# Implementation and Use of The K-Means Algorithm

Swetha Varadarajan

September 11, 2014

## Contents



1. first
2. second

**Abstract**

The *k-means* algorithm is described and an implementation in python is presented. The algorithm is applied to a manually created data-set for demonstation and also to a data set obtained from the UCI Machine Learning Repository. The result of each iteration of the *k-means* algorithm is illustrated and variations in the results are studied for different values of $k$.

# 1 Introduction

Finding groupings of data samples consisting of more than two or three dimensions.

Objective of *k-means* is to find points in the data space that are near the centers of groupings of data (Bishop, 2006, pp. 424–428). Difficulties: don't know how many groupings there are, so don't know how many centers to use. Will consider a performance measure to help choose.

# 2 Python Implementation of *k-means*

This section contains a summary of the *k-means* algorithm, an implementation of it in python, and some demonstrations of the algorithm's behavior on two-dimensional data.

## 2.1 *k-means*

The naive idea of using training values, finding the cost function and later optimizing for the best result is followed in this algorithm. The *k-means* algorithm operates by 5 steps.

step1: Initialize the centers by taking random $k$ values of centers (training values).

step2: Find the distances given by the equation-1 below

$$(x_n - \mu_k)^2$$

step3: Estimate a better center value and find *distortion measure* (cost function-J) given by equation-2 below.

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} (x_n - \mu_k)^2$$

step4: Update the cluster centers.

step5: Test for convergence. i.e, repeat steps 2 and 3 for n iterations with the updated center values at each iteration (optimizing).

## 2.2 Implementation

The implementation is done using python scripting. The code structure consists of 4 parts. Part-1 has the importing of necessary packages. Part-2 is the definition of functions. Part-3 consists of reading or creating the datasets and calling the defined functions. Part-4 consists of plotting the obtained results. Part-2 is discussed here. Next section discusses Part-3 and 4 in detail.

```
def kmeans(d, k, n):
    #Step-1
    centers = d[np.random.choice(range(d.shape[0]),k,replace=False), :]
    J = []

    #Step-5
    for iteration in range(n):

        #Step-2
        # Which center is each sample closest to?
        sqdistances = np.sum((centers[:,np.newaxis,:] - d)**2, axis=2)
        #Step-3
        closest = np.argmin(sqdistances, axis=0)

        # Calculate J and append to list J
        J.append(calcJ(d,centers))

        # Step-4
        for i in range(k):
            centers[i,:] = d[closest==i,:].mean(axis=0)

    J.append(calcJ(d,centers))
    return centers,J,closest
```

In the above piece of code, the function of k-means is defined for k number of clusters and n number of iterations. The 5 steps are mentioned in the comments. In step-1, the centers are randomly selected from the range of data set. step2 finds the equation-1. The syntax says that the equation-1 has to be performed along axis-2. Axis-2 here denotes the individual column(attributes). In step-3, the appropriate center is found out by picking the index value of the minimum distance. In step-4, the new center is found by taking the mean of values closest to the kth center. Step-5 is the repetition of the above steps. The J value is determined at each step using a seperate function listed below. This involves the double summation as shown in the equation-2. The convergence can be found by monitoring the recorded J's value at each iteration.

```
def calcJ(d,centers):
    diffsq = (centers[:,np.newaxis,:] - d)**2
    return np.sum(np.min(np.sum(diffsq,axis=2), axis=0))
```

### 2.3 Demonstrations

To demonstrate the above explained *k-means* algorithm and the distortion measure *J*, let us consider a set of *two-dimensional* data. Let a=[1,4],[2,3],[3,2],[4,1],[1,2.5],[2,2.5],[3,2.6],[4,2.4] be the set.

