Noisy-or classifier[*]

Jiří Vomlel
Laboratory for Intelligent Systems
University of Economics,
Prague, Czech Republic

Institute of Information Theory and Automation
Academy of Sciences of the Czech Republic
Prague, Czech Republic
`vomlel@utia.cas.cz`

Abstract

We discuss an application of a family of Bayesian network models – known as models of independence of causal influence (ICI) – to classification tasks with large numbers of attributes. An example of such a task is categorization of text documents, where attributes are single words from the documents. The key that enabled application of the ICI models is their compact representation using a hidden variable. We address the issue of learning these classifiers by an computationally efficient implementation of the EM-algorithm. We pay special attention to the noisy-or model – probably the best known example of an ICI model. The classification using the noisy-or model corresponds to a statistical method known as logistic discrimination. We describe the correspondence. Tests of the noisy-or classifier on the Reuters dataset show that, despite its simplicity, it has a competitive performance.

1 Introduction

Automatic classification is one of the basic tasks in the area of artificial intelligence. A classifier is a function that assigns instances represented by attributes to a class. A number of different approaches were used to solve this problem: decision trees, neural networks, support vector machines, etc. Bayesian network classifiers is a group of classifiers that use a Bayesian network – a probabilistic model – to represent relations between attributes and classes. A good overview of Bayesian network classifiers is given in [4].

Let $\{A_1, \ldots, A_k\}$ be a set of attributes and C be a class variable. By $\mathbf{A}$ we will denote the multidimensional variable $(A_1, \ldots, A_k)$ and by $\mathbf{a} = (a_1, \ldots, a_k)$

[*]This work was supported by the Grant Agency of the Czech Republic through grant nr. 201/04/0393 and through the Czech-Austrian grant AKTION KONTAKT 2004/19

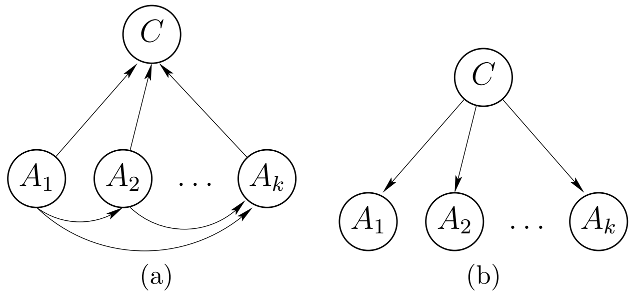

Figure 1: Two examples of Bayesian network classifiers

we will denote its states. In this paper we assume binary attributes having states labeled 0 and 1 and a binary class variable with states also labeled 0 and 1. In Figure 1 (a) we present an example of a Bayesian network model used as a classifier, whose structure is a complete graph. A disadvantage of this model is that the representation of this classifier is exponential with respect to the number of attributes. Consequently, it is difficult to estimate an exponential number of parameters from limited data and perform computations with the model.

On the other end of the complexity scale is the Naïve Bayes classifier, the simplest Bayesian network classifier. An example of this classifier is presented in Figure 1 (b). It relies on a strong assumption of conditional independence of the attributes given the class. Its advantage is that the parameter estimation from data can be done efficiently and also class predictions can be very fast.

In this paper we discuss application of a class of simple Bayesian network models – the models of independence of causal influence (ICI)[1] – to classification with large number of attributes. In an ICI classifier for each attribute variable $A_j, j = 1, \ldots, k$ we have one child A'_j that has assigned a conditional probability distribution $P(A'_j \mid A_j)$. Variables $A'_j, j = 1, \ldots, k$ are parents of the class variable C. $P_M(C \mid \mathbf{A}')$ represents a deterministic function f that assigns to each combination of values $(a'_1, \ldots, a'_k)$ a class c. Examples of ICI models are noisy-or, noisy-and, noisy-max, noisy-min, noisy-add, etc. Following Srinivas [14] we can represent an ICI model using the Bayesian network with the structure given in Figure 2.

A Bayesian network model M defines a unique probability distribution P_M. Since we restrict ourselves to ICI models, which form a subclass of Bayesian network models, the probability distribution P_M is defined for all combinations of values $(c, \mathbf{a}', \mathbf{a})$ of variables $C, \mathbf{A}'$, and $\mathbf{A}$, respectively. The probability dis-

[1]The term *independence of causal influence* was first used by Heckerman [5]. Previously, the ICI property was also called *causal independence*. Since this type of independence need not be present in causal models only, in our context it might be better to say that attributes have independent influence on the class variable.

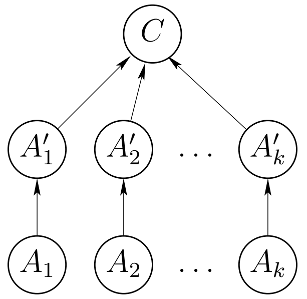

Figure 2: Model of an ICI classifier

tribution P_M is given by

$$P_M(c, \mathbf{a}', \mathbf{a}) \quad = \quad P_M(c \mid \mathbf{a}') \cdot \prod_{j=1}^{k} P_M(a'_j \mid a_j) \cdot P_M(a_j) \ , \qquad (1)$$

where conditional probability $P_M(c \mid \mathbf{a}')$ is one iff $c = f(\mathbf{a}')$, otherwise it is zero.

