Machine Learning User Domain Discovery

Olgierd Grodzki, Mateusz Rzeszutek

Institute of Applied Computer science
AGH University of Science and Technology

Geist Winter of Code
April 19, 2013, Kraków

http://geist.agh.edu.pl

Outline

Presentation Outline

Problem description

The goal of this project is to create an Android application which changes its user interface depending on the user's location - the collected GPS data has to be clustered into 'domains' using a clustering algotrithm and possibly some other data.

Problem description, continued

For algorithm prototyping we used the Python Programming Language with NumPy, SciPy, pandas and matplotlib libraries.

Presentation Outline

K-means

Input:

- K – number of clusters
- $\{x^{(1)}, ..., x^{(m)}\}$ – training set

Algorithm:

1. Randomly initialize K cluster centroids $\mu_1, \mu_2, \ldots, \mu_K \in \mathbb{R}^n$
2. Repeat until convergence:
   1. **Cluster assignment step** – for $i \in \{1, ..., m\}$:
      $c^{(i)}$:= index (from 1 to K) of cluster centroid closest to $x^{(i)}$
      $(\arg\min_k \|x^{(i)} - \mu_k\|^2)$
   2. **Update step** – for $k \in \{1, ..., K\}$:
      μ_K := average (mean) of points assigned to cluster k

Distortion

The distortion is defined as follows:

$$\frac{1}{p} \min_{\mathbf{c_1},\ldots,\mathbf{c_K}} E\left[(\mathbf{X}-\mathbf{c_X})\Gamma^{-1}(\mathbf{X}-\mathbf{c_X})^T\right]$$

where

- $\mathbf{X}$ – a p-dimensional random variable; a mixture of G components
- Γ – covariance matrix
- $\mathbf{c}_1, \ldots, \mathbf{c}_K$ – a set of all K cluster centers
- $\mathbf{c_X}$ – the closest cluster center to a given sample of X
- E – expected value

Calculating distortion

It is impossible to calculate a minimum for **all** possible sets of clusters. Distortion calculation is usually implemented in one of those two ways:

- calculate the distortion for some chosen set of clusters, or
- calculate distortions for a few sets of clusters, and choose the smallest value

Distortion function implementation

Our implementation:

```
import numpy as np
import scipy.linalg as la

def distortion(data, idx, centroids, gamma = None):
    # data dimensions
    M, N = data.shape
    K = len(centroids)

    # if no covariance matrix is passed, use an identity matrix
    # in this case the distortion is simply mean squared error
    if gamma is None:
        gamma = np.eye(N)
    cov = np.matrix(la.inv(gamma))

    # calculate distortion
    distortion = 0
    for i in range(M):
        temp = np.matrix(data[i] - centroids[idx[i]])
        distortion += temp * cov * temp.T
    distortion = distortion / (M * N)

    return distortion
```

Finding the number of clusters in a data set

An information theoretic approach algorithm [1]:

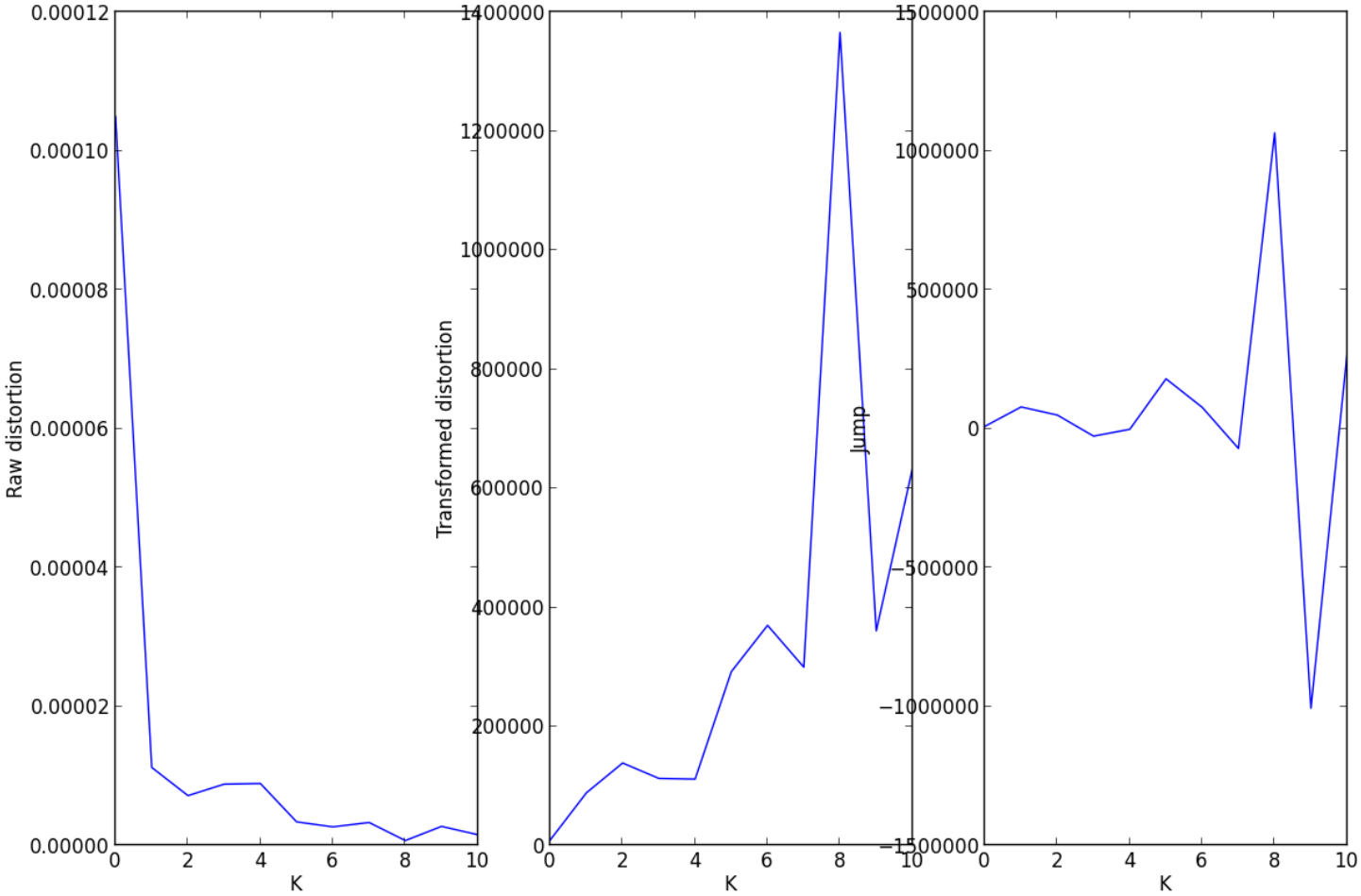
1. compute distortion $d(k)$ for all $1 < K < n$, using a standard clustering algorithm (K-means)
2. choose a transformation power, $Y > 0$; a typical value is $\frac{p}{2}$
3. transform the distortion curve by a negative power: $D(K) = d(K)^{-Y}$
4. calculate jumps: $J(K) = D(K) - D(K-1)$
5. the largest jump ($K^* = \arg\max_K J(K)$) represents the best choice for the number of clusters