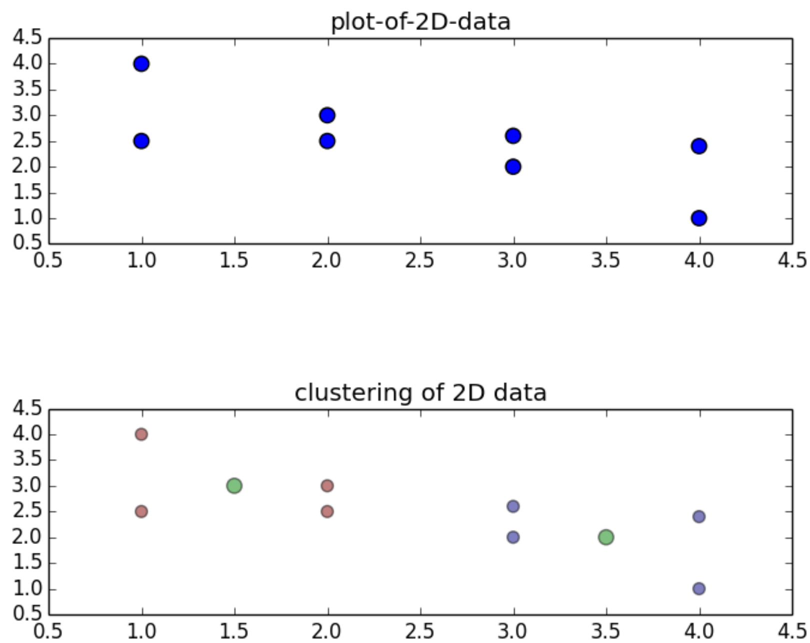


Figure 1: Demonstration of *k-means* on two-dimensional data.

The subplot-1 gives a plot of the 2-D data. The subplot-2 gives the clustering result for k=2 and the number of iterations as 7. The green spots represent the centers. Since k=2, there are 2 different clusters represented by blue and red.

The convergence happens after the 2nd iteration. This was found out from the J value obtained at each iteration as seen in figure-2 since the J value remains constant after 2nd iteration.

Thus, *k-means* algorithm as simple as seen in the demonstration, is useful for bigger data sets. One such example is shown in the next section.

## 3 Experiments

This section describes the data set which is being used to study the *k-means* algorithm and the obtained results.

### 3.1 Data Sets

From UCI Machine Learning Repository, the data set of TAE `http://archive.ics.uci.edu/ml/machine-learning-databa` `tae/` has been selected. It has 6 attributes describing the performance of the Teaching assistants at UW-

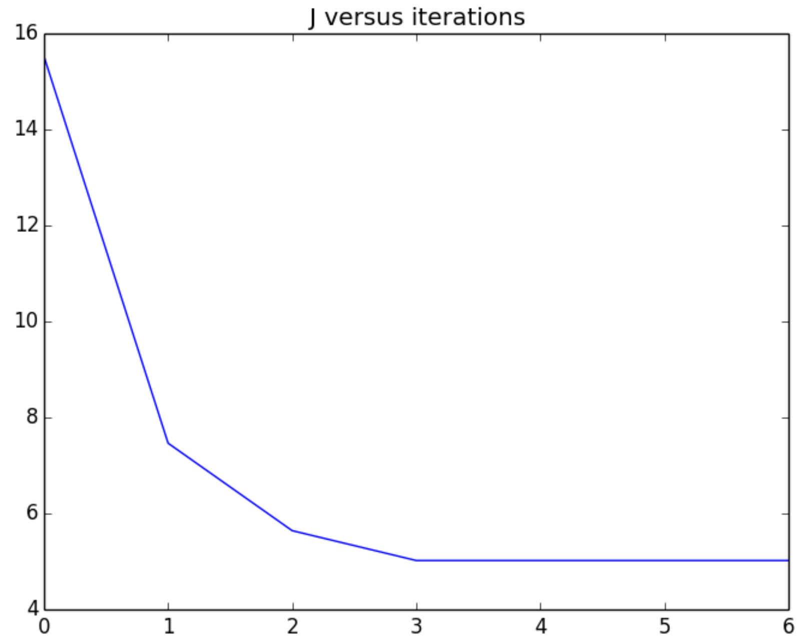


Figure 2: Demonstration of $J$ *measure* on two-dimensional data.

Madison.

### 3.2 Loading Data Set into python

The data set was downloaded from the url and the following command was used to load into python. The pandas package was used.

```
data = pandas.read_csv(open('tae.data'))
data = data[data.isnull().any(axis=1)==False]
d=data.iloc[:,:].values
```

### 3.3 Result set-1

The k-means algorithm has been applied to the entire data-set and the distortion measure obtained as a function of iteration is shown in figure-3

[!htbp]

The corresponding values of J at each iteration are

$$J = [39750, 25345, 21341, 19196, 18604, 18404, 18444, 18393, 18393, 18393, 18393]$$

thus, convergence happens after iteration 8.

