Weighted K-Nearest Neighbor Revisited

M. Bicego
University of Verona
Verona, Italy
Email: manuele.bicego@univr.it

M. Loog
Delft University of Technology
Delft, The Netherlands
Email: m.loog@tudelft.nl

***Abstract*—In this paper we show that weighted K-Nearest Neighbor, a variation of the classic K-Nearest Neighbor, can be reinterpreted from a classifier combining perspective, specifically as a fixed combiner rule, the sum rule. Subsequently, we experimentally demonstrate that it can be rather beneficial to consider other combining schemes as well. In particular, we focus on trained combiners and illustrate the positive effect these can have on classification performance.**

I. Introduction

The K-nearest neighbor (KNN) rule is a widely used and easy to implement classification rule. It assigns a point x to the class most present among the K points in the training set nearest to x [1], [2], [3], [4]. Deciding which points are nearest is done according to some prespecified distance. In this procedure, all points within the neighborhood contribute equally to the final decision for x. It seems obvious, therefore, to allow for weighted voting (of some sort) in order to improve performance. Royall was probably the first to seriously consider this option [5]: he demonstrated that improvements are indeed possible in the regression setting under squared loss.

In the classification setting, Dudani [6] was the first to introduce a specific distance-weighted KNN rule and provided empirical evidence of its admissibility. He discussed some alternatives to define the weights, all with weights dropping in terms of the distance to x – with a weight of 1 for the first nearest neighbor and a weight of 0 for the Kth. Given the weights, each neighbor of x contributes to the final decision with its own weight: in particular, the Weighted K-Nearest-Neighbor (WKNN) rule assigns x to that class for which the weights of the representatives among the K nearest neighbors sum to the greatest value [6] (see. Fig. (1)).

The weighting scheme introduced by Dudani [6], even when weights are cleverly chosen, is not necessarily helpful as, for instance, demonstrated in [7]. The paper showed that, asymptotically, unweighted KNN is to be preferred over any weighted version in case we fix K. However, when dealing with the realistic setting of finite samples, improvements are possible (see [8] for instance). Clearly, whether weighting can help also depends on what we consider as improvement [5], [8], [9]. Though weighted KNN rules are used in various applications, little conceptual, theoretical, or methodological advances have been made in the past decades. Two recent additions to this literature include [10] and [11]. In [10], a so-called dual distance function is considered, which turns out to be less sensitive to the choice of K and supposedly avoids degradation of the classification accuracy in the small sample case and when dealing with outliers. In [11], the authors derive an asymptotically optimal way of defining nonnegative weights to be used within the WKNN scheme.

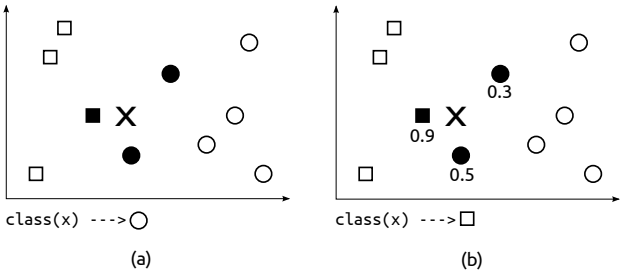

Fig. 1: Example of (a) K-Nearest Neighbor and (b) Weighted K-Nearest Neighbor (K = 3). With KNN, every neighbor counts in the same way for the final decision: in the case shown in figure, the cross is assigned to the circle class, the most frequent class in the neighborhood. On the contrary, with Weighted KNN every neighbor has associated a weight; in the final decision, each neighbor counts with its own weight: in the example, since the sum of the weights of the neighbors from the square class is larger than that of the neighbors of the circle class, the cross is assigned to the square class.

In this work, we reinterpret the Weighted KNN (and the KNN) from a classifier combining perspective [12]: we show that KNN can be seen as a plain *majority voting* scheme and, generally, the weighted KNN as a *fixed combiner rule* (the sum rule). This view opens the door to the use of other classifier combiners and we show that it can indeed be quite beneficial to consider alternative and more advanced schemes. In particular, here we focus on trained combining schemes [13], [14], for which with our experiments demonstrate potentially significant improvements in classification performances over the original KNN weighting scheme by Dudani [6].

