LSS Classification Toolbox

Stefan Uhlich

May 6, 2009

This manual gives a short introduction into the LSS Classification Toolbox. This Toolbox was originally intended for emotion recognition using speech signals but has been rewritten to be also useful for other tasks.

1 Data structures

Each classifier that is available in this Toolbox has the same parameter interface: The two struct variables `DATA` and `PARAMETERS`.

- The `DATA` argument is a structure array which contains two fields – `TRAINING` and `CLASSIFICATION`. The patterns in `DATA.TRAINING` are used for the supervised learning of the decision rules. These decision rules are then tested with `DATA.CLASSIFICATION`, which contains the classification patterns.

  Each row of *`DATA.TRAINING`* and `DATA.CLASSIFICATION` corresponds to one pattern where the elements `1:end-1` are the feature values for this pattern and the last entry in each row is the class label which has to be an integer value.

  Example: Assume, we have four features. One row of `DATA.TRAINING`/`DATA.CLASSIFICATION` might look like

  $$\begin{array}{ccccc} \textit{1. Feature} & \textit{2. Feature} & \textit{3. Feature} & \textit{4. Feature} & \textit{Class label} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.5 & 3.2 & -2.3 & -1.2 & 3 \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{array} \tag{1}$$

- `PARAMETERS` is also a structure array. It contains classifier-specific parameters, e.g. the number of nearest neighbours that should be used for the classifier `k_nearest_neighbour_classifier` can be set with the following Matlab command

```
PARAMETERS.NEAREST_NEIGHBOURS = 3;
```

The return value for all classifiers has also the same form: It is a matrix `EVAL_MATRIX` which has two columns. For each row in `DATA.CLASSIFICATION` it has in the first column the estimated class and in the second column the true class (which was given in the last column of `DATA.CLASSIFICATION`). The `EVAL_MATRIX` can e.g. be evaluated with the help of

```
generate_confusion_matrix(EVAL_MATRIX)
```

which prints a confusion matrix.

2 Classifiers

The user can choose among nine different classifiers. The classifiers `bayes_classifier` and `linear_discriminant_function_classifier` *(_quadratic/_cubic)* return in addition a matrix `PROB_EVAL_MATRIX` that contains the a-posteriori probabilities for each pattern of `DATA.CLASSIFICATION`. This information can e.g. be used to give a confidence value for a decision.

The following table gives an overview of all available classifiers. The values in brackets are the default settings if the corresponding parameters are not defined by the user.

Classifier Name	General Description	Available Parameters	Parameter Description
`ann_classifier`	three-layer feed-forward neural network. The user can choose among the well-known backpropagation or the Levenberg-Marquardt method for the training. The weight matrices are initialized randomly at the start.	*HIDDEN_LAYER_LENGTH* *ANN_ITERATIONS* *USE_MOMENTUM_TERM* *USE_MARQUARDT*	Number of nodes in the hidden layer *[3]* Number of restarts of the learning algorithm. The weights, that have the minimal training error are used during classification. *[10]* In the case of backpropagation (`USE_MARQUARDT=0`), `USE_MOMENTUM_TERM=1` will switch on the momentum term. This often gives a speed-up in the convergence rate. `USE_MOMENTUM_TERM=0` will use the standard backpropagation. *[1]* If `0` then the backpropagation algorithm is used, `USE_MARQUARDT=1` uses the Levenberg-Marquardt optimization. *[0]*
`bayes_classifier`	uses the Bayes plug-in rule. The class-conditional distributions are assumed to be Gaussian. This choice makes sense for unimodal distributed feature vectors.	*COVARIANCE_MATRIX*	gives the type of the covariance matrix. Can be `'full'`, `'diagonal'` or `'spherical'`. The use of `'diagonal'` and `'spherical'` is useful in the case that there is not sufficient training material to estimate the full covariance matrix. *['full']*
`decision_tree_classifier`	uses a decision tree for the classification. The training and classification is done with help of the *Matlab Statistics Toolbox*. There are many parameters that can be set during the classification, see `help` *treefit* for more information.		
`gmm_classifier`	uses the Bayes plug-in rule. The class-conditional distributions are assumed to be distributed according to a Gaussian mixture model. For the training of the unknown parameters the EM algorithm is used. Such a model is useful for multimodal distributed feature vectors.	*GMM_DIM*	Model order of the Gaussian mixture model, i.e. number of Gaussians. Can be set to a constant value to force all distributions to use this model order. With `GMM_DIM=NaN` the AIC is used to select the best suited distribution among all candidate models. To do so, the EM is used with $m = 1, \ldots, 4$ and the model with the minimum AIC value is selected. *[NaN]*
`k_nearest_neighbours_classifier`	uses the simple nearest neighbours rule, i.e. the test pattern is assigned to the class, that is most frequent in the neighbourhood of the test pattern.	*NEAREST_NEIGHBOURS* *METRIC*	Number of neighbours, that should be considered. *[1]* is `'euclidian'` or `'mahalanobis'`. The mahalanobis distance has the advantage, that it is invariant to all non-singular transformations. *['euclidian']*
`linear_discriminant-_function_classifier` *_quadratic/cubic*	uses a least squares estimation to determine the unknown parameters of the discriminant function. It is also possible to include quadratic or cubic terms.		

