# 30540 - Mapping from Aerial and Satellite Images
# Supervised Classification

Anders Richard Pankoke Holm
s144125

Hand-in date: 13/03-2019

## Quadratic Maximum Likelihood (QML)

We have earlier tried to identify different classes of images using Unsupervised Classification, where we based on clusters in feature space would assign pixels to classes based on the Euclidean distance to each of the cluster centroids. With Supervised Classification we have some training data, which we use to determine the overall appearance of a class, based on the intensities of the different bandwidths. Our data consist of a 6-band image (red, green, blue and 3 infrared) of the northern part of the capital region of Denmark, thus we see Copenhagen, Amager and the municipalities right outside as well as Roskilde. In addition we have many fields used for agriculture, the water surrounding Copenhagen and finally we see some miscellaneous vegetation. This gives us 4 classes: Water, forest, urban areas and fields. For our training images, we draw polygons on the map using the `roipoly` function, which will assign a user-defined area to one of the chosen classes. The polygons for our 4 drawn areas as well as the original photo can be seen in figure 1.

When we have our 4 classes, we would like to classify all the pixels according to the general properties of the training images. To do this we use the Max A Posteriori Probability (MAP) estimation, which builds on Bayes' rule.

$$P(\omega_c|\mathbf{X}) \propto P(\mathbf{X}|\omega_c)P(\omega_c) \tag{1}$$

where $P(\omega_c)$ is the a priori probability, which is derived from some known properties of the problem. We set $P(\omega_c) = 1/k$ for this specific problem, such that we say each class is equally likely ($k = 4$). $P(\mathbf{X}|\omega_c)$ is the likelihood that we would have some data given the class $c$ for $c = 1, \ldots, k$. We set this to be given by the determinant of the dispersion matrix of the class $|\Sigma_c|$. $P(\omega_c|\mathbf{X})$ is the a posteriori, which is the probability of obtaining class $c$ given the data $\boldsymbol{X}$. We want to maximize the Gaussian likelihood of the $p$-dimensional problem ($p$ being the number of bands used to classify), where we compute the log-likelihood of the class $c$ given the data $\mathbf{X}$. This is computed using eq. (2).

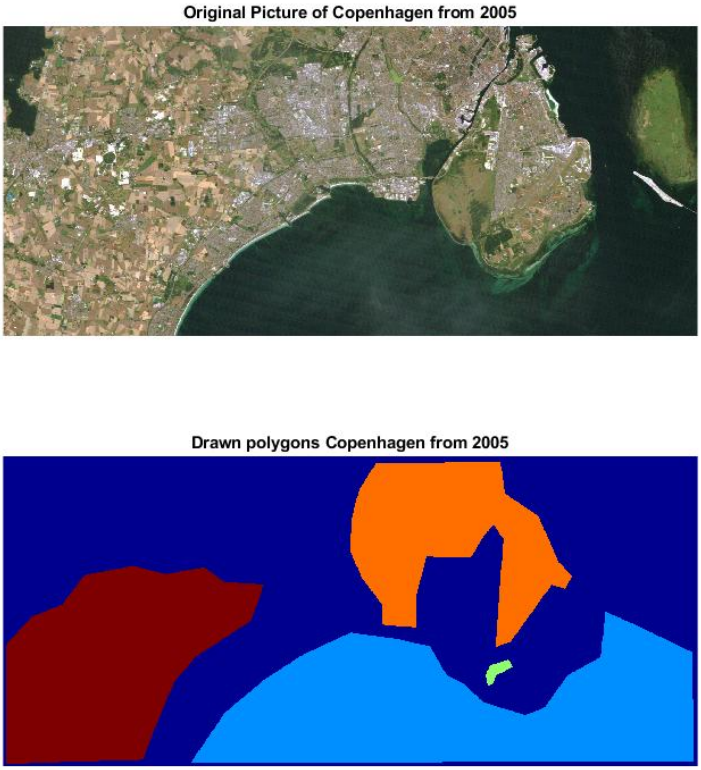


Figure 1: **Top:** Image of Copenhagen displayed using RGB. **Bottom:** Drawn class polygons.