The variation of J with respect to k is been observed and the plot shown in figure-4 says that as k increases, the J value decreases.

### 3.4 Result set-2

This result set is obtained for a subset(40 TA's report) of 2 of the attributes (class size and class attribute) of the selected data set.

Figure-5 shows the distribution of data. Next 5 figures(Figure-6 to Figure-11)show the movement of centers which finally reaches convergence in figure 6.

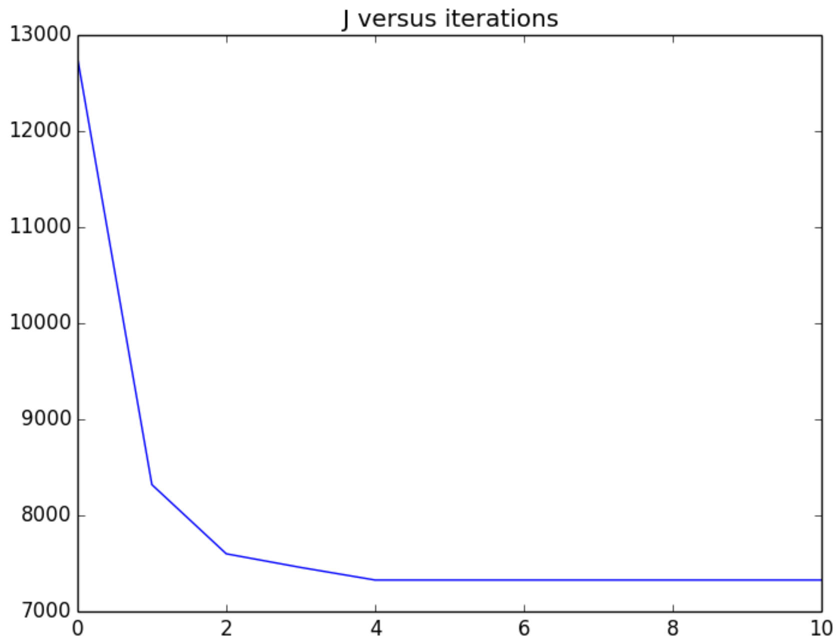


Figure 3: J versus iteration

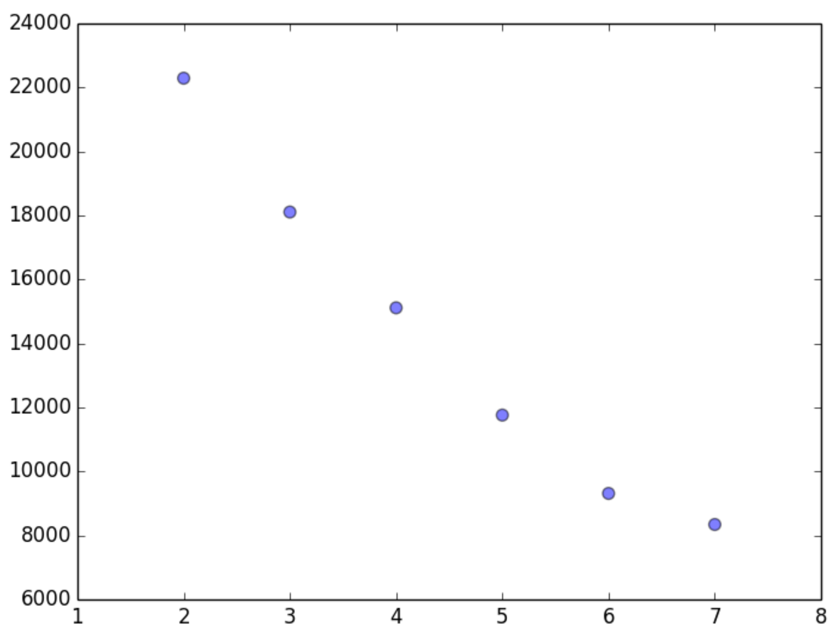


Figure 4: J versus k value

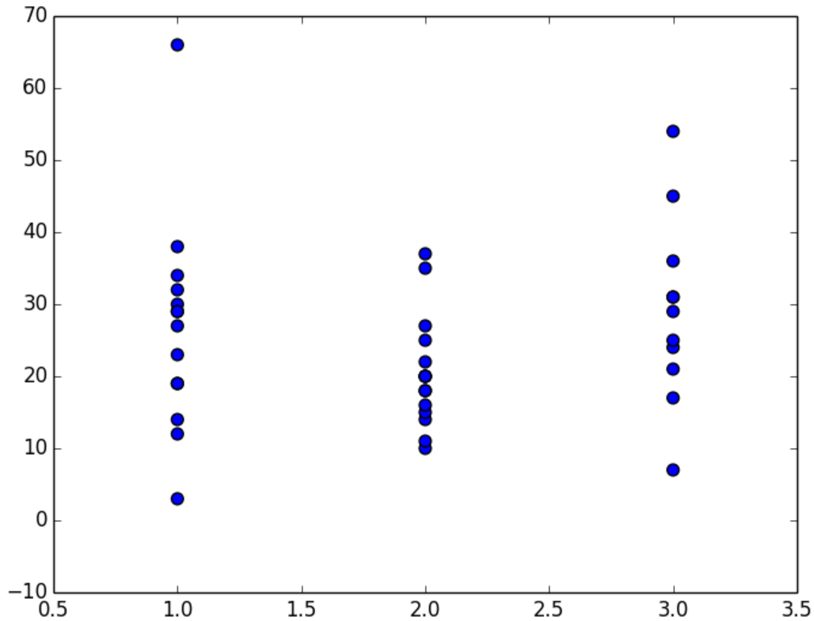


Figure 5: Demonstration of *k-means* on subset.

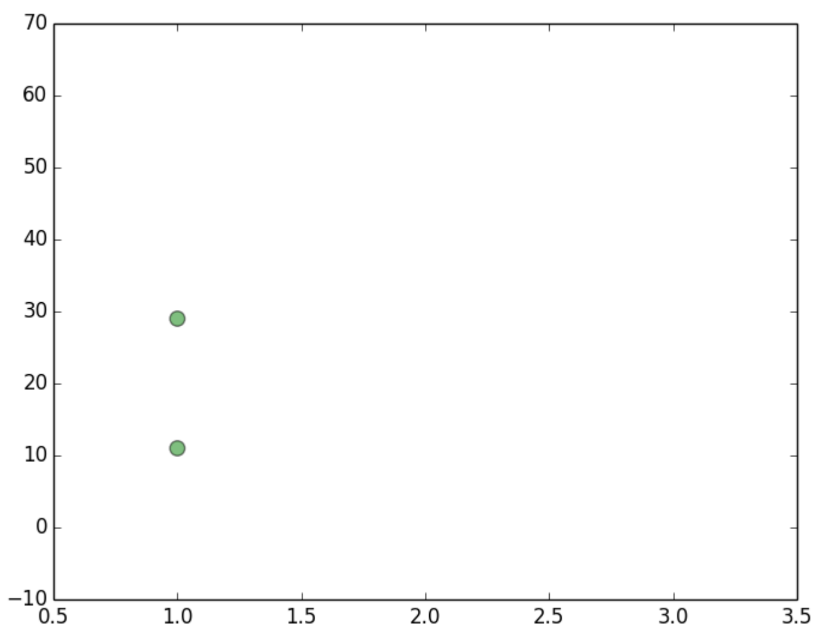


Figure 6: Demonstration of *k-means* on subset.

