

CB1D - Clustering based 1D learner

Jiří Iša, 0536563

December 22, 2005

Abstract

In this paper we will discuss a 1D learner based on RBF (Radial Basis Functions) networks. It differs from the classical RBF networks by the selection of cluster centers dependent of class classification in the training set. After the method is described, it is compared to other methods (KNN, RBF) and improvement suggestions are made.

Introduction

The CB1D learner was developed as the response to the task given at Automated Learning lectures in the University of Amsterdam in the school year 2005/2006.

1 Task description

Purpose of this assignment is to invent a method, develop a system and justify this, for predicting a class from a single numerical variable. Training data are a set of cases, where a case consists of a single value and a class label. There are only two classes (called + and -). [1]

2 Data set

The used data set is available from <http://www.ics.uci.edu/~mlearn/databases/credit-screening/>. It describes, if someone gets a loan or not. The data come from real financial institution, only the labels were changed in order not to reveal sensitive information.

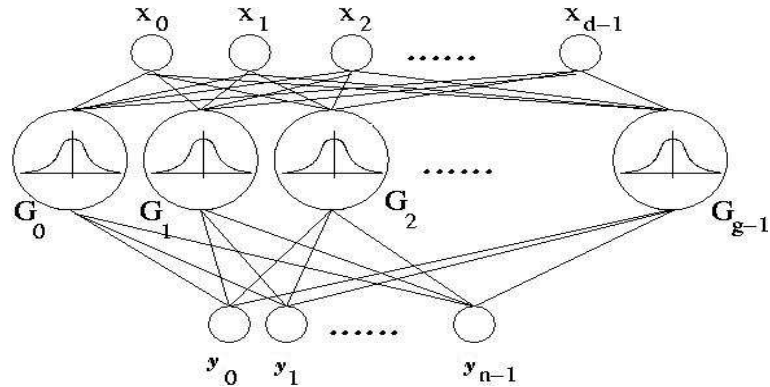
There are several attributes available. It is allowed to select any single continuous variable to make the +/- classification.

Algorithm

In this section we provide a brief description of the Radial Basis Function (RBF) networks and then we naturally extend them to the CB1D algorithm, making single, but significant modification.

3 Classical RBF network

3.1 Architecture



The RBF network consists of three layers, including the input layer. As the input comes, every Gaussian unit in the hidden layer calculates its output as a function of its center and width (variance):

$$G_j(x) = \frac{1}{\sqrt{2\pi b_j^2}} e^{-\frac{\|x-c_j\|^2}{2b_j^2}}$$

- x ... inputs
- $G_j(x)$... output of the j -th hidden unit
- c_j ... center of the unit j
- b_j^2 ... variance of the j -th unit (b_j is called also *width* in the rest of the text)

The output is calculated as a weighted sum of outputs of hidden units:

$$y_k = \sum_j w_{kj} G_j(x)$$

- y_k ... k -th output
- w_{kj} ... weight of the j -th hidden unit for the k -th output

3.2 Learning of RBF networks

Learning of RBF networks usually consists [2] of the following steps:

- Hidden units centers estimation
- Hidden units variance estimation
- Weights to output estimation

3.2.1 Centers estimation

Several different techniques are widely used for hidden units centers estimation. Among the most common belong:

- Random sample selection - random training examples are taken as centers of hidden units
- K-means clustering
- Kohonen's neural network algorithm

3.2.2 Variance estimation

Very often the variance of the unit is taken proportional to the distance to its nearest neighbors [3]. In extreme cases only one neighbor is taken into account.

3.2.3 Weights estimation

This is the first moment in RBF networks, when the class prediction attribute in the data is taken into account.

For every example in the training set the outputs of the hidden units are calculated. Together with the prediction class they form new example used for weights training.

