Project: Principal Component Analysis — identifying dietary patterns in someone’s diet

Traditionally tables of food content contain many nutrients. Sometimes we do not care about each single nutrient but rather about their combinations. It would be nice to identify the driving combination of nutrients in someone's diet. We may notice that some nutrients appear in natural foods in combinations: for example, protein foods often contain a large amount of fat and no vitamin C, and vitamin C foods often contain fiber. In other words, we note that the presence of fat in the food item is positively correlated with the presence of protein and negatively correlated with the presence of vitamin C.

What if we want to identify such correlated combination of nutrients in a systematic way. We will apply a technique called principal component analysis (PCA).

For a detailed derivation of the method see “Additional Background” section. We consider the following example.

Explore the nutritional content of food in Table 2.9. What is the best way to differentiate food items? We can try a differentiation by vitamin C content or protein levels. Perhaps a combination of both can give us a better differentiation? Hidden categories or clusters of foods could be identified this way!

Among food items we may identify broad categories like meat and vegetables, as well as sub-categories such as types of vegetables. How do we derive the variables that best differentiate items? PCA finds underlying variables (known as principal components) that best differentiate your data points. Principal components are dimensions along which your data points are most spread out:

A principal component can be expressed by one or more existing variables. For example, we may use a single variable – vitamin C – to differentiate food items. Because vitamin C is present in vegetables but absent in meat, it helps to differentiate vegetables from meat but obviously cannot differentiate meat items from each other. To spread the meat items out, we can use fat content in addition to vitamin C levels, since fat is present in meat but is essentially absent in most vegetables. However, fat and vitamin C levels are measured in different units. To combine the two variables, we first have to scale it (for each value in the feature column we subtract the minimum value of this column and divide by the difference between maximum and minimum values), which would allow us to calculate a new variable – vitamin C minus fat. Combining the two variables helps to spread out both vegetable and meat items. The spread can be further improved by adding fiber, of which vegetable items have varying levels. This new variable – (vitamin C + fiber) minus fat – achieves the best data spread yet. Scaled data is shown in Table 2.10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Table 2.9 Data for Principal Component Analysis (PCA) demonstration | | | | |
| Foods | Total\_Fat g | Protein g | Dietary\_Fiber g | Vit\_C IU |
| Couscous | 0.1 | 3.4 | 1.3 | 0 |
| Peanut Butter | 16 | 7.7 | 2.1 | 0 |
| Potatoes | 0.2 | 3.7 | 3.2 | 15.6 |
| White Rice | 0.2 | 2.1 | 0.3 | 0 |
| Pork | 72.2 | 13.8 | 0 | 0 |
| Grapes | 0.1 | 0.2 | 0.2 | 1 |
| Oatmeal | 2.3 | 6.1 | 4 | 0 |
| Celery | 0.1 | 0.3 | 0.7 | 2.8 |
| Popcorn | 1.2 | 3.4 | 4.3 | 0 |
| Lettuce | 0 | 0.2 | 0.3 | 0.8 |
| White Tuna | 2.1 | 22.7 | 0 | 0 |
| Apple | 0.5 | 0.3 | 3.7 | 7.9 |
| Chicken | 10.8 | 42.2 | 0 | 0 |
| Banana | 0.5 | 1.2 | 2.7 | 10.4 |
| Tofu | 5.5 | 9.4 | 1.4 | 0.1 |
| Corn | 0.6 | 2.5 | 2 | 5.2 |
| Kiwi | 0.3 | 0.8 | 2.6 | 74.5 |
| Oranges | 0.2 | 1.2 | 3.1 | 69.7 |
| Eggs | 7.3 | 6.7 | 0 | 0.1 |
| Peppers | 0.1 | 0.7 | 1.3 | 66.1 |
| Tomato | 0.4 | 1 | 1.4 | 23.5 |
| Broccoli | 0.8 | 8 | 8.5 | 160.2 |
| Carrots | 0.1 | 0.6 | 1.6 | 5.1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Table 2.10 Data for Principal Component Analysis (PCA) demonstration –Scaled | | | | |
| Food | Fat | Protein | Fiber | Vit C |
| Couscous | 0.001385 | 0.07619 | 0.152941 | 0 |
| Peanut Butter | 0.221607 | 0.178571 | 0.247059 | 0 |
| Potatoes | 0.00277 | 0.083333 | 0.376471 | 0.097378 |
| White Rice | 0.00277 | 0.045238 | 0.035294 | 0 |
| Pork | 1 | 0.32381 | 0 | 0 |
| Grapes | 0.001385 | 0 | 0.023529 | 0.006242 |
| Oatmeal | 0.031856 | 0.140476 | 0.470588 | 0 |
| Celery | 0.001385 | 0.002381 | 0.082353 | 0.017478 |
| Popcorn | 0.01662 | 0.07619 | 0.505882 | 0 |
| Lettuce | 0 | 0 | 0.035294 | 0.004994 |
| White Tuna | 0.029086 | 0.535714 | 0 | 0 |
| Apple | 0.006925 | 0.002381 | 0.435294 | 0.049313 |
| Chicken | 0.149584 | 1 | 0 | 0 |
| Banana | 0.006925 | 0.02381 | 0.317647 | 0.064919 |
| Tofu | 0.076177 | 0.219048 | 0.164706 | 0.000624 |
| Corn | 0.00831 | 0.054762 | 0.235294 | 0.032459 |
| Kiwi | 0.004155 | 0.014286 | 0.305882 | 0.465044 |
| Oranges | 0.00277 | 0.02381 | 0.364706 | 0.435081 |
| Eggs | 0.101108 | 0.154762 | 0 | 0.000624 |
| Peppers | 0.001385 | 0.011905 | 0.152941 | 0.412609 |
| Tomato | 0.00554 | 0.019048 | 0.164706 | 0.146692 |
| Broccoli | 0.01108 | 0.185714 | 1 | 1 |
| Carrots | 0.001385 | 0.009524 | 0.188235 | 0.031835 |

