Doktorandský den ’02

Ústav informatiky
Akademie věd České republiky

Praha, 17. říjen 2002

Obsah

Reflexion in C++

doktorand:
PAVEL KRUŠINA

krusina@rychnov.cz

školitel:
ROMAN NERUDA

roman@cs.cas.cz

obor studia:
I-1

Abstrakt

This article deals with the topic of reflexion in the C++ programming language and the ways of its implementation. We assume that the reader has the basic knowledge of the C++ and its object terminology.

1. What is *reflexion*

Reflexion is the ability of a program to see itself. This means that from a class point of view, the class is able to get information about its elements and parents and their types and exact possitions in itself and about the type of inheritance. This information should be easily available to the class and its methods, possibly as the return values of well-known functions. This makes possible to create generic functions working on a broad set of data, getting the exact information about the class from itself.

We work on a design of a theoretical computational model called ASP (for ASynchronous Parallel). This model is an extension of Valliants's BSP model of parallel computation. It focus in on decribing the asynchronous work and communication of the computational system. We aim to create a realistic theoretical model to be able – to some extent – predict the time complexity of algorithms on real-world computers and clusters.

This is a theoretical foundation of our work on a unified, modular, agent-oriented framework [**?**] for computations which enables to test our complexity estimates and validate them on real problems. The following article deals with one building block of this system: the reflexion library which makes data inspecting and transmissions possible with minimalizing the user assistence.

2. Existing implementations

2.1. RTTI

Specification: RTTI means run-time type information. It is composed of standard `dynamic_cast` and `typeid` operators.

Pros: it is wide-spread standard.

Cons: its function is limited to class inheritance detection, the content of structures is invisible for this method and so the RTTI is not sufficient for our needs.

2.2. Parsing source code

Specification: Though the wanted information is available to the C++ compiler, it is not usually exported in any form for the program. This approach uses special program which mimics the first phases of C++ compiler's work – lexical and syntactical analysis – and exports the result for the program. OpenC++ [**?**] is an example of such a program – it parses the C++ sources and stores the information in the form of C++ objects linked together with the inspected program to make it available.

Pros: the complete information is available.

Cons: the parser complexity is high and the parser program has to be kept up-to-date as the C++ language syntax slowly changes.

2.3. User-supplied data

Specification: the user is asked to write down all needed information manually. For example MPI uses this method for MPI derived types – here it is the user's responsibility to provide the system with accurate type information for each new type: the exact type and possition.

Pros: it is the easiest for the programmer – all the responsibility is on the user.

Cons: this method is prone to user errors, though this works for small or not-too-much complicated data types, it is not useful for large systems.

2.4. Avoiding reflexion at all

Specification: the metadata is not collected at all. Instead, virtual functions are used where the polymorphism of different data structures is needed.

Pros: it solves the problem, if it is possible to use this method.

Cons: in many cases, this method is not possible at all, or it is only a hidden form of the previous case. For example if we want to avoid the need of collecting metadata for generic serialization routine, we may design our system to call for each given data structure its respective virtual serialization routine. But the problem is how to write the serialization for composite objects – obviously by calling serialization routines for all the components. Here we encounter similar problems as with the previous approach, because the user has to manually write down information about the components.

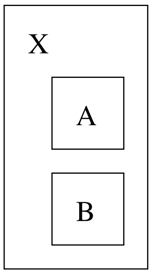
3. RTI method

Not beeing satisfied with existing approaches, we designed our own one to solve the problem of reflexion implementation. We call it RTI (for Rox Type Info). This work is part of the bang3 project [**?**] and you can see the source code of our implementation at the project site.

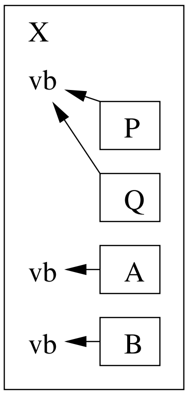
3.1. Class elements

First we detect which classes the given class consists of (Figure 1).

To solve this problem, we exploit the fact that the class elements constructors are called during each object creation. We only need to call from each class constructor our function saving the *this* pointer for further processing. Of course we need not to do so each time but only once for each class type – usually during initialization phase of program start-up.



Obrázek 1: The structure X consists of elements A and B.



Obrázek 2: The structure X inherits from P and Q and consists of elements A and B.

3.2. Class parents

Second, we want to know, what are the parents of a given class (Figure 2). Notice, that the memory layout for the class elements and the parents is the same – with the parents going first.

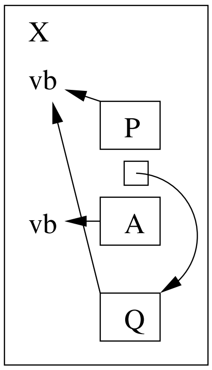
```
class P : public virtual VirtBase { ... };
class Q : public virtual VirtBase { ... };
class A : public virtual VirtBase { ... };
class B : public virtual VirtBase { ... };

class X : public P, public Q {

        A       a;
        B       b;
};
```

To distinguish between these two cases we used the fact, that the virtually inherited base classes are shared among the parents, but not among the elements. So we put one common virtual base class into each investigated class type. Now we can test whether the virtual base of each executed constructor class has the same address as the virtual base of the whole class, or not. In the former case the subclass is a parent, in the later it is the element.

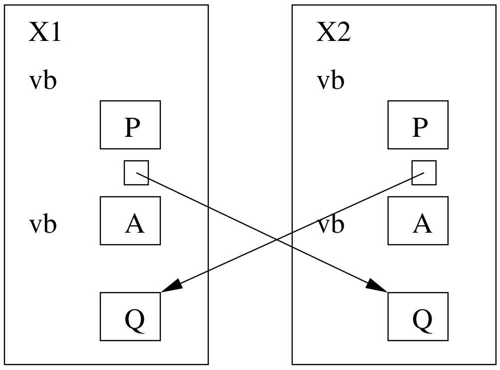
3.3. Virtual inheritance

Finally we want to detect, if the parent class P is inherited into class X virtually or not (Figure 3).

```
class P : public virtual VirtBase { ... };
class Q : public virtual VirtBase { ... };
class A : public virtual VirtBase { ... };
```



Obrázek 3: The structure X inherits from staticly P and from Q virtually.



Obrázek 4: Two X structures after content swap.

```
class X : public P, virtual public Q {

        A       a;
};
```

To detect the virtuallity of inherited base class we create two instances of the investigated class and swap their contents using the low-level `memcpy` function. From the changes of `this` pointers between construction and destruction of such a modified pair we can distinguish the virtual base from non-virtual. The virtually inherited base classes are destructed at the changed locations, because they are accessed using single pointer stored inside the inherited class[1]. The statically inherited base classes are destructed at the original locations (Figure 4).

4. Example

Let us look at the working library now. In this example we create the diamond scheme of classes, each with some member variables. We also flag a variable with a label or a hidden flag. Then we use generic function to convert the structure to a printable format.

```
#include ″../../../base/banglib.h″
#include ″../../../dict/all.h″

class   P : public CStruct
{       STRUCTIMPL( P )
```

---

[1] According to the Cfront-style virtual inheritance implementation, which is common in todays compilers.

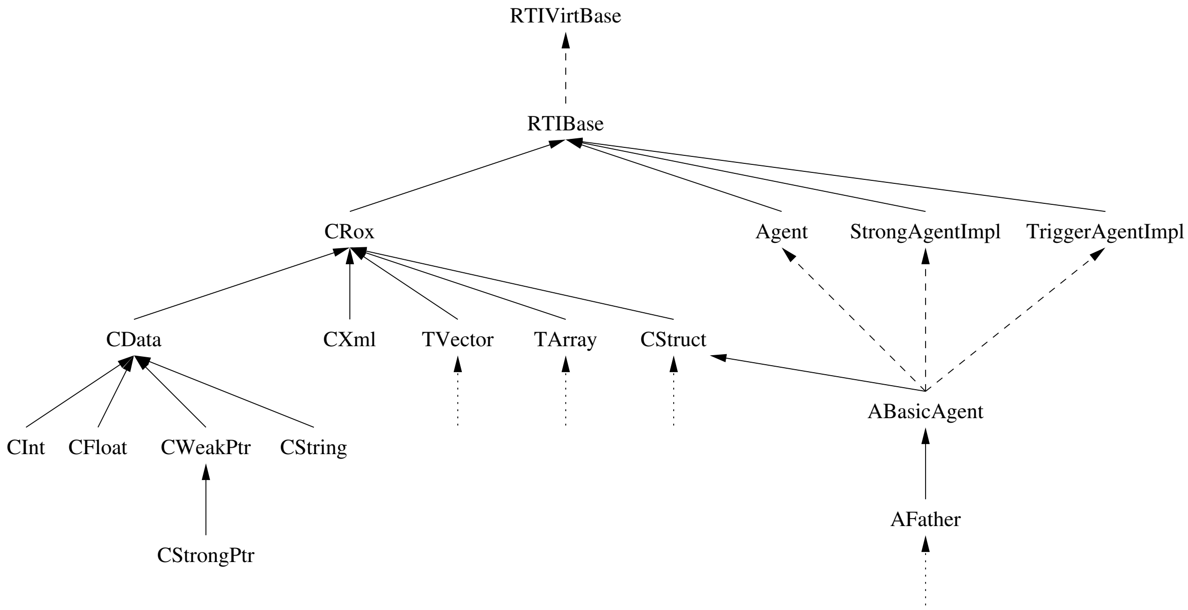
```
        CString p;
};
REGROX( P )

class   Q : public CStruct
{       STRUCTIMPL( Q )

        CInt    q;
};
REGROX( Q )

class   X : public P, public Q
{       STRUCTIMPL( X )
        DOMINANCE( P )
                        CString m;
        LABEL(alpha)    CFloat  a;
        HIDDEN          CString b;
                        CString c;
};
REGROX( X )

int main()
{       AutoExec();

        X mm;
        mm.m = ”prealpha”;
        mm.a =  3.14;
        mm.b = ”BETA”;
        mm.c = ”CORRINO”;
        mm.p = ”pH”;
        mm.q =  12;

        const RStr& mms = *mm.ToCRox();
        puts( mms.ToChar() );

        return 0;
}
```

```
Program output:

<X>
    <CString value=”pH”/>
    <CInt value=”12”/>
    <CString value=”prealpha”/>
    <alpha>
        <CFloat value=”3.14000000000000012e+00”/>
    </alpha>
    <CString value=”CORRINO”/>
</X>
```

In a similar way we can serialize this new structure of ours into a block of binary data and recreate it again[2].

---

[2] The whole example is available on bang3 project CVS in directory bang3/test/auto/210_xize.



Obrázek 5: Appendix: Bang3 data types hierarchy.

5. Conclusion

We successfully designed and implemented the reflexion method for C++ data structures. This allows for easier programming of sophisticated data types which are able to print, serialize, deserialize and parse themselves in an automatic manner.

We can use these data types to communicate throughout the network without additional effort from the agent programmer. We also can use such a communicating system as a demonstration and a test case of our ASP model.

Genetic and Eugenic Learning of RBF Networks

doktorand:
PETRA KUDOVÁ
Ústav informatiky AV ČR
Pod vodárenskou věží 2,
18 207 Praha 8
petra@cs.cas.cz

školitel:
ROMAN NERUDA
Ústav informatiky AV ČR
Pod vodárenskou věží 2,
18 207 Praha 8
roman@cs.cas.cz

obor studia:
Teoretická informatika

Abstrakt

Learning methods for RBF networks are briefly reviewed. The focus is on evolutionary methods. As an alternative to the genetic algorithms we introduce the Eugenic learning, evolving separately a pool of hidden units and complete networks. Described algorithms are demonstrated on some experiments.

1. RBF network

An *RBF network* is a feed-forward neural network with one hidden layer of RBF units and a linear output layer. By an *RBF unit* we mean a neuron with multiple real inputs $\vec{x} = (x_1, \dots, x_n)$ and one output y computed as:

$$y = \varphi(\xi); \quad \xi = \frac{\| \vec{x} - \vec{c} \|_C}{b} \tag{1}$$

where $\varphi : \mathbb{R} \to \mathbb{R}$ is a suitable activation function, let us consider Gaussian $\varphi(z) = e^{-z^2}$. The center $\vec{c} \in \mathbb{R}^n$, the width $b \in \mathbb{R}$ and an $n \times n$ real matrix C are a unit's parameters, $|| \cdot ||_C$ denotes a weighted norm defined as $\| \vec{x} \|^2_C = (\mathbf{C}\vec{x})^T(\mathbf{C}\vec{x}) = \vec{x}^T\mathbf{C}^T\mathbf{C}\vec{x}$.

Thus, the network represents the following real function $\vec{f} : \mathbb{R}^n \to \mathbb{R}^m$:

$$f_s(\vec{x}) = \sum_{j=1}^{h} w_{js} e^{-\left(\frac{\| \vec{x} - \vec{c}_j \|_{C_j}}{b_j}\right)^2}, \qquad s = 1, \dots, m, \tag{2}$$

where $w_{js} \in \mathbb{R}$ are weights of s-th output unit and f_s is the s-th network output.

The goal of an RBF network learning is to find suitable values of RBF units' parameters and the output layer's weights, so that the RBF network function approximates a function given by a set of examples of inputs and desired outputs $T = \{\vec{x}(t), \vec{d}(t); t = 1, \ldots, k\}$, called a *training set*. The quality of the learned RBF network is measured by the *error function*:

$$E = \frac{1}{2} \sum_{t=1}^{k} \sum_{j=1}^{m} e_j^2(t), \qquad e_j(t) = d_j(t) - f_j(t). \tag{3}$$

2. RBF network learning

There is quite a wide range of methods used for RBF networks learning, three main approaches were described in [**?**].

From one point of view an RBF network is a special type of a feed-forward network, like a multilayer perceptron (MLP). Therefore we can use a modification of the Back Propagation algorithm, designed for MLPs. It is an iterative algorithm based on a straightforward gradient minimization of the error function E from (3). Starting with random initialization, we shift the vector of all network parameters in the direction given by the gradient of the error function, and iterate until a local minimum or the desired value of the error function is reached. We call it the *Gradient learning*, since there is only one hidden layer and therefore no "back propagation" during the evaluation of the gradient.

The Gradient learning unifies all parameters by treating them in the same way, even though the meaning of the RBF network parameters is well defined. Another method, *Tree-step method* takes advantage of this knowledge, divides the parameters into three categories and learns each category separately using a customized method.

First we need to set the centers of RBF units. They should reflect the density of training data points and therefore some vector quantization algorithm is usually used. Then we find suitable values for additional units' parameters (widths, norm matrices), which determine the size and the shape of the area controlled by a unit. Some heuristics are used to cover the whole input space with these areas and at the same time to keep their local character. Remaining weights of a linear combination are determined by a linear least squares method.

A completely different approach to RBF learning is represented by evolutionary methods, such as *genetic algorithms*. It is a stochastic optimization method, working with a population of individuals and operators of crossover, mutation and selection constructing new generations. Each individual represents one feasible setting of network parameters, the better the represented network, the higher the individual's *fitness* (a probability of being selected into a new generation).

In [**?**] the canonical version of the genetic algorithm is introduced. It reduces the searched parameter space using the fact that RBF networks with the same but permuted RBF units represent the same function and thus are equivalent.

Another evolutionary method, based on Eugenic evolution [**?**], is described in the next section.

All mentioned methods are candidates for RBF network learning. The real choice depends on actual computational resources or a character of a given data set. The Three-step method is the best choice from the computational time point of view, the Gradient method usually finds the most accurate solution. Evolutionary methods are quite time consuming but can be used also for determination of a number of hidden units or for learning of networks with different types of units in the hidden layer.

3. RBF network learning based on Eugenic evolution

Since an RBF network represents a linear combination of RBF units, we can understand it as a *team* of cooperating units. Thanks to the local character of RBF units, we can assume that if some unit works well on its local area it would probably be a good member of a team. Thus, if we had a good supply of those units, we would probably be able to select teams forming good networks.

Now we introduce a learning technique based on the eugenic evolution described in [**?**].

The evolution takes place on two levels. On the first level we evolve a pool of RBF units that serves as a store of building blocks for RBF networks. Good units are those that are able to cooperate with other units well enough to solve a given task.

A network, understood as a team of units, is represented by a *blueprint* specifying which units form its hidden layer. A population of blueprints is evolved on the second level.

RBF units evolution

First we describe one step of the first level evolution, i.e. evolution of units:

- According to a chosen blueprint we create a network that is used to solve a given task (i.e. is applied on a given training set) in a trial, and is rewarded by a fitness based on the error function. All units of the current team increase their trial count and the network fitness is added to their fitness count.
- After enough trials have been performed, each neuron average fitness is evaluated dividing its fitness count by its trial count.
- New units are constructed using the operators of selection, crossover and mutation and they replace the lower ranking part of the population.

Starting with randomly initialized neurons and blueprints, we iterate until a suitable solution is found.

To form a network we could choose a team in some random way, but we would prefer to build teams of units that are likely to work well together. Therefore the evolution on the second level is done by means of a eugenic algorithm.

Here, an individual represents one blueprint and it is coded by a bitmap. Each bit represents one gene and corresponds to one RBF unit from a pool. It is set to one, if this unit is included in the hidden layer of a particular network.

The introduction to the general Eugenic algorithm follows.

Eugenic algorithm

Unlike a standard genetic algorithm, where crossover, mutation and selection are used to form a new individual, in Eugenic algorithm a new individual is constructed one gene after another according to the statistics over the entire population.

Starting with a random initialization, we iterate until a suitable solution is found.

In each generation only one new individual $\mathcal{N}$ is generated:

1. Start with the set of all genes $U = \{x_1, x_2, \dots, x_n\}$
2. Select the most significant gene x_{sig}. This should be the gene that is most strongly correlated with the fitness of the individuals. Assume that the larger the $|f(x_{g,0}) - f(x_{g,1})|$, the stronger the influence of the gene x_g is. ($f(x_{g,0})$ is the average fitness of those individuals having 0 at gene x_g, similarly for $f(x_{g,1})$).
3. Set the value of the gene x_{sig} according to the probability $f(x_{sig,a})/(f(x_{sig,1}) + f(x_{sig,0}))$, where a is either 1 or 0. Higher probability have those values that are more likely (according to a current population) to form good individuals.
4. The gene x_{sig} is removed from U.
5. A rough estimate of *epistasis* is evaluated for the population.

   It is a measure of how strongly the fitness $f(x_{g,a})$ depends on the values of the other genes. If for all genes g the difference between $f(x_{g,0})$ and $f(x_{g,1})$ is low, i.e. both choices seems to be equally good, we speak about high epistasis. High epistasis indicates that there is no really significant gene and therefore the values for remaining genes should not be chosen independently.

6. If epistasis is high, the population is restricted to those individuals having the same value of gene x_{sig} as the individual $\mathcal{N}$.
7. Use the restricted population and continue with step 2 until U is empty.
8. Compute the fitness of the individual $\mathcal{N}$ (including error function evaluation of the corresponding network) and replace the individual with worst fitness.

The described algorithm doesn't control the number of units in the hidden layer. To avoid large or empty networks, we explicitly define the minimal and maximal number of units. Creation of a new individual is stopped after the maximal number of units is reached (i.e. the same number of bits is set to 1). If the created individual has less than minimal number of units, randomly chosen bits are set to 1. Then the algorithm determines not only the values of networks parameters, but also the number of hidden units.

4. Experiments

In this section we present some results of our experiments.

All experiments were run on a Linux cluster, each computation on an individual node with a Celeron 1.1GHz processor and 0.5 GB of memory.

The well-known data set *Iris plants* was used. It contains 3 classes of 50 instances each, where each class refers to a type of an iris plant. We used a network with three output neurons, one neuron for each class. The class number is then coded by three binary values, value 1 on the position corresponding to the number of class and zeros on the others. So each training sample consists of 4 input values describing some features of a plant and 3 output values coding its class.

We tried to learn an RBF network with 3, 6, 9 and 12 RBF units, using genetic algorithms and eugenic learning. Each computation was run 5 times, an average computation was picked up. The average time of 100 iterations was 72.0 s for genetic learning, 8.6 s for eugenic learning.

Figures 6 and 7 shows the typical fall of the error function comparing the canonical and the classical version of genetic algorithms.

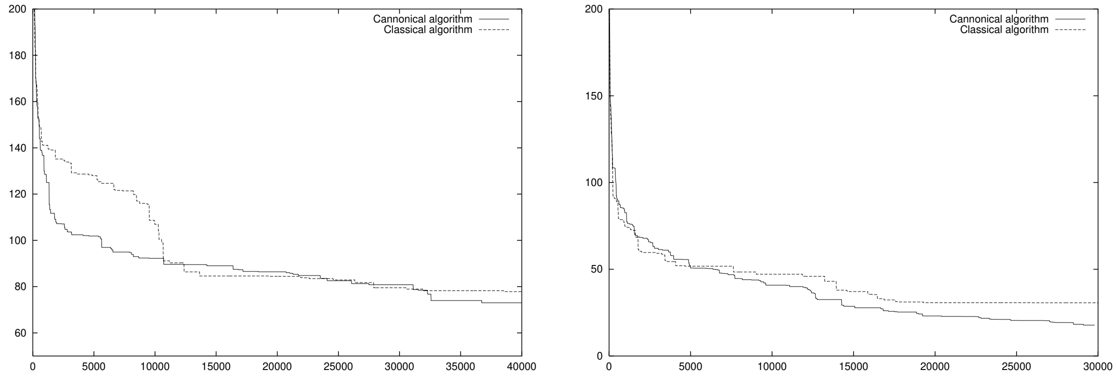
On figure 8a) you can see the typical fall of the error function using eugenic learning for networks with 3, 6, 9 and 12 units.

Eugenic learning was run also with a varying number of RBF units, the minimal number of units was 3, maximal 12. Figure 8b) shows the fall of the error function.

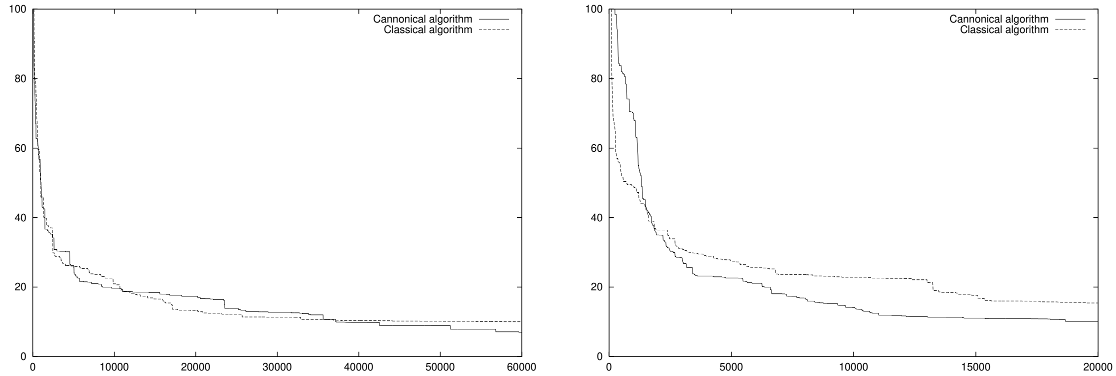
Genetic and Eugenic learning of an RBF network with 12 hidden units are compared in the table 1, that shows the number of iterations and time needed to achieve given ε.

Note that, while Genetic learning creates a whole population of new individuals and evaluates the value of the error function for all of them each iteration, the Eugenic learning evaluates the error function only for the new individual and those individuals that mutated during last iteration. Therefore there is no use of comparing the numbers of iterations and we have to measure time.

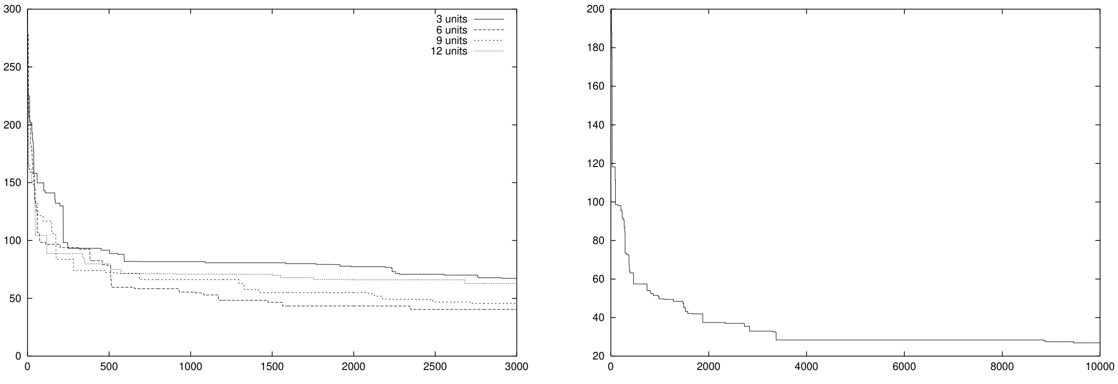
On the *Iris plant* problem we demonstrated, that Eugenic algorithm can achieve results comparable to genetic algorithms in shorter time.



Obrázek 6: The fall of error function in canonical genetic learning and classical genetic learning, for the RBF network with a) 3 units b) 6 units.



Obrázek 7: The fall of error function in canonical genetic learning and classical genetic learning, for the RBF network with a) 9 units b) 12 units.



