Constrained Learning Vector Quantization or Relaxed k-Separability

Marek Grochowski and Włodzisław Duch

Department of Informatics, Nicolaus Copernicus University, Toruń, Poland,
grochu@is.umk.pl; Google: W Duch

Abstract. Neural networks and other sophisticated machine learning algorithms frequently miss simple solutions that can be discovered by a more constrained learning methods. Transition from a single neuron solving linearly separable problems, to multithreshold neuron solving k-separable problems, to neurons implementing prototypes solving q-separable problems, is investigated. Using Learning Vector Quantization (LVQ) approach this transition is presented as going from two prototypes defining a single hyperplane, to many co-linear prototypes defining parallel hyperplanes, to unconstrained prototypes defining Voronoi tesselation. For most datasets relaxing the co-linearity condition improves accuracy increasing complexity of the model, but for data with inherent logical structure LVQ algorithms with constraints significantly outperforms original LVQ and many other algorithms.

1 Introduction

Problems with Complex logical structure are difficult to solve with feedforward neural networks, such as the Multilayer Perceptrons (MLPs), the basis set expansion (Radial Basis Function, RBF) networks, or Support Vector Machines (SVMs). The k-separability index [1] breaks the class of non-separable classification problems into subclasses that require at least k intervals for separation of data after a single linear projection (equivalent to k parallel hyperplanes). Such problems are called k-separable, with $k = 2$ for linear separability. High values of k characterize problems that are difficult to learn for feedforward neural networks and kernel classifiers [1], although in principle they may be solved by a single multi-threshold neuron. For example, the d-bit parity problem is very difficult because it is $d + 1$-separable [1], with vectors forming compact clusters only after suitable projection, not before. Many problems in bioinformatics, text analysis and other fields have inherent complex logic that makes learning difficult. Changing the cost function from a typical error function evaluating separability [2] to function that rewards discovery of interesting patterns in the data helps to create good internal representations. For k-separable problems reward for creation of large clusters of pure vectors after projection to the hidden space creates features that greatly simplify the learning process [3].

Direct use of linear projections is not always the best idea. For example, image segmentation requires distinguishing complex shapes defined in two- or three-dimensional space. This type of difficulty is solved by using localized kernels to project original

Lecture Notes in Computer Science Vol. 5768: 151-160

data to the high-dimensional space [4], providing flexible decision borders in the original space. However, for highly non-separable problems with non-compact distribution of data clusters, as is the case of Boole'an functions, flexible decision borders are not sufficient, and significant simplifications may be achieved if instead of linearly separable k-separable learning target is defined [1]. Such approach may discover simple and comprehensible data models, but needs to be extended if simple solutions do not exist.

In this paper systematic increase of model complexity is investigated, from the simplest, single neuron perceptron that solves separable problems, through k-separable projections equivalent to constrained vector quantization methods, to unconstrained prototype-based LVQ solutions. This is achieved by investigating relations between projection-based and distance-based solutions. In the first case constructive neural networks with perceptron-type nodes may be used [5, 6], while in the second case localized function networks, or learning vector quantization methods, provide prototype-based rules [7]. In the next section some drawbacks of existing systems are briefly pointed out and relations between k-separable solutions and constrained Learning Vector Quantization (cLVQ) approaches explored. Section three describes two such cLVQ algorithms that are tested on some benchmark datasets in section four.

2 *k*-separability and prototypes

There is a growing evidence that existing algorithms easily miss simple solutions for various datasets. Consider the Australian Credit dataset [8] that has been used in the Statlog project [9]. 690 cases of credit card applications are characterized using 6 numerical and 8 categorical attributes, with some missing values. All attribute names and values have been changed to meaningless symbols to protect confidentiality of the data. Almost the same number of approval (55.5%) as disapproval cases (44.5%) has been selected. In the Statlog project [9] over 20 classifiers have been applied to this data, with the best results reached in 10-fold crossvalidation by the Cal5 decision trees, around 87$\pm$4% accuracy with an average of about 6 nodes per tree. Other trees showed an average of about 85.5% and lower, with as many as 70 nodes. Radial Basis Function (RBF) networks gave also about 85.5%, and MLP about 1% lower, while linear SVM gives 85.2$\pm$3.0%, with over 190 support vectors. Optimized Gaussian-kernel SVM is slightly better, with 86.2$\pm$4.1%, for optimized $C = 0.1, \sigma = 0.01$, leaving over 460 support vectors [1]. LVQ gives here even worse results at the 80% accuracy level. Unfortunately all these learning methods missed the simplest rule that has Acc=85.5$\pm$4.0% accuracy, based on a single binary feature: if A8=True then Yes, if A8=False then No. This solution is statistically not worse than the best results for this dataset, and much more comprehensible than those provided by alternative methods. Regularization in neural networks [2] does not provide bias towards this type of simple solutions, preferring many small weights instead of binarized weights.

