Geometric Properties of Naive Bayes in Nominal Domains

Huajie Zhang and Charles X. Ling

Department of Computer Science
The University of Western Ontario
London, Ontario, Canada N6A 5B7
{hzhang, ling}@csd.uwo.ca

Abstract. It is well known that the naive Bayesian classifier is linear in binary domains. However, little work is done on the learnability of the naive Bayesian classifier in nominal domains, a general case of binary domains. This paper explores the geometric properties of the naive Bayesian classifier in nominal domains. First we propose a three-layer measure for the linearity of functions in nominal domains: hard linear, soft nonlinear, and hard nonlinear. We examine the learnability of the naive Bayesian classifier in terms of that linearity measure. We show that the naive Bayesian classifier can learn some hard linear and some soft nonlinear nominal functions, but still cannot learn any hard nonlinear functions.

1 Introduction

Learning classifiers from examples is an important issue in machine learning research. A classifier is a function that assigns a class label to an example. Assume A_1, A_2,$\cdots$, A_n are n attributes. An example E is represented by a vector $(a_1, a_2, , \cdots, a_n)$, where a_i is the value of A_i. There are two types of attributes: nominal (taking values from a finite set) and numeric (taking values from a continuous range). We restrict our discussion to nominal attributes in this paper. Let C represent the classification variable, which takes values $+$ (positive class) or $-$ (negative class), and let c be the value that C takes.

Numerous approaches to learning classifiers, such as decision trees, neural networks, and instance-based learning, have been studied. In recent years, probability approaches to learning classifiers have been extensively investigated. According to Bayes Theorem, the probability of an example $E = (a_1, a_2, \cdots, a_n)$ being in class c is

$$p(c|E) = \frac{p(a_1, a_2, \cdots, a_n|c)p(c)}{p(a_1, a_2, \cdots, a_n)}.$$

E belongs to the class $C = +$ iff

$$g(E) = \frac{p(C = +)p(a_1, a_2, \cdots, a_n|C = +)}{p(C = -)p(a_1, a_2, \cdots, a_n|C = -)} \geq 1,$$

where $g(E)$ is called a Bayesian classifier.

Assume all attributes are independent given the class value (conditional independence), then

$$p(a_1, a_2, \cdots, a_n|c) = \prod_{i=1}^{n} p(a_i|c).$$

The corresponding Bayesian classifier $g(E)$ is then:

$$g(E) = \frac{p(C=+)}{p(C=-)} \prod_{i=1}^{n} \frac{p(a_i|C=+)}{p(a_i|C=-)},$$

where $g(E)$ is called a naive Bayesian classifier or, in short, Naive Bayes.

Naive Bayes is easy to construct simply by estimating the value of $p(a_i|c)$ from training examples. Intuitively this might be not accurate, because the conditional independence assumption rarely holds true. To some extent, however, this intuition is not correct. Many empirical comparisons between Naive Bayes and C4.5 [9] showed that Naive Bayes predicts just as well as C4.5 [7, 6].

In recent years, researchers have attempted to uncover reasons for the good performance of Naive Bayes. Domingos and Pazzani [1] presented an explanation: even though Naive Bayes alters the probability distribution of a class, the class with the maximum probability may still be the same. This is verified by Frank's [3] work, which shows that the performance of Naive Bayes is much worse when it is used for regression (predicting a continuous value).

One interesting and fundamental question is the learnability of Naive Bayes. It is well-known that Naive Bayes can create only linear frontiers in binary domains [2]. That is, Naive Bayes can learn only linearly separable concepts in binary domains. For nominal domains, a general case of binary domains, there is no satisfying result. Attributes in nominal domains can have more than two values. Assume A_1, A_2, $\cdots$, A_n are n nominal attributes, each attribute A_i may have m values a_{i1}, a_{i2}, $\cdots$, and a_{im} ($m \geq 2$). Domingos and Pazzani [1] and Peot [8] introduced m new Boolean attributes B_{i1}, B_{i2}, $\cdots$, and B_{im} for each attribute A_i, and proved that Naive Bayes is linear over these new binary attributes. However, the linear separability on n original attributes is transformed to $m_1 \times m_2 \cdots \times m_n$ new attributes. To our knowledge, there is no general result for the linearity of Naive Bayes on original nominal attributes.

