Machine Learning User Domain Discovery

Olgierd Grodzki, Mateusz Rzeszutek

Institute of Applied Computer science
AGH University of Science and Technology

April 16, 2013

Outline

Presentation Outline

Opis zagadnienia

- problem klasyfikacji
- wykorzsytanie k-means i automatycznego doboru K

Presentation Outline

K-means

Distortion

The distortion is defined as follows:

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

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:

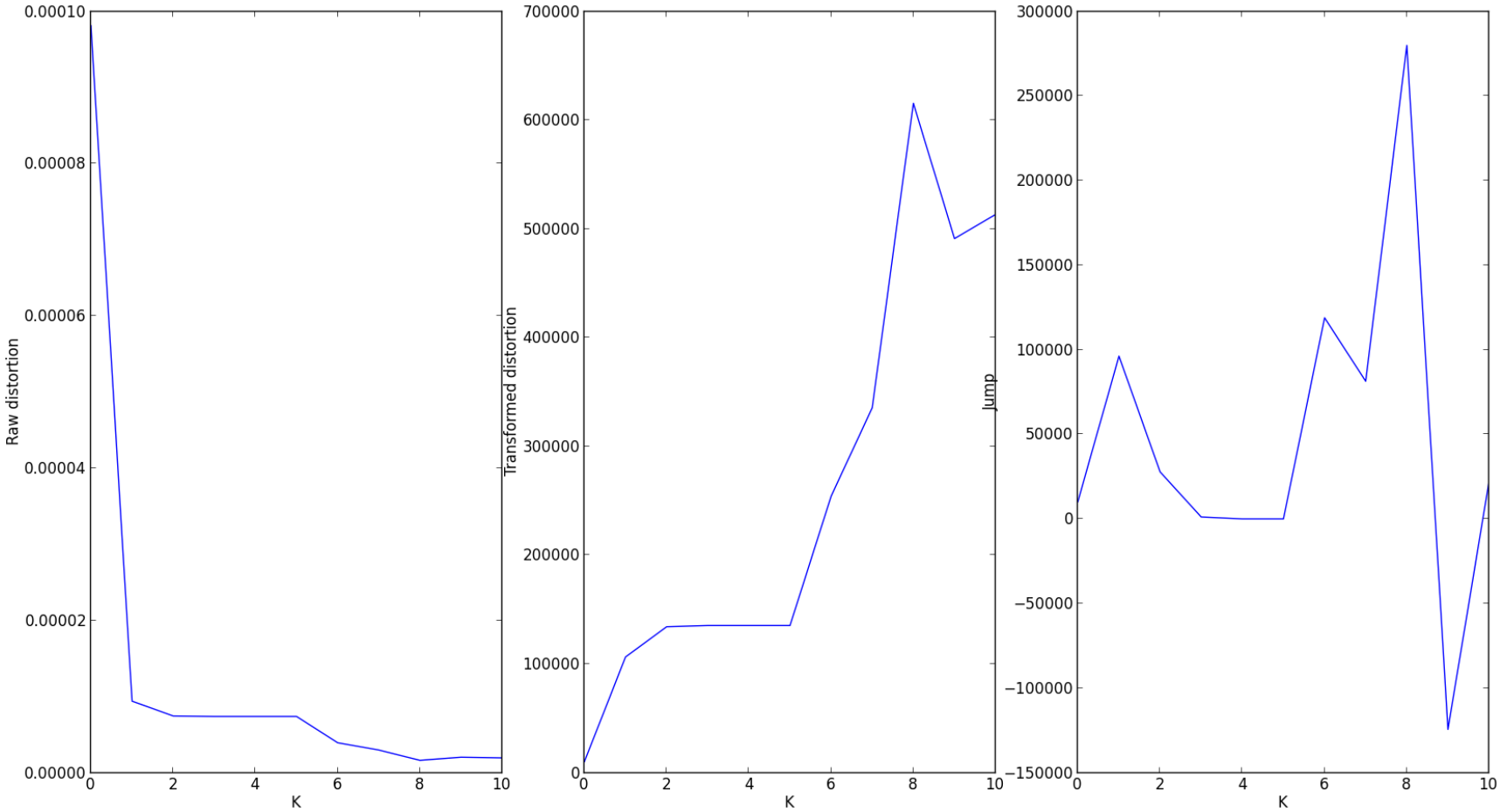
- compute distortion $d(k)$ for all $1 < K < n$, using a standard clustering algorithm (K-means)
- choose a transformation power, $Y > 0$; a typical value is $\frac{p}{2}$
- transform the distortion curve by a negative power: $D(K) = d(K)^{-Y}$
- calculate jumps: $J(K) = D(K) - D(K-1)$
- the largest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

An information theoretic approach – implementation

Out implementation:

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

def jump_method(data, n = None, max_iterations = 10):
    if n is None:
        n = int(np.sqrt(M))
    M, N = data.shape
    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

Wyykresyyy

Presentation Outline

Logger na Androida

- Akcelerometr + GPS

Clustering dla odcinków w trasie

Implementacja na Androidzie

- Jak?
- Zbieranie danych i co 24h obróbka
- Biblioteki numeryczne

Integracja z silnikami reguł

Thank you for your attention!

Any questions?

http://geist.agh.edu.pl