Machine Learning E16 - Handin 4
Representative-based Clustering Algorithms

Group 22
Mark Medum Bundgaard, Morten Jensen, Martin Sand Nielsen

December 19, 2016, Aarhus University

Requirements from Ira's description

- The status of the work, i.e., does it work, if not, then why.

- A discussion of plots of at least two runs of your algorithm implementations detailing what you can see. Make sure that you relate this to the discussion in the lecture or textbook about the strengths and weaknesses of the algorithms.

- A discussion of plots of the evaluation measures F1 and silhouette coefficient, detailing what you can learn from them. Include an explanation of what the evaluation measures reflect.

- Describe how you can use one of the clustering algorithms for image compression, and demonstrate the results for at least one algorithm on both images, discussing their quality and giving a reasoning for the differences. (Done, når billederne er komprimeret!)

K-keans clustering

Gaussian Mixture-model Expectation Maximization

Silhouette measures & F1 scores

To evaluate clustering performance, some sort of measure is useful. The F_1 score is based on simply evaluating the ratios of correctly clustered data points. The F_1 score is defined as,

$$F_1 = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}},$$

where precision is the ratio of selected elements which are relevant, while recall is the ratio of relevant elements being selected. The F_1 score is thus bound to the interval between 0 and 1, where 1 denotes a perfect clustering.

F_1 scores for various clusterings performed with both K-means and EM-GM and a combination, where K-means is used to initialize GM-EM centers, can be seen in Figure 1 and 3. To illustrate the variating results for some for some configurations all clustering has been performed 20times and averaged. The
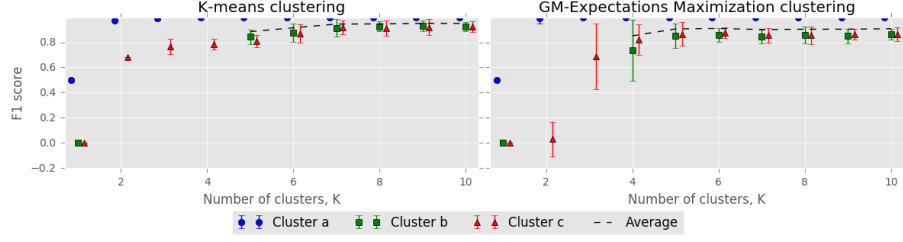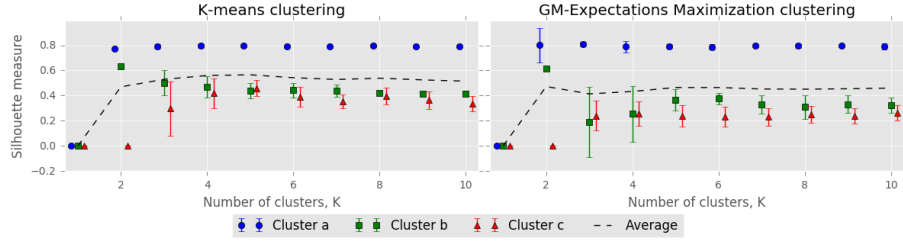
Figure 1:

Figure 2:

standard deviation is marked for each such average. The clusters are assigned
labels by voting of its assigned data points (supervised).

The Silhouette is another clustering performance measure, which relies on
a measure of similarity of points clustered. This allows for some evaluation
of unsupervised clusterings. For Silhouette a similarity measure between two
data points has to be chosen. For this exercise a measure of Euclidean distance
has been used as such. Silhouette is calculated for each data point, and can
then be averaged for each cluster, which again can be averaged to evaluate the
clustering. For the i'th data point,

$$\text{silhouette}(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}},$$

where the dissimilarity $a(i)$ is the average distance to all other data points in
its cluster, and the dissimilarity $b(i)$ is the average distance to all data points
in its nearest neighbouring cluster. This is fairly many computations when
working with large data sets. The measure is bounded to the interval between
-1 and 1. A plot of Silhouette measures for various number of clusters k, can
be seen in Figure 2. Naturally only positive silhouette measures is found for
our clusters, since both clustering methods use distance for descissions.

Both measures indicate that $k = 4$ is the most suitable for the data set.
One of the three labels seems to be best described by two subclusters(when
using K-means), or sum of two Gaussian distributions (when using GM-EM).
Naturally only bad results occur with $k < 3 =$ number of labels.
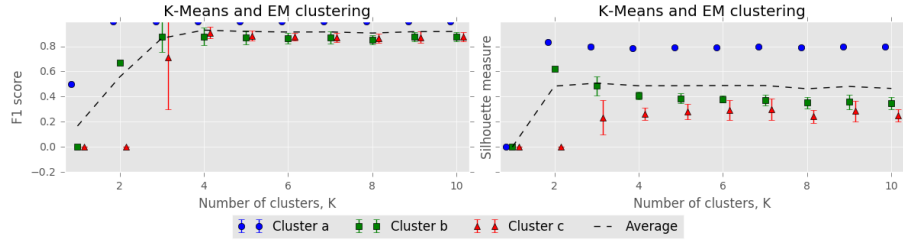
Figure 3:

Image color compression with K-means

The individual pixels in an normal RGB image holds information about the intensity of the three color channels. The color depth is given by the possible color combinations, which is usually 24 bit; 8 bit for each color channel. In a normal picture far from all possibilities af colors are present, so much of the color depth is unused. By limiting the number of possible colors to a small sub set of the full color depth, and thus let each pixel refer to a color in this color set, disk usage can be decreased significantly. The color compression would then be a algorithm deciding a color set and transform each pixel as to refer these. Likewise a decompression algorithm would then do the reverse transformation by looking up each pixels reference color. So the space savings comes at two additional computational costs.

