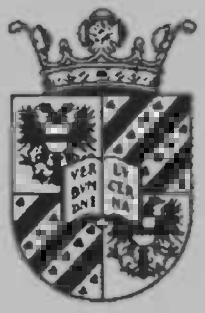
Handling Large Data Files

A Deterministic Approach

by

Michel Jaring



Department of Computing Science
Rijksuniversiteit Groningen
Groningen, The Netherlands
March 1999

A thesis submitted in fulfillment of the requirements
for the degree of Master of Science at the
Rijksuniversiteit Groningen

Supervised by J.A.G. Nijhuis

Abstract

Handling Large Data Files
A Deterministic Approach

Master of Science
Department of Computing Science
Rijksuniversiteit Groningen
March 1999

Induction systems have been successfully applied in a wide range of learning applications. However, they do not scale up to large scientific and business data sets. Applying a large training set (e.g., one million patterns) to a learning algorithm will result in:

- An excessive amount of training time;
- The inability to address the training set.

This thesis presents a feasible solution to the problems generated by the limited amount of the resources time (e.g., training time) and space (e.g., main memory). Both problems have a joint cause, a too large data set (e.g., a training set) is applied to an algorithm (e.g., a machine learning algorithm).

One problem occurs as a shortage of time, the other as a shortage of space. Generalizing both problems will yield a single problem, and a deterministic approach to this problem is necessary to provide a convenient premise. In other words, the joint cause of both problems implies a joint solution which can be found by a deterministic approach to the matter.

The essence of the solution is a histogram of each dimension of the data space (the data space is defined by the data set). The histograms are equalized by using an operation closely related to histogram equalizing, namely bin (bar) equalizing. By combining all histograms into a single data structure, a so-called mirror image of the data set is acquired. The mirror image provides information on the data set, and its resolution or accuracy depends on the number of bins of the histograms of which it is composed.

An equalized histogram of a specific dimension can be interpreted as an intersection of the data space. This intersection provides information on the dimension at issue, it does not provide information on other dimensions, i.e., a single intersection is one-dimensional. The mirror image combines the intersections, and, as a result, it does provide information on all dimensions of the data space. The mirror image is a small sized structure which efficiently provides information on the data set.

Each record in the data set defines a data point in the data space at a specific location. By verifying the location by means of the mirror image (one record a time), a record is either copied into a reduced data set (i.e., the sample set) set or is rejected. In other words, a record is either suitable or not suitable (i.e., it can or it cannot provide useful information to the sample set). This process is called:

- Deterministic sampling.

If a record has be to retrieved from a data set, the same process can be maintained. The only difference is the source of the properties of a record. The properties are now supplied by, e.g., the learning algorithm and not by the record itself. Addressing by means of the mirror image is virtually similar to deterministic sampling, and it is therefore denominated:

- Deterministic addressing.

Except for their premise, deterministic sampling and deterministic addressing do not differ. After all, both resource related problems have a joint cause, and a joint cause implies a joint solution.

Contents

Chapter 1 Introduction

1.1 Induction Systems

The method often used in the field of machine learning is to encode the knowledge of human specialists into a computer program. A so-called expert system, which uses some specific data set for tuning and testing. This method is costly and time consuming, because a programmer has to interpret the specific knowledge of the specialist to encode it.

The part of encoding the knowledge can be considered as 'learning' from the examples provided by the specialist. The programmer is an intermediate between the expert and the actual computer program. Replacing the programmer by a computer program which performs his task as an intermediate will result in an induction system (a system which generates general rules from specific facts) [3]. A popular family of induction programs represent the classifier they produce in the form of a decision tree.

Additional programs exist which convert decision trees into production rules like an `if-then-else` statement. The same knowledge is used, but the representation is different. For small data sets, decision trees and rules are easy to produce and to understand by humans. However, they have a limited amount of freedom to fit the model they represent to the data [3]. For specific tasks (particularly when large data sets are involved), the generalization of the presented data is limited, as well as the discrimination power [6][11]. The accuracy of the classifier is restricted and therefore not always the right choice to solve a specific problem. There is an approach which is able to solve such problems, namely neural networks.

