Weighted Clusterwise Linear Regression using Adaptive Quadratic Distance

OTML Project

Dinesh Kumaran K - CS23B2057
Jovan Moris D - CS23B2058

December 13, 2024

Contents

1 Introduction

Weighted Clusterwise Linear Regression using Adaptive Quadratic Distance is a powerful approach for analyzing complex datasets. It combines clustering and regression by introducing adaptive weight matrices and quadratic distance measures to enhance accuracy and interpretability. The algorithm effectively partitions data into meaningful clusters and assigns specific regression models to each cluster.

Applications:

- Housing price prediction
- Customer segmentation
- Anomaly detection
- Financial forecasting

2 Dataset Description

The dataset used in this analysis consists of the following attributes:

- **CRIM:** Per capita crime rate by town.
- **ZN:** Proportion of residential land zoned for lots over 25,000 sq.ft.
- **INDUS:** Proportion of non-retail business acres per town.
- **CHAS:** Charles River dummy variable (1 if tract bounds river; 0 otherwise).
- **NOX:** Nitric oxides concentration (parts per 10 million).
- **RM:** Average number of rooms per dwelling.
- **AGE:** Proportion of owner-occupied units built prior to 1940.
- **DIS:** Weighted distances to five Boston employment centres.
- **RAD:** Index of accessibility to radial highways.
- **TAX:** Full-value property-tax rate per $10,000.
- **PTRATIO:** Pupil-teacher ratio by town.
- **B:** $1000(Bk - 0.63)^2$ where Bk is the proportion of Black residents by town.
- **LSTAT:** % lower status of the population.
- **MEDV:** Median value of owner-occupied homes in $1000s.

3 Algorithm

The algorithm follows these steps:

1. **Initialization:** Assign data points randomly to clusters and initialize weight matrices.
2. **Representation:** Update cluster centroids using mean of assigned points.
3. **Weighting:** Compute adaptive weight matrices for each cluster.
4. **Modeling:** Train a linear regression model for each cluster.
5. **Assignment:** Reassign data points to clusters based on weighted distances and regression errors.
6. **Stopping Criteria:** Repeat until cluster assignments stabilize.

3.1 Implementation Details

The following sections describe the implementation steps in detail, including code snippets.

3.1.1 Initialization

In this step, data points are assigned to initial clusters, and weight matrices are initialized.

```
from sklearn.cluster import KMeans

# Random initialization using KMeans
kmeans = KMeans(n_clusters=K, random_state=42)
clusters = kmeans.fit_predict(X)

# Initialize weight matrices (identity matrix for each
   cluster)
weights = [np.eye(X.shape[1]) for _ in range(K)]

def initialize_clusters(X, K):
    return kmeans.fit_predict(X), weights
```

3.1.2 Representation

Cluster centroids are updated as the mean of the data points assigned to each cluster.

```
def update_centroids(X, clusters, K):
    centroids = []
    for k in range(K):
        cluster_points = X[clusters == k]
        centroids.append(cluster_points.mean(axis=0))
    return np.array(centroids)
```

3.1.3 Weighting

Adaptive weight matrices are calculated using the inverse covariance of cluster points.

```
def compute_weights(X, clusters, K, regularization=1e-6):
    weights = []
    for k in range(K):
        cluster_points = X[clusters == k]
        if len(cluster_points) > 1:
            cov_matrix = np.cov(cluster_points, rowvar=False)
                + regularization * np.eye(X.shape[1])
            weights.append(np.linalg.inv(cov_matrix))
        else:
            weights.append(np.eye(X.shape[1]))
    return weights
```

3.1.4 Modeling

Linear regression models are trained for each cluster.

```
from sklearn.linear_model import LinearRegression

models = [LinearRegression() for _ in range(K)]
for k in range(K):
    cluster_points = X[clusters == k]
    cluster_targets = y[clusters == k]
    if len(cluster_points) > 0:
        models[k].fit(cluster_points, cluster_targets)
```

3.1.5 Assignment

Reassign each data point to the cluster that minimizes the quadratic distance plus regression error.

```
def assign_clusters(X, models, weights, K):
    new_clusters = np.zeros(X.shape[0], dtype=int)
    for i, x in enumerate(X):
        distances = [weighted_distance(x, models[k].predict(x
            .reshape(1, -1)), weights[k]) for k in range(K)]
        new_clusters[i] = np.argmin(distances)
    return new_clusters
```

3.1.6 Stopping Criteria

The algorithm stops when cluster assignments no longer change or after a maximum number of iterations.

```
for iteration in range(max_iter):
    new_clusters = assign_clusters(X, models, weights, K)
    if np.array_equal(clusters, new_clusters):
        print("Convergence achieved.")
        break
    clusters = new_clusters
    weights = compute_weights(X, clusters, K)
```

4 Visualization

The following figures illustrate the clustering results and the fitted regression models.

