

# Investigating Noise Tolerance in Generalised Nearest Neighbour Learning

---

10 November 2005

Alexander U. J. Wong      Supervisor: Dr. Brent Martin

---

## **Abstract**

In this report we investigate the effects of integrating techniques and methods that tolerate noise well in nearest neighbour systems into generalised nearest neighbour systems and find whether or not this similarly helps in their tolerance of noise. We use Nearest Neighbour with Generalised Exemplars (NNGE) as our base generalised nearest neighbour system and create alternative variations,  $k$ -NNGE and NNGE-S, which we predict will perform better than the original NNGE in noisy domains. Our findings show that this is not in fact the case but insightful discoveries from this outcome has resulted in a beneficial investigation.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Generalised Nearest Neighbour Learning . . . . .	2
1.2	Noise . . . . .	3
1.3	Aims and Motivations . . . . .	4
1.3.1	$k$ -NN . . . . .	4
1.3.2	IB3 . . . . .	5
<b>2</b>	<b>Background</b>	<b>6</b>
2.1	Noise in Machine Learning . . . . .	6
2.1.1	The effect of noise on machine learning algorithms . . . . .	7
2.2	The problem of small disjuncts . . . . .	7
2.3	Nearest Neighbour . . . . .	8
2.3.1	$k$ -NN . . . . .	9
2.3.2	IBL family . . . . .	10
2.4	NNGE . . . . .	12
2.4.1	Creating/growing hyperrectangles . . . . .	12
2.4.2	Algorithm Design and Noise Tolerance . . . . .	13
<b>3</b>	<b><math>k</math>-NNGE</b>	<b>15</b>
3.1	Direct implementation . . . . .	15
3.1.1	Non-weighted $k$ -NNGE Algorithm . . . . .	16
3.2	Weighted $k$ -NNGE . . . . .	17
3.2.1	Weighted Algorithm . . . . .	18
<b>4</b>	<b>NNGE-S</b>	<b>19</b>
4.1	NNGE-S Algorithm . . . . .	21
<b>5</b>	<b>Experiments</b>	<b>23</b>
5.1	Test Domains . . . . .	23
5.2	Changes made to the original NNGE . . . . .	27
5.3	$k$ -NNGE . . . . .	28
5.3.1	Weighted $k$ -NNGE . . . . .	28
5.3.2	Performance gain from NNGE to $k$ -NNGE . . . . .	30
5.4	NNGE-S . . . . .	32
5.4.1	Discussion . . . . .	33
<b>6</b>	<b>Conclusion</b>	<b>35</b>
6.1	Further Research . . . . .	36
6.2	Acknowledgements . . . . .	37

<b>A</b>	<b>Changes from original NNGE implementation</b>	<b>41</b>
<b>B</b>	<b>Non-weighted <math>k</math>-NNGE results</b>	<b>42</b>
<b>C</b>	<b>Weighted <math>k</math>-NNGE results</b>	<b>43</b>
<b>D</b>	<b>IBk results</b>	<b>44</b>
<b>E</b>	<b>NNGE-S Results</b>	<b>45</b>

# Chapter 1

## Introduction

People take for granted the ability to make predictions based on previous knowledge. What we learn from experience usually helps us every day when we encounter same, similar or new experiences. When a situation is not exactly identical, previous knowledge can still be used to deduce the likely outcome or how to deal with a certain situation. Studies performed by psychologists have concluded that people recall past experiences to aid their solution of new problems when performing tasks [16]. For example, a doctor attending a patient considers not only the patient's symptoms, but also his or her previous experiences of other patients with similar symptoms. Such information can also be gleaned from other sources; for instance, the doctor may look up previous cases in a medical book.

Nearest neighbour is a machine learning algorithm introduced by Cover and Hart [7] that 'learns' by comparing each new case to previous examples, in a similar way to that practised by humans as described above. Machine learning is an area of artificial intelligence concerned with the development of techniques which allow computers to 'learn'. More specifically, machine learning is a method for creating computer programs for the analysis of data sets. Instance-based learning, of which nearest neighbour is a subset, is a branch of machine learning algorithms; other branches include: rule-based, artificial neural networks, genetic algorithms, and support vector machines.

An instance, also known as an example, is represented by a number of attributes related to a given domain. Instances have concept labels if they have been recorded and these can be used for training a machine learning algorithm. If no concept label is recorded, we try to guess the concept label of the instance based on what has previously been learnt. Consider a scenario where the doctor is taking symptoms from a patient and these may consist of temperature (hot, normal or cold), nausea (yes or no) and migraine (yes or no). The concept to be learnt here is whether a patient has a migraine or not and an example of an instance here could be {normal, yes, yes}, representing a case where a previous patient had normal temperature, nausea and was diagnosed with having a migraine.

In the nearest neighbour algorithm, all instances are generally stored in memory during training. When a new query instance is received the memory is searched to find the instance that matches the query instance most closely. Nearest neighbour will then infer that the concept label of the query instance is the same as the concept label of the most similar instance stored in memory.

There are advantages immediately evident in this approach when compared with human capability. Many instances can be stored by nearest neighbour so long as there

is enough memory to accommodate them. However, it is assumed that the capacity of memory to store information are far greater than human capabilities. The recall time of all the instances in memory is also much faster than what a human being is capable of. These advantages reduce the information overload on what people have to remember. Although machine learners in general are unable to make decisions that are as well informed as those of a human, they can provide a valuable aid in helping people make better and faster decisions. This is especially so in cases where machine learners can find important relationships between instance attributes and concepts that people did not previously know about.

One major difficulty that nearest neighbour has in general is producing accurate classifications for query instances when the data (especially training but also test) contains erroneous information. In machine learning this is known as noise. In this report we investigate noise tolerance in generalised nearest neighbour learning. We consider the effects of integrating techniques and methods that tolerate noise well in nearest neighbour systems into generalised nearest neighbour systems and find whether or not this similarly helps in their tolerance of noise. Our findings show that this is not in fact the case but insightful discoveries from this outcome has resulted in a beneficial investigation.

## 1.1 Generalised Nearest Neighbour Learning

Non-Nested Generalised Exemplars [17] (NNGE) extends on nearest neighbour by introducing generalised exemplars. Generalised exemplars are a bounded group of instances that share the same concept and are close in proximity within an  $n$ -dimensional problem space, where  $n$  is the number of attributes in each instance. The bounding of groups are implemented by axis-parallel  $n$ -dimensional rectangles, or hyperrectangles. Hyperrectangles represent each generalisation by an exemplar where each attribute value is replaced by either a range of values for a continuous-valued domain or a list of possible values for a discrete-valued domain [17]. This enables hyperrectangles to represent a more general rule more fully than many single instances. For the remainder of this investigation the terms exemplar and hyperrectangle will be used interchangeably but represent the same meaning.

NNGE was developed as an improvement to Nested Generalised Exemplar (NGE) theory, which Salzberg [25] implemented into his program Exemplar-Aided Constructor of Hyperrectangles (EACH). Investigations into the performance of EACH suggested that there were problems that needed to be addressed. Wettscherech and Dietterich [31] cited poor performance in NGE caused by nested and overlapping hyperrectangles, as well as the deficiencies in the search algorithm. NNGE sought to rectify the performance problems by using non-nested, non-overlapping hyperrectangles.

Martin [17] put forward that using generalised exemplars were desirable because they: improved the performance of nearest neighbour systems by improving the representation of large disjuncts; resulted in a useful set of rules that were comparable with those produced by other rule induction methods; and reduced classification time without sacrificing accuracy.

NNGE tries to find a balance between a general and specific bias so that useful generalisation is performed but not at the expense of the benefits of instance-based learning. A bias is a rule or method that causes an algorithm to choose one generalised output over another [18]. When the bias is shifted towards the middle of generality, the performance of a system tends to deteriorate in the area it previously performed best.

Instance-based learners do well with poorly represented concepts due to their specific bias, while induction systems generally perform well where the concepts are strongly represented due to their general bias. Generalised exemplars are an alternative way of combining these two opposite methods in a form that does not diminish the strengths of each approach.

NNGE has been found to perform poorly in domains that contain noise. This is because of two main problems: firstly, the underlying specific bias of the nearest neighbour component works well when presented with clean data sets only; and secondly, noise prevents its attempts to generalise. If NNGE can be modified so that it is more tolerant to noise, it would become a more versatile machine learning algorithm with improved performance and wider applicability to many domains. However, modification may mean that NNGE works well for noisy datasets but loses its ability to accurately learn about domains when provided with clean datasets. In this case, NNGE will become another machine learner that is particularly suited to certain applications. This is undesirable but is a common problem in machine learning and is known as the selective superiority problem [5].

## 1.2 Noise

Noise within data is a significant problem preventing machine learners from being more reliable, or applicable to a wide range of domains. When training an algorithm with real-world data, it is reasonable to assume that noise will be present. Noise is any incorrect attribute and/or concept value information and can be a result of errors in data entry, collection, measurement or corruption of data. If the potential for noise is not acknowledged, this can lead to machine learning algorithms “fitting the noise”. “Fitting the noise” is when the machine learner (mistakenly) ‘learns’ the noisy data as if it were not noisy data. Machine learners are typically gullible in this sense and because of this, noisy data can hide concepts and generally make learning harder.

Returning to the doctor example from the introduction, consider what would happen if the doctor was basing his diagnosis on erroneous information. The doctor may have learnt (wrongly) from a source that when a patient has a high temperature the diagnosis should always be that the patient is fine. The doctor will make an incorrect diagnosis every time a patient has a high temperature. This is an example of “fitting the noise” and nearest neighbour will do exactly the same thing. However, the doctor will normally realise that the information from the source was incorrect, and hence will correct it by finding the correct information and rejecting the erroneously learned information from memory. This type of intelligence is much harder to replicate in machine learning. Noise will often make instances in memory contradict each other. For instance, it is hard to say what diagnosis should be made in the case of a doctor with a leukemia patient who has in his experience similar, yet contradictory cases. Contradictory instances in memory is another common situation that machine learners struggle with.

Noise can also occur when attributes are missing, and with outliers in the dataset. Attributes can be missing for a number of reasons; for example, when there is a form to fill out that has optional fields. The machine learner has to come up with some way to deal with this. Whatever option is chosen the machine learner has to make some bias that will desirably not hide the underlying concept of instances with missing attributes. Outliers are values that are inconsistent with the values represented by the majority of instances within the domain for a certain concept. Although probably genuine values,

such rare cases will skew concepts and this can confuse a machine learner; how will it be able to tell the difference? This can be especially true if the machine learner works well with noisy domains.

## 1.3 Aims and Motivations

It is evident that improving machine learners in general would be of significant benefit to many fields. Relationships within large datasets, such as those stored by organisations, can be harvested via data mining, and complex decision-making can aid professionals such as doctors. Over the past decade many organisations have begun to routinely capture huge volumes of historical data describing their operations, their products, and their customers [19]. Finding important correlations within the vast amounts of information here can provide an organisation with an edge over competitors and more importantly drive profits. Although this is more technically data-mining, machine learning is central to the process. Other applications include predicting customer purchase behaviour, customer retention, and the quality of goods produced by a particular manufacturing line.

NNGE does not work well with noise. The ultimate goal of machine learning is to have an algorithm that has a classification performance of 100% over any dataset. This is already difficult to achieve, if not non-trivially impossible, when there is no noise present in the dataset. When noise is present in the dataset, classification performance is generally worse than if there was no noise present. The aim in this research is to improve NNGE when classifying noisy domains by using techniques and methods that have similarly improved nearest neighbour.

NNGE has not been investigated to a large extent. However, much research has been performed on nearest neighbour systems in general.  $k$ -NN and IB3 are solutions that have been previously applied to 1-nearest neighbour so that they better tolerate noisy domains. 1-nearest neighbour is the underlying algorithm that drives NNGE's classification process. Our research makes predictions about and then examines the application of these same solutions on NNGE to see if similar improvements in noisy domains can be achieved. These solutions and their potential effect when incorporated into the NNGE algorithm are detailed in the sections below.

