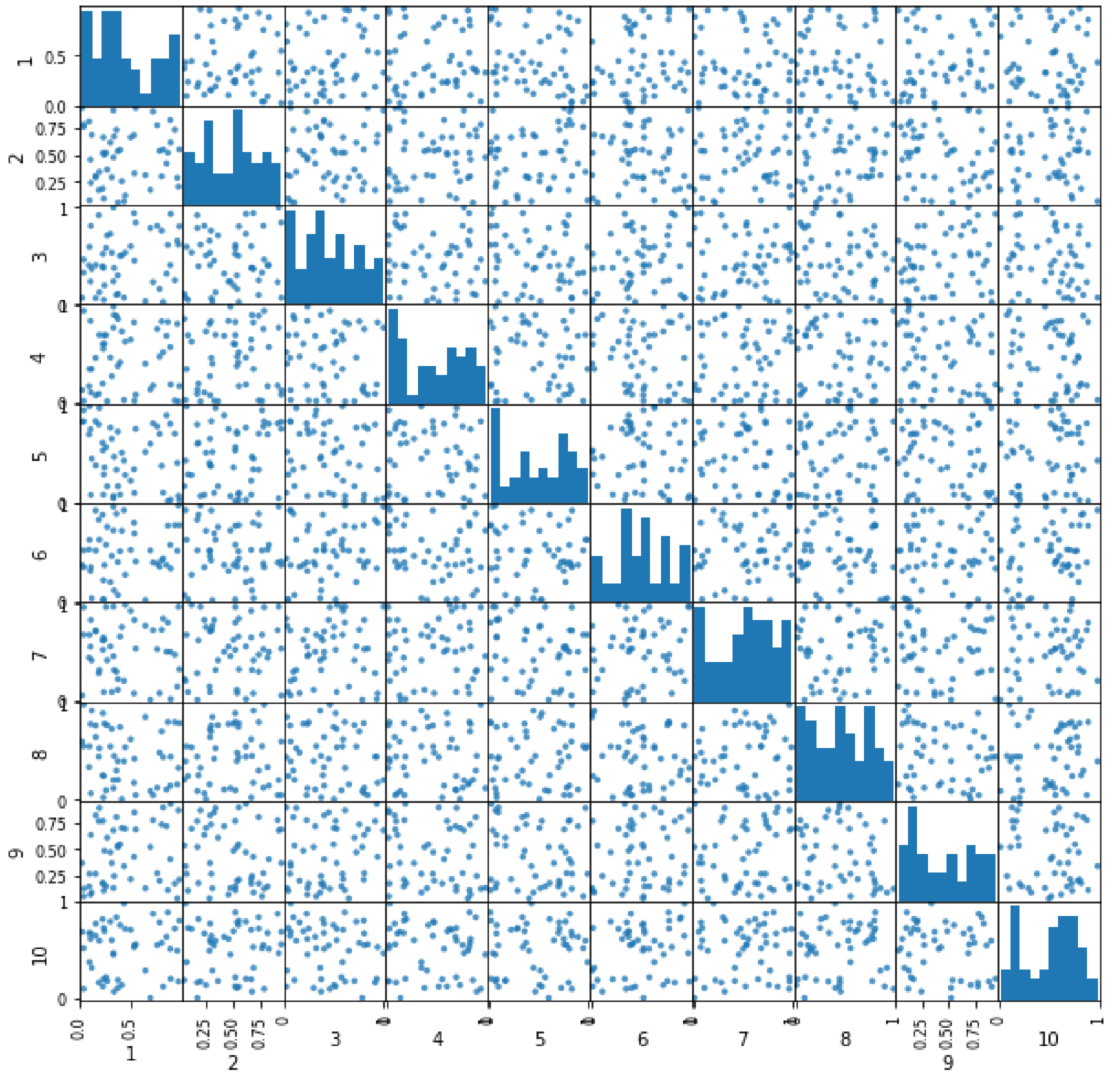
```
Maximal select-able number of dimensions 10
Selected columns:  []
Cumulative correlation = 0.0
Iteration : 0
Coefficient threshold = 0.95
Selected columns:  [4]
Cumulative correlation = 0.7244636945332077
Iteration : 1
Coefficient threshold = 0.95
Selected columns:  [4, 5]
Cumulative correlation = 0.8462894477247339
Iteration : 2
Coefficient threshold = 0.95
Selected columns:  [4, 5, 7]
Cumulative correlation = 0.9051257211714339
Iteration : 3
Coefficient threshold = 0.95
```

