WP3 Algorithm development for S2 and S3 (pre-processing and data reduction)
D3.6 Data mining BEAM module

BC
2014-11
GLaSS is funded by the European Commission (FP7)
Grant number 313256

WP3 Algorithm development for S2 and S3 (pre-processing and data reduction)

D3.6 Data mining BEAM module
BC

Due date: 2014-11-30
Submitted: 2014-12-19

Change records

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description</th>
<th>Contributors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>2014-12-09</td>
<td>First version of compiled document</td>
<td>Ana Ruescas, Tonio Fincke, Kerstin Stelzer, Norman Fomferra, Carsten Brockmann</td>
</tr>
<tr>
<td>final</td>
<td>2014-12-19</td>
<td>Second version of document</td>
<td>Ana Ruescas, Annelies Hommerson, Kersti Kangro, Samsa Koponen, Tonio Finke, Kerstin Stelzer</td>
</tr>
</tbody>
</table>

Consortium

<table>
<thead>
<tr>
<th>No</th>
<th>Name</th>
<th>Short Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>WATER INSIGHT BV</td>
<td>WI</td>
</tr>
<tr>
<td>2</td>
<td>SUOMEN YMPARISTOKESKUS</td>
<td>SYKE</td>
</tr>
<tr>
<td>3</td>
<td>EOMAP GmbH &amp; Co.KG</td>
<td>EOMAP</td>
</tr>
<tr>
<td>4</td>
<td>STICHTING VU-VUMC</td>
<td>VU/VUmc</td>
</tr>
<tr>
<td>5</td>
<td>BROCKMANN CONSULT GMBH</td>
<td>BC</td>
</tr>
<tr>
<td>6</td>
<td>CONSIGLIO NAZIONALE DELLE RICERCHE</td>
<td>CNR</td>
</tr>
<tr>
<td>7</td>
<td>TARTU OBSERVATORY</td>
<td>TO</td>
</tr>
<tr>
<td>8</td>
<td>BROCKMANN GEOMATICS SWEDEN AB</td>
<td>BG</td>
</tr>
</tbody>
</table>

Reference

Please refer to this report as: GLaSS Deliverable 3.6, 2014. Global Lakes Sentinel Services, D3.6: Data mining BEAM module, BC. Available via: www.glass-project.eu/downloads
Task objective (from DoW)

The objective of the task is to extend the capabilities of the optical data toolbox BEAM with a large set of data mining functions. The original idea was to use the competences and models of Rapid Miner and integrate them with BEAM, at least those tools that the partners and users required for lake applications. In view of the development of the use cases in WP5, and making an analysis of the requirements, analyzing the programming model of RapidMiner and making the necessary modifications to integrate the tools in BEAM were the steps to be followed.

Update of the Task objectives

This initial idea had to be changed when the license rights of RapidMiner change from open source to commercial and other data mining tools were considered. As part of WP3, task 3.3, the implementation of the Optical Water Classification Tool (OWT) had already started; besides, BC presented two options for discussion:

- Local/desktop server solutions (extension of BEAM): i.e. data mining through prediction tools, like a supervised classification operator. This type of tool would be available to public.
- Server side solutions (extension of the GLaSS core system): tools to query and preselect data based on indexing exclusively for the GLaSS consortium.

A third option was discussed, coming as a proposal from some partners, to create a Time Series Tool to download data and statistics from a certain location (pixel wise) directly from the core system.

The first and third options were finally accepted, and a predicting tool and the time series extraction tool are now in development together with the finalized OWT tool.

Scope of this document

The scope of this document is to report on the developments, functionality and technical details of the prediction tool (supervised classification). The developments are partially based on existing elements already implemented in BEAM, which provides functionality for definition of training data sets based on predefined masks and extracts/calculates basic statistics necessary for the model to run. For a description of the OWT tool please refer to Deliverable 3.3.
# List of abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROI</td>
<td>Region Of Interest</td>
</tr>
<tr>
<td>BEAM</td>
<td>Basic ENVISAT Toolbox for (A)ATSR and MERIS</td>
</tr>
<tr>
<td>WFD</td>
<td>Water Framework Directive</td>
</tr>
<tr>
<td>DoW</td>
<td>Description of Work</td>
</tr>
<tr>
<td>ESA</td>
<td>European Space Agency</td>
</tr>
<tr>
<td>GPT</td>
<td>Graph Processing Tool</td>
</tr>
<tr>
<td>OWT</td>
<td>Optical Classification Tool</td>
</tr>
<tr>
<td>ML</td>
<td>Maximum Likelihood</td>
</tr>
<tr>
<td>PDFS</td>
<td>Probability Density Functions</td>
</tr>
<tr>
<td>MLP</td>
<td>Multi-Layer Perceptron</td>
</tr>
<tr>
<td>PNN</td>
<td>Probabilistic Neural Network</td>
</tr>
</tbody>
</table>