Obrázek 8: The fall of the error function in eugenic learning for the RBF network with a) 3, 6, 9 and 12 units. b) the varying number of units, from 3 to 12.

ε	**EuGA**		CanGA		GA	
100	**95**	**8 s**	30	22 s	20	14 s
80	**291**	**25** s	240	3 min 18 s	100	1 min 22 s
60	**463**	**40 s**	1320	18 min 12 s	720	9 min 55 s
40	**1879**	**2 min 42 s**	3420	47 min 11 s	6810	1 hours 33 min 11 s
20	**151717**	**3 hours 37 min 27 s**	22390	5 hours 8 min 58 s	28590	6 hours 35 min 54 s

Tabulka 1: Average number of iterations and time needed for the fall of the error function under the given ε.

5. Conclusion

We reviewed common learning methods for RBF networks, including evolutionary methods. The Eugenic learning based on two-level evolution was introduced. It takes advantage of local character of hidden units and more sophisticated way for creating new individuals is used. It seems to be a promising alternative to genetic algorithms.

Our future focus should be moved to hybrid methods combining known algorithms.

In [**?**] it was already shown that the Three step method combined with Gradient learning could bring better results than these methods alone. Also genetic algorithms could be easily combined with other methods, for instance they are very good at setting centers in the Three step method.

And finally, the Eugenic learning described in this paper takes advantage of local character of units while evolving a pool of them. But we have to learn also the weights of the output layer, which are highly dependent on other weights and units in a current network. Therefore we want to let the Eugenic learning learn only the hidden layer and set the output weights using some linear least squares method.

Ontologie v Bangu

doktorand:
ZUZANA PETROVÁ
Ústav informatiky AV ČR,
Pod vodárenskou věží 2, 182 07, Praha 8
petrova@cs.cas.cz

školitel:
ROMAN NERUDA
Ústav informatiky AV ČR,
Pod vodárenskou věží 2, 182 07, Praha 8
roman@cs.cas.cz

obor studia:
Teoretická informatika

Abstrakt

Systém Bang je knihovna, obsahující datové struktury a algoritmy spadající do oblasti umělé inteligence, zejména neuronové sítě a algoritmy pro jejich učení a genetické algoritmy. Navíc je i platformou jak pro vyvíjení nových datových struktur (např. nových typů neuronových sítí) a jejich učících algoritmů, tak pro učení stávajících struktur a uschovávání naučených konfigurací.

Předpokládejme, že Bang bude mít v budoucnu poměrně velký počet uživatelů, kteří si budou do centrální databáze ukládat své vlastní struktury i konfigurace, naučené na určitý problém. Budou ovšem také chtít nalézt datovou strukturu, která je schopná se naučit řešit zadaný problém, nebo ještě lépe její už naučenou konfiguraci.

Každá datová struktura bude tedy obsahovat záznam o tom, čemu je schopna se naučit a každá konfigurace podobný záznam o tom, co všechno umí, popřípadě jak moc to umí. Předpokládá se, že jedna struktura či konfigurace může mít více záznamů a hierarchické uspořádání struktur (například RBF síť umí to, co každá neuronová síť a ještě něco navíc).

Cílem článku je navrhnout jazyk, vhodný pro vytváření takovýchto záznamů. Dále je třeba, abychom poznali, že záznamy vytvářené různými uživateli pro stejnou či podobnou konfiguraci si jsou podobné. Tento problém alespoň částečně řeší ontologie a dalším cílem mého článku je proto diskuse o použití ontologií v Bangu.

1. Úvod

Hybridní modely, zejména kombinace metod umělé inteligence, jako jsou například neuronové sítě, genetické algoritmy či fuzzy logické kontrolery, se zdají být slibnou oblastí výzkumu a jsou též směrem v současné době poměrně rozšířeným [1]. My jsme s nimi experimentovali v předchozí implementaci našeho systému (popsán v práci [3]) s povzbuzujícími výsledky soudě alespoň podle srovnávacích testů [3].

Bang sestává z populace agentů žijících v prostředí, které poskytuje nezbytné služby, jako vytváření nových agentů, komunikaci pomocí zpráv a distribuované spoustění v počítačové síti a mobilitu. Ostatní důležité služby jako databáze algoritmů, seznam bežících procesů, zajištění persistence dat, load balancing, textové a grafické rozhraní pro uživatele už jsou zajišťovány na "uživatelské úrovni", tedy agenty.

Pojem inteligetního softwarového agenta vznikl z potřeby mít kusy softwaru, které by byly do jisté míry autonomní, samostatně se rozhodující a popřípadě schopné si sehnat informace, které ke svému rozhodování potřebují. Příklady dnes používaných agentů jsou roboti vyhledávající informace na internetu, či elektronické sekretářky. Nicméně ač je softwarový agent velice vděčné téma pro pro vědecké články neexistuje pro něj, na rozdíl od například objektu (ve smyslu objektově orientovaného programování) žádná jednotná definice. Různé definice agenta jsou prodiskutovány například ve výborném úvodním článku od pana Franklina [2].

Obecně, softwarový agent je počítačový program, který je autonomní, vnímá stav a změny ve svém okolí a odpovídá na ně, sleduje tím vlastní cíle. Takovýto agent může být adaptivní či inteligentní, ve smyslu být schopen sbírat informace, které potřebuje, nějakým sofistikovaným způsobem. Naši agenti jsou navíc mobilní (mohou se přemístit na jiný počítač) a trvalí (persistentní), tedy schopní přežít i poté, co přestane existovat jejich prostředí. Neuvažujeme o jiných "typických" vlastnostech agentů jako simulace lidských emocí, nálady atp.

Prostředí v Bangu dává agentům možnost komunikovat pomocí zpráv. Zprávy mohou být synchronní či asynchronní a za podstatné považuji též tzv. location transparency, tedy skrytí rozdílů mezi tím, jestli agenti, kteří spolu komunikují, jsou na stejném nebo na různých počítačích.

Celý Bang je psán v C++. Agenti jsou třídy odvozené od základní třídy Agent, která umí reagovat na několik základních zpráv (ping aneb žiješ?, přemísti se na jiný počítač, ukonči se). Programátor nového agenta pak jen připisuje triggery. Trigger je procedura reagující na jednu konkrétní zprávu. Prostředí přijímá zprávy a spouští odpovídající triggery agentů.

2. Zlaté a bílé stránky

Bang obsahuje velkou spoustu agentů, kteří jsou dobře popsáni v on-line manuálu [9]. Zmíním se tedy jen agentech bezprostředně souvisejících s naším úkolem.

Nejprve se vrátím k definicím pojmů, které už jsem použila v abstraktu.

Datová struktura popisuje všechny volné proměnné objektu, který chceme učit. V případě dopředné sítě to jsou váhy včetně prahů, a případně konstanty pro učící algoritmy.

Konfigurace je konkrétní nastavení proměnných v datové struktuře, například váhový vektor už naučené sítě.

Agent obsahuje libovolný počet datových struktur a učících či pomocných algoritmů.

Agenti v Bangu jsou různých druhů. Agenti systémoví zajišťují mnoho užitečných služeb, které prostředí díky snaze být co nejjednodušší nedělá. UI agenti dělají tu práci, kvůli které je Bang psán, tj. obalují nějakou datovou strukturu a její učící algoritmy a nakonec pomocní agenti poskytují různé, většinou numerické, operace pro ostatní agenty.

Zmíním se o dvou systémových agentech, Zlatých a Bílých stránkách. *Zlaté stránky* jsou databází žijících (rozuměj běžících) agentů. Uchovávají informace o jejich jménech, struktuře a schopnostech. Každý agent se ihned po vytvoření registruje u Zlatých stránek a informuje je při přemístění na jiný počítač či při ukončení.

Bílé stránky udržují informace o agentech, kteří neběží, nicméně je možné je spustit. Také zajišťují persistenci dat, to znamená, že ukládají stav agentů, kteří o to požádají, ať už jde o naučenou konfiguraci, či dočasně přerušený výpočet.

Záznam pro jednu strukturu či konfiguraci má následující položky: typ, popis, vnitřní stav a pomocné informace. Typ a popis třídí datové struktury podle dvou různých hledisek. *Typ* třídí podle struktury, hierarchicky, například orientovaný graf → neuronová síť → konfigurace neuronové sítě. *Popis* podle funkce, taktéž hierarchicky.

3. Jazyky

Máme několik možností, jak zaznamenávat schopnosti datové struktury či konfigurace. Buď přirozeným jazykem, ovšem potom je třeba řešit jeho nejednoznačnost a pomalé vyhledávání, nebo jednoznačně určenými hesly (jednoznačnost by se zaručila pomocí ontologií (viz následující odstavec)), což považuji za ne dost obecné řešení. Nebo pomocí nějakého jazyka pro popis znalostí agentů. Vzhledem k tomu, že agenti jsou poměrně populární, je takovýchto jazyků poměrně dost.

Výhody posledního řešení jsou následující: jazyk dost obecný, aby byl schopen popsat totéž, co přirozený jazyk, v případě, že použijeme jazyk dostatečně rozšířený, získáme možnost komunikovat s agenty za hranicemi Bangu.

Jazyky, o kterých budu psát, je možné rozdělit do tří skupin. V první skupině jsou jazyky navržené přímo pro reprezentaci znalostí. Dost rozšířené na to, aby připadaly v úvahu, jsou dva, KIF [5] a ACL-Lisp [4]. Jsou si velice podobné, používají lispovskou syntax, jsou založené na predikátové logice. Potom vyhledávání nemusí dávat přesné výsledky v důsledku nerozhodnutelnosti predikátové logiky. Výhody těchto jazyků jsou v jejich značném rozšíření mezi agenty, nicméně jsou asi o něco obecnější, a tedy i složitější, než potřebujeme.

V druhé skupině jsou jazyky navržené pro přenos dat ve formátu čitelném pro uživatele i počítač. Bývají založeny na XML, což je sympatické, neboť Bang používá XML pro komunikaci mezi agenty. Příklady takovýchto jazyků jsou PMML [6] či XSIL [7].

4. Ontologie

Slova v přirozeném jazyce jsou nejen definovaná poměrně nepřesně, ale také mají v různých kontextech různé významy. Tento problém se alespoň částečně snaží řešit ontologie.

Ontologie jsou seznam definic, věnovaných určité oblasti. Agenti, kteří znají stejnou ontologii, mohou v komunikaci používat termíny v ní definované a očekávat, že si budou rozumět.

Existuje projekt Ontolingua s cílem vytvořit globální databázi ontologií [10]. Jazyk KIF byl navržen právě pro tento účel.

5. Ontologie v Bangu

Námi řešený problém, jak jednoznačně popsat funkčnost částí Bangu, se dá redukovat na problém vytvořit ontologii pro Bang. Vzhledem k tomu, že Ontolingua je poměrně používaná, je vhodné používat jazyk velmi blízký KIFu. Na druhou stranu Bang používá XML pro komunikaci mezi agenty. Navržený jazyk bude podmnožinou KIFu, definovaný v XML.

6. Závěr a výhledy do budoucna

Jazyk pro popis agentů je navržen, zbývá ho naimplementovat do Bangu a ukázat jeho užitečnost na konkrétních příkladech. Vytvoření jazyka, který srozumitelně pro člověka i počítač popisuje strukturu a schopnosti datových struktur a konfigurací, je prvním krokem k daleko zajímavějším problémům, které mimochodem také osvětlují použití agentů v Bangu namísto prostých objektů, a to je vzájemná vyjednávání mezi agenty na spolupráci při řešení problémů a dokonce (polo)automatické vytváření schémat pro hybridní systémy.

References

[1] Pietro P. Bonissone, “Soft computing: the convergence of emerging reasoning technologies”, *Soft Computing*, vol. 1, pp. 6–18, 1997.

[2] Stan Franklin and Art Graesser, “Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents”, *Intelligent Agents III*, pp. 21–35, 1997.

[3] Roman Neruda and Pavel Krušina, “Creating Hybrid AI Models with Bang”, *Signal Processing, Communications and Computer Science*, vol. I, pp. 228–233, 2000.

[4] “Agent Communication Language”, Technical report, Foundation for Intelligent Physical Agents, 1998.

[5] Michael Genesereth and Richard Fikes, “Knowledge interchange format, v3.0 reference manual”, Technical report, Computer Science Department, Stanford University, 1995.

[6] “PMML v1.1 Predictive Model Markup Language Specification”, Technical report, Data Mining Group, 2000.

[7] Roy Williams, “JAVA/XML for Scientific Data”, Technical report, California Institute of Technology, 2000.

[8] Extensible Markup Language (XML) 1.0 (Second Edition) W3C Recommendation, 2000.

[9] Bang2 homepage, http://bang2.sourceforge.net, 2001.

[10] Ontolingua homepage, http://ksl-web.stanford.edu/kst/ontolingua.html.

Comparison of Linear and Neural Network Approximation

doktorand:
TEREZIE ŠIDLOFOVÁ
Institute of Computer Science, Academy of Sciences of the Czech Republic
Pod vodárenskou věží 2, P.0. Box 5, 182 07 Prague 8, Czech Republic
terka@cs.cas.cz

školitel:
VĚRA KŮRKOVÁ
Institute of Computer Science, Academy of Sciences of the Czech Republic
Pod vodárenskou věží 2, P.0. Box 5, 182 07 Prague 8, Czech Republic
vera@cs.cas.cz

obor studia:
Teoretická Informatika

Abstrakt

We prove Maurey-Jones-Barron rates of approximation (extended to $\mathcal{L}^p$-spaces to be valid for one-hidden-layer neural network schemes with activation functions from $\mathcal{L}^\infty$-spaces. We propose extension to this result. We use operator notation to compare NN-approximation to the linear one.

1. Introduction

Neural Network approximation of functions from $\mathbb{R}^d$ to $\mathbb{R}^j$ has been widely studied in recent years. The existence of an arbitrarily close approximation of a continuous or $\mathcal{L}^p$ function by one-hidden-layer network with quite general units has been proven (see e.g. Leshno et al., 1993, Park, Sandberg, 1993. [7], [10])

These are very nice results but insufficient. An important quality of the approximation scheme is the speed of convergence. It is well known that linear approximation schemes have also the property of being able to approximate ”any” function arbitrarily closely. The advantage of neural network schemes is in the flexibility of the interior of this approximation scheme. While the only thing the linear approximation schemes can do to improve closeness to the desired function is to increase the number of parameters, nonlinear schemes, including neural networks, are able to reflect the shape of the desired function also in the properties of the fundamental parts (units) of the approximation scheme, not necessarily increasing the number of units.

We have to mention Maurey, Jones and Barron, who coped with the curse of dimensionality (exponential increase of parameters with growing input dimension) by proving rates of approximation of some schemes to be of the order of $O(1/\sqrt{n})$ for functions satisfying certain conditions on their Fourier transform. These results can be used also for neural network approximation.

In this paper we show a broad range of functions that don't exhibit the curse of dimensionality when approximated by neural networks.

2. Preliminaries

Let $(X, \|.\|)$ be a normed linear space. We denote by $B_r(\|.\|)$ the ball of radius r in norm $\|.\|$ i.e. $B_r(\|.\|) = \{x \in X; \|x\| \leq r\}$.

For $\mathcal{G}$ a subset of $(X, \|.\|)$ and $c \in \mathbb{R}$ we define

$$\mathcal{G}(c) := \{wg; g \in \mathcal{G}, w \in \mathbb{R} \ \& \ |w| \leq c\}.$$

We will denote the $\|.\|$-closure of $\mathcal{G}$ by $\mathrm{cl}_{\|.\|}\, \mathcal{G}$. The linear combination of elements of $\mathcal{G}$ will be denoted by $\operatorname{span} \mathcal{G}$ and linear combination of at most n elements by $\operatorname{span}_n \mathcal{G} := \{\sum_{i=1}^n w_i g_i; w_i \in \mathbb{R}, g_i \in \mathcal{G}\}$. Convex hull of $\mathcal{G}$ shall be denoted by $\operatorname{conv} \mathcal{G}$ and analogously we define convex hull of at most n elements of $\mathcal{G}$: $\operatorname{conv}_n \mathcal{G} := \{\sum_{i=1}^n v_i g_i;\ v_i \in [0,1], \sum_{i=1}^n v_i = 1, g_i \in \mathcal{G}\}$. By $\|f - \mathcal{G}\| := inf_{g \in \mathcal{G}} \|f - g\|$ we denote the distance of f from $\mathcal{G}$. We define the notion of $\mathcal{G}$-variation as the Minkovski functional of the set $\mathrm{cl}\, \mathcal{G}(1)$:

$$\|f\|_{\mathcal{G}} := \inf\{c \in \mathbb{R}^+; f \in \mathrm{cl}_{\|.\|} \operatorname{conv} \mathcal{G}(c)\}.$$

To describe linear approximation we use the notion of Kolmogorov width defined for $Y \subseteq (X, \|.\|)$ as

$$d_n(Y) = \inf_{X_n} \sup_{f \in Y} \|f - X_n\|,$$

where X_n are n-dimensional subspaces of X.

Neural networks consist of interconnected computational units with activation functions depending on parameters and input variables: $\phi(x, a) : \mathbb{R}^d \times \mathbb{R}^k \to \mathbb{R}$, where a are parameters and x input. One hidden layer network with n ϕ-type units computes a function of d variables of the form:

$$f(x) = \sum_{i=1}^n w_i \phi(x, a_i),$$

where $w_i \in \mathbb{R}$, $a_i \in \mathbb{R}^k$ and $n \in \mathbb{N}^+$.

We extend this notion to continuum of hidden units obtaining

$$f(x) = \int_A w(a) \phi(x, a) \mathrm{d}a,$$

where $x \in (H \subseteq \mathbb{R}^d)$, $a \in (A \subseteq \mathbb{R}^k)$. We can consider this scheme in operator terms having

$$f = T_\phi(w),$$

where T_ϕ is an operator from the set of function on A (for example $\mathcal{L}^p(A, \lambda_k)$) into a set of functions on $\mathbb{R}^d$ (possibly also $\mathcal{L}^p(H, \lambda_d)$, where $H \subseteq \mathbb{R}^d$.) By $s_n(T_\phi)$ we denote the n-th singular number of the operator T_ϕ.

We would like to remind the reader of the Luzin's Theorem, that characterizes approximation of functions by continuous ones (for the proof see for example [9].)

Theorem 2.1 *Let (P, μ) be a locally compact space, where μ is complete Radon measure (for example the Lebesgue measure on $\mathbb{R}^n$) and f be μ-almost everywhere finite function on P. Then the following conditions are equivalent:*

- *(i) f is μ-measurable;*
- *(ii) for any $\varepsilon > 0$ and any compact set $K \subset P$ there is an open set E so that $\mu E < \varepsilon$ and $f\big|_{K \setminus E}$ is continuous;*
- *(iii) for any $\varepsilon > 0$ and any compact set $K \subset P$ there exists a continuous function $\widetilde{f}$ on P such that*

$$\mu\{x \in K : f(x) \neq \widetilde{f}(x)\} < \varepsilon;$$

(iv) for every compact set $K \subset P$ there exists a sequence $\{\widetilde{f}_n\}$ of continuous functions on P such that

$$\widetilde{f}_n \to f \;\; \mu - \textit{almost everywhere on } K.$$

An easy consequence of the Luzin's theorem for P a countable union of compact sets (any $\mathbb{R}^n$) gives a more useful form of statement (iii):

(iii') for any $\varepsilon > 0$ there exists a continuous function $\widetilde{f}$ on P and an open set E such that $\mu E < \varepsilon$ and

$$\widetilde{f} = f \textit{ on } P \setminus E.$$

3. Deriving Rates of Approximation

It is useful to consider neural network approximation in the form of its limit as an integral thus having instead of the finite sum $f(x) = \sum_{i=1}^{n} w_i \phi(x, a_i)$ the integral of the form $f(x) = \int_A w(a)\phi(x,a)\mathrm{d}a$, $f : H \to \mathbb{R}$, where A is a compact subset of $\mathbb{R}^k$, H a compact subset of $\mathbb{R}^d$ (compactness is reasonable to demand in real case.) We use an extension to the Maurey-Jones-Barron theorem by Darken et al. [3], reformulated in [6], that estimates rates of approximation by functions from $\mathrm{span}_n\, \mathcal{G}$, where $\mathcal{G}$ is a bounded subset of $\mathcal{L}_p$-space, $1 < p < \infty$. These estimates don't exhibit the curse of dimensionality.

Theorem 3.1 *Let X be a finite measure space. Let $\mathcal{G}$ be a bounded subset of $\mathcal{L}_p(X)$, $1 < p < \infty$ and $s_{\mathcal{G}} = \sup_{g \in \mathcal{G}} \|g\|$. Then for every $f \in \mathcal{L}_p(X)$ and $n \in \mathbb{N}^+$*

$$\|f - \mathrm{span}_n\, \mathcal{G}\|_p \leq \frac{2^{1+1/\bar{p}} \|f\|_{\mathcal{G}} s_{\mathcal{G}}}{n^{1/\bar{q}}},$$

where $q = p/(p-1)$, $\bar{p} = \min(p,q)$ and $\bar{q} = \max(p,q)$.

Note that for $p = 2$ we get the original Maurey-Jones-Barron rates of approximation of the order of $\mathcal{O}(1/\sqrt{n})$.

Observing the results of theorem 3.1 we see that reasonable rates of approximation are obtainable for functions with finite $\|f\|_{\mathcal{G}}$. We prove this to be true for $\mathcal{G}$ containing quite general activation functions of neural network units.

We have $\mathcal{G}$ as the set of functions computable by NN units. The linear combination of n such functions can be computed by an n-hidden-unit network with one linear output unit.

Now we need to find conditions that would guarantee the desired function $f(x) = \int_A w(a)\phi(x,a)\mathrm{d}a$ to be in the convex closure $\mathrm{cl\,conv}_{\|.\|}\, \mathcal{G}$ (thus the norm $\|f\|_{\mathcal{G}}$ be finite) to apply Theorem 3.1 for neural networks. The function f has been proven to be in the convex closure of $\mathcal{G}$ for ϕ, w continuous functions, see Kůrková et al. [4].

Theorem 3.2 *Let d, k be any positive integers, H be a compact subset of $\mathbb{R}^d$ and let $f \in C(H)$ be any function that can be represented as $f(x) = \int_A w(a)\phi(x,a)\mathrm{d}a$, where $A \subseteq \mathbb{R}^k$, $w \in C(A)$ is compactly supported and $\phi \in C(\mathbb{R}^d \times A)$. Denote $\mathcal{G} = \{\phi(x,a), a \in A\}$, $M = \{a \in A; \exists x \in H :: w(a)\phi(x,a) \neq 0\}$. Then*

$$\|f\|_{\mathcal{G}} \leq \int\limits_M |w(a)|\mathrm{d}a$$

Theorem 3.2 can be reformulated in terms of operators and $\mathcal{G}$-variation as follows:

Corollary 3.3 *Under assumptions of the previous theorem it holds:*

$$\|T_\phi(w)\|_{\mathcal{G}_\phi} \leq \|w\|_1$$

Now we present extension of this theorem to $\mathcal{L}_p$-spaces. We make use of the Luzin's Theorem 2.1 to prove $\|f\|_{\mathcal{G}}$ to be finite for f computed by one-hidden-layer neural network with $\mathcal{L}^\infty$ activation functions.

Theorem 3.4 *Let k, d be positive integers, A a compact subset of $\mathbb{R}^k$ and H a compact subset of $\mathbb{R}^d$. Let $w \in \mathcal{L}_p(A, \lambda_k)$ and $\phi \in \mathcal{L}_p(A \times H, \lambda_{k+d})$ for some $1 < p < \infty$ such that there exists a b so that $w \leq b$ λ_k-almost everywhere on A and $\phi \leq b$ λ_{k+d}-almost everywhere on $A \times H$. Let $f(x) := \int_A w(a)\phi(x,a)\mathrm{d}a$ and $\mathcal{G} = \{\phi(x,a), a \in A\}$. Then*

$$\|f\|_{\mathcal{G}} \leq \|w\|_1.$$

Proof:

The statement can be equivalently rewritten as follows:

$$f(x) = \int_A w(a)\phi(x,a)\mathrm{d}a \in \mathrm{cl}_p \operatorname{conv}\{c_i\phi(x,a_i), a_i \in A, |c_i| \leq \|w\|_1\}.$$

It suffices to show:

$$(\forall \varepsilon > 0)(\exists g \in \operatorname{conv}\{c_i\phi(x,a_i), a \in A, c_i \leq \|w\|_1\})$$

such that

$$\|f - g\|_p < \varepsilon,$$

where

$$f(x) := \int_A w(a)\phi(x,a)\mathrm{d}a$$

Thus we are searching for a g of the form $g = \sum_{i=1}^m c_i\phi(x,a_i)$ that is within ε from f and $\sum_{i=1}^m c_i \leq \|w\|_1$.

