Setting Attribute Weights for *k*-NN Based Binary Classification via Quadratic Programming

Lu Zhang*, Frans Coenen and Paul Leng

Department of Computer Science, University of Liverpool, Liverpool L69 3BX, UK

E-mails: {lzhang,frans,phl}@csc.liv.ac.uk

Tel: 44-151-7943792

Fax: 44-151-7943715

*The corresponding author is Lu Zhang

Abstract. The *k*-Nearest Neighbour (*k*-NN) method is a typical lazy learning paradigm for solving classification problems. Although this method was originally proposed as a non-parameterised method, attribute weight setting has been commonly adopted to deal with irrelevant attributes. In this paper, we propose a new attribute weight setting method for *k*-NN based classifiers using quadratic programming, which is particularly suitable for binary classification problems. Our method formalises the attribute weight setting problem as a quadratic programming problem and exploits commercial software to calculate attribute weights. To evaluate our method, we carried out a series of experiments on six established data sets. Experiments show that our method is quite practical for various problems and can achieve a stable increase in accuracy over the standard *k*-NN method as well as a competitive performance. Another merit of the method is that it can use small training sets.

Keywords: classification, machine learning, *k*-Nearest Neighbour, attribute weight, quadratic programming

1. INTRODUCTION

Classification is a typical problem in machine learning and knowledge discovery that has been studied by many researchers. Up to now, there have been various approaches reported in the literature. Typical techniques include decision trees [7], neural networks [5], Bayesian estimation [3], linear discriminant functions [17], and lazy learning [2] etc.

The *k*-Nearest Neighbour (*k*-NN) algorithm [13] is a basic algorithm for lazy classification. When classifying an instance, *k*-NN selects *k* most similar instances in the training set of instances, and uses the *k* similar instances to determine the class of the instance under classification via some voting mechanism. Usually, each instance is described as a sequence of attributes, i.e. $\{A_1,A_2,\ldots A_q\}$ (where q is the number of attributes), and the similarity between instance $X=\{X_1,X_2,..X_q\}$ and instance $Y=\{Y_1,Y_2,\ldots Y_q\}$ can be calculated through formula (1), in which, $Simi(X,Y)$ is the similarity between instance X and instance Y, and $Simi(X_i,Y_i)$ is the similarity of the ith attribute between X and Y.

$$Simi(X,Y) = \sum_{i=1}^{q} Simi(X_i, Y_i) \qquad (1)$$

In formula (1), all the attributes for describing instances are equally treated. However, for a real world problem some attributes may be less important than others, and some attributes may even be irrelevant. Therefore, many *k*-NN based classifiers parameterise the similarity function (or the distance function) to deal with irrelevant attributes (see e.g. VDM [36], CCF [11], MVDM [10], MI [12], Relief-F [21], and *k*-NN_{VSM} [38] etc.). Intuitively, more important attributes will be assigned higher weights, and less important attributes will be assigned lower weights. In reality, an attribute weight setting algorithm is needed for a weighted *k*-NN based classifier. In [39], a survey and empirical analysis of such algorithms is provided. For an attribute-

weighted k-NN classifier, similarity between instances can be calculated through formula (2), in which W_i is the weight on the ith attribute.

$$Simi(X,Y) = \sum_{i=1}^{q} W_i * Simi(X_i, Y_i) \quad (2)$$

In this paper, we propose a novel attribute weight setting method using quadratic programming, which is particularly suitable for binary classification problems. Compared with previously proposed methods, our method has the following advantages. First, our method has a sound theoretical foundation, while most other methods are empirical. Secondly, from our experiments, the performance of our method improves previously proposed attribute weight setting methods. Thirdly, our method can use a small-size training set, and still get a good performance. Some preliminary results for this method have been reported in [40].

2. RELATED WORK

2.1 Attribute Weight Setting Problem

The attribute weight setting problem in a weighted k-NN classifier can be described as follows. There are n training instances in the training set, each having a value in each of q attributes and being assigned to a class. These training instances will be used to calculate a set of attribute weights with the aim of making the classifier achieve a high performance when using the weights and the training instances to classify new instances. The set of attribute weights obtained will be used in formula (2) to calculate the similarity between two instances.

The optimal set of attribute weights is a set of weights that, when used to classify new instances, minimises the number of misclassified instances. Obviously, the optimal attribute weight setting is related to both the training instances and the instances under classification. Therefore, the optimal attribute weights cannot be

calculated by only using the training set. However, if we can assume that the training instances can fully represent the instances under classification, it might be possible to get the set of optimal attribute weights that can achieve the smallest prediction error only using the training set itself. This is the attribute weight setting problem we will discuss in this paper.

2.2 Previous Methods

There have been numerous attribute weight setting methods for k-NN proposed in the literature. A thorough classification and survey of these can be found in [39]. We here summarise some frequently referenced methods.

In [30], an attribute weight setting method named EACH is proposed. The idea of EACH is to change the attribute weights of previous training instances after classifying a new training instance. The weights of all the matched attributes for correct classifications are increased by a fixed amount F, and the weights of all the matched attributes for incorrect classifications are decreased by F. On the other hand, the weights of all the mismatched attributes for correct classifications are decreased by F, and the weights of all the mismatched attributes for incorrect classifications are increased by F. Similar to EACH, IB4 [1] uses another more sophisticated formula to calculate the new attribute weights of previous training instances, and Relief-F [21] uses different amounts for different instances. A common property of the above approaches is to continuously adjust the weights when processing a new training instance.

In [20] and [34], two approaches using genetic algorithms to learn attribute weights are reported to have higher accuracies than standard k-NN on some data sets. In [25], the variable kernel similarity metric (VSM) is reported to use conjugate gradient to minimise summed leave-one-out classification error (LOOCE) for the

training instances. In [38], a simplification of VSM with the name k-NN_{VSM} is reported to have similar performance to VSM for a variety of data sets. These approaches try to optimise the weights by processing all the training instances together to get the weights.