All learning methods mentioned above also fail when sharp decision borders are needed. On most data for which logical rules gave good results [10] neural networks, LVQ, SVM and the nearest neighbor methods are much worse than decision trees. For

[1] calculations performed with the Ghostminer package, www.fqs.pl/ghostminer

example, the Thyroid Disease data set [8], with 3772 examples for learning (primary hypothyroid, compensated hypothyroid, normal, 2.47%, 5.06%, 92.47% respectively), and 3428 for testing (similar class distribution) has 21 attributes (15 binary, 6 continuous). Rules based on 4 continuous and 2 logical features give 99.36% accuracy on the test set, while the best neural approaches are not better than 98.5% and SVM with optimized Gaussian kernel achieves only 96.1%. Results on complex Boolean problems are even worse, but in this case also decision trees fail completely [5, 6].

Thus in a number of relatively simple problems popular machine learning methods are not able to find good models. One may of course develop a methodology that would start from simple rules [10], for example highly pruned decision trees, and proceed to more complex learning algorithms testing different methods. However, it would be better to have a model that could deal with such problems directly. To this aim relations between projection and distance-based methods are explored below.

A linearly separable data is divided by a hyperplane into two distinct classes. The $\boldsymbol{w}$ vector is perpendicular to this hyperplane and defines a direction for the projection $z(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x}/||\boldsymbol{w}||$ of d-dimensional data points $\boldsymbol{x}$, mapping all points from one class on $z < \theta$, and from the other class on $z > \theta$. The conditional probability densities $p(z|C)p(C)$ after projection of the training data may be used to draw a histogram and estimate probability distributions (Fig. 1). Identical solution is obtained using two prototypes $\boldsymbol{t}_i, i = 1, 2$ placed on the $\boldsymbol{w}$ line symmetrically around the threshold $\theta = |z(\boldsymbol{t}_1) - z(\boldsymbol{t}_2)|/2 = |z_1 - z_2|/2$, each associated with its class C_i, and a rule: choose C_1 if $|z(\boldsymbol{x}) - z_1| < |z(\boldsymbol{x}) - z_2|$, else choose C_2. However, the interpretation of such prototypes may be problematic, as their placement is not unique and cannot in general match both probability density peaks. To facilitate arbitrary placements each prototype should have its own threshold θ_i, with $z_1 + \theta_1 = \theta = z_2 - \theta_2$. Then the decision rule becomes: C_1 if $z(\boldsymbol{x}) < z_1 + \theta_1$, or C_2 if $z(\boldsymbol{x}) < z_2 - \theta_2$. Using two instead of one threshold allows also to define a confidence margin around θ where probability densities may overlap decreasing reliability of predictions.

Some non-separable problems are k-separable; a projection exists that gives k pure clusters of vectors that belong to different classes. It requires only $d+k-1$ parameters, one direction $\boldsymbol{w}$ and $k-1$ intervals. Most Boole'an problems for d-bit strings have $k \sim n/2$ with maximum k probably equal to $d+1$ [1]. This is equivalent to k colinear LVQ prototypes $\boldsymbol{t}_i, i = 1..k$. For example, in d-bit parity problem projection on the direction of the main diagonal $\boldsymbol{w} = [1, 1, ...1]$ creates clusters for alternating odd and even-parity classes containing strings with 0, 1, 2 ... d bits equal to 1, separated from each other by $\sqrt{n}/(n+1)$. The decision rule says: if $z(\boldsymbol{x}) \in [\theta_{i-1,i}, \theta_{i,i+1}]$ than $\boldsymbol{x}$ belongs to the class associated with z_i prototype, where $\theta_{i-1,i}$ is the Bayesian threshold calculated from probability distributions for $z(\boldsymbol{x})$ for vectors that fall into adjacent clusters. Prototypes $z_i = z(\boldsymbol{t}_i; \boldsymbol{w})$ may fall at any point in this interval.

