K-Means Clustering Algorithm

CSEC620 - Project 2 - Mehul Sen

Overview

This report summarizes the implementation and evaluating of a K-Means clustering algorithm on the Iris and Wine datasets. A standard Euclidean distance metric and a weighted Mahalanobis distance metric are tested. Feature weights are tuned to optimize the Mahalanobis distance metric. The Mahalanobis metric with tuned weights improves clustering performance compared to the standard K-Means algorithm.

Quick Start

This section will outline the necessary steps to use both the K-Means and the K-Means Mahalanobis algorithms. This project was completed using Visual Studio Code with Jupyter Notebook. If you require assistance, we recommend referring to the following resource for setting up the notebook: Running Jupyter Notebook on Visual Studio Code | Medium

1. *Setup Dataset Folder*: To set up the dataset folder, start by unzipping and extracting the Iris and wine .zip files and placing them in a Dataset folder in the root directory. The dataset should be correctly located using the provided zip file, `project_2_sen.zip`. The desired folder structure should be as follows with the required files:

```
- requirements.txt
- project_2_kmean_clustering_algoritm.ipynb
- Dataset/
    - Iris/
        - iris.data
        - iris.name
    - wine/
        - wine.data
        - wine.name
```

2. *Import Libraries*: Next, import the necessary libraries from requirements.txt using the following command: `pip install -r requirements.txt`. (Note: The only external library used for this project is numpy for data handling, the implementation of K-Means is done manually.)
3. *Run Notebook*: Finally, run the Jupyter Notebook (`project_2_kmean_clustering_algorithm.ipynb`) and execute each cell block sequence from top to bottom.

Dataset

There are two datasets used in this project. Namely Iris and Wine.

The Iris dataset, which contains measurements of Iris flowers belonging to 3 species, has three categories: Iris Setosa, Iris Versicolour, and Iris Virginica.

This dataset has the following four features:

- sepal length in cm

- sepal width in cm
- petal length in cm
- petal width in cm

The Wine dataset contains chemical properties of wines from 3 cultivars: 1, 2, and 3.

This dataset has the following thirteen features:

- Alcohol
- Malic acid
- Ash
- Alkalinity of ash
- Magnesium
- Total phenols
- Flavanoids
- Nonflavanoid phenols
- Proanthocyanins
- Color intensity
- Hue
- OD280/OD315 of diluted wines
- Proline

Additional information on each dataset can be found in their respective .name files.

K-Means Algorithm

The K-Means algorithm partitions data points into a specified number of clusters by iteratively:

1. Assigning points to their closest cluster centroid
2. Updating the centroids based on the mean of assigned points

This repeats until the assignments converge. The key steps are:

- Initialize K random centroids
- Calculate the distance of each point to each centroid
- Assign points to the nearest centroid
- Recompute centroids as cluster means
- Repeat steps 2-4 until convergence

The KMeans class implements this in Python, supporting the customization of distance metrics and maximum iterations.

Evaluation

The algorithm is run 1000 times on the Iris and Wine datasets to evaluate performance. The average accuracy, precision, recall, and F1 score are calculated.

Iris Dataset

Using Euclidean distance, K-Means achieves:

- **Confusion Matrix**:

16.485	**17.490**	**16.025**
16.936	**16.640**	**16.424**
15.996	**17.017**	**16.987**

- **Accuracy**: 0.334
- **Precision**: 0.364
- **Recall**: 0.339
- **F1 Score**: 0.317

Wine Dataset

With Euclidean distance, K-Means has:

- **Confusion Matrix**:

19.907	**19.870**	**19.223**
24.214	**22.003**	**24.783**
16.689	**14.940**	**16.371**

- **Accuracy**: 0.327
- **Precision**: 0.236
- **Recall**: 0.341
- **F1 Score**: 0.261

Mahalanobis K-Means

The Mahalanobis distance uses a weighted covariance matrix C when calculating distances:

```
d(x, y) = (x - y)^T * C * (x - y)
```

This accounts for the correlation between features.
The performance is first evaluated using default weights. Then, weights are tuned for each feature to maximize accuracy.

Evaluation

The algorithm is run 1000 times on the Iris and Wine datasets to evaluate performance. The average accuracy, precision, recall, and F1 score are calculated.
To finetune the C value for the dataset, I set the weight for each feature as '10', calculated the accuracy, and then individually changed the values for each feature, ranging from 1, 10, 100, and 1000. I used accuracy as the measurement metric. After collecting these values, the values that significantly differed in accuracy were used.

Iris Dataset

Using tuned Mahalanobis distances improves performance:

Default:

- **Default Accuracy**: 0.320

Feature	1	100	1000	Chosen Weights
1	0.336	0.335	0.336	1
2	0.316	0.325	0.336	1
3	0.330	0.342	0.332	100
4	0.330	0.339	0.323	100

Finetuned:

- **Confusion Matrix**:

16.824	**16.893**	**16.283**
15.868	**17.134**	**16.998**
15.568	**16.928**	**17.504**

- **Accuracy**: 0.343
- **Precision**: 0.345
- **Recall**: 0.350
- **F1 Score**: 0.321

This suggests that weighting helps separate the Iris species clusters.

Wine Dataset

The Mahalanobis metric provides smaller gains on the Wine data:

Default:

- **Default Accuracy**: 0.326

Feature	1	100	1000	Chosen Weights
1	0.327	0.332	0.331	100
2	0.335	0.323	0.327	1
3	0.342	0.334	0.337	1
4	0.337	0.330	0.336	1
5	0.332	0.342	0.337	100

Feature	1	100	1000	Chosen Weights
6	0.332	0.324	0.327	1
7	0.327	0.328	0.341	1000
8	0.335	0.342	0.333	100
9	0.337	0.333	0.331	1
10	0.332	0.333	0.339	1000
11	0.323	0.318	0.326	10
12	0.333	0.325	0.338	1000
13	0.332	0.341	0.336	100

Finetuned:

- **Confusion Matrix**:

20.698	19.826	18.476
22.297	24.627	24.076
15.419	16.916	15.665

- **Accuracy**: 0.342
- **Precision**: 0.228
- **Recall**: 0.326
- **F1 Score**: 0.251

But all metrics still increase from default K-Means.

Results

The default k-means clustering algorithm has performed poorly on the Iris and Wine datasets, with low accuracy, precision, recall, and F1 scores.
The Mahalanobis k-means algorithm with finetuned feature weights improved the clustering performance, especially on the Iris dataset. The finetuned Mahalanobis approach increased the accuracy from 0.334 to 0.343, precision from 0.364 to 0.345, and recall from 0.339 to 0.350 compared to the default k-means. This indicates that weighting the features helped the algorithm better separate the Iris data points into distinct clusters.
The improvements from finetuning the Mahalanobis weights on the Wine dataset were minor, with accuracy going from 0.327 to 0.342. However, all the evaluation metrics still increased, suggesting the weighting helped improve clustering performance on this dataset.
Overall, the Mahalanobis distance metric allowed the k-means algorithm to account for differences in scale and variance between features. Finetuning the feature weights further improved performance by letting the algorithm focus more on the most discriminative features for clustering the data points correctly. This enabled better cluster separation and assignment compared to the default algorithm.

Conclusion

The Mahalanobis distance metric improves K-Means clustering by accounting for feature differences. Tuning the feature weights enhances this by focusing on the most discriminative attributes. This enables better cluster separation and assignment. The improvements are most pronounced on the Iris dataset but also benefit the Wine data.