Python 语言实现 K-means 算法

总结报告

撰写人：孙宗敏

指导老师：陈凯

完成日期：**2012.2.8**

目录

1. 摘要

本次作业分两个部分。

第一部分需要完成 windows 环境下基于 Python 语言的 K-means 算法实现，这要求熟悉该算法的用途、详细流程、python 语言的运用，并且设计方法验证算法编写得是否正确。

第二部分需要在 windows 环境下安装 VMWARE 虚拟机，并在其上运行一个 centOS 系统，将前面完成的算法运行于该虚拟机上。

最后，需要写一份详细的技术报告，阐述自己对 K-means 算法的认识、作业中各个步骤的详细说明以及实际结果。

2. 作业要求

2.1. Python 实现 K-Means 算法

在windows或Linux下安装python 2.X版本

实现k-means算法

http://en.wikipedia.org/wiki/K-means_clustering

完成一个技术报告，说明python的安装使用，并说明k-means流程图，并设计验证 k-means 算法设计的试验（就是说，程序编完成了，如何验证编写是否正确）。

2.2. windows 下安装 VMWARE 虚拟机软件

在windows下安装VMWARE虚拟机软件，并安装一个Linux CENTOS 5.X的虚拟机，在CentOS上运行第一个python实现的k-means算法。

CentOS下载：http://centos.ustc.edu.cn/centos/5/

3. 作业一

3.1. 背景介绍

3.1.1.K-means

“In statistics and data mining, ***k*-means clustering** is a method of cluster analysis which aims to partition *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest mean. This results into a partitioning of the data space into Voronoi cells.”

——en.wikipedia.org

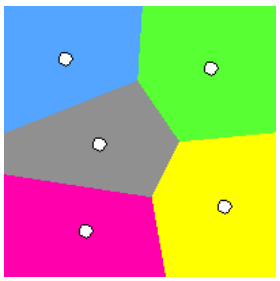
“给定 n 个对象的数据集 D，以及要生成的簇的数目 k，划分算法将对象组织为 k 个划分（k<=n），每个划分代表一个簇。这些簇的形成旨在优化一个目标划分准则，如基于距离的相异度函数，使得根据数据集的属性，在同一个簇中的对象是‘相似的’，而不同簇中的对象是‘相异的’。最著名和最常用的划分方法是 k 均值、k 中心点和它们的变种……k 均值算法以 k 为输入参数，把 n 个对象的集合分为 k 个簇，使得结果簇内的相似度高，而簇间的相似度低。簇的相似度是关于簇中对象的新均值。这个过程不断重复，直到准则函数收敛。”

——《Data Mining Concepts and Techniques 2nd》

“The k-means algorithm takes as input the number of clusters to generate, k, and a set of observation vectors to cluster. It returns a set of centroids, one for each of the k clusters. An observation vector is classified with the cluster number or centroid index of the centroid closest to it.”

——docs.scipy.org

本次作业中，实现该算法是作业一的主要任务。



3.1.2.Numpy & Scipy

“NumPy and SciPy are two of many open-source packages for scientific computing that use the Python programming language...SciPy (pronounced "Sigh Pie") is open-source software for mathematics, science, and engineering. It is also the name of a very popular conference on scientific programming with Python. The SciPy library depends on NumPy, which provides convenient and fast N-dimensional array manipulation. The SciPy library is built to work with NumPy arrays, and provides many user-friendly and efficient numerical routines such as routines for numerical integration and optimization. Together, they run on all popular operating systems, are quick to install, and are free of charge. NumPy and SciPy are easy to use, but powerful enough to be depended upon by some of the world's leading scientists and engineers. ”

——www.scipy.org

本次作业中，主要用它们来进行算法检验。

3.1.3.matplotpy

“ matplotlib is a python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. matplotlib can be used in python scripts, the python and ipython shell (ala MATLAB®* or Mathematica®†), web application servers, and six graphical user interface toolkits.”