The original contribution of this paper is the application of ICI models to classification. In case of the noisy-or model, the classification is equivalent to logistic regression, which means we do not get any new classifier. However, we get a new probabilistic interpretation of logistic regression classifiers and a new algorithm for learning this classifier - the EM-algorithm. An advantage of the EM-algorithm is that it can be used also for learning from incomplete data, i.e., from data where some attribute values are missing. However, the major contribution is that the suggested learning method is directly applicable to other types of ICI models, namely to noisy-add, noisy-max, and noisy-min. In case of other ICI-models (in contrast to the equivalence of noisy-or and logistic regression classifiers) we are not aware of any previously described equivalent classifiers.

The paper is organized as follows. In Section 2 we propose an efficient implementation of the EM-algorithm that can be used to learn parameters of an ICI model. In Section 3 the best known example of an ICI model - the noisy-or model - is described. In Section 4 we discuss the correspondence between classification using the noisy-or model and a statistical method known as *logistic discrimination.* In Section 5 we compare noisy-or with other classifiers using the well known Reuters-21578 text categorization collection.

2 Learning parameters of the ICI classifiers

Let D be a set of data, which are independently randomly generated from an ICI model M with a known function f. Further assume that no additional information about the class C is available. Let $D = \{\mathbf{e}^1, \ldots, \mathbf{e}^n\}$, where the instances are

$$\mathbf{e}^i = \{c^i, \mathbf{a}^i\} = \{c^i, a_1^i, \ldots, a_k^i\}, \quad \text{for i=1,...,n.}$$

The learning process aims at parameters (i.e. values of conditional probability distributions) of the ICI model M that maximize the ability to correctly predict class C. This can be formalized as maximization of the conditional likelihood CL of a model M given data D

$$CL(M \mid D) \quad = \quad \prod_{i=1}^{n} P_M(c^i \mid \mathbf{a}^i)$$

or, equivalently, as maximization of the conditional log-likelihood CCL of a model M given data D

$$CLL(M \mid D) \quad = \quad \sum_{i=1}^{n} \log P_M(c^i \mid \mathbf{a}^i) \ . \tag{2}$$

Note that this is generally different from learning a model M that fits the data best, for which the maximization of the (unconditional) log-likelihood of the model given observed data

$$LL(M \mid D) \quad = \quad \sum_{i=1}^{n} \log P_M(c^i, \mathbf{a}^i)$$

is appropriate[2].

It is important to realize that for an ICI model M the maximization of conditional log-likelihood $CLL(P_M \mid D)$ and the maximization of log-likelihood yield equivalent resulting distributions. It follows from the following observation.

The log-likelihood of an ICI model M given data D can be expressed as

$$\begin{aligned} LL(M \mid D) &= \sum_{i=1}^{n} \log P_M(c^i, \mathbf{a}^i) \\ &= \sum_{i=1}^{n} \log \left(P_M(c^i | a_1^i, \ldots, a_k^i) \cdot \prod_{j=1}^{k} P_M(a_j^i) \right) \\ &= \sum_{i=1}^{n} \log P_M(c^i | a_1^i, \ldots, a_k^i) + \sum_{i=1}^{n} \log \prod_{j=1}^{k} P_M(a_j^i) \\ &= CLL(P_M \mid D) + \sum_{i=1}^{n} \log \prod_{j=1}^{k} P_M(a_k^i) \ . \end{aligned}$$

Both expressions on the right hand side of the previous formula are negative and $P_M(C|A_1, \ldots, A_k)$ and $P_M(A_j), j = 1, \ldots, k$ are assumed to have independent

[2]Later we will see that for a certain family of classifiers it does not make any difference whether we maximize CLL or LL.

parameterizations. If we find a $P_M(C, A_1, \ldots, A_k)$ that maximizes LL it has to maximize CLL as well. Therefore we can use the EM-algorithm to learn model parameters from the training data (in Section 2.2).

2.1 Transformation of ICI models using a hidden variable

We will derive an efficient implementation of the EM-algorithm that is based on the transformation of an ICI model using a hidden variable [3, 15]. We exploit the fact that for most commonly-used functions f the joint probability of an ICI model can be represented by use of a hidden variable as a product of two-dimensional potentials.

Let $P_M(\cdot)$ denote the joint distribution of an ICI model (as defined by formula 1). After the application of the transformation by use of a hidden variable B (for details see [3] or [15]) we can write:

$$P_M(\cdot) \quad = \quad \sum_B \left(\prod_{j=1}^{k} \psi_j(A'_j, B) \right) \cdot \varphi(B, C) \cdot \left(\prod_{j=1}^{k} P_M(A'_j, A_j) \right) \quad , \tag{3}$$

where B is a hidden variable. This factorization is always possible [15], but for computational efficiency it is important to define potentials so that variable B has the minimal number of states.

Minimal factorizations are known for the following functions [13]: maximization, minimization, addition, and the parity function. Note that the noisy-or model corresponds to the noisy-max model in the case of binary variables. The factorization of noisy-max was originally proposed in [3]. See also [15] for a discussion on how the transformation of some other models containing functional dependence can be done. In the following example we will describe a minimal factorization of the potential representing the OR function.

Example 1 Let B take two states labeled 0 and 1. Define for $j \in \{1, \ldots, k\}$, $a'_j \in \{0, 1\}$, and $b \in \{0, 1\}$

$$\psi_j(a'_j, b) \quad = \quad \begin{cases} 1 & a'_j \leq b \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

and for $c \in \{0, 1\}$, $b \in \{0, 1\}$

$$\varphi(b, c) \quad = \quad \begin{cases} +1 & b = c \\ -1 & b = c - 1 \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

It is not difficult to verify that the potentials $\psi_j, j = 1, \ldots, k$ and φ satisfy formula (3) for the noisy-or model. □

We can express the suggested transformation of an ICI model graphically: the original ICI model presented in Figure 2 can be transformed to a decomposable model depicted in Figure 3, where each arc corresponds to one potential.

