Dependency networks based classifiers: learning models by using independence tests.

José A. Gámez and Juan L. Mateo and José M. Puerta
Computing Systems Department
Intelligent Systems and Data Mining Group – $i^3\mathcal{A}$
University of Castilla-La Mancha
Albacete, 02071, Spain

Abstract

In this paper we propose using dependency networks (Heckerman et al., 2000), that is a probabilistic graphical model similar to Bayesian networks, to model classifiers. The main difference between these two models is that in dependency networks cycles are allowed, and this fact has the consequence that the automatic learning process is much easier and can be parallelized. These properties make dependency networks a valuable model especially when it is needed to deal with large databases. Because of these promising characteristics we analyse the usefulness of dependency networks-based Bayesian classifiers. We present an approach that uses independence tests based on chi square distribution, in order to find relationships between predictive variables. We show that this algorithm is as good as some state-of the-art Bayesian classifiers, like TAN and an implementation of the BAN model, and has, in addition, other interesting proprierties like scalability or good quality for visualizing relationships.

1 Introduction

Classification is basically the task of assigning a label to an instance formed by a set of attributes. The automatic approach for this task is one of the main parts in machine learning. Thus, the problem consists on learning classifiers from datasets in which each instance has been previously labelled. Many methods have been used for solving this problem based on various representations such as decision trees, neural networks, rule based systems, support vector machines among others. However a probabilistic approach can be used for this purpose, Bayesian classifiers (Duda and Hart, 1973; Friedman et al., 1997). Bayesian classifiers make use of a probabilistic graphical model such as Bayesian networks, that have been a very popular way of representing probabilistic relationships between variables in a domain. Thus a Bayesian classifier takes advantage of probability theory, especially the Bayes rule, in conjunction with a graphical representation for qualitative knowledge. The classification task can be expressed as determining the probability for the class variable knowing the value for all other variables:

$$P(C|X_1 = x_1, X_2 = x_2, \cdots, X_n = x_n).$$

After this computation, the class variable's value with the highest probability is returned as result. Obviously computing this probability distribution for the class variable can be very difficult and inefficient if it is carried out directly, but based on the independencies stated by the selected model this problem can be simplified enormously.

In this paper we present a new algorithm for the automatic learning of Bayesian classifiers from data, but using *dependency network classifiers* instead of Bayesian networks. Dependency networks (DNs) were proposed by Heckerman et al. (2000) as a new probabilistic graphical model similar to BNs, but with a key difference: the graph that encodes the model structure does not have to be acyclic. As in BNs each node in a DN has a conditional probability distribution given its parents in the graph. Although the presence of cycles can be observed as the capability to represent richer models, the price we have to pay is that usual BNs inference algorithms cannot be applied and Gibbs sampling has to be used in order to recover the joint probability distribution (see (Heckerman et al., 2000)). An initial approach for introducing dependency networks in automatic classification can be found in (Mateo, 2006).

Learning a DN from data is easier than learning a BN, especially because restrictions about introducing cycles are not taken into account. In (Heckerman et al., 2000) was presented a method for learning DNs based on learning the conditional probability distribution for each node independently, by using any classification or regression algorithm to discover the parents of each node. Also a feature selection scheme can be use in conjunction. It is clear that this learning approach produces scalable algorithms, because any learning algorithm with this approach can be easily parallelized just distributing groups of variables between all nodes available in a multi-computer system.

This paper is organized as follows. In section 2 dependency networks are briefly described. In section 3 we review some notes about how classification task is done with probabilistic graphical models. In section 4 we describe the algorithm to learn dependency networks classifiers from data by using independence tests. In section 5 we show the experimental results for this algorithm and compare them with state-of-the-art BN classifiers, like TAN algorithm and an implementation of the BAN model. In section 6 we present our conclusions and some open research lines for the future.

