Density-Based Unsupervised Classification for Remote Sensing

C.H.M. van Kemenade, J.A. La Poutré, R.J. Mokken

Software Engineering (SEN)

SEN-R9810 July 31, 1998

Report SEN-R9810
ISSN 1386-369X

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

Density-Based Unsupervised Classification for Remote Sensing

Cees H.M. van Kemenade
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
Cees.van.Kemenade@cwi.nl

Han La Poutré
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
Han.La.Poutre@cwi.nl

Robert J. Mokken
Center for Computer Science in Organization and Management (CCSOM),
Department of Statistics and Methodology, PSCW,
University of Amsterdam,
Sarphatistraat 143, 1018 GD Amsterdam, The Netherlands
mokken@ccsom.uva.nl

ABSTRACT

Most image classification methods are supervised and use a parametric model of the classes that have to be detected. The models of the different classes are trained by means of a set of training regions that usually have to be marked and classified by a human interpreter. Unsupervised classification methods are data-driven methods that do not use such a set of training samples. Instead, these methods look for (repeated) structures in the data.

In this paper we describe a non-parametric unsupervised classification method. The method uses biased sampling to obtain a learning sample with little noise. Next, density estimation based clustering is used to find the structure in the learning data. The method generates a non-parametric model for each of the classes and uses these models to classify the pixels in the image.

1991 Mathematics Subject Classification: 62D05, 62G07, 62H30, 62Pxx, 68U10
1991 Computing Reviews Classification System: I.4.6, I.4.9, I.5.3, I.5.4
Keywords and Phrases: image classification, biased sampling, spatial statistics, density-based clustering, multi-spectral data
Note: Research carried out under theme "Evolutionary and Applied Algorithmics" and supported under US Govt. European Research Office contract #N68171-95-C-9124.

1. Introduction

Classification in remote sensing involves the mapping of the pixels of an image to a (relatively small) set of classes, such that pixels in the same class are having properties in common.

Until recently, most satellite imagery was at a relative low-resolution, where the width of a single pixel was between 10 meters and 1 kilometer. Nowadays new satellites come into orbit that produce imagery with high spatial resolution. Such high-resolution images offer new opportunities for applications. In theory, a better classification is possible, as high-resolution images offer more information. In practice, lots of new problems appear. Due to the high spatial resolution, many objects that are too small to locate in low-resolution images are visible in high-resolution images. As a result, the number of different classes that can be detected increases, and the discrimination between classes becomes

more difficult. In case of low-resolution imagery one often assumes that a pixel value consists of the spectral vector of the underlying ground cover class plus some Gaussian distributed noise component. It is questionable whether this model is still applicable for high-resolution imagery. Therefore, we use non-parametric models. These models are more flexible with respect to the shapes of the distributions modeled.

Apart from the spectral information, there is also spatial information available in the image. This spatial information is heavily used by the human interpreter but is not used during a purely spectral classification. So, these methods disregard the similarities between neighboring pixels during classification. Incorporation of the properties of the neighboring pixels means, that we try to exploit the spatial structure present in the image.

The outline of this paper is as follows. In section 2 we discus (non-)parametric unsupervised classification methods. The outline of our approach is given in section 3. The selection of the learning-sample is discussed in section 4. Section 5 gives a brief introduction on density estimation, followed by a discussion of the curse of dimensionality in section 6. A more detailed investigation of the application of density estimation in the remote sensing domain in section 7. The density-based clustering method is given in section 8. Details about the usage of these methods for remote sensing are given in section 9. Conclusions and directions for further research follow in section 10.

2. Unsupervised non-parametric classification

We have developed an unsupervised non-parametric classification method. This classification method uses a clustering method to find the structure in the data. A good introduction on clustering is given in Kaufman and Rousseeuw [8], and in Ripley [11]. First we explain the differences between supervised and unsupervised classification method, next the differences between parametric and non-parametric methods are discussed. At the end of this section, some examples of some other unsupervised clustering methods are given.

In case of a supervised classification method, a human interpreter has to do a pre-classification of part of the image, during which a set of regions is marked and each marked region is classified by the interpreter. Next, the supervised method can use these classified regions to obtain models that are used to classify the other parts of the image. During unsupervised classification we do not use a set of pre-classified training-samples. The method has to find a classification of the image just based on the image itself, although some general assumptions about images can be used.

A clustering methods can either be parametric or non-parametric. Parametric models usually assume a spherical or Gaussian model for the shape of a class. Let us assume that the noise contribution is mainly determined by small size objects that have a spectral vector that differs significantly from the spectral vector of its surroundings. During classification, we have to detect the class of the most important ground cover in the pixel. In case of low-resolution imagery, a single pixel can contain many different small objects, and their combined surface will usually be relatively small. If the sub-pixel objects belong to different classes, then a further cancellation of their spectral contributions is likely. The combined noise contribution can be modeled by a Gaussian distributed noise component. In case of high-resolution imagery, a pixel will only contain a small number of these disturbing objects. It is then questionable whether the Gaussian noise model is applicable, and even if it is applicable, then the variance is likely to be much larger as the disturbing objects are likely to cover a significant part of the pixel. A further complication stems from the fact that we have to discriminate between many more clases in case of high-resolution imagery.

A parametric method will fit the free parameters to the actual data as good as possible. If the model matches the actual data, then these parametric methods can provide fast and efficient ways of obtaining the actual clustering of the data. If the data does not match the model, then the quality of the clustering can decrease severely. A method for this type of problem is the isodata clustering procedure [5, 6]. This method does a minimization of the sum of the squared distances between the points and the corresponding cluster center. This corresponds to the assumption that the clusters are approximately spherical.

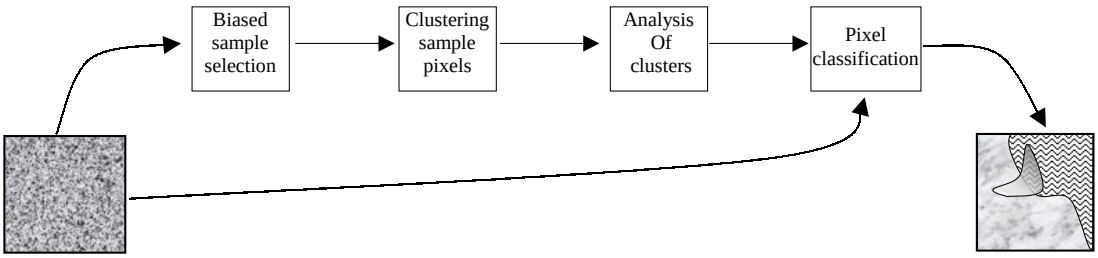

Figure 1: Schematic representation of the classification method

More flexible modeling of data is possible when applying non-parametric clustering methods. We develop a non-parametric clustering method that is based on density-estimation [12, 13]. Non-parametric methods are more general in the sense that less prior information with respect to the actual shape of clusters are made. The resulting problem is more complex since the search for a good clustering of the data now involves both model selection and model fitting. Unfortunately, these two modeling phases cannot be performed separately. A model selection has to be performed before the model fitting can take place. But model fitting is required to assess the quality of the selected model. In fact, the model selection and model fitting often are so tightly integrated that we often cannot differentiate between these two anymore.

3. Outline of the method

We use a classification method that is based on density-estimation. The output of the method is a non-parametric clustering of the set of input points. A parametric clustering will assume a certain pre-specified model of the clusters, where an instantiation of this model can be described by a (small) number of parameters. Figure 1 gives a schematic representation of the unsupervised classification method introduced in this paper. On the left we see the unclassified image, on the right we have a classified image. The four steps of the method are:

Sample selection: A biased training-sample is selected from the image. The bias is used to reduce the amount of noise and the fraction of mixed pixels in the sample. During this selection, spatial information is used. The training sample consist of the spectral vectors of the selected pixels.

Clustering: An unsupervised non-parametric clustering method is used to find the clusters. This clustering method is based on density estimation in the sample-space

Analysis: The clusters are analysed. For each cluster, the principal component is determined, and the distribution of the pixels when mapped at this principal component is investigated

Pixel classification: Using cluster information and the original image, a classified image is generated.

The sample selection is discussed in section 4 the clustering method is given in section 8. The last two steps are discussed briefly in section 9.

4. Biased sampling of pixels

We develop a biased sampling method, where the bias is directed towards the selection of pixels containing primarily one ground cover, and with little noise. We use it in our final algorithm inside a stratified random sampling framework. In this section, we first describe a stratified sampling procedure for images. Next, the local spatial homogeneity assumption is presented. A similarity measure that compares pixels to their neighborhood is introduced, and it is shown theoretically that this similarity

measure allows a biased sampling procedure that reduces the noise in the sample for a one-class problem. By means of an example, it is shown that this sampling procedure also works for multiple-class sampling problems. Finally, we apply the sampling procedure to a set of real images.