### 1.3.1 $k$ -NN

The  $k$ -Nearest Neighbour ( $k$ -NN) [7] algorithm has been shown to help reduce the effect of noisy instances. It does this by predicting the class of a new example based on  $k > 1$  nearest neighbours instead of  $k = 1$  nearest neighbours, which is equivalent to the simplest nearest neighbour implementation. NNGE uses  $k = 1$  for the nearest neighbour component of its algorithm. However,  $k$ -NN's overall classification performance decreases as its bias becomes more generalised, that is as  $k \rightarrow \infty$ . NNGE has a bias that is generalised through generalised exemplars and this suggests that integrating  $k$ -NN into NNGE may not give a significant increase in classification performance for NNGE in general. This is compared to the increase in classification performance that  $k$ -NN can gain on 1-nearest-neighbour in noisy domains.

***Hypothesis 1:** The improvement in classification performance from NNGE to  $k$ -NNGE is less significant than the improvement from nearest neighbour to  $k$ -NN.*

### 1.3.2 IB3

IB3 is a member of the IBL [1] family of algorithms that introduces a statistical wait-and-see approach. IB3 applies an accuracy filter on instances and does not use an instance in classifying decisions until it has proved itself to be worthy of being used in the decision making process. We can apply this technique to NNGE in that if an instance is thought to be noise we can monitor its performance and make a decision later on whether to use it or discard it, depending on how well it performs. IB3 is difficult to implement directly into NNGE, but variations can be made.

Martin [17] suggests that NNGE does not work well with noise because it does not allow any conflict of class within a hyperrectangle. If a noisy instance falls within a hyperrectangle of a different concept, it forces that hyperrectangle to be pruned. A statistical filter on instances, such as that used in IB3, in NNGE could prevent this. Using such an approach could increase NNGE's tolerance in noisy domains, while keeping its performance in non-noisy domains similar to before.

***Hypothesis 2:** Using a statistical acceptability on instances approach, NNGE will have improved classification performance over normal NNGE in noisy domains.*

***Hypothesis 3:** Using a statistical acceptability on instances approach, NNGE will perform as well as normal NNGE in non-noisy domains.*

## Chapter 2

# Background

### 2.1 Noise in Machine Learning

The effects of noise on machine learning algorithms has been investigated since their inception. Wilson [32] describes two problems that can occur for algorithms as a result of learning with noisy data. Firstly, very few instances will be removed from the training set because they are needed to maintain the noisy decision boundaries (“fitting the noise”). To explain this further, consider a large area of instances in two-point space that contains the same concept. If you remove all instances that are not on the boundary of this area, the concept can still be adequately represented. However, if a noisy instance is in the middle of this area, it is necessary to keep an inner boundary of non-noisy instances around the noisy instance as well. This retention of instances is undesirable because this uses more memory and generally slows down classification time. The second problem is degradation of generalisation accuracy, especially if noisy instances are retained while good instances are removed as a result of the noisy instances. Noisy instances are not useful in classification, so it would be helpful to identify and discard them so that subsequent instances do not become misclassified, or in the case of NNGE, have less chance of being generalised.

Although noise can be present in attribute, or concept values, or both, Quinlan [21] demonstrated that attribute noise occurring simultaneously in all attributes describing the instance can result in faster degradation in classification accuracy than noise found only in the concept label [1]. Because of this, machine learning researchers usually focus on attribute values when considering noise tolerance, for example, the IBL family of algorithms (see section 2.3.2).

Missing attributes can be handled in a number of ways. Options include replacing the attribute value with one that is maximally different from possible values; or not to take that certain attribute value or even instance into consideration during classification.

Skewed distributions can fool machine learners that tolerate noise well into thinking there is noise. These are cases where an unusual or interesting class is rare among the general population. A good example here is financial fraud detected within transaction data. The majority of transactions are non-fraudulent but the very small percentage of fraudulent transactions can be misinterpreted as noise by a machine learning algorithm. The inability to distinguish between these rare cases (that is, true exceptions) and noise can be responsible for the difficulty in learning in the presence of noise [30].

### 2.1.1 The effect of noise on machine learning algorithms

Machine learners that are biased towards the generalised end of the range of generality tend to work well in certain domains, but usually not in the domains where a specific bias works well in, and vice versa. This is often because of the problem of small disjuncts described further in section 2.2. Decision trees and nearest neighbour have a very general and specific bias, respectively, and are used to explain this “range of generality bias” phenomenon below.

Decision trees, like ID3 [21], assume attributes make big decisions. If an attribute of an instance is noisy a big decision can possibly be made on erroneous data. The decision tree will generally keep those decisions indefinitely once made. However, because decision trees have a maximum generality bias that looks for well represented concepts, they are unlikely to be affected in this way unless there is a great deal of noise. Noise can also disguise the information gain of attributes and make them look less important causing the decision tree to create rules that model the noise.

Nearest neighbour has a specific bias that can over-represent small rules whilst not being able to represent more general concepts at all. If there is a noisy instance in memory and it is picked as the nearest neighbour to a query instance, the error has propagated. A noisy training instance can split exemplars unnecessarily in NNGE, while single-instance exemplars have the same problems as nearest neighbour.

Although C4.5 [23] and  $k$ -NN can help reduce the effect of noisy instances for ID3 and nearest neighbour, respectively, such techniques can also hurt generalisation in some cases, especially when noise is not present.

## 2.2 The problem of small disjuncts

The coverage or size of each disjunct is defined as the number of training examples that it correctly classifies [14]. A disjunct could be a rule found in decision-trees, an exemplar in NNGE, or many other kinds of representations found in other machine learning algorithms. Small disjuncts are those disjuncts that cover only a few training examples. How many constitutes a few is arbitrary and defining a cutoff value is one of the problems of small disjuncts.

Noise tends to appear in the learning system as spurious small disjuncts. Small disjuncts are primarily responsible for learning being difficult in the presence of noise, but Weiss and Hirsh [30] also show that even without noise, small disjuncts usually cover a disproportionate number of errors. For example, small disjuncts may collectively be responsible for 50% of the errors, but only cover 5% of the total instances. Again, the ambiguity of the size of a small disjunct is prevalent.

If small disjuncts are the cause for high error rates, one remedy may be to just prune them altogether. However, Holte et al. [14] found that small disjuncts cannot be totally eliminated if high predictive accuracy is to be achieved. The main reason for this is that small disjuncts correspond to the usually present rare cases within the domain under study.

Weiss and Hirsh [29] felt that: 1. noise tends to decrease the number of large disjuncts and increase the number of small disjuncts in the learned concept; 2. relatively low levels of noise will increase the percentage of errors contributed by small disjuncts, but this effect will diminish as higher levels of noise are applied; and 3. noise in the test set has an equalising effect that decreases the impact of the small disjuncts on learning.

Decision tree and rule inducing methods favour more general rules or large disjuncts because they implement a maximum generality bias. Instance-based learners are the opposite and have a maximum specificity bias that favours small disjuncts. We can adopt a more general bias by considering  $k$ -NNs rather than the single closest exemplar, thus shifting the underlying bias as close to the middle as possible. However, this often tends to reduce overall performance [17].

One approach involves employing a maximum specificity bias for learning small disjuncts, while continuing to use the more common maximum generality bias for the large disjuncts [14, 28]. Unfortunately these efforts have produced, at best, only marginal improvements. Given the prevalence of noise in real-world problem domains, a better understanding of small disjuncts and their role in learning may be required before further advances are possible [30].

## 2.3 Nearest Neighbour

Dasarathy [8] prepared an extensive survey of nearest neighbour systems in 1991. Nearest neighbour assumes that instances can be represented as individual points in an  $n$ -dimensional space, where  $n$  is the number of attributes needed to represent each instance. Nearest neighbour in its simplest form learns by storing the presented training data in memory verbatim. A Euclidean distance function is used to measure the similarity between instances with smaller distances implying that instances are more similar. The inductive bias corresponds to an assumption that the classification of an instance will be most similar to the classification of other instances that are nearby in Euclidean distance [18]. This is the most specific bias possible in machine learning and works well only if instances in memory that match a query instance are all of the same concept. Contradicting instances, most often caused by noise, makes it difficult for nearest neighbour systems to make an accurate decision.

Nearest neighbour systems work incrementally, learning over time. When learning from the available data gets close to covering the entire domain space, classification should approach 100%. In practise this will also mean that the database will grow large, slowing down the time it takes to classify a new example and/or fill up memory to capacity.

When there is only a small dataset nearest neighbour can identify concepts well compared with other machine learning techniques. This is because it can make useful predictions from as few as one example per class. Rule induction methods require a reasonable representation of each rule before they can be induced. This is because they need to consider the entire dataset each time the information content of attributes needs to be measured. The higher the information content, the more important an attribute is when deriving rules about classes. The smaller the dataset, the less accurate the measurement on information content for attributes are likely to be.

Nearest neighbour is sometimes referred to as a “lazy” learning method because it does not use much effort in learning from the dataset, while most of the effort is used when classifying a new example. During classification a search over all instances in memory is performed to find the closest instance to the query instance. With this delayed, or lazy learning, instead of estimating the target function once for the entire instance space, these methods can estimate it locally and differently for each new instance to be classified [18]. This is in contrast to a classification function used over the entire instance space. These are usually produced by rule induction systems that are “greedy”. Most of the effort is used during learning to generalise the dataset into a

small set of decision rules, making up a global function.

Two issues have been a problem for nearest neighbour systems throughout its lifetime. The first issue is that it is difficult to define a distance function that works well for both continuous and discrete attributes. The second issue is that nearest neighbour systems assign equal importance to each attribute.

Continuous attributes naturally lend themselves to the Euclidean distance function because instances can lie within attribute ranges corresponding to a point in  $n$ -dimensional space. Points closer together are theoretically more similar and as a group can represent a larger concept. When discrete attributes are introduced the Euclidean distance function becomes weaker because they are not so comparable. Conventionally, the distance between two discrete values are zero if the values are the same, and one if they are not. An example for the mismatch between these two types of attributes is where two continuous attributes may be a distance of ten apart. This is close if the range is in the thousands, but conversely be relatively far if the range is from twenty. Discrete attributes can only be at most a distance of one apart, given the conventional method. A popular solution is to linearly normalise continuous attributes to a range between zero and one. This is better but there is still a mismatch as continuous attributes can have any value between zero and one, whereas discrete attributes can only have a value of zero or one. Also, not all continuous attributes may be linear.

It is reasonable to assume that some attributes are more predictive than others. Conversely, there will be attributes that are completely irrelevant to the concepts being learned. If the distance between instances are calculated using equal weightings for each attribute of the instance, the real similarity between instances can become lost. Similar instances may seem far apart if these irrelevant attributes are included in the calculation and this is known as the “curse of dimensionality”. In the “curse of dimensionality” extraneous attributes look like noise in the distance function. In contrast, rule-based systems make decisions based on a subset of instance attributes when forming the hypothesis and thus suffer less from the “curse of dimensionality” than instance-based learners. However, this can lead to over-generalisation when small disjuncts are present.

A solution for overcoming the “curse of dimensionality” is using weighted attributes. This stretches the axes in the Euclidean space with the aim being to shorten the axes that correspond to less relevant attributes, while lengthening the axes that correspond to more relevant attributes [18].

### 2.3.1 $k$ -NN

The  $k$ -Nearest Neighbour algorithm is one of the most thoroughly analysed algorithms in machine learning, due in part to its age and in part to its simplicity [18]. Cover and Hart [7] presented early theoretical results, while Duda and Hart [11] provide a good overview.

$K$ -NN is a simple extension to the nearest neighbour theory. For  $k > 1$ , the algorithm assigns the most common concept value among the  $k$  nearest training examples. This has the effect of smoothing concept boundaries and prevents a single noisy example from incorrectly classifying the new one. Determining the value of  $k$  is arbitrary but as a rough guide the more noise a dataset contains the higher  $k$  should be. Also note that as  $k$  becomes larger the neighbourhood around the query instance becomes less local. Because it is not known *a priori* what the value of  $k$  should be, cross validation is a popular method here where different values of  $k$  are tested and the best result is

used. As  $k$  increases,  $k$ -NN's bias become more generalised, which can dampen overall performance as small disjuncts become less influential.

### 2.3.2 IBL family

Aha [1] developed a family of Instance-Based Learning (IBL) algorithms where iterative improvements were made in each successive algorithm. Starting with IB1, which is very similar to nearest neighbour, the goal of the IBL algorithms were to improve their behaviour in the presence of noise, uncertain attribute relevance and novel attributes.

