Usage LVQTools
Bachelor project: implementing LVQ in R

Sander Kelders

June 14, 2010

Introduction

This document is part of a bachelor project which implements several LVQ algorithms in the statistical language R. It describes in short the usage of the package. The `validate` function is the entry-point and thus various parameters can be specified here. This document lists these parameters with a short explanation of its function.

Parameters

Input

1. *datapath = NA*: The location of the file containing data used for training or nfoldcrossvalidation. If this value is *NA inp* will be used, otherwise the specified file will be used. Default value: *NA*.

2. *normalizescheme = 'none'*: Determines how the data is normalized. *ztransform* substracts the mean and divides by variance. *iqr* substracts the median and divides by the InterQuantile Range. *sumone* makes each datapoint sum up to one by dividing all values by the datapiont's sum. Available schemes: *ztransform*, *iqr*, *sumone*, *none*. Default value: *none*

3. *normalclasswise = 'none'*: Determines the class on which the normalisation is based for classwise normalisation. Default value: *none*.

4. *inp = NA*: The data used for training. If *datapath* is *NA inp* will be used, otherwise *datapath* will be used. Default value: *NA*.

5. *replaceNA = FALSE*: Determines whether or not the *NA* values in the input will be replaced. If *TRUE* they will be replaced by the overall median, unless classwise replacement is used. Available values: *TRUE*, *FALSE*. Default value: *FALSE*.

6. *replaceclasswise = FALSE*: Determines whether or not the replacement of *NA* values will be classwise. If *TRUE NA* values will be replaced by the median of the class to which the datapoint belongs to, otherwise the overall median is used.

7. *testdatapath = NA* The location of the file containing data used for testing. If this value is *NA testinp* will be used, otherwise the specified file will be used. Default value: *NA*.

8. *testinp = NA*: The data used for training. If *datapath* is *NA* *inp* will be used, otherwise *datapath* will be used. Default value: *NA*.

LVQ

1. *alfa = 2*: A variable used only in conjunction with the *renyi* LVQscheme. Determines the variant of renyi-divergence to be used.

2. *distscheme = 'euclidean'*: The distance measure used for determining the difference between prototype and datapoint. Together with *LVQscheme*, *relevancemode*, *relevancescheme* and *optimisationscheme* this determines the complete distancemeasure. When using scheme *custom* a custom differencemeasure can be used by setting the *customdist* parameter. The *distscheme* variable is only used in conjunction with the *LVQ1* LVQscheme and not in conjunction with *cauchyschwarz* or *renyi*.
   Available schemes: *manhattan*, *euclidean*, *custom*.
   Default scheme: *euclidean*.

3. *epochs = 10*: The number of epochs used in training. Default value: *10*.

4. *initscheme = 'zero'*: Determines the way the prototypes are initialized. *mean* initializes all prototypes at the mean of all the datapoints. *randomsample* initializes all prototypes by selecting a different random sample for each prototype and using its values for initialisation. *randomwindow* initializes all prototypes by constructing a window which includes all datapoints and initialising each prototype randomly within this window. *zero* initializes all prototypes by setting all values to *0*.
   Available schemes: *mean*, *randomsample*, *randomwindow*, *zero*.
   Default scheme: *zero*.

5. *learningrate = 0.01*: Determines the rate at which the prototypes are adjusted. This can be a single value to be used throughout the whole training process or a vector of length *epochs*, which will use each value once in order. Default value: *0.01*.

6. *LVQscheme = 'LVQ1'*: Determines which version of LVQ is used. Together with *distscheme*, *relevancemode*, *relevancescheme* and *optimisationscheme* this determines the complete distancemeasure. Available values: *LVQ1*, *cauchyschwarz*, *renyi*. Default value: *LVQ1*.

7. *customdist = 3*: When using *distancemeasure custom*, this parameter determines the distance-measure used. *customdist* is p in $\sqrt[p]{|datapoint - prototype|^p}$

8. *optimisationscheme = 'normal'*: Determines which type of costfunction is used and thus how the prototypes are updated. The *normal* optimisationscheme uses the winner takes all principle and only updates the closest prototype. The *general* optimisationscheme is used for generalized LVQ. It uses stochastic gradient descent to determine the prototype updates. The following function is used for this purpose: $\Sigma_i \Phi(\mu)$ with $\mu = \frac{d_J^\Lambda - d_K^\Lambda}{d_J^\Lambda + d_K^\Lambda}$. And with d_J^Λ as the distance to the nearest prototype of the appropriate class and d_K^Λ as the distance to the nearest prototype of another class.
   Available uses: *normal*, *general*.
   Default value: *normal*.