As we noted before, fat and protein levels seem to move in the same direction with each other and in the opposite direction from fiber and vitamin C levels. To confirm our hypothesis, we can check for correlations between the nutrition variables. This is how correlation is computed between fat and protein, fiber and vitamin C, vitamin C and fat, fiber and fat, fiber and protein, and protein and vitamin C:

from numpy import genfromtxt

import numpy as np

import scipy.linalg as lna

from sklearn import decomposition as dc

import matplotlib.pyplot as pl

data\_file = 'food\_pca.csv'

my\_data = genfromtxt(data\_file, delimiter = ',', skip\_header=1)

#n - number of features

#m - number of food items

#first column contains food names

n = 4

m = 24

X0=my\_data[0:m,1:n+1]

#data normalization

X = (X0 - X0.min(0)) / X0.ptp(0)

cor\_fat\_protein = np.corrcoef(X[:,0],X[:,1])[0,1]

cor\_fat\_vitC = np.corrcoef(X[:,0],X[:,3])[0,1]

cor\_fat\_fiber = np.corrcoef(X[:,0],X[:,2])[0,1]

cor\_fiber\_vitC = np.corrcoef(X[:,2],X[:,3])[0,1]

cor\_fiber\_protein = np.corrcoef(X[:,2],X[:,1])[0,1]

cor\_vitC\_protein = np.corrcoef(X[:,3],X[:,1])[0,1]

As expected, there are positive correlations between fat and protein levels (r = 0.32), as well as between fiber and vitamin C levels (r = 0.69), and negative correlation between fat and vitamin C (r = -0.17), as well as between protein and vitamin C (r = -0.13), between protein and fiber (r = -0.23), and fat and fiber (r = -0.25)

Therefore, instead of analyzing all 4 nutrition variables, we can combine highly-correlated variables, leaving just 2 dimensions to consider. This is the same strategy used in PCA – it examines correlations between variables to reduce the number of dimensions in the dataset. This is why PCA is called a dimension reduction technique.

Applying PCA to the food dataset (Table 2.9) results in the principal components shown in Table 2.11.

|  |  |  |  |
| --- | --- | --- | --- |
| Table 2.11 Weights of initial features in principal components | | | |
|  | PC1 | PC2 | |
| Fat | -0.29757 | 0.50290 |
| Protein | -0.32004 | 0.73846 |
| Fiber | 0.63745 | 0.21054 |
| Vitamin C | 0.63456 | 0.39677 |

Table 2.11 represents weights used in combining variables to derive principal components. For example, to get the top principal component (PC1) value for a particular food item, we add up the amount of Fiber and Vitamin C it contains, with slightly more emphasis on Fiber, and then from that we subtract the amount of Fat and Protein it contains, with Protein negated to a larger extent.