All IBL algorithms are specified by a framework of 3 components [1]:

1. similarity function
2. classification function
3. concept description updater

#### IB1

This machine learner is almost the same as nearest neighbour with two differences. Firstly, IB1 linearly normalises all instance's continuous attribute values. This has the effect of making the ranges of continuous attributes the same, independent of their actual scale. Secondly, IB1 uses a simple method for tolerating missing attribute values. Missing attributes are assumed to be maximally different from the other value. IB1's similarity function is Euclidean distance, it classifies using nearest neighbour and when updating its concept description it saves all processed training instances.

#### IB2

IB2 improves on IB1 in that it requires less in storage but this is at the expense of accuracy being slightly worse. IB2 uses less storage space as its updater saves only misclassified instances after starting with a single instance for each important region of feature space. The theory behind this is that only instances close to the concept boundary edges need to be saved. Consequently, storage space is saved because of this. However, this places a bias towards noisy instances in training sets because they will be regularly misclassified. The saved noisy instances will then be used by IB2 to misclassify subsequent, similar instances [1]. Accuracy is especially poor early on because there are not enough instances of conflicting classes to accurately portray the differences between the new instance and the nearest neighbour [17]. IB2 has to assume that all saved instances are non-noisy to achieve high classification accuracy.

#### IB3

IB3 improves on IB2 by improving the algorithm's behaviour in the presence of noise. It applies an accuracy filter on instances and allows only those instances with significantly high classification records to be used in classification decisions.

A confidence interval of proportions significance test<sup>1</sup> is used to decide whether instances are acceptable, noisy or neither. A confidence interval is placed on both the instance's classification performance and its class's observed frequency. If the lower bound of the instance's confidence interval is greater than the upper bound of the class's

---

<sup>1</sup>This formula is found on page 276 in [2]

confidence interval, the instance is acceptable for classification decisions. Conversely, if the upper bound of the instance is less than the lower bound of the class, the instance is removed from memory indefinitely. Cases that lie in-between acceptable and removal places the instance in an *unsure* state where it is not allowed to be used in classification decisions but still kept in memory.

Aha [1] designed IB3's confidence interval test to make it difficult for an instance to be accepted using a high confidence level (90%) for acceptance but a lower confidence level (75%) for dropping instances, making it statistically easier for an instance to be dropped than be accepted.

IB3 differs from IB2 in three ways,

1. IB3 maintains classification records for all saved instances (that is, the number of correct and incorrect classifications of subsequently presented training instances);
2. Only those saved instances with significantly good classification records are acceptable for use in subsequent classification tasks; and
3. Those saved instances that appear to be noisy (that is, those instances with significantly poor classification performance) are discarded.

Noisy instances that are distant from concept boundaries should be detectable because their classification accuracies will be relatively low. This means we are pruning the database of only those small disjuncts that we are confident are noise.

Table 2.1 [1] shows that IB2 uses less storage space than IB1 but has worse classification performance due to its vulnerability of saving noisy instances. There is some correlation between the amount of instances saved in IB2/IB3 and how noisy the dataset is; if there is more noise there are more instances that are likely to be saved. IB3's significance test on instances significantly improves classification performance from IB2 to figures similar to IB1. IB3 uses less space than IB2 because noisy and uncertain instances are no longer being saved.

LED-7 and Waveform-21 [4] are artificial noisy domains where IB3 has high classification accuracy compared to IB1 and IB2, but uses much less storage. The Congressional Voting database contains easier to learn concepts with little noise. This is also indicative from the few instances that are needed to be saved by IB3, but its performance gains are not as large. The Cleveland and Hungarian databases are noisy and not linearly separable and in comparison to IB2, IB3 has significant improvements in classification accuracies and reduced storage requirements. This is especially evident in the Hungarian database, which contains many noisy instances [1]. LED-24 contains 17 irrelevant attributes and IB3 is sensitive to these as shown by its poor classification accuracy. The databases in table 2.1 are discussed more fully in section 5.1.

#### **IB4 and IB5**

IB4 differs from IB3 by improving its performance with regard to attribute relevance by using attribute weights. IB5 then improves on IB4 by being able to adapt quickly to the introduction of novel, relevant attributes during the training process. These improvements are outside of the scope of this investigation so will not be discussed further.

Database	IB1	IB2	IB3
LED-7	72.2/100	63.1/43.5	72.0/28.7
Waveform-21	74.7/100	70.3/31.8	74.7/14.1
Voting	91.7/100	90.7/11.4	91.9/7.5
Cleveland	76.2/100	70.5/29.9	77.9/18.6
Hungarian	59.5/100	56.6/36.0	78.6/18.7
LED-24	47.2/100	42.0/60.5	45.8/25.8

Table 2.1: IB3 performs better than IB1 and IB2. (Accuracy %/Storage %)

## 2.4 NNGE

NNGE extends on nearest neighbour with the introduction of generalised exemplars and how they are biased towards creation, growth and pruning. The inductive bias is that an instance will always try to generalise to the closest neighbour of the same class. If the growth of a hyperrectangle (generalisation) to cover a new instance conflicts with other instances or hyperrectangles, it is not performed. If a new instance conflicts with an existing hyperrectangle, that hyperrectangle is pruned.

NNGE differs from nearest neighbour with respect to dealing with missing attributes, attribute weighting and exemplar weighting. Missing attributes are ignored when calculating the distance function. For attribute weighting, NNGE has a dynamic attribute weighting method that only alters weights when exemplars are classified incorrectly, and for only those attributes that differed. For exemplar weighting, NNGE has a dynamic exemplar weighting method, which rewards, rather than penalises, exemplars in memory [17].

Martin [17] evaluated three motivations for using generalised exemplars in NNGE: better classification performance, the ability to more easily induce rules, and faster speed of classification. Generalised exemplars improves the performance of nearest neighbour systems by improving the representation of large disjuncts. Exclusive generalised exemplars produce a useful set of rules that may be compared with those produced by other rule methods. Generalising exemplars reduces classification time without sacrificing accuracy as less exemplars need to be examined.

### 2.4.1 Creating/growing hyperrectangles

A hyperrectangle in NNGE can only contain instances of the same class. For each continuous attribute, minimum and maximum values are stored representing the range of values covered by the hyperrectangle corresponding to the instances contained within. For each discrete attribute, a list of feature-values are maintained with each feature-value being a Boolean value. If the feature is represented by some instance within the hyperrectangle, that feature's value is true.

When an instance is added to the hyperrectangle, continuous attribute ranges are extended if the new instance contains continuous attributes that fall outside the hyperrectangle's current ranges. With discrete attributes, for every feature that has a true value within the instance, the corresponding feature in the hyperrectangle will be set to true if it is not already.

## 2.4.2 Algorithm Design and Noise Tolerance

The way in which a machine learning algorithm tolerates noise can generally be found in its underlying bias. When NNGE is broken down into components where noise can be a factor we can have a general idea of how changes to the implementation may improve noise tolerating capabilities.

The original NNGE algorithm [17] can be broken down into two categories: distance function and generalisation bias.

### Distance function

Distances between instances can be calculated using the Euclidean distance function. Because NNGE uses hyperrectangles, the simple Euclidean distance function must be adjusted to accommodate this feature. For continuous attributes, we calculate the distance from the closest boundary of a hyperrectangle to the query instance. Continuous attribute values are also normalised; a technique also implemented in the IBL family of algorithms [1]. For discrete attributes, we trivialise the distance to zero if the feature-values of the query instance are a subset of the hyperrectangle's feature-values, and one if they are not.

Apart from the changes above, there are three main implementation decisions in NNGE that affect the distance function: missing attributes, dynamic attribute weighting and exemplar reliability weighting.

**Missing attributes:** NNGE ignores missing attributes by excluding their contribution from the distance function. The final distance is divided by the number of non-missing attributes so that exemplars with all attribute values present are not penalised. Note that this is the same method that IB5 [1] applies, where its aim was to improve the classification performance after the introduction of a novel, relevant attribute, at any stage of the processing of the training set.

**Dynamic attribute weighting:** NNGE keeps a weighting for each attribute that it has seen. A larger weight means that an attribute is more important. If the values of this attribute are sparsely distributed, that is, an irrelevant attribute, it will be more heavily discriminated against.

If NNGE classifies a new example correctly, it leaves attribute weights unchanged; otherwise it increases the weights for those attributes that differed, accentuating the difference. The weights are increased by a multiplier constant,  $d$ . The value of  $d$  is arbitrary and performs well for any reasonable value, regardless of domain [17].

**Exemplar reliability weighting:** NNGE keeps a weight for each exemplar that it stores. The weight is determined by classification performance, where  $p$  is the number of correct predictions that the exemplar has made and  $n$  the number of incorrect predictions. The weight is calculated as

$$\text{exemplar weight} = \frac{n}{p+n}$$

Therefore, exemplars with more correct predictions will seem closer in distance. Noisy exemplars should thus become further away. This is in contrast to NGE [25], from which NNGE was adapted from (see section 1.1), which uses the formula

$$\text{exemplar weight}' = \frac{p+n}{p}$$

Martin [17] felt that this formula penalised exemplars that did not start well enough too heavily and suggested the use of the former formula instead for NNGE.

### **Generalisation Bias**

The way in which NNGE decides to generalise or prune exemplars can affect the representation of the underlying concepts that the algorithm is trying to find. Martin [17] sought to prevent over-generalisation.

A new instance can conflict with a hyperrectangle of a different class in a number of ways. The hyperrectangle maybe over-generalised, representing a concept outside its real concept area. A new instance may also be noisy and accidentally fall within a hyperrectangle of a different class.

To prevent a new instance from conflicting with a hyperrectangle of a different class, NNGE performs a pruning operation reducing the size of the hyperrectangle along a single feature axis. For discrete features, NNGE selects the feature where the feature-value to be removed predicts the class of the hyperrectangle the least well. For continuous features the range between the two adjacent examples along the axis under test is used, as this will determine the two new boundaries of the shrunken hyperrectangle. A conflicting instance can reduce the coverage of an over-generalised hyperrectangle towards the more realistic representation of the concept it is supposed to cover.

NNGE always tries to generalise new instances to their closest neighbour of the same class, that is, grow a hyperrectangle large enough to cover it. This generalisation is also performed after pruning operations on hyperrectangles. If generalisation causes a conflict, it will not be performed. Generalisation conflicts occur when a hyperrectangle is overlapping with another hyperrectangle.

Roy [24] introduced a generalisation parameter,  $g$ , that corresponds to the number of attempts the algorithm will make to generalise an instance with an existing hyperrectangle of the same class. At each failure of generalisation with the nearest neighbour the algorithm will search for the next nearest neighbour and attempt to generalise the instance with that until  $g$  attempts have been made. The higher the number of generalisation attempts,  $g$ , the more generalised the bias becomes. While ad hoc, Roy reported that  $g = 5$  generally works well for increasing the classification performance of NNGE.

## Chapter 3

### $k$ -NNGE

The pivotal decision in the implementation of  $k$ -NNGE ( $k$ -nearest neighbour with generalised exemplars) relates to the treatment of hyperrectangles, in terms of counting the  $k$  nearest neighbours. A direct implementation could be kept, where the  $k$  nearest neighbours are the closest hyperrectangles, independent of size, to a query instance. However, this approach may be biased towards trivial hyperrectangles because there are likely to be far more of these than non-trivial hyperrectangles. Implementing  $k$ -NNGE in this way may increase specificity instead of generality. This approach is also weakened by the fact that noise is most likely to be represented by a trivial hyperrectangle

Non-trivial hyperrectangles represent a much larger rule than trivial hyperrectangles. It would therefore make sense to give a weighting to hyperrectangles corresponding to their size. A hyperrectangle's size could be number of instances that it contains, or its  $n$ -dimensional volume.

$K$ -NN has been shown to improve classification performance over nearest neighbour in noisy domains [18], but degraded performance over non-noisy domains. This is due to the shift in bias toward generalisation, weakening the influence of small disjuncts that are needed to maintain high classification performance.

Nearest neighbour starts with the most specific bias possible, but  $k$ -NN shifts this more towards the middle of the range of generality. This effect generally increases the classification performance of  $k$ -NN over nearest neighbour in noisy domains. NNGE already has a generalised bias from nearest neighbour with hyperrectangles. Therefore, applying  $k$ -NN may not increase its performance greatly. Classification accuracy may even degrade if  $k$ -NN shifts NNGE's bias back towards the specific side of the range of generality.

