k-Nearest neighbour classifiers

Article *in* ACM Computing Surveys · April 2007

DOI: 10.1145/3459665

2 authors:

Padraig Cunningham
University College Dublin
361 PUBLICATIONS **11,846** CITATIONS
SEE PROFILE

Sarah Jane Delany
Technological University Dublin - City Campus
90 PUBLICATIONS **2,776** CITATIONS
SEE PROFILE

k-Nearest Neighbour Classifiers

Pádraig Cunningham[1] and Sarah Jane Delany[2]

[1] University College Dublin `Padraig.Cunningham@ucd.ie`
[2] Dublin Institute of Technology `Sarahjane.Delany@comp.dit.ie`

Technical Report UCD-CSI-2007-4
March 27, 2007

Abstract. Perhaps the most straightforward classifier in the arsenal or machine learning techniques is the Nearest Neighbour Classifier – classification is achieved by identifying the nearest neighbours to a query example and using those neighbours to determine the class of the query. This approach to classification is of particular importance today because issues of poor run-time performance is not such a problem these days with the computational power that is available. This paper presents an overview of techniques for Nearest Neighbour classification focusing on; mechanisms for assessing similarity (distance), computational issues in identifying nearest neighbours and mechanisms for reducing the dimension of the data.

1 Introduction

The intuition underlying Nearest Neighbour Classification is quite straightforward, examples are classified based on the class of their nearest neighbours. It is often useful to take more than one neighbour into account so the technique is more commonly referred to as k-Nearest Neighbour (k-NN) Classification where k nearest neighbours are used in determining the class. Since the training examples are needed at run-time, i.e. they need to be in memory at run-time, it is sometimes also called Memory-Based Classification. Because induction is delayed to run time, it is considered a *Lazy Learning* technique. Because classification is based directly on the training examples it is also called Example-Based Classification or Case-Based Classification.

The basic idea is as shown in Figure 1 which depicts a 3-Nearest Neighbour Classifier on a two-class problem in a two-dimensional feature space. In this example the decision for q_1 is straightforward – all three of its nearest neighbours are of class O so it is classified as an O. The situation for q_2 is a bit more complicated at it has two neighbours of class X and one of class O. This can be resolved by simple majority voting or by distance weighted voting (see below).

So $k-$NN classification has two stages; the first is the determination of the nearest neighbours and the second is the determination of the class using those neighbours.

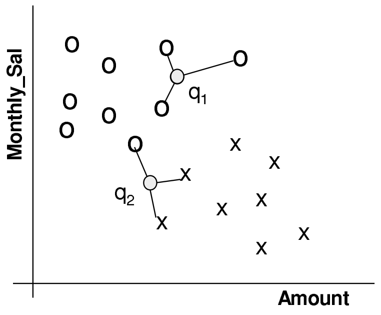

Fig. 1. A simple example of 3-Nearest Neighbour Classification

Let us assume that we have a training dataset D made up of $(\mathbf{x}_i)_{i\in[1,|D|]}$ training samples. The examples are described by a set of features F and any numeric features have been normalised to the range [0,1]. Each training example is labelled with a class label $y_j \in Y$. Our objective is to classify an unknown example $\mathbf{q}$. For each $\mathbf{x}_i \in D$ we can calculate the distance between $\mathbf{q}$ and $\mathbf{x}_i$ as follows:

$$d(\mathbf{q},\mathbf{x}_i) = \sum_{f\in F} w_f \delta(\mathbf{q}_f, \mathbf{x}_{if}) \tag{1}$$

There are a large range of possibilities for this distance metric; a basic version for continuous and discrete attributes would be:

$$\delta(\mathbf{q}_f, \mathbf{x}_{if}) = \begin{cases} 0 & f \text{ discrete and } \mathbf{q}_f = \mathbf{x}_{if} \\ 1 & f \text{ discrete and } \mathbf{q}_f \neq \mathbf{x}_{if} \\ |\mathbf{q}_f - \mathbf{x}_{if}| & f \text{ continuous} \end{cases} \tag{2}$$

The k nearest neighbours are selected based on this distance metric. Then there are a variety of ways in which the k nearest neighbours can be used to determine the class of $\mathbf{q}$. The most straightforward approach is to assign the majority class among the nearest neighbours to the query.

It will often make sense to assign more weight to the nearer neighbours in deciding the class of the query. A fairly general technique to achieve this is distance weighted voting where the neighbours get to vote on the class of the query case with votes weighted by the inverse of their distance to the query.

$$Vote(y_j) = \sum_{c=1}^{k} \frac{1}{d(\mathbf{q},\mathbf{x}_c)^n} 1(y_j, y_c) \tag{3}$$

Thus the vote assigned to class y_j by neighbour $\mathbf{x}_c$ is 1 divided by the distance to that neighbour, i.e. $1(y_j, y_c)$ returns 1 if the class labels match and 0 otherwise. In equation 3 n would normally be 1 but values greater than 1 can be used to further reduce the influence of more distant neighbours.