# List of related documents

<table>
<thead>
<tr>
<th>Short</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>D3.3</td>
<td>Optical Preclassification Method</td>
<td>October 2014</td>
</tr>
</tbody>
</table>
# Table of contents

Scope of this document ................................................................................................................. 3  
List of related documents ............................................................................................................... 4  
Table of contents .......................................................................................................................... 5  
1 Introduction ................................................................................................................................... 6  
2 BEAM Prediction Tool ............................................................................................................... 8  
   2.1 Selection of training data ........................................................................................................ 8  
      2.1.1. The Mask Manager ............................................................................................................ 9  
      2.1.2. Pin Manager .......................................................................................................................10  
      2.1.3 Drawing tools - Geometries ...............................................................................................10  
      2.1.4 Magic Wand .......................................................................................................................11  
      2.1.5 Band math - Creating masks by logical band math expression .................................... 12  
      2.1.6 Import vector data .............................................................................................................13  
      2.1.7 Create Masks from tool windows .................................................................................... 13  
   2.2 Definition of valid pixels ......................................................................................................... 13  
   2.3 Models ...................................................................................................................................... 15  
      2.3.1 Supervised classification and its methods .........................................................................15  
      2.3.2 Regression Analysis ..........................................................................................................15  
      2.3.3 Neural networks ................................................................................................................16  
      2.3.4 Unsupervised Classification .............................................................................................16  
   2.4. Accuracy measurements .......................................................................................................16  
3. Implementation ........................................................................................................................... 19  
4. How to apply the Prediction Tool in BEAM ........................................................................... 23  
5. Future Work ............................................................................................................................... 27  
References ....................................................................................................................................... 28  
ANNEX I The Magic Wand Algorithm .......................................................................................... 29
1 Introduction

Data mining is the computational process of discovering patterns in large data sets involving methods at the intersection of artificial intelligence, machine learning, statistics and database systems. The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use. It involves database and data management aspects, data pre-processing, model and inference considerations, interestingness metrics, complexity considerations, post-processing of discovered structures, visualization and online updating. The aim of the data mining task is the automatic or semi-automatic analysis of large quantities of data to extract previously unknown interesting patterns such as unusual records (anomaly detection), dependencies (association rule mining) and groups of data records (cluster analysis).

Related to the cluster analysis, one of the main purposes of satellite remote sensing is to interpret the observed data and classify features. In addition to the approach of photointerpretation, quantitative analysis, which uses computer to label each pixel to particular spectral classes (called classification), is commonly used. There are two broads of classification procedures: supervised classification and unsupervised classification. In the supervised classification the analyst has available sufficient known pixels to generate representative parameters for each class of interest. This step is called training. Once trained, the classifier is then used to attach labels to all the image pixels according to the trained parameters. The most commonly used supervised classification is maximum likelihood classification (MLC), which assumes that each spectral class can be described by a multivariate normal distribution. However, the effectiveness of maximum likelihood classification depends on reasonably accurate estimation of the mean vector \( \mu \) and the covariance matrix for each spectral class data. Another broad of classification is unsupervised classification, which does not require human interaction to acquire the foreknowledge of the classes, and mainly using some clustering algorithm to classify an image data. The unsupervised classification is already available in BEAM through cluster analysis.

Another tool for analyzing the data due to their spectral behavior is the linear spectral unmixing. Using known spectra of pure surfaces as endmembers and calculating how much of each surface is included in the measured spectrum of the pixels enables the unmixing of each pixel. This functionality is also available with BEAM and provides another tool for data analysis.

The concept of a prediction tool - focusing so far in the supervised learning and modeling implemented in BEAM- is shown in Figure 1. The training data consist of a set of training examples (pixels, ROIs, etc.). In supervised learning, each example is a pair consisting of an input object and a desired output value (label). A supervised learning algorithm analyzes the training data and produces an inferred function ("Train model"), which can be used for mapping new examples ("Apply to images"). An optimal scenario will allow for the algorithm to correctly determine the class labels for pixels not in the training data set. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

The steps that a user should follow are:

- Determine the type of training examples. After the classification method is set, the user needs to decide what kind of data is to be used as a training set. In the case of remote sensing, the information is extracted from a pixel or a group of pixels or region of interest.
• Gather the training set (Section 2.1). The training set needs to be representative enough of the set of data used as input. The selection of the pixels to be used for the training can be done in several ways using some of the BEAM tools already available, which all end up in having masked defined as training sets.

