Description of the Software System *Constud* 2

Kalle Remm, Madli Linder, Hendrik Proosa

Chair of Cartography and Geoinformatics
Department of Geography
Institute of Ecology and Earth Sciences
University of Tartu

Tartu 2010

1. Introduction

Machine learning and prediction system *Constud* is created for: 1) calculating spatial pattern indices to a database table or to a binary raster file; 2) predicting the nominal and numerical variables to a database table or to a binary raster file; 3) machine learning – search and weighting of features and exemplars needed for the most reliable similarity based predictions.

The system consists of 3 parts: 1) **data layers** of explanatory variables, pre-classifiers and interpolation polygons; 2) **data base(s)**: knowledge base(s) of observations (cases), parameters of machine learning, results of machine learning, and predicted values; 3) the **software application** *Constud*. The first part is the component of the system only when spatial data is used. In the case of non-spatial data, the system consists of two components—knowledge base and the software.

The main differences and advantages of *Constud* compared to traditional methods and packages of statistical modelling are the following.

1. Computation of local statistics automatically from many raster files without the need to know the map sheet number or the need to relaunch the application upon condition that the raster files of data layers are stored in requested format and in certain registered[1] folders.
2. Wide possibilities for choice of local kernel and sample parameters.
3. Possibility to compute local indices using pixel masks, i.e., restricting the calculations with particular polygons.
4. *Constud* allows calculating indices of spatial pattern rarely found in other applications, e.g., *gradient strength*, *stripeness*, and *mode weighted by inverse distance*.
5. The possibility to store the observations, the metadata and parameters of the data layers, the features calculated to the training data, the parameters and results of machine learning, and the predictions in a single knowledge base.
6. Search for the best solution is continuous iterative process. Experience obtained during the process is saved in a knowledge base as values of actuality of features and observations. Features and observations can be added to and excluded from the knowledge base without interrupting the learning process.
7. Hitherto the best set of weights of features and exemplars can anytime be used for map generation.
8. Possibility to teach feature and exemplar weights separately for every class of a multinomial variable.

The software code of *Constud* is written by Kalle Remm. Madli Linder, Hendrik Proosa, and Tiiu Kelviste participated in testing and writing technical documentation. The compiled application is freeware. The source code of *Constud* as a property of the University of Tartu, Estonia and is not distributed. The tested functionality is given in Appendix 1. System files, recent updates, database examples and system description can be found at http://www.geo.ut.ee/CONSTUD/. Error reports and suggestions should be sent to kalle.remm@ut.ee.

The abbreviated descriptions of the system are published in Remm and Linder, 2007; Linder et al., 2008, 2009; Remm and Remm, 2008, 2009, 2010; Remm et al., 2009; Tamm and Remm, 2009.

[1] The data layers and their metadata must be registered in an MS Access format data base.

1.1. Changes in Constud 2

The following changes have been made to the software system Constud beginning with version 25.06.2010.

Short integers (2 bytes) are allowed for: the layer of preclassifying polygons, interpolation polygons, data layers treated as numerical predictors. Byte format is still obligatory for the preclassifier feature of the dependent variable, for data layers treated as nominal, values of predictors (incl. those calculated from integer format data layers). Dependent numerical variable is treated as 4 bytes real.

The fields [precl_k], [precl_layer_k], [radius_k] of the table [EXPL_VAR] are any more not used in the Access format knowledge base. The corresponding parameters are read from the fields [precl], [precl_layer], [radius]. A single format field [EXPL_VAR].[divisor] is added for converting integer format source layer to byte format indices.

Only the compact format of log tables applied since 2008 is allowed. The field indicating substitute features in the table [EXPL_VAR] can only be [substitute] and not any more alternatively [replaced_by].

Prediction fit is calculated relative to total sample not relative to the calculated observations. The results are different if the prediction of the dependent variable of some observations is not possible due to e.g. absence of similar exemplars, a unique preclassifier value or applicable date interval restrictions.

The main dialog window of Constud enables to use a common prefix added to the beginning of all folder names in the table [DATA_LAYERS]. The data layers can be located in a FTP folder and used over internet.

2. Data layers

2.1. Requirements for data layers

In *Constud*, the main utility of the data layers is the use for extracting the local spatial[2] indices from these. In addition, one of the data layers selected by user can be utilised as pre-classifier (see chapter 2.2.) and in generation process of predictive maps the pre-calculated data layers of explanatory variables (*substitute feature layers*) and interpolation polygon layers can be used.

The data layers of spatial data, pre-classifiers and interpolation polygons used in system *Constud* must correspond to the requirements and must be organised as follows.

1. All data layers must have unpacked binary raster format saved by rows and without header (*Idrisi32 rst*-format).
2. Data type of the raster layers should be byte (integer values 0…255) or signed 16-bit (two bytes integers ranging in value from -32768 through 32767).
3. All raster files used must be in the same projection and in the same coordinate system[3].
4. The name of each raster file must contain only digits and the extension *.rst*[4] in its name. The number expressed by digits must not exceed integer format (2147483647).

[2] This is the reason why the data layers are not used in case of non-spatial data.

[3] E.g., in the case of Estonian data the base map projection and cartesian coordinates (Lambert-Est97) have been used to date, and following from this, every raster file represents the 10 × 10 km map sheet.

[4] In the case of Estonian data, the name is comprising the 4-digit number of the 10 × 10 km Estonian base map sheet, e.g., *5434.rst*.

5. The number of rows and columns in all raster must either be equal or both minimum and maximum coordinates of every map sheet must be registered in the table *map_sheets*.
6. The cell size in raster files can be fixed by user, but within the same data layer it has to remain constant.
7. Every data layer must be located in a different subdirectory.
8. The data layers must be registered and their metadata must be fixed in the table *D_Layers* (see chapter 2.1.2) that is related to the machine learning and prediction data base currently used.

2.1.2. Table of data layers

Registration of data layers in table *D_Layers* is required only for spatial data. During calculation of the pattern indices the exact folders of the files, from which the features are calculated, must be indicated. In the case of non-spatial data, the temporal limits of data layers, the table of spatial indices and pre-computed features are not used. The nominality of a feature is determined by the nominality of a data layer.

Required fields and field formats in table *D_Layers* are described in Table 2.1.

Table 2.1. Required fields in table *D_Layers*.

Field name	Format	Comment
KID	2-byte integer	ID of the data layer
p_edge	4-byte float	Pixel edge in the same units as coordinates of locations.
folder	text	Path to the files of this data layer without prefix input through the main dialog window.
nominal	yes/no	*yes* means layer values will be treated as nominal
void	byte	Pixel value that is treated as absence of the data layer
from	date, format depends on op-system settings	Date of the data layer
until	date, format depends on op-system settings	Valid through date of the data layer
bytes	byte	Data format: 1 = byte, 2 = short integer; nominal layer can be in byte format only

2.1.3. Table *Map_sheets*

Table *Map_sheets* stores boundary coordinates of map sheets and helps the user to get the overview of which data layers exist for one or another map sheet (Table 2.2). This table is not obligatory. VBA module *Update_Map-sheets* can be used to record the presence of map sheets (Appendix 2). This module assumes the first character in name of a raster file denotes the number of a map sheet and that the extension of file name is *rst* or *rdc*. Longer file names are cut down during execution of this module.

Table 2.2. Fields in table *Map_sheets*. First three fields are required for the calculation of location indices.

Field name	Format	Comment
NR	4-byte integer	Number of map sheet
min-x	8-byte double	Cartesian coordinate of the western border of the map sheet
min-y	8-byte double	Cartesian coordinate of the southern border of the map sheet
max-x	8-byte double	Cartesian coordinate of the eastern border of the map sheet, needed if option square-shaped map-sheets is deselected
max-y	8-byte double	Cartesian coordinate of the northern border of the map sheet, needed if option square-shaped map-sheets is deselected
<No. of data layer>	yes/no	*yes* means data file for layer exists in path defined in the table *D_Layers*

2.2. Data layers as pre-classifiers

Constud enables to use three types of pre-classifiers: 1) a pre-classifier of the dependent variable, which determines the set of exemplars to be used for predictions, 2) a pre-classifier limiting the extent of the calculation of map and image pattern indices, and 3) interpolation polygons limiting the interpolation during map generation. The first and the last one can limit the area of calculations for predictive and similarity maps. The pre-classifiers can be calculated from raster data like all other spatial indices. The layers of a pre-classifier and the interpolation polygons may be used to delimit the area of map generation and the extent of the calculation of spatial indices (see 2.2, 5.3.2).

When using a pre-classifier of a dependent variable, similar exemplars for every observation are sought only among exemplars of the same pre-class. The pre-classifier of a dependent variable is one of the explanatory variables having its number in the table *DEP_VAR*. The pre-classifier of a dependent variable can be used in machine learning and for calculation of predictions.

The polygon layer to be used in calculation spatial indices is determined by the layer identification number in the table *EXPL_VAR*. When using polygons, only pixels within the same polygon and within kernel radius are used in index calculation. Different combinations of kernel radii and polygon layers can be used in index definitions. For example, pattern indices for field observations are calculated using polygons marking the spatial range of description validity. For map generation, polygons covering the whole map area without gaps are more appropriate.

Subsequently, using the land cover categories on the Estonian 1: 10 000 base map as the pre-classification layer is introduced. The land cover categories can be used as pre-classifiers in calculations of spatial pattern indices at locations and for calculations of predictive maps. In the first case, the base map polygons form spatial boundaries that allow calculating the indices at the observation location only at pixels of the same unit of the preclassifier (spatial preclassifier). In predictions, the most similar training sites are searched only from the subset of the same category as at the predictable location (a thematic pre-classifier). Pre-classification accelerates the map generation process and allows considering the most accurate spatial information, maintaining the base map polygon boundaries in the vegetation map.

2.3. Layers of pre-computed features

When using high-resolution data or large datasets, it is advisable to use formerly pre-computed feature layers to speed up the computation process. These data layers are stored for later use to suppress the need of repetitive feature computation. It is most useful in the case of large raster files (over 10 MB) and computationally intensive spatial indices.

3. Database

All *Constud* data tables must be incorporated in relational databases allowing SQL queries. To date, *MS Access 2000* and *MS Access 2003* format databases have been tested. The order of records is irrelevant for *Constud* in all database objects. Uppercase and lowercase letters are not distinguished in database object names.

3.1. Observations

Database of training observations (cases) contains table(s) comprising values of features for every observation (feature vectors), and tables[5] with the meanings codes.

Required fields in the database object (a table or a query) of observations are: separate fields for all explanatory variables, fields: *VID, from, until*, and in the case of spatial data, coordinate fields *x* and *y* (Table 3.1). A SQL query can be used to select needed fields, rename them and pass their values to *Constud*.

Numerical dependent variable type can be byte, integer, or real; multinomial variables can be represented only by integer values between 0 and 254 (byte format), binomial variables must be in Boolean format. Fields *from* and *until* limit the temporal validity and the use of observations related to that. For example, observations describing the clearings are valid only starting from the date of clear-cutting fixed in the temporal limits.

Table 3.1. Required fields in the database object of observations.

Field name	Format	Comment
VID	4-byte integer	Observation ID
F	4-byte real if *ftype* = 2, yes/no if *ftype* = 3, byte in other cases	Value of the dependent variable
x	8-byte double	Cartesian coordinate x
y	8-byte double	Cartesian coordinate y
from	date, format depends on settings of operating system	Beginning date of the observation validity; the field may be empty
until	date, format depends on settings of operating system settings	End date of the observation validity; the field may be empty
<No. of explanatory variable>	byte or integer depending on data layer	Value of explanatory value

[5] The tables of code meanings are not essential for the system to function but suggested to keep everything as interpretable as possible.

