Improving k-Nearest Neighbour Classification with Distance Functions Based on Receiver Operating Characteristics

Md. Rafiul Hassan[1], M. Maruf Hossain[1], James Bailey[1,2], and Kotagiri Ramamohanarao[1,2]

[1] Department of Computer Science and Software Engineering,
The University of Melbourne, Australia
[2] NICTA Victoria Laboratory, The University of Melbourne, Australia
{mrhassan,hossain,jbailey,rao}@csse.unimelb.edu.au

Abstract. The k-nearest neighbour (k-NN) technique, due to its interpretable nature, is a simple and very intuitively appealing method to address classification problems. However, choosing an appropriate distance function for k-NN can be challenging and an inferior choice can make the classifier highly vulnerable to noise in the data. In this paper, we propose a new method for determining a good distance function for k-NN. Our method is based on consideration of the area under the Receiver Operating Characteristics (ROC) curve, which is a well known method to measure the quality of binary classifiers. It computes weights for the distance function, based on ROC properties within an appropriate neighbourhood for the instances whose distance is being computed. We experimentally compare the effect of our scheme with a number of other well-known k-NN distance metrics, as well as with a range of different classifiers. Experiments show that our method can substantially boost the classification performance of the k-NN algorithm. Furthermore, in a number of cases our technique is even able to deliver better accuracy than state-of-the-art non k-NN classifiers, such as support vector machines.

Keywords: Receiver Operating Characteristics (ROC), k-Nearest Neighbour, Feature Weighting, Classification, Gene Expression.

1 Introduction

The k-nearest neighbour (k-NN) technique is a classic, simple and appealing method to address classification problems. It has a very intuitive interpretation and its predictions are easily explained to domain experts. Although k-NN has been applied for classification in many domains, it tends to suffer from poor classification accuracy when i) there are very many features, ii) when there are very few instances, or iii) the data is very noisy. These weaknesses make it difficult to use k-NN for datasets such as gene expression data, which are very noisy and typically have thousands of features, but only tens or at most hundreds of instances. Indeed in the gene expression domain, k-NN has been adopted by very few researchers (e.g. [1, 2]), due to its generally inferior performance.

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 489–504, 2008.

Example 1. *In the ALL-AML leukaemia gene expression dataset, k-NN has an average accuracy of only* 83.33±3.49% *using* 10×10-*fold cross validation, whereas Support Vector Machines (SVM) yield an average accuracy of* 98.61±1.26%. *This lower accuracy of the k-NN classifier can discourage biologists to use it.*

A wide range of proposals have been made to improve k-NN. These principally propose alternative ways of computing the distance function, since using different distance functions can yield vastly different classification performance. Indeed ideally, the distance function used for k-NN should be adapted to the particular problem being solved [3].

Our aim in the paper is to improve the classification performance of k-NN using a new type of distance function. It is based on a feature weighting scheme that considers receiver operating characteristics that are appropriate to the points whose distance is being computed.

In a nutshell, we present a method to derive a distance function for k-NN based on feature weighting. The weight for each feature is calculated by considering the area under the Receiver Operating Characteristics (ROC) curve [4]. This value is equivalent to the Mann-Whitney U statistic normalized by the number of possible pairings of positive and negative values, also known as the two sample Wilcoxon rank-sum statistic [5]. The area under the ROC curve (AUC) actually represents the probability that a randomly chosen positive example is correctly ranked with greater suspicion than a randomly chosen negative example. Moreover, this probability of correct ranking is the same quantity estimated by the non-parametric Wilcoxon statistic [6].

The intuitive outline of the technique is as follows:

For a given dataset $\mathcal{D}$ of n instances comprising m features: $x_1, x_2, x_3, \ldots, x_m$, each feature x_i (where $1 \leqslant i \leqslant m$) has some discriminative power, i.e., the influence of each feature on the classification accuracy can be measured. The ROC curve is plotted for a series of pairs which are each formed by a threshold value for the "classifier" feature x_i and the corresponding class label Y_i. Then, when calculating the distance of a new test instance from a training example, the distance measure is modified using the AUC score as weight for that feature.

A crucial question faced by the technique is which values of a feature should be used to help derive the ROC curve whose area corresponds to each weight. We shall show that, surprisingly, using all values of a feature is not the best strategy. Instead, considerably better performance can be obtained by selecting from an appropriate neighborhood for each feature, specific to the instances whose distance is being calculated.

Related Work. Since there exist many works related to k-NN, we can only briefly cover a representative selection. Early works addressing the improvement of k-NN include Kira and Rendell [7] and Salzberg [8], whose approaches rely on an interactive system architecture in which users are asked to rate a given similarity prediction, and then use reinforcement learning to enhance the distance function based on the user feedback. Kononenko [9] proposed an extension to this for updating feature weights based on intracluster weights.

Klein *et al.* [10] proposed a shortest path algorithm to modify a Euclidean distance function based on prior knowledge. Stein and Niggemann [11] used a neural network approach to learn weights of distance functions based on training examples.

Other approaches rely on an underlying class structure to evaluate distance functions. Han *et al.* [12] employed a randomized hill-climbing approach to learn weights of distance functions for classification tasks. In their approach, k-NN queries were used to evaluate distance functions; the k-neighbourhood of each object is analysed to determine to which extent the class labels agree with the class label of each object. Zhang [13] suggested the use of kernel functions and multidimensional scaling to learn Euclidean metrics. Hastie and Tibshirani [3] proposed algorithms that learn adaptive rectangular neighborhoods (rather than distance functions) to enhance nearest-neighbour classifiers.

Other types of approaches includes work by Hastie and Tibshirani [14] and Domeniconi *et al.* [15], who considered schemes for locally adaptive distance functions that vary throughout the input space. In particular, Domeniconi *et al.* [15] suggested using the decision boundaries of SVMs to induce a locally adaptive distance function for k-NN.

Driessens *et al.* [16] presented a two-stage classifier, YATSI, that improves its predictive accuracy by making use of the available unlabeled data. It used a weighted nearest neighbor classification algorithm using the combined example-sets as a knowledge base.