Another approach to voting is based on Shepard's work [25] and uses an exponential function rather than inverse distance, i.e:

$$Vote(y_j) = \sum_{c=1}^{k} e^{-\frac{d(\mathbf{q},\mathbf{x}_c)}{h}} 1(y_j, y_c) \tag{4}$$

In this paper we consider three important issues that arise with the use of k-NN classifiers. In the next section we look at the core issue of similarity and distance measures and explore some *exotic* (dis)similarity measures to illustrate the generality of the k-NN idea. In section 3 we look at computational complexity issues and review some speed-up techniques for k-NN. In section 4 we look at dimension reduction – both feature selection and sample selection. Dimension reduction is of particular importance with k-NN as it has a big impact on computational performance and accuracy. The paper concludes with a summary of the advantages and disadvantages of k-NN.

2 Similarity and Distance Metrics

While the terms *similarity metric* and *distance metric* are often used colloquially to refer to any measure of affinity between two objects, the term *metric* has a formal meaning in mathematics. A metric must conform to the following four criteria (where $d(x, y)$ refers to the distance between two objects x and y):

1. $d(x, y) \geq 0$; non-negativity
2. $d(x, y) = 0$ only if $x = y$; identity
3. $d(x, y) = d(y, x)$; symmetry
4. $d(x, z) \geq d(x, y) + d(y, z)$; triangle inequality

It is possible to build a k-NN classifier that incorporates an affinity measure that is not a proper metric, however there are some performance optimisations to the basic k-NN algorithm that require the use of a proper metric [24, 2]. In brief, these techniques can identify the nearest neighbour of an object without comparing that object to every other object but the affinity measure must be a metric, in particular it must satisfy the triangle inequality.

The basic distance metric described in equations 1 and 2 is a special case of the Minkowski Distance metric – in fact it is the 1-norm (L_1)Minkowski distance. The general formula for the Minkowski distance is

$$MD_p(\mathbf{q}, \mathbf{x}_i) = \left(\sum_{f \in F} |\mathbf{q}_f - \mathbf{x}_{if}|^p \right)^{\frac{1}{p}} \tag{5}$$

The L_1 Minkowski distance is the Manhattan distance and the L_2 distance is the Euclidean distance. It is unusual but not unheard of to use p values greater than 2. Larger values of p have the effect of giving greater weight to the attributes on which the objects differ most. To illustrate this we can consider three points in 2D space; $A = (1, 1), B = (5, 1)$ and $C = (4, 4)$. Since A and B differ on one attribute only the $MD_p(A,B)$ is 4 for all p, whereas $MD_p(A,C)$ is 6, 4.24 and 3.78 for p values of 1, 2 and 3 respectively. So C becomes the nearer neighbour to A for p values of 3 and greater.

The other important Minkowski distance is the L_∞ or Chebyshev distance.

$$MD_\infty(\mathbf{q},\mathbf{x}_i) = \max_{f\in F} |\mathbf{q}_f - \mathbf{x}_{if}|$$

This is simply the distance in the dimension in which the two examples are most different; it is sometimes referred to as the chessboard distance as it is the number of moves it takes a chess king to reach any square on the board.

In the remainder of this section we will review a selection of other metrics distances that are important in multimedia analysis.

2.1 Other Distances Metrics for Multimedia Data

The Minkowski distance defined in (5) is a very general metric that can be used in a k-NN classifier for any data that is represented as a feature vector. When working with image data a convenient representation for the purpose of calculating distances is a colour histogram. An image can be considered as a grey-scale histogram H of N levels or bins where h_i is the number of pixels that fall into the interval represented by bin i (this vector h is the feature vector). The Minkowski distance formula (5) can be used to compare two images described as histograms. L_1, L_2 and less often L_∞ norms are used.

Other popular measures for comparing histograms are the Kullback-Leibler divergence (6) [12] and the χ^2 statistic (7) [23].

$$d_{KL}(H,K) = \sum_{i=1}^{N} h_i \log\left(\frac{h_i}{k_i}\right) \tag{6}$$

$$d_{\chi^2}(H,K) = \sum_{i=1}^{N} \frac{h_i - m_i}{h_i} \tag{7}$$

where H and K are two histograms, h and k are the corresponding vectors of bin values and $m_i = \frac{h_i+k_i}{2}$.

While these measures have sound theoretical support in information theory and in statistics they have some significant drawbacks. The first drawback is that they are not metrics in that they do not satisfy the symmetry requirement. However, this problem can easily be overcome by defining a modified distance between x and y that is in some way an average of $d(x,y)$ and $d(y,x)$ – see [23] for the Jeffrey divergence which is a symmetric version of the Kullback-Leibler divergence.

A more significant drawback is that these measures are prone to errors due to bin boundaries. The distance between an image and a slightly darker version of itself can be great if pixels fall into an adjacent bin as there is no consideration of adjacency of bins in these measures.

Earth Mover Distance The Earth Mover Distance (EMD) is a distance measure that overcomes many of these problems that arise from the arbitrariness of binning. As the name implies, the distance is based on the notion of the amount of effort required to convert one image to another based on the analogy of transporting *mass* from one distribution to another. If we think of two images as distributions and view one distribution as a mass of earth in space and the other distribution as a hole (or set of holes) in the same space then the EMD is the minimum amount of work involved in filling the holes with the earth.

In their analysis of the EMD Rubner et al. argue that a measure based on the notion of a *signature* is better than one based on a histogram. A signature $\{\mathbf{s}_j = \mathbf{m}_j, w_{\mathbf{m}_j}\}$ is a set of j clusters where $\mathbf{m}_j$ is a vector describing the mode of cluster j and $w_{\mathbf{m}_j}$ is the fraction of pixels falling into that cluster. Thus a signature is a generalisation of the notion of a histogram where boundaries and the number of partitions are not set in advance; instead j should be 'appropriate' to the complexity of the image [23].