3.2. Machine learning

A database for communications between machine learning in *Constud* and tabular data is a central unit of the system. The name of the database is not fixed — it is selected from *Constud* main window along with data path to its location (Fig. 5.1). The data base consists of database objects or links to the objects given below. The objects having obligatory fixed names are marked with asterisk; other names of objects can be modified.

*D_Layers** — metadata of data layers (description in chapter 2.1.2).
*DEP_VAR** — parameters of predictable dependent variables.
*EXPL_VAR**— parameters of explanatory variables.
*SP_Indices** — table of spatial indices.
LOG_F1, *LOG_F2**, etc. — tables used to store feature weights, optimized value of similarity, leave-one-out cross validation fits in training sample and in validation sample calculated after every learning iteration (chapter 4.3.4, Table 4.9).
Feature vectors referred to on the field *data query* in the table *DEP_VAR*.
*Self_Sim_Tables** — a table containing the names of the tables describing the similarities between categories of nominal variables. Contains fields *AID* — the identificator of an explanatory variable, *sim_table* — the database object containing similarity values of nominal feature categories.
Self-similarity tables — the names of tables describing similarities between categories of nominal variables.
*STDEV** — contains fields *AID* — the identificator of a numerical variable, *precl* — the category of the preclassifier of a numerical variable. Values 0 and 255 mean pre-classification is not used, *SD* — standard deviation of feature in pre-class. This table is necessary only if standard deviations of numerical values are read from table. Otherwise standard deviations will be computed from training data sample referred to in table *DEP_VAR* (field *dataquery*).
The following tables are just one possible way to organize data. *Constud* communicates with the feature vectors linking the tables.
Observations — a database object containing feature vectors of observations.
Results — a database object containing hitherto the best sets of weights and actualities for exemplars; predicted values for observations.

3.2.1. Dependent variables

Learning process is driven through *DEP_VAR* table — application queries it for process tasks before each learning iteration. List and parameters of dependent variables in table *DEP_VAR* must be formatted as described in Table 3.2.

Table 3.2. Required fields in table *DEP_VAR*.

Field name	Format	Comment
calc	yes/no	Whether to calculate this variable or not
FID	byte	ID of a dependent variable
given	byte	The code of the category learned separately from other categories of the multinomial variable
logID	4-byte integer	ID of hitherto the best result corresponding to the one in the respective *LOG* table
precl	byte	ID of the explanatory variable used as a pre-classifier
name	text	A name given to the dependent variable by the user
ftype	byte	The type of dependent variable
no_part_feature	byte	The number of dimensions (partial features) of multidimensional numerical variable
sumsimmax	4-byte real	The initial value for the sum of similarity searched for decision making
dataquery	text	The name of the data query of features of training observations and fields for results
t_sample	4-byte integer	Approximate[6] size of the training sample (needed only for machine learning). Values >2000 are automatically changed to 2000.
no_features	4-byte integer	Approximate number of features used in learning iterations
v_sample	4-byte integer	Approximate size of the validation sample (needed only for machine learning). Values >5000 are automatically changed to 5000.
0_dist	4- byte real	A distance correction factor for reducing the effect of spatial autocorrelation (needed only for machine learning of spatial variables)

Additional specifications on the table of dependent variables follow.

The combination of *given* and *FID* must be unique for the rows where *calc = yes*.

Empty *precl* field or values 0 and 255 mean learning and/or prediction without pre-classification.

Types of dependent variable specified on the field *ftype* are coded as follows: 0 = multinomial, 2 = numerical, 3 = binomial, 4 = one category of a multinomial variable opposed to all other categories of the same variable. Code 1 is reserved for multidimensional numerical variables, the option is hitherto not in functionality. An example of a multidimensional feature is the forest stand formula describing the relative coverage of separate tree species in a forest stand. If *ftype* = 4, a separate row for every class is required in the table *DEP_VAR*.

The sum of similarity (*sumsimmax*) is the total value of similarity between predictable observation and exemplars, after which the process of searching for the next similar exemplar is stopped. The maximum total similarity determines when to stop searching for additional exemplars. The most similar exemplars are sought during prediction of an observation. If the

[6] The size of sample is calculated combining random probabilities and the expected proportion of the sample; it is stored in the respective log-table.

exemplars having the highest similarity do not sum up to the level of *sumsimmax*, the next most similar exemplar is added and so forth. The level of similarity is decreased by 1%. When the summed similarity of the most similar exemplars equals or exceeds the maximum sum of similarity, no more exemplars are added to the set and prediction is calculated. The optimal value of parameter *sumsimmax* depends on the data being used; its initial value is set by the user. Smaller values of *sumsimmax* increase prediction sensitivity and help to separate classes consisting of a small number of observations but increase error rate for classes that include more cases. Greater values tend to lead to generalizations and are more useful for the recognition of numerous classes.

The distance correction parameter (field *0_dist*) regulates the extent of reciprocal prediction between observations at close distance. The spatially close observations tend to be similar due to spatial autocorrelation. For example, pattern indices calculated within radius 30 m are almost identical for observations only 5 m apart. Since spatially close observations predict each-other with great accuracy, the results tend to express deceptively high prediction accuracy and the application of the same set of features and exemplars for more distant observations is not so reliable. To extenuate the effect of autocorrelation, the similarity between observations is decreased in proportion to the inverse distance between observations. The amount of decrease is regulated by the distance correction value. If the distance between observations remains below the correction value, similarity between them is set to zero. If distance equals to double correction value, similarity will be decreased by 50%. As a result, *Constud* prefers exemplars that are more dispersed (Fig. 3.1). If the distance correction factor is 50 m and the similarity between two observations equal to 90% and the observations are 50 m apart, their similarity will be corrected to zero; similarity will be corrected to 40% at distance 100 m, 70% at 250 m apart, and 500 m leads to a corrected similarity value of 80%.

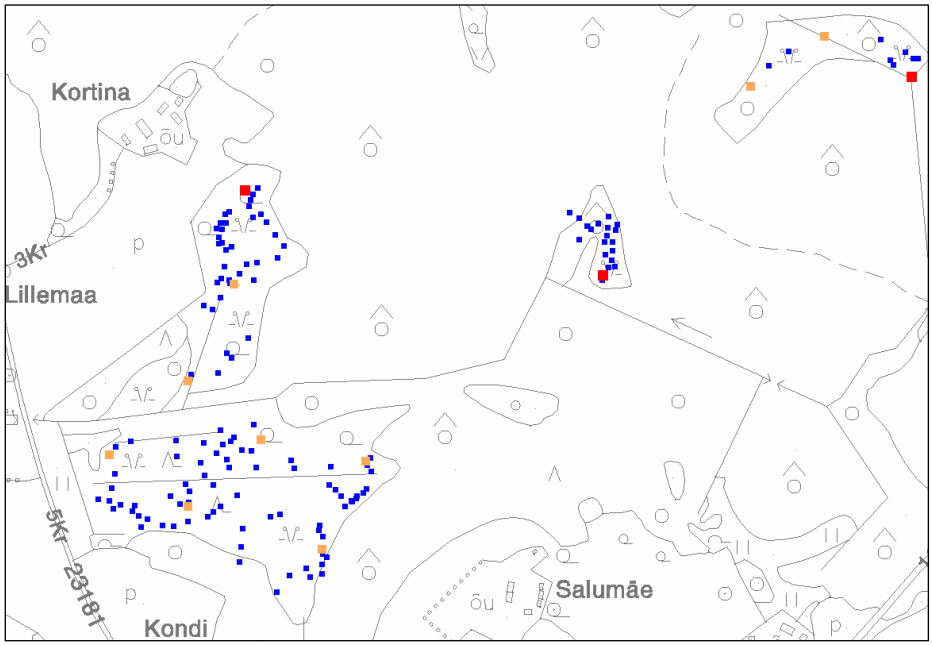

Figure 3.1. Find locations of an orchid *Epipactis palustris* (blue), locations chosen as training cases (rouge) and locations selected to the set of exemplars (red) in a fraction of Otepää Nature Park.

3.2.2. Explanatory variables

Locational features (explanatory variables describing a location and its neighbourhood) are spatial indices calculated from satellite and orthophoto images and from different rasterized map layers. In addition, all other characteristics coded into byte format can be used as features describing observations. The latter possibility is applicable in the case of spatial, as well as in the case of non-spatial data. Features can be easily created or recomputed from existing data layers.

The number of features is virtually unlimited but a large number of descriptive variables slows down iterative weighting and can lead to over-fitting (the validity of predictions only within training data). Therefore, up to approximately 100 features more related to the dependent variable and less inter-correlated should be preselected by the user.

3.2.2.1. Parameters for index calculation

For index calculation, user must define 1) the index, 2) the data layer to be used, 3) the kernel radius, 4) sample proportion (computation time of indices that use pair-wise comparisons increases geometrically depending on the number of pixels in sample), 5) usage of block sample (required for indices that use neighbouring pixels and when using sampling of pixels), 6) a given code if applied for a particular index, and 7) spatial pre-classifier (whether to use a spatial preclassifier layer and the name of the layer). Names and parameters for index calculation must be stored in the table *EXPL_VAR* arranged and formatted as described in Table 3.3.

Table 3.3. Required fields in table *EXPL_VAR*.

Field name	Format	Comment
AID	integer	ID of an explanatory feature
name	text	name of the feature given by the user
KID	byte	ID of the data layer from table *D_Layers*
index_ID	byte	number of the spatial index from table *SP_Indices*
radius	4-byte real	Radius in meters used when the index is calculated to the database table (needed only for the calculation of spatial indices)
in_radius	4-byte real	Internal radius of the annulus kernel (needed only for calculation of spatial indices)
sample	4-byte real	Approximate sample proportion in range 0...1 (not required for machine learning)
blocksample	yes/no	The use of block-sampling (not required for machine learning)
given	byte	given code is needed for indices 1 and 13 (frequency of a category and distance to a category)
precl	yes/no	The use of a pre-classifier to restrict the calculation of spatial index to a certain category (is used when the index or prediction is calculated to a database table)
precl_layer	byte	Number of the nominal data layer in table *D_Layers* used as a pre-classifier (is used when the index or prediction is calculated to a database table). Empty field and 0 means the preclassifier is not applied.
BO_folder	text	Folder of the binary output if the index is calculated to a raster layer (needed only for calculation of indices to a raster layer). Raster output is calculated from all files having extension *rst* in the folder of the data layer of the selected feature referred to in the field *KID*.
substitute	byte	ID of a feature replaced by this feature. Value of the field must be empty or 0 if the feature is not a substitute feature. Substitute features can be used only for spatial data.
divisor	single	Divisor used to to divide the values of indices calculated from an integer format data layer.
F1, F2, F3 …	yes/no	The involvement of features in calculation of indices and/or during machine learning. The field is relevant if the matching *FID* in the table *DEP_VAR* is switched on. *DEP_VAR.F1* directs *Constud* to *EXPL_VAR.F1*. Applied if *ftype* ≠ 4.
AF1, AF2, AF3, …	byte	Actuality of features in range 1…200 in machine learning of the dependent variable *FID* = 1. Applied if *ftype* ≠ 4.
F1_1, F1_2, F1_3, …	yes/no	The application of features for calculation of indices and in machine learning of a category of a nominal dependent variable *FID* = 1 having the marked given code. Applied if *ftype* = 4.
AF1_1, AF1_2, AF1_3, …	byte	Actuality of features in range 1…200 for machine learning of a category of a nominal dependent variable *FID* = 1 having the marked given code. Applied if *ftype* = 4.

