Classification based on contextual probability

Hui Wang
School of Computing and Mathematics
University of Ulster at Jordanstown
BT37 0QB, Northern Ireland, UK
h.wang@ulster.ac.uk

Sally McClean
School of Computing and Information Engineering
University of Ulster at Coleraine
BT52 1SA, Northern Ireland, UK
si.mcclean@ulster.ac.uk

Abstract

Data Mining is often concerned with large and complex datasets. However the data space may be sparse and contain regions that are poorly covered. In such cases we would like to make inferences about the sparser regions using data from the denser regions. Another fundamental challenge with multivariate data is that a complete probabilistic model involves a state space that grows exponentially with the number of variables, rapidly becoming computationally intractable.

Current solutions to these problems include model-based, or parametric, approaches that may have relatively high bias when the data do not obey the assumed functional form of the model, and non-parametric methods that often include measures of locality that may affect performance and lead to additional algorithm complexity. In addition non-parametric methods are often ad-hoc and may thus fail to provide generic solutions that relate well to well-founded concepts such as probability.

We develop a principled approach that utilises a novel definition of contextual probability, which we show is a well-founded distribution, termed the G-contextual probability function. We also provide an efficient algorithm for the estimation of G, which may be utilised for sparse as well as dense areas of the data space. Based on this estimate we propose a classification procedure that is polynomial in the number of data instances and constitutes a lazy approach, as it does not build models. Finally the approach is evaluated and shown to compare favourably with well-known methods (C5.0 and SVM) using a range of standard datasets from a variety of domains.

Keywords: Data mining, classification, contextual probability, sparse data

1 Introduction

Data Mining is often concerned with large and complex datasets. However the data space may be sparse in the sense that it contains regions that are poorly covered by the data. In such cases we would like *inter alia* to make inferences about the sparse regions using data from the dense regions. Another fundamental challenge with multivariate data is that a complete probabilistic model involves a state space that grows exponentially with the number of variables. Therefore it becomes computationally impracticable to estimate the complete joint probability distribution for a large set of variables.

One way to overcome such problems is to use a model-based, or parametric, approach. Parametric models assume a particular functional form for the distribution such as a Gaussian. Such parametric models are often characterised by a small number of parameters e.g. the mean and variance-covariance parameters in the Gaussian case. These models have the advantage of simplicity (easy to estimate and interpret). They are also able to fit the model to one (dense) region of the data space and use it for another (sparse) region. However parametric models may have relatively high bias when the data do not obey the assumed functional form of the model [2].

Another solution is through non-parametric estimation. In non-parametric models the probability distribution estimation is data-driven and relatively few assumptions are made *a priori* about the functional form. Histograms, kernel methods and k-nearest neighbours (kNN) are all examples of non-parametric methods, where a histogram is a relatively primitive version of a kernel method, and kNN is a special case of a kernel method [4]. Kernel methods are based on the assumption that the density function is constant locally; however the extent of the locality is prescribed by parameters, which must be specified or estimated; such parameters have a critical bearing on the performance of the methods and may lead to additional complexity of the algorithms. In addition non-parametric methods are often ad-hoc and may thus fail to provide generic solutions that relate well to well- founded concepts such as probability.

With these ideas in mind, we here develop a principled approach that utilises a novel definition of contextual probability, which we show is a well-founded probability distribution, which we term the G-contextual probability function. Our definition of contextual probability allows all sets that overlap with the set for which we seek to determine the probability, to contribute to the estimation. We are hence able to include information from all possible neighbourhoods (the context). Thus G is different from the classical frequentist probability P; however we will show that G is a linear function of P. We also provide an efficient algorithm for the estimation of G, which may be utilised for sparse as well as dense areas of the data space. This estimator provides an approximation to G, where we show that classification on the basis of G is equivalent to classification based on P. However by basing our classification on the estimator for G rather

than P, the classification algorithm is much more efficient and can handle sparse regions in an intuitive manner.

Finally the approach is evaluated and shown to compare favourably with two well known methods (C5.0 and SVM) using a range of standard datasets from a variety of domains.

2 Related work

We review four multivariate classification models that are related to ours one way or another. Data are described by a set R of n *predictor attribute (variables)* and a discrete *class attribute*. Classification learning is simplified as building a mapping from the predictor attributes to the class attribute subject to some constraints.

The review is mainly based on [13, 8] and `http://www.faqs.org/faqs/ai-faq`.

2.1 The Bayes classifier

In a Bayes classifier [10, 13, 12], the distribution of values of the predictor attributes given any class is assumed to be known exactly, and the prior probabilities of the classes are assumed known, so that the posterior probabilities can be computed by an application of Bayes' theorem. The Bayes classifier involves no learning – we know everything that needs to be known! The Bayes classifier is a gold standard that can almost never be used in real life but is useful in theoretical work and in simulation studies that compare classification methods.

2.2 The naive Bayes classifier

The naive Bayes classifier [16, 13, 12, 14] assumes that the predictor attributes are conditionally independent given the class attribute. The naive Bayes classifier is usually applied with categorical inputs, and the distribution of each predictor attribute is estimated by the proportions in the training set; hence the naive Bayes classifier is a frequentist method.

Let V be a data space defined by n predictor attributes. Each instance $x \in V$ is then a vector of predictor attribute values $\langle v_1, v_2, \cdots, v_n \rangle$ and that the discrete valued classification function we are trying to learn is $f : V \to C$ where $C = \{c_1, c_2, \cdots, c_m\}$ is the set of class values.