#### 3.1 Direct implementation

NNGE stores hyperrectangles in memory, which in its trivial form is no different to how  $k$ -NN stores instances. Non-trivial hyperrectangles contain only instances of the same class and thus can be deemed "large instances". NNGE uses an adapted Euclidean distance function, where any query instance that falls within a hyperrectangle is given a distance of zero from that hyperrectangle. This is equivalent to a query instance having the exact same attribute values as an instance already in memory in  $k$ -NN. Section 3.1.1

summarises the algorithm<sup>1</sup>.

### 3.1.1 Non-weighted $k$ -NNGE Algorithm

```
For each instance
  replace missing attributes
  adjust attribute ranges

classify instance:
  while(list  $l$  has less than  $k$  exemplars)
    for each exemplar in memory
      compute distance from new instance
      if (distance < min distance so far)
        nearest neighbour = exemplar
      if (distance = min distance so far)
        nearest neighbour = exemplar with more instances

    add nearest neighbour to  $l$ 
    remove nearest neighbour from next search

  find the most popular class in  $l$ 
  return the nearest neighbour of that class
  (no nearest? Store new instance verbatim)

adjust model:
  if (correct prediction)
    increment positive count for exemplar
  else
    increment negative count for exemplar

  if (instance falls within exemplar)
    prune over-generalised exemplar
  else
    adjust weights for attributes with differing values

generalise the new instance
  extend attribute ranges of the exemplar to include the new instance

  if (exemplar is overlapping another exemplar of a different class)
    restore exemplar to original size
    store new instance verbatim
```

<sup>1</sup>The majority of the algorithm is explained in section 2.4. Roy's [24] generalisation feature is not included because it is not a part of this investigation. "replace missing attributes" and "if (exemplar is overlapping...)" are lines of the algorithm that differ from the original implementation but are not relevant to the  $k$ -NN implementation and are discussed in section 5.2.

## 3.2 Weighted $k$ -NNGE

Two weighted schemes will be discussed: a simple threshold count and a more complex volumetric version.

The simple version uses the number of instances contained within a hyperrectangle as its contribution to the  $k$  count. Each time a nearest hyperrectangle neighbour is found, the number of instances it contains is added to an accumulating counter,  $n$ . The algorithm for finding nearest neighbours stops when,  $n$ , reaches, or exceeds,  $k$ . It is important to note that  $k$  no longer corresponds to finding the  $k$  nearest neighbours, instead it is now a threshold for when to stop finding nearest neighbours. The class of the query instance is defined as the most popular class from the list of nearest neighbours found; the same as  $k$ -NN. In the event of a tie, the class of the hyperrectangle containing the most instances is chosen. A random choice is made if there is still no distinct winner.

This implementation is deliberately biased towards larger hyperrectangles. Larger hyperrectangles represent a more certain rule and thus can be more trustworthy than a single instance (trivial hyperrectangle). However, this can dampen overall performance as small disjuncts become less influential; this is usually the case with  $k$ -NN.

A more complex approach to weighted  $k$ -NNGE uses an  $n$ -dimensional sweep of a certain distance, calculating volumes for hyperrectangles to get their respective size. When the sweep only covers a portion of a hyperrectangle, we calculate only that portion of the hyperrectangle as its contribution to the sweep. This affects the hyperrectangles that are at the edge of the sweep. For this to work, two passes over the hyperrectangles in memory need to be performed. The first pass finds the  $k$  nearest hyperrectangles, independent of size. The second pass uses the distance to the  $k^{\text{th}}$  hyperrectangle to perform the  $n$ -dimensional sweep. Hyperrectangles covered by the sweep contribute to class counters, with the size of the hyperrectangles being directly proportional to the count added, including those fractional calculations at the edge of the sweep boundary. This implementation is more complex and computationally more expensive than the simple version approach described before.

One major complexity is calculating the volume of an  $n$ -dimensional hyperrectangle when both continuous and discrete attributes are present. The Euclidean distance function has this problem, but uses a simple linear metric, whereas an  $n$ -dimensional metric is calculated in this case. Calculating an  $n$ -dimensional metric will probably only exacerbate the incomparable types. Discrete attributes, always play a greater part in the volume calculated because there can only be a distance of zero or one. This also brings up the zero multiplier problem, where any number multiplied by zero will be equal to zero. Continuous attributes are often normalised to a range between zero and one, keeping some consistency with discrete attributes. However, it is unlikely that there would be a continuous distances at the extreme ends of its ranges all the time (although zero may occur often when instances fall inside hyperrectangles).

We will experiment with the simple approach in section 5 and put aside the more complex version as a novel approach that either requires more thought or may not even be feasible. Section 3.2.1 summarises the differing component of the simple weighted  $k$ -NNGE from the algorithm in section 3.1.1.

### 3.2.1 Weighted Algorithm

Same as the  $k$ -NNGE algorithm (section 3.1.1) except:

**classify instance:**

while( $n$  is less than  $k$ )

  for each exemplar in memory

    compute distance from new instance

    if (distance < min distance so far)

      nearest neighbour = exemplar

    if (distance = min distance so far)

      nearest neighbour = exemplar with more instances

  add nearest neighbour to  $l$

  add the number of instances within exemplar to  $n$

  remove nearest neighbour from next search

find the most popular class in  $l$

return the nearest neighbour of that class

(no nearest? Store new instance verbatim)

## Chapter 4

# NNGE-S

Three domains Martin [17] tested in his thesis contained noise and NNGE did not perform well in these. The reasons given for this were that NNGE does not allow any conflict of class within a rectangle. When noisy examples fall into a hyperrectangle of a different class, it requires the hyperrectangle to be pruned unnecessarily. In time the hyperrectangle may grow around the noisy instance but the correct concept cannot be fully represented again. This problem may be overcome by allowing a small amount of conflict of class within a rectangle, an approach similar to decision tree pruning.

The statistical acceptability test on instances, used in IB3, is maybe what is needed to allow some conflict within NNGE. However, unlike  $k$ -NN, IB3 cannot be directly implemented into NNGE. IB3 only saves instances when they are misclassified and only uses them during classification time if they are statistically acceptable. This bias is the opposite to NNGE, which must save all processed instances in order to create, grow and prune hyperrectangles. Compression of instances in memory can only be performed in NNGE after training, where instances within hyperrectangles can be discarded and only the boundaries of hyperrectangles are needed to be kept. However, we can still use the idea of statistical acceptability for hyperrectangles and call this variant NNGE-S.

Following IB3, hyperrectangles in NNGE-S can be in one of three states:

- Acceptable: used in classification decisions.
- Unsure: not used in classification decisions, but during classification, weights can be updated if it is truly the closest neighbour to a query instance.
- Rejected: removed from memory indefinitely.

### Acceptability

Statistical acceptability was introduced in IB3 so that the noisy instances saved by IB2 (because IB2 is biased towards them) are ignored. Using statistical acceptability in a case where all instances are saved, for example, NNGE, should not have any negative effects. Instead, it should have the effect of boosting the acceptability of instances that represent a concept well, while also increasing the chances of noisy instances being rejected.

Noisy examples are most commonly spurious small disjuncts and they should not be allowed to “generalise” or “split” existing hyperrectangles. Can we achieve this

if we do not know whether an instance is noisy *a priori*? IB3 stores instances in an *unsure* state to begin with. In NNGE, exemplars that are in an *unsure* state, should not be allowed to generalise or split hyperrectangles. As time passes, an exemplar may only be allowed these behaviours if it becomes statistically acceptable.

IB3's significance test uses the number of instances from its entire memory and the number of instances of a given class in its class frequency count. This same method will be applied to NNGE-S. That is we, disregard hyperrectangles, and consider single instances instead to avoid skewing the significance test towards smaller disjuncts.

### **Non-trivial exemplar bias**

Theoretically, noisy instances should not grow into hyperrectangles that contain two or more instances. This is because noisy instances have no relationship with any other instances and for at least two of the same class to be close together in Euclidean space would be highly improbable. We can make an assumption that hyperrectangles, which contain two or more instances, indicate that a more generalised concept has been found and should remain in an *acceptable* state indefinitely. This means that any hyperrectangle that contains two or more instances does not have to undergo statistical acceptance tests. Our greater assumption is that all non-trivial hyperrectangles will not contain noise. This is a variation on IB3, which does a statistical check after every classification and instances can always change states, unless they have been rejected.

### **Pruning**

Pruning hyperrectangles is a complex issue. Above, it has been decided that non-trivial hyperrectangles are indefinitely acceptable. However, they can still be pruned if a new instance conflicts with it. The case where a noisy example unnecessarily splits a hyperrectangle has been discussed as a reason why NNGE works poorly with noise. If instances were initially in an *unsure* state, hyperrectangles would not be split up straight away. However, in IB3, it is easy to calculate whether an instance is closer to an *unsure* instance than any other instance in memory. This is not the case with hyperrectangles in NNGE. If an instance falls within a hyperrectangle it is assigned a distance of zero. If there is an *unsure* exemplar lurking within that same hyperrectangle how can it be updated? To avoid this, when a query instance falls within a hyperrectangle, distances must instead be calculated with regard to the individual instances contained within the hyperrectangle. Now it is possible to find if an *unsure* instance is *truly* nearest neighbour and can be updated accordingly.

### **Seeding of acceptable exemplars**

If some instances are needed to start in an *acceptable* state, IB3, seeds its memory with an *acceptable* instance in each important concept area. Without knowing the important concept areas of a dataset *a priori*, this is impossible to do. Instead, we come up with a new parameter,  $u$ , which is the probability that a new instance starts in an *unsure* state, given that no nearest neighbour is found for it. Section 4.1 summarises the NNGE-S algorithm.

## 4.1 NNGE-S Algorithm

### For each instance

replace missing attributes  
adjust attribute ranges

### classify instance:

for each *acceptable* exemplar in memory  
  compute distance from new instance  
  if (distance < min distance so far)  
    nearest neighbour = exemplar  
  if (distance = min distance so far)  
    nearest neighbour = exemplar with more instances

return the nearest neighbour  
(no nearest? Store new instance verbatim with *u* chance of  
  being in *unsure* state)

### adjust model:

if (correct prediction)  
  increment positive count for exemplar  
else  
  increment negative count for exemplar  
  
if (instance falls within exemplar)  
  prune over-generalised exemplar  
else  
  if (exemplar contains more than two instances)  
    adjust weights for attributes with differing values  
  else if (exemplar tests as acceptable)  
    adjust weights for attributes with differing values  
  else  
    if (exemplar tests as rejectable)  
      remove exemplar from database  
    else  
      change exemplar state to *unsure*

### generalise the new instance

extend attribute ranges of the exemplar to include the new instance

if (exemplar is overlapping another exemplar of a different class)  
  restore exemplar to original size  
  store new instance verbatim

*continued on next page*

**check *unsure* exemplars**

```
for each exemplar in memory (acceptable or unsure)
  compute distance from new instance
  if (instance falls within exemplar)
    compute distance from closest instance within exemplar
  if (distance < min distance so far)
    nearest neighbour = exemplar
  if (distance = min distance so far)
    nearest neighbour = exemplar with more instances
```

```
return the nearest neighbour
```

```
if (nearest neighbour returned is unsure)
```

```
  if (nearest neighbour is of same class as instance)
    increment positive count for exemplar
```

```
  if (exemplar tests as acceptable)
    change exemplar state to acceptable
```

```
    if (conflict)
      prune over-generalised exemplar
```

```
    generalise (same as “generalise new instance” above):
  else
```

```
    increment negative count for exemplar
```

```
    if (exemplar tests as rejectable)
      remove exemplar from database
```

## Chapter 5

# Experiments

We created a new version of NNGE for these experiments that follows the original [17] implementation as closely as possible. We used the Waikato Environment for Knowledge Analysis [33] (WEKA) machine learning workbench to implement and evaluate NNGE and its variants. WEKA is a free open source collection of machine learning algorithms for data mining tasks. In this chapter, we first make comparisons between the new version and the original using the results produced by WEKA and results obtained from Martin’s thesis, respectively. The remainder of the results produced by NNGE variants ( $k$ -NNGE and NNGE-S) and other algorithms within WEKA are compared against the new NNGE version.

In his thesis, Martin [17] performed a 25-fold cross validation over datasets using a number of training examples consistent with experiments performed with NGE [25], composite learner and C4.5 [28]. In 25-fold cross-validation, the original dataset is partitioned into 25 subsets. Of the 25 subsets, a single subset is retained as the validation data for testing the model, and the remaining 24 subsets are used as training data. The cross-validation process is then repeated 25 times (the folds), with each of the 25 subsets used exactly once as the validation data. The 25 results from the folds can then be averaged to produce a single estimation.