——matplotlib.sourceforge.net

本次作业中，该工具主要用于实现 2D 的平面图像。

3.2. 算法分析

3.2.1.算法步骤

输入：

K：簇的数目

D：包含 n 个对象的数据集

输出：

K 个簇的集合

步骤：

1）从 D 中任意选择 K 个对象作为初始簇中心；

2）Repeat

3） 根据簇中的对象的均值，将每个对象（再）指派到最相似的簇；

4） 更新簇均值，即计算每个簇中对象的均值；

5）Until 不再发生变化

注：通常采用**平方误差准则**，即

$$E=\sum_{i=1}^{K}\sum_{p\in C_i}|\ p-m_i\ |^2$$

E 是数据集中所有对象的平方误差和，p 是空间中的点，表示给定对象，m_i 是簇 C_i 的均值。

均值的求法为

$$\mathbf{m}_i^{(t+1)}=\frac{1}{|S_i^{(t)}|}\sum_{\mathbf{x}_j\in S_i^{(t)}}\mathbf{x}_j$$

3.2.2.算法优缺点及其发展

K-means 的许多特征既是它的优点也是它的缺点：

1）该算法以欧几里得距离为度量标准，以方差为衡量簇的松散度的标准。这样使得 k 均值算法不适合于发现非凸形状的簇，或者大小差别很大的簇。此外，它对于噪声和离群点数据是敏感的，因为少量的这类数据能够对均值产生极大的影响。

2）该算法的 K 值是个输入值。不合理的 K 值会产生非常糟糕的结果。因此在使用算法前做一些 diagnostic checks 来决定 K 值是很有必要的。

K-means 的一个变体是 **K-modes 算法**，它扩展了 k 均值模式来聚类（cluster）分类数据，用簇的 mode 取代 mean，采用新的相异性度量处理分类对象，采用基于频率的方法更新 mode（即众数）。

EM 算法（Expectation-Maximization）以不同的方式对 k 均值模式进行了扩展。与 k-means

算法将每个对象指派到一个簇相反，在 EM 算法中，每个对象按照权重指派到每个簇，其中权重表示对象的隶属概率。

3.3. 实验环境

操作系统：Windows 7 旗舰版
开发平台：Eclipse Indigo
开发语言：Python 2.7.1

3.4. 代码分析

以下将逐个介绍实现K-means算法的必要函数：

```
def distortion(node1,node2):
    sum=0
    for i in range(F):
        dis=node1[i]-node2[i]
        sum=sum+dis*dis
    return sum
```

distortion函数计算节点node1和节点node2的欧几里得距离，**F为的属性个数**。

```
def assignCode(code_book,node):
    final_dis=-1
    code=-1

    for i in range(K):
        temp_dis=distortion(node,code_book[i])
        if final_dis is -1:
            final_dis=temp_dis
            code=i
            continue
        if final_dis >= temp_dis:
            final_dis=temp_dis
            code=i
    return code
```

assignCode函数计算结点node到各个中心的距离，然后为node选择最近的中心，该中心的序号为code。**code_book中包含了所有的中心的具体数值。K为中心数，即K-means的K**。

```
def isEnd(codes,code_book,data):
    new_E=0

    for n in range(N):
        new_E=new_E+distortion(data[n],code_book[codes[n]])
    if E[0] is -1:
        E[0]=new_E
        return False
    else:
        if E[0] == new_E:
            return True
        else:
            E[0]=new_E
            return False
```

isEnd函数判断算法中的循环是否结束，结束的话返回True。判断依据为

$$E=\sum_{i=1}^{K}\sum_{p\in C_i}|p-m_i|^2$$

当E趋向收敛时，循环停止。**N为结点数。data包含所有节点的具体数值。**

```
def updateCodebook(codes,code_book,data):
    count=[0 for x in range(K)]
    new_book=[[0 for x in range(F)] for y in range(N)]

    for n in range(N):
        count[codes[n]]=count[codes[n]]+1
        for f in range(F):
            new_book[codes[n]][f]=new_book[codes[n]][f]+data[n][f]

    for k in range(K):
        if count[k] is 0:
            continue
        for f in range(F):
                code_book[k][f]=new_book[k][f]/count[k]
```

updateCodebook函数更行code_book中的各个均值的具体数值。**codes中保留着各个节点的所属中心序号。**其中count记录了每个中心下连接的节点数，以供后面求均值使用。