A neural network is another example of an induction system, and is mostly described by connectionism. Connectionism is the study of a certain class of massively parallel architectures for artificial intelligence [3]. By massively interconnecting very simple so-called neurons, artificial neural networks attempt to mimic the computational power of the mammalian brain. The human brain consists of approximately 10^{11} neurons and each with an average of $10^3 - 10^4$ connections. The immense computing power of the brain is said to be the result of the parallel and distributed computing performed by these neurons [18]. The design of massively interconnecting simple units has provided models which have proved to be successful in a number of applications and in various fields (e.g., text to speech conversion, protein structure analysis, autonomous navigation, game playing, character recognition (including handwriting), image and signal processing, etc.) [14][18].

Neural networks tend to learn the target concept better than commonly used data mining methods [6]. They also have been successfully in terms of their learning ability, high discrimination power and excellent generalization ability [26]. Nevertheless, they have their limitations which make them poorly suited to tasks which make use of large data sets (particularly data mining and data warehousing tasks). Training times are often excessive, and the training set does not fit into main memory [24][2]. Whether the data set is small or large, one would like to use the advantages of neural networks, run the models fast and generate useful results in real time [26][17].

1.2 Brief Overview of Neural Networks

Neural networks (NNs) can be thought of as a nonlinear model which accepts inputs and produces outputs. NNs consist of processing elements, the neurons, and weighted connections. The network is composed of several layers, and each layer contains a number of neurons. Each neuron collects the values from all of its input connections, and performs a predefined mathematical operation to produce a single output value.

The value of the weights is often determined by a learning procedure, although sometimes they are predefined and hardwired into the network. The adjustment of the connection weights enables the NN to store a generalization of the applied training set.

A neuron itself processes information by means of three basic elements [14]:

- A set of connecting links, each of which is characterized by a weight;
- An adder to sum the input signals weighted by the respective links of the neuron;
- An activation function to limit the amplitude of the output of the neuron.

A model of a neuron is shown in Figure 1.1. Neurons are usually nonlinear due to a nonlinear activation function. In mathematical terms, a neuron k is described by

$$u_k = \sum_{j=1}^{p} w_{kj} x_j, \tag{1.1}$$

and

$$y_k = \varphi(u_k). \tag{1.2}$$

where $x_1, x_2, \cdots, x_p$ are the input signals, $w_{k1}, w_{k2}, \cdots, w_{kp}$ are the weights of neuron k, u_k is the output of the summing junction (linear combiner output), and y_k is the output signal of the neuron [14].

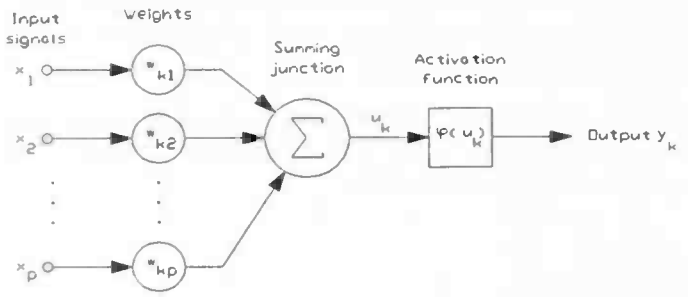


Figure 1.1 Model of a neuron.

There are several important features which apply to all NNs [25]:

- Each neuron acts independently of all other neurons, and the output of a neuron relies only on its constantly available inputs from abutting connections;
- Each neuron relies only on local information, it does not require the state of any of the other neurons where it does not have an explicit connection with;
- The large number of connections provides redundancy, and facilitates a distributed representation.

Learning algorithms can be divided into two classes, supervised and unsupervised. The class of supervised learning algorithms provides the NN with a training vector, sometimes referred to as a pattern, and the desired or target response for that training vector. A collection of training vectors is a so-called training set.

The most widely used supervised learning algorithm is the back-propagation algorithm. This learning algorithm makes use of two distinct phases, namely the forward phase and the backward phase. In the forward phase, the signals propagate through the network layer by layer, eventually producing some response at the output of the network (the weights of the network are all fixed). The actual response of the network is subtracted from the target response to produce an error signal. This error signal is then propagated backward through the network against the direction of the connections (error signals are propagated backwards in comparison with function signals). Hence the name back-propagation algorithm.

