Toolbox for brain image recognition using artificial neural networks

Available from:
http://www.iba.muni.cz/index-en.php?pg=research--data-analysis-tools--annbrainrecog_toolbox

This toolbox is focused on brain image classification using artificial neural networks. The functions implemented in MATLAB® were invented during the experimentation, whose goal was to create a classification scheme that would be able to detect first-episode schizophrenia from images acquired from magnetic resonance device. The experiments resulted in two classification schemes. The first one uses single classifiers based on artificial neural networks (ANN) and the second scheme takes advantage of Random Subspace Ensemble Multi-layer Perceptron (RSE-MLP). The list of the implemented functions and their hierarchy are shown below.

Single ANNs scheme

The first scheme performs features selection using two-sample t-test. Then it uses one of the classifiers Multi-layer perceptron, Radian Basis Function Network and Learning Vector Quantization Neural Network that are trained and tested. The scheme is validated using leave-one-out cross-validation. The details are in the help of the functions.

The example for this first classification scheme can be tried by running the script *demo1*. It comprises 3 type of functions. The first one is for data preprocessing, where N 3D images with M voxels in NIfTI format are transformed to matrix of size $N \times M$. The second function is the classification scheme with particular type of ANN and the third one is function to plot the results.

RSE-MLP scheme

The second classification scheme employs RSE-MLP. It uses two-sample t-tests applied on each image variable to create a features pool, from which the feature vectors are taken to train the ensemble of MLPs. This ensemble is used to predict the class of the testing subjects and the results are evaluated. The details are in the help of the functions.

Similarly, script *demo2* can be tried to run the example for Random Subspace Ensemble Multi-layer Perceptron. There are four types of functions. The first one is to transform images to matrix the same way as in *demo1*. The second performs the learning of the classifiers and predicting the classes of the testing subjects. The third one evaluates the results and finally, there is a function to plot the results.

Requirements

Neural Network Toolbox™
Statistics and Machine Learning Toolbox™
NIfTI toolbox - name this toolbox 'NIFTI' and place it to the directory of this toolbox 'annBrainRecog_toolbox'

Note

Input data matrix X (or X_LOO) should include brain images in rows and features in columns. diseased must be in first rows and healthy controls must follow. Similarly, column vector y (or y_LOO) that contains classification classes of the brain images must be in this form: diseased must be denoted by 1 and must be in the first part of this vector and healthy controls must be denoted by 0 and must follow the diseased e.g. [1 1 1 0 0]', otherwise not all the functions work properly.

The rest of documentation is organized as follows:

i. Hierarchy of the implemented functions
ii. List of functions with brief description, list of inputs and list of outputs
iii. Full list of functions in alphabetical order
iv. DEMO1: Single Artificial Neural Network Classification Scheme
v. DEMO2: Random Subspace Ensemble Multi-layer Perceptron Classification Scheme

i. Hierarchy of the implemented functions

1. Single Artificial Neural Network Classification Scheme

Example in *demo1*

- → prepInTar
- → mlpClas
    - → createFP
    - → mlp
    - → evalANN
    - → resTab
- → rbfClas
    - → createFP
    - → evalANN
    - → resTab
- → lvqClas
    - → createFP
    - → som
    - → somLab
    - → lvq1
    - → lvq2_1
    - → evalANN
    - → resTab
- → plotANN

2. Random Subspace Ensemble Multi-layer Perceptron Classification Scheme

Example in *demo2*

- → prepInTar
- → rseClas
    - → createFP
    - → chooseFV
    - → mlpEnsClas
        - → mlp
- → rseEvalAVG
    - → rseEvalComb
        - → genComb
            - → genCombOverLim
        - → perfMeas
- → plotRSE

ii. List of functions with brief description and list of inputs and outputs

Name	Brief description	Input(s)	Output(s)
createFP	creates feature vector or feature pool	i, y, X_LOO, FP_size	pval, positions
evalANN	evaluates the predictions provided by artificial neural networks	predict, y	overall_accuracy, sensitivity, specificity
genComb	generates combinations of classifiers for evaluation	ens_size, ens_size_odd	combinations
genCombOverLim	generates limited combinations of classifiers for evaluation	ens_size, ens_size_odd	combinations
chooseFV	chooses feature vector from the feature pool	num_in, X_LOO, ts_LOO, feature_pool	X_LOO_ens, ts_LOO_ens, input_features_save
lvq1	performs learning vector quantization (LVQ1) algorithm	X_LOO, y_LOO, ts_LOO, num_neur, num_epoch_lvq1, som_net, labels	lvq1_net, predict
lvq2_1	performs learning vector quantization (LVQ2.1) algorithm	predict	X_LOO, y_LOO, ts_LOO, num_epoch_lvq2_1, lvq1_net
lvqClas	learning vector quantization classification scheme	X, y, num_in, num_neur, num_iter, lvq_type, num_epoch_lvq1, num_epoch_lvq2 _1, som_preproc, num_epoch_som	T1, T2
mlp	performs multi-layer perceptron classification	X_LOO, y_LOO, ts_LOO, num_neur	predict
mlpClas	multi-layer perceptron classification scheme	X, y, num_in, num_neur, num_iter	T
mlpEnsClas	random subspace ensemble multi-layer perceptron classification scheme	X_LOO_ens, y_LOO, ts_LOO_ens, num_neur, ens_size_wei_init	voting_predict_mlp, predict_mlp
perfMeas	Computes performance measures for random subspace ensemble based classifiers	predict_rse_clas, ens_size_odd, y, combinations	oa_mean, sen_mean, spe_mean

plotANN	plots result of single artificial neural network classification scheme	T	graph
plotRSE	plots results for random subspace ensemble classification	performance_measure_mlp, FP_size, num_in, performance_measure_svm	graph
prepInTar	prepares input matrix and target vector	n, mask	X, y
rbfClas	radial basis function neural network classification scheme	X, y, num_in, max_num_neur, goal, spread	T, num_neur_real
resTab	creates table with results for classification using artificial neural networks (ANN)	name_of_classifier, overall_accuracy, sensitivity, specificity, num_neur, num_iter, num_in, spread	T
rseClas	random subspace ensemble classification scheme	X, y, ens_size, num_in, num_neur, FP_size, ens_size_wei_init, num_iter, svm_comp	predict_mlp_iter, feature_pool_save_iter, pvals_iter, predict_mlp_save_iter, input_features_save_iter, predict_svm_iter
rseEvalAVG	computes average of random subspace ensemble classification evaluation	y, predict_mlp_iter, ens_size, num_iter, num_in, predict_svm_iter	oa_avg_mlp, sen_avg_mlp, spe_avg_mlp, oa_avg_svm, sen_avg_svm, spe_avg_svm
rseEvalComb	computes mean performance measures for random subspace ensemble classification scheme over all combinations of classifiers in ensemble	y, predict_mlp, ens_size, num_in, predict_svm	oa_mean_mlp, sen_mean_mlp, spe_mean_mlp, oa_mean_svm, sen_mean_svm, spe_mean_svm
som	self-organizing map neurons preprocessing	X_LOO, num_neur, num_epoch_som	netsom
somLab	self-organizing map neurons labeling	X_LOO, y, i, num_neur, som_net	labels

iii. Full list of functions in alphabetical order

chooseFV - chooses feature vector from the feature pool

[X_LOO_ens, ts_LOO_ens, input_features_save] = chooseFV(num_in, X_LOO, ts_LOO, feature_pool)

Description:

The function 'chooseFV' randomly chooses a subspace of the feature pool to create the features vector. It takes this subspace from input matrix 'X_LOO' and testing sample 'ts_LOO' to prepare the information for training and testing of one classifier from the ensemble.

Input variables:

num_in ... scalar, defines length of the feature vector
X_LOO ... matrix, contains preprocessed brain images in rows and features in columns, testing subject is left out
ts_LOO ... vector, contains the testing sample
feature_pool ... vector with indices that define the positions of the feature pool in the image vectors

Output variables:

X_LOO_ens ... matrix, contains the chosen feature vectors from feature pool to train the classifiers
ts_LOO_ens ... vector, contains corresponding features for testing subject
input_features_save ... vector, contains the positions of the chosen features in the image vector

Algorithm:

1. Random number generator is seed based on current time to ensure the randomness.
2. The n (= 'num_in') random numbers from feature_pool are chosen.
3. Only columns of 'X_LOO' matrix and elements of 'ts_LOO' vector corresponding to the chosen feature vector are preserved, the others are removed.

***createFP* - creates feature vector or feature pool**

[pval, positions] = createFP(i, y, X_LOO, FP_size)

Description:

The function 'createFP' creates the feature vector that serves as an input to classification algorithms or it creates so-called feature pool. This function applies two-sample t-test on the X_LOO matrix. It returns the p-value and the position of each voxel in the X_LOO matrix. The p-values help to choose acquired amount of the most significant voxels that are used to create the feature vector or the feature pool. This feature pool is later used as a bag from which the features (voxels) are chosen to train the ensemble of classifiers.