Given a function in nominal domains, however, how can we define linearity or linear separability? Typically, linearity is a geometric term on the Euclidean space $\mathcal{R}^n$ ($\mathcal{R}$ is the set of all real numbers), but nominal attributes do not have direct geometric meaning. We must map nominal attributes into numeric ones to discuss the linearity property of a function. The tricky issue is that different mappings may result in different results of linearity.

On the other hand, it is obvious that different functions in nominal domains may present different difficulties for Naive Bayes to learn. It is natural to ask the following questions: Can the complexity of a function be measured in terms of its geometric properties? Is there any relation between the geometric properties of a function and the learnability of Naive Bayes?

The motivation of this paper is to explore the learnability of Naive Bayes by answering the two questions above. The remainder of this paper is organized as follows. Section 2 introduces necessary definitions and briefly reviews the results in the upper bound on the learnability of Naive Bayes. Section 3 proposes a three-layer measure for the linearity of a function, and proves a sufficient and necessary condition for hard nonlinearity, and then examines the learnability of Naive Bayes in terms of that measure. In the conclusions, we summarize our results and outline our future work.

2 The Upper Bound on the Learnability of Naive Bayes

As mentioned above, Naive Bayes is a linear classifier in binary domains. Let us briefly review the relevant results [2].

Suppose that attributes A_1, A_2, $\cdots$, A_n are binary, taking value 0 or 1. Let p_i and q_i represent the probability $p(A_i = 1|C = +)$ and $p(A_i = 1|C = -)$ respectively, $E = (a_1, \cdots, a_n)$ be an example. Then the corresponding Naive Bayes $G(E)$ is:

$$G(E) = \frac{p(C=+)}{p(C=-)} \prod_{i=1}^{n} \frac{{p_i}^{a_i}(1-p_i)^{1-a_i}}{{q_i}^{a_i}(1-q_i)^{1-a_i}}. \tag{1}$$

It is straightforward to obtain a linear classifier by applying a logarithm to the above equation.

Two points should be noted here. First, we actually implicitly use a mapping from the two nominal values (such as *red* and *blue*) of a binary attribute to $\{0, 1\}$. Second, we cannot get a similar result when any of A_i has more than two values. Thus, we cannot simply extend the result in binary domains to nominal domains.

We begin our discussion on the linearity of Naive Bayes with a few definitions.

Definition 1 *Given n nominal attributes A_1, A_2, $\cdots$, A_n, and two classification labels $\{+, -\}$, a function f from $A_1 \times A_2 \cdots \times A_n$ to $\{+, -\}$ is called an n-dimensional nominal function.*

Zhang and Ling [12] proved that if a nominal function f "contains" an XOR, then no Naive Bayes can represent it. So, roughly, XOR is the upper bound of Naive Bayes. We give detailed definitions of "contain" below, since we will use this term in the next section.

Definition 2 *Assume f is an n-dimensional nominal function on A_1, A_2, $\cdots$, A_n. An (n-1)-dimensional partial function f_p of f on A_1, $\cdots$, A_{i-1}, A_{i+1}, $\cdots$, A_n, and $A_i = a_{ij}$, is called an (n-1)-dimensional subfunction at $A_i = a_{ij}$, denoted by $f(a_{ij})$, where $1 \leq i \leq n$.*

To get a k-dimensional subfunction of f is straightforward, by fixing $n-k$ attributes, $2 \leq k \leq n-1$.

Definition 3 *An n-dimensional nominal function f is said to contain a tangent XOR, if there is a 2-dimensional subfunction f_p on attributes A_i and A_j, and each of them has two distinct values a_i and $\bar{a}_i$, a_j and $\bar{a}_j$, respectively, such that a partial function $f_{p'}$ of f_p on $\{a_i, \bar{a}_i\} \times \{a_j, \bar{a}_j\}$ is an XOR function.*