A. Outline

Section II introduces the necessary background on KNN, its weighted variant, and classifier combiners, while fixing notation. Section III offers our interpretation of KNN as a combining scheme and sketches how various combiners could be integrated using the terminology of matching scores. The next section, Section IV, describes the experiments that were carried out with our revisited KNN using a trained combiner. It

also reports on the results and discusses them. Finally, Section V concludes.

II. Preliminaries and Additional Background

In this section we introduce the necessary background on KNN, the WKNN, and the theory of classifier combiners, while fixing notation.

A. K-Nearest Neighbor

Let us start with some definitions:

- x: the pattern to be classified;
- $\{x_i\}$ (with $1 \leq i \leq N$): the set of N points in the training set; each training pattern is equipped with a label $\{y_i\}$ (with $1 \leq i \leq N$). The label y_i can be one of the possible values $1 \dots C$, where C is the number of classes of the problem at hand
- $ne_K(x) = \{n_1, ..., n_K\}$: the K points in the training set which are nearest to x according to a certain distance $d(\cdot, \cdot)$; $y_{n_1}, ...y_{n_K}$ are the corresponding labels; please note that we consider $\{n_1, ..., n_K\}$ as ordered according to the distance from x — n_1 is the nearest neighbor, n_K is the farthest of the K nearest neighbors.

Given these definitions, the standard KNN rule assigns x to the class $\hat{c}$ more frequent in the set $ne_K(x)$, i.e.

$$x \longleftarrow \arg\max_c |\{n_i : y_{n_i} = c\}| \tag{1}$$

where $|X|$ denotes the cardinality of the set X. Rule (1) can be rewritten as

$$x \longleftarrow \arg\max_c \left[\sum_{i=1}^{K} I_c(n_i)\right] \tag{2}$$

where $I_c(z)$ is the indicator function for class c

$$I_c(z) = \begin{cases} 1 & \text{if } z \text{ belongs to class } c \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

The summation in (2), for a given c, simply counts the number of points in the neighborhood $ne_K(x)$ belonging to class c.

B. Weighted K-Nearest Neighbor

Within the Weighted K-Nearest Neighbor rule [6], each neighbor $n_i \in ne_K(x)$ is equipped with a weight w_{n_i}, which can be computed using for example the methods presented in [6], [11]. Note that in the general setting, we may have a different set of weights for every point to be classified: when changing the point x to be classified, also the neighborhood $ne_K(x)$ changes and therefore the corresponding weights, as they typically depend directly on the relation between the neighbors and the point x. This is clear, for instance, when considering the definition of weights introduced in Equation (2) of [6]:

$$w_{n_i} = \frac{1}{d(x, n_i)} \tag{4}$$

With this definition, the weight of a given training example is different when changing the point x to be classified – it depends on the distance from such x: the more distant to the neighbor, the lower its weight/importance in the classification of x. This definition of weights takes inspiration from ideas typical of the Parzen Windows estimator [15].

Given neighbors and weights, the Weighted K-nearest neighbor rule assigns x to the class $\hat{c}$ for which the weights of its representatives in the neighborhood $ne(x)$ sum to the greatest value. Following the notation of Equation (2),

$$x \longleftarrow \arg\max_c \left[\sum_{i=1}^{K} I_c(n_i) w_{n_i}\right] \tag{5}$$

Clearly, the KNN and the Weighted KNN rules are equivalent when K=1.

C. Classifier Combining

The main idea behind the Classifier Combining theory [16], [12] is that it is possible to improve the classification accuracy by exploiting the diversity present in different pattern recognition systems. Such diversity can derive from the employment of different sensors, different features, different training sets, different classifiers or others [12]. In particular, here we focus on the following scenario: we have a set of M different classifiers (experts) $E_1, ...E_M$. Given a classification problem involving C classes, and a pattern x to be classified, every classifier E_ℓ returns a set of values $E_\ell(x)$:

$$E_\ell(x) = [e_{\ell 1}(x), e_{\ell 2}(x), \cdots, e_{\ell C}(x)]$$

where $e_{\ell c}(x)$ can be a posterior of the class c —i.e $e_{\ell c}(x) = P(c|x)$— or simply a *matching score*, i.e. a number indicating how likely is that the class of x is c (called *confidences* in [12]). A given classifier (expert) E_ℓ takes a decision on x with the following rule

$$x \longleftarrow \arg\max_c e_{\ell c}(x) \tag{6}$$

Given a pool of M classifiers, the goal is to combine the values present in the following matrix

$$E(x) = \begin{bmatrix} e_{11}(x) & e_{12}(x) & \cdots & e_{1C}(x) \\ e_{21}(x) & e_{22}(x) & \cdots & e_{2C}(x) \\ \vdots & \vdots & \ddots & \vdots \\ e_{M1}(x) & e_{M2}(x) & \cdots & e_{MC}(x) \end{bmatrix}$$

to reach a classification that is potentially better than those of the single classifiers. Many methods have been proposed in the past to address this problem ([16], [12], [17], [18], just to cite a few), which are based on different ideas, intuitions, or hypotheses. Here we summarize the following three classes of approaches, which will become useful in the remainder of this work.