2 Preliminaries

A Bayesian network (Jensen, 2001) $\mathcal{B}$ over the domain $\mathbf{X} = \{X_1, X_2, \cdots, X_n\}$, can be defined as a tuple ($\mathbb{G}$,$\mathbf{P}$) where $\mathbb{G}$ is a directed acyclic graph, and $\mathbf{P}$ is a joint probability distribution. Due to the conditional independence constraints encoded in the model, each variable is independent of its non-descendants given its parents. Thus, P can be factorized as follows:

$$P(\mathbf{X}) = \prod_{i=1}^{n} P(X_i|Pa_i) \qquad (1)$$

A dependency network is similar to a BN, but the former can have directed cycles in its associated graph. In this model, based on its conditional independence constraints, each variable is independent of *all other variables* given its parents. A DN $\mathcal{D}$ can be represented by ($\mathbb{G}$,$\mathbf{P}$) where $\mathbb{G}$ is a directed graph not necessarily acyclic, and as in a BN $\mathbf{P} = \{\forall i, P(X_i|\mathbf{Pa}_i)\}$ is the set of local probability distributions, one for each variable. In (Heckerman et al., 2000) it is shown that inference over DNs, recovering joint probability distribution of $\mathbf{X}$ from the local probability distributions, is done by means of Gibbs sampling instead of the traditional inference algorithms used for BNs due to the potential existence of cycles.

A dependency network is said *consistent* if $P(\mathbf{X})$ can be obtained from $\mathbf{P}$ via its factorization from the graph (equation 1). By this definition, can be shown that exists an equivalence between a consistent DN and a Markov network. This fact suggests an approach for learning DNs from Markov network learned from data. The problem with this approach is that can be computationally inefficient in many cases.

Due to this inconvenient, consistent DNs became very restrictive, so in (Heckerman et al., 2000) the authors defined *general* DNs in which consistency about joint probability distribution is not required. General DNs are interesting for automatic learning due to the fact that each local probability distribution can be learned independently, although this local way of processing can lead to inconsistencies in the joint probability distribution. Furthermore, structural inconsistencies ($X_i \in pa(X_j)$ but not $X_j \in pa(X_i)$) can be found in general DNs. Nonetheless, in (Heckerman et al., 2000) the authors argued that these inconsistencies can disappear, or at least be reduced to the minimum when a reasonably large dataset is used to learn from. In this work we only consider general DNs.

An important concept in probabilistic graphical models is *Markov blanket*. This concept is defined for any variable as the set of variables which made it independent from the rest of variables in the graph. In a BN the Markov blanket is formed by the set of parents, children and parent of the children for every variable. However, in a DN, the Markov blanket for a variable is exactly the parents set. That is the main reason why DN are useful for visualizing relationships among variables, because these relationships are explicitly shown.

3 Probabilistic graphical models in classification

As it has been said before, we can use Bayesian models in order to perform a classification task. Within this area, the more classical Bayesian classifier is the one called *naive* (Duda and Hart, 1973; Langley et al., 1992). This model considers all predictive attributes independent given the class variable. It has been shown that this classifier is one of the most effective and competitive in spite of its very restrictive model since it does not allow dependencies between predictive variables. However this model is used in some applications and is employed as initial point for other Bayesian classifiers. In the augmented naive bayes family models, the learning algorithm begins with the structure proposed by naive Bayesian classifier and then try to find dependencies among variables using a variety of strategies. One of this strategies can be looking for relationships between predictive variables represented by a simple structure like a tree as is done in TAN (Tree Augmented Naive Bayes) algorithm (Friedman et al., 1997). Another is to adapt a learning algorithm employed in machine learning with general Bayesian networks. In this case the learning method is modified in order to take into account the relationship between every variable and the class.

Apart from this approach to learn Bayesian classifiers, it must be mentioned another one based on the Markov blanket concept. These methods focus their efforts in searching the set of variables that will be the class variable's Markov blanket. Nonetheless, it has been shown that methods based on extending Naive Bayes classifier, in general, outperform the ones based on the Markov blanket.

When the classifier model has been learned is time to use inference in order to find what class value has to be assigned for the values of a given instance. To accomplish this task the MAP (*Maximum at posterior*) hypothesis is used, which returns as result the value for the class variable that maximize the probability given the values of predictive variables. This is represented by the following expression:

$$\begin{aligned} c_{MAP} &= \arg\max_{c \in \Omega_C} \quad p(c|x_1, \ldots, x_n) \\ &= \arg\max_{c \in \Omega_C} \quad \frac{p(x_1, \ldots, x_n|c) \cdot p(c)}{p(x_1, \ldots, x_n)} \\ &= \arg\max_{c \in \Omega_C} \quad p(c, x_1, \ldots, x_n) \end{aligned}$$

Thus, the classification problem is reduced to compute joint probability for every value of the class variable with the values for the predictive variables given by the instance to by classified.

