sheet07

Unknown Author

November 27, 2013

Exercise Sheet 7: K-means Clustering (100 P) **Intro:** In this scenario we will simulate finding good locations for warehouses of a multi-national company and some scenarios which can occur. The company is expanding to Europe and surrounding countries and is looking for locations which would maximize its profits. We will attempt to find the number of warehouses and their locations by using k-means clustering. We will look at several scenarios and how they impact the proposed model.

In this exercise we will using data available at: http://sedac.ciesin.columbia.edu/ . The data contains a number of (x,y) coordinates on the map, the number of inhabitants of that section and the corresponding country.

In [46]:
```
import pickle,numpy
import scipy
import scipy.spatial.distance
import os
import random
import io,IPython,Image
import matplotlib.pyplot as plt
```

The following helper function reads the data provided. It returns:

- **xnid** : an array of integers indicating the country at each latitude/longitude on the map
- **xpop** : an array of integers counting the population size at each latitude/longitude on the map
- **nx** : the number of latitudes considered
- **ny** : the number of longitudes considered

In [29]:
```
def getData():

    def reading(filename):
        with open(filename, 'rb') as st:
            source = pickle.loads(st.read())
        return source

    xpop = reading('data/pop.pkl')
    xpop = numpy.array([xpop[i::5,j::5] for i in range(5) for j in range(
    nx,ny = xpop.shape


    xnid = reading('data/nid.pkl')
    xnid = xnid[2::5,2::5]
    return xnid, xpop, nx,ny
```

Part 1: Examine the Data and Generate Various Maps (10 P) In this section we will utilize the data to show 4 different maps:

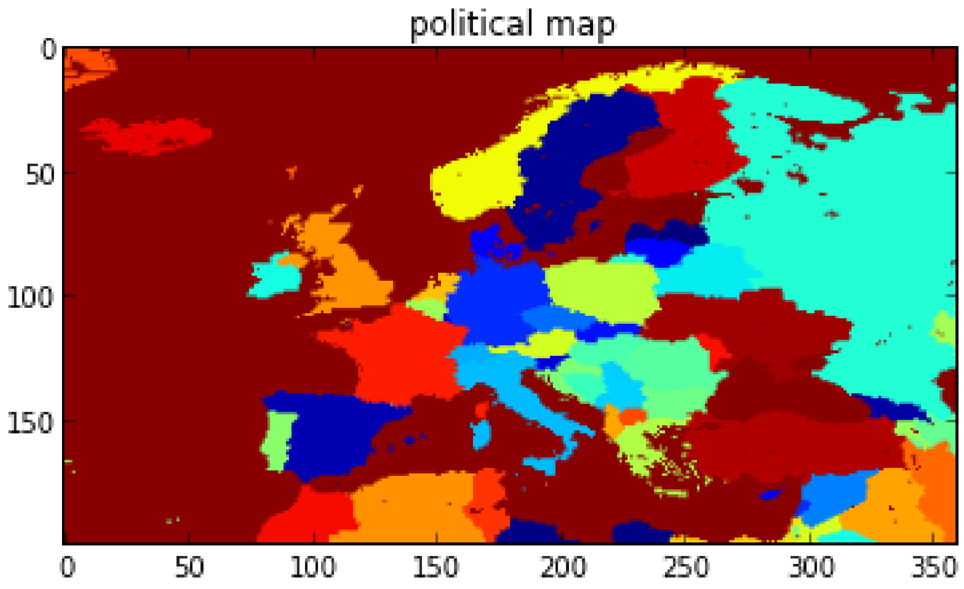
- A map containing the borders between countries in our data (already provided for you)
- A map where every country is colored differently (or at least assignes different colors to neighbouring countries)

- A map which shows population density in our data
- A map that combines border indicators and population density

Sample maps are given in the folder /maps. Your maps can be generated using the function imshow of the matplotlib library. The last hybrid map will be used in the rest of the exercise sheet.