In case of large images, we have to select a limited sample of pixels from an image for learning. We would like this learning-sample to be representative for the complete image. Therefore we select a stratified sample by means of the following procedure. If we want a sample of size N, we partition the complete image in a set of N rectangular regions of approximately equal size. Next we select a pixel at random from each of these square regions. Using such a procedure guarantees that the points in the learning-sample are distributed uniformly over the complete image. Especially in the case that N is small, such a procedure is important.

The human eye is very sensitive to spatial structure in an image. We almost never observe individual pixels. Instead we observe regions of pixels that are relatively similar. Within these regions the pixels have approximately the same spectral vectors, or the region shows a texture that is quite apparant. We call this property the local spatial homogeneity of images. This homogeneity is due to the fact that pixels close to one another are likely to have the same primary ground cover. Next, we show how to use the local spatial homogeneity to reduce noise in the sample.

If a region contains a single ground-cover, it is possible to get a sample with pixels that contain little noise by means of biased sampling. Here, the probability that a pixel is selected is determined by a similarity measure. We introduce a similarity measure that computes the similarity $sim(p)$ between a pixel p and the pixels in its neighborhood $N(p)$. We show that the set of pixels with $sim(p) \leq \delta$, i.e. the pixels that are relatively similar to their neighbors, comprise a biased sample of pixels containing relatively little noise. Let the function $d(\vec{s}_1, \vec{s}_2)$ denote the spectral distance between two pixels. A similarity measure can be obtained by calculating $d(\cdot,\cdot)$ for all neighbors. Furthermore, the rank with respect to their distances is given by $r(i)$. The similarity measure we use is:

$$sim(p) = \sum_{i \in N(p)} W_{r(i)} d(\vec{s}_p, \vec{s}_i)$$

where p is the current pixel, $N(p)$ is its neighborhood, and $\vec{W}$ is a weight vector such that $|\vec{W}| = 1$. The best choice of $\vec{W}$ will depend on the structures one is interested in. If thin structures should be detectable, such as rivers or roads that have a width comparable to the width of a pixels, then W_1 and W_2 should be the only non-zero values. If one is not interested in such thin structures, it is better to have $W_k > 0$ for higher values of k, as this puts a stronger emphasis on the similarity of a pixel with respect to its neighbors, and therefore a stronger emphasis on the homogeneity of the pixel. In the theoretical analysis that follows, we assume that we have a set of t pixels that are drawn according to the same distribution. From this set of pixels, the biased sampling method should select those pixels that contain relatively little noise, without knowing the center of the distribution. This analysis is mainly a proof of the principle for the biased sampling. In case of such a nice uni-modal distribution, it is also possible to estimate the location of the center by statistical methods, and select pixels that are close to this center. But such a simple approach fails in case of a multi-modal distribution. If all peaks of the multi-modal have approximately the same density, then it would be possible to select pixels in regions of high density, to get a set of relatively noise-less pixels. When densities of peaks differ strongly, a peak can only be recognized by the fact that its local spectral density is higher than the spectral density in a spectral neighborhood.

4.1 Theoretical analysis of a one-class problem

A theoretical analysis is possible for the case that $W_k = 1$, for a certain k. This corresponds to the case where $sim(p)$ equals the distance to the k^{th} nearest neighbor. Assume that the spectral vector of a pixel is composed of the the spectral vector of the primary ground cover within the area of this pixel and Gaussian distributed noise with variance σ. Given a spatial region containing a set of pixels that have the same primary ground cover, we can observe the distribution of these pixels in the spectral

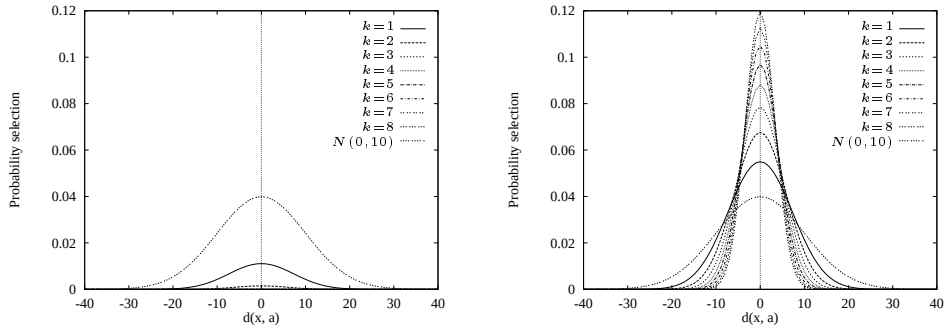

Figure 2: Density functions of samples obtained by using a series of values for k before (left) and after (right) normalization

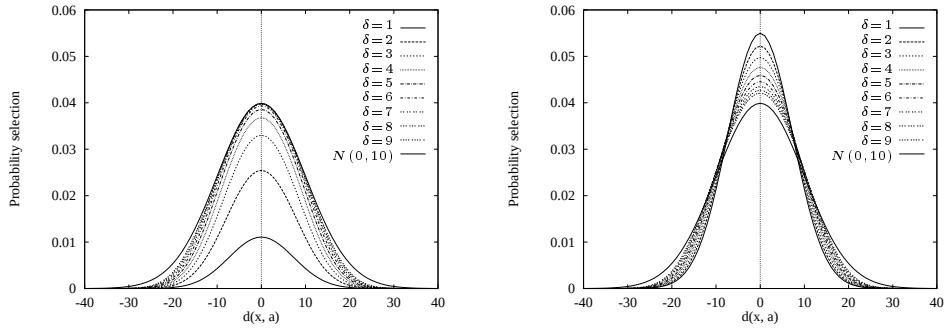

Figure 3: Density functions of samples obtained by using a series of values for δ before (left) and after (right) normalization

space. Let the distribution of pixels in the spectral space be given by the density function $f(\vec{s})$. We select pixels p such that $sim(p) \leq \delta$. Now the distribution $g(x)$ of te obtained sample is given by the formula

$$g(\vec{s}) = f(\vec{s}) \sum_{j=k}^{t} B\left(t, j, p(\vec{s}, \delta)\right)$$

$$\text{where: } p(\vec{s}, \delta) = \int_{Sphere(\vec{s},\delta)} f(\vec{s}) \mathrm{d}\vec{s}.$$

Here $B(n, k, p)$ is the binomial distribution, where $0 \leq k \leq n$, t is the number of pixels in neighborhood $N(p)$, and $Sphere(\vec{s}, \delta)$ denotes a d-dimensional ball of radius δ around the point $\vec{s}$. So, after performing the sampling step we get the original density multiplied by a factor that depends on the averaged local density around spectral point $\vec{s}$. The value of k can be varied between one and t. The case $k = 1$ corresponds to computing the distance to the nearest neighbor, while the case $k = t$ corresponds to computing the distance with respect to the most dissimilar neighbor. Large values of k correspond to a stricter criterion for homogeneity and therefore result in the selection of less pixels.

We computed the curves for one-dimensional data and a Gaussian noise model. We assume that the noise is given by the Gaussian distribution with $\mu = 0$, $\sigma = 10$. The neighborhood $N(p)$ consist of the 8 adjacent pixels. Figure 2 shows the unnormalized and normalized plots of the formula in one dimension for varying values of k and $\delta = 0.5$. Along the horizontal axis the distance between the actual value and the noise-less value $d(x, a)$ is shown. This value can be interpreted as the amount of

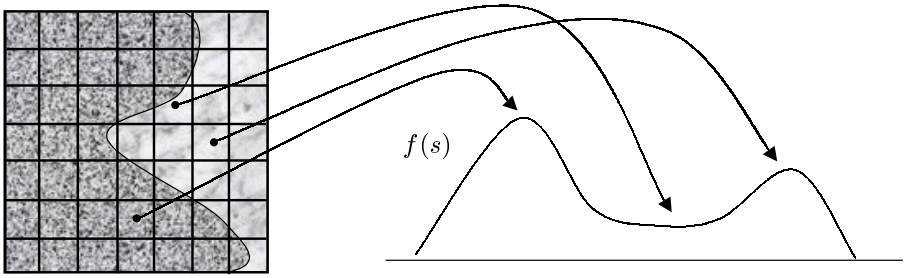

Figure 4: A square region containing two types of ground cover and the corresponding density function obtained when mapping all pixels to the spectral space.

noise in a pixel. The vertical axis corresponds to the probability that a corresponding pixel is selected. As k increases the probability of selection decreases rapidly, so the size of the sample gets smaller. When normalizing the results, it can be observed that the distributions tend to be more strongly peaked as k increases. So the selection is biased towards relatively noiseless pixels.