An information theoretic approach – implementation

Our implementation:

```
import numpy as np
import scipy.cluster.vq as cluster

def jump_method(data, n = None, max_iterations = 10):
    M, N = data.shape
    if n is None:
        n = int(np.sqrt(M))
    Y = 0.5 * N
    tf_dist = np.zeros(n + 1)
    jump = np.zeros(n)

    # for all k = 1..n
    for k in range(1, n + 1):
        centroids, idx = cluster.kmeans2(data, k, minit = 'points',
                                         iter = max_iterations)
        # calculate distortion
        dist = distortion(data, idx, centroids)
        # calculate transformed distortion
        tf_dist[k] = dist[k - 1]**(-Y)

    for i in range(n):
        # calculate jumps
        jump[i] = tf_dist[i + 1] - tf_dist[i]

    return np.argmax(jump)
```

Example distortion curves

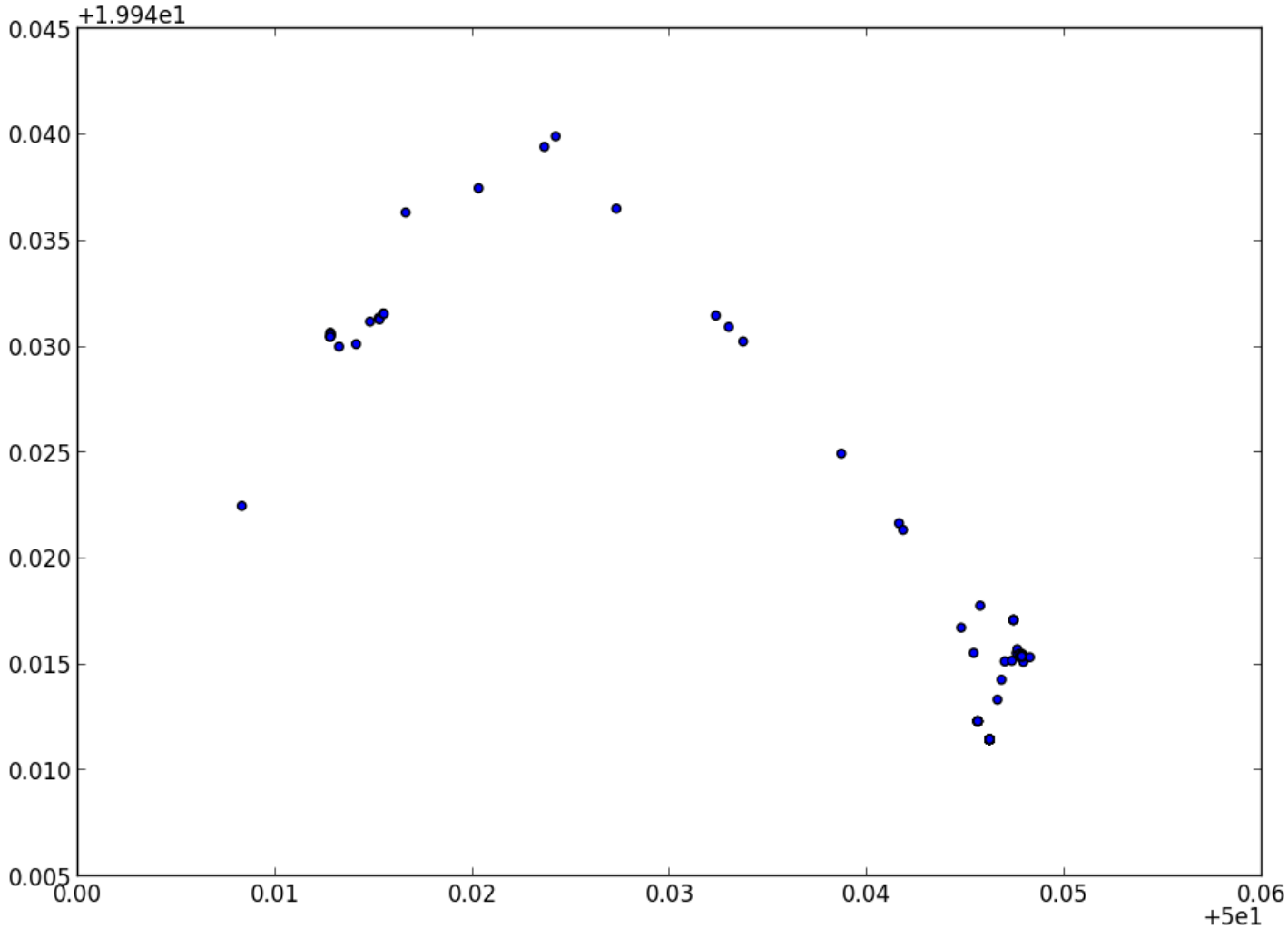


Our data clustering algorithm

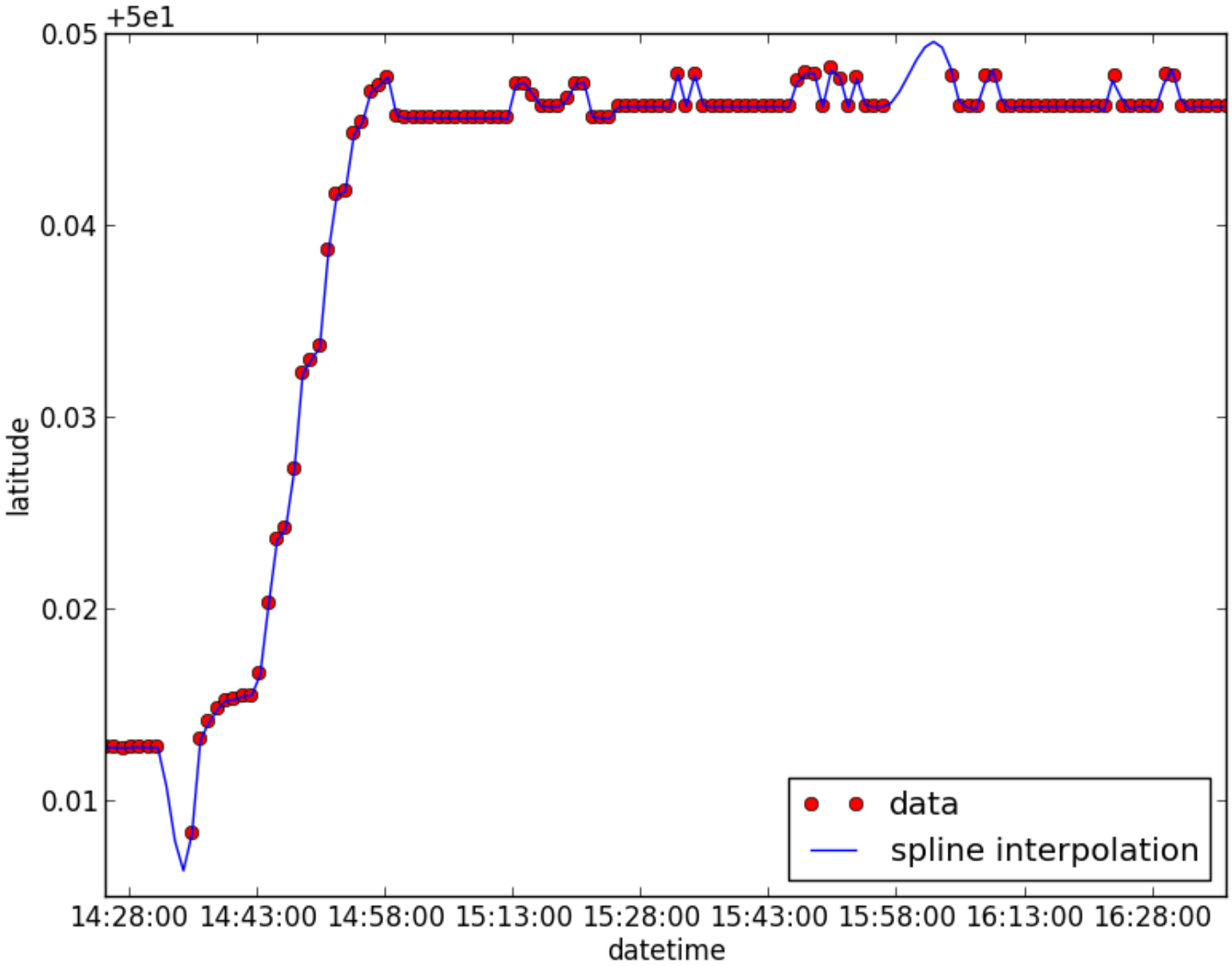
1. Spline interpolation
2. Sampling (every minute)
3. Finding the number of clusters
4. K-means