$$\mathcal{L}(\mathbf{X}) = \ln P(\omega_c) - \frac{1}{2}\ln|\Sigma_c| - \frac{1}{2}(\mathbf{X}-\mu_c)^T\Sigma_c^{-1}(\mathbf{X}-\mu_c) \tag{2}$$

where $(\mathbf{X}-\mu_c)^T\Sigma_c^{-1}(\mathbf{X}-\mu_c)$ is the Mahalanobis distance. Different from the Euclidean distance, points that reside on the confidence hyper-ellipsoids in feature space, all have the same Mahalanobis distance to the cluster centroids $\mu_c$. As an alternative, we could also have minimized the Mahalanobis distance in order to classify, which will produce a different result. When we have the likelihoods, we simply assign a pixel to the class it is most likely to be a member of. A result of the classification can be seen in figure 2.

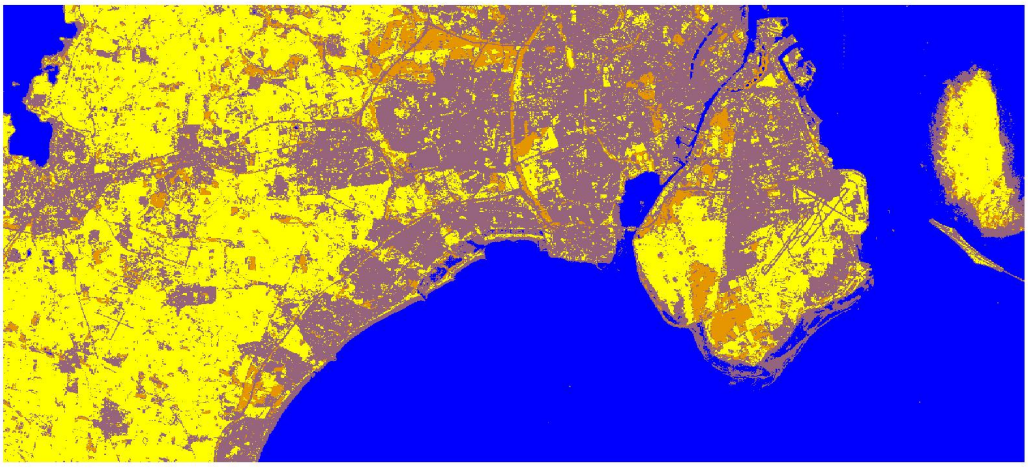

Figure 2: Classification of Copenhagen. **Blue:** Water. **Grey:** Urban. **Yellow:** Fields. **Orange:** Forest.

The method perfectly classifies the water, as we both see it around Copenhagen and in Roskilde Fjord. We see that a lot of the coastal areas are classified as urban areas as well, which is wrong given that these areas mainly contain a lot of sand. We get a few areas with forests and the rest is deemed to be fields.

We produce a confusion matrix, which tells us how many of the pixels were initially classified as one of the 4 classes, and how many of them ended up in either the original class or some other. We see that the consumer precision of the forest is low, given that the training area was very small, but many other pixels were classified as forest.

| Known class\Classified as | Water | Forest | Urban | Fields | Row total | Producers accuracy (%) |
|---|---|---|---|---|---|---|
| Water | 539104 | 1 | 405 | 0 | 539510 | 99.9 |
| Forest | 0 | 3443 | 105 | 30 | 3578 | 96.2 |
| Urban | 2747 | 13069 | 208289 | 32980 | 257085 | 81 |
| Fields | 91 | 14444 | 71593 | 296386 | 382514 | 77.5 |
| Column total | 541942 | 30957 | 280392 | 329396 | 1182687 | |
| Consumer's accuracy (%) | 99.5 | 11.1 | 74.3 | 90 | | |

We could very likely improve the classification by introducing more classes, like the coastal areas or different urban environments. We will also try performing a PCA and only do the classification from the first couple of principal components. With the first 2 PC's, we get more than 95% of the variance, which will thus suffice for further analysis. Result of classification can be seen in figure 3. The difference between the two can be seen in figure 4.