A set of N training instances of the form $\langle x_i, c_i \rangle$ is presented followed by a new instance x. What is the most likely value of $f(x)$?

$$c_* = \arg \max_{c_j \in C} P(x|c_j)P(c_j) \tag{1}$$

$$= \arg \max_{c_j \in C} P(v_1, v_2, \cdots, v_n|c_j)P(c_j) \tag{2}$$

The prior probability of c_j is easy to estimate as the relative frequency of c_j in the training instances. But what about $P(x|c_j)$? To estimate this we need to count the number of times the n-tuple $\langle v_1, v_2, \cdots, v_n \rangle$ occurs in the subset of the training data for which $f(x) = c_j$. Technically this is infeasible.

So we make a simplifying assumption that the v_i are conditionally independent of each other given c_j. That is, $P(v_1, v_2,$
$\cdots, v_n|c_j) = \prod_{i=1}^{n} P(v_i|c_j)$. If we substitute the above in Eq. 2, we get the expression for the classification given by the naive Bayes classifier:

$$c_{NB} = \arg \max_{c_j \in C} \prod_{i=1}^{n} P(v_i|c_j)P(c_j) \tag{3}$$

Note that $P(v_i|c_j)$ can be easily estimated as the ratio of the frequency of the value v_i in the subset of the training instances for which $f(x) = c_j$ to the size of this subset.

2.3 Kernel methods for classification

Most non-parametric methods are based on the assumption that a function is locally constant. This implies that in order to classify a data instance, we only need to look at those instances with known classification that are close to this instance, in other words, those in the neighbourhood of this instance.

Much of the difficulty in the use of these methods is deciding what is meant by "local" in a high dimensional space [13]. Two prominent methods are *kernel methods* and *k-nearest neighbour methods*.

Kernel methods [13, 8] aim to model the class-conditional densities in many dimensions. A kernel Λ is a bounded function on a data space with integral one. Suitable examples include probability density functions such as the multivariate normal.

It is usually assumed that Λ is in some sense peaked about 0. Then $\Lambda(x-y)$ is used as a measure of the proximity of x and y. A commonly used assumption is $\Lambda(-x) = \Lambda(x)$. The empirical distribution of x within a class j gives mass $1/N_j$ to each of the instances. This suggests that a local estimate of the density $p_j(x)$ can be found by summing each of these contributions with weight $\Lambda(x - x_i)$, that is

$$\bar{p}_j(x) = \frac{1}{N_j} \sum_i \Lambda(x - x_i)$$

and this can also be interpreted as an average of kernel functions centred on each instance from the class.

When the prior probabilities are estimated by N_j/N we have

$$\bar{p}(j|x) = \frac{\sum_{[i]=j} \Lambda(x - x_i)}{\sum_i \Lambda(x - x_i)},$$

the weighted proportion of instances around x which have class j.

The difficulty with kernel methods is the choice of Λ, especially if some attributes are categorical. Kernel methods (in fact, any density estimation methods) need a very large training set to be successful [13].

2.4 The k-nearest neighbour classifier

The k-Nearest neighbour methods [11, 6, 1, 3] are closely related to the kernel methods for classification. The basic kernel method defines a cell by a fixed bandwidth and calculates the proportion of instances within this cell that belong to each class. This means that the denominator in the proportion is a random variable. The basic k-nearest neighbour method fixes the proportion (at k/N) and lets the bandwidth be a random variable. More sophisticated extensions of both methods (for example, smoothly decaying kernel functions, differential weights on the nearest neighbour instances according to their distance from x, or choice of bandwidth that varies according to x) often lead to methods that are barely distinguishable in practice [8].

k-Nearest neighbour methods are a special case of the kernel methods in the sense that the kernel Λ is specified such that only nearest neighbours have constant non-zero values.

k-Nearest neighbour methods are critically dependent on the distance metric and the selection of k.

3 Classical probability

Probability theory is the body of knowledge that enables us to reason formally about uncertain events or propositions. There are different approaches to probability theory, most notably the *frequentist* and *Bayesian* approaches [7, 2, 9].

In the frequentist point of view, the probability of an event is taken to be equal to the limit of the relative frequency of the chosen event with respect to all possible events as the number of trials goes to infinity. The appeal of the frequentist approach for scientists lies in the apparent objectivity of its treatment of data.

On the other hand, the Bayesian approach extends the interpretation of probability to include degrees of belief or knowledge in propositions. We pass from the probability of events (frequentist) to the probability of propositions (Bayesian). Nevertheless the axioms used to define the mathematical properties of probability remain unchanged. Consequently many of the statistical procedures of the two approaches are identical.

Here we focus on the mathematical properties of probability. In particular we take probability to be defined in terms of probability distribution. Let V be a set consisting of the basic elements of interest in a domain. A probability distribution function is $p : V \to [0, 1]$ such that $\sum_{x \in V} p(x) = 1$. A (classical)

probability function is $P : 2^V \to [0,1]$ such that, for any $E \subseteq V$,

$$P(E) = \sum_{x \in E} p(x). \tag{4}$$

$P(E)$ is the probability that an arbitrary element $x \in V$ belongs to a well-defined subset $E \subseteq V$.

The above probability function satisfies the *Axioms of Probability*: for any *event* $E \subseteq V$,

- $P(E) \geq 0$.
- $P(V) = 1$.
- If $E_1 \cap E_2 = \emptyset$ then $P(E_1 \cup E_2) = P(E_1) + P(E_2)$.

In general any function satisfying the Axioms of Probability, however defined, is a probability function [7].