Now the weights training problem becomes a well known single linear neuron learning problem. We can either calculate the weights directly, if possible, or we can use any of the common iterative algorithms, like delta rule [2]:

$$\Delta w_{kj}^t = \sum_{x \in Examples} \alpha (y_k^*(x) - y_k(x)) * G_j(x)$$

- $G_j(x)$... the output of j-th hidden units, whose weight is being adapted
- Δw_{kj}^t ... the weight is updated according to the rule: $w_{kj}^{t+1} = w_{kj}^t + \Delta w_{kj}^t$
- α ... a learning speed
- $y_k^*(x)$... desired k-th output for training example x
- $y_k(x)$... actual k-th output for training example x

4 CB1D

The major drawback of the RBF networks is the fact, that they do not consider classification attribute during the clustering phase. The obvious solution would be to improve the centers according to the training examples including the classification attribute.

4.1 Algorithm

The major and the only difference of CB1D is compared to RBF is the fact, that the hidden units centers calculation is split into as many pieces, as there are classification possibilities.

4.1.1 Centers estimation

The centers of hidden units may be estimated by any algorithm usually used in RBF networks. The only difference is in the fact, that first the training data is split according to the class attribute, then centers of clusters (hidden units in our case) for each partial data set are found and after that these centers are merged together to form the single set.

4.1.2 Variance estimation

The width of each unit in the hidden layer may be found by any algorithm used for RBF networks.

4.1.3 Weights estimation

The same as for variance estimation holds for weights estimation. Any common method may be used.

Tests

This section describes the implementation used both for RBF networks and for CB1D and the results obtained from their comparison and comparison to the *Nearest neighbor* and the *Four nearest neighbors* method.

5 Implementation

5.1 Centers estimation

The Kohonen's neural network algorithm was used to estimate centers of the hidden units. Because of the one-dimensional nature of our task, the one-dimensional Kohonen's neural network algorithm was chosen.

The *neighborhood* for this algorithm is set to 1, because of the low number of hidden units we use in our example. It means that not only the best fitting (nearest) cluster/center is moved in the direction of the example, like in *k-means clustering*, but also both neighboring clusters on both sides are moved.

5.1.1 Stop conditions

Fix amount of iterations (600) over the training set if performed. This is a hand-tuned parameter.

5.2 Variance estimation

The simple method of using the average distance to the neighbors was used.

5.2.1 Stop conditions

As this is a single pass over all hidden units, no stop condition is needed.

5.3 Weights estimation

The delta rule as described in 3.2.3 was used.

5.3.1 Improvement

The best weights combination achieved so far is remembered and after the learning is finished, this combination is used. This is called *bucketing*.

5.3.2 Stop conditions

The maximal number of iterations over training set is limited to 8.000 cycles. If the performance did not improve during last 200 iterations over the training set, learning is stopped sooner and the best weights combination achieved so far is restored.

6 Results and discussion

All the results come from the 10-fold cross-validation over the given training data.

For our purpose we use A8 variable, which seems to offer better classification chance than the other variables (according to simple analysis performed using *Weka toolkit*). As the variable itself doesn't seem to give any strong predictive power, it might be interesting to see, how different methods deal with overfitting.

The own implementation of CB1D is compared to the RBF network implemented by the same way (centers finding, width assignment, weights adaptation). To understand the significance of the obtained results the *Weka toolkit* [4] is used to get the results for more advanced implementation of *RBF network* and for *k nearest neighbors* method.

