Reject Options and Confidence Measures for kNN Classifiers*

Christoph Dalitz
Hochschule Niederrhein
Fachbereich Elektrotechnik und Informatik
Reinarzstr. 49, 47805 Krefeld, Germany

Abstract

This survey summarizes proposals made in the pattern recognition literature for detecting uncertain patterns that should rather be rejected than classified by a classifier. Beyond reviewing methods applicable to distance based nearest neighbor classifiers, this article describes an interface for computing confidences, storing them with classified images and querying this information, as it is implemented in the Gamera framework for document analysis and recognition. Based on this interface, a method for detecting broken, touching, and unknown characters, as well as noise, is proposed. The method is applied to two historic prints, showing that this method works well for detecting broken and touching characters, but less reliable for identifying glyphs representing noise.

1 Introduction

The classical problem of statistical pattern recognition is to assign an unknown pattern to one of M different classes $\{\omega_1, \ldots, \omega_M\}$. There are a number of textbooks entirely devoted to this problem, for instance [Webb2002]. Basically, it boils down to picking the class with the highest *posterior probability* $P(\omega_i|x)$, which denotes the probability that the pattern is of class ω_i when the feature vector x is observed. When all probabilities are known, this leads to Bayes' decision rule. When the probabilities are unknown, the decision must be based on training samples from known classes. A simple and often deployed decision rule is the k nearest neighbor (kNN) rule: decide for the class that is most frequent among the k nearest neighbors of the unknown pattern in feature space (see Fig. 1).

*Published in C. Dalitz (Ed.): "Document Image Analysis with the Gamera Framework." Schriftenreihe des Fachbereichs Elektrotechnik und Informatik, Hochschule Niederrhein, vol. 8, pp. 16-38, Shaker Verlag (2009)

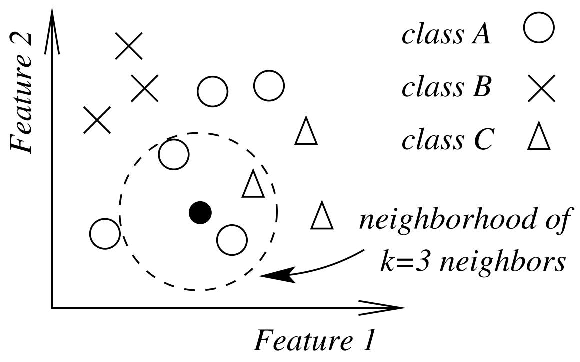

Figure 1: An example for the kNN rule for a two dimensional feature space and $k = 3$. The black test point is assigned to class A, because among its k nearest neighbors the majority is of class A.

In practice however, the assumption that all classes are known beforehand does not always hold. One case is the situation of a large number of classes (e.g. Chinese characters), of which some are rather rare. In such cases, training sets stemming from realistic data will generally be incomplete. Another typical case is the presence of missegmented samples, which can represent arbitrary fragments or combinations of proper class patterns. It is therefore desirable to have a *reject option*, i.e. an option to withhold a classifier decision.

It should be noted that a reject option cannot simply be integrated into the standard model merely by adding class $M + 1$ as class “unknown”, because the distribution of class “unknown” is also unknown and cannot be estimated from training data. Therefore the decision criteria for class “unknown” will be fundamentally different from the decision rule for all other classes. A natural approach is to extend the classifier to not only make a class decision, but to also provide some kind of *confidence measure* for the decision. This confidence can then be used in a second step to decide for acceptance or rejection.

Such a two step approach is particularly useful for modular recognition systems like the *Gamera framework* for document analysis and recognition [Gamera]. As this approach separates the classification and rejection steps, it allows for flexibility in the part of the system that receives the results from the classifier. Actually, the kNN classifier currently implemented in Gamera already returns a primitive “confidence” value (see section 3 for details), which is utilized by Gamera’s optional grouping algorithm to pick the best match from a set of fragment combination hypotheses [Droettboom2003]. It is one of the aims of the present paper to make suggestions for confidence measures and rejection rules suitable for modular recognition frameworks like *Gamera*.

This paper is organized as follows: section 2 introduces general concepts and a taxonomy of approaches. In section 3, proposals for establishing a “confidence measure” for classification decisions are reviewed. Sections 4 and 5 cover decision rules for rejection. Section 6 describes the confidence measure interface implemented in *Gamera*, and section 7 presents a case study how this interface can be deployed to detect broken or touching characters, as well as noise.

2 General Concepts

A first step in in the direction of adding alternatives to classification was made by Chow, who introduced a *reject option* within the context of Bayesian decision theory [Chow1970]. Chow still assumed that all classes and probabilities are known beforehand, and his motivation was to avoid classifications with a high probability of error. Because feature space regions with a low probability $P(\omega_i|x)$ contribute most to the error rate, Chow suggested to choose a threshold t and then reject a pattern when

$$P(\omega_i|x) < 1 - t \quad \text{for all classes } \omega_1, \ldots, \omega_M \tag{1}$$

Even though this rule reduces the error rate e, which becomes a function of the threshold t, it introduces a rejection rate $r(t)$. According to Chow, the above rejection rule is "optimal" in the way that for a given error rate, it minimizes the rejection rate. When the posterior probability is estimated from training data, Fumera et al. observed that the "optimality" of Chow's rule does not necessarily hold and showed that a class dependent threshold can improve the error-reject tradeoff [Fumera2000].

Chow made the assumption that all classes and their respective a priori and class conditional probabilities are known. In this situation, the only reason for rejection is ambiguity in the class decision. In the case of incomplete knowledge about all possible classes however, another possibility beside class ambiguity exists: that the pattern belongs to neither class.

Dubuisson and Masson therefore called Chow's rejection an *ambiguity reject* and introduced a new type of rejection, the *distance reject* [Debuisson1993]. "Distance" alludes to the decision criterion that a pattern is so far away from the training patterns in feature space that it is unlikely to be of any of the training classes. A "distance reject" can also be considered as the detection of a novel class. Markou and Singh therefore called it *novelty detection* and devoted a review to this topic [Markou2003].

Both for ambiguity and distance reject, the reject criterion is generally based on some kind of *confidence measure*. In Chow's rule (1), the confidence measure is simply the posterior probability of the most probable class ω_i. As this is not known in practical situations, it must be replaced by an estimator thereof or by some other measure. Actually, the confidence measures for ambiguity and distance reject need not be the same, and can even be used in different ways. Let us therefore first review suggested confidence measures in a separate section, and then consider the two reject cases.