1) Combination of decisions: In this case, each expert E_ℓ takes its own decision; the final classification is then obtained by combining such decisions. One relevant example is the *majority voting rule*, where the final decision is taken by looking at the class which received the *majority of votes*. More formally,

$$x \longleftarrow \arg\max_c \sum_{\ell=1}^{M} \Delta_{\ell c}(x) \tag{7}$$

where

$$\Delta_{\ell c}(x) = \begin{cases} 1 & \text{if } e_{\ell c}(x) = \max_j e_{\ell j}(x) \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

In other words $\Delta_{\ell c}(x)$ is 1 only if the classifier E_ℓ assigns x to the class c.

2) Fixed combination of matching scores: In this case, for a given class c, the matching scores $e_{\ell c}(x)$ of the different classifiers (with $1 \leq \ell \leq M$) are combined together, in order to return an unique matching score for the considered class c. The combination of these scores follows *fixed rules*, such as the sum or the product of them, the max or the min among them, the linear combination of them, and similar [12]. The final decision is finally taken by looking at these aggregated matching scores. For example, with the *Sum Rule*, a pattern x is classified with the following rule:

$$x \longleftarrow \arg\max_c \left[\sum_{\ell=1}^{M} e_{\ell c}(x) \right] \quad (9)$$

whereas with the *Prod Rule* we have

$$x \longleftarrow \arg\max_c \left[\prod_{\ell=1}^{M} e_{\ell c}(x) \right] \quad (10)$$

3) Trained Combiners: This represents a more advanced scheme [13], [14], in which the idea is to directly use the scores derived in the matrix $E(x)$ as *new features* for the pattern x: in this way a classifier is learned on the ouputs of other classifiers, following what is sometimes referred to as stacked combination [19]. In more detail, a pattern x is described with $vec(E(x))$, where the so-called $vec(\cdot)$ operator (vectorization) takes a matrix argument and returns a vector with the matrix elements stacked column by column. In the training phase, the vectorized $E(x_i)$ matrix is computed for all objects x_i of the training set, resulting in a novel training set, which is used to train a classifier f. In the testing phase, the testing object x is firstly encoded with $vec(E(x))$ and then classified using the classifier f.

III. The Weighted KNN Rule Revisited

In this section we propose an interpretation of the WKNN rule (and the KNN rule) from a combining classifier perspective. The main idea behind our interpretation is the following: in the (W)KNN the final decision on x is obtained by combining information provided by the K nearest neighbors $ne_K(x) = n_1, \cdots n_K$ of x. Therefore it seems reasonable to consider these K points as K different experts/classifiers, which provide information to be combined for reaching the final decision. Let us clarify our vision by firstly considering the KNN: we will show how to build the $E(x)$ matrix, and which combination rule should be used to get exactly the KNN rule.

As said before, we have K experts/classifiers, which we indicate as $E_{n_1}, \cdots E_{n_K}$, each one related to one specific neighbor n_ℓ. In the KNN case, the elements of the matrix $E^{KNN}(x)$ are defined, $\forall \ell \in \{1..K\}$ and $c \in \{1..C\}$, as:

$$e_{n_\ell c}(x) = \begin{cases} a & \text{if } y_{n_\ell} = c \\ 0 & \text{otherwise} \end{cases} \quad (11)$$

with a a fixed positive number[1] (it can also be 1). For example, if $K = 3$, $C = 4$, and $y_{n_1} = 1$, $y_{n_2} = 1$, and $y_{n_3} = 2$, the matrix $E^{KNN}(x)$ is

$$E^{KNN}(x) = \begin{bmatrix} a & 0 & 0 & 0 \\ a & 0 & 0 & 0 \\ 0 & a & 0 & 0 \end{bmatrix}$$

Given this formulation, if we apply the *majority voting rule* defined in Equation (7) we have to perform two steps: i) to take a decision for each classifier (each row), and this is done by taking the maximum over the row; ii) then to assign x to the class which received the majority of votes. Actually in this way we obtain exactly the K-nearest neighbor classifier: given the definition in Equation (11), every expert (neighbor) votes for the class corresponding to its label, and the final class is decided by looking at the most voted class, which is exactly the most frequent class in the neighborhood[2].

