Neural Network Methods for Rule Induction

Ricardo Bezerra de Andrade e Silva, Teresa Bernarda Ludermir
{rbas, tbl}@di.ufpe.br
Departamento de Informática
Universidade Federal de Pernambuco – UFPE, Brazil

Abstract

Local basis function networks are a useful category of classifiers, with known variations developed in neural networks, machine learning and statistics communities. The localized range of activation of the hidden units have many similarities with rule-based representations. Neurofuzzy systems are a common example of an framework that explicitly integrates these approaches. Following this concept, we study alternatives for the development of rule-neural hybrid systems with the purpose of inducing robust and interpretable classifiers. Local fitting of parameters is done by a gradient descent optimization that modifies the covering produced by a rule induction algorithm. Two tasks are accomplished: how to select a small number of rules and how to improve precision. Accuracy is not the only target: indeed, the use of this architecture is better suited when one wants to achieve a good compromise between classification performance and simplicity.

1. Motivation

Most machine learning and data mining methods can be seen as extensions or hybrids of a few basic principles [7]. However, there has been difficulties of communication among researchers and practitioners due most to the use of alternative nomenclatures and motivation principles. It is clearer now that potential synergies exist in methods with apparently different inductive bias and function representation.

The study of hybrid learning systems [10][20] sheds some light in these differences. The term itself is very imprecise and basically means a system that combines two or more techniques in a single framework, which makes necessary the comprehension of the similarities among those methodologies.

There are many ways to perform this combination [12] [20], and depending on subjective points of view, one can accept an architecture as a hybrid system or not. However, it will not make sense if it does not have some kind of advantage over the original methods. It is very tempting to create a hybrid system just because it is possible. Further, there can be an increased danger of misuse, since it is necessary some expertise in different domains [11].

Under this context, it is discussed here an approach for inducing models of local basis functions by the combination of fast rule induction algorithms of partition or covering of an input space and slow gradient-based optimization methods. The purpose is the refinement of a interpretable model of classification (also known as theory refinement [22][28]) with special emphasis on model simplification and local measures of attribute importance.

2. Previous work

Andrews and Geva [1] survey models of local function networks for classification and their relationship to rule-based representations. A local response unit can be interpreted as a propositional rule of the form

IF $(x_1 \in [x_{1min}, x_{1max}]) \wedge ... \wedge (x_n \in [x_{nmin}, x_{nmax}])$
THEN pattern belongs to class c

where each range $[x_{kmin}, x_{kmax}]$, $k \in \{1, ..., n\}$, is a function of the center and spread parameters of the unit. The class c is the class of the majority of the data bounded by this rule. Although the authors focus specially in non-overlapping units that use indicator (binary) functions, in models that apply smooth functions (like symmetric gaussians), these "rules" will overlap, and the classification is given by a combination of responses of the units in the network.

Distance-based methods like radial basis functions (RBFs, [21][3]) and probabilistic neural networks (PNNs, [26]) are often criticized for being more sensitive to the curse of dimensionality than, for example, multilayer perceptrons, specially when there are irrelevant attributes [24].

Initializing a local network using the partition generated by a tree or rule induction algorithm [16] partially solves this problem, since the hyper-rectangles generated by theses methods are often described by a small subset of the available features that are considered important for that region. We have approximated measures of local relevance of each feature since the spread (or even presence) of an attribute varies in each element of the rule base. Many tasks can be solved by this approach: for example, [5] describes a fuzzy rule set derived from a ID3 tree for handwritten numeral recognition. Smyth *et al.* [25] perform probability density estimation in subspaces determined by the leaves of a induced decision tree. The RecBF model [14] is a rectangular basis function network that is fitted to the data by a constructive on-line procedure, similar to classical concept learning algorithms for induction of hyper-rectangles [19]. The output layer, however, is not a linear combination of the complete hidden layer, as we can find in radial RBFs, but it is analog to PNNs, where each rectangular basis is linked only to the output unit of the same class.

