M11-HW1

November 26, 2023

1 Problem 1

1.1 Problem Description

In this problem you will use DBSCAN to cluster two melt pool images from a powder bed fusion metal 3D printer. Often times during printing there can be spatter around the main melt pool, which is undesirable. If we can successfully train a model to identify images with large amounts of spatter, we can automatically monitor the printing process.

Fill out the notebook as instructed, making the requested plots and printing necessary values.

You are welcome to use any of the code provided in the lecture activities.

Summary of deliverables:

- Bitmap visualization of melt pool images 1 and 2
- Visualization of final DBSCAN clustering result for both melt pool images
- Discussion of tuning, final number of clusters, and the sensitivity of the model parameters for the two images.

Imports and Utility Functions:

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn.cluster import DBSCAN

def points_to_bitmap(x):
    bitmap = np.zeros((64, 64), dtype=int)
    cols, rows = x[:, 0], x[:, 1]
    bitmap[rows, cols] = 1
    return bitmap

def plot_bitmap(bitmap):
    _, ax = plt.subplots(figsize=(3,3), dpi = 200)
    colors = ListedColormap(['black', 'white'])
    ax.imshow(bitmap, cmap = colors, origin = 'lower')
    ax.axis('off')

def plot_points(x, labels):
```

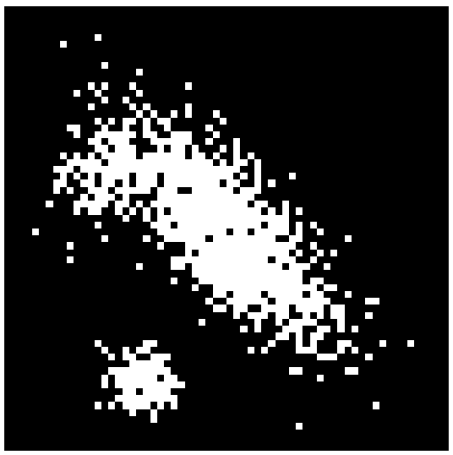
```
    fig = plt.figure(figsize = (5,4), dpi = 150)
    for i in range(min(labels),max(labels)+1):
        plt.scatter(x[labels == i, 0], x[labels == i, 1], alpha = 0.5, marker =␣
↪'s')
    plt.gca().set_aspect('equal')
    plt.tight_layout()
    plt.show()
```

1.2 Melt Pool Image #1

Load the first meltpool scan from the `m11-hw1-data1.txt` file using `np.loadtxt()`, and pass the `dtype = int` argument to ensure all values are loaded with their integer coordinates. You can convert these points to a binary bitmap using the provided `points_to_bitmap()` function, and then visualize the image using the provided `plot_bitmap()` function.

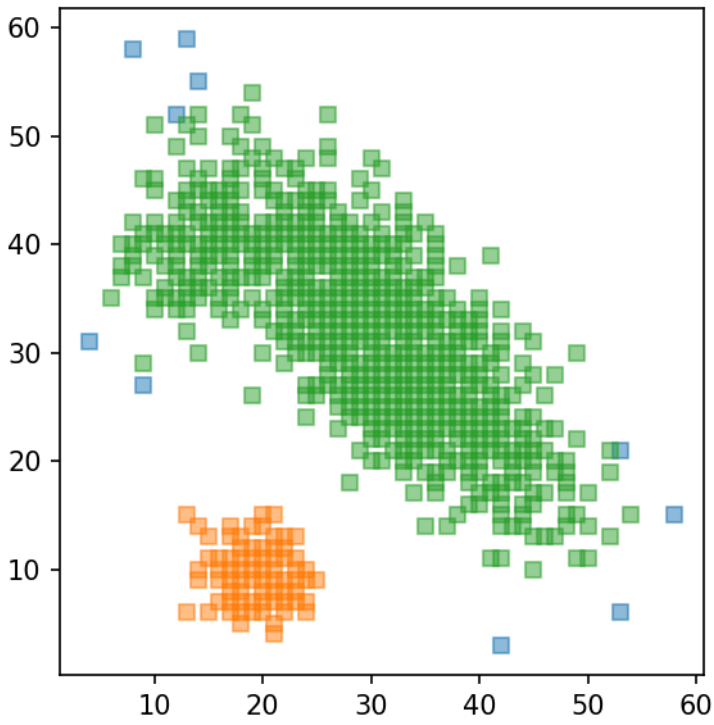
Note: you will use the integer coordinates for clustering, the bitmap is just for visualizing the data.

[ ]:
```
x1 = np.loadtxt("data/m11-hw1-data1.txt", dtype=int)
bitmap1 = points_to_bitmap(x1)
plot_bitmap(bitmap1)
```