Definition 4 *Assume f is an n-dimensional nominal function. f is said to contain a diagonal XOR if there are two attributes A_i and A_j, each of which has two distinct values, a_i, $\bar{a}_i$, and a_j, $\bar{a}_j$, respectively, such that:*

$$f(a_1, \cdots, a_i, \cdots, a_j, \cdots, a_n) = + \tag{2}$$
$$f(a_1, \cdots, a_i, \cdots, \bar{a}_j, \cdots, a_n) = - \tag{3}$$
$$f(\bar{a}_1, \cdots, \bar{a}_i, \cdots, a_j, \cdots, \bar{a}_n) = - \tag{4}$$
$$f(\bar{a}_1, \cdots, \bar{a}_i, \cdots, \bar{a}_j, \cdots, \bar{a}_n) = + \tag{5}$$

where a_l and $\bar{a}_l$ are two distinct values of A_l, $l \neq i$ and j.

Definition 5 *An n-dimensional nominal function f is said to contain an XOR if and only if f contains a tangent XOR or a diagonal XOR.*

3 Geometric Properties of Naive Bayes

Containing an XOR of a nominal function is a basic concept in this paper. It has been proved that Naive Bayes cannot learn any function containing XOR [12]. However, what is the relation between a function containing an XOR and its geometric linearity? We propose a measure for the geometric linearity of a function, and show that there is a close relation between a function's containing an XOR and its linear separability. Then we examine the learnability of Naive Bayes on functions with different linearity.

3.1 Measure for the Linearity of Nominal Functions

We have to map nominal attributes of a function into numeric ones in order to discuss its linearity. Since the values of a nominal attribute have no order, it is reasonable to map a nominal value into an arbitrary real number without conflicting with the mapping of another attribute value.

Definition 6 *Given n numeric attributes A_1, A_2, $\cdots$, A_n, and two classification labels $\{+, -\}$, a function f from $A_1 \times A_2 \cdots \times A_n$ to $\{+, -\}$ is called n-dimensional numeric function.*

Definition 7 *Given a nominal attribute $A = \{a_1, a_2, \cdots, a_m\}$, a numeric attribute mapping Γ is defined as a function from A to $\mathcal{R}$, such that for each a_i, $1 \leq i \leq m$, we have $\Gamma(a_i) = x_i$, $x_i \in \mathcal{R}$ and $x_i \neq x_j$ if $i \neq j$.*

Let us review the conclusion that Naive Bayes is a linear classifier in binary domains. As discussed in Section 2, this conclusion comes from an implicit assumption that the two values of each binary attribute are mapped into $\{0, 1\}$. Obviously, other reasonable mappings also exist, since we can map the two values of a binary attribute into any two different real numbers. Then we can no longer get Equation 1. Since the mapping affects the geometric properties of a nominal function, what we are interested in are the properties that are independent of mappings.

Definition 8 *Given a nominal function f on nominal attributes $A_1, A_2, \cdots, A_n$, an attribute mapping vector $\Omega = (\Gamma_1, \Gamma_2, \cdots, \Gamma_n)$ is called a numeric function mapping of f, where Γ_i is the numeric attribute mapping of A_i, $i = 1, 2, \cdots, n$. The resulting numeric function f_Ω is defined below:*

$$f_\Omega(\Gamma_1(A_1), \Gamma_2(A_2), \cdots, \Gamma_n(A_n)) = f(A_1, A_2, \cdots, A_n).$$

After f is mapped to f_Ω in the Euclidean space, geometrically, f_Ω corresponds to an n-dimensional hypercube, and each $(n-1)$-dimensional subfunction $f(a_{ij})$ corresponds to an $(n-1)$-dimensional surface of the hypercube, denoted by H_{ij}. Note that an $(n-1)$-dimensional surface is also an n-dimensional hyperplane. Each assignment of all attributes corresponds to a vertex of the hypercube with a class label. Let W^+ and W^- represent all positive and negative vertices respectively. Then we have the following definition.