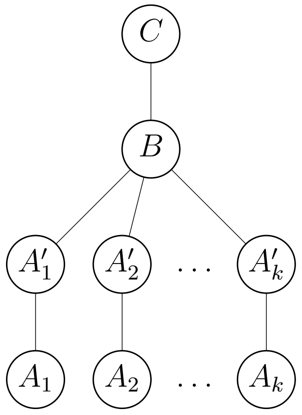

Figure 3: Transformed model of an ICI classifier

Lemma 1 provides a recipe for efficient computation of conditional probabilities $P_M(A'_j \mid \mathbf{e}), j = 1, \dots, k$ given an instance (data vector) $\mathbf{e}$. It will be used in the E-step of the EM-algorithm (as it will be described in Section 2.2). We will use the following notation. Let for $j = 1, \dots, k$

$$\begin{aligned} \psi'_j(A'_j, B \mid \mathbf{e}) &= \psi_j(A'_j, B) \cdot P_M(A'_j \mid A_j = a_j) \\ \psi'_j(B \mid \mathbf{e}) &= \sum_{A'_j} \psi'_j(A'_j, B \mid \mathbf{e}) \ . \end{aligned}$$

Lemma 1 *Assume a data vector* $\mathbf{e}$ *corresponding to evidence* $A_1 = a_1, \dots, A_k = a_k, C = c$. *Then for* $\ell = 1, \dots, k$

$$P_M(A'_\ell \mid \mathbf{e}) \propto \sum_B \varphi(B, C = c) \cdot \psi'_\ell(A'_\ell, B \mid \mathbf{e}) \cdot \prod_{j \neq \ell} \psi'_j(B \mid \mathbf{e}) \ .$$

Proof. From formula 3 it follows that

$$\begin{aligned} P_M(A'_1, \dots, A'_k \mid \mathbf{e}) &\propto \sum_B \varphi(B, C = c) \cdot \left(\prod_{j=1}^{k} \psi_j(A'_j, B) \cdot P_M(A'_j \mid A_j = a_j) \right) \\ &\propto \sum_B \varphi(B, C = c) \cdot \prod_{j=1}^{k} \psi'_j(A'_j, B \mid \mathbf{e}) \ . \end{aligned}$$

For $\ell = 1, \dots, k$ we get $P_M(A'_\ell \mid \mathbf{e})$ by marginalizing out variables $A'_j \neq A'_\ell$ from $P_M(A'_1, \dots, A'_k \mid \mathbf{e})$, which immediately proves the assertion of the lemma. $\square$

2.2 EM-algorithm

The EM-algorithm, first introduced in [2], is a broadly applicable algorithm for computing maximum likelihood estimates from incomplete data. For a thorough description of the algorithm see [8]. An efficient implementation of the algorithm for graphical models was proposed by Lauritzen [7]. However, his implementation is intractable if applied directly to an ICI model with hundreds of attributes[3].

Every iteration of the EM-algorithm consists of two steps: the expectation step (E-step) and maximization step (M-step). In our transformed decomposable model the E-step corresponds to computing the expected marginal count $n(A'_\ell, A_\ell)$ given data $D = \{\mathbf{e}^1, \ldots, \mathbf{e}^n\}$ and model M from the previous step[4]:

$$n(A'_\ell, A_\ell) \quad = \quad \sum_{i=1}^{n} P_M(A'_\ell, A_\ell \mid \mathbf{e}^i) \text{ for all } \ell = 1, \ldots k \ ,$$

where for each (a'_ℓ, a_ℓ)

$$P_M(A'_\ell = a'_\ell, A_\ell = a_\ell \mid \mathbf{e}^i) \quad = \quad \begin{cases} P_M(A'_\ell = a'_\ell \mid \mathbf{e}^i) & \text{if } a_\ell = a^i_\ell \\ 0 & \text{otherwise.} \end{cases}$$

The maximization step corresponds to setting

$$P^\star_M(A'_\ell \mid A_\ell) \quad = \quad \frac{n(A'_\ell, A_\ell)}{n(A_\ell)}, \text{ for all } \ell = 1, \ldots k \ .$$

These distributions are used in the next iteration of the EM-algorithm[5].

Note that, during one iteration of the EM-algorithm, the largest tables we manipulate are two-dimensional; they have a number of entries equal to the number of states of variable B times number of states of the respective variable. The lower the number of states of B, the more computationally efficient the procedure.

3 Noisy-or classifier

The noisy-or model was first introduced by Pearl [9]. As its name suggests it is a generalization of the deterministic OR relation. It is an ICI model where f is the OR function, i.e.,

$$P_M(C = 0 \mid \mathbf{A}' = \mathbf{0}) = 1 \text{ and } P_M(C = 0 \mid \mathbf{A}' \neq \mathbf{0}) = 0 \ .$$

Probability distributions $P_M(A'_i \mid A_i)$, $j = 1, \ldots, k$ represent a noise. The joint probability distribution of the noisy-or model is

$$P_M(\cdot) \quad = \quad P_M(C \mid A'_1, \ldots, A'_k) \cdot \left(\prod_{j=1}^{k} P_M(A'_j \mid A_j) \cdot P_M(A_j) \right) \ .$$

[3]When constructing a junction tree used in the EM-algorithm all parents of node C would be married. It would result in one clique containing all attributes and the class node.

[4]Recall that $\mathbf{e}^i = \{c^i, \mathbf{a}^i\} = \{c^i, a^i_1, \ldots, a^i_k\}$ for $i = 1, \ldots, n$.