Using the `sklearn.cluster.DBSCAN()` function, cluster the melt pool until you get well defined clusters. You will have to modify the `eps` and `min_samples` parameters to get satisfactory results. You can visualize the clustering with the provided `plot_points(x, labels)` function, where `x` is the integer coordinates of all the points, and labels are the labels assigned by `DBSCAN`. Plot the results of your final clustering using the `plot_points()` function

[ ]:
```
model1 = DBSCAN(eps=5, min_samples=20).fit(x1)
plot_points(x1, model1.labels_)
```



1.3 Melt Pool Image #2

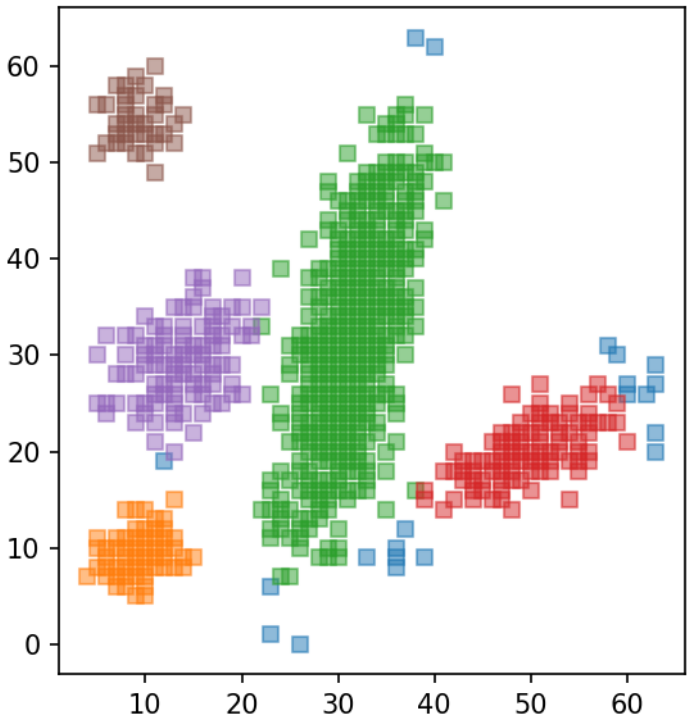
Now load the second melt pool scan from the `m11-hw1-data2.txt` file using `np.loadtxt()`, and the `dtype = int` argument to ensure all values are loaded with their integer coordinates. Again, convert the points to a binary bitmap, and visualize the bitmap using the provided functions.

[ ]:
```
x2 = np.loadtxt("data/m11-hw1-data2.txt", dtype=int)
bitmap2 = points_to_bitmap(x2)
plot_bitmap(bitmap2)
```

Using the `sklearn.cluster.DBSCAN()` function, cluster the meltpool until you get well defined clusters. You will have to modify the `eps` and `min_samples` parameters to get satisfactory results. You can visualize the clustering with the provided `plot_points(x, labels)` function, where `x` is the integer coordinates of all the points, and labels are the labels assigned by `DBSCAN`. Plot the results of your final clustering using the `plot_points()` function.

Note: this melt pool is significantly noisier than the last and therefore requires more sensitive tuning of the two DBSCAN parameters.

[ ]:
```
model2 = DBSCAN(eps=4, min_samples=15).fit(x2)
plot_points(x2, model2.labels_)
```



1.4 Discussion

Discuss how you tuned the `eps` and `min_samples` parameters for the two models. How many clusters did you end up finding in each image? Why does a wider range of `eps` and `min_samples` values successfully cluster melt pool image #1 compared to melt pool image #2?

The eps parameter was tuned according to the minimum distance within visible clusters. For example, on the first dataset, the smallest cluster had a width of about 10-20 with a large density of points so the eps was made to be about half the width and then adjusted from there. The min_samples was tuned by visually inspecting the number of samples that appeared to be in each

distinct cluster and then lowballing this estimate for the final min_samples value. For the first image this ended up finding 2 clusters and for the second image this ended up finding 5 clusters. The wider parameters for image 1 work better because there are less clusters and they are more spread out than in the second image.