There are also some earlier approaches using statistical properties of the training instances to calculate attribute weights. In [36], the *value-difference metric* (VDM) for discrete attributes is introduced. The basic idea of VDM is to assign higher weights to attributes whose distribution across different classes is more skewed. In [10], a modified version of VDM named MVDM is reported to have similar performance to VDM. In [11], two attribute weight setting methods based on VDM are reported, which are the CCF method and the PCF method. The CCF method assigns higher weights to attributes that occur in fewer classes. The PCF method modifies CCF to assign the same attribute different weights in different classes. In [28], it is reported that CCF outperforms PCF on most tested data sets. In [12], a method based on the *mutual information* (MI) theory (see e.g. [33]) is reported. The basic idea of this approach is to calculate the contribution of individual attributes to the class of each training instance. An attribute with a larger contribution will be assigned a higher weight.

Besides the above, other methods can be found in [31] and [23] etc.

3. QUADRATIC PROGRAMING

A quadratic programming (QP) problem is a particular case of an optimisation problem, which is to calculate the maximum or minimum value of an objective function of a set of variables subject to a set of constraints on the variables. For a quadratic programming problem, each of the constraints is a linear equation or a linear

inequality, and the objective function is at most quadratic [15]. Therefore, a QP problem can be represented in the following form:

$$\text{maximise or minimise } \sum_{j=1}^{n} c_j x_j + \sum_{j=1}^{n}\sum_{k=j}^{n} C_{jk} x_j x_k$$

$$\text{subject to Constraint}_i \ (\text{i} = 1,2,...\text{m})$$

$$\text{Constraint}_i \text{ is of one of the three forms :}$$

$$\sum_{j=1}^{n} a_{ij} x_j \geq b_i, \text{ or} \qquad (3)$$

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, \text{ or}$$

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i$$

$$x_j \geq 0 \ (\text{j} = 1,2,...\text{n})$$

In (3), x_1, x_2, … x_n are the *variables*; $\sum_{j=1}^{n} c_j x_j + \sum_{j=1}^{n}\sum_{k=j}^{n} C_{jk} x_j x_k$ is the *objective function*; n is the number of *variables*; and m is the number of *constraints.*

Quadratic programming is a well-studied area in optimisation. Many methods for solving linear programming problems (see [4] for linear programming) can be extended to quadratic programming problems. There have already been commercial software packages (such as IBM OSL [19]) to solve quadratic programming problems. Although there is lack of theoretical analysis of the computational complexity of methods for quadratic programming, it is shown in [27] that current quadratic programming software is able to solve problems with several thousand variables and several thousand constraints.

4. SETTING ATTRIBUTE WEIGHTS VIA QUADRATIC PROGRAMMING

In this paper, our focus is on a particular subset of classification problems – binary classification. In a binary classification problem, each instance will be classified between two classes. In this section, we will demonstrate how the attribute weight

setting problem in *k*-NN based binary classification can be reduced to a quadratic programming problem.

4.1 *Assumption*

When using a *k*-NN based classifier to classify instances, the classifier will classify an instance to the class of a similar instance. However, if there are many irrelevant attributes and/or different attributes have different importance, the calculated similarities may not reflect the real similarities. It is the responsibility of an attribute weight setting method to acquire the set of weights that can make the similarities calculated from formula (2) approximate to the real similarities. The weight of each attribute then reflects the importance of that attribute.

For the binary classification problem, we assume that the real similarity between instances in the same class is 1, and the real similarity between instances in different classes is 0. Thus, the objective of the training process is to seek a set of weights, which, when applied to instances in the training set through formula (2), will lead to similarities that approximate to the real similarities.

4.2 *Formalisation*

Based on the above assumption, the attribute weight setting problem in *k*-NN based binary classification can be viewed as an optimisation problem whose aim is to minimise the differences between the similarities calculated from formula (2) and the real similarities obtained by comparing the classes. However, as there is only one objective function in an optimisation problem, all the differences between corresponding pairs of similarities should be summed into the objective function. As the aim of the optimisation problem is to minimise all the individual differences, we use the sum of the squares of the differences, instead of the arithmetic sum of the

differences. Therefore, the optimisation problem can be summarised as the following quadratic programming problem.

Supposing there are n training instances in the training set, each having q attributes; the constraints in the problem can be represented as equation (4). In (4), S_{ijk} is the similarity on the kth attribute between instance i and instance j. $\sum_{k=1}^{q} S_{ijk} W_k$ is then the similarity between instance i and instance j calculated using formula (2). L_{ij} is the value by which the calculated similarity is less than the real similarity, M_{ij} is the value by which the calculated similarity is greater than the real similarity, and then R_{ij} is the real similarity between instance i and instance j. We will call the L_{ij} and M_{ij} as the *difference variables*.

$$\sum_{k=1}^{q} S_{ijk} W_k + L_{ij} - M_{ij} = R_{ij} \, (i, j = 1..n, i < j) \tag{4}$$

Usually we also require the sum of the weights to be 1. So, there is also another constraint:

$$\sum_{k=1}^{q} W_k = 1 \tag{5}$$

As analysed above, the objective function in this problem is to minimise the sum of the squares of L_{ij} and M_{ij}. In fact, our aim is to minimise each L_{ij} and M_{ij}, but as we have to express our aim in one objective function, we choose the sum of the squares of L_{ij} and M_{ij} as a device to prevent any of them from being too large. Therefore, the objective function can be represented as (6).

$$\text{minimise} \sum_{i=1}^{n} \sum_{j=i+1}^{n} (L_{ij}^2 + M_{ij}^2) \tag{6}$$

4.3 *Complexity analysis*

We provide here a brief analysis of the size of this quadratic programming problem. As there is still no comprehensive analysis of the complexity of quadratic programming problems, this analysis can only indicate how large this quadratic programming problem can be, but not the actual complexity of the problem.