It is possible to constrain the calculation of spatial pattern indices with pixels within the same category on a particular map layer, or also by observation plot boundaries. For the computation of predictive maps, an all-cover pre-classification layer (e.g., base map) can be used instead of sparse separate polygons of observation areas. Indices are calculated only for observations not excluded according to the void code of pre-classification layer.

Spatial pattern indices are calculated in *Constud* within circular kernels having a user-defined radius. For features indicating distance to a particular category, calculations should be made without pre-classifier and within a rather large radius. Feature values at the centre of the observation can be calculated using radius = 0.

Block sample enables to use more neighbouring pixels than simple random sample (Fig. 3.2). Block sample means the selection of 3 × 3 pixel blocks (Fig. 3.3). Distance between blocks depends on the proportion of sample — the number of selected pixels matches the predetermined share as precisely as possible.

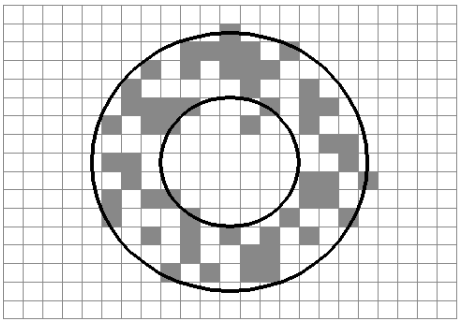

Figure 3.2. Random sample having a half proportion of grey squares in an annulus-kernel (inner radius 3.5, outer radius 7 pixels).

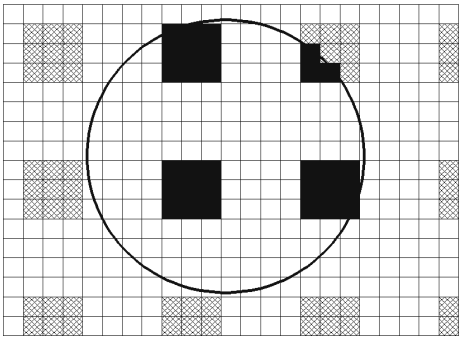

Figure 3.3. A block sample from raster file having proportion of 9/49 (black and striped squares) in a circular kernel (radius 7 pixels) and the squares of a block sample within the kernel (black squares).

Field *F<FID>* as e.g. *F2* or as *F<FID>_<given code>* (e.g. *F2_12*) in the table *EXPL_VAR* is required for every feature used in machine learning or in calculation of spatial indices.

Field *AF<FID>* as e.g. *AF2* or as *AF<FID>_<given code>* (e.g. *AF2_12*) is required for every feature used in machine learning. Automatically stored feature weights and actualities can be manually modified by the user (Chapter. 5.4.2).

The fields *F<FID>* and *AF<FID>* are not used in prediction.

Pre-computed substitute features are used only for map generation and not in machine learning. These features should be disabled in *F*-fields during machine learning. A pre-computed feature must have greater AID value than the matching feature being replaced.

3.2.2.2. Spatial indices in *Constud*

It is possible to calculate 31 different indices in Constud version 27-11-08. The indices are registered in the table *SP_Indices* formatted as described in Table 3.4. All indices are calculated using round kernel having radius (in the same units as Decartian coordinates of observation sites, usually meters) set through table *EXPL_VAR*. Distance values for indices *distance from class* and *distance from class border* are given in pixels of the source data layer used for calculations. The identificator numbers and the usage of indices as a nominal or a numerical variable are given in Table 3.5. The list of indices and descriptions of algorithms is following.

Indices calculated from nominal data layers.

Proportion of a given category — the share [%] of a given class within a given kernel.

Mode — the code of category having the largest share in the kernel.

Shannon's index of diversity — the diversity of pixel classes according to the formula $H = -10 \cdot \sum p_i \cdot log_2 p_i$, where p_i **—** share of the class *i* in the kernel.

Lloyd's index of equitability — calculated according to the formula $10 \cdot H / lg(s)$, where H **—** Shannon's diversity, s **—** number of classes within the kernel.

Dominance — calculated using the formula $100 \cdot \sum p_i^2$, where p_i **—** share of the class *i* in the kernel.

Number of classes — the number of different categories in the kernel.

Class adjacency — the ratio of edges between the pixels of the same class to the number of all edges. Equals 100 when all pixels within the kernel belong to the same category.

Direction of patches — 0 means one pixel wide vertical stripes, 90 means horizontal one pixel wide stripes. Value 255 marks the absence of patch borders.

Class proximity — the ratio of the sum of inverse distances between pixel centres of the same class to the sum of inverse distances of all pixel pairs. Given as percentages.

Share of different class pairs — the ratio of pixel pairs with pixels belonging to different class to the number of all pixel pairs within the kernel. Given as percentages.

Distance weighted mode — the mode weighted by inverse distance. The category for which the sum of inverse distances of pixels from the kernel centre is the greatest. The focal pixel of the kernel is not included.

Distance to class boundary — the minimum distance in pixels from kernel centre to the closest class boundary.

Minimum distance to a given class — the distance from focal pixel to the closest pixel of a given category in pixels.

Distance weighted frequency — frequency of a given class weighted by inverse distance.

Second mode — the next frequent category after mode.

Third mode — the third frequent category after mode and second mode.

Indices calculated from numerical data layers.

Share of pixel values above the mean — the share of pixel values [%] exceeding the mean value within the kernel expressing the asymmetry of the distribution of values.

Mean — the arithmetic mean of pixel values within the kernel.

Standard deviation — the square root of the sum of squared deviations of pixel values from the local mean.

Median — a pixel value for which the number of smaller pixel values is equal to the number of pixels having higher values.

Moran's *I* of 8 neighbouring pixels — spatial autocorrelation according to Moran's *I* of 8 neighbouring pixels. 0 means maximum possible negative spatial autocorrelation, 100 means absence of spatial autocorrelation and 200 **—** maximum positive spatial autocorrelation (similar values adjoin).

Distance weighted Moran's *I* — the inverse distance weighted Moran's *I* having the same scale as the previous index.

Difference of neighbouring pixels — the mean difference between adjacent pixels.

Coefficient of variation — the ratio of the standard deviation to the mean × 100.

Gradient direction — direction of inclination of the linear trend surface within the kernel. 255 means no gradient, 50 **—** increasing values in direction of y-axis, 150 **—** decreasing values in direction of y-axis, 100 **—** increasing values in direction of x-axis, 1 and 200 **—** decreasing values in the direction of x-axis.

Difference of border to centre — difference between the mean value of pixels within half radius and the mean value of pixels located between half and full radius of the kernel. A constant = 100 is added to the difference and values less than 0 and above 254 are truncated, so the final statistic is in range 0…254, 100 meaning no difference between central and peripheral part of a pixel.

Minimum — the minimum value within a given kernel.

Maximum — the maximum value within a given kernel.

Factor of kurtosis — calculated using the formula: $E = \frac{10}{n}\sum\left(\frac{x_i - \bar{x}}{\sigma}\right)^4$, where x_i **—** pixel value, $\bar{x}$ **—** the mean of pixel values, n – the number of pixels within the kernel and σ **—** the standard deviation of pixel values. Greater values appear where areas of different brightness are close together.

Gradient smoothness — homogeneity of neighbouring pixels. Calculated using the formula: $S = 100 \cdot \frac{\sum_i \sum_j \frac{1}{1+(x_i - x_j)^2}}{n}$, where x_i and x_j — pixel values of neighbouring pixels in horizontal, vertical and diagonal directions and n — the number of pixels within the kernel.

Difference from the mean — relative value as the difference between the focal pixel value and the arithmetic mean of all pixels within the kernel.

Gradient intensity — the angle of inclination of the linear trend surface of pixel values within kernel. Difference of pixels within kernel to mean pixel value is multiplied by pixel distance in gradient direction. Values range from 0 to 200. Zero means no gradient (sum of weighted pixel distances on gradient equals the sum of unweighted pixel values on gradient). Value 10 means the pixel values on a gradient change by one when distance is increased by one pixel.

Distance weighted mean — the mean of pixel values weighted by inverse distance from the focal pixel.

Directional structures — stripeness calculated first as the largest difference between the average of 9 pixels and 3 pixels in a line in four directions (north to south, east to west, north-west to south-east, and north-east to south-west) around every pixel and thereafter as the mean of all partial differences within the sample in direction of the maximum difference. The stripeness = 0 if no directional structures are present.

Table 3.4. Required fields in table *SP_Indices*.

Field name	Format	Comment
Index_ID	byte	ID of the index
name	text	name of the index
nomlayer	yes/no	*yes* means the index is calculated from nominal layer
nomind	yes/no	*yes* means the values of the index represent nominal categories

Table 3.5. Spatial pattern indices available in *Constud*. The index numbers correspond to the ID-numbers in table *EXPL_VAR*, field *indexID* (Table 3.3). The values of indices are handled as class codes of a nominal feature if *nomlayer = yes* and as numerical features if nomlayer = *no*. If nomind = *yes* then the index is nominal, if *no* then numerical.

No.	Name	nomlayer	nomind
1	share of given class	*yes*	*no*
2	mode	*yes*	*yes*
3	Shannon	*yes*	*no*
4	Lloyd	*yes*	*no*
5	dominance	*yes*	*no*
6	number of classes	*yes*	*no*
7	class adjacency	*yes*	*no*
8	direction of patches	*yes*	*no*
9	class proximity	*yes*	*no*
10	share of different class pairs	*yes*	*no*
11	distance weighted mode	*yes*	*yes*
12	distance to class boundary	*yes*	*no*
13	distance from a given class	*yes*	*no*
14	distance weighted frequency	*yes*	*no*
15	second mode	*yes*	*yes*
16	third mode	*yes*	*yes*
101	share of values above mean	*no*	*no*
102	mean	*no*	*no*
103	standard deviation	*no*	*no*
104	median	*no*	*no*
105	Moran's *I* of 8 neighbours	*no*	*no*
106	distance weighted Moran's *I*	*no*	*no*
107	mean difference of adjacent pixels	*no*	*no*
108	coefficent of variation	*no*	*no*
109	gradient direction	*no*	*no*
110	difference of border to centre	*no*	*no*
111	minimum	*no*	*no*
112	maximum	*no*	*no*
113	factor of kurtosis	*no*	*no*
114	gradient smoothness	*no*	*no*
115	difference from mean	*no*	*no*
116	gradient intensity	*no*	*no*
117	distance weighted mean	*no*	*no*
118	stripeness	*no*	*no*

3.2.2.3. Table of descriptive features

Indices describing image and map properties at observation locations have to be computed before machine learning and predictions. Values of descriptive features can be stored in the table of observations or in a separate table. Required fields in the table of descriptive features are *VID* — observation number (format 8-byte integer) and fields containing values of descriptive features (format byte). If feature vectors are accessed through a feature vector query, the format of the query has to match application's requirements. The name and format of the table of descriptive features is free.

4.3.3. Feature vectors of observations

To prevent data redundancy, it is advisable to create feature vectors as queries requesting data from the tables of observations and results. If training data are in a table containing more than one predictable variable, names of predictable variables must be redefined in a query. For every predictable variable the field name for the values of the dependent variable has to be *F*. Fields *pred*, *sim*, *actuality*, and *w* have to be for storage of the results of a learning process (Table 4.8).

