Mathware & Soft Computing 11 (2004) 109-124

An Evolutionary Approach to Constraint-Regularized Learning

E. Hüllermeier
Informatics Institute, Marburg University
e-mail: eyke@informatik.uni-marburg.de

I. Renners and A. Grauel
Center of Computational Intelligence and Cognitive Systems
59505, Soest, Lübecker Ring 2, FB 16
e-mail: ingo.renners@syetec.com and *AdolfGrauel@web.de*

Abstract

The success of machine learning methods for inducing models from data crucially depends on the proper incorporation of background knowledge about the model to be learned. The idea of *constraint-regularized learning* is to employ fuzzy set-based modeling techniques in order to express such knowledge in a flexible way, and to formalize it in terms of fuzzy constraints. Thus, background knowledge can be used to appropriately bias the learning process within the regularization framework of inductive inference. After a brief review of this idea, the paper offers an operationalization of constraint-regularized learning. The corresponding framework is based on evolutionary methods for model optimization and employs fuzzy rule bases of the Takagi-Sugeno type as flexible function approximators.

1 Introduction

Learning models from data is the central theme in several research fields, including machine learning and inferential statistics. Model induction may serve different purposes, such as accurate *prediction* of future observations or intelligible *description* of dependencies between variables in the domain under investigation. The *data-driven* approach to model construction can be contrasted with a *knowledge-driven* one, where a model is built through the acquisition and formalization of expert knowledge. The classical expert system paradigm was mainly based on this latter approach, though in the course of time the knowledge acquisition process turned out to be a real bottleneck.

In fact, knowledge acquisition will usually be more successful when exploiting both, background knowledge as well as observed data. And indeed, both of these

sources are usually available in applications, at least to some extent. Combining them in an adequate manner, however, is a challenging problem not supported by many methods.[1]

One possibility of incorporating background knowledge in learning from data is to restrict the class of potential candidate models – called the *hypothesis space* in machine learning – to models satisfying particular structural assumptions, such as a linear dependency between input and output variables of a system. The incorporation of (at least approximately) valid structural assumptions of this type will usually improve the learning process. On the other hand, incorrect assumptions can be misleading and might badly deteriorate results.

Indeed, more often than not the available background knowledge does hardly justify strong assumptions about the global structure of a model. Rather, the knowledge is usually *local* in the sense that it refers to local properties of the model under consideration, e.g. to subregions of the input space. Most of the time, the knowledge is also *imprecise* and *vague*, possibly expressed in a qualitative rather than a quantitative way, e.g. in terms of natural language rather than mathematical formulas. In [4], one of the authors has introduced a method which appears to be particularly suitable for utilizing this type of knowledge in learning from data. The simple yet powerful idea of this method, subsequently referred to as *constraint-regularized learning* (CRL), is to embed fuzzy modeling in so-called *regularized learning*: Using fuzzy set-based (linguistic) modeling techniques, background knowledge is formalized in terms of flexible constraints. Thus, learning actually comes down to solving an optimization problem, namely to finding an optimal tradeoff between satisfying the constraints and reproducing the data.

The objective of this paper is to operationalize the conceptual framework of CRL as outlined in [4]. Particularly, this means specifying an adequate model class, which is flexible enough to comprise (local) structural constraints, and providing an efficient and easy way to implement optimization (search) method for that model class. After some brief comments on supervised learning and regularization in general (Section 2), we review the basic concepts underlying CRL in Section 3. In Section 4, we present a framework in which CRL is realized by means of evolutionary methods. To illustrate this framework, we finally present an example in Section 5.

2 Supervised Learning and Regularization

2.1 Supervised Learning

In supervised learning, one usually seeks to induce a (functional) relationship between an input space $\mathfrak{D}_X$ and an output space $\mathfrak{D}_Y$, proceeding from a set of observed data in the form of a (finite) sample

$$S = \{ (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n) \} \in (\mathfrak{D}_X \times \mathfrak{D}_Y)^n. \tag{1}$$

[1] Notable exceptions include Bayesian learning, inductive logic programming, and knowledge-based neurocomputing.

Depending on the nature of the data involved, different types of learning problems can be distinguished. The problems considered most often are *classification* and *regression*. In classification, the response is categorical, i.e. the output space $\mathfrak{D}_Y$ is a finite and unordered set. In regression, the response variable is taken from a numeric scale, usually the real number line. Subsequently, we shall focus on regression as a learning task, even though the method of constraint-regularized learning can be developed for classification as well.

A set of candidate mappings $\mathfrak{D}_X \to \mathfrak{D}_Y$ is commonly referred to as the *hypothesis space*. For elements of this space we shall employ the terms *hypothesis* and *model* interchangeably. Without loss of generality, we can assume the hypothesis space $\mathcal{H}$ to be a parameterized class of functions, that is

$$\mathcal{H} = \{h(\cdot, \omega) \,|\, \omega \in \Omega\}.$$

Here, ω is a parameter (perhaps of very high or even infinite dimension) that uniquely identifies the function $h = h(\cdot, \omega)$, and $h(x, \omega)$ is the value of this function for the input x.