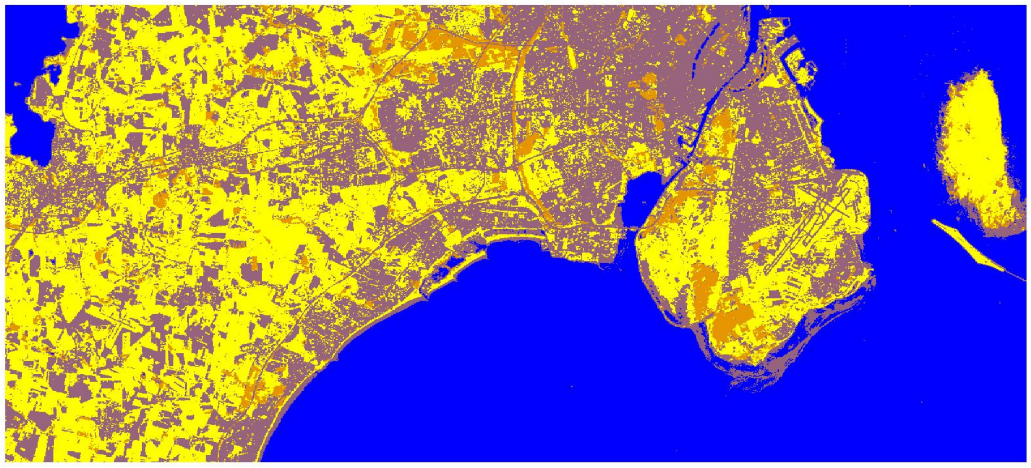

Figure 3: Classification of Copenhagen. with first 2 Principal Components **Blue:** Water. **Grey:** Urban. **Yellow:** Fields. **Orange:** Forest.

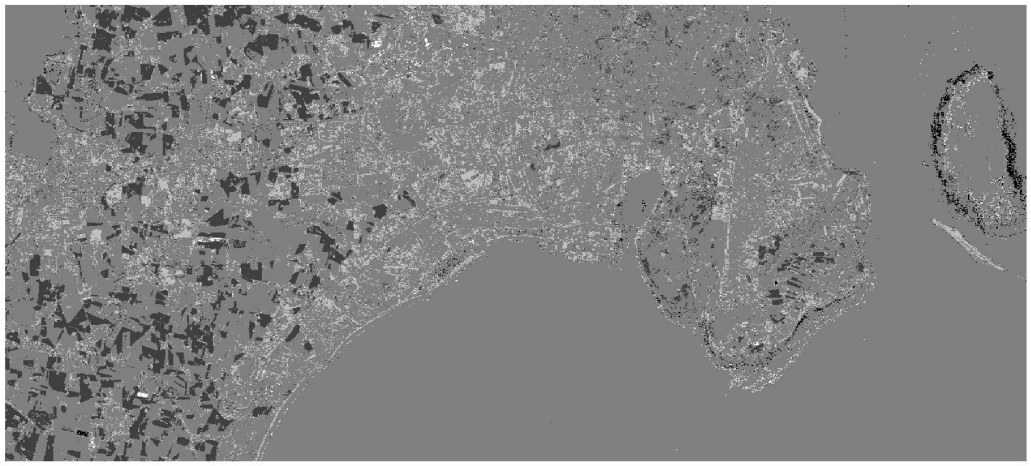

Figure 4: Difference between first classification from all 6 spectral bands and the classification from the first 2 principal components

The biggest difference that we see is in the classification of the fields, where many are now classified as urban areas, where in reality, they it is more likely to just be different types of crops or crops planted at different time periods.

## Support Vector Machine (SVM)

For this part, we will do binary classification. This means that we only pick 2 classes from the `igalmss` data set. Just like in former section, we draw polygons on the image for the desired classes. We pick water and everything else for our classes. In order to work with the concept of SVM, we would like to reduce the number of bands we use, such that we will not have to visualize the data in hyperspace. For this we perform a PCA and see that the data variance is represented at 98.6% with the first 2 principal components, meaning we can visualize the data in 2D.

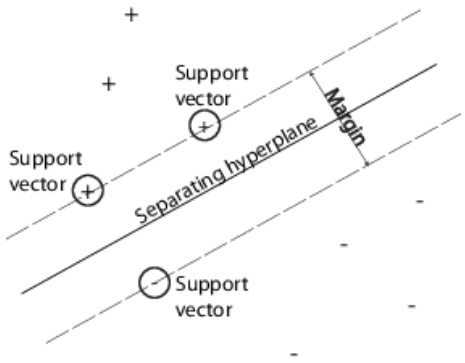


Figure 5: Separation of classes using SVM (from MathWorks website's page on SVM)

The principle of SVM is to draw a hyperplane in feature space (line in 2D feature space) which will separate the data into 2 classes like shown in figure 5. The data points located either very close to or beyond the separating line are deemed "support vectors". When using a kernel to perform the SVM procedure, we get the results seen in figure 6.