• Determine the structure of the learned function and corresponding learning algorithm. In the present case only the supervised classification is available, but in the near future some other functions, like neural nets or regression models will be selectable (Section 2.3).

• The accuracy of the learned function depends strongly on how the input object/mask is represented (Section 2.4). Typically, the input object is transformed into a feature vector, which contains a number of features that are descriptive of the object. In this way the model starts to be trained.

• Complete the design. Run the learning algorithm on the gathered training set. Some supervised learning algorithms require the user to determine certain control parameters. These parameters may be adjusted by optimizing performance on a subset (called a validation set) of the training set, or via cross-validation.

• Evaluate the accuracy of the learned function. After parameter adjustment and learning, the performance of the resulting function should be measured on a test set that is separate from the training set.

• Apply the model to image to obtain the final target product.

*Figure 1. Concept for the supervised classification in BEAM*
2 BEAM Prediction Tool

2.1 Selection of training data

The selection and gather of the training set is the most delicate step of the prediction tool. In the present case the focus is put on the supervised classification, so the terms and concepts related to this method are used here. The training set must be representative of the target classes of interest. The aim of the training is to derive a representative sample of the spectral signatures or other parameters of interest of each class. Thus, the quality of the training datasets is directly influencing the performance of the algorithm and so the results. In the case of the maximum likelihood and other classifiers based on parametric statistics, the accuracy of the standard deviation derived from the training affect the accuracy of the decision rules. Several well-known paper remark the importance of the training stage for achieving reliable classification results (Chen and Stow, 2002; Chuvieco and Congalton, 1988; Foody, 1999).

The training data is usually derived from an image using a priori knowledge of the scene. Several spatial sampling objects are used to select the training data: a single pixel, polygons or blocks of pixels, similar contiguous pixels, pixels following certain arithmetic expressions, etc. (see Figure 2). Some factors to be considered are the number of pixels used, the effect of spatial auto-correlation, intra-image variance, time and labor cost, and the functionality of the processing system (Chen and Stow, 2002). Autocorrelation is a big issue when working with medium resolution images, because it reduces the variance and makes the data less representative.

![Figure 2. Selection of the training sets](image)

In the prediction tool implemented in BEAM, the training data is introduced in the model by creating masks (Figure 3). The image to be classified has to be selected previously and the mask manager opened. Training pixels are saved into masks using different tools for their acquisition.
In data modelling terms, a mask is a product node similar to a band or tie-point grid. It has a unique name and comprises an image (raster data) whose sample data type is Boolean. Each data product may comprise virtually any number of masks.

Not only the mask definitions but also their use in conjunction with a raster data set, such as a band or tie-point grid, is part of the data model stored within the product. A product "remembers" for a certain band or tie-point grid:

- which masks have been switched visible and
- which masks are in use as region of interest.

A number of product formats define a default mask set. E.g. the Envisat MERIS L1 and L2 product types define a mask for each of their quality flags.

Several tools already in place in BEAM-VISAT allow the user to create masks (training classes).

For a future version of the tool, it is planned to enable the users to create the training set based on data from several inputs and external datasets, for instance from several images or using a data table with coordinates of in situ data.

### 2.1.1. The Mask Manager

Masks are managed by the Mask Manager tool window (Figure 4). The manager allows creating new masks, editing mask properties and delete existing masks. It also allows for creating new masks based on logical combinations of existing masks. Furthermore masks may be imported and exported. If an Image View is selected, the manager tool window can also be used to control the visibility for the currently displayed band. Each mask becomes automatically available in the raster data analysis tools, such as the Statistics, Histogram, and Scatter Plot tool windows.

![Figure 3. Input of the training set into the model using the Mask Manager](image-url)
2.1.2. Pin Manager

A pin is a marker for a certain geographical position within a geo-referenced image. The properties of a pin are: the geographical coordinate, a graphical symbol, the name or label and the textual description. Pins can be used to store the pixel information. New pins can be created with the Pin Tool and removed using the delete pin command in the Edit menu or using directly the Pin Manager. Selecting existing pins can be done with the Select Tool. The Pin Manager display all pins stored in the current product within a table and provide some pin related operations. Pins are automatically stored in the Mask Manager as a mask derived form a geometry (a point or groups of points).