The threshold δ can vary between 0 and ∞. By increasing δ the selection criterion is weakened and more pixels are included. In the limit of $\delta \to \infty$ all pixels are selected, and the distribution will converges to the Gaussian. Figure 3 shows the unnormalized and normalized plots for different values of δ and $k = 1$.

4.2 Discussion of a multi-class problem

In section 4, a biased sampling method was introduced, and in the previouys subsection, it was shown that this method is able to select a set of relatively noiseless pixels in case of a one-class region. Now, we show intuitively that the sampling procedure also works in case of multiple classes, and hence in case of a multi-modal density function of the pixels. Figure 4 shows a region containing two classes, i.e. two types of ground cover. The Figure also shows the density function obtained when mapping all pixels from the square patch to the spectral space. The arrows show the locations of three different pixels in the density-plot. The peak on the left of the graph corresponds to the main ground cover within the square patch. The peak on the right corresponds to the other ground cover. The ridge inbetween these two peaks corresponds to the mixed pixels, that are located on the boundery of the two sub-regions. When computing the similarity measure $sim(\cdot)$ for these three pixels, it is likely that the two non-mixed pixels have a relatively low value of $sim(\cdot)$, as these pixels are surrounded by other pixels with the same primary ground cover. The mixed pixel probably has a high value of $sim(\cdot)$. Therefore the two pixels containing a single ground cover are more likely to be selected than the mixed pixel.

To summarize, the local spatial homogeneity is exploited by means of the assumption that pixels in the spatial neighborhood $N(p)$ of point p are likely to belong to the same class as point p. Pixels with lots of noise and mixed pixels are both likely to have a large value of $sim(p)$. So, if a biased sample is selected according to the rule $sim(p) \leq \delta$, then this sample is likely to contain the pixels with little noise and containing only a single ground-cover.

Next, we describe the results of applying the biased sampling method to a number of real images, and compare it with a stratified random sample. All images are three-band images, with a sample size of 4000 pixels. The stratified random sample is taken by dividing the image in a set of non-overlapping rectangular areas, and taking a random pixel from each area for the real images. The distribution in spectral space of such a random sample is compared to the distribution when using a biased sampling. During the biased sampling a weightvector with $W_3 = 1$ is used. The value of δ was adapted such that exactly 4000 pixels were selected, so in fact we select the 4000 most homogeneous pixels. The results are shown in Figures 5, 6, 7, and 8. In each of these Figures, the left graph corresponds to

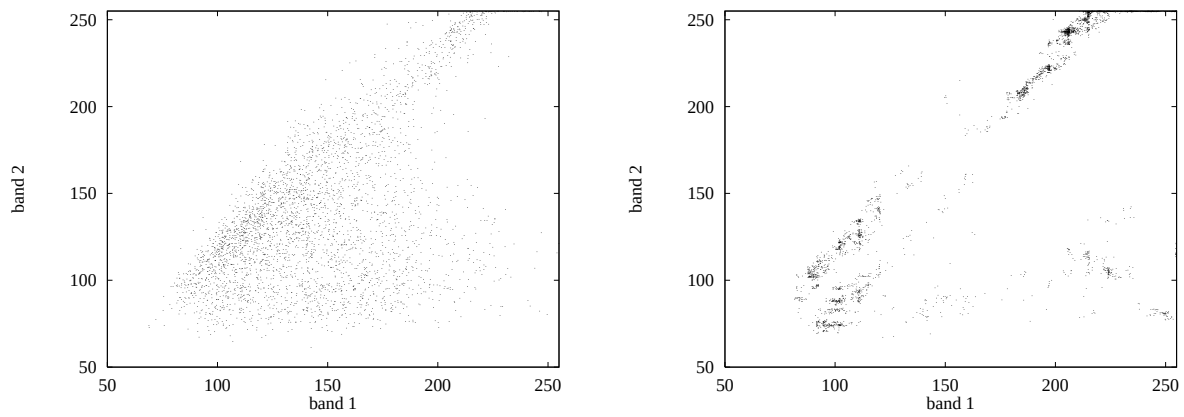

Figure 5: Results for image 1 with stratified random sampling (left) and biased sampling with $k = 3$ (right)

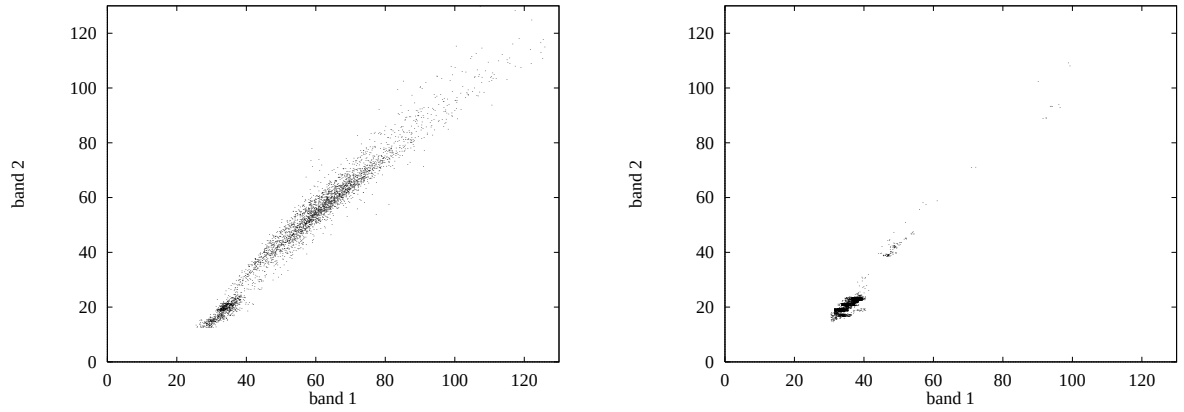

Figure 6: Results for image 2 with stratified random sampling (left) and biased sampling with $k = 3$ (right)

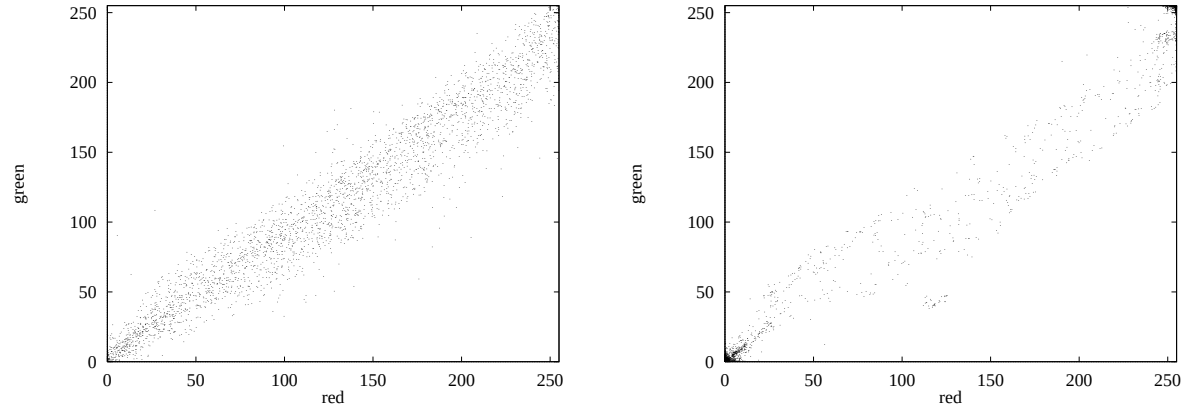

Figure 7: Results for image 3 with stratified random sampling (left) and biased sampling with $k = 3$ (right)

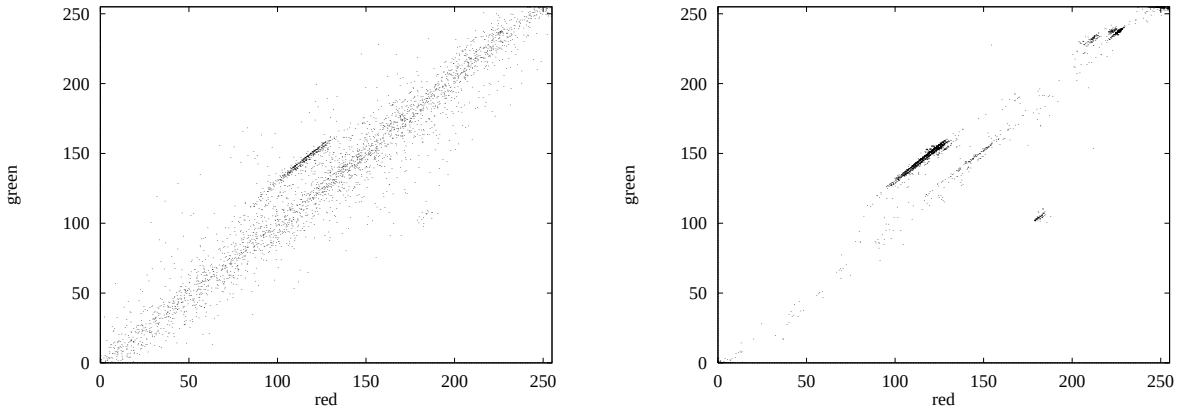

Figure 8: Results for image 4 with stratified random sampling (left) and biased sampling with $k = 3$ (right)

the stratified random sample, and the right graph corresponds to the biased sample. In all cases, the biased sample is reveals more structure than the random sample. This shows that the biased sampling method is able to highlight the cluster structure in the image-data.