From the above formalisation, there are q weight variables and $n*(n-1)$ difference variables. The number of the constraints in (4) and (5) is $n*(n-1)+1$. Therefore, the above quadratic programming problem is a quadratic programming problem with $n*(n-1)+q$ variables and $n*(n-1)+1$ constraints.

4.4 *Simplification*

As the above complexity analysis shows, the size of the formalised quadratic programming problem is not linear to the number of training instances. Therefore, with the increase of the number of the training instances, the formalised quadratic programming problem may become unmanageable. In such a case, some simplification mechanism may help to reduce the size of the quadratic programming problem, which may also reduce the performance of the classifier in some extent.

In the above formalisation, each instance is compared with all the other instances. If we only compare each instance with a subset of other instances, we can reduce the size of the formalised quadratic programming problem. If an instance is only compared with another p ($p=n-1$) instances, we will have q weight variables and $n*p$ difference variables, and the number of constraints will be $n*p+1$. In this case it becomes a quadratic programming problem with $n*p+q$ variables and $n*p+1$ constraints.

4.5 *An Illustrative Example*

To understand the above formalisation, let us consider the following example. There are four training instances in the example, which are listed in Table 1. Each instance has a value in each of the three attributes, and each instance is classified as A or B.

Table 1. Training instances of the example

Instance Number	Attribute 1	Attribute 2	Attribute 3	Class
1	1	1	0	A
2	1	0	1	A
3	0	0	0	B
4	0	1	1	B

Using the above method, we can get the following QP problem. The constraints of the formalised QP problem are listed in (7).

$$\begin{aligned}
&W_1 + L_{12}\text{-}?_{12} = 1\\
&W_3 + L_{13}\text{-}?_{13} = 0\\
&W_2 + L_{14}\text{-}?_{14} = 0\\
&W_2 + L_{23}\text{-}?_{23} = 0\\
&W_3 + L_{24}\text{-}?_{24} = 0\\
&W_1 + L_{34}\text{-}?_{34} = 1\\
&W_1 + W_2 + W_3 = 1
\end{aligned} \qquad (7)$$

The objective function is to minimise the sum in (8).

$$L_{12}^2 + M_{12}^2 + L_{13}^2 + M_{13}^2 + L_{14}^2 + M_{14}^2 + L_{23}^2 + M_{23}^2 + L_{24}^2 + M_{24}^2 + L_{34}^2 + M_{34}^2 \qquad (8)$$

Obviously, if we set W_1 to 1, and set W_2, W_3, and all the L_{ij} and M_{ij} to 0, all the constraints in (7) can be satisfied, and the sum in (8) can reach 0 - the minimum. This weight setting represents the classification using only the first attribute, and it can correctly classify all the four training instances.

5. EXPERIMENTAL RESULTS

5.1 Experimental Method

To test the performance of our method, we applied it to six binary classification data sets, acquired from the UCI Machine Learning Repository [6]. We randomly selected a fixed number of instances from each data set as training instances. Some initial experiments indicated that about 100 training instances was sufficient for our method to achieve a good and stable performance. We therefore selected 100 as the size of all the training sets for the six data sets. The advantage of this is that, as 100 training instances are quite manageable for our training process, it is not necessary to use any simplification of the type discussed in section 4.4. We view the latter only as a possible strategy for managing large training sets with some sacrifice of accuracy. Our results suggest, however, that our method produces good results even when this small training set is used, an advantage which we discuss later.

For the Thyroid Disease data set, we randomly chose 50 positive instances and 50 negative instances in the data set as the training data, as there were very few positive instances in the data set. For each other data set, we randomly chose 100 instances in the data set as the training data. Based on the 100 instances, we calculated the attribute weight setting using our method. Then, the calculated weights were used to classify the remaining instances in the data set. The accuracy of classifying the remaining instances was recorded, and for comparison, the accuracy of standard k-NN classification without using any attribute weights was also recorded. To calculate the attribute weight setting, we generated the corresponding quadratic programming problem in MPS format [29], and used IBM OSL [19] to solve the problem and get the weight setting. To avoid occasional results, we performed the experiment ten times for each data set. All the experiments were performed on a Pentium III 500MHz

PC with 128M RAM running Windows NT 4.0. The CPU time of each weight setting calculation was also recorded as the training time to indicate the manageability of each formalised quadratic programming problem.

In a k-NN based classifier, the value of k will also affect the accuracy of classification. As usually the value of k is a small odd number, we tested our method when k is odd and $1=k=19$. For each k, the average accuracy and the standard deviation for both our method and the standard k-NN method were recorded. To test the significance of the difference in accuracy between our method and the standard k-NN, we used Yates' $?^2$ (chi-square) test, whose meaning is interpreted in [22] as follows: When the $?^2$ value is less than 3.84, the difference is not significant; when the $?^2$ value is no less than 3.84 but less than 6.63, the difference is probably significant; and when the $?^2$ value is no less than 6.63, the difference is significant.

5.2 Descriptions of the Tested Data Sets

The six data sets used were the Mushroom data set, the Congressional Voting data set, the Thyroid Disease data set, the Breast Cancer data set, the Pima Indians Diabetes data set, and the Credit Screening data set. The sets were chosen to provide a range of cases of binary classification. They included three cases of noise-free, and three of noisy data, and cases for which the attributes are all discrete, all continuous, and a mixture. We also chose data which has been the subject of published experiments for other methods, to enable us to make comparisons with these.

In the Mushroom data set, an instance represents one type of mushroom. Each instance is characterised by 22 discrete attributes, and each instance is classified as either edible or poisonous. There are 8124 instances in the data set in total.