In the case of Weighted KNN, we define the elements of the matrix $E^{WKNN}(x)$, $\forall \ell \in \{1..K\}$ and $c \in \{1..C\}$, as:

$$e_{n_\ell c}(x) = \begin{cases} w_{n_\ell} & \text{if } y_{n_\ell} = c \\ 0 & \text{otherwise} \end{cases} \quad (12)$$

For example, for the problem introduced before ($K = 3$, $C = 4$, $y_{n_1} = 1$, $y_{n_2} = 1$, and $y_{n_3} = 2$)

$$E^{WKNN}(x) = \begin{bmatrix} w_{n_1} & 0 & 0 & 0 \\ w_{n_2} & 0 & 0 & 0 \\ 0 & w_{n_3} & 0 & 0 \end{bmatrix}$$

Given this definition, if we apply *Sum Rule* described in Equation (9) to $E^{WKNN}(x)$, we have to perform two steps: i) aggregate the scores for every class, and this is done by summing the values contained in each column; ii) assign x to the class for which this aggregated score is maximum. It is straightforward to note that this is exactly the decision rule proposed by the Weighted KNN rule described in Equation (5).

A. Normalization of $E(x)$

In many combination rules, before applying a combination scheme, the matching scores (confidences) $E_\ell(x)$ should be normalized, in order to get values which are comparable among the different classifiers (see [12], chap. 5) – this is especially true when using trained combiners [13], as those described in the previous section. In the following we provide some intuitions on what happens when using a common and established normalization scheme, the so-called *Soft-Max*

[1]Please note that $e_{n_\ell c}(x)$ can be defined in a more compact way using the Indicator Function $I_c(z)$ used in Equation (2). However, for clarity, here he presented this more verbose formulation.

[2]Please note that in this case we also get the KNN rule by applying the sum rule (since a is a costant).

normalization [15]. After this normalization the matching scores are in the range $[0-1]$; moreover, for every classifier, they sums to 1, so that they can be interpreted – with somehow an abuse of interpretation – as posterior probabilities – this is especially useful when trying to derive theoretical properties as in [16]. When applied to our case, each $e_{\ell c}(x)$ of $E(x)$ is transformed in $\hat{e}_{\ell c}(x)$ via the following formula:

$$\hat{e}_{\ell c}(x) = \frac{e^{e_{\ell c}(x)}}{\sum_{j=1}^{C} e^{e_{\ell j}(x)}} \tag{13}$$

With this normalization, $E^{KNN}(x)$ is transformed in $\hat{E}^{KNN}(x)$, where

$$\hat{e}_{n_\ell c}(x) = \begin{cases} e^a/R & \text{if } y_{n_\ell} = c \\ 1/R & \text{otherwise} \end{cases} \tag{14}$$

where

$$R = (C-1) + e^a \tag{15}$$

is the normalization factor present in the denominator of Equation (13). It is straightforward to observe that, given this normalized $\hat{E}^{KNN}(x)$, the KNN rule is still obtained by applying the majority voting rule to $\hat{E}^{KNN}(x)$. On the contrary, after this normalization, the Weighted KNN rule becomes equivalent to another fixed rule, namely the prod rule. Actually, $\hat{E}^{WKNN}(x)$ is defined as

$$\hat{e}_{n_\ell c}(x) = \begin{cases} e^{w_{n_\ell}}/R_\ell & \text{if } y_{n_\ell} = c \\ 1/R_\ell & \text{otherwise} \end{cases} \tag{16}$$

where R_ℓ is again the normalization factor of Equation (13), which in this case is different for different neighbors n_ℓ, and is defined as

$$R_\ell = (C-1) + e^{w_{n_\ell}} \tag{17}$$

If we consider the prod rule in Equation (10) applied to $\hat{E}^{WKNN}(x)$, we have

$$x \longleftarrow \arg\max_c \left[\prod_{\ell=1}^{M} \hat{e}_{\ell c}(x) \right] \tag{18}$$