Predicted values for observations, observation weights and actualities are stored by *Constud* application. The mean value of a continuous dependent variable is recorded if prediction can not be calculated. Missing data is recorded as 255 in the case of other types of dependent variables.

The same feature vector is used for all categories in separate estimation of categories of a multinomial variable (*ftype* = 4). Hence, feature vectors match *FID* codes and not to rows marking categories in table *DEP_VAR*.

Table 4.8. Required fields in feature vectors.

Field name	Format	Comment
VID	8-byte integer	Identifier of observations
F	4-byte real if *ftype* = 2, yes/no or byte if *ftype* = 3, byte if *ftype* = 1 or 4	Value of the dependent variable
pred	same as for field *F*	Value predicted using training data (needed only for machine learning and predictions stored in database table)
sim	byte	Similarity to exemplars chosen from the rest of training data. Values are in the range 0…100 (needed only for machine learning and when predictions are stored in a database table)
actuality	byte	Actuality of the observation (needed only for machine learning)
w	4-byte real	Weight of observation as an exemplar
x	8-byte double	West-east direction in Cartesian coordinates
y	8-byte double	South-north direction in Cartesian coordinates
from	date, format depends on op-system settings	Beginning of the observation validity. Leave empty if not applied.
until	date, format depends on op-system settings	End of the observation validity. Leave empty if not applied.
<number of feature>	byte	Values of explanatory variables

Fields containing the temporal constraints of observations can be left void, temporal validity of data layers must always be indicated. For machine learning and predictions, only those combinations of features and observations are used, which have overlapping temporal ranges. For example, *Constud* skips remote sensing data not valid before clear-cut date for

observations located on a clearing, although can still use soil maps because these are valid during a long period. If temporal constraints of an observation are left void the validity of this observation is not constrained.

4.3.4. Log tables

Log tables store the results of machine learning iterations. The name of a log table must begin with string *LOG_F* followed by the number of a dependent variable (for example *LOG_F18*). There must be a separate log table for every dependent variable used in machine learning or in predictions. In individual estimation of classes of a multinomial variable (*ftype* = 4), the learning results of every class are stored in the same table, feature vectors corresponding to *FID* codes and not to rows in the table *DEP_VAR*. Required fields and formats for log tables are described in Table 4.9. Earlier versions of *Constud* (before 2008) required fields of weights separately for every explanatory variable even if the weight of the variable was zero. *Constud_2008* stores the variable weights exceeding zero in one field — *F_weights*. *Constud_2008* can read feature weights from both older and newer formats of log tables.

Table 4.9. Required fields in log tables.

Field name	Format	Comment
ID	8-byte integer	Number of iteration
given	byte	Code of the category that is learned separately from the other categories of the multinomial variable
date	date	Date and time of saving the record
t_fit	4-byte real	training fit of observations and predictions (leave-one-out cross-validation — LOOC in the training sample)
v_fit	4-byte real	Validation fit of observations and predictions (LOOC in the validation sample)
sumsimmax	4-byte real	Sum of similarity required for decision
etn	4-byte integer	Number of exemplars selected in this learning iteration
sample	4-byte integer	Sample size
<feature No.>	4-byte real	Weights for features (necessary for versions preceding *Constud_2008*)
F_weights	memo	Weights for features as: [no of a feature] [weight] [no of a feature] [weight], e.g.: 32 1.2342 35 0.328 87 0.9387.

Log table must not be completely empty. For the first usage, a single row where ID = 0 has to be manually inserted by the user.

4.3.6. Table of results

The table of results must contain fields for dependent variables: 1) field *VID*, 2) predicted value for every observation (field *pred*), 3) similarity of every predicted value to exemplars

used for prediction (field *sim*), 4) actuality of every used observation (field *actuality*), 5) weight of every observation used as an exemplar (*w*).

In machine learning, the field of similarity is used to store mean similarity of exemplars used in prediction of a numerical variable. In the case of nominal variables, it stores the share of similarity expressing the confidence of decision (ratio of the sum similarity of exemplars of the predicted class to the sum of similarity of all examples chosen for decision). During separate learning of nominal categories, the share of similarity to exemplars of current class to all exemplars selected for decision is recorded in results. When computing similarity to a given class, the mean similarity to the given class is stored for every observation.

Prediction fit is recorded as kappa index of agreement of multinomial variables and as a modified true skill statistic (TSS) (Allouche et al., 2006) for a binomial variable. The modification uses the sum of similarities rather than counts (Remm and Remm, 2008). Root mean squared error is the objective function for a numerical variable, and the matching share of predictions and observations for a complex variable.

Table 4.5. Required fields in self-similarity tables.

Field name	Format	Comment
1	byte	Category of the first nominal feature
2	byte	Category of the second nominal feature
s	byte	Similarity in scale 0…100 (percentage points)

5. Application *Constud*

5.1. Technical preconditions

Constud is installed using Microsoft Windows Installer deployment. The *Constud* program is installed to the client's computer over Internet. The program checks for updates every time before starting. *Constud* runs on low processor priority, minimizing the interference to other processes and applications.

5.2. Main window

The header of *Constud* main window contains application version. Four sections of different tasks are marked in colour: green — new data (calculation of values for observations or to a raster file), yellow — learning (finding of optimal weights for features and exemplars), pink — knowledge test (validation of the fit of predictive sets and reselection of exemplars), blue — knowledge application (calculation of predictions and spatial indices to a database table or to a raster file (Figure 5.1). If the checkbox of spatial data is unchecked, coordinates of observations in the feature vector and the distance correction value in the table *DEP_VAR* are not required, and spatial data layers are not used. Fitting of the sum of similarity is active only when checked. Machine learning stops when the number of iterations exceeds the pre-set maximum. Standard deviations can be read from the table *STDEV* if the table is in database and contains standard deviations of all numerical explanatory variables used in machine learning or in predictions. Standard deviations have to be stored separately for every pre-class if a preclassification is used. If the option *read SD from the table STDEV* is left unchecked, standard deviations are calculated from the training sample defined in table *DEP_VAR*. Standard deviation from a training sample is usually less reliable estimation of variability than standard deviation calculated from all observations.

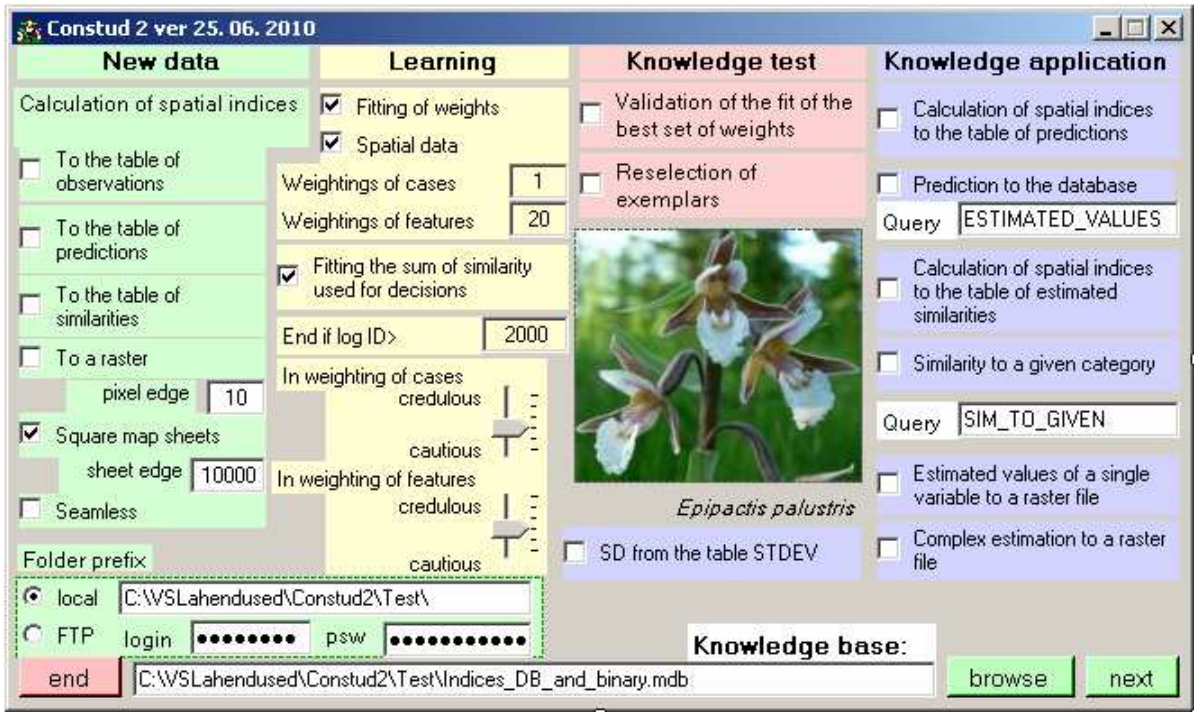

Figure 5.1. *Constud* version 25.06.2010 main dialog window.

5.2.1. Configuration file

Configuration file app.config can be opened using any text editor. It contains default values for parameters defined in user dialog windows.

5.3. Calculation of spatial pattern indices

Tools for the calculation of spatial indices can be found in *Constud* main window in the division *New Data*. When calculating map or image pattern indices to database, the name of the table or query is received from table *DEP_VAR*. If indices are stored in the table of predictions or similarities, the name of the output table or query must be determined in a text box in the division *Knowledge Application* (Fig. 5.1).

If the calculated indices are stored in a raster file, *Constud* reads the output path from the field *BO_catalog* in the table *EXPL_VAR*. *Constud* computes only indices that are marked as *true* in the table *EXPL_VAR* (field *F<FID>*), where the number of a dependent variable matches the name of the field *F<FID>* in the table *EXPL_VAR*, and field *calc* in the table *DEP_VAR* is set to „true“. Indices are computed from every raster file in the marked folder. To calculate indices from selected raster files, these files should be moved to a separate folder and the path to the folder must be temporarily modified in the table *D_Layers*.

For calculation of indices to the database, each observation must have unique identification number along with coordinates *x* and *y*.

5.4. Machine learning

Machine learning in *Constud* is an iterative mining of weights for exemplars and for features to find the best predictive sets of weights for similarity-based alias case-based predictions. The machine learning section is marked in yellow background in the *Constud* main window (Fig. 5.1).

The two main goals of similarity based machine learning are: 1) to find the best set of examples for the prediction of a dependent variable, and 2) to find the best combination of features for recognition of the predictable variable. All observations are not suitable as etalons because are redundant with existing data, are odd, outdated, or unreliable. Also, all features are not equally useful for similarity measurement. Some duplicate each-other, others are not useful in finding object of interest, and some data layers may be outdated due to landscape changes. Maps use different sign systems and generalization; remote sensing data depends on apparatus, state of the atmosphere and vegetation at the moment of registration. Both maps and remote sensing imagery may contain random data errors. A rule of thumb is — the smaller is the number of exemplars and features used, the faster is the computation of similarity-based predictions.

Similarity in some features does not mean similarity in all features. For example spruce, Siberian larch and birch are all somewhat similar yet differ greatly in some other aspects. Spruce and larch are coniferous trees; birch and larch are deciduous trees; spruce and birch share their habitats. It is important to find the features for similarity-based reasoning that describe phenomenon of interest as best as possible.

Intensity of the red band of an image has been a helpful feature for recognition of habitats of an orchid species *Dactylorhiza russowii* from digital orthophotos, although this does not mean a clear correlation between the intensity of red band and the probability of occurrence of the orchid. Merely, it is important to take into consideration the intensity of red tone in measuring the similarity between a location and the species’ find and absent locations.