In the Congressional Voting data set, an instance represents one vote. Each instance is characterised by 16 discrete attributes, and each instance is classified as either for democrats or for republicans. There are 435 instances in the data set in total.

In the Thyroid Disease data set, an instance represents one case of the thyroid disease. Each instance is characterised by 7 continuous attributes and 18 discrete attributes, and each instance is classified as either hypothyroid or negative. There are 3163 instances in the data set in total.

In the Breast Cancer data set, an instance represents one case of the breast cancer disease. Each instance is characterised by 4 continuous attributes and 5 discrete attributes, and each instance is classified as either recurrent or not recurrent. There are 286 instances in the data set in total.

In the Pima Indians Diabetes data set, an instance represents one report of the diabetes test. Each instance is characterised by 8 continuous attributes, and each instance is classified as either positive or negative. There are 768 instances in the data set in total.

In the Credit Screening data set, an instance represents one credit card application. Each instance is characterised by 6 continuous attributes and 9 discrete attributes, and each instance is classified as either plus (+) or minus (-). There are 690 instances in the data set in total.

The features of the six data sets are summarised in table 2.

Table 2. Features of the six tested data sets

Data Set	Number of Instances	Number Attributes
Mushroom	8124	22
Congressional Voting	435	16
Thyroid Disease	3163	25

Breast Cancer	286	9
Pima Indians Diabetes	768	8
Credit Screening	690	15

5.3 Results on Less Noisy Data Sets

For the Mushroom data set, the Congressional Voting data set, and the Thyroid Disease data set, there is not much noise in the data sets. Therefore, there have been highly accurate methods reported in the literature, and the standard k-NN classifier can also solve the problems with high accuracy. Previous results for the Mushroom data set can be found in [32], [18] and [14]. Previous results for the Congressional Voting data set can be found in [32], [39] and [14]. Previous results for the Thyroid Disease data set can be found in [14].

Our results for the Mushroom data set are summarised in Table 3. The average training time for the quadratic programming approach was 1940 seconds with a standard deviation of 343 seconds. The results tabulated show the average accuracy of classification by our method in comparison with that of standard *k*-NN.

Table 3. Results on the Mushroom data set

k	**Quadratic Programming (%)**	**Standard *k*-NN (%)**	**Difference**	**Significant (Yates' $?^2$)**
1	98.67±0.90	98.13±0.87	+0.54	Yes (73.97)
3	97.64±1.61	96.59±1.56	+1.05	Yes (157.50)
5	96.13±1.90	95.26±1.46	+0.87	Yes (73.50)
7	95.16±1.75	93.16±2.09	+2.00	Yes (291.47)
9	95.41±1.32	91.90±1.64	+3.51	Yes (831.20)

11	94.82±0.67	90.58±1.31	+4.24	Yes (1065.21)
13	95.02±0.79	89.84±0.83	+5.18	Yes (1537.81)
15	94.99±0.73	89.47±0.47	+5.52	Yes (1705.09)
17	94.96±0.67	89.21±0.26	+5.75	Yes (1819.15)
19	95.08±0.63	89.21±0.23	+5.87	Yes (1909.13)

Table 3 shows that our method improves on the standard *k*-NN for this data set. Our method demonstrates significant improvements for all the values of *k*, although some of the increases are small.

The results for the Congressional Voting data set are summarised in Table 4. The average training time for the quadratic programming approach was 1159 seconds with a standard deviation of 206 seconds.

Table 4. Results on the Congressional Voting data set

k	**Quadratic Programming (%)**	**Standard k-NN (%)**	**Difference**	**Significant (Yates' ?^{2})**
1	94.63±1.03	91.11±1.74	+3.52	Yes (30.81)
3	94.99±1.40	91.67±1.59	+3.32	Yes (29.13)
5	95.52±0.63	90.98±1.67	+4.54	Yes (54.13)
7	95.58±0.58	90.78±1.62	+4.80	Yes (59.98)
9	95.67±0.64	90.63±1.71	+5.04	Yes (65.89)
11	95.64±0.66	90.39±1.68	+5.25	Yes (70.25)
13	95.67±0.57	90.09±1.81	+5.58	Yes (78.02)
15	95.70±0.63	89.55±1.68	+6.15	Yes (91.84)
17	95.64±0.59	89.61±1.77	+6.03	Yes (88.28)

19	95.67±0.64	89.31±1.57	+6.36	Yes (96.63)

Again, Table 4, shows our method to improve on the standard *k*-NN for this data set. For all the values of *k*, our method achieves significant increases in accuracy. In contrast with the Mushroom data set, for which the best results were obtained with *k* = 1, this data illustrates a case where improved performance is obtained from a larger value of *k*. The figures suggest our method exploits this more effectively than *k*-NN, which shows no improvement for $k > 3$.

The results for the Thyroid Disease data set are summarised in Table 5. The average training time for the quadratic programming approach was 2008 seconds with a standard deviation of 363 seconds.

Table 5. Results on the Thyroid Disease data set

k	**Quadratic Programming (%)**	**Standard k-NN (%)**	**Difference**	**Significant (Yates' $?^2$)**
1	86.41±3.55	83.83±3.99	+2.58	Yes (80.28)
3	89.14±2.02	85.54±3.48	+3.60	Yes (179.18)
5	91.33±2.18	87.30±4.28	+4.03	Yes (260.21)
7	91.86±2.05	88.36±5.10	+3.50	Yes (210.12)
9	90.56±2.17	85.93±8.59	+4.63	Yes (316.05)
11	90.38±2.47	83.72±9.82	+6.66	Yes (602.01)
13	90.02±2.72	82.74±11.81	+7.28	Yes (689.29)
15	89.94±3.20	81.32±14.45	+8.62	Yes (924.10)
17	89.39±3.31	78.69±14.40	+10.70	Yes (1306.47)
19	89.01±3.85	74.61±15.70	+14.40	Yes (2133.07)

This case also shows that our method improves on the standard k-NN for this data set. For all the values of k, our method achieves significant increases, many of which are quite large. As with the previous cases, the accuracy of the method seems to become relatively stable once a sufficient threshold value of k is reached. This contrasts with standard k-NN, for which accuracy declines significantly for values of k that are too large.

