CHAPTER VII
Principal Component Analysis (PCA)

PHOK Ponna and NEANG Pheak

Institute of Technology of Cambodia
Department of Applied Mathematics and Statistics (AMS)

2022–2023

Contents

1. Expected Value and Mean
2. Variance and Standard Deviation
3. Covariance and Correlation
4. Covariance Matrix
5. What is Principal Component Analysis
6. Calculate Principal Component Analysis
7. Principal Component Analysis in scikit-learn

Contents

1. Expected Value and Mean
2. Variance and Standard Deviation
3. Covariance and Correlation
4. Covariance Matrix
5. What is Principal Component Analysis
6. Calculate Principal Component Analysis
7. Principal Component Analysis in scikit-learn

Definition 1

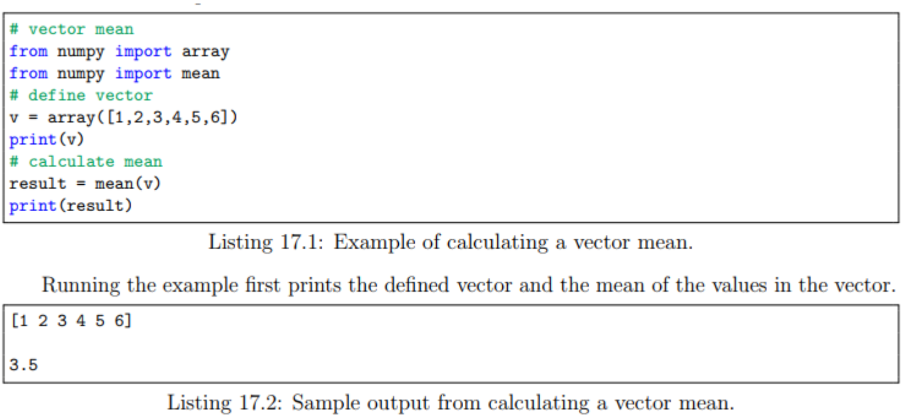
Let X be a random variable whose range is D. In probability, the expected value or expectation of X is denoted by $E(X)$ or μ :

$$\mu = E(X) = \sum_{x \in D} x.p(x), \qquad p(x) = P(X = x)$$

In statistics, the mean, or more technically the arithmetic mean or sample mean, can be estimated from a sample of examples drawn from the domain. If $X = (x_1, \ldots, x_n)$ is a vector of observations, then the sample mean $\bar{x}$ is defined by

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

The arithmetic mean can be calculated for a vector or matrix in NumPy by using the mean()function. The example below defines a 6-element vector and calculates the mean.



```
# vector mean
from numpy import array
from numpy import mean
# define vector
v = array([1,2,3,4,5,6])
print(v)
# calculate mean
result = mean(v)
print(result)
```

Listing 17.1: Example of calculating a vector mean.

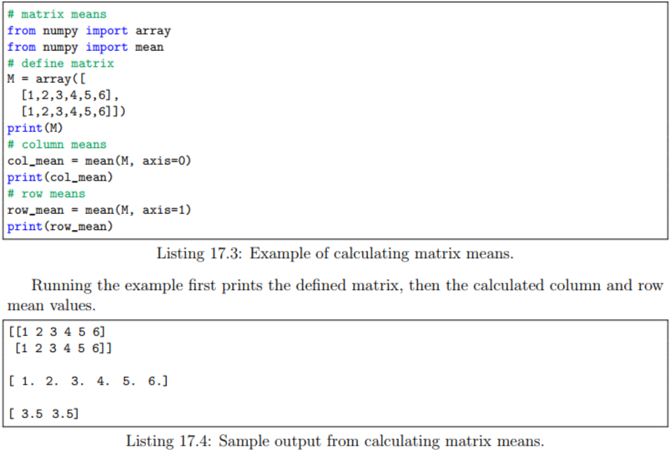
Running the example first prints the defined vector and the mean of the values in the vector.

```
[1 2 3 4 5 6]

3.5
```

Listing 17.2: Sample output from calculating a vector mean.