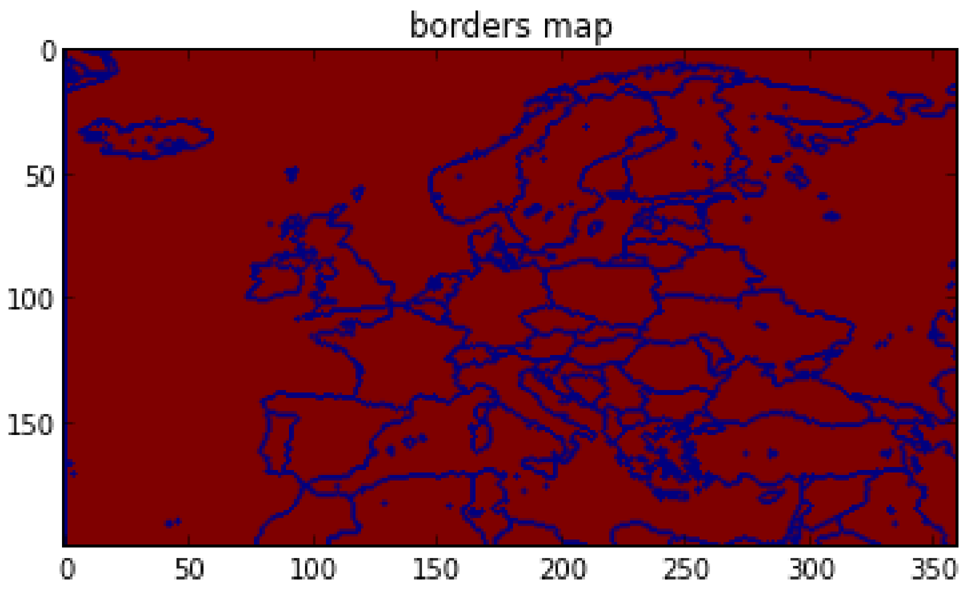
In [47]:
```
# 1a: Political map
xnid,xpop,nx,ny = getData()
plt.imshow(xnid*(1029743.9384958475)%1)
plt.title('political map')
```

Out [47]:
```
<matplotlib.text.Text at 0xefb8d0c>
```



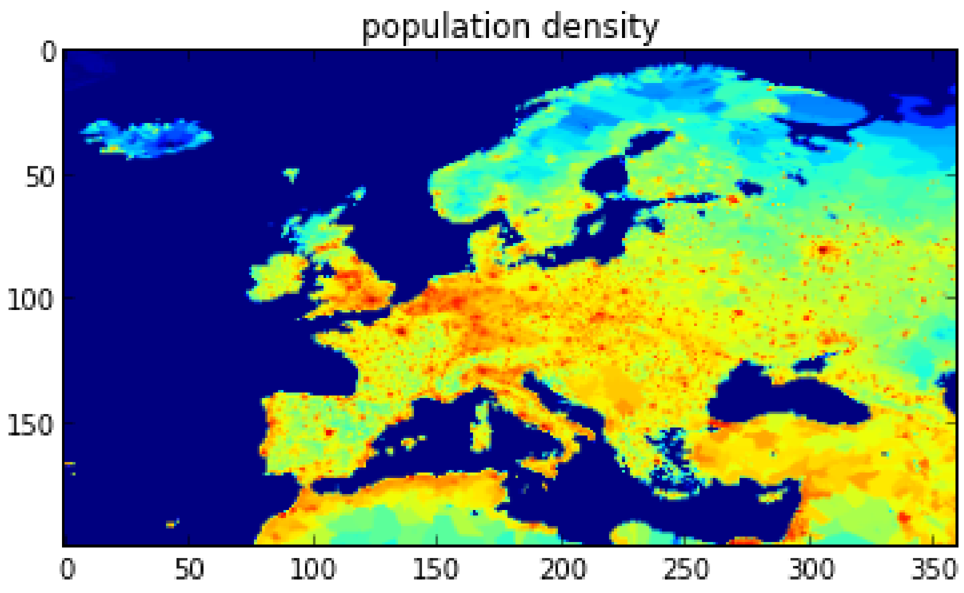
In [48]:
```
# 1b: Map showing the borders between countries
xbor = xnid*0
xbor[1:-1,1:-1] = ((xnid[1:-1,1:-1] == xnid[1:-1,2:]) *
    (xnid[1:-1,1:-1] == xnid[1:-1,:-2]) *
    (xnid[1:-1,1:-1] == xnid[2:,1:-1]) *
    (xnid[1:-1,1:-1] == xnid[:-2,1:-1]))
plt.imshow(xbor)
plt.title('borders map')
```