The weights are adjusted according to the error-correction learning rule to make the actual response of the network to move closer to the desired response. The purpose of the error-correction learning rule is to minimize a cost function based on the error signal in such a manner that the actual response of each output neuron approaches the target response for that neuron in some statistical sense [14].

Figure 1.2 illustrates a NN which consists of 2 source nodes, 4 computation nodes in the hidden layer, and 2 computation nodes in the output layer (a so-called 2-4-2 network).

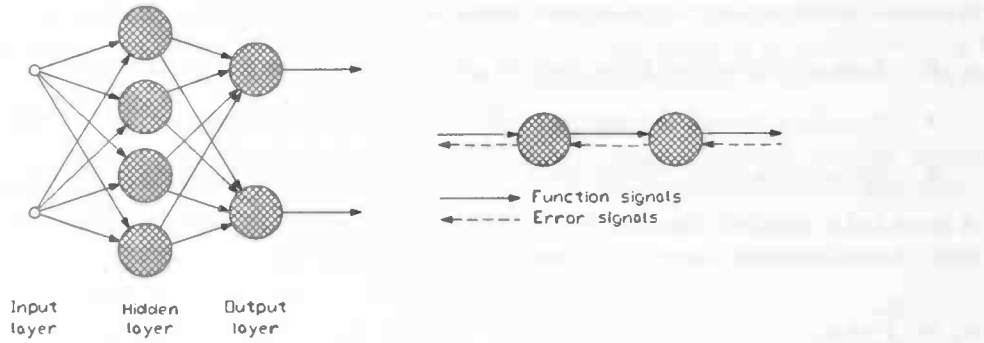


Figure 1.2 A 2–4–2 neural network (for the sake of clarity, the network does not show the dashed arrows).

An unsupervised learning algorithm omits the target response, and a task independent measure of the quality of the representation of the network is required to learn. The weights of the network are optimized with respect to that measure. The network develops the ability to form internal representations for encoding features of the input data [14]. Supervised learning is like learning how to drive a car with the assistance of an instructor, and unsupervised learning is like a baby learning how to crawl around.

A multilayer perceptron (MLP) is a commonly used class of NNs which consists of a set of source nodes which constitute the input layer, one or more hidden layers of computation nodes, and an output layer of computation nodes. The network in Figure 1.2 is a MLP.

This thesis does not provide an introduction to NNs, but instead refers the interested reader to one of the good textbooks or papers in the field (e.g., *Neural Networks, A Comprehensive Foundation* by Haykin [14] or the *Foundations of Neural Networks* by Simpson [25]).

1.3 Resources

1.3.1 Introduction

NNs have been successfully applied by engineers and scientists in various fields. However, the field of NNs is highly interdisciplinary, and each approach has viewpoints on topics concerning how NNs should be put into practice. These viewpoints have a common characteristic, the training set which is used to train the NN has to be small [6][14][26][8]. As a consequence, the domain of NN applications is limited. This is particularly true if the relevance of an input feature depends on the value of other input features (a feature is derived from one or more (raw) patterns and emphasizes a specific property) [24].

The meaning of the phrase "large training set" changes as fast as the hardware does. About a decade ago it meant hundreds or thousands of patterns [3]. Nowadays (at least in this thesis) it means hundreds of thousands or even millions of patterns, and in the nearby future probably billions. In fact, the size of the training set reflects the available hardware, and it should be considered accordingly, namely relatively and comparatively.

An algorithm basically needs two resources, namely time (e.g., training time) and space (e.g., main memory). If the demand for at least one of these resources exceeds a threshold, the algorithm cannot properly produce its results. It may seem that the resource time is unlimited, but the results have to be produced within a reasonable amount of time (i.e., the results still have to be useful).

1.3.2 Demand for Time

Learning time, sometimes referred to as learning speed, is an important practical consideration if it grows beyond minutes to days or worse. Figure 1.3 shows a typical example of the nonlinear increase of the learning time to create a decision tree [3]. The training sets originated from NASA its Space Shuttle to diagnose a subsystem of it (the space radiators). Seven classes represent the possible states of the radiators, and the attributes comprise nine measurements from three sensors. Extra information becomes available due to the increased size of the training set, and the tree is allowed to grow to store this extra information [3].