Even though a single projection may sometimes be sufficient for correct classification a larger number of good projections may be useful for building a robust predictor. This may be achieved by sequential constructive network algorithms using projection pursuit based on the Quality of Projected Clusters (QPC) [5, 6]. Note that solutions provided by MLP, RBF or SVM classifiers for k-separable problems are much more complex and cannot generalize well. If not all data samples are separated by projections

extraction of information based on similarity should be tried. One of the simplest such transformations is done by finding prototypes in the original data space, for example by the Learning Vector Quantization (LVQ) algorithm [11, 12], and then use distances to these prototypes as new dimensions. The number of dimensions is then equal to the number of prototypes. LVQ algorithm may also be used to look for optimal projections, but there are two problems with this solution. First, k unconstrained prototypes require kd parameters, while for k-separable problems models $d + k - 1$ parameters are sufficient. Second, it is very difficult to find good unconstrained solution for problems that are highly-nonseparable. Even in the Australian Credit problem LVQ has failed to find binary feature that separates the data fairly well. Finding optimal scaling factors for features used in the distance function is quite hard, therefore all distance-based methods, including LVQ, easily miss it.

In the spirit of the structural risk minimization approach [13] data models of progressively increasing complexity will be created by adding prototypes and relaxing constraints. The simplest model searches for separable solutions starting from testing projections on each of the input features x_i. A line $\boldsymbol{w}_{ij} = \boldsymbol{m}_i - vm_j$ connecting means $\boldsymbol{m}_i$ of vectors for each class for Gaussian distributions gives optimal solution for pairwise class discrimination. This requires d parameters for $\boldsymbol{w}$ plus one parameter for the threshold. In general optimal direction may be found using the QPC criterion [5]. Fixing the reference point at $\boldsymbol{m}_1$ mean similar direction $\boldsymbol{w} \sim \boldsymbol{m}_1 - \boldsymbol{t}_1$ should be found by the LVQ optimization of a single prototype $\boldsymbol{t}_1$. k-separable solutions are only slightly more complex, defining along $z(\boldsymbol{x})$ projection line k intervals and requiring $d + k - 1$ parameters. This is equivalent to k co-linear prototypes $\boldsymbol{t}_i$, with one prototype per resulting cluster. LVQ algorithm should find similar solution if a prototype $\boldsymbol{t}_1$ is adapted without contraints, providing direction $\boldsymbol{w} \sim \boldsymbol{m}_1 - \boldsymbol{t}_1$. The remaining $k - 1$ prototypes are placed on this line, adapting only their positions $z_i = \boldsymbol{w} \cdot \boldsymbol{t}_i$.

Decision borders created by co-linear prototypes divide whole feature space with parallel hyperplanes. If this solution is not sufficient constraints should be relaxed to allow for LVQ adaptation of all prototypes. This creates Voronoi tessellation using kd parameters. Allowing for adaptation of the scaling factors and the type of distance functions creates even more complex decision regions [14, 15], but we shall not investigate it here. Starting from one prototype, adding more prototypes with constraints, and finally allowing for full LVQ adaptation, should systematically explore different biases leading to progressively more complex models. Constraints of this type are a form of regularization [2] that should help to find solutions that are not easy to discover.

3 Constrained Learning Vector Quantization (cLVQ)

The constructive cLVQ algorithm combines (non-local) projection pursuit scheme and (local) LVQ learning. Learning starts from the simplest classification model (in terms of k-separability), defined by a single projection and a single prototype per class. For two-class linearly separable problems this is sufficient. If this is not the case the number of parameters of the model is increased until results will show no significant improvement. Complexity of the model may be increased in two ways: by adding consecutive projections, or by increasing the number of prototypes for each projection.

The cLVQ model may take the form of a two layer feedforward neural network. Let $\mathcal{W} = \{\boldsymbol{w}_i\}, i = 1, \ldots, h$ denote weights of connections between the d input and the h hidden nodes, performing linear projections $\mathcal{W}\boldsymbol{x}$, scaling and combining input features. Let $\mathcal{T} = \{\boldsymbol{t}_j\}, j = 1, \ldots, k$ denote connection between hidden to the k output nodes, one per class, with each weight $\boldsymbol{t}_{jl}$ coding positions of prototypes $\boldsymbol{t}_j$ after projection $\boldsymbol{w}_l$. Classification is done using the winner takes all (WTA) rule, where for a given input $\boldsymbol{x}$ the network returns class label associated with prototype $\boldsymbol{t}_r$ most similar to $\boldsymbol{x}$: $r = \arg\min_j ||\mathcal{W}\boldsymbol{x} - \boldsymbol{t}_j||$. Fully connected network has $(d + k)h$ parameters, where d is the number of inputs, k the number of prototypes (output nodes) and h is the number of projections (hidden nodes). Removing connections with small weight values may further reduce model complexity.