3 Classifier Confidence

The most obvious attempt to measure classification confidence is the use of an estimator for the posterior probability $P(\omega_i|x)$. In order to apply Chow's rejection rule (1), it is actually not necessary to estimate the posterior probability directly, but it is sufficient to find an estimator for any monotonic mapping thereof. If such a mapping g exists for a confidence measure f_i for class ω_i, i.e.

$$f_i(x) = g(P(\omega_i|x)) \tag{2}$$

Lin et al. called f_i a *generalized confidence* for class ω_i [Lin1998]. In the case of the kNN rule, a direct estimator for the posterior probability is given by [Cover1967]

$$\hat{P}(\omega_i|x) = k_i/k \tag{3}$$

where k_i is the number of training samples from class ω_i among the k nearest neighbors. This estimator is only applicable in the large sample limit $k, N \to \infty$ (where N is the number of training samples), for which Cover and Hart have proved the (probabilistic) convergence of the estimator to the true probability [Cover1967]. They have also shown that in this limiting case, the distance of a test point to its nearest neighbor approaches zero with probability one. This is mirrored in the fact that the actual distances to the nearest neighbors play no role in the estimator (3).

For finite sample sizes however, all neighbors have different and finite distances from the test point and it is natural to utilize this distance information in the confidence computation. Atiya proposed not to use the actual distance values in the confidence estimation, but to weigh the neighbor classes according to their *neighborship rank*, i.e. the nearest neighbor gets more weight than the second and so on [Atiya2005]. He replaced the kNN posterior probability estimator (3) with

$$\hat{P}(\omega_i|x) \;=\; \sum_{j=i}^{k} v_j \cdot \delta(j, \omega_i) \tag{4}$$

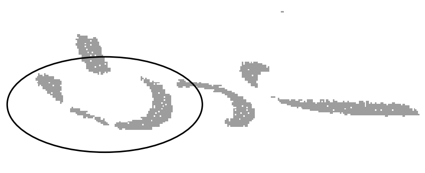

Figure 2: When the individual fragments of the marked broken symbol are classified, the estimator for the posterior probability might not be an appropriate confidence measure, because all class probabilities add to one, which ignores alternative segmentations.[1]

$$\text{with} \quad \delta(j,\omega_i) = \begin{cases} 1 \text{ when } \omega_i = \text{class of } j\text{-th neighbor} \\ 0 \text{ otherwise} \end{cases}$$

where $v_1 \geq v_2 \geq \ldots \geq v_k$ are weights for the neighbor ranks, which are to be determined by a maximum-likelihood method from the training data under the constraint $\sum v_i = 1$.

Building a reject option on an estimator for $P(\omega_i|x)$ like (3) or (4) makes the implicit assumption that all possible classes are known beforehand and that the training data contains samples of all classes, because of the constraint $\sum_{i=1}^{M} P(\omega_i|x) = 1$. This is insofar dissatisfying as it does not leave room for unknown classes as in the example in Fig. 2. An indicator of an unknown class could be that the distance to the nearest training sample is unusually large. It can therefore be reasonable to build a confidence measure directly on the distances rather than on an estimator for the posterior density.

Debuisson and Masson considered the average distance of the test point x to the k nearest neighbors $y_i, \ldots, y_j$ as a confidence measure [Debuisson1993]. Unlike a probability estimator, this has however the drawback that it is not normalized to a meaningful range, and even when all neighbors are of the same class and one of the distances is zero, the confidence is not at the boundary of its range. Therefore, other confidence measures normalized to the range $[0, 1]$ have been suggested, with the boundary value one meaning an almost sure decision. Droettboom does not give the confidence measure used in [Droettboom2003], but an inspection of his published Gamera source code shows that he used as a confidence for class ω_i

$$f_i(x) = \left(1 - \frac{\min\{d(x,y_j)\,|\,j = 1,\ldots k \text{ and } y_j \text{ belongs to } \omega_i\}}{\max\{d(x,y_j)\,|\,j = 1,\ldots,n\}}\right)^{10} \tag{5}$$

where the minimum in the numerator goes over the k nearest neighbors only, while the maximum in the denominator goes over all training samples. This confidence measure is normalized to $0 \leq f_i \leq 1$ with the desirable property that it is one when x coincides with one of the training points y_j. The division by the

[1]The image is a detail from the Greek chant book "Anastasimatarion" from 1847. For more information on the source and the notation see [Dalitz2008].

maximum distance over all training samples makes it however dependant from the global spread of the training data in feature space and not only from the local distances of the test point to its neighbors. On the other hand, this confidence measure is even applicable for $k = 1$, and also works in the presence of rare classes, of which only single items occur in the training data.

Arlandis et al. used the following confidence measure, which is built on the distances to the k nearest neighbors alone [Arlandis2002]:

$$f_i(x) = \left(\sum_{y_j \text{of class } \omega_i} \frac{1}{d(x, y_j)} \right) \Bigg/ \left(\sum_{j=1}^{k} \frac{1}{d(x, y_j)} \right) \tag{6}$$

This confidence measure can be considered as a distance weighted version of eq. (3). Actually it already appeared in Dudani's paper on distance weighted decision rules [Dudani1976] [Zavrel1997]. Dudani proposed it beside other distance weighting methods as a replacement for the kNN decision rule. The general form of a distance weighted confidence is

$$f_i(x) = \left(\sum_{y_j \text{of class } \omega_i} v_j \right) \Bigg/ \left(\sum_{j=1}^{k} v_j \right) \tag{7}$$

where v_j is a weight assigned to the distance $d(x, y_j)$. In addition to the inverse distance weighting (3), Dudani also proposed linear distance weighting by a linear interpolation between the nearest neighbor y_1 and the farest y_k among the k nearest neighbors:

$$v_j = \frac{d(x, y_k) - d(x, y_j)}{d(x, y_k) - d(x, y_1)} \tag{8}$$

Even though Dudani only proposed it as a confidence measure for deciding class membership, (7) can also be used in a reject criterion. Both (6) and (8) are normalized to $0 \leq f_i \leq 1$ and are equal to one when all k neighbors are of the same class. Moreover, (6) has the attractive property that it is one when the test point is identical to a training sample, because $\lim_{x \to y_j} f_i(x) = 1$. It should be noted however that neither confidence measure is meaningful for $k = 1$.