When we deal with a classifier based on a Bayesian network we can use joint probability factorization shown in equation 1 in order to recover these probabilities. Initially inference with dependency networks must be done by means of Gibbs sampling, but due to the fact that we focus on classification, and under the assumption of complete data (i.e. no missing data neither in the training set nor in the test set), we can avoid Gibbs sampling. This result comes from the fact that we can use *modified ordered Gibbs sampling* as designed in (Heckerman et al., 2000). This way of doing inference over dependency networks is based on decomposing the inference task into a set of inference tasks on single variables. For example, if we consider a simple classifier based on dependency network with two predictive variables, which show dependency between them, so with the following graphical structure: $X_1 \longleftrightarrow X_2; C \longrightarrow X_1; C \longrightarrow X_2$; its probability model will be:

$$P(C, X_1, X_2) = P(C)P(X_1|C, X_2)P(X_2|C, X_1).$$

With this method joint probability can be obtained by computing each term separately. The determination of the first term does not requires Gibbs sampling. The second and third terms should be determined by using Gibbs sampling but in this case we know all values for their conditioning variables, so they can be determined directly too by looking at their conditional probability tables. If we assume that the class variable will be determined before other variables (as it is the case), by this procedure we can compute any joint probability avoiding the Gibbs sampling process.

4 Finding dependencies

In this section we present our algorithm to build a classifier model from data based on dependency networks. Our idea is to use independence tests in order to find dependencies between predictive variables X_i and X_j, but taking into account the class variable, and in this way extend the Naive Bayes structure. It is known that statistic

$$2 \cdot Nc \cdot I(X_i; X_j | C = c)$$

follows a χ^2 distribution with $(|X_i|-1)\cdot(|X_j|-1)$ degrees of freedom under the null hypothesis of independence, where Nc is the number of input instances in which $C = c$ and $I(X_i; X_j|C = c)$ is the mutual information between variables X_i and X_j when the class variable C is equal to c. Using this expression, for any variable can be found all other variables (in)dependent given the class.

But it is not this set what we are looking for, we need a set of variables which made that variable independent from the other, that is, its Markov blanket. So, we try to discover this set by carrying out an iterative process, for every predictive variable independently, in which in each step is selected as a new parent the variable not still chosen which shows more dependency with the variable under study (X_i). This selection is done considering all the parents previously chosen. Thus the statistic used actually is

$$2 \cdot Nc \cdot I(X_i; X_j | C = c, \mathbf{Pa}_i),$$

where $\mathbf{Pa}_i$ is the current set of X_i's parents minus C. We assess the goodness of every candidate parent by the difference between the percentile of the χ^2 distribution (with $\alpha = 0.025$) and the statistic value. Here the degrees of freedom must be

$$df = (|X_i| - 1) \cdot (|X_j| - 1) \prod_{Pa_i \in \mathbf{Pa}_i} (|Pa_i|).$$

Obviously this process finishes when all candidate parents not selected yet shows independence with X_i according to this test, and in order to make search more efficient, every candidate parent that appear independent in any step in the algorithm, is rejected and is not taken into account in next iterations. The reason of rejecting these variables is just for efficiency, although we know that any of these rejected variables can become dependent in posterior interations.

Figure 1 shows the pseudo-code of this algorithm called ChiSqDN.

```
Initialize structure to Naive Bayes

For each variable $X_i$
  $\mathbf{Pa}_i = \emptyset$
  $\mathbf{Cand} = \mathbf{X} \setminus \{X_i\}$ // Candidate parents
  While $\mathbf{Cand} \neq \emptyset$
    For each variable $X_j \in \mathbf{Cand}$
      Let $val$ be assessment for $X_j$ by $\chi^2$
          test
      If $val \leq 0$ Then
        Remove $X_j$ from $\mathbf{Cand}$

    Let $X_{max}$ be the best variable found
    $\mathbf{Pa}_i = \mathbf{Pa}_i \cup X_{max}$
    Remove $X_{max}$ from $\mathbf{Cand}$

  For each variable $Pa_i \in \mathbf{Pa}_i$
    Make new link $Pa_i \rightarrow X_i$
```

Figure 1: Pseudo-code for ChiSqDN algorithm.