Out [48]:
```
<matplotlib.text.Text at 0xb46dfac>
```

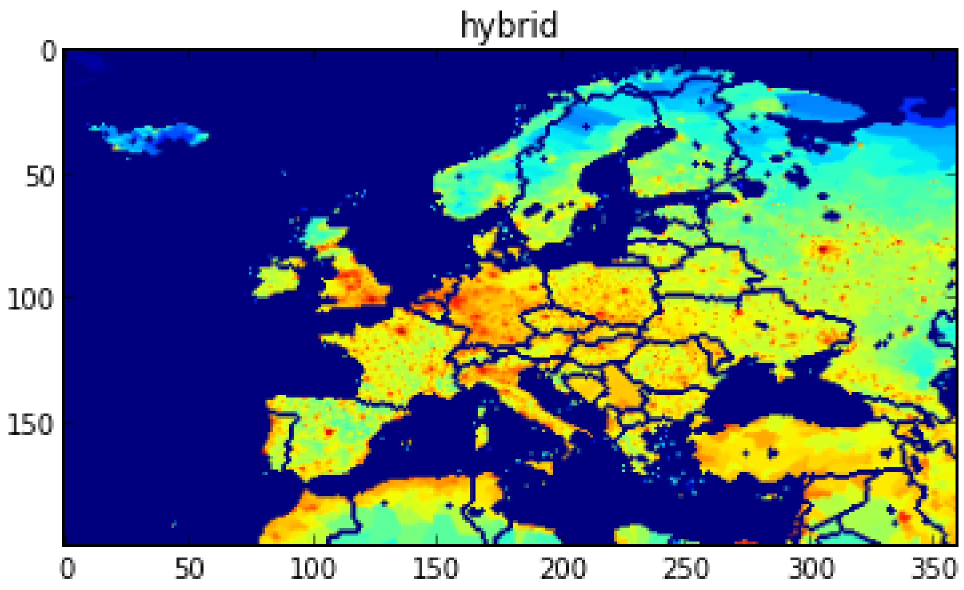


In [49]:
```
# 1c: Map showing population density
plt.imshow(numpy.log(1+xpop))
plt.title('population density')
```

Out [49]:
```
<matplotlib.text.Text at 0xe13640c>
```



```
In [50]: # 1c: Map superposing population density and borders
         plt.imshow(numpy.log(1+xpop)*xbor)
         plt.title('hybrid')

Out [50]: <matplotlib.text.Text at 0xf82292c>
```



Part 2: Implementing one step of weighted k-means (20 P) Your task is to implement the k-means algorithm using weighted data. Here, instead of considering each inhabitant as a separate point, we consider each geographic coordinate as an inhabitant with a different weight corresponding to the number of people living at this coordinate.

- **initial centroids**: the x and y coordinates of initial centroids
- **points** : x and y coordinates of points with number of inhabitants > 0
- **weights** : Containing the number of people living in every section

The function should output the following:

- **final centroids** : the final centroids after many iterations
- **cost function** : the value of the cost function $J(c) = \sum_{x_i \in \text{points}} w_i ||x_i - c(x_i)||_2^2$ where $c(x_i)$ denotes the centroid associated to x_i and w_i is the weight of each data point x_i.

```
In [51]: def wkmeans(centroids, points, weights):

             d = scipy.spatial.distance.cdist(centroids,points,'sqeuclidean')

             J = (d.min(axis=0) * w[:,0]).sum()

             ind = numpy.argmin(d,axis=0)

             wpoints = weights*points
             for i in range(len(c)):
                 c[i] = wpoints[ind==i,:].mean(axis=0)/weights[ind==i].mean(axis=0

             return centroids,J
```

Part 3: Building an initialization heuristic (20 P) Here, you are requested to implement an heuristic for choosing initial centroids. The idea is the following: We assign a random score $s(i, j)$ to each geographical coordinate (i, j). The score is computed as $s(i,j) = w(i,j) - N$ where $w(i,j)$ is the number of inhabitants at a certain location and N is a random variable drawn from an exponential distribution of scale parameter λ to be determined. Then, the k locations with largest score are assigned a centroid.

(a) Implement the initialization heuristic

(b) Generate a map superposing the centroids to the population density map

(c) Test different parameters λ and k

In [52]:
```
def initialize(pop,k):
    popn = pop - numpy.random.exponential(25000000,pop.shape)
    popt = numpy.sort(popn.flatten())[-k]
    cx,cy = numpy.nonzero((popn>=popt)*1.0)
    return numpy.array([cx,cy]).T
```