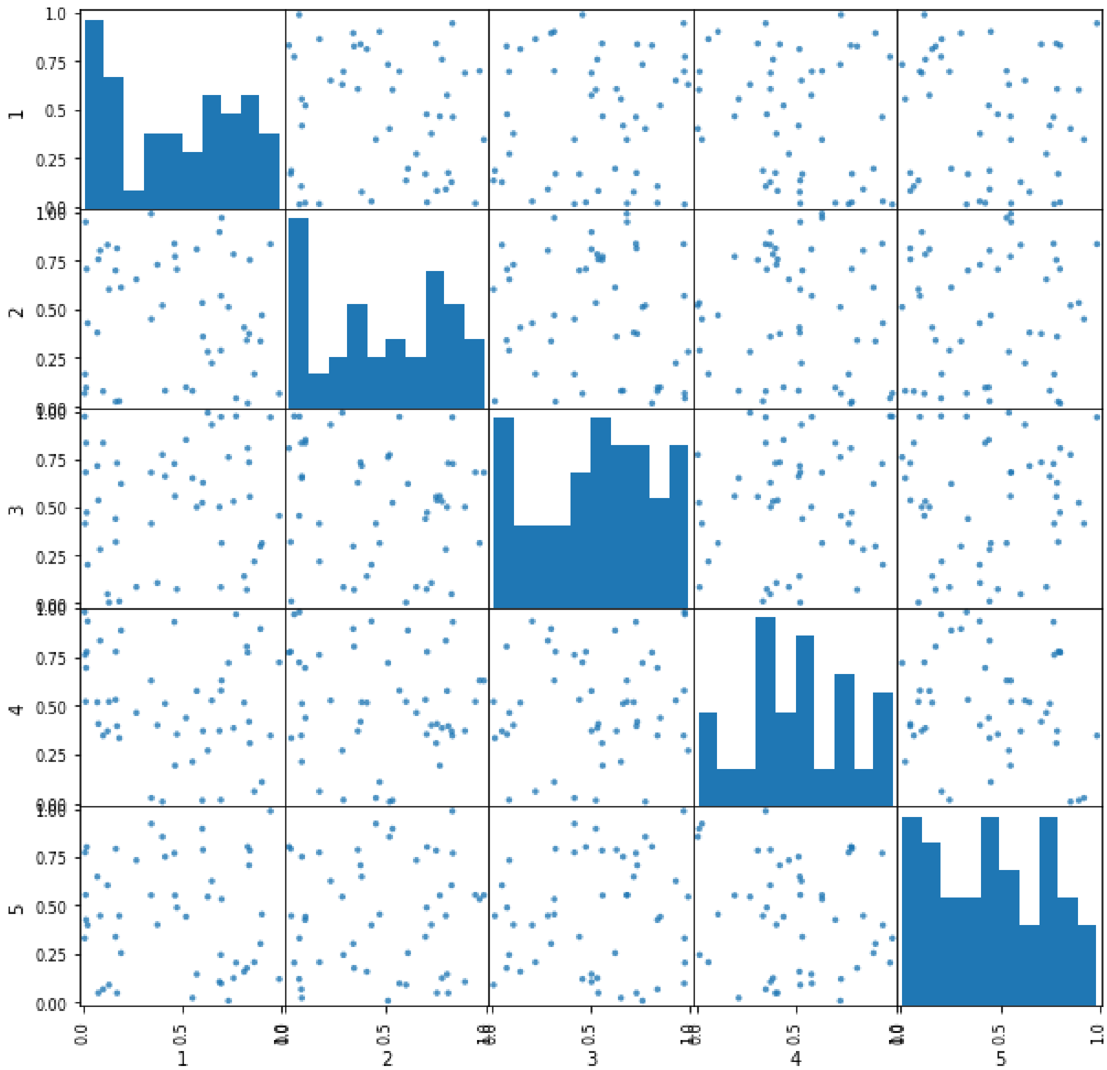
```
Selected columns:  [4, 5, 7, 6]
Cumulative correlation = 0.9389179466360492
Iteration : 4
Coefficient threshold = 0.95
Output selected columns:  [4, 5, 7, 6, 8]
Final cumulative coefficient (coefficient between full distance matrix and
selected column distance matrix):  0.9590956327563951
```

[5]:
```python
members_index = [x - 1 for x in selectedCols]
X_df = pd.DataFrame(data=X[0:,0:],index=X[0:,0],columns=myRange(1,(np.
 ↪shape(X)[1]+1),1))
X_reduced = X[:,members_index]
X_reduced_df = pd.DataFrame(data=X_reduced[0:,0:],index=X_reduced[0:
 ↪,0],columns=myRange(1,(np.shape(X_reduced)[1]+1),1))

g1 = pd.plotting.scatter_matrix(X_df, figsize=(10,10), marker = 'o', hist_kwds␣
 ↪= {'bins': 10}, s = 12, alpha = 0.8)
new_labels = [round(float(i.get_text()), 2) for i in g1[0,0].get_yticklabels()]
g1[0,0].set_yticklabels(new_labels)
plt.show()

g2 = pd.plotting.scatter_matrix(X_reduced_df, figsize=(10,10), marker = 'o',␣
 ↪hist_kwds = {'bins': 10}, s = 12, alpha = 0.8)
new_labels = [round(float(i.get_text()), 2) for i in g2[0,0].get_yticklabels()]
g2[0,0].set_yticklabels(new_labels)
plt.show()
```


1
2
3
4
5
6
7
8
9
10
0.0
0.5
0.25
0.50
0.75
0
1



1
2
3
4
5
1.0
0.75
0.5
0.25
0.0
1.00
0.50
0.00
0.0
0.5
1.0