Definition 9 *A numeric function f_Ω is linearly separable if there is a hyperplane H to separate W^+ from W^-, where $H = \sum_{i=1}^{n} w_i A_i + w_0$, $w_i \in \mathcal{R}$.*

Geometrically, if f_Ω is linearly separable, then there is an n-dimensional hyperplane slicing the hypercube of f_Ω, such that all positive vertices lie on the one side of the hyperplane and all negative vertices on its opposite side. Therefore, the problem of linear separability becomes the problem in which a hyperplane slices a hypercube [10]. As mentioned earlier, since we only care about the properties independent of mappings, we can freely move a surface H_{ij} of f_Ω (corresponding to an f's $(n-1)$-dimensional subfunction $f(a_{ij})$), which corresponds to assigning a different mapping value to a_{ij}; or we can freely exchange the positions of any two distinct surfaces H_{ij} and H_{ik}, which corresponds to exchanging the mapping values of a_{ij} and a_{ik}. Since moving a surface or exchanging two surfaces corresponds to a different mapping, we will use these two operations in the proof of theorems later.

Definition 10 *(1) A nominal function f is called hard linear, if for any numeric function mapping, the resulting numeric function is linearly separable.*

(2) A nominal function f is called soft nonlinear if it is not a hard linear function, and there exists a numeric function mapping such that the resulting numeric function is linearly separable.

(3) A nominal function f is called hard nonlinear if for any numeric function mapping, the resulting numeric function is not linearly separable.

Definition 10 proposes three layers to measure the linearity of nominal functions, independent of specific mappings. In binary domains, however, only two layers are needed, as given by the following Theorem.

Theorem 1. *There is no soft nonlinear Boolean function in binary domains.*

Proof: Suppose that f is a Boolean function on binary attributes $B_1, \cdots, B_n$, and there is a numeric function mapping $\Omega = (\Gamma_1, \cdots, \Gamma_n)$, where $\Gamma_i(b_{ij}) = x_{ij}$ (b_{ij} is the value of B_i, $i = 1, \cdots, n$, $j = 1, 2$), such that the resulting numeric function f_Ω is linearly separable. Then there is a hyperplane H to separate f_Ω. Assume $H = \sum_{i=1}^{n} w_i x_i + w_0$, $w_i \in \mathcal{R}$.

For any other mapping $\Omega' = (\Gamma'_1, \cdots, \Gamma'_n)$, where $\Gamma'_i(b_{ij}) = x'_{ij}$ ($i = 1, \cdots, n$, $j = 1, 2$), the following hyperplane

$$H' = \sum_{i=1}^{n} w_i\left(\frac{x_i - x'_{i1}}{x'_{i2} - x'_{i1}}(x_{i2} - x_{i1}) + x_{i1}\right) + w_0$$

will separate the resulting function $f_{\Omega'}$.

A straightforward result from Theorem 1 is that Naive Bayes is linear in binary domains regardless of mapping. That means that linearity in binary domains is simple. However, the linearity in nominal domains is more complex. As we will show in the following subsections, the learnability of Naive Bayes in the three layers of linearity is different.

3.2 Hard Linear Nominal Functions

Consider the m-of-n functions. An m-of-n function is a Boolean function that is true if m or more out of n Boolean variables are true. Clearly, if the two values of each Boolean variable are mapped into $\{0, 1\}$, an m-of-n function is linearly separable by the hyperplane $\sum_{i=1}^{n} x_i - m = 0$, where $x_i \in \{0, 1\}$. So it is hard linear by Theorem 1. It can be verified that some m-of-n functions are learnable to Naive Bayes, such as 13-of-25, 30-of-60, 31-of-60 [11]. However, not all m-of-n functions are learnable to Naive Bayes. Domingos and Pazzani [1] showed that for the concept 8-of-25, Naive Bayes gives an incorrect answer of 1 (instead of 0), when just six or seven input Boolean variables are true. This result is independent of mapping, since all variables are Boolean in an m-of-n function.

The above example shows that Naive Bayes can learn only a subset of hard linear nominal functions. Zhang and Ling [11] presented a sufficient and necessary condition for an m-of-n function to be learnable to Naive Bayes. But for an arbitrary linear function, such a condition is still unknown.

3.3 Soft Nonlinear Nominal Functions

Soft nonlinear nominal functions are those functions that are nonlinear under some mappings, but linear under others. The following is an example.