5.4 *Results on Noisy Data Sets*

There is much noise in the Breast Cancer data set, the Pima Indians Diabetes data set and the Credit Screening data set. Therefore, there has been no very accurate method reported in the literature and the standard k-NN classifier can only solve the problems with low accuracy. Previous results for the Breast Cancer data set can be found in [26], [8], [9], [37] and [14]. Previous results for the Pima Indians Diabetes data set can be found in [35] and [14]. Previous results for the Credit Screening data set can be found in [14].

The results for the Breast Cancer data set are summarised in Table 6. The average training time for the quadratic programming approach was 523 seconds with a standard deviation of 91 seconds.

Table 6. Results on the Breast Cancer data set

k	**Quadratic Programming (%)**	**Standard k-NN (%)**	**Difference**	**Significant (Yates' ?^{2})**
1	66.93±3.86	67.37±3.56	-0.44	No (0.0629)
3	69.30±2.51	69.52±2.78	-0.22	No (0.0121)
5	71.29±3.78	72.20±2.07	-0.91	No (0.336)

7	71.45±3.02	72.74±1.78	-1.29	No (0.706)
9	72.74±2.33	71.88±2.42	+0.86	No (0.302)
11	72.37±2.15	72.47±2.63	-0.10	No (0.000995)
13	73.28±2.20	72.31±2.75	+0.97	No (0.394)
15	73.50±2.52	72.42±2.62	+1.08	No (0.496)
17	73.55±2.55	71.83±2.14	+1.72	No (1.30)
19	73.01±2.41	71.72±2.27	+1.29	No (0.711)

From Table 6, it appears that the two methods are about the same for this data set. Each method has five increases and five decreases, and none of the differences are significant or probably significant. However, weight-setting again appears to enable better exploitation of a larger value of k. For all $k > 11$, the method gives better results than *any* of the cases of k-NN.

The results for the Pima Indians Diabetes data set are summarised in Table 7. The average training time for the quadratic programming approach was 394 seconds with a standard deviation of 27 seconds.

Table 7. Results on the Pima Indians Diabetes data set

k	**Quadratic Programming (%)**	**Standard k-NN (%)**	**Difference**	**Significant (Yates' $?^2$)**
1	67.46±1.88	68.24±2.13	-0.78	No (0.896)
3	70.94±2.09	70.31±1.51	+0.63	No (0.609)
5	72.01±0.88	71.12±1.60	+0.89	No (1.26)
7	72.63±1.39	71.45±1.45	+1.18	No (2.25)
9	72.52±1.28	71.93±1.76	+0.59	No (0.551)

11	73.47±1.11	71.54±2.14	+1.93	Probably (6.14)
13	73.61±1.30	71.57±1.56	+2.04	Yes (6.88)
15	73.40±1.61	71.30±1.88	+2.10	Yes (7.26)
17	72.98±1.79	71.36±2.37	+1.62	Probably (4.28)
19	72.86±1.83	70.76±1.93	+2.10	Yes (7.17)

In this case, our method is slightly better than the standard k-NN for this data set. For all values of $k > 1$, our method demonstrates increases, many of which are significant or probably significant.

The results for the Credit Screening data set are summarised in Table 8. The average training time for the quadratic programming approach was 1022 seconds with a standard deviation of 400 seconds.

Table 8. Results on the Credit Screening data set

k	**Quadratic Programming**	**Standard k-NN**	**Difference**	**Significant (Yates' $?^2$)**
1	79.29±3.08	77.19±3.80	+2.10	Yes (7.52)
3	83.22±1.57	81.81±2.33	+1.41	Probably (3.96)
5	83.97±2.40	83.44±1.66	+0.53	No (0.569)
7	84.54±1.33	83.80±1.08	+0.74	No (1.16)
9	84.92±0.96	83.76±1.58	+1.16	No (2.92)
11	85.17±0.71	83.97±1.61	+1.20	No (3.16)
13	85.25±0.52	83.80±1.68	+1.45	Probably (4.63)

15	85.12±0.70	83.80±1.99	+1.32	No (3.82)
17	85.10±1.04	83.95±2.24	+1.15	No (2.90)
19	85.02±0.90	83.47±2.29	+1.55	Probably (5.22)

Again, our method appears slightly better than the standard k-NN for this data set, with improved results for all the values of k.

5.5 Analysis of the Experimental Results

Based on the above experimental results, we can find the following properties of our method.

5.5.1 Stable increase in accuracy

For an attribute setting method for k-NN, the basic evaluation is the increase in accuracy over the standard k-NN. In our experiments, we tested ten values of k for each data set. For the three less noisy data sets, our method achieves increases in accuracy for all the different values of k unanimously. For the three noisy data sets, our method achieves increases in accuracy for 24 values of k, and some decrease for the other 6 values of k. However, if k is tuned into the optimal values for our method, there is always an increase for each data set, and the accuracy achieved in this case is always greater than for *any* case of standard k-NN tested. For the three less noisy data sets, all the increases are significant. Moreover, whereas the accuracy of standard k-NN sometimes reduces significantly if too large a value of k is chosen, the weighted classification appears to be relatively stable once a sufficiently high value of k is reached. For the three noisy data sets, nine increases are significant or probably significant, but no decreases are significant or probably significant. In general, although our method may not guarantee a better performance over the standard k-NN,

our method is not likely to be much worse. The increases and decreases of our method are summarised in Table 9.

