Prof. M. Pollefeyes

Filippo Ficarra: Lab 5 - Image Segmentation

fficarra@student.ethz.ch, 22–938–062.

22/11/2023 - 10:43h

1 Mean-Shift Algorithm

The Mean-Shift Algorithm aims to find modes in a given dataset. It is used to perform unsupervised clustering. The principal benefit of this algorithm is that we don't have to impose a fixed number of cluster a priori but it will just depend on a variable called "bandwidth".

1.1 Distance function

The first function completed is the distance function. This function it will further used in the gaussian kernel to compute the weight of each point. The following function simply computes the eucledian distance of the point $\boldsymbol{x}$ and all points in $\boldsymbol{X}$.

Listing 1: Distance function

```python
def distance(x, X):
  return np.sqrt(np.sum((x - X) ** 2, axis=1))
```

1.2 Gaussian Kernel

The Gaussian kernel is used since it provides continuous and smooth distributions and are suitable to extract information also in noisy data.

Give a bandwith, the gaussian kernel is simply the following:

Listing 2: Gaussian Kernel

```python
def gaussian(dist, bandwidth):
  K = np.exp(-(dist ** 2) / (2 * bandwidth ** 2))
  return K
```

1.3 Update point

Updating the point is the part in where the algorithm start to find the clusters. The update of a point $y^{(t+1)}$ is computed as following.

$$y_j^{(t)} \leftarrow \frac{\sum_{i=1}^{n} X_i w_i(j)}{\sum_{i=1}^{n} w_i(j)}$$

where the $w_i(j)$ are the weights for the j-th point computed with the gaussian function above.

1.4 Iterate the algorithm

The final step is to compute the update for each point and compute it for a number of iteration equal to 20 and then compute the centroids:

Listing 3: Mean-shift

```python
def meanshift_step(X, bandwidth=1):
  for i in range(X.shape[0]):
    dist = distance(X[i], X)
    weight = gaussian(dist, bandwidth)
    X[i] = update_point(weight, X)
  return X

centroids, labels = np.unique((X / 4).round(), return_inverse=True, axis=0)
```

2 Results

First, the number of colors was 24 so we were constrained with the number of clusters, if the results gave more clusters than the number of colors we would have an error when running. This has been noticed when using different values of the bandwidth in the gaussian kernels. Lower bandwidth correspond to local understanding of data and so to a greater number of clusters, while higher bandwidth will have more general understanding of the data and tend to reduce the number of clusters. Without any further fucntions, for example, the number of clusters for bandwidth equal to 1 and 2 were 284 and 44, that are more than the number of colors. After running, the bandwith suitable for this exercise were from 2.5 on. Below the images corresponding to bandwidth = 2.5, 3, 4, 5, 6, 7. To fix the problem of too many cluster, we can aggregate cluster, for example using K means or as I did in the code in the function shrink_labels, just include each cluster to the nearest of the 24 biggest clusters. The results were obviously more imprecise. The last 2 images correspond to the images for which we aggregate the clusters.

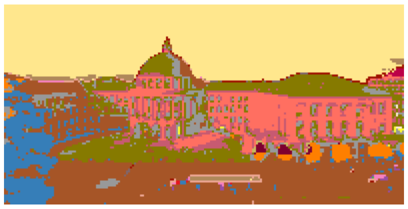
Figure 1: bandwidth = 2.5, clusters = 21

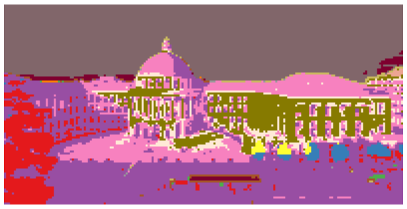
Figure 2: bandwidth = 3, clusters = 15

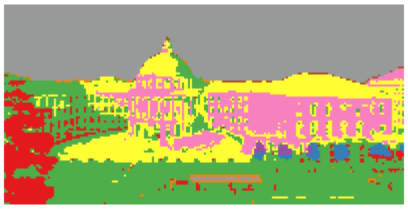
Figure 3: bandwidth = 4, clusters = 9

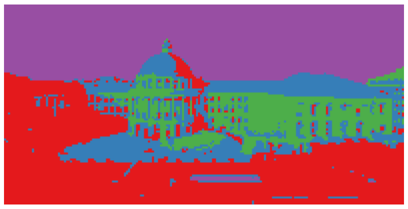
Figure 4: bandwidth = 5, clusters = 4

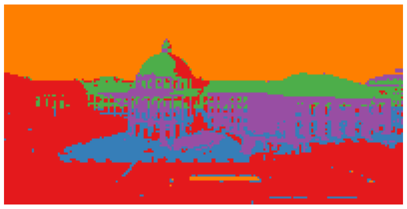
Figure 5: bandwidth = 6, clusters = 5

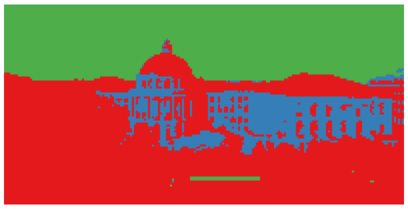
Figure 6: bandwidth = 7, clusters = 3

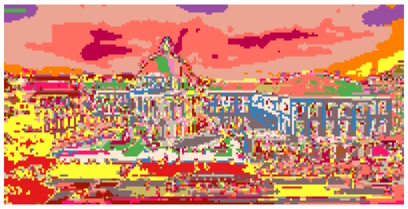
Figure 7: bandwidth = 1, clusters = 24, original clusters = 284

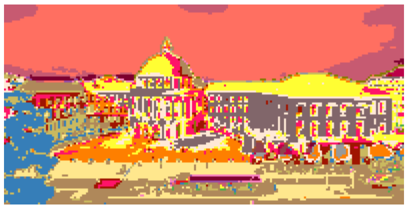
Figure 8: bandwidth = 2, clusters = 24, original clusters = 44