Let $A = \{a_1, a_2, a_3\}$, $B = \{b_1, b_2, b_3\}$, the nominal function f_1 is defined as follows: $f_1(a_1, b_1) = +$, $f_1(a_1, b_3) = +$, $f_1(a_3, b_1) = +$, $f_1(a_3, b_3) = +$, $f_1(a_2, *) = -$, $f_1(*, b_2) = -$, where * means any valid values. Figure 1 (a) is the result of mapping a_i and b_i to i, $i = 1, 2, 3$. It is not linearly separable. However if we map a_1, b_1 to 1, and a_3, b_3 to 2, and a_2 to 5, b_2 to 4, the result, which is linearly separable, is shown in Figure 1 (b). Therefore, f_1 is soft nonlinear.

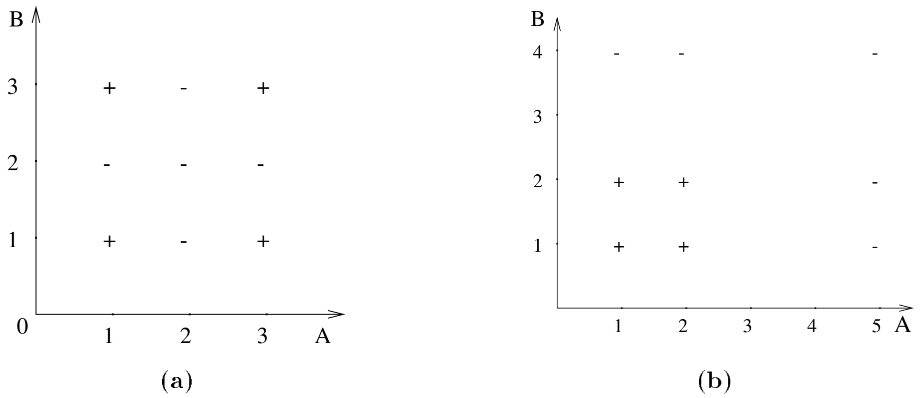


Fig. 1. (a) nonlinear result of f_1 (b) linear result of f_1

Definition 11 *A numeric attribute mapping Γ on a nominal attribute $A = \{a_1, a_2, \cdots, a_m\}$ is called an integer mapping, if Γ is a one-to-one mapping from $\{a_1, a_2, \cdots, a_m\}$ to $\{1, 2, \cdots, m\}$, denoted by Γ_I.*

From Definition 7, we know that Γ_I is a special case of arbitrary Γ, since nominal attribute values are mapped one-to-one into a fixed set of integers. Indeed, application of Naive Bayes on real-world datasets often requires that all attributes be converted into nominal attributes, which are commonly represented internally by integers from 1 to k for k different values. Therefore, integer mapping is a typical and useful mapping in real applications.

Consider a Naive Bayes G on two specific nominal attributes A and B, where $A = \{a_1, a_2, a_3\}$, $B = \{b_1, b_2, b_3\}$, and Table 1 is the conditional probability table (CPT) for A, and B has the same CPT as A. Assume Γ_{IA} and Γ_{IB} are integer mappings from A and B to $\{1, 2, 3\}$ respectively, such that $\Gamma_{IA}(a_i) = i, \Gamma_{IB}(b_i) = i, i = 1, 2, 3$. It is easy to verify that the classifications of G are the same as in Figure 1 (a) after mapped to a two-dimensional Euclidean plane. So, f_1 is learnable to G.

Table 1. The conditional probability table for A.

	$A = a_1$	$A = a_2$	$A = a_3$
$C = -$	0.3	0.4	0.3
$C = +$	0.5	0	0.5

Therefore, Naive Bayes can learn some soft nonlinear nominal functions. This conclusion is quite different from that in binary domains. As we have shown, there are no soft nonlinear functions in binary domains; a function is either hard linear, or hard nonlinear, and Naive Bayes can only learn some of hard linear functions. In nominal domains, however, there are soft nonlinear functions, which are nonlinear under some mappings, and Naive Bayes can actually learn some of them. This may explain why the performance of Naive Bayes is quite good in real-world datasets in which attributes are nominal (since continuous attributes are often discretized to nominal attributes).

3.4 Hard Nonlinear Nominal Functions

Consider the XOR function. Let $A = \{a, \bar{a}\}$, $B = \{b, \bar{b}\}$, the XOR function f_2 is defined as: $f_2(a, b) = +$, $f_2(a, \bar{b}) = -$, $f_2(\bar{a}, b) = -$, $f_2(\bar{a}, \bar{b}) = +$. It is easy to verify that f_2 is hard nonlinear.