The example in Figure 2 illustrates this idea. We can think of the clustering as a quantization of the image in some colour space so that the image is represented by a set of cluster modes and their weights. In the figure the source image is represented in a 2D space as two points of weights 0.6 and 0.4; the target image is represented by three points with weights 0.5, 0.3 and 0.2. In this example the EMD is calculated to be the sum of the amounts moved (0.2, 0.2, 0.1 and 0.5) multiplied by the distances they are moved. Calculating the EMD involves discovering an assignment that minimizes this amount.

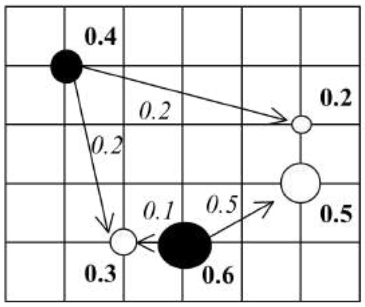

Fig. 2. An example of the EMD between two 2D signatures with two points (clusters) in one signature and three in the other (based on example in [22]).

For two images described by signatures $S = \{\mathbf{m}_j, w_{\mathbf{m}_j}\}_{j=1}^{n}$ and $Q = \{\mathbf{p}_k, w_{\mathbf{p}_k}\}_{k=1}^{r}$ we are interested in the work required to transfer from one to the other for a given flow pattern $\mathbf{F}$:

$$WORK(S, Q, \mathbf{F}) = \sum_{j=1}^{n} \sum_{k=1}^{r} d_{jk} f_{jk} \qquad (8)$$

where d_{jk} is the distance between clusters $\mathbf{m}_j$ and $\mathbf{p}_k$ and f_{jk} is the flow between $\mathbf{m}_j$ and $\mathbf{p}_k$ that minimises overall cost. An example of this in a 2D colour space is shown in Figure **??**. Once the transportation problem of identifying the flow that minimises effort is solved (using dynamic programming) the EMD is defined to be:

$$EMD(S,Q) = \frac{\sum_{j=1}^{n}\sum_{k=1}^{r} d_{jk} f_{jk}}{\sum_{j=1}^{n}\sum_{k=1}^{r} f_{jk}} \tag{9}$$

Efficient algorithms for the EMD are described in [23] however this measure is expensive to compute with cost increasing more than linearly with the number of clusters. Nevertheless it is an effective measure for capturing similarity between images.

Compression-Based Dissimilarity In recent years the idea of basing a similarity metric on compression has received a lot of attention. [15, 10]. Indeed Li et al. [15] refer to this as *The* similarity metric. The basic idea is quite straightforward; if two documents are very similar then the compressed size of the two documents concatenated together will not be much greater than the compressed size of a single document. This will not be true for two documents that are very different. Slightly more formally, the difference between two documents A and B is related to the compressed size of document B when compressed using the codebook produced when compressing document A.

The theoretical basis of this metric is in the field of Kolmogorov complexity, specifically in conditional Kolmogorov complexity.

$$d_{Kv}(x,y) = \frac{Kv(x|y) + Kv(y|x)}{Kv(xy)} \tag{10}$$

where $Kv(x|y)$ is the length of the shortest program that computes x when y is given as an auxiliary input to the program and $Kv(xy)$ is the length of the shortest program that outputs y concatenated to x. While this is an abstract idea it can be approximated using compression

$$d_C(x,y) = \frac{C(x|y) + C(y|x)}{C(xy)} \tag{11}$$

$C(x)$ is the size of data x after compression, and $C(x|y)$ is the size of x after compressing it with the compression model built for y. If we assume that $Kv(x|y) \approx Kv(xy) - Kv(y)$ then we can define a normalised compression distance

$$d_{NC}(x,y) = \frac{C(xy) - \min(C(x), C(y))}{\max(C(x), C(y))} \tag{12}$$

It is important that $C(.)$ should be an appropriate compression metric for the data. Delany and Bridge [6] show that compression using Lempel-Ziv (GZip) is effective for text. They show that this compression based metric is more accurate in k-NN classification than distance based metrics on a bag-of-words representation of the text.

3 Computational Complexity

Computationally expensive metrics such as the Earth-Mover's Distance and compression based (dis)similarity metrics focus attention on the computational issues associated with k-NN classifiers. Basic k-NN classifiers that use a simple Minkowski distance will have a time behaviour that is $O(|D||F|)$ where D is the training set and F is the set of features that describe the data, i.e. the distance metric is linear in the number of features and the comparison process increases linearly with the amount of data. The computational complexity of the EMD and compression metrics is more difficult to characterise but a k-NN classifier that incorporates an EMD metric is likely to be $O(|D|n^3 \log n)$ where n is the number of clusters [23].