9. *prototypes = vector()*: Determines the number of prototypes for each class. This vector must have entries accesible by strings representing the classlabels. Each entry lists the number of prototypes for the class whose label was used for accessing it. A usable default value is not present. This parameter has to be specified manually.

10. *relevances = NA*: When *mode relevance* or *matrix* is used this parameter contains the relevance-vector or matrix respectively. The relevances can be specified manually using this parameter or when no relevances are provided they will be randomly initialized. Default value: *NA*.

11. *relevancemode = 'normal'*: Determines if relevances should be used or not. *normal* mode does not use relevances at all. *relevance* mode uses a relevancevector to assign relevances to each dimension. *matrix* mode uses a square relevancematrix to assign relevances to dimensions and correlations between them. When using mode *matrix* only *euclidean distancescheme* is available.
    Relevances are not available when using *cauchyschwarz*- or *renyi*-LVQscheme.
    Available values: *normal*, *relevance*, *matrix*.
    Default value: *normal*.

12. *relevancescheme = 'global'*: Determines how many different sets of relevances should be used. When using *global*-relevances only 1 set of relevances is used for all prototypes. When using *local*-relevances, each prototype has its own set of relevances. When using *classwise*-relevances all prototypes of the same class share a set of relevances.

Available values: *global*, *local*, *classwise*
Default value: *global*.

13. *relrate = 0.001*: When using relevances determines the rate at which the relevance-vector or matrix adapts. This can be a single value to be used throughout the whole training process or a vector of length *epochs*, which will use each value once in order. Default value: *0.001*

Output

1. *costcurve = FALSE*: When *TRUE* the cost is calculated after each epoch and the value stored. When the program ends this cost (possibly among other things) is returned. Available values: *TRUE*, *FALSE*. Default value: *FALSE*.

2. *progress = FALSE*: When *TRUE* records the value of all prototypes before the first and after each epoch and returns it (possibly among other things) after terminating. Possible values: *TRUE*, *FALSE*. Default value: *FALSE*.

3. *prototypeoutput = TRUE*: When *TRUE* records the endconfiguration of the prototypes of a training and returns it (possibly among other things) after terminating. Possible values: *TRUE*, *FALSE*. Default value: *TRUE*.

4. *relevanceoutput = FALSE*: When *TRUE* records the endconfiguration of the relevance-vector or matrix of a training and returns it (possibly among other things) after terminating. Possible values: *TRUE*, *FALSE*. Default value: *FALSE*.

5. *relevanceprogress = FALSE*: When *TRUE* records the value of the relevance-vector or matrix before the first and after each epoch and returns it (possibly among other things) after terminating. Possible values: *TRUE*, *FALSE*. Default value: *FALSE*.

6. *testerror = FALSE*: When testing with a different set than the trainingset stores the number of missclassifications after training and returns (possibly among other things) it after terminating. Possible values: *TRUE*, *FALSE*. Default value: *FALSE*.

7. *testerrorprogress = FALSE*: When using testdata to test the outcome of a training and *testerrorprogress* is *TRUE* calculates the testerror after very epoch and stores it to return (possibly) among other output.

8. *trainerror = FALSE*: After training tests with the trainingset and stores the number of missclassifications and returns (possibly among other things) it after terminating. Possible values: *TRUE*, *FALSE*. Default value: *FALSE*.

9. *trainerrorprogress = FALSE*: When *TRUE* the trainerror is calculated after every epoch and stored to be returned (possibly) among other output.

Progress

1. *graphics = FALSE*: When *TRUE* and the data is 2-dimensional the progress of the prototypes will be plotted after every epoch. Available values: *TRUE*, *FALSE*. Default value: *FALSE*.

2. *plotcurve = FALSE*: When *TRUE* and *costcurve* is also set to *TRUE* the costcurve will be plotted after every training. Available values: *TRUE*, *FALSE*. Default value: *FALSE*.

3. *show = FALSE*: When *TRUE* prints the prototype configuration and if applicable the relevance-vector or matrix and the costcurve to the console. Available values: *TRUE*, *FALSE*. Default value: *FALSE*.

Validation

1. *nfold = 8*: Determines the number of sets the data is divided in when using nfoldcross-validation. Available values: any whole positive number in the range of $[2...numberofdatapoints]$. Default value: *8*.

2. *validatescheme = 'train'*: Determines how training and testing is to be executed. *train* only trains the prototypes, while *traintest* also tests the prototypes after training with a different set of testdata. The *nfold*-scheme will apply nfoldcross-validation with the *nfold*-parameter determining in how many sets the data is to be divided. The sets are divided randomly without consideration to class. Available values: *train*, *traintest*, *nfold*. Default value: *train*