We now establish a sufficient and necessary condition for the hard nonlinearity of a function.

Lemma 1. *If an n-dimensional nominal function f has an (n-1)-dimensional subfunction $f(a_{ij})$ with an identical class (called identical subfunction), where a_{ij} is a value of attribute A_i, then f is hard nonlinear if and only if f's partial function f_p on A_1, A_2, $\cdots$, $A_i - \{a_{ij}\}$, $\cdots$, A_n is hard nonlinear.*

Proof: It is obvious that if f_p is hard nonlinear, f is hard nonlinear too.

If f_p is not hard nonlinear, then there is a numeric function mapping Ω, such that after applying it on f_p, the resulting numeric function $f_{p\Omega}$ is linearly separable. Then there is a hyperplane H: $\sum_{j=1}^{n} w_j A_j + w_0 = 0$ to separate $f_{p\Omega}$.

Suppose that the vertices above H have class label $+$, and the vertices under H have class label $-$. Assume that H_{ik} is the surface corresponding to subfunction $f(a_{ik})$, $k = 1, \cdots, m$. We put H_{ij} on the side of H with the same class label and move it any distance from other surfaces H_{ik}, $k \neq j$. If H is exactly perpendicular to H_{ij}, we can adjust H slightly to make it not perpendicular to H_{ij}, while keeping the separation. This is possible since there are only finite vertices on the hypercube. Then it is obvious that if we move H_{ij} far enough, it will be totally on the one side of H, since H is not perpendicular to H_{ij}.

The current positions of H_{ik} for $k = 1, \cdots, m$, correspond to a numeric function transform of f. Applying this transform, the resulting numeric function f_Ω must be linearly separable by H. Therefore, f is not hard nonlinear too.

Lemma 1 shows that the identical surface has no influence on the linearity of a function, so it can be freely removed without affecting the linearity.

Now we begin to discuss the relation between a nominal function containing XOR and its nonlinearity.

Lemma 2. *Suppose f is a 2-dimensional nominal function. If f does not contain an XOR, then f is not hard nonlinear.*

Proof: Suppose that f is a 2-dimensional nominal function on attributes A and B and does not contain an XOR. After applying a numeric function mapping Ω, we get a corresponding numeric function f_Ω. f_Ω can be illustrated by a matrix M on $\mathcal{R}^2$ which consists of a set of vertices with class labels, as shown in Figure 1. Because f does not contain an XOR, it can be proved that M will be empty after successively removing the identical rows and columns of M. According to Lemma 1, f is not hard nonlinear.

Theorem 2 below establishes a close relation between a function containing an XOR and its nonlinearity.

Theorem 2. *An n-dimensional nominal function f is hard nonlinear if and only if it contains an XOR.*

See Appendix 1 for the proof of this theorem.

Interestingly enough, XOR is also related to the linearity of other classifiers. For example, the perceptron is linear since it cannot represent XOR.

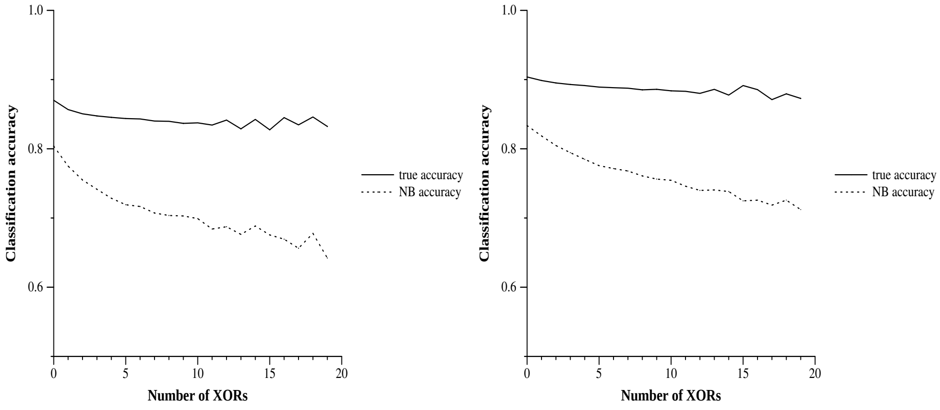
Theorem 3. *Naive Bayes cannot learn any hard nonlinear nominal function.*

Proof: Since Naive Bayes cannot learn any nominal functions containing an XOR and, according to Theorem 2, a nominal function is hard nonlinear if and only if it contains an XOR, Naive Bayes cannot learn any hard nonlinear nominal function.