5.4.1. Preparation for machine learning

The dependent variables to estimate, the explanatory variables, and the training data to use have to be decided before starting machine learning. Descriptive variables of the training observations must be calculated into the table of training data before starting the machine learning. Indicative values of descriptive indices can be evaluated beforehand to choose only the features having higher indicative values to speed up the machine learning process.

For learning a new dependent variable, the following preparations have to be made: add a new row with a unique FID-code to the table *DEP_VAR*, add columns (*F<FID>* and *AF<FID>*) to the table *EXPL_VAR*, add a log table named *LOG_F<FID>*; add four columns into the table of results: for predicted values, similarity with exemplars, exemplar weights, and actuality; generate a query to display aforementioned columns as features having names (*pred*, *sim*, *w*, *actuality)*, values of the dependent variable as *F*, and values of explanatory features. Write to the table *DEP_VAR*: the name of the query for feature vectors to the field *Dataquery*, type of the dependent variable as *ftype*, ID of the explanatory variable used as a preclassifier (if a preclassifier is applied), initial value for the critical amount of similarity needed for decision as *sumsimmax*, desired volume of training sample and for validation sample as *t_sample* and *v_sample*, desired number of features used as *no_features* zero-distance for similarity reduction of close observations as *0_dist*, and the code-number of data layer of the preclassifier when using pre-classification. *Constud* does not accept learning samples larger than 2000 observations. Only approximately 2000 random observations will be used in training iterations if sample volume is set to be larger.

Field *AF* in the table *EXPL_VAR* can be empty before the learning process — the actuality value 100 is applied by default. Log table must include at least one row with any ID value (preferably 0). Fields *pred*, *sim*, *w*, *actuality* may be empty before machine learning.

5.4.2. Machine learning procedure

After creating necessary database objects and parameters, machine learning can be started from *Constud* main window. Primary available machine learning options are listed hereafter.

1. A single learning iteration. All features and observations are used at once. The computation may take a substantial amount of time if the number of observations exceeds 1000 and the number of features is over 30.

2. Single or iterative machine learning (continuous study) using a fixed set of features selected by the user. Explanatory features have to be selected in table *EXPL_VAR* and the number of features indicated in the table *DEP_VAR* must be equal to or greater than the number of chosen features.

3. Continuous study using repetitive random sampling of features and observations. Actuality value for each feature and observation is stored in the connected database and its value will be changed automatically according to the usefulness of the feature/observations in current sample. The probability of feature/observation being used in further samples is proportional to the actuality. This method is suitable if there are more than a few hundred observations and over 30 features and the user does not wish to interrupt the learning process.

4. Continuous study using random sampling of features. User can remove features that have reached actuality below a threshold and thereafter start single learning using all preserved features and all observations. Method is recommendable when there are less than 1000 observations and more than 30 features.

5. Continuous study without weighting of observations using samples of a few hundred observations. This method is relatively fast in resolving the more indicative features. After that, the less actual features can be deactivated and the best solution be calculated using less features and larger samples. It is suitable when there are more than a few hundred

observations and more than 30 features and user wishes to speed up feature selection by deactivating useless features from further use.

6. Separate learning in every preclass and/or region. The dependent variable in every preclass and/or region has to be registered as a separate variable in the table *DEP_VAR*. A separate query of feature vectors and fields *F* and *AF* in the table *EXPL_VAR* for every preclass has to be prepared. Explanatory features and exemplars are chosen separately in every preclass. A common table of results for all preclasses is suggested as each observation belongs to a single preclass. Method is suitable if more accurate estimates are expected using different features in different preclasses or geographical regions.

7. For a multinomial dependent variable, it is possible to choose machine learning of categories simultaneously (*ftype* = 0) using the same features for all categories, or to prefer separate learning of categories (*ftype* = 4) using possibly different sets of features for every category.

If preliminary results are not satisfactory, better prediction accuracy might be sought by adding new features, by joining classes that were difficult to separate and restarting learning process on the joined class or using new explanatory features. The training data should also be critically revised since possible errors in case descriptions, in data layers, and also outliers in training data alter prediction results.

5.4.3. Machine learning phases

Learning iterations in *Constud* consist of five stages in the given fixed order: 1) selection of features, 2) weighting of features, 3) selection of exemplars, 4) weighting of exemplars, 5) changing the actuality of features and observations.

Selection of features is one by one inclusion of features in the order of usefulness while gradually decreasing feature weights. All training observations in the training sample are used in this process. The best predictive set is used in the next phase — feature weighting. The weights of selected features are increased and decreased by a small random value, if the goodness-of-fit improves; the new weights are applied in further learning. Hitherto the best set of feature weights is used for the selection and weighting of observations.

5.4.3.1. Selection and weighting of features

In the machine learning process, features are selected first. Features are tested one-by-one and the one giving the best goodness of fit between predictions and the training data is chosen. During the next iteration, features are added one-by-one to the feature(s) selected during previous iterations. One-by-one addition of features ends when all features are added. Goodness-of-fit of the prediction is calculated in the end of each iteration.

Weights of added features are decreased using formula:

featureweight(p) = 1 – (feature rank - 1) / (number of features being used + 1),

where: rank is the order of inclusion of the feature, number of features being used is the size of feature sample, *p* is the feature index.

Weights of the added features are divided by the mean weight of features so that the mean weight of features in the predictive set is always equal to one. The set of features that gave the best results in feature selection phase is used in the feature weighting phase. The best feature set usually does not contain all features from the sample of features because some features are not related to the predictable variable, others are highly inter-correlated and only one of them is chosen.

Inclusion of features is followed by a weighting process during which the feature weights are repeatedly changed to improve the goodness-of-fit. The altered weights are calculated according to formula:

changedweight(p) = featureweight(p) – 0.5 + Rnd()
where: *Rnd()* is a random number in range 0...1.

The impact of changing the amount of similarity to the prediction accuracy is evaluated in addition to the changes in feature weights. The amount of similarity is changed according to formula:

$changed\_amount\_of\_similarity = amout\_of\_similarity - 1 + 2 \cdot Rnd()$.

The set of changed weights and amount of similarity that gave the best matching predictions is used in the next iteration. The number of weightings of features can be set by user in the *Constud* main window.

5.4.3.2. Selection and weighting of exemplars

Before the selection of exemplars, goodness-of-fit is calculated in the training data given all observations are used as exemplars. Thereafter, the last observation is removed and goodness-of-fit is calculated again. If goodness-of-fit decreased, the removed observation will be used as an exemplar, if not the observation is excluded from the set of exemplars. Observations are handled in the decreasing order of *VID*. The removal of observations back to forth helps to maintain conservativeness of the set of exemplars — the older observations having a smaller *VID* among similar ones will be preferred to the predictive set. An observation is not removed when it is the last one in its pre-class or if the number of exemplars for a predicted category is less than 2 + truncated value of the amount of similarity looked for decision.

Parameters of learning can be set in *Constud* main window (Fig. 5.1). If the number of weightings is set to 0, only selection of exemplars is performed — all selected observations have the same weight. The weighting of exemplars also takes place in decreasing order of identification numbers. All exemplars within the sample, including those having weight set to zero in the previous iteration, are used in the weighting process. Three alternatives are being compared in the feature weighting: 1) increase of exemplar weight, 2) weight left unchanged, and 3) decrease of weight. The initial amount of change is 0.5; it is divided by 2 for every following iteration. Three iterations are usually sufficient.

5.4.3.3. Changing the actualities of observations and features

Actuality values are used during the generation of feature and exemplar samples – the higher the value, the more likely for it is to fall into the sample. By default, the initial actuality of a feature/exemplar is 100 units, if not set otherwise by the user. The actuality is raised for a feature and/or for an exemplar that has been selected from the learning sample and has turned out to be useful (the prediction accuracy has increased). The actuality of features and exemplars selected to the sample but which did not improve the prediction, the actuality is decreased. The maximum actuality value is 200; the minimal possible value is 1. No feature or an observation is totally discounted and therefore, there is always a chance it will be used in the learning process and to be selected into the best predictive set.[7]

[7] Features can be removed from machine learning by disabling the corresponding *F*-field in table *EXPL_VAR*. To remove the cases, the case selection query has to be modified.

After approximately ten learning iterations, actualities of features and exemplars are normalized the mean actuality of features and exemplars currently in use to be equal to 100.

The extent of changes in actuality codes is regulated by a *factor of credulousness* which can be adjusted using sliders in the *Constud* main window (Fig. 5.1). The factor of credulousness is an integer value between 1...6. Slider units match the factor values. When the learning process is set to be more credulous, the actualities are changed faster and the more actual features and observations are bought fourth in a less number of iterations.

When an observation has been selected as an exemplar, its actuality is increased by the value of observation weight multiplied by the factor of credulousness. If observation is in training sample but appears to be useless as an exemplar, its weight is decreased by the ratio of the sum of exemplar weights to the number of observations not being used as exemplars. Due to this, the total increase and decrease of actualities is balanced.

Feature actualities are changed similarly to observation actualities. The mean weight of features is constantly normalized to 1, which means that the sum of weights equals the number of features. Therefore the process of changing actualities of features is computationally simpler.

5.4.3.4. Generation of samples

Generally, the observations are added into the sample when their actuality code is greater than:

sum of actualities of all observations / sample size · Rnd(),

where *Rnd()* is a random number between 0...1.

If the number of presence observations of a binomial (presence/absence) variable in the training dataset is less than the sample size, all presence observations are included to the sample. If the number of absence locations is less than a half of the sample, they all will be included to the sample. If the sample size set in table *DEP_VAR* is equal or greater than the number of observations, all observations will be included.

Observations not included in the training sample are preferred for the validation sample. If the number of observations not included in training sample is less than validation sample size, they are all used for validation, and the lacking are randomly chosen from the training sample.

Selection of features is also random and the probability to be included into sample is proportional to the actuality of a feature.

5.4.3.5. Similarity calculation

Before similarity calculations, temporal validity of observations is verified first (Chapter 4.3.3). After verification, similarity between observations (partial similarity) is calculated in respect of every feature. Second temporal check is needed to ensure that temporal limits of the data layer used for feature calculation and that of observations are overlapping. When temporal limits do not overlap, the partial similarity between the observations is not calculated.

The rule of calculation of partial similarity depends on whether the explanatory variable is nominal or numerical. If categories of a nominal variable match, the partial similarity equals to one. If classes differ and self similarities of classes are not considered, partial similarity equals to zero. When self similarity tables are used, partial similarity equals the similarity between the classes as set in self similarity tables.

For numerical features, similarity between an observation and an exemplar is calculated using inverse difference. Difference is calculated using formula (Remm, 2004a):

$$D(T_p, E_p) = \frac{|T_p - E_p|}{V_{p,precl} \cdot w_E \cdot w_p},$$

where:

p — feature index,
$D(T_p,E_p)$ — standardized difference between values T_p and E_p,
E — exemplar,
T — observation,
$V_{p,precl}$ — normalizer of feature p value in pre-class *precl*,
w_E — weight of feature E,
w_p — weight of feature p.

Normalizing vector V equals double standard deviation of the feature p over all training observations in pre-class *precl*. If pre-classifier is not used, standard deviation is calculated using all training observations. Weights in the denominator regulate the range of equal difference along feature value axis. Greater differences in feature values of the features and exemplars that have greater weights are comparable to smaller differences in feature values if the features and exemplars have smaller weights. For observation pre-classes 0 and 255, observations from all pre-classes will be used. If calculated difference is greater than one, partial similarity is zero. Otherwise, partial similarity is calculated by subtracting difference from one.