Input variables:

i ... index of current iteration of leave-one-out cross-validation loop
X_LOO ... matrix, contains brain images in rows (testing subject is left out) and features in columns; diseased must be in first rows and healthy controls must follow
y ... vector, contains classification classes of the brain images, diseased must be denoted by 1 and must be in the first part of this vector!!! e.g. [1 1 1 0 0]'
FP_size ... scalar, defines the length of the feature vector (for single classifier approach) or size of the feature pool (for ensemble approach)

Output variables:

pval ... vector, contains the p-values of the features (voxels)
positions ... vector, contains the positions of the chosen features in image vector

Algorithm:

1. Number of patients (thus also number of healthy controls) is allocated.
2. Two-sample t-test is applied on the 'X_LOO' matrix.
3. P-values are sorted in ascending order.
4. The positions and p-values of most significant voxels are returned.

Notes:

Requires Statistics and Machine Learning Toolbox™

evalANN - evaluates the predictions provided by artificial neural networks

[overall_accuracy, sensitivity, specificity] = evalANN(predict, y)

Description:

The function 'evalANN' takes matrix of predicted classes for testing subjects, compare it with targets and computes the performance measures, which are overall accuracy, sensitivity and specificity. This function can be used to evaluate predictions performed by functions mlp, rbf, lvq1 and lvq2_1.

Input variables:

y ... vector, contains classification classes of the brain images
predict ... matrix, contains the classes of the testing subjects predicted by one of the artificial neural networks

Output variables:

overall_accuracy ... scalar, the value of overall accuracy
sensitivity ... scalar, the value of sensitivity
specificity ... scalar, the value of specificity

Algorithm:

1. The matrix with predictions is transformed to vector.
2. The numbers of well- and mis- classified subjects are computed.
3. The overall accuracy, sensitivity and specificity are computed.

genComb - generates combinations of classifiers for evaluation

[combinations] = genComb(ens_size, ens_size_odd)

Description:

The function 'genComb' generate combinations of indices that are later used to take subspace of predictions. These subspaces are supposed to be used for voting during evaluation of random subspace ensemble classifiers.

Input variables:

ens_size ... scalar, defines number of classifiers in ensemble, must be an odd number
ens_size_odd ... scalar, contains odd numbers of predictions that are used to vote

Output variable:

combinations ... matrix, contains generated combinations of indeces that can later take subspace of predictions and use them to vote for final class of testing sample

Algorithm:

```
1. Compute number of all possible combinations of predictions, when 'ens_size_odd' predictions
among 'ens_size' are used to vote.
IF the number of the combinations is <= 10000 THEN
   2. Generate all possible combinations of predictions.
ELSE
   3. Call function 'genCombOverLim' to return 10000 random combinations.
```

genCombOverLim - generates limited combinations of classifiers for evaluation

[combinations] = genCombOverLim(ens_size, ens_size_odd)

Description:

The function 'genCombOverLim' generate 10000 random combinations of indices that are later used to take subspace of predictions. These subspaces are supposed to be used for voting during evaluation of random subspace ensemble classifiers. The combination limit helps to decrease the computational time and it is still big enough to keep robustness of the evaluation.

Input variables:

ens_size ... scalar defines number of classifiers in ensemble, must be an odd number
ensemble_size_odd ... vector, contains odd numbers of predictions that can be used to vote

Output variable:

combinations ... matrix, contains generated combinations of indices that can later take subspace of predictions and use them to vote for final class of testing sample

Algorithm:

1. Random number generator is seed based on current time to ensure the randomness. Number of combination is set to 10000. And a vector to store the combinations is defined.
2. Loop over number of combinations starts here:
   3. 10000 random combinations are generated.
4. Loop over number of combinations ends here.
5. Duplicities among random combinations are deleted.
6. WHILE the number of combinations is less than 10000, generate more combinations and simultaneously check for duplicities.

lvq1 – performs learning vector quantization (LVQ1) algorithm

[lvq1_net, predict] = lvq1(X_LOO, y_LOO, ts_LOO, num_neur, num_epoch_lvq1, som_net, labels)

Description:

The function 'lvq1' takes advantage of the Neural Network Toolbox™ and use its functions to train the LVQ network by LVQ1 algorithm [1] and to predict the class of the testing subject. The LVQ can take advantage of SOM for weight preprocessing, but it also can initialize the weights randomly. The adapted network can be stored and passed to another function 'LVQ2_1' to do additional training.

Input variables:

X_LOO ... matrix, includes preprocessed brain images in rows and features in columns, testing subject is left out
y_LOO ... vector containing classes of the training subjects
ts_LOO ... vector, contains the testing sample image
num_neur ... scalar, defines number of neurons
num_epoch_lvq1 ... scalar, number of LVQ1 learning epochs
som_net ... object, trained SOM
labels ... matrix, contains the labels of the neurons preprocessed by SOM

Output variables:

lvq1_net ... object, LVQ network trained by LVQ1 algorithm predict ... scalar, the class of the testing subject predicted by LVQ network adapted on the 'X_LOO' matrix

Algorithm:

```
1. LVQ network is created using 'lvqnet' function from Neural Network Toolbox™ and configured.
IF 'som_net' parameter was input to the function THEN
  2. The weights are initialized using information from 'som_net'.
  3. The neurons are labeled with aid of parameter 'labels'.
ELSE the weights are initialized randomly.
4. Number of epochs is set.
5. Window showing the status of the training is turned off.
6. The LVQ network is trained using LVQ1 algorithm.
7. The class of the testing subject is predicted.
```

References:

[1] T. Kohonen, „The self-organizing map", Proc. IEEE, Vol. 78, No. 9, pp. 1464–1480, 1990.

Notes:

Requires Neural Network Toolbox™

lvq2_1 – performs learning vector quantization (LVQ2.1) algorithm

[predict] = lvq2_1(X_LOO, y_LOO, ts_LOO, num_epoch_lvq2_1, lvq1_net)

Description:

The function 'lvq2_1' takes advantage of the Neural Network Toolbox™ and use its functions to do additional training of the LVQ network using LVQ2.1 algorithm [1] after the LVQ1 training was done and to predict the class of the testing subject.

Input variables:

X_LOO ... matrix, contains preprocessed brain images in rows and features in columns, testing subject is left out
y_LOO ... vector containing classes of the training subjects
ts_LOO ... vector, contains the testing sample image
num_epoch_lvq2_1 ... scalar, number of training epochs
lvq1_net ... object, LVQ network trained using LVQ1 algorithm

Output variable:

predict ... scalar, the class of the testing subject predicted by LVQ network adapted on the X_LOO matrix

Algorithm:

1. Learning algorithm is set to LVQ2.1.

2. Number of epochs is set.
3. The LVQ network is trained using LVQ2.1 algorithm.
4. The class of the testing subject is predicted.

References:

[1] T. Kohonen, „The self-organizing map", Proc. IEEE, Vol. 78, No. 9, pp. 1464–1480, 1990.

Notes:

Requires Neural Network Toolbox™

lvqClas - learning vector quantization classification scheme

[T1, T2] = lvqClas(X, y, num_in, num_neur, num_iter, lvq_type, num_epoch_lvq1, num_epoch_lvq2_1, som_preproc, num_epoch_som)

Description:

The function 'lvqClas' mainly employs LVQ network for brain image classification. It contains steps for feature selection, neuron weights initialization using Kohonen self-organizing map (optional, but recommended step by author of these algorithms [1]), training of LVQ network by LVQ1 algorithm and additional optional training by LVQ2.1. Last steps are performance evaluation and creation of a overview table(s) with results. Feature selection is voxel-wise and it is performed using two-sample t-test. Adaptation of networks and classification of the testing subject is performed using functions from Neural Network Toolbox™. The classification scheme is validated using leave-one-out cross-validation (LOO-CV). This function takes input data from matrix X and target data from vector y. Parameters that can be adjusted by user are first those that can set the architecture of the network (number of inputs and neurons) and second those that can define number of epochs of SOM and LVQ algorithms. Since the LVQ's or SOM's weights are initialized randomly, the classification results are dependent on it, thus there is another parameter 'num_iter' that defines number of repetitions of the classification scheme. Repeating of the experiment could be used to gain more robust results by averaging the performance measures (overall accuracy, sensitivity and specificity).