2.1.3 Drawing tools - Geometries

Once an image view is opened, new geometries are created by using the various drawing tools provided by VISAT through the Interactions Toolbar:

- **Line**: Press left mouse button for the start point, drag line to end point and release left mouse button.
- **Polyline**: Single-click (press and release) left mouse button for the start point, move line segment and click to add a vertex point, move to end point and double-click to finalize the polyline.
- **Rectangle**: Press left mouse button for the start point, drag line to end point and release left mouse button.
- **Ellipse**: Similar to rectangle; Press left mouse button for the start point, drag line to end point and release left mouse button.
- **Polygon**: Similar to polyline; Single-click (press and release) left mouse button for the start point, move line segment and click to add a vertex point, move to end point and double-click to close the polygon.
Geometries are organized in geometry containers. All geometries within one container belong together and are handled “as one” geometry (concerning statistics, training definition etc.)

The geometries in a geometry container can be directly used as a ROI for raster data analysis. Once a new geometry container has been added to the data product, an associated geometry mask is created by rendering the geometry onto the product's intrinsic raster data grid (Figure 5). The associated mask will always have the same name as the geometry container which created it and can serve as possible mask for the selected band or tie-point grid without any additional user interaction. Once the geometry is created its associated geometry mask can be used as ROI in the various analysis tools.

Figure 5. Tools available for creating training data via geometries

2.1.4 Magic Wand

The purpose of this tool is allowing creating region of interests (ROIs) within images comprising pixels that are similar to a set of manual picked pixels. The similarity takes into account any number of bands or the entire spectra. Different similarity metrics are provided which may operate either on the derivative, integral or the plain spectrum (Figure 7).

The Magic Wand tool is accessible by clicking on the icon on the right panel of the VISAT view window. When the tool is opened, several options can be selected (Figure 7). The first step to make good use of the tool is to open the image of interest and check in the Colour Manipulation window or using the Histogram tool, the distribution of the data. This is important because the Tolerance range should be change from the default (0 – 1) to values closer to the ones in the image. For instance, if the image is in the range of 0 to 256 values, to set the threshold from 0 to 100 would be a good way to start. Afterwards, click on a representative pixel with the right characteristics of the region of interest you want to identify. There are several options in the tool that can be selected depending on the characteristics of the region of interest and pixel features. In the Annex I there is a detail explanation of each one of the options for the spectrum transformation and the inclusion/exclusion test.
2.1.5 Band math - Creating masks by logical band math expression

Masks and thus training sets can be created by defining band math expressions. It is done by using the $f(x)$ button in the mask manager. This opens the band math expression editor (Figure 8), which allows the user to combine all bands, tie point grids, flags and other masks in logical expressions (with C syntax). The source data can originate from all currently open and spatially compatible input products. The data sources can be combined by a number of comparison, arithmetic, logical and binary operators or use them as arguments for mathematical functions.
2.1.6 Import vector data

Another way of creating geometries for mask is using the possibility of import vector data in BEAM: files of comma separated values files, ESRI shapefiles, MERMAID extraction files, SeaBASS data and SeaDAS track data can be imported, and automatically masks are generated.

2.1.7 Create Masks from tool windows

Masks can also be created from tool windows that show plots, e.g. from a histogram. Simply click into the plot, drag the cursor to select an area. This area refers to a subset of pixels and can be used to form a mask.

2.2 Definition of valid pixels

To define valid pixels is a necessary pre-processing step before data extraction and further analysis. This step erases unwanted pixels from the training data set and thus improves the statistics of the model. Unwanted pixels, i.e. not valid pixels, could be pixels affected by clouds, cloud shadow, shallow areas or correspond to any other criteria that excludes pixels that are erroneously interpreted at some step in the selected line of processing, e.g. calibration or atmospheric correction. It will be up to the user to define the appropriate valid pixel expression as the line of processing can vary very much. The valid pixel expression can be specified in the prediction model window, which is further described below (Section 3). An example is given in Figure 9 below, which displays the chlorophyll a concentration (FUB) in Lake Mälaren before and after masking of valid pixels. The valid pixel definition can also reduce the training data set to a certain region of the image by drawing a bounding box.
Table 1 shows some examples of valid pixel expressions for different processing algorithms:

<table>
<thead>
<tr>
<th>Processing algorithm</th>
<th>Chlorophyll band</th>
<th>Valid pixel expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUB with CC L1P pixel identification</td>
<td>algal_2</td>
<td>fneq(algal_2,5.0) and (NOT l1p_flags.CC_LAND AND NOT l1p_flags.CC_MIXEDPIXEL AND NOT l1p_flags.CC_CLOUD AND NOT l1p_flags.CC_CLOUD_SHADOW )</td>
</tr>
<tr>
<td>CC L2W</td>
<td>conc_chl_merged</td>
<td>NOT l2w_flags.INVALID AND NOT l1p_flags.CC_LAND AND NOT l1p_flags.CC_MIXEDPIXEL AND NOT l1p_flags.CC_CLOUD AND NOT l1p_flags.CC_CLOUD_SHADOW AND NOT l1p_flags.CC_CLOUD_BUFFER</td>
</tr>
<tr>
<td>IPF L2</td>
<td>algal_2</td>
<td>l2_flags.WATER AND (l2_flags.PCD_16 OR l2_flags.PCD_17)</td>
</tr>
</tbody>
</table>
2.3 Models

2.3.1 Supervised classification and its methods

The supervised classification is based on the idea that a user can select sample pixels in an image that are representative of specific classes and then direct the image processing software to use these training sites as references for the classification of all other pixels in the image. Training sites (also known as testing sets or input classes) are selected based on the knowledge of the user. The user also designates the number of classes that the image is classified into.