Roy and Madhvanath proposed to make the number k of neighbors considered in the confidence measure (6) dependent on the class ω_i, such that it is chosen higher for classes more frequent in the training set [Roy2008]. It should be noted however that this suggestion is in contradiction to Davies' observation that the class frequencies in the training set implicitly take care of the a priori class

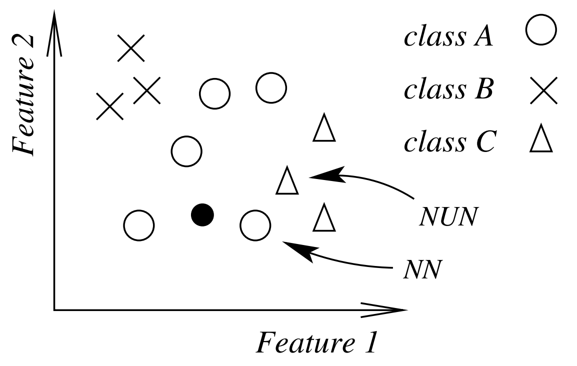

Figure 3: In this example, the nearest neighbor (NN) of the black test point is of class A. the nearest neighbor among the remaining classes is of class C and is called the *nearest unlike neighbor* (NUN).

probabilities [Davies1988]. Roy and Madhvanath's suggestion may therefore do good or harm, depending on how much the class frequencies in the training set are skewed compared to the a priori probabilities.

A different approach, which is independent of the chosen value for k, is Dasarathy's concept of the *nearest unlike neighbor* [Dasarathy1995], which is the nearest neighbor among the training samples not belonging to the closest class (see Fig. 3). Let us denote the distance of the test point x to the closest training sample from class ω_i with $d_i(x)$, that is

$$d_i(x) = \min\{d(x, y_j) \,|\, j = 1, \ldots N \text{ and } y_j \text{ belongs to } \omega_i\}$$

Then Dasarathy's confidence measure reads

$$f_i(x) = 1 - \frac{d_i(x)}{\min_{j \neq i}(d_j(x))} \tag{9}$$

Lin et al. argued in [Lin1998] that (9) is an estimator for a generalized confidence, but their argumentation made the implicit assumption that the expectation value E has the property $E(X/Y) = E(X)/E(Y)$. This identity does not hold in general, and it seems dubious that it holds in the special case $X = d_i(x)$ and $Y = \min_{k \neq i} d_k(x)$. Even though if might not be justifiable as a generalized confidence, the confidence measure (9) is nevertheless appealing, because it is one when x is a training sample and zero when the two nearest classes are evenly apart. Moreover, it is meaningful even for $k = 1$. For $k > 1$, it does however not take the class counts into account and is thus very susceptible to outliers in the training data. A generalization of the confidence measure (9) is the proposal by Chen and Ding to use the n-th nearest class rather than the second nearest class in the denominator to take care of the situation that the recognition problem has several very similar classes [Chen2003].

As a confidence measure that is also only based on the local neighborhood of the test point x, but does not take the class information into account, Tax and

Duin proposed [Tax1998]

$$f_i(x) = \frac{d(x, N(x))}{d(N(x), N(N(x)))} \tag{10}$$

where $N(x)$ denotes the nearest training point to x. Their reasoning for this measure was that the distance of an outlier to the closest training sample should be larger than the distance among adjacent training samples. In their experiments described in [Tax1998] however, this measure did not work for outlier detection. In the successive paper [Tax2000], they were able to detect outliers with this measure on a particular data set, but only in the special case of very few training samples per class.

Barbu et al. suggested to use as the denominator in (10) not the distance to the nearest, but to the *farthest* training sample [Barbu2007]. Let y_k denote the k-th nearest neighbor to the test point x and $FN(y)$ the training sample most far away from the point y, then their confidence measure reads

$$f_i(x) = \frac{1}{k} \sum_{i=1}^{k} \frac{d(x, x_j)}{d(x_j, FN(x_j))} \tag{11}$$

Barbu et al. only considered the problem of detecting outliers from a single class, but the definition (11) naturally makes sense for different classes, when for each class only the training samples belonging to this class are considered. (11) is smallest for points x near the mean value of the class distribution, so it measures somehow, how far a point deviates from the distribution mean. Therefore this measure can only be expected to be meaningful for unimodal distributions. In [Barbu2007], Barbu et al. do not present any theoretical or experimental properties of (11), so that further investigations are necessary to say anything about the usefulness of this confidence measure.

4 Ambiguity Rejection

In the first paper introducing a reject option for kNN classifiers, Hellman proposed to apply a threshold on the kNN probability estimator (3), but made no suggestion how to find an appropriate threshold [Hellman1970]. As (3) is limited to the large sample case, it seems more reasonable to use one of the other confidence measures (4), (7) or (9) instead, because they also take the actual distances into account. Whatever the used confidence measure may be, the problem of finding an appropriate reject threshold always remains.

Depending on the confidence measure and whether the reject or error rates should be limited, different approaches are possible:

- When the confidence measure is an estimator for the posterior probability, the threshold is a bound of the error rate and can thus directly be chosen.
- The threshold can experimentally be determined on the training data, such that a cross-correlation or bootstrap estimator of either the error rate or the reject rate is below a predefined value.
- A different criterion can be defined that is to be optimized by the threshold on the training data.

The first point is easily understood from the fact that $1 - P(\omega_i|x)$ is the probability of making an error when deciding for class ω_i. Therefore, Chow's rule (1) guarantees that the error rate will be smaller than the threshold t. Even when the confidence measure is not a direct estimator for $P(\omega_i|x)$, but a generalized confidence in the sense of (2), the mapping function $g(y)$ can be estimated from the training data by Lin et al.'s "adaptive confidence transform" [Lin1998]. Then Chow's rejection rule becomes $\max_i f_i(x) < g(1-t)$.

An alternative to an upper bound for the error rate $e(t)$ is an upper bound on the reject rate $r(t)$. Unlike for $e(t)$, there is no simple relationship between t and $r(t)$, so that the appropriate threshold t leading to a given reject rate $r(t)$ must be estimated form the training data. Chen and Ding proposed to experimentally find the lowest value for t on a second independent training set such that $r(t)$ is still below the predefined value [Chen2003]. In the absence of an abundance of training data, the same could be done with the leave-one-out method[2]. Formally, for a predefined rejection rate r_0 and N training samples, the threshold is then experimentally determined as ("$|M|$" stands for the number of elements in M):

$$t = \arg\max_t \{ |\{x_j | \max_i f_i(x_j) < 1 - t,\, j = 1, \ldots, N\}| < N \cdot r_0 \} \qquad (12)$$

The above two approaches to threshold selection try to limit either the error or the reject rate by some predefined value. Neither of these approaches yields an "optimal" threshold, a term obviously depending on the criterion used for "optimality". Once such a criterion is defined, the "optimal" threshold can be estimated by optimizing this criterion on the training set. The most simple "optimality criterion" is a linear combination of the error and reject rate, $v_e e(t) + v_r r(t)$,

[2]The resubstitution method is inappropriate for confidence measures that become one for data points from the training set, like (6).