5. Density estimation

As a basis for our clustering method we need a density estimation method [12, 13]. Assume that a set of points is drawn according to an unknown distribution. A density estimation method constructs a density function based on this data-set, that explains the distribution of the data-points. This density function is related to a distribution function, which should approximate the original, but unknown, distribution function that generated the data.

This method has to be able to handle multi-dimensional data. Furthermore, it is important that it operates fast, as the density estimation method is used as part of the clustering method. The oldest method for density estimation is the histogram method. In a univariate case a range of data-values is split up in a number of bins, and the number of data-points in each of this bin is counted and used as an estimate of the local density. The width of the bins is called the window size. Histogram methods do not perform well in high-dimensional spaces, as the number of bins rises exponential with the dimension, so one either needs an excessively large number of data-points, or one should use a large window size, which results in a very coarse estimation of the density function.

Most modern density estimation methods are based on kernel estimators. The basic kernel estimator may be written compactly [12] as

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) = \frac{1}{n} \sum_{i=1}^{n} K_h(x - x_i),$$

where $K_h(t) = K(t/h)/h$. In this formula x is the point where the density is estimated, n is the number of data points, h is the window size, x_i are the actual data points, and $K_h(t)$ is the kernel of the density estimation function. This kernel determines how points at distance t from the point x influence the density at point x.

We use a density estimation method with an adaptive window size h_x that is based on the distance to the k^{th} nearest neighbor, the k-NN estimator [13]. The k-NN method and the kernel discrimination were first given by Fix & Hodges [2], according to Ripley [11]. This method is fast, it performs good in high dimensional spaces, and it fits well within the framework of the classification method described in section 8. Let the distance to the k^{th}nearest neighbor be denoted by $d_k(x, \{x_i\})$, where $\{x_i\}$ denotes

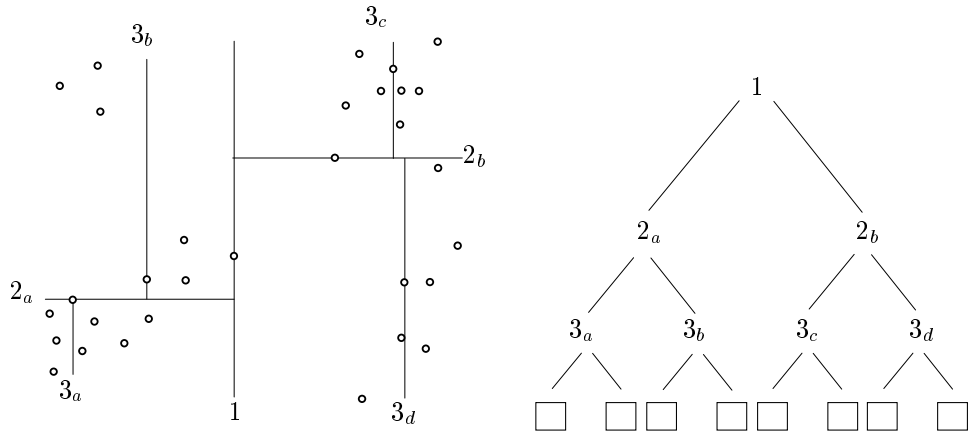

Figure 9: Example of a kd-tree on a two-dimensional dataset. On the left the data-points and the hyperplanes that subsequently split the dataset are shown, and on the right the corresponding kd-tree is shown.

a list with all data-points. Now the local density at a query point x can be estimated by the assuming a uniform density over a sphere centered at x with radius $d_k(x, \{x_i\})$, which leads to the formula

$$\hat{f}(x) = \frac{k}{d_k(x, \{x_i\})^d N V_d(1)},$$

where N is the sample size and $V_d(1)$ is the volume of a d-dimensional sphere with radius one. This formula fits within the kernel-based framework, mentioned earlier in this section, with an adaptive window width. The window width depends upon the local density. If the local density increases, then the window width decreases. Adaptive window widths become important as the dimension of the data increases. The k-NN estimator is seen to outperform the fixed kernel estimator when the dimension is larger than or equal to five [12].

The k-NN density estimator requires a method to find the k^{th}nearest neighbor. To compute the k^{th} nearest neighbor, we use an advanced data structure, viz., the kd-trees introduced by Friedman et al. [3, 4] and Bentley [1]. We use these trees because a brute-force computation of the nearest neighbors in a data set of size N will require $N - 1$ distance computations, while by means of kd-trees the same computation will only take a number of steps proportional to log$N$. The kd-tree is a multi-dimensional extension of the binary tree, where each level of the tree uses a different dimension to obtain the discriminator. Figure 9 shows an example of a kd-tree for a two-dimensional dataset. On the left we see the set of data-points, given by the circles and the partitioning hyperplanes, given as lines. Hyperplane 1 partitions the data-set in two sub-sets of equal size along the x-axis. These subsets are partitioned by the hyperplanes 2_a and 2_b along the y-axis. The third set of hyperplanes will partition the resulting 4 sub-sets along the x-axis again. On the right side of Figure 9 we see the corresponding tree. In each node, the location of the hyperplane is stored. Given this kd-tree we can check in a number of steps proportional to log$N$ whether a point is in the data-set, where N is the number of data-points. To perform this check one starts at the root of the tree, and compute on which side of hyperplane 1 the point is located. If the x-value of the point is smaller or equal to the value of hyper-plane 1, then we know the point is located in the left branch, otherwise it is located in the right branch. Assuming the point is in the right branch we go one level down in the tree and compare y-value of the point against hyperplane 2_b. As the depth of the tree is $O(\log N)$, a leaf is reached in a number of steps proportional to log$N$. If the point is in the tree, then it is equal to the point located in this leaf.

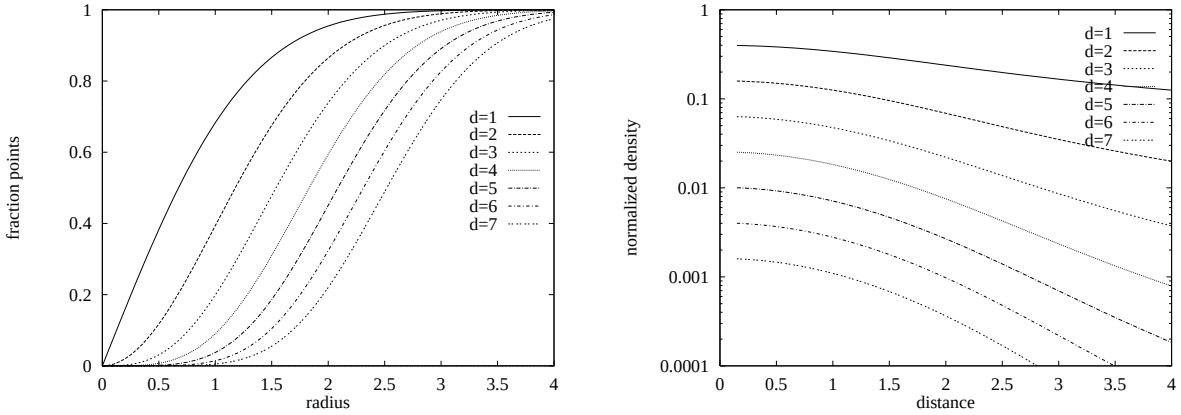

Figure 10: Fraction of points in sphere($\vec{\mu}, r$) (left) and average uniform density over this sphere (right) for different numbers of dimensions d.

Construction of the kd-tree takes $O(N \log N)$ time, where N is the size of the data set. A query for the k^{th}nearest neighbor takes $O(\log N)$ time. For our application we used the optimized kd-tree [4]. In these trees the discriminating dimension is computed for each node separately, so different nodes at the same level can use a different discriminator, this contrary to the standard kd-tree. The optimized kd-tree performs better in practice, but the theoretical time-complexities for building the tree and searching an element are the same for both types of kd-trees.

6. Curse of dimensionality for density estimation

In this section we investigate the requirements for a cluster to be detectable by means of k-NN density estimation method. We first look at the estimated density of the center of a cluster, and next we take a look at a two cluster problem, to investigate conditions under which a two classes are separable.