Weighted mean total similarity is calculated from partial similarities. As in the previous formula, weights are the cross product of observation and feature weights. Zero-weight features, features not suitable due to temporal limits and features not applicable for other reasons are not used for the calculation of the weighted mean of partial similarities. For example, if soil type data for one observation location is missing, soil types do not affect the total similarity between observations.

5.4.3.6. Decision making

Decision process in *Constud* is driven by similarities. If only the most similar exemplar is used for prediction, prediction will be equal to the value of dependent variable of that exemplar. More generalized predictions are given using more than one exemplar. The number of exemplars used is limited by the sum of similarity needed for decision (total similarity). Initial value of the total similarity is read from table *DEP_VAR* and is attuned by *Constud* during the feature weighting process. Most similar exemplars are sought for as long as the total similarity needed for decision is reached or exemplars with similarity greater than zero exist. When a pre-classifier is used, only exemplars among the given pre-class are weighed.

Value of a continuous dependent variable is calculated as the mean of the values of exemplars used for the decision weighted by the similarity of values of dependent variable. For a nominal dependent variable, predicted class will be the one with greatest sum of similarities of the exemplars used. In the learning of classes of a multinomial feature, the dependent variable is handled as being Boolean: whether it belongs to the given class or to some other class.

During the learning process, the level of similarity at the moment of decision is stored along with estimated values. For numerical features, it is calculated as the arithmetic mean of similarities of exemplars used for decision. For binominal features — the mean similarity to exemplars with greater value or to category coded as „*true*“ (similarity to exemplars of occurrence), for multinomial features — the mean value of similarity to exemplars of the given code. As for binominal features it is presumed that occurrence observations are marked by greater number than absence observations, it is advisable to mark the observations of

occurrence by value one and observations of absence by value zero or to use yes/no format of the field.

5.4.3.7. Goodness-of-fit of the estimations

Constud calculates the goodness-of-fit between observations and estimations using leave-one-out cross validation (*LOOC*). LOOC means that the predicted value for every observation is calculated by all exemplars, leaving this observation out. Goodness-of-fit calculated for test sample using the best set of weights found from learning sample is called validation fit.

In case of a multinomial feature, goodness-of-fit is measured by the Kappa index of agreement.

The basis of goodness-of-fit for binominal features are similarities. The fit of a binominal variable is calculated as: true positive similarity + true negative similarity –1. The resulting objective function, called true skill statistic (*TSS*), assigns equal weight to both correct negative and correct positive estimations and is similar to the Pearson's coefficient of correlation having interval of possible values between –1 and +1. Goodness-of-fit =1 expresses absolute fit between observations and estimations, –1 means that all estimations are false, goodness-of-fit equals zero if a half of positive and a half of negative observations have been estimated correctly. *Constud* does not count positive and negative estimates but instead summarizes the similarity to positive and negative exemplars. For example if the sum of similarity with positive exemplars is 1.8 and the sum of similarity with negative exemplars is 0.6, then, 1.8 is added to the count of true positives if the observation is actually positive, and 0.6 is added to the count of true negatives if the observation is actually negative. For TSS calculation, summarized true positive similarity and summarized true negative similarity are both divided by total sum of corresponding similarities.

For multidimensional numerical variable, goodness-of-fit is measured by the amount of co-incident sub features. One-dimensional numerical variable is handled as a continuous variable. For such variables, goodness-of-fit is characterized by standard deviation where the number of features used for prediction is subtracted from the number of observations.

5.4.3.8. Storage of machine learning results

Feature weights found during learning iterations and the similarity levels needed for decision making are stored in log tables. In case of *ftype* = 4, log table field *etn* is used to store the number of exemplars of a given class, not the number of all exemplars as for other types of functional variable. Field *date* is used to store the exact time of recording and field *t_fit* to store the goodness-of-fit achieved during given iteration. Validation fit that is calculated by using the best set of weights is stored in field *v_fit*. Validation fit is calculated only when the training fit after iteration is better than hitherto the best validation fit.

Feature actualities are stored in table *EXPL_VAR*. Actualities of observations, values predicted using hitherto the best set of weights, observation weights in hitherto the best set and the level of similarity for decision making are stored in a table or query as set in table *DEP_VAR*.

Field *LogID* in table *DEP_VAR* is used to store the identification number of feature weights set with hitherto the best goodness-of-fit. The value of this field, feature weights, predicted values and level of similarity for decision making are modified only when a set of weights giving better goodness-of-fit than the hitherto the best set is found during learning iteration. It is essential to remember that actualities of features and observations are changed at the end of each iteration and feature weights are stored in log tables even if prediction was not hitherto the best.

5.5. Knowledge test

Knowledge test section is marked by pink background in *Constud* main window (Fig. 5.1). It features two check-buttons: fit validation of the best set of weights and reselection of exemplars.

5.5.1. Fit validation of the best set of weights

Fit validation is necessary when training set or the set of exemplars and features has been modified during the machine learning process or by the user (observations added or removed, feature set or feature weights being modified). On the contrary to machine learning when exemplar weights are read from memory and optimized during learning process, fit validation uses exemplar weights stored in the database. When learning, the validation fit is calculated using a training sample; separate validation fit is calculated using all observations.

In separate learning of categories of a nominal feature (*ftype* = 4), fit validation found during the learning process differs from the result of standalone calculation. The optimal exemplars needed for class separation and a set of exemplars representing all other classes are chosen during separate learning of every class. Only weights for the currently learned class are stored after each iteration giving hitherto the best results. For *ftype* = 4, fit validation is calculated separately for every category of the nominal dependent variable. All observations are used, but only with features optimized for the given category. The rest of the classes are represented by exemplars from hitherto the best set for particular category, which may not be suitable for separation of categories features with chosen for separation of the given category. In similarity based estimations, exemplar set and feature set are related to each other. Recombining of features and exemplars may yield unpredictable results. Learning of classes of a multinomial feature using different features for each class still needs additional research and testing.

5.5.2. Reselection of exemplars

Reselection and weighting of exemplars can help in seeking the best predictive set when hitherto the best estimates have been obtained using observation sample (training fit is better than validation fit). For reselection, exemplars with higher actuality value are preferred. During exemplar selection, at least one observation for every pre-class is preserved. If possible, the number of observations representing categories of nominal feature is kept higher than the value of total similarity sought for decision.

The reselection of exemplars may be time consuming and might not give as good a set of exemplars than hitherto the best set found in learning process. It takes approximately 10 minutes if the number of exemplars is about 1000. The process slows down significantly when the number of exemplars increases. In case of a large training set (more than 2000 exemplars) and limited time, it is recommended to use machine learning with less than 1000 random observations instead of exemplar reselection.

During reselection exemplars of one-by-one learned categories of multinomial feature (*ftype* = 4), only observations of given class are selected and weighted. Observations of other classes are used with weights previously stored in the knowledge base.

5.6. Knowledge application

Knowledge is applied for giving estimations. The estimated value can be a code of predicted class, a value of a continuous variable, or similarity to a predicted class, to exemplars or to exemplars of a given class. *Constud* can store predicted values and

similarities to a database table or into a raster file in *Idrisi32* binary format. Tasks related to estimates are on blue background under the knowledge application subsection of the main window (Fig. 5.1).

Calculation of predictions in knowledge application is similar to prediction in machine learning except for the case of multinomial feature classes. In prediction of separately learned classes of a multinomial variable, similarities of an observation or a location to every single category are calculated and the class code of the most similar category will become the predicted value. For separate estimation of separately learned classes, estimates for every class must be calculated one-by-one and saved in different files (option *estimated values to a raster file* in *Constud* main window; Fig 5.1). In this case, all categories of the multinomial variable are handled as binominal variables and the decision describes whether the location is more similar to given class than to other classes.

Prediction of a continuous variable is replaced by the mean value of the variable in the training sample when similar exemplars are absent. For, value 255 is used for nominal, false for Boolean variables.

5.6.1. Prediction with similarity estimates

The similarity calculated along with prediction describes the similarity between the location and exemplars used for decision. Similarity to the most similar exemplars is usually high but it does not mean that similarity to exemplars of other classes or to different values of a continuous variable should be smaller. Sometimes, many exemplars with different values of the dependent variable can resemble the given observation or location.

For numerical variables, the mean similarity of exemplars used for decision is stored. For multinomial and binominal variables, the ratio of the sum of similarities of predicted class to the sum of similarities of all exemplars used for decision is stored. Similarity value of a binomial variable in predicted find locations describes the similarity to exemplars of find locations, in absent locations the similarity to exemplars of absent locations. Similarity value 50% means the location is equally similar to both find and absent locations.

To calculate similarity to given categories, all given categories must be selected in table *DEP_VAR*.

5.6.2. Predictions to a database table

A database table can be used to store both the predicted values and the similarity to exemplars of a given class. Name of the table to be used for storing predicted values has to be specified in *Constud* main window (Fig. 5.1), the table must contain fields *F, sim,* and *VID*.

Field *F* must match the format of the dependent variable; field *sim* must be in byte format as it is used to store similarity to exemplars in percentage points as integers. Field *VID* is used for observation indentifier in integer format. In addition to aforementioned fields, fields for descriptive variables are required, having names matching the identification numbers of explanatory variables in table *EXPL_VAR*.

When calculating similarity to exemplars of a given class, table structure must be similar to that of predictions table except that field *F* must contain the code of the given class in byte format. Field *sim* is used to store the mean similarity of the most similar exemplars. The number of the most similar exemplars is limited by the value of the sum of similarity sought four decision, which is recorded in log table on the row of hitherto the best feature weights.

For the calculation of both similarity estimation and prediction, data query defined in table *DEP_VAR* is used. Queried data is used for the calculation of exemplar attributes and standard deviations of descriptive variables. Pre-computed features are not used in these

calculations and descriptive features must be calculated and stored in knowledge base before starting the process.

5.6.3. Estimated values to a raster file

All raster files saved by *Constud* application are in without header binary raster format (*Idrisi32 rst*-file, top-down row major). These raster file can be used to store estimated values of both nominal and numerical variables (prediction maps), level of similarity at decision making (decision confidence) and similarity of locations to exemplars of a given class (Fig. 5.2). The speed of map calculations depends on the number of features, the number of exemplars and in the case of estimation of single categories of a nominal variable (*ftype* = 4) also on the number of active classes in the *DEP_VAR* table (when the map of estimated values is calculated in addition to similarity map).

For both similarity- and estimated value maps, pre-computed features can be used. It can significantly reduce the time needed for map generation. The number of data layers used for generation of estimation map must not exceed 15 and the number of features is limited to 255.