where $v_{e/r}$ are weights representing the costs of an error and a reject. In other words, the “optimal threshold” is the value t that minimizes the *normalized risk*

$$\text{risk}(t) = e(t) + \beta \cdot r(t) \quad \text{with } \beta = \frac{\text{cost of a reject}}{\text{cost of an error}} \tag{13}$$

Chow has shown that in this situation, the optimal threshold for rule (1) can be found analytically with the use of a remarkable relationship between $r(t)$ and $e(t)$ [Chow1970]:

$$e(t) = \int_0^t r(s)\,ds - t \cdot r(t) \tag{14}$$

It follows that $e'(t) = -tr'(t)$, and the total cost is minimal when

$$0 = e'(t) + \beta r'(t) \iff t = \beta$$

In practical situations, the costs are often hard to quantify and thus to a certain degree arbitrary. Nevertheless, this result gives a direct interpretation of the optimal reject threshold for confidence measures that are estimators of the posterior class probability. For “generalized” confidences (in the sense of (2)), the normalized risk (13) can experimentally be minimized on the training data to find an optimal threshold for the generalized confidence measure.

Fumera et al. used as a criterion for the optimal class dependent thresholds $\mathbf{t} = (t_1, \ldots, t_M)$ the maximum of the *accuracy*, defined as [Fumera2000]

$$A(\mathbf{t}) = P(\text{correct}|\text{accept}) = \frac{1 - e(\mathbf{t}) - r(\mathbf{t})}{1 - r(\mathbf{t})} \tag{15}$$

under the additional constraint that the total reject rate is below a predefined value $r(\mathbf{t}) \leq r_{max}$. In the case of a single global threshold t, it can be concluded from de l’Hospital’s rule and (14) that the accuracy (15) approaches one as t goes to zero:

$$\begin{aligned} \lim_{t\to 0} A(t) &= \lim_{t\to 0} \left(\frac{1-r}{1-r} - \frac{e}{1-r} \right) = 1 - \lim_{t\to 0} \frac{e}{1-r} \\ &= 1 - \lim_{t\to 0} \frac{e'}{(1-r)'} = 1 - \lim_{t\to 0} \frac{-tr'}{-r'} = 1 \end{aligned}$$

Therefore, (15) is not a useful criterion for finding an optimal global reject threshold t.

5 Distance Rejection

The posterior probability estimators used for ambiguity rejection might coincidentally find some outliers that should also be distance rejected. In general however, this is a mere random side effect because the distribution of unknown classes is unknown and cannot be estimated from the training data. Therefore a threshold on the absolute distance of a test point from its neighboring training samples must be set, which is in the simplest case the mean distance to the k nearest neighbors [Debuisson1993]

$$g(x) = \frac{1}{k}\sum_{j=1}^{k} d(x, y_j) \tag{16}$$

In the case $k = 1$, this is simply the distance to the nearest neighbor. In the case $k = n$ (all training samples), this is for the Euclidean distance simply the distance between x and the mean of the training samples $\overline{x}$ plus an additive constant (which happens to be the empirical variance of the training data):

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n}(x - x_i)^2 &= \frac{1}{n}\sum_{i=1}^{n}\Big((x-\overline{x}) - (x_i - \overline{x})\Big)^2 \\ &= (x-\overline{x})^2 + \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2\end{aligned}$$

The measure (16) is not bounded and depends on the absolute magnitudes in feature space. Therefore the absolute value (16) must be compared to or normalized by some value learnt from the training data.

It should be noted that many other outlier detection schemes have been proposed, many of which try to estimate parameters of the class conditional densities in the feature space [Markou2003]. An appealing advantage of considering the distribution of (16) among the training data is not only that it makes no assumption on the distribution in feature space, but also that it reduces the problem of estimating a multi-variate distribution of feature values to estimating a univariate distribution of distance values. This makes empirical techniques for univariate distributions like box plots or the empirical CDF applicable.

Let $d_1, \ldots, d_N$ denote the values (16) for the M training points. The simplest approach is to base the distance reject threshold on $d_{max} = \max_i\{d_i\}$ as in [Guttormsson1999]. More detailed information can be obtained by measuring the empirical cumulative distribution function (CDF)

$$F(d) = P(g(x) \leq d) \tag{17}$$

of the $\{d_i\}$ [Arlandis2002]. The problem of the CDF is that it does not completely describe the distribution, because outliers are to be expected in the impossible range where $F(d) = 1$. Arlandis et al. therefore proposed to extrapolate $F(d)$ for values $d > d_{max}$ by a linear fit through the top 5% of the $\{d_i\}$. They then chose a threshold t_d of one or slightly above and rejected a test point when $F(g(x)) > t_d$.

In case the training set contains outlier data, the distance threshold for (16) can be set experimentally by demanding an appropriate reject rate on the training data.

6 Gamera's Interface to Confidence Computation

Starting with Gamera version 3.2.0, a new interface for computing, storing and querying confidence values has been implemented. Earlier versions of Gamera already used a builtin confidence value computed as in (5), but did not allow for additional confidence values. The builtin confidence value was used, e.g., in the grouping algorithm [Droettboom2003] and maybe in third party applications. Therefore, the new implementation was made in such a way that no legacy applications are broken, and the new features are optional new additions. To make the design decisions clear, we must first have a look at the old confidence interface in Gamera versions up to 3.1.x.

6.1 The legacy interface

In Gamera, classification state and result are stored in the image properties *classification_state* and *id_name*. The classification state can be one of *UNCLASSIFIED, AUTOMATIC, HEURISTIC,* or *MANUAL*. *id_name* does not only store the class decision, but all classes among the k nearest neighbors. This means that *id_name* is a list where each entry is a tuple (*confidence*, *classname*). The first list entry `id_name[0]` is the actual class decision made by the classifier.

For the confidence value, eq. (5) was used. As this confidence is based on the distance to the closest class representant alone, the class decision made by the knn classifier does not necessarily correspond to the class with the highest confidence value in *id_name*. The reasoning for the choice of (5) as a confidence measure was that it is defined for all values of k and also for all classes appearing among the k neighbors. Moreover, it has the advantage that it does not require an a priori statistical estimation process on the training data, but can directly be computed from the distances between test point and training data.

The interface to *Image.id_name* is given by the plugins in the category "Classification". The plugins with the prefix *classify_* receive obtain the list *id_name* as input. *classify_automaic* is hereby called by the knn classifier to store the classification result in the image. *get_main_id*, *get_confidence* and *match_id_name* only work on the first entry of *id_name*, i.e. the actual class decision.