First, a one-class problem is studied. Let the pixels of the class be distribution according to the Gaussian distribution with variance σ, denoted by $G(x, \mu, \sigma)$. This corresponds to a spherical symmetric distribution, so the probability of finding a pixel with spectral vector $\vec{s}$ is only dependent upon the distance $|\vec{s} - \vec{\mu}|$, where $\vec{\mu}$ is the noise-less spectral vector of the pure ground cover this class corresponds to. We compute the distance to the k^{th}nearest neighbor in a d-dimensional space. To do so, we need the volume of a d-dimensional hyper-sphere of radius r, which is given by the formula $V_d(r) = V_d(1)r^d$. Here $V_d(1)$ is the volume of the unit-sphere in d dimensions, which is given by the formula

$$V_d(1) = \frac{\pi^{\frac{d}{2}}}{\Gamma[\frac{d}{2} + 1]}$$

where:

$$\Gamma[n] = \int_0^\infty x^{n-1} e^{-x} \mathrm{d}x.$$

Without loss of generality, assume that $\sigma = 1$, and $\vec{\mu} = \vec{0}$. So, the plots correspond to the scaled distance, where distances are scaled with respect to the actual σ. The fraction of pixels that fall within a sphere of radius r around the pure spectral vector $\vec{\mu}$ is given by the formula

$$c_d(r) = \frac{\int_0^r G(x, 0, 1) \frac{\partial V_d(x)}{\partial x} \mathrm{d}x}{\int_0^\infty G(x, 0, 1) \frac{\partial V_d(x)}{\partial x} \mathrm{d}x}$$

The left graph of Figure 10 shows the curves $c_d(r)$ for dimensions ranging from one to seven. This

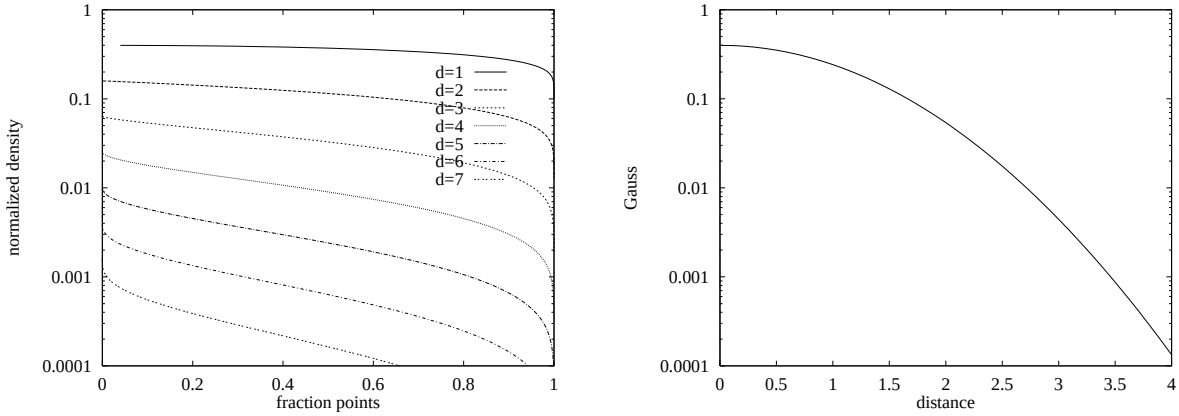

Figure 11: Density as a function of the fraction of points in the k-NN sphere for different numbers of dimensions d (left), and the function density as function of the distance $G(r, 0, 1)$ (right).

graph shows the curse of dimensionality. In a one-dimensional space, 50% of the points are located within a sphere with radius 0.68. In seven dimension only 0.01% of the points is such a sphere, and a sphere of a radius of 2.6 is required to cover more than 50% of the points. So, if the dimension increases, then the densities get lower and a larger fraction of the points is located in the tail of the distribution. We can also compute the average uniform density over sphere$(\vec{\mu}, r)$ by the formula $\hat{f}_d(r) = \frac{c_d(r)}{V_d(r)}$. The right graph of Figure 10 shows this density as a function of the radius of this sphere. These densities have been normalized, such that the integral over the densities is one. So, if a class covers only a fraction α of the image, then these normalized densities should be multiplied by α, to get the actual density.

The k-NN density estimator uses the uniform density over a k-NN sphere as a density estimate. Let a class contains n sample points, and k be given. Now the k-NN sphere around the center of the class contains a fraction k/n of the sample points that belong to this class. Using the above results, the estimated density over this k-NN sphere is computed. The results are shown in the left graph of Figure 11. Note that the y-axis is logarithmic. The density decreases rapidly as the dimension increases, so as the dimension increases, it becomes more difficult to detect a class. Furthermore, the curves show a downward slope. This effect gets stronger as the dimension gets higher. Classes with relatively few samples, require a large k-NN sphere. Therefore such classes are more difficult to detect in high-dimensional spaces. For example, let use assume that we have two classes. The sample contains $10k$ points that belong to the first class and $5k$ points that belong to the second class, so the fraction of samples in the k-NN sphere is respectively 0.1 and 0.2. Now, graph in Figure 11 is used to find the normalized densities of the peaks of the clusters that belong to these two classes. In case of a one-dimensional data-set the normalized peak-density is 0.398 for the first class and 0.395 for the second class. So a one-dimensional space, the two peak-densities of the corresponding clusters differ approximately by a factor two. In case of the seven-dimensional space the peak-densities are 0.0056 and 0.00039. These densities differ a lot, and therefore the class with fewer sample points is more difficult to detect, and much more likely to disappear in the background noise.

Next, we take a look at the separability of two classes. The right graph of Figure 11 shows the normalized density as a function of the distance to the center of a cluster, denoted by $G(r, 0, 1)$. Let us take two clusters, where the first cluster contains n sample-points, and the second cluster contains αn sample-points, for $0 < \alpha \ll 1$. The left graph of Figure 11 can be used to find an approximation of the normalized density at the center of the second cluster. Multiplying the obtained value by α gives the relative density with respect to the first cluster. Now given this density, the right graph of Figure 11 is used to determine the distance with respect to the center of the first cluster, where

the tail-density of the first cluster corresponds to the peak-density of the second cluster. To give a concrete example. Let us image a 7-dimensional space where the first cluster contains $10k$ points, and the second cluster contains $5k$ points. Using these values α becomes 0.5. Now using the left graph we find that the normalized density at the center of the second cluster is approximately $4 \cdot 10^{-4}$. We have to multiply this value by α to get the relative density when comparing to the first cluster, so the value becomes $2 \cdot 10^{-4}$. Using the right graph it can be seen that the density of the first cluster is $2 \cdot 10^{-4}$ at a scaled distance of 3.9 from the center of this cluster. Doing the same exercise for the 1-dimensional case shows that the comparable density is then attained at a scaled distance of approximately 1.2.

Density estimation becomes more difficult as the number of dimensions increases. Given a set of points drawn according to a Gaussian distribution, an increasing fraction of the points is located in the tails of the distribution as the dimension of the space increases. As a result densities decrease rapidly. When using a k-NN density estimator, a uniform distribution over a k-NN sphere is assumed. As a result the peak-density of small clusters will be underestimated, because the the k points in the k-NN sphere form a significant fraction of all points in case of a small cluster. The underestimation of peak-density for small classes gets more severe as the dimension increases. As a result, small clusters are more likely to disappear in the tail one of the larger clusters.

7. Density based sampling for Remote Sensing

In section 6 we investigated the influence of the dimensionality of the data-set on the detectability of clusters. Here we investigate the consequences in case of a remote sensing application. We show that the number of sample-points belonging to the different classes are likely to differ strongly. As a result the classes that cover only a small part of the image are difficult to detect, as the densities of the peaks of the corresponding clusters will be relatively low. Next, a variant of the biased sampling procedure is introduced. This sampling method takes relatively many samples of classes covering only a small part of the image.

7.1 Density estimation in Remote Sensing

By means of a simple example, we show that the number of pixels assigned to different types of ground cover, are likely to differ strongly. Let us imagine an image consisting of 1000×1000 pixels, that is mainly covered with green grass. The image also contains a grey road with a width of 3 pixels and 10 red houses, each having dimension 10×10 pixels. If we compute the number of pixels that are assigned to each type of ground cover, the following figures are obtained. The road covers 0.3%, the houses cover 0.1%, and the grass covers the remaining 99.6% of the image. In many applications of remote sensing, it is important to detect such structures that cover only a very small part of the complete image.