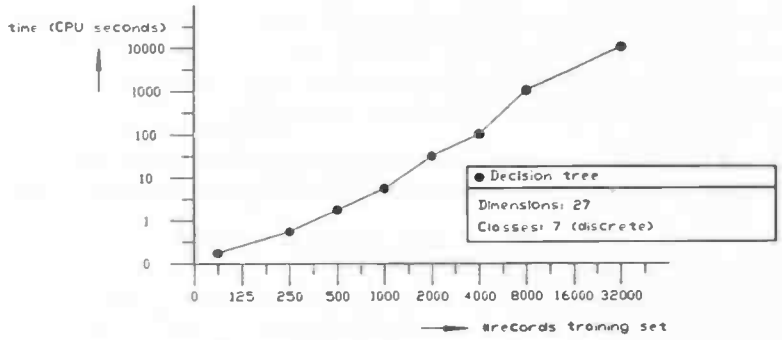


Figure 1.3 Nonlinear increase of the learning time of a decision tree.

The best way to indicate the training time or computational complexity of a NN is the number of connection traversals [21]. Figure 1.4 demonstrates the increase of connections between neurons when the size of a NN is growing. The figure is obtained by constantly adding two neurons to each hidden layer of a NN which initially is a 27–1–1–7 network.

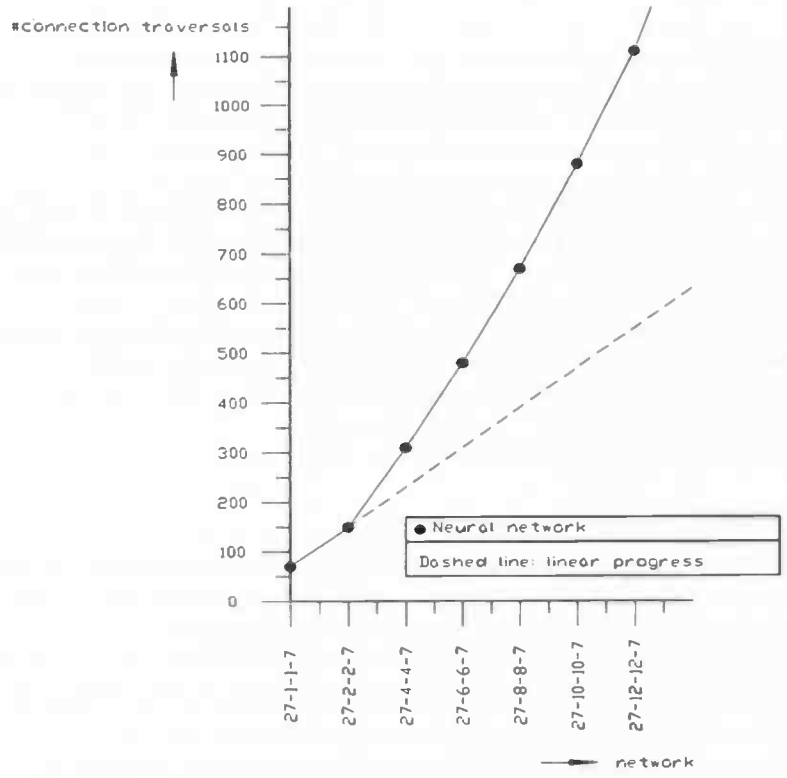


Figure 1.4 Nonlinear increase of the number of connection traversals of a NN.

Let L be the number of computation layers of a multilayer NN. The effect of a weight in the first layer on the output of the network depends on its interactions with approximately F_i^L other weights, where F_i is the so-called fan-in, the average number of incoming links of neurons. When the size of the network increases, the network becomes more computationally intensive, and the learning time will nonlinearly grow [14].

Decision trees and NNs have the same scaling behavior (both nonlinear), but there is an important difference. With respect to the same training set, it usually takes extra time to train a NN than to build a decision tree. This is due to the forward and backward phase of the supervised learning algorithm. Each neuron (node) of a NN is visited at least two times for each pattern applied to the network. However, a decision tree can be traversed by using comparisons, thereby skipping irrelevant parts of the tree (unless it has to be balanced again). Besides the number of node visits, NNs learn the (classification) rules by multiple passes over the training set, as contrasted to the single pass of decision trees [17].