[5]Since we optimize only CLL and not LL we need not update $P^\star_M(A_\ell)$. However, even if we did, we would simply set $P^\star_M(A_\ell) = \frac{n(A_\ell)}{n}$.

It follows that

$$P_M(C=0 \mid \mathbf{A}=\mathbf{a}) = \prod_j P_M(A'_j=0 \mid A_j=a_j) \quad (6)$$

$$P_M(C=1 \mid \mathbf{A}=\mathbf{a}) = 1-\prod_j P_M(A'_j=0 \mid A_i=a_j) \ . \quad (7)$$

Using a threshold $0 \leq t \leq 1$ all data vectors $\mathbf{a}=(a_1,\ldots,a_k)$ such that

$$P_M(C=0 \mid \mathbf{A}=\mathbf{a}) \quad < \quad t$$

are classified to class $C=1$.

Remark 1 In order to maximize the conditional log-likelihood $CLL(P_M \mid D)$ on training data D the threshold is set to 0.5.

The noisy-or classifier can have the following semantics. If an attribute, say A_j, is in a state a_j then the instance $(a_1,\ldots,a_j,\ldots,a_k)$ is classified to class $C=1$ unless there is an inhibitory effect, which has probability $P_M(A'_j = 0|A_j=a_j)$. All inhibitory effects are assumed to be independent. Therefore the probability that an instance does not belong to class C, i.e. $C=0$, is a product of all inhibitory effects $\prod_j P_M(A'_j=0 \mid A_j=a_j)$[6].

3.1 E-step of the EM-algorithm for the noisy-or

The following lemma is a special case of Lemma 1 for the case of the noisy-or model. It shows how the updating of the model can be done efficiently in the E-step of the EM-algorithm proposed in Section 2.2.

Lemma 2 *Assume a noisy-or classifier P_M and an evidence $C=c,\mathbf{A}=\mathbf{a}$. The updated probabilities of A'_ℓ for $\ell=1,\ldots,k$ can be computed as*

$$P_M(A'_\ell = a'_\ell \mid C=c,\mathbf{a})$$
$$= \begin{cases} 1 & \text{if } c=0 \text{ and } a'_\ell=0 \\ 0 & \text{if } c=0 \text{ and } a'_\ell=1 \\ \frac{1}{z}\cdot\left(\begin{array}{c} P_M(A'_\ell=0 \mid A_\ell=a_\ell) \\ -\prod_j P_M(A'_j=0 \mid A_j=a_j)) \end{array}\right) & \text{if } c=1 \text{ and } a'_\ell=0 \\ \frac{1}{z}\cdot P_M(A'_\ell=1 \mid A_\ell=a_\ell) & \text{if } c=1 \text{ and } a'_\ell=1 \end{cases}$$

where z is a normalization constant.

Proof. From Lemma 1 we have that

$$P_M(A'_\ell \mid \mathbf{e}) \quad \propto \quad \sum_B \varphi(B,C=c)\cdot\psi'_\ell(A'_\ell,B \mid \mathbf{e})\cdot\prod_{j\neq\ell}\psi'_j(B \mid \mathbf{e}) \ . \quad (8)$$

[6]Note that the model is asymmetric with respect to the coding of the values of C. The symmetric model to the noisy-or is the noisy-and model.

where for $j = 1, \ldots, k$:

$$\begin{aligned}\psi'_j(A'_j, B \mid \mathbf{e}) &= \psi_j(A'_j, B) \cdot P_M(A'_j \mid A_j = a_j) && (9)\\ \psi'_j(B \mid \mathbf{e}) &= \sum_{A'_j} \psi'_j(A'_j, B \mid \mathbf{e}) \ . && (10)\end{aligned}$$

When we substitute potentials $\psi_j, j = 1, \ldots, k$ and φ defined in Example 1 into equations 9 and 10 we get for $j = 1, \ldots, k$

$$\begin{aligned}\psi'_j(A'_j = 0, B = 0 \mid \mathbf{e}) &= P_M(A'_j = 0 \mid A_j = a_j)\\ \psi'_j(A'_j = 1, B = 0 \mid \mathbf{e}) &= 0\\ \psi'_j(A'_j, B = 1 \mid \mathbf{e}) &= P_M(A'_j \mid A_j = a_j)\\ \psi'_j(B = 0 \mid \mathbf{e}) &= P_M(A'_j = 0 \mid A_j = a_j)\\ \psi'_j(B = 1 \mid \mathbf{e}) &= P_M(A'_j = 0 \mid A_j = a_j) + P_M(A'_j = 1 \mid A_j = a_j) \quad = \quad 1 \ .\end{aligned}$$

Substitution into equation 8 leads to

$$\begin{aligned}P_M(A'_\ell = 0 \mid 0, \mathbf{a}) &= 1\\ P_M(A'_\ell = 1 \mid 0, \mathbf{a}) &= 0\\ P_M(A'_\ell = 0 \mid 1, \mathbf{a}) &\propto P_M(A'_\ell = 0 \mid A_\ell = a_\ell) \cdot (1 - \prod_{j \neq \ell} P_M(A'_j = 0 \mid A_j = a_j))\\ P_M(A'_\ell = 1 \mid 1, \mathbf{a}) &\propto P_M(A'_\ell = 1 \mid A_\ell = a_\ell)\end{aligned}$$

□

3.2 Convergence of the EM-algorithm

Generally, the EM-algorithm need not converge to a global maxima of CLL. The usual way to get around this problem is to restart the EM-algorithm from different starting points and finally select a limit probability distribution that has the maximal target value.