It is clear that determining degrees of freedom is critical in order to perform properly this test. The scheme outlined before is the general way for assessing this parameter, nonetheless, in some cases, especially for deterministic or almost deterministic relationships, the tests carried out using degrees of freedom in this way can fail. Deterministic relationships are pretty common in automatic learning when input cases are relatively few. Examining the datasets used in the experiments (table 1) can be seen that there are some of them with a low number of instances. So, to make the tests in our algorithm properly, we use a method for computing degrees of freedom and handling this kind of relationships proposed in (Yilmaz et al., 2002). Basically and considering discrete variables and its conditional probability tables (CPT), this proposal reduces degrees of freedom based on the level of determinism and this level is determined by the number of columns or rows which are full of zeros.

For example, if we wanted to compute degrees of freedom for variables X_i and X_j whose probability distribution is shown in figure 2, we should use $(3 - 1) \cdot (3 - 1) = 4$ as value for this parameter, as both variables have 3 states. Nonetheless, is clear that the relation between these two variables is not probabilistic but deterministic, because is similar to consider that variable X_i has only one state. Thus, if we use this new way for computing degrees of freedom we have to take into account that this table has two rows and a column full of zeros. Then the degrees of freedom will be $(3 - 2 - 1) \cdot (3 - 1 - 1) = 0$.

	X_j			MT
X_i	13	16	0	29
	0	0	0	0
	0	0	0	0
MT	13	16	0	

Figure 2: Probability table for variables X_i and X_j.

In these cases, i. e, when the degrees of freedom are zero, it does not make sense perform the test because in this case it always holds. In (Yilmaz et al., 2002), the authors propose skipping the test and mark the relation in order to be handled by human experts. In our case, when this algorithm yields degrees of freedom equal to zero we prefer an automatic approach that is to accept dependence between tested variables if statistic value is greater than zero.

5 Experimental results

In order to evaluate our proposed algorithm and to measure their quality, we have selected a set of datasets from the UCI repository (Newman et al., 1998). These datasets are described in table 1. We have preprocessed these datasets in order to remove missing values and to discretize continuous variables. Missing values for each variable have been replaced by the mean or mode depending on whether it is a continuous o discrete variable respectively. For discretization we have used the algorithm proposed in (Fayyad and Irani, 1993), which is especially suggested for classification.

Datasets	inst.	attrib.	$\lvert C \rvert$	cont.?	missing?
australian	690	15	2	yes	no
heart	270	14	2	yes	no
hepatitis	155	20	2	yes	yes
iris	150	5	3	yes	no
lung	32	57	3	no	yes
pima	768	9	2	yes	no
post-op	90	9	3	yes	yes
segment	2310	20	7	yes	no
soybean	683	36	19	no	yes
vehicle	846	19	4	yes	no
vote	435	17	2	no	yes

Table 1: Description of the datasets used in the experiments.

We have chosen a pair of algorithms from the state-of-the-art in Bayesian classification which allow relationships between predictive variables. One of this algorithms is the one known as TAN (Tree Augmented Naive Bayes). The other employs a general Bayesian network in order to represent relationships between variables and is based on BAN model (Bayesian network Augmented Naive Bayes (Friedman et al., 1997)). In our experiments we use B algorithm (Buntine, 1994) as learning algorithm with BIC (Schwarz, 1978) metric to guide the search. Both algorithms use mutual information as base statistic in order to determine relationships, directly in TAN algorithm and a penalized version in BIC metric.

The experimentation process consists on running each algorithm for each database in a 5x2 cross validation. This kind of validation is an extension of the well known k-fold cross validation in which a 2-fold cv is performed five times and in any repetition a different partition is used. With this process we have the validation power from the 10-fold cv but with less correlation between each result (Dietterich, 1998).

Once we have performed our experiments in that way, we have the results for each classifier in table 2.