To keep consistency, we ran WEKA with its 25-fold cross validation setting and use the default parameters of any machine learners chosen, unless otherwise stated. All classification accuracies reported are rounded to one decimal place, including those found in appendices.

### 5.1 Test Domains

The datasets used for the following experiments have been obtained from the UCI machine learning repository [20]. Table 5.1 summarises the the properties of the datasets, including the number of classes, the number of feature attributes, the number of instances within the database, and the frequency of the class that appears the most often within the database.

The datasets have been converted into WEKA’s Attribute-Relation File Format (ARFF) but this has not affected the integrity of the data, except for Cleveland for which the changes are described below. All datasets were used by Martin for his experiments with NNGE, apart from Promoter and Monks-2, which we found difficultly in converting to ARFF. These particular datasets had difficult concepts to learn but did

Database	# classes	# features	# instances	# frequency
Iris	3	4	150	33.3%*
LED-7	10	7	500	10.0%*
LED-24	10	24	500	10.0%*
Waveform-21	3	21	500	33.3%*
Waveform-40	3	41	500	33.3%*
Cleveland	2	13	303	54.1%
Hungary	2	13	294	64.0%
Voting	2	16	435	61.4%
BcW	2	9	699	65.5%
Diabetes	2	8	768	65.1%
Hepatitis	2	19	155	79.4%

Table 5.1: Test domain details (\* equal distribution)

not contain noise, therefore the loss of these datasets is not of major concern to this investigation.

***Iris plants.*** This is a well known database found in the pattern recognition literature. There are 150 iris flowers represented, each described by four continuous attributes corresponding to its dimensions. One class (Setosa) is linearly separable from the other two, while those two are not linearly separable from each other. There are no missing values. It is an easy domain to learn with many systems achieving around 95% accuracy.

***LED-7 display.*** A simple LED display contains seven Light-Emitting Diodes (Boolean attributes) used to represent a digit, thus there are ten concepts. 500 instances were generated using `led-creator`, requesting 10% noise, with 11.03% noise reported. This means that each attribute, excluding the class has a 11.03% chance of being inverted. Without noise this domain is simple with only one instance per concept needed to fully represent the entire domain. Several machine learning researchers have used this domain for testing noise tolerance. Table 5.2 lists some reported classification performances [4, 22, 27]. The Bayesian classifier theoretically provides optimal classification performance and is a valuable comparison figure.

System	Accuracy
Optimal Bayes	74
CART	71
Nearest Neighbour	71
C4	72.6
IWN	73.3

Table 5.2: LED-7 performance

***LED-24.*** This domain is an extension of the *LED-7* problem with an additional 17 irrelevant Boolean attributes added to the instance space making the domain more difficult to learn. 500 instances were generated using `led-creator+17`, requesting

10% noise, with 9.77% noise reported. Table 5.3 lists some reported classification performances<sup>1</sup> [4].

System	Accuracy
Optimal Bayes	74
CART	70
Nearest Neighbour	41
NTgrowth	71.5

Table 5.3: LED-24 performance

**Waveform-21** data generator. This domain has difficult concepts to learn containing three classes of waves. Each instance has 21 continuous-valued attributes between 0 and 6, which all include noise (mean 0, variance 1). Each class is generated from a combination of two out of three possible “base” waves. 500 instances were generated with a seed of 7 using waveform. Table 5.4 lists some reported classification performances [4].

System	Accuracy
Optimal Bayes	86
CART	72
Nearest Neighbour	78

Table 5.4: Waveform-21 performance

**Waveform-40**. This domain is an extension of the *Waveform-21* problem with an additional 19 irrelevant continuous-valued attributes which are all noise attributes with mean 0 and variance 1. 500 instances were generated with a seed of 7 using waveform+noise. Table 5.5 lists some reported classification performances [4].

System	Accuracy
Optimal Bayes	86
CART	72
Nearest Neighbour	38

Table 5.5: Waveform-40 performance

**Cleveland heart disease**<sup>2</sup>. This domain is from a set of four databases concerning heart disease diagnosis from four different locations. These domains are noisy and are not linearly separable<sup>3</sup>. Of the four databases, this one has been used most by machine learning researchers. Each database has the same instance format using only 13 of a

<sup>1</sup>NTgrowth: Aha,D.W., & Kibler,D. (1988). Unpublished data. [20]

<sup>2</sup>Compiled by Robert Detrano, M.D., Ph.D., Cleveland Clinic Foundation

<sup>3</sup>Detrano[9] reported that his discriminant analysis method for predicting heart disease resulted with accuracies of approximately 75%. [1]

possible 75 attributes for prediction. All attributes are continuously valued. The concept to be learned is the presence of heart disease in a patient. This is an integer value from 0 (no presence) to 4 (extreme presence). Previous experiments have reduced this domain into a two-class variety; presence of heart disease or not. This change has been made to the original dataset for the experiments here. There are six missing attributes. Table 5.6 lists some reported classification performances [15, 13].

System	Accuracy
NTgrowth	77
C4	74.8
CLASSIT	78.9

Table 5.6: Cleveland performance

**Hungarian heart disease**<sup>4</sup>. This domain is also part of the heart-disease directory that includes the *Cleveland heart disease dataset*. However, there are many more missing attributes in this dataset though. Detrano et al. [9] reported classification accuracies of 77% and 74% for CDF and CADENZA respectively.

**Voting records**. This data set includes votes for each of the U.S. House of Representatives Congressmen on the 16 key votes identified by the Congressional Quarterly Almanac, 1984. The class identifies Democrat or Republican party affiliation. All attributes are Boolean valued corresponding to a “yea” or “nea” vote<sup>5</sup>, with approximately 4% of attributes missing values. Missing attributes here are not unknown, they are just neither “yea” or “nay” (although the machine learner will still treat them as a missing attribute). STAGGER [26] achieved an accuracy of around 90-95%.

**Breast cancer - Wisconsin (BcW)**. This domain represents breast cancer sufferers who had either malignant or benign diagnoses. Each instance has nine integer-valued attributes. Three pairs of parallel hyperplanes were found to be consistent with 67% of data [34], while accuracy on the remaining 33% of dataset is 95.9%. If these hyperplanes cannot be found, it results in very low classification accuracy. Zhang [35] achieved 93.7% accuracy with 1-nearest-neighbour (using the original 369 instance group). Sixteen instances contain a single missing attribute value.

**Pima Indian diabetes**. Originally owned by the National Institute of Diabetes and Digestive and Kidney Diseases, this domain contains a binary concept for whether a patient has tested positive or negative for diabetes according to World Health Organisation criteria. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage. All eight attributes are continuous-valued.

**Hepatitis**. This domain represents information about patients with hepatitis and whether they lived or died. Each instance contains 19 attributes, with 13 Boolean attributes and the remaining 6 attributes continuous-valued. Gong [10] reported 80%

<sup>4</sup>Compiled by Andras Janosi, M.D., Hungarian Institute of Cardiology, Budapest

<sup>5</sup>This has been simplified down from a variety of possible votes

classification accuracy and Assistant-86 [6] achieved 83% accuracy.

## 5.2 Changes made to the original NNGE

The implementation of NNGE within WEKA has been kept as close as possible to the implementation made by Martin [17] with two exceptions: the handling of missing attributes in the distance function, and allowing overlapping hyperrectangles of the same class.

Martin’s implementation excludes missing attributes from contributing to the Euclidean distance measure, while the implementation we use replaces continuous attributes with a value of zero and discrete attributes with a dummy feature-value. Roy [24] implemented this in his version of NNGE<sup>6</sup> and it seemed to perform better than Martin’s implementation, which uses the same approach as IB5. We discuss the resulting performance differences further on.

The original NNGE did not allow overlapping hyperrectangles. Martin chose this because he wanted to prevent over-generalisation. We made a decision that hyperrectangles of the same class could overlap, which would increase generalisation, but not harm performance. If an instance falls within an overlapping region, the hyperrectangle containing the most instances will be defined as the closest neighbour and eventually contain that instance.

Applying our changes to the original NNGE implementation produced similar results with the changed implementation performing better in three domains, worse in three and equal in five, compared with the original implementation<sup>7</sup> (see appendix A). Generally, accuracies were comparatively close when they differed, except for the Hungarian dataset, which the new implementation reported a 15.3% point improvement over the original. Because of this, we used the new implementation with the changes described above.

Database	Martin’s	new NNGE	Roy’s
Iris	94.7	<b>95.3</b>	96.0
LED-7	69.4	<b>71.4</b>	67.6
LED-24	55.2	<b>60.2</b>	63.0
Waveform-21	68.6	<b>74.8</b>	74.8
Waveform-40	64.6	<b>70.2</b>	78.2
Cleveland	<b>80.7</b>	76.6	76.2
Hungary	81.5	<b>81.6</b>	79.9
Voting	92.9	<b>94.7</b>	94.7
BcW	<b>95.4</b>	93.3	—
Diabetes	71.6	<b>72.5</b>	73.2
Hepatitis	<b>83.2</b>	82.6	83.2

Table 5.7: NNGE performance differences

Table 5.7<sup>8</sup> compares the performance of different NNGE implementations. The

<sup>6</sup>Listed as **NNGe** in the WEKA distribution

<sup>7</sup>This is a comparison between the new version of NNGE with different settings, namely missing attributes and exemplar overlapping

<sup>8</sup>Results from Martin are taken from his Master’s thesis [17]. Results from Roy were run using his version

new version seemed to benefit from the changes above. It performed better than the Martin’s NNGE in eight of the eleven domains. Interestingly, there is quite a significant difference in performance over the waveform databases with improvements of 6.2% and 5.6% over the 21 and 40 attribute versions respectively. The new version also gave a large improvement of 5.8% in the LED-24 database, suggesting that the slight changes made to the original implementation benefited irrelevant attribute tolerance.

Out of interest, we have included the performance of Roy’s [24] NNGE implementation within WEKA. The default setting of five generalisations and five folds for mutual information were used. A downside to Roy’s implementation is that the dynamic feature weighting (mutual information) used does not work incrementally. This means that if one of a new instance’s attribute values falls outside of the ranges currently defined by the previous instances in memory, the mutual information for that attribute has to be re-computed.

Bakari [3] successfully used mutual information for feature weighting and Wettschereck and Dietterich [31] improved the results of NGE [25] using it. The mutual information between the values of a feature and the class of an instance determines the feature weight value. The mutual information will be zero for a feature that does not give any information about the class and proportional to the log of the number of classes for a feature that completely determines the class [24]. Theoretically, this should work well in the presence of irrelevant attributes. Table 5.7 shows that this is the case with LED-24 and Waveform-40, (which include 17 and 19 irrelevant artificially generated attributes, respectively) where Roy’s version recorded 6.0% and 8.0% accuracy increases over the next best version, respectively.

## 5.3 $k$ -NNGE

### 5.3.1 Weighted $k$ -NNGE

Implementing  $k$ -NNGE leaves a vulnerability where exemplars of all sizes are considered equivalent when finding the  $k$  nearest neighbours. The generality of larger exemplars can be eroded because the  $k$  nearest neighbours are more likely to consist of the usually more abundant smaller exemplars. For example, a large exemplar may cover over 30% of the domain, while small examples can cover less than 1%. Both are seen as a single exemplar when straightforwardly finding the  $k$  nearest neighbours. Applying a simple weighting scheme corresponding to the number of instances that an exemplar holds can neutralise the loss of generality when using  $k$ -NNGE (see section 3.2).

Table 5.8 shows the classification performances between the non-weighted and weighted versions of  $k$ -NNGE using the best  $k$ -values found in tables 5.9 and 5.10. The weighted version recorded accuracies that were equal in three domains, and better than seven domains, compared with the non-weighted version. The non-weighted version bettered the weighted version for classification accuracy in only the BcW domain by 0.3% points. In the seven domains that the weighted version performed better than the non-weighted version, there were improvements between 0.2% and 0.9% points. Because of the better classification performance of the weighted version of  $k$ -NNGE compared to the non-weighted version, we use its classification accuracies for comparison in the remainder of the experiments.

---

currently in WEKA. There was no result for the BcW dataset using Roy’s version because of unknown problems.