This is the Python code to compute first two principal components

pca = dc.PCA(n\_components=2)

res = pca.fit(X)

cmp=res.components\_

expl\_var=res.explained\_variance\_ratio\_

The maximum number of principal components should be equal to the number of features. However, the decomposition shows that 76% of the variance can be explained by the first 2 principal components (see explained variance ratio in the above code sample: 50% for PC1, 26% for PC2).

The preceding analysis is very helpful for visualizing the food items to differentiate them between each other. Let us recall the food items in Figure 2.1. When they are differentiated by protein and fat level some points are clustered together (see Figure 2.10)

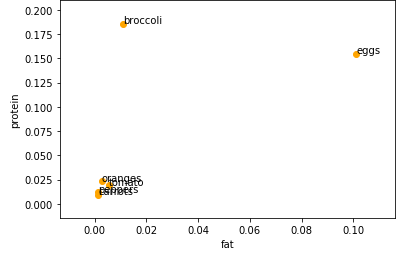


Figure 2.10 Depiction of several food items in (scaled) fat/protein space

For better differentiation between food items we can draw a dependency between the first 2 principal components rather than single foods, see Figure 2.11.

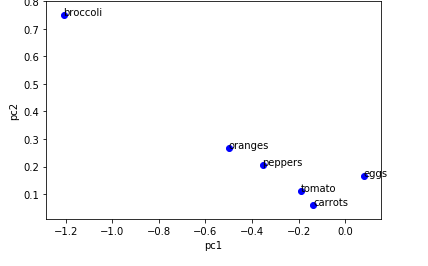


Figure 2.11 Depiction of several food items in the space of two largest principal components

Besides good visualization, we now care about two components instead of the original four features!

This is Python code that produces Figures 2.10 and 2.11:

cv = np.cov(X,rowvar=False)

eigenvals,eigenvecs = lna.eig(cv)

r = np.matmul(X,eigenvecs)

food\_labels = ['oranges', 'eggs', 'peppers', 'tomato','broccoli','carrots']

pl.figure()

pl.title('Depiction of several food items in fat/protein space')

xxx=X[17:23,0]

yyy=X[17:23,1]

pl.scatter(xxx,yyy,facecolor='orange')

pl.ylabel('protein')

pl.xlabel('fat')

for i, txt1 in enumerate(food\_labels):

pl.annotate(txt1, (xxx[i], yyy[i]))

pl.show()

pl.figure()

pl.title('Depiction of several food items in the space of two largest principal components')

xxx=r[17:23,0]

yyy=r[17:23,1]

pl.scatter(xxx,yyy,facecolor='blue')

for j, txt in enumerate(food\_labels):

pl.annotate(txt, (xxx[j], yyy[j]))

pl.ylabel('pc2')

pl.xlabel('pc1')

pl.show()

PCA is applied in many fields for dimension reduction – the first few principal components can usually be used in place of the original features. One of common applications of PCA in the nutrition / dieting world is identifying dietary patterns in different populations of people.

Additional Background

Principal component analysis can be formulated as an optimization problem. (More precisely, this problem of factorization is transformed into a series of sub problems of finding only factors.)

Let X be a random vector where X=( are interpreted as measurable characteristics of some object. If n is too large we’d like to replace n characteristics with one. How? New characteristics can be chosen as linear combination of old n characteristics:

How do we choose C=(?

Let the expectation of be 0 for each i:



If it is not 0, we always can subtract a constant to make it 0.

We choose C in such a way that the dispersion of a new characteristic reaches its maximum. *Maximum dispersion means taking into account more information*.(Simply put, such a sample gives more information.)





How do we solve it? The Lagrange method will be used.

Note that



where  is the covariance matrix



The Lagrange function follows:



Setting the derivative to 0 we get the following:



In other words, C is an eigenvector. This problem fits into stochastic approximation formulation. More details on matrix transformation can be found in [3].

Questions and Exercises

1. What is the purpose of PCA?

Which Python packages and functions can be used to perform PCA?

1. How did PCA help to differentiate food items considered in the project?