Classified as:	+	-
Desired value: +	203	104
Desired value: -	109	274

Table 1: CB1D (8 hidden units) classification results with error 30.87%

Classified as:	+	-
Desired value: +	200	107
Desired value: -	108	275

Table 2: RBF with implementation similar to CB1D (8 hidden units) results with error 31.16%

Classified as:	+	-
Desired value: +	194	113
Desired value: -	108	275

Table 3: Weka implementation of RBF (8 hidden units) classification results with error 32.03%

Classified as:	+	-
Desired value: +	168	139
Desired value: -	163	220

Table 4: Nearest neighbor classification (Weka, 1B1) classification results with error 43.72%

Classified as:	+	-
Desired value: +	172	135
Desired value: -	108	275

Table 5: Four nearest neighbors (Weka, 1Bk, k = 4) with error 35.22%

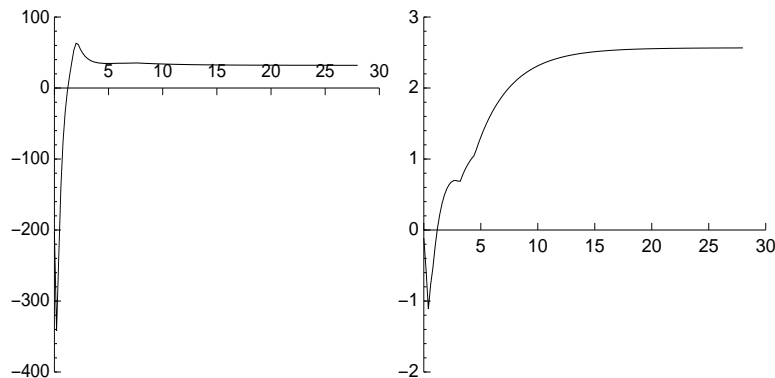


Figure 1: CB1D (left) and RBF (right) classification. Values with the curve above zero are classified as +, values below zero are classified as -

Learner	10-fold cross-validation error
CB1D:	30.87%
RBF:	31.16%
Weka RBF:	32.03%
Weka lBk, k=4:	35.22%
Weka lB1:	43.77%

Table 6: 10-fold cross-validation error summary

In the tables 1 to 5 we see few interesting topics. First of all, RBF networks perform better than kNN algorithm. This might be due the lower sensitivity of RBF networks to training data.

It is also clear, that CB1D performs slightly better, than RBF in the same representation and the same number of hidden units. It would be interesting to see the difference on more predictive attribute, maybe even with n-dimensional input. However, no such attribute is available in our training set.

The better performance of the hand-written algorithms compared to Weka implementation are believed to be caused by the hand-tuned cluster width definition against the general heuristic rules in Weka toolkit.

On the figure 6 we can compare different approaches of the similar implementations of CB1D and RBF. While RBF considers examples with high value of the attribute to be "surely class +", CB1D reflects the fact, that there are not enough training examples to make any strict conclusion.

Conclusion

7 Results

The improvement suggested to learning of RBF networks really slightly improved the performance of the learner on our data set. Also the classifying curve looks "smarter" for CB1D.

From Figure 6 it is obvious that it would be possible to obtain a simple linear classifier with the same performance on our data set. The advantages of RBF based classifier would be visible on the less homogeneous data.

The 10-fold cross-validation errors of all compared learners are in Table 6.

8 Suggestions for further work

It might be interesting to check the overfitting inclination of CB1D, as it might, and usually does, end up with cluster centers close one to another to reflect slight differences in class distributions in training examples and their classification.

For such a case some pruning of hidden units might be involved. Early experiments show, that the performance according to the 10-fold cross-validation degrades, when pruning is introduced, but it is not clear yet, if the better generalization is gained. The influence of pruning would be different than pruning on RBF, where clusters are not so closed one to another. There are also two moments of pruning possible: before merging of centers and after it.

References

- [1] Automated Learning task description: <http://staff.science.uva.nl/~maarten/ml2/> (2005)
- [2] Stuart Russel, Peter Norvig: *Artificial Intelligence - A Modern Approach, second edition*, Prentice Hall, (2003), ISBN: 0-13-080302-2
- [3] Nabil Benoudjit, Michel Verleysen: *On the Kernel Widths in Radial-Basis Function Networks*, Neural Processing Letters 18: 139-154, 2003
- [4] Weka AI toolkit: <http://www.cs.waikato.ac.nz/ml/weka/>