Faculty of Engineering - Alexandria University

Pattern Recognition

# Image Segmentation

Abdelrahman Mohamed Abdelnabi 3466
Ahmed Essam Shaheen 3321
Amr Mohamed El Sayed 3400

*Supervisor*
Dr. Marwan Torki

May 17, 2018

# 1 Introduction

Image segmentation can be performed by unsupervised clustering algorithms. Clustering is a form of unsupervised learning because it can perdict labels of datapoints without the need of having true labels at training time. K-means is a simple clusternig algorithm that is very good in clustering spherical data. Spectral clustering clusters data by modeling the similarity of points as a graph and tranforms the clustering problem to a minimum (or normalized) cut problem on the similarity graph. Similarity graphs can be k-NN graphs, rbf kernel similarity or, euclidean distance similarity.

# 2 Spectral Clustering with Normalized Cut

Pixel pairwise similarity is very expensive in memory. Images of size 321x481 have 154000 pixels. Computing the pixel pairwise similarity for such image requires a matrix of size $154000^2$ which is roughly 88.8 GBs!

However, by noting that the k-NN graph of an image this size is very sparse. Each row in the matrix of k-NN graph have only k entries of 1's in it compared to the row size of 154000. We can use scipy's sparse matrices to be able to store the similarity matrix in memory and perform eigenvalue decomposition on its normalized Laplacian.

The eigenvalue decomposition is very expensive and may not converge in some cases.

We use scipy's sparse solver to perform eigenvalue decomposition. The solver has a few handy parameters. One is `k`, the number of eigenvalues/eigenvectors required, and `which`, whether the largest magnitude, largest algebraic, smallest magnitude, or smallest algebraic eigenvalues/eigenvectors required. We need the smallest eigenvalues, so we try getting the smallest magnitude eigenvalues but the solver takes a very long time and may not converge so we tried the smallest algebraic eigenvalues and it worked.

For each image, we construct the K-NN graph for its pixels. The K-NN graph is represented with an adjacency matrix, where `matrix[i][j]` is 1 iff $j$ is in the KNN of $i$, otherwise it is 0. However, this matrix is asymmetric because the KNN similarity between two nodes is asymmetric. To make the matrix symmetric, we take its average between itself and its transpose [**?**].

# 3 Encoding Spatial Layout

Previously, the pixels given to the clustering algorithm only had 3 components, red, green, and blue. The algorithm clustering the pixels based on their color components only, which means pixels with similar colors can easily end up in the same cluster even if they are too far in the actual image.

To encode the information abou the pixel layout, we add two extra components to each pixel, its x and y cooridnates in the actual image. This was the pixel similarity is also affected by the position of each pixel not just its color components.