This model is trained by two approaches: first, maximization of the QPC projection pursuit index to find projection directions (QPC-LVQ), and second, based on the first PCA eigenvector (PCA-LVQ). The cLVQ network learning starts with an empty hidden layer and k (the number of classes) output nodes. First optimization of projections $\boldsymbol{w}_i$ that separate data clusters is described. Centers of these clusters are determined by positions of prototypes $\boldsymbol{t}_j$ placed along the projected line $\boldsymbol{w}_i^T\boldsymbol{x}$. Positions of these prototypes are adjusted using LVQ1 algorithm [11], with additional term that attracts prototypes to the line. Position of the winning prototype $\boldsymbol{t}$ (closest to the training vector $\boldsymbol{x}$) is adjusted using the following formula:

$$\Delta\boldsymbol{t} = \alpha\delta(C_{\boldsymbol{x}}, C_{\boldsymbol{t}})(\boldsymbol{x} - \boldsymbol{t}) + \beta\left((\boldsymbol{t} - \boldsymbol{\mu}) - \boldsymbol{w}^T(\boldsymbol{t} - \boldsymbol{\mu})\boldsymbol{w}\right) \tag{1}$$

The first term of Eq. 1, scaled by coefficient α, comes from standard LVQ1 procedure, where $C_{\boldsymbol{x}}$ denotes class of the vector $\boldsymbol{x}$ and $\delta(C_{\boldsymbol{x}}, C_{\boldsymbol{t}}) = +1$ if $C_{\boldsymbol{x}} = C_{\boldsymbol{t}}$ and -1 otherwise. The second term, scaled by β, pushes prototypes in the direction that brings them closer to the $\boldsymbol{x} = \boldsymbol{w}z + \boldsymbol{\mu}$ line, where $\boldsymbol{w}$ defines a direction in space and $\boldsymbol{\mu}$ is a certain point on the line, for example a mean position of all prototypes $\mu = \frac{1}{N}\sum \boldsymbol{t}_i$. For $\beta = 0$ Eq. 1 reduces to the original LVQ1 algorithm. If $\beta = 1$ the second term of Eq. 1 will put the prototypes exactly along the line, i.e. this corresponds to the orthogonal projection of prototypes on the direction $\boldsymbol{w}$. The effective number of parameters is reduced by adding these constraints.

The first PCA eigenvector calculated on co-linear prototypes (PCA-LVQ) is equivalent to fitting points to a line defining projection direction $\boldsymbol{w}$. Another approach is based on the Quality of Projected Clusters (QPC) index, a supervised algorithm that finds interesting linear projections for multiclass datasets [6]. Assuming that a set of prototypes has been created by the LVQ procedure the QPC index is defined as:

$$QPC(\boldsymbol{w}) = \sum_{i=1}^{n}\left(A_i \sum_{j=1}^{k} \delta(C_{\boldsymbol{x}_i}, C_{\boldsymbol{t}_j}) G\left(\boldsymbol{w}^T(\boldsymbol{x}_i - \boldsymbol{t}_j)\right)\right) \tag{2}$$

where $G(x)$ is a function with localized support and maximum in $x = 0$ (e.g. a Gaussian function), $C_{\boldsymbol{x}_i}$ is the class of vector $\boldsymbol{x}_i$, $C_{\boldsymbol{t}_j}$ the class of prototype $\boldsymbol{t}_j$, and $\delta(C_{\boldsymbol{x}_i}, C_{\boldsymbol{t}_j}) = \pm 1$. This index achieves maximum value when linear projection on the direction defined by $\boldsymbol{w}$ groups vectors $\boldsymbol{x}_i$ from a single $\mathcal{C}_{\boldsymbol{x}_i}$ class into compact clusters close to the prototype $\boldsymbol{t}_j$ associated with this class. Parameters A_i control influence of

each term in Eq. (2). If $A_i = A^+$ for all i that satisfies $C_{\boldsymbol{x}_i} = C_{\boldsymbol{t}_j}$ and $A_i = A^-$ otherwise, then large value of A^- enforces strong separation between clusters, while A^+ has influence mostly on compactness and purity of clusters. Optimization of QPC index is performed by a simple gradiend-based method. Since it may requre a large number of iterations to converge, to speed up learning one can search for $\boldsymbol{w}$ only after some specified number of learning epochs during LVQ optimization phase.

Procedure 1 describes steps performed to expand the network, creating a sequence of hidden nodes, where weights $\boldsymbol{w}_i$ of each successive node represent unique direction in the input space according to the projection pursuit scheme.