Passing from a sample (1) to a complete model is a problem of *induction*, and the selection of an (apparently) optimal hypothesis $h_0 \in \mathcal{H}$ is guided by some *inductive principle*. An important and widely used inductive principle is *empirical risk minimization* (ERM) [15]. The risk of a hypothesis $h(\cdot, \omega)$ is defined as

$$R(\omega) =_{\text{def}} \int_{\mathfrak{D}_X \times \mathfrak{D}_Y} L(y, h(x, \omega))\, dP(x, y),$$

where L is a loss function and P is a probability measure over the input-output space $\mathfrak{D}_X \times \mathfrak{D}_Y$, specifying the probability of observing an input vector x together with an output y. The empirical risk is an approximation of the true risk:

$$R_{emp}(\omega) \;=_{\text{def}}\; \frac{1}{n} \sum_{i=1}^{n} L(y_i, h(x_i, \omega)) \;. \tag{2}$$

The ERM principle prescribes to choose the parameter ω_0 resp. the associated hypothesis $h_0 = h(\cdot, \omega_0)$ that minimizes (2).

2.2 Regularization

The specification of a suitable hypothesis space is crucial for successful learning. A point of critical importance in this connection concerns the *complexity* of $\mathcal{H}$. On the one hand, $\mathcal{H}$ must be "rich" enough to guarantee the existence of a good approximation, i.e. a model with a low risk. On the other hand, $\mathcal{H}$ must not be too flexible in order to avoid the problem of *overfitting*. Roughly speaking, overfitting means adapting a model to the data in too exact a manner. Quite often, such models reproduce the sample data rather well but perform poorly when it comes to *generalizing* beyond that data. In the case of noisy data, for instance, very flexible models tend to reproduce not only the deterministic part of the underlying (true) model but also the noise itself.

One approach to counteract the problem of overfitting is *regularized learning*. The basic idea of this approach is to "punish" candidate models that have a high complexity. More specifically, let $\phi(\omega) = \phi(h(\cdot, \omega))$ be a measure of the complexity of the hypothesis $h(\cdot, \omega)$.[2] The *penalized risk* to be minimized for the *regularization inductive principle* is then given by a weighted sum of the empirical risk and a penalty term:

$$R_{pen}(\omega) =_{\text{def}} R_{emp}(\omega) + \lambda \cdot \phi(\omega). \tag{3}$$

As can be seen, the penalized risk is intended to find a tradeoff between the accuracy and the complexity of a model. This tradeoff is controlled by the parameter λ, called the regularization parameter. The model $h(\cdot, \omega_0)$ minimizing (3) is supposed to have a smaller *predictive risk* than the function minimizing the empirical risk. In other words, it is assumed to be a better generalization in the sense that it predicts future outputs more accurately.

3 Constraint-Regularized Learning

The regularization framework presented in the previous section provides a means for avoiding the problem of overfitting. Even though overfitting is the much more frequent problem, there is of course also a risk of *underfitting* a model. An example, especially interesting as a motivation for the method of constraint-regularized learning, is illustrated in Fig. 1. Here, the function to be learned has a "sharp peak" around $x = 0$. Unfortunately, the data in the random sample does not give any hint at this peak, since none of the x_i-values is close enough to 0. Consequently, the data is approximated by a rather flat curve (in this example, an approximation with b-splines was used, see Section 5).

The approximation in the above example could clearly be improved by incorporating knowledge about the peak in the learning process. This type of knowledge is obviously quite different from knowledge which is typically expressed in terms of hypothesis spaces or complexity constraints. First, it is *local* in the sense that it refers to local properties of the model to be learned. Second, it is *vague* in the sense that a peak is a fuzzy rather than a crisp concept: For the shape of a curve there is no clear boundary between having a (sharp) peak and not having a peak.

The method of constraint-regularized learning is motivated by the fact that background knowledge does indeed often refer to local aspects of a model, characterizing them in a vague rather than an exact manner, e.g. in terms of natural language descriptions. The basic idea of CRL is to formalize this type of background knowledge as flexible (fuzzy) constraints and to use these constraints in place of the complexity measure ϕ in the penalized risk functional. Thus, one arrives at the following measure:

$$R_{pen}(\omega) =_{\text{def}} R_{emp}(\omega) - \lambda \cdot C(\omega), \tag{4}$$

[2] It is worth mentioning that "intuitively" complex models are not necessarily flexible in the sense of data adaptation, and vice versa. Moreover, theoretically well-founded complexity measures such as the VC-dimension [15] actually refer to the flexibility of complete hypothesis spaces rather than single hypotheses.

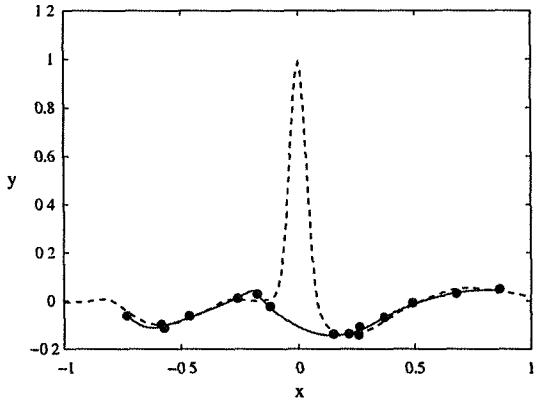


Figure 1: The approximation (solid line) of the true function is rather flat, even though the latter (dashed line) has a "sharp peak" around $x = 0$.

where C is a fuzzy constraint: $C(\omega) = 1$ means that the model $h(\cdot, \omega)$ satisfies the constraint completely, whereas $C(\omega) = 0$ indicates that the constraint is not satisfied at all. Any intermediate value $0 < C(\omega) < 1$ means that the constraint is violated to a certain extent, though not completely. As can be seen, an evaluation $R_{pen}(\omega)$ is a tradeoff between the accuracy of the hypothesis $h(\cdot, \omega)$, as expressed by the empirical risk $R_{emp}(\omega)$, and the extent to which $h(\cdot, \omega)$ is in accordance with the background knowledge, as expressed by $C(\omega)$.