We will make use of the Luzin's Theorem 2.1 (iii') to derive useful facts from 3.2: For all $\varepsilon_n > 0$ there exist continuous functions $\widetilde{w}_n$ and $\widetilde{\phi}_n$ on A and $A \times H$ respectively and sets E_n and F_n, such that

$$\widetilde{w}_n = w \text{ on } A \setminus E_n$$

and

$$\widetilde{\phi}_n = \phi \text{ on } (A \times H) \setminus F_n,$$

where $\lambda_k(E_n) < \varepsilon_n$ and $\lambda_{k+d}(F_n) < \varepsilon_n$. Now we define an open set

$$U_n \supseteq \big\{\{a \in A; \lambda_d\{(a,-) \in F_n\} > \sqrt{\varepsilon_n}\} \cup E_n\big\}$$

We observe that $\lambda_k(U_n) < \sqrt{\varepsilon_n} + \varepsilon_n$ (use the Fubini's Theorem for the first part). Thus we have $\lambda_k(U_n) < 2\sqrt{\varepsilon_n}$ for $\varepsilon_n < 1$. Now we define

$$f_n(x) := \int_{A \setminus U_n} \widetilde{\phi}_n(x,a)\widetilde{w}_n(a)\mathrm{d}a.$$

Now split the further proof (the formula $\|f - g\|_p$ into three parts:

$$\|f - g\|_p \leq \overbrace{\|f - f_n\|_p}^{\mathbf{A}} + \overbrace{\|f_n - \sum_{i=1}^{m_n} c_i\widetilde{\phi}_n(-,a_i)\|_p}^{\mathbf{B}} + \overbrace{\|\sum_{i=1}^{m_n} c_i\widetilde{\phi}_n(-,a_i) \ - \sum_{i=1}^{m_n} c_i\phi(-,a_i)\|_p}^{\mathbf{C}}$$

and prove all of them to be less or equal to $\varepsilon/3$, where ε was fixed at the beginning.

(A) Without loss of generality we put $p = 1$ for better transparency.

$$\|f - f_n\|_p = \overbrace{\int_H \int_A |w(a)\phi(x,a) - \widetilde{w}_n(a)\widetilde{\phi}_n(x,a)|\mathrm{d}a\mathrm{d}x}^{(1)} + \overbrace{\int_H \int_{U_n} |\widetilde{w}_n(a)\widetilde{\phi}_n(x,a)|\mathrm{d}a\mathrm{d}x}^{(2)}$$

We deal with both the parts separately (and use Fubini's Theorem):

$$(1) = \int_{\left(H\times(A\setminus E_n)\right)\setminus F_n} \underbrace{|w(a)\phi(x,a) - \widetilde{w}_n(a)\widetilde{\phi}_n(x,a)|}_{=0}\,\mathrm{d}a\mathrm{d}x +$$

$$+ \int_{F_n} \underbrace{|w(a)\phi(x,a) - \widetilde{w}_n(a)\widetilde{\phi}_n(x,a)|}_{\leq 2b^2}\,\mathrm{d}a\mathrm{d}x +$$

$$+ \int_{E_n}\int_H \underbrace{|w(a)\phi(x,a) - \widetilde{w}_n(a)\widetilde{\phi}_n(x,a)|}_{\leq 2b^2}\,\mathrm{d}a\mathrm{d}x \leq$$

$$\leq 0 + \varepsilon_n 2b^2 + \varepsilon_n\lambda_d(H)2b^2 = \varepsilon_n 2b^2(1+\lambda_d(H))$$

The second part is easier:

$$(2) = \int_{U_n}\int_H \underbrace{|\widetilde{w}_n(a)\widetilde{\phi}_n(x,a)|}_{\leq b^2}\,\mathrm{d}a\mathrm{d}x \leq \sqrt{\varepsilon_n}\lambda_d(H)b^2.$$

Altogether we get:

$$\|f - f_n\| \leq \lambda_d(H)b^2(4\varepsilon_n + \sqrt{\varepsilon_n}).$$

Now we can easily find n such that $\|f - f_n\| \leq \varepsilon/3$. We fix this n and continue with the second part:

(B) We have f_n defined as integral over a compact set $A \setminus U_n$ of continuous functions $\widetilde{w}_n(a)$ and $\widetilde{\phi}_n(x,a)$.

$$f_n(x) := \int_{A\setminus U_n} \widetilde{\phi}_n(x,a)\widetilde{w}_n(a)\mathrm{d}a.$$

Thus using Theorem 3.2 we know:

$$\forall\xi > 0\ \exists m_n :: \|f_n - \sum_{i=1}^{m_n} c_i\widetilde{\phi}_n(-,a_i)\|_C < \xi$$

We find such m_n so that $\|f_n - \sum_{i=1}^{m_n} c_i\widetilde{\phi}_n(-,a_i)\|_C < \varepsilon/(3\lambda_d(H))$. Thus we get

$$\|f_n - \sum_{i=1}^{m_n} c_i\widetilde{\phi}_n(-,a_i)\|_p = \int_H \underbrace{|f_n(x) - \sum_{i=1}^{m_n} c_i\widetilde{\phi}_n(x,a_i)|}_{<\varepsilon/(3\lambda_d(H))}\,\mathrm{d}x \leq \frac{\varepsilon}{3}$$

We proceed to the third part.

(C) We have to notice that by $F_n\big|_H$ we mean projection of F_n onto H.

$$\|\sum_{i=1}^{m_n} c_i\widetilde{\phi}_n(-,a_i) - \sum_{i=1}^{m_n} c_i\phi(-,a_i)\|_p = \int_H |\sum_{i=1}^{m_n} c_i\widetilde{\phi}_n(x,a_i) - \sum_{i=1}^{m_n} c_i\phi(x,a_i)|\mathrm{d}x \leq$$

$$\int_{H\setminus(F_n\big|_H)} \underbrace{|\sum_{i=1}^{m_n} c_i\widetilde{\phi}_n(x,a_i) - \sum_{i=1}^{m_n} c_i\phi(x,a_i)|}_{=0}\,\mathrm{d}x + \int_{F_n\big|_H} |\sum_{i=1}^{m_n} c_i\widetilde{\phi}_n(x,a_i) - \sum_{i=1}^{m_n} c_i\phi(x,a_i)|\mathrm{d}x$$

$$\leq 0 + \int_{F_n\big|_H} |\sum_{i=1}^{m_n} c_i(\widetilde{\phi}_n - \phi)(x,a_i)|\mathrm{d}x \leq \sum_{i=1}^{m_n}|c_i|\int_{F_n\big|_H} \underbrace{|(\widetilde{\phi}_n - \phi)(x,a_i)|}_{\leq 2b}\,\mathrm{d}x = *$$

As of $a_i \notin U_n$ we obtain the constraint on the measure of the set over which we are integrating:

$$\lambda_d(F_n\big|_H) \leq \sqrt{\varepsilon_n}$$

for all a_i. We conclude

$$* \leq 2b\sqrt{\varepsilon_n}\sum_{i=1}^{m_n} c_i \leq 2b\sqrt{\varepsilon_n}\|w\|_1,$$

because

$$\sum_{i=1}^{m_n} c_i \leq \int_{A\setminus U_n} |\widetilde{w}_n(a)|\mathrm{d}a = \int_{A\setminus U_n} |w(a)|\mathrm{d}a \leq \|w\|_1$$

It is easy to find an ε_n that secures all of the A, B, C parts to be $\leq \varepsilon/3$ which is what we wanted.

□

Corollary 3.5 *Under the assumtpions of Theorem 3.4 we have:*

$$\|T_\phi(w)\|_{\mathcal{G}_\phi} \leq \|w\|_1.$$

Proof: An easy consequence of Theorem 3.4 and definitions of T_ϕ and $\|.\|_{\mathcal{G}_\phi}$. □

4. Linear versus Nonlinear Approximation

To derive estimates of Kolmogorov n-width of balls in $\mathcal{G}$-variation we can use the following Theorem from Pinkus [11, p. 65].

Theorem 4.1 *Let $(X, \|.\|_X)$ and $(Y, \|.\|_Y)$ be two Hilbert spaces and $T : X \to Y$ be a compact operator. Then for every positive integer n*

$$d_n(T(B_1(X)), Y) = s_{n+1}(T)\,.$$

We use this Theorem together with the results of the previous section having for $f(x) := \int_A w(a)\phi(x,a)\mathrm{d}a$ and $\mathcal{G} = \{\phi(x,a), a \in A\}$:

$$\|f\|_{\mathcal{G}} \leq \|w\|_1$$

We apply the operator notation introduced at the beginning to reformulate this result. We have $f = T_\phi(w)$. Using the previous inequality we obtain:

$$T_\phi(B_1(\|.\|_1)) \subseteq B_1(\|.\|_{\mathcal{G}}).$$

Using Theorem 4.1 for $p = 2$ we have:

$$d_n(B_1(\|.\|_{\mathcal{G}})) \geq d_n(T_\phi(B_1(\|.\|_1)) = s_{n+1}(T)$$

Our futher focus is on the speed of s_n going to zero. We want to derive conditions under which the singular numbers converge to zero slower than $O(\sqrt{n})$ thus proving neural network approximation to be better than linear approximation (its speed of convergence given by d_n).

5. Discussion

In Theorem 3.4 (in combination with Theorem 3.1) we have proven the rates of approximation of the neural network scheme with one hidden layer and $\mathcal{L}^\infty$ activation functions to be of the order $O(n^{1/\bar{q}})$ (q derived from the embedding into $\mathcal{L}^p$-space, see 3.1) thus avoiding the curse of dimensionality. We have shown the possible progress in the field using operator notation and deriving rates of convergence of singular numbers of compact operators.

We expect to extend the results of Theorem 3.4 to general $\mathcal{L}^p$-functions on finite measure spaces without the restriction on up to measure zero finiteness of the function.

References

[1] Barron, A. R. (1992) Neural net approximation. In *Proceedings of the 7th Yale Workshop on Adaptive and Learning Systems* (pp. 69–72)

[2] Barron, A. R. (1993) Universal approximation bounds for superposition of a sigmoidal function. *IEEE Transactions on Information Theory*, **39**, 930–945

[3] Darken C., Donahue M., Gurvits L., Sontag E., Rate of Approximation Results Motivated by Robust Neural Network Learning, *In proceedings of the 6th Annual ACM Conference on Computational Learning Theory*, Santa Cruz, CA, pp. 303-309, 1993.

[4] Kůrková V., Kainen P.C., Kreinowich V., Estimates of the Number of Hidden Units and Variation with Respect to Half-Spaces, *Neural Networks*, vol. 10, no. 6, pp. 1061-1068, 1997.

[5] Kůrková V., Neural Networks as Nonlinear Approximators, *Proceedings of the Second ICSC Symposium on Neural Computation*, Berlin, pp. 29-35, 2000

[6] Kůrková V., High-dimensional approximation by neural networks. *In proceedings of Theory and Practice* (to appear.)

[7] Leshno M., Lin V., Pinkus A., Schocken S. (1993) Multilayer feedforward networks with a non-polynomial activation function can approximate and function *Neural Networks*, **6**, 861-867.

[8] Lukeš J., *Zápisky z funkcionální analýzy*, Karolinum, UK Praha, 2002.

[9] Lukeš J., Malý J., *Measure and Integral*, Matfyzpress, Praha, 1995.

[10] Park J., Sandberg I. W. (1993) Approximation and radial-basis-function networks *Neural Computation*, **5**, 305-316.

[11] Pinkus A., *n-Widths in Approximation Theory*, Springer-Verlag, Berlin-Heidelberg, Germany, 1985.

Radial Implicative Fuzzy Inference Systems

doktorand:
ING. DAVID COUFAL
Institute of Computer Science

david.coufal@cs.cas.cz

školitel:
DOC. ING. STANISLAV KREJČÍ, CSC.
University of Pardubice

Stanislav.Krejci@upce.cz

obor studia:
Technical cybernetics

Abstrakt

The paper surveys the basic knowledge about the special class of fuzzy inference systems which is the class of the radial implicative fuzzy inference systems. It is presented their definition together with several important properties of them such as coherence and universal approximation property.

1. Radial fuzzy inference systems

The concept of fuzzy inference system (FIS) is well known over thirty years, e.g., [1]. The architecture of standard FIS is given by four building blocks, a fuzzifier, a rule base, an inference engine and a deffuzifier. The input signal flows from fuzzifier, through inference engine, which cooperates with a rule base, to the deffuzifier. Mathematically, a FIS (in MISO configuration) performs a function from $\mathcal{R}^n$ to $\mathcal{R}$.

A "knowledge" of a FIS is stored in the rule base. This is traditionally given by a set of m IF-THEN rules. An IF-THEN rule has the canonical form

$$\text{IF } \; x_1 \; is \; A_{j1} \;\; and \;\; x_2 \; is \; A_{j2} \;\; and \; \ldots \; and \;\; x_n \; is \; A_{jn} \;\; \text{THEN } \; y \; is \; B_j, \qquad \text{(4)span}$$

where $A_{ji}, B_j, i = 1, \ldots, n, j = 1, \ldots, m$ are fuzzy sets defined on respective universal sets $X_1, \ldots, X_n$, $X_i \subseteq \mathcal{R}$, $Y \subseteq \mathcal{R}$. The linguistic connective and is represented by a t-norm (associative, commutative, monotone and conjunction-like operation from $[0, 1]^2$ to $[0, 1]$, see [1, 2] for exact definition).

Considering $n > 1$ we have antecedent of a rule representing a fuzzy relation on $X_1 \times X_2 \times \cdots \times X_n$ given as

$$A_j(\boldsymbol{x}) = A_{j1}(x_1) \star A_{j2}(x_2) \star \cdots \star A_{jn}(x_n), \qquad \text{(5)span}$$

where $\star$ symbol represents a t-norm. Having particular fuzzy sets given as Gaussians and t-norm as product we have above as

$$A_j(\boldsymbol{x}) = \exp\left[-\frac{(x_1 - a_{j1})^2}{b_{j1}^2}\right] \cdot \exp\left[-\frac{(x_2 - a_{j2})^2}{b_{j2}^2}\right] \cdot \cdots \cdot \exp\left[-\frac{(x_n - a_{jn})^2}{b_{jn}^2}\right]. \qquad \text{(6)span}$$

which is on base of Gaussians properties

$$A_j(\boldsymbol{x}) = \exp\left[-\sum_{i=1}^{n}\frac{(x_i - a_{ji})^2}{b_{ji}^2}\right] = \exp\left[-||\boldsymbol{x} - \boldsymbol{a}_j||_{E_{\boldsymbol{b}}}^2\right], \tag*{(7)span}$$

where $\boldsymbol{a}_j = (a_{j1}, \ldots, a_{jn})$, $\boldsymbol{a}_j \in \mathcal{R}^n$, is the vector of centers of particular fuzzy sets A_{ji}, $\boldsymbol{b}_j = (b_{j1}, \ldots, b_{jn})$, $\boldsymbol{b} \in \mathcal{R}_+^n$ is the vector of theirs width parameters and $|| \cdot ||_{E_{\boldsymbol{b}}}$ is the scaled Euclidean norm defined as

$$||\boldsymbol{u}||_{E_{\boldsymbol{b}}} = \sqrt{\sum_{i=1}^{n}\frac{u_i^2}{b_i^2}}. \tag*{(8)span}$$

Comparing the form of antecedent and succedent,

$$A_j(\boldsymbol{x}) = \exp\left[-||\boldsymbol{x} - \boldsymbol{a}_j||_{\boldsymbol{b}}^2\right] \qquad B_j(y) = \exp\left[-(y-c)^2/d^2\right], \tag*{(9)span}$$

we see that they have the same form. That is, their computation is given, up to dimension, by the same radial basis function. Actually, this property gives the formal definition of radial FIS.

A FIS is called radial if there exists a non-increasing function $act : [0, +\infty) \to [0, 1]$, $act(0) = 1$, $\lim_{s\to+\infty} act(s) = 0$, and a strictly increasing function $g : [0, +\infty) \to [0, +\infty)$, $g(0) = 0$, both continuous, such that antecedent $A_j(\boldsymbol{x})$ and succedent $B_j(y)$ of the jth rule, $j = 1, \ldots, m$, can be written as

$$A_j(\boldsymbol{x}) = act(\ g(||\boldsymbol{x} - \boldsymbol{a}_j||_{\boldsymbol{b}_j})\ ) \quad \text{and} \quad B_j(y) = act\,(\,g\,(|y - c_j|/d_j)\,)\,. \tag*{(10)span}$$

On base of this definition we see that in above example the *act* function is given as $act(s) = \exp(-s)$ and g function as $g(s) = s^2$. There is a question if there can be defined another radial FISs on base of other well known t-norms such as Lukasiewicz, defined as $x \star y = \max\{0, x + y - 1\}$ and minimum $x \star y = \min\{x, y\}$ and other shapes of fuzzy sets such as triangular ones. We have these two lemmas.

Lemma 1.1 *Let A_{ji}, B_j be triangular fuzzy sets, i.e.,*

$$A_{ji} = \max\left\{0, 1 - \left|\frac{x_i - a_{ji}}{b_{ji}}\right|\right\}, \qquad B_j = \max\left\{0, 1 - \left|\frac{y - c_j}{d_j}\right|\right\}, \tag*{(11)span}$$

and t-norm be chosen as minimum then resulting FIS *is radial.*

Proof: Considering antecedent of particular rule we have

$$\begin{aligned}
A_j(\boldsymbol{x}) &= \min_i\left\{\max\left\{0, 1 - \left|\frac{x_i - a_{ji}}{b_{ji}}\right|\right\}\right\}, && (12)\\
A_j(\boldsymbol{x}) &= \max\left\{0, \min_i\left\{1 - \left|\frac{x_i - a_{ji}}{b_{ji}}\right|\right\}\right\}, && (13)\\
A_j(\boldsymbol{x}) &= \max\left\{0, 1 - \max_i\left\{\left|\frac{x_i - a_{ji}}{b_{ji}}\right|\right\}\right\}. && (14)
\end{aligned}$$

Hence it is

$$A_j(\boldsymbol{x}) = \max\{0, 1 - ||\boldsymbol{x} - \boldsymbol{a}_j||_{C_{\boldsymbol{b}_j}}\}, \tag*{(15)span}$$

where $|| \cdot ||_{C_{\boldsymbol{b}_j}}$ is the scaled cubic norm given as

$$||\boldsymbol{u}||_{C_{\boldsymbol{b}_j}} = \max\left\{\left|\frac{u_1}{b_1}\right|, \ldots, \left|\frac{u_n}{b_n}\right|\right\}, \tag*{(16)span}$$

for $\boldsymbol{b} = (b_1, \ldots, b_n)$, $\boldsymbol{b} \in \mathcal{R}_+^n$.

Setting $act(s) = \max\{0, 1 - s\}$ for $s \in [0, +\infty)$ and $g(s) = s$, we have

$$\begin{aligned} A_j(\boldsymbol{x}) &= act(\, g(||\boldsymbol{x} - \boldsymbol{a}||_{C_{\boldsymbol{b}_j}}\,)\,), && (17) \\ B_j(y) &= \max\left\{0, 1 - \left|\frac{y - c_j}{d_j}\right|\right\} = act\left(g\left(\left|\frac{y - c_j}{d_j}\right|\right)\right). && (18) \end{aligned}$$

Hence Mamdani I-FIS is radial. □

Lemma 1.2 *Let* A_{ji}, B_j *be triangular fuzzy sets, i.e.,*

$$A_{ji} = \max\left\{0, 1 - \left|\frac{x_i - a_{ji}}{b_{ji}}\right|\right\}, \qquad B_j = \max\left\{0, 1 - \left|\frac{y - c_j}{d_j}\right|\right\}, \qquad (19)\text{span}$$

and t*-norm be chosen as Lukasiewicz one then resulting* FIS *is radial.*

Proof: By induction. Let $n = 2$ then we have for $r_1 = (x - a_{j1})/b_{j1}$, $r_2 = (x - a_{j2})/b_{j2}$,

$$\begin{aligned} A_j(\boldsymbol{x}) &= \max\{0, \max\{0, 1 - |r_1|\} + \max\{0, 1 - |r_2|\} - 1\}, && (20) \\ A_j(\boldsymbol{x}) &= \max\{0, 1 - (|r_1| + |r_2|)\}. && (21) \end{aligned}$$

By the same manipulation we prove that for $n > 2$ we have

$$A_j(\boldsymbol{x}) = \max\left\{0, 1 - \sum_{i=1}^{n} |r_i|\right\}. \qquad (22)\text{span}$$

Considering the scaled octaedric norm in $\mathcal{R}^n$ defined as

$$||\boldsymbol{u}||_{O_{\boldsymbol{b}}} = \sum_{i=1}^{n} \left|\frac{u_i}{b_i}\right|, \qquad (23)\text{span}$$

we have

$$A_j(\boldsymbol{x}) = \max\{0, 1 - ||\boldsymbol{x} - \boldsymbol{a}_j||_{O_{\boldsymbol{b}}}\}. \qquad (24)\text{span}$$

Setting again $act(s) = \max\{0, 1 - s\}$ for $s \in [0, +\infty)$ and $g(s) = s$ we obtain a radial FIS. □

Note that considering l_p norms in $\mathcal{R}^n$ defined for $p \geq 1$ by

$$l_p(\boldsymbol{u}) = (|u_1|^p + \dots |u_n|^p)^{1/p} \qquad (25)\text{span}$$

then their scaled counterparts for $\boldsymbol{b} \in \mathcal{R}^n_+$

$$l_{p_{\boldsymbol{b}}}(\boldsymbol{u}) = \left(\left|\frac{u_1}{b_1}\right|^p + \dots + \left|\frac{u_n}{b_n}\right|^p\right)^{1/p} \qquad (26)\text{span}$$

are norms in $\mathcal{R}^n$ as well. Further it is

$$l_{(p=1)_{\boldsymbol{b}}}(\boldsymbol{u}) = \left|\frac{u_1}{b_1}\right| + \dots + \left|\frac{u_n}{b_n}\right| = ||\boldsymbol{u}||_{O_{\boldsymbol{b}}}, \qquad (27)\text{span}$$

$$\lim_{p \to +\infty} l_{p_{\boldsymbol{b}}}(\boldsymbol{u}) = \max\left\{\left|\frac{u_1}{b_1}\right| + \dots + \left|\frac{u_n}{b_n}\right|\right\} = ||\boldsymbol{u}||_{C_{\boldsymbol{b}}}. \qquad (28)\text{span}$$

Thus we see that for the most important t-norms (any t-norm can be build from Lukasiewicz, product and minimum ones [2]) and usual shapes of fuzzy sets (triangular, Gaussians) there exists corresponding radial FIS.

2. Radial implicative fuzzy inference systems

A particular IF-THEN rule of a rule base is mathematically represented by a fuzzy relation. Its form depends on the shapes of fuzzy sets, used t-norm and interpretation of IF-THEN structure of a rule. Actually, there are two approaches known in the literature. It is the conjunctive approach and the implicative approach [1]. On the base of conjunctive one so called conjunctive FISs are defined and on base of the implicative one the implicative FISs (I-FISs) are defined.

In an implicative FIS antecedent and succedent are combined by a proper fuzzy implication given as the residuum of a t-norm used for and connective representation. A residuum $\rightarrow$ of a t-norm is generally given as (it is an operation from $[0,1]^2$ to [0,1], see [1, 2])

$$x \rightarrow y = \sup_z \{x \star z \leq y\}. \tag{29}$$span

For the three basic norms it is given as $x \rightarrow y$=1 iff $x \leq y$ (this is a general property of all residua) and for $x > y$ as

- Lukasiewicz t-norm: $x \rightarrow y = 1 - x + y$
- product t-norm: $x \rightarrow y = y/x$
- minimum t-norm: $x \rightarrow y = y$

Hence in an I-FIS a particular rule representation is given as

$$\begin{aligned} R_j(\boldsymbol{x}, y) &= A_j(\boldsymbol{x}) \rightarrow B_j(y), & (30)\\ R_j(\boldsymbol{x}, y) &= (A_{j1}(x_1) \star \cdots \star A_{jn}(x_n)) \rightarrow B_j(y). & (31)\end{aligned}$$

On base of this representation particular rules are combined to compound relation giving a fuzzy relation representing the whole rule base. The combination is for implicative FIS given by the t-norm representing a fuzzy intersection. The t-norm is the same as used for and connective representation. Thus we have

$$RB(\boldsymbol{x}, y) = \bigcap_{j=1}^{m} R_j(\boldsymbol{x}, y) = R_1(\boldsymbol{x}, y) \star \cdots \star R_n(\boldsymbol{x}, y). \tag{32}$$span

Now we can state the definition of radial I-FIS. A FIS is radial implicative one if it is radial according to definition 1 and it has the implicative representation of rule base.