For these reasons there has been considerable research on editing down the training data and on reducing the number of features used to describe the data (see section 4). There has also been considerable research on alternatives to the exhaustive search strategy that is used in the standard k-NN algorithm. Here is a summary of four of the strategies for speeding up nearest-neighbour retrieval:

– **Case-Retrieval Nets:** k-NN retrieval is widely used in Case-Based Reasoning and Case-Retrieval Nets (CRNs) are perhaps the most popular technique for speeding up the retrieval process. Again, the cases are pre-processed, but this time to form a network structure that is used at retrieval time. The retrieval process is done by *spreading activation* in this network structure. CRNs can be configured to return exactly the same cases as k-NN [14, 13].
– **Footprint-Based Retrieval:** As with all strategies for speeding up nearest-neighbour retrieval, Footprint-Based Retrieval involves a preprocessing stage to organise the training data into a two level hierarchy on which a two stage retrieval process operates. The preprocessing constructs a competence model which identifies 'footprint' cases which are landmark cases in the data. This process is not guaranteed to retrieve the same cases as k-NN but the results of the evaluation of speed-up and retrieval quality are nevertheless impressive [27].
– **Fish & Shrink:** This technique requires the distance to be a true metric as it exploits the triangle inequality property to produce an organisation of the case-base into candidate neighbours and cases excluded from consideration. Cases that are remote from the query can be bounded out so that they need not be considered in the retrieval process. Fish & Shrink can be guaranteed to be equivalent to k-NN [24].
– **Cover Trees for Nearest Neighbor:** This technique might be considered the state-of-the-art in nearest-neighbour speed-up. It uses a data-structure called a Cover Tree to organise the cases for efficient retrieval. The use of Cover Trees requires that the distance measure is a true metric, however they have attractive characteristics in terms of space requirements and speed-up

performance. The space requirement is $O(n)$ where n is the number of cases; the construction time is $O(c^6 n \log n)$ and the retrieval time is $O(c^{12} \log n)$ where c is a measure of the inherent dimensionality of the data [2].

These techniques involve additional preprocessing to construct data structures that are used to speed up retrieval. Consequently they are more difficult to implement than the standard k-NN algorithm. As emphasised at the beginning of this section, the alternative speed-up strategy is to reduce the dimension of the data – this is covered in the next section.

4 Dimension Reduction

Given the high dimension nature of the data, Dimension Reduction is a core research topic in the processing of multimedia data. Research on Dimension Reduction has itself two dimensions; the dimensions of a dataset of $|D|$ examples described by $|F|$ features can be reduced by selecting a subset of the examples or by selecting a subset of the features (an alternative to this is to transform the data into a representation with less features). Dimension reduction as achieved by supervised feature selection is described in section 4.1. Unsupervised feature transformation is described elsewhere in this book in the chapter on Unsupervised Learning and Clustering. The other aspect of dimension reduction is the deletion of redundant or noisy instances in the training data – this is reviewed in section 4.2.

4.1 Feature Selection

When the objective is to reduce the number of features used to describe data there are two strategies that can be employed. Techniques such as Principle Components Analysis (PCA) may be employed to *transform* the data into a lower dimension represention. Alternatively feature selection may be employed to *discard* some of the features. In using k-NN with high dimension data there are several reasons why it is useful to perform feature selection:

– For many distance measures, the retrieval time increases directly with the number of features (see section 3).
– Noisy or irrelevant features can have the same influence on retrieval as predictive features so they will impact negatively on accuracy.
– Things look more similar on average the more features used to describe them (see Figure 3).

Feature Selection techniques typically incorporate a search strategy for exploring the space of feature subsets, including methods for determining a suitable starting point and generating successive candidate subsets, and an evaluation criterion to rate and compare the candidates, which serves to guide the search process. The evaluation schemes can be divided into two broad categories:

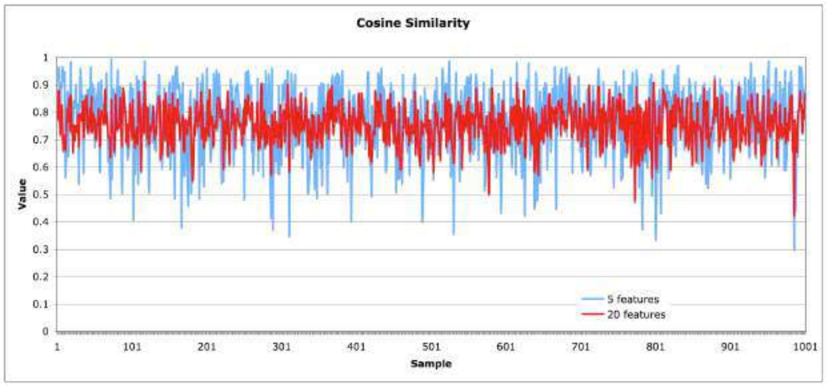

Fig. 3. The more dimensions used to describe objects the more similar on average things appear. This figure shows the cosine similarity between objects described by 5 and by 20 features. It is clear that in 20 dimensions similarity has a lower variance than in 5.

– **Filter** approaches attempt to remove irrelevant features from the feature set prior to the application of the learning algorithm. Initially, the data is analysed to identify those dimensions that are most relevant for describing its structure. The chosen feature subset is subsequently used to train the learning algorithm. Feedback regarding an algorithms performance is not required during the selection process, though it may be useful when attempting to gauge the effectiveness of the filter.
– **Wrapper** methods for feature selection make use of the learning algorithm itself to choose a set of relevant features. The wrapper conducts a search through the feature space, evaluating candidate feature subsets by estimating the predictive accuracy of the classifier built on that subset. The goal of the search is to find the subset that maximises this criterion.