6.2 The new interface (since version 3.2)

To avoid breaking existing applications, the image property *id_name* is left unchanged. This means that it still is a list of class names with a hard coded confidence (5) attached. An additional image property *confidence* is introduced that refers to the confidence of the principal class name (*id_name[0][1]*). To allow for different confidence measures, it is not a single value, but a mapping object ("dictionary" in python lingo) where the key is a unique identifier number (aliased to the uppercase constants described in Tbl. 1) for the confidence measure, and the value is the corresponding confidence value. This approach is quite flexible:

- it allows for multiple confidence values, making it possible to decide both about ambiguity and distance reject
- the user can decide which confidence measures to use and the information

Identifier Constant	*Description*
`CONFIDENCE_DEFAULT`	eq. (5)
`CONFIDENCE_KNNFRACTION`	eq. (3)
`CONFIDENCE_INVERSEWEIGHT`	eq. (6)
`CONFIDENCE_LINEARWEIGHT`	eq. (8)
`CONFIDENCE_NUN`	eq. (9)
`CONFIDENCE_NNDISTANCE`	distance to nearest neighbor
`CONFIDENCE_AVGDISTANCE`	eq. (16)

Table 1: Confidence types and their identifier constants in Gamera 3.2

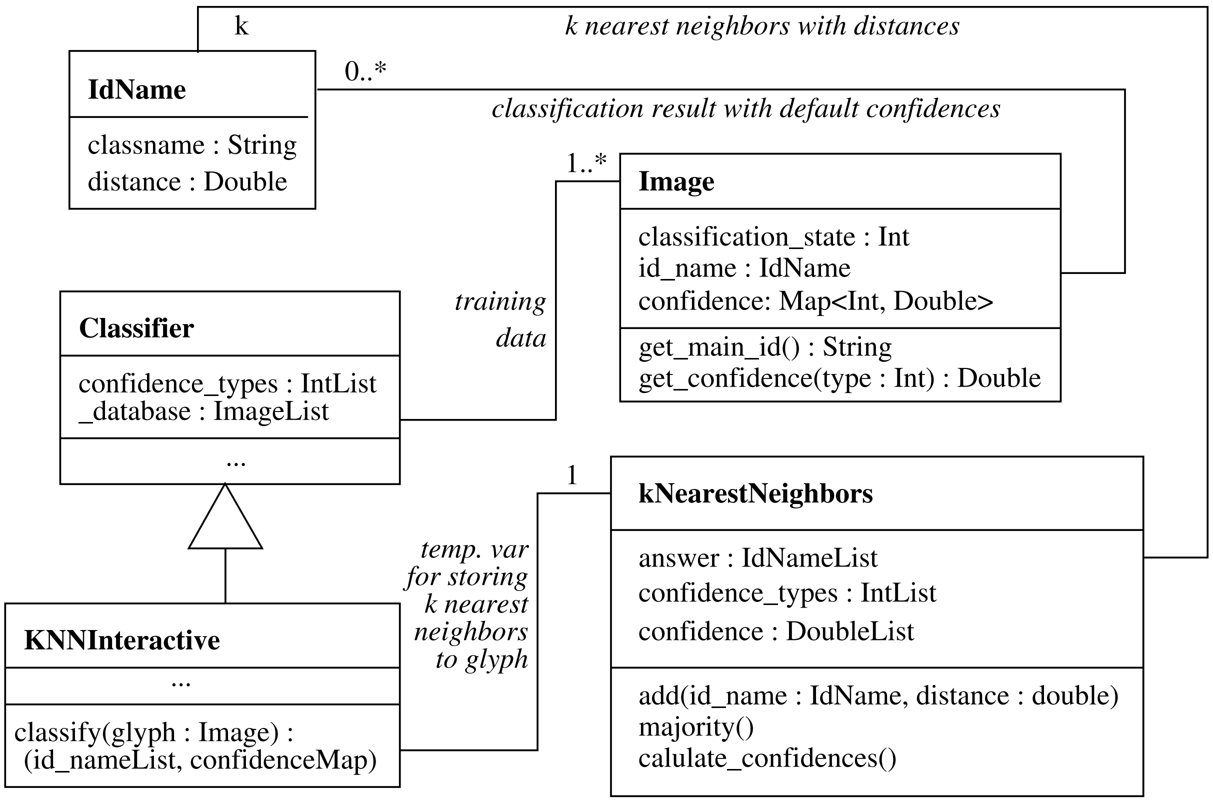
k
k nearest neighbors with distances
IdName
classname : String
distance : Double
0..*
classification result with default confidences
1..*
Image
classification_state : Int
id_name : IdName
confidence: Map<Int, Double>
get_main_id() : String
get_confidence(type : Int) : Double
training data
Classifier
confidence_types : IntList
_database : ImageList
...
1
kNearestNeighbors
answer : IdNameList
confidence_types : IntList
confidence : DoubleList
add(id_name : IdName, distance : double)
majority()
calulate_confidences()
temp. var for storing k nearest neighbors to glyph
KNNInteractive
...
classify(glyph : Image) : (id_nameList, confidenceMap)

Figure 4: Class diagram of the classes in the Gamera code that are involved in confidence calcuation.

about the used confidence measures is stored with *Image.confidence*

To extend the method *Image.get_confidence* in a backward compatible way, it now has an optional argument, the confidence measure identifier. When this argument is omitted, the old confidence value (5) stored in *id_name[0]* is returned.

An overview of the classes involved in the distance calculation in Gamera is shown in Fig. 4. The actual calculation is done in *kNearestNeighbor.calculate_confidences()*, based upon the distances to the k nearest neighbors and optionally additional information collected during the loop over the distances to the training prototypes. A more detailed overview of the knn classification process can be seen in Fig. 5.

The fact that the confidence calculation is performed in a class that does not have access to the training data, but only to the distances between the test glyph and the prototypes, restricts the choice for confidence measures to those that are to be computed from this information alone, like (5), (9) or (7). It also allows for distance rejection based on summary statistics for (16), because the confidence calculation can simply return the average distance, which can then be transformed to a meaningful “confidence” value in an independant second step