Taking the $\log$ does not affect the argument of the max, therefore an equivalent rule is:

$$x \longleftarrow \arg\max_c \left[\log \prod_{\ell=1}^{M} \hat{e}_{\ell c}(x) \right] \tag{19}$$

which becomes

$$x \longleftarrow \arg\max_c \left[\sum_{\ell=1}^{M} \log \hat{e}_{\ell c}(x) \right] \tag{20}$$

$$= \arg\max_c \left[\sum_{\ell=1}^{M} [e_{\ell c}(x) - \log R_\ell] \right] \tag{21}$$

$$= \arg\max_c \left[\sum_{\ell=1}^{M} e_{\ell c}(x) - \sum_{\ell=1}^{M} \log R_\ell \right] \tag{22}$$

$$= \arg\max_c \left[\sum_{\ell=1}^{M} e_{\ell c}(x) \right] \tag{23}$$

where we dropped the last term because is equal among all classes. The resulting rule is equivalent to the Weighted KNN rule of Equation (5).

Summarizing, here we provided a revisitation of the KNN and the Weighted KNN rules from the Classifier Combining perspective: this opens the door to the possibility of using different (even complex) combination strategies. We will provide some evidence, in the experimental section, that using a trained combiner permits to improve the performances of both the KNN and the WKNN rule.

IV. Experimental Results

In this section, we provide some empirical evidence that the perspective introduced in this paper permits to exploit advanced combination techniques, such as those represented by trained combiners [13], [14]. In particular, in our empirical evaluation we compare three techniques:

1) **KNN**: this is the classic K-Nearest Neighbor rule. As we have shown in Section III, this corresponds to the *majority vote rule* applied to the E^{KNN} matrix defined in Equation (11) – as well as to the $\hat{E}^{WKNN}$ defined in Equation (16).
2) **WKNN (Dudani)**: this is the Original Weighted K-Nearest Neighbor rule described in [6] and presented in Section II-B. This corresponds to the *sum rule* applied to the E^{WKNN} matrix defined in Equation (12) – or to the *prod rule* applied to the $\hat{E}^{WKNN}$ defined in Equation (16).
3) **WKNN (TrainedComb)**: in this case we applied a trained combiner scheme: as explained in Section II-C, with this scheme every pattern is described with the vectorization of its corresponding matrix of scores, which is used as *feature vector* to represent it. In other words, all the objects of the problem are mapped in a novel feature space, where another classifier is used. Here we adopt the *decision template scheme* proposed in [12], which represents one of the first and most basic trained combiner. More in details, for every pattern x_i of the training set we compute the matrix $\hat{E}^{WKNN}(x_i)$, as defined in Equation (16); we used the normalized scores, as suggested in [13]. For every training point x_i, the corresponding neighborhood $ne_K(x_i)$ is determined without considering x_i (this can partially prevent the overtraining situation which may occur with trained combiners – for a discussion on these aspects see [13]). Given this novel feature space, the Nearest Mean Classifier [15] is used as classifier. In particular, for every class c, we compute the mean of the vectorized scores of the x_i belonging to class c: this averaged vectorized score then represents the *template* t_c of such class:

$$t_c = \underset{x_i s.t. y_{n_i} = c}{\text{mean}} vec(\hat{E}(x_i)) \tag{24}$$

Finally, the testing object x is classified by looking at the similarity between its $vec(\hat{E}(x))$ and the different

classes templates t_c, assigning it to the nearest template:

$$x \longleftarrow \arg\min_c d(vec(\hat{E}(x)), t_c) \quad (25)$$

where $d(\cdot,\cdot)$ is a distance between vectors. For more details interested readers can refer to Subsection 5.3.1 of [12].

The three techniques have been tested using 6 different classic datasets (from the UCI-ML repository), which characteristics are summarized in table I. All datasets have been normalized so that every feature has zero mean and unit variance. As distance to compute neighbors we used the classical

Dataset	Objects	Classes	Features
Sonar	208	2	60
Soybean2	136	4	35
Ionosphere	351	2	34
Wine	178	3	13
Breast	699	2	9
Bananas	100	2	2

TABLE I: Description of the datasets

Euclidean distance. Weighted KNN weights are computed using Equation (1) of Dudani's paper [6]:

$$w_{n_i} = \begin{cases} \dfrac{d(x,n_K) - d(x,n_i)}{d(x,n_K) - d(x,n_1)} & \text{if } d(x,n_K) \neq d(x,n_1) \\ 1 & \text{otherwise} \end{cases}$$

In this way weights are normalized between 0 and 1 (1 the weight of the nearest neighbor, 0 the weight of the farthest neighbor). We let K vary from 5 to 45 (step 2). Classification errors have been computed using the Averaged Holdout Cross validation protocol: the dataset is randomly split into two parts, one used for training and the other used for testing; the procedure has been repeated 30 times. Results are shown in Figure 2.