Parameters of the predicted variable can be modified through dialog (Fig. 5.3). The parameters are as follows:

1. Map sheet number which determines the files of descriptive variables to be used.
2. Pixel size in used files.
3. Interpolation parameters: initial length of step in pixels, maximum difference of similarity between the pixels to be interpolated, in case of continuous variable, difference of similarity between the pixels in a data layer of a continous variable. This input box is visible only when *ftype* = 2 in the table *DEP_VAR*. The algorithm of interpolation is described in chapter 5.6.3.1.
4. File name of the prediction map. Not visible when only similarity is calculated.
5. File name of similarity map. The mean similarity to exemplars of the given class is stored if calculating similarity to given class is selected. Otherwise mean similarity to exemplars of predicted class is stored.
6. Path to the file of interpolation polygons.
7. Path to the file of spatial pre-classification variable. Visible only when field *precl* in *DEP_VAR* table contains the code of pre-classification feature.
8. Values to be ignored in pre-classification files. Calculations are not performed on areas containing these values. It enables to compute predictions on only these areas that are covered with certain land-cover class or defined by the file of interpolation polygons.
9. File of given categories. Visible only when output type is similarity to the given category. It defines the binary raster file that contains class codes to be used for similarity calculation. It is possible to choose whether to calculate similarity to the predicted class of to the given class and whether to calculate also the predicted class or not.
10. Storage of intermediate results. When checked, intermediate stages of interpolation are saved into separate files. The number of interpolation step is added to file name.
11. Warning messages of missing layers. If checked, a message box containing layer number appears every time when data layer was not found. To continue, OK-button must be pressed or the calculations interrupted to correct names or location of data layers. Reasons for missing files may be an incorrect file name, an image layer does not extend to map sheet being calculated, or a file is not at specified data path.

A

B

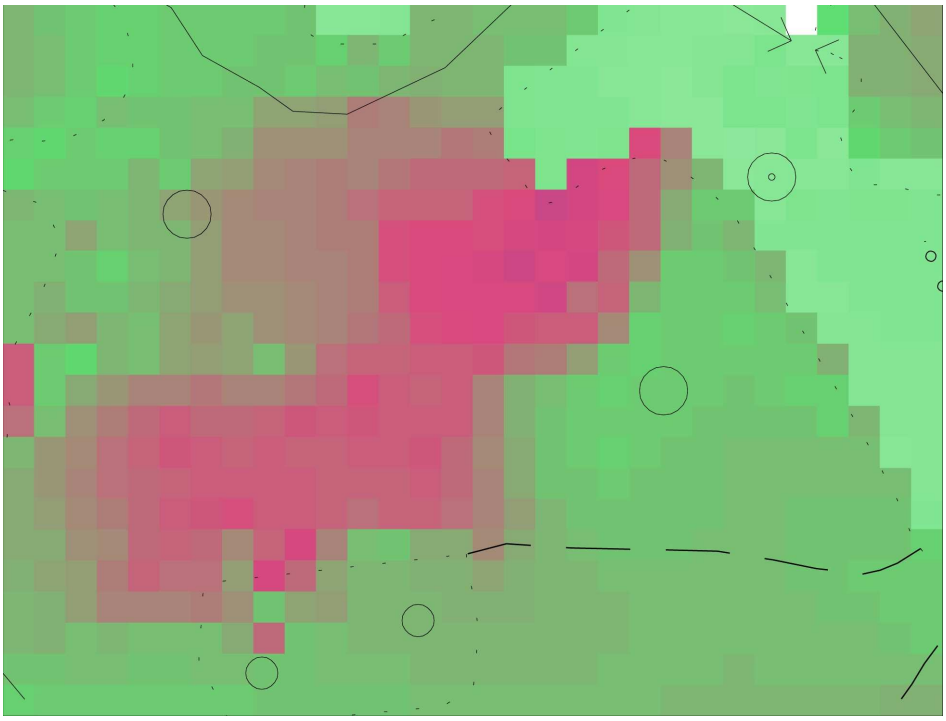

Figure 5.2. A — a clip of an orthophoto from 2005 with overlying basic map; B — similarity of exemplars from year 2001 to the orthophoto (similarity is the smallest at a new clearing).

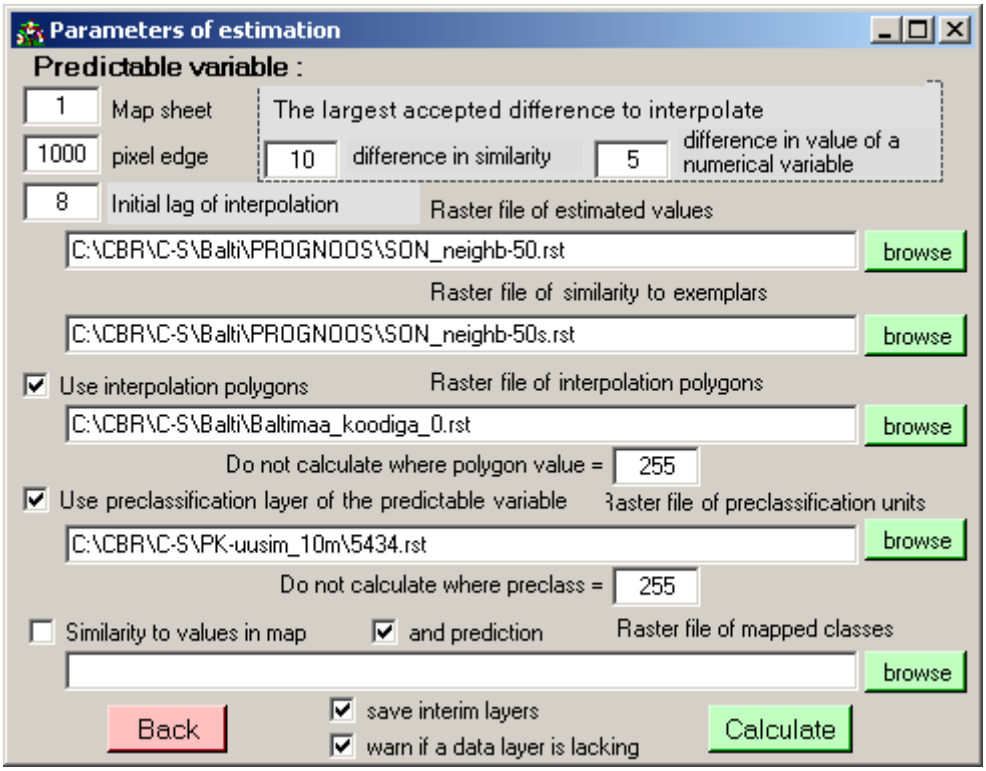

Figure 5.3. Dialog window of the map generation parameters showing all possible options.

5.6.3.1. Map interpolation

To speed up map generation, all predicted values in areas of similar pixels pixels are not calculated. Instead, the pixels to be calculated are selected at a regular interval (lag) — the intermediate values are interpolated. User can specify the initial lag, which is recommended to be a power of two: 2, 4, 8, 16, 32, or 64. During the first iteration, values are calculated only for pixels separated horizontally and vertically by the lag value. The lag is decreased by half at every subsequent iteration. Already calculated values are not recalculated. The intermediate pixel values are interpolated after every iteration. For nominal variables, the values are interpolated only if four bounding corner pixels belong to the same class. The values of a nominal variable are interpolated when bounding pixels differ less than the value set in map generation dialog window (*difference in similarity*; Figure 5.2). When interpolation polygons (as spatial pre-classifiers) are used, pixel values are interpolated only if all four bounding corner pixels are located within the same polygon.

Predictive map for find locations of *Epipactis palustris* in Otepää Nature Park in Estonia is presented as an example of iterative map generation (Fig. 5.4, 5.5).

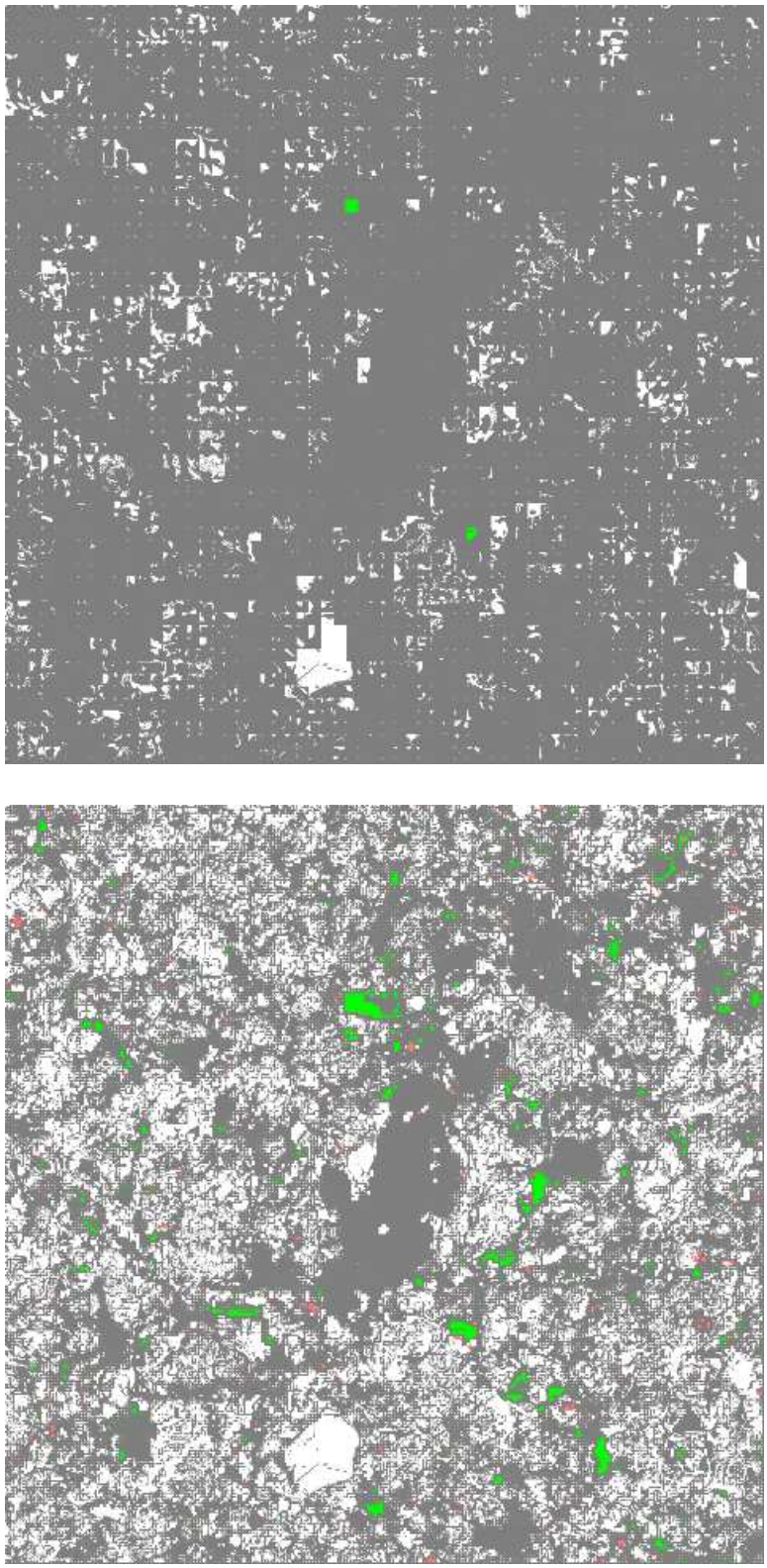

Figure 5.4. Generation of the predictive map of *Epipactis palustris* occurrence in Otepää Nature Park, Estonia. A — interpolation lag 16 pixels, B — lag 4 pixels. Gray — unspecified area, white — probable absence location, green — probable find location, scarlet — location similar to both absence and find locations. Calculation of predictions on water bodies and built-up areas was impeded by the pre-classification layer (base map).