	TAN	BAN+BIC	ChiSqDN
australian	0.838841	**0.866957**	0.843478
heart	0.819259	**0.826667**	0.823704
hepatitis	0.852930	**0.872311**	0.865801
iris	0.948000	**0.962667**	**0.962667**
lung	**0.525000**	**0.525000**	**0.525000**
pima	0.772135	**0.773177**	0.744531
post_op	0.653333	**0.673333**	**0.673333**
segment	**0.936970**	0.909437	0.931169
soybean	0.898661	**0.906893**	0.903950
vehicle	**0.719385**	0.697163	0.698345
vote	**0.945299**	0.944840	0.931045

Table 2: Classification accuracy estimated for algorithms tested.

5.1 Analysis of the results

The analysis done for the previous results consists on carring out a statistical test. The test selected is the paired Wilcoxon signed ranks, that is a non parametric test which examines two samples in order to decide whether the differences shown by them can be assumed as not significant. In our case, if this test holds, then we can conclude that the classifiers which yield the results analysed have similar classification power.

Thus, this test is done with a level of significance of 0.05 ($\alpha = 0.05$) and it shows that both reference classifiers do not differ significantly from our classifier learned with ChiSqDN algorithm. When we compare our algorithm with TAN we obtain a p-value of 0.9188, and for the comparison with BAN model the p-value is 0.1415. Therefore ChiSqDN algorithm is as good as the Bayesian classifier considered, in spite of the approximation introduced in the factorization of the joint probability distribution due to the use of general dependency networks. Nonetheless, also as consequence of the use of dependency networks we can take advantage of the ease of learning and possibility of doing this process in parallel without introduce an extra workload.

Apart from these characteristics that can be exploited from a dependency network, we can focus on its descriptive capability, i.e. how good is the graph for each model to show relations over the domain. Taking into account pima dataset, in figures 3 and 4 are shown the graphs yielded by the algorithms BAN+BIC and ChiSqDN respectively.

Obviously, for every variable in these graphs, those that form its Markov blanket are the more informative or more related. The difference is that for a Bayesian network graph the Markov blanket set is formed by parents, children and parent of children for every variable, but for a dependency network graph the Markov blanket set are all variables directly connected. Thus in tables 3 and 4 we show the Markov blanket extracted from the corresponding graph for every variable. Evidently the sets do not match due to the automatic learning process but are quite similar. Apart from that, it is clear that discovering these relationships from the dependency network graph is easier, especially for people not used to deal with Bayesian networks, than doing it from a Bayesian network. We think that this point is an important feature of dependency networks because can lead us to a tuning process helped by human experts after the automatic learning.

Variable	Markov blanket
mass	skin, class
skin	age, mass, insu, class
insu	skin, plas, class
plas	insu, class
age	pres, preg, skin, class
pres	age, class
preg	age, class
pedi	class

Table 3: Markov blanket for each predictive variable from the BAN+BIC classifier.

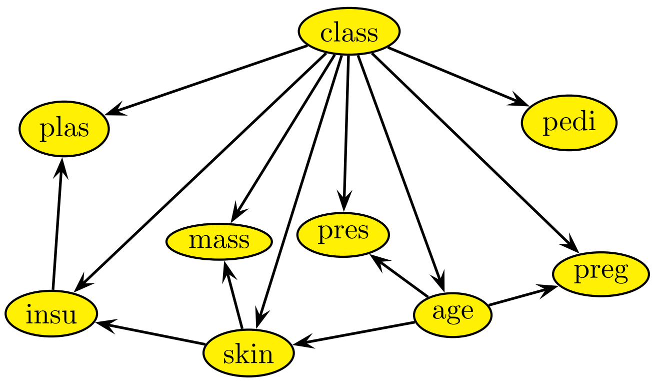

Figure 3: Network yielded by BAN algorithm with BIC metric for the pima dataset.

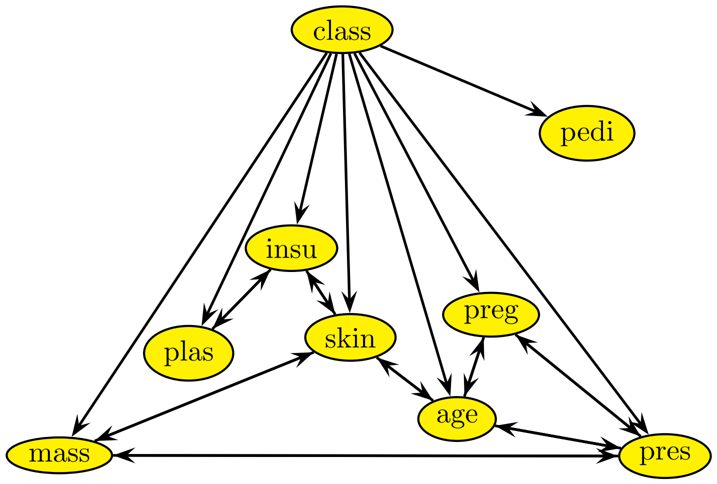

Figure 4: Network yielded by ChiSqDN algorithm for the pima dataset.