Database	non-weighted	weighted
Iris	95.3	<b>96.0</b>
LED-7	<b>70.5</b>	<b>70.5</b>
LED-24	61.7	<b>62.0</b>
Waveform-21	<b>74.8</b>	<b>74.8</b>
Waveform-40	70.2	<b>70.4</b>
Cleveland	76.6	<b>76.9</b>
Hungary	<b>81.6</b>	<b>81.6</b>
Voting	94.7	<b>94.9</b>
BcW	<b>93.7</b>	93.4
Diabetes	72.5	<b>73.4</b>
Hepatitis	83.2	<b>83.9</b>

Table 5.8: Non-weighted vs. weighted  $k$ -NNGE (best found  $k$ -value)

### Discussion

The weighted version’s better classification performance over the non-weighted version can be explained using tables 5.9 and 5.10<sup>9</sup>. Table 5.9 shows that for eight of the domains, non-weighted  $k$ -NNGE’s best classification accuracy is when  $k = 1$ . For those domains where  $k = 1$  does not produce the best classification accuracy, the  $k$ -value is still low with the LED-24, BcW, and Hepatitis domains having best  $k$ -values equal to 3, 3, and 5, respectively. For all domains, as  $k$  becomes larger, the classification performance seems to trend away from its best accuracy found in lower values of  $k$ . This observation leads us back to the corresponding mismatch between finding the  $k$  nearest neighbours when exemplars of all sizes are treated equivalently.

Database	1	3	5	7	9	15
Iris	<b>95.3</b>	94.7	<b>95.3</b>	94.7	94.7	94.7
LED-7*	<b>70.5</b>	68.2	68.0	65.8	66.5	67.3
LED-24*	60.2	<b>61.7</b>	60.5	56.2	56.7	57.5
Waveform-21	<b>74.8</b>	70.4	69.0	69.6	69.8	70.2
Waveform-40	<b>70.2</b>	66.4	66.8	67.6	66.8	65.0
Cleveland	<b>76.6</b>	71.0	73.3	69.6	73.6	69.0
Hungary	<b>81.6</b>	78.9	76.5	75.1	76.5	75.5
Voting	<b>94.7</b>	94.3	<b>94.7</b>	<b>94.7</b>	94.5	94.0
BcW	93.3	<b>93.7</b>	92.1	91.8	91.4	92.0
Diabetes	<b>72.5</b>	71.8	72.1	70.1	70.6	70.8
Hepatitis	82.6	81.9	<b>83.2</b>	82.6	82.6	81.9

Table 5.9: Classification performance of non-weighted  $k$ -NNGE for different  $k$ -values

Table 5.10 shows that by applying a weighting scheme to exemplars,  $k$ -NNGE does not prematurely find its best classification in the lower values of  $k$ , as it does in the non-weighted version. The best  $k$ -values for classification accuracy are more spread out for the weighted  $k$ -NNGE version because it uses a threshold for counting up to  $k$  or more single instances when consecutively finding nearest neighbours. If the nearest

<sup>9</sup>(\*Mean values. See appendices B and C)

neighbour is an exemplar containing more than one instance, each instance within that exemplar contributes towards the threshold. With weighted  $k$ -NNGE we find the best classification accuracies are generally found when  $k > 1$ . This gives more room to find a better classification accuracy as  $k$  increases, whereas it is unlikely that the best accuracy can be achieved once  $k > 1$  using non-weighted  $k$ -NNGE.

Database	1	5	9	15	21	27
Iris	95.3	95.3	95.3	<b>96.0</b>	95.3	95.3
LED-7*	<b>70.5</b>	66.1	66.3	66.3	65.8	66.8
LED-24*	60.2	59.8	59.0	60.5	<b>62.0</b>	57.9
Waveform-21	<b>74.8</b>	<b>74.8</b>	<b>74.8</b>	74.2	72.8	73.2
Waveform-40	70.2	70.2	70.2	70.0	<b>70.4</b>	<b>70.4</b>
Cleveland	76.6	75.2	74.3	<b>76.9</b>	73.3	73.9
Hungary	<b>81.6</b>	80.3	78.9	78.2	79.3	78.6
Voting	94.7	94.7	<b>94.9</b>	<b>94.9</b>	<b>94.9</b>	<b>94.9</b>
BcW	93.3	93.3	93.3	<b>93.4</b>	93.3	<b>93.4</b>
Diabetes	72.5	73.2	<b>73.4</b>	72.7	72.8	71.5
Hepatitis	82.6	82.6	81.9	81.9	<b>83.9</b>	83.2

Table 5.10: Classification performance of weighted  $k$ -NNGE for different  $k$ -values

### 5.3.2 Performance gain from NNGE to $k$ -NNGE

In the introduction, we made a hypothesis that the improvement in classification performance from NNGE to  $k$ -NNGE would be less significant than the improvement from nearest neighbour to  $k$ -NN. This is because nearest neighbour starts with the most specific bias possible and then  $k$ -NN gradually generalises that bias as  $k$  increases. In contrast, NNGE is deliberately more generalised than nearest neighbour due to its use of generalised exemplars. If increasing the general bias in nearest neighbour by applying  $k$ -NN is the reason for its performance increase, because NNGE has a generalised component, we would expect that this would dampen the performance increases sought from  $k$ -NNGE.

We used IBk, a machine learner distributed with WEKA, as the  $k$ -NN classifier used to prove our hypothesis. IBk is also used as a comparative nearest neighbour machine learner by setting  $k = 1$ . To obtain the best possible classification accuracy for  $k$ -NN, we ran IBk twice for each domain. For the first run, we used hold-one-out (also known as ‘leave-one-out’) cross validation<sup>10</sup> to find the best  $k$ -value. For the second run, we performed a 25-fold cross-validation using the  $k$ -value found from the first run. The best  $k$ -value for classification accuracies in IBk, for each domain, are presented in appendix D. With  $k$ -NNGE, we found the “best” classification accuracies by testing values of  $k$  arbitrarily.

Table 5.11 shows the performance improvements from nearest neighbour (IBk,  $k = 1$ ) to  $k$ -NN (IBk), and from NNGE to  $k$ -NNGE. Firstly, we compare nearest neighbour and NNGE; secondly, we compare the improvement in classification performance from nearest neighbour to  $k$ -NN, and from NNGE to  $k$ -NNGE; and finally, we find that  $k$ -NNGE does not have the same performance increase effect over NNGE, as  $k$ -NN has over nearest neighbour.

<sup>10</sup>This is a parameter of IBk that can be set in the WEKA

Nearest neighbour and NNGE is first compared to see how they perform before applying the  $k$ -NN component. Table 5.11 shows that nearest neighbour performs better in four domains: LED-7, Waveform-21, Cleveland, and BcW; while NNGE performs better in six domains: LED-24, Waveform-40, Hungary, Voting, Diabetes, and Hepatitis; with the simplest domain, Iris, producing an equivalent classification accuracy.

There is an improvement in classification accuracy from nearest neighbour to the best reported  $k$ -NN accuracy in every domain, except the BcW domain. This confirms that the BcW domain contains very little noise. Seven of the eleven domains had increases of more than 4.2% points, a large gain for machine learners, and two had increases of 15.6% and 19.4% points. The two large increases came from the LED-24 and Waveform-40 domains, respectively.

Of the classification accuracy improvements from NNGE to the best reported  $k$ -NNGE, no improvements were found in the Led-7, Waveform-21, and Hungary domains. Led-7 and Waveform-21 are artificial noisy domains, while the Hungary domain contains many missing attributes. Waveform-40, Cleveland, Voting and BcW domains reported classification performance increases that were negligible. However, relatively large performance increases were achieved in the Iris, LED-24 and Hepatitis domains.

Database	IBk ( $k=1$ )	IBk (best)	diff.	NNGE	$k$ -NNGE (best)	diff.
Iris	95.3	96.7	1.4	95.3	96.9	1.6
LED-7	76.8	78.6	1.8	70.5	70.5	0.0
LED-24	45.6	65.0	19.4	60.2	62.0	1.8
Waveform-21	78.4	86.6	8.2	74.8	74.8	0.0
Waveform-40	67.4	83.0	15.6	70.2	70.4	0.2
Cleveland	77.6	85.8	8.2	76.6	76.9	0.3
Hungary	78.9	83.7	4.6	81.6	81.6	0.0
Voting	92.6	93.3	0.7	94.7	94.9	0.2
BcW	95.9	95.9	0.0	93.3	93.4	0.1
Diabetes	69.8	75.9	6.1	72.5	73.4	0.9
Hepatitis	81.3	85.8	4.2	82.6	83.9	1.3

Table 5.11: IBk vs.  $k$ -NNGE

## Discussion

Interestingly, nearest neighbour performs considerably better than NNGE in the LED-7 and Waveform-21 domains, which are both deliberately noisy domains. Although NNGE is vulnerable to noisy domains, the same should apply to nearest neighbour, in theory, if not more so. More predictably, nearest neighbour performs poorly in the irrelevant-attribute domains, LED-24 and Waveform-40, which is a well known weakness of nearest neighbour[18]. Conversely, NNGE is more robust when learning domains with irrelevant attributes.

The large performance increases in accuracy from nearest neighbour to  $k$ -NN in LED-21, and Waveform-40, shows the power of  $k$ -NN to generalise around irrelevant attributes in domains. These large improvements have occurred even though nearest neighbour began with quite similar classification accuracies compared with NNGE.

Although we mentioned that NNGE is robust in the presence of irrelevant attributes,  $k$ -NNGE does not improve its performance in the area further.

With no classification performance increase from NNGE to  $k$ -NNGE in the LED-7, Waveform-21, and Hungary domains, it could be that NNGE tolerates noise to a degree where  $k$ -NNGE cannot help further. However, if we will look more closely into the characteristics of the LED-7 domain, we see that IBk reports its best accuracy when  $k=3$ <sup>11</sup>, and then accuracy drops off as  $k$  becomes larger. This indicates that for this domain, generalisation is unlikely to be the key to performance improvement. Also, the optimal Bayes classification accuracy for LED-7 was reported as 74% [4], while IBk managed 78.6%.

$K$ -NN produced a large improvement in the classification accuracy from nearest neighbour in the LED-24 domain, and  $k$ -NNGE also improved in this domain from NNGE. This indicates generalisation is a good technique to overcome the “curse of dimensionality” caused by irrelevant attributes.

It is worth noting again that we found the “best” classification accuracies reported by  $k$ -NNGE by testing values of  $k$  arbitrarily. In comparison, we used hold-one-out cross-validation in IBk to find the best  $k$ -value<sup>12</sup>. Because of this, it is quite possible that  $k$ -NNGE produces better classification accuracies for values of  $k$  different to those that we tested. However, tables 5.9 and 5.10 suggest that classification accuracies would not be much better anyway. Although, the relationship between  $k$  and classification accuracy is often not linear.

We have shown that the improvement in classification performance from NNGE to  $k$ -NNGE is less significant than the improvement from nearest neighbour to  $k$ -NN.  $k$ -NN also shows great versatility, performing better than  $k$ -NNGE in all but two domains (Iris and Voting). In the LED-7, Waveform-21, Waveform-40 and Cleveland domains  $k$ -NN performed considerably better than NNGE.  $K$ -NN improves from nearest neighbour in classification performance and this can be by a considerable amount in some cases. However,  $k$ -NN cannot be universally applied to nearest neighbour systems in general as a simple “add-on” to boost classification performance. This is supported by  $k$ -NNGE’s lack of performance increase from NNGE.

## 5.4 NNGE-S

In the introduction, we hypothesised that by using a statistical acceptability on instances approach, NNGE-S will have improved classification performance over NNGE in noisy domains, while keeping performance at least the same in non-noisy domains.

We ran NNGE-S with the parameter  $u=0.95$ <sup>13</sup>. We found that this parameter was not overly important and any setting between 0.0 and 0.95 gave similar results. Setting  $u=1.0$  makes all exemplars start in an *unsure* state and this frequently gave 0% classification accuracy because no exemplars changed into an *acceptable* state.  $Z$ -values for the acceptance and rejection confidence tests were set to 1.2817<sup>14</sup> and 0.6742<sup>15</sup>, respectively. These are the same  $z$ -values used by IB3 [2]. We could not find another combination of  $z$ -values that gave consistently better results using NNGE-S. Table 5.12

---

<sup>11</sup>See appendix D

<sup>12</sup>This saved a considerable amount of experimental time

<sup>13</sup>The probability that a new instance starts in an *unsure* state, given that no nearest neighbour is found for it. See section 4.