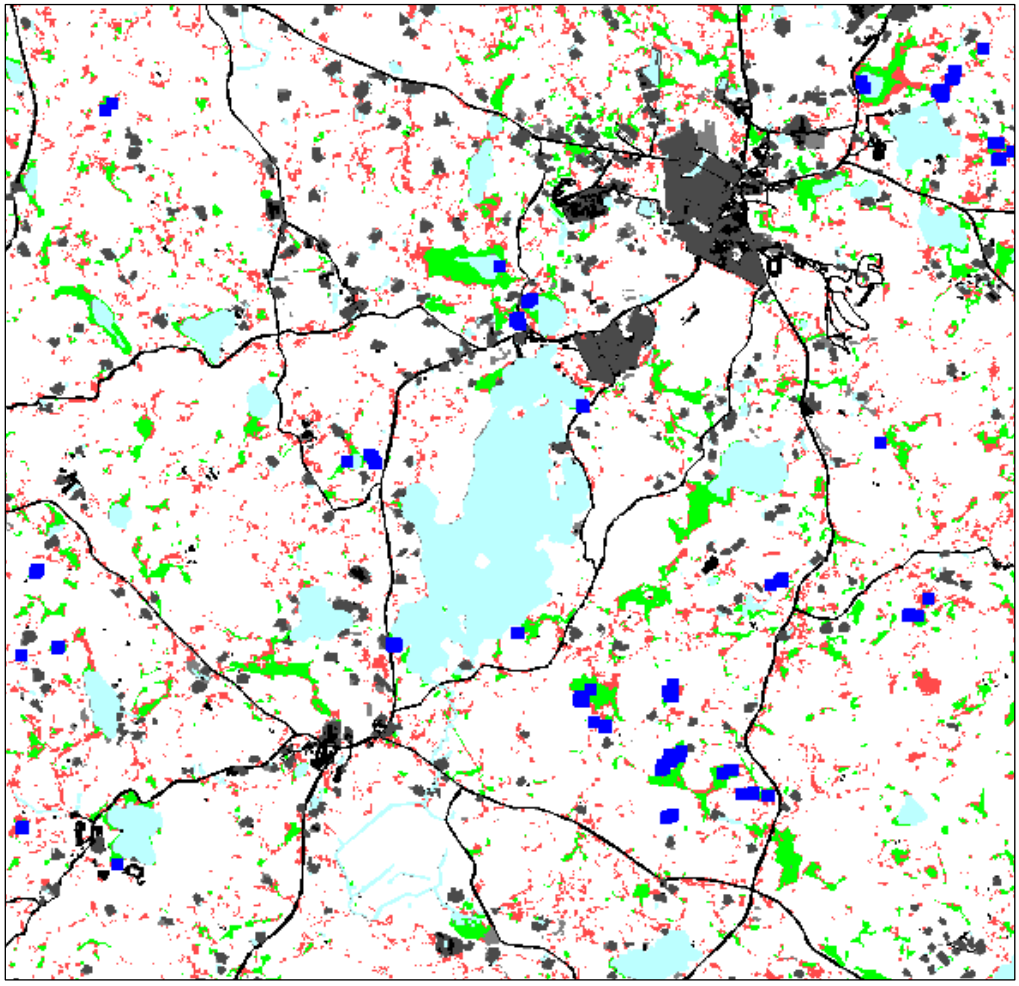

Figure 5.5. Predictive map of *Epipactis palustris* occurrence. Green — probable find location, white — probable absence location, scarlet — location similar to both absence and find locations, light blue — water bodies, dark blue — find locations, black — build-up areas and roads. Map edge 10 km.

References

Allouche, O., Tsoar, A., Kadmon, R. 2006. Assessing the accuracy of species distribution models: prevalence, kappa and the true skill statistic (TSS). *Journal of Applied Ecology* 43, 1223–1232.

Eesti põhikaardi 1: 10 000 digitaalkaardistuse juhend. 2000. Maa-amet. http://www.maaamet.ee/index.php?lang_id=1&page_id=64&menu_id=7. 20.02.2006.

Keskkonnaministri 5. veebruari 2004. a määrus nr 4. Geodeetilise süsteemi kehtestamine. Riigi Teataja Lisa, 18.02.2004, 17, 267.

Linder, M., Remm, K., Absalon, E. 2009. Tehisõppesüsteemi Pidevstuudium/CONSTUD kasutused. Mander, Ü.; Uuemaa, E.; Pae, T. (Toim.). Uurimusi eestikeelse geograafia 90. aastapäeval. Publicationes Instituti Geographici Universitatis Tartuensis 108 (52–62). Tartu: Tartu Ülikooli Kirjastus.

Linder, M., Remm, K., Proosa, H. 2008. The application of the concept of indicative neighbourhood on Landsat ETM+ images and orthophotos using circular and annulus kernels. *Ruas, A.; Gold, C. (Eds.). Headway in Spatial Data Handling. Proceedings of the 13th International Symposium on Spatial Data Handling in Montpellier, France.* 147–162. Springer.

Meiner, A. 1999. Eesti maakate. CORINE Land Cover projekti täitmine Eestis. Tallinn, lk. 22.

Paal, J. 1997. Eesti taimkatte kasvukohatüüpide klassifikatsioon. *Classification of Estonian Vegetation Site Types*. Tallinn, 297 lk.

Palo, A., Aunap, R., Mander, Ü. 2005. Predictive vegetation mapping based on soil and topographical data: A case study from Saare County, Estonia. *Journal for Nature Conservation* 13, 2–3, 197–211.

Remm, K. 2002. Otepää looduspargi taimkatte kasvukohatüüpide kaart. *In*: T. Frey (toim.), Eesti süsinikubilansi ökoloogiast ja ökonoomikast. Tartu, lk. 62–76.

Remm, K. 2004a. Case-based prediction for species and habitat mapping. *Ecological Modelling* 177(3-4), 259–281.

Remm, K. 2004b. Nähtuste paiknemissuhteid Eesti põhikaardi lehel 5434. *Publicationes Instituti Geographici Universitatis Tartuensis* 89, 209–236.

Remm, K., Linder, M. 2006. Otepää looduspargi taimkatte inventuuri esialgse kaardistus-etapi aruanne. Käsikiri Tartu Ülikooli geograafia instituudis ja Riikliku Looduskaitsekeskuse Põlva-Valga-Võru regiooni halduskeskuses.

Remm, K., Linder, M. 2007. Prognoosisüsteemi Pidevstuudium tutvustus. *Geodeet*, 34(58), 37–43.

Remm, K., Linder, M., Remm, L. 2009. Relative density of finds for assessing similarity-based maps of orchid occurrence. *Ecological Modelling*, 220(3), 294–309.

Remm, K., Remm, L. 2009. Similarity-based large-scale distribution mapping of orchids. Biodiversity and Conservation, 18(6), 1629–1647.

Remm, K., Remm, M. 2010. Geographical aspects of enterobiasis in Estonia. Health & Place, 16, 291–300.

Remm, M., Remm, K. 2008. Case-based estimation of the risk of enterobiasis. *Artificial Intelligence in Medicine*. 43(3), 167–177.

Tamm, T., Remm, K. 2009. Estimating the parameters of forest inventory using machine learning and the reduction of remote sensing features. International Journal of Applied Earth Observation and Geoinformation, 11(4), 290–297.

Appendix 1. Tested functionality and data bases used in testing the software Constud English version.

Function	Option 1	Index/Variable	Database	Date	Comments
Calculation of spatial indices	To the table of observations	Various	Indices_DB.mdb	1.06.09	Table Observations
	To the table of predictions	Various		1.06.09	Table PREDICTIONS
	To the table similarities	Various		1.06.09	Table ESTIMATED_OBS
	To the binary raster	Various	Indices_RASTER.mdb	1.06.09	With and without preclassifying polygons
Learning	Non-spatial data	Binomial	Non_spatial.mdb	20.05.08	SD from table, with and without weighting of cases, with and without fitting the sum of similarity
		Multinomial		20.05.08	
		Continuous		20.05.08	
	Spatial data	nominal	BM_areal_cat_2008.mdb	1.06.09	
		nominal classes separately		1.06.09	
Knowledge testing	Validation of the fit of the best set of weights	Binomial Multinomial Continuous	Non_spatial.mdb	20.05.08	
	Reselection of exemplars	Multinomial	BM_areal_cat_2008.mdb	1.06.09	
Knowledge application (results to database)	Calculation of spatial indices to the table of predictions	Various	BM_areal_cat_2008.mdb	21.05.08	
	Prediction	Binomial Multinomial Continuous	Non_spatial.mdb	21.05.08	ESTIMATED_1 ESTIMATED_2 ESTIMATED_3 ESTIMATED_4 ESTIMATED_5
		Multinomial	BM_areal_cat_2008.mdb	21.05.08	ESTIMATED_VALUES
		nominal classes separately	BM_areal_cat_2008.mdb	21.05.08	ESTIMATED_Complex
	Calculation of spatial indices to the table of estimated similarities	Various	BM_areal_cat_2008.mdb	21.05.08	SIM_TO_GIVEN
	Similarity to given category	nominal classes separately	BM_areal_cat_2008.mdb	21.05.08	SIM_TO_GIVEN
Knowledge application (results to raster)	Estimation of one variable	binominal	BM_areal_cat_2008.mdb		Forest
		multinomial	BM_areal_cat_2008.mdb	21.05.08	BM_ALL
		continuous	BM_areal_cat_2008.mdb	21.05.08	Coverage
	Complex estimation	nominal classes separately	BM_areal_cat_2008.mdb	21.05.08	
Application of non-square data layers			Balt_precipitations.mdb	26.11.08	

Appendix 2. VBA-module *Registration of data layers*

```
Option Explicit
Option Compare Database
Sub Andmekihtide_olemasolu()
'Märgib tabelisse olemasolevad andmekihid ja kärbib rst-failide nimed neljatäheliseks
 Dim dbs As Database
 Dim SQLT, SQLT1, f, fs, fc, f1 ' kõik on variandi tüüpi
 Dim i As Integer, n As Integer
 Dim laiend As String, failinimi, päring As String
 Dim KID As String, väli As String, directory As String, loba As String

Set dbs = CurrentDb 'loetakse samast andmebaasist
DoCmd.SetWarnings False 'lülitab välja hoiatused, et kas soovid ikka faili nime muuta
Set SQLT = dbs.OpenRecordset("D_layers") 'avab andmekihtide tabeli
SQLT.MoveLast
n = SQLT.RecordCount 'loendab kihtide arvu
SQLT.MoveFirst
For i = 1 To n
   KID = SQLT.Fields("KID") 'kihi identifikaator
   ' If KID > 180 Or KID < 156 Then GoTo Line10
   directory = SQLT.Fields("folder") 'kihi kataloog
   SQLT.MoveNext
   Set fs = CreateObject("Scripting.FileSystemObject")
   If fs.FolderExists(directory) = False Then GoTo Line10 'kui kataloogi ei ole
      Set f = fs.GetFolder(directory)
      Set fc = f.files
      For Each f1 In fc ' kõik selles kataloogis olevad failid
         failinimi = f1.Name
         laiend = Right(failinimi, 4) 'loeb faili laiendi
         If laiend = ".rst" Or laiend = ".RST" Or laiend = ".rdc" Or laiend = ".RDC" Then
            If Left(failinimi, 1) > 0 And Left(failinimi, 1) <= 9 Then
               väli = Left(failinimi, 1)
               päring = "Update Map_sheets set [" + KID + "]=true where NR=" + väli + ";"
               On Error Resume Next
               DoCmd.RunSQL päring
               If Left(failinimi, 1) > 0 And Left(failinimi, 1) <= 9 Then
                 f1.Name = Left(failinimi, 1) & laiend
               End If
            End If
         End If
Line9: Next
Line10: Next i
   SQLT.Close
End Sub
```