In case of the general noisy-or model we observed that there were several global maxima of the log-likelihood. This means that several noisy-or models with different parameter values maximized the log-likelihood. Consequently, the parameters of the original noisy-or model were not identifiable. However, in the experiments it did not cause any problem since all the limit distributions we got when applying the EM-algorithm to different starting points had the same log-likelihood value. Furthermore, for each classification task all of the limit distributions were equivalent to a noisy-or classifier in the *canonical form* that will be defined in Section 4.1.

A disadvantage of the EM-algorithm is that it typically converges slowly near the optimum. However, in the experiments we did, the number of iterations sufficient to get reasonably close to the limit distribution was always less than fifty.

3.3 A restricted noisy-or classifier

One possibility for getting the parameters of noisy-or models identifiable is to restrict the family of the noisy-or models by certain constraints. One solution is to regard only noisy-or classifiers satisfying the property that

$$P_M(A'_j = 0 \mid A_j = 0) \quad \geq \quad P_M(A'_j = 0 \mid A_j = 1) \quad > \quad 0$$

We will call this condition the *monotonicity condition*.

This requirement means that it is more probable that the instance does not belong to the class C for $A_j = 0$ then for $A_j = 1$. For example, assume the interpretation that $A_j = 1$ means that attribute A_j is present and that $A_j = 0$ means that attribute A_j is not present. Then the presence of the attribute implies a higher probability that the instance belongs to the class C.

Now assume the threshold t to be a parameter of the noisy-or model that can be learned. Having restricted the family of the noisy-or models by the monotonicity condition we can require the value

$$P_M(A'_j = 0 \mid A_j = 0) \quad \overset{df}{=} \quad 1 \tag{11}$$

for all $j = 1, \ldots, k$ and still construct a noisy-or classifier that has the same behavior as any classifier from the restricted family. A noisy-or classifier satisfying condition 11 will be referred to as a *restricted* noisy-or classifier. The restricted noisy-or classifier has only $k+1$ parameters[7] to be learned from data.

Lemma 3 *For any noisy-or classifier satisfying the monotonicity condition we can construct a restricted noisy-or classifier that assigns instances to the same class.*

Proof. The decision rule of the noisy-or classifier with threshold t is

"If $P_M(C = 0 \mid \mathbf{A} = \mathbf{a}) < t$ then instance $\mathbf{a}$ is classified to class $C = 1$."

For a noisy-or classifier P_M and every $\mathbf{a}$ it holds

$$\begin{aligned} P_M(C = 0 \mid \mathbf{A} = \mathbf{a}) &= \prod_j P_M(A'_j = 0 \mid A_j = a_j) \\ &= \prod_{j:a_j=0} P_M(A'_j = 0 \mid A_j = 0) \cdot \prod_{j:a_j=1} P_M(A'_j = 0 \mid A_j = 1) \end{aligned}$$

Under the assumptions of the lemma we can rewrite it as

$$P_M(C = 0 \mid \mathbf{A} = \mathbf{a}) = \prod_j P_M(A'_j = 0 \mid A_j = 0) \cdot \prod_{j:a_j=1} \frac{P_M(A'_j = 0 \mid A_j = 1)}{P_M(A'_j = 0 \mid A_j = 0)}$$

[7] I.e., k model parameters plus the threshold.

Define another noisy-or classifier P'_M by

$$\begin{aligned} P'_M(A'_j = 0 \mid A_j = 0) &= 1 \\ P'_M(A'_j = 0 \mid A_j = 1) &= \frac{P_M(A'_j = 0 \mid A_j = 1)}{P_M(A'_j = 0 \mid A_j = 0)} \\ t' &= \frac{t}{\prod_j P_M(A'_j = 0 \mid A_j = 0)} \end{aligned}$$

Note that $P'_M(A'_j = 0 \mid A_j = 1) \leq 1$.

Observe that the decision rule of this noisy-or classifier

"If $P'_M(C = 0 \mid \mathbf{A} = \mathbf{a}) < t'$ then instance $\mathbf{a}$ is classified to class $C = 1$."

is equivalent to the decision rule of the original noisy-or classifier P_M. □

4 Correspondence between noisy-or and logistic discrimination

The fundamental assumption of *logistic discrimination* [1] is that the log-likelihood ratio is assumed to be linear:

$$\log \frac{P_M(C = 0 \mid \mathbf{A} = \mathbf{a})}{1 - P_M(C = 0 \mid \mathbf{A} = \mathbf{a})} = \beta_0 + \sum_j \beta_j \cdot a_j = \beta_0 + \sum_{j:a_j=1} \beta_j \ . \quad (12)$$

Typically, $\beta_0 = \beta'_0 + \log \frac{P_D(C=0)}{P_D(C=1)}$. From (12) we have that

$$P_M(C = 0 \mid \mathbf{A} = \mathbf{a}) = \frac{\exp(\beta_0 + \sum_{j:a_j=1} \beta_j)}{1 + \exp(\beta_0 + \sum_{j:a_j=1} \beta_j)} \ . \quad (13)$$

In logistic discrimination, the overall posterior probability of correct assignment is maximized if a data vector $\mathbf{a}$ is classified to $C = 1$ provided

$$\log \frac{P_M(C = 0 \mid \mathbf{A} = \mathbf{a})}{1 - P_M(C = 0 \mid \mathbf{A} = \mathbf{a})} < 0 \ . \quad (14)$$

Logistic discrimination and classification using a noisy-or classifier are equivalent in the sense that, for any logistic regression classifier, we can construct a noisy-or classifier that provides equivalent results and vice versa. This assertion is formalized in the following lemmas. Their proofs are constructive, i.e., they show how an equivalent classifier can be constructed.