From the plots it can be observed that the Trained Combiner rule permits to improve the accuracy of both KNN and WKNN. This is more evident when the problem lives in a high dimensional space. For moderately dimensional space we cannot observe such a drastic improvement. One interesting observation derives by looking at the behavior for large K. Apparently, the Trained Combiner scheme does not suffer too much from a bad choice of K; this may be due to the fact that adding neighbors to the analysis simply corresponds to a different normalization of the feature space induced by $vec(\hat{E(x)})$. More in details, adding neighbors changes $d(x,n_K)$, which results in a shift (the numerator) and in a rescaling (the denominator) of the weight defined in Equation (26). Since we consider such weights as features in the novel space, adding neighbors simply result in a different scaling, which seems to not affect too much the final classification.

V. Conclusions

In this paper we revisited the Weighted K-Nearest Neighbor (and the K-Nearest Neighbor) scheme under a classifier combining perspective. Assuming this view, WKNN implements a fixed combiner rule, whereas KNN a majority voting rule. Then we provided some evidence that classification improvements are possible when using other classifier combining techniques, such as trained combiners.

Acknowledgements

This work was partially supported by the University of Verona through the CooperInt Program 2014 Edition. The authors are extremely grateful for all the guidance and inspiration that O. Ai Preti offered.

References

[1] E. Fix and J. L. Hodges Jr, "Discriminatory analysis-nonparametric discrimination: consistency properties," DTIC Document, Tech. Rep., 1951.

[2] ——, "Discriminatory analysis-nonparametric discrimination: Small sample performance," DTIC Document, Tech. Rep., 1952.

[3] T. Cover and P. Hart, "The nearest neighbor decision rule," *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 21–27, 1967.

[4] L. Devroye, L. Györfi, and G. Lugosi, *A probabilistic theory of pattern recognition*. Springer Science & Business Media, 2013, vol. 31.

[5] R. M. Royall, "A class of non-parametric estimates of a smooth regression function," Ph.D. dissertation, Dept. of Statistics, Stanford University., 1966.

[6] S. Dudani, "The distance-weighted k-nearest-neighbor rule," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. SMC-6, no. 4, pp. 325–327, 1976.

[7] T. Bailey and A. Jain, "A note on distance-weighted k-nearest neighbor rules," *IEEE Transactions on Systems, Man, and Cybernetics*, no. 4, pp. 311–313, 1978.

[8] J. E. MacLeod, A. Luk, and D. M. Titterington, "A re-examination of the distance-weighted k-nearest neighbor classification rule," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 17, no. 4, pp. 689–696, 1987.

[9] J. F. Banzhaf III, "Weighted voting doesn't work: A mathematical analysis," *Rutgers L. Rev.*, vol. 19, p. 317, 1964.

[10] J. Gou, L. Du, Y. Zhang, and T. Xiong, "A new distance-weighted k-nearest neighbor classifier," *Journal of Information & Computational Science*, vol. 9, no. 6, pp. 1429–1436, 2012.

[11] R. Samworth, "Optimal weighted nearest neighbor classifiers," *The Annals of Statistics*, vol. 40, no. 5, pp. 2733–2763, 2012.

[12] L. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*. Wiley, 2004.

[13] R. Duin, "The combining classifier: To train or not to train?" in *Proc. Int. Conf. on Pattern Recognition*, 2002, pp. 765–770.

[14] L. Kuncheva, J. Bezdek, and R. Duin, "Decision templates for multiple classifier fusion: an experimental comparison," *Pattern Recognition*, vol. 34, no. 2, p. 299314, 2001.

[15] R. Duda, P. Hart, and D. Stork, *Pattern Classification*, 2nd ed. John Wiley & Sons, 2001.

[16] J. Kittler, M. Hatef, R. Duin, and J. Matas, "On combining classifiers," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 3, pp. 226–239, 1998.