The fuzzy constraint C will usually be a conjunction of several individual constraints C^i, $1 \leq i \leq m$, pertaining to different pieces of (local) knowledge. Using a t-norm $\otimes$ as a generalized logical conjunction, it can hence be modeled as follows:

$$C(\omega) =_{\text{def}} C^1(\omega) \otimes C^2(\omega) \otimes \ldots \otimes C^m(\omega).$$

The simplest type of individual constraint is a restriction on the absolute values of the function. Knowledge of this type can be expressed in terms of a *fuzzy rule* such as

$$\text{IF} \quad x \text{ is close to } 0 \quad \text{THEN} \quad f(x) \text{ is approximately } 1 \tag{5}$$

or, more formally, as

$$\forall x \in \mathfrak{D}_X : C_1(x) \rightsquigarrow C_2(f(x)),$$

where the fuzzy sets C_1 and C_2 model, respectively, the constraints "close to 0" and "approximately 1". The degree to which a hypothesis $h(\cdot, \omega)$ satisfies this constraint is given by

$$C(\omega) =_{\text{def}} \inf_{x \in \mathfrak{D}_X} C_1(x) \rightsquigarrow C_2(h(x, \omega)), \tag{6}$$

where $\rightsquigarrow$ is a generalized implication operator; the infimum operator in (6) generalizes the universal quantifier in classical logic.

Constraints of the above type can be extended quite easily to the case of more than one input variable, interactive input variables, first or higher order (partial) derivatives, or restrictions on the relative (rather than the absolute) values of a function. Apart from that, it is of course also possible to model "non-standard" constraints such as the peak of the function in Fig. 1. See [4] for a more detailed discussion.

Finally, let us note that (4) can thoroughly be combined with the standard regularization approach (3). One possibility is to supplement $R_{emp}(\omega)$ with both a term $\lambda\,\phi(\omega)$ that penalizes the complexity and a term $\lambda' C(\omega)$ that takes the background knowledge into consideration. A second possibility is to incorporate the former in the latter, i.e., to consider "low complexity" simply as an additional constraint.[3]

4 Operationalizing CRL

Learning in the context of CRL comes down to minimizing the penalized risk functional $R_{pen}(\cdot)$ in (4), i.e., to finding an optimal parameter in the parameter space Ω. Since the corresponding optimization problem is usually nonlinear, efficient optimization methods constitute a main prerequisite for putting the conceptual framework presented in the previous section into practice. Needless to say, specific setups of that framework will usually call for specific optimization tools. The specification of a setup includes:

- the hypothesis space $\mathcal{H}$ defined by the chosen model class and the valid model structures;
- the search method that identifies a (nearly) optimal model;
- the "language" for expressing constraints on the model;
- the fuzzy sets and generalized logical connectives used in formalizing the constraints.

The problem of optimization usually becomes much more practicable if the objective function is smooth, or at least continuous, and ideally even convex. In this regard, notice that the constraint C in (4) is a function composed of fuzzy sets, hypotheses $h \in \mathcal{H}$, generalized logical connectives and the inf-operator. Thus, it is possible to guarantee continuity of C, namely by restricting oneself to continuous fuzzy sets, hypotheses and logical connectives. To guarantee the convexity of C will usually not be possible, though.

There are basically two directions for approaching the optimization problem. The first one is to develop specialized methods for restricted setups, i.e. methods that exploit a certain structure of (4). It seems, however, that this idea can only be realized at the cost of considerable restrictions on the expressiveness of constraints.

In this paper, we pursue a different approach, namely the use of evolutionary algorithms as a general purpose optimization method. In such a case, the structure of (4) is less important. Still, in order to guarantee good performance of the

[3]This assumes a fuzzy set for the concept "low complexity" that scales the complexity measure to the unit interval.

evolutionary approach, the possibility of evaluating a particular solution ω by computing $R_{pen}(\omega)$ in an efficient way remains a crucial point. Note, for instance, that the formalization of most constraints involves an inf-operator, which means that evaluating such constraints might in principle become quite costly. Moreover, the dimension of the parameter ω is in direct correspondence with the dimension of the search space. Thus, a reasonable balance must be found between the expressiveness of $\mathcal{H}$ and the number of parameters to be optimized.