The mean function can calculate the row or column means of a matrix by specifying the axis argument and the value 0 or 1 respectively. The example below defines a 2×6 matrix andcalculates both column and row means.



```
# matrix means
from numpy import array
from numpy import mean
# define matrix
M = array([
  [1,2,3,4,5,6],
  [1,2,3,4,5,6]])
print(M)
# column means
col_mean = mean(M, axis=0)
print(col_mean)
# row means
row_mean = mean(M, axis=1)
print(row_mean)
```

Listing 17.3: Example of calculating matrix means.

Running the example first prints the defined matrix, then the calculated column and row mean values.

```
[[1 2 3 4 5 6]
 [1 2 3 4 5 6]]

[ 1.  2.  3.  4.  5.  6.]

[ 3.5  3.5]
```

Listing 17.4: Sample output from calculating matrix means.

Contents

1. Expected Value and Mean
2. Variance and Standard Deviation
3. Covariance and Correlation
4. Covariance Matrix
5. What is Principal Component Analysis
6. Calculate Principal Component Analysis
7. Principal Component Analysis in scikit-learn

Definition 2

In probability, the variance of some random variable X is a measure of how much values in thedistribution vary on average with respect to the mean. The variance of X, denoted by $Var(X)$, is defined by

$$Var(X) = E[(X - E(X))^2]$$

In statistics, the variance can be estimated from a sample of examples drawn from the domain. The population variance, denoted by σ^2, is defined by

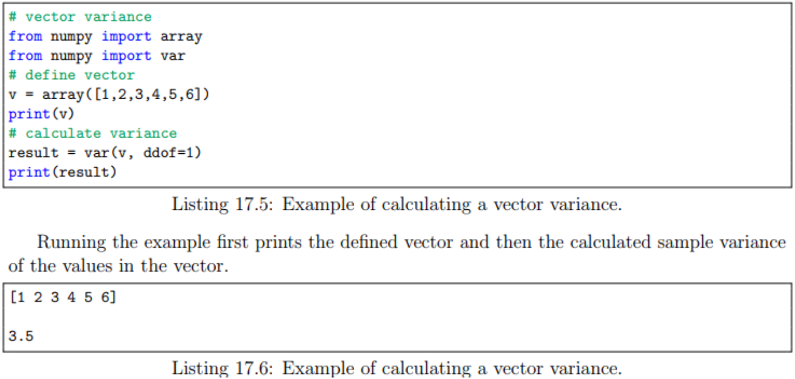
$$\sigma^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)^2$$

The sample variance, denoted by s^2, is defined by

$$s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_1 - \bar{x})^2$$

The standard deviation is the square root of the variance.

In NumPy, the variance can be calculated for a vector or a matrix using the var() function. By default, the var() function calculates the population variance. To calculate the sample variance, you must set the ddof argument to the value 1. The example below defines a 6-element vector and calculates the sample variance.



```
# vector variance
from numpy import array
from numpy import var
# define vector
v = array([1,2,3,4,5,6])
print(v)
# calculate variance
result = var(v, ddof=1)
print(result)
```

Listing 17.5: Example of calculating a vector variance.

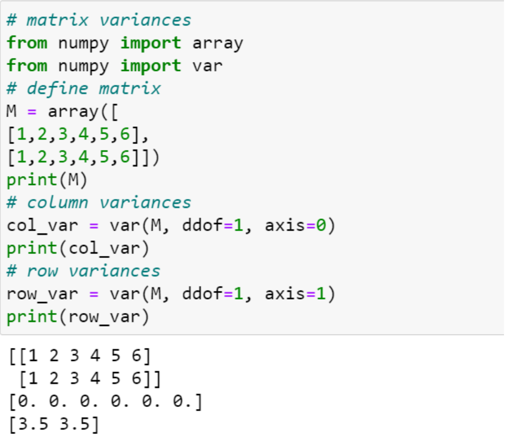
Running the example first prints the defined vector and then the calculated sample variance of the values in the vector.

```
[1 2 3 4 5 6]

3.5
```

Listing 17.6: Example of calculating a vector variance.