更新方法为：

$$\mathbf{m}_i^{(t+1)}=\frac{1}{|S_i^{(t)}|}\sum_{\mathbf{x}_j\in S_i^{(t)}}\mathbf{x}_j$$

```
def k_means(code_book,data):
    global E
    E=[-1]
    global F
    F=len(code_book[0])
    global N
    N=len(data)
    global K
    K=len(code_book)
    codes=[-1 for x in range(N)]

    while not isEnd(codes,code_book,data):

        for n in range(N):
            codes[n]=assignCode(code_book,data[n])
        updateCodebook(codes,code_book,data)
    updateCodebook(codes,code_book,data)

    return code_book,codes
```

K_means函数中实现了算法的整个流程。

```
if __name__ == '__main__':

    random.seed()
    F=random.randint(1,10)
    N=random.randint(5,50)
    K=int((N+9)/10)
    data=[]
    code_book=[]
    for n in range(N):
        data.append([random.random() for f in range(F)])
    for k in range(K):
        code_book.append([random.random() for f in range(F)])

    code_book,codes=k_means(code_book,data)
    print "generated code_book:"
    for i in range(K):
        print code_book[i]
    print "generated codes:"
    for i in range(N/K):
        print codes[i:i+N/K]
```

Main 函数中可见，节点数和节点的属性数是随机的。K 取了节点数的十分之一（进一步的改进是加入 diagnostic checks 来决定 k 的取值）。data 中的每个节点的各个属性取值都是随机的（在[0.0,1.0]之间）。**code_book 中的初始中心的取值也是随机的（也可以改成从 data 中抽取）**。

该部分代码保存在 kmeans.py 中。

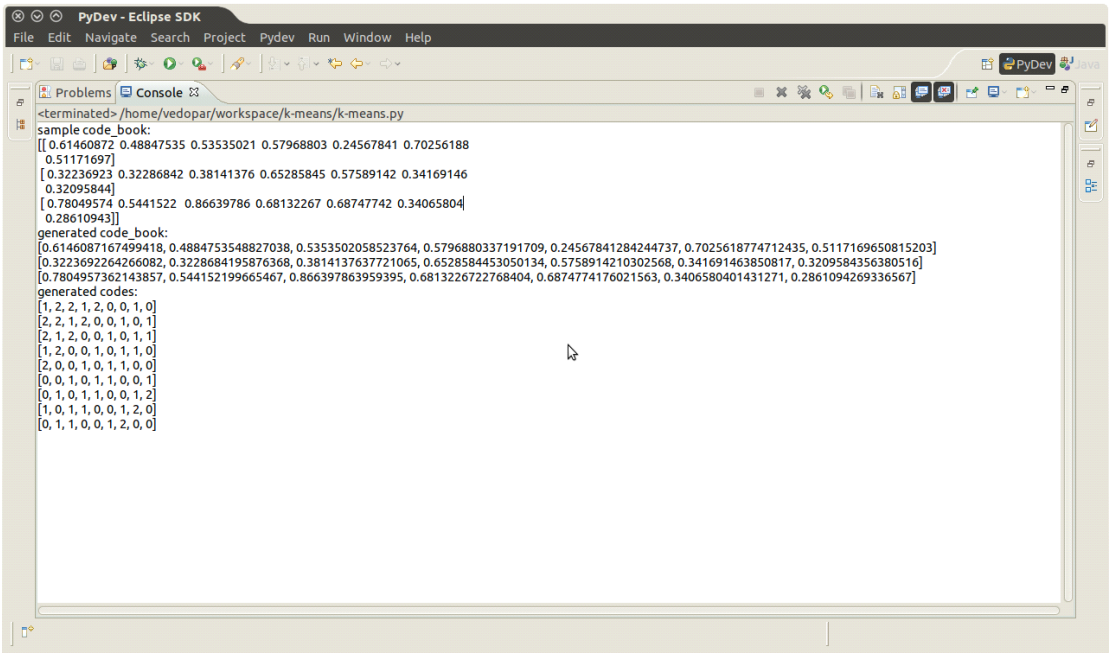
3.5. 实现与算法检验

本次检验使用 Scipy&Numpy 中的 kmeans 接口作为结果对照。具体代码如下：

```
from numpy import array
from scipy.cluster.vq import vq,kmeans

print "sample code_book:"
print kmeans(array(data),array(code_book))[0]
```