3. Computation of radial I-FIS

A computation of standard FIS is given by the compositional rule of inference [1]. On base of this rule output (fuzzy set B') of inference engine, given as a response on input $\boldsymbol{x}^*$, has the form

$$B'(y) = \sup_{\boldsymbol{x}} \{A'_{\boldsymbol{x}^*}(\boldsymbol{x}) \star RB(\boldsymbol{x}, y)\}, \tag{33}$$span

where $A'_{\boldsymbol{x}^*}(\boldsymbol{x})$ is the fuzzy set given by a fuzzifier as a response on the crisp input $\boldsymbol{x}^* \in \mathcal{R}$ and $\star$ is the t-norm used for and connective representation. Using singleton fuzzifier, which is the most common choice in practice, transforming a crisp input on fuzzy singleton, i.e.,

$$fuzz(\boldsymbol{x}^*) = A'_{\boldsymbol{x}^*}(\boldsymbol{x}) = \left\{ \begin{array}{ll} 1 & \text{for } \boldsymbol{x} = \boldsymbol{x}^* \\ 0 & \text{for } \boldsymbol{x} \neq \boldsymbol{x}^* \end{array} \right. , \tag{34}$$span

and employing the fact that for any t-norm $0 \star x = x \star 0 = 0$ we have above CRI rule (33) in the simpler form of

$$B'(y) = RB(\boldsymbol{x}^*, y). \tag{35}$$span

Further considering the computation of CRI rule in a more restrictive form, so called Δ-CRI rule, we have above

$$B'(y) = \Delta(RB(\boldsymbol{x}^*, y)), \qquad \text{(36)span}$$

where Δ is an operation from [0,1] to $\{0, 1\}$ defined as

$$\Delta(x) = \begin{cases} 1 & \text{for } x = 1, \\ 0 & \text{for } x \in [0, 1). \end{cases} \qquad \text{(37)span}$$

Since for any t-norm it is $x \star y = 1$ if and only if $x = 1$ and $y = 1$ we can (36) write as

$$B'(y) = \Delta(R_1(\boldsymbol{x}^*, y)) \star \cdots \star \Delta(R_n(\boldsymbol{x}^*, y)). \qquad \text{(38)span}$$

So $B'(y) = 1$ if and only if $R_j(\boldsymbol{x}^*, y) = 1$ simultaneously for all $j = 1, \ldots, m$. Otherwise $B'(y) = 0$. Hence B' is a crisp set.

Having particular rule represented in an implicative way it is $R_j(\boldsymbol{x}^*, y) = 1$ if and only if it is $A_j(\boldsymbol{x}^*) \leq B_j(y)$, which is given by the properties of residua. Considering an implicative FIS to be radial we can state these inequalities.

$$\begin{array}{rcll} d_j \cdot ||\boldsymbol{x}^* - \boldsymbol{a}_j||_{\boldsymbol{b}_j} & \geq & |y - c_j|, & (39) \\ ||\boldsymbol{x}^* - \boldsymbol{a}_j||_{\boldsymbol{b}_j} & \geq & |y - c_j|/d_j, & (40) \\ g(||\boldsymbol{x}^* - \boldsymbol{a}_j||_{\boldsymbol{b}_j}) & \geq & g(|y - c_j|/d_j), & (41) \\ act(g(||\boldsymbol{x}^* - \boldsymbol{a}_j||_{\boldsymbol{b}_j})) & \leq & act(g(|y - c_j|/d_j)), & (42) \\ A_j(\boldsymbol{x}^*) & \leq & B_j(y). & (43) \end{array}$$

Hence the set of those y satisfying $A_j(\boldsymbol{x}^*) \leq B_j(y)$contains at least the closed interval

$$I_j = [c_j - d_j \cdot ||\boldsymbol{x}^* - \boldsymbol{a}_j||_{\boldsymbol{b}_j}\ ,\ c_j + d_j \cdot ||\boldsymbol{x}^* - \boldsymbol{a}_j||_{\boldsymbol{b}_j}]. \qquad \text{(44)span}$$

Considering by definition interval I_j such a set of those y for which $R_j(\boldsymbol{x}^*, y) = 1$ we see that the set of y for which $B'(y) = 1$ is given by the intersection of I_j for $j = 1, \ldots, m$. Formally, we can write computation of an inference engine using Δ-CRI rule in the case of radial I-FIS as

$$B' = \bigcap_{j=1}^{m} I_j. \qquad \text{(45)span}$$

Since an intersection of intervals is an interval as well (we consider that it is non empty) it is straithforward to consider as the final deffuzified output of a radial I-FIS the middle point of B', i.e.,

$$y^* = \frac{L(I_{B'}) + R(I_{B'})}{2}, \qquad \text{(46)span}$$

where $L(I_{B'})$, $R(I_{B'})$ are the left or the right limit point of B', respectively.

Having stated the computation of a radial I-FIS we can investigate some of its properties such as coherence and universal approximation property. In the following sections we state important theorems regarding these properties without proofs.

4. Coherence

The question of coherence is the question if for any input is the output of radial I-FIS always defined. That is, for any $\boldsymbol{x}^* \in \mathcal{R}^n$ it is $\bigcap I_j \neq \emptyset$. The answer on this question is given by the following theorem.

Theorem 4.1 *Let be w_{kl} for $k, l = 1, \ldots, m$ given as*

$$w_{kl} = \begin{cases} \dfrac{|c_k - c_l|}{||\boldsymbol{a}_k - \boldsymbol{a}_l||_E} & \text{for } k \neq l \\ 0 & \text{for } k = l \end{cases}, \qquad \text{(47)span}$$

Then radial $\mathrm{I-FIS}$ *is coherent if and only if for all elements* w_{kl} *it is*

$$w_{kl} \leq \min\{d_k\alpha_k, d_l\alpha_l\}, \tag{48}$$span

where α_j*,* $j = 1, \ldots, m$*, are positive numbers such that*

$$\alpha_j \cdot ||\boldsymbol{u}||_E \leq ||\boldsymbol{u}||_{\boldsymbol{b}_j} \tag{49}$$span

for all $\boldsymbol{u} \in \mathcal{R}^n$*.*

5. Universal approximation property

The universal approximation property is an important property which justifies the employment of radial I-FISs as controllers or other approximation tools. We are able to prove this property in the following form.

A system of functions $\mathcal{G}$ defined on hypercube $H = [p_1, q_1] \times \cdots \times [p_n, q_n]$, i.e., $H \subseteq R^n$, exhibits the universal approximation property if for given $\varepsilon > 0$ and any continuous function $f : H \to \mathcal{R}$ there exists a function $g \in \mathcal{G}$ such that for all $\boldsymbol{x} \in H$ it is

$$|f(\boldsymbol{x}) - g(\boldsymbol{x})| < \varepsilon. \tag{50}$$span

Theorem 5.1 *Let* $\mathcal{G}$ *be the system of functions given by the computation of all coherent radial* $\mathrm{I-FISs}$ *defined on given hypercube* H*,* $H \subseteq \mathcal{R}^n$*. Then* $\mathcal{G}$ *has the universal approximation property in the sense of definition 5.*

6. Conclusion

It was defined the class of fuzzy inference systems which was the class of radial implicative fuzzy inference systems. It was shown that important t-norms can be combined with important shapes of fuzzy sets to obtain radial I-FISs. It was shown that the computation of radial implicative FISs is given by intersection of intervals and there was presented sufficient and necessary condition to radial I-FIS be coherent. Moreover, the class of radial implicative fuzzy inference systems exhibits the universal approximation property.

References

[1] Klir G.J., Yuan B. Fuzzy sets and Fuzzy logic - Theory and Applications, Prentice Hall, 1995

[2] Hájek P., Metamathematics of Fuzzy Logic, Kluwer Academic Publishers, 1998

Generalised Target Functions for Multilayer Neural Networks

doktorand:
MILAN RYDVAN

Ústav informatiky AV ČR
Pod vodárenskou věží 2,
182 07 Praha 8

Czech Republic

rydvan@cs.cas.cz

školitel:
LADISLAV ANDREJ

Ústav informatiky AV ČR
Pod vodárenskou věží 2,
182 07 Praha 8

Czech Republic

andre@cs.cas.cz

obor studia:
I1 - Teoretická informatika

Abstrakt

This paper discusses application of alternative target functions on multilayer neural networks, comparing them with the commonly used square error function $E = (y - d)^2$. Genetic training is used to allow generalised target functions, possibly non-continuous and non-differentiable. The proposed ideas are tested on the problem of stock price prediction, using a model of profit as the target function.

1. Motivation

This work deals with the model of *multi-layer neural networks*. The model is widely known; the definition can be found for example in the original work [6]. Multi-layer neural networks employ supervised training, using a finite training set $T = \{(\vec{x}_i, \vec{d}_i)\}$ of pairs of input vectors and desired output vectors. The aim of training is to find such parameters of the network (weights, thresholds) that minimise a *target function* $E(d_{ij}, y_{ij})$, summed over all the output neurons and all the training patterns (y_{ij} stands for the actual output of the network; for simplification, we will omit the indices in the following text).

Rummelhart ([6]) proposed square error function $E = (y - d)^2$ as the target function and it is widely used till today. Its advantage include the fact is that it is simple and natural. The fact that it penalises the distance between the desired and the actual output makes it applicable, with a better or worse success, on all kinds of problems without requiring a special knowledge about the character of the problem.

This article however studies the case when we have a specific knowledge about the problem and wish the network to learn this knowledge. This can be, for example, the case of economic (e.g. stock-price) predictions. There is a difference between the following two cases:

1. The system predicts a (significant) price growth g and the real price growth is 0.

2. The system predicts a zero price growth and the real price growth is g.

A broker following the advice of the system buys the stock in the first case, expecting a price growth and a profit. He does not achieve a it, but he loses only the transaction costs, as the stock he purchased has retained its value. In the second case, the trader does nothing, because stagnation is expected. His profit is zero, but his suffers an opportunity loss — he could have made profit by buying the stock.

This knowledge cannot be taught using the standard error function $E = (y - d)^2$, because E depends only on the difference between the actual and the desired output, which is equal in the two cases. We need a more general target function that would be able to incorporate this kind of problem-specific knowledge.

2. Alternative Target Functions

The standard training algorithm for multilayer neural networks is the Back-Propagation algorithm ([6]). This is a gradient-descent algorithm, requiring the target function E to be differentiable in y. While the standard error function $E = (y - d)^2$ fulfils this condition, the desired target function that incorporate our special knowledge may not fulfil it.

A possible way how to deal with this problem is to approximate the desired target function by a differentiable function. Works [4] and [5] use this solution. The first of them approximates the desired target function by a bi-quadratic function of the form $A_2d^2 + A_1d + B_2y^2 + B_1y + Cdy + D$; the second one uses a modular approach, training a special neural network, called relief error network, to approximate the desired target function. Both of these solutions achieved good results, improving both the networks' performance and generalisation ability, compared to the standard error function. Yet they are mere approximations; the next logical step was to apply the desired target function directly.

An example of such target function follows — it is a model of the daily profit a broker would gain if he followed the advice of a stock price prediction system, in our case of a neural network. In this case, the actual output y is the price growth/fall predicted by our system. The desired output d is the real growth/fall achieved on the stock exchange. To make the model more realistic, we will take into account the transaction costs c. All of y, d and c are expressed as a ratio of the price. The profit P is then modelled as follows:

$P = d - c$ iff $y > c$, (price rise prediction, recommendation to buy)

$P = -d - c$ iff $y < -c$, (price fall prediction, recommendation to sell)

$P = 0$ otherwise, (stagnation or small change prediction, no action recommended)

and the target function we employ is simply $-P$ (we want to maximise the profit, while the target function is being minimised).

Note that the model, though rather simple, reflects the behaviour of the real system described in the previous section — for $g > c$, $E(0, g) \neq E(g, 0)$. The first case leads to the loss of the transaction costs; the second one causes no factual loss, but represents the opportunity loss.

Note that the target function is not differentiable in y; it is even discontinuous in points, where $y = c$ or $y = -c$. We thus cannot apply the gradient descent BP-algorithm to teach the network using the profit-based target function.

3. Genetic Training

We will train our network using genetic algorithms, in order to be able to use the target function $-P$. Genetic algorithms (see for example [3] for more detailed information) perform distributed cooperating search in the solution space. Each prospective solution is coded in the form of a chromosome, a string of one-bit, two-bit or real values. Each chromosome is assigned a *fitness*, reflecting how suitable the corresponding solution is. The GA maximises the fitness using genetic operators on a population of chromosomes. *Selection* ensures the overall improvement of fitness, *crossover* combines schemes in existing individuals in order to create new patterns in new individuals and *mutation* makes random modifications, helping the system to produce brand new prospective solutions and avoid local minima.

When training neural networks using GAs, the chromosome can consist of real-valued genes, each representing a single parameter of the network — a weight or a threshold. The fitness of such chromosome-network is the negative value (because GAs maximise fitness) of the target function applied on the network and the training set, divided by the size of the training set, i.e. the average value reached per training pattern.

Genetic training of neural networks usually have several drawbacks compared to gradient methods — it tends to be slower and its results are poorer. On the other hand, it does not suffer from the local minima problem so much. However, the main benefit of genetic training for us is that it allows usage of general target functions. Our hypothesis is that the use of these target functions will balance (and hopefully outperform) the drawbacks of genetic training.

4. Stock Price Prediction

We have compared the performance of the proposed approach with that of using the standard error function on the problem of stock price prediction, using the profit-based target function described in Section 2. The aim was to predict the stock price change in the following trading day, knowing a history of (five) previous price changes and additional information about the trading, such as the volume of trade, the position of the latest known price in the long-term history, the supply/demand ratio etc. The raw data from the stock exchange required extensive pre-processing, from tranforming absolute prices on price changes to application of Principal Component Analysis (PCA) (see e.g. [2]).

We used Matlab as the platform for programming the experiments. In order to implement genetic training, we have interconnected Matlab's neural network toolbox with a GA toolbox developed by Houck, Joines and Kay ([1]). Series of experiments were carried out, in order to determine and tune the parameters of the tested methods. We used the same architecture for both the standard error function and the profit-based target function, in order to keep the conditions as similar as possible. The architecture was 9-15-1, which means that the network had 150 weights and 16 thresholds and was thus represented by a real-value chromosome with 166 genes.

For the genetic training we achieved the best results with a population of 1000 chromosomes, using the normalised geometric ranking, simple one-point crossover and uniform mutation as the genetic operators. Using normalised geometric ranking, the probability of selecting the i-th individual from the population equals

$$P_i = \frac{q}{1-(1-q)^P}(1-q)^{r-1},$$

where r is the rank of the i-th individual according to the fitness, q is the probability of selecting the best individual and P is the population size. The parameter q was set to 0.08. Simple crossover just randomly selects a point in the chromosomes of the parents and creates the offspring by exchanging the parents' genes located rightwards of the position. Uniform mutation randomly selects one gene and assigns it a uniform random number from the permitted space of values (interval $(-10, 10)$ was used as the permitted space for the gene values).[1] The probabilities of crossover and mutation were 0.5 and 0.2, respectively. The evolution continued until the best individual reached the fitness of 0.43 or for 200 generations.

Generalisation: In order to estimate and compare generalisation abilities of the methods, we divided the known data into a *training set*, which was used during the training period, and the *test set*, unseen by the networks during training and used for measuring their performance on unknown data. The training set contained 75% of the data; the test set contained the remaining 25%. We propose *randomisation* of the training data as a method for improving generalisation. This method adds small random noise to the input component of each training pattern during each run of the training process (or evolutionary process, respectively). The aim is to prevent overfitting of the network and to teach the network that similar inputs

---

[1] For detailed definition of the mentioned genetic operators, see [1].

should produce similar outputs. We have tested the method for BP-training earlier and it really improved generalisation. The question was whether it would be efficient also when applied on the genetic training, where the role of the training set is slightly different — it is in fact a part of the fitness function.

We compared the proposed methods by carrying out 200 experiments. During each of them, three networks were trained - one using the standard error function, one using the profit-based target function with randomisation and one using the profit-based target function without randomisation. The table below contains the averaged results both on the training set and the test set. The division into the training/test set was carried out randomly for each of the 200 experiments.

Target function	Set	Square error	Direction correctness	Profit
Standard	Train	0.033	81.8%	0.474%
	Test	0.050	73.6%	0.164%
Profit-randomised	Train	0.182	78.4%	0.403%
	Test	0.221	73.9%	0.196%
Profit-pure	Train	0.178	79.5%	0.433%
	Test	0.220	73.3%	0.188%

Tabulka 2: Comparison of performance of the standard error function and the proposed target function on the problem of stock price prediction, separately for the training set and the test set. The first couple of rows contains the results for networks trained by BP-training algorithm and applying the standard error function. The second and third couple show the results of networks trained genetically and using the profit-based target function. Rows 3-4 contain the results when applying training data randomisation; rows 5-6 use pure training data. Several measures of success are presented - summed-square error, direction correctness (the percentage of correct prediction of price rise/decrease) and the average daily profit model defined in Section 2

Several facts are worth mentioning as regards the results. The profits all the models achieved, even on the test set, are quite high. A daily net profit, including the transaction costs, between 0.15% and 0.20% is not bad at all, even though we must be aware of the fact that it is just a simple model, not modelling e.g. demand/offer excesses etc. The fact that the networks are able to predict correctly whether the stock will rise or fall in almost three cases out of four is also quite good, despite of the fact that the data include about 10% of cases with no change at all, where each of the predictions rise/fall is correct.

We can see that the profit-based target functions show higher profit than the standard error function on the test data, while their performance on the training data is worse. This could suggest that the generalisation ability of the method we propose is higher on this problem. A similar result, though less distinct, give the profit-trained networks with and without randomisation. This supports the assumption that randomisation improves generalisation ability also in the case of genetic training.

The direction correctness results all three methods achieved on the test set are roughly similar. However, the networks trained by the standard error function needed visibly higher correctness on the training set to achieve this, which would once again suggest that the generalisation of the networks trained on the profit was higher.

In the case of summed square errors, the standard error function distinctly beats the profit-based target function, no matter if randomisation is employed or not. That is however rather natural — the alternative target function did not teach the networks to minimise the difference between the desired output and the actual one, but to maximise the (modelled) profit. The networks have apparently learnt that it pays off to be courageous — they propose trading in almost every trading day (more than 98% of cases), while the standard error function sticks to the training data, proposing trading in slightly more than 70% of cases; the share of trading days when buying or selling was actually advantageous according to the real data was 73%. We can say that the networks trained using the profit-based target function have proven that to undergo the danger of loss of the transaction costs is under this model more advantageous than to risk the opportunity loss.

The price we pay for the better profit and generalisation is speed. Training one network genetically with the parameters we found as the best takes many times longer than training the same network using the

BP-training algorithm and the standard error function. The reason is that the size of the population as well as the number of generations is high. Further fine-tuning of the parameters of the evolutionary process could probably somewhat speed up the training process, but it will hardly get close to the speed achieved by the standard training. These results once again support the rule that genetic teaching is slower than gradient methods.

5. Conclusion

This article proposed a method of applying general types of target functions for training multilayer neural networks. Training the network using genetic algorithms allowed us to use non-continuous and non-differentiable target functions, which can be useful for teaching the network a specific knowledge we might have about the particular problem.

We have illustrated the approach on the problem of stock price prediction. A generalised target function based on a model of profit succeeded to teach the network a special knowledge — that a risk of unnecessary buy or sell order is more acceptable than a rise of an opportunity loss. The outcomes also suggest that randomisation of the training data somewhat improves the generalisation of neural networks, even if they are trained using genetic algorithms.

Further attention should be paid to fine-tuning the model of stock price prediction using profit-based target functions and genetic training, especially as regards the length of training, which is very high in comparison with standard training. A summary and comparison of the methods of alternative, generalised target functions proposed in this article and the previous ones is supposed to follow.

References

[1] Chris Houck, Jeff Joines, Mike Kay, *“A Genetic Algorithm for Function Optimization: A Matlab Implementation”*, NCSU-IE TR 95-09, 1995,
see `http://www.ie.ncsu.edu/mirage/#GAOT`.

[2] T. Masters, *“Advanced Algorithms for Neural Networks”*, John Wiley & Sons, 1995.

[3] M. Mitchell, *“An Introduction to Genetic Algorithms”*, MIT Press, 1997.

[4] M. Rydvan, *“Biquadratic Error Functions for the BP-networks”*, TR No 98/10, Charles University Prague, MFF.

[5] I. Mrázová, M.Rydvan, *“Generalised Error Function for BP-Network”*, TR No 99/1, Charles University Prague, MFF.

[6] D. E. Rummelhart, G. E. Hilton,R. J. Williams, *“Learning representations by backpropagating errors”*, Nature (323), (1986), pp. 533-536, MFF.

Použití lokálně omezeného kroku v metodách vnitřních bodů nelineárního programování

doktorand:
MGR. CTIRAD MATONOHA
ÚI AVČR, Pod vodárenskou věží 2, Praha 8

školitel:
DOC. RNDR. JAN ZÍTKO, CSC.
Sokolovská 83, Praha 2

obor studia:
M6 - vědecko-technické výpočty

Abstrakt

Metody vnitřních bodů jsou efektivním nástrojem pro řešení obecných problémů nelineárního programování, zejména tehdy, je-li úloha velká a strukturovaná. Abychom zaručili globální konvergenci algoritmu, je výhodné používat myšlenku metod s lokálně omezeným krokem ve spojení s vícekriteriálním rozhodováním při výběru nového přiblížení. V tomto příspěvku je zformulována úloha nalezení lokálně omezeného kroku pro metody vnitřních bodů a ukázána iterační metoda, založená na předpodmíněné metodě sdružených gradientů, sloužící k jejímu řešení.

Compactness of various fuzzy logics

doktorand:
PETR CINTULA
Ústav informatiky, Akademie věd Českě Republiky, Pod vodárenskou
věží 2, 182 07 Prague 8
Cintula@cs.cas.cz

školitel:
PETR HÁJEK
Ústav informatiky, Akademie věd Českě Republiky, Pod vodárenskou
věží 2, 182 07 Prague 8
hajek@cs.cas.cz

obor studia:
Matematické inženýrství

Abstrakt

Compactness is an important property of classical logic. It states that simultaneous satisfiability of an infinite set of formulas is equivalent to the satisfiability of all its finite subsets. In fuzzy logics, we have different degrees of satisfiability, hence the questions of compactness become more complicated. Here we give an overview of recent results on compactness and we extend them to various fuzzy logics.

This paper is a joint work with Mirko Navara from the Czech Technical University (cf. [6]).

1. Introduction

Dealing with vague or uncertain information, we often replace the two classical truth degrees $0, 1$ by a continuous scale $[0, 1]$. Then we have also various possibilities how to define the interpretation of the basic logical connectives. Depending on this choice, we obtain various fuzzy logics. We refer to [2, 9, 10] for a detailed description of the most frequent approaches.

In classical logic, a set of formulas is satisfiable if there is an evaluation which evaluates them all by 1. The compactness theorem holds for classical logic: A set of formulas is satisfiable if and only if all its finite subsets are satisfiable. It is natural to ask in which fuzzy logics analogues of this theorem hold. As the first step, we have to generalize the notion of satisfiability. We may again require all formulas to be evaluated by 1, but this is not the only possibility. Sometimes other alternatives are well-motivated, hence, following [2], we work with K-satisfiability, where K can be an arbitrary subset of $[0, 1]$. We say that a set of formulas is K-satisfiable if there is an evaluation which evaluates them all by values in K. (In particular, we get the former case if we choose $K = \{1\}$.) Using K-satisfiability, we may formulate various types of compactness. Here we present new results about validity of compactness in the most frequently used fuzzy propositional logics.

In particular, we prove that product and Gödel logic do not satisfy the compactness property, but we present also partial positive results. Then we extend these observations to logics with Baaz $\triangle$ operator, logics with involutive negations, and $L\Pi$ logics which form a common extension of Lukasiewicz and product logic. We prove that most of these stronger logics do not satisfy the compactness property and even more—they are not K-compact for almost all forms of the set K.

2. Basic fuzzy logical operations

Following [9, 10], we deal here with logics which have the real interval $[0,1]$ as the set of truth values and the following basic connectives:

- nullary false statement $\mathbf{0}$, interpreted by 0,
- binary conjunction $\wedge$, interpreted by a *t-norm* $T\colon [0,1]^2 \to [0,1]$, i.e., a commutative, associative, non-decreasing operation with a neutral element 1,
- binary implication $\to$, interpreted by the *residuum* R of T, i.e.,

$$R(x,y) = \sup\,\{z \in [0,1] : T(x,z) \le y\}\ .$$

We start from a nonempty countable set A of *atomic symbols* (*atoms*) and we define the class of *well-formed formulas* in a fuzzy logic (*formulas* for short) in the standard way. Each function which assigns truth values to atomic formulas is uniquely extended (using the interpretations of connectives) to an *evaluation* $e\colon \mathcal{F}_{\mathcal{P}} \to [0,1]$.

In this approach, the semantics of the logic is fully determined by the choice of the t-norm T. The three basic triangular norms lead to the following three main examples of fuzzy logics:

- For the minimum $T_{\mathbf{G}}(x,y) = \min(x,y)$ we obtain *Gödel logic* G.
- For the triangular norm $T_{\mathbf{L}}(x,y) = \max(x+y-1,0)$ we obtain *Lukasiewicz logic* L.
- For the algebraic product $T_{\mathbf{P}}(x,y) = x \cdot y$ we obtain *product logic* Π.

The respective residua in these logics are:

$$\begin{aligned}
R_{\mathbf{G}}(x,y) &= \begin{cases} 1 & \text{if } x \le y\ , \\ y & \text{otherwise}\ , \end{cases} \\
R_{\mathbf{L}}(x,y) &= \begin{cases} 1 & \text{if } x \le y\ , \\ 1-x+y & \text{otherwise}\ . \end{cases} \\
R_{\mathbf{P}}(x,y) &= \begin{cases} 1 & \text{if } x \le y\ , \\ \dfrac{y}{x} & \text{otherwise}\ . \end{cases}
\end{aligned}$$

The residuum $R_{\mathbf{L}}$ is continuous, $R_{\mathbf{G}}$ is not continuous in the points (x,x), $0 \le x < 1$, and $R_{\mathbf{P}}$ has a discontinuity in $(0,0)$.