Procedure 1 (Construction of the sequence of hidden nodes)

1. start with k output nodes and no hidden nodes;
2. add a new hidden node, optimize its weights $\boldsymbol{w}_i$ and positions of prototypes $\boldsymbol{t}_j$ (see Procedure 2);
3. if the error does not decreases return the network which gives the lowest error;
4. else increase the number of prototypes splitting the node that makes the largest number of errors;
5. orthogonalize training data to the i-th projection, and repeat, starting from step 2.

Each new projection $\boldsymbol{w}_i$ is trained separately on dataset that is orthogonal to all directions $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{i-1}$ found earlier. Another way to gain unique projections is to perform learning only on a subset of training samples misclassified by the network. Only weights of one hidden node $\boldsymbol{w}_i$ are adjusted in step 2, all projections found earlier are fixed. Each change of weights $\boldsymbol{w}_i$ requires proper modification of weights betwen the i-th hidden node and all outputs. Number of output nodes needs to be optimized as well, because each new direction $\boldsymbol{w}_i$ produces different input space projection image. The region covered by the prototype $\boldsymbol{t}_j$ that leads to the largest number of errors (defined by the WTA rule) is divided by adding new prototype associated with the class that has most errors (step 3).

Procedure 2 describes optimization of hidden nodes $\boldsymbol{w}_i$.

Procedure 2 (Searching for an optimal hidden node after adding a new hidden node.)

1. optimize weights $\boldsymbol{w}_i$ and prototype positions $\boldsymbol{t}_j$ to find the best projection that minimizes classification error (using PCA-LVQ or QPC-LVQ algorithms);
2. remove redundant prototypes using condensed nearest neighbor (CNN) method;
3. adjust positions of the remaining prototypes (LVQ learning in the hidden space);
4. if the current projection does not lead to lower error than projections already found return;
5. else increase the number of prototypes splitting the output node that makes the largest number of errors, and go to step 2;

The final network after learning may contain prototypes which are useless (e.g. LVQ learning can expel them far from the training data points). Redundant prototypes that do not decrease accuracy are removed using Hart's condensed nearest neighbor (CNN) method [16, 17], where prototypes are used as reference vectors. Positions of the remaining prototypes are then adjusted with few steps of LVQ to improve cluster separation in the hidden space (after $\boldsymbol{w} \cdot \boldsymbol{x}$ projection).

The cost of computations of Eq. 1 is $O(knd)$ for QPC-LVQ network and $O(knd^2)$ for PCA-LVQ, where d denotes dimensionality of the data, n is the number of instances and k the number of prototypes. For each LVQ learning step the winner prototype must be chosen, which requires $O(nkd)$ operations. Only one hidden node is adapted at a time and the computation of QPC index is linear respect to number of instances, number of prototypes and dimensionality of data. Both LVQ and QPC may require trying a few initializations and performing many iterations, thus evaluation of QPC (or PCA, respectively) should be done only after some LVQ learning. Removal of redundant prototypes using CNN method also requires $O(knd)$ time. Thus the overall computational cost of searching for optimal solution by the methods described above should not exceed $O(knd)$ for QPC and $O(knd^2)$ for PCA, respectively.

4 Results

Results of constrained LVQ models – QPC-LVQ and PCA-LVQ – are compared to the original LVQ, support vector machine (SVM), k-nearest-neighbors (kNN) and the multilayer perceptron network (MLP) classifiers. Accuracy of classification has been estimated using 10 fold stratified cross-validation method for several datasets with different types of features. Following datasets from the UCI repository [8] have been used: Appendicitis, Australian, Breast Cancer Wisconsin, Glass, Heart, Ionosphere, LED (with 500 generated samples), Ljubljana Breast Cancer, Voting and Wine, plus the melanoma skin cancer data collected in the Outpatient Center of Dermatology in Rzeszów in Poland [18]. In addition two Boolean artificial datasets have been used: 10-dimensional parity problem, and a Mirror Symmetry dataset with 1000 strings of 20 bits, where half of them are symmetric. This parity problem is 11-separable while the symmetry problem is 3-separable, but requires a set of exponentially growing weights. Results are collected in Table 4. For QPC-LVQ and PCA-LVQ average number of projections (#P) and number of prototypes (#K) is reported, for LVQ number of prototypes(#K, the size of codebook), for SVM average number of support vectors (#SV), and for MLP networks average number of hidden neurons (#N) are reported.