4.1 Cluster Visualization

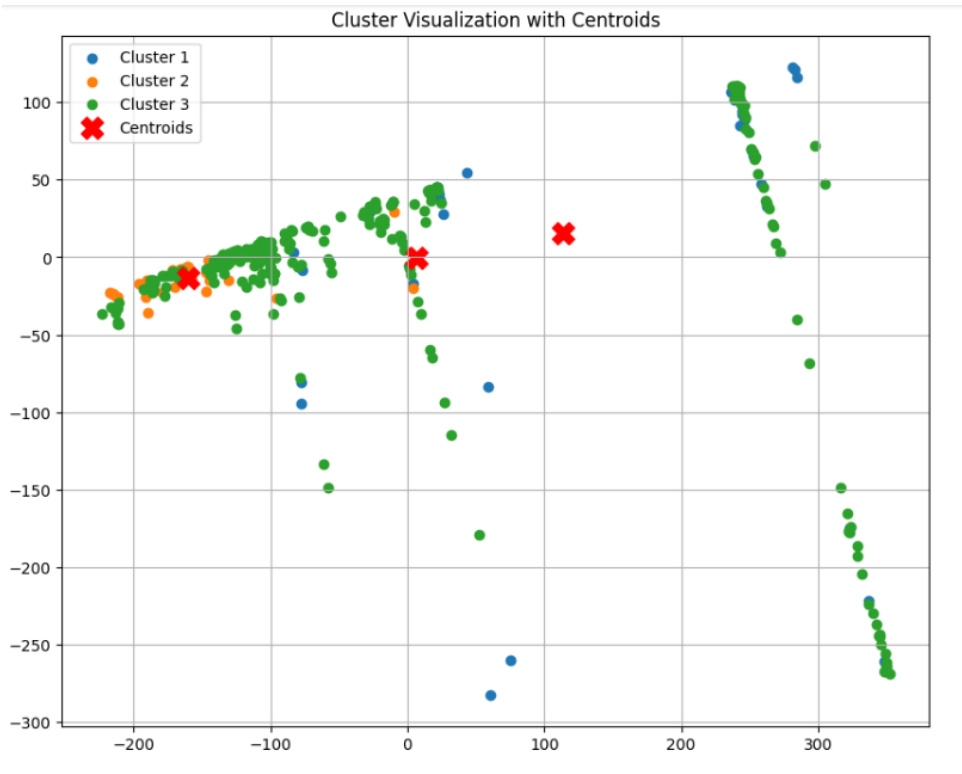

Figure 1: Cluster Visualization after Weighted CLR

4.2 Regression Analysis

```
Cluster 1:
  Number of points: 25
  Coefficients:
    Feature 1: 6.198616348097052
    Feature 2: 0.00933543741120401
    Feature 3: 0.09497076592139249
    Feature 4: 0.1833440311558824
    Feature 5: -28.109505103541665
    Feature 6: 9.450066567631113
    Feature 7: -0.05483854281183476
    Feature 8: -1.1361648885313762
    Feature 9: 0.042058683111608465
    Feature 10: -0.03758885036459734
    Feature 11: -1.0910028017893962
    Feature 12: 0.005512441403385852
    Feature 13: -0.37720410913567504
  Intercept: 14.27

Cluster 2:
  Number of points: 318
  Coefficients:
    Feature 1: -0.1152760240638963
    Feature 2: 0.06056909582035256
    Feature 3: 0.06765932764489307
    Feature 4: 6.591099883606772
    Feature 5: -18.683849124562222
    Feature 6: -0.40521324514915635
    Feature 7: -0.015717982542628663
    Feature 8: -1.248619038444088
    Feature 9: 0.23197270070904827
    Feature 10: -0.009608425851890354
    Feature 11: -0.5064617521109985
    Feature 12: 0.005712751336150679
    Feature 13: -0.5144873863841111
  Intercept: 53.93

Cluster 3:
  Number of points: 51
  Coefficients:
    Feature 1: 0.02236221145047671
    Feature 2: 0.053885590932255256
    Feature 3: -0.33982965988579705
    Feature 4: 1.8481463730860352
    Feature 5: -1.6999291420670481
    Feature 6: 5.889036913509647
    Feature 7: -0.034150639416345475
    Feature 8: -1.63762200287785
    Feature 9: -0.5864052737411888
```

Figure 2: Regression Models for Clusters

5 Results

Predicted Value for New Point: `$456,789.00`
Assigned Cluster: 2

6 Conclusion

The Weighted Clusterwise Linear Regression algorithm using Adaptive Quadratic Distance effectively partitions data into meaningful clusters and models them individually using regression. Its adaptive weighting and robust distance metrics significantly improve clustering and regression accuracy. Future enhancements include applying this algorithm to large-scale datasets and experimenting with alternative weighting schemes.