Feature weighting schemes for k-NN have also been proposed. Im and Park [17] proposed a hybrid expert system of case-based reasoning and neural network, which uses a value difference metric as the distance function for symbolic features. In another study, Vivencio *et al.* [18] proposed a feature weighting method based on the χ-squared test for k-NN.

Receiver operating characteristics have previously been used to improve classifiers. These include algorithms such as ROC-tree [19] and work by Ferri *et al.* [20], which both propose using ROC information to build decision trees. A justification for using ROC for feature evaluation, is given in Deng *et al.* [21], who demonstrate that in the context of gene expression data, ROC is a superior method to the t-test.

Contributions. Our main contributions in this paper are as follows:

- Development of a new feature weighting technique to derive a distance function for k-NN. This technique employs a range-wise feature weighting scheme based on ROC, that can dynamically determine which weights are most appropriate for the points whose distance is being measured.
- An experimental investigation which demonstrates that our new algorithm (known as *ROC-kNN*) performs strongly compared to and often outperforms well-known techniques like C4.5 [22], Random Forest, Vivencio *et al.*'s [18] χ^2-FW weighted k-NN, YATSI [16], Ferri *et al.*'s [20] AUCsplit, ROC-tree [19], and SVMs in terms of accuracy as well as overall AUC value. Surprisingly, the predictive power of *ROC-kNN* is comparable to the "black box" SVM approach, making it a very attractive tool for domain experts.

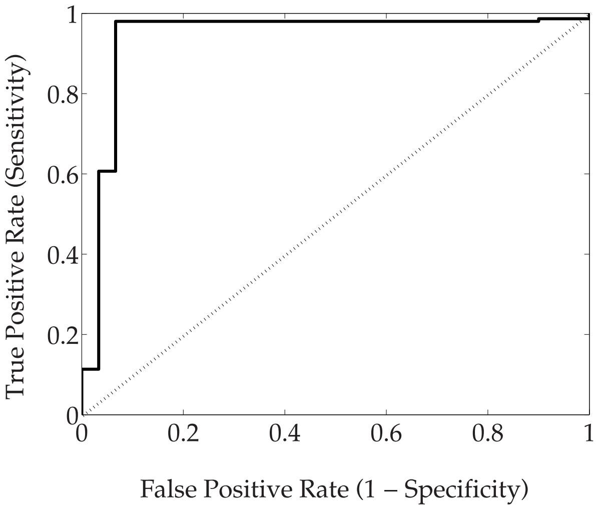


Fig. 1. A typical ROC curve

2 The Receiver Operating Characteristic Curve (ROC): Preliminaries

ROC curves were first used in signal detection theory [4]. In machine learning, the ROC curve is used to evaluate the discriminative performance of binary classifiers. This is obtained by plotting the curve of the true positive rate (*Sensitivity*) versus the false positive rate (1 – *Specificity*) for a binary classifier by varying the discrimination threshold. Figure 1 shows a typical ROC curve.

All the calculations of true positive rate and false positive rate are attained when using a particular classifier threshold. By varying the threshold, a set of values for these measurements is obtained. This set of values is plotted in a two-dimensional Cartesian graph to yield the ROC curve. The ROC curve takes into account all the possible solutions by varying the discriminative threshold. The best performance would be produced, if the ROC curve matches with the upper left corner of the ROC space (which yields 100% *sensitivity* and 100% *specificity*). The closer the ROC curve is to the upper part of the ROC space, the better the performance of the classifier.

An ROC curve is a two dimensional illustration of classifier performance. Reducing ROC performance to a single scalar value to represent expected performance helps compare classifiers. A popular method is to calculate the *area under the ROC curve (AUC)* [5].

The AUC, being a part of the area of the unit square, has a value between 0 and 1. Since random guessing could produce the diagonal line between $(0, 0)$ and $(1, 1)$ with an area of 0.5, a classifier with an AUC less than 0.5 is undesirable [23]. An AUC value close to 1 indicates better performance for a binary classifier. [24].

3 ROC-kNN: A k-NN Algorithm Using ROC Information

We now describe the steps in our algorithm, which we call *ROC-kNN*.

3.1 ROC for Feature Weighting

Previous work [25, 19, 20] has established the use of an ROC curve for feature ranking and selection, to identify the discriminative features in the context of gene expression microarray data. First, the ROC curve is plotted for each of the pairs formed by each of the features and the class label. This means treating a single feature as a classifier and calculating the classification in terms of the *sensitivity* and *specificity* by varying the operating point. We shall build on this kind of idea to derive a feature weighting method to use in distance functions. For each feature, the AUC is calculated.

Example 2. *Let us consider a dataset $\mathcal{D}$ of N instances, where each instance comprises m features: x_1, x_2, x_3, $\ldots$, x_m. Each of the m features has a differing discriminative power reflected by its respective AUC. To calculate the discriminative power that is expressed in terms of AUC, we plot the ROC curve for each feature paired with the class label, (i.e., $\{x_i, Y_i\}$, where $1 \leqslant i \leqslant m$ and Y is the vector of class labels) and calculate the AUC of this ROC curve.*

As alluded to earlier, there is a strong mathematical justification for using the ROC to measure discriminative power. It is equivalent to the Mann-Whitney U Test (also known as Wilcoxon Rank sum), a non-parametric statistical test. Not employing any distributional assumptions makes it especially useful for small sample size, noisy datasets [21], such as gene expression microarrays.

3.2 Weighted Distance Metrics

For calculating the distance of a new test instance from a training instance, we modify the standard distance measure using the AUC score as a weight. Example 3 describes how we employ the weight in **Minkowski distance** of order p (**ℓ_p-norm distance**). Recall that Euclidean distance is ℓ_2 distance, rectilinear, Manhattan or Hamming distances are ℓ_1 distance, and Chebyshev distance is ℓ_∞ distance.