Another point of view is a generalization of nearest neighbor algorithms, where instead of using the whole data base, one induces "general cases" for reducing memory amount and retrieval time required by these models. Domingos [6] describes a bottom-up approach for combining data points in hyper-rectangles when there is enough similarity according to a given measure. The final model is formed of hyper-rectangles (rules) and single instances. Wettschereck and Dietterich [30] discuss some problems and solutions for specific nearest hyper-rectangles algorithms. Friedman *et al.* [8] present the k-d tree as a fast technique to nearest neighbor search.

Fast rule-based initialization of local basis networks offers another advantage when it is considered as a good starting point for a supervised learning of centers and dispersion parameters. In a fully supervised learning procedure, Moody and Darken experiments [21] resulted in units of very large receptive fields, losing the concept of locality, and taking much more time than the combination of clustering and least-squares fitting of output weights. A survey of feature weighting methods in the context of lazy learning algorithms is presented in [29]

Finally, the equivalence of basis functions and rules is well known in the neurofuzzy literature. Kosko [15] provides an overview, including a chapter about optimization of fuzzy rules using gradient descent.

3. Proposed approach

What typical techniques used by neural networks methods can offer for better rule induction processes? Two problems that can be explored are:

- the number of rules produced by rule induction algorithms is often high. A big number of rules is not good at the point of view of comprehensibility. Also, many of them cover few instances and are very error prone, but can not just be pruned because can increase even more the misclassification cost. This is the so called *small disjuncts* problem [13];

- the boundaries of each rule can be optimized in some degree for increased accuracy;

In this paper, we study how a parameterization of a rule base in terms of a neural network can be useful for reducing the complexity (number of rules) of a model and achieving some improvement in classification accuracy. For accomplishing these tasks, we cast a rule base produced by a given algorithm as an architecture analog to an PNN: the output layer is constructed using an unit for each class, the hidden units are labeled according to one of the classes and linked only to the corresponding output unit.

Each hidden unit is built from a specific rule r as a local basis function. Let D_r be the set of examples in the training set bounded by the corresponding rule, where it is considered only those examples that are of the same class of r, and let $\mathbf{x}_r$ be an element of D_r. We describe here a parameterization for numeric attributes only. The unit parameters are initialized as follow:

- the class of the unit is the same of the rule;

- the center parameters m_{rk} are computed for each attribute k:

  1. if $a \leq x_{rk} \leq b$, $a, b \in R$, m_{rk} is given by the median of x_{rk} in D_r;
  2. if $x_{rk} \leq b$, $b \in R$, m_{rk} is given by the lowest possible value of the attribute k;
  3. if $x_{rk} \geq a$, $a \in R$, m_{rk} is given by the highest possible value of the attribute k;

- in general, two spread parameters, s_{rk+} and s_{rk-}, are computed for each attribute k. Let h_k and l_k be the highest and lowest value of attribute k in D_r, respectively. Then, we have $s_{rk+} = |m_{rk} - h_k| / s$ and $s_{rk-} = |m_{rk} - l_k| / s$. Good values of s are usually 2 or 3. If k is not used in the definition of the rule, we consider $s_{rk+} = s_{rk-} = \infty$. If k is used in the rule as an one-sided interval, $s_{rk+} = \infty$ or $s_{rk} = \infty$, for lower bounded or upper bounded intervals, respectively;

The activation function $f_r(\mathbf{x})$ of a hidden unit r given an input $\mathbf{x}$ is:

$$f_r(\mathbf{x}) = \min_k f_{rk}(\mathbf{x}) \quad (1)$$

$$f_{rk}(\mathbf{x}) = e^{-\frac{(m_{rk}-x_k)^2}{s_{rk+}^2}}, \text{ if } x_k > m_{rk} \text{ and } s_{rk+} \neq \infty \quad (2)$$
$$= e^{-\frac{(m_{rk}-x_k)^2}{s_{rk-}^2}}, \text{ if } x_k < m_{rk} \text{ and } s_{rk-} \neq \infty$$
$$= 1, \text{ otherwise}$$

where $k \in \{1, 2, ..., n\}$, and n is the number of domain attributes. This is very similar to traditional models of fuzzy neural networks.

