

The Integration of Neural Networks with Symbolic Knowledge Processing

A. Ultsch
University of Dortmund
Lehrstuhl Informatik VI
P.O. Box 500 500
4600 Dortmund 50

Summary:

A part of the project WINA, sponsored by the German Ministry of Science and Technology, we are studying the integration of Artificial Neural Networks (ANN) with a rule based Expert Systems. The methods of integrating ANN with XPS can be classified into four different approaches: Neural Approximative Reasoning, Neural Unification, Intro spection and Integrated Knowledge Acquisition. Each approach is explained and results of our own realizations of the different approaches are briefly discussed. Keywords: neural networks, machine learning, symbolic data analysis, expert systems, learning.

1. Introduction

In this paper we classify the different approaches for the integration of ANN with symbolic knowledge processing, in particular with Expert Systems (XPS), and give an overview on our own results in the realization of different approaches. Under an ANN we understand a biologically motivated architecture for information processing systems with a corresponding method of programming. This architecture is mainly characterized by a large number of simple processors, called units or neurons, and a network of weighted connections that can be changed. The programming style is called learning or training. Under XPS we understand a computer program which contains an explicit and formal representation of "knowledge" in the form of a knowl edgebase. AN XPS is capable to draw conclusions using this knowledge (inference) and is able to explain the drawn conclusions. Under knowledge we understand a symbolic representation of objects, facts and rules for a interpreter with symbol pro cessing capability, e.g. a human. In particular knowledge is communicable by word or writing. In order to illustrate the distinction between a symbolic representation of data and a subsymbolic representation, consider an example taken from the domain of music. In Figure 1 we see the first few notes of George Bizet's opera "Carmen".

Figure 1: Notes from Georges Bizet's Figure 2: Example for subsymbolic representa opera Carmen tion of data

In Figure 2 we see a set of numbers. They are extracted from a CD with an in terpretation of the same piece of music. So both figures represent the same content: the music of Carmen. The content of Figure 1 we would call a symbolic represen

tation. The notes can be read and understood by humans and in particular be "interpreted", i.e. played. A single number in Figure 2 has no particular meaning. As a matter of fact we could change or omit a single number and only a trained listener would detect it - if the CD player would not even have "corrected" the number. Subsymbolic representation of data is characterized by the property that one single data element does not have a meaning (interpretation) for itself alone. A single element can only be understood in a broader context, and by taking other informations into account. In many practical applications of XPSs knowledge does not exist at first in a communicable, formal or symbolic way. In most applications, however a more or less large data base of case data that has been previously processed by experts exists. These data are typically in the form of measurements and can be considered subsymbolic data. While XPS are very efficient and capable of dealing with symbolic knowledge (e.g. rules), they are in most cases not able to deal with subsymbolic data. This is, however, an appropriate field for ANNs. An integration of both approaches seems therefore to be advisable. In the Project WINA (Wissensverarbeitung in neuronalen Architekturen) sponsored by the German Ministry of Science and Technology (BMFT) we are studying the integration of ANN with a rule based XPS. In this paper we present briefly some of the results of the WINA project. In Section 2 we identify different types of integration. The following sections present some results of the realization of some integration approaches we have made in the WINA project. Section 3 describes an implementation of the central part of an XPS, the unification algorithm in an ANN. Section 4 shows how ANNs can be used to improve an XPS' performance by learning from experience. Finally in Section 5 we present some results in transforming the learned structures from subsymbolic data in ANN into symbolic rules that can be used by XPS.

2. Methods for the Integration of ANN and XPS

Four different methods to integrate ANN and XPS can be distinguished. We call them "Neural Approximative Reasoning", "Neural Unification", "Introspection" and "Integrated Knowledge Acquisition". Neural Approximative Reasoning: Under the term Neural Approximative Reasoning we understand techniques that make use of ANNs for the XPS's process of drawing approximate conclusions. One very obvious way for this integration type is the realization of a system for approximative reasoning through an ANN. This approach has been presented, for example by Gallant [Gallant 88]. In this approach confidence values are assigned to units/weights. In analogy to a Bayesian-type reasoning process the confidence values are used to derive the probability/plausibility of conclusions. This and many other approaches are, roughly speaking, such that the ideas of probabilistic reasoning (such as Bayesian, Dempster-Shafer, Belief-networks and others) are implemented using an ANN with a local (symbolic) representation of knowledge.