由于 windows 下配置 scipy 和 numpy 所需环境比较繁琐，该检验在 ubuntu10.04 下进行：

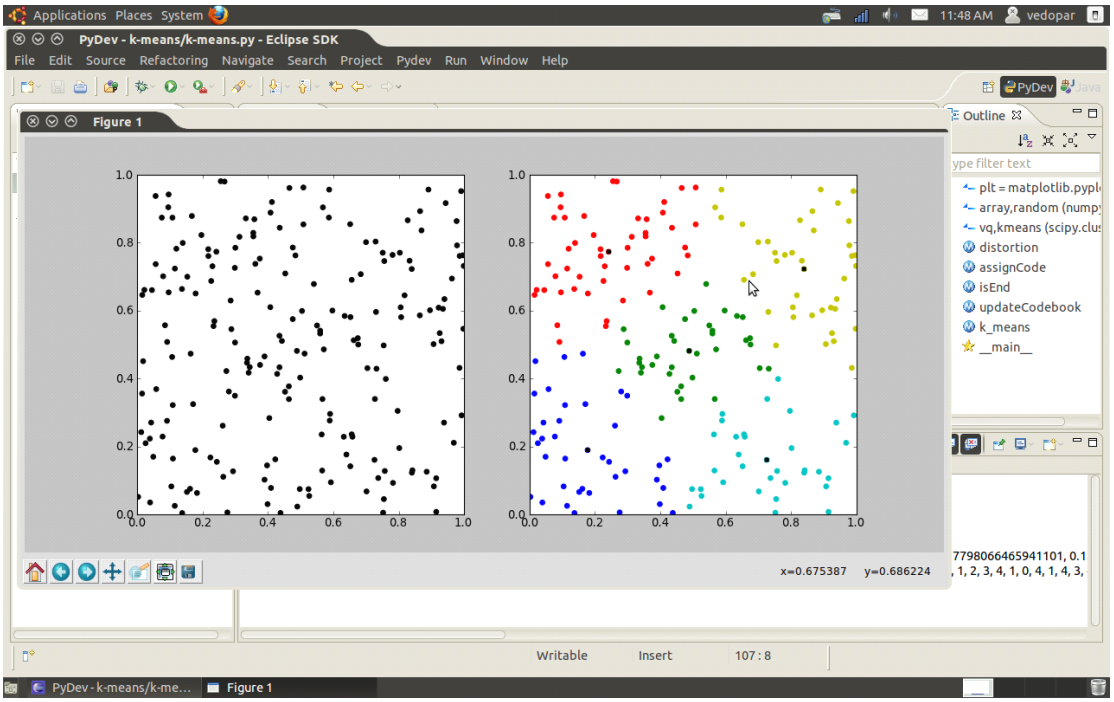


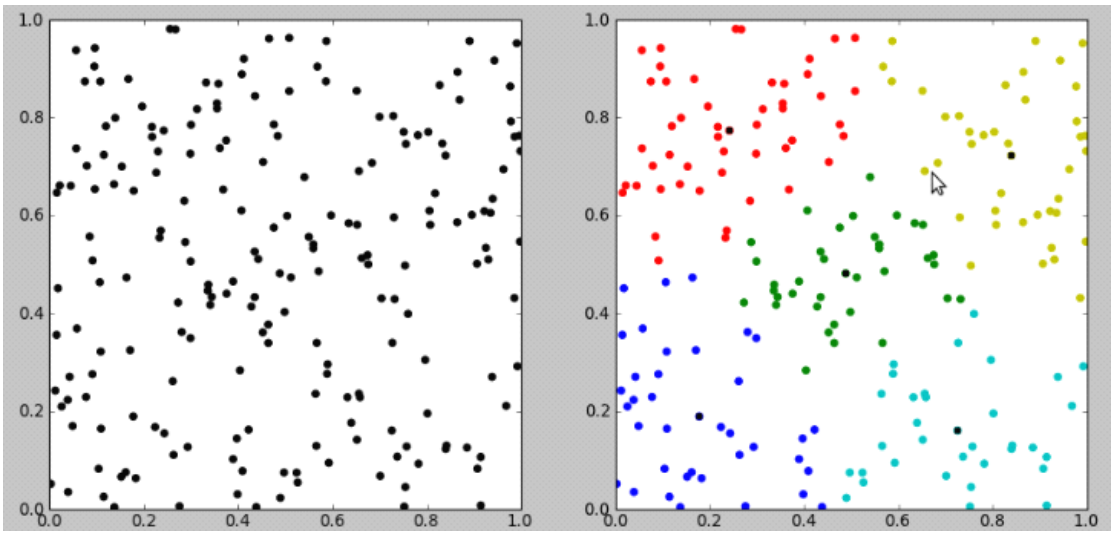

```
sample code_book:
[[ 0.61460872 0.48847535 0.53535021 0.57968803 0.24567841 0.70256188
  0.51171697]
 [ 0.32236923 0.32286842 0.38141376 0.65285845 0.57589142 0.34169146
  0.32095844]
 [ 0.78049574 0.5441522  0.86639786 0.68132267 0.68747742 0.34065804
  0.28610943]]
generated code_book:
[0.6146087167499418, 0.4884753548827038, 0.5353502058523764, 0.579688
[0.3223692264266082, 0.3228684195876368, 0.3814137637721065, 0.652858
[0.7804957362143857, 0.544152199665467, 0.866397863959395, 0.68132267
```

由图可见 scipy 标准库中的 kmeans 接口生成的“sample code_book”与作业中编写出的 kmeans 的结果“generated book”一样（精度不同）。**该部分代码保存在**

kmeans-test.py 文件中。

除此以外，我还利用 matplotpy 做了一个图形化的演示，**代码保存在 kmeans-map.py 文件中**。这里取 N=200，K=5，初始中心取 data 的前 K 个值。其他一样。结果如图：