The weight w_r, linking unit r to the output unit of the same class, is computed as

$$w_r = \frac{n_+}{n_+ + n_-} \quad (3)$$

where n_+ is the number of examples in the training set that are of the same class of unit r and are closer to r than to any other unit (r has the highest activation among all hidden units for these examples); n_- is the number of examples in the training set that are of a different class and are closer to r than to any other unit.

The activation of an output unit $o_i(\mathbf{x})$ is calculated as:

$$o_i(\mathbf{x}) = \frac{s_i(\mathbf{x})}{S(\mathbf{x})} \quad (4)$$

$$s_i(\mathbf{x}) = \sum_r f_r(\mathbf{x}) w_r \quad (5)$$

$$S(\mathbf{x}) = \sum_m s_i(\mathbf{x}) \quad (6)$$

where $i \in \{1, 2, ..., m\}$, and m is the number of classes.

For the two tasks approached in this work, network simplification and accuracy increase, gradient descent optimization of network parameters is performed. The error function is the mean squared error. The free parameters are the set of $\mathbf{s}_r$. The output weights are recomputed at each step according to (3). In this process, we search for a good compromise between the number of rules and accuracy gain. A cycle of pruning and fitting is performed as a control strategy. The process is summarized below.

Task: find a good compromise between classification performance and simplicity

1. Parameterize a given rule base as a neural network;
2. Eliminate the unit from the network with the minor weight. If the error in the training set does not increase by more than a small percentage, repeat this step. Otherwise, do not eliminate this unit;
3. Optimize the networks with a gradient-based learning algorithm (gradient descent, conjugate gradients, etc.);
4. Perform a pruning analog to step 2;
5. If in Step 4 we pruned some unit, go back to step 3. Otherwise, stop the process and output the network;

4. Experiments and results

We performed experiments with some of the data sets used in [17] for the comparison of thirty three classifiers. These data sets, already divided in ten folders for cross-validation procedures, can be found in http://www.stat.wisc.edu/~limt/treeprogs.html. They are originally from the UCI Machine Learning Repository [2].

The rule induction algorithm utilized for network initialization is the Patient Rule Induction Method (PRIM, [9]). It was chosen because it is a robust, non-greedy, state-of-the-art method. The fact it performs a relative slow adjustment of boundaries, in contrast with the very greedy partition used in most rule and tree-based learning techniques, gives a fairer comparison with gradient-based optimization techniques. The conflict resolution used when a given case is covered by more than one rule is to label it according to the class of the rule with the highest mean. The mean is defined in an analog way to the output weights (3).

The sets used are the Wisconsin breast cancer database (bcw, 683 cases), BUPA live disorders (bld, 345 cases), PIMA indian diabetes (pid, 532 cases) and StatLog vehicle silhouette (veh, 846 cases). Table 1 shows results for the rule induction alone, and table 2, results for the neural network trained by the first procedure. The increase in error percentage used for allowing pruning is 1%.

	PRIM		
	error	std. deviation	#rules
bcw	0.0541	0.02	7
bld	0.3305	0.09	16.5
pid	0.2390	0.02	17.5
veh	0.2885	0.03	34.5

Table 1: results for the PRIM algorithm alone with default options. The first column is the name of the database, followed by the mean classification error in test set, the error standard deviation and the mean number of rules that remains after the same pruning used in the second step of network training.

	Neural casting, prune and fit procedure			
	error	std. dev.	#rules	#params.
bcw	0.0438	0.02	6	17
bld	0.3103	0.03	11	28
pid	0.2315	0.02	10	42
veh	0.2680	0.04	28.5	135

Table 2: results for the neural network process of optimization. The last column show the mean number of parameters.