Example 3. *Consider a training instance and a test instance each with m feature values: x_1, x_2, x_3, $\ldots$, x_m and y_1, y_2, y_3, $\ldots$, y_m, respectively. Then the Minkowski distance between the instances is $\delta = \left(\sum_{i=1}^{m} |x_i - y_i|^p\right)^{\frac{1}{p}}$. We can associate a weight for each feature based on its AUC score. The weighted distance function is $\Delta = \left(\sum_{i=1}^{m} (\mathcal{A}_i \cdot |x_i - y_i|)^p\right)^{\frac{1}{p}}$, where each $\mathcal{A}_i$ is the AUC value for the i-th feature.*

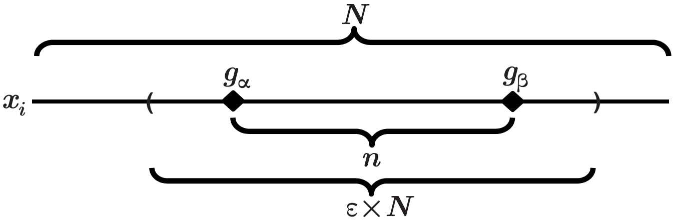


Fig. 2. Widening $[g_\alpha, g_\beta]$ so that it covers $\epsilon \times N$ values of feature x_i

3.3 Considering a Smaller Range of Feature Values to Calculate AUC

Using all values of a feature to derive the ROC curve whose area will be calculated, may not be the best way to measure the weight or "importance" of a feature. Consider the following (somewhat artificial) example.

Example 4. *Consider the power of the feature voice_pitch for predicting the class label* sex *(male/female). Suppose the* voice_pitch *feature values for the population range between 50Hz (low pitch) to 350Hz (high pitch). In general, voice_pitch is likely to be a good feature for discriminating between males and females. However, if we are only dealing with a sub-population of young children, then voice_pitch is likely to be a far less useful feature, since there is much less variation for values of this feature across young children.*

Thus, the power of a feature may need to be evaluated within some context or sub-population. A natural way to form such a context or range for the ROC calculation, is to consider the two points between which the distance is being computed. Suppose we are computing the distance between two instances P_1 and P_2 with respect to feature x_i an $P_1[x_i] = g_\alpha$ and $P_2[x_i] = g_\beta$[1]. Rather than using all values that occur for feature x_i in ROC calculation, we just use the values that lie in the interval $[g_\alpha, g_\beta]$. This range of values $[g_\alpha, g_\beta]$ corresponds to a sub-population that is more appropriate for computing the ROC of feature x_i, in the context of P_1 and P_2.

Now, it could be the case that the range $[g_\alpha, g_\beta]$ is very small. This could then to lead to low confidence in the resulting AUC calculation and estimate of feature significance, since the sample size would be too small to be statistically significant. We, therefore, generalise this idea by employing a parameter, ϵ, which can be thought of as a "coverage factor". The interval $[g_\alpha, g_\beta]$ covers some number of values (say n) of the total number of values (say N) that instances can have for feature x_i [2]. Thus, the number of values not covered by $[g_\alpha, g_\beta]$ on x_i is $N - n$. Now, ϵ varies between 0% and 100% and it is a lower bound on how much coverage we require for our interval. For example, suppose $\epsilon = 50\%$, then we require at least half the values in x_i to be covered by the

[1] Where $P_i[x_i]$ denotes the value for the instance P_i on feature x_i.

[2] Of course different instances can share the same value on x_i. Any such value will be counted more than once when calculating n or N.

Algorithm 1. CalculateROC

```
Input(s): $\mathcal{C}$: A two column matrix of training examples with the first column being the values for feature $x_i$ and the last column being values for the class label, $\epsilon$: The percentage of instance values to be covered, $g_\alpha$: Value of the training instance on $x_i$, $g_\beta$: Value of the test sample on $x_i$
Output: $\mathcal{A}$: The AUC of the attribute $x_i$
Sort $\mathcal{C}$ in descending order of $x_i$
if $g_\alpha \leqslant g_\beta$ then
    $startPoint \leftarrow g_\alpha$
    $endPoint \leftarrow g_\beta$
else
    $startPoint \leftarrow g_\beta$
    $endPoint \leftarrow g_\alpha$
end if
$N \leftarrow$ The number of instances
$SamplesToUse \leftarrow \epsilon \times N$
$SamplesInRange \leftarrow$ The collection of all values for $x_i$ which are between $[startPoint, endPoint]$
$SizeSIR \leftarrow$ Calculate the size of $SamplesInRange$
while $SizeSIR < SamplesToUse$ do
    $SamplesInRange \leftarrow SamplesInRange$ union one instance value that is adjacent to it
    Update $startPoint$ and $endPoint$ accordingly
    if No more samples can be added to either $startPoint$ or $endPoint$ then
        $startPoint \leftarrow$ The last available sample's on that side
        $endPoint \leftarrow$ The last available sample on that side
    end if
    $SizeSIR \leftarrow$ Calculate the size of $SamplesInRange$
end while
$\mathcal{RC} \leftarrow$ The samples between the range $[startPoint, endPoint]$
$\mathcal{A} \leftarrow$ AuROC($\mathcal{RC}$)
return $\mathcal{A}$
end
```

interval $[g_\alpha, g_\beta]$. Now, if $[g_\alpha, g_\beta]$ doesn't cover 50% of N, then our strategy is to widen it (symmetrically) just enough until it does achieve this level of coverage. If $\epsilon = 0$, then the interval $[g_\alpha, g_\beta]$ will never widened. If $\epsilon = 100\%$, then $[g_\alpha, g_\beta]$ will always be widened to encompass the entire range of values for feature x_i. Figure 2 illustrates the general idea.

Intuitively, range adjustment using ϵ allows the interval to be widened sufficiently so that enough feature values are available to ensure statistical significance of the ROC calculation. However, we do not want to widen the interval too much (by always choosing a high ϵ), since the resulting measure of feature discriminative power may not be focused on an appropriate population for the chosen instances P_1 and P_2 (recall Example 4). Hence, choosing a good ϵ is important. In practice, this can be done by empirically comparing classification performance for different choices of ϵ.