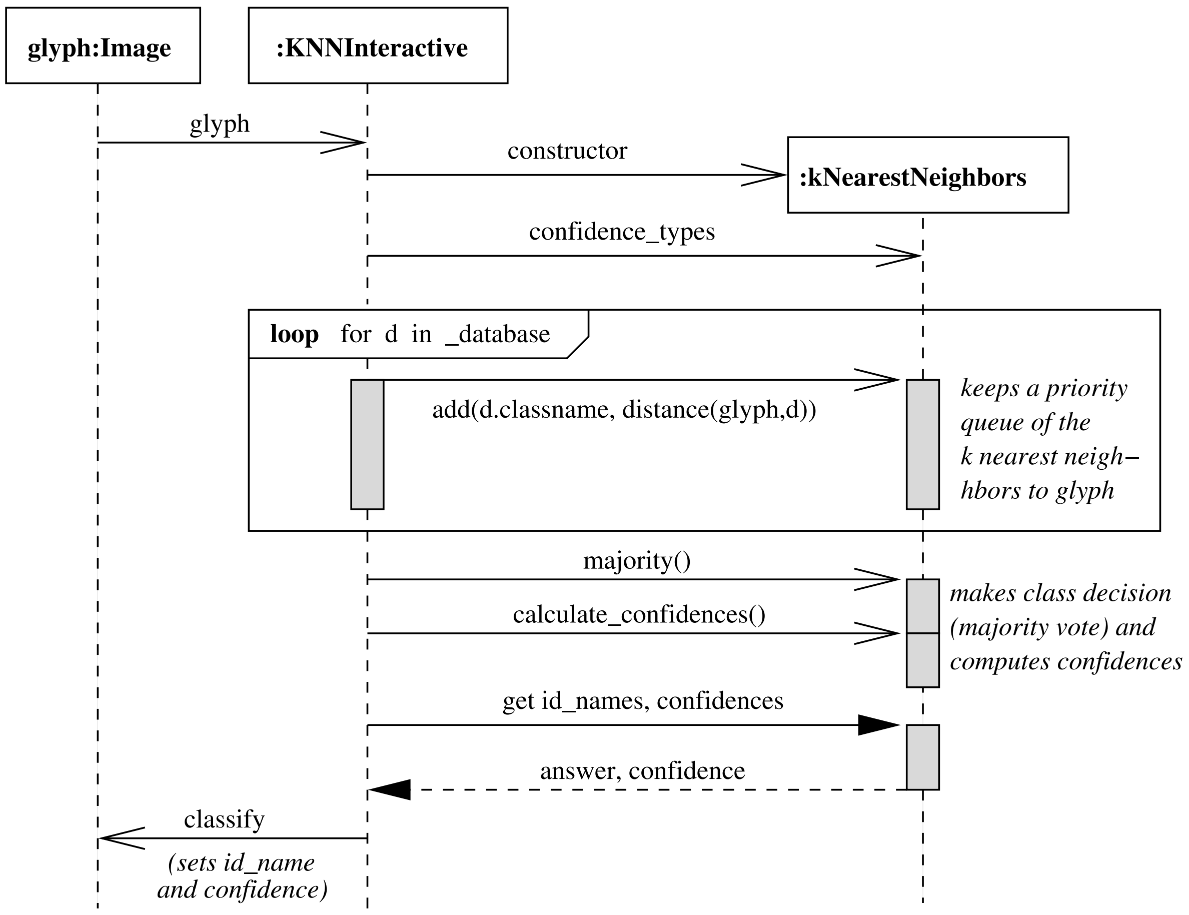
glyph:Image
:KNNInteractive
glyph
constructor
:kNearestNeighbors
confidence_types
loop for d in _database
add(d.classname, distance(glyph,d))
keeps a priority queue of the k nearest neigh–hbors to glyph
majority()
calculate_confidences()
makes class decision (majority vote) and computes confidences
get id_names, confidences
answer, confidence
classify
(sets id_name and confidence)

Figure 5: Sequence diagram showing the knn classification process of a test image *glyph* in the Gamera framework.

thereafter.

To allow for estimating a rejection threshold for (16) from the training data, a new method *knndistance_statistics(k)* has been added to kNN classifiers. This computes (16) for all training points with the leave-one-out method. The resulting list can then be further analyzed with the statistical methods from the SciPy module [SciPy], like *scipy.stats.gaussian_kde* for a gaussian kernel estimate of the probability density function. If several evaluations of the empirical distribution of these values are necessary, the class *EmpiricalCDF* from Listing 1 can be used. This code can also be used as a starting point for implementing extrapolations like those suggested in [Arlandis2002].

```python
class EmpiricalCDF(object):

    def __init__(self, data):
        self.n = len(data)
        self.data = [x for x in data] # copy data
        self.data.sort()

    def cdf(self, x):
        from bisect import bisect
        pos = bisect(self.data, x)
        if pos < 1:
            return 0.0
        elif pos >= self.n:
            return 1.0
        else:
            # linear interpolation between neighboring points
            a = self.data[pos-1];
            b = self.data[pos]
            return float(pos + float(x - a)/(b-a)) / self.n

    def invcdf(self, q):
        pos = int(q*self.n)
        if pos >= self.n:
            return self.data[self.n-1]
        elif pos <= 0:
            return self.data[0]
        else:
            # linear interpolation between neighboring points
            a = self.data[pos-1]
            b = self.data[pos]
            fa = float(pos)/self.n
            fb = float(pos+1)/self.n
            return (q-fa)*(b-a)/(fb-fa) + a
```

Listing 1: A Python implementation to compute the cumulative distribution function (17) and its inverse from a list of data points

7 Detecting Touching and Broken Characters - A Case Study

To demonstrate how the interface described in the preceding section can be deployed in a practical application, let us consider the problem of rejecting noise and distorted characters. In [Dalitz2008], a recognition system for byzantine chant notation is described. The printed books, on which the system was tested, contained not only considerable noise, but also many broken symbols, touching symbols, and also unknown symbols not present in the training data, like large (and mostly broken) intial characters or rare symbols not found on the pages used for training.

The system described in [Dalitz2008] dealt with noise by including sample noise in the training data, and with broken symbols by Droettboom's grouping algorithm [Droettboom2003]. No attempt was made to sort out touching and unknown symbols. The training and test data from that study provides a nice test case how well both noise and distorted characters can be detected by distance rejection. Here the term "distorted characters" is used for the set of broken, touching, or unknown symbols, and "noise" means symbols not representing any document content.

7.1 The method

The method is based on the mean distance $g(x)$ of a test symbol x to its k nearest neighbor training samples, as defined in (16). The rejection threshold for this value is chosen in such a way that the probability for rejecting a sane symbol ("false positive") is below a predefined value e_{fp}.