Using the basic logical connectives $\wedge, \to$ and $\mathbf{0}$, we can define derived logical connectives. Negation $\neg$ is defined as

$$\neg\varphi = \varphi \to \mathbf{0}\ .$$

Its interpretation is the fuzzy negation N given by

$$N(x) = R(x,0) = \sup\,\{z \in [0,1] : T(x,z) = 0\}\ .$$

In Lukasiewicz logic this leads to *standard negation* $N_{\mathbf{S}}(x) = 1-x$, in Gödel and product logic we obtain *Gödel negation*

$$N_{\mathbf{G}}(x) = \begin{cases} 1 & \text{if } x = 0\ , \\ 0 & \text{if } x > 0\ . \end{cases}$$

For additional information on these logics, we refer to [9, 10]. Their detailed study and the proofs of completeness can be found in [10, 11].

Furthermore we examine the compactness property for logics with the set of connectives extended by an additional unary connective $\triangle$ (0-1 *projector* or *Baaz delta*) with interpretation Δ defined by

$$\Delta(x) = \begin{cases} 1 & \text{if} \quad x = 1\ , \\ 0 & \text{if} \quad x < 1\ . \end{cases}$$

Taking the Gödel, Lukasiewicz, or product t-norm as the interpretation of conjunction, we obtain Gödel, Lukasiewicz, or product logic with $\triangle$ denoted by $\mathrm{G}_\triangle$, $\mathrm{L}_\triangle$, $\Pi_\triangle$, respectively.

Then we examine the compactness property for Gödel and product logics with the set of connectives extended by an additional unary connective $\sim$ (*involutive negation*) interpreted so that $e(\sim\varphi) = 1 - e(\varphi)$. We call these logics Gödel involutive and product involutive logics. We will denote them by $\mathrm{G}_\sim$ and $\Pi_\sim$. They were introduced by Esteva, Godo, Hájek and Navara in [7]. (For Lukasiewicz logic, this notion does not bring anything new as it already contains an involutive negation.)

Then we investigate the compactness of $\mathrm{L}\Pi$ and $\mathrm{L}\Pi\frac{1}{2}$ propositional logics. These logics were introduced by Esteva, Godo, and Montagna in [8]. Then they were studied mainly in [4] and [5]. They unify Lukasiewicz and product logics (and many others) and have many interesting properties (such as completeness), but we shall see that they lack the compactness property and are not K-compact for an almost arbitrary set K.

The $\mathrm{L}\Pi$ logic has three basic connectives: Lukasiewicz implication and product conjunction and product implication. Furthermore, it has all other connectives of product, Gödel and Lukasiewicz logics, including the $\triangle$ connective.

$\mathrm{L}\Pi$ *logic* has the same standard semantics as product involutive logic. And the $\mathrm{L}\Pi\frac{1}{2}$ *logic* is $\mathrm{L}\Pi$ logic with an additional nullary connective **c** (a *non-trivial constant statement*) interpreted by an arbitrary element c from the open interval $(0, 1)$.

3. Satisfiability and compactness property

We present an analogue of the notion of satisfiability in fuzzy logics. It is natural to admit various degrees of simultaneous satisfiability of a set of formulas.

Definition 1 *For a set Γ of formulas in a fuzzy logic and $K \subseteq [0, 1]$, we say that Γ is K-*satisfiable *if there exists an evaluation e such that $e(\varphi) \in K$ for all $\varphi \in \Gamma$. The set Γ is said to be* finitely K-satisfiable *if each finite subset of Γ is K-satisfiable. Formula φ is called K-satisfiable if the set $\{\varphi\}$ is K-satisfiable.*

K-satisfiability obviously implies finite K-satisfiability. The reverse implication holds in classical logic, as well as in some fuzzy logics. This property is called compactness of a logic.

Definition 2 *We say that a logic is K-*compact *if K-satisfiability is equivalent to finite K-satisfiability. A logic satisfies the* compactness property *if it is K-compact for each* closed *subset K of $[0, 1]$.*

In the latter definition, it is necessary to consider only sets K which are closed (hence also compact), as we shall show in the next section. The following two observations will simplify our study:

At first observe that if we extend the set of connectives of a logic, L, then the resulting logic, L', becomes "less compact", i.e., if L' is K-compact (resp. has the compactness property) then L is K-compact (resp. has the compactness property). This is obvious since each counterexample to K-compactness in L is also a counterexample to K-compactness in L'.

Then observe that if $\{0, 1\} \subseteq K$, then for any formula φ there is an evaluation e such that $e(\varphi) \in K$ (it suffices to evaluate all atomic symbols by 0, then the evaluation of each each formula becomes either 0 or 1). Thus we will restrict ourselves to sets K not containing 0 and 1 simultaneously. We will also observe that in various logics K-compactness depends on presence 1 in K. Therefore the following classes of subsets of $[0, 1]$ will be important in the study of satisfiability and compactness:

Definition 3 *A nonempty subset K of $[0, 1]$ is of type C if $0 \notin K$ or $1 \notin K$. Furthermore, if K is of type C we define other type C_1 if $1 \in K$.*

4. The results

The following tables summarize our results. The first column indicates the types of logics, the second shows whether the logic enjoys the compactness property. The third column deals with K-compactness property for the sets of type C and the last one deals with the remaining sets. The possible elements of these tables are

All	logic is K-compact for all sets of this type
None	logic is K-compact for no set of this type
C_1	logic is K-compact at least for all sets of type C_1
Compact	logic is K-compact exactly for all compact subsets
Dense	logic is K-compact at most for those sets with a dense subset
Q	logic is K-compact at most for those sets containing all rationals

We also present a diagram of logics studied in this paper ordered by the richness of their sets of connectives. Notice that the results depends on the cardinality of the set of atoms for G, $\mathrm{G}_\triangle$, $\mathrm{G}_\sim$ logics.

Tabulka 3: K-compactness for finite and infinite sets of atoms

K-compactness for finite sets of atoms

Logic	Compactness	C	non C
L	Yes	Compact	All
Π	No	C_1	All
G, $\mathrm{G}_\triangle$, $\mathrm{G}_\sim$	Yes	All	All
$\mathrm{L}_\triangle$, $\Pi_\triangle$, $\Pi_\sim$, LΠ	No	None	All
L$\Pi\frac{1}{2}$	No	None	Q

K-compactness for infinite sets of atoms

Logic	Compactness	C	non C
L	Yes	Compact	All
Π, G	No	C_1	All
$\mathrm{G}_\triangle$, $\mathrm{G}_\sim$	No	Dense	All
$\mathrm{L}_\triangle$, $\Pi_\triangle$, $\Pi_\sim$, LΠ	No	None	All
L$\Pi\frac{1}{2}$	No	None	Q

$$\begin{tikzcd}
\mathrm{L} \arrow[d] & \Pi \arrow[d] & \mathrm{G} \arrow[d] \\
\mathrm{L}_\triangle \arrow[ddr] & \Pi_\triangle \arrow[d] & \mathrm{G}_\triangle \arrow[d] \\
 & \Pi_\sim \arrow[d] & \mathrm{G}_\sim \arrow[dl] \\
 & \mathrm{L}\Pi \arrow[d] & \\
 & \mathrm{L}\Pi\frac{1}{2} &
\end{tikzcd}$$

5. Conclusion

We studied various types of fuzzy logics—Lukasiewicz, Gödel, and product logic, logics with $\triangle$, logics with involutive negation, LΠ logic and L$\Pi\frac{1}{2}$ logic. We have found out that the analogue of classical compactness property holds for Lukasiewicz logic. In some other logics at least partial positive results were obtained. In general, enriching the set of connectives we weaken the compactness of the logic. The L$\Pi\frac{1}{2}$ logic represents the extreme case as it does not satisfy K-compactness for an almost arbitrary subset $K \subseteq [0,1]$.

There are several open questions. Is Gödel or product logic K-compact for some sets K of type C which are not of type C_1? (The answer is positive for the set $\{0\}$.) Is Gödel logic with $\triangle$ or Gödel involutive logic K-compact for all sets K containing a dense subset? Is L$\Pi\frac{1}{2}$ logic K-compact for all sets K containing all rationals from $[0,1]$? How many atoms are really needed to prove our theorems? (We know that all results presented here can be proven with three atoms, some properties with two atoms, and sometimes only one atom is sufficient.)

On the other hand, these are the only questions unanswered in our paper—for all the other combinations of a logic and a subset $K \subseteq [0,1]$ the problem of K-compactness has been solved here.

References

[1] M. Baaz and R. Zach: *Compact propositional Gödel logics.* In Proc. 28th Int. Symp. on Multiple Valued Logic. IEEE Computer Society Press, Los Alamitos, CA, 1998.

[2] D. Butnariu, E.P. Klement, S. Zafrany: *On triangular norm-based propositional fuzzy logics.* Fuzzy Sets and Systems 69:241–255, 1995.

[3] R. Cignoli, I.M.L. D'Ottaviano, D. Mundici: *Algebraic Foundations of Many-Valued Reasoning.* Kluwer, Dordrecht, 1999.

[4] P. Cintula: *The* $\mathrm{L\Pi}$ *and* $\mathrm{L\Pi}\frac{1}{2}$ *propositional and predicate logics.* Fuzzy Sets and Systems 124/3:21–34, 2001.

[5] P. Cintula: *An alternative approach to the* $\mathrm{L\Pi}$ *logic.* Neural Network World 11:561–572, 2001.

[6] P. Cintula, M. Navara: *Compactness of various fuzzy logics.* Submitted to Fuzzy Sets and Systems.

[7] F. Esteva, L. Godo, P. Hájek, M. Navara: *Residuated fuzzy logics with an involutive negation.* Arch. Math. Logic 39:103–124, 2000.

[8] F. Esteva, L. Godo, F. Montagna: *The* $\mathrm{L\Pi}$ *and* $\mathrm{L\Pi}\frac{1}{2}$ *logics: two complete fuzzy systems joining Lukasiewicz and product logics.* Archive of Math. Logic 40:39–67, 2001.

[9] S. Gottwald: *A Treatise on Many-Valued Logics.* Studies in Logic and Computation. Research Studies Press, Baldock, 2001.

[10] P. Hájek: *Metamathematics of Fuzzy Logic.* Kluwer, Dordrecht, 1998.

[11] P. Hájek, L. Godo, F. Esteva: *A complete many-valued logic with product conjunction.* Arch. Math. Logic 35:191–208, 1996.

[12] P. Hájek: *Personal communication.* August 2000.

[13] J. Hekrdla, E.P. Klement, M. Navara: *Two approaches to fuzzy propositional logics.* Multiple-valued Logic, accepted.

[14] E.P. Klement, M. Navara: *A survey of different triangular norm-based fuzzy logics.* Fuzzy Sets and Systems 101:241–251, 1999.

[15] M. Navara: *Satisfiability in fuzzy logics.* Neural Network World 10:845–858, 2000.

[16] M. Navara, U. Bodenhofer: *Compactness of fuzzy logics.* In Proc. 2nd Int. ICSC Congress on Computational Intelligence: Methods and Applications (CIMA 2001), Bangor, Wales, UK, 654–657, 2001.

Complexity of the propositional tautology problem for t-norm logics

doktorand:
ZUZANA HANIKOVÁ
Ústav informatiky AV ČR, 182 07 Praha 8
zuzana@cs.cas.cz

školitel:
PROF. RNDR. PETR HÁJEK, DRSC.
Ústav informatiky AV ČR, 182 07 Praha 8
hajek@cs.cas.cz

obor studia:
Matematická logika

Abstrakt

The propositional tautology problem for any logic given by a continuous t-norm is coNP complete.

1. Foreword

This paper is a preliminary and incomplete version of [5]; some proofs have been omitted for the sake of brevity, and for the same reason, it is impossible to give a full introduction to the topic. A comprehensive treatment of the approach we follow will be found, e. g., in [4].

2. Introduction

t-algebras (or *standard* algebras) are a frequently used class of algebras of truth values for many-valued logics. Each t-algebra is determined in a unique manner by a continuous t-norm on $[0, 1]$ (hence the term).

It is known that the propositional logic BL, investigated in [4], is complete w. r. t. the tautologies of all t-algebras. This result comes from [3]. It is also known that some t-algebras are BL-generic; [1] gives a characterization of these. Moreover, [2] shows the tautologies of all t-algebras (or equivalently, the propositional BL) to be coNP complete. Thus the complexity of the propositional tautology problem is settled for BL-generic t-algebras.

Three important schematic extensions of BL, namely the logic of Lukasiewicz, of Gödel, and the product logic, have been investigated thoroughly, and their propositional tautologies have also been proved to be coNP complete ([4] gives further references; in particular, the coNP completeness of propositional L-tautologies comes from [7]).

The aim of this paper is to adjust the algorithm presented in [2] and prove the following claim:

Theorem 2.1 *For any t-algebra, the propositional tautology problem is in coNP.*

Once established, this theorem settles the question of complexity of propositional tautologies for an arbitrary t-algebra, in combination with an earlier result:

Theorem 2.2 *For any t-algebra, the propositional tautology problem is coNP hard.*

This comes from [2] for t-algebras starting with an Ł (proved via reduction of propositional Ł-tautologies, prefixing a negation to each propositional variable) and from [4] for t-algebras not starting with an Ł (proved via reduction of propositional Boolean tautologies, prefixing a double negation to each propositional variable).

Throughout the paper we use heavily the Mostert-Shields decomposition theorem for continuous t-norms, originating in [6], and employ some rather informal notation based on it. The statement of the theorem is that the "backbone" of any continuous t-norm is formed by a countable closed subset I of $[0, 1]$ (we use the term 'cutpoints' for the elements of I), and on each of the closures of the open intervals which form the complement of I, the t-norm is isomorphic to either Lukasiewicz, Gödel, or product t-norm (on $[0, 1]$). For this reason each t-algebra is an ordinal sum of copies of Lukasiewicz, Gödel, and product algebras, which we habitually call *segments* and denote with symbols Ł, G, and Π. We stress that each copy of Gödel counts as one segment, thus, e. g., $[0, 1]_{\text{Ł}\oplus\text{G}\oplus\Pi}$ is a t-algebra with three segments, namely a sum of a copy of the Lukasiewicz algebra, a copy of the Gödel algebra and a copy of the product algebra; the type of the sum is Ł⊕G⊕Π. We disregard the exact positioning of the set I within $[0, 1]$.

3. Finite ordinal sums

[2] gives an NP algorithm recognizing BL-couterexamples. In fact, it shows more than that: by a trivial modification, for any finite sum of Ł-segments only, the set of its non-tautologies is in NP.

To prove our claim for finite ordinal sums, we generalize the algorithm of [2] to recognize non-tautologies (i.e., formulas for which there is a counterexample evaluation) in an arbitrary fixed t-algebra which is a finite ordinal sum. Fix A as such a t-algebra, and let n be its cardinality (i. e., the number of segments in the sum). For a propositional formula φ, let $m = 2|\varphi|$, where $|\varphi|$ is the number of occurrences of propositional variables in φ (so m is an upper bound on the number of the subformulas of φ).

What follows is, we claim, an NP algorithm which for an input formula φ decides whether there is an evaluation e in A s. t. $e(\varphi) < 1$. It is a modification of the algorithm of [2]: we drop, for the moment, the step which guesses the cardinality of the sum, since A is fixed. The generalization, which adds a check for G-segments and Π-segments, comes in the `checkInternal()` step, which will be discussed subsequently.

`// algorithm for finite sum A`

{

`cutpointVariables()` Introduce variables $z_0 < \cdots < z_n$ for the cutpoints of A (thus z_0 is intended for 0 and z_n is intended for 1).

`intervalVariables()` For each $i = 0, \ldots, n-1$ introduce variables $z_i = y_{i0} < y_{i1} < \cdots < y_{im} = z_{i+1}$ We call these *the variables belonging to* i. By convention, two variables which are equal are interchangeable in all contexts (thus also z_i, z_{i+1} belong to i, $i = 0, \ldots, n-1$).

Since the values of all subformulas could belong to a single segment and each subformula could evaluate to a different element of the segment, it is vital to have enough variables belonging to each $i = 0, \ldots, n-1$. Note that this is so, since each i contains $m + 1 = 2|\varphi| + 1$ variables, of which two represent the cutpoints, while the total number of subformulas is at most $2|\varphi| - 1$; so the number of variables is sufficient for any type of evaluation.

Set $C = \{z_0, \ldots, z_n\} \cup \{y_{ij} | i = 0, \ldots, n-1,\ j = 0, \ldots, m\}$.

`guessAssignment()` Guess an assignment f of variables in C to subformulas of φ (an "evaluation" of subformulas of φ with variables in C), s. t. $f(\varphi)$ is not z_n.

`checkExternal()` Check external soundness of f: if $u, v \in C$, $f(\varphi_1) = u$, $f(\varphi_2) = v$, then

- if $\varphi_1 \& \varphi_2$ is a subformula of φ and, for some i, $u \leq z_i \leq v$, then $f(\varphi_1 \& \varphi_2) = f(\varphi_1) = u$;
- if $\varphi_1 \to \varphi_2$ is a subformula of φ and $u \leq v$ then $f(\varphi_1 \to \varphi_2) = z_n$;
- if $\varphi_2 \to \varphi_1$ is a subformula of φ and for some i, $u < z_i \leq v$, then $f(\varphi_2 \to \varphi_1) = f(\varphi_1) = u$.

`checkInternal()` Check internal soundness of f for each segment. Consider the i-th segment. For each subformula $\varphi_1 \& \varphi_2$ s. t. $f(\varphi_1) = y_{ij}$ and $f(\varphi_2) = y_{ik}$, if $f(\varphi_1 \& \varphi_2) = y_{il}$ (that is, all three variables involved belong to i), put down an equation $y_{ij} * y_{ik} = y_{il}$, and for each subformula $\varphi_1 \to \varphi_2$ s. t. $f(\varphi_1) = y_{ij}$ and $f(\varphi_2) = y_{ik}$, where $j > k$, if $f(\varphi_1 \to \varphi_2) = y_{il}$, put down an equation $y_{ij} \Rightarrow y_{ik} = y_{il}$. Check whether these equations, together with the sharp inequalities $y_{i0} < \cdots < y_{im}$, have a solution in the i-th segment of the sum, s. t. y_{i0} and y_{im} evaluate to the lower and upper cutpoint of the segment, respectively.

}

The last check in the above algorithm is the same as finding a solution in the Lukasiewicz, Gödel, or product t-algebra (depending on the type of the i-th segment in A), s. t. y_{i0} and y_{im} are evaluated by 0 and 1, respectively. [2] presents an NP algorithm which performs this check for L-segments, so it remains to show how to perform it for G-segments and for Π-segments.

Observation 1 *The solvability of the above system of equations and sharp inequalities in G can be checked in linear time (w. r. t. $|\varphi|$).*

For the product t-algebra we use the following lemma.

Lemma 3.1 *The abovementioned system of equations and sharp inequalities is solvable in* Π *iff it has a solution in an algebra of type* L⊕L *such that* y_{i0} *is* $0_{\mathrm{L}\oplus\mathrm{L}}$ *and* $y_{i1}, \ldots, y_{im}$ *are evaluated in* $(h, 1]$*, where* h *is the non-extremal cutpoint.*

Proof: Follows from the isomorphism of the cut product algebra with L. An $m + 1$-tuple $0 = a_0 < \cdots < a_m = 1$ is a solution in Π iff, introducing a cut c so that $a_0 < c < a_1$ and using an isomorphism g to map $a_1, \ldots, a_m$ into $(h, 1]$ in L⊕L, $0_{\mathrm{L}\oplus\mathrm{L}}$ together with $g(a_1), \ldots, g(a_m)$ form a solution in L⊕L. QED

Thus, to check solvability in the product t-algebra, we first eliminate all equations involving y_{i0}; the soundness of any such equation can be, and indeed has been in part, checked "externally"; for the remaining cases, check, for any u, v belonging to i, that if $u * v = y_{i0}$ then either u or v is y_{i0}, that if $u \Rightarrow v = y_{i0}$ (and $u > v$) then v is y_{i0}, and that $u \Rightarrow y_{i0} = y_{i0}$. Then we consider the remaining equations and sharp inequalities in L, introducing a new inequality $0 < y_{i1}$, and check solvability of this system of equations and inequalities using the NP algorithm for solvability in L, referred to in [2].

Finally, it is obvious from the construction of the algorithm that the output is 'yes' (on at least one branch) iff the formula φ has a counterexample evaluation in A, i. e., is not an A-tautology. Thus the set of A-tautologies is in coNP.

4. Infinite ordinal sums

It is known ([1]) that a t-algebra is BL-generic iff it is an ordinal sum starting with an L and with infinitely many copies of L. Since the tautologies of BL are coNP complete, so are the tautologies of each BL-generic t-algebra.

Also, it is easy to see that t-algebras which are ordinal sums not starting with an Ł and having infinitely many copies of Ł are SBL-generic. To follow this observation, recall that a counterexample evaluation in Π can be locally embedded into Ł⊕Ł. Now let A be a t-algebra with infinitely many copies of Ł, not starting with an Ł. Assume φ is not an SBL-tautology, and let B be an SBL-algebra in which φ does not hold. We may assume that B is a finite sum of Ł's and Π's only (thus starting with a Π). Then the counterexample evaluation can be locally embedded in A, mapping the initial Π segment of B to any two Ł-segments of A (not necessarily adjacent), each of the following Ł-segments of B to arbitrary Ł segments of A, and each of the following Π-segments of B to any two Ł-segments of A, all in increasing order w. r. t. the ordering of the intervals in $[0, 1]$.

Theorem 4.1 *The propositional logic SBL is coNP complete.*

Proof: If φ is not an SBL-tautology, then it has a counterexample in a finite ordinal sum whose first element is not an Ł. Thus we may modify our algorithm by prefixing steps guessing the cardinality of the sum and its type. Let k be the number of propositional variables in φ.

`// algorithm for SBL`

{

`guessCardinality()` Pick at random a natural n, $0 < n \leq k + 1$.

Lemma 4.2 *Let k be the number of propositional variables in a formula φ. If φ has an evaluation $e(\varphi) < 1$ in any t-algebra, then it has an evaluation $e'(\varphi) < 1$ in a t-algebra which is an ordinal sum with cardinality at most $k + 1$.*[1]

`guessLayout()` Assign to each $i = 1, \ldots, n$ one of the symbols Ł, G, Π, signifying the type of the i-th segment of the sum, in such a way that the first symbol is not an Ł. We use the term 'constructed sum' and the symbol C to denote this finite sum.

`cutpointVariables()`

`intervalVariables()`

`guessAssignment()`

`checkExternal()`

`checkInternal()`

}

This modification is an NP algorithm recognizing SBL counterexamples, so the propositional tautology problem for the logic SBL is in coNP. QED

It remains to discuss the complexity of tautologies of an arbitrary infinite ordinal sum with only finitely many (possibly no) copies of Ł.

Fix such an algebra A, denote p the number of its Ł-segments, and define its representation S^A: a finite sequence of length $p + 1$, each element $S^A[i]$ determining the type of the subsum between two consecutive Ł-segments ($S^A[0]$ before the first Ł-segment and $S^A[p]$ after the last Ł-segment in A). $S^A[i], i = 0, \ldots, p$ is one of the following:

- $\emptyset$ if the subsum is void;

---

[1] This is just a variant of a similar result in [2].

- ∞ if there are infinitely many G-segments (thus there is an infinite alternating subsum of G's and Π's);
- (for finite number p_i of G-segments) a sequence $S^A[i]$ of length $p_i + 1$, determining the number of Π-segments between each two consecutive G-segments (also before the first and after the last G-segment). The j-th element of the sequence, $j = 0, \ldots, p_i$ is a natural number in the range $[0, \infty]$.

This is a handy finite representation of A. Note that using S^A, we may introduce indices for the segments of A in the following way:

- Any L-segment is uniquely determined by a natural number in the range $[1, p]$.
- A G-segment is either determined by a tuple of natural numbers $\langle i_1, i_2 \rangle, i_1 \in [0, p], i_2 \in [1, p_{i_1}]$, if it is the i_2-th G-segment after the i_1-th L-segment in A, where $S^A[i_1]$ is not ∞; or, if $S^A[i_1], i_1 = 0, \ldots, p$ is ∞, the G-segments in the i_1-th subsum may be for our purposes referred to by a tuple $\langle i_1, \mathtt{ANY} \rangle$.
- A Π-segment is either determined by a triple $\langle i_1, i_2, i_3 \rangle, i_1 \in [0, p], i_2 \in [0, p_{i_1}], i_3 \in N$, if it is the i_3-th Π-segment after the i_2-th G-segment after the i_1-th L-segment in A, where $S^A[i_1]$ is not ∞ and $S^A[i_1][i_2]$ is not ∞; or, if $S^A[i_1]$ is not ∞ but $S^A[i_1][i_2]$ is ∞, $i_1 = 0, \ldots, p, i_2 = 0, \ldots, p_{i_1}$, all the Π-segments in the subsum of Π's after the i_2-th G-segment after the i_1-th L-segment in A may be referred to by a triple $\langle i_1, i_2, \mathtt{ANY} \rangle$; or, if $S^A[i_1]$ is ∞, $i_1 = 0, \ldots, p$, all the Π-segments in the i_1-th subsum may be referred to by a tuple $\langle i_1, \mathtt{ANY} \rangle$.