Neural Unification: A different type of integrated reasoning is the realization of an important part of the reasoning process, unification, using ANNs. Unification is the process of making two terms equal. For example, to solve the following equation in the variables X and Y, $f(g(a,X)) = f(g(Y,Y))$, the constant a can be identified with the Variable Y and consequently X with Y. A successful unification yields therefore: $f(g(a,a)) = f(g(a,a))$, with the substitutions: $Y = a$, $Y = X$ and $X = a$. Unification plays a central role in logic programming (e.g. in the language Prolog) and is also a central feature for the implementation of many XPS. The idea of this approach is to realize the matching and unification part of the reasoning process in

447

a suitable ANN (see Section 3).

Introspection: Under Introspection we understand methods and techniques whereby an XPS

observes its own behaviour and improves its performance. This approach can be realized using ANNs that observe the sequence of steps an XPS takes in the derivation of a conclusion. This is often called Control Knowledge. When the observed behaviour of the XPS is appropriately encoded, an ANN can learn how to avoid misleading paths and how to arrive faster at its conclusions (see Section 4).

Integrated Knowledge Acquisition: XPS depend mostly on a human expert to formulate knowledge in symbolic rules. It is next to impossible for an expert to describe his knowledge sufficiently in the form of rules. In particular it is very difficult to describe knowledge acquired by experience. An XPS may therefore not be able to diagnose a case which the expert is able to. The question is, how to extract experience from a set of examples for the use of XPS. Symbolic algorithms to do this have been proposed in the AI discipline called "Machine Learning". Symbolic systems such as, for example, ID3 [Quinlan 85] and version-space [Mitchell 82] are well known examples of this type of algorithms. Under Integrated Knowledge Acquisition we understand subsymbolic approaches, i.e. the usage of ANN, to gain symbolic knowledge. ANN can easily process subsymbolic raw data. This includes all the positive features of ANNs like handling noisy and inconsistent data. An intrinsic property of ANNs is however, that no "knowledge" in the sense defined in Section 1 can be identified in the trained ANN. The central problem for Integrated Knowledge Acquisition is reformat how to transform whatever an ANN has learned into a symbolic form (see Section 5).

3. Neural Unification

Several systems for unification using ANN have been proposed [Ajjanagadde/Shastri 89, Ballard 86, Hölldobler 90, Stolcke 89, Touretzki/Hinton 89]. One of the best known is the approach of [Ballard 86]. In this approach a logical statement is encoded in a relaxation network using dedicated units and special types of links to describe clauses of the statement. Unification (reasoning) is performed by an energy minimization process while updating the state of the ANN. Problems with this system are that recursion is not allowed and certain restrictions on the structure of terms are imposed. For example, a clause may only be used once in an inference process. We have investigated a different approach that is closer to the problem representation proposed by Hölldobler [Hölldobler 90]. The main idea is to use Kohonen Feature Maps (KFM) [Kohonen 84] for the representation of the atoms and functors in a logical statement. KFMs have the property that similar input data are represented in a close neighborhood in the Feature Map. For each atom, respectively functor, in the logical statement a vector as input data for the KFM is generated as follows: each component of the input vector represents the number of occurrences of the given atom/functor in a (sub-)term. The length of the vector is the number of possible (sub-)terms. For each subterm in a logical statement a trained KFM represents the atoms and functors. A special designed relaxation network performs the unification. This network is constructed such that when a unification is performed the corresponding locations in a KFM are activated. Special links ensure that no term is a subterm of itself (occur-check). The unifiability of two terms can be detected in this approach for all subterms in parallel. For details see [Ultsch et al. 92].

448

4. Introspection

Many symbolic knowledge processing systems rely on programs that are able to perform symbolic proofs. Interpreters for the programming language Prolog are examples of such programs. The usage of Prolog interpreters for symbolic proofs, however, implies a certain

proof strategy: first, selection of the leftmost partial goal in the resolvent as goal to be solved and second, selection of the clauses in written order for the resolution of the selected partial goal. In case of failure of a partial goal, the interpreter backtracks systematically to the last choice made without analyzing the cause of failure. Even for simple programs, this implicit control strategy is not sufficient to obtain efficient computations. The formulation of an explicit control strategy in a logical program is, however, contradictory to the declarative programming paradigm. ANN can be used to automatically optimize symbolic proofs without the need of an explicit formulation of Control Knowledge [Ultsch et al 91]. We have realized an approach to learn and store Control Knowledge in an ANN. Input to the ANN is the Prolog clause to be proved. The output is an encoded structural description of the subgoal that is to be proved next. In order to do a comparison we have realized three different ANN for that problem [Ultsch et al 91]: n~71

- ART1 extended to supervised learning mode [Carpenter/Grossberg 87]

- Backpropagation

- Kohonen's self organizing Feature Maps [Kohonen 84].

