K-means1.py：

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

import seaborn as sns

import tensorflow as tf

num\_vectors = 1000

num\_clusters = 3

num\_steps = 100

vector\_values = []

for i in range(num\_vectors):

if np.random.random() > 0.5:

vector\_values.append([np.random.normal(0.5, 0.6),

np.random.normal(0.3, 0.9)])

else:

vector\_values.append([np.random.normal(2.5, 0.4),

np.random.normal(0.8, 0.5)])

#df = pd.DataFrame({"x": [v[0] for v in vector\_values], "y": [v[1] for v in vector\_values]})

#sns.lmplot("x", "y", data=df, fit\_reg=False, size=7)

vectors = tf.constant(vector\_values)

centroids = tf.Variable(tf.slice(tf.random\_shuffle(vectors), [0,0],[num\_clusters,-1]))

expanded\_vectors = tf.expand\_dims(vectors, 0)

expanded\_centroids = tf.expand\_dims(centroids, 1)

#print expanded\_vectors.get\_shape()

#print expanded\_centroids.get\_shape()

distances = tf.reduce\_sum(tf.square(tf.sub(expanded\_vectors, expanded\_centroids)), 2)

assignments = tf.argmin(distances, 0)

means = tf.concat(0, [

tf.reduce\_mean(

tf.gather(vectors,

tf.reshape(

tf.where(

tf.equal(assignments, c)

),[1,-1])

),reduction\_indices=[1])

for c in range(num\_clusters)])

update\_centroids = tf.assign(centroids, means)

init\_op = tf.initialize\_all\_variables()

with tf.Session() as sess:

sess.run(init\_op)

for step in range(num\_steps):

\_, centroid\_values, assignment\_values = sess.run([update\_centroids,

centroids,

assignments])

# print "centroids"

# print centroid\_values

data = {"x": [], "y": [], "cluster": []}

for i in range(len(assignment\_values)):

data["x"].append(vector\_values[i][0])

data["y"].append(vector\_values[i][1])

data["cluster"].append(assignment\_values[i])

df = pd.DataFrame(data)

sns.lmplot("x", "y", data=df,

fit\_reg=False, size=7,

hue="cluster", legend=False)

plt.show()

注意：pandas在python3才支持，然而python3里是range()不是xrange()。