Input variables:

X ... matrix, contains brain images in rows and features in columns; diseased must be in first rows and healthy controls must follow

y ... vector containing classification classes of the brain images, diseased must be denoted by 1 and must be in the first part of this vector!!! e.g. [1 1 1 0 0]'

num_in ... scalar defining a length of a feature vector, that is an input to the classification model

num_neur ... scalar, defines number of neurons

num_iter ... scalar, number of repetitions of the classification process to gain more robust results

lvq_type ... string, defines if 'LVQ1' algorithm is used or both 'LVQ1' and 'LVQ2.1' algorithms are used
num_epoch_lvq1 ... scalar, number of LVQ1 learning epochs
num_epoch_lvq2_1 ... scalar, number of LVQ2.1 learning epochs
som_preproc ... logical, defines if SOM weight preprocessing is used
num_epoch_som ... scalar, number of SOM learning epochs

Output variables:

T1 ... table, includes information about applied model (name of the classifier (LVQ1), number of neurons and length of the feature vector) and values of performance measures (overall accuracy, sensitivity and specificity)
T2 ... table, used only if the LVQ2.1 is applied, it includes the same parameters as T1 only in the column for name of classifier is LVQ2.1

Algorithm:

```
1. The parameters n - number of inputs, oe - expected output and matrices to store predictions from LVQ1 and LVQ2.1 are allocated.
2. LOO-CV loop starts here:
  3. Testing subject is left out.
  4. Feature selection based on two-sample t-tests is done by function 'createFP'.
  IF som_preproc == 1 THEN
     5. The function 'som' is called. This function takes input matrix and initialize the weights of the neurons. For details see description of 'som' function in this documentation.
     6. A class name is allocated to each neuron by 'somLab' function.
  7. The LVQ network is trained by LVQ1 algorithm with aid of function 'lvq1' and the testing sample class is predicted.
  IF lvq_type == 'LVQ2.1'
     8. LVQ2.1 algorithm is applied to train the network with aid of function 'LVQ2_1'.
9. LOO-CV ends here.
7. Variables to store the performance measures are declared.
8. Function 'evalANN' is called. This function takes matrix with predicted classes of the testing samples and target vector and it returns overall accuracy, sensitivity and specificity. If LVQ2.1 algorithm is used, the performance meassures are computed for the predictions provided by this additionally trained network.
9. Function 'resTab' is called. It returns table(s) with results. See description of 'Output variables'.
```

References:

[1] T. Kohonen, „The self-organizing map", Proc. IEEE, Vol. 78, No. 9, pp. 1464–1480, 1990.

Notes:

Requires Neural Network Toolbox™ and Statistics and Machine Learning Toolbox™

mlp – performs multi-layer perceptron classification

[predict] = mlp(X_LOO, y_LOO, ts_LOO, num_neur)

Description:

The function 'mlp' takes advantage of the Neural Network Toolbox™ and use its functions to train the MLP and to predict the class of the testing subject. In this function, the basic settings of MLP is adjusted in order to train the network fast and perform well on neuroimaging data. For further information about the parameters used here see Matlab documentation: http://www.mathworks.com/help/nnet/index.html

Input variables:

X_LOO ... matrix, contains preprocessed brain images in rows and features in columns, testing subject is left out
y_LOO ... vector containing classes of the training subjects
ts_LOO ... vector, contains the testing sample image
num_neur ... scalar or vector containing number of neurons, where each element of this vector determines one hidden layer

Output variable:

predict ... scalar, the class of the testing subject predicted by multi-layer perceptron adapted on the 'X_LOO' matrix

Algorithm:

1. MLP network is created using 'patternnet' function from Neural Network Toolbox™.
2. All data in matrix 'X_LOO' are defined as training data.
3. Training function is defined scaled conjugate gradient backpropagation.
4. The function that is minimized during training phase is defined as cross-entropy.
5. Window showing the status of the training is turned off.
6. The MLP is trained.
7. The class of the testing subject is predicted.

Notes:

Requires Neural Network Toolbox™

mlpClas - multi-layer perceptron classification scheme

[T] = mlpClas(X, y, num_in, num_neur, num_iter)

Description:

The function 'mlpClas' employs multi-layer perceptron for brain image classification. It contains steps for feature selection, classifier adaptation, classification of a testing subjects, performance evaluation and creation of an overview table with results. Feature selection is voxel-wise and it is performed using two-sample t-test. Adaptation and classification of the testing subject is performed using functions from Neural

Network Toolbox™. The classification scheme is validated using leave-one-out cross-validation (LOO-CV). This function takes input data from matrix X and target data from vector y. Number of inputs and neurons are parameters of this function that can be adjusted by user. Since the multi-layer perceptron's weights are initialized randomly, the classification results are affected by it, thus there is another parameter 'num_iter' that defines number of repetitions of the classification scheme. Repeating of the experiment could be used to gain more robust results by averaging the performance measures (overall accuracy, sensitivity and specificity).

Input variables:

X ... matrix, contains brain images in rows and features in columns; diseased must be in first rows and healthy controls must follow
y ... vector, contains classification classes of the brain images, diseased must be denoted by 1 and must be in the first part of this vector!!! e.g. [1 1 1 0 0]'
num_in ... scalar, defines a length of a feature vector, that is an input to the classification model
num_neur ... scalar or vector, contains number of neurons, where each element of this vector determines one hidden layer
num_iter ... scalar, number of repetitions of the classification process to gain more robust results

Output variable:

T ... table, that includes information about applied model (name of the classifier (MLP), number of neurons and length of the feature vector) and values of performance measures (overall accuracy, sensitivity and specificity)

Algorithm:

1. The parameters n - number of inputs, oe - expected output are defined and matrix to store prediction is allocated.
2. LOO-CV loop starts here:
  3. Testing subject is left out.
  4. Feature selection based on two-sample t-tests is done by function 'createFP'.
  5. Function 'mlp' is called. This function takes input matrix, target vector, testing subject and information about MLP architecture and returns the predicted class (see in documentation function 'mlp').
6. LOO-CV ends here.
7. Variables to store the performance measures are allocated.
8. Function 'evalANN' is called. This function takes matrix with predicted classes of the testing samples and target vector and it returns overall accuracy, sensitivity and specificity.
9. Function 'resTab' is called. It returns table with results. See description of 'Output variable'.

Notes:

Requires Neural Network Toolbox™ and Statistics and Machine Learning Toolbox™

mlpEnsClas - random subspace ensemble multi-layer perceptron classification scheme

[voting_predict_mlp, predict_mlp] = mlpEnsClas(X_LOO_ens, y_LOO, ts_LOO_ens, num_neur, ens_size_wei_init)

The function 'mlpEnsClas' performs repeating MLP training and testing. Since the weights are initialized randomly, in this function the MLP training is repeated 'ens_size_wei_init' times to gain multiple predictions of the testing sample class. These predictions are used to vote for the final testing sample class.

Input variables:

X_LOO_ens ... matrix, contains the chosen feature vectors from the feature pool to train the classifiers
y_LOO ... vector, includes class the subjects belong to
ts_LOO_ens ... vector, contains corresponding features for testing subject
num_neur ... scalar or vector containing number of neurons, where each element of this vector determines one hidden layer
ens_size_wei_init ... scalar, odd number, defines the number of MLPs that are trained and tested here and used to predict the class of the testing sample

Output variables:

voting_predict_mlp ... scalar, the class of the testing subject predicted by multi-layer perceptron adapted on the 'X_LOO' matrix
predict_mlp ... matrix, stores predictions of each MLP used in this function

Algorithm:

1. A matrix 'predict_mlp' is declared.
2. Loop for training the MLPs starts here:
   3. The function 'mlp' is called to train MLP and predict class of the testing subject.
4. Loop for training the MLPs ends here.
5. The predictions of all the MLPs are used to vote the final class of the testing subject.

Notes:

Requires Neural Network Toolbox™

perfMeas - computes performance measures for random subspace ensemble based classifiers

[oa_mean, sen_mean, spe_mean] = perfMeas(predict_rse_clas, ens_size_odd, y, combinations)

Description:

The function 'perfMeas' computes performance measures (overall accuracy, sensitivity and specificity) for each combination of predictions generated by function 'genComb'. The results are then averaged and thus more robust estimation of performance measures is computed.

Input variables:

predict_rse_clas ... matrix, contains the testing subject class prediction of each classifier in ensemble
ens_size_odd ... scalar, number of voting classifiers
y ... vector containing classes the subjects belong to, diseased must be denoted by 1 and must be in the first part of this vector!!! e.g. [1 1 1 0 0]'
combinations ... matrix, contains generated combinations of indices that take subspace of predictions and use them to vote for final class of testing sample

Output variables:

oa_mean ... scalar, contains overall accuracy for defined feature vector length and for defined odd number of classifiers in ensemble
sen_mean ... scalar, contains sensitivity for defined feature vector length and for defined odd number of classifiers in ensemble
spe_mean ... scalar, contains specificity for defined feature vector length and for defined odd number of classifiers in ensemble

Algorithm:

```
1. 3D matrix to save confusion matrices for each combination of predictions is declared.
2. Loop over prediction combinations number starts here:
  3. The voting of all possible combinations of predictions is performed.
4. Loop over prediction combinations number ends here.
5. Performance measures (overall accuracy, sensitivity and specificity) for all combinations of predictions
are computed.
6. Mean performance measures are computed over all combinations of prediction.
```

plotANN - plots result of single artificial neural network classification scheme

plotANN(T)

Description:

The function 'plotANN' makes graph with the results from classifications of brain images with aid of artificial neural networks. It takes information from table with results and draw point values or boxplots of three performance measures overall accuracy, sensitivity and specificity. The graphs are fully described. The layout of the figure can be further adjusted by investigator in the figure window.