5. Discussion and future work

The procedure described was responsible for gains in prediction accuracy as well as a decrease in the number of used rules. Since most algorithms induce rules one at a time, a process that refines the classification capabilities of the model as a whole is an useful complement. As usual, some datasets give better results than others. In comparison with the results reported in [17], the combination of PRIM and the fitting strategies described remains in a good rank. It must be emphasized that we are working under some restrictions of interpretability: the local basis functions are axis-parallel so they can be interpreted as a set of rules. Models like that decision trees that use linear combination of attributes at each node often give lower classification error and a less number of nodes, but the model representation may not be interesting for issues of interpretation.

Initializing a local basis network by using the output generated by a rule or tree induction algorithm has clear advantages over methods like unsupervised learning: it is not necessary to choose the number of clusters, nor defining a similarity metric; only a subset of the variables are used at each basis; they can be associated with a class label, adding more interpretability. However, it usually results in more units than those produced by optimized clustering procedures. That was one of the motivations for the present work.

Depending on the application, accuracy gains as those reported in this work may not be satisfactory. Better results can be expected with more sophisticated pruning methods and control strategies of overfitting. Even when the number of parameters is not very high in comparison with the amount of available data, techniques as training with early stopping can result in lower generalization error. Lowe [18] and Tresp *et al.* [27] suggest the use of weight decay. A possible approach for reducing the number of parameters is initializing the network with non-overlapping regions, like those induced by decision trees. Parameter sharing can be done if one represents the boundary of two adjacent regions as a single parameter. These topics will be treated in a future work.

The neural network representation of hidden units do not use information of probability density of the cases bounded in a given hyper-rectangle: the parameters of spread are a function of the boundaries of the boxes, independent of how they are distributed inside these boundaries. Probability estimates are not necessary for good classification, and often can be unreliable in comparison with simpler techniques [4]. The only information concerning distribution that is computed is the median, used as the measure of center for those regions that are limited in both upper and lower boundaries, but it is expected that the model is not sensitive for this choice, as early experiments showed. Asymmetric parameters of spread are used as a substitute for adaptive centers, making the refinement of the model simpler, where all parameters are of the same kind. Other issues on local learning feature weights and asymmetric measures of distance are raised in [23].

The activation function (1) is not so common as a kernel that multiplies the contribution of all attributes, but has many advantages: the gradient descent optimization is considerably faster, it is easier to interpret, it is a more natural extension to the rule-based representation (it resembles a hyper-rectangle with soft boundaries) and induction (where at each step a boundary is changed without affecting the others). For the tasks discussed here, the multiplicative kernel achieved similar results, so the function (1) was preferred.

The weights connecting the hidden layer to the output layer are not treated as free parameters. Fully connecting these layers and optimizing them could give better accuracy rates. However, we choose to keep the model as simple as possible, making interpretation easier.

As another possible extension of this work, it can be studied how the combination of the refined rules and the introduction of misclassified cases as extras units can improve accuracy in a manner similar to hybrid systems of nearest neighbors and nearest hyper-rectangles [6].

References

[1] R. Andrews and S. Geva. “Rules and local function networks”. *Rules and Networks – Proceedings of the Rule Extraction from Trained Artificial Neural Networks Workshop* 1996.

[2] C. Blake, E. Keogh and C. J. Merz. *UCI Repository of machine learning databases.* Irvine, CA: University of California, Departament of Information and Computer Science. URL: http://www.ics.uci.edu/~mlearn/MLRepository.html.

[3] C. M Bishop. *Neural Networks for Pattern Recognition*, Oxford Press, 1995.

[4] V. Cherkassky and F. Mulier. *Learning From Data: Concepts, Theory and Methods*, Wiley Interscience, 1998.

[5] Z. Chi, and H. Yan. “ID3-derived fuzzy rules and optimized defuzzification for handwritten numeral recognition”. *IEEE Transactions on Fuzzy Systems* 4, 24-31, 1996.

[6] P. Domingos. “Unifying instance-based and rule-based induction”. *Machine Learning*, 24, 141-168, 1996.

[7] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. “From data mining to knowledge discovery: an overview”. *Advances in Knowledge Discovery and Data Mining*, AAAI Press, 1-34, 1996.