A meta-interpreter generates training patterns for the ANN. It encodes successful Prolog proofs. Trained with these examples of proofs the ANN generalizes a control strategy to select clauses. Another meta-interpreter, called generating meta interpreter (GMI), is asked to prove a goal. The GMI constructs the optimal proof for the given goal, i.e. the proof with the minimal number of resolutionsteps. The optimal proof is found by generating all possible proofs and comparing them with reference to the number of resolutionsteps. For an optimal proof each clause-selection situation is recorded. A clause-selection-situation is described by the features of the partial goal to be proved and the clause which is selected to solve that particular goal. The clause is described by a unique identification and two different sorts of information concerning the structure of arguments are used: the types of argument and their possible identity.

constant

(atom (~greyd"J~.
empty list) (~

¥ P~ P E3 ART I E~ ICohocn ~ o~.~n~

Figure 4 Resolution steps for different Figure 3: Type tree of Prolog types ANN averaged on all tests

For the types of arguments a hierarchical ordering of the possible argument types, as shown in Figure 3, is used. The encoder takes the clause-selection-situation and produces a training pattern for the ANN. The encoding preserves similarities among the types, e.g. the code for 'integer' is more similar to the code for 'atom' than to the code for 'list'. The ANN is trained with the encoded training patterns until it

inspected by a human expert and added to an XPS. When a case is presented to the XPS, the system first tries to reason with the rules that have been acquired from an expert in order to produce a suitable diagnosis. If this fails to produce a diagnosis, the new rules produced by the process described above can be used. If the case can

Eupert

stem

Figure 5: Integrated Knowledge Acquisition

be handled in such a way, all steps of the reasoning process may be inspected by and explained to a user of the system. The system, however, may not be able to produce a suitable diagnosis in this way. This could be because data is missing, or the input is erroneous. It may also be that no rule fits the data, since the presented case has not been considered while building the knowledge base. In these cases the XPS can turn the question over to the ANN. The ANN, with its ability to associate and generalize, searches for the best matching case that has been learned before. The diagnosis which has been associated with such a case, is then returned as a possible diagnosis.

Figure 6: Kohonen self organizing Feature Map method
Figure 7: Example of a particular U-matrix

One of the ANNs we used was a KFM consisting of two layers (see Figure 6). The input layer has n units representing the n components of a data vector. The other layer called unit layer is a two dimensional array of units arranged on a grid. The number of the units is determined experimentally: Typical sizes of unit layers are 64×64 128×128 or 256×256 . Each unit in the input layer is connected to every unit in the unit layer with a weighted link. The weights are initialized randomly in the

450

is able to reproduce the choice of a clause for a partial goal. A query is passed to an optimizing meta-interpreter (OMI). For each partial goal the OMI presents the description of the partial goal as input to the ANN and obtains a candidate-clause for resolution. With this candidate the resolution is attempted. If resolution fails, the OMI uses Prolog search strategy as default. Our system allows to generalize the learned Control Knowledge to new programs. In order to do this, structural similarities between the new program and the learned one are used to generate a mapping of the corresponding selection situations of different programs. We have tested our approach using several different Prolog programs, for example programs for map coloring, travelling salesman, symbolic differentiation and a small XPS. The following figure depicts the obtained results. The percent average numbers of Resolution steps for a proof of different programs are given. As 100% the number of resolution steps needed by a Prolog interpreter for the given program is used. The results in Figure 4 called "trained" are obtained with programs the ANNs were trained with. For queries using the same program, but that were not used as training data, the results are termed "untrained" in Figure 4. The "generalized" results are for completely new programs.

