K-Shape Algorithm

Shape-based Time Series Clustering

Key Features

- **Cross-correlation** for similarity
- **Scale & shift invariant** distance
- Efficient **shape-based centroid**
- **Iterative refinement** like K-means
- **Faster** than DTW-based methods

✓ Best For

Normalized time series data

Method Comparison

Feature	K-Shape	DTW
Speed	Fast	Slow
Scale Invariant	✓	✗
Shift Invariant	✓	✗
Time Warp	✗	✓
Complexity	O(nm)	$O(n^2m^2)$

K-Shape Algorithm Flow

1. Initialize k cluster centroids
2. Assign series to nearest centroid (cross-correlation)
3. Update centroids (shape extraction)
4. Repeat until convergence

K-Shape Algorithm Example

Step-by-Step Calculation with 2D Vectors

📌 Input Data: 4 Time Series (k=2 clusters)

$x_1 = [1, 2]$ $x_2 = [2, 3]$ $x_3 = [8, 9]$ $x_4 = [9, 10]$

🎯 Step 1: Normalization (z-score)

$$z = (x - \mu) / \sigma$$

$z_1 = [-1.0, 1.0]$ $z_2 = [-1.0, 1.0]$ $z_3 = [-1.0, 1.0]$ $z_4 = [-1.0, 1.0]$

💡 Example: Normalize $x_1 = [1, 2]$

$\mu = (1 + 2) / 2 = 1.5$

$\sigma = \sqrt{[((1-1.5)^2 + (2-1.5)^2) / 2]} = 0.5$

$z_1[0] = (1 - 1.5) / 0.5 = -1.0$

$z_1[1] = (2 - 1.5) / 0.5 = 1.0$

Result

$z_1 = [-1.0, 1.0]$

📝 Key Insight

All normalized vectors have the same shape pattern [-1.0, 1.0], showing they follow the same increasing trend!

This normalization makes K-Shape **scale-invariant** - it focuses on shape, not magnitude.

K-Shape Algorithm Example

Iteration 0: Initial Cluster Assignment

Iteration 0

Step 2: Initialize Centroids (Random)

$\mu_1 = [-1.0, 1.0]$ $\mu_2 = [-1.0, 1.0]$

Cross-Correlation Distance (SBD)

```
SBD(x, y) = 1 - max(CC(x, y)) / (||x|| × ||y||)
CC(x, y) = Σ x[i] × y[i] (Cross-Correlation)
```

Calculate distance for z_1 to μ_1:

$CC(z_1, \mu_1) = (-1.0)\times(-1.0) + (1.0)\times(1.0) = 2.0$

$||z_1|| = \sqrt{(1.0^2 + 1.0^2)} = 1.414$

$||\mu_1|| = 1.414$

$SBD(z_1, \mu_1) = 1 - 2.0/(1.414 \times 1.414) = 0.0$

Distance Result

$z_1 \rightarrow$ Cluster 1 (distance = 0.0)

Initial Cluster Assignment

Cluster 1

$z_1 = [-1.0, 1.0]$

Cluster 2

$z_3 = [-1.0, 1.0]$

z_2 = [-1.0, 1.0]

z_4 = [-1.0, 1.0]

K-Shape Algorithm Example

Iteration 1: Update Centroids

Iteration 1

🔄 Step 3: Update Cluster 1 Centroid

```
μ = argmax Σ CC(x_i, y) / ||y||²
Simplified: Average of cluster members
```

Members: z_1 , z_2 = [-1.0, 1.0]

μ_1 [0] = (-1.0 + -1.0) / 2 = -1.0

μ_1 [1] = (1.0 + 1.0) / 2 = 1.0

Updated Centroid

μ_1 = [-1.0, 1.0]

🔄 Step 3: Update Cluster 2 Centroid

Members: z_3 , z_4 = [-1.0, 1.0]

μ_2 [0] = (-1.0 + -1.0) / 2 = -1.0

μ_2 [1] = (1.0 + 1.0) / 2 = 1.0

Updated Centroid

μ_2 = [-1.0, 1.0]

⚠️ Convergence Check

Centroids unchanged!

$\mu_1 = \mu_2 = [-1.0, 1.0]$

✓ Algorithm converged!

💡 Why did it converge so quickly?

All normalized vectors have identical shape [-1.0, 1.0], so they all have perfect cross-correlation with any centroid of the same shape. The algorithm converged in just one iteration!

K-Shape Algorithm Example

Final Results & Analysis

Final Cluster Assignment

Cluster 1

$x_1 = [1, 2]$

$x_2 = [2, 3]$

Centroid: [-1.0, 1.0]

Cluster 2

$x_3 = [8, 9]$

$x_4 = [9, 10]$

Centroid: [-1.0, 1.0]

Key Observations

1. Shape-based clustering:

Groups series by pattern, not absolute values

2. Same normalized shape:

All series show increasing trend [-1, 1]

3. Scale invariance:

[1,2] and [8,9] treated similarly

Algorithm Insights

✓ **Fast convergence:**

Converged in 1 iteration

✓ **Cross-correlation:**

Measures shape similarity efficiently

✓ **Normalization effect:**

All vectors → same pattern

Complexity Analysis

For n=4 series, m=2 dimensions, k=2 clusters:

K-Shape	DTW
$O(4 \times 2) = O(8)$	$O(16 \times 4) = O(64)$

K-Shape is 8× faster! ⚡