It is worth mentioning at this point that some other classification techniques perform implicit feature selection. For instance the process of building a decision tree will very often not select all the features for use in the tree. Features not used in the tree have not role then in classification.

Filter Techniques Central to the Filter strategy for feature selection is the criterion used to score the predictivness of the features. In recent years Information Gain (IG) has become perhaps the most popular criterion for feature selection. The Information Gain of a feature is a measure of the amount of information that a feature brings to the training set [19]. It is defined as the expected reduction in entropy caused by partitioning the training set D using the feature f as shown in Equation 13 where D_v is that subset of the training set D where feature f has value v.

$$IG(D,f) = Entropy(D) - \sum_{v \in values(f)} \frac{|D_v|}{|D|} Entropy(D_v) \tag{13}$$

Entropy is a measure of how much randomness or impurity there is in the data set. It is defined in Equation 14 where c equals the number of classes in the training set and p_i is the proportion of class i in the data – entropy is highest when the proportions are equal.

$$Entropy(D) = \sum_{i=1}^{c} -p_i \log_2 p_i \tag{14}$$

In binary classification, the $Entropy(D)$ can be simplified to $Entropy(D) = -p_+ \log_2 p_+ - p_- \log_2 p_-$ where p_+ represents the class and p_- the non-class.

For comparison purposes it is we will also consider Odds Ratio (OR)[17] which is an alternative filtering criterion. For binary classification OR calculates the ratio of the odds of a feature occuring in the class to the odds of the feature occuring in the non-class.

$$OR(f,c) = \frac{Odds(f|c)}{Odds(f|\bar{c})} \tag{15}$$

Where a specific feature does not occur in a class, it can be assigned a small fixed value so that the OR can still be calculated. For feature selection, the features can be ranked according to their OR with high values indicating features that are very predictive of the class. The same can be done for the non-class to highlight features that are predictive of the non-class.

We can look at the impact of these feature selection criteria in an email spam classification task. In this experiment we selected the n features with the highest IG value and $\frac{n}{2}$ features each from $OR(f, spam)$ and $OR(f, nonspam)$ sets. The results, displayed in Figure 4, show that IG performed significantly better than OR. The reason for this is that OR is inclined to select features that occur rarely but are very strong indicators of the class. This means that some objects (emails) are described by no features and thus have no similarity to any cases in the case base. In this experiment this occurs in 8.8% of cases with OR compared with 0.2% for the IG technique.

This shows a simple but effective strategy for feature selection in very high dimension data. IG can be used to rank features, then a cross validation process can be employed to identify the number of features above which classification accuracy is not improved. This evaluation suggests that the top 350 features as ranked by IG are adequate.

While this is an effective strategy for feature selection it has the drawback that features are considered in isolation so redundancies or dependancies are ignored. Two strongly correlated features may both have high IG scores but one may be redundant once the other is selected. More sophisticated Filter techniques that address these issues using Mutual Information to score *groups* of features have been researched by Novovičová et al. [18] and have been shown to be more effective than these simple Filter techniques.

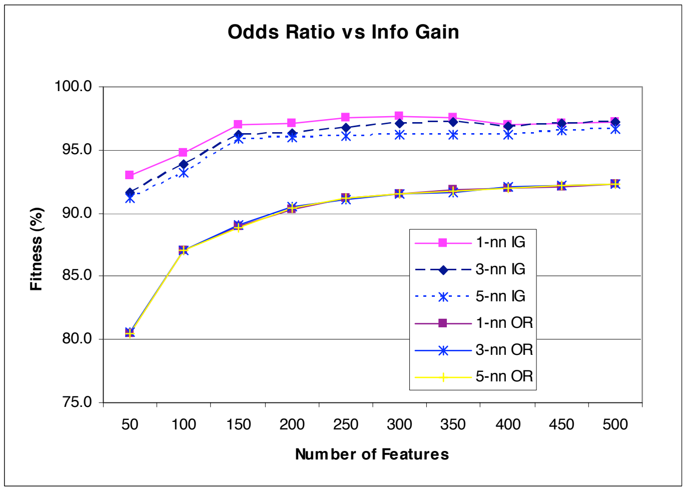

Fig. 4. Comparing Information Gain with Odds Ratio. Results of the average of three 10-fold cross validation experiments on a dataset of 1000 emails, 500 spam and 500 legitimate where word features only were used.

Wrapper Techniques The obvious criticism of the Filter approach to feature selection is that the filter criterion is separate from the induction algorithm used in the classifier. This is overcome in the Wrapper approach by using the performance of the classifier to guide search in feature selection – the classifier is *wrapped* in the feature selection process [11]. In this way the merit of a feature subset is the generalisation accuracy if offers as estimated using cross-validation on the training data. If 10-fold cross validation is used then 10 classifiers will be built and tested for each feature subset evaluated – so the wrapper strategy is very computationally expensive. If there are p features under consideration then the search space is of size 2^p so it is an exponential search problem.

A simple example of the search space for feature selection where $p = 4$ is shown in Figure 5. Each node is defined by a feature mask; the node at the top of the figure has no features selected while the node at the bottom has all features selected. For large values of p an exhaustive search is not practical because of the exponential nature of the search. The two most popular strategies are:

– **Forward Selection** which starts with no features selected, evaluates all the options with just one feature, selects the best of these and considers the options with that feature plus one other, etc.
– **Backward Elimination** starts with all features selected, considers the options with one feature deleted, selects the best of these and continues to eliminate features.