Maximum Likelihood classifier

The most common of the traditional parametric approaches to supervise classification is maximum likelihood (ML) (Richards 2005). The ML classifier provides an approach for classifying remotely sensed data that is relatively easy to understand and undertake. In the ML approach, mean vectors and variance–covariance matrices for each class are generated from the training data, providing estimates of the parameters of multivariate Gaussian probability density functions (pdfs) for all classes. The posterior probabilities of class membership are estimated for each pixel and pixels are then allocated to the class to which they have the greatest probability of membership. If the data approximate a normal distribution, this approach can produce accurate classifications (Benediktsson et al. 1990), as the pdfs are Gaussian. The ML classifier is also relatively simple to use, as the analyst does not have to set any parameters for the classification to run, just select suitable training data and classes. In some classifiers such as the MLP (Multi-Layer Perceptron) and PNN (Probabilistic Neural Network), parameters need to be specified, which can impact on the accuracy of the output classification. However, the ML classifier does have limitations. For example, it assumes the data approximate a multivariate Gaussian distribution, whereas this is often not the case, especially when a class contains a great deal of within-class variation. In addition, the input data layers should not be correlated, otherwise the statistical assumptions made when using the ML classifier are not met and this may result in a reduction in the accuracy of the classification. Although there are limitations with the ML classifier, it outputs posterior probabilities and these outputs may be used directly as estimates of thematic uncertainty (Foody et al. 1992, Brown et al. 2009).

Support Vector Machines

A Support Vector Machine (SVM) is a classifier which employs hyperplanes: During training the goal is to find hyperplanes which are situated between classes, so that they separate features from different classes. To this aim, hyperplanes sometimes lie in higher-dimensional spaces. After a SVM has been trained, features are classified by determining their position in relation to the hyperplanes (Vapnik and Chervonenkis 1974, Cortes and Vapnik 1995).

2.3.2 Regression Analysis

Multiple linear regression is a generalization of linear regression by considering more than one independent variable, and a specific case of general linear models formed by restricting the number of dependent variables to one (Mardia et al. 1979, Rawlings et al 2002). It is used in a wide variety of applications and can be applied to retrieve the dependent variables (geophysical parameter) from a number of input spectral values as independent variables.
2.3.3 Neural networks

Neural networks are based on the biologic model of neurons. Neurons are cells which react to stimuli and can send stimuli themselves to other neurons. This means that a signal coming from the outside is received by one or more neurons, processed in a specific manner and then passed on to other neurons, which again process and output. During training, features are presented to the network repeatedly, causing neurons to change their properties. Neural networks have proven to be a very rich source of inspiration for algorithm developers and today there exist numerous different variations of neural networks which serve for a multitude of different purposes (Bishop and Hinton 1995).

2.3.4 Unsupervised Classification

Unsupervised classification is not based on a training data set and pre-knowledge, but on purely statistic behavior of the input data. The knowledge based interpretation of the derived classification is done afterwards.

Clustering

Like classification, during training clustering algorithms determine the properties of groups of features. These groups are then called clusters. Features of one cluster are with respect to some attribute similar to features of the same cluster but different to features of another cluster. After a cluster has been created, features can be assigned to it.

Unlike classifiers, clustering algorithms create clusters, not classes. Clusters may overlap, so that features might be assigned to more than one cluster. Also, clusters are usually not known before training, so that clustering algorithms can be used to detect structures or patterns in data. After training, labels can be attributed to clusters. This step is optional, however (Bailey 1994).

2.4. Accuracy measurements

To test the accuracy of the thematic uncertainty (probability) prediction, the outputs of the classifiers are compared with validation data. This validation set is taken from the training set prepared by the user, and it is not used for the training itself, but for the validation of the model phase.