When the size of the training set increases linearly, a NN requires extra time with respect to the decision tree in a nonlinear fashion. Training a NN with the training set of Figure 1.3 will result in an initial learning time which is several times the magnitude of the initial learning time of a decision tree (e.g., four times). If the premise is learning time, a decision tree is a better choice than a NN.

If it is assumed that a NN (back-propagation learning algorithm, initial network is 27-10-7) is trained by means of the same environment which was used to obtain Figure 1.3, then Figure 1.5 can be deduced.

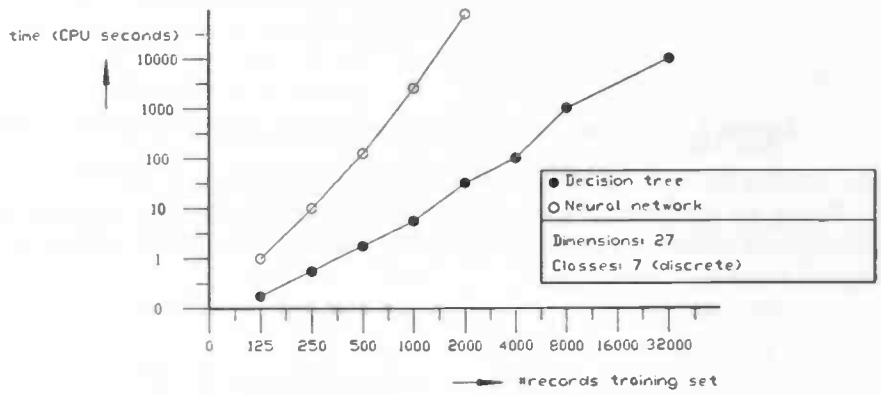


Figure 1.5 The learning time of a NN is not proportional to the learning time of a decision tree.

1.3.3 Demand for Space

Difficulties also appear whilst addressing a large training set, because most learning algorithms require that the entire training set or a portion of it permanently remain in main memory [24]. A large training set will probably exceed the available amount of main memory. Standard techniques to address a large training set are available (e.g., hashing, B-trees, B^+-trees, etc.) [9]. However, these techniques have a common drawback. They require a specific structure which is independent of the remaining part of the (learning) algorithm. In other words, when a large data set is being sampled by means of some specific structure, then the same structure should be (reversely) used to retrieve a specific record from this data set. It should be two-way traffic instead of one-way traffic.

1.4 Rationale

Whilst training an induction system, the demand for the resource time should be minimized, and, regardless of the resource space, addressing a training set should always be possible. These problems emanate directly or indirectly from the insufficiency of the hardware (CPU and main memory), and the only way to overcome them, within certain limits, is to use expensive hardware [10][22]. Nevertheless, at this moment, the every year growth of available data exceeds the development of hardware in that same year. Therefore, the problem is stated as follows:

- *The limited amount of the resources time and space will generate problems when a too large data file is applied to an algorithm (e.g., a machine learning algorithm). Research techniques and methods to find a meaningful solution to these problems.*

NNs and the back-propagation learning algorithm are the premise, and throughout this thesis they will serve as an example to elucidate the theory. However, the theory also applies to machine learning algorithms in general, and any other algorithm which has to process (too) large data sets.

1.5 Precis of Thesis

The second chapter investigates the stated problem. It also provides, based on the investigation, a survey of techniques relevant to the reduction of large data sets. Chapter three enunciates a feasible solution (algorithm) to the problem. Then, in chapter four, the behavior of the algorithm is examined and explained. In the same chapter, the results of a field-test are presented to determine whether or not the solution is an applicable one. The next and final chapter concludes with a list of advantages and disadvantages of the solution.

Chapter 2 Survey

2.1 Investigating the Problem

2.1.1 Introduction

The only way to reduce the demand for the resources time and space is by adjusting the size of the training set. One could argue that by optimizing the learning algorithm in some specific way, the demand for the resources is also reduced. This is a correct statement if a learning algorithm is not yet optimized, but it does not hold in general. Throughout this thesis it is assumed that each aspect of the learning algorithm is already optimized. This assumption is easily supported by the fact that even if a learning algorithm is maximally efficient, it is still nonlinear with respect to the resource time and linear with respect to the resource space.

These lower bounds do not change, no matter how well the learning algorithm is optimized. However, this does not imply that the learning algorithm (e.g., back-propagation) cannot be improved [10][28][30]. A third order learning time is an improvement in comparison with a fourth order learning time, but it does not change the nature of the problem. The time consuming training process also inhibits an exhaustive exploration of alternative network design [29].