[8] J. H. Friedman, J. L. Bently, and R. A. Finkel. “An algorithm for finding best matches in logarithmic expected time”. *ACM Transactions on Mathematical Software* 3, 209-226, 1977.

[9] J. H. Friedman and N. I. Fisher. “Bump hunting in high dimensional data (with discussion)”. *Statistics and Computing* (to appear), 1999.

[10] C. L. Giles and R. Sun and J. M. Zurada. "Neural networks and hybrid intelligent models: foundations, theory, and applications". Guest Editorial, *IEEE Transactions on Neural Networks* 9, 721-723, 1998.

[11] A. Gray. *Hybrid Systems FAQ*. URL: http://divcom.otago.ac.nz:800/COM/INFOSCI/SMRL/people/andrew/publications/faq/hybrid/parts/Info.htm, 1997.

[12] S. Goonatilake and S. Khebbal, eds. *Intelligent Hybrid Systems*, John Wiley & Sons, 1995.

[13] R. C. Holte, L. E. Acker, and B. W. Porter. “Concept learning and the problem of small disjuncts”. *Proceedings of the International Conference of Artificial Intelligence*, 813-818, 1989.

[14] K. Huber and M. Berthold. “Building Precise Classifiers with Automatic Rule Extraction”. *Proceedings of the IEEE International Conference on Neural Networks* 3, 1263-1268, 1996.

[15] B. Kosko. *Fuzzy Engineering*. Prentice Hall, 1997.

[16] M. Kubat. "Decision trees can initialize radial-basis function networks". *IEEE Transactions on Neural Networks* 9, 813-821, 1998.

[17] T.-S. Lim, W._Y. Loh, and Y.-S. Shih. "A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms". *The Machine Learning Journal* (to appear), 1999.

[18] D. Lowe. "Similarity metric learning for a variable-kernel classifier". *Neural Computation* 7, 72-85, 1995.

[19] T. Mitchell. *Machine Learning*, McGraw Hill, 1997.

[20] K. McGarry, S. Wermter and J. MacIntyre. “Hybrid neural systems: from simple coupling to fully integrated neural networks”. *Neural Computing Surveys* 2, 62-93, 1999.

[21] J. Moody and C. J. Darken. “Fast learning in networks of locally-tuned processing units”. *Neural Computation* 1, 281-294, 1989.

[22] S. Ramachandran, and R. Mooney. **“**Theory Refinement for Bayesian Networks with Hidden Variables”. *Proceedings of the Fifteenth International Conference on Machine Learning*, 454-462, 1998.

[23] F. Ricci, and P. Avesani. “Learning a Local Similarity Metric for Case-Based Reasoning”. *Proceedings of the First International Conference on Case-Based Reasoning*, 301-312, 1995.

[24] W. S. Sarle, ed. *Neural Network FAQ*, periodic posting to the Usenet newsgroup comp.ai.neural-nets, URL: ftp://ftp.sas.com/pub/neural/FAQ.html, 1999.

[25] P. Smyth, A. Gray, and U. M. Fayyad. “Retrofitting decision tree classifiers using kernel density estimation”. *Proceedings of the Twelfth International Conference on Machine Learning*, 506-514, Morgan Kaufmann, 1995.

[26] D. Specht. “Probabilistic neural networks”. *Neural Networks* 3, 109-118, 1990.

[27] V. Tresp, J. Hollatz, and S. Ahmad. “Network structuring and training using rule-based knowledge”. *Advances in Neural Information Processing Systems* 5, 871-878, Morgan Kaufmann, 1993.

[28] G. G. Towell and J. W. Shavlik. “Knowledge-based artificial neural networks”. *Artificial Intelligence* 70, 119-165, 1994.

[29] D. Wettschereck, D. W. Aha, and T. Mohri. “A review and comparative evaluation of feature weighting methods for lazy learning algorithms”. *Artificial Intelligence Review* 11, 273-314, 1997.

[30] D. Wettschereck and T. G. Dietterich. “An experimental comparison of the nearest-neighbor and nearest-hyperrectangle algorithms”. *Machine Learning* 19, 1-25, 1995.