These strategies will terminate when adding (or deleting) a feature will not produce an improvement in classification accuracy as assessed by cross validation.

Both of these are greedy search strategies and so are not guaranteed to discover the best feature subset. More sophisticated search strategies can be employed to better explore the search space; however, Reunanen [20] cautions that more intensive search strategies are more likely to overfit the training data.

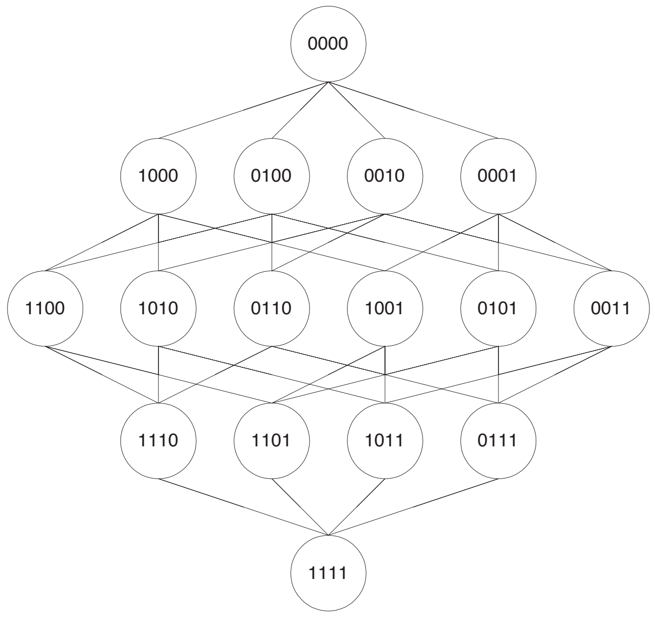

Fig. 5. The Feature Subspace

4.2 Instance Selection and Noise Reduction

An area of instance-based learning that has prompted much recent research is case-base editing, which involves reducing the number of cases in the training set while maintaining or even improving performance.

Early Techniques Case-base editing techniques have been categorised by [3] as competence preservation or competence enhancement techniques. Competence preservation corresponds to redundancy reduction, removing superfluous cases that do not contribute to classification competence. Competence enhancement is effectively noise reduction, removing noisy or corrupt cases from the training set. Figure 6 illustrates both of these, where cases of one class are represented by stars and cases of the other class are represented by circles. Competence preservation techniques aim to remove internal cases in a cluster of cases of the same class and can predispose towards preserving noisy cases as exceptions or border cases.

Noise reduction on the other hand aims to remove noisy or corrupt cases but can remove exceptional or border cases which may not be distinguishable from true noise, so a balance of both can be useful.

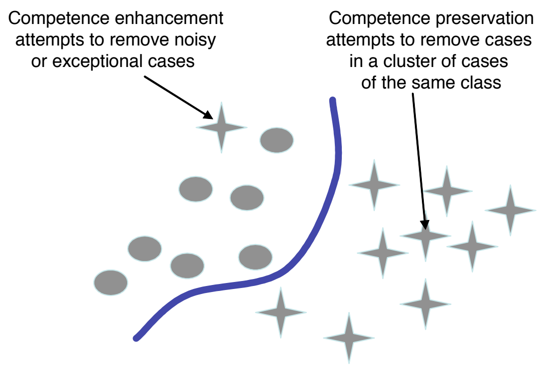
Competence enhancement attempts to remove noisy or exceptional cases
Competence preservation attempts to remove cases in a cluster of cases of the same class

Fig. 6. Case-base editing techniques demonstrating competence preservation and competence enhancement

Editing strategies normally operate in one of two ways; *incremental* which involves adding selected cases from the training set to an initially empty edited set, and *decremental* which involves contracting the training set by removing selected cases.

An early competence preservation technique is Hart's Condensed Nearest Neighbour (CNN) [9]. CNN is an incremental technique which adds to an initially empty edited set any case from the training set that cannot be classified correctly by the edited set. This technique is very sensitive to noise and to the order of presentation of the training set cases, in fact CNN by definition will tend to preserve noisy cases. Ritter et al. [21] reported improvements on the CNN with their Selective Nearest Neighbour (SNN) which imposes the rule that every case in the training set must be closer to a case of the same class in the edited set than to any other training case of a different class. Gates [8] introduced a decremental technique which starts with the edited set equal to the training set and removes a case from the edited set where its removal does not cause any other training case to be misclassified. This technique will allow for the removal of noisy cases but is sensitive to the order of presentation of cases.

Competence enhancement or noise reduction techniques start with Wilson's Edited Nearest Neighbour (ENN) algorithm [30], a decremental strategy, which removes cases from the training set which do not agree with their k nearest neighbours. These cases are considered to be noise and appear as exceptional cases in a group of cases of the same class.

Tomek [28] extended this with his repeated ENN (RENN) and his *all k-NN* algorithms. Both make multiple passes over the training set, the former repeating the ENN algorithm until no further eliminations can be made from the training set and the latter using incrementing values of k. These techniques focus on noisy or exceptional cases and do not result in the same storage reduction gains as the competence preservation approaches.