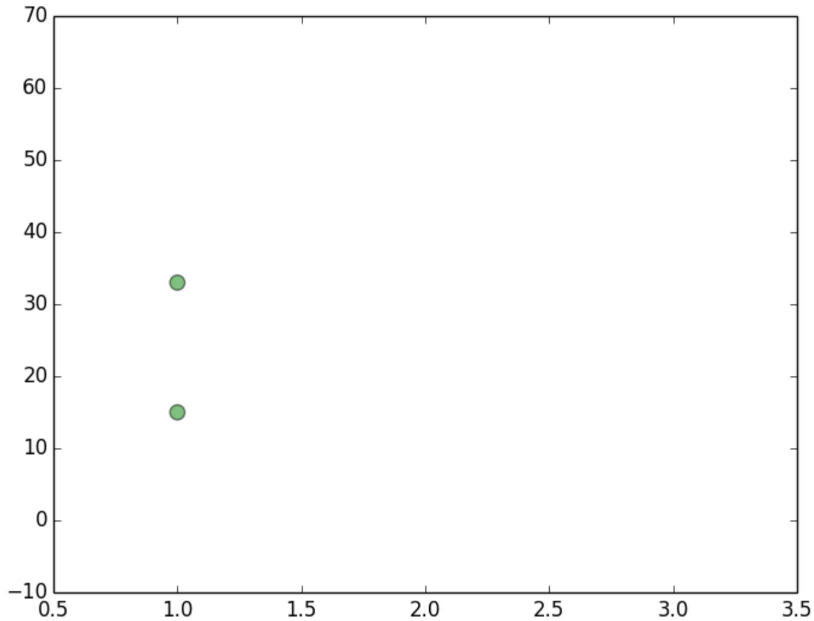


Figure 7: Demonstration of *k-means* on subset.

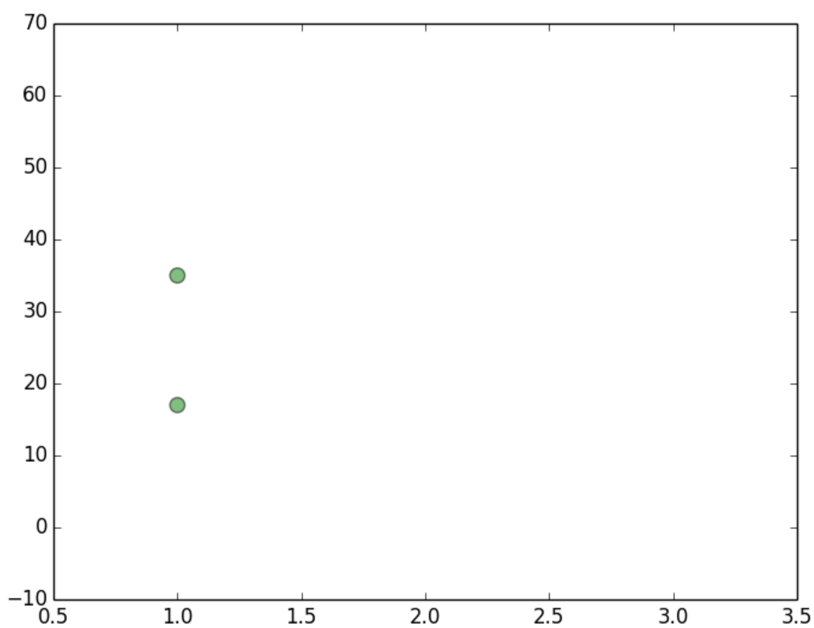


Figure 8: Demonstration of *k-means* on subset.

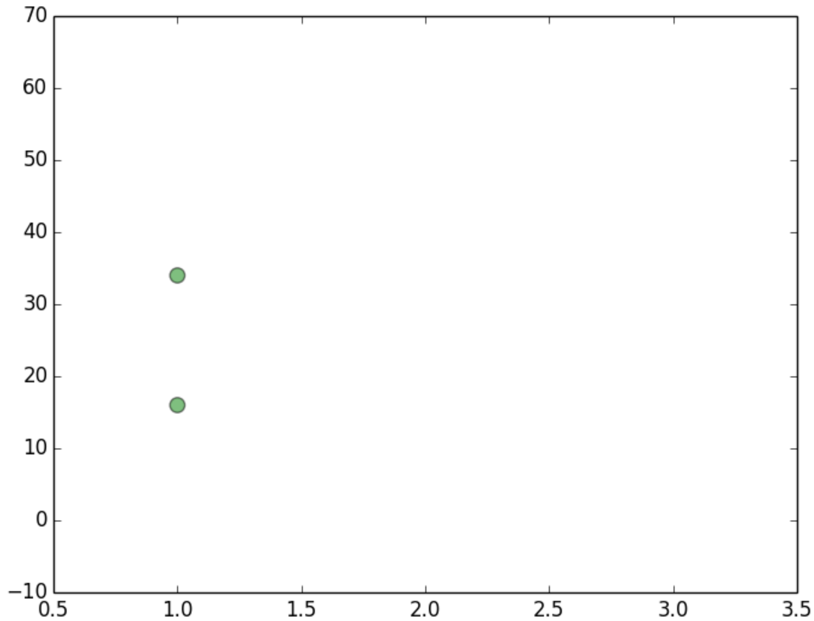


Figure 9: Demonstration of *k-means* on subset.