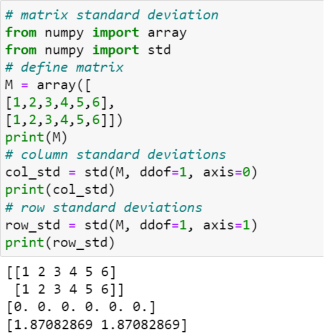
The var function can calculate the row or column variances of a matrix by specifying the axis argument and the value 0 or 1 respectively, the same as the mean function above. The example below defines a 2×6 matrix and calculates both column and row sample variances.



```
# matrix variances
from numpy import array
from numpy import var
# define matrix
M = array([
[1,2,3,4,5,6],
[1,2,3,4,5,6]])
print(M)
# column variances
col_var = var(M, ddof=1, axis=0)
print(col_var)
# row variances
row_var = var(M, ddof=1, axis=1)
print(row_var)
```

```
[[1 2 3 4 5 6]
 [1 2 3 4 5 6]]
[0. 0. 0. 0. 0. 0.]
[3.5 3.5]
```

The standard deviation is calculated as the square root of the variance and is denoted as σ for population and s for sample.
NumPy also provides a function for calculating the standard deviation directly via the std() function. As with the var() function, the ddof argument must be set to 1 to calculate the unbiased sample standard deviation and column and row standard deviations can be calculated by setting the axis argument to 0 and 1 respectively.The example below demonstrates how to calculate the sample standard deviation for the rows and columns of a matrix.



```
# matrix standard deviation
from numpy import array
from numpy import std
# define matrix
M = array([
[1,2,3,4,5,6],
[1,2,3,4,5,6]])
print(M)
# column standard deviations
col_std = std(M, ddof=1, axis=0)
print(col_std)
# row standard deviations
row_std = std(M, ddof=1, axis=1)
print(row_std)
```

```
[[1 2 3 4 5 6]
 [1 2 3 4 5 6]]
[0. 0. 0. 0. 0. 0.]
[1.87082869 1.87082869]
```

Contents

1. Expected Value and Mean
2. Variance and Standard Deviation
3. Covariance and Correlation
4. Covariance Matrix
5. What is Principal Component Analysis
6. Calculate Principal Component Analysis
7. Principal Component Analysis in scikit-learn

Definition 3

In probability, covariance is the measure of the joint probability for two random variables. It describes how the two variables change together. It is denoted as the function $cov(X, Y)$, where X and Y are the two random variables being considered. Covariance is calculated as expected value or average of the product of the differences of each random variable from their expected values, where $E[X]$ is the expected value for X and $E[Y]$ is the expected value of Y.

$$cov(X, Y) = E\left[(X - E[X])(Y - E[Y])\right]$$

Assuming the expected values for X and Y have been calculated, the covariance can be calculated as the sum of the difference of x values from their expected value multiplied by the difference of the y values from their expected values multiplied by the reciprocal of the number of examples in the population.

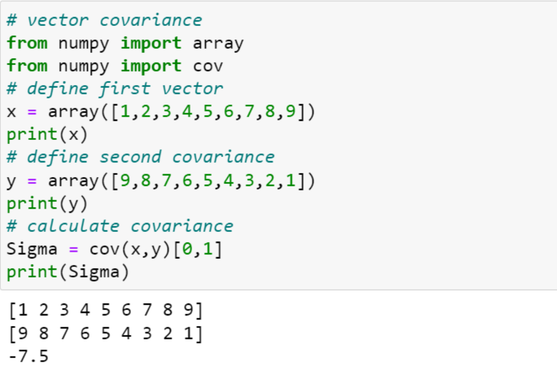
$$cov(X, Y) = \frac{1}{n}\sum(x - E[X])(y - E[Y])$$

In statistics, the sample covariance can be calculated in the same way, although with a bias correction, the same as with the variance.

$$cov(X, Y) = \frac{1}{n-1} \sum (x - E[X])(y - E[Y])$$

The sign of the covariance can be interpreted as whether the two variables increase together(positive) or decrease together (negative). The magnitude of the covariance is not easily interpreted. A covariance value of zero indicates that both variables are completely independent. NumPy does not have a function to calculate the covariance between two variables directly.Instead, it has a function for calculating a covariance matrix called cov() that we can use to retrieve the covariance. By default, the cov()function will calculate the unbiased or sample covariance between the provided random variables.