Given the image described above, the human eye can easily discover the structure. In a split second, we observe groups of pixels that are close together and have roughly the same color, even in the presence of noise. So, the usage of the local spatial structure in the image seems to be the key to the human visual recognition. If we map pixels to the spectral space, then the spatial information gets lost. In section 6, it was noted that detection of small clusters gets increasingly more difficult as the dimension of the space increases. Our method does a spectral clustering on a sample. During the sampling step, spatial information is available. Next, we show a sampling method that uses this spatial information to get a sample containing relatively many points that belong to the clusters that cover only a small part of the image.

7.2 Using local density estimates during sampling

We introduce a sampling method that incorporates spatial information. To do so, the method makes a comparison of local and global density estimates. Pixels are selected based on the ratio between the local and the global density estimates. Our goal is to get a sample that contains a more even distribution of points over all classes. This means that pixels belonging to the classes that cover a small part of the image, should get a relatively high probability of being selected.

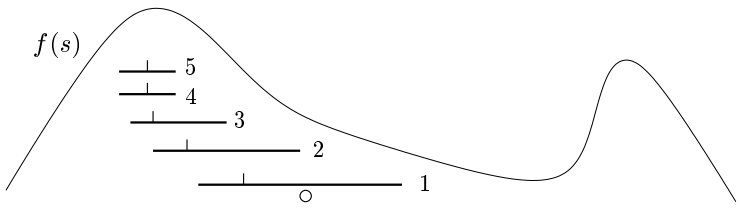

Figure 12: Search for high density region by iterated taking of k-neighborhoods around median points.

To get the local density estimates, we use small patches from the complete image. Within such a patch the diversity is much smaller than in the complete image. This is a result of the limited number of pixels in the patch, but the effect is magnified by the local spatial homogeneity. As a result, it is relatively easy to discover the different classes in such a small patch. Furthermore, the ground-covers present in such a small patch, tend to cover a relatively large part of the patch. In the example image mentioned above, if a patch contains part of a house, then the proportion of the patch filled by the house is likely to be much larger than the overall 0.1%. So, the local proportion of houses is likely to be much larger than the global proportion of image, which was 0.1%. To get the global density estimates the method first draws a random sample S_g from the image. This sample will contain a high density in regions corresponding to spectral features of the most important ground covers in the image.

We select the learning sample by means of the following procedure. To select a single point we extract a $l \times l$ patch of pixels from the image. From this patch, a pixel is selected at random. Given the spectral vector of this pixel, the nearest peak in the density landscape has to be located. We need to find this peak in order to get reliable comparisons of the local and global density estimates. This peak is likely to correspond to a relatively noise-less pixels containing only a single ground cover. So, locating this peak results in a reduction of noise. Furthermore, we need to locate this peak in order to get a reliable estimates of the local and global density estimates. The location of this peak is detemined by the following procedure. The k_l neighborhood of the point in the spectral space is taken, and the median point of the k_l pixels in this neighborhood is computed. A median point is determined by computing the median value for each dimension, so this median point does not have to correspond with a real point in the data-set. Next, the k_l neighborhood of this median point is computed, resulting in a new median point. This process is continued until a k_l-neighborhood is obtained, containing only contains points that where visited already. The median point now is likely to correspond to a local maximum in the density landscape. Figure 12 gives a graphical example of this procedure. The curve represents the density in spectral space. The circle denotes the location of the random starting point int the spectral space, and the numbered lines below the figure correspond to the subsequent k_l neigborhoods that are computed. The vertical marker on each line denotes the location of the median point of the sample. These median points are likely to be located on the side of the neighborhood that corresponds to the highest density. The fifth neighborhood does not contain any new points, and thus the iterated search is terminated. Now given this median point, we compute the the ratio between the local density, and the global density around this median point, both in spectral space. The local density is computed by means of the k_l^{th}nearest neighbor of the median point over all pixels in the patch, the global density is computed by means of the k_g^{th}nearest neighbor in sample S_g.

Given a $l \times l$ patch this whole procedure of extracting a random pixel, computing the nearest peak, and estimating the ratio between local and global density is repeated β times. The median point with the largest ratio is selected, and used as a sample point. The selected point typically has a ratio in the range 10^2 to 10^4. The whole procedure is embedded within a stratification framework. We stratify the image by covering the image with N non-overlapping rectangular regions, where N is the size of

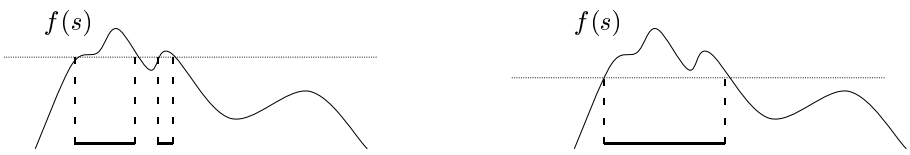

Figure 13: Density based hierarchical clustering

the learning sample. Each rectangular region provides one $l \times l$ patch, and therefore provides a single sample point.

A pixel belonging to a ground cover that covers a small part of the image has a small global density. Therefore, taking the ratio between the local and the global density is large. For a ground cover that covers large part of the image the global density is high, and therefore the ratio gets smaller. Noise reduction is obtained by searching for the peaks in the density landscape. It is possible that we select a noisy point that is located within a transitively closed neighborhood. The median point now is close to this point. Probably the the local density is roughly equal to its global density and therefore the ratio is relatively small. As the typical ratios are larger than 10^2, such a noisy pixel is still unlikely to have the largest ratio amongst the β pixels that are selected from the patch.

8. Hierarchical clustering

In this section a detailed description is given of the hierarchical clustering method we developed. This method takes a set of points as an input and produces a set of clusters. Simultaneously, it computes a measure for the separability of all the clusters. We start with a rather intuitive example, to sketch the basic approach of the clustering method. Next, we are going to describe the method more in detail.

8.1 Water-level model

The operation of the hierarchical clustering method can be though of as method that counts the number of islands when the water-level of the lake is decreasing. If the water-level drops, then a new peak that surfaces increases the number of islands by one. It can also happen that the region inbetween two islands gets dry when the water-level drops. In that case the number of islands decreases by one. A graphical representation is given in Figure 13. The left side of this figure shows a density curve over a one-dimensional space. This density curve has three local maxima. The left peak corresponds to the highest density. The horizontal line in Figure 13 represents the decreasing threshold, used by the method. On the left of the figure the threshold is still high. Each connected region above the threshold results in two separate cluster, so in this case we have two clusters, denoted by the solid line-segments below the graph. On the right side of Figure 13, the threshold is lowered. The density in the region inbetween the two clusters is above the threshold. As a result the two clusters have been merged into a single cluster. If we lower the threshold even further, then the points corresponding to the third peak will be detected. In practice we do not known the density function of a data-set, but by means of density estimation methods we can approximate the density function.

8.2 Hierarchical clustering method

After this brief outline of the approach given in the previous subsection, we now give a detailed description of the algorithm. The algorithm keeps track of three lists. The first list contains all data-points, the second list contains all clusters, and the third list contains delayed cluster merge operations. Initially, the last two lists are empty. For each of the points the local density is estimated, and the list of points is ordered on decreasing density. So, a position closer to the start of this list