We shall now present an NP algorithm recognizing counterexamples in A. As before, let the input formula φ be given, k be the number of its variables, and $m = 2|\varphi|$.

`// algorithm for infinite sum A`

{

`guessCardinality()` Pick at random a natural n, $0 < n \leq k + 1$.

`guessLayout()` Assign to each $i = 1, \ldots, n$ one of the symbols L, G, Π, signifying the type of the i-th segment of the sum.

We use the term 'constructed sum' and the symbol C to denote this finite sum.

`// from now on the algorithm works with` C

`checkEmbedding()` Check whether the constructed sum is $1 - 1$ embeddable into A (as a sequence of symbols into a sequence of symbols), in such a way that a potential initial L of the constructed sum is mapped to an initial L in A. It is vital that initial L remains initial in A, since otherwise a counterexample in the constructed sum need not be a counterexample in A.

`cutpointVariables()`

`intervalVariables()`

`guessAssignment()`

`checkExternal()`

`checkInternal()`

}

We discuss in more detail why the `checkEmbedding()` step does not violate the NP nature of the algorithm.

Lemma 4.3 *The embeddability of the constructed sum C into A can be checked by an NP algorithm (w. r. t. the length n of C).*

Proof: The (nondeterministic) algorithm constructs the embedding a by assigning to each segment of C an index of its image in A, using the abovedescribed indices.

Denote max the maximum of the numbers $p, p_0, \ldots, p_p, n$. This number is the maximum natural number that can occur in any index guessed by the algorithm. Note that this number is independent of the input C. (Although some Π-segments could have indices with arbitrarily high numbers (as the third element), we use n as an upper bound, since C has the cardinality n, thus a suitable embedding can be always found in an initial n-segment fragment of the infinite subsum.)

First the algorithm guesses an index for each segment of C: indices of L-segments are natural numbers; indices of G-segments are tuples, the first element of which is a natural number and the second element is a natural number or the symbol `ANY`; indices of Π-segments are either tuples, consisting of a natural number and the symbol `ANY`, or triples, the first and second element of which are natural numbers and the third element is a natural number or the symbol `ANY`. Any number occurring in any index must be within $[0, max]$.

Subsequently the algorithm performs two checks, to find out whether there are segments in A referred to by the indices (this is checked using S^A) and whether the assignment of indices is 1-1 and increasing (w. r. t. the ordering of segments in C and in A). Both these checks can be performed in polynomial time (the detailed proof is omitted), thus the algorithm is NP. QED

Again, it is clear that the output of the algorithm is 'yes' (on at least one branch) iff the formula φ is not an A-tautology, thus A-tautologies are in coNP.

5. Thanks

Partial support from the grant IAA 103 0004, Grant Agency of the Academy of Sciences, Czech Republic, is acknowledged.

6. PhD thesis

The author is a PhD student of mathematical logic at the Faculty of Mathematics and Physics, Charles University, since October 1999. The topic of her thesis is "Mathematical and metamathematical properties of fuzzy logic".

References

[1] P. Aglianò, F. Montagna, "Varieties of BL-algebras I: general properties", preprint, 2002.

[2] M. Baaz, P. Hájek, F. Montagna, H. Veith, "Complexity of t-tautologies", *Ann. Pure Appl. Logic 113*, No.1-3, pp. 2-11, 2001.

[3] R. Cignoli, F. Esteva, L. Godo, A. Torrens, "Basic Fuzzy Logic is the logic of continuous t-norms and their residua", *Soft Comput. 4*, pp. 106-112, 2000.

[4] P. Hájek, "Metamathematics of Fuzzy Logic", Kluwer Academic Publishers, Dordrecht, 1998.

[5] Z. Haniková, "A note on the complexity of propositional tautologies of individual t-algebras", to appear in NNW.

[6] P. S. Mostert, A. L. Shields, "On the structure of semigroups on a compact manifold with boundary", *Annals of Math. 65*, pp. 117-143, 1957.

[7] D. Mundici, "Satisfiability in many-valued sentential logic is NP-complete", *Theor. Comput. Sci. 52*, pp. 145-153, 1987.

Analýza dat z projektu MAGIC-telescope pomocí rozhodovacích stromů a lesů

doktorand:
EMIL KOTRČ
Ústav informatiky AV ČR, Pod Vodárenskou věží 2, Praha 8
kotrc@cs.cas.cz

školitel:
RNDR. PETR SAVICKÝ, CSC.
Ústav informatiky AV ČR, Pod Vodárenskou věží 2, Praha 8
savicky@cs.cas.cz

obor studia:
Matematické inženýrství

Abstrakt

V tomto příspěvku se budeme stručně věnovat problematice rozhodovacích stromů a lesů. Ukážeme, že tyto metody jsou poměrně zajímavé z hlediska řešení klasifikačních problémů, ke kterým jsou zejména využívány. V úvodu článku se nejdříve velice stručně seznámíme se základními známými metodami pro konstrukci rozhodovacích stromů a lesů a poté si ukážeme výsledky experimentů na datech z projektu MAGIC-telescope a z nich plynoucí závěry a podněty pro další práci.

1. Úvod

V této úvodní části si zavedeme některé důležité pojmy, se kterými budeme dále v článku pracovat. Rozhodovací stromy a lesy se často používají jako klasifikátory, tedy nástroje, které umí klasifikovat neznámé případy do různých tříd. Abychom mohli nějaký strom či les natrénovat potřebujeme k tomu *učící množinu*, což není nic jiného než množina vektorů (dat) se známou klasifikací, na základě níž takový strom dokážeme zkonstruovat. Učení s učící množinou nazýváme *učení s učitelem*.
Předpokládejme nyní, že data, která zkoumáme, jsou charakterizována r atributy (proměnnými) x_i. Tyto atributy se většinou uvažují buď numerické (reálné) nebo kategoriální (nabývající hodnot z nějaké konečné množiny). Pak vektor $\mathbf{x} = (x_1, \ldots, x_r)$ nechť je pro nás vektorem měřených hodnot. Označme si X jako množinu všech takových možných měření. Dále nechť množina $C = \{C_1, \ldots, C_k\}$ je pro nás množina k tříd, do nichž chceme případy z X klasifikovat. Abychom ovšem mohli sestrojit nějaký klasifikátor, budeme potřebovat množinu případů se známou klasifikací. Označme si proto množinu $L = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ jako naší učící množinu (přesněji multi-množinu, neboť připouštíme i násobný výskyt vektorů), kde vektorům $\mathbf{x}_i$ přidáme k r atributům speciální $r+1$ složku, která bude určovat třídu, do níž známý vektor náleží. Formálně můžeme zapsat, že $L \subset X \times C$ a každý vektor $\mathbf{x}_i \in L, i \in \{1, \ldots, N\}$ má tvar $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,r}, x_{i,r+1})$, kde první až r-tá složka jsou měřené veličiny (numerické či kategoriální) a $r+1$ indikuje třídu, tedy $x_{i,r+1} \in C$. Jinými slovy, jestliže vektor $\mathbf{x}_i$ náleží do třídy C_j, pak $x_{i,r+1} = C_j$.

Naším cílem je pak na základě této trénovací množiny L vytvořit klasifikátor, který by po předložení případu s neznámou klasifikací dokázal určit (či spíše odhadnout) třídu, do které případ patří. Klasifikátor vlastně není nic jiného než zobrazení $h : X \to C$, tedy $h(\mathbf{x}) = C_j$ značí, že neznámé $\mathbf{x} \in X$ klasifikátor h zařadil do třídy C_j. V tomto článku se zabýváme případem, kdy klasifikátor h je *rozhodovacím stromem*. Jestliže ke klasifikaci budeme používat více stromů než jeden, nazýváme takový klasifikátor *rozhodovacím lesem*. Tedy, máme-li množinu t stromů $\{h_1, \ldots, h_t\}$ a nějaké pravidlo pro kombinování predikcí jednotlivých stromů, je rozhodovací les opět zobrazení $h^* : X \to C$.

Abychom mohli klasifikátory nějakým způsobem porovnávat, potřebujeme ověřit jejich kvalitu. Za tímto účelem se většinou pracuje s takzvanou *testovací množinou*, která je definována podobně jako množina učící a obsahuje rovněž případy se známou klasifikací. Nechť $K = \{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$ je pro nás testovací množina (multi-množina). Na ní potom můžeme definovat chybu klasifikátoru h například jako

$$\varepsilon_K^h = \frac{|\{\mathbf{x} \in K | x_{r+1} \neq h(\mathbf{x})\}|}{|K|} \qquad (51)\mathrm{span}$$

což je relativní počet nesprávně klasifikovaných vektorů z množiny K. Tento vzorec lze ovšem zobecnit pro případ, kdy máme apriorní informaci o pravděpodobnosti tříd. Toto zobecnění je použito v sekci věnované experimentům. Nyní se již budeme věnovat stručnému výčtu a popisu známých metod.

2. Metody

Metod na konstrukci klasifikátorů ve formě rozhodovacích stromů a lesů je celá řada. Mezi nejznámější patří bezesporu *C4.5/C5.0* od J.R.Quinlana ([1]) a *CART* ([2]). Většinu experimentů provádíme právě s pomocí těchto dvou metod. Mezi další můžeme zařadit novou metodu na konstrukci rozhodovacích lesů *Random Forest* ([4]), kterou zkoumáme v současné době, a dále například metody *Quest* a *CRUISE* ([8], [6], [7]).
V tomto článku se nebudeme se zabývat detailním popisem těchto metod. Pouze poukážeme na základní vlastnosti a výhody či nevýhody. Podrobnosti lze najít v literatuře.

2.1. Konstrukce stromů

Vytváření rozhodovacích stromů je ve většině implementací založeno na rekurzivním dělení učící množiny. Metody *C4.5/C5.0* a *CART* jsou, co se týče růstu stromů do jisté míry podobné. Na základě entropie (míry neurčitosti) se provede výběr nejlepšího splitu na podmnožině učící množiny. Jakmile je vytvořen celý strom, přechází se ke druhé části zvané *prořezávání*, kdy dochází k odstraňování některých uzlů a k jejich nahrazení listem. Důvod pro tento krok je zlepšení generalizace prořezaného stromu.
Metoda *Random Forest* funguje trošku odlišně při konstrukci jednotlivých stromů v lese a je založena na výběru nejlepšího náhodného splitu (řezu). Přesněji, zvolí se náhodně F proměnných, na nichž se bude vyhledávat nejlepší řez a splitem se stane ten nejlepší. Možná se to zdá být překvapivé, ale výsledky ukazují, že jde o velmi dobrou metodu. Navíc, F můžeme položit rovno 1, což stírá veškeré složité vyhledávání nejlepšího splitu a výsledky jsou překvapivě dobré (viz [4]).
Námi nejčastěji používaná metoda je *C4.5/C5.0*, protože se poměrně dlouhou dobu zdála být nejlepší. Jejím nedostatkem však je skutečnost, že implementace *C5.0* je komerční, tudíž nemáme přístup ke zdrojovým textům, a proto není vždy zcela zřejmé jak program pracuje. V současnosti se začínáme zabývat metodou *Random Forest*, kterážto je volně šiřitelná, takže při jejím používání nevznikají potíže jako s *C5.0*, a jak jsme již zmiňovali výše, je to metoda poměrně nová a zajímavá. Metody *C4.5/C5.0*, *CART* a samozřejmě *Random Forest* lze použít kromě konstrukce rozhodovacích stromů rovněž pro vytváření rozhodovacích lesů, kterým je věnována další stručná kapitola.

2.2. Konstrukce lesů

Při konstrukci rozhodovacích lesů je potřeba ze stejné učící množiny vytvořit několik různých stromů. Metod existuje několik a my si tu uvedeme dvě základní – *bagging* a *boosting*.
Bagging spočívá v náhodném výběru s/bez (terminologie není ustálena) vracení z učící množiny a následném natrénování klasifikátoru. Na druhou stranu *boosting* je založen na vážení učících případů. V prvním kroku

mají všechny případy stejnou váhu a při každém dalším se tyto váhy mění v závislosti na chybě klasifikátoru. Případy, které jsou v předchozím kroku klasifikovány chybně, dostanou větší váhu a více ovlivní následující vznikající klasifikátor. V každém kroku jsou tedy případy z učící množiny vybírány dle rozdělení dané váhami v kroku předchozím utvořené. Více informací lze nalézt v článku [5], který nás přivedl na nové směry a myšlenky.
Jakmile máme vytvořen rozhodovací les, přichází na řadu další otázka. Jak klasifikovat pomocí více stromů ? To lze samozřejmě opět provádět více způsoby. Při *boosting*u se hlas každého stromu nějakým způsobem váží, narozdíl od *bagging*u, kde má většinou každý strom stejnou váhu, tedy klasifikování se provádí prostým většinovým hlasováním, což ovšem samozřejmě není podmínkou. V současnosti se právě zaměřujeme na zkoumání volby vhodných vah při hlasování více stromů (náš projekt *forestanalysis*).

3. Experimenty

V této kapitole se budeme krátce věnovat experimentům, které se stromy a lesy provádíme, a z nich plynoucím výsledkům. Experimenty provádíme na datech z projektu *MAGIC–Telescope*[1], která jsou zatím pouze simulovaná, neboť teleskop je stále ve výstavbě. Jedná se o data se dvanácti prediktory, přičemž učení se provádí na prvních deseti atributech, a se dvěmi třídami - signál a šum. Našim úkolem tedy je na základě učící množiny natrénovat klasifikátor, který bude schopen rozlišit tyto dvě třídy.
Formálně proto můžeme zapsat, že v našem případě $r = 12$ a $C = \{C_0, C_1\}$, kde C_0 představuje šum (hadrony) a C_1 signál, který představují gamma částice. Data dostáváme ve třech souborech - *GA*, *ON* a *OFF*. V *GA*-souboru jsou pouze částice gamma, tedy pro nás signál, v *OFF*-souboru pouze částice šumu a *ON*-soubor obsahuje směs gamma částic i šumu. Na základě těchto tří souborů se pak mohou provádět dva různé experimenty – *GA-OFF* a *ON-OFF*, přičemž pro trénování se použijí z každého souboru první dvě třetiny dat a zbytek pro testování. Testy *GA-OFF* jsme nejdříve prováděli s pomocí *C5.0* a výsledky dopadli v porovnání s ostatními metodami poměrně dobře. Poté, co jsme ale chtěli vyzkoušet i experiment *ON-OFF*, zjistili jsme, že boosting v *C5.0* je pro tento typ experimentu nepoužitelný, protože program po prvním kroku končí chybou. To je způsobeno právě *ON*-souborem, kterýžto obsahuje jak gamma částice, tak částice představující šum. Boosting se pak zastaví po prvním kroku kvůli velké klasifikační chybě. Abychom mohli v experimentech pokračovat, museli jsme růst více stromů zajistit jiným způsobem, pomocí našich nástrojů, k tomuto účelu vytvořených. Celý postup vlastně spočívá ve "vytažení" boostingu mimo *C5.0*. Postup je stručně popsán v tabulce 4 a v následujícím odstavci .
Nejdříve se učící a testovací množina rozdělí na třetiny. Dostaneme tak vlastně tři různé experimenty, ve kterých se každá z těchto třetin využije jako učící i jako trénovací. Tedy přesněji, ve fázi 1 se první třetina dat použije jako testovací a zbytek jako učící. Ve fázi 2 se na testování použije druhá třetina a v poslední fázi poslední třetina. Na konci jsou výsledky zprůměrovány. Postup uvedený v tabulce 4 se tedy použije pro každou fázi zvláště a stejným způsobem. Naše nástroje rovněž požadují zadat nějaké vstupní parametry - počet stromů, velikost učící množiny a rozsah penalizací za chybnou klasifikaci. Parametr počet stromů je zřejmý, prostě znamená kolik stromů se pro klasifikaci použije. Protože na vytváření více stromů používáme bagging, je nezbytné zadat, jak velký výběr (bez vracení) z trénovací množiny se použije pro učení každého stromu. Při konstrukci stromu umožňuje *C5.0* zadat penalizaci (*misclassification cost*) za chybné klasifikování, čili něco jako váhu chyby. Podobný mechanismus obsahuje i *CART*, kde se nastavují takzvané *priory*. Penalizace zadáváme ve formě geometrické posloupnosti, určené prvním a posledním členem a počtem prvků.

---

[1] http://hegra1.mppmu.mpg.de/MAGICWeb/

```
1. $p = min.penále, \ldots, max.penále$; přičemž $p$ bereme jako geometrickou posloupnost
2. $t = 1, \ldots, T$; kde $T$ je počet stromů
3. Náhodně vyber (bez vracení) z učící množiny podmnožinu o velikosti $c$
4. Pro danou podmnožinu a penále pomocí C5.0 zkonstruuj rozhodovací strom
5. Pokud $t \leq$ počet stromů, pokračuj krokem 2
6. Pokud $p \leq max.penále$, pokračuj krokem 1
   V této fázi máme zkonstruováno celkem $T*$ počet penalizací stromů
7. Vezmi všech $T$ stromů pro danou penalizaci $p$ jako jeden les a spočti chybu
```

Tabulka 4: Postup při experimentech s využitím C5.0

Abychom mohli metody porovnávat, potřebujeme opět nějaké vyjádření chyby. Zaveďme si proto veličinu

$$p(i,j) = |\{\mathbf{x} \in K | h(\mathbf{x}) = C_i \wedge x_{r+1} = C_j\}|, \;\; i,j \in \{0,1\} \qquad (52)\text{span}$$

s jejíž pomocí můžeme definovat chybu našeho klasifikátoru jako

$$\varepsilon = \Big(\frac{p(1,0)}{p(1,0)+p(0,0)}\,,\;\frac{p(1,1)}{p(1,1)+p(0,1)}\Big) \qquad (53)\text{span}$$

Z definice snadno vidíme, že optimálním bodem je bod $(0,1)$, tedy že žádný šum se neklasifikuje jako signál a každá gamma částice se klasifikuje správně. Otázkou je, jak dostat více hodnot ε. Ve výše uvedeném postupu pro generování stromů jsme pracovali s pojmem *penalizace*. Jak jsme již zmiňovali, *C5.0* umožňuje nastavit různou váhu pro klasifikační chybu, defaultně jsou tyto váhy nastaveny na 1 a my měníme hodnotu váhy při špatné klasifikaci gamma částic. Toto je tedy parametr, který variujeme (od min.penále po max.penále) a dostáváme tak vícero bodů na křivce.

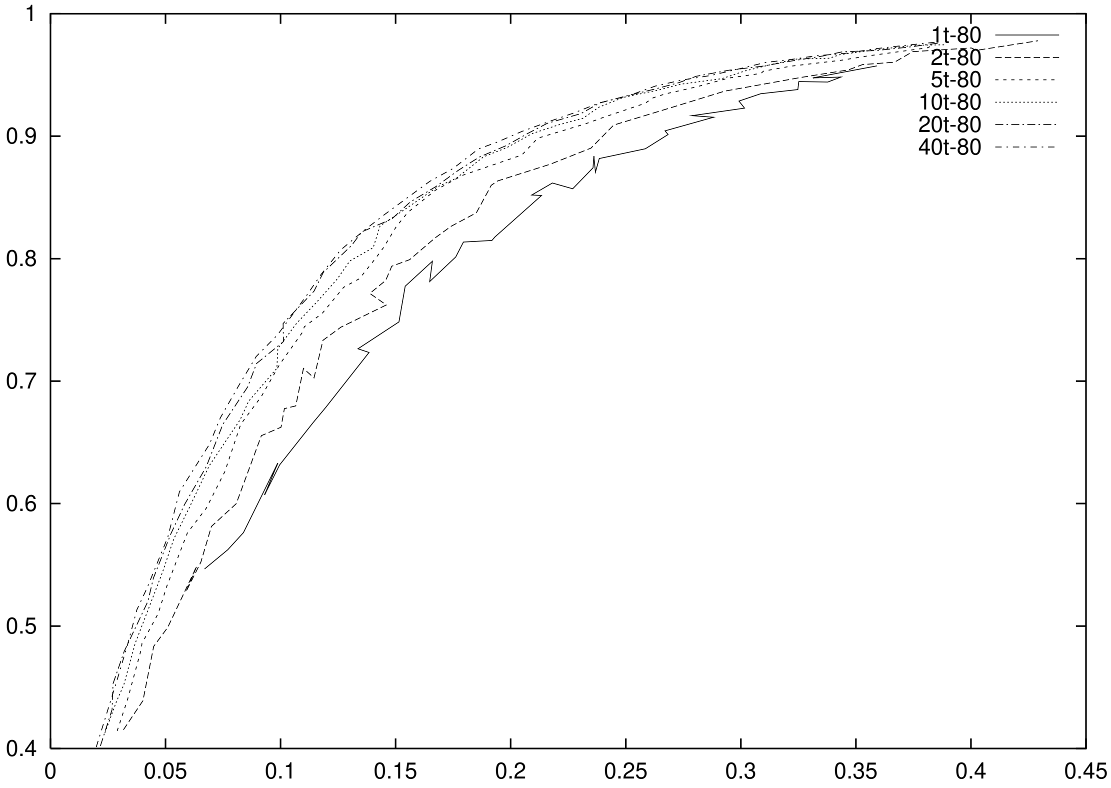
Nyní se konečně dostáváme k popisu grafů z experimentu *GA-OFF*. Vidíme na nich srovnání metod *C5.0* a *CART* s ostatními metodami (viz. [9]) používanými jinými lidmi na projektu spolupracujími (obr.10). Zajímavý je graf, při jehož konstrukci jsme měnili velikost učící množiny (obr.11). Vidíme, že velký rozdíl v kvalitě klasifikátoru je mezi 10 a 30 procenty učící množiny, dále už nárust přesnosti není až tak výrazný. Nejhůře výsledky dopadly, když jsme na učení použili celou trénovací množinu, protože tak vlastně dostaneme několik úplně shodných stromů, čili se jedná o hlasování jediného stromu. Při změně počtu stromů (obr.9) v *C5.0* nedochází od 10ti stromů k výraznému zlepšení, ale rozhodně více stromů klasifikaci zlepší. Při hlasování jednotlivých stromů používáme mechanismus boostingu v *C5.0* použitý, tedy že hlas každého stromu má jinou váhu (tyto váhy se v *C5.0* nazývají *konfidence*) . V našem případě se konfidence pro třídy C_0 a C_1 počítají následujícím způsobem (v *C5.0* označovaným jako *Laplaceova korekce*)

$$k_i = \frac{N_i + 1}{N + 2}, \;\; i \in \{0,1\}, \;\; N = N_0 + N_1 \qquad (54)\text{span}$$

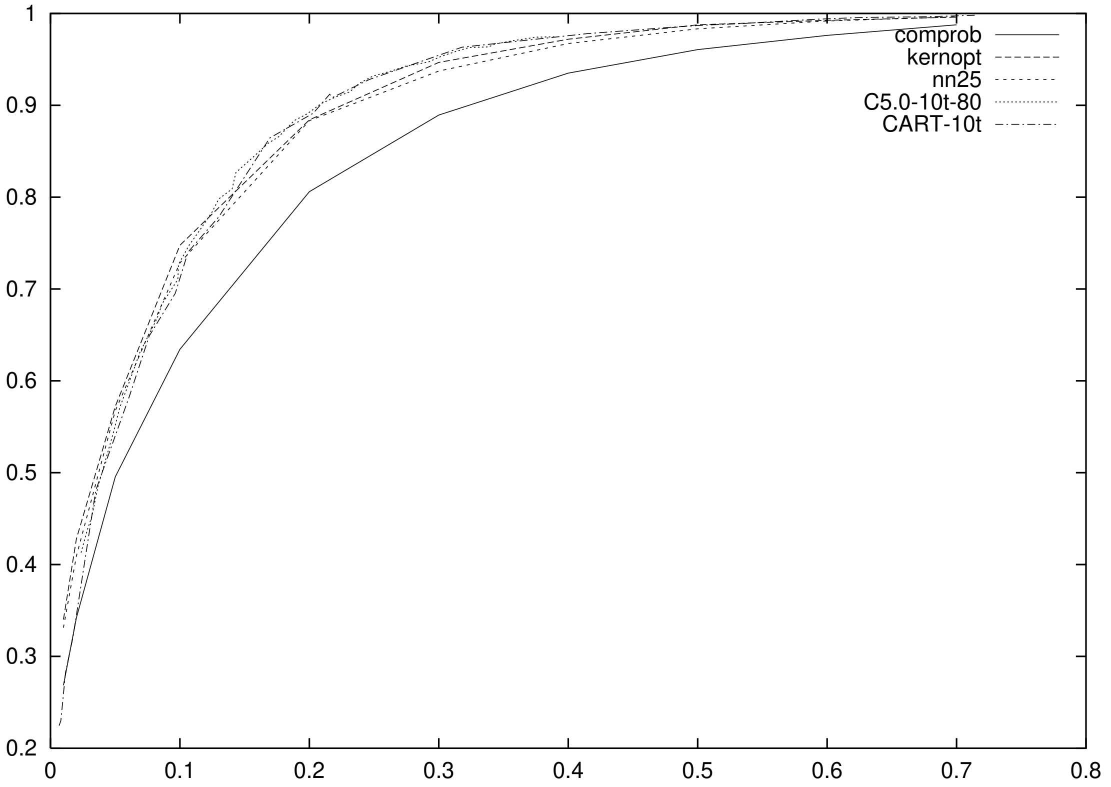
kde N_i je počet případů z trénovací množiny, náležící do třídy C_i, které se při učení dostali až do onoho listu, kam se nyní "prosypal" neznámý případ (respektive případ z testovací množiny).