4.1 Takagi-Sugeno Fuzzy Models

As we deal with a combination of data-driven and knowledge-driven learning, a convenient representation of hypotheses $h \in \mathcal{H}$ are so-called Takagi-Sugeno fuzzy models (TS-FMs) [14]. These are rule-based fuzzy models with rules r of the following form:

$$\text{IF } x_1 \text{ is } A_1^r \text{ AND } \cdots \text{ AND } x_N \text{ is } A_N^r \text{ THEN } f(x) = y_r = y(x, c^r),$$

where A_n^r is an element of a class $\mathcal{F}_n$ of fuzzy sets defining a (fuzzy) partition of the domain of the n^{th} input variable, $1 \leq n \leq N$, and y_r denotes the model output. The latter is a function of the input vector x, parameterized by a vector of coefficients c^r.

Given a rule base with R rules, the overall model output y (coefficient vector c) of a TS-FM is a weighted sum of the rule outputs y_r (vectors c^r) coming from the different rules:

$$c = \frac{\sum_{r=1}^{R} c^r \mu^r(x)}{\sum_{r=1}^{R} \mu^r(x)}, \tag{7}$$

where c^r is the parameter vector of the r^{th} rule and $\mu^r(x)$ denotes the corresponding rule firing level (degree to which x satisfies the premise of the rule).

Now, consider the special case of a complete set of rules, which means that each combination of input fuzzy sets $A_n \in \mathcal{F}_n$ is represented by one rule premise (hence $R = \prod_n |\mathcal{F}_n|$) and zero$^{\text{th}}$ order TS-FMs (each output y_r is a constant). Moreover, if we guarantee that the $\mathcal{F}_n$, $1 \leq n \leq N$, form a partition of unity, the denominator in (7) becomes one and the output of a TS-FM simplifies to

$$y = \sum_{r=1}^{R} c^r \mu^r(x). \tag{8}$$

This construction obviously yields axis-orthogonal membership functions [4] (MFs) on a grid-like partition of the input space. Thus, using a complete rule base with MFs forming a partition of unity on each input, a singleton TS-FM becomes equivalent to a normalized lattice based (radial)-basis function network [7, 8]. An extension of this result, namely that TS-FMs are equivalent to local model networks, has been established in [5].

[4] Membership functions in the high-dimensional input space are (pointwise) products of one-dimensional membership functions defined on the corresponding axes.

Regarding the choice of MFs for fuzzy sets, *b(asis)-splines* offer some interesting characteristics (e.g. [6]). B-splines are recursively defined over a *knot-vector* λ, i.e. b-splines of order $k+1$ can be derived from those of order k:

$$B_j^{k+1}(x) = \frac{x - \lambda_j}{\lambda_{j+k-1} - \lambda_j} B_j^k(x) + \frac{\lambda_{j+k} - x}{\lambda_{j+k} - \lambda_{j+1}} B_{j+1}^k(x)$$

$$B_j^1(x) = \begin{cases} 1 & \text{if } x \in [\lambda_j, \lambda_{j+1}) \\ 0 & \text{otherwise} \end{cases},$$

with x = input value and $B_j^k(x)$ = activation value of the j^{th} b-spline defined over the knots λ_j to λ_{j+k} (see Fig. 2).

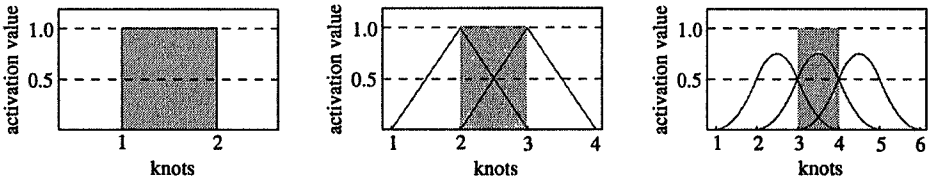


Figure 2: Univariate b-spline functions of order 1, 2, and 3 (from left to right). The shaded areas highlight the domain where the partition of unity is valid.

The concept of a knot-vector and the characteristics of b-splines meet our demands in several ways: (1) Each single knot vector defines a complete set of MFs in the form of a partition of unity for one linguistic variable. (2) The granularity of the grid-like partition of the input space can be adapted in a very flexible manner, simply by inserting or removing knots in the knot-vectors. (3) Changing the order of splines or the positions of knots allows for very flexible approximations. (4) Derivatives of a spline function are easy to calculate. The v^{th}-order derivative of a spline function of order k is itself a spline of order $k - v$ having the same knots.

The above discussion suggests the use of zero$^{\text{th}}$ order TS-FMs with b-splines as MFs as a convenient representation of hypotheses, and we shall subsequently employ this type of representation. Of course, due to the *curse of dimensionality*, the number of fuzzy sets in each dimension has to be limited. On the other hand, the approach allows for a very efficient model adaptation on the basis of least squares methods, and guarantees the interpretability of induced models. Finally, it is easy to calculate the derivatives of the model, which is important in the context of CRL. For example, local constraints such as e.g. "$f(\cdot)$ is rather flat around the origin" are naturally formalized in terms of fuzzy restrictions on the derivative.

4.2 Model Adaptation via Evolutionary Computation

In the remainder of this section, we present a framework for adapting TS-FMs using methods from *evolutionary computation* (EC). We assume some general familiarity with evolutionary methods and terminology; see [10] for an introduction and current trends.