SVM with Gaussian kernel was used with γ and C parameters fully optimized using an inner 5-fold crossvalidation procedure (the Ghostminer software was used[2]) MLP networks were trained with error backpropagation several times changing the number of hidden nodes from 0 to 20, and selecting the network that gave the best training results. For k-class problems MLP networks with k outputs were used (1-of-k output coding), except for two classes when a single output was used. For LVQ the number of prototypes (codebook vectors) was increased from 1 to 20, and only the best solution is reported. The optimal number of neighbors for kNN method was estimated each time during 5-fold inner crossvalidation, and the best one model was chosen for training. Euclidean measure was used to calculate distance to neighbors.

The SVM is quite accurate for almost all UCI datasets, but failed completely for the parity problem. In most cases large number of support vectors are created even when simple solutions exists. MLP was able to handle all problems with good accuracy and

[2] http://www.fqs.pl/ghostminer/

Table 1. Average classification accuracy for 10-fold crossvalidation test (see text for explanation); bold numbers = accuracy within variance from the best result.

Dataset	**QPC-LVQ**			**PCA-LVQ**			**LVQ**		**SVM**		**kNN**		**MLP**	
	acc.	#P	#K	**acc.**	#P	#K	**acc.**	#K	**acc.**	#SV	**acc.**	#K	**acc.**	#N
Appendicitis	**86.1** $\pm$ 8.5	1.0	2.5	**82.2** $\pm$ 7.4	1.0	3.1	**86.6** $\pm$ 6.8	2	**86.7** $\pm$ 10.8	31.4	**84.0** $\pm$ 6.1	4.8	**87.8** $\pm$ 10.1	3
Australian	**86.1** $\pm$ 4.1	1.0	2.0	**86.1** $\pm$ 4.1	1.0	2.0	**85.6** $\pm$ 4.6	2	**84.9** $\pm$ 1.6	206.1	**85.1** $\pm$ 2.9	8.2	**86.8** $\pm$ 4.9	2
Mirror Symmetry	87.3 $\pm$ 4.2	1.2	3.5	78.1 $\pm$ 11.4	1.8	5.3	75.7 $\pm$ 4.3	12	**98.0** $\pm$ 1.4	296.5	89.3 $\pm$ 2.6	5.8	92.1 $\pm$ 10.7	3
Breast C.W.	**96.2** $\pm$ 1.8	1.0	2.0	**96.2** $\pm$ 2.1	1.0	2.0	**96.5** $\pm$ 2.5	2	**96.6** $\pm$ 1.7	51.1	**97.1** $\pm$ 1.4	5.6	**96.9** $\pm$ 1.0	1
Melanoma	85.7 $\pm$ 6.2	1.0	4.0	70.8 $\pm$ 9.0	1.7	4.3	76.8 $\pm$ 7.2	4	85.2 $\pm$ 5.7	240.3	86.0 $\pm$ 7.4	1.0	**96.0** $\pm$ 2.7	4
Glass	**60.4** $\pm$ 10.3	1.1	4.5	58.9 $\pm$ 7.3	1.4	4.0	**66.3** $\pm$ 10.5	7	**64.9** $\pm$ 6.2	283.6	**68.8** $\pm$ 9.2	1.4	**69.2** $\pm$ 10.4	7
Heart	**78.9** $\pm$ 8.5	1.0	2.0	**80.7** $\pm$ 9.0	1.0	2.0	**82.2** $\pm$ 8.6	2	**81.5** $\pm$ 9.1	101.5	**78.5** $\pm$ 7.6	8.5	**79.6** $\pm$ 9.8	1
Ionosphere	78.4 $\pm$ 8.0	1.0	3.1	75.5 $\pm$ 8.5	1.1	3.5	81.9 $\pm$ 7.0	4	**93.5** $\pm$ 4.7	61.0	84.0 $\pm$ 7.7	1.2	81.5 $\pm$ 5.5	5
Iris	**96.0** $\pm$ 4.4	1.0	3.0	**94.7** $\pm$ 4.0	1.0	3.0	**97.3** $\pm$ 3.3	3	**96.7** $\pm$ 4.7	39.6	**94.5** $\pm$ 6.9	5.8	**94.7** $\pm$ 8.2	0
L. Breast	**72.6** $\pm$ 4.4	1.0	2.0	**74.0** $\pm$ 6.9	1.0	2.1	**74.7** $\pm$ 6.0	3	**73.3** $\pm$ 9.6	143.6	**73.7** $\pm$ 5.5	6.9	**72.2** $\pm$ 6.7	1
LED500	58.5 $\pm$ 8.0	1.1	9.6	45.5 $\pm$ 8.6	1.5	8.2	**72.0** $\pm$ 5.1	10	65.2 $\pm$ 5.5	664.1	**71.2** $\pm$ 6.6	8.5	**70.8** $\pm$ 7.5	0
Parity 10	**96.1** $\pm$ 3.9	1.0	6.8	**97.6** $\pm$ 0.8	1.0	7.5	51.3 $\pm$ 4.9	8	44.2 $\pm$ 5.7	921.2	80.7 $\pm$ 3.4	20.0	82.9 $\pm$ 17.2	14
Voting	**95.2** $\pm$ 3.5	1.0	2.0	90.6 $\pm$ 5.5	1.3	2.5	**93.8** $\pm$ 2.7	5	**95.9** $\pm$ 2.4	57.0	93.3 $\pm$ 3.2	4.6	**95.9** $\pm$ 2.4	1
Wine	**96.0** $\pm$ 5.7	1.9	3.1	94.9 $\pm$ 4.7	2.0	3.0	**97.7** $\pm$ 2.8	4	**96.6** $\pm$ 2.9	63.7	95.0 $\pm$ 4.1	6.2	**98.3** $\pm$ 2.7	0