Variable	Markov blanket
mass	skin, pres, class
skin	insu, mass, age, class
insu	plas, skin, class
plas	insu, class
age	pres, preg, skin, class
pres	age, mass, preg, class
preg	age, pres, class
pedi	class

Table 4: Markov blanket for each predictive variable from the ChiSqDN classifier.

6 Conclusions and Future Work

We have shown how dependency networks can be used as a model for Bayesian classifiers beyond their initial purpose. We have presented a new learning algorithm which use independence tests in order to find the structural model for a classifier based on dependency networks. By means of the analysis developed in this paper, we have shown that the proposed classifier model based on dependency networks can be as good as some classifiers based on Bayesian networks. Besides this important feature, classifiers obtained by our algorithm are visually easier to understand. We mean that relationships automatically learned by the algorithm are easily understood if the classifier is modelled with a dependency network. This characteristic is especially valuable because for people is often difficult discover all relationships inside a Bayesian networks if they do not know their theory, however these relationships are clearly shown in a dependency network. Besides, we can not forget the main contributions of dependency networks: ease of learning and possibility to perform this learning in parallel.

The ChiSqDN algorithm shown here learn structural information for every variable independently, so we plan to study a parallelized version in the future. Also we will try to improve this algorithm, in terms of complexity, by using simpler statistics. Our idea is to use an approximation for the mutual information over multiples variables using a decomposition in simpler terms (Roure Alcobé, 2004).

Acknowledgments

This work has been partially supported by Spanish Ministerio de Ciencia y Tecnología, Junta de Comunidades de Castilla-La Mancha and European Social Fund under projects TIN2004-06204-C03-03 and PBI-05-022.

References

Wray L. Buntine. 1994. Operations for Learning with Graphical Models. *Journal of Artificial Intelligence Research*, 2:159–225.

Thomas G. Dietterich. 1998. Approximate statistical test for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923.

R. O. Duda and P. E. Hart. 1973. *Pattern classification and scene analisys*. John Wiley and Sons.

U. Fayyad and K. Irani. 1993. Multi-interval discretization of continuous-valued attributes for classification learning. In Morgan and Kaufmann, editors, *International Joint conference on Artificial Intellegence (IJCAI)*, pages 1022–1029.

Nir Friedman, Dan Geiger, and Moises Goldszmidt. 1997. Bayesian Network Classifiers. *Machine Learning*, 29(2-3):131–163.

David Heckerman, David Maxwell Chickering, Christopher Meek, Robert Rounthwaite, and Carl Kadie. 2000. Dependency Networks for Inference, Collaborative Filtering, and Data Visualization. *Journal of Machine Learning Research*, 1:49–75.

Finn V. Jensen. 2001. *Bayesian networks and decision graphs*. Springer.

P. Langley, W. Iba, and K. Thompson. 1992. An Analisys of bayesian Classifiers. In *Proceedings of the 10th National Conference on Artificial Intelligence*, pages 223–228.

Juan L. Mateo. 2006. Dependency networks: Use in automatic classification and Estimation of Distribution Algorithms (in spanish). University of Castilla-La Mancha.

D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. 1998. UCI Repository of machine learning databases. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

Josep Roure Alcobé. 2004. An approximated MDL score for learning Bayesian networks. In *Proceedings of the Second European Workshop on Probabilistic Graphical Models 2004 (PGM'04)*.

Gideon Schwarz. 1978. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464.

Yusuf Kenan Yilmaz, Ethem Alpaydin, Levent Akin, and Taner Bilgi. 2002. Handling of Deterministic Relationships in Constraint-based Causal Discovery. In *First European Workshop on Probabilistic Graphical Models (PGPM02)*, pages 204–211.