Large data sets tend to be redundant [3][20][23]. Removing the redundancy results in a smaller data set, a so-called sample set. An optimal sample set contains the same amount of information as the original data set without any redundancy. This is a theoretical optimum, and it does not necessarily imply an attainable one. If a data set comprises more than one dimension, it is not always possible to remove all redundancy or to preserve all information. However, at least one sample set of a specific size always exists in such a manner that it is optimized with respect to the data set [3]. Applying such a sample set to a learning algorithm will limit the demand for the resources time and space, making no concessions to the accuracy of the resulting network within certain limits of the reduction of the training set.

A clear distinction should be made between compression and reduction. Compression transforms the data set into another form by encoding, but in such a manner that the sample set can be decoded into the exact original. No information is lost in the process, but the intermediate form (the sample set) is more compact. A reduction of a data set removes irrelevant information from the data set to produce a sample set, but in an irreversible manner. Whether or not information is irrelevant depends on the processing of the information by the receiver (e.g., a learning algorithm) [1].

The sample set referred to in this thesis cannot be entirely decoded into the original data set, i.e., not always. By claiming that a sample set is a compressed data set, the data set can always be entirely recovered. Therefore, a sample set is a reduced data set and not a compressed data set.

2.1.2 Generalizing Time and Space

Two difficulties related to time and space appear whilst sampling a data set. The demand for both resources has to be linear with respect to the size of the data set. A nonlinear sampling algorithm may produce a suitable sample set, but it shifts the problem from an excessive training time to an excessive sampling time. Therefore, the selection algorithm must display linear time behavior with respect to the size of the data set. Another problem is the size of the data set, it probably does not fit into main memory. This problem is identical to the problem which occurs if a large data set is presented to a learning algorithm. The need to permanently store the entire data set or a portion of it into main memory whilst sampling must be circumvented.

The limited amount of the resources time and space seem to generate different problems. Seem to, because their only difference is their occurrence. One can interpret time as space, a 'box' of one dimension. An excessive amount of training time does not 'fit' into the box. Space can be interpreted as time, i.e., three 'lines' that represent time (or two or one, that depends on the number of dimensions), each of which ticking out their (CPU) seconds. The available amount of main memory can be regarded as a one-dimensional space in which, e.g., the training set has to fit into.

The previous results in five lines (dimensions) of some specific 'length'. Each of these lines can be interpreted as a specific amount of time. The underlying idea is to represent time and space by their components (the di-

mensions) of which they are composed. The intention is to generalize time and space through which the resource related problems merge into a single problem.

If a large training set exhausts the resources time and / or space, the length of at least one dimension is exceeded. Time and space seem different, but they are one and interchangeable. It is exactly like a space of two dimensions which can be replaced by two spaces of one dimension. Therefore, if a large training set is applied to a learning algorithm, the limited resources time and space will generate the same problem and not two different problems. Figure 2.1 supports the previous.

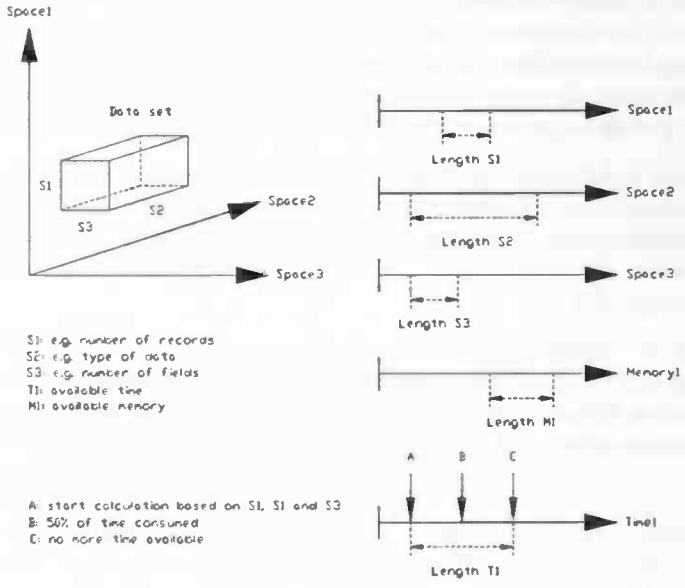


Figure 2.1 Interchanging time and space. S2 does not 'fit' into T1 (the calculation based on S2 is too lengthy), and the data set does not fit into main memory (the combined 'length' of S1, S2 and S3 exceeds the length of M1).