Under the assumption that the training data contains neither noise nor distorted symbols, then the cumulative distribution function $F(d)$ (see (17)) for the mean distance (16) among the sane symbols can be estimated with a leave-one-out method from the training data. The rejection threshold d_t leading to a false positive rate e_{fp} is then simply

$$e_{fp} \geq P(g(x) > d_t) = 1 - F(d_t) \quad \Longleftrightarrow \quad d_t \geq F^{-1}(1 - e_{fp}) \tag{18}$$

In the Gamera framework, d_t can thus be estimated with the classifier method *knndistance_statistics* and the class *EmpiricalCDF* from listing 1. The resulting Python code for rejecting noise and distorted symbols is shown in listing 2.

```
# initialize classifier
classifier=knn.kNNInteractive([], ['feature1',...], 0)
classifier.num_k = k
classifier.from_xml_filename(training_datafile)
classifier.confidence_types = [CONFIDENCE_AVGDISTANCE]

# compute rejection threshold d_t
stats = classifier.knndistance_statistics()
cdf = EmpiricalCDF([s[0] for s in stats])
d_t = cdf.invcdf(e_fp)

# classification with distance reject option
classifier.classify_list_automatic(glyphs)
for g in glyphs:
    if g.get_confidence(CONFIDENCE_AVGDISTANCE) > d_t:
        # do some reject operation
```

Listing 2: Python code using the library functions described in section 6 for distance rejection with a chosen false positive rate e_{fp}.

7.2 Experimental results

The rejection method was tested on 22 pages from the prints HA-1825 (Heirmologion Argon, Constantinople, 1825) and AM-1847 (Anastasimatarion, Constantinople, 1847). The training data was the same as in [Dalitz2008], but with all glyphs representing noise removed, so that the training data only contained sane symbols. The test images were segmented with connected component (CC) labeling, and each CC was classified on the training data. The sizes of training and test data are listed in Tbl. 2.

According to (18), choosing a rejection threshold based on a predefined value e_{fp} should have the effect that the fraction of sane symbols to be rejected is about e_{fp}. Fig. 6 shows that this holds indeed, though only approximately. Actually, in the present experimental setup, the observed false positive rate was slightly higher. This is a general tendency when the distance distribution func-

	training	*test glyphs*		
source	*glyphs*	*total*	*distorted*	*noise*
HA-1825	3411	5269	315	920
AM-1847	2051	6280	528	1851
total	5462	11549	843	2771

Table 2: Size of training and test data in the experimental study.

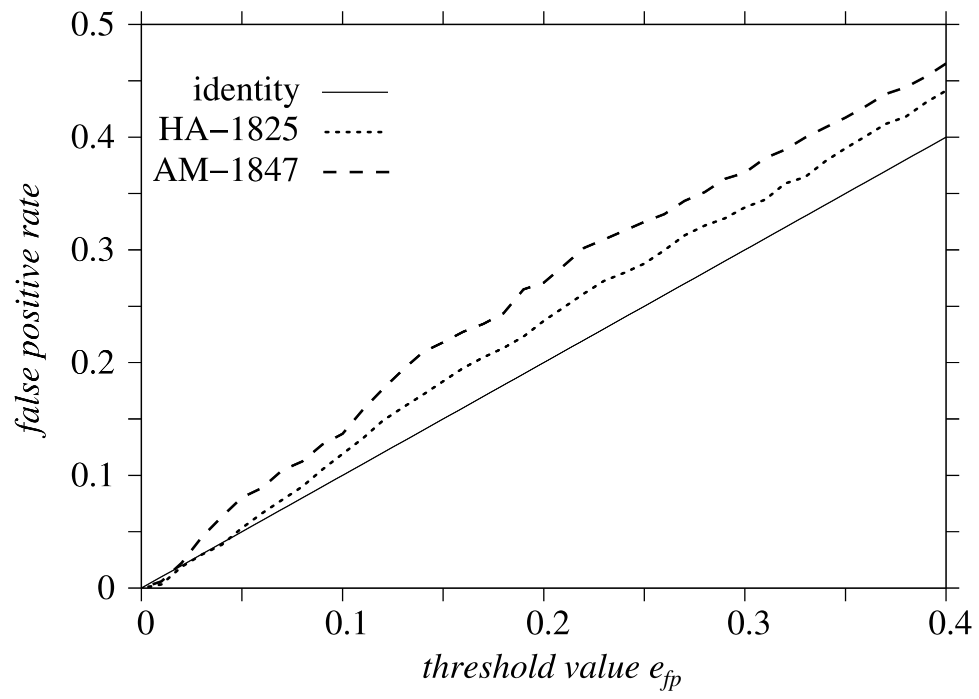

Figure 6: The actual false positive rates achieved with the threshold based on the CDF estimated from the training data.

tion is estimated from the training data. A theoretical argument that might come to mind is that although leave-one-out estimators have been found be nearly unbiased, they have a high variance for small sample sizes [Hand1986] [Isaksson2008]. This should however lead both to over- and underestimation. From a practical point of view, when the training prototypes have been manually selected, they tend to contain fewer poor (or “untypical”) prototypes than the real test data. This holds particularly for the present setup, because the training data purposely only contains “sane” symbols. This has the effect that the distance variation in the test data tends to be higher than in the training data, thus leading to more distance rejects, some of which also are false positives.

The performance of the distance rejection were very similar on both prints, so that in Fig. 7 the summarized error rates over both prints are shown. In the ROC curve [Webb2002], the observed false positive rate is drawn on the horizontal axis, and the rate of the correctly rejected glyphs (the “true positives”) on the vertical axis. It turned out that the performances for noise and for distorted characters were quite different. While distorted symbols could be detected with good accuracy even at low false positives rates, the same did not hold for noise. This is due to the fact that touching or broken characters tend to be rather different from the sane symbols, while noise can be of any shape and is often similar to dots or commas also present among the sane symbols.

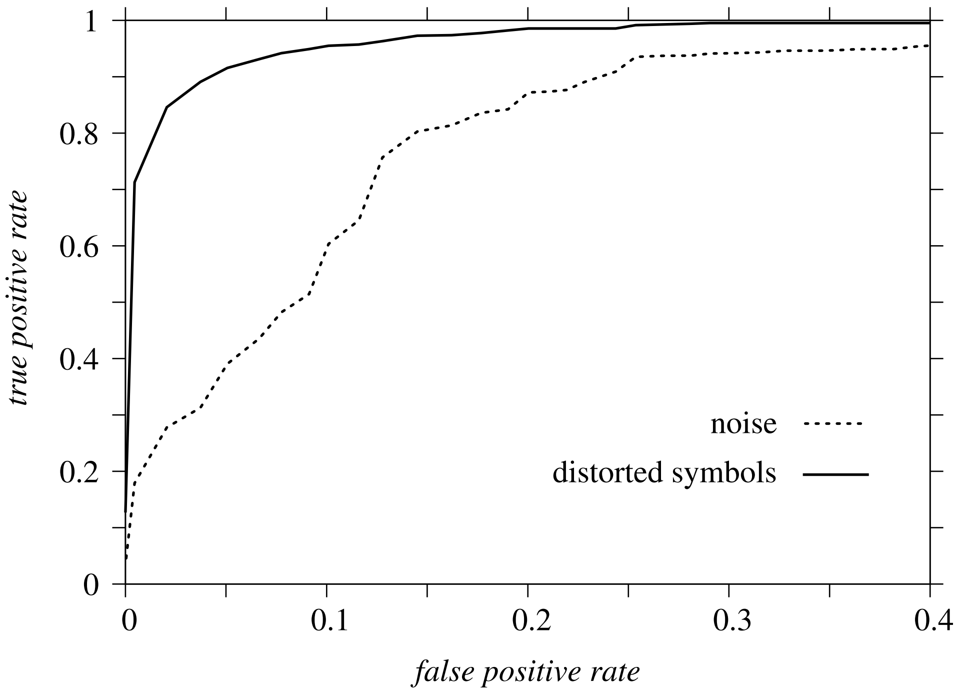

Figure 7: ROC curve for the detection of distorted symbols and noise in two historic prints with Byzantine neume notation.