Table 9. Summary of increases

Data Set	Number of increases (significant)	Number of decreases (significant)	Maximum increase	Minimum increase	Increase for optimal *k*
Mushroom	10 (10)	0 (0)	5.87 (k=19)	0.54 (k=1)	0.54 (k=1)
Congressional Voting	10 (10)	0 (0)	6.36 (k=19)	3.32 (k=3)	6.15 (k=15)
Thyroid Disease	10 (10)	0 (0)	14.40 (k=19)	2.58 (k=1)	3.50 (k=7)
Breast Cancer	5 (0)	5 (0)	1.72 (k=17)	-1.29 (k=7)	1.72 (k=17)
Pima Indians Diabetes	9 (3)	1 (0)	2.10 (k=15, 19)	-0.78 (k=1)	2.04 (k=13)
Credit Screening	10 (1)	0 (0)	2.10 (k=1)	0.53 (k=5)	1.45 (k=13)

In [39], six classification methods were evaluated using six data sets with no irrelevant attributes. All of the previous weighted approaches analysed in that paper demonstrated decreases in accuracy for some data sets. In contrast, our method seems more stable in achieving increases in accuracy for different data sets. It should be noted that, in the absence of irrelevant attributes, it is natural that our method does not achieve dramatic increases in accuracy.

A summary of the results reported [39] is shown in Table 10, in which the increase is achieved when k is tuned to the values that let the corresponding weighting method achieve the highest accuracy.

Table 10. Previous results on data sets without irrelevant attributes

Data Set	Relief-F	k-NN$_{VSM}$	CCF	VDM	MVDM	MI
LED-7	-1.0	0.0	-1.5	-1.4	-1.3	-1.2
Waveform-21	0.3	-0.5	-6.1	-3.7	-3.9	0.5
Cleveland	-0.5	0.0	-1.3	0.2	0.7	-0.6
Congressional Voting	2.9	2.5	1.0	2.1	2.1	2.0
Isolet	0.4	1.9	-1.1	-3.9	1.6	1.6
NETtalk	9.2	6.6	7.7	10.0	12.1	9.7

The average increases in accuracy in percentage points for the six data sets are respectively 1.88 (Relief-F [21]), 1.75 (k-NN$_{VSM}$ [38]), -0.22 (CCF [11]), 0.55 (VDM [36]), 1.88 (MVDM [10]), and 2.00 (MI [12]). Our method achieves an average increase of 2.57 percentage points on the tested six data sets when k is tuned to the optimal values. Please note that this is not a justified comparison of accuracy increases as many data sets used in Table 10 are not binary classification data sets and are not used in our experiments. This review does suggest, however, that weighted k-NN methods typically show only small increases in accuracy compared with the standard non-weighted k-NN when the tested data sets have no irrelevant attributes.

5.5.2 Bearable training time

The idea of using optimisation for machine learning has already been proposed in the literature (see e.g. [24] and [16]). The main drawback of these approaches is that they usually need much computation, which may mean a long training time. However,

with the increase of the capacity of computation, it seems that we can already often overcome this drawback. In our experiments, the average training times for the six data sets were respectively 32.33±5.72 minutes (Mushroom), 19.32±3.43 minutes (Congressional Voting), 33.47±6.05 minutes (Thyroid Disease), 8.72±1,52 minutes (Breast Cancer), 6.57±0.45 minutes (Pima Indians Diabetes), and 17.03±6.67 minutes (Credit Screening). Any of the above training times is bearable, and it can be predicted that the training time of our method for larger training sets and/or data sets with more attributes should also be bearable.

5.5.3 Competitive performance

As previously mentioned, the data sets chosen for our experiments have been the subject of experiments published for many other methods, with which we are able to make comparisons. The results of these are summarised in table 11. This shows that, compared with previous results, if k is tuned to the optimal values for our method, the overall accuracies for our method are competitive.

In [14], the RISE method was evaluated using all the data sets we have used. In 4 of the 6 cases, we have been able to show improved accuracy. The STAGGER method [32] was applied to both the Mushroom data set and to the Congressional Voting data set, and the Mushroom data set was used in the method described in [18] also. In [39], six weighted k-NN approaches were tested on the Congressional Voting data set and the accuracies achieved were respectively 95.5% (Relief-F [21]), 95.1% (k-NN_{VSM} [38]), 93.6% (CCF [11]), 94.7% (VDM [36]), 94.7% (MVDM [10]) and 94.6% (MI [12]). In all these cases a greater accuracy was achieved by our method. The Breast Cancer data set has been used to evaluate methods described in [26], [8], [9] and [37], as well as [14]. Only one of these, [8] demonstrates higher accuracy than that of our method. This is also the case for the Pima Indians Diabetes data set in [35],

although on this data our method improves on RISE. Overall, our method performs better than most or all other methods on every data set except the Thyroid Disease data, for which the only comparison, RISE, is superior.

Table 11. Comparison of average performance

Method	Ours	[14]	[32]	[18]	[39]	[26]	[8]	[9]	[37]	[35]
Mushroom	98.67%	100%	95%	95%	-	-	-	-	-	-
Congressional Voting	95.70%	95.2%	90%-95%	-	93.6%-95.5%	-	-	-	-	-
Thyroid Disease	91.86%	97.5%	-	-	-	-	-	-	-	-
Breast Cancer	73.55%	67.7%	-	-	-	66%-72%	78%	65%-72%	68%-73.5%	-
Pima Indians Diabetes	73.61%	70.4%	-	-	-	-	-	-	-	76%
Credit Screening	85.25%	83.3%	-	-	-	-	-	-	-	-

5.5.4 Small training sets

Intuitively, our method can take all the comparisons between any two training instances into consideration in the training process, and therefore, our method may acquire enough knowledge from fewer training instances to get a good performance. The experimental results also support this.