[17] A. Ross, K. Nandakumar, and A. Jain, *Handbook of Multibiometrics*. Springer, 2006.

[18] G. Fumera and F. Roli, "A theoretical and experimental analysis of linear combiners for multiple classifier systems," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 6, pp. 942–956, 2005.

[19] D. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992.

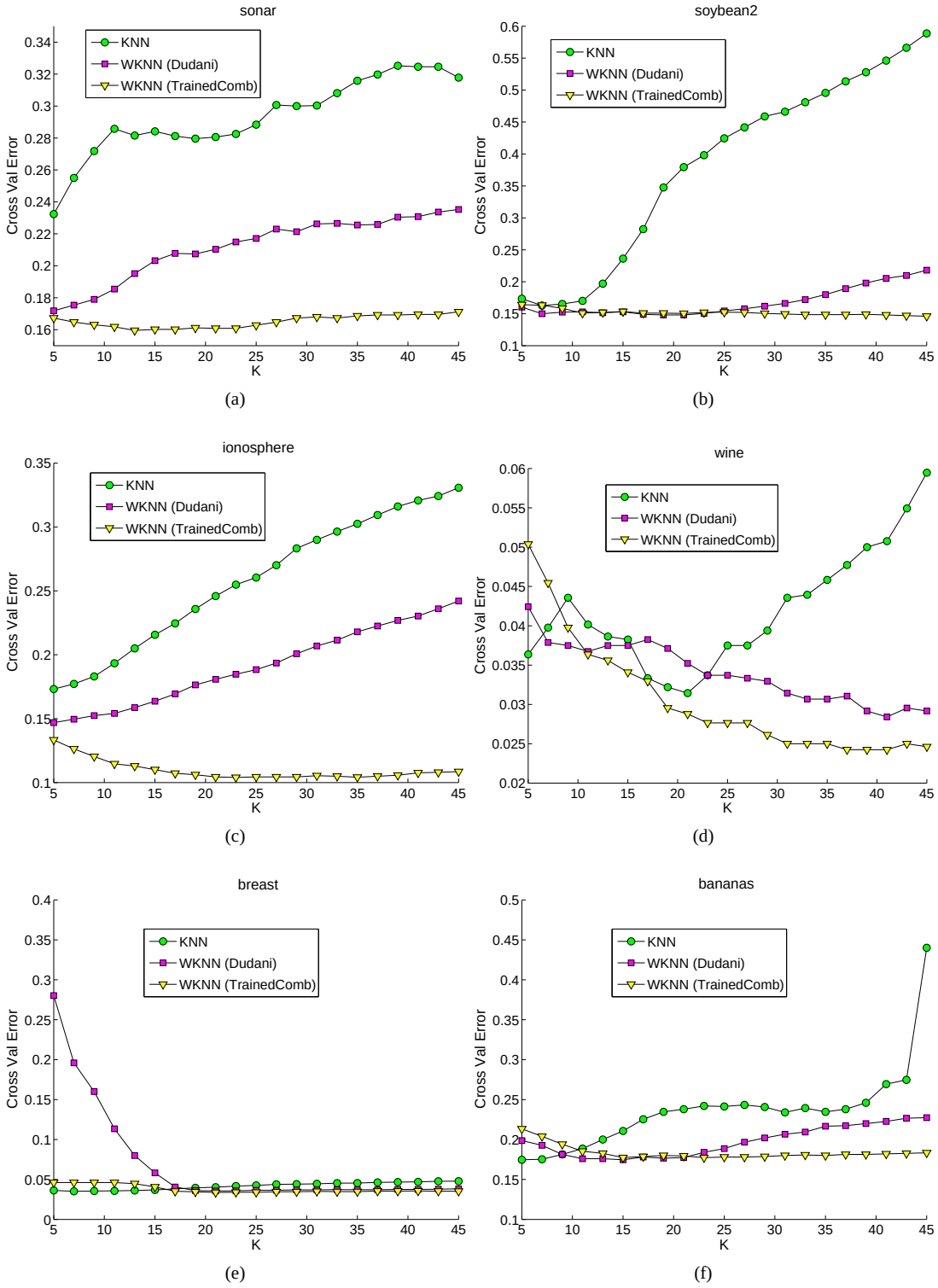

Fig. 2: Cross validation errors of the tested techniques for different datasets: (a) Sonar; (b) Soybean2; (c) Ionosphere; (d) Wine; (e) Breast; (f) Bananas.