Algorithm 2. ROC-kNN

Input(s): $\mathcal{D} = \{(\mathbf{x}_1, Y_1), \ldots, (\mathbf{x}_n, Y_n)\}$: The matrix of n training examples with the last column being the class, $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_n)$: The test sample, k: The number of neighbours, p: The order for Minkowski distance function, ϵ: The percentage of training instances to be covered when weighting each feature

Output: $\mathcal{C}$: The class label for the test sample $\boldsymbol{\tau}$

```
for each labelled instance $(\mathbf{x}_i, Y_i), (i = 1, \ldots, n)$ do
    for each feature $a_j, 1 < j <$ Number of features, $m$ do
        $\mathcal{A}_j = \textsc{CalculateROC}(\{x_j, Y\}, \epsilon, \mathbf{x}_i, \boldsymbol{\tau})$   /* $\mathcal{A}$ is a vector of AUC scores for all features */
    end for
    Calculate $\Delta(\mathbf{x}_i, \boldsymbol{\tau}) = \left( \sum_{j=1}^{m} (\mathcal{A}_j \cdot |\mathbf{x}_i[a_j] - \boldsymbol{\tau}[a_j]|)^p \right)^{\frac{1}{p}}$
end for
Sort $\Delta(\mathbf{x}_i, \boldsymbol{\tau})$ in ascending order
$\mathcal{D}_{\boldsymbol{\tau}}^k$=$k$ nearest instances to $\boldsymbol{\tau}$
$\mathcal{C} \leftarrow$ most frequent class in $\mathcal{D}_{\boldsymbol{\tau}}^k$
return $\mathcal{C}$
end
```

Example 5. *Suppose the 15 instances of the training dataset have values* $\{1, 2, 2, 3, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 12\}$ *for feature* x_i *and* $P_1[x_i] = 8$, $P_2[x_i] = 9$ *and* $\epsilon = 40\%$. *Then the interval* $[8, 9]$ *must be widened so that it includes* $0.4 \times 15 = 6$ *of the values that occur for* x_i *in training instances. This can be accomplished by (symmetrically) widening it to be* $[6, 11]$. *So the discriminative power of* x_i *in this situation will be measured by the AUC of the ROC curve derived from when* x_i *takes values* $\{6, 7, 8, 9, 10, 11\}$.

Definition 1. *Given instances* P_1 *and* P_2 *from a dataset with* m *features* $x_1, \ldots, x_m$ *and values for parameters* ϵ *and* p. *The weighted distance between* P_1 *and* P_2 *using range adjusted ROC is calculated as*

$$\left(\sum_{i=1}^{m} (\mathcal{A}_i \cdot |P_1[x_i] - P_2[x_i]|)^p \right)^{\frac{1}{p}},$$

where each $\mathcal{A}_i$ *measures the discriminative power using ROC of feature* x_i *in the interval* $r = [P_1[x_i] - \alpha_1, P_2[x_i] + \alpha_2]$ *for some* α_1, α_2. *The interval* r *covers at least* $\epsilon\%$ *of the values taken by feature* x_i *for instances in the training dataset.*

Thus, the weighting of each feature x_i is specific to the points whose distance is being computed. Carrying out the weight calculation can be done at either runtime (k-NN classification time) or during training. If the latter, then one must assume test instances do not contain any feature values that are not present in the training data. In that situation, one precomputes the ROC for every contiguous interval of values for each feature x_i and then selects the appropriate ROC weight value at runtime, according to the values on x_i of the instances which are being compared.

Table 1. Properties of the datasets used in this study

Dataset	No. of Attributes	No. of Instances	Collected from	First used by
GE1	24,481	97	Integrated Tumor Transcriptome Array and Clinical data Analysis database [27]	van 't Veer *et al.* [26]
GE2	3,226	22	National Human Genome Research Institute	Hedenfalk *et al.* [28]
GE3	12,533	181	Division of Thoracic and Surgery [30], Brigham Women's Hospital, Boston	Gordon *et al.* [29]
GE4	12,600	21	Cancer Program [31], Broad Institute of MIT and Harvard	Singh *et al.* [32]
GE5	12,600	136	Cancer Program [31], Broad Institute of MIT and Harvard	Singh *et al.* [32]
GE6	7,129	72	Cancer Program [33], Broad Institute of MIT and Harvard	Golub *et al.* [34]
Hepatitis	19	155	UCI ML Repository [35]	–
Ionosphere	34	351	UCI ML Repository [35]	–
Pima	8	768	UCI ML Repository [35]	–
WBC	9	699	UCI ML Repository [35]	–
WDBC	30	569	UCI ML Repository [35]	–
WPBC	33	198	UCI ML Repository [35]	–

Algorithm 1 presents the pseudocode for calculating the AUC for a specified feature x_i. It relies the existence on the function AuROC function provided in [19]. The overall procedure is described in Algorithm 2 (which assumes weight computation is done at runtime).

4 Experimental Design

k-NN is actually a family of techniques, according to k value and distance function used. In our evaluation, we tested using different values of k (1, 3 and 5) and different choices of p for the Minkowswki ℓ_p-norm distance (1, 2 and ∞). For our algorithm, *ROC-kNN*, we also needed to test using different values for the parameter ϵ (100%, 95%, 90%, …, 0%).

Based on this testing, for k-NN and *ROC-kNN* on each dataset, we identified the values of p, k and ϵ that produced the best classification performance and this is what is reported in the result tables.

In addition to comparing against traditional k-NN, we compared *ROC-kNN* against thirteen other techniques. These are: Vivencio *et al.*'s [18] χ^2-FW weighted k-NN, YATSI [16], ROC-tree [19], C5.0, its predecessor C4.5 [22], Random Forest, Ferri *et al.*'s [20] AUCsplit technique for decision trees, Naïve Bayes and SVMs using two different kernels: polynomial and radial basis function (RBF). Where applicable, each of these classifiers was run multiple times on each dataset by varying its parameters. We report the best result of each such classifier on each dataset across the variation of its parameters. Since the χ-squared test can only handle

Table 2. Comparison of accuracy results from 10 × 10-fold cross validation on six gene expression datasets