Lemma 4 *For any logistic regression classifier, we can construct a noisy-or classifier that assigns instances to the same class as the logistic regression classifier.*

Proof. The decision rule for a logistic regression classifier is:

"If $\beta_0 + \sum_{j:a_j=1} \beta_j < 0$ then instance $\mathbf{a}$ is classified to class $C = 1$."

By exponentiation we get

$$\exp(\beta_0) \cdot \prod_{j:a_j=1} \exp(\beta_j) \quad < \quad 1 \tag{15}$$

Let the parameters of a noisy-or classifier be defined as

$$P_M(A'_j = 0 \mid A_j = a_j) \quad = \quad \begin{cases} \frac{\exp(\beta_j)}{1+\exp(\beta_j)} & \text{for } a_j = 1 \\ \frac{1}{1+\exp(\beta_j)} & \text{for } a_j = 0 \ . \end{cases}$$

Note that for $a_j = 0, 1$ it holds that $0 < P_M(A'_j = 0 \mid A_j = a_j) < 1$. We can rewrite inequality 15 as

$$\begin{aligned} \exp(\beta_0) \cdot \left(\prod_j (1 + \exp(\beta_j)) \right) \cdot \left(\prod_j P_M(A'_j = 0 \mid A_j = a_j) \right) \quad &< \quad 1 \\ \prod_j P_M(A'_j = 0 \mid A_j = a_j) \quad &< \quad t \\ P_M(C = 0 \mid \mathbf{A} = \mathbf{a}) \quad &< \quad t \ , \end{aligned}$$

where

$$t \quad = \quad \frac{1}{\exp(\beta_0) \cdot \left(\prod_j (1 + \exp(\beta_j)) \right)} \ .$$

Note that

"If $P_M(C = 0 \mid \mathbf{A} = \mathbf{a}) < t$ then instance $\mathbf{a}$ is classified to class $C = 1$."

is the decision rule of the noisy-or classifier with threshold t. □

Lemma 5 *For any noisy-or classifier such that for every j and $a_j = 0, 1$*

$$P_M(A'_j = 0 \mid A_j = a_j) \quad > \quad 0$$

we can construct a logistic regression classifier that assigns instances to the same class as the noisy-or classifier.

Proof. The decision rule for the noisy-or classifier with threshold t is

"If $P_M(C = 0 \mid \mathbf{A} = \mathbf{a}) < t$ then instance $\mathbf{a}$ is classified to class $C = 1$."

Let

$$\begin{aligned} p_j(a_j) \quad &= \quad \log P_M(A'_j = 0 \mid A_j = a_j) \\ r_j \quad &= \quad p_j(1) - p_j(0) \\ r_0 \quad &= \quad \sum_j p_j(0) \ . \end{aligned}$$

Then we can write

$$\begin{aligned}
\log P_M(C=0 \mid \mathbf{A}=\mathbf{a}) &= \sum_j \log P_M(A'_j=0 \mid A_j=a_j) \\
&= \sum_{j:a_j=0} p_j(0) + \sum_{j:a_j=1} p_j(1) \\
&= \sum_j p_j(0) + \sum_{j:a_j=1} p_j(1) - p_j(0) \\
&= r_0 + \sum_{j:a_j=1} r_j \ .
\end{aligned}$$

Since log is a monotonous function the decision rule for the noisy-or classifier with threshold t is equivalent to

"If $r_0 - \log t + \sum_{j:a_j=1} r_j < 0$ then instance $\mathbf{a}$ is classified to class $C=1$."

Thus a logistic regression classifier with parameters

$$\begin{aligned}
\beta_j &= r_j = \log \frac{P_M(A'_j=0 \mid A_j=1)}{P_M(A'_j=0 \mid A_j=0)} \\
\beta_0 &= r_0 - \log t = \sum_j \log P_M(A'_j=0 \mid A_j=0) - \log t
\end{aligned}$$

provides equivalent results as the noisy-or classifier. □

4.1 Canonical form of a noisy-or classifier

What would be the parameters of the noisy-or classifier P'_M we would get after the transformation of the original noisy-or classifier P_M to logistic classifier (Lemma 5) and back to the noisy-or (Lemma 4)? Of course, we would get a classifier that assigns instances to the same class as the original one: the question is 'what are its parameters?' After a little algebra we get

$$P'_M(A'_j=0 \mid A_j=a_j) = \frac{P_M(A'_j=0 \mid A_j=a_j)}{P_M(A'_j=0 \mid A_j=0) + P_M(A'_j=0 \mid A_j=1)} \quad (16)$$

$$t' = \frac{t}{\prod_j \left(P_M(A'_j=0 \mid A_j=0) + P_M(A'_j=0 \mid A_j=1)\right)} \quad (17)$$

Observe that if $P_M(A'_j=0 \mid A_j=0) + P_M(A'_j=0 \mid A_j=1) = 1$ for all j then noisy-or classifier P'_M has the same parameters as the original one. We will say that such a noisy-or classifier is in the *canonical form*. Formulas 16 and 17 can be used for the transformation of a noisy-or classifier to its canonical form.