As turned out almost all ANNs were in principle able to learn a proof strategy. Best results in reproducing learned strategies were obtained with the modified ART1 network which reproduced the optimal number of resolutions for a known proof. For queries of the same type (same program) that were not used as training data, however, the KFMs turned out to be the best. A proof strategy using this ANN averaged slightly over the optimal number of resolutions even for completely new programs but well below the number of resolutions a Prolog interpreter needs. The Backpropagation network we used was the worst for both cases. This may be due to a architectural problem for very large BP networks. Different architectures instead of the 3-layer type could ease that problem. The networks we used had several thousands of units and took a considerable time to converge to a satisfactory state.

5. Knowledge Acquisition

In our project WINA we are realizing a system that uses ANNs for the automatic acquisition of knowledge out of a set of examples. Our system enhances the reasoning capabilities of classical XPS with the ability to generalize and the handling of incomplete cases. It uses ANN with unsupervised learning algorithms to extract regularities out of case data. A symbolic rule generator transforms these regularities into PROLOG rules. The generated rules and the trained ANN are embedded into the XPS as knowledge bases. In the system's diagnosis phase it is possible to use these knowledge bases together with human experts' knowledge bases in order to diagnose an unknown case. Furthermore the system is able to process and diagnose inconsistent data using the trained ANN, exploiting their ability to generalize. Figure 5 gives an overview of parts of our system. Case data that are presented to an XPS are usually stored in a case data base. A Data Transformation module encodes such cases in a suitable way in order to be learned by ANNs. This module performs several tasks: first it transforms the data so that the components have equal scopes. One of the possibilities is the use of a standardisation. The second task of the transformation module is to encode the data into a input pattern for ANNs. This could mean the recoding of the data, for

example using FFT or other algorithms. With the so transformed data different ANNs with unsupervised learning algorithms are trained. These ANNs have the ability to adapt their internal structures (weights) to the structure of the data. In a Rule Generation module the structures learned by the ANNs are detected, examined and transformed into XPS rules. These rules can be

451

range of the smallest and the greatest value of each component of all cases. They are adjusted according to Kohonen's learning rule [Kohonen 84]. The applied rule uses the Euclidean distance and a simulated mexican-hat function to realize lateral inhibition. In the unit layer neighboring units form regions, which correspond to similar input vectors. The KFM are implemented on a transputer system connected to a SUN workstation [Guimaraes/Korus 92]. The automatic detection of classes using KFMs, however, is difficult because the Kohonen algorithm converges to an equal distribution of the units corresponding to a case in the output layer. A special algorithm, the so called U-matrix method was developed in order to detect classes that are in the data [Ultsch/Siemon 90, Ultsch 92]. The idea of the U-matrix method is to make use of the Feature Map's topology. At each point of the unit layer the weights are analyzed with respect to the neighbouring weights. A very simple, but very effective way to do this is to display the distance between two neighbour units as height (see Figure 7). We have tested U-matrix methods with several examples stemming from different areas of application such as medicine, meteorology, process control, environmental protection and statistics. In the following figures we see a U-matrix for a data set containing blood analysis values from 20 patients (20 vectors with 11 realvalued components) selected from a set of 1500 patients [Deichsel/ Trampisch 85]. In the U-matrix of Figure 8 three major classes can be distinguished: the upper right corner the, left part, and the lower right part. The latter two may be subdivided further into two subgroups each. It turned out that this clustering corresponds nicely with the different patient's diagnoses, as Figure 9 shows.

O ,so

_ ~ _ 11

Figure 8: U-Matrix of acidosis data Figure 9: U-matrix with diagnoses

```

~hie~se Y pu~ients c(II Tec~ | -.vrom~
lin~mo~e~ o~e~
normochr. _~
hvr~hrome I ~ I ~ I I
Eisenm; noel _~ '~
.ch~erE~I 16 ~ 16 T ~)
chron.PTozeL~ I I | ~)
idero~chr. L
hv~erchr. i I I
Polv~llohul I)
o.B. ~

~um I l~l I 11'~ 1 .l

```

Figure 10: sig*-generated diagnoses vs. true diagnoses

In summary, by using this method, structure in the data can be detected as classes. These classes represent sets of data that have something in common. Having detected