Presentation Outline

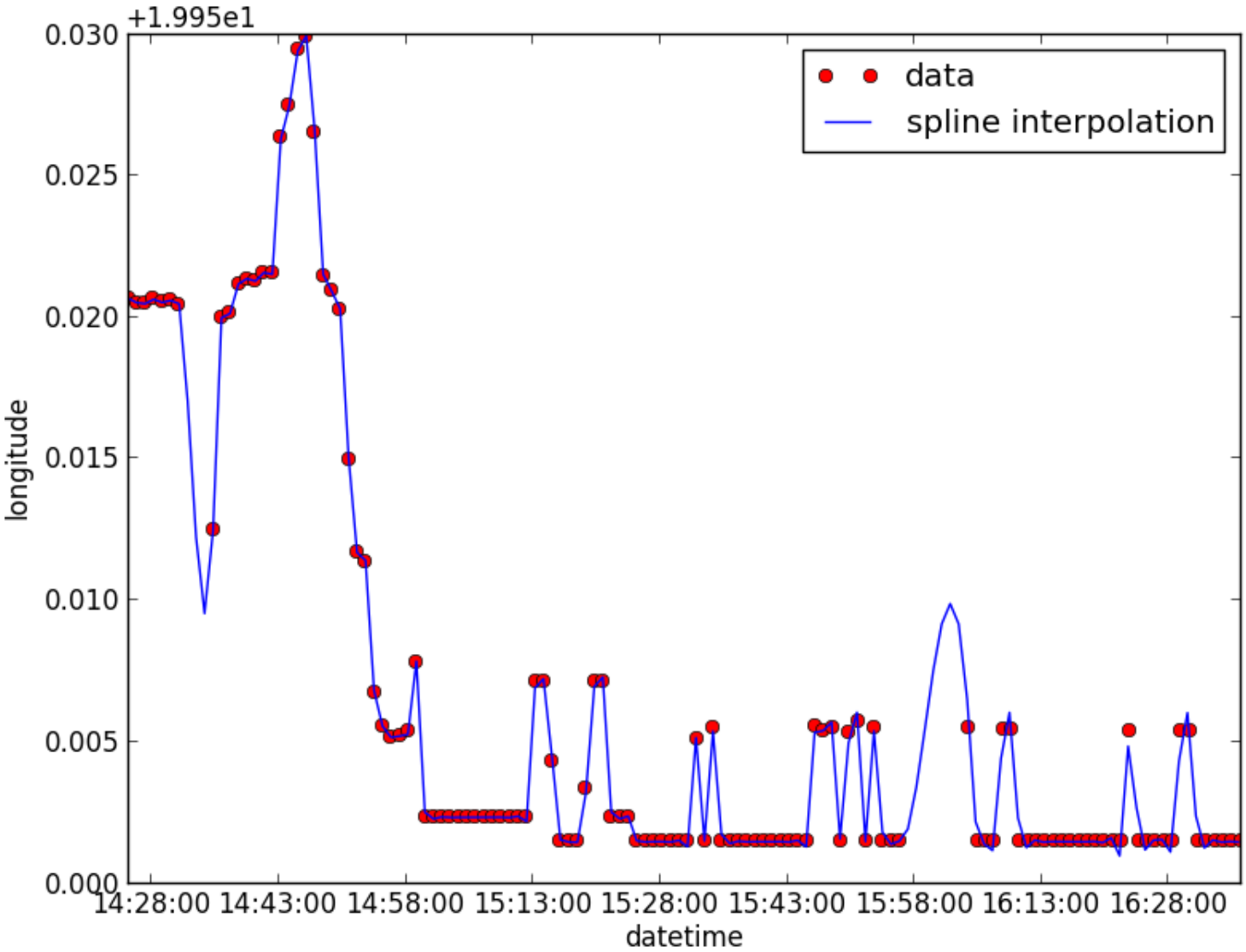
Data


+1.994e1
0.045
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.00
0.01
0.02
0.03
0.04
0.05
0.06
+5e1