The current implementation of the Prediction Tool offers several classifier specific accuracy measures. The first one of these is a confusion matrix. A confusion matrix displays the relationship between the reference data (or ground truth) and the classification result. It is a square matrix with the number of rows and columns corresponding to the number of classes. The table displays which features have been classified onto which class. If there is a perfect fit, all values that do not lie on the main diagonal are 0. Figure 10 shows an example of a confusion matrix. The rows show the features of the reference data, the columns show the classification results.
Further measures are the producer and the user accuracies. Both measures are calculated for each class and can be derived from the confusion matrix. Whereas the producer accuracy is the fraction of correctly classified pixels with regard to all pixels of that ground truth class, that means, it is the accuracy. For each class of ground truth pixels (row), the number of correctly classified pixels is divided by the total number of ground truth or test pixels of that class. Figure 11 shows the accuracy values derived for the confusion matrix above. It can be seen that the land class has a producer accuracy of 0.64, since the result of 19 853 140/31 057 641 features that have been classified correctly (576 474 and 10 628 027 have been classified as water and clouds, respectively). The user accuracy is also known as the reliability of the classification. It is the fraction of correctly classified pixels with regard to all pixels classified as this class in the classified image (in this case only the training data). For instance, for the land class, the reliability is 19 853 140/24 445 075 = 0. 81215.

The overall accuracy is the ratio of correctly classified features against the total sum of features. Finally, the kappa coefficient κ, also called Cohen’s kappa (COHEN 1960), is calculated from the confusion matrix. Its calculation is given in Equation 1. The kappa coefficient is a more robust measure, as it considers agreement by chance.
The Kappa coefficient is given by Error! Reference source not found., where \( c_{i,j} \) stands for the element in the \( i \)-th row and the \( j \)-th column of the confusion Matrix and \( C \) denotes the total number of classified features. The left term in the numerator is 1 if all values lie on the diagonal, i.e., if all features have been classified correctly. The factor on the right side will be the smaller the more the row sums and the column sums for a class differ. Therefore, models where classes tend to classify many features incorrectly will receive a lower kappa coefficient.

\[
\kappa = \frac{\sum_i \frac{c_{i,i}}{C} - \sum_i \sum_j \frac{c_{j,i}}{C} \sum_j \frac{c_{i,j}}{C^2}}{1 - \sum_i \sum_j \frac{c_{j,i}}{C} \sum_j \frac{c_{i,j}}{C^2}}
\]

*Equation 1: Kappa’s coefficient*

The Kappa coefficient is always less than or equal to 1. A value of 1 implies perfect agreement and values less than 1 imply less than perfect agreement. In rare situations, Kappa can be negative. It is rare that we get perfect agreement. Different people have different interpretations as to what is a good level of agreement.

Here is one possible interpretation of Kappa (Altman, 1991):

- Poor agreement = Less than 0.20
- Fair agreement = 0.20 to 0.40
- Moderate agreement = 0.40 to 0.60
- Good agreement = 0.60 to 0.80
- Very good agreement = 0.80 to 1.00
3. Implementation

The implementation of the Prediction Tool is an ongoing task. So far, one operator has been developed in order to ensure that it is possible to create a model and apply it to a product.

In Figure 12 the three main steps of the Prediction Tool are outlined:

- Select & Label pixels
- Train model
- Apply to Images

In the “Select & Label” step, training data is gathered and categorized. Training data are organized via masks. Chapter 2.1 outlines a variety of methods already offered by BEAM to create or derive masks from images. The current version of the tool, allows for the selection of the input bands from the opened product and its masks while the transfer of masks to other products is still limited.

In the remainder of this section, masks will be referred to as training areas and input bands as source dimensions.

The second step is the “Training of a model”. It requires the implementation of the models themselves and to define interfaces for different categories of models. The current version includes the Maximum Likelihood Supervised Classification only. Therefore no dedicated user interface for different methods is available, yet.

The third step is the application of the trained model onto images, which is the actual classification step. This step also requires the definitions of interfaces to account for various categories.

The implementation of the Prediction Tool is complex, especially concerning the output. It creates various outputs: the first type is the model itself, which shall be applied to one or more images and can be saved for later use; the second type of output is the image which is the result of the application of the model. Two tools were implemented to deal with this complexity: the main task of the first tool is to create a model; the second tool has the purpose of applying it. The first tool has been already implemented, while the second tool is currently being developed.

The handling of the model training tool and an overview about its functionalities is presented in Section 4. This section will focus on the software design beneath the user interface, mainly on how the models are organized and how the training is performed. At the core of the program resides the concept of the Model. Figure 12 shows the model design.
At the top, there is the _Model_ interface, which needs to be implemented by all model classes. The methods provided by the _Model_ interface are for information or training purposes. The training procedure will be explained later.