Input variable:

T ... table, contains information about applied classification model (name of the classifier, number of neurons and length of the feature vector, spread - in case of RBF network) and values of performance measures (overall accuracy, sensitivity and specificity)

Output:

graph ... the output of this function is a window with drawn graph

Algorithm:

```
IF table 'T' includes only one row with result THEN
  1. Plot point values of the performance measures.
  2. Set labels, colors and title.
ELSE
  3. Plot boxplots of the performance measures.
  4. Set labels, colors and title.
```

plotRSE - plots results for random subspace ensemble classification

plotRSE(performance_measure_mlp, FP_size, num_in, performance_measure_svm)

Description:

The function 'plotRSE' plots a figure for RSE-based classification. Using this figure, the trends of ensemble learning can be observed and RSE-MLP and RSE-SVM can be compared easily. Function is prepared only for 3 or less sizes of input vectors not to make the figure difficult to read.

Input variables:

performance_measure_mlp ... matrix, contains results of some performance measure for RSE-MLP for all explored lengths of feature vector and all odd numbers of voting classifiers
FP_size ... scalar, defines the size of the feature pool
num_in ... scalar or vector, defines lengths of feature vectors
performance_measure_svm ... matrix, contains performance measures for RSE-SVM for all explored lengths feature vector and all odd numbers of voting classifiers

Output:

Graph that shows results for RSE-MLP and eventually compare it with RSE-SVM

Algorithm:

```
1. Vectors of symbols (for MLP) and colors used in chart are defined.
IF nargin > 3 i.e. SVM performance measures are input of this function.
   2. Vector of symbols (for SVM) used in chart are defined.
3. Matrix 'legend_text' to store information used in legend is allocated.
4. Loop over different feature vectors lengths starts here:
   5. Performance measure for RSE-MLP and defined feature vector length is plotted.
   6. Legend text of this plot is generated and saved.
IF nargin > 3 i.e. SVM performance measures are input of this function.
   7. Performance measure for RSE-SVM and defined feature vector length is plotted.
   8. Legend text of this plot is generated and saved.
9. Loop over different feature vectors lengths ends here.
10. The title, labels and values of axes x and y and legend are added to the plot.
```

prepInTar - prepares input matrix and target vector

[X, y] = prepInTar(n, mask)

Description:

The function 'prepInTar' prepares example NIfTI image data format to the required form i.e. matrix X, where rows are subjects and columns are variables and target vector y, where first part of this vector are ones (denotes patients) and are second part are zeros (denotes healthy controls). The example dataset contains pictures of circles and triangles. During the preparation of the required data format a mask that was prepared to delete background of the pictures is applied. This function was prepared for DEMOs of this toolbox, however it can be used for any other dataset. The images must be located in example_data directory and their names and class must be in index.xls in this directory (see this file to understand its structure).

Input variable:

n ... number of images

Output variables:

X ... matrix, contains brain images in rows and features in columns
y ... vector, contains classification classes of the brain images

Algorithm:

1. The directory is changed to 'example_data' directory that is in the toolbox. This directory must include images in nifty format and excel document with names of the brain images in column A (from A2 to AN) and labels of the brain images in column B (from B2 to BN).

2. Excel document i read and names of images and labels are stored.
3. The directory is changed to NIfTI directory, that must be in the toolbox.
4. Images are loaded, transformed (the mask is applied) and saved to the matrix X.

Note:

Requires NIfTI toolbox.

rbfClas - radial basis function neural network classification scheme

[T, num_neur_real] = rbfClas(X, y, num_in, max_num_neur, goal, spread)

Description:

The function 'rbfClas' takes advantage of RBF network for brain image classification. It contains steps for feature selection, classifier adaptation, classification of a testing subject, performance evaluation and creation of an overview table with results. Feature selection is voxel-wise and it is performed using two-sample t-test. Training of RBF network [1] and classification of the testing subject is performed using functions from Neural Network Toolbox™. The classification scheme is validated using leave-one-out cross-validation (LOO-CV). This function takes input data from matrix X and target data from vector y. Number of inputs, neurons, goal and spread are parameters of this function that can be adjusted by user.

Input variables:

X ... matrix, contains brain images in rows and features in columns; diseased must be in first rows and healthy controls must follow
y ... vector containing classification classes of the brain images, diseased must be denoted by 1 and must be in the first part of this vector!!! e.g. [1 1 1 0 0]'
num_in ... scalar defining a length of a feature vector, that is an input to the classification model
max_num_neur ... scalar, defines maximal number of neurons
goal ... scalar defining the value of error function, when the adaptation should be stopped; when goal is zero, then 'max_num_neur' = final number of neurons
spread ... scalar, defines the size of the area around the RBF neurons with influence on the output of this RBF neuron

Output variables:

T ... table, contains information about applied model (name of the classifier (RBF), number of neurons, length of the feature vector and spread) and values of performance measures (overall accuracy, sensitivity and specificity)
num_neur_real ... real number of neurons, when goal is not equal zero

Algorithm:

1. The parameters n - number of inputs, oe - expected output are defined and matrix to store prediction is allocated.
2. LOO-CV loop starts here:
  3. Testing subject is left out.

4. Feature selection based on two-sample t-tests is done by function 'createFP'.
5. Function 'newrb' from 'Neural Network Toolbox' is called. This function takes input matrix, target vector, goal, spread and maximal number of neurons and train the RBF network (see the documentation of 'Neural Network Toolbox').
6. The class of the testing subject is predicted and stored.
7. LOO-CV ends here.
8. Variables to store the performance measures are allocated.
9. Function 'evalANN' is called. This function takes matrix with predicted classes of the testing samples and target vector and it returns overall accuracy, sensitivity and specificity.
10. Function 'resTab' is called. It returns table with results. See description of 'Output variable'.

References:

[1] S. Chen, C. F. N. Cowan, a P. M. Grant, „Orthogonal least squares learning algorithm for radial basis function networks", Vol. 2, No. 2, pp. 302–309, 1991.

Notes:

Requires Neural Network Toolbox™ and Statistics and Machine Learning Toolbox™

resTab – creates table with results for classification using artificial neural networks (ANN)

[T] = resTab(name_of_classifier, overall_accuracy, sensitivity, specificity, num_neur, num_iter, num_in, spread)

Description:

The function 'resTab' creates a table to summarize parameters used for ANN-based brain image classification and the results.

Input variables:

name_of_classifier ... character, name of the ANN classifier
overall_accuracy ... vector, the n values of overall accuracies (n='num_iter')
sensitivity ... vector, the n values of sensitivities (n='num_iter')
specificity ... vector, the n values of specificities (n='num_iter')
num_in ... scalar, defines a length of a feature vector, that is an input to the classification model
num_neur ... scalar or vector containing number of neurons, where each element of this vector determines one hidden layer
num_iter ... scalar, number of repetitions of the classification process to gain more robust results
spread ... scalar, defines the size of the area around the RBF neurons with influence on the output of this RBF neuron

Output variable:

T ... table, includes information about applied classification model (name of the classifier, number of neurons and length of the feature vector, spread - in case of RBF network) and values of performance measures (overall accuracy, sensitivity and specificity)

Algorithm:

```
1. Prepare future columns of the table (number of neurons, number of inputs and performance measures).
IF nargin < 8 i.e. spread is not an input THEN
  2. Create table
ELSE
  3. Define column for spread.
  4. Create table.
```

rseClas - random subspace ensemble classification scheme

[predict_mlp_iter, feature_pool_save_iter, pvals_iter, predict_mlp_save_iter, input_features_save_iter, predict_svm_iter] = rseClas(X, y, ens_size, num_in, num_neur, FP_size, ens_size_wei_init, num_iter, svm_comp)

Description:

The function 'rseComp' performs computation of Random Subspace Ensemble Multi-layer Perceptron. The classification scheme is validated with aid of leave-one-out cross-validation. It takes input data from matrix X and target data from vector y. The experiment settings are adjusted by input parameters. The RSE-MLP can be compared to Random Subspace Ensemble Support Vector Machines. The robustness of the results can be further ensured by repetition of this experiment using parameter num_iter.