4. Závěr

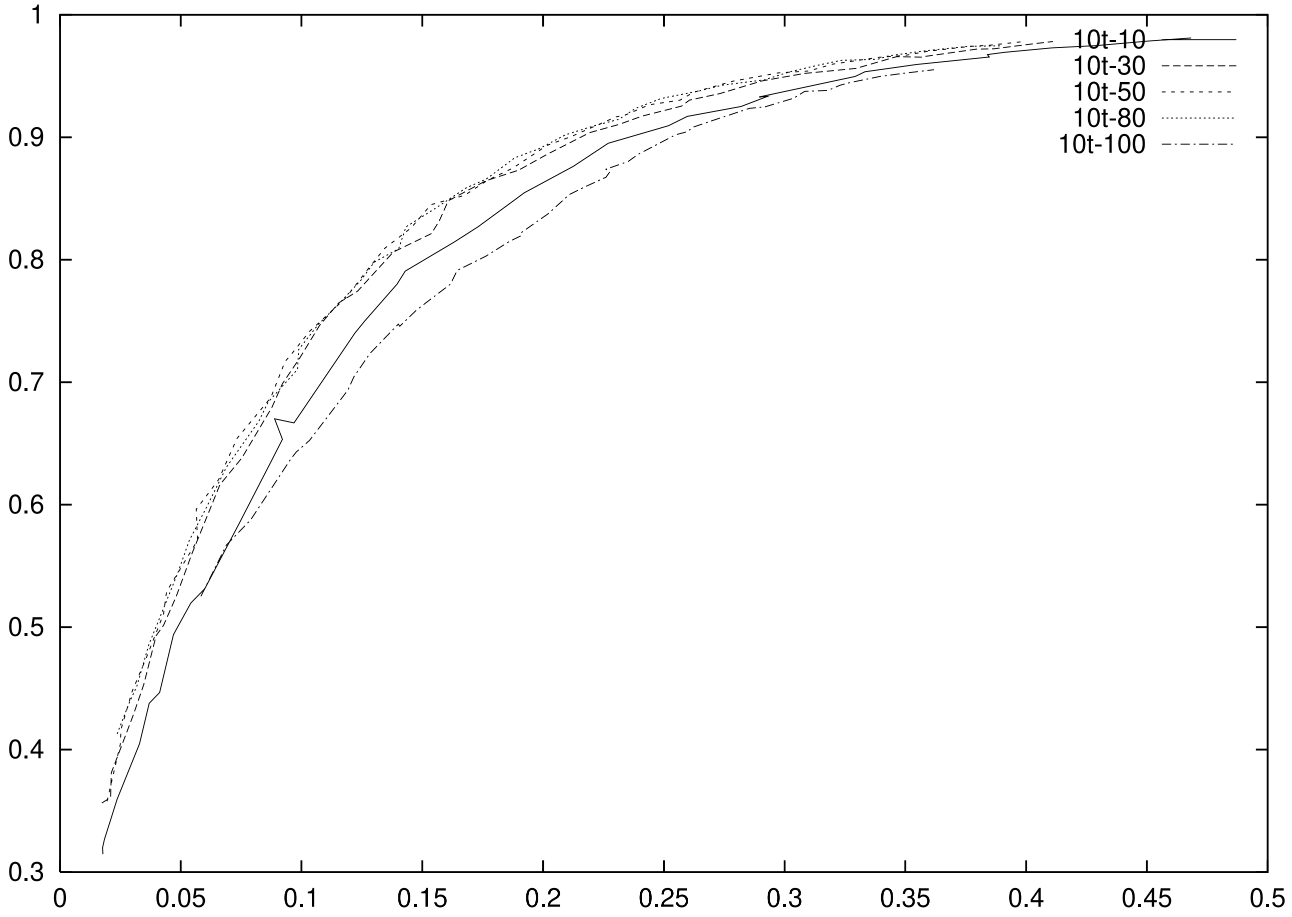
Při experimentování s metodami jsme narazili na místa, v nichž by se podle našeho mínění dali vylepšit. Tyto problémy lze shrnout do tří oblastí – způsoby získání více stromů, prořezávání a kombinování stromů. V současnosti pracujeme na knihovně *forestanalysis* pro softwarový balík *R*, ve které se pokoušíme realizovat naše myšlenky pro analýzu rozhodovacích lesů. Vstupem pro tuto knihovnu mohou být stromy vytvořené pomocí *C5.0*, *CART* i *Random Forest* a snažíme se najít vhodné váhy při hlasování více stromů. *Random Forests* a *CART* stromy v lesích neváží vůbec a *C5.0* je váží pomocí konfidencí (54). Do budoucna bychom chtěli pomocí vhodného vážení stromů tyto metody vylepšit.



Obrázek 9: Závislost chyby klasifikátoru na počtu stromů. 10t-80 znamená, že klasifikátor se skládá z 10ti stromů a při učení se použilo 80% případů z trénovací množiny



Obrázek 10: Srovnání metod, CART i C5.0 používají 10 stromů



Obrázek 11: Závislost na změně velikosti učící množiny, použito bylo 10 stromů

References

[1] J.R. Quinlan, "C4.5:Programs for Machine learning", *Morgan Kaufmann Publishers*, 1993.

[2] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, "Classification and regression trees", *Belmont CA: Wadsworth*, 1984.

[3] J.R. Quinlan, "Boosting, bagging and C4.5", *Proceedings of AAAI'96.*, 1996.

[4] L. Breiman, "Random forests", *Machine Learning*, vol. 45, p. 5–32, 2001.

[5] R.E. Shapire, Y. Singer, "Improved boosting algorithms using confidence-rated predictions", *Machine Learning*, vol. 37, p. 297–336, 1999.

[6] H. Kim, W. Loh, "CRUISE User manual", *Technical report 989, University of Wisconsin, Madison*, 2000.

[7] H. Kim, W. Loh, "Classification trees with unbiased multiway splits", *Journal of the American Statistical Association*, vol. 96, p. 589-604, 2001

[8] W.-Y. Loh, Y.-S. Shih, "Split selection methods for classification trees", *Statistica Sinica*, vol. 7, p. 815-840, 1997

[9] R.K. Bock, A. Chilingarian, M. Gaug, F. Hakl, T. Hengstebeck, M. Jiřina, J. Klaschka, E. Kotrč, P. Savický, S. Towers, A. Vaiciulis, W. Wittek, "Methods for multidimensional event classification: a case study using images from a Cherenkov gamma ray telescope", *v tisku*, 2002

Symbolic Rule Extraction using Artificial Neural Networks

doktorand:
MICHAL JAKOB
Katedra kybernetiky, FEL ČVUT v Praze
Technická 2
166 27 Praha 6
jakob@labe.felk.cvut.cz

školitel:
MARTIN HOLEŇA
Ústav informatiky AV ČR
Pod vodárenskou věží 2
182 07 Praha 8
martin@cs.cas.cz

obor studia:
Umělá inteligence a biokybernetika

Abstrakt

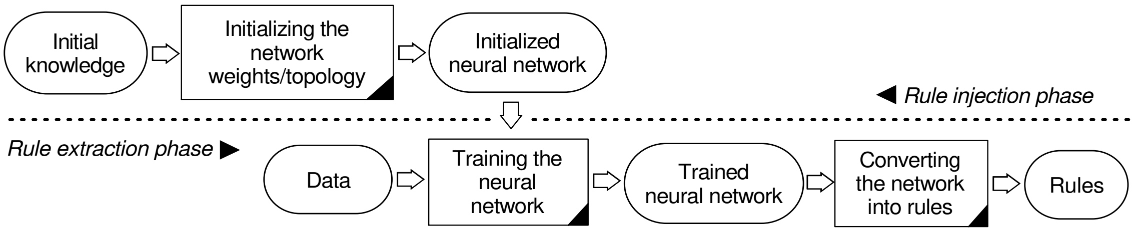
Due to ever increasing amount of collected data, automatic knowledge acquisition has become a key concern in artificial intelligence. *Rule extraction using neural networks* presents an attractive approach to this problem because it combines strengths of both its constituents – the straightforward manner in which neural networks can learn from training data and the comprehensibility of the rule set representation. Extraction methods that employ structural learning in the training of neural network seem particularly promising because they produce skeletal networks which facilitate subsequent rule extraction. As a prominent example of this class of methods, rule extraction by successive regularization is described in this paper. When applied to the well-known mushroom classification problem, the method yielded so far the simplest rule classifying all examples correctly.

Keywords: neural networks, rule extraction, rule refinement, machine learning, classification, knowledge discovery, knowledge transformation

1. Introduction

The development of methods for rule extraction from neural networks was originally motivated by an effort to overcome persistent problems of neural network – their very low comprehensibility and inability to provide justification and explanation of their outputs. Later on, these methods started to be used also as a tool with a primary goal of extracting rules from data. This application of neural networks in rule extraction is termed *rule extraction using neural networks* to stress the fact that neural networks are used as an intermediary link between data and extracted rule sets.

The process of rule extraction proceeds by first training a neural network on the data being analyzed, followed by transformation of the resulting network into a corresponding rule set representation (see Figure 12). By means of this transformation, rules *implicitly* represented by numerical connection weights and topology of the trained neural network are expressed in an explicit form. Moreover, by linking rule extraction with a mechanism for converting rules into an a equivalent neural network (so-called *rule injection* – see Figure 12), neural networks can also be used for the refinement of existing symbolic theories, thus addressing a



Obrázek 12: Rule refinement using neural networks

long-standing machine learning problem of how to integrate prior expert knowledge with learning from data.

2. Rule Extraction

Rule extraction is a special case of knowledge extraction, which can be viewed as searching for an optimum concept in a given concept class with respect to the data being analyzed. There are two alternative ways to define this optimum [2]. *Mental fit* measures how comprehensible the extracted concept is to a human, and how does it thus help to *understand* the data under consideration. *Data fit* concerns how closely the extracted model approximates actual relationships in the data. Mental fit and data fit are often competing goals and it is hence highly desirable that the extraction method allows the user to choose whether, and to what extent, he prefers high accuracy over the comprehensibility of extracted models or vice versa.

2.1. Rule Sets

In the context given above, *rule extraction* (also termed *rule induction*) can be viewed as knowledge extraction with rule sets as a target concept class. Rule sets represent embedded knowledge by means of symbolic rules. Although several types of rules exist, we focus here on classical Boolean rules as they are the most comprehensible, yet offer sufficiently good classification accuracy for a wide range of tasks. They are of the following form

$$\text{IF } x_1 \in S_1^{(k)} \wedge x_2 \in S_2^{(k)} \wedge \cdots \wedge x_N \in S_N^{(k)} \text{ THEN Class(X)} = C_k \qquad (55)\mathrm{span}$$

where x_i are values of attributes of example X and $S_i^{(k)}$ are sets of symbolic values, discrete numerical values, or intervals for continuous features.

The main attraction of rule sets is the high mental fit they provide. Symbolic rules are a common way of communicating knowledge among people; likewise, expert knowledge is also often expressed in a form of rules. Other advantages include the possibility to apply various symbolic manipulation and inferencing techniques to rule sets. From the knowledge discovery perspective, rule sets are thus a favorable way of expressing extracted knowledge.

2.2. Why Rule Extraction using Neural Networks?

While we have already explained while rule sets are a suitable target concept class for knowledge extraction, a question remains what are the advantages of using neural networks to extract rules in comparison to extracting rules directly from data, as performed e.g. by C4.5, CN2 or AQ-family algorithms. The arguments include:

handling continuous attributes Neural networks can be trained on data with continuous attributes; discretization of continuous inputs[1] – and the loss of information which such discretization necessarily entails – can be postponed to later stages of the overall extraction process, thus potentially improving the quality of extracted rules.

---

[1] Discretization of continuous attributes is in either case necessary in order to obtain rules of the form (55).

universal approximation Neural networks with sufficiently high number of hidden units are capable of approximating continuous functions to an arbitrary degree of accuracy, thus providing a very good data fit to the data being analyzed.

straightforward training Training of a neural network consists in the search for network weights that, for a given training set, minimize the network error function. The search space of network weights is normally continuous and the error function differentiable. This allows more informed and thus more effective search than in case of discrete search spaces, which are often the case for algorithms inducing rules directly from data.

multivariate search Neural network training algorithms perform multivariate search, i.e., they take into consideration all input attributes at the same time. This may allow them to arrive at better solutions than techniques working in a ”attribute by attribute” manner, e.g. decision tree induction algorithms.

fine-grained representation of knowledge Neural networks provide a finer-grained representation of knowledge than rule sets. This allows more subtle modification of the incorporated knowledge during learning, which is especially beneficial for knowledge refinement.

Additionally, there are other well-known advantages of neural networks such as high tolerance to noise and very good generalization capability; at the same time, these are the properties which methods for direct rule extraction often lack. So although, due to their low comprehensibility, neural networks are less useful as a target knowledge representation, the above mentioned properties make them a valuable intermediary tool for rule extraction. By first training a neural network on the data and then transforming it into a rule set, the best of both approaches can be exploited simultaneously. For a comprehensive overview and taxonomy of rule extraction methods using neural networks see [1, 9].

3. Rule Extraction by Successive Regularization

Many rule extraction algorithms proceed by first extracting rules at the level of individual network units and subsequently aggregating them to form global relationships. This process is, however, generally quite complicated and requires some mechanism to approximate functions computed by network units using Boolean functions, which can be subsequently expressed as formulas.

3.1. Structural Learning

As a possible solution to this problem, *structural learning* methods, which aim at producing networks that simplify the rule extraction step, have been proposed. Structural learning methods define the error function of the network in such a way that its minimization leads to a network with a skeletal structure of connection weights. Conversion of the resulting skeletal network into a rule set is then a relatively straightforward task. Various ad hoc techniques that are often used in network-to-rule conversion (such as ignoring weak connections and clustering/unification of weights) are no longer necessary.

3.2. Successive Regularization

A prominent example of extraction methods employing structural learning is *rule extraction by successive regularization* proposed by M. Ishikawa [4, 5]. Depending on the complexity of the task, a network with one or two hidden layers of is used. The network units compute standard sigmoid transfer function $h(x) = 1/(1 + e^{-x})$. In its basic form, the method requires binary input values. This is not a restrictive requirement since discrete attributes can be easily converted into a binary representation; continuous attributes have to be either discretized first, or a modified method which can deal with continuous attributes directly might be used (see [6] for details).

Rule extraction by successive regularization introduces three additional terms into the network error function – *the forgetting term*, *the hidden unit clarification term*, and *the selective forgetting term*. The training proceeds in three steps, in each step a different combination of the above mentioned terms is included in the error function (See section [Network training] in Figure 13). The network resulting from regularization training has (almost) binary outputs of its hidden units and only a small proportion of non-zero network weights. Put together, it allows to express each hidden unit as an equivalent, simple Boolean formula. To

```
1. Set the initial value of the regularization parameter $\lambda$
2. [Rule acquiring step]
    a. [Network training]
        Apply the learning with forgetting
        Apply the learning with forgetting and with hidden unit clarification
        Apply the learning with selective forgetting and hidden unit clarification
    b. [Rule extraction]
        Represent each hidden unit as a minimum Boolean function of input units
        Represent each output unit as a minimum Boolean function of hidden units
        By combining the above two Boolean functions, express each output unit as a Boolean function of
        input units
3. If the extracted rules does not classify training samples satisfactorily
    then
        Freeze the connection weights corresponding to extracted formulas
        Decrease the regularization parameter $\lambda$
        Go to step 2
    else Stop
```

Obrázek 13: Algorithm for rule extraction by successive regularization

derive a minimum formula representing a given Boolean hidden unit function, we employ Quine-McClusky minimization method [7], which – in contrast to Karnaugh map method used by Ishikawa – does not impose any limit on the number of inputs and is easily automatized. By advancing layer by layer, each output of the whole network can be represented as a Boolean formula of network inputs.

3.3. Hierarchical Rule Extraction

An important feature of rule extraction by successive regularization is its ability to extract a *hierarchically* ordered set of rules; this is achieved by repeating the rule acquiring step several times with successively decreased regularization parameter while freezing connection weights corresponding to already extracted rules. The overall scheme of rule extraction by successive regularization is depicted in Figure 13.

3.4. Advantages of Rule Extraction by Successive Regularization

In comparison to other methods for rule extraction using neural networks, successive regularization posses several advantages [5]:

hierarchical extraction – the method first extracts a small number of dominant rules at an earlier stage and less dominant rules or exceptions at later stages, which agrees with a human tendency to interpret data and thus improves rule comprehensibility

scalability in comprehensibility – through the choice of the regularization parameter λ, the method allows to control the complexity of extracted rule sets

robustness – thanks to successive regularization, only relatively small networks are generated in each rule acquiring step. The structure of resulting network, and in turn also the extracted rules, are therefore less sensitive to the initial network connection weights.

4. Experimental Results

Ishikawa applied rule extraction by successive regularization to various tasks, both with discrete and continuous valued attributes. In these experiments, the method exhibited very good scalability in comprehensibility; in several cases, it extracted the simplest rule set known for a given level of accuracy. See [6, 5] for a detailed description of results.

Our version of rule extraction by successive regularization was implemented using Matlab and Matlab

Attribute	No*	Attribute	No*
cap-shape	6	stalk-surface-above-ring	4
cap-surface	4	**stalk-surface-below-ring**	**4**
cap-color	10	stalk-color-above-ring	9
bruises?	2	stalk-color-below-ring	9
odor	**9**	veil-type	2
gill-attachment	4	veil-color	4
gill-spacing	3	ring-number	3
gill-size	**2**	ring-type	8
gill-color	12	**spore-print-color**	**9**
stalk-shape	2	**population**	**6**
stalk-root	7	habitat	7

span span span $*numberofattributevalues$

span span span $Theattributesappearinginthefinalruleareprintedinbold$

Tabulka 5: Mushroom dataset attributes

Optimization Toolbox. It employs a subspace trust region algorithm based on the interior-reflective Newton method for network training. Here, we present results of its application to the mushroom classification dataset, which ranks among the most widely-used machine learning benchmark problems.

4.1. Mushroom Classification

The mushroom dataset and its comprehensive description can be found in the UCI learning repository[2]. Patterns for the problem are described by 22 discrete attributes with a total of 126 different attribute values (See Table 5). The mushroom dataset consists of 8124 examples, 51.8% of which represent edible and the rest nonedible (mostly poisonous) mushrooms. One tenth of the dataset, i.e. 812 examples, was randomly chosen for training. The extraction proceeded in four steps with gradually decreasing regularization parameter λ. The progress of extraction, together with the increasing accuracy of the extracted rule, is depicted in Table 6. The final rule

$$\begin{aligned}\text{IF (odor} \quad &= \text{almond} \vee \text{anise} \vee \text{none}) \wedge (\text{spore-print-color} \neq \text{green})\\ &\wedge((\text{gill-size = broad}) \vee (\text{stalk-surface-below-ring} \neq \text{scaly}))\\ &\wedge(\text{population} \neq \text{clustered})\ \text{THEN edible}\end{aligned} \tag*{(56)span}$$

is the same as obtained by Ishikawa, however, the order in which it was obtained is slightly different (Ishikawa used only three rule acquiring steps). To our best knowledge, it is the most compact classification rule for mushroom dataset achieving 100% accuracy.

The comparison of rule extraction by successive regularization with other neural network rule extraction methods as well as with the C4.5 decision tree induction algorithm is depicted in Table 7. The description of C-MLP2LN, M-of-N3 and Boolean approximation algorithms can be found in [3, 10, 8], respectively.

---

[2] http://www1.ics.uci.edu/ mlearn/MLRepository.html

Rule	Misclassified*
1. (odor = almond $\vee$ anise $\vee$ none)	11 / 120
2. $\wedge$ (spore-print-color $\neq$ green)	6 / 48
3. $\wedge$ ((gill-size = broad) $\vee$ (s-s-b-r** $\neq$ scaly))	0 / 8
4. $\wedge$ (population $\neq$ clustered)	0 / 0

span $*trainingset/entiredataset$

span $**stalk-surface-below-ring$

Tabulka 6: Progress of mushroom rule extraction

Method	Rule accuracy (%)	Rule complexity*
Successive regularization	100.0	5 / 7
C-MLP2LN	99.9	4 / 7
M-of-N3	98.1	? / 9
Boolean approximation	96.5	2 / 8
C4.5	98.7	1 / 8

span span $*numberofattributesused/numberoftermsintheruleantecedent$

Tabulka 7: Method comparison on mushroom dataset

5. Conclusions

In this paper, a short introduction to rule extraction using neural networks was presented. The strength of this approach to knowledge extraction lies in the combination of high learning capability of neural networks with the comprehensibility of target rule set representation. Methods that use structural learning for neural network training show a particularly promising direction since they integrate a substantial part of rule extraction task into neural network training, thus making the finale network-to-rule conversion relatively easy. A prominent example of this class of rule extraction methods is rule extraction by successive regularization. When applied to a range of benchmark problems, this method proved the ability to extract compact rules while maintaining very high accuracy of extracted rule sets. Our variant of rule extraction by successive regularization implements Quine-McClusky logic minimization procedure during the rule extraction phase. In preliminary experiments on the mushroom classification dataset, it achieved results almost identical to those reported by Ishikawa in [5]. Results on additional benchmark problems as well as on a selected real-world problem will be given in the workshop presentation.

Acknowledgements

The research presented in this paper was supported by CTU internal grant No. CTU0208713.

References

[1] R. Andrews, J. Diederich, and A.B. Tickle. Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Based Systems*, 8(6):373–389, 1995.

[2] M. Berthold and D.J. Hand, editors. *Intelligent Data Analysis*, chapter Rule Induction, pages 195–216. Springer, 1999.

[3] W. Duch, R. Adamczak, and K. Grabczewski. A new methodology of extraction, optimization and application of crisp and fuzzy logical rules. *IEEE Transactions on Neural Networks*, 12(2):277–306, 2001.

[4] M. Ishikawa. Structural learning with forgetting. *Neural Networks*, 9(3):509–521, 1996.

[5] M. Ishikawa. Rule extraction by successive regularization. *Neural Networks*, 13(10):1171–1183, 2000.

[6] M. Ishikawa and H. Ueda. Structural learning approach to rule discovery form data with continuous valued inputs. In *ICONIP '97*, pages 898–901. Springer, New York, 1997.

[7] William V. Quine. The problem of simplifying truth functions. *American Mathematical Monthly*, 59:521–531, October 1952.

[8] R. Setiono. Extracting m-of-n rules from trained neural networks. *IEEE Transactions on Neural Networks*, 11(2):512–519, 2000.

[9] A.B. Tickle, R. Andrews, M. Golea, and J. Diederich. The truth will come to light: Directions and challenges in extracting the knowledge embedded within trained artificial neural networks. *IEEE Transactions on Neural Networks*, 9(6):1057–1068, 1998.

[10] H. Tsukimoto. Extracting rules from trained neural networks. *IEEE Transactions on Neural Networks*, 11(2):377–389, 2000.

Elektronická zdravotní dokumentace

doktorand:
ING. PETR HANZLÍČEK
Ústav informatiky, EuroMISE Centrum, Pod vodárenskou věží 2, 180 00, Praha 8
hanzlicek@euromise.cz

školitel:
PROF. RNDR. JANA ZVÁROVÁ, DRSC.
Ústav informatiky, EuroMISE Centrum, Pod vodárenskou věží 2, 180 00, Praha 8
zvarova@euromise.cz

obor studia:
Biomedicínská informatika

Úvod

Zdravotní záznamy jsou v současnosti zavedenou součástí klinické praxe. Tyto záznamy obsahují důležité informace pro léčebnou péči a užívají se různými způsoby pro různé účely. Snahou je reprezentovat tyto záznamy na elektronickém médiu tak, aby byly zpracovatelné počítačovým systémem. Elektronický zdravotní záznam lze pak chápat jako technologický prostředek pro dokumentaci léčebného procesu.

V Evropě i ve světě pracuje řada organizací, které se zabývají výzkumem a vývojem prototypů elektronického zdravotního záznamu, oblast elektronického zdravotního záznamu je i předmětem činnosti národních i mezinárodních standardizačních organizací. V tomto příspěvku ukazuji současnou situaci v oblasti výzkumných projektů a existujících standardů a předkládám návrh implementace systému, kompatibilního s vybranými evropskými normami a standardy.

Normy a standardy

V současné době existuje ve světě celá řada organizací, zabývajících se přípravou norem a standardů z oblasti elektronické zdravotní dokumentace. Jedná se o dokumenty, pokrývající oblast komunikace, reprezentace dat, terminologie, bezpečnosti a interoperability.

Na mezinárodní úrovni hraje významnou roli technická komise č. 215 mezinárodní organizace pro standardizaci ISO (ISO/TC215) [1]. Jednotlivé pracovní skupiny se věnují koordinaci elektronických zdravotních záznamů a tvorbě informačních modelů, problematice komunikace a přenosu zpráv, reprezentaci zdravotních konceptů, bezpečnosti a zdravotním kartám. Komise je přímým autorem dvou standardů, další byly připraveny ve spolupráci s dalšími standardizačními institucemi.