The space savings would be closely related to the size of a color set. And how few bits required for each pixel to index the color map. To further reduce the size, certain very closely related colors, could be said to yield same color, since small color variations can be almost impossible to notice to the human eye. This would add a quality loss to the compression, but often at a great reduction in color space. To create a color set, where similar colors are mapped together, K-means and GM-EM has been tried on the two test images shown in Figure 4.

A reasonable depth would be some 2^n, where n is the number of bits necessary for each pixel to reference the color map. In Figure 5 images are compressed to $2^4 = 16$ colors using both K-means and GM-EM. Likewise in Figure 6 images are compressed to $2^8 = 256$ colors using both K-means and GM-EM. The space savings relative to a raw 24bit bitmap approximate

$$\text{saving ratio} \approx \frac{3 \cdot 8\,\text{bit} - n\text{bit}}{3 \cdot 8\,\text{bit}},$$

for large images, where the additional space needed for the color map is negligible. For the 16 and 256 color maps, the savings are 83% and 67%.

Image quality loss acceptance limit will always be subject to personal taste and dependent on purpose. Compression of images can be done in various ways, and is done most image formats used for photographs. Color depth compression with ie. K-means to determine optimal color map of a given size, is very computational demanding, and probably not very useful. A fixed colormap could decrease image quality, depending on how well the pixel colors matches the color map.
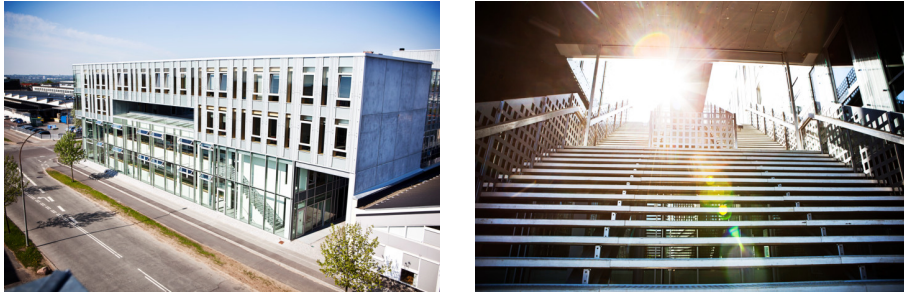
3

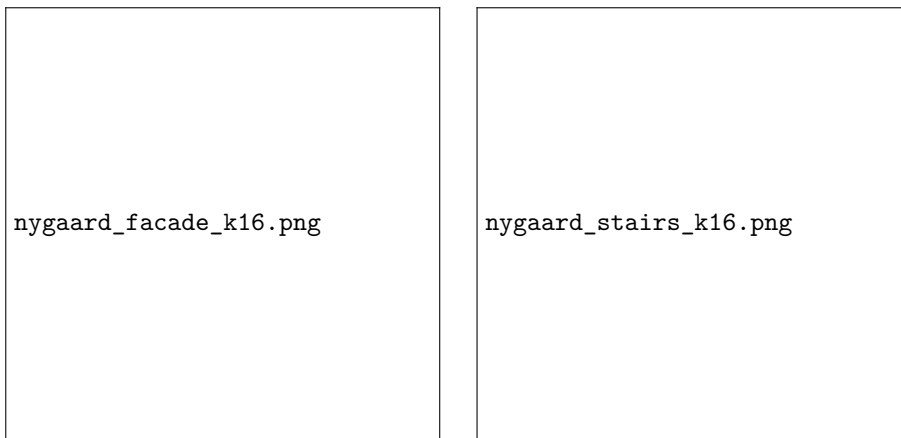Figure 4: *Uncompressed* nygaard_facade.jpg *and* nygaard_stairs.jpg *images.*

nygaard_facade_k16.png

nygaard_stairs_k16.png

Figure 5: *Images with color depth compressed to a set of 16 colors (4bit) with the K-means algorithm.*
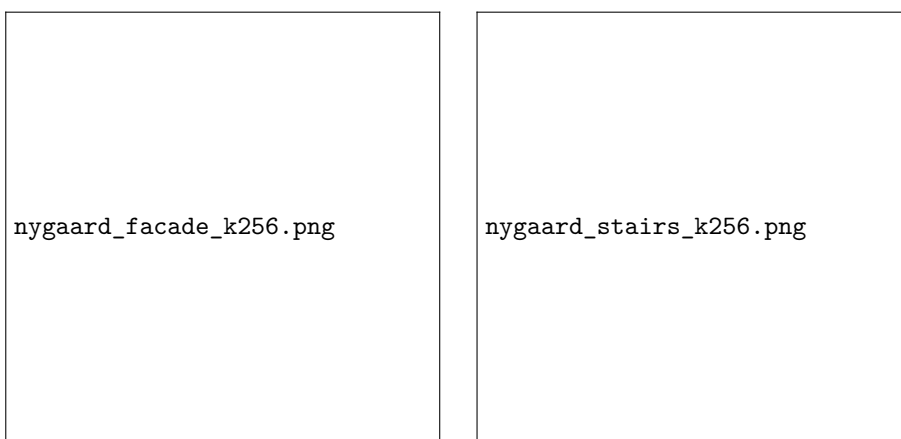

nygaard_facade_k256.png

nygaard_stairs_k256.png

Figure 6: *Images with color depth compressed to a set of 256 colors (8bit) with the K-means algorithm.*