The example below defines two vectors of equal length with one increasing and one decreasing. We would expect the covariance between these variables to be negative. We access just the covariance for the two variables as the [0; 1] element of the square covariance matrix returned.



```
# vector covariance
from numpy import array
from numpy import cov
# define first vector
x = array([1,2,3,4,5,6,7,8,9])
print(x)
# define second covariance
y = array([9,8,7,6,5,4,3,2,1])
print(y)
# calculate covariance
Sigma = cov(x,y)[0,1]
print(Sigma)
```

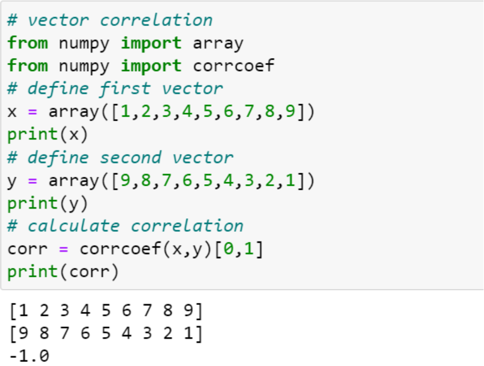
```
[1 2 3 4 5 6 7 8 9]
[9 8 7 6 5 4 3 2 1]
-7.5
```

The covariance can be normalized to a score between -1 and 1 to make the magnitude interpretable by dividing it by the standard deviation of X and Y . The result is called the **correlation** of the variables, also called the **Pearson correlation coefficient**, named for the developer of the method.

$$r = \frac{cov(X,Y)}{s_X . s_Y}$$

Where r is the correlation coefficient of X and Y , $cov(X,Y)$ is the sample covariance of X and Y and s_X and s_Y are the standard deviations of X and Y respectively.

NumPy provides the corrcoef() function for calculating the correlation between two variables directly. Likecov(), it returns a matrix, in this case a correlation matrix. As with the results from cov() we can access just the correlation of interest from the [0,1] value from the returned squared matrix.



```
# vector correlation
from numpy import array
from numpy import corrcoef
# define first vector
x = array([1,2,3,4,5,6,7,8,9])
print(x)
# define second vector
y = array([9,8,7,6,5,4,3,2,1])
print(y)
# calculate correlation
corr = corrcoef(x,y)[0,1]
print(corr)
```

```
[1 2 3 4 5 6 7 8 9]
[9 8 7 6 5 4 3 2 1]
-1.0
```

Contents

1 Expected Value and Mean

2 Variance and Standard Deviation

3 Covariance and Correlation

4 Covariance Matrix

5 What is Principal Component Analysis

6 Calculate Principal Component Analysis

7 Principal Component Analysis in scikit-learn

Definition 4

The covariance matrix is a square and symmetric matrix that describes the covariance between two or more random variables. The diagonal of the covariance matrix are the variances of each of the random variables, as such it is often called the variance-covariance matrix. A covariance matrix is a generalization of the covariance of two variables and captures the way in which all variables in the dataset may change together. The covariance matrix is denoted as the uppercase Greek letter Sigma, e.g. Σ. The covariance for each pair of random variables is calculated asabove.

$$\Sigma = E\left[(X - E[X])(Y - E[Y])\right]$$

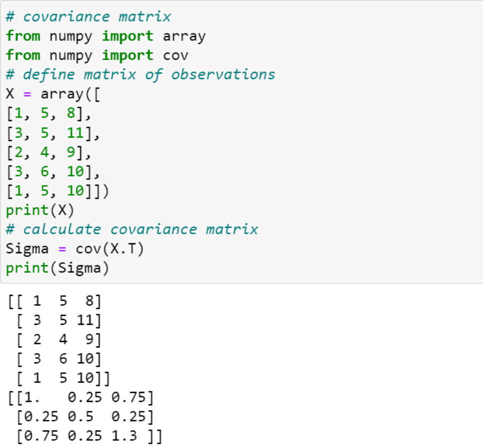
Where:

$$\Sigma_{ij} = cov(X_i, X_j)$$

And X is a matrix where each column represents a random variable.

The covariance matrix provides a useful tool for separating the structured relationships in a matrix of random variables.This can be used to decorrelate variables or applied as a transform to other variables. It is a key element used in the Principal Component Analysis data reduction method, or PCA for short.
The covariance matrix can be calculated in NumPy using the cov() function. By default,this function will calculate the sample covariance matrix. The cov() function can be called with a single 2D array where each sub-array contains a feature (e.g. column). If this function is calle dwith your data defined in a normal matrix format (rows then columns), then a transpose of the matrix will need to be provided to the function in order to correctly calculate the covariance of the columns.