Input variables:

X ... matrix, contains brain images in rows and features in columns; diseased must be in first rows and healthy controls must follow

y ... vector containing classification classes of the brain images, diseased must be denoted by 1 and must be in the first part of this vector!!! e.g. [1 1 1 0 0]'

ens_size ... scalar, defines number of classifiers in ensemble, must be an odd number

num_in ... scalar or vector, defines lengths of feature vectors

num_neur ... scalar or vector containing number of neurons, where each element of this vector determines one hidden layer

FP_size ... scalar, defines the size of the feature pool

ens_size_wei_init ... scalar, defines the number of the MLPs that votes for class in first voting, must be an odd number

num_iter ... scalar, number of repetitions of the experiment to gain robust results

svm_comp ... logical, defines if the comparison with SVM is performed (1) or not (0)

Output variables:

predict_mlp_iter ... matrix with prediction of the testing subject class by MLP for each iteration of LOO, each classifier in ensemble, each length of the feature vector and each experiment repetition

feature_pool_save_iter ... matrix, contains indices of voxels that form the feature pool for each iteration of LOO and each experiment repetition

pvals_iter ... matrix, contains p-values of voxels that form the feature pool for each iteration of LOO and each experiment repetition

predict_mlp_save_iter ... matrix with predictions of the testing subject class by each MLP trained during the experiment, before the first voting

input_features_save_iter ... cell array, stores the randomly chose subspace ensembles for each iteration of LOO, each classifier in ensemble, each length of the feature vector and each experiment repetition

predict_svm_iter ... matrix with prediction of the testing subject class by SVM for each iteration of LOO, each classifier in ensemble, each length of the feature vector and each experiment repetition

Algorithm:

```
1. The declaration of parameters n - number of subjects, eo – expected outputs and matrices to store the results.
2. Loop for the experiment repetition starts here:
  3. Leave-one-out cross-validation loop starts here:
    4. The testing subject is left out from the data matrix and target vector.
    5. Feature pool of defined size is created using function 'createFP'.
    6. Loop for computing over defined lengths of feature vectors starts here:
      7. Loop for computing over the number of classifiers in ensemble starts here:
        8. The feature vector of defined size is chosen using function 'chooseFV'.
        9. The function 'mlpEnsClas' is called. This function returns class of the testing subject based on classification by MLP. For details see description of this function.
        IF SVM comparison is required:
          10. SVM is trained and the classification of the testing subject is performed by functions from Statistics and Machine Learning Toolbox™.
      11. Loop for counting over the number of classifiers in ensemble ends here.
        12. Loop for counting over defined lengths of feature vectors ends here.
  13. Leave-one-out cross-validation loop ends here.
  14. The predicted classes of testing subjects are stored.
15. Loop for experiment repetition ends here.
```

Notes:

Requires Neural Network Toolbox™ and Statistics and Machine Learning Toolbox™

To understand the structure of output matrices, please, see comments in the code in the place, where these matrices are allocated

rseEvalAVG - computes average of random subspace ensemble classification evaluation

[oa_avg_mlp, sen_avg_mlp, spe_avg_mlp, oa_avg_svm, sen_avg_svm, spe_avg_svm] = rseEvalAVG(y, predict_mlp_iter, ens_size, num_iter, num_in, predict_svm_iter)

Description:

The function 'rseEvalAVG' average the results gained by repeating the RSE-MLP and RSE-SVM experiments. It can be used either only for RSE-MLP or for both RSE-MLP and RSE-SVM, if RSE-SVM experiment was performed.

Input variables:

y ... vector containing classes the subjects belong to, diseased must be denoted by 1 and must be in the first part of this vector!!! e.g. [1 1 1 0 0]'
predict_mlp_iter ... matrix with prediction of the testing subject class by MLP for each iteration of LOO, each classifier in ensemble, each length of the feature vector and each experiment repetition
ens_size ... scalar, defines number of classifiers in ensemble, must be an odd number
num_iter ... scalar, number of repetitions of the experiment to gain robust results
num_in ... scalar or vector, defines lengths of feature vectors
predict_svm_iter ... matrix with prediction of the testing subject class by SVM for each iteration of LOO, each classifier in ensemble, each length of the feature vector and each experiment repetition

Output variable:

oa_avg_mlp ... matrix, contains averaged overall accuracies for all explored feature vector lengths and for all odd numbers of MLP classifiers in ensemble
sen_avg_mlp ... matrix, contains averaged sensitivities for all explored feature vector lengths and for all odd numbers of MLP classifiers in ensemble
spe_avg_mlp ... matrix, contains averaged specificities for all explored feature vector lengths and for all odd numbers of MLP classifiers in ensemble
oa_avg_svm ... matrix, contains averaged overall accuracies for all explored feature vector lengths and for all odd numbers of SVM classifiers in ensemble
sen_avg_svm ... matrix, contains averaged sensitivities for all explored feature vector lengths and for all odd numbers of SVM classifiers in ensemble
spe_avg_svm ... matrix, contains averaged specificities for all explored feature vector lengths and for all odd numbers of SVM classifiers in ensemble

Algorithm:

```
1. The matrices to store the performance measures (overall accuracy, sensitivity and specificity) for RSE-MLP are declared.
IF nargin == 6 i.e. the RSE-SVM was used in the experiment THEN
   2. The matrices to store the performance measures (overall accuracy, sensitivity and specificity) for RSE-SVM are declared.
```

```
3. Loop over 'num_iter' starts here:
   4. Function 'compPerform' is called to compute the performance measures for RSE-MLP.
   IF nargin < 6 i.e. the RSE-SVM was used in the experiment THEN
      5. Function 'compPerform' is called to compute the performance measures for both RSE-MLP and RSE-SVM.
      6. The performance measures of RSE-SVM are added to storage matrix to be later averaged
   7. The performance measures of RSE-MLP are added to storage matrix to be later averaged
8. Loop over 'num_iter' ends here.
9. Performance measures of RSE-MLP are divided by 'num_iter' to gain the average.
IF nargin < 6 i.e. the RSE-SVM was used in the experiment THEN
   10. Performance measures of RSE-SVM are divided by 'num_iter' to gain the average.
```

Notes:

To understand the structure of output matrices, please, see comments in the code in the place, where these matrices are allocated

rseEvalComb – computes mean performance measures for random subspace ensemble classification scheme over all combinations of classifiers in ensemble

[oa_mean_mlp, sen_mean_mlp, spe_mean_mlp, oa_mean_svm, sen_mean_svm, spe_mean_svm] = rseEvalComb(y, predict_mlp, ens_size, num_in, predict_svm)

Description:

The function 'rseEvalComb' compute mean performance measures of classifiers RSE-MLP and RSE-SVM. The ensemble of classifiers generates N predictions. To explore how well the ensemble help, 1, 3, 5, ...and other odd number of predictions can vote for the final classification class and the trend can be explored. Furthermore, all the combinations of 1, 3, 5,... predictions among N can vote. This function takes these combinations into consideration and call the function to evaluate the RSE-MLP and RSE-SVM approaches.

Input variables:

y ... vector containing classes the subjects belong to diseased must be denoted by 1 and must be in the first part of this vector!!! e.g. [1 1 1 0 0]'
predict_mlp ... matrix, contains predictions for all testing subjects, used classifiers in ensemble and input feature vectors
ens_size ... scalar defines number of classifiers in ensemble, must be an odd number
num_in ... scalar or vector, defines lengths of feature vectors
predict_svm ... matrix, contains predictions for all testing subjects, used classifiers in ensemble and input feature vectors

Output variables:

oa_mean_mlp ... matrix, contains overall accuracies for all odd number of MLP classifiers in ensemble (averaged over many combinations of voting classifiers) and explored feature vector lengths sen_mean_mlp ... matrix, contains sensitivities for all odd number of MLP classifiers in ensemble (averaged over many combinations of voting classifiers) and explored feature vector lengths

spe_mean_mlp ... matrix, contains specificities for all odd number of MLP classifiers in ensemble (averaged over many combinations of voting classifiers) and explored feature vector lengths

oa_mean_svm ... matrix, contains overall accuracies for all odd number of SVM classifiers in ensemble (averaged over many combinations of voting classifiers) and explored feature vector lengths

sen_mean_svm ... matrix, contains sensitivities for all odd number of SVM classifiers in ensemble (averaged over many combinations of voting classifiers) and explored feature vector lengths

spe_mean_svm ... matrix, contains specificities for all odd number of SVM of SVM classifiers in ensemble (averaged over many combinations of voting classifiers) and explored feature vector lengths

Algorithm:

```
1. Definition of vector 'ens_size_odd' (its elements are all odd numbers of predictions that can vote) and matrices to store the performance measures (overall accuracy, sensitivity and specificity) for RSE-MLP are allocated.
IF nargin > 4 i.e. the RSE-SVM was used in the experiment THEN
  2. Matrices to store the performance measures (overall accuracy, sensitivity and specificity) for RSE-MLP are allocated.
3. Loop over feature vectors lengths starts here:
  4. Loop over all odd numbers of predictions starts here:
    5. Function 'genComb' is called to generate all combinations of defined number of classification predictions that vote among all classification predictions.
    6. Function 'perfMeas' is called to count mean performance measures for MLP over all combinations defined by 'genComb'.
    IF nargin > 4 i.e. the RSE-SVM was used in the experiment THEN
      7. Function 'perfMeas' is called to count mean performance measures for SVM over all combinations defined by 'genComb'.
    8. Loop over all odd numbers of predictions ends here.
  9. Loop over feature vectors lengths ends here.
```

Notes:

To understand the structure of output matrices, please, see comments in the code in the place, where these matrices are allocated

som - self-organizing map neurons preprocessing

```
[netsom] = som(X_LOO, num_neur, num_epoch_som)
```

Description:

The function 'som' takes advantage of the Neural Network Toolbox™ and use its functions to train the SOM [1]. It approximates the distribution of data and initialize the neurons weights. This step is recommended by T. Kohonen - the author of SOM and LVQ networks - before LVQ network is used for classification to improve the results. This function offers only one dimensional SOM and initial neighborhood size equal 3 and should be used after taking these issues into account or should be manually changed to fulfil the requirements. This function is recommended only when the data set includes similar number of patients and healthy controls, since using function 'somLab' the same number of neurons is assigned to each group.