In our experiments, our method uses 100 training instances. Conversely, the experiments reported for RISE [14], STAGGER [32], and for the method used in [35] and the various methods reported in [39] all use larger training sets, in some cases much larger. Despite this, our method achieves higher accuracy in almost all cases. The only exceptions are for the Mushroom data set and Thyroid Disease data set, for which RISE obtains greater accuracy using 5416 training cases and 2108 training cases respectively, and for the method described in [35], which obtains greater accuracy on the Pima Indians Diabetes set using 576 training cases. Overall, it appears that other methods only perform better than ours, if at all, in cases where very much larger training sets are used. These results are summarised in Table 12.

Table 12. Comparison of training set sizes

Data Set	Previous Methods		Our Method	
	Training Set Size	Average Accuracy	Training Set Size	Average Accuracy
Mushroom	1000 ([32])	95%	100	98.67%
	5416 ([14])	100%		
Congressional Voting	305 ([39])	93.6%-95.5%	100	95.70%
	290 ([14])	95.2%		
Thyroid Disease	2108 ([14])	97.5%	100	91.86%
Breast Cancer	190 ([14])	67.7%	100	73.55%
Pima Indians Diabetes	576 ([35])	76%	100	73.61%
	512 ([14])	70.4%		
Credit Screening	460 ([14])	83.3%	100	85.25%

6. FUTURE WORK

An interesting and useful study might be a study of the learning curve of our method. This can be done by repeating the experiments in this paper for various different sizes of the training sets. It may also be worthwhile to investigate to what extent the simplification strategy suggested in section 4.4 will work. We would expect a slightly better performance for our method when the size of the training set is tuned to the optimal value with some light simplification.

A limitation of our method may be the assumption that the real similarity between instances in the same class is 1 and the real similarity between instances in different classes is 0. Although this seems natural for many problems, it is not always the case.

For example, when classifying instances as normal or abnormal, those normal instances should be similar in nature, but those abnormal instances may not be similar at all. Furthermore, instances in different classes may not be totally dissimilar to each other. This is more the case when the problem is not a binary classification problem. It is always in nature that some classes are somewhat similar to each other while other classes are dissimilar to each other.

For this reason, we think this method can achieve good performance only on binary classification problems, and will require some adaptation before it can be applied to other classification problems.

In the future, we plan to identify some mechanisms to acquire the real similarities between classes rather than using the assumed similarities. One possible approach may be to use statistical information for setting the similarities. Another possible approach may be to recursively apply our method and gradually adjust the real similarities accordingly. We think if we can acquire the genuine real similarities, it will be possible to extend our method to general classification problems and/or further improve the performance of our method.

7. CONCLUSION

There have been quite a few attribute weight setting algorithms for k-NN reported in the literature. In this paper, we proposed a new attribute weight setting method for k-NN based binary classification using quadratic programming. We also performed a series of experiments on six previously known binary classification data sets. For the three less noisy data sets, our method achieved significant increases for all the tested values of k over the standard k-NN. For the three noisy data sets, our method also achieved significant or probably significant increases and no significant or probably significant decreases over the standard k-NN in most cases.

In conclusion, our experimental results suggest that our method may have the following properties: a stable increase in accuracy over standard *k*-NN, bearable training time, a good performance compared with other methods, and the ability to achieve this performance using small training sets.

ACKNOWLEDGEMENTS

The work in this paper was supported by the UK DTI under the Foresight 'LINK' programme (FLA009). Our thanks are due to Colin Johnson and John Huckssteppe of Stoves PLC, Mike Delves of NA Software Ltd., and Stan Price for their assistance with the project.

REFERENCES

[1] D. W. Aha, "Tolerating Noise, Irrelevant, and Novel Attributes in Instance-Based Learning Algorithms," International Journal of Man-Machine Studies, vol. 36, pp. 267-287, 1992.

[2] D. W. Aha, Lazy Learning, Kluwer, Boston, MA, 1997.

[3] J. M. Bernardo and A. F. Smith, Bayesian Theory, Wiley, New York, 1996.

[4] D. Bertsimas and J. N. Tsitsiklis, Introduction to Linear Optimization, Athena Scientific, Belmont, Massachusetts, 1997.

[5] C. M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, Oxford, UK, 1995.

[6] C. L. Blake and C. J Merz, UCI Repository of Machine Learning Databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science, 1998.

[7] C. E. Brodley and P. E. Utgoff, "Multivariate Decision Trees," Machine Learning 19 (1): 45-77, 1995.

[8] G. Cestnik, I. Konenenko, and I. Bratko, "Assistant-86: A Knowledge-Elicitation Tool for Sophisticated Users," In Proceedings of the 2nd European Working Session on Learning. I. Bratko & N. Lavrac (Eds.) Progress in Machine Learning, 31-45, Sigma Press, 1987.

[9] P. Clark, and T. Niblett, "Induction in Noisy Domains," In Proceedings of the 2nd European Working Session on Learning. I. Bratko & N. Lavrac (Eds.) Progress in Machine Learning, 11-30, Bled, Yugoslavia: Sigma Press, 1987.

[10] S. Cost, and S. Salzberg, "A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features," Machine Learning, 10, 57-78, 1993.

[11] R. H. Creecy, B. M. Masand, S. J. Smith, and D. L. Waltz, "Trading Mips and Memory for Knowledge Engineering," Communications of the ACM, vol. 35, pp. 48-64, 1992.

[12] W. Daelemans, S. Gills, and G. Durieux, Learnability and Markedness in Data-Driven Acquisition of Stress (Technical Report 43). Tilburg, Netherlands: Tilburg University, Institute for Language Technology and Artificial Intelligence, 1993.