Method	GE1	GE2	GE3	GE4	GE5	GE6
ROC-kNN	63.29 ± 3.13	71.07 ± 4.22	98.66 ± 0.39	61.49 ± 2.12	84.65 ± 1.22	90.33 ± 0.89
k-NN	58.45 ± 2.88	61.82 ± 6.14	94.64 ± 0.27	57.14 ± 3.89	82.35 ± 1.80	88.89 ± 0.93
χ^2-FW	49.90 ± 3.64	50.00 ± 10.05	90.77 ± 0.45	58.57 ± 5.96	80.81 ± 1.78	89.72 ± 1.63
YATSI	56.70 ± 3.21	45.45 ± 4.19	93.92 ± 1.09	57.14 ± 1.93	72.06 ± 1.99	84.72 ± 1.57
ROC-tree	$\mathbf{72.16 \pm 4.32}$	$\mathbf{77.27 \pm 2.45}$	98.34 ± 0.89	38.10 ± 5.95	88.24 ± 2.33	94.44 ± 2.96
AUCsplit	63.58 ± 4.59	74.39 ± 1.63	96.14 ± 1.36	34.01 ± 2.87	82.47 ± 3.96	81.61 ± 3.28
C5.0	64.95 ± 6.21	59.09 ± 4.52	92.82 ± 1.21	23.81 ± 4.65	81.62 ± 4.12	80.55 ± 3.74
C4.5	62.89 ± 3.11	72.73 ± 1.36	95.03 ± 1.05	33.33 ± 4.59	79.42 ± 5.45	79.17 ± 4.87
ADTree	61.86 ± 4.28	68.18 ± 5.68	92.82 ± 2.19	32.86 ± 3.44	86.76 ± 2.63	86.11 ± 3.77
REPTree	52.18 ± 5.45	59.09 ± 3.92	95.03 ± 1.28	32.86 ± 3.46	80.88 ± 3.33	81.94 ± 4.26
Random Tree	55.67 ± 3.54	63.64 ± 2.58	79.56 ± 2.69	32.86 ± 3.12	62.50 ± 5.23	75.00 ± 3.90
Random Forest	62.89 ± 6.43	50.00 ± 5.33	93.92 ± 1.22	38.10 ± 5.27	80.88 ± 2.56	79.17 ± 2.36
Naïve Bayes	54.64 ± 3.38	59.09 ± 4.58	98.34 ± 0.03	33.33 ± 0.78	55.88 ± 4.76	98.61 ± 1.03
SVM (poly)	68.04 ± 2.14	59.09 ± 2.98	$\mathbf{99.45 \pm 0.11}$	47.62 ± 5.63	$\mathbf{91.18 \pm 3.12}$	$\mathbf{98.61 \pm 1.26}$
SVM (RBF)	67.01 ± 2.36	63.64 ± 0.94	98.34 ± 1.41	$\mathbf{61.90 \pm 1.39}$	69.12 ± 5.31	80.56 ± 2.18

discrete data, we used entropy-based discretization [36] to discretize the datasets before applying χ^2-FW.

Each of the classifiers was applied on 12 datasets, of which 6 were gene expression datasets and 6 are non-gene expression datasets having rather different characteristics. The properties of the datasets are illustrated in Table 1. Recall that gene expression datasets represent one of the most challenging scenarios for k-NN algorithms. For each classifier and dataset, a 10-fold cross validation (CV) scheme was used 10 times.

Using a fixed 10-fold cross validation scheme, we also conducted a win-draw-loss analysis based on a paired t-test with 5% significance level, for six of the classifiers.

5 Results and Discussion

The classification results for all 14 techniques on the considered datasets are presented in Table 2 and 3. The best performances among that of the reported classifiers are marked in bold. Since for the non-gene expression datasets, different ϵ values yielded best classification accuracy on different datasets, we also include a separate entry for *ROC-kNN* with $\epsilon = 100\%$. For the gene expression datasets, $\epsilon = 100\%$ always yielded the best result. We show the chosen ϵ values for *ROC-kNN* in Table 4. We also show in that table the chosen values for k and and p for both *ROC-kNN* and k-NN.

ROC-kNN versus k-NN Techniques. Looking at Table. 2 and 3, the classification performance of *ROC-kNN* shows considerable improvement over the traditional k-NN technique. For all datasets and different k values, *ROC-kNN* has improved accuracy over k-NN.

Results of statistical significance tests we conducted (Bonferroni-corrected t-tests [37]) confirm that the improvement in accuracy is statistically significant. For example, when comparing the best results of the *ROC-kNN* with that of the k-NN, on 5 of the 6 non-gene expression datasets (hepatitis, ionosphere, WBC,

Table 3. Comparison of accuracy results from 10 × 10-fold cross validation on six non-gene expression datasets

Method	Hepatitis	Ionosphere	WBC	WDBC	WPBC	Pima
ROC-kNN	$\mathbf{85.59 \pm 0.55}$	90.56 ± 0.22	$\mathbf{98.26 \pm 0.23}$	$\mathbf{98.49 \pm 0.21}$	79.26 ± 0.51	$\mathbf{88.51 \pm 0.96}$
ROC-kNN ($\epsilon = 100\%$)	82.98 ± 0.87	88.60 ± 0.12	97.07 ± 0.14	97.47 ± 0.24	77.02 ± 0.68	86.60 ± 1.09
k-NN	82.58 ± 0.80	87.35 ± 0.59	96.88 ± 0.39	97.24 ± 0.29	76.21 ± 1.31	85.28 ± 1.87
χ^2-FW	78.84 ± 1.60	53.05 ± 13.27	95.55 ± 0.28	95.87 ± 0.59	70.76 ± 1.36	70.83 ± 0.61
YATSI	80.00 ± 2.12	83.48 ± 2.39	96.28 ± 1.01	94.73 ± 1.48	71.21 ± 2.76	73.70 ± 1.98
ROC-tree	78.71 ± 7.65	84.05 ± 9.87	92.56 ± 5.43	90.69 ± 6.78	69.67 ± 8.33	63.54 ± 8.65
AUCsplit	82.10 ± 3.43	86.00 ± 7.31	95.88 ± 1.94	93.75 ± 3.39	70.53 ± 9.67	73.82 ± 5.35
C5.0	76.13 ± 2.35	89.46 ± 1.23	93.64 ± 1.65	93.29 ± 2.23	70.70 ± 4.12	73.94 ± 2.76
C4.5	80.00 ± 4.45	91.45 ± 3.36	93.84 ± 2.63	93.15 ± 1.26	75.25 ± 3.32	73.83 ± 2.89
ADTree	76.13 ± 2.96	$\mathbf{93.16 \pm 1.65}$	95.14 ± 1.77	94.02 ± 1.06	77.78 ± 5.42	72.92 ± 3.23
REPTree	78.71 ± 4.23	89.46 ± 1.46	93.99 ± 2.14	92.44 ± 2.33	73.74 ± 4.85	75.39 ± 4.55
Random Tree	72.91 ± 9.21	87.75 ± 3.64	94.13 ± 2.85	89.46 ± 3.67	68.18 ± 5.45	67.97 ± 6.49
Random Forest	81.94 ± 1.26	92.59 ± 3.26	95.99 ± 1.45	95.25 ± 1.37	78.28 ± 3.47	73.70 ± 4.98
Naïve Bayes	83.87 ± 1.71	82.62 ± 3.48	95.99 ± 0.74	92.97 ± 2.58	67.68 ± 5.08	76.30 ± 3.49
SVM (poly)	76.77 ± 4.23	88.60 ± 2.43	96.85 ± 1.07	97.72 ± 1.04	76.26 ± 4.78	77.34 ± 5.01
SVM (RBF)	84.52 ± 4.02	91.74 ± 5.15	96.85 ± 1.29	96.07 ± 3.12	$\mathbf{81.31 \pm 2.36}$	77.47 ± 3.73