A central aspect in EC concerns the distinction between the *genotype* and the *phenotype* representation of an individual. Roughly speaking, a phenotype corresponds to the solution of the (optimization) problem under consideration, which is in our context an optimal hypothesis $h \in \mathcal{H}$. The corresponding genotype is an *encoding* of the phenotype. In evolutionary algorithms, search is performed in the space of genotypes. In other words, the space of phenotypes (solutions) is not searched directly, but only in an implicit way. The real challenge for evolutionary search methods is hence to find an adequate genotype representation, such that this implicit search becomes as efficient as possible.

Two types of information represented by a genotype can roughly be distinguished, namely *structure information* and *variable information.* In genetic algorithms, for example, the genotypes are usually represented by a linear structure, namely a string $g = (g_1 \dots g_n)$ with binary variables $g_i \in \{0, 1\}$. For our purpose, *trees* appear to be a more convenient structure for genotypes. Especially, sub-trees inherently offer the possibility to classify functional entities of the target phenotype. By labeling nodes of the tree, it is possible to define the space of possible genotypes by using genotype-templates based on a grammar [11], where a grammar is a high-level notation used to describe the structure of data.

4.2.1 Tree-Based Genotypes and Genotype-Templates

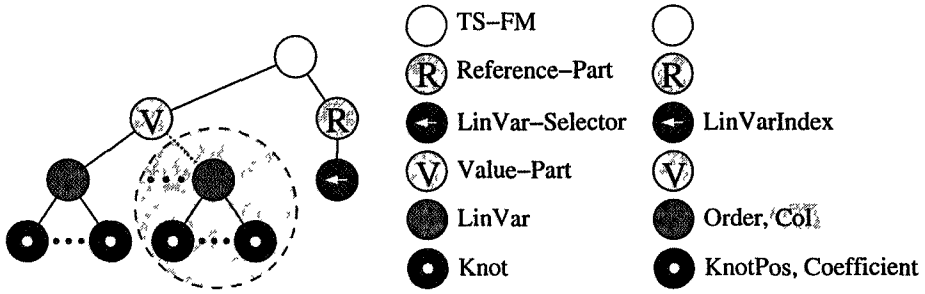
<genotype-template> ::= <node-list>,<conjunction-list>,<constraint-list>
<node-list> ::= **empty** | <node>,<node-list>
<node> ::= <node-label>,<min-succ-size>,<max-succ-size>,<var-list>
<conjunction-list> ::= **empty** | <conjunction>,<conjunction-list>
<conjunction> ::= <node>,<node>,<predetermined>
<var-list> ::= **empty** | <var>,<var-list>
<var> ::= <var-name>,<min-value>,<max-value>,<var-type>,<untouchable>
<constraint-list> ::= **empty** | <constraint>,<constraint-list>
<constraint> ::= <node>, <var-name>,<condition>,<scope>

<predetermined>,<untouchable> ::= **boolean**
<min-succ-size>,<max-succ-size> ::= **integer**
<min-value>,<max-value> ::= **double**
<node-label>,<var-name>,<var-type> ::= **string**
<condition> ::= **"=="** | **"!="** | **">"** | **"<"**
<scope> ::= **"sibling"** | **"level"** | **"all"**

Grammar 4.1: Grammar defining a genotype space on the basis of trees.

We make use of a context-free grammar in *Backus-Naur form* as a framework to formulate genotype-templates. For TS-FMs as hypotheses (phenotypes), the genotype search space $\mathcal{G}$ can be defined by instances (genotypes) of a genotype-template based on grammar 4.1. Different classes of nodes of the tree-based genotypes are

distinguished by different node labels. The non-terminal symbol <predetermined> indicates whether the corresponding node must be present to assure a mapping from genotypes to valid phenotypes. With the non-terminal symbol <untouchable> it is possible to detach certain variables from evolutionary operations such as mutation. Obviously, the structure information of a phenotype can be constrained by the formulation of the genotype-template. The restriction of variable information, regarding the variables embedded in the nodes of the tree-based genotype, is often inevitable. Simple restrictions can be formulated with the non-terminal symbol <constraint>, e.g. the mathematically claimed increasing alignment of b-spline knot-positions in each knot-vector.



(a) Structure. (b) Labels. (c) Variables.

Figure 3: A tree-based genotype representation showing (a) the structure, (b) the node-labels and (c) the embedded variables. The gray shaded areas in (a) and (c) illustrate the extension to a model capable to perform feature selection.

To exemplify the above concepts, suppose that phenotypes are TS-FMs, with b-splines of order two or three as MFs, for one-dimensional function approximation. Fig. 3 shows the corresponding genotype. By using grammar 4.1, this genotype can easily be described by genotype-template 4.1.

4.2.2 Evolutionary Operations on Tree-Based Genotypes

Regarding mutation operations, we have to distinguish between mutation of variable information and structure information. The former is implemented by traversing all nodes of the tree-based genotype representation. The variables at a node, if any, are modified by applying a (randomized) mutation operator. Usually, real-valued variables are changed by means of evolutionary strategy concepts, whereas integers are modified by increasing, decreasing or replacing the original value at random. In any case, the grammar-based genotype-template specifies the legal range for each variable. The mutation of structure information is implemented by adding, deleting or expanding a node chosen at random. Again, the grammar-based genotype-template provides all information necessary for validating the correctness of a structure information mutation.