[13] B. V. Dasarathy, Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques. Los Alamitos, CA: IEEE Computer Society Press, 1991.

[14] P. Domingos, "Unifying Instance-Based and Rule-Based Induction," Machine Learning, 24(2), pages 141-168, 1996.

[15] R. Fourer, Nonlinear Programming Frequently Asked Questions, [http://www-unix.mcs.anl.gov/otc/Guide/faq/nonlinear-programming-faq.html], 2001.

[16] A. J. Grove, N. Littlestone, and D. Schuurmans, "General Convergence Results for Linear Discriminant Updates," In Proceedings of the COLT 97, pp. 171-183, ACM Press, 1997.

[17] I. Guyon and D. G. Stork, "Linear Discriminant and Support Vector Classifiers," In Alex Smola, Peter Bartlett, Bernhard Scholkopf, and Dale Schuurmans (eds.) Advances in Large Margin Classifiers. MIT Press, Cambridge, MA, 1999.

[18] W. Iba, J. Wogulis, and P. Langley, "Trading off Simplicity and Coverage in Incremental Concept Learning," In Proceedings of the 5th International Conference on Machine Learning, 73-79. Ann Arbor, Michigan: Morgan Kaufmann, 1988.

[19] IBM, Optimization Solutions and Library (Version 3), [http://www-3.ibm.com/software/data/bi/osl/index.html], 2001.

[20] J. D. Kelly, and L. Davis, "A hybrid genetic algorithm for classification," In Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (pp. 645-650). Sydney, Australia: Morgan Kaufmann, 1991.

[21] I. Kononenko, "Estimating Attributes: Analysis and Extensions of RELIE-F," In Proceedings of the 1994 European Conference on Machine Learning, pp. 171-182, Catania, Italy: Springer Verlag, 1994.

[22] R. Langley, Practical Statistics Simply Explained, pp. 285-291, Dover Publications, New York, 1971.

[23] C. X. Ling and H. Wang, "Computing Optimal Attribute Weight Settings for Nearest Neighbor Algorithms," Artificial Intelligence Review, vol. 11, pp. 255-272, 1997.

[24] N. Littlestone, "Learning Quickly When Irrelevant Attributes Abound: A New Linear-Threshold Algorithm," Machine Learning, vol. 2, no. 4, pp. 285-318, 1988.

[25] D. Lowe, "Similarity metric learning for a variable-kernel classifier," Neural Computation, 7, 72-85, 1995.

[26] R. S. Michalski, I. Mozetic, J. Hong, and N. Lavrac, "The Multi-Purpose Incremental Learning System AQ15 and its Testing Application to Three Medical Domains," In Proceedings of the Fifth National Conference on Artificial Intelligence, pp. 1041-1045, Philadelphia, PA: Morgan Kaufmann, 1986.

[27] H. D. Mittelmann, "Benchmarking Interior Point LP/QP Solvers," Optimization Methods and Software 12, 655-670, 1999.

[28] M. Mohri and H. Tanaka, "An Optimal Weighting Criterion of Case Indexing for Both Numeric and Symbolic Attributes," In D. Aha (ed.) Case-Based Reasoning: Papers from the 1994 Workshop, Menlo Park, CA: AAAI Press, 1994.

[29] B. Murtagh, Advanced Linear Programming: Computation and Practice. McGraw-Hill, 1981.

[30] S. L. Salzberg, "A Nearest Hyperrectangle Learning Method," Machine Learning, vol. 6, pp. 251-276, 1991.

[31] K. Satoh and S. Okamoto, "Toward PAC-Learning Of Weights from Qualitative Distance Information," In D. Aha (ed.) Case-Based Reasoning: Papers from the 1994 Workshop, Menlo Park, CA: AAAI Press, 1994.

[32] J. S. Schlimmer, Concept Acquisition Through Representational Adjustment (Technical Report 87-19). Doctoral dissertation, Department of Information and Computer Science, University of California, Irvine. 1987.

[33] C. E. Shannon, "A mathematical theory of communication," Bell Systems Technology Journal, 27, 379-423, 1948.

[34] D. Skalak, "Prototype and feature selection by sampling and random mutation hill climbing algorithms," In Proceedings of the Eleventh International Machine Learning Conference (pp. 293-301). New Brunswick, NJ: Morgan Kaufmann, 1994.

[35] J.W. Smith, J.E. Everhart, W.C. Dickson, W.C. Knowler, and R.S. Johannes, "Using the ADAP Learning Algorithm to Forecast the Onset of Diabetes Mellitus," In Proceedings of the Symposium on Computer Applications and Medical Care, pp. 261--265. IEEE Computer Society Press, 1988.

[36] C. Stanfill and D. Waltz, "Toward Memory-Based Reasoning," Communications of the ACM, vol. 29, pp. 1213-1228, 1986.

[37] M. Tan, and L. Eshelman, "Using Weighted Networks to Represent Classification Knowledge in Noisy Domains," In Proceedings of the Fifth International Conference on Machine Learning, pp. 121-134, Ann Arbor, MI, 1988.

[38] D. Wettschereck, A Description of the Mutual Information Approach and the Variable Similarity Metric (Technical Report 944). Sankt Augustin, Germany, German National Research Center for Computer Science, Artificial Intelligence Research Division. 1995.

[39] D. Wettschereck, D. W. Aha, and T. Mohri, "A Review and Empirical Evaluation of Feature Weighting Methods for a Class of Lazy Learning Algorithms," Artificial Intelligence Review, vol. 11, pp. 273-314, 1997.

[40] L. Zhang, F. Coenen, and P. Leng, "An Attribute Weight Setting Method for *k*-NN Based Binary Classification using Quadratic Programming," In Proceedings of 15th European Conference on Artificial Intelligence (ECAI), 21-26 July 2002, pp. 325-329.