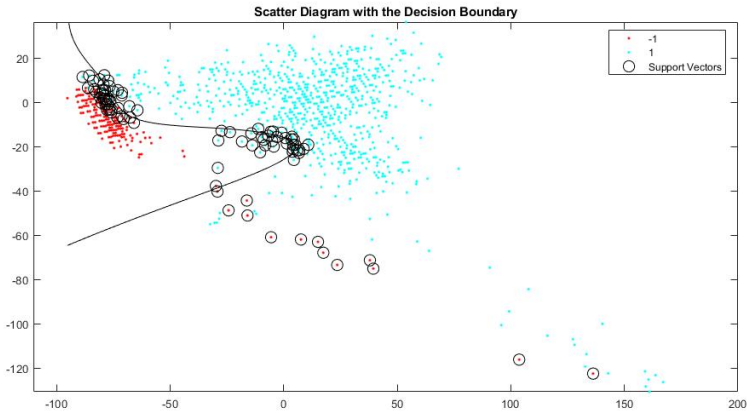


Figure 6: Separation of 1000 random samples from both training images using SVM with kernel

With this model, we can then classify the remaining pixels of the image.

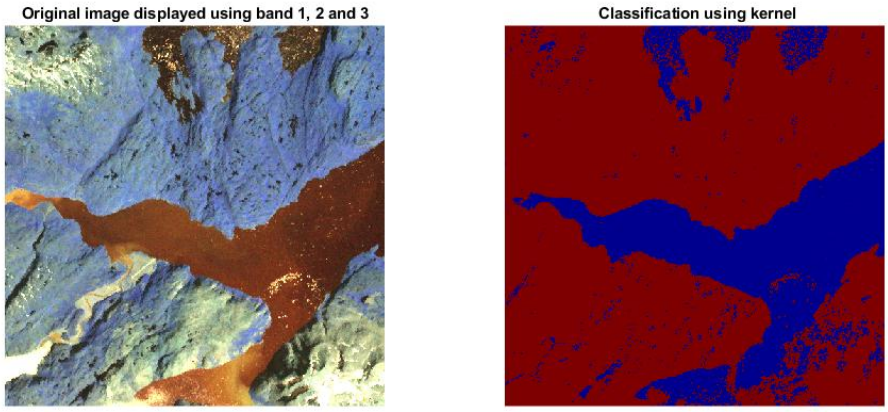


Figure 7: Classification of pixels using SVM with kernel

If we don't use the kernel and do the linear SVM, we get the following result.

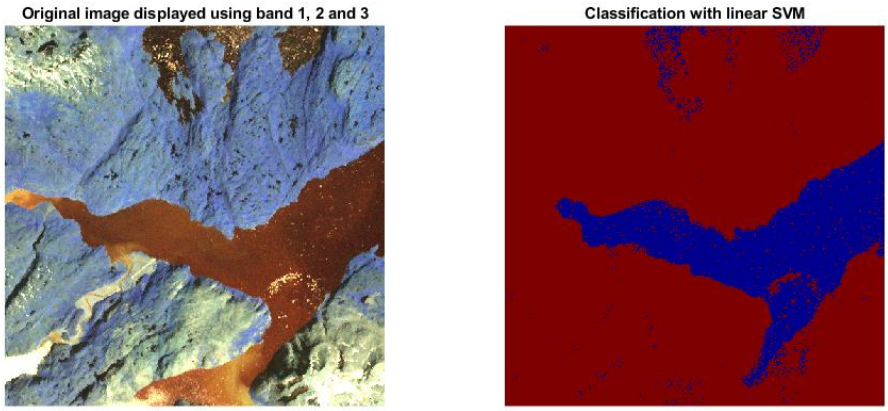


Figure 8: Classification of pixels using linear SVM

With the kernel, we get a lot more of the water, but we also see some of the shadowy mountainsides being labelled as water. We also see less noise in the areas we know to be water, when we use the kernel.