Below is an example that defines a dataset with 5 observations across 3 features and calculates the covariance matrix.



```
# covariance matrix
from numpy import array
from numpy import cov
# define matrix of observations
X = array([
[1, 5, 8],
[3, 5, 11],
[2, 4, 9],
[3, 6, 10],
[1, 5, 10]])
print(X)
# calculate covariance matrix
Sigma = cov(X.T)
print(Sigma)
```

```
[[ 1  5  8]
 [ 3  5 11]
 [ 2  4  9]
 [ 3  6 10]
 [ 1  5 10]]
[[1.   0.25 0.75]
 [0.25 0.5  0.25]
 [0.75 0.25 1.3 ]]
```

Contents

1. Expected Value and Mean
2. Variance and Standard Deviation
3. Covariance and Correlation
4. Covariance Matrix
5. What is Principal Component Analysis
6. Calculate Principal Component Analysis
7. Principal Component Analysis in scikit-learn

Definition 5

Principal Component Analysis, or PCA for short, is a method for reducing the dimensionality of data. It can be thought of as a projection method where data with m-columns (features) is projected into a subspace with m or fewer columns, whilst retaining the essence of the original data. The PCA method can be described and implemented using the tools of linear algebra.

PCA is an operation applied to a dataset, represented by an $n \times m$ matrix A that results in a projection of A which we will call B. Let's walk through the steps of this operation.

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

$$B = PCA(A)$$

1. The first step is to calculate the mean values of each column.

$$M = mean(A)$$

2. Next, we need to center the values in each column by subtracting the mean column value.

$$C = A - M$$

3. The next step is to calculate the covariance matrix of the centered matrix C.

$$V = cov(C)$$

4. Finally, we calculate the eigendecomposition of the covariance matrix V . This results in a list of eigenvalues and a list of eigenvectors.

$$values, vectors = eig(V)$$

The eigenvectors represent the directions or components for the reduced subspace of B,whereas the eigenvalues represent the magnitudes for the directions. The eigenvectors can be sorted by the eigenvalues in descending order to provide a ranking of the components or axes of the new subspace for A. If all eigenvalues have a similar value, then we know that the existing representation may already be reasonably compressed or dense and that the projection may offer little. If there are eigenvalues close to zero, they represent components or axes of B that may be discarded. A total of m or less components must be selected to comprise the chosen subspace. Ideally, we would select k eigenvectors, called **principal components**, that have the k largest eigenvalues.

$$B = select(values, vectors)$$

Once chosen, data can be projected into the subspace via matrix multiplication.

$$P = B^T A$$

Contents

1 Expected Value and Mean

2 Variance and Standard Deviation

3 Covariance and Correlation

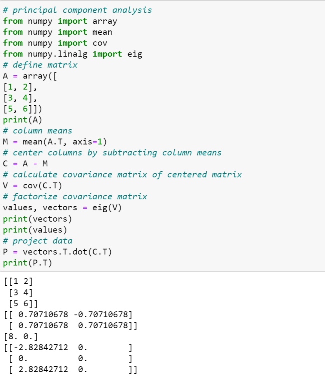
4 Covariance Matrix

5 What is Principal Component Analysis

6 Calculate Principal Component Analysis

7 Principal Component Analysis in scikit-learn

There is no pca() function in NumPy, but we can easily calculate the Principal Component Analysis step-by-step using NumPy functions. The example below defines a small 3×2 matrix,centers the data in the matrix, calculates the covariance matrix of the centered data, and then the eigendecomposition of the covariance matrix. The eigenvectors and eigenvalues are taken as the principal components and singular values and used to project the original data.



```python
# principal component analysis
from numpy import array
from numpy import mean
from numpy import cov
from numpy.linalg import eig
# define matrix
A = array([
[1, 2],
[3, 4],
[5, 6]])
print(A)
# column means
M = mean(A.T, axis=1)
# center columns by subtracting column means
C = A - M
# calculate covariance matrix of centered matrix
V = cov(C.T)
# factorize covariance matrix
values, vectors = eig(V)
print(vectors)
print(values)
# project data
P = vectors.T.dot(C.T)
print(P.T)
```

```
[[1 2]
 [3 4]
 [5 6]]
[[ 0.70710678 -0.70710678]
 [ 0.70710678  0.70710678]]
[8. 0.]
[[-2.82842712  0.        ]
 [ 0.          0.        ]
 [ 2.82842712  0.        ]]
```