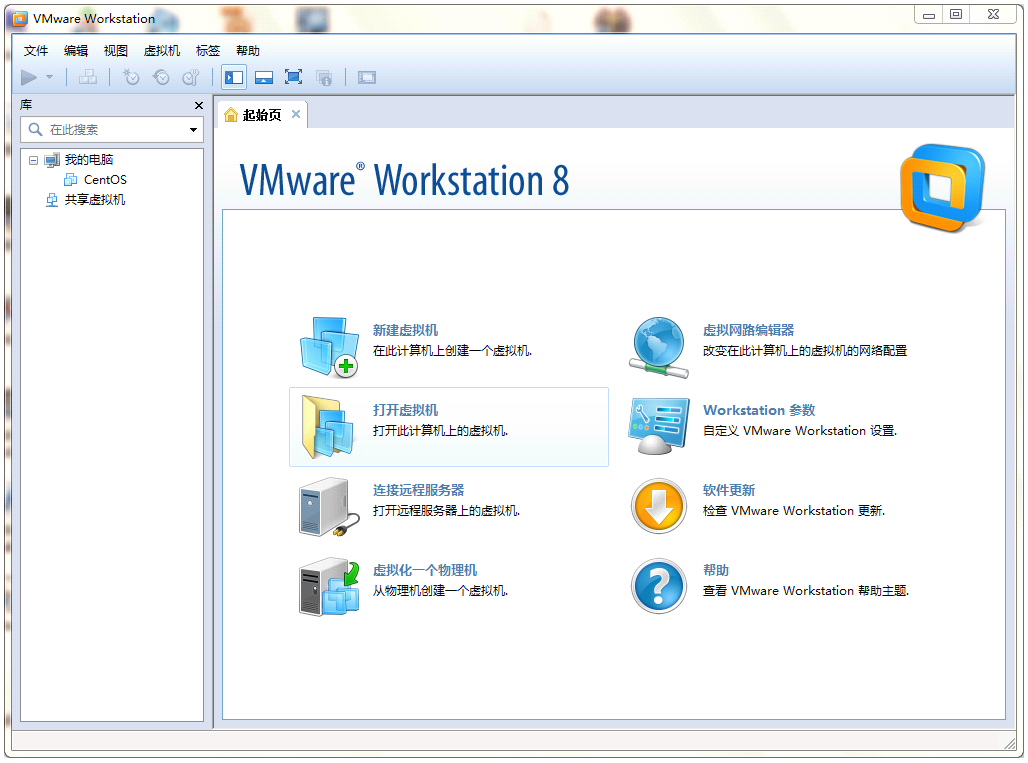
如图所示，左边图为未处理数据。右边图为处理过数据，其中黑点为新的中心点。

4. 作业二

4.1. 背景介绍

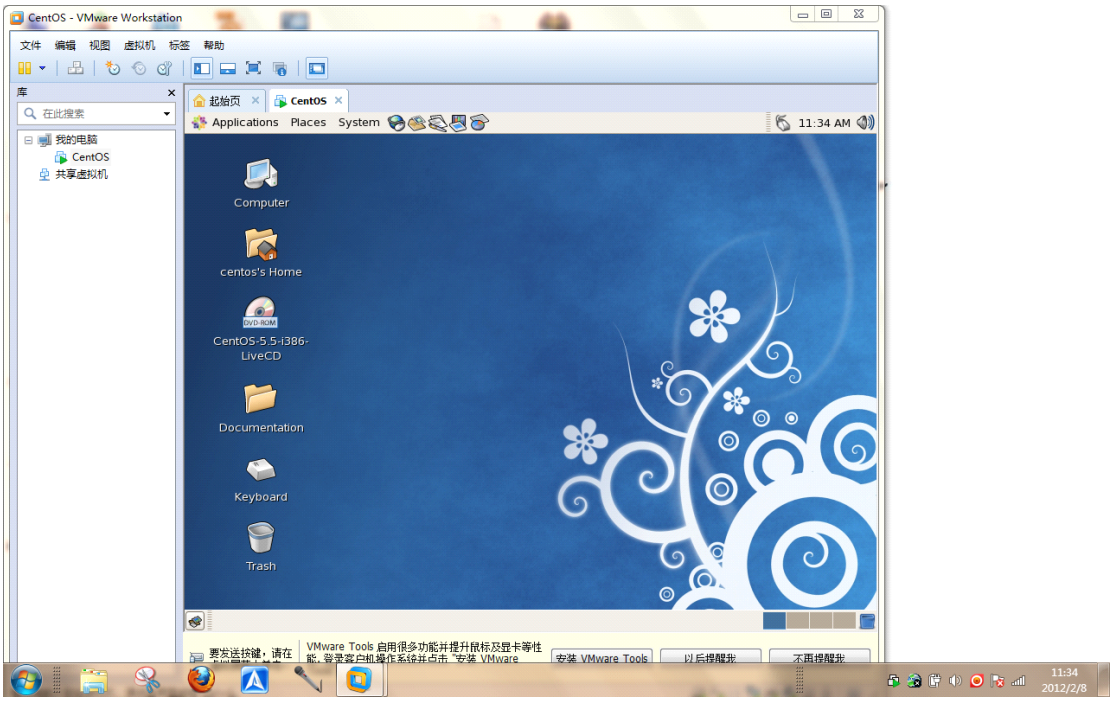
4.2. VMWARE 的安装

从 VMWARE 官网上下载试用版的 VMWARE Workstation，在 windows 7 环境下安装。安装完成以后打开如下。



4.3. centOS 的安装

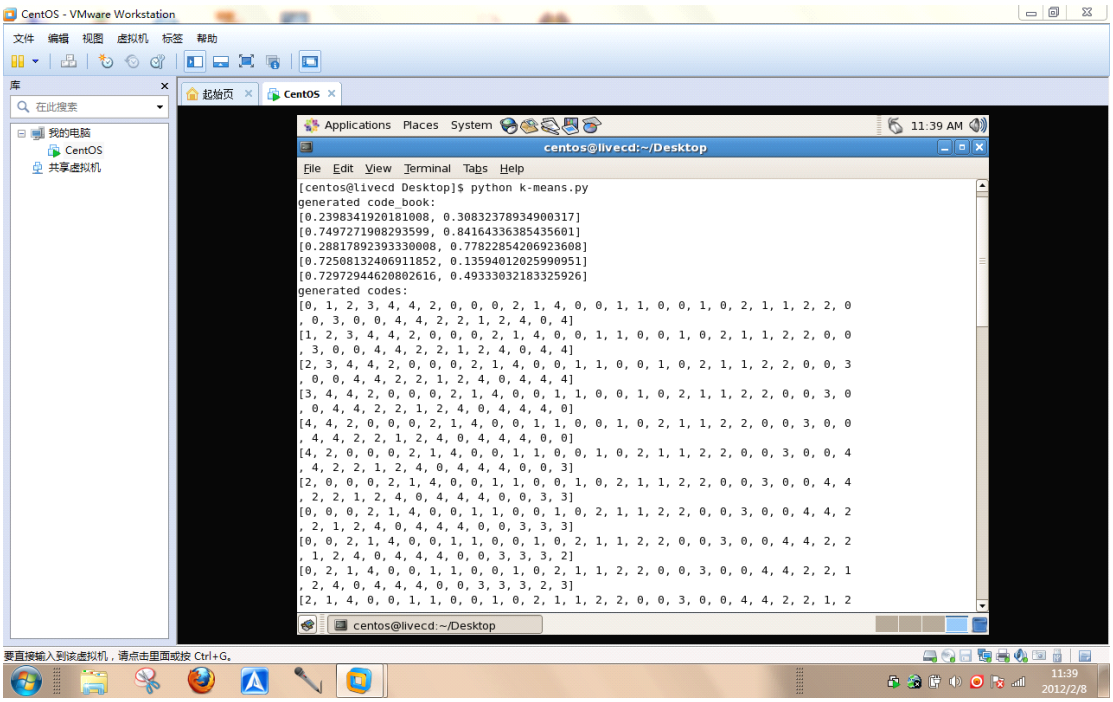
下载 centOS 的 liveCD 的 iso（本次作业中用的），新建虚拟机，加载该 iso。
centOS 版本：CentOS-5.5-i386-LiveCD



centOS 虚拟机运行

4.4. 运行作业一的脚本

由于 centOS 自带 python2.4，故可以直接在虚拟机中运行 python 脚本。运行结果如下。