V Evropě existuje obdobná organizace jako ISO - Evropský výbor pro normalizaci CEN. Problematikou zdravotnické informatiky se v rámci CENu zabývá jeho technická komise č. 251, informující o své činnosti na svých webovských stránkách http://www.centc251.org/ [2]. Komise za dobu své existence připravila přes 40 dokumentů předběžných evropských norem. Z hlediska dlouhodobé snahy naší republiky o vstup do Evropské unie jsou výsledky práce této evropské organizace pro nás klíčové a jsou také postupně zaváděny do české soustavy norem.

Další technickou komisí, jejíž činnost souvisí s oblastí medicínské informatiky, je technická komise č. 224, zabývající se problematikou elektronických karet.

Další organizací, produkující standardy, jejíž činnost v poslední době nabývá na významu, je americká organizace Health Level Seven [4], připravující standard pro komunikaci ve zdravotnictví HL7. Ze standardu, původně specifikujícího pouze obsah a formát komunikace na aplikační úrovni se má v připravované verzi 3 stát komplexní soubor dokumentů, věnující se celé problematice medicínské informatiky. Ve verzi 3 je výměna zpráv založena na datovém modelu. V základním RIMu (Referenční informační model, Reference Information Model, viz Obrázek 16) a v DIMech (Doménové informační modely, Domain Information Models) jsou modelovány komunikační vazby ve zdravotnických systémech a slouží jako základ pro všechny pracovní skupiny používající HL7.

Další organizace a výzkumné projekty

Dalším projektem, který se zabývá hledáním optimálních způsobů realizace elektronického zdravotního záznamu, je australský projekt Good Electronic Health Record (GEHR) [5]. Základem jeho koncepce je formální sémantický model (GEHR Object Model, GOM), popisující koncepty ve třech úrovních - EHR (záhlaví, transakce), struktury pro reprezentaci obecných znalostí (zjištění, subjektivní informace, instrukce) a nízkoúrovňová data (datové typy, jednotky, multimédia). Klinické modely jsou popisovány mimo GOM pomocí tzv. archetypů - základních typických stavebních jednotek jako např. délka nebo hmotnost. Tyto archetypy jsou pak odkazovány ve struktuře pojmů, používaných pro popis zjištění u pacienta v elektronickém zdravotním záznamu.

Problematice strukturovaného elektronického zdravotního záznamu a optimálního uživatelského rozhraní se věnoval evropský projekt I4C/TripleC [6]. Jeho cílem bylo vyvinout systém, poskytující integrovaný přístup k datům bez ohledu na místo jejich uložení, podporu lékařské péče konzultacemi na dálku a konzistentní záznam pacientských dat, obrazů a biosignálů v rámci jednoho elektronického záznamu. Obdobný přístup byl zvolen v projektu Synapses, ve kterém byla snaha o využití existujících informačních systémů pro klinické informace, laboratorní data a vytvořit otevřené řešení pro sdílení dat mezi těmito systémy. Základem byla hierarchická struktura údajů, udržovaná jedním centrálním serverem, vlastní data pak byla získávána přímo z konkrétního systému, ve kterém byla uložena.

Hlavní zdroje inspirace

Hlavními zdroji myšlenek a inspirací pro náš vývoj byly jednak předběžné evropské normy, připravované technickou komisí CEN/TC251, jednak výsledky evropského projektu I4C/TripleC 4. rámcového programu, v jehož rámci byl vyvinut software elektronického zdravotního záznamu ORCA (Open Record for Care). Architekturu elektronického zdravotního záznamu popisuje dokument CEN/TC251 ENV13606 "Sdělování elektronických zdravotních záznamů" [3], který je od roku 2001 součástí české soustavy norem. Tento předběžný standard má následující 4 části:

- Část 1: Rozšířená architektura
- Část 2: Seznam pojmů domény
- Část 3: Distribuční pravidla
- Část 4: Zprávy pro výměnu informací

Hlavní význam tohoto předběžného standardu je v oblasti komunikace mezi elektronickými zdravotními záznamy. V první části je popsán konceptuální model struktury a obsahu, vhodný pro výměnu informací mezi elektronickými zdravotními záznamy. Dokument odkazuje na řadu dalších standardů CEN a ISO.

Druhým zdrojem inspirace byl software elektronického zdravotního záznamu ORCA. Tento software využíval dvouvrstvou architekturu klient-server (databázová a uživatelská vrstva) a strukturovaný způsob uložení dat kombinovaný s možností uložení multimediálních objektů v historii pacienta [7]. Primárním způsobem vkládání dat byl vstup strukturované informace s možností uvádět k jednotlivým položkám textové komentáře. Množina sbíraných dat byla definována hierarchicky strukturovanou znalostní bází, jejíž součástí byl popis obsažených termínů v několika evropských jazycích. Vložené strukturované údaje tak bylo možné snadno přeložit do jiného jazyka bez ohledu na to, jaký jazyk byl použit při jejich vkládání. EuroMISE centrum se účastnilo druhé části projektu - projektu TripleC. Cílem projektu byla adaptace software Orca na české podmínky a otestování použitelnosti strukturovaného záznamu o pacientovi sběrem dat pro výzkumné účely [8].

Vlastní návrh

Naše zkušenosti a myšlenky inspirované projektem I4C/TripleC vyústily v následující seznam požadavků, které by elektronický zdravotní záznam měl splňovat.

- strukturovaný způsob uložení informací kombinovaný s volným textem
- nástroje k usnadnění strukturalizace uložené informace (převod volného textu na strukturovaná data)
- nástroje pro vyhodnocování a vizualizaci uložených dat
- množina sbíraných údajů dynamicky rozšiřitelná a modifikovatelná bez změny databázové struktury
- systém pro řízení přístupu k pacientským datům a znalostní bázi
- nástroje pro kontrolu správnosti uložených dat a souhlasu s lékařskými doporučeními
- podpora práce ve více jazycích
- informace o rodinných vztazích mezi pacienty pro výzkum v oblasti genetiky
- sdružování dat příslušející k jednotlivým událostem u pacienta
- záznam o všech změnách pacientských dat, znalostní báze, změn přístupových práv apod.
- minimální závislost na použitém databázovém nebo operačním systému
- široká využitelnost (od jednotlivé pracovní stanice na stole praktického lékaře až po distribuované prostředí ve velké nemocnici)
- možnost přístupu k uloženým datům z mobilních terminálů (PDA, mobilní telefony)
- motivační systém pro lékaře (automatizace rutinní administrativní práce, reporty pro zdravotní pojišťovny, apod.)
- multimediální informace jako součást elektronického záznamu.

Na základě zmíněných požadavků byl zahájen výzkum a vývoj nových způsobů reprezentace a realizace elektronického zdravotního záznamu. Základem námi navrhované struktury je třívrstvá architektura, sestávající z datové vrstvy, funkční vrstvy a uživatelského interface (prezentační vrstva). Výhodou tohoto řešení je snadnější údržba a ladění aplikace z důvodu oddělení prezentační a funkční vrstvy, optimalizace zdrojů rozdělením výkonu (možnost distribuce aplikace v rámci sítě), spolehlivost a odolnost proti výpadkům a zvýšení bezpečnosti. Architektura systému tak umožňuje plynulý přechod mezi systémem instalovaném na jediném PC až po instalace klient-server v rámci distribuované sítě.

Základem datové reprezentace je myšlenka oddělení popisu a hodnot veličin, které v elektronickém zdravotním záznamu uchováváme. Veličiny a jejich vzájemné vztahy definuje znalostní báze, reprezentovaná grafovou strukturou, vlastní data jsou uložena pomocí jednoduché univerzální grafové struktury, obsahující pro každý údaj: identifikační údaje pacienta a lékaře, identifikaci vyšetřované veličiny, její hodnotu, spolehlivost, se kterou byl údaj zjištěn, údaje o období platnosti daného údaje a další. Tato struktura umožňuje aplikovat jednotný postup při uložení nebo prezentaci libovolného typu informace, bez změny databázové struktury je možné přidávat nové zjišťované veličiny. Návrh zachovává historii změn znalostní báze i zjištěných hodnot jednotlivých veličin. V rámci funkční vrstvy je řešena logika aplikace, součástí vrstvy je implementace formalizovaných lékařských doporučení. Jedna z připravovaných realizací uživatelského rozhraní zahrnuje systém, kombinující záznam pomocí volného textu s poloautomatickým výběrem zadávané veličiny ze znalostní báze na základě zadávaného textu. Text je pak uložen jako XML dokument, obsahující odkazy na veličiny ve znalostní bázi a jejich hodnoty. Paralelně jsou data uložena v databázi pro další zpracování.

References

[1] International Organization for Standardization, http://www.iso.ch/

[2] European Committee for Normalization, Technical Committee 251 Health Informatics (CEN/TC251), http://www.centc251.org/

[3] ČSN ENV 13606, CEN/TC251

[4] Health Level Seven, http://www.hl7.org/

[5] Good Electronic Health Record, http://www.gehr.org/

[6] Pierik F.H., van Ginneken A.M., Timmers T., Stam H., Weber R.F.: Restructuring routinely collected patient data: ORCA applied to andrology. Yearbook of Medical Informatics 98, Schattauer, Stuttgart 1998

[7] A.M. van Ginneken, M.J. Verkoien, A multi-disciplinary approach to a user interface for structured data entry. Medinfo. 2001;10(Pt 1):693-7

[8] Zvárová J, Hanzlíček P, Přibík V. Application of Orca Multimedia EPR in Czech Hospitals. EuroRec'99 Proceedings Book - Sevilla, ProRec 1999: pp. 160-165

Psychologické dotazníky a detekce psychosomatických stavů

doktorand:
ING. PETR ZAVADIL

EuroMISE
Ústav informatiky AV ČR
Pod Vodárenskou vě«í 2
Praha 8

zavadil@euromise.cz

školitel:
DOC. ING. VLADIMÍR ECK, CSC.

Katedra kybernetiky
FEL ČVUT v Praze
Technická 2
Praha 6

eck@lab.felk.cvut.cz

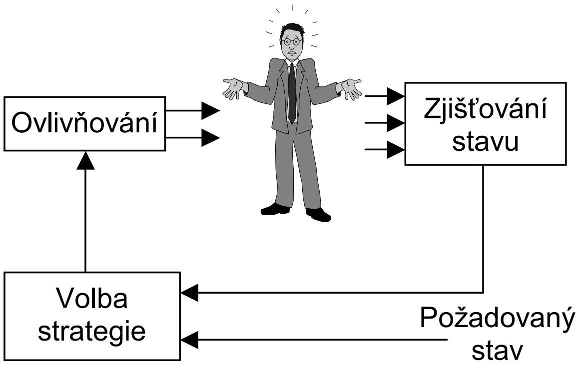
obor studia:
Umělá inteligence a biokybernetika

Souhrn

Článek popisuje metodu detekce psychosomatických stavů operátora s vyu«itím psychologických sebehodnotících Ãkálových dotazníků. Tato metoda doplňuje standardní metody detekce. Znalost psychosomatických stavů mů«e být vyu«ita při uzavření vnějÃí zpětné vazby u člověka. Popisovaná metoda je podpořena experimentálně získanými daty.

1. Úvod

Psychosomatický operátorský přístup spadá do oblasti zpětné vazby (ZV) u člověka. ZV je v reálném čase (obr. 14) reprezentována třemi částmi:



Obr. 14: Uzavření vnějÃí zpětné vazby u člověka

- zjiÃtěním psychosomatického stavu,
- porovnáním zjiÃtěného stavu se stavem po«adovaným a stanovením dalÃí strategie na základě individuálního modelu chování,
- řízeným ovlivňováním (zvýÃením nebo sní«ením zátě«e) na základě zvolené strategie.

Stavy operátora je nutné individuálně zmapovat v identifikačním procesu, který předchází uzavření ZV. Standardní metody sledování psychosomatických stavů člověka se zaměřují na střídání fází klidu a různých druhů krátkodobé i dlouhodobé zátě«e [1]. V průběhu testování jsou měřeny a zaznamenány reakční a fyziologické signály. Na základě znalostí fází a záznamů jsou následně individuálně určeny zjistitelné psychosomatické stavy. Příklad takového přístupu je systém Inteligentního rozhraní vyvíjený na Univerzitě Kyoto, jeho« součástí je analýza a odhad poznávacích stavů soustavy ”Man-machine interface” [2].

V této práci je nadále předkládán odliÃný postup umo«ňující detekovat psychosomatické stavy psychologickými sebehodnotícími metodami. Výsledky tohoto postupu by měly standardní identifikační operátorský přístup doplnit a zpřesnit.

Cílem práce je orientovat se na *metody psychologie*, které vyu«ívají *sebediagnostické dotazníky* pro záznam krátkodobého psychického stavu člověka a stanovit konkrétní postup, který umo«ní *rozdělit* otestovanou *skupinu osob*-operátorů *podle* detekovaných *stavů*.

2. ^kálové emoční dotazníky

V psychologii se záznam krátkodobých emočních stavů vyu«ívá při hodnocení pracovní zátě«e a stresu [3]. Jednou z metod pou«ívaných v této oblasti jsou Ãkálové dotazníky, kde odpovědi probandů vyjadřují míru ztoto«nění s otázkou (např. s adjektivem popisujícím specifický pocit). Míra souhlasu mů«e být vyjádřena výběrem z předkládaného seznamu slovního hodnocení, graficky na ose nebo přímo jednoduchou klasifikací (např. 1 - 5). Po vyplnění jsou otázky řazeny do skupin, které jsou hodnoceny součtem numericky vyjádřených odpovědí. Tento přístup předpokládá u probandů ochotu spolupracovat a koncentraci na sebehodnocení.

Pro dalÃí pou«ití byl vybrán dotazník Ãkály MIND (Ãkály Likertova typu), který byl pou«it při testování pracovní náročnosti ve Státním zdravotním ústavu, v sekci hygieny práce. Obdobný Ãkálový dotazník *12 Faktorů ”Mood Adjective Checklist”* byl poprvé publikován v roce 1965 (Nowlis, University of Rochester N.Y.). Navazující Ãvédská studie [4] se zabývala optimalizací počtu otázek v těchto dotaznících.

Pou«itý dotazník má 2 části, první obsahuje otázky na krátkodobou minulost a druhý pak 40 otázek na okam«itý stav s 5-ti hodnotovými Ãkálami odpovědí. Inovace pro pou«ití dotazníku v této práci je počítačové zpracování. Dosud byl vyplňován v papírové formě a podobné otázky bylo mo«no vizuálně korigovat. Nová softwarová realizace umo«ňuje předkládat otázky jednotlivě, otázky nejdou bezprostředně po sobě, zároveň se zaznamenává čas strávený nad jednotlivými otázkami a vzápětí po vyplnění jsou k dispozici výsledky - individuální v dvourozměrných grafech a souhrnné o skupině či podskupinách v grafech třírozměrných.

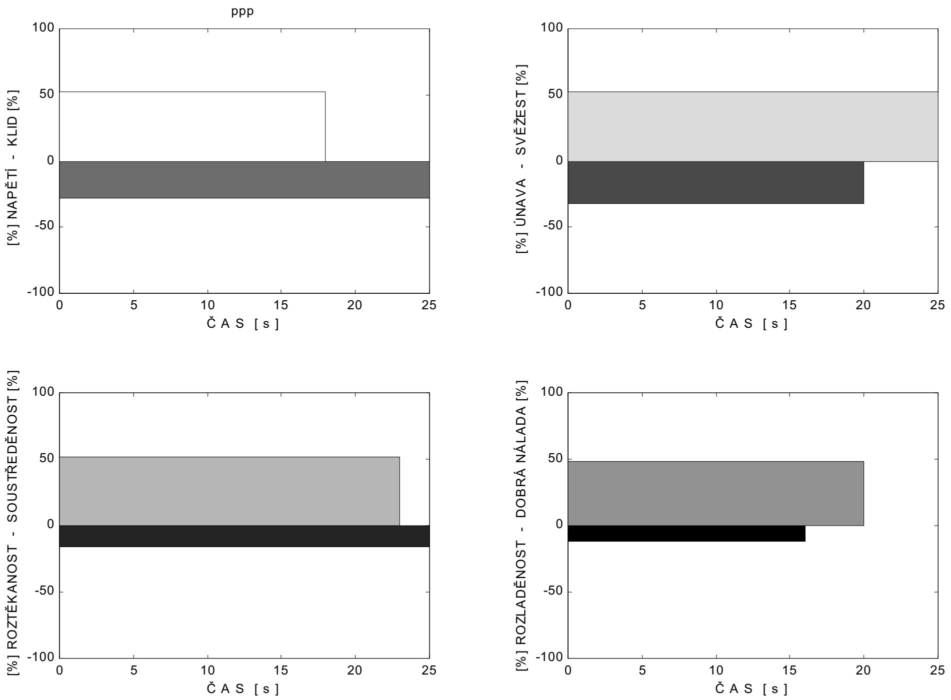
Pro dalÃí zpracování je významnějÃí druhá část s 8-mi skupinami otázek. VÃechny odpovědi jsou vyjádřeny jednou z pěti hodnot Ãkály. Závěrečné vyhodnocení skupin 5-ti otázek je procentuálně vzta«eno k minimálnímu a maximálnímu mo«nému součtu odpovědí ve skupině (obr. 15). Takto zpracované dotazníky umo«ňují individuálně stanovit psychický stav a při testování více probandů tyto stavy lépe heuristicky porovnávat.

3. Prováděné experimenty

3.1. Navr«ené cíle

Cílem provedených experimentů bylo vyu«ít psychologické metody pro detekci psychosomatického stavu operátora. Tento hlavní cíl mů«eme rozdělit na několik dílčích:

- otestovat homogenní skupinu osob vytvořeným počítačově zpracovaným Ãkálovým dotazníkem



Obr. 15: Individuální vyhodnocení emočního dotazníku

- během testování zaznamenávat fyziologické a reakční parametry s předpokládanou psychofyziologickou vazbou a jednoduchou aplikací při testech
- na základě testu rozdělit skupinu podle individuálně vyhodnocených dotazníků
- podpořit rozdělení do podskupin s významnými emočními stavy reakčními a fyziologickými parametry (hledání psychosomatické vazba)
- zhodnotit a optimalizovat test (redukce počtu otázek i jejich skupin, úprava parametrizace, ...)

3.2. Průběh testů

Pro testování byla vytvořena internetová stránka, která slou«ila jako objednávací systém a zájemcům nastínila průběh měření z pohledu testované osoby. Pro ka«dou osobu byla vyhrazena hodina v standardizovaném klidném prostředí. Měření probíhalo v odpoledních hodinách během přibli«ně jednoho měsíce.

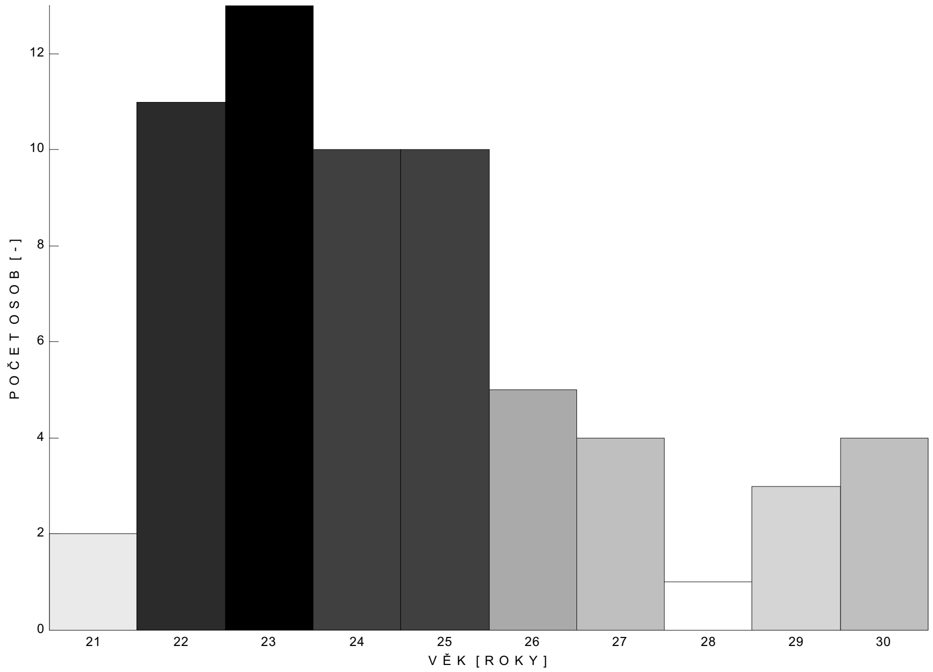
Výběr fyziologických signálů a následné parametrizace byl omezen dostupností a předpokladem o psychosomatické vazbě [5] a [6].

Při testování byl vyplňován vytvořený dotazník pomocí klávesnice a myÃi. Před a po vyplňování byl měřen krevní tlak a počet tepů za 60 s, během vyplňování pak srdeční frekvence s vzorkovací frekvencí 5 s - senzorem na hrudníku (přístroj POLAR) a synchronně byl zaznamenáván čas ka«dé odpovědi.

Po ukončení byli vÃichni testovaní informováni o výsledcích dotazníku na grafickém znázornění a zároveň konzultováni a dotazováni (pochopení otázek, nejasnosti, celkové sebehodnocení apod.)

4. Dosa«ené výsledky

Bylo otestováno 63 osob (na Katedře kybernetiky FEL ČVUT v Praze). Jednalo se převá«ně o studenty, PhD studenty a zaměstnance katedry mezi 21 a 30 lety (histogram je na obr. 16). Průměrný věk testované skupiny byl 24,5 let.



Obr. 16: Histogram testované skupiny

Na základě heuristického přístupu vycházející ze struktury dat, iteračních postupů, pozorování a porovnávání podskupin v 3D grafech byly stanoveny limitní hodnoty pro dvojice skupin otázek. Tyto hodnoty umo«ní bipolárně rozdělit celou testovanou skupinu ve čtyřech daných oblastech. Ka«dá z veličin (reprezentujících odpovědi na emoční dotazník) přispívá pouze do jedné z oblastí. Jednotlivá rozdělení dělí skupinu na dvě významné podskupiny, jednu s jasně vymezeným emočním stavem (konkrétně v«dy pozitivně) a druhou zahrnující zbytek. Výsledky jsou názorně shrnuty v tab. 8.

Tab. 8: Tabulka rozdělení testované skupiny
(hodnoty vyjadřují průměr odpovědí celé skupiny v % a počty osob s daným vymezením odpovědí)

-bb-error = =

5. DalÃí zpracování

Při dalÃím zpracování se autor zaměří na porovnání heuristického stanovení skupin s emočním stavem se standardními statistikými klasifikačními metodami (postupy vyu«ité např. v [7] a [8]).

V následném kroku pak na porovnání skupin emočních stavů s reakčními a fyziologickými parametry, případně s jejich významnými skupinami.

V závěrečné fázi pak autor zhodnotí pou«itou dotazníkovou metodu. Vzhledem ke zjiÃtěným výsledkům pak případně navrhne pro detekci psychosomatických stavů úpravu dotazníků a modifikaci měřených signálů a parametrů.

6. Závěr

Data získaná experimenty a navr«ený postup mapuje mo«nosti detekce psychosomatických stavů operátora pomocí sebediagnostických psychologických dotazníků. Popsaná metodika se nesna«í zachytit operátorské stavy v plném rozsahu, slou«í jen jako zdroj doplňujících informací o zatě«ovaném člověku. Výsledky vyplývající z experimentů je třeba chápat v omezené míře v kontextu testované skupiny osob.

Literatura

[1] ***Zavadil, P.*: Identification and Control of Psycho-physiological Stress**; In: Proceedings of Workshop 2000, Prague: CTU, 2000, p. 531.

[2] ***Yoshikawa, H.; Nakagawa, T.; Nakatani, Y.; Furuta, T.; Hasegawa, A.*: Development of an analysis support system for man-machine system design information**; Journal Paper, Control Engineering Practice, Elsevier, UK, 1997: Vol: 5 Iss: 3 p. 417-25.

[3] ***Hladký A. a kol.*: Zdravotní aspekty zátě«e a stresu**; Praha, UK Karolinum, 1993.

[4] ***Kjellberg, A.; Iwanowski, S.*: Stress/Energi Formulärt: Utveckling av en metod för skattming av sinnesstmäning i arbetet**; Solna, NIOH, Undersöknigsrapport 26, 1989.

[5] ***BaÃtecký, J., ^avlík, J., ^imek, J. a kol.*: Psychosomatická medicína**; Praha, Grada Avicenum, 1993.

[6] ***Brichcín, M., Hoskovec, J., ^tikar, J.*: Přístrojové metody v psychologické diagnostice**; Bratislava, Psychodiagnostické a didaktické testy, 1983.

[7] ***Hebák, P.; Hustopecký, J.*: Vícerozmirné statistické metody a aplikace**; Praha, SNTL, 1987.

[8] ***Máče, M.*: Aplikace statistických klasifikačních metod na biologické objekty**; Diplomová práce na katedře řídící techniky, Praha, ČVUT FEL, 1981.