Classifier Name	General Description	Available Parameters	Parameter Description
`naive_bayes_classifier`	uses the Bayes plug-in rule with the assumption that the feature distributions are independent.	*`FEATURE_DISTRIBUTION`*	Defines the feature distribution that is used. Possible values are `'gaussian'`, `'uniform'` and `'gaussian_kernel'`. `'gaussian_kernel'` corresponds to the parzen window method with a gaussian kernel function.
`nearest_mean_classifier`	Each class is represented by its mean vector. A test pattern is assigned to the class with the nearest mean vector. The euclidean distance is used as similarity measure.		

3 Feature extraction

The LSS Classification Toolbox provides two transformation methods to reduce the dimension of the feature vectors: `principal_component_analysis` which uses the principal components of the scatter matrix and `fisher_discriminant_analysis` which is the well known linear discriminant analysis by R. A. Fisher.

Both functions expect two parameters: `DATA` with the two fields `DATA.TRAINING` and `DATA.CLASSIFICATION` as well as `DIM` which gives the new dimension. Only `DATA.TRAINING` is used to learn the linear transformations.

```
DATA_NEW = principal_component_analysis(DATA, DIM)
[DATA_NEW, DATA_NEW2] = fisher_discriminant_analysis(DATA, DIM)
```

The fisher discriminant analysis returns two feature matrices. `DATA_NEW` uses the *between-scatter* matrix $\mathbf{S}_B$ in the numerator of the fisher criterion and `DATA_NEW2` uses the *total-scatter* matrix $\mathbf{S}_T$ instead. In the literature [1], it is stated, that `DATA_NEW2` is more robust for small number of patterns than `DATA_NEW`.

4 Feature selection

For the selection of a subset, the function `sequential_floating_forward_selection` can be used. It uses the SFFS algorithm and expects two parameters: `DATA.FEATURES` is a feature matrix with all available features. The SFFS algorithm will choose a subset of all available features. `PARAMETERS` gives the user the possibility to change the default settings.

```
[INDICES, P_CORRECT] = sequential_floating_forward_selection(DATA, ...
                                            PARAMETERS)
```

`INDICES` is a cell array which contains for each step the selected features. `P_CORRECT` gives for each step the rate of patterns that where correctly classified.

The function `sequential_floating_forward_selection` uses internally `evaluate_classifier`. `evaluate_classifier` can also be used to determine the performance of a classifier. It returns a `EVAL_MATRIX` that the SFFS algorithm uses to choose the best subset.

The following table gives an overview of all settings for the SFFS algorithm.

`PARAMETERS.SFFS_DIMENSION` Number of features, that the SFFS algorithm should select. *[10]*

`PARAMETERS.CLASSIFIER` String with the name of the classifier, that should be used to calculate the classification error. The classification error is used during the execution of the SFFS algorithm to select a subset. The selected subset is therefore adapted to the used classifier. *['bayes_classifier']*

`PARAMETERS.TCR` Defines the ratio between training patterns and all available patterns. E.g. `PARAMETERS.TCR = 0.5` corresponds to the case that half of the patterns are used for training and the other half for classification. Note, that there are two special cases:

- `NaN` Use all patterns for training and classification
- `0` Use the leave-one-out method for classification. `PARAMETERS.ITERATIONS` is automatically set to the number of patterns.

[0.9]

`PARAMETERS.ITERATIONS` Number of iterations to avoid statistical side-effects. *[20]*

`PARAMETERS.FEATURE_REDUCTION` If `PARAMETERS.FEATURE_REDUCTION` is

- `NaN` then apply no feature reduction technique
- `1` then use the fisher linear discriminant analysis (determinant criterion)
- `2` then use the fisher linear discriminant analysis (trace criterion)
- `3` then use the principal component analysis

[NaN]

`PARAMETERS.REDUCED_DIMENSION` Number of feature after the reduction has been applied. Only used if `PARAMETERS.FEATURE_REDUCTION` is not `NaN` *[3]*

`PARAMETERS.VERBOSE` Takes the following values:

- `0` Suppress all output
- `1` Show intermediate results and progress bar (Default)
- `2` Additionally show status of `evaluate_classifier`

References

[1] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification and Scene Analysis*, 2001.