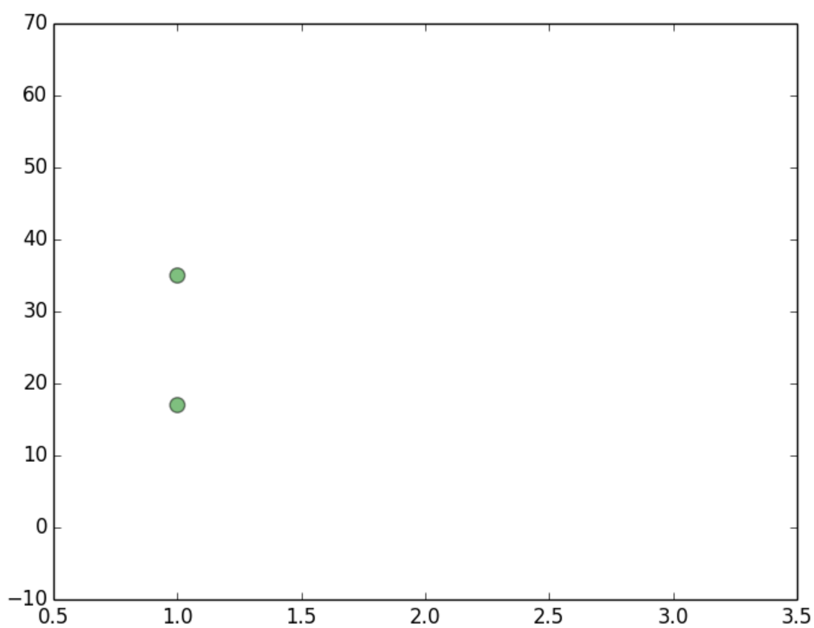


Figure 10: Demonstration of *k-means* on subset.

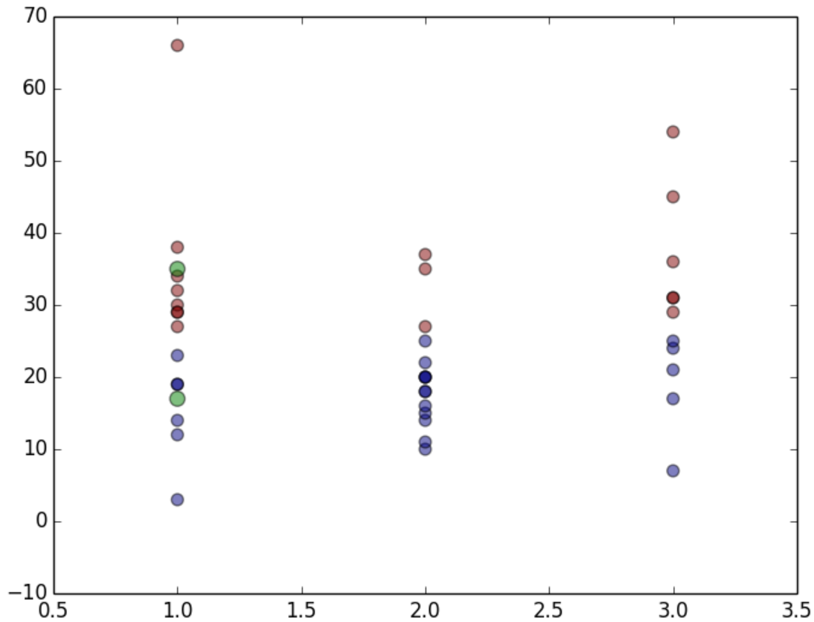


Figure 11: Demonstration of *k-means* on subset.

### 3.5 Assignmet-1 experience

1. Clear topics: Importance and implementation of k-means algorithm
2. Not-clear : Why should J decrease with increase in k ?
3. Difficulties: Plotting the results and inserting in latex.

## 4 References

1.Class notes
2.UCI's Machine Learning Repository (http://archive.ics.uci.edu/ml/machine-learning-databases/tae/)

## Bibliography

Christopher Bishop. *Pattern Recognition and Machine Learning.* Springer, 2006.