4 Contextual probability

Let V be a **finite** set. We define a function $G : 2^V \to [0,1]$ such that for any $E \subseteq V$

$$G(E) = \sum_{X \subseteq V} \frac{P(X)}{K} \times \frac{|E \cap X|}{|X|}, \tag{5}$$

where $|X|$ is the number of elements in set X, and K is a normalising factor such that $\sum_{X \subseteq V} P(X)/K = 1$.

An interpretation of this function is as follows. Our knowledge about (or belief in) X is contained in the probability function P and represented by $P(X)$. A fraction of $P(X)$ is attributed to E. Since we do not know how $P(X)$ is distributed among the components in X, we can assume it is evenly distributed over its components according to the *principle of indifference*. The sum of the fractions of $P(X)$ for all possible X attributable to E gives us the G value for E.

Theorem 1. *G is a probability function on V. That is to say,*

1. *For any $E \subseteq V$, $G(E) \geq 0$;*
2. *$G(V) = 1$;*
3. *For $E_1, E_2 \in V$, $G(E_1 \cup E_2) = G(E_1) + G(E_2)$ if $E_1 \cap E_2 = \emptyset$.*

Proof. The first claim is true following the fact that $P(X) \geq 0$ for any $X \subseteq V$. The equation holds when $E = \emptyset$.

The second claim is true since $G(V) = \sum_{X \subseteq V} P(X)/K = 1$.

Let's now consider the third claim. $X \cap (E_1 \cup E_2) = (X \cap E_1) \cup (X \cap E_2)$. If $E_1 \cap E_2 = \emptyset$ then $|X \cap (E_1 \cup E_2)| = |X \cap E_1| + |X \cap E_2|$. As a result we have

$$\begin{aligned} G(E_1 \cup E_2) &= \sum_{X \subseteq V} \frac{P(X)}{K} \times \frac{|X \cap (E_1 \cup E_2)|}{|X|} \\ &= \sum_{X \subseteq V} \frac{P(X)}{K} \times \frac{|X \cap E_1| + |X \cap E_2|}{|X|} \\ &= \sum_{X \subseteq V} \frac{P(X)}{K} \times \frac{|X \cap E_1|}{|X|} + \sum_{X \subseteq V} \frac{P(X)}{K} \times \frac{|X \cap E_2|}{|X|} \\ &= G(E_1) + G(E_2) \end{aligned}$$

□

We therefore call G a *contextual probability function* [1]. We can define *conditional contextual probability* in the usual way. For $E_1, E_2 \subseteq V$, the conditional (contextual) probability of E_1 given E_2 is

$$G(E_1|E_2) = \frac{G(E_1 \cap E_2)}{G(E_2)}.$$

For simplicity, if E is a singleton set, e.g., $E = \{a\}$, we write $G(a)$ for $G(\{a\})$.

Now we investigate the relationship between $G(E)$ and $P(E)$ for $E \subseteq V$. Let $\binom{N}{n}$ be the combinatorial number representing the number of ways of picking n unordered outcomes from N possibilities. From Combinatorial Theory we know that $\binom{N}{n} = \frac{N!}{(N-n)!n!}$.

Lemma 1. *Let $N = |V|$. Then $K = \sum_{i=1}^{N} \binom{N-1}{i-1} = 2^{N-1}$.*

Proof.

$$\begin{aligned} K &= \sum_{X \subseteq V} P(X) = \sum_{i=1}^{N} \sum_{X \subseteq V, |X|=i} P(X) \\ &= \sum_{i=1}^{N} \sum_{x \in V} \tbinom{N-1}{i-1} p(x) = \sum_{i=1}^{N} \tbinom{N-1}{i-1} \sum_{x \in V} p(x) \\ &= \sum_{i=1}^{N} \tbinom{N-1}{i-1} = 2^{N-1} \end{aligned}$$

□

Theorem 2. *Let $\alpha \stackrel{\text{def}}{=} \frac{1}{K} \sum_{i=1}^{N} \frac{\binom{N-2}{i-1}}{i}$, and $\beta \stackrel{\text{def}}{=} \frac{1}{K} \sum_{i=1}^{N} \frac{\binom{N-2}{i-2}}{i}$. Then $G(x) = \alpha p(x) + \beta$ for $x \in V$.*

[1] We note that the use of the term "contextual probability" here is different and more general than that used previously in computational linguistics, e.g. [4], where such probabilities relate to tagged grammars.

Proof.

$$\begin{aligned}
G(x) &= \sum_{Y\subseteq V} \frac{x\cap Y}{|Y|}\frac{P(Y)}{K} = \sum_{Y\subseteq V, x\in Y} \frac{1}{|Y|}\frac{P(Y)}{K} \\
&= \frac{1}{K}\sum_{Y\subseteq V, x\in Y} \frac{P(Y)}{|Y|} = \frac{1}{K}\sum_{Y\subseteq V, x\in Y} \frac{\sum_{z\in Y} p(z)}{|Y|} \\
&= \frac{1}{K}\sum_{i=1}^{N}\frac{1}{i}\sum_{Y\subseteq V, |Y|=i, x\in Y}\sum_{z\in Y} p(z) \\
&= \frac{1}{K}\sum_{i=1}^{N}\frac{1}{i}\left(\tbinom{N-1}{i-1}p(x) + \tbinom{N-2}{i-2}\sum_{z\neq x} p(z)\right) \\
&= \frac{1}{K}\sum_{i=1}^{N}\frac{1}{i}\left(\tbinom{N-1}{i-1}p(x) + \tbinom{N-2}{i-2}(1-p(x))\right) \\
&= \frac{1}{K}\sum_{i=1}^{N}\frac{1}{i}\left(\tbinom{N-2}{i-1}p(x) + \tbinom{N-2}{i-2}\right) = \alpha p(x) + \beta
\end{aligned}$$

The claim then follows. □

Since both P and G are probability functions we have $\sum_{x\in V} P(x) = 1$ and $\sum_{x\in V} G(x) = 1$. According to Theorem 2 we then have

Corollary 2. $\alpha + |V| \times \beta = 1$.