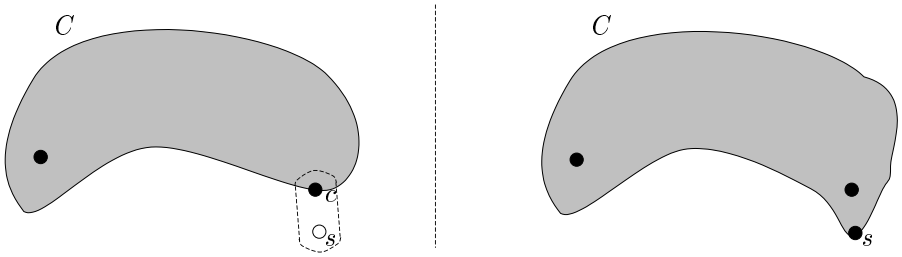

Figure 14: Extending a cluster by one point

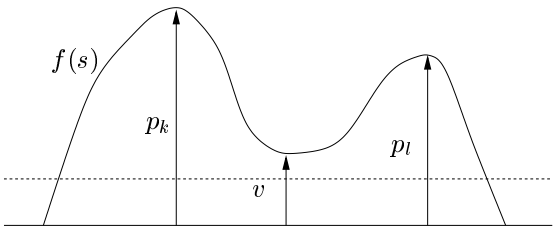

Figure 15: Separability of clusters

corresponds to a higher density estimate. If the data is distributed according to the density given in Figure 13, then the sample-points close to center of the left cluster will be at the start of this list. If we process a sample-point s, then we set the current threshold equal to its local density estimate. Before actually processing the point, we first process all delayed merge operations that have to be performed at the current threshold. Once an operation is performed, it is removed from the merge operation list. Next, we generate a new cluster containing sample-point s only, and we add this new cluster to the list of clusters. Now, we have to determine when this new cluster can be merged with any of the other clusters. The computation for a single cluster C is shown in Figure 14. The new cluster consist of a single point denoted by the small circle. The cluster C is denoted by the grey region. The density at which this merge operation can be performed is computed as follows. The nearest neighbor of s in C is located, let us denote this point by c. Now the joint density of s and c is taken as an estimate for the density at which the cluster containing s is merged with C. This density is computed by constructing the cylindrical envelope of the k-NN neighborhoods of point s and c. The cylindrical envelope is a cylinder with rounded sides. The shape resembles the shape of a pill. A detailed description of the computation of the volume is given later in this section. Given the volume of this cylindrical envelope and the fact that it contains at least $2k$ points, we can compute the density over this volume. Using this approach, the separability of two clusters is determined by the density of the densest connection between these two clusters. When using this approach, the algorithm will be sensitive to the minimal allowed density during the clustering process. If this density is too low, all clusters are likely to be merged in a single cluster, if this density is too high, then low density clusters, corresponding to classes of ground cover that are relatively seldom in the image, will not be detected. If the dimension of the spectral space increases, its sensitivity with respect to the minimal density will increase. Therefore, we introduce a second measure for the separability of two clusters, and use this measure to put an additional restriction on the merge of two clusters. The new measure is the separation measure, given by the formula

$$\frac{v}{\min\{p_k, p_l\}},$$

where p_i is the maximum of the density in cluster i, and v is the density inbetween the two clusters. This measure determines the density ratio between the peak density of the low-density cluster, and the density inbetween the clusters. An example is given in Figure 15. This figure shows a density function over a one dimensional search space. The dashed line denotes the current *densityThreshold*. The solid horizontal line denotes density 0. The value of the separation measure is between zero and one. A value close to one means that the low-density cluster has a density that is close to the density of the region inbetween the clusters. As the densities we are working with are estimated densities, we can merge the two clusters in this case. If the value is close to zero, then the low-density peak has a significant higher density than the region inbetween the two clusters, and it is likely that the low-density cluster corresponds to another class. Thus, no merge is performed.

8.3 Pseudo code of the hierarchical clustering method
In the full description of the algorithm the following data-types are used.

point	$p = [x, radius]$;
cluster	$c = [\{p_j\}, topDensity]$;
mergeOperation	$m = [x_k, x_l, density]$;

The point p consist of a spectral vector denoted by x and the radius of the sphere, centered at x, that contains exactly $k-1$ other points. A cluster c consists of a list of points denoted by $\{p_j\}$ and the highest density within this cluster *topDensity*. A mergeOperation m_i consists of the two points that can be merged x_k and x_l, and the *density*. The *density* determines when the merge operations has to be performed. A merge means that the clusters containing x_k and x_l are combined into a single cluster.

The complete algorithm is as follows:

hierarchicalClustering($\{x_i\}$, *minimalDensity*, *separation*)
points : list of point;
clusters : list of cluster;
mergeOperations : list of mergeOperation;

compute the nearest neighbors of all points
for $j = 1$ to n do
    $x_k = k$-NearestNeighbor(x_j, $\{x_i\}$);
    $radius = d(x_j, x_k)$;
    if (sphereDensity($radius, k$) $\geq$ *minimalDensity*) then
        add $[x_j, radius]$ to list *points*;
    fi
od

sort all point on decreasing density (increasing *radius*)
so the first point of the list has the highest density (after sorting)
$sort_{radius}(points)$;

$clusters = \{\}$;
$mergeOperations = \{\}$;

process all elements p_j of list *points* in sequence
for $j = 1$ to n do
    $densityThreshold$ = sphereDensity($p_j.radius$, k);

```
        ## process all operations above the threshold
        processOperations(mergeOperations, densityThreshold, separation);

        ## create a new cluster containing point p_j only
        c_j = newCluster(p_j.x, densityThreshold);
        add [p_j.x, densityThreshold] to list clusters;

        ## compute all possible merge operations
        ## and store them in the list of mergeOperations
        for k = 1 to |clusters| do
            p_l = findNearestPoint(c_k, p_j, minimalDensity);
            density = jointDensity(p_l, p_j, 2k);
            minDensity = min {threshold, density };
            if minDensity > minimalDensity then
                add [x_l, x_k, minDensity] to list mergeOperations;
            fi
        od
    od

    ## process all operations above the threshold
    processOperations(mergeOperations, densityThreshold, separation);
end
```

The function processOperations(*mergeOperations*, *densityThreshold*, *separation*) processes all operations in list *mergeOperations*, that have a density larger than *densityThreshold*. Here the *separation* determines a lower bound on the separability of the clusters, as shown in Figure 15. The list *mergeOperations* is used to store merge operations that can not be applied yet, as their density is below the *densityThreshold*. This list is implemented as an priority-queue such that the element with the highest density can be extracted in a number of steps proportional to $mboxlog\, N$) [9].

The function findNearestPoint($c_k, p_j, minimalDensity$) searches for the point in cluster c_k that is closest to the point p_j. The additional parameter *minimalDensity* is used to increase the efficiency of the search, as one only has to look for points that are close enough to the point p_j. Close enough here means that the joint density is higher than *minimalDensity*. Using the upper bound on the distance, the search can be limited to a subset of all points by means of a variant of the kd-tree algorithm.

To determine when to merge two clusters, we have to estimate the density in the region between the two clusters. We estimate this density by computing the joint density of a point and its nearest neighbor within another cluster. The joint density of two points is computed by constructing a cylindrical envelope, described earlier in this section, that encloses the k^{th}nearest neighborhoods of both points. Figure 16 shows the construction of the cylindrical envelope for a point y and its nearest neighbor in a cluster denoted by point x. The volume of a cylinder is $V_{d-1}(1)lR^{d-1}$, where l is the length, R is the radius, and d is the dimension of the space. $V_m(1)$ denotes the volume of the unit sphere in a m-dimensional space. The jointDensity(p_k, p_l, m) is computed by the following program.

```
jointDensity(p_k, p_l, m)
    l = d(p_k.x, p_l.x) + p_k.radius + p_l.radius;
    R = max{l/aspectRatio, p_l.radius};
    if l < 2R then
        V = V_d(R);
    else
        V = V_d(R) + (l − 2R)V_{d−1}(1)R^{d−1};
    fi
    jointDensity = m/V;
```

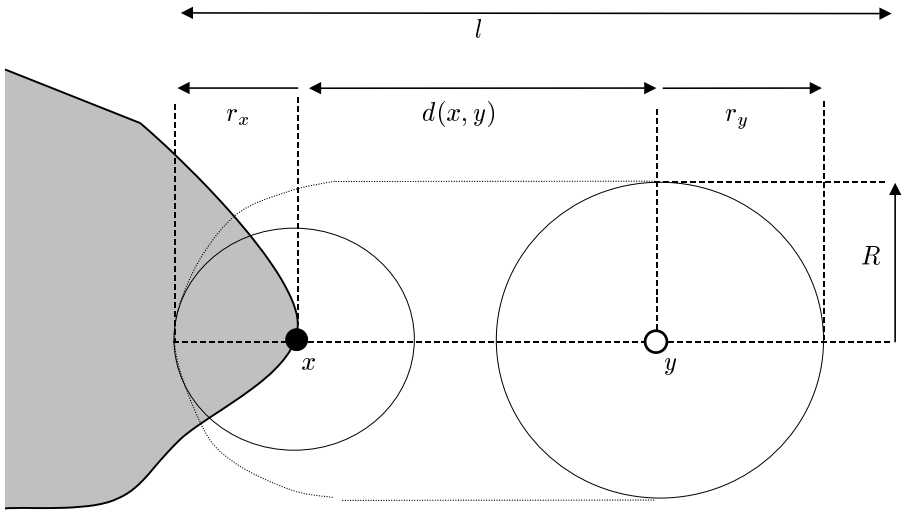

Figure 16: Estimating the combined density around two points by the construction of their cylindrical envelope.