8 Conclusion

Hopefully, this paper can serve as a good starting point for implementing reject options for recognition applications based on the Gamera framework. The experimental results presented in section 7 show that the methods now implemented in Gamera already work well for detecting distorted characters. Should the confidence measures built into Gamera turn out to be not sufficient for the application in question, the detailed description of the confidence calculation process in section 6 should make it easy to add custom confidence measures to the Gamera code.

References

[Arlandis2002] J. Arlandis, J.C. Perez-Cortes, J. Cano: *Rejection Strategies and Confidence Measures for a k-NN Classifier in an OCR Task.* 16th International Conference on Pattern Recognition, Proceedings, pp. 576-579 (2002)

[Atiya2005] A.F. Atiya: *Estimating the Posterior Probabilities Using the K-Nearest Neighbor Rule.* Neural Computation 17, pp. 731-740 (2005)

[Barbu2007] E. Barbu, C. Chatelain, S. Adam, P. Heroux, E. Trupin: *A simple one class classifier with rejection strategy: application to symbol classification.* Seventh IAPR International Workshop on Graphics Recognition (GREC) (2007)

[Chen2003] Z. Chen, X. Ding: *Rejection Algorithm for Mis-segmented Characters in Multilingual Document Recognition.* 7th Int. Conf. on Document Analysis and Recognition (ICDAR), Proceedings, pp. 746-749 (2003)

[Chow1970] C.K. Chow: *On optimum error and reject tradeoff.* IEEE Transactions on Information Theory 16, pp. 41-46 (1970)

[Cover1967] T.M. Cover, P.E. Hart: *Nearest neighbor pattern classification.* IEEE Transactions on Information Theory 13, pp. 21-27 (1967)

[Dasarathy1995] B.V. Dasarathy: *Nearest unlike neighbor (NUN): an aid to decision confidence estimation.* Optical Engineering 34, pp. 2785-2792 (1995)

[Dalitz2008] C. Dalitz, G.K. Michalakis, C. Pranzas: *Optical Recognition of Psaltic Byzantine Chant Notation.* International Journal of Document Analysis and Recognition 11, pp. 143-158 (2008)

[Davies1988] E.R. Davies: *Training sets and a priori probabilities with the nearest neighbour method of pattern recognition.* Pattern Recognition Letters 8, pp. 11-13 (1988)

[Debuisson1993] B. Dubuisson, M. Masson: *A Statistical Decision Rule With Incomplete Knowledge About Classes.* Pattern Recognition 26, pp. 155-165 (2002)

[Droettboom2003] M. Droettboom: *Correcting broken characters in the recognition of historical printed documents.* Joint Conference on Digital Libraries (JCDL'03), pp. 364-366 (2003)

[Dudani1976] S.A. Dudani: *The Distance-Weighted k-Nearest Neighbor Rule.* IEEE Transactions on Systems, Man, and Cybernetics 6, pp. 325-327 (1976)

[Fumera2000] G. Fumera, F. Roli, G. Giacinto: *Multiple Reject Thresholds for Improving Classification Reliability.* Proceedings of the Joint IAPR

Int. Workshop SSPR&SPR 2000, Springer, Lecture Notes in Computer Science 1876, pp. 863-871 (2000)

[Gamera] M. Droettboom, C. Dalitz: *The Gamera Project.* `http://gamera.informatik.hsnr.de/` (2008-2009)

[Guttormsson1999] S.E. Guttormsson, R.J. Marks II, M.A. El-Sharkawi, I. Kerszenbaum: *Elliptical Novelty Grouping for On-Line Short-Turn Detection of Excited Running Rotors.* IEEE Transactions on Energy Conversion 14, pp. 16-22 (1999)

[Hand1986] D.J. Hand: *Recent advances in error rate estimation.* Pattern Recogniiton Letters 4, pp. 335-246 (1986)

[Hellman1970] M.E. Hellman: *The Nearest Neighbor Classification Rule with a Reject Option.* IEEE Transactions on Systems, Science and Cybernetics 6, pp. 179-185 (1970)

[Isaksson2008] A. Isaksson, M. Wallman, H. Göransson, M.G. Gustafsson: *Cross-validation and bootstrapping are unreliable in small sample classification.* Pattern Recognition Letters 29, pp. 1960-1965 (2008)

[Lin1998] X. Lin, X. Ding, M. Cheng, R. Zhang, Y. Wu: *Adaptive confidence transform based classifier combination for Chinese character recognition.* Pattern Recognition Letters 19, pp. 975-988 (1998)

[Markou2003] M. Markou, S. Singh: *Novelty Detection: A Review. Part 1: Statistical Approaches.* Signal Processing 83, pp. 2481-2497 (2003)

[Roy2008] V. Roy, S. Madhvanath: *A Skew-tolerant Strategy and Confidence Measure for k-NN Classification of Online Handwritten Characters.* HP Labs, Technical Report HPL-2008-52 (2008)

[SciPy] E. Jones, T. Oliphant, P. Peterson et al.: *SciPy: Open Source Scientific Tools for Python.* `http://www.scipy.org/` (2001-2009)

[Tax1998] D.M.J. Tax, R.P.W. Duin: *Outlier Detection Using Classifier Instability.* Proceedings of the Joint IAPR Int. Workshop 1998, Springer, Lecture Notes In Computer Science 1451, pp. 593-601 (1998)

[Tax2000] D.M.J. Tax, R.P.W. Duin: *Data Description in Subspaces.* Proceedings of the 15th International Conference on Pattern Recognition, pp. 672-675 (2000)

[Webb2002] A. Webb: *Statistical Pattern Recognition.* John Wiley & Sons, second edition (2002)

[Zavrel1997] J. Zavrel: *An Empirical Re-Examination of Weighted Voting for k-NN.* Proceedings of the 7th Belgian-Dutch Conference on Machine Learning, pp. 139-148 (1997)