Input variables:

X_LOO ... matrix, contains brain images in rows (testing subject is left out) and features in columns; diseased must be in first rows and healthy controls must follow (required when function 'somLab' is used afterwards)
num_neur ... scalar, defines number of neurons
num_epoch_som ... number of training epochs of SOM

Output variable:

netsom ... object, trained SOM

Algorithm:

1. Parameters related to SOM training (dimension of map, number of neighbors when starting the adaptation and distance function) are defined.
2. SOM network is created using 'selforgmap' function from Neural Network Toolbox™.
3. Window showing the status of the training is turned off.
4. The SOM is trained.

References:

[1] T. Kohonen, „The self-organizing map", Proc. IEEE, Vol. 78, No. 9, pp. 1464–1480, 1990.

Notes:

Requires Neural Network Toolbox™

somLab - self-organizing map neurons labeling

[labels] = somLab(X_LOO, y, i, num_neur, som_net)

Description:

The function 'somLab' adds the labels to the neurons initialized by the function 'som' that is a part of this toolbox. Both functions are recommended when the data set includes similar number of patients and healthy controls, since the same number of neurons is assigned to each group.

Input variables:

X_LOO ... matrix, contains brain images in rows (testing subject is left out) and features in columns; diseased must be in first rows and healthy controls must follow
y ... vector, contains classes the training subjects belong to, diseased must be denoted by 1 and must be in the first part of this vector!!! e.g. [1 1 1 0 0]'
i ... index of current iteration of leave-one-out cross-validation loop
num_neur ... scalar, defines number of neurons
netsom ... object, trained SOM

Output variable:

labels ... matrix, defines what class the neurons represent

Algorithm:

1. Matrices ED and labels are allocated.
2. Matrix ED is filled with Euclidean distances between all neurons and training subjects.
3. The closest neuron to each training subject is found.
4. Tables with neurons and number of subjects, for which these neurons are winners, are made. The tables are two, one for patients and one for healthy subjects.
5. Loop over the (half) number of neurons starts here:
   6. The neuron that is the winner neuron for most patients is labeled as 'patient neuron'. The class of this neuron cannot be later changed.
   7. The neuron that is the winner neuron for most healthy controls is labeled as 'healthy neuron'. The class of this neuron cannot be later changed.
8. Loop over the (half) number of neurons ends here.
9. Assign label to the unlabeled neurons. This is correction for situation, if the Nth neuron (and previous neurons) never won for any subject, then these neurons do not appear in the tables from 4. and therefore less than N neurons are labeled during 5-8.

iv: DEMO1: Single Artificial Neural Network Classification Scheme

The purpose of this DEMO is to present how to use the 'Toolbox for Brain Image Recognition Using Artificial Neural Networks' (ANN) to classify images using single classifiers based on artificial neural networks. After going throug this DEMO, the user should be able to apply the classification scheme that consists of feature selection using two-sample t-tests, training of the classifier (both within leave-one-out cross-validation) and evaluation of the results. Finally the results can be shown in the graph. The neural network classifiers available are Multi-layer Perceptron (MLP) Neural Network, Radial Basis Function (RBF) Neural Network and Learning Vector Quantization (LVQ) Neural Network.

demo1 contains 4 steps:

1. Data Preparation

In the first step, the function to load the example data is applied.

2. Classifier Selection

In the second step, function that uses a neural network in the classification scheme is chosen. It is shown how to use schemes with MLP, RBF and LVQ. Several parameters can be defined when applying one of the ANNs for classification. These parameters are architecture of ANNs i.e. number of neurons, layers of neurons, goal of error function and spread of RBF neurons for RBF network and number of training epochs for LVQ networks. To train the LVQ network, LVQ1 learning algorithm or both LVQ1 and LVQ2.1 learning algorithms can be applied. Furthermore, Kohonen's Self-organizing Map can be used to preprocess the weights of neurons used in LVQ network. Parameter 'num_iter' defines number of experiment repetitions, since MLP and LVQ contain random steps. The results can be then for example averaged. The parameters are defined in this DEMO to show the examples that lead to good results. The output of this function is table with the results.

In this step, the user selects the type of classifier:

For example:

```
Choose neural network type: options - 'MLP','RBF','LVQ1','LVQ2.1': 'MLP'
```

The single ANN scheme is then applied on the example data set and the table(s) with results are returned after all iteration of leave-one-out cross-validation are done:

```
LOO - iteration: 1/20
LOO - iteration: 2/20
LOO - iteration: 3/20
LOO - iteration: 4/20
…
LOO - iteration: 20/20
```

3. The Results

In the third step, the performance measures (accuracy, sensitivity and specificity) are printed:

MLP:

```
Mean overall accuracy:0.91
Mean sensitivity:0.88
Mean specificity:0.94
```

RBF:

```
Overall accuracy:0.7
Sensitivity:0.8
Specificity:0.6
```

LVQ trained by LVQ1 algorithm:

```
Mean overall accuracy:0.58333
Mean sensitivity:0.73333
Mean specificity:0.43333
```

LVQ trained by LVQ1 and LVQ2.1 algorithms and preprocessed by SOM:

LVQ1

```
Mean overall accuracy:0.78333
Mean sensitivity:0.86667
Mean secificity:0.7
```

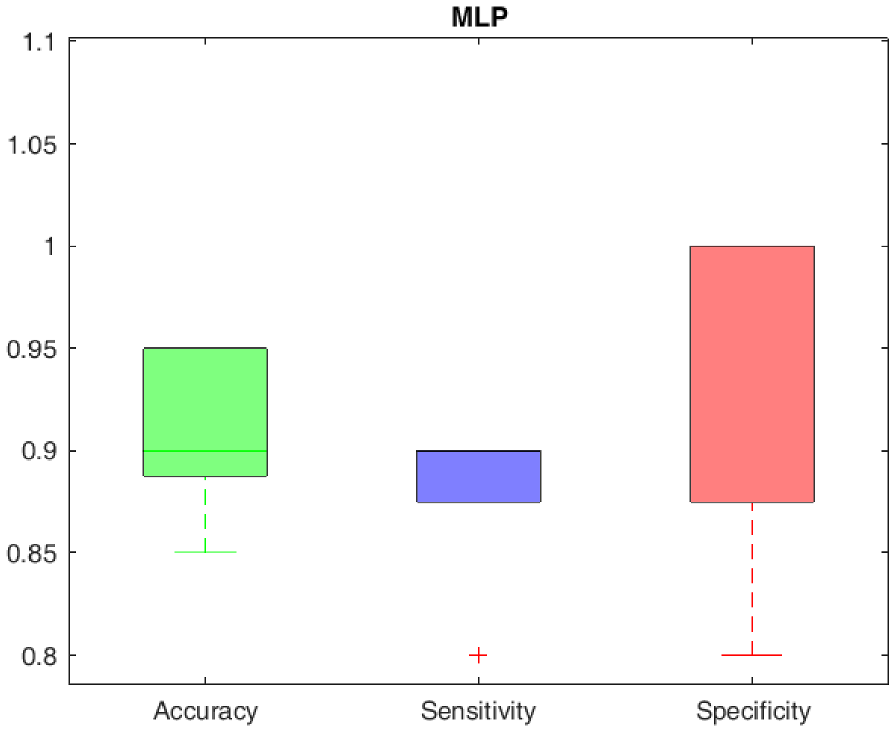
LVQ2.1

```
Mean overall accuracy:0.78333
Mean sensitivity:0.86667
Mean specificity:0.7
```