452

classes in the data, the next step is to extract symbolic rules from the ANNs. As a first approach to generate rules from the classified data we have tested a well known machine learning algorithm: ID3 [Quinlan 85, Ultsch/Panda 91]. While being able to generate rules this algorithm has a serious problem [Ultsch 91]: it uses a minimalization criterion that seems to be unnatural for a human expert. Rules are generated that use only a minimal set of decisions to come to a conclusion. This is not what must be done, for example, in a medical domain. Here the number of decisions is based on the type of the disease. In simple cases, i.e. where the symptoms are clear, very few tests are made, while in difficult cases a diagnosis must be based on all available information. In order to solve this problem, we have developed a rule generation algorithm, called sig*, that takes the significance of a symptom (range of a component value) into account (for details see [Ultsch 91]). One of the data sets we tested the algorithm with was the diagnosis of iron deficiency. We had a test set of 242 patients with 11 clinical test values each. The rules generated with sig* showed, first, a high degree of coincidence with expert's diagnosis rules and, second, exhibited knowledge not previously known to us while making sense to the experts [Ultsch 91]. We have tested this algorithm in another way: the data set was randomly divided into two subsets and only one of them was used to generate rules. The other subset was used to compare the diagnosis generated by the rules with the actual disease. In 119 out of 121 cases (98%) the system produced the right diagnoses. In eleven cases a wrong diagnosis was given. Most of these cases were additional wrong diagnoses. For example in six cases healthy patients (diagnoses "o.B.") were also considered to have a slight form of iron deficiency. In other cases, like "sidero-acrestic", the data set that had been learned by the ANN was too small (one case) in order to produce a meaningful diagnostic rule.

6. Conclusion

The methods of integrating ANNs with XPSs can be classified into four different approaches: Neural Approximative Reasoning, Neural Unification, Introspection and Integrated

Knowledge Acquisition. Neural Approximative Reasoning is the usage of ANNs for approximative reasoning; Neural Unification means to implement symbolic unification with ANNs resulting in an algorithm that is both efficient (parallel execution) and has the ability to do similarity-based "fuzzy" unification; Introspection is the observation of an XPS' internal behaviour using ANNs with the aim to improve the system's performance; Integrated Knowledge Acquisition uses subsymbolic knowledge sources (raw data or measurements) as training data for ANNs in order to transform the structural features recognized by ANNs into symbolic knowledge. Concerning Neural Unification we have studied a neural unification algorithm using Kohonen Feature Maps. This neural unification algorithm is capable of doing the ordinary unification with ANNs whereby important problems like the occurcheck and the calculation of a most common unifier can be done in parallel [Ultsch et al. 92]. In Introspection we have tested several different ANNs for their ability to detect and learn proof strategies. A modified Feature Map has been identified to yield best results concerning the reproduction of proofs made before and for generalizing to completely new programs [Ultsch et al. 91]. In the field of Integrated Knowledge Acquisition we have developed methods to detect structures that an ANN has learned from raw data. Data stemming from measurements with typically high dimensionality can be analyzed by using an apt visualisation of an ANN's weights (U-matrix methods) [Ultsch 92]. The detected structures can be reformulated in the form of symbolic rules. This transformation has been realized with a machine learning algorithm (ID3). The usage of this algorithm, however, seemed not to be optimal for the

purpose of rule generation for XPS at least in the domain of diagnosis. ID3 minimizes the number of decisions necessary to arrive at a diagnosis. This is, however, counterintuitive for human experts, where the number of decisions (rule conditions) is proportional to the significance of the decisions. We have developed an algorithm, called sig* to overcome these problems. Up to now sig* has been successful on a 9 level for generating meaningful rules in several different problem domains [Ultsch 91, 92]. First results of the WINA project demonstrate the usefulness of the combination of a symbolic knowledge processing system with ANN. Algorithms for Neural Unification allow for an efficient realization of the central part of a symbolic knowledge processing system and may also be used for Neural Approximative Reasoning. Introspection with ANN frees the user and programmer of knowledge processing systems to formulate Control Knowledge explicitly. As the system gains more and more experience with the way it is used its performance improves gradually. The usage of ANN for integrated subsymbolic and symbolic knowledge acquisition realizes a new type of learning from examples. Unsupervised learning ANNs are capable of extracting regularities from data. Due to the distributed subsymbolic representation, ANNs are, however, not able to explain their inferences. Our system avoids this disadvantage by extracting symbolic rules out of the ANN. The acquired rules can be used like the expert's rules. In particular it is possible to give an explanation of the inferences made by the ANNs. By exploiting the properties of the ANNs the system is also able to effectively handle noisy and incomplete data.