The _Model_ interface is to be extended by interfaces corresponding to model categories. A _Model_ shall not be derived directly from the _Model_ interface, but by one of the category interfaces. Currently, there is one such interface for the Supervised Classification. It extends the _Model_ interface by two methods, which define the output of a classification. There is one implementation of the _SupervisedClassifier_ interface: The _MaximumLikelihoodClassifier_.

![Figure 12. Organization of models](image)
Figure 13 shows how models are accessed within the software. As with Models, there is a dedicated interface ModelPlugin, which needs to be implemented. For each implementation of a model exists an implementation of a ModelPlugin (e.g., for the MaximumLikelihoodClassifier exists a MaximumLikelihoodClassifierPlugin). A ModelPlugin describes what is needed for its instantiation. Instances of a model are created by calling the createModel-method of the ModelPlugin interface. Instances of the ModelPlugin overwrite the method to ensure that only models of the corresponding types are created, i.e., a MaximumLikelihoodClassifierPlugin can only create MaximumLikelihoodClassifiers.

The ModelPlugins are registered in a ModelRegistry. This registry can be called to retrieve information on which models are available - it is also possible to derive only the models of a distinct category – and to get the ModelPlugin for a certain type.

The use of ModelPlugins facilitates the handling of the models, as it regulates the way models are instantiated, thereby, helping to prevent unstable model states. Also, future models might be costly in instantiation, so using the ModelPlugins during the model setup ensures that the user interface responds quickly.
Figure 14. The training cycle

Figure 14 shows how a model is trained. The image supposes that the user has adjusted model specific settings and defined a training data set consisting of source dimensions and training areas. First, these parameters are passed onto the ModelPlugin, so that it will create a model of the corresponding type with the given settings. In the next step, the untrained model is given to a dedicated trainer class along with the training product and an optional valid pixel expression. Using the information provided by the model, the trainer will extract features from the product. Each feature is passed onto the model, where a model-specific training step will be performed. The features are passed individually, as this allows keeping the amount of data required in memory small.

For some models it might be necessary that features are presented multiple times. This is achieved through passes: one pass corresponds to one training cycle over all features. Each model specifies how many passes it requires. After a pass, the model is informed that all features have been presented so that a pass-specific step can be performed. E.g., when an arithmetic mean should be calculated, during each feature step the value of the current feature is added to a sum. At the end of the pass, calling finishPass() will cause this sum of values to be divided by the number of features. The calculated mean could then be used in a consecutive pass. After all passes have been performed, the model is trained and can be applied to a product.

Future implementations of models of different categories will afford other ways of preparation. The setting up of a training data set is likely to be different for models of other categories. This would in turn require category-dependent ways of setting up the user interface. As there is currently only one category, only one set of components that allows the gathering of a training data set for supervised classification has been implemented.
4. How to apply the Prediction Tool in BEAM

In this section there is a concise explanation on how to use the Prediction Tool in BEAM 5-VISAT.

The first step consists on the installation of the corresponding jar files in the modules folder of the current BEAM version (usually denominated beam-5.0/modules). The Prediction Tool will be installed next time that VISAT is opened.

The second step is to read the image that is going to be analyzed in VISAT (File/Open Product). This image will be the source of all input information needed for the training of the model, and it will be later for the product the model is applied to by the model operator. Once the definition of the training data has been done as explained in Section 2, the Prediction Tool can be opened from the View or the Processing/Image Analysis labels or clicking on the icon. The Training Set label is highlighted first, and the user needs to select the masks that contain the training classes (Figure 15).

![Figure 15. Training Set label from the Prediction Tool window. The reference image or source is selected and the corresponding source dimensions show in a list format](image)

The Training Set label has a link to the Mask Manager, from where the classes are selected and transferred into the Training Areas section. A valid pixel expression can be already stated here, in order to select only valid pixels from each of the defined training classes (Figure 16 and Figure 17).
Figure 16. The mask manager view is opened automatically and the masks to be used as training areas can be transferred. Their name, color and description modified manually by the user.

Figure 17. Training areas selected and a valid pixel expression has been introduced for the training set.

Once the training classes have been selected, the Model is chosen and trained. In Figure 18, two selections are available: select the Model Category and Choose Method. The model category refers to the prediction model to be use. Only the Supervised Classification is now active, but some other models will be available in the following version such as Support
Vector Machines, Neural Networks.

The method refers to the statistical and cluster analysis to be done on the training data. Only the Maximum Likelihood is available until now. Once both boxes have been selected, the model can be trained (click on Train Model label).

The validation of the trained model is done by the metric explained in Section 2.4. With the derived statistics and the confusion matrix, the user can decide to go further and apply the model to the full image, or to improve the training set if the results are not sufficient.