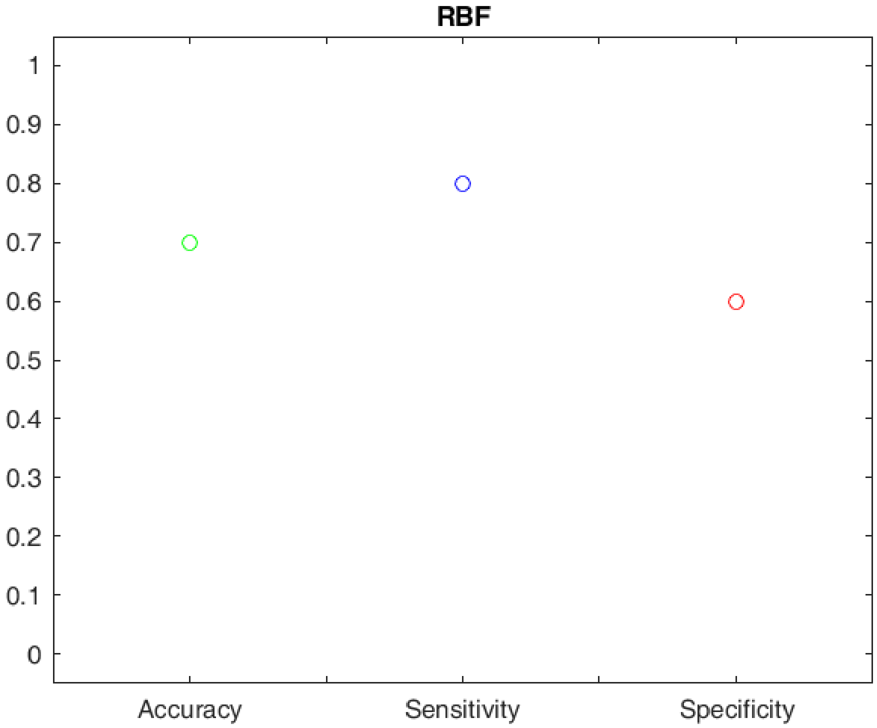
4. Plot Figure(s) with Performance Measures

The graphs for chosen neural network classifier are drawn in the last step.

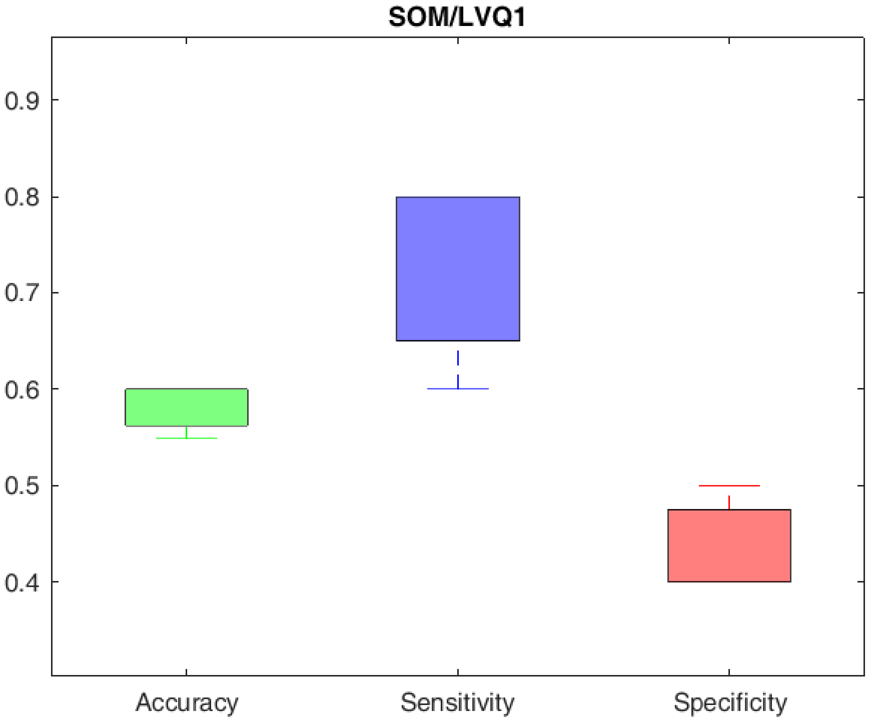
MLP:



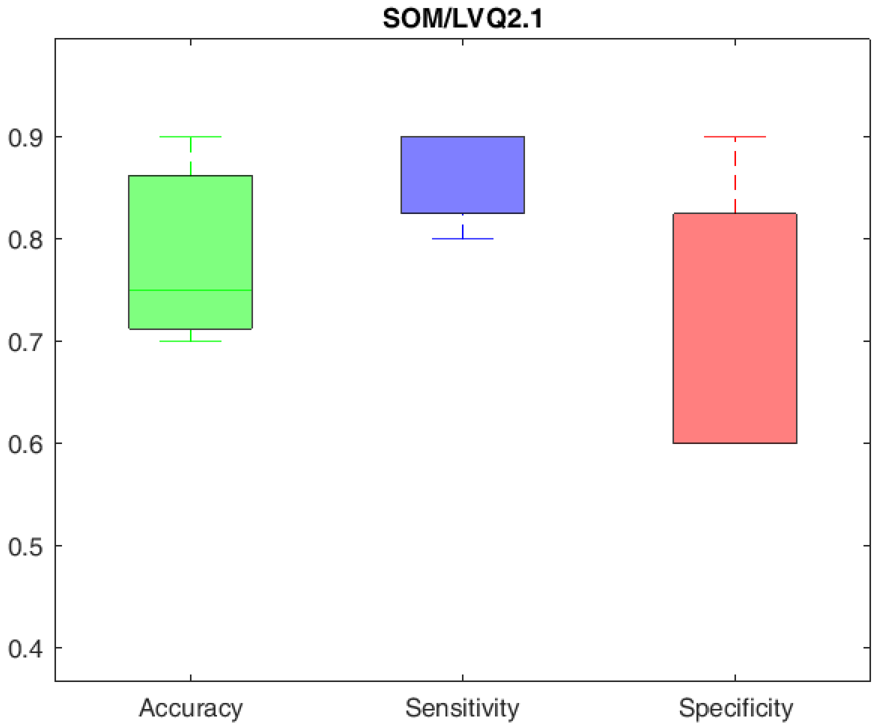
RBF:

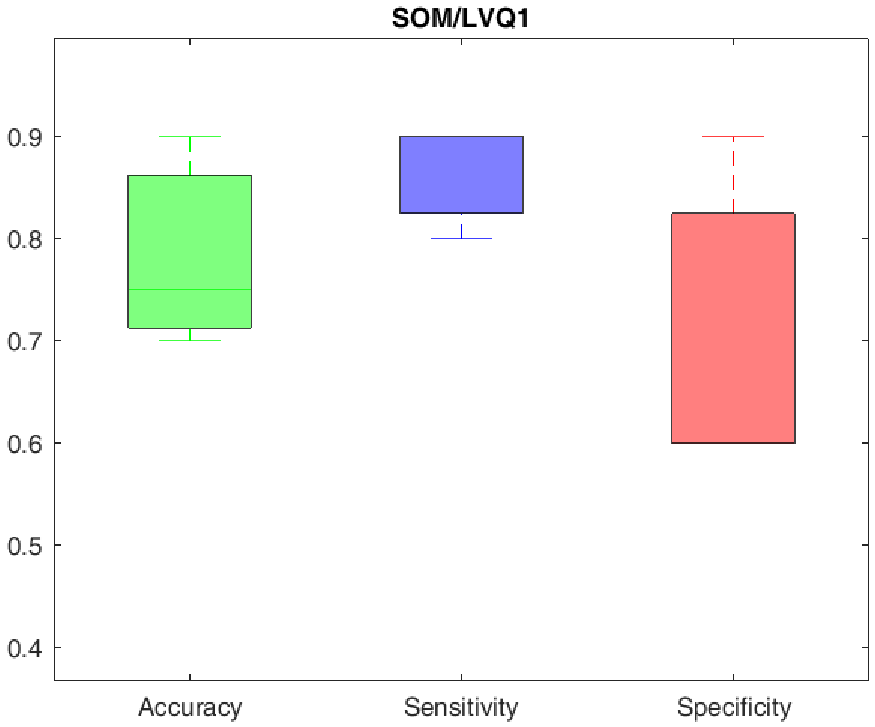


LVQ trained by LVQ1 agorithm:



LVQ trained by LVQ1 and LVQ2.1 algorithms and preprocessed by SOM:





v. DEMO2: Random Subspace Ensemble Multi-layer Perceptron Classification Scheme

The main goal of *demo2* is to go through the functions that perform the classification scheme based on Random Subspace Ensemble Multi-layer Perceptrons (RSE-MLP). In this classification scheme a feature pool that contains only limited amount of the most significant voxels that distingusish between the groups of diseased and healthy controls based on two sample t-tests is prepared. The ensemble of classifiers is based on the randomly chosen features from this feature pool. Leave-one-out cross-validation method is used and includes both feature pool creation and classification phases. Several parameters can be defined and explored within the scheme, therefore it is said here how to adjust them. The classification accuracy is finally computed and corresponding graphs drawn.

demo2 contains 4 steps:

1. Data Preparation

In the first step, the function to load the example data is applied.

2. Train the Random Subspace Ensemble Multi-layer Perceptron

The purpose of the second step, is to use the function that train and test the ensembles of Multi-layer perceptrons (MLP) on the subspaces of features. Several parameters can be explored when applying this scheme: number of classifiers in ensemble, architecture of MLPs (nuber of inputs and neurons in each layer) and size of the feature pool. The RSE-MLP can be compared to similar approach, where instead of MLPs, SVMs are used as classifiers (RSE-SVM). Furthermore, the experiment can be repeated several times (parameter 'num_iter') and the results can be avaraged. This step should eliminate the influence of random feature selection.