Contents

1 Expected Value and Mean

2 Variance and Standard Deviation

3 Covariance and Correlation

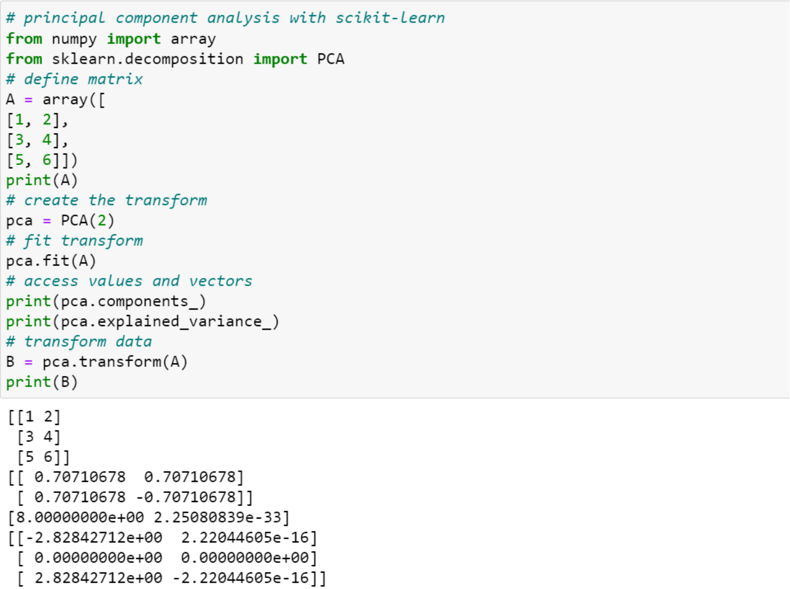
4 Covariance Matrix

5 What is Principal Component Analysis

6 Calculate Principal Component Analysis

7 Principal Component Analysis in scikit-learn

We can calculate a Principal Component Analysis on a dataset using the PCA() class in the scikit-learn library. The benefit of this approach is that once the projection is calculated, it can be applied to new data again and again quite easily. When creating the class, the number of components can be specified as a parameter. The class is first fit on a dataset by calling the fit()function, and then the original dataset or other data can be projected into a subspace with the chosen number of dimensions by calling the **transform**() function. Once fit, the singular values and principal components can be accessed on the PCA class via the explained variance and components attributes. The example below demonstrates using this class by first creating an instance, fitting it on a 3×2 matrix, accessing the values and vectors of the projection, and transforming the original data.



```
# principal component analysis with scikit-learn
from numpy import array
from sklearn.decomposition import PCA
# define matrix
A = array([
[1, 2],
[3, 4],
[5, 6]])
print(A)
# create the transform
pca = PCA(2)
# fit transform
pca.fit(A)
# access values and vectors
print(pca.components_)
print(pca.explained_variance_)
# transform data
B = pca.transform(A)
print(B)
```

```
[[1 2]
 [3 4]
 [5 6]]
[[ 0.70710678  0.70710678]
 [ 0.70710678 -0.70710678]]
[8.00000000e+00 2.25080839e-33]
[[-2.82842712e+00  2.22044605e-16]
 [ 0.00000000e+00  0.00000000e+00]
 [ 2.82842712e+00 -2.22044605e-16]]
```

Steps for Computing PCA

1. Step 1: Standardize the dataset.
2. Step 2: Calculate the covariance matrix for the features in the dataset.
3. Step 3: Calculate the eigenvalues and eigenvectors for the covariance matrix.
4. Step 4: Sort eigenvalues and their corresponding eigenvectors.
5. Step 5: Pick k eigenvalues and form a matrix of eigenvectors.
6. Step 6: Transform the original matrix.

Example 1

Compute the PCA for the following dataset which has 4 features and a total of 5 training examples.

f1	f2	f3	f4
1	2	3	4
5	5	6	7
1	4	2	3
5	3	2	1
8	1	2	2