Figure 18. Model Training selection from the Prediction Tool window

The last label corresponds to the Model Application (Figure 19): this is the section of the tool that allows to apply the trained model to the input data, and to define the desired output. The output, in this case the supervised classification, can be written in the input file as additional
band, with a desired suffix, or can be saved as an independent product (Target Product, Figure 20). The directory can also be indicated here.

In the next version of the tool, expected by the end of January 2015, the model created and conveniently saved, will have the possibility to be applied to other images of the same sensor in the same processing level (e.g. Rrs) and same bands (number of bands, naming of bands, etc.).

Figure 19. Model Application label from the Prediction Tool window

Figure 20. Results of the supervised classification: on the left RGB composite; on the left classification of the pixels in the image in land, water and clouds
5. Future Work

The present implementation has been performed to train and apply one supervised classification model. However, there are still several issues that need to be attended, like the integration of the predictor into the BEAM Graph Processing Framework. This will facilitate the re-use of trained models and their application to products not used for training. This will also enable the users to run the tool in batch mode.

Furthermore, it is planned to include additional models, also from new categories. Candidates for these categories are clustering, regression or neural networks. These additions would bring with them new affordances for the user interface, as the definitions of the training data set would change. Also, new measures for evaluating the model accuracy will be required.

The next planned step is the implementation of a supervised classification method based on the concatenation of multiple Support Vector Machines (SVMs).

We further plan to show additional measures to estimate the distinctness of training areas. For these purpose, we want to employ Jeffrey-Matusita-Distances (Richards 1999).
References

- COHEN 1960, A coefficient of agreement for nominal scale, Educat Psychol Measure 20: 37-46.
ANNEX I The Magic Wand Algorithm

Let A be the input “spectrum” comprising the sample values of N bands at a selected pixel position

\[ A = (a_0, a_1, a_2, \ldots, a_{N-2}, a_{N-1}) \]

The magic wand tool will create a new mask which includes all pixels whose spectrum is close or similar to a set of spectra selected by the user. The similarity measure used in the pixel inclusion test is computed by three different metrics: Distance, Average and Limits.

Before spectra are compared, they can be transformed using their derivatives or the integrals. It is also possible to normalise the spectra before transformation and comparison.

The normalised spectrum is given by:

\[ A = (1, a_1/a_0, a_2/a_0, \ldots, a_{N-2}/a_0, a_{N-1}/a_0) \]

The transformations are:

- **Integral**: Pixel similarity comparison is performed on the sums of subsequent band values
  \[ B = integral(A) = (b_0, b_1, b_2, \ldots, b_{N-2}, b_{N-1}) \]
  where \( b_0 = a_0, b_i = b_{i-1} + a_i \) with \( i = \{1 \ldots N-1\} \)

- **Identity**: Pixel similarity comparison is performed on the original or normalised band values
  \[ B = identity(A) = A \]

- **Derivative**: Pixel similarity comparison is performed on the differences of subsequent band values
  \[ B = derivative(A) = (b_0, b_1, b_2, \ldots, b_{N-2}, b_{N-1}) \]
  where \( b_i = a_{i+1} - a_i \) with \( i = \{0 \ldots N-2\} \)

The similarity comparison is using to sets of selected spectra. User can put spectra into the

- **Inclusion set** \( B^+ = \{B^+_0, B^+_1, \ldots\} \) or the
- **Exclusion set** \( B^- = \{B^-_0, B^-_1, \ldots\} \).

Given a tolerance \( t \) in units of the transformed spectra and a spectra \( C \) at any pixel position in the image, the metrics used for the similarity results \( s^+ \) and \( s^- \) against \( C \) are:

- **Distance**: Tests if the minimum of Euclidian distances of a pixel to each collected spectrum is below the threshold \( t \):
  \[ s^+ = \Lambda((C - B^+_i) < t) \]
  \[ s^- = \Lambda((C - B^-_i) < t) \]

- **Average**: Tests if the Euclidian distances of a pixel to the band-wise average of all collected spectra is below the threshold \( t \):
  \[ s^+ = \Lambda((C - avg(B^+_i)) < t) \]
  \[ s^- = \Lambda((C - avg(B^-_i)) < t) \]

- **Min-Max**: Tests if a pixel is within the band-wise min/max limits of collected spectra plus/minus the tolerance \( t \):
  \[ s^+ = \Lambda((C \geq min(B^+_i) - t) \land (C \leq max(B^+_i) + t)) \]
  \[ s^- = \Lambda((C \geq min(B^-_i) - t) \land (C \leq max(B^-_i) + t)) \]

Finally, a pixel with spectrum \( C \) is included in the mask if \( s^+ \land \neg s^- \)