As a result we only need to calculate either of α and β, and the other can be determined according to the corollary.

Since both P and G are probability functions they satisfy the additive axiom. In other words for $E \subseteq V$, $P(E) = \sum_{x\in E} p(x)$ and $G(E) = \sum_{x\in E} G(x)$. Following Theorem 2 we then have:

Corollary 3.

$$G(E) = \alpha P(E) + \beta|E|$$

Theorem 2 and Corollary 3 establish the relationship between G and P. If we know P we can calculate G, and vice versa.

Lemma 4. $\alpha = \frac{2^{N-1}-1}{2^{N-1}\times(N-1)}$ *and* $\beta = \frac{N\times 2^{N-1}-2^N+1}{N\times(N-1)\times 2^{N-1}}$.

Proof. It can be shown that

$$\begin{aligned}
\alpha =& \frac{1}{K}\sum_{i=1}^{N}\frac{\binom{N-2}{i-1}}{i} = \frac{1}{2^{N-1}}\sum_{i=1}^{N}\frac{(N-2)!}{i\times(i-1)!\times(N-i-1)!} \\
=& \frac{1}{2^{N-1}}\sum_{i=1}^{N}\frac{(N-1)!}{i!\times(N-i-1)!\times(N-1)} \\
=& \frac{1}{2^{N-1}\times(N-1)}\sum_{i=1}^{N}\tbinom{N-1}{i} = \frac{2^{N-1}-1}{2^{N-1}\times(N-1)}
\end{aligned}$$

E	$\emptyset$	$\{a\}$	$\{b\}$	$\{c\}$	$\{d\}$	$\{a,b\}$	$\{a,c\}$	$\{a,d\}$
P	0	0.1	0.3	0.4	0.2	0.4	0.5	0.3
G	0	198/960	254/960	282/960	226/960	452/960	480/960	424/960
E	$\{b,c\}$	$\{b,d\}$	$\{c,d\}$	$\{a,b,c\}$	$\{a,b,d\}$	$\{a,c,d\}$	$\{b,c,d\}$	$\{a,b,c,d\}$
P	0.7	0.5	0.6	0.8	0.6	0.7	0.9	1.0
G	536/960	480/960	508/960	734/960	678/960	706/960	762/960	1.0

Table 1: The set is $V = \{a, b, c, d\}$. The probability distribution is assumed to be $\{a : 0.1, b : 0.3, c : 0.4, d : 0.2\}$.

According to Corollary 2 we have

$$\beta = \frac{N \times 2^{N-1} - 2^N + 1}{N \times (N-1) \times 2^{N-1}}$$

□

If N is large α and β can be approximated as follows:

$$\alpha = \frac{2^{N-1}}{2^{N-1} \times (N-1)} = \frac{1}{N-1}$$
$$\beta = \frac{N \times 2^{N-1} - 2^N}{N \times (N-1) \times 2^{N-1}} = \frac{N-2}{N(N-1)}$$

Example 1. *Consider a set $\{a, b, c, d\}$, whose probability distribution is assumed to be $\{0.1, 0.3, 0.4, 0.2\}$. Following definition, the P and G values can be calculated for all the subsets and are shown in Table 1.*

Clearly the G values for singleton subsets are slightly different from the those P values given in probability distribution.

It can be verified that $G(E) = \alpha P(E) + \beta|E|$ for any $E \subseteq V$. Note that $N = 4$ and hence $K = 2^{N-1} = 2^3 = 8$ according to Lemma 1. Note also that $\alpha = 28/96$ and $\beta = 17/96$.

5 Classification based on probability

In this section we discuss the task of classification and how probability is usually used as a basis for classification.

Let V be a finite data space. The instances in V can be labelled by a finite set C of class values. The correspondence between V and C is represented by a classification function (or simply put, classifier) $f : V \to C$.

In a typical classification learning task we are given a finite dataset D, which is usually considered a sample of V according to a probability distribution P over $V \times C$. The labelling of elements in D can be represented by a function $f' : D \to C$. The aim of classification learning is to build a classifier $\bar{f} : V \to C$

such that $\bar{f}(x) = f'(x)$ if $x \in D$. $\bar{f}$ is consistent with D, and serves as an approximation to the unknown "true" classification function f.

If we know probability P, we can classify data instances according to the following rule - this is in fact the Bayes classifier (Section 2.1).

Rule 1. *For $x \in V$, $f(x) = \arg\ \max_{c \in C} P(c|x)$.*

Similarly, if we know contextual probability G, we can classify according to the following rule

Rule 2. *For $x \in V$, $f(x) = \arg\ \max_{c \in C} G(c|x)$.*

An important question is, are the above two rules equivalent for classification? The following theorem answers this question.