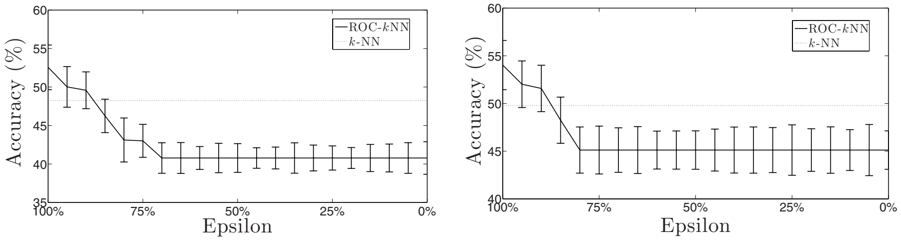
Table 4. The parameters of k-NN and *ROC-kNN*

Dataset	k-NN		*ROC-kNN*			
	k	p	k	p	ϵ	$\epsilon \times N$
GE1	3	∞	1	∞	100%	88 ± 0.0
GE2	1	2	1	2	100%	20 ± 0.0
GE3	1	2	3	∞	100%	163 ± 0.0
GE4	5	2	1	2	100%	19 ± 0.0
GE5	3	1	1	1	100%	123 ± 0.0
GE6	1	1	1	1	100%	65 ± 0.0
Hepatitis	3	2	5	2	65%	91 ± 1.8
Ionosphere	1	∞	5	∞	70%	221 ± 2.3
WBC	5	2	5	2	60%	378 ± 1.5
WDBC	3	1	3	1	75%	384 ± 2.75
WPBC	3	1	3	1	85%	152 ± 2.15
Pima Indians	3	2	3	2	55%	380 ± 2.6

WDBC and Pima Indians) the p-value is less than 0.05 (0.041533, 0.011329, 0.029657, 0.046574 and 0.034136, respectively). WPBC showed milder improvement (p-value is 0.1056256). For gene expression data, Bonferroni-corrected t-tests confirmed significant improvement of *ROC-kNN* over k-NN (p-values are 0.04123, 0.01298, 0.03192, 0.04871, 0.03907 and 0.04516 for GE1 to GE6 datasets, respectively).

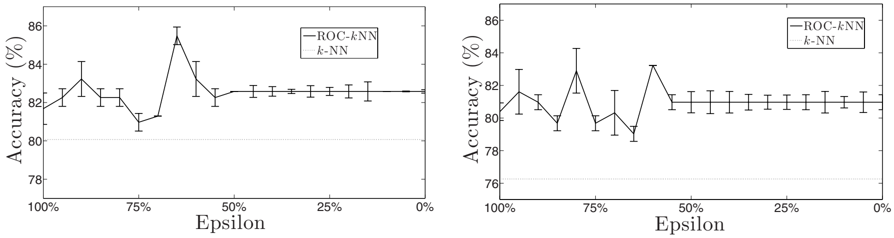
The results for the χ^2-FW weighted k-NN technique show it mostly failed to improve classification accuracy over k-NN. This is interesting, since both *ROC-kNN* and χ^2-FW use feature weighting to extend k-NN, yet each uses a rather different method. The poor performance of the χ^2-FW technique may be because it is intended more for discrete data, rather than the continuous datasets used in our evaluation (and thus it required an extra discretisation step).

ROC-kNN versus non k-NN Classifiers. Surprisingly and pleasingly, the classification performance of *ROC-kNN* on the UCI ML repository datasets is extremely strong. It outperforms all the other non k-NN classifiers, except on



(a) ℓ_1-norm distance (b) ℓ_2-norm distance

Fig. 3. The effect of ϵ on classification accuracy for GE1 dataset



(a) ℓ_2-norm distance (b) ℓ_∞-norm distance

Fig. 4. The effect of ϵ on classification accuracy for the hepatitis dataset

Table 5. Comparison of AUC value from 10 × 10-fold cross validation on six gene expression datasets

Method	GE1	GE2	GE3	GE4	GE5	GE6
ROC-kNN	0.5992 ± 0.02	$\mathbf{0.6721 \pm 0.11}$	0.9665 ± 0.02	0.4812 ± 0.12	$\mathbf{0.8386 \pm 0.02}$	0.8743 ± 0.03
k-NN	0.5793 ± 0.04	0.5064 ± 0.14	0.8463 ± 0.04	0.4378 ± 0.13	0.8213 ± 0.02	0.8585 ± 0.03
χ^2-FW	0.4986 ± 0.08	0.4958 ± 0.12	0.7418 ± 0.05	$\mathbf{0.5174 \pm 0.11}$	0.8015 ± 0.03	0.8682 ± 0.04
YATSI	$\mathbf{0.6230 \pm 0.00}$	0.4150 ± 0.02	$\mathbf{0.9820 \pm 0.00}$	0.4760 ± 0.01	0.8050 ± 0.00	$\mathbf{0.905 \pm 0.00}$

ionosphere and WPBC. This provides some evidence that *ROC-kNN* is a classifier which is able to compete with and even surpass mainstream state-of-the-art techniques in these circumstances. On the gene expression datasets, *ROC-kNN* performs quite strongly compared to the non-k-NN classifiers, but it is not the standout performer. This could be because for these datasets, the number of instances is very small and thus the ROC measure of feature discriminative power is less reliable.