Compared to string-based genotypes, the recombination operations simplify to sub-tree swapping. Furthermore, in the case of grammar-based genotype-templates,

```
node ("TS-FM", 2, 2, empty)
node ("Reference-Part", 1, 1, empty)
node ("LinVar-Selector", 0, 0, "LinVarIndex")
node ("Value-Part", 1, 1, empty)
node ("LinVar", 6, 11, "Order")
node ("Knot", 0, 0, ("KnotPos","Coefficient"))
var ("LinVarIndex", 0, 0, "integer", FALSE)
var ("Order", 2, 3, "integer", FALSE)
var ("KnotPos", 0.0, 1.0, "real", FALSE)
var ("Coefficient", -100.0, 100.0, "real", FALSE)
conjunction ("TS-FM", "Reference-Part", TRUE)
conjunction ("ReferencePart", "LinVar-Selector", TRUE)
conjunction ("TS-FM", "ValuePart", TRUE)
conjunction ("ValuePart", "LinVar", FALSE)
conjunction ("LinVar", "Knot", FALSE)
constraint ("Knot", "KnotPos", ">", "sibling")
```

Genotype-Template 4.1: Simple genotype-template for the above example.

verifying whether the sub-tree swapping results in valid genotypes can be done efficiently by checking the grammar.

4.3 Implementation

The task of the evolutionary algorithm (EA) is to minimize the risk functional (4). This is done by finding a (sub)optimal b-spline distribution (rule-antecedents) covering the input[5] and, simultaneously, optimal coefficients (rule-consequents). In pure data-driven approaches the coefficients can be derived solely by least squares (LS) methods. In our case, the coefficients are encoded in the genotype instead. In each generation the corresponding genotype variable value (the variable "Coefficient" in node "Knot") is calculated by by LS (i.e. Householder algorithm). If the calculated change of the coefficient is below a certain threshold the corresponding value in the genotype is replaced by this calculated value. If the change is equal or above the threshold, it is assumed that an important change in the individual has happened, due to the individual's attempt to satisfy (one of) the constraints. Thus, the adaptation of such coefficients are better driven by mutation rather than LS. The implementation concerning mutation and recombination follows the description of evolutionary operations on tree-based genotypes as outlined above. We

[5] Because we claim that the model input(s) are normalized to the interval [-1,1] the following was done: In the genotype to phenotype mapping, the outermost (for order two) respectively the two outermost (for order three) knots (see information variable definition in Genotype-Template 4.1) are rescaled to $[-2, -1.001]$ and $[1.001, 2]$. The internal knots are scaled to $[-1, 1]$. Thus, the input dimension is covered with b-splines fulfilling the partition of unity.

used an elite EA with tournament selection (tournament size 7) and a population size of 100 individuals. The crossover probability was chosen at random while the mutation probability of integer-valued variables was set to 0.001.

4.4 Related Methods

During recent years, a large number of methods for evolutionary learning or tuning of fuzzy (rule-based) systems has been devised; see e.g. [9, 2, 12, 16] for particular approaches and [3] for a comprehensive overview. The learning approach is purely data-driven in the sense that both the structure and the parameters (membership functions) of a fuzzy rule base are induced from a given set of data. In the tuning approach, the data is merely used in order to optimize the parameters of the fuzzy system while the structure is pre-specified. Needless to say, the transition between pure parameter optimization (tuning) and learning of complete rule bases is gradual with many methods falling in-between.

Specifying the structure of a rule-based system first and calibrating the system by means of a tuning-step afterwards seems to be an obvious alternative to combining knowledge-based and data-based modeling. It deserves mentioning, however, that the adaptation strategy as realized by CRL is quite different from such methods: In the tuning approach, the background knowledge is basically used for coming up with a good initial solution, specified in terms of the original rule base. This initial solution is then adapted to the data. This way, the original rule base actually influences the final model through implementing a kind of *search bias*, while the method itself is after all a more data-driven one. In fact, the deviation from the original model and, hence, the discrepancy between the final model and the background knowledge cannot be controlled.

As opposed to this, CRL tries to adapt the model to the data and the constraints more or less simultaneously, as the risk functional requires a real compromise between knowledge and data. That is, both pieces of information are guaranteed to have an influence on the result and the tradeoff between fitting the data and agreeing with the knowledge can be controlled in an explicit way. Apart from that, it should be noted that CRL is more flexible in the sense that it does not require the specification of a complete rule base. Moreover, it also allows for constraints that are not expressed in the form of rules.

5 Example

At present, a thorough empirical validation of our approach is difficult for at least two reasons: Firstly, we are currently not aware of alternative methods that are directly comparable. Secondly, real benchmark problems that provide both empirical data and background knowledge are not publicly available so far. Therefore, we restrict ourselves to illustrating CRL by means of a simple example that involves only one constraint. Moreover, to ease a graphical representation the function to be learned has only one input variable. Let us mention that even though CRL can be applied to higher-dimensional problems in a straightforward way, its complexity

does of course critically depend on the input dimension. In this respect, however, CRL does not differ from any other evolutionary method for learning fuzzy systems. In practice, all these methods do suffer from the curse of dimensionality.

An an example, consider again the function shown in Fig. 1, together with the random sample shown as black points. As background knowledge we seek to incorporate information about the peak around $x = 0$.