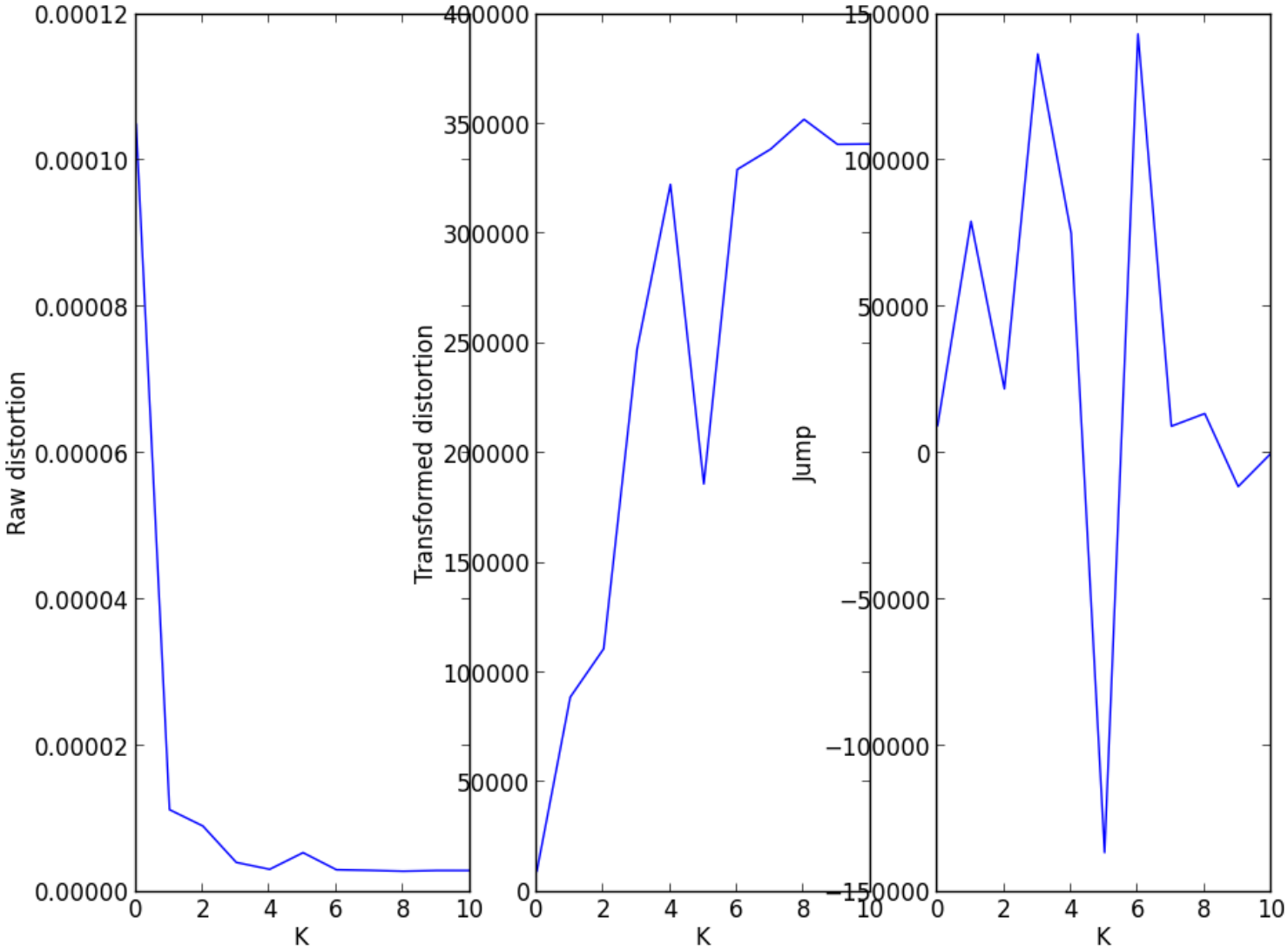
Interpolation: latitude



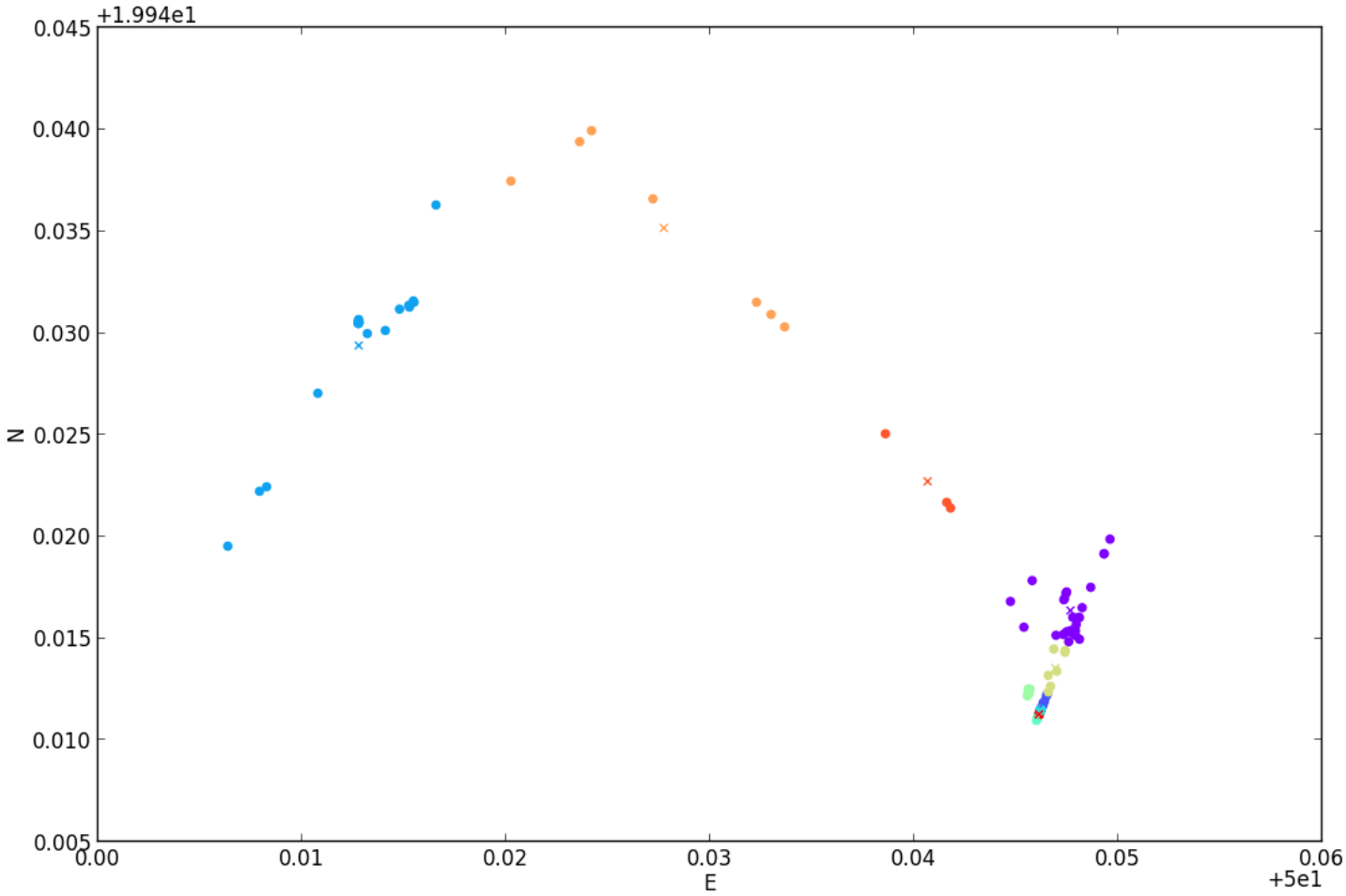
Interpolation: longitude



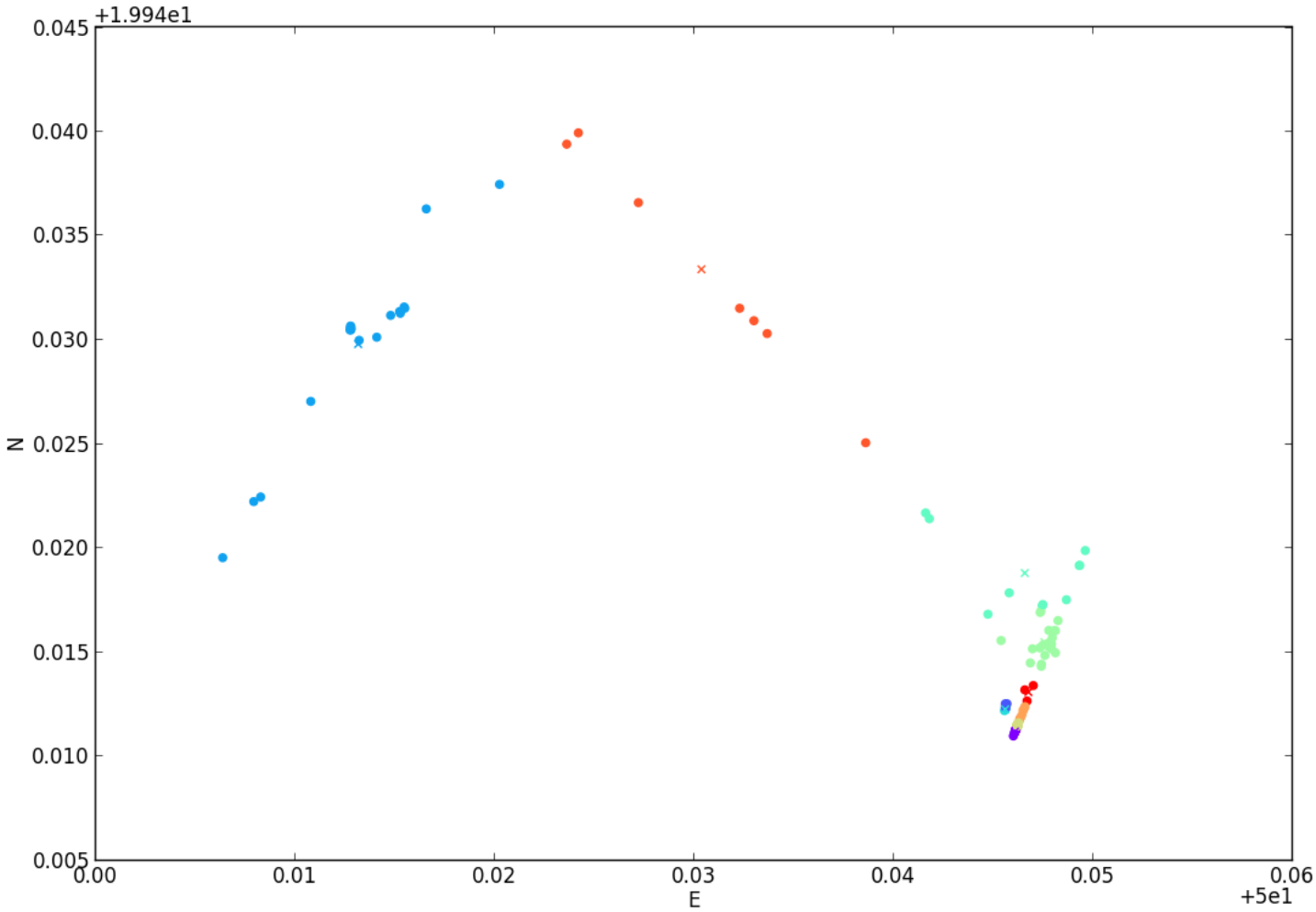
Distortion curve



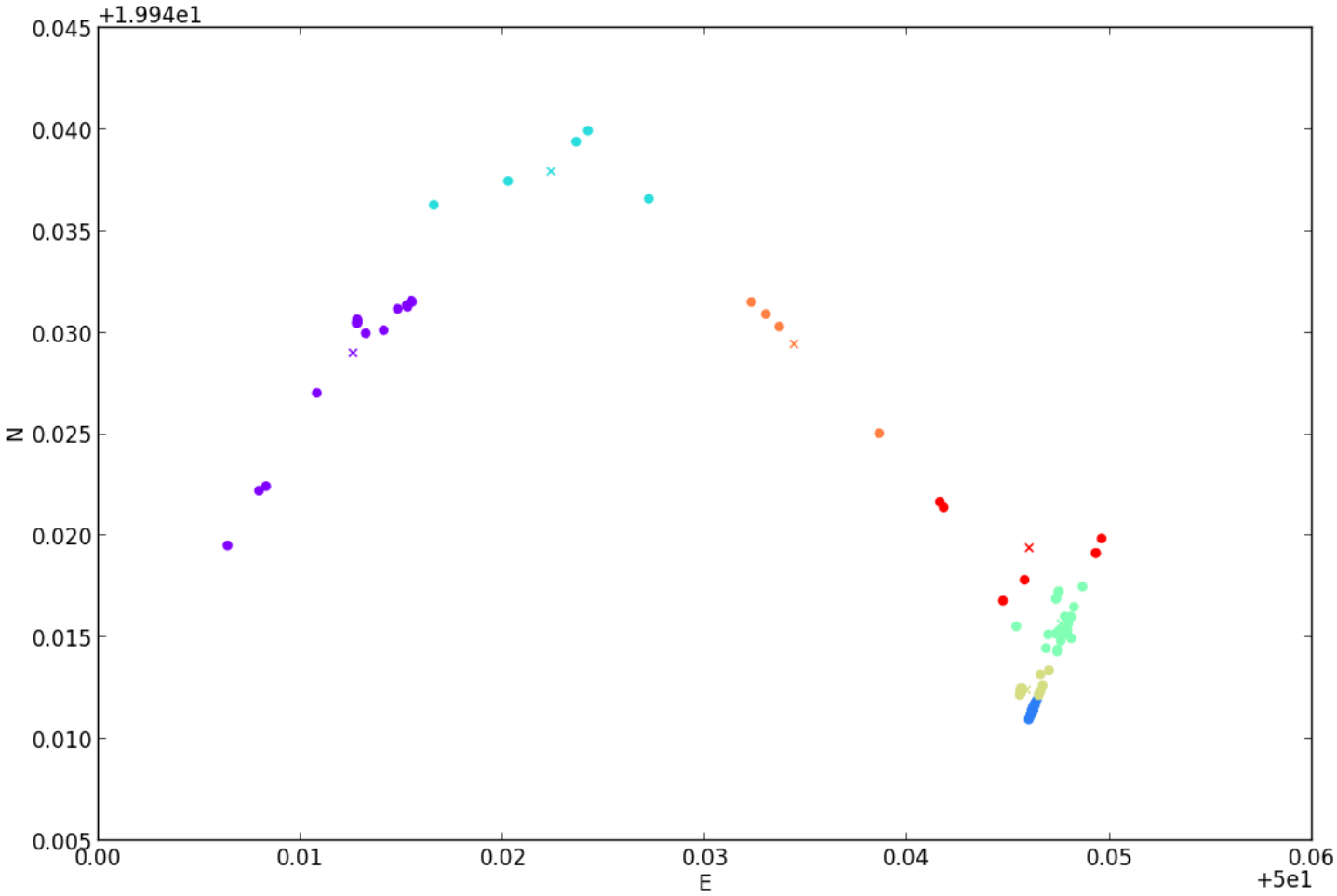
Clustered data



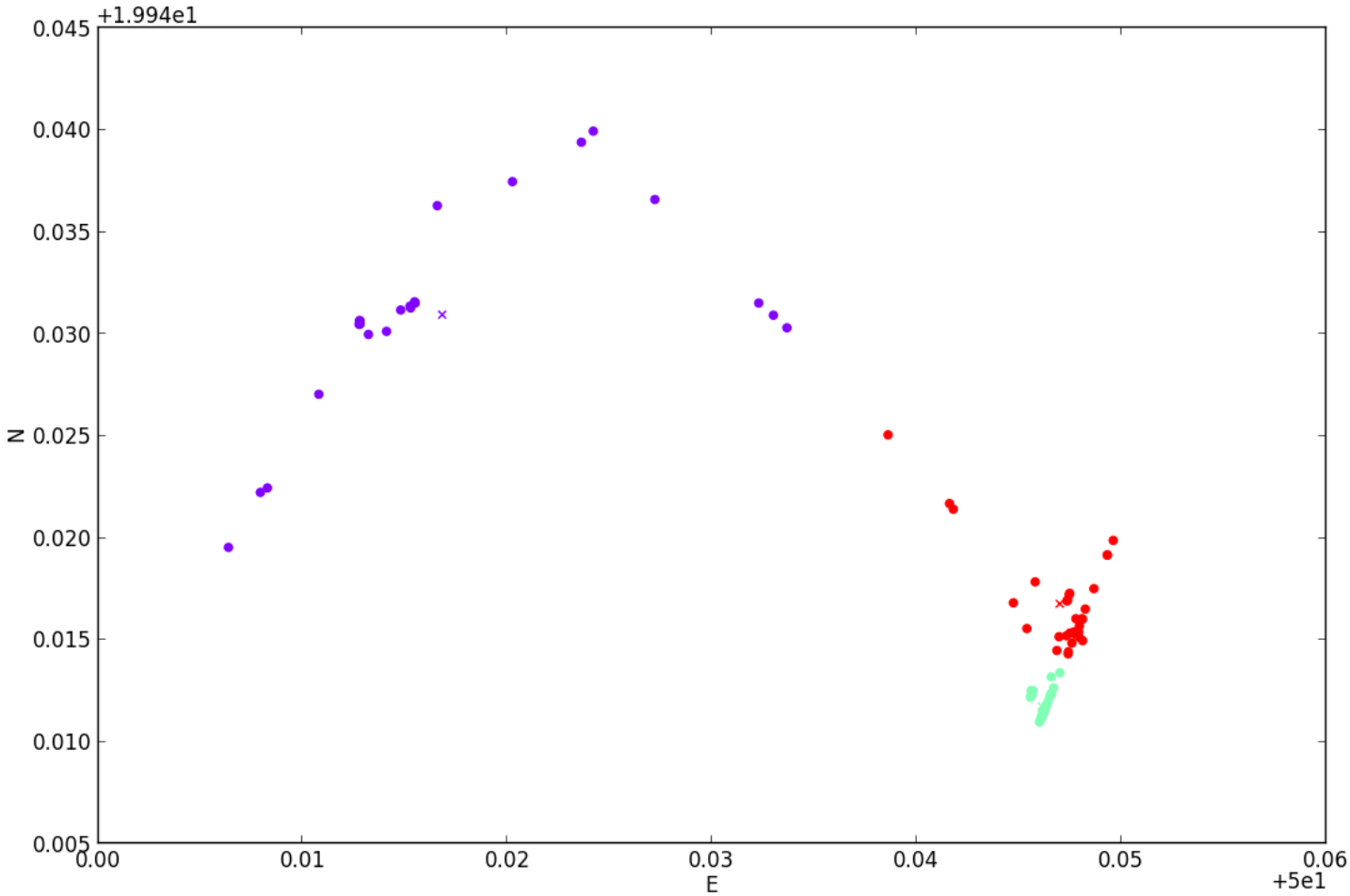
Clustered data



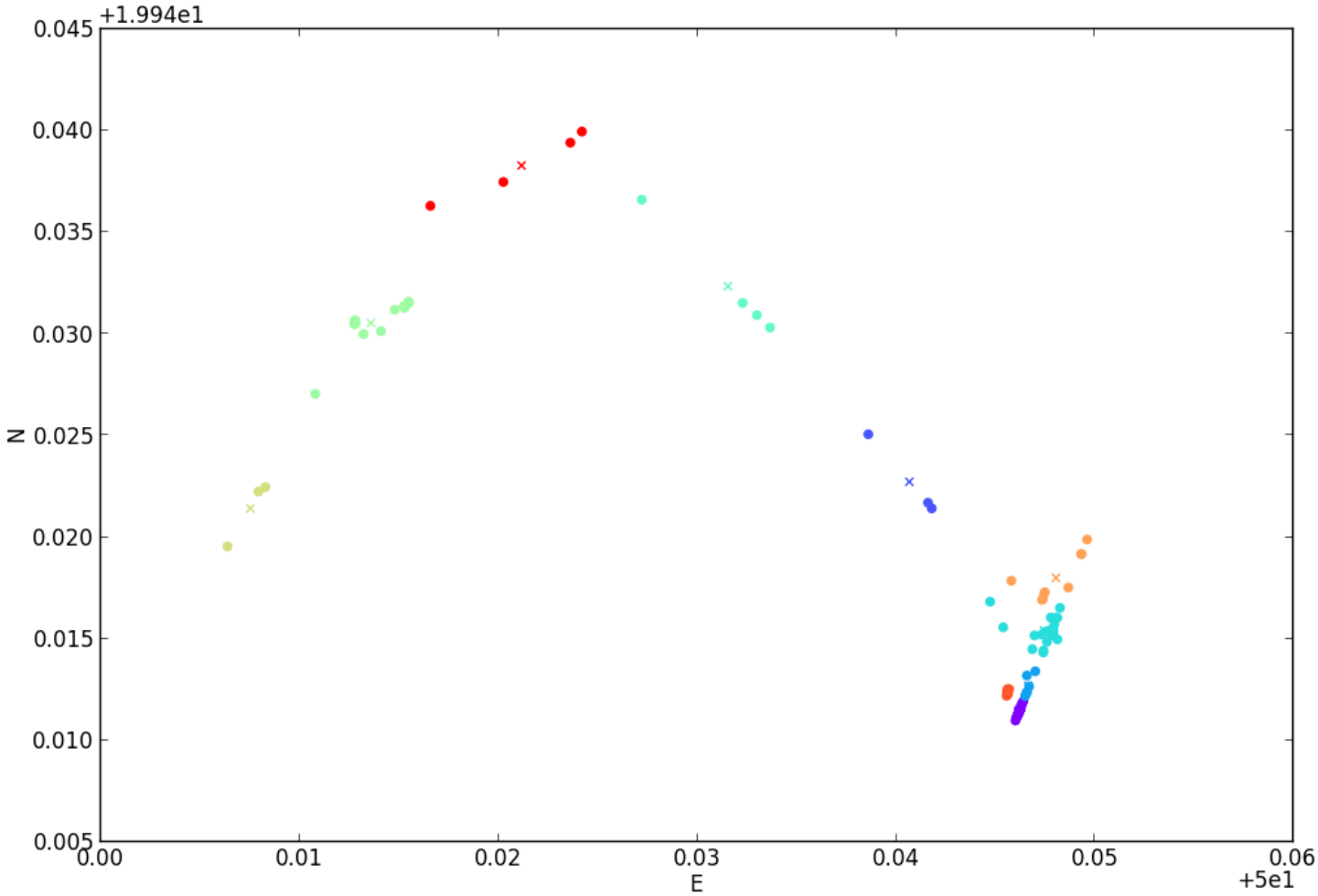
Clustered data



Clustered data



Clustered data



Presentation Outline

Android implementation

1. GPS Logger has to be rewritten from scratch, this time fully implementing Location Strategies described in Android API Guides
2. Logger could be extended with the addition of accelerometer data collection, which could prove useful in discovering of e.g. the 'Traffic' or 'Movement' domains, as well as in extending the interval between consecutive GPS data logs.
3. Data collection and processing model has to bo designed
4. A numerical library has to be picked, 2 solutions proposed:
   - Java libraries – translation into Dalvik format (whol JARs or selected classes)
   - NDK with C libraries
5. Integration with a rule framework, e.g HeaRTDroid

Algorithm improvement

Clustering while in travel
How to cluster the data collected while travelling?

- Several clusters, or
- One cluster ("the road")

References


Catherine A. Sugar and Gareth M. James
Finding the number of clusters in a data set: An information theoretic approach
Marshall School of Business, University of Southern California

Thank you for your attention!

Any questions?

http://geist.agh.edu.pl