Influence of the ϵ parameter: Figure 3 and 4 show the effect on classification performance for varying ϵ on both the GE1 dataset and the hepatitis dataset along with the improvement over k-NN, which is represented by a dotted line in the figures. Table 4 also shows the best ϵ values for each dataset. Though the effect of ϵ is not entirely the same on every dataset, there are some clear trends.

Table 6. Comparison of AUC value from 10×10-fold cross validation on six non-gene expression datasets

Method	Hepatitis	Ionosphere	WBC	WDBC	WPBC	Pima
ROC-kNN	0.7491 ± 0.02	0.8790 ± 0.02	0.9816 ± 0.01	0.9799 ± 0.00	**0.7815 ± 0.04**	**0.8728 ± 0.01**
k-NN	0.7306 ± 0.03	0.8322 ± 0.02	0.9675 ± 0.00	0.9694 ± 0.01	0.6278 ± 0.03	0.7894 ± 0.03
χ^2-FW	0.5928 ± 0.04	0.5139 ± 0.03	0.9530 ± 0.01	0.9499 ± 0.01	0.5228 ± 0.04	0.6508 ± 0.01
YATSI	**0.7820 ± 0.00**	**0.9200 ± 0.00**	**0.9890 ± 0.00**	**0.9830 ± 0.00**	0.575 ± 0.01	0.7600 ± 0.01

Table 7. Win-Draw-Loss results for the top six top classifiers using t-test on 72 test combinations for each classifier

Method	Win	Draw	Loss
ROC-kNN	**46**	**20**	**6**
k-NN	29	26	17
χ^2-FW	8	20	44
YATSI	11	24	37
ROC-tree	22	28	22
SVM (poly)	19	38	15
SVM (RBF)	18	42	12

For gene expression data, $\epsilon = 100\%$ is (unsurprisingly) always the best choice, due to the small number of instances in this type of data and classification accuracy falls monotonically with decreasing ϵ. However, in non-gene expression data, the trend is quite different. Classification accuracy can increase and decrease according to varying ϵ and the best accuracy is achieved with different ϵ values for each dataset. It is worth noting that for all datasets, the performance of the classifier becomes constant once ϵ drops below a certain value (see Fig. 3 and 4). This is because, after a point, the ϵ constraint becomes too loose and never forces any widening of a feature interval.

Comparison of AUC Values: We also computed the overall AUC value of selected k-NN style classifiers, shown in Table 5 and 6, resulting from the 10×10 cross validation over the gene expression and UCI ML repository datasets. The AUC values of the four considered k-NN classifiers: *ROC-kNN*, k-NN, χ^2-FW weighted k-NN and YATSI reflect the much the same behaviour as seen for classification accuracy, except for the YATSI classifier. Though its classification accuracy is not as good as *ROC-kNN* or even traditional k-NN, YATSI performed slightly better than *ROC-kNN* for 3 out of 6 gene expression datasets and 4 out of 6 non-gene expression data in terms of AUC. *ROC-kNN* has improved AUC compared to k-NN in both the gene expression and non-gene expression data.

Statistical significance tests: We also carried out win-draw-loss analysis based on paired t-test with 5% significance level for the seven most promising classifiers (see Tab. 7). In this analysis, *ROC-kNN* arguably outperforms all the other techniques, as it has a significant number of wins and only a few losses, compared to the other classifiers. A more detailed look at *ROC-kNN*'s performance against the other classifiers (see Tab. 8), also reveals that it has significant

Table 8. Win-Draw-Loss result for *ROC-kNN* versus the top five classifiers using t-test

Method	GE1	GE2	GE3	GE4	GE5	GE6	Hepatitis	Ionosphere	WBC	WDBC	WPBC	Pima
k-NN	Win	Win	Win	Win	Win	Win	Win	Win	Win	Win	Win	Win
χ^2-FW	Win	Win	Win	Win	Win	Win	Win	Win	Win	Win	Win	Win
YATSI	Win	Win	Draw	Win	Win	Win	Win	Win	Draw	Win	Win	Win
ROC-tree	Loss	Draw	Draw	Loss	Draw	Draw	Win	Win	Win	Win	Win	Win
SVM (poly)	Draw	Win	Draw	Win	Loss	Loss	Win	Draw	Draw	Draw	Draw	Win
SVM (RBF)	Loss	Draw	Draw	Loss	Draw	Draw	Win	Draw	Draw	Draw	Draw	Win

improvements over all other considered k-NN techniques. Against ROC-tree, *ROC-kNN* significantly improves for the non-gene expression datasets, but performs mostly the same on gene expression data. Against the two SVM classifiers, *ROC-kNN* no worse and indeed actually improves for two of the six considered non-gene expression datasets. However, on gene expression data, the improvement is more marginal.

6 Conclusion

This paper has presented a new k-NN style algorithm for classification, based on a new method for defining a weighted distance function. Our method is based on considering the ROC characteristics of each feature, within a neighbourhood appropriate to the instances whose distance is being computed.

Experimental analysis shows our technique, *ROC-kNN*, is able to substantially improve over basic k-NN. Furthermore, it performs very strongly compared to other state-of-the-art classifiers and is even able to deliver improved accuracy in many cases.

For future work, we would like to consider extending our approach i) to incorporate pruning of irrelevant features and ii) to situations where there are three or more classes in the data, a well known challenge for ROC computation.

Acknowledgements. This work is partially supported by National ICT Australia. National ICT Australia is funded by the Australian Government's Backing Australia's Ability initiative in part through the Australian Research Council.