4 Empirical Experiment

In the above section, we proved that any hard nonlinear function is not learnable to Naive Bayes, and that a nominal function is hard nonlinear if and only if it contains an XOR. According to that result, it is intuitive that the more XORs contained in a nominal function, the more difficulty for it to be learnable to Naive Bayes. We have verified this statement by the empirical experiment below.

We randomly generate an augmented Naive Bayes (ANB) G [4], and count the number of XORs contained in G. Then we draw a dataset from G at random by logical sampling [5]. A Naive Bayes N_G is trained and applied to that dataset by 5-fold cross-validation . Then we can observe the relation between the number of XORs contained in G and the classification accuracy of N_G. We try different ANBs, and the results are similar. The following figure shows the results of two



(a) (b)

Fig. 2. (a) target ANB with six nodes (b) target ANB with eight nodes

cases, in which the ANBs have six nodes and five edges, and eight nodes and seven edges respectively. We repeat the above process 10,000 times in both cases.

In Figure 2, the solid lines plot the classification accuracy of the target ANBs. Note that the accuracy is not 1, because not every example drawn by logical sampling is consistent to the classification of the target ANB. The dotted lines are the classification accuracy of Naive Bayes. Clearly, the difference between the solid and dotted lines increases significantly in both cases, as the number of XORs contained in the target ANB increases. Approximately, the difference grows from six percent to nineteen percent.

The above experiment verifies our expectation that the more XORs contained in a nominal function, the more difficulty for it to be learnable to Naive Bayes. This provides us with the evidence that Naive Bayes cannot learn any hard nonlinear functions.

5 Conclusions

We investigated the linearity of nominal functions by introducing a three-layer measure: hard linear, soft nonlinear, and hard nonlinear. We discussed the learnability of Naive Bayes in term of the three layers of linearity. Our results showed that Naive Bayes can learn some hard linear and some soft nonlinear functions, but it cannot learn any hard nonlinear functions. Our experiment also verified that hard nonlinear functions are more difficult to be learned by Naive Bayes.

Our work establishes two fundamental facts. One is the limitation of Naive Bayes in terms of geometric properties of the nominal functions. The other is that linearity is an appropriate measure for the complexity of nominal functions in Bayesian learning.

From the theoretical results of our paper, Naive Bayes is quite limited in its learnability in nominal domains. A natural question is, what is the reachable upper bound on the learnability of Naive Bayes? That is, how well can Naive Bayes approximate a nonlinear function, even though it cannot represent it perfectly? This is one of our future research topics.

Acknowledgements

We thank our reviewers for their valuable suggestions to improve this paper.

Appendix: Proof of Theorem 2

Suppose that f contains an XOR. By Definition 5, there is a tangent XOR or a diagonal XOR in f. When f is mapped to $\mathcal{R}^n$, there should exist a 2-dimensional plane H_2 containing an XOR. It is obvious that the XOR cannot disappear whatever numeric function mappings are applied. Moreover, for any hyperplane H to separate f, its projection onto H_2 should separate the XOR on H_2. Therefore, such a hyperplane H does not exist. So function f is nonlinear regardless of mappings. That is, f is hard nonlinear.

Suppose f does not contain an XOR. We apply induction on f's dimension n.

Let $n = 2$. From Lemma 2, we know that f is not hard nonlinear.

Suppose when $n = m - 1$, all (m-1)-dimension nominal functions not containing an XOR are not hard nonlinear, where $m \geq 3$.

Consider $n = m$. Suppose f is an m-dimensional nominal function on attributes A_1, A_2, $\cdots$, A_m and does not contain an XOR. Applying a numeric function mapping Ω on f, the resulting numeric function is f_Ω and $f_\Omega(a_{m1})$, $\cdots$, $f_\Omega(a_{mk})$ correspond to all of f's (m-1)-dimensional subfunctions $f(a_{m1})$, $\cdots$, $f(a_{mk})$ on A_m. Suppose all $f(a_{m1})$, $\cdots$, $f(a_{mk})$ are not identical (otherwise we can delete it by Lemma 1).