a small number of neurons, but at a rather high cost, requiring many initializations with increasing number of hidden nodes from 0 to 20. Setting the right number of hidden nodes has been very important to reach high quality solutions but also reduces time to convergence. The number of parameters adapted during training of MLPs with one hidden layer is $(d+1)h + (h+1)k$, where d denotes number of inputs, h - number of hidden nodes and k the number of output nodes.

The QPC-LVQ algorithm was able to find in most cases, with very few hidden nodes, solution of comparable accuracy to all other classifiers. Moreover, for some datasets (mirror symmetry, melanoma and parity) it has significantly outperformed original LVQ1. Only for the LED data the QPC-LVQ model has been too simple and gave worse results. The PCA-LVQ performs worse in most cases, comparison of PCA-LVQ and QPC-LVQ using Wilcoxon [19] test shows significant differences, at confidence level of 95%, in terms of accuracy (p-value of 0.048) and number of parameters (p-value of 0.003) both in favor of QPC-LVQ. QPC-LVQ compared with LVQ, SVM and kNN shows no significant difference in accuracy, giving p-value equal to 0.62, 0.07 and 0.87, respectively. Only for the MLP Wilcoxon test find significant differences (p-value of 0.048) against QPC-LVQ. On the other hand MLP and LVQ needed on average 3 times more parameters than QPC-LVQ.

5 Discussion

Drawbacks of existing learning methods that fail on relatively simple problems, as well as on highly-non separable problems, may be remedied by structural risk minimization approach, in which models of growing complexity are introduced by relaxing various constraints. One such approach has been analyzed here, based on the constrained LVQ model, allowing for systematic relaxation of constraints to create progressively more

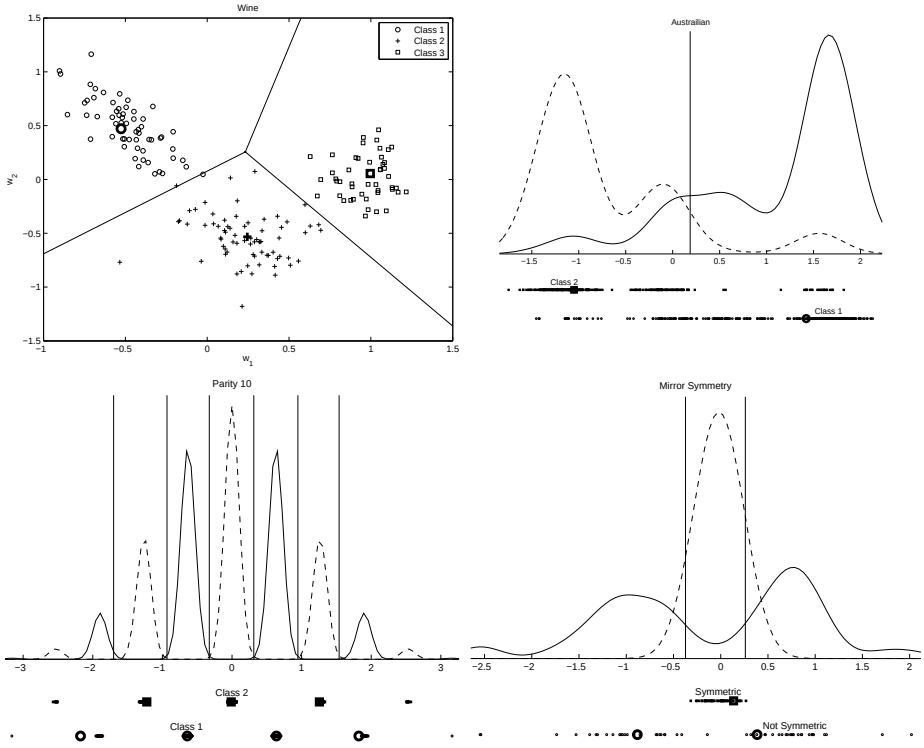

Fig. 1. Visualization of hidden layer of the QPC-LVQ network for Wine, Australian, Parity 10 and the Mirror Symmetry dataset.