5 Experimental results

We used the Reuters-21578 text categorization collection (Distribution 1.0) containing the Reuters new stories. More specifically, we used data preprocessed by Karčiauskas [6]. The split of data to the training and testing sets was according to time of publication of the documents (ModApte). Classes that contained only one document were eliminated together with the corresponding documents. The resulting training set contained 7769 documents and testing set 3018 documents. After removing function words and words that appear only in one document the feature set contained 15715 words. The tests were performed on the ten classes containing the most documents. We did not fix the number of features in advance. Instead, for each class we selected the most informative features out of 15715 binary features using the expected information gain as a feature selection criteria. Specifically, for each attribute A we computed the expected information gain $IG(A) = \sum_a P(A=a) \cdot (H(P(C)) - H(P(C \mid A=a)))$, where H denotes the Shannon entropy. We selected all attributes for which $IG(A)$ was higher or equal to a threshold[8].

We measured the performance of the noisy-or classifiers by commonly used measures. For each class we counted the number of tested documents:

- classified correctly to the class $C = 1$ ... *tp* (true positive),
- classified incorrectly to the class $C = 1$ ... *fp* (false positive),
- classified correctly out of the class $C = 1$ ... *tn* (true negative), and
- classified incorrectly out of the class $C = 1$ ... *fn* (false negative).

Commonly used measures based on these counts are

$$\begin{array}{llll} \textit{precision} & \pi = \frac{tp}{tp+fp}, & \textit{recall} & \varrho = \frac{tp}{tp+fn}, \\ \textit{accuracy} & \eta = \frac{tp+tn}{tp+tn+fn+fp}, & F_1\textit{-measure} & F_1 = \frac{2\pi\varrho}{\pi+\varrho} . \end{array}$$

We provide these measures in percentage scale (i.e., multiplied by 100). *Macro-average* is the average of local values of π and ϱ in each class. *Micro-average* is computed so that, first, the total sums of *tp*,*fp*, and *fn* are computed over all ten tested classes. Second, π and ϱ are computed according to their definitions, but from the totals of *tp*,*fp*, and *fn*.

Most classifiers can be tuned so that they sacrifice precision to high recall or vice versa. We can tune the value of threshold[9] t of the restricted noisy-or classifier[10]. The point where precision equals to recall is called the break-even point.

For the comparisons of rN-OR with other classifiers the threshold t was tuned on the training data for each class separately, first, with respect to the

[8] After few experiments we decided for value 0.005.

[9] Recall that threshold t is used to assign an instance described by attributes $\mathbf{a}$ to class $C = 1$ if $P(C = 0 \mid \mathbf{A} = \mathbf{a}) < t$.

[10] I.e., a noisy-or classifier that has the value of $P_M(A'_j = 0 \mid A_j = 0)$ fixed to one.

accuracy (rN-OR-a) and second, with respect to the F_1-measure (rN-OR-f). In Figure 4 we present dependence of the micro-average precision and recall of the restricted noisy-or classifier on the value of the threshold. The micro-average break-even point was 86%. It corresponds to threshold $t = 0.55$.

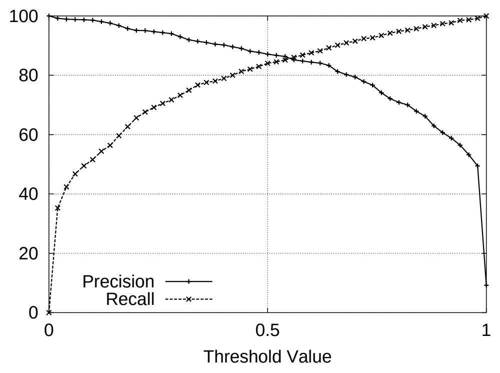

Figure 4: Micro-average precision and recall with respect to the threshold value.

In Table 1 and 2 we compare the accuracy η and the F_1-measure, respectively, of the noisy-or classifiers – two restricted (rN-OR-a and rN-OR-f) and the general (non-restricted) noisy-or classifier (N-OR) with the fixed threshold $t = 0.5$ – with four other classifiers implemented in the Weka system [16]:

- the support vector machine (SMO) - with the degree of the polynomial kernel equal to one and learned by John C. Platt's sequential minimal optimization algorithm [10],
- the logistic classifier (LOG) - a logistic regression model with a ridge estimator [12],
- the Naïve Bayes classifier (NB), and
- the Decision Tree classifier (J48) - the Weka implementation of C4.5 decision tree [11],

all used with the default parameters. We tested the classifiers on a subset of the Reuters data consisting of the ten largest classes. The classes are ordered according to the number of documents (nd) in each class.

If we compare two simple probabilistic classifiers – the popular Naïve Bayes model (NB) and the noisy-or model (N-OR) – we can see that N-OR often provide better results.

Note that the comparisons are somewhat unfair to probabilistic classifiers since they were learned to maximize the conditional log-likelihood not the F_1-measure nor the accuracy[11]. In contrast to non-probabilistic classifiers the prob-

[11]In order to be able to compare with classifiers based on different principles we could not use the conditional log-likelihood measured on testing data as a criteria for the comparisons.

Table 1: Comparisons of the accuracy.

class	*nd*	J48	NB	LOG	SMO	rN-OR-a	rN-OR-f	N-OR
earn	1087	97.1	94.8	98.0	**98.5**	96.3	96.3	96.3
acq	719	95.1	95.3	96.3	**96.7**	93.2	93.3	93.2
crude	189	98.1	97.1	98.4	**98.5**	98.2	98.1	98.1
money-fx	179	96.1	94.1	**96.4**	96.3	95.7	95.6	95.8
grain	149	99.1	96.4	98.9	99.3	**99.4**	99.3	99.2
interest	131	96.7	95.8	**97.1**	96.9	96.8	96.4	96.5
trade	117	97.1	93.8	97.9	**98.0**	96.3	96.3	96.6
ship	89	98.9	98.8	**99.0**	98.9	**99.0**	**99.0**	98.9
wheat	71	**99.5**	97.6	99.4	**99.5**	**99.5**	**99.5**	**99.5**
corn	56	**99.7**	97.0	99.6	**99.7**	**99.7**	**99.7**	**99.7**