References

1. Theilhaber, J., et al.: Finding genes in the C2C12 osteogenic pathway by k-nearest-neighbor classification of expression data. Genome Research 12(1), 165–176 (2002)
2. Wu, W., Xing, E., Mian, I., Bissell, M.: Evaluation of normalization methods for cDNA microarray data by k-NN classification. BMC Bioinformatics 6(191), 1–21 (2005)
3. Hastie, T., Tibshirani, R.: Discriminant adaptive nearest-neighbor classification. IEEE Transactions on Pattern Analysis Machine Intelligence 18(6), 607–616 (1996)

4. Green, D.M., Swets, J.M.: Signal detection theory and psychophysics. John Wiley & Sons Inc., New York (1966)
5. Hanley, J.A., McNeil, B.J.: The meaning and use of the area under a receiver operating characteristic (ROC) curve. Radiology 143, 29–36 (1982)
6. Bradley, A.P.: The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition 30(7), 1145–1159 (1997)
7. Kira, K., Rendell, L.: A practical approach to feature selection. In: Proc. of ICML, pp. 249–256 (1992)
8. Salzberg, S.: Distance metrics for instance-based Learning. In: Raś, Z.W., Zemankova, M. (eds.) ISMIS 1991. LNCS, vol. 542, pp. 399–408. Springer, Heidelberg (1991)
9. Kononenko, I.: Estimating attributes: Analysis and extensions of Relief. In: Bergadano, F., De Raedt, L. (eds.) ECML 1994. LNCS, vol. 784, pp. 171–182. Springer, Heidelberg (1994)
10. Klein, D., Kamvar, S.D., Manning, C.D.: From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In: Proc. of ICML, pp. 307–314 (2002)
11. Stein, B., Niggemann, O.: Generation of similarity measures from different sources. In: 14th International Conference on Industrial Engineering Applications of Artificial Intelligence and Expert Systems, pp. 197–206 (2001)
12. Han, E.H., Karypis, G., Kumar, V.: Text categorization using weight-adjusted nearest-neighbor classification. In: Conference on Knowledge Discovery and Data Mining, Hong Kong, China, pp. 53–65 (2001)
13. Zhang, Z.: Learning metrics via discriminant kernels and multidimensional scaling: toward expected Euclidean representation. In: Proc. of ICML, pp. 872–879 (2003)
14. Hastie, T., Tibshirani, R.: Discriminant Adaptive Nearest Neighbor Classification and Regression. In: Advances in Neural Information Processing Systems, vol. 8, pp. 409–415 (1996)
15. Domeniconi, C., Gunopulos, D., Peng, J.: Large margin nearest neighbor classifiers. IEEE Transactions on Neural Networks 16(4), 899–909 (2005)
16. Driessens, K., Reutemann, P., Pfahringer, B., Leschi, C.: Using Weighted Nearest Neighbor to Benefit from Unlabeled Data. In: Ng, W.-K., Kitsuregawa, M., Li, J., Chang, K. (eds.) PAKDD 2006. LNCS (LNAI), vol. 3918, pp. 60–69. Springer, Heidelberg (2006)
17. Im, K.H., Park, S.C.: Case-based reasoning and neural network based expert system for personalization. Expert Systems with Applications 32, 77–85 (2007)
18. Vivencio, D.P., et al.: Feature-weighted k-Nearest Neighbor Classifier. In: Proc. of FOCI, pp. 481–486 (2007)
19. Hossain, M.M., Hassan, M.R., Bailey, J.: ROC-tree: A Novel Decision Tree Induction Algorithm Based on Receiver Operating Characteristics to Classify Gene Expression Data. In: Proc. of 8th SIAM Int'l Conf. on Data Mining (SDM 2008), pp. 455–465 (2008)
20. Ferri, C., Flach, P., Hernández-Orallo, J.: Learning decision trees using the area under the ROC curve. In: Proc. of ICML, pp. 139–146 (2002)
21. Deng, L., Pei, J., Ma, J., Lee, D.L.: A Rank Sum Test Method for Informative Gene Discovery. In: Proc. of KDD, pp. 410–419 (2004)
22. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo (1993)
23. Fawcett, T.: ROC Graphs: Notes and Practical Considerations for Researchers. Technical Report MS 1143, HP Laboratories (2004)

24. Egan, J.P.: Signal Detection Theory and ROC Analysis. Academic Press Series in Cognition and Perception. Academic Press, London (1975)
25. Mamitsuka, H.: Selecting features in microarray classification using ROC curves. Pattern Recognition 39(12), 2393–2404 (2006)
26. van 't Veer, L.J., et al.: Gene expression profiling predicts clinical outcome of breast cancer. Nature 415, 530–535 (2002)
27. Integrated Tumor Transcriptome Array and Clinical Data Analysis (2006), `http://bioinfo-out.curie.fr/ittaca`
28. Hedenfalk, I., et al.: Gene-expression profiles in hereditary breast cancer. The New England Journal of Medicine 344(8), 539–548 (2001)
29. Gordon, G.J., et al.: Translation of Microarray Data into Clinically Relevant Cancer Diagnostic Tests Using Gene Expression Ratios in Lung Cancer And Mesothelioma. Cancer Research 62, 4963–4967 (2002)
30. The Division of Thoracic Surgery (2002), `http://www.chestsurg.org/publications/2002-microarray.aspx`
31. Broad Institute Cancer Program (2002), `http://www.broad.mit.edu/cgi-bin/cancer/publications/pub_paper.cgi?mode=view&paper_id=75`
32. Singh, D., et al.: Gene Expression Correlates of Clinical Prostate Cancer Behavior. Cancer Cell 1, 203–209 (2002)
33. Broad Institute Cancer Program (1999), `http://www.broad.mit.edu/cgi-bin/cancer/publications/pub_paper.cgi?mode=view&paper_id=43`
34. Golub, T.R., et al.: Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. Science 286, 531–537 (1999)
35. Newman, D., et al.: UCI Repository of machine learning databases. Online Repository (1998), `http://www.ics.uci.edu/~mlearn/MLRepository.html`
36. Perner, P.: Methods for Data Mining. In: Data Mining on Multimedia Data. LNCS, vol. 2558, pp. 23–89. Springer, Heidelberg (2002)
37. Glantz, S.A.: Primer of BioStatistics, pp. 309–310. McGraw-Hill, NY (1992)