Theorem 3. *Let $t \in V$, and $c_1, c_2 \in C$. Then $P(c_1|t) \leq P(c_2|t) \iff G(c_1|t) \leq G(c_2|t)$.*

Proof.

$$
\begin{aligned}
& G(c_1|t) \leq G(c_2|t) && \iff \\
& \frac{G(t,c_1)}{G(t)} \leq \frac{G(t,c_2)}{G(t)} && \iff \\
& G(t,c_1) \leq G(t,c_2) && \iff \\
& \alpha P(t,c_1) + \beta \leq \alpha P(t,c_2) + \beta && \iff \\
& P(t,c_1) \leq P(t,c_2) && \iff \\
& \frac{P(t,c_1)}{P(t)} \leq \frac{P(t,c_2)}{P(t)} && \iff \\
& P(c_1|t) \leq P(c_2|t)
\end{aligned}
$$

□

This theorem effectively says that Rules 2 and 1 are equivalent. The advantage of Rule 2 is that G can be estimated with less dependence on some locality parameters and probably higher resilience to sparsity of data.

6 Estimating contextual probability for classification

In practical applications the probability distribution is usually not known. In this section we discuss how to estimate contextual probability from data for classification.

In order to calculate $G(t)$ for $t \in V$ we need to consider all $E \in V$ such that $t \in E$, i.e., all neighbourhoods of t, along with the probability $P(E)$. However in a classification task we only know a sample D of V, therefore we can estimate $G(t)$ by considering all neighbourhoods of t in D.

Let $\mathcal{E}(t) = \{E \subseteq D : t \in E\}$. Each element of $\mathcal{E}(t)$ is a neighbourhood of t, so $\mathcal{E}(t)$ is the set of all neighbourhoods of t.

Now we discuss how to estimate $P(E)$ for $E \in \mathcal{E}(t)$. Given no further knowledge about V or D we can apply the *principle of indifference* and assume that all elements in D have equal probability - a common practice. For a class $c \in C$ let $E^c = \{x \in E : f'(x) = c\}$. Recall that f' is the classification function implied in dataset D. Then we have

$$\begin{aligned} P(E,c) &= |E^c|/|D| \\ P(E) &= \sum_{c \in C} P(E,c) = |E|/|D| \\ P(c|E) &= P(E,c)/P(E) = |E^c|/|E| \end{aligned}$$

According to the definition in Eq.5 we calculate $G(t,c)$, $G(t)$ and $G(c|t)$ as follows. Note that $\{t\} \cap E = \{t\}$ and $|\{t\}| = 1$.

$$G(t,c) = \sum_{E \in \mathcal{E}(t)} \frac{P(E,c)}{K \times |E|} = \sum_{E \in \mathcal{E}(t)} \frac{|E^c|}{K \times |D| \times |E|} \tag{6}$$

$$G(t) = \sum_{c \in C} G(t,c) \tag{7}$$

$$G(c|t) = G(t,c)/G(t) \tag{8}$$

$$= \frac{\sum_{E \in \mathcal{E}(t)} |E^c|/(K \times |D| \times |E|)}{G(t)} \tag{9}$$

$$= \frac{\sum_{E \in \mathcal{E}(t)} |E^c|/|E|}{K \times |D| \times G(t)} = \delta(t) \sum_{E \in \mathcal{E}(t)} |E^c|/|E| \tag{10}$$

where K is a normalising factor as discussed before and $\delta(t)^{-1} = |D| \times K \times G(t)$.

It is worth noting that Eq.10 is not tied up with any specific interpretation of neighbourhood. We can use the usual distance-based interpretation, which is widely adopted in kNN studies. Section 7.1 will detail our interpretation of neighbourhood.

Once we get an estimate of $G(c|t)$ we can use Rule 2 for classification. If we consider only one neighbourhood E we have $G(c|t) = \delta(t)|E^c|/|E|$; as a result the classification rule is effectively the voting kNN rule if neighbourhood E is specified as the set of nearest neighbours in the usual distance-based interpretation. In this sense we can understand our classification rule as a generalisation of the voting kNN rule.

If we consider all neighbourhoods we will end up with an unbiased estimate of G. As the dataset grows larger we will get an estimate closer to the true G and in the limit we will get the true G – a frequentist view.

If dataset D is large, there are a very large number of neighbourhoods for t. This makes estimating the contextual probability with all neighbourhoods being computationally difficult. The next section will present a solution to reduce the number of neighbourhoods along with a specification of neighbourhood.

7 An algorithm for G-based classification

The classification rules above are theoretical framework. They serve as a guide in designing practical algorithms, where the notion of neighbourhood must be interpreted, the selection of neighbourhoods must be given, and the formula for G must be specified based on the chosen neighbourhood. Clearly the notion of neighbourhood can be interpreted in different ways.

In this section we present our interpretation of neighbourhood and a classification algorithm based on this interpretation. The algorithm does not use all neighbourhoods; instead it uses a small number of neighbourhoods defined by the given data instances and the to-be-classified instance.

7.1 Hypertuples as neighbourhoods

To calculate $G(t)$ for $t \in V$ we need to consider $E \subseteq V$ such that $t \in E$. Consider a one-dimensional example: $E = \{1, 2, 4, 5\}$ and $t = 3$. If E is interpreted as a discrete set it is clear that $t \notin E$. However if E is interpreted as an interval $[1, 5]$ supported by the given numbers we then have $t \in E$ in the sense that $1 \leq t$ and $t \leq 5$. Therefore it is necessary to interpret E and the operation $t \in E$ in a more general context. Our interpretation is based on the notion of hypertuple [15].