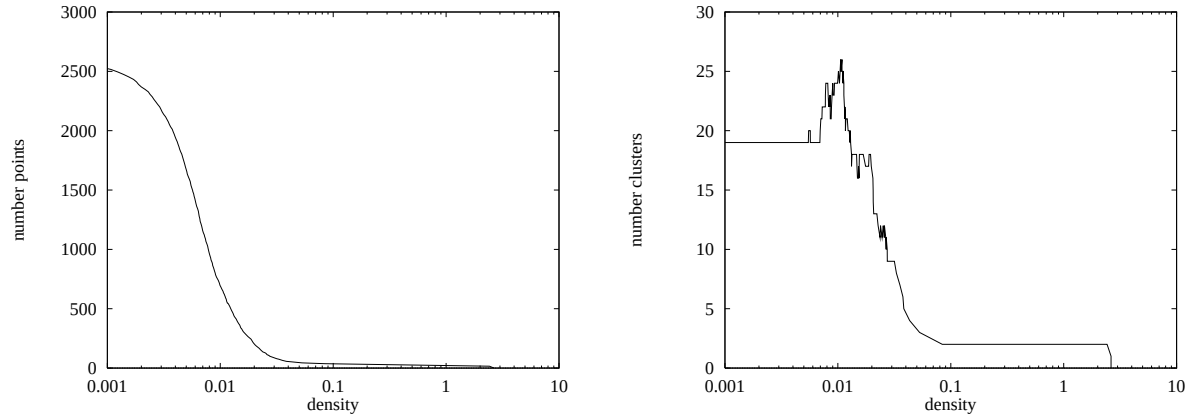

Figure 17: The number of points classified (left) and the number of clusters (right) as a function of the density

end

where the *aspectRatio* is used to put a lower bound on the ratio l/R. This bound is needed to take care the the joint density scales proportional to l^d. We use $aspectRatio = 4$, which corresponds to the ratio of a cylinder containing two toughing spheres of equal radius.

We also state that the theoretical estimated time complexity can be quantified as $\leq d \cdot (NC\log N)$ steps for some constant d, where N is the number of data-points and C is the maximal number of clusters that exist simultaneously.

When the hierarchicalClustering methods is applied to a data-set, it produces an output which is shown in Figure 17. The left graph shows the number of points that have been classified as a function of the density. The right graph shows the actual number of clusters obtained as a function of the density. If we choose a new density threshold that is larger than the current *minimalDensity*, then the corresponding classification can be generated almost instantly, based on an internal data-structure generated by the hierarchicalClustering method. This in contrast to most other clustering methods,

that require that one determines the number of clusters beforehand, and where a change in the number of clusters requires a new run of the method.

9. RS application

The hierarchical clustering method produces a set of classes. A pixel-classification method is needed to assign all pixels to a class. The clustering method can produce classes that correspond to non-convex regions in the spectral space. Therefore, the pixel-classification method should be non-parametric. We use a nearest-neighbor classifier. A pixel is classified by finding the (spectral) nearest pixel in the learning-sample, and assigning the class of this pixel. We use a kd-tree to find the nearest neighbor of a pixel in a number of steps proportional to $\log N$, where N is the size of the learning sample. As this operation has to be repeated for each pixel, this pixel classification method turns out to be slow on large data-sets, where only a small fraction of the pixels is in the learning sample. As an alternative we implemented a method that does a principal component analysis, by means of singular value decomposition [7], on each of the clusters. Next each cluster is reduced to a line-segment. The direction of the line-segment is determined by the direction of its primary principal component, the center of the line-segment is determined by the center of the cluster, and the length of the line-segment is set equal to two times the standard-deviation along the primary principal component [7]. Now a pixel is classified by mapping it on the line-segment of each of the clusters and selecting the cluster that corresponds to the nearest line-segment. The position of the mapping of the point on the line-segment, is used as luminance-value in the output of the image. The time needed for the classification of a pixel now is proportional to the number of clusters.

We have applied the methods presented in this paper to remote sensing data. We have tested the method on a high-resolution three-band aerial photograph of 500×538 pixels and on 7-band Landsat scene with 960×1130 pixels. In case of the Landsat images the sixth band, which corresponds to thermal emission, was removed from the data-set, following [10]. In both cases we used a sample of 4000 points for learning. The tests were performed on a SUN workstation running at 180 MHz. The sampling method used during this experiment involves the computation of local and global density estimates. In case of the three-band image the generation of the biased sample took approximately 33 seconds, and a classification containing 13 clusters was obtained in approximately 16 seconds. In case of the six-band image the generation of the biased sample took approximately 95 second, and a classification containing 26 clusters was obtained in approximately 59 seconds. When using a sample containing 11,000 points for the 7-band image the sampling step takes 460 seconds, and the clustering step takes 421 seconds.

In case of the 7-band Landsat scene we also had a map, showing the results of a supervised classification of the land usage of part of this region. The resolution of the map and the Landsat scene were different, and geometric corrections were applied to the map. Therefore, we can only give a qualitative comparison between the map and the classification obtained by our tool. The types of ground-usage shown in the map are agriculture, industry, city, residential, water, and natural vegetation. When comparing the map to our results we observe that our method finds more classes. For example, we observe many different classes in the agriculture region. It is interesting to see that most of the regions found by our method are rectangular regions, that are aligned with the neighboring regions. The shape, orientation, and size of these regions corresponds to the typical plots of land in agriculture regions. It seems that our method is able to discriminate between the different types of agricultural use of the land in this region. Also the water regions, that cover only approximately 0.6% of the total surface, come out clearly when using our method. We also find two classifications for the urban regions, the first class is mainly located near the center of urban regions, while the second class is located more towards the boundaries of the urban areas. This can correspond to the discrimination between city and residential area in the map. The boundaries between city and residential are different in our case, though we can easily imagine that these boundaries are not very well defined, and we see it as a promising result that our method already detects that you have different types of urban area's. There is only little industry in this region, and it seems like the industrial regions are classified as

residential area's in our method. The area's with natural vegetation are split over two classes.

When doing the same analysis for a lower value of the separability parameter we get a more course grain classification. The agriculture regions is less diverse, and the city and residential area are merged in a single class. Natural vegetation is covered by a single class too.

10. Conclusions

We developed a biased sampling method and a hierarchical clustering method. The sampling method exploits spatial information in order to select those pixels that correspond to a single ground cover, and contain relatively little noise. The sampling method has been tested by means of a theoretical model and on real data. In both cases, we observe that the clusters in the data-set are more clearly present when using a biased sample instead of a random sample.

The hierarchical clustering method is a fast, unsupervised clustering method that takes a set of points as an input and produces a set of non-parametric classes describing the input-data. The method is purely data-driven, and therefore the number of clusters obtained is dependent upon this data-set. In fact, the algorithm produces a whole range of clusterings simultaneously, and afterwards a number of clusterings can be extracted almost instantly. Apart from the sample-sizes and neighborhood sizes, the method uses a separability parameter. This parameter determines under what conditions two clusters can be merged into a single cluster, and therefore affects the final number of clusters. This parameter has an intuitive basis in terms of the ratio of the peak-densities of clusters and the density of the ridge connecting the clusters.

Anticipated further work is the development of non-parametric models out of the learning-data by means of radial basis neural networks, the use of evolutionary computation methods to search for models that allow a demixing of clusters consisting of multiple classes, and the usage of a Bayesian approach to exploit the spatial structure during the pixel classifcation. Spatial structure is exploited by computing computing prior probabilities over a spatial neighborhood, and use these to compute posterior pixel classification probabilities.

References

1. J.L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.
2. E. Fix and J.L. Hodges. Nonparametric discrimination: Consistency properties. Technical Report Report Number 4, USAF School of Aviation Medicine, Randolph Field, Texas, 1951.
3. J.H. Friedman, F. Baskett, and L.J. Shustuk. An algorithm for finding nearest neighbors. *IEEE transactions on Computers*, C-24:1000–1006, 1975.
4. J.H. Friedman, J.L. Bentley, and R.A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, 1977.
5. D.J. Hall and G.B. Ball. ISODATA: a novel method of data analysis and pattern classification. Technical report, Stanford Research Institute, Menlo Park CA, 1965.
6. D.J. Hall and D. Khanna. The ISODATA method of computation for relative perception of similarities and differences in complex and real computers. In K. enslein, A. Ralston, and H.S. Wilf, editors, *Statistical methods for Digital Computer 3*, pages 340–373. Wiley, New York, 1977.
7. I.T. Jolliffe. *Principal component analysis.* Springer Series in Statistics. Springer-Verlag New York, 1986.
8. L. Kaufman and P.J. Rousseeuw. *Finding Groups in data, an introduction to cluster analysis.* John Wiley & Sons, Inc., 1990.
9. D.E. Knuth. *The Art of Computer Programming*, volume Sorting and Searching (vol. 3). Addison-Wesley publishing company, 1973.
10. J.A. Richards. *Remote Sensing Digital Image Analysis.* Springer-Verlag, Berlin, 1993.
11. B.D. Ripley. *Pattern Recognition and Neural Networks.* Cambridge University Press, 1996.
12. D.W. Scott. *Multivariate Density Estimation.* Wiley series in probability and Mathematical Statistics. John Wiley & Sons, INC., New York, 1993.
13. B.W. Silverman. *Density Estimation for Statistics and Data Analysis.* Monographs on Statistics and Applied Probability. Chapman and Hall, London, 1986.