Table 2: Comparisons of the F_1-measure.

class	*nd*	J48	NB	LOG	SMO	rN-OR-a	rN-OR-f	N-OR
earn	1087	96.0	93.1	97.3	**97.9**	95.0	94.9	95.0
acq	719	89.4	90.2	91.9	**92.9**	85.2	86.1	85.5
crude	189	84.6	79.5	87.6	**88.0**	86.0	85.6	84.5
money-fx	179	66.1	60.1	**67.1**	65.6	61.4	63.9	60.9
grain	149	91.0	71.7	88.7	**92.8**	93.9	93.5	92.7
interest	131	58.0	**59.9**	59.7	56.1	51.0	56.1	40.2
trade	117	63.1	49.5	72.5	**72.9**	61.7	63.2	51.0
ship	89	78.2	**81.9**	81.5	79.0	81.0	81.2	79.5
wheat	71	**90.3**	65.0	88.6	**90.3**	**90.3**	**90.3**	**90.3**
corn	56	**92.6**	51.9	89.7	91.8	91.8	91.8	91.8

abilistic classifiers not only classify an instance to a class but also provide the probability that the classification is correct.

One might expect that due to the equivalence of classification using the logistic regression (LOG) and using the noisy-or model (N-OR) (as it is described in Section 4) the two methods should provide equivalent results. But, since LOG is learned by a different method it need not provide equivalent results as N-OR. In the experiments with the Reuters dataset often LOG performs better than N-OR. This suggests that the LOG learning method, which minimizes the negative log-likelihood plus a ridge estimator is more suitable for the tested data than the EM-algorithm for N-OR.

However, to provide statistically significant comparisons we would need to compare the classifiers on several different datasets that would allow us to compute not only the average performance but also the standard errors. The Reuters data set with the split of data to the training and testing sets according to time of publication of the documents does not allow us to do that. Another problem of this split of of data is that the classifier properly tuned on training data need not preform best on the testing data. We can observe this effect on the rN-OR classifier, which sometimes – when tuned with respect to a criteria on training data – performs worse on testing data (measured by the criteria used for tuning) than its non-tuned version.

6 Conclusions

In this paper we have applied Bayesian network models with attributes of independent influence on the class variable to the classification task. Our main motivation was the classification of instances described by a large number of attributes.

We have described the correspondence between classification using the noisy-or and the logistic discrimination. We proposed, implemented, and tested an efficient version of the EM-algorithm, which we used to learn parameters of the noisy-or classifiers.

We tested the general and restricted versions of the noisy-or classifier on the Reuters text categorization collection. We observed that despite their simplicity they have a competitive performance. In the future we plan to study other ICI classifiers, namely, noisy-add, noisy-max, and noisy-min.

Acknowledgments

I would like to thank Gytis Karčiauskas who provided me the preprocessed Reuters data. I am also grateful to Nevin L. Zhang who provided first feedback and encouraged me to work on the topic of this paper, Radim Jiroušek and Gernot Kleiter for their suggestions for improving the paper, and anonymous reviewers of the ECSQARU 2003 conference for their valuable comments and pointing out the correspondence to logistic discrimination.

References

[1] J. A. Anderson. Logistic discrimination. In P. R. Krishnaiah and L. N. Kanal, editors, *Handbook of Statistics*, volume 2, pages 169–191. North-Holland Publishing Company, 1982.

[2] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Statist. Soc. Ser. B*, 39:1–38, 1977.

[3] F. J. Díez and S. F. Galán. An efficient factorization for the noisy MAX. *International Journal of Intelligent Systems*, 18(2):165–177, 2003.

[4] N. Friedman, D. Geiger, and M. Goldszmitdt. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997.

[5] David Heckerman and John S. Breese. A new look at causal independence. In R. Lopez de Mantaras and David Poole, editors, *Proc. of the Tenth Conf. on Uncertainty in AI*, pages 286–292, 1994.

[6] G. Karčiauskas. Text categorization using hierarchical Bayesian networks classifiers. Master's thesis, Aalborg University, `http://www.cs.auc.dk/library`, 2002.

[7] S. L. Lauritzen. The EM algorithm for graphical association models with missing data. *Computational Statistics & Data Analysis*, 19:191–201, 1995.

[8] G. J. McLachlan and T. Krishnan. *The EM algorithm and extensions.* Wiley series in probability and statistics. John Wiley & Sons, Inc., 1997.

[9] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference.* Morgan Kaufmann, 1988.

[10] J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning.* MIT Press, 1998.

[11] Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[12] J.C. van Houwelingen S. le Cessie. Ridge estimators in logistic regression. *Journal of the Royal Statistical Society - Series C: Applied Statistics*, 41(1):191–201, 1992.

[13] P. Savicky and J. Vomlel. Factorized representation of functional dependence. *International Journal of Approximate Reasoning (submitted)*, 2004.

[14] Sampath Srinivas. A generalization of the Noisy-Or model. In David Heckerman and Abe Mamdani, editors, *Proc. of the Ninth Conf. on Uncertainty in AI*, pages 208–215, 1993.

[15] J. Vomlel. Exploiting functional dependence in Bayesian network inference. In A. Darwiche and N. Friedman, editors, *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence (UAI 2002)*, pages 528–535, 2002.

[16] Weka 3.2 - data mining with open source machine learning software. `http://www.cs.waikato.ac.nz/ml/weka/`, 2002.