Let $R = \{a_1, a_2, \cdots, a_n\}$ be the set of predictor attributes used to describe the data space V, and y be the class attribute. Furthermore, let $dom(a)$ be the domain of attribute $a \in R \cup \{y\}$. Then $C = dom(y)$, and the data space is defined by $V \stackrel{\text{def}}{=} \prod_{i=1}^{n} dom(a_i)$. If we write a data instance t by $\langle v_1, v_2, \cdots, v_n \rangle$ then $v_i \in dom(a_i)$. A *hypertuple* h is a member of $\prod_{i=1}^{n} 2^{dom(a_i)}$. If we write h by $\langle s_1, s_2, \cdots, s_n \rangle$ then $s_i \in 2^{dom(a_i)}$ or $s_i \subseteq dom(a_i)$. The key difference between an instance (or tuple) and a hypertuple is that a field in an instance is a value (hence *value-based*) while a field in a hypertuple is a set (hence *set-based*). If we interpret $v_i \in dom(a_i)$ as a singleton set $\{v_i\}$, then an instance t is a special hypertuple.

Consider two hypertuples h_1 and h_2, where $h_1 = \langle s_{11}, s_{12},$
$\cdots, s_{1n} \rangle$ and $h_2 = \langle s_{21}, s_{22}, \cdots, s_{2n} \rangle$. h_1 is *covered* by h_2, written $h_1 \leq h_2$, if for $i \in \{1, 2, \cdots, n\}$,

$$\begin{cases} \forall x \in s_{1i}, \min(s_{2i}) \leq x \leq \max(s_{2i}) & \text{if } a_i \text{ is ordinal} \\ s_{1i} \subseteq s_{2i} & \text{if } a_i \text{ is categorical} \end{cases} \tag{11}$$

Furthermore the *sum* of h_1 and h_2, written by $h_1 + h_2 \stackrel{\text{def}}{=} \langle s_1, s_2, \cdots, s_n \rangle$, is: for each $i \in \{1, 2, \cdots, n\}$,

$$s_i = \begin{cases} \{x \in dom(a_i) : & \min(s_{1i} \cup s_{2i}) \leq x \leq \max(s_{1i} \cup s_{2i})\}, \\ & \text{if } a_i \text{ is ordinal} \\ s_{1i} \cup s_{2i}, & \text{if } a_i \text{ is categorical} \end{cases} \tag{12}$$

Clearly $h_1 \leq h_1 + h$, where h is any hypertuple.

Table (a) below shows a set of data instances (tuples) and Table (b) a set of hypertuples.

a_1	a_2	y
a	0	α
a	1	α
b	2	β

(a)

a_1	a_2	y
$\{a\}$	$\{0,1\}$	α
$\{b\}$	$\{2\}$	β

(b)

Now we look at the original question: how to interpret $E \subseteq V$ and the operation $t \in E$. We interpret E as a hypertuple $\sum_{x \in E} x$, $|E|$ as the number of elements in E or the *coverage* of E, and the operation $t \in E \Longleftrightarrow t \leq \sum_{x \in E} x$. Note that t is a special hypertuple, and $\leq$ and $\sum$ are defined in Eq.11 and Eq.12 respectively. E is in fact a hyper rectangular region surrounding t, which is our interpretation of a neighbourhood of t.

7.2 The algorithm

Given a data instance some neighbourhoods are very large: some elements within them are "similar" to t and some are not, and overall most of the elements are not very "similar". Therefore we can ignore those "not-very-similar" neighbourhoods without generating much bias in the estimation. In other words, we consider only a small number of neighbourhoods such that each contains elements fairly close to t. There may be different ways of doing this. One obvious way is to adopt the usual distance-based interpretation of neighbourhood, and consider $E_1, E_2, \cdots, E_m$ in estimating G. Here E_i contains i nearest neighbours. Clearly $E_i \subseteq E_{i+j}$ for $j \geq 0$. In this approach we need to specify the parameter m – the number of neighbourhoods to be used.

Here we adopt the hypertuple-based interpretation of neighbourhood since both categorical and ordinal data can be treated in a uniform way [15]. Our solution is presented in the following algorithm.

Let $t \in V$ be a data instance to be classified. The following algorithm calculates $G(c|t)$ for all $c \in C$, and classify according to Rule 2. The algorithm is called *CP-based classifier* or *CPC* for short.

Algorithm [CPC]

- Set $\mathcal{E}(t) = \emptyset$.
- For each $x \in D$
  - calculate $t + x$;
  - calculate $E(t, x) \stackrel{\text{def}}{=} \{y \in D : y \leq t + x\}$;
  - add $E(t, x)$ to $\mathcal{E}(t)$;

- Calculate $G(c|t)$ according to Eq.10.
- Classify t according to Rule 2.

□

Since $t \leq t + x$ it is clear that $E(t, x)$ is a neighbourhood of t and hence $\mathcal{E}(t)$ is a selection of neighbourhoods.

The complexity of the algorithm is dominated by the loop, which runs $|D|$ times. The complexity of each loop is dominated by the operation to calculate $E(t, x)$, which at worst needs to look at all $|D|$ instances in the dataset. Therefore the complexity of the algorithm is at worst $O(|D|^2)$.

7.3 Examples

We consider two examples here. Example 2 illustrates classification with all neighbourhoods, and Example 3 illustrates the CPC algorithm.