<sup>14</sup>90% confidence

<sup>15</sup>75% confidence

summarises the classification performance between NNGE and NNGE-S.

NNGE reported better classification accuracies than NNGE-S in the LED-7 and Waveform-21 domains, amongst others. We specifically mention these two because they are artificially noisy domains, which NNGE-S was hypothetically supposed to perform better in. NNGE also performed better in the Diabetes domain, and negligibly better in the Hungary and Hepatitis domains. These three domains are also noisy and are not linearly separable.

NNGE-S reported better classification accuracies than NNGE in the Iris, LED-24, Waveform-40, Cleveland, Voting and BcW domains. Iris, Voting and BcW are the least noisy domains available in this experiment. Of these three domains, Iris and BcW produced small classification accuracy increases of 0.4% and 0.5% points, respectively, while the Voting domain produced a more significant increase of 1.3% points from NNGE. LED-24 and Waveform-40 are domains that contain numerous irrelevant-attributes and NNGE-S produced relatively significant classification performance increases in these domains of 1.5% and 2.1%, respectively.

Database	NNGE	NNGE-S	(best recorded)
Iris	95.3	<b>95.7</b>	96.0
LED-7	<b>70.5</b>	69.8	71.0
LED-24	60.2	<b>61.7</b>	63.2
Waveform-21	<b>74.8</b>	73.0	75.0
Waveform-40	70.2	<b>72.3</b>	73.4
Cleveland	76.6	<b>76.9</b>	80.9
Hungary	<b>81.6</b>	81.5	82.7
Voting	94.7	<b>95.8</b>	96.3
BcW	93.3	<b>93.8</b>	94.3
Diabetes	<b>72.5</b>	71.8	73.0
Hepatitis	<b>82.6</b>	82.4	83.2

Table 5.12: NNGE vs. NNGE-S

### 5.4.1 Discussion

The results from above proved contrary to our hypothesis that by using a statistical acceptability on instances approach, NNGE-S would have improved classification performance over NNGE in noisy domains, while keeping performance at least the same in non-noisy domains. NNGE-S did not perform better than NNGE in noisy domains, instead improving classification performance in less-noisy domains.

LED-24 and Waveform-40 contain irrelevant attributes, and similarly with  $k$ -NNGE, NNGE-S seems to handle these types of attributes better than noisy attributes. Again, we mention that generalisation seems to be a key factor in overcoming the “curse of dimensionality”. Also similar with  $k$ -NNGE, NNGE-S performed poorly in the noisy domains, LED-7 and Waveform-21. We can reason that noise in relevant attributes can make them seem like irrelevant attributes. Because the underlying general bias of NNGE works well for irrelevant attributes, this harms noisy relevant attributes that are treated in a similar way.

Coincidentally, the domains in which NNGE-S showed classification accuracy improvements from NNGE, also happened to be the same domains that  $k$ -NNGE showed

improvements from NNGE. This suggests that  $k$ -NN and IB3 may be trying to do the same thing. It is obvious that  $k$ -NN increases the generality bias from nearest neighbour. IB3 surreptitiously does the same thing too, even though its aim is to rather single out noisy instances and remove them. By trying to get rid of noisy instances, IB3 is trying to maintain concept boundaries (represented by misclassified acceptable instances) that are as large as possible, effectively producing generalised areas. This is just trying to increase the generality bias from nearest neighbour, which is akin to  $k$ -NN.

Overall, the classification performance of NNGE-S is still much poorer than what IBk, a far simpler machine learner, can achieve. Our explanation for this is similar to why we think  $k$ -NNGE also does not perform better than expected.  $K$ -NNGE struggles with appropriately being able to consider the size of exemplars when choosing the  $k$  nearest neighbours. Analogously, NNGE-S struggles to keep smooth concept boundaries when there are exemplars of different sizes, which can make them jagged. Generalised exemplars of different sizes in NNGE do not seem to be “nicely” compatible when integrated with  $k$ -NN and IB3 techniques.

NNGE is consistent in that there is no variance when producing classification accuracies, except for the LED-7 and LED-24 domains. When we start considering NNGE-S’s best recorded classification accuracy, in table 5.12, its classification performance is better than NNGE in all domains except for LED-7<sup>16</sup>. This is promising, and if we can find how to achieve this “best” accuracy all of the time for NNGE-S, we will have improved NNGE. The shortcoming of NNGE’s variance in classification accuracy suggests there is further research to determine how to get that best result all the time. A more consistent system is always desirable.

Also worth noting is the use of  $u$  as a probabilistic parameter to initially seed the exemplars in memory that begin in an *acceptable* state. In comparison, IB3 initially seeds important regions of feature space with *acceptable* instances. Aha [2] is ambiguous with how this is implemented within IB3, but we predict that an instance of each class is chosen as an *acceptable* instance. It would therefore be interesting to know if this technique is still suitable for linearly-separable domains. Also, if this seeding technique were to be implemented in NNGE-S, would it have performed better?

---

<sup>16</sup>To be fair, we must consider LED-7’s best classification accuracy of 71.8%. LED-24’s best was 61.0%. These are the only two domains that varied under NNGE.

## Chapter 6

# Conclusion

During this investigation into noise tolerance in generalised nearest neighbour learning, we found that the classification performance did not improve in noisy domains when applying  $k$ -NN, or statistical methods used by IB3, to NNGE. Although our aims of finding the reverse outcome were not met, we justify why this is the case.  $K$ -NNGE, and NNGE-S are variants of NNGE that integrate noise tolerating techniques from  $k$ -NN and a statistical approach adopted from IB3, respectively. NNGE,  $k$ -NNGE, NNGE-S, and IBk were tested over eleven domains to gather evidence on whether  $k$ -NNGE and/or NNGE-S could improve classification performance from NNGE.

We first experimented with  $k$ -NNGE, hypothesising that its improvement in classification performance from NNGE to  $k$ -NNGE is less significant than the improvement from nearest neighbour to  $k$ -NN and found this to be the case. However, the best classification accuracies for  $k$ -NNGE may not have been found during experiments because we only arbitrarily tested  $k$ -values. Although, there is also evidence that classification accuracies found by testing other  $k$ -values would not have been likely to be much better than those already found. IBk was shown to be a versatile performer, and after finding the best  $k$ -value, achieved better classification accuracies than NNGE,  $k$ -NNGE and NNGE-S over all domains under test, except Voting. LED-7 was a key noisy domain for our experiments and we found that generality, which  $k$ -NN systems provide, was not a solution for improving performance accuracy in this domain.

We also showed that a weighted exemplar approach to finding the  $k$  nearest neighbours in  $k$ -NNGE was better than a non-weighted approach. A non-weighted approach unfairly considers all exemplars as equivalent when finding the  $k$  nearest neighbours. This is biased towards small disjuncts and neutralises the generality of NNGE. A simple weighted approach considers the sizes of exemplars by applying a threshold. When nearest neighbours are found, the number of instances contained within the exemplar contributes towards the threshold until it is reached. Although this weighted approach worked better than the non-weighted approach, it did not perform satisfactorily. Further investigation into methods which consider the size of exemplars more appropriately when finding the  $k$  nearest neighbours in  $k$ -NNGE still need to be made.

Our second experiment concerning NNGE-S hypothesised that by using a statistical acceptability on instances approach, NNGE-S will have improved classification performance over NNGE in noisy domains, while keeping performance at least the same in non-noisy domains. Classification results from NNGE-S proved to be contrary to this hypothesis and instead did not perform better than NNGE in noisy domains, while improving classification performance in less-noisy domains.

NNGE-S reported better classification accuracies than NNGE in almost every domain, except for those that contained noise. However, unlike NNGE, which had variance in classification accuracies in the two LED domains, NNGE-S had variations in all domains. Putting aside the variations in NNGE-S and instead considering the best classification accuracies for each domain, NNGE-S outperformed NNGE in all but the LED-7 domain. We conclude that if consistency can be found in obtaining the “best” classification accuracy with NNGE-S, it will supersede NNGE as a machine learner.

Following the coincidental finding where  $k$ -NNGE and NNGE-S showed improved classification accuracies on the same domains from NNGE, we discovered  $k$ -NN and IB3 to be similar in their generalisation bias. This penultimate discovery lead us to find why NNGE-S, like  $k$ -NNGE does not integrate “nicely” with NNGE.

It is well known that increasing  $k$  in  $k$ -NN increases its generality bias. Although IB3 works to single out and then remove noisy instances from memory, surreptitiously, this also inadvertently increases the generality bias. By removing noisy instances, IB3 tries to maintain concept boundaries (represented by misclassified acceptable instances) that are as large as possible, effectively making generalised areas. Extending from this discovery we found that NNGE-S has problems analogous to  $k$ -NNGE’s consideration of exemplar size problem. NNGE-S struggles to keep smooth concept boundaries when there are exemplars of different sizes, which can make the boundary jagged. We conclude from this that generalised exemplars of different sizes in NNGE do not seem to be “nicely” compatible when integrated with  $k$ -NN and IB3 techniques.

Finally, from both experiments performed with  $k$ -NNGE and NNGE-S, we found that their generalisation bias generally worked well for overcoming the “curse of dimensionality” problem, with NNGE-S more so than  $k$ -NNGE. However, because both NNGE variations did not work well with noisy domains, we determined that noisy relevant attributes were most likely being mis-interpreted as irrelevant attributes by the machine learners. This causes  $k$ -NNGE and NNGE-S to generalise around the noisy relevant attributes, effectively losing important classification information, and explains their poor performance in noisy domains.

## 6.1 Further Research

We first discuss further research that is directly related to the carrying out of this investigation, and then finally discuss further research that has not been previously mentioned or mentioned very little thus far.

A simple weighted approach for finding the  $k$  nearest neighbours in  $k$ -NNGE was discussed and implemented within this investigation. However, its results were not satisfactory and it is possible that a better scheme can be produced to provide more satisfactory results. A starting point could be the more complex version discussed in section 3.2, which involved calculating volumes of hyperrectangles in an  $n$ -dimensional sweep.

We obtained performance accuracies of  $k$ -NNGE were by arbitrarily choosing values of  $k$  to test. With IB $k$ , we used its hold-one-out cross-validation parameter in WEKA to first find the best  $k$ -value for a domain. This same operation can be implemented within  $k$ -NNGE to find its best  $k$ -values for domains so that we can find its best classification accuracy and be more conclusive of the findings in this investigation.

We found that  $k$ -NNGE struggles with appropriately considering exemplar sizes when finding the  $k$  nearest neighbours. We also found that the different sizes of exemplars in NNGE-S can prevent the concept boundaries from being smooth. They are

disjunct problems, and both machine learners will work better when considering disjuncts of the same size. Research into the ratio of small disjuncts to large disjuncts of  $k$ -NNGE and NNGE-S after training and their relationship with classification performance. It would be expected that the higher the ratio, or less similar in size that the disjuncts are overall, the less likely it would be for  $k$ -NNGE and/or NNGE-S to obtain good classification accuracies.

Onto further research that has not been previously mentioned or mentioned little thus far, we begin with Roy's [24] generalisation feature. This feature attempts to generalise up to the first  $g$  nearest neighbours of the same class and was not considered in this investigation. Roy did not perform much analysis on this feature and extensive analysis could be beneficial. This feature is already implemented and on inspection seems like it could perform better than  $k$ -NNGE and/or NNGE-S without making immediate changes to its implementation.

Another way to reduce noise is to consider other alternative statistical methods. One such method that we can adapt from tree node splitting is the use of the Hoeffding bound [12]. The Hoeffding bound estimates the true mean of an attribute within a given error. We can use this to statistically check certain attributes of query or stored instances for whether they are noisy or not. However, applying this method to a nearest neighbour system may not be computationally cost-effective as the memory becomes large,

## 6.2 Acknowledgements

The author would like to firstly thank his supervisor, Dr. Brent Martin, for his helpfulness and flexibility towards the carrying out of this investigation, especially the readings of all the drafts and answering of e-mails towards the end. Also, a huge thank you to Karthik Nilakant, Lara Rennie, and Amy Chisholm for giving up their valuable time in proof reading my work. Warwick Irwin, who allowed us to incorporate our research projects within an assessment in his course. It saved seriously valuable amounts of time. Finally, the COSC honours room for an enjoyable and memorable year.