One possibility to model a "peak constraint" is to specify conditions on the function's derivative of first and maybe second order. For example, the derivative of a hypothesis h should be rather large for negative inputs x with a small (though not too small) distance from 0. Experimentally, however, we have found that in our example almost the same effect can be obtained with a more simple type of constraint, namely an absolute value constraint of the form (5).

More concretely, we have used the fuzzy rule "If x is near 0, then $f(x)$ is close to 1", formalized as

$$C(\omega) =_{\text{def}} \inf_{x\in[-1,1]} C_1(x) \rightsquigarrow C_2(h(x,\omega)). \tag{9}$$

The fuzzy sets C_1 and C_2 are specified through Gaussian membership functions, and as an implication we employed the Lukasiewicz operator $(\alpha,\beta) \mapsto \min\{1, 1-\alpha+\beta\}$. In this example, the parameter vector ω identifying a hypothesis comprises information about the spline approximation (knot-vector, coefficients of spline functions).

Computing $C(\omega)$, i.e. the infimum on the right-hand side of (9), amounts to finding

$$\min_{x\in[-1,1]} 1 - C_1(x) + C_2(h(x,\omega)) = \min_{x\in[-1,1]} 1 - C_1(x) + C_2\left(\sum_i \gamma_i B_i(x)\right),$$

where the B_i are b-splines (the coefficients γ_i are part of the parameter vector ω). Unfortunately, this minimization problem cannot be solved analytically. However, since the function to be minimized is smooth enough, standard numerical tools such as the Newton-Raphson method can be used. Taking the center of the fuzzy set C_1 as an initial value, this method converges extremely fast.

In our evolutionary algorithm, we generated 50 populations of size 100 (taking around 10 seconds on an AMD XP2000+ CPU). The number of b-splines forming a partition was lower and upper bounded by 3 and 7, respectively. The result of the optimization is shown in Fig. 4 for the regularization parameter $\lambda = 1$ and in Fig. 5 for $\lambda = 2$. The same figures also show the underlying b-spline partitions. As can be seen, the original function, including the peak, can now be approximated quite accurately. To compare, the approximation presented in Fig. 1 (Section 3) corresponds to the result we obtained when leaving the constraint out of account (i.e. when setting $\lambda = 0$). Moreover, the influence of the regularization parameter can be seen nicely in our example: The larger λ is, the higher the peak becomes.

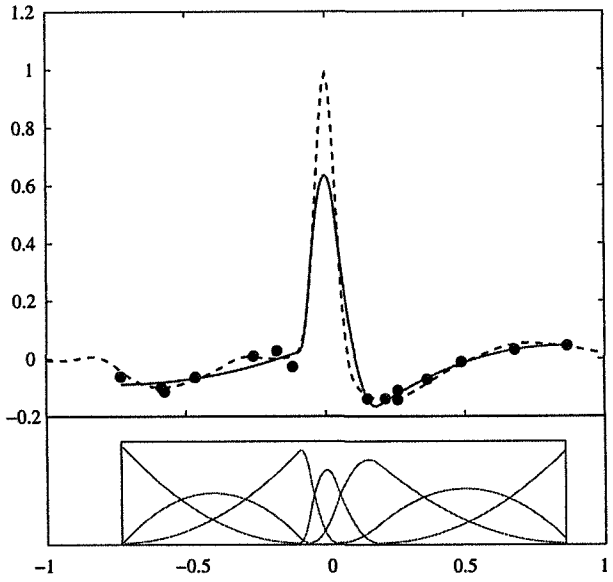


Figure 4: Optimal approximation (solid line) found by the evolutionary algorithm with $\lambda = 1$ (top), together with the underlying spline partition (bottom).

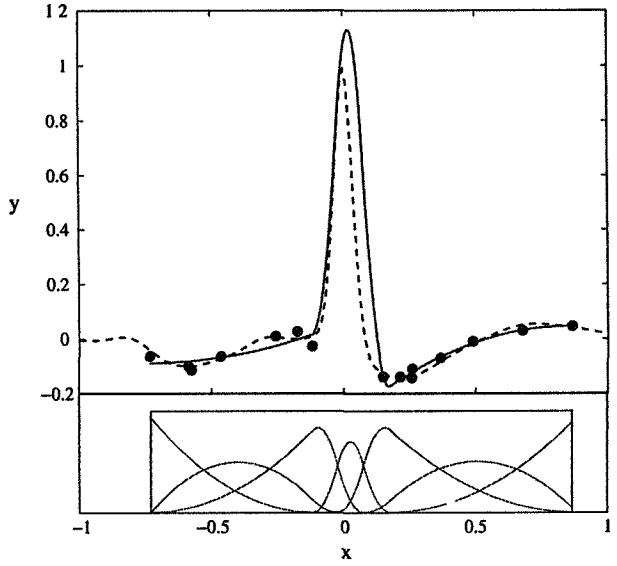


Figure 5: Optimal approximation (solid line) found by the evolutionary algorithm with $\lambda = 2$ (top), together with the underlying spline partition (bottom).