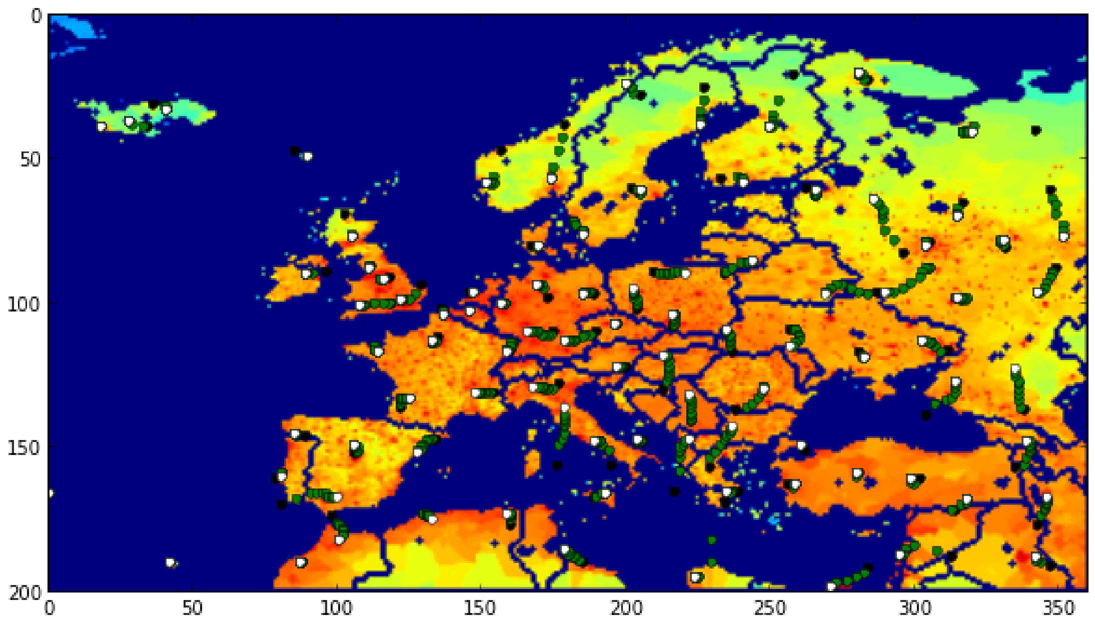
Part 4: Running weighted k-means (20 P) The company wishes to find good locations of warehouses. We assume that the same percentage of inhabitants of every country uses their services, so the number of users can be approximated with the number of inhabitants. We assume that the cost of transporting their products to users is proportional to the squared euclidian distance to the closest warehouse.

(a) Run the k-means algorithm with the number of clusters $k = 100$. Save your results, as they will be used later. Run at least 20 iterations of k-means.

(b) Plot the path of the centroids throughout the training procedure (for example, the initial centroids as a black dot, the final centroids as a white dot and the path that they followed as a black line). The path should be plotted in superposition to the hybrid map built in Part 1.

In [45]:
```
xnid, xpop, nx,ny = getData()

c = initialize(xpop,100)
gx,gy = numpy.nonzero(xpop)
w = numpy.array([xpop[x,y] for x,y in zip(gx,gy)])[:,numpy.newaxis]
g = numpy.array([gx,gy]).T

f = plt.figure(figsize=(10,10))
p = f.add_subplot(1,1,1)
p.imshow(numpy.log(0.001+xbor*xpop))

# start k-means
for t in range(10):

    c,J = wkmeans(c,g,w)
    print(J)
    if t == 0: p.plot(c[:,1],c[:,0],'o',mec='black',mfc='black',ms=5)
    elif t == 9: p.plot(c[:,1],c[:,0],'o',mec='black',mfc='white',ms=5)
    else: p.plot(c[:,1],c[:,0],'o',mec='black',mfc='green',ms=5)

p.axis([0,ny,nx,0])
```

```
130067180547.0
74413084630.3
65226655381.6
59722916706.5
56544180772.0
54088168039.9
53244317424.8
52574234266.3
52114862277.7
51797337973.0
```



Part 5: Focusing on the German market (10 P) We have determined that our initial assumption that the number of customers in a country is proportional to its number of inhabitants. Analysis has shown that customers in Germany (xnid=111), Austria(xnid=104) and Switzerland (xnid=109) are three times more likely to buy the product than customers in other countries.