# Bibliography

- [1] Aha, D. W. Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies*, 36(2):267–287, 1992.
- [2] Aha, D. W., Kibler, D. and Albert, M. K. Instance-Based Learning Algorithms. *Machine Learning*, 6(1):37–66, 1991.
- [3] Bakiri, G. *Converting English Text To Speech: A Machine Learning Approach*. PhD thesis, Oregon State University, 1991. (Unpublished).
- [4] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. *Classification and Regression Trees*. Wadsworth International, 1984. See pages 43–49.
- [5] Brodley, C. Addressing the selective superiority problem: Automatic algorithm/model class selection. In *Machine Learning: Proceedings of the Tenth International Conference*, pages 17–24. Morgan Kaufmann, 1983.
- [6] Cestnik, B., Kononenko, I. and Bratko, I. ASSISTANT 86: A Knowledge-Elicitation Tool for Sophisticated Users. In *EWSL*, pages 31–45, 1987.
- [7] Cover, T. M. and Hart, P. E. Nearest neighbor pattern classification. *IEEE, Transactions on Information Theory*, IT-13, 1:21–27, 1967.
- [8] Dasarathy, B. V., editor. *Nearest Neighbour (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, 1991.
- [9] Detrano, R., Janosi, A., Steinbrunn, W., Pfisterer, M., Schmid, J., Sandhu, S., Guppy, K., Lee, S. and Froelicher, V. International application of a new probability algorithm for the diagnosis of coronary artery disease. *American Journal of Cardiology*, 64:304–310, 1989.
- [10] Diaconis, P. and Efron, B. Computer-Intensive Methods in Statistics. *Scientific American*, 248(5), May 1983.
- [11] Duda, R. O. and Hart, P. E. *Pattern Classification and Scene Analysis*. Wiley, 1973.
- [12] Gama, J. and Medas, P. Learning in Dynamic Environments: Decision Trees for Data Streams. In *PRIS*, pages 149–158, 2004.
- [13] Gennari, J. H., Langley, P. and Fisher, D. Models of incremental concept formation. *Journal of Artificial Intelligence*, 40:11–61, 1989.

- [14] Holte, R. C., Acker, L. E. and Porter, B. W. Concept Learning and the Problem of Small Disjuncts. In *Proc. of the 11th IJCAI*, pages 813–818, Detroit, MI, 1989.
- [15] Kibler, D. and Aha, D. Learning representative exemplars of concepts: An initial case study. In *Proceedings of the Fourth International Workshop on Machine Learning*, pages 24–30, Irvine, CA, 1987. Morgan Kaufmann.
- [16] Kolodner, J. L. *Retrieval and organizational strategies in conceptual memory: a computer model*. PhD thesis, Yale University, 1980.
- [17] Martin, B. Instance-Based Learning: Nearest Neighbour with Generalisation. Master's thesis, University of Waikato, 1995.
- [18] Mitchell, T. M. *Machine Learning*. McGraw-Hill Higher Education, 1997.
- [19] Mitchell, T. M. Machine learning and data mining. *Communications of the ACM*, 42(11):30–36, 1999.
- [20] Newman, D. J., Hettich, S., Black, C. L. and Merz, C. J. UCI Repository of machine learning databases, 1998.
- [21] Quinlan, J. R. The effect of noise on concept learning. In *Machine Learning Volume II: An Artificial Intelligence Approach*, 1986.
- [22] Quinlan, J. R. Simplifying decision trees. *Int. J. Man-Mach. Stud.*, 27(3):221–234, 1987.
- [23] Quinlan, J. R. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.
- [24] Roy, S. Rapport de Projet de Fin d'Étude. Unpublished, 2003.
- [25] Salzberg, S. A Nearest Hyperrectangle Learning Method. *Machine Learning*, 6(3):251–276, 1991.
- [26] Schlimmer, J. C. *Concept acquisition through representational adjustment*. PhD thesis, Department of Information and Computer Science, University of California, Irvine, CA., 1987.
- [27] Tan, M. and Eshelman, L. Using Weighted Networks to Represent Classification Knowledge in Noisy Domains. In *5th International Conference on Machine Learning*, pages 121–134, Ann Arbor, Michigan, 1988. Morgan Kaufmann.
- [28] Ting, K. M. The problem of small disjuncts: its remedy in decision trees. In *Proc. 10th Canadian Conf. Artificial Intelligence*, pages 91–97, 1994.
- [29] Weiss, G. M. and Hirsh, H. The Problem with Noise and Small Disjuncts. In *ICML*, pages 574–578, 1998.
- [30] Weiss, G. M. and Hirsh, H. A Quantitative Study of Small Disjuncts. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 665–670. AAAI Press / The MIT Press, 2000.

- [31] Wettshereck, D. and Dietterich, T. G. An Experimental Comparison of the Nearest-Neighbor and Nearest-Hyperrectangle Algorithms. *Machine Learning*, 19(1):5–27, 1995.
- [32] Wilson, D. R. *Advances in Instance-Based Learning Algorithms*. PhD thesis, Brigham Young University, 1995.
- [33] Witten, I. H. and Frank, E. *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, 2000.
- [34] Wolberg, W. H. and Mangasarian, O. L. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. In *Proceedings of the National Academy of Sciences, U.S.A.*, volume 87, pages 9193–9196, December 1990.
- [35] Zhang, J. Selecting typical instances in instance-based learning. In *In Proceedings of the Ninth International Machine Learning Conference*, pages 470–479, Aberdeen, Scotland, 1992. Morgan Kaufmann.

## Appendix A

# Changes from original NNGE implementation

Database	New	Original
Iris	<b>95.3</b>	<b>95.3</b>
LED-7*	<b>70.5</b>	70.2
LED-24*	60.2	<b>60.3</b>
Waveform-21	<b>74.8</b>	<b>74.8</b>
Waveform-40	<b>70.2</b>	<b>70.2</b>
Cleveland	<b>76.6</b>	75.9
Hungary	<b>81.6</b>	66.3
Voting	94.7	<b>95.6</b>
BcW	<b>93.3</b>	<b>93.3</b>
Diabetes	<b>72.5</b>	<b>72.5</b>
Hepatitis	82.6	<b>84.5</b>

Table A.1: Comparison of NNGE implementations, where the new implementation uses a differing missing attribute handling scheme (see section 5.2) and overlapping examples of the same class are allowed. (\* mean values. Refer table 5.9, from appendix B for the “New” column values and table A.2 for the “Original” column values.)

Database	Brent’s							mean	st.dev.	
LED-7	70.4	69.4	71.4	71.4	69.6	69.2	70.0	70.2	70.2	0.8
LED-24	59.6	60.6	60.6	61.0	60.4	60.0	60.0	60.0	60.3	0.5

Table A.2:

## Appendix B

### Non-weighted $k$ -NNGE results

Database	$k$		mean	st.dev.
LED-7	1	70.8 71.4 69.8 71.8 69.6 70.6 71.2 69.0	70.5	1.0
	3	68.0 68.0 68.4 68.8 67.8 68.2 67.4 68.6	68.2	0.5
	5	68.4 66.0 67.0 67.8 67.8 68.8 69.4 68.8	68.0	1.1
	7	67.4 66.2 66.0 63.2 66.2 65.8 64.0 67.4	65.8	1.5
	9	66.0 66.2 66.0 66.4 67.0 66.8 67.0 66.4	66.5	0.4
	15	65.0 67.6 68.2 67.6 66.0 68.0 67.2 68.4	67.3	1.2
LED-24	1	60.4 59.2 60.6 61.0 60.0 60.6 59.8 59.8	60.2	0.6
	3	62.2 62.0 61.6 61.2 61.6 61.8 61.6 61.6	61.7	0.3
	5	60.4 60.8 60.4 60.4 59.8 60.4 60.8 60.6	60.5	0.3
	7	56.6 55.4 56.2 56.0 56.6 56.4 56.0 56.0	56.2	0.4
	9	56.4 57.0 56.2 56.4 56.8 57.0 57.4 56.0	56.7	0.5
	15	56.8 57.8 57.8 57.0 57.4 58.4 57.0 57.8	57.5	0.5

Table B.1: Classification performance of non-weighted  $k$ -NNGE for arbitrarily different  $k$  values

## Appendix C

### Weighted $k$ -NNGE results

Database	$k$		mean	st.dev.
LED-7	1	use same as non-weighted $k$ -NNGE		
	5	65.6 65.8 68.0 66.8 66.4 63.4 64.2 68.8	66.1	1.8
	9	66.6 66.0 69.2 65.6 64.8 67.2 63.8 66.8	66.3	1.6
	15	63.6 66.6 66.8 68.6 66.2 65.8 67.0 65.6	66.3	1.4
	21	68.0 63.2 66.1 65.6 66.4 68.0 66.0 62.8	65.8	1.9
	27	64.6 66.2 67.4 68.2 65.8 68.4 66.6 67.0	66.8	1.3
LED-24	1	use same as non-weighted $k$ -NNGE		
	5	59.4 59.4 59.8 60.4 60.0 59.6 60.4 59.4	59.8	0.4
	9	59.2 58.8 59.0 58.8 58.8 58.8 59.4 59.4	59.0	0.3
	15	60.4 61.0 60.2 60.6 60.4 60.4 60.6 60.6	60.5	0.2
	21	62.6 61.4 62.4 62.6 61.6 61.6 62.4 61.4	62.0	0.5
	27	58.2 58.6 57.6 57.6 57.8 57.8 58.2 57.6	57.9	0.4

Table C.1: Classification performance of weighted  $k$ -NNGE for arbitrarily different  $k$  values

## Appendix D

### IBk results

Default WEKA settings. Change on  $k$  only.

Database	$k = 1$	3	5	7	9	15	using $k$ obtained from x-validation
Iris	95.3	95.3	95.3	<b>96.0</b>	95.3	95.3	96.7 (6)
LED-7	76.8	<b>78.6</b>	77.2	76.6	76.6	77.0	78.6 (3)
LED-24	45.6	52.0	57.6	60.2	61.6	<b>64.0</b>	65.0 (27)
Waveform-21	78.4	81.8	81.4	82.4	<b>83.2</b>	<b>83.2</b>	86.6 (46)
Waveform-40	67.4	71.2	76.4	76.2	80.4	<b>81.2</b>	83.0 (28)
Cleveland	77.6	82.5	<b>85.8</b>	83.2	82.3	81.5	85.8 (5)
Hungary	78.9	82.3	82.0	82.3	<b>83.3</b>	83.0	83.7 (41)
Voting	92.6	<b>93.1</b>	<b>93.1</b>	<b>93.1</b>	92.9	92.4	93.3 (4)
BcW	<b>95.9</b>	95.4	95.3	94.8	94.4	93.7	95.9 (1)
Diabetes	69.8	74.2	73.2	<b>74.7</b>	73.6	<b>74.7</b>	75.9 (33)
Hepatitis	81.3	81.3	<b>85.8</b>	85.2	84.5	82.6	85.8 (5)

Table D.1: Classification performance of IBk for arbitrarily different  $k$  values

## Appendix E

### NNGE-S Results

Database		mean	st.dev.
Iris	96.0 95.3 94.7 95.3 96.0 96.0 94.7 97.3	95.7	0.9
LED-7	69.8 71.0 69.8 69.8 68.4 70.0 69.0 70.8	69.8	0.9
LED-24	60.6 62.0 61.4 61.2 60.8 63.2 61.8 62.6	61.7	0.9
Waveform-21	73.6 75.0 73.4 73.2 73.2 73.2 71.2 71.0	73.0	1.3
Waveform-40	72.6 72.6 73.4 72.4 72.4 72.6 72.0 70.6	72.3	0.8
Cleveland	76.7 77.9 75.9 75.2 75.9 75.2 77.2 80.9	76.9	1.9
Hungary	80.3 82.3 81.0 82.0 82.0 82.0 82.7 79.6	81.5	1.1
Voting	95.4 95.2 96.1 96.3 95.9 95.6 95.2 96.3	95.8	0.5
BcW	94.0 94.0 93.7 93.8 94.0 93.8 93.1 94.3	93.8	0.4
Diabetes	72.5 70.2 73.0 71.7 71.0 71.4 72.8 71.5	71.8	1.0
Hepatitis	81.3 83.2 81.9 83.2 82.6 82.6 83.2 81.3	82.4	0.8

Table E.1: Classification performance of NNGE-S ( $u = 0.95$ )