6 Concluding Remarks

The idea of constraint-regularized learning (CRL) is to embed fuzzy modeling techniques in regularized learning. This approach provides a simple yet elegant means for considering background knowledge, expressed in terms of flexible constraints, in learning from data. The paper has proposed an operationalization of CRL, using Takagi-Sugeno fuzzy rule bases as a flexible model class and methods from evolutionary computation for finding a model which is (nearly) optimal in the sense of minimizing the modified risk functional. Our evolutionary approach makes use of an efficient encoding of models in terms of tree-based genotypes. These are specified by means of genotype-templates defined by grammars.

CRL naturally supports a kind of "interactive" learning and optimization process. Namely, the optimal approximation obtained for a certain set of constraints might give a human expert cause to modify these constraints or to adapt the regularization parameter. That is, the inspection of the supposedly optimal function might suggest adding further constraints not made explicit so far or strengthening some of the original constraints. For instance, in our example the expert might vary the parameter λ until the peak satisfies his conception. Furthermore, it would be useful to inform the expert about the consistency of the constraints that he has specified,[6] or to point at particular constraints that are critical in the sense of being in conflict with parts of the observed data. An important aspect of future work is to extend our current implementation in these directions. Eventually, a tool should be obtained that supports the above type of interactive learning in a convenient way.

Acknowledgments
We are grateful to two anonymous reviewers whose suggestions helped to improve the paper. Two of the authors (A. Grauel and I. Renners) thank the Ministry of Science and Research, North Rhine-Westphalia, for financial support.

References

[1] Brown, M.; Harris, C.J.: *Neurofuzzy Adaptive Modelling and Control.* Prentice Hall, New York, (1994).

[2] Cordon, O.; Herrera, F.: *Hybridizing genetic algorithms with sharing scheme and evolution strategies for designing approximate fuzzy rule-based systems.* Fuzzy Sets and Systems, vol.118, no.2, pp.235–255, (2001).

[3] Cordon, O.; Gomide, F.; Herrera, F.; Hoffmann, F.; Magdalena, L.: *Ten years of genetic fuzzy systems: Current framework and new trends.* Fuzzy Sets and Systems, vol.141, no.1, pp.5–31, (2004).

[4] Hüllermeier, E.: *Regularized Learning with Flexible Constraints.* Proc. IDA–03, 5th International Symposium on Intelligent Data Analysis (LLNC 2810), pp. 13–24, Berlin, (2003).

[6] More often than not, human expert knowledge is not completely consistent.

[5] Hunt, K.J.; Haas, R.; Murray-Smith, R.: *Extending the functional equivalence of radial basis function networks and fuzzy inference systems.* IEEE Transactions on Neural Networks, vol.7(3), pp.776–781, (1996).

[6] Ichihashi, H.; Shirai, T.; Nagasaka, K.; Miyoshi, T.: *Neuro-fuzzy ID3: A method of inducing fuzzy decision trees with linear programming for maximizing entropy and an algebraic method for incremental learning.* Fuzzy Sets and Systems, vol.81, no.1, pp.157–167, (1996).

[7] Jang, S.R.; Sun, C-T.: *Functional equivalence between radial basis function networks and fuzzy inference systems.* IEEE Transactions on Neural Networks, vol.4(1), pp.156–159, (1993).

[8] Kecman, V.; Pfeiffer, B-M.: *Exploiting the structural equivalence of learning fuzzy systems and radial basis function networks.* European Congress on Intelligent Techniques and Soft Computing (EUFIT), pp.58–66, Aachen, Germany, (1994).

[9] Krone, A.; Kiendl, H.: *Evolutionary concept for generating relevant fuzzy rules from data.* Int. Journal of of Knowledge-based Intelligent Engineering Systems,vol.1, no.4, pp.207–213, (1997).

[10] Paun, G.; Rozenberg, G.; Salomaa, A. (Editors): *Current Trends in Theoretical Computer Science, Entering the 21th Century.* Publisher: World Scientific, ISBN 981-02-4473-8, chapter "Natural Computing", pp.546–690, (2001).

[11] Renners, I.; Grauel, A.: *Variable length coding of genotypes using n-ary trees.* Proc. of the 9^{th} Int. Conference on Soft Computing – Mendel, (2003).

[12] L. Sanchez, I. Couso, and JA. Corrales. Combining GP operators with SA search to evolve fuzzy rule based-classifiers. *Information Sciences*, 136(1–4):175–191, 2001.

[13] Tanaka, K.; Sano, M.; Watanabe, H.: *Modeling and Control of Carbon Monoxide Concentration Using a Neuro-Fuzzy Technique.* IEEE Trans. Fuzzy Systems, vol.3, pp.271–279, (1995).

[14] Tanaka, K.; Sugeno, M.: *Introduction to Fuzzy Modeling.* Fuzzy Systems Modeling and Control (Nguyen, H.T. and Sugeno, M., Eds.), Kluwer Academic Publ., Boston, MA, pp.63-89, (1998).

[15] V.N. Vapnik: *The Nature of Statistical Learning Theory.* Springer-Verlag, New York, second edition, (2000).

[16] Xiong, N.; Litz, L.: *Reduction of fuzzy control rules by means of premise learning – method and case study.* Fuzzy Sets and Systems, vol.132, no.2, pp.217–231, (2002).

[17] Zadeh, L.A.: *The Concept of a Linguistic Variable and its Application to Approximate Reasoning.* Information Sciences - Application to Approximate Reasoning I - III, (1975).