(a) Explain how the model should be modified to take into account this new constraint

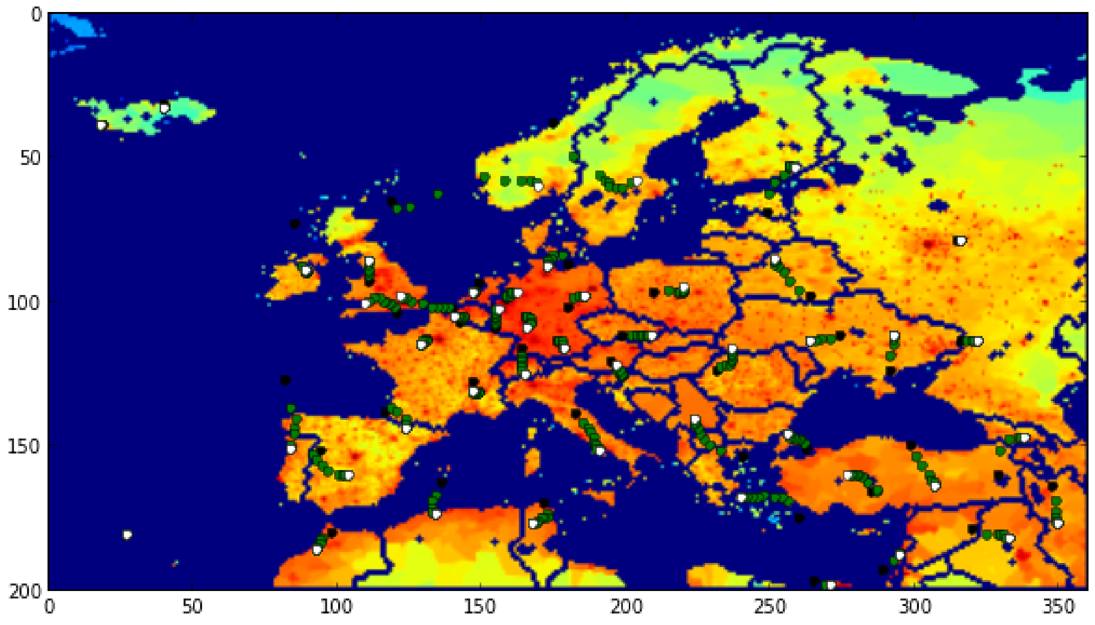
(b) Run k-means on this new problem with $k = 50$ and show the same type of map as in Part 5 for this new setting.

(c) Describe the qualitative change in the allocation of factories accross Europe.

In [54]:
```
xnid, xpop, nx,ny = getData()

xpopnew = xpop*(0.4+0.6*(xnid==104)+0.6*(xnid==109)+0.6*(xnid==111))
xpop = xpopnew / xpopnew.mean() * xpop.mean()

c = initialize(xpop,50)
gx,gy = numpy.nonzero(xpop)
w = numpy.array([xpop[x,y] for x,y in zip(gx,gy)])[:,numpy.newaxis]
g = numpy.array([gx,gy]).T

f = plt.figure(figsize=(10,10))
p = f.add_subplot(1,1,1)
p.imshow(numpy.log(0.001+xbor*xpop))

# start k-means
for t in range(10):

    c,J = wkmeans(c,g,w)
    print(J)
    if t == 0: p.plot(c[:,1],c[:,0],'o',mec='black',mfc='black',ms=5)
    elif t == 9: p.plot(c[:,1],c[:,0],'o',mec='black',mfc='white',ms=5)
    else: p.plot(c[:,1],c[:,0],'o',mec='black',mfc='green',ms=5)

p.axis([0,ny,nx,0])
```

```
256088005691.0
151770074547.0
128313187074.0
118108414270.0
112676274179.0
109198241839.0
106831559872.0
105370187467.0
104648711043.0
104094669955.0
```

```
Out [54]: [0, 360, 200, 0]
```



Part 6: Shipping restrictions (10pt) Due to new shipping regulations in the EU, transport the product across borders is now taxed heavily.

(a) Explain how the new problem can be understood as a k-means problem in a higher-dimensional space (i.e. extending the original longitude/latitude two-dimensional space to more dimensions).

(b) Implement the higher-dimensional k-means problem and produce a map similar to the one produced in Part 5.

(c) Explain what are the qualitative difference between the distribution of factories as a function of the level of cross-border taxation.

```
In [ ]:
```

Part 7: Discussion (10pt) Discuss in approximately three paragraphs what are the advantages and limitation of the k-means model to the problem of optimal resource allocation. The discussion could cover points such as validity of the Euclidean distance as a measure of shipping cost, temporal evolution of consumer demand, supply side constraints. Then, discuss how the basic model could be extended to these more realistic scenarios.Submission guidelines To facilitate grading, please export the notebook to PDF format. This can be done easily by installing the required packages and running

ipython nbconvert –to latex sheet07.ipynb && pdflatex sheet07.tex