Parameters definition for example in *demo2*:

```
% Define 7 classifiers in ensemble
ens_size = 7;

% Define lengths of the feature vectors, that are explored in the experiment
num_in = [100 1000 10000];

% Define 5 neurons for each MLP in ensemble
num_neur = 5;

% Set size of the features pool to 20000 (the random subspaces of features
% are chosen from this feature pool)
FP_size = 20000;

% Define 3 MLPs to train them on the same subspace of features
% (these MLPs are used to vote for the class of the testing subject to
% reduce the influence of the random initialization of the neuron weights)
ens_size_wei_init = 3;

% Set number of repetitions of the experiment
num_iter = 2;

% Comparison with SVM is required
svm_comp = 1;
```

The RSE-MLP scheme is then applied on the example data set and the matrices with results are returned after all iteration of leave-one-out cross-validation and repetitions of experiment are done:

```
Loop for the robustness of the results - iteration: 1/2
LOO - iteration: 1/20
…
LOO - iteration: 20/20
Loop for the robustness of the results - iteration: 2/2
LOO - iteration: 1/20
…
LOO - iteration: 20/20
```

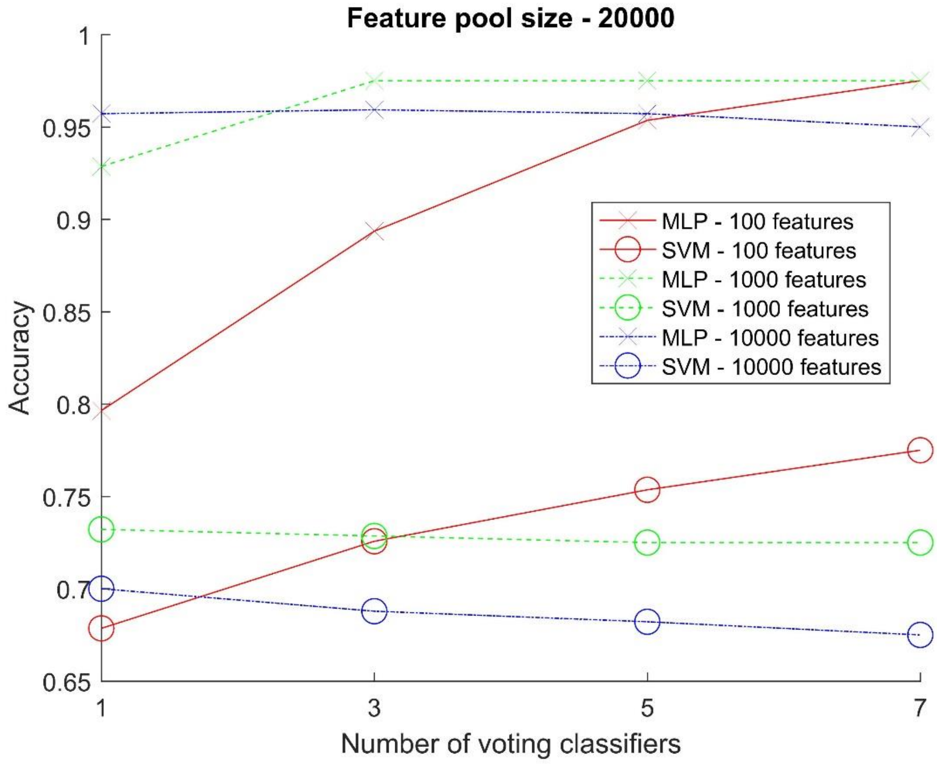
3. Evaluate the Outcomes

In the third step, the performance measures (accuracy, sensitivity and specificity) are computed for all odd numbers of voting classifiers and input vectors' lenghts. Furthermore, all posible (but not more than 10000) combinations of particular number of classifiers out of all classifiers in ensemble are used to vote and the results are avaraged. The results are avaraged also over repetitions of the experiment.

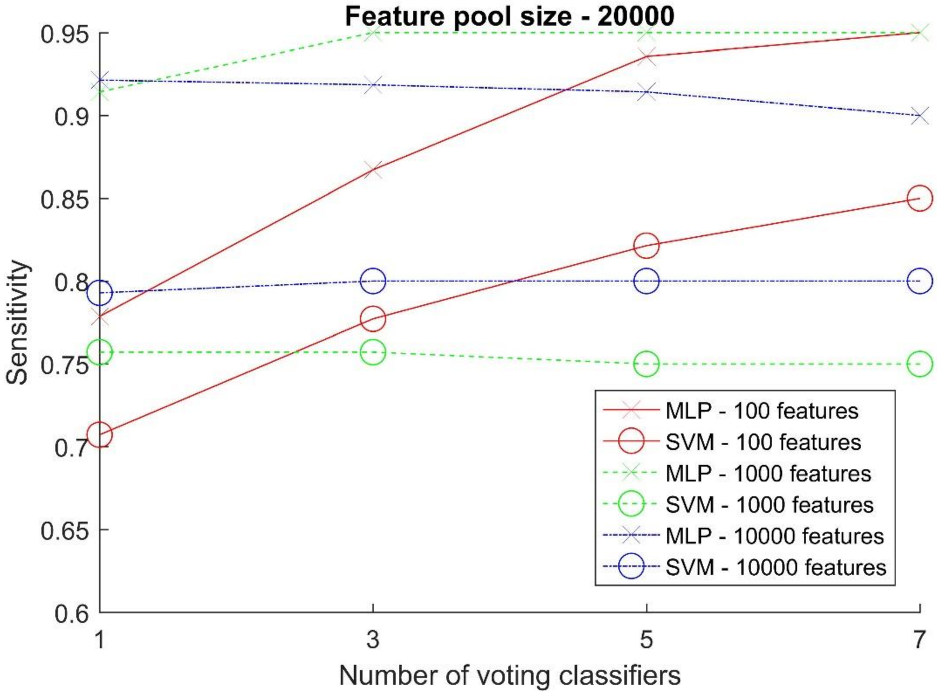
4. Plot Figures with Performance Measures

The graphs for this classification scheme are drawn in the last step.

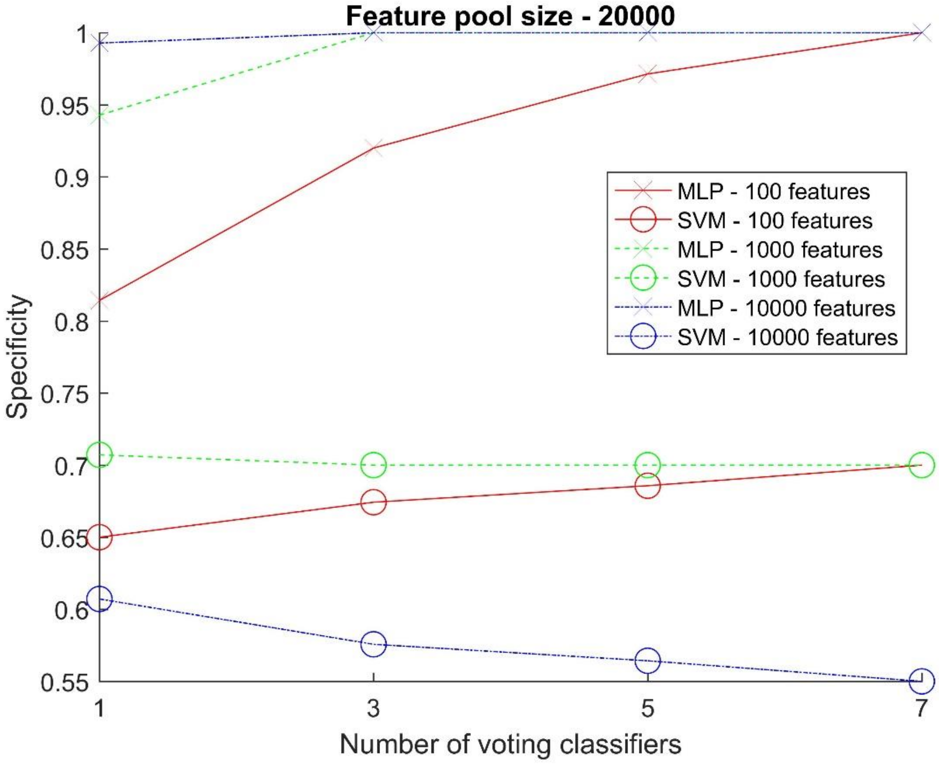
1. Accuracy



2. Sensitivity



3. Specificity


Feature pool size - 20000
Specificity
Number of voting classifiers
MLP - 100 features
SVM - 100 features
MLP - 1000 features
SVM - 1000 features
MLP - 10000 features
SVM - 10000 features