Example 2. *Suppose Table 7.1(a) is the dataset D. Assume that a_1 and a_2 are predictor attributes and y is the class attribute. Assume further that a_1 and y are categorical, and a_2 is ordinal; $dom(a_1) = \{a, b\}$, $dom(y) = \{\alpha, \beta\}$, and $dom(a_2) = [0, 2]$ – an interval. Then $V = dom(a_1) \times dom(a_2)$. Our question is: given D as the only source of information, what is the most likely class value of $t = \langle b, 1\rangle$?*

To answer this question we can use the contextual probability. In order to find all neighbourhoods of t we consider all hypertuples in 2^D and remove those that do not cover t. This leads to the following hypertuples along with their coverage. Those with $*$ are neighbourhoods of t.

hypertuple	*coverage*	$\lvert E^\alpha \rvert$	$\lvert E^\beta \rvert$	
$\emptyset$	*0*	*0*	*0*	
$\langle a, 0\rangle$	*1*	*1*	*0*	
$\langle a, 1\rangle$	*1*	*1*	*0*	
$\langle b, 2\rangle$	*1*	*0*	*1*	
$\langle a, \{0, 1\}\rangle$	*2*	*2*	*0*	
$\langle \{a, b\}, \{0, 2\}\rangle$	*3*	*2*	*1*	$*$
$\langle \{a, b\}, \{1, 2\}\rangle$	*2*	*1*	*1*	$*$
$\langle \{a, b\}, \{0, 1, 2\}\rangle$	*3*	*2*	*1*	$*$

Then we calculate joint contextual probability as follows:

$$G(t, \alpha) = \frac{2}{3 \times K \times 3} + \frac{1}{3 \times K \times 2} + \frac{2}{3 \times K \times 3} = \frac{11}{18 \times K}$$
$$G(t, \beta) = \frac{1}{3 \times K \times 3} + \frac{1}{3 \times K \times 2} + \frac{1}{3 \times K \times 3} = \frac{7}{18 \times K}$$

Therefore the (marginal) contextual probability is

$$G(t) = G(t, \alpha) + G(t, \beta) = \frac{1}{K}$$

The conditional contextual probability is as follows:

$$G(\alpha|t) = \frac{G(t, \alpha)}{G(t)} = \frac{11}{18}$$
$$G(\beta|t) = \frac{G(t, \beta)}{G(t)} = \frac{7}{18}$$

As a result we can conclude that t is more likely to be in α class than β class.

Example 3. *Figure 1 shows a dataset D displayed on a 2D (ordinal) grid. Each unit square represents an instance, which is the coordinate of the square. The D is $\{\langle 3,2\rangle, \langle 2,3\rangle, \langle 4,4\rangle, \langle 5,4\rangle, \langle 4,5\rangle\}$ and C is $\{+,-\}$. Our task is to classify $t = \langle 1,1\rangle$.*

Following the CPC algorithm we take the following steps:

- *$E(t,x) = \cup_{c\in C} E^c(t,x)$, and the $E^c(t,x)$ are the following:*

$$\begin{array}{ll}
E^+(t,\langle 3,2\rangle) = \{\langle 3,2\rangle\} & E^-(t,\langle 3,2\rangle) = \emptyset \\
E^+(t,\langle 2,3\rangle) = \{\langle 2,3\rangle\} & E^-(t,\langle 2,3\rangle) = \emptyset \\
E^+(t,\langle 4,4\rangle) = \{\langle 3,2\rangle, \langle 2,3\rangle\} & E^-(t,\langle 4,4\rangle) = \{\langle 4,4\rangle\} \\
E^+(t,\langle 5,4\rangle) = \{\langle 3,2\rangle, \langle 2,3\rangle\} & E^-(t,\langle 5,4\rangle) = \{\langle 4,4\rangle, \langle 5,4\rangle\} \\
E^+(t,\langle 4,5\rangle) = \{\langle 3,2\rangle, \langle 2,3\rangle\} & E^-(t,\langle 4,5\rangle) = \{\langle 4,4\rangle, \langle 4,5\rangle\}
\end{array}$$

- *$G(t,c)$ are calculated as follows: $G(t,+) = 5/13$, and $G(t,-) = 3/13$. So $G(+|t) = 5/8$ and $G(-|t) = 3/8$.*
- *Classify t by $+$ since $G(+|t) > G(-|t)$.*

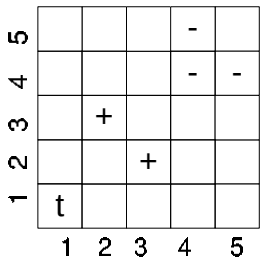

Figure 1: *Given some instances with known classification, classify a new instance t.*

7.4 Evaluation

The CPC algorithm was evaluated using some public datasets from UC Irvine Machine Learning Repository [2]. General information about these datasets is shown in Table 2.

The evaluation results are shown also in Table 2. As a comparison the results by C5.0 and SVM (Support Vector Machine) are also presented. The C5.0 algorithm we used is an implementation in the SPSS-Clementine package. The default parameters were used and the validation method was 5 fold cross validation. SVM is an implementation by Chih-Chung Chang and Chih-Jen Lin [5]. The command line used was "-s 0 -t 0 -c 1 -v 5", which means the type of SVM is C-SVC (C-support vector classification), the kernel function is linear, the cost upper bound is 1 and the cross validation fold is 5. In order to make the results comparable we used the same method of partioning data into train data and test data at each fold: a dataset is read from the first record to the last and partioned into 5 records a group. At fold 1, the first records in all groups are used for testing and the rest are used for training. At fold 2, the second records are used for testing and the rest are used for training, and so on.