complex solutions. Such models may learns optimal solutions to complex Boolean functions as well as problems that are almost linearly separable using single feature, while unconstrained LVQ converges to worse solutions. Two versions of such approaches have been presented here, based on PCA and QCP algorithm to find good projection lines. Straightforward LVQ modification allows for reduction of the number of effective parameters by forcing the LVQ prototypes to stay near those projection lines. In tests one or two lines have usually been sufficient, with very small number of prototypes.

Great advantage of algorithms developed here is that the hidden layer contains only one or two nodes, and therefore activations of those nodes may be used to visualize inherent dataset relations. Fig. 1 illustrates this, with prototypes and decision boundaries set by prototypes (the winner-takes-all rule leads to Voronoi tessellation) for Wine, Australian, Parity 10 and the Mirror Symmetry datasets. Such visualizations allows to estimate confidence of classification for each vector. The main point ot this paper was to show that the strategy of starting with constrained models and gradually relaxing constraints (in Eq. 1 decreasing β to 0) should help to discover simple models that more complex approaches will miss, and also in some complex cases find better solutions. This is indeed observed in Tab. 4: for some data LVQ-QCP has ineed advantages, while for other data that require more complex decision borders unconstrained optimization is

significantly more accurate. Thus it is recommended to always use methods that allow for such control of model complexity.

References

1. Duch, W.: k-separability. Lecture Notes in Computer Science **4131** (2006) 188–197
2. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer Verlag (2006)
3. Grochowski, M., Duch, W.: Learning highly non-separable Boolean functions using Constructive Feedforward Neural Network. Lecture Notes in Computer Science **4668** (2007) 180–189
4. Schölkopf, B., Smola, A.: Learning with Kernels. Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge, MA (2001)
5. Grochowski, M., Duch, W.: A Comparison of Methods for Learning of Highly Non-Separable Problems. Lecture Notes in Computer Science **5097** (2008) 566–577
6. Grochowski, M., Duch, W.: Projection Pursuit Constructive Neural Networks Based on Quality of Projected Clusters. Lecture Notes in Computer Science **5164** (2008) 754–762
7. Duch, W., Blachnik, M.: Fuzzy rule-based systems derived from similarity to prototypes. In Pal, N., Kasabov, N., Mudi, R., Pal, S., Parui, S., eds.: Lecture Notes in Computer Science. Volume 3316. Physica Verlag, Springer, New York (2004) 912–917
8. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
9. Michie, D., Spiegelhalter, D.J., Taylor, C.C.: Machine learning, neural and statistical classification. Elis Horwood, London (1994)
10. Duch, W., Setiono, R., Zurada, J.: Computational intelligence methods for understanding of data. Proceedings of the IEEE **92** (2004) 771–805
11. Kohonen, T.: Self-organizing maps. Springer-Verlag, Heidelberg Berlin (1995)
12. Biehl, M., Ghosh, A., Hammer, B.: Dynamics and generalization ability of lvq algorithms. J. Mach. Learn. Res. **8** (2007) 323–360
13. Vapnik, V.: The Nature of Statistical Learning Theory. Springer-Verlag, New York (1995)
14. Duch, W., Adamczak, R., Diercksen, G.: Classification, association and pattern completion using neural similarity based methods. Applied Mathematics and Computer Science **10** (2000) 101–120
15. Duch, W.: Similarity based methods: a general framework for classification, approximation and association. Control and Cybernetics **29** (2000) 937–968
16. Hart, P.E.: The condensed nearest neighbor rule. IEEE Transactions on Information Theory **114** (1968) 515–516
17. Jankowski, N., Grochowski, M.: Comparison of instance selection algorithms. i. algorithms survey. Lecture Notes in Computer Science **3070** (2004) 598–603
18. Hippe, Z.: Data mining in medical diagnosis. In Kącki, E., ed.: Computers in Medicine. Volume 1., Łódź, Poland, Polish Society of Medical Informatics (1999) 25–34
19. Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics **1** (1945) 80–83