Later editing techniques can be classified as hybrid techniques incorporating both competence preservation and competence enhancement stages. [1] presented a series of instance based learning algorithms to reduce storage requirements and tolerate noisy instances. IB2 is similar to CNN adding only cases that cannot be classified correctly by the reduced training set. IB2's susceptibility to noise is handled by IB3 which records how well cases are classifying and only keeps those that classify correctly to a statistically significant degree. Other researchers have provided variations on the IBn algorithms [4, 5, 31].

Competence-Based Case-Base Editing More recent approaches to case-base editing build a competence model of the training data and use the competence properties of the cases to determine which cases to include in the edited set. Measuring and using case competence to guide case-base maintenance was first introduced by Smyth and Keane [26] and developed by Zu and Yang [32]. Smyth and Keane [26] introduce two important competence properties, the *reachability* and *coverage* sets for a case in a case-base. The *reachability set* of a case c is the set of all cases that can successfully classify c, and the *coverage set* of a case c is the set of all cases that c can successfully classify. The coverage and reachability sets represent the local competence characteristics of a case and are used as the basis of a number of editing techniques.

McKenna and Smyth [16] presented a family of competence-guided editing methods for case-bases which combine both incremental and decremental strategies. The family of algorithms is based on four features;

(i) an *ordering policy* for the presentation of the cases that is based on the competence characteristics of the cases,
(ii) an *addition rule* to determine the cases to be added to the edited set,
(iii) a *deletion rule* to determine the cases to be removed from the training set and
(iv) an *update policy* which indicates whether the competence model is updated after each editing step.

The different combinations of ordering policy, addition rule, deletion rule and update policy produce the family of algorithms.

Brighton and Mellish [3] also use the coverage and reachability properties of cases in their Iterative Case Filtering (ICF) algorithm. ICF is a decremental strategy contracting the training set by removing those cases c, where the number of other cases that can correctly classify c is higher that the number of cases that c can correctly classify. This strategy focuses on removing cases far from class borders. After each pass over the training set, the competence model is updated and the process repeated until no more cases can be removed. ICF includes a pre-processing noise reduction stage, effectively RENN, to remove noisy cases. McKenna and Smyth compared their family of algorithms to ICF and concluded that the overall best algorithm of the family delivered improved accuracy (albeit marginal, 0.22%) with less than 50% of the cases needed by the ICF edited set [16].

Wilson and Martinez [29] present a series of Reduction Technique (RT) algorithms, RT1, RT2 and RT3 which, although published before the definitions of coverage and reachability, could also be considered to use a competence model. They define the set of associates of a case c which is comparable to the coverage set of McKenna and Smyth except that the associates set will include cases of a different class from case c whereas the coverage set will only include cases of the same class as c. The RTn algorithms use a decremental strategy. RT1, the basic algorithm, removes a case c if at least as many of its associates would still be classified correctly without c. This algorithm focuses on removing noisy cases and cases at the centre of clusters of cases of the same class as their associates which will most probably still be classified correctly without them. RT2 fixes the order of presentation of cases as those furthest from their nearest unlike neighbour (i.e. nearest case of a different class) to remove cases furthest from the class borders first. RT2 also uses the original set of associates when making the deletion decision, which effectively means that the associate's competence model is not rebuilt after each editing step which is done in RT1. RT3 adds a noise reduction pre-processing pass based on Wilson's noise reduction algorithm.

Wilson and Martinez [29] concluded from their evaluation of the RTn algorithms against IB3 that RT3 had a higher average generalization accuracy and lower storage requirements overall but that certain datasets seem well suited to the techniques while others were unsuited. [3] evaluated their ICF against RT3 and found that neither algorithm consistently out performed the other and both represented the "cutting edge in instance set reduction techniques".

5 Conclusion: Advantages and Disadvantages

k-NN is very simple to understand and easy to implement. So it should be considered in seeking a solution to any classification problem. Some advantages of k-NN are as follows (many of these derive from its simplicity and interpretability):

- Because the process is transparent, it is easy to implement and debug.
- In situations where an explanation of the output of the classifier is useful, k-NN can be very effective if an analysis of the neighbours is useful as explanation.
- There are some noise reduction techniques that work only for k-NN that can be effective in improving the accuracy of the classifier [7].
- Case-Retrieval Nets [13] are an elaboration of the Memory-Based Classifier idea that can greatly improve run-time performance on large case-bases.

These advantages of k-NN, particularly those that derive from it's interpretability, should not be underestimated. On the other hand, some significant disadvantages are as follows:

– Because all the work is done at run-time, k-NN can have poor run-time performance if the training set is large.
– k-NN is very sensitive to irrelevant or redundant features because all features contribute to the similarity (see Eq. 1) and thus to the classification. This can be ameliorated by careful feature selection or feature weighting.
– On very difficult classification tasks, k-NN may be outperformed by more *exotic* techniques such as Support Vector Machines or Neural Networks.