Dataset	#Attribute	#Example	#Class	Classification Accuracy (%)		
				C5.0	SVM	CPC
Australian	14	690	2	89.7	85.29	91.88
Auto	25	205	6	70.7	68.29	70.73
Diabetes	8	768	2	73.6	77.65	75.0
German	20	1000	2	71.7		73.8
Glass	9	214	3	79.4	81.90	85.05
Heart	13	270	2	80.4	83.33	84.81
Hepatitis	19	155	2	80.0	80.65	82.58
Horse-Colic	22	368	2	85.3	83.01	83.17
Iris	4	150	3	94.0	94.00	96.0
Sonar	60	208	2	75.9	72.20	87.5
TTT	9	958	2	86.2		97.39
Vote	18	232	2	96.5		96.13
Wine	13	178	3	91.0	94.29	94.94
Average				82.65		86.08

Table 2: *General information about the datasets and the classification accuracy of C5.0, SVM and CPC. The validation method used is 5 fold cross validation. Note that the SVM values for "German", "TTT" and "Vote" datasets are not available because they have categorical attributes and the SVM package we used does not work with categorical attributes.*

From Table 2 we can see that "sonar" has 60 predictor attributes, but only

[2] `http://www.ics.uci.edu/~mlearn/MLRepository.html`

208 data instances. This is clearly a sparse dataset in relative terms. A test accuracy of 87.5% is achieved by CPC, compared with 75.9% by C5.0 and 72.20% by SVM.

8 Summary and conclusion

In this paper we have presented a probability function G – the contextual probability function, and have shown that G is a linear function of the classical probability function P. We can estimate G from data (sparse or dense) directly, and obtain P indirectly through a linear transformation of G. We have shown that G is equivalent to P for classification.

Based on the theoretical framework we have developed a classification procedure CPC, where first of all G is estimated using a subset of neighbourhoods defined here as hypertuples (using all neighbourhoods is infeasible for large datasets), then the posterior probabilities are calculated for all classes given a data instance to be classified, and finally the data instance is classified by the most G-probable class. This procedure has a polynomial time in the number of instances in the data. It is a lazy approach to classification. Experiments show that CPC compares very favourably with C5.0 - a state-of-the-art decision tree classifier.

Our procedure is a generalisation of the voting kNN classifier in the sense that CPC becomes kNN when only one neighbourhood is considered. Neighbourhoods are interpreted as hypertuples in CPC, different from the distanced-based interpretation in the standard kNN. However distance-based neighbourhood can also be used in CPC.

References

[1] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM*, 45(6):891–923, 1998.

[2] R.B. Ash. *Real Analysis and Probability*. Academic Press, New York, 1972.

[3] Christopher G. Atkeson, Andrew W. Moore, and Stefan Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11(1-5):11–73, 1997.

[4] Eric Brill. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565, 1995.

[5] Chih-Chung Chang and Chih-Jen Lin. Libsvm – a library for support vector machines. Technical report, National Taiwan University. `http://www.csie.ntu.edu.tw/~cjlin/libsvm/index.html`.

[6] Belur V. Dasarathy, editor. *Nearest Neighbor(NN) Norms: NN Pattern Classificatiion Techniques*. IEEE Computer Society Press, Los Alamitos, California, 1991.

[7] William Feller. *An Introduction to Probability Theory and Its Applications*. Wiley, 1968.

[8] D. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining*. The MIT Press, 2001.

[9] Edwin T. Jaynes. *Probability Theory: The Logic of Science*. `http://bayes.wustl.edu`.

[10] P. Langley, W. Iba, and K. Thompson. An analysis of bayesian classifiers. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, page 399406, San Jose, CA, 1992. AAAI Press.

[11] J. E. Macleod, A. Luk, and D. M. Titterington. A re-examination of the distance weighted k-nearest neighbor classification rule. *IEEE Trans. Syst. Man Cyber.*, 17(4):689–696, 1987.

[12] T. M. Mitchell. *Machine Learning*. The McGraw-Hill Companies, Inc, 1997.

[13] B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[14] I. Rish. An empirical study of the naive bayes classifier. In *Proceedings of IJCAI-01 workshop on Empirical Methods in AI*, pages 41–46, 2001.

[15] Hui Wang, Ivo Düntsch, and Günther Gediga. Classificatory filtering in decision systems. *International Journal of Approximate Reasoning*, 23:111–136, 2000.

[16] H.R. Warner, A.F. Toronto, L.G. Veasy, and R. Stephenson. A mathematical approach to medical diagnosis: application to congenital heart disease. *Journal of the American Medical Association*, 177:177 – 183, 1961.

A Notation

a_i	:	an attribute
c	:	a class value
C	:	the set of all class values
D	:	a training dataset
E	:	a neighbourhood
E^c	:	a neighbourhood projected to class c
$E(t, x)$	:	a neighbourhood of t defined by t and data instance x
$\mathcal{E}(t)$	:	the set of all neighbourhoods of t
f	:	a true classification function
f'	:	the classification function implied by a dataset
$\bar{f}$	:	an approximation of a true classification function
G	:	contextual probability function
h	:	a hypertuple
K	:	normalising factor
n	:	the number of predictor attributes
N	:	the number of data instances
N_j	:	the number of data instances in class j
p	:	classical probability distribution function
P	:	classical probability function
R	:	the set of all predictor attributes
s	:	a subset of the domain of an attribute
t	:	a designated data instance
v_i	:	a value of attribute i
V	:	data space defined by a set of attributes
x and X	:	passing variables
y	:	the class attribute or a passing variable
α and β	:	two constants in contextual probability
Λ	:	any Kernel function