Acknowledgements

Gabriella Guimaraes, Heng Li and Dieter Korus presently work fulltime for the project WINA and have contributed among many other assistants and students of the University of Dortmund to the results presented in this paper. This work has been supported in part by the project WINA (Wissensverarbeitung in neuronalen Architekturen) sponsored by the German Ministry of Research and Technology (BMFT) under contract No. 413-5839-01 IN 103 C/3 and in part by the Bennisen-Foerder Forschungspreis Nordrhein-Westfalen granted to the author.

References:

AJJANAGADE, V., SHASTRI, L.: Efficient inference with multiple predicates and vari

ables in a connectionist systems, Proc. Ann. Conf. Cognitive Science Soc., 1989, pp 396-403.

BALLARD, D.H.: Parallel logic inference and energy minimization. Proc. AAAI, Philadelphia, pp 203-208, 1986.

CARPENTER, G.A., GROSSBERG, S.: Self-Organization of Stable Category Recognition Codes for Analog Input Patterns, Applied Optics, Vol. 26, S. 4.919-4.930, 1987.

DEICHSEL G., TRAMPISCH H.J.: Clusteranalyse u. Diskriminanzanalyse, Fisher Verl., Stuttgart 1985.

GALLANT, S.I.: Connectionist Expert Systems, Communications of the ACM, Vol. 31, 1988, pp 152 - 169.

GUIMARAES, G. KORUS, D.: Neuronale Netze auf l~ansputern, Proc. TAT, Aachen, 09/1992, pp. 75-76.

HÖLLDOBLER, S.: On high level inferencing and the variable binding problem in connectionist networks, in Dörfner, G.(ed.): Konnektionismus in Artificial Intelligence und Kognitionsforschung, Springer 1990, pp. 180-185.

QUINLAN, J.R.: Learning Efficient Classific. Procedures and their Application to Chess

454

K~d Games, i~: Ru~elh~t D~., Xipser D.: ~at~e disc. by Co~petitive Le~ Co6~itve Scie~ce
lg85 ~1. g, pp. 75- 112.

KO~ONKN, T~ Sel~Or6a~isatio~ a-d Associati~ ~e~or~, Spri~6er ~rla6, ~er~n, lg84.

~ITC~LL, T.~ Genera~ation ~s Se~ch, in: Arti~cial InteUi6ence 18, pp. 203-226.

STOLCKE, A~ ~ni~c. as constrai~ satisfi~ction in struc. connectionist net~orks, Neural Co~putation, 4, lg8g.

TOVR~T%KI, D.S~ ~INTON, G.E.: A d~tributed connectionist production syste~, Co6 n~ive Science 12, lg8g, pp. 423-466.

~SC~, A., et aL: Opti~in6 1O6ic~1 proo~ ~h connectionist networks, in Kohoneu et aL (eds.~ Arti~cial Ne~al Net~orks, Elsevier lggl, pp. 585-Sg0.

U~SC~, A., GVI~ARAES, G., WEBER, V.: SelE Or6anisinE ~ature ~aps ~r Logical Vni~cation, Univers~y of Dort~nd, sub~itted to IJCAI lgg3, AlbertviUe.

U~SC~, A., PANDA, PG.: Die Kopplun6 konnektionistischer ~odelle ~it ~issensbasiert~ Systemen, ~6un6sband Experte~e~6e Dort~nd, ~bru~ lggl, VDI ~a6, pp. 74 g4.

U~SC~, A~ SIE~ON, H.P~ Kohonen's Se~ Or6ani~in6 ~at~e ~aps ~r Exploratory Data Analys~, Proc. I~ern. Ne~al Net~orks, Kl~er Acade~ic Press, Parb, lgg0, pp. 305-308.

U~SC~, A.: Konnektio~ist~che ~odeUe und ihre Inte6ration ~ ~issensbasierten Systeme~n, ~esearch Report Nr 3g6, Dep~t~e~ of Co~puter Science Univ. Dort~nd, ~bruary lggl.

U~SC~, A.: Sel~Or6ani~. Neural Net~orks ~r Kno~led6e Ac~isition, Proc. ECAI, W~n, lgg2, pp. 208-210.