References

1. D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.
2. A. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor. In *Proceedings of 23rd International Conference on Machine Learning (ICML 2006)*, 2006.
3. H. Brighton and C. Mellish. Advances in instance selection for instance-based learning algorithms. *Data Mining and Knowledge Discovery*, 6(2):153–172, 2002.
4. C. Brodley. Addressing the selective superiority problem: Automatic algorithm/mode class selection. In *Proceedings of the 10th International Conference on Machine Learning (ICML 93)*, pages 17–24. Morgan Kaufmann Publishers Inc., 1993.
5. R. M. Cameron-Jones. Minimum description length instance-based learning. In *Proceedings of the 5th Australian Joint Conference on Artificial Intelligence*, pages 368–373. Morgan Kaufmann Publishers Inc., 1992.
6. S.J. Delany and D. Bridge. Feature-based and feature-free textual cbr: A comparison in spam filtering. In D. Bell, P. Milligan, and P. Sage, editors, *Proceedings of the 17th Irish Conference on Artificial Intelligence and Cognitive Science (AICS'06)*, pages 244–253, 2006.
7. S.J. Delany and P. Cunningham. An analysis of case-base editing in a spam filtering system. In *7th European Conference on Case-Based Reasoning*. Springer Verlag, 2004.
8. G. W. Gates. The reduced nearest neighbor rule. *IEEE Transactions on Information Theory*, 18(3):431–433, 1972.
9. P. E. Hart. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, 14(3):515–516, 1968.
10. E. J. Keogh, S. Lonardi, and C. Ratanamahatana. Towards parameter-free data mining. In W. Kim, R. Kohavi, J. Gehrke, and W. DuMouchel, editors, *KDD*, pages 206–215. ACM, 2004.
11. R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artif. Intell.*, 97(1-2):273–324, 1997.
12. S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951.
13. M. Lenz and H-D. Burkhard. Case retrieval nets: Basic ideas and extensions. In *KI - Kunstliche Intelligenz*, pages 227–239, 1996.
14. M. Lenz, H.-D.Burkhard, and S. Brückner. Applying case retrieval nets to diagnostic tasks in technical domains. In Ian F. C. Smith and Boi Faltings, editors, *EWCBR*, volume 1168 of *Lecture Notes in Computer Science*, pages 219–233. Springer, 1996.

15. M. Li, X. Chen, X. Li, B. Ma, and P. M. B. Vitányi. The similarity metric. *IEEE Transactions on Information Theory*, 50(12):3250–3264, 2004.
16. E. McKenna and B. Smyth. Competence-guided editing methods for lazy learning. In W. Horn, editor, *ECAI 2000, Proceedings of the 14th European Conference on Artificial Intelligence*, pages 60–64. IOS Press, 2000.
17. D. Mladenic. Feature subset selection in text-learning. In C. Nedellec and C. Rouveirol, editors, *ECML*, volume 1398 of *Lecture Notes in Computer Science*, pages 95–100. Springer, 1998.
18. J. Novovičová, A. Malík, and P. Pudil. Feature selection using improved mutual information for text classification. In A. L. N. Fred, T. Caelli, R. P. W. Duin, A. C. Campilho, and D. de Ridder, editors, *SSPR/SPR*, volume 3138 of *Lecture Notes in Computer Science*, pages 1010–1017. Springer, 2004.
19. J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
20. J. Reunanen. Overfitting in making comparisons between variable selection methods. *Journal of Machine Learning Research*, 3:1371–1382, 2003.
21. G. L. Ritter, H. B. Woodruff, S. R. Lowry, and T. L. Isenhour. An algorithm for a selective nearest neighbor decision rule. *IEEE Transactions on Information Theory*, 21(6):665–669, 1975.
22. Y. Rubner, L. J. Guibas, and C. Tomasi. The earth mover's distance, multi-dimensional scaling, and color-based image retrieval. In *Proceedings of the ARPA Image Understanding Workshop*, pages 661–668, 1997.
23. Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.
24. J.W. Schaaf. Fish and Shrink. A Next Step Towards Efficient Case Retrieval in Large-Scale Case Bases. In I. Smith and B. Faltings, editors, *European Conference on Case-Based Reasoning (EWCBR'96*, pages 362–376. Springer, 1996.
25. R.N. Shepard. Toward a universal law of generalization for psychological science. *Science*, 237:1317–1228, 1987.
26. B. Smyth and M. Keane. Remembering to forget: A competence preserving case deletion policy for cbr system. In C. Mellish, editor, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI (1995)*, pages 337–382. Morgan Kaufmann, 1995.
27. B. Smyth and E. McKenna. Footprint-based retrieval. In Klaus-Dieter Althoff, Ralph Bergmann, and Karl Branting, editors, *ICCBR*, volume 1650 of *Lecture Notes in Computer Science*, pages 343–357. Springer, 1999.
28. I. Tomek. An experiment with the nearest neighbor rule. *IEEE Transactions on Information Theory*, 6(6):448–452, 1976.
29. D. Wilson and T. Martinez. Instance pruning techniques. In *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*, pages 403–411. Morgan Kaufmann Publishers Inc., 1997.
30. D. L. Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man and Cybernetics*, 2(3):408–421, 1972.
31. J. Zhang. Selecting typical instances in instance-based learning. In *Proceedings of the 9th International Conference on Machine Learning (ICML 92)*, pages 470–479. Morgan Kaufmann Publishers Inc., 1992.
32. J. Zu and Q. Yang. Remembering to add: competence preserving case-addition policies for case-base maintenance. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI 97)*, pages 234–239. Morgan Kaufmann Publishers Inc., 1997.