5. 参考文献

[1]"data mining concepts and techniques 2nd".Jiawei Han , Micheline Kamber . 2007
[2]*k*-means clustering, http://en.wikipedia.org/wiki/K-means_clustering
[3]k-means clustering implementation in Python with scipy,
http://docs.scipy.org/doc/scipy/reference/cluster.vq.html
[4]matplotlib 官网, http://matplotlib.sourceforge.net/
[5]The Python Tutorial , http://docs.python.org/tutorial/

6. 总结

本次作业陈老师给了我们一个寒假的时间，但很惭愧由于过年种种事情拖沓，着手开始这项作业却比较晚，虽然如此，但是我也尽力认真去对待。由于直接看 wikipedia 的关于 k-means 算法的介绍，其中有很多专业的概念，看得自己云里雾里的，只好先找来一本数据挖掘的书细细从头看。认真的过了 3-4 章以后我，开始有了头绪，再看原来的介绍明白了许多。

通过多方学习，我明白了 k-means 算法的用途非常的广，我本来想做一个压缩图片的基于 k-means 的 python 应用，由于时间限制，未完成，先提交一个简单的用 matplotlib 画出的 k-means 处理数据图。由于涉及到专业的科学运算，在 windows 下的安装 scipy&numpy 等工具需要安装许多其他的专业软件做基础。我在 ubuntu 下用 apt-get 解决这些依赖问题却非常简单，所以一想，索性直接在 ubuntu 下作检验工作了，这里偷了个懒，请老师见谅。

写这篇总结报告已经是 2 月 8 日截止日期的晚上了，虽然完成了作业的各个要求，但是我自己总觉得做得比较仓促，本可以做得更好，心里不免有些自责，也不免有这里那里不足的地方，恳请陈老师指正！