Since none of $f(a_{m1})$, $\cdots$, $f(a_{mk})$ contains an XOR, $f_\Omega(a_{m1})$, $\cdots$, $f_\Omega(a_{mk})$ are all linearly separable. So there are k (m-1)-dimensional hyperplanes H_1, $\cdots$, H_k to separate $f_\Omega(a_{m1})$, $\cdots$, $f_\Omega(a_{mk})$ respectively.

When $k = 2$, if H_1 and H_2 are parallel, we can construct an m-dimensional hyperplane H from H_1 and H_2 to separate f_Ω. When H_1 and H_2 have the same orientation, [1] we can also construct an m-dimensional hyperplane H from H_1 and H_2 to separate f_Ω. When H_1 and H_2 have different orientation, f should contain an XOR.

If $k > 2$, H_1, H_2, $\cdots$, H_k should have the same orientation. Then we can sort H_1, H_2, $\cdots$, H_k in an decreasing order of the vertex number above each hyperplane. Suppose that the resulting sequence is H_{k1}, H_{k2}, $\cdots$, H_{kk}. We construct

[1] Here orientation is the angle of the intersection of H_i and the hyperplane perpendicular to $f_\Omega(a_{mi})$

an m-dimensional hyperplane H from H_{k1}, H_{k2}. Then for each H_{ki} from $i = 3$ to k, since the number of vertices above H_{ki} is not greater than those above $H_{k1}, \cdots, H_{k(i-1)}$ and there are only finite vertices, it is easy to adjust the angle of H by moving $H_{k1}, \cdots, H_{k(i-1)}$ away from H_{ki} to separate $f_\Omega(a_{mk_i})$ while separating $f_\Omega(a_{mk_1}), \cdots, f_\Omega(a_{mk_{(i-1)}})$. Since k is finite, the hyperplane H constructed above will separate all $H_{k1}, H_{k2}, \cdots, H_{kk}$. So f_Ω is linearly separable. Therefore, f is not hard nonlinear.

References

1. Domingos P., Pazzani M.: Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier. Machine Learning **29** (1997) 103-130
2. Duda R. O., Hart P. E.: Pattern Classification and Scene Analysis. A Wiley-Interscience Publication (1973)
3. Frank E., Trigg L., Holmes G., Witten I. H.: Naive Bayes for Regression. Machine Learning **41(1)** (2000) 5-15
4. Friedman N., Greiger D., Goldszmidt M.: Bayesian Network Classifiers. Machine Learning **29** (1997) 103–130
5. Henrion M.: Propagation of Uncertainty in Bayesian Networks by Probabilistic Logic Sampling. In Lemmer J. F. and Kanal L. N. (Eds): Uncertainty in Artificial Intelligence 2. Elsevier/North-Holland (1988) 149-163.
6. Kononenko I.: Comparison of Inductive and Naive Bayesian Learning Approaches to Automatic Knowledge Acquisition. Current Trends in Knowledge Acquisition. IOS Press (1990)
7. Langley P., Iba W., Thomas K.: An Analysis of Bayesian Classifiers. Proceedings of the Tenth National Conference of Artificial Intelligence. AAAI Press (1992) 223-228
8. Peot M. A.: Geometric Implications of the Naive Bayes Assumption. Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence. Morgan Kaufmann (1996).
9. Quinlan J. R.: C4.5: Programs for Machine Learning. Morgan Kaufmann: San Mateo, CA (1993)
10. Saks M. E.: Slicing the Hypercube. Survey in Combinatorics 1993. Cambridge University Press (1993) 211-255
11. Zhang H., Ling, C. X., Zhao Z.: The Learnability of Naive Bayes. In Hamilton H. J. and Yang Q. (Eds.): Advances in Artificial Intelligence. Springer (2000) 432-441
12. Zhang H., Ling, C. X.: Learnability of Augmented Naive Bayes in Nominal Domains. Proceedings of the Eighteenth International Conference on Machine Learning (to appear). Morgan Kaufmann (2001)