By solving the problem related to the resource time, the problem related to the resource space is also solved in a indirect manner. The problem is focussed on the resource time, and referring to the resource time is indirectly referring to the resource space. On that account, the premise is sampling a data set, and not addressing a data set (the training time can be reduced by applying a smaller training set).

2.1.3 Subproblems

Sampling a data set implies two subproblems:

- How to maximize the information contents of a sample set?
- How to minimize the redundancy of a sample set?

The demand for training time is nonlinearly related to the size of the training set as opposed to the linear demand for main memory. However, the demand for sampling time should linearly increase with respect to the size of the data set. Otherwise the problem is shifted from the learning algorithm to the sampling algorithm. As a consequence, the complexity of the sampling algorithm has to be linear.

The semantics of data set can be very diverse. If it is possible to comprehend the semantics, then there is no reason to use a learning algorithm. A data set is considered to be a collection of bits and bytes, representing values that define an unknown structure. Each record in the data set is a combination of values which defines a data point. All data points can be compared with a single valued pixel of an image (i.e., the pixels denoted by a record are, e.g., white and the remaining pixels black). It is not necessary to look at an image to understand the structure of it in a mathematical way. A similar approach to the sampling algorithm ensures that the selection process does not depend on some interpretation of the data set. It also ensures independence of the type of fields of the data set. The type of a field is either continuous (e.g., 1.23) or discrete (e.g., either male or female). In other words, the observation of the data set is generalized to maximize the flexibility of the sampling algorithm.

The demand for time and space is minimized if the common case is generalized and the uncommon case specialized. Common cases add redundancy to a data set, and uncommon cases add information. A small percentage of a data set can be uncommon when the entire data set is considered, but common for some specific state (e.g., a query). An example will clarify this. A bank its database holds records which stores the monthly income of clients. Let 99% of the clients have an average income of fl 5000,– or less per month, and the remaining part of 1% more than fl 5000,– per month. In general, the 1% group is considered to be uncommon in relation to the 99% group. This difference is not as obvious as it may seem, it even is very subtle.

A member of the set of uncommon cases is considered common within this same set, e.g., if the bank wants to mail all the clients who earn more than fl 5000,– per month, then a client in the 1% group is considered as being common within this group. This results in two more subproblems:

- When is a data point common?
- When is a data point uncommon?

2.1.4 Multiple Field Relations

Each field (a record is composed of fields) in the sample set should come from the same distribution as the corresponding field in the data set. Hypothesis tests are usually designed to minimize the probability of falsely claiming that the two distributions are different. A univariate hypothesis test provides no guarantee that the bivariate hypothesis is correct. One could as well run bivariate tests, but then there is no guarantee that the trivariate statistics will be correct [15][20]. Table 2.1 and Table 2.2 illustrate that all dimensions have to be considered in relation to each other instead of one by one (i.e., a trivariate test). The type of the test is determined by the number of fields of a data set.

DATA SET				
	U	V	W	
U	50	50	50	150
V	50	50	50	150
W	50	50	50	150
	150	150	150	

Table 2.1 Counts of a data set containing 50 copies of records <UU>, <UV>, <UW>, <VU>, <VV>, <VW>, <WU>, <WV> and <WW>.

SAMPLE SET				
	U	V	W	
U	75	0	0	75
V	0	75	0	75
W	0	0	75	75
	75	75	75	

Table 2.2 Counts of a sample set containing 75 copies of records <UU>, <VV> and <WW>. To univariate sampling this is a suitable sample set, but not according to bivariate and trivariate sampling.

2.2 Static Sampling

The aim of static sampling is to determine whether a sample set is sufficiently similar to the entire data set. The criteria are static in the sense that they are independently used of the following analysis to be performed on the sample set. A statistically valid sample set implies that the sample set and the data set come from the same distribution.

Hypothesis tests are usually designed to minimize the probability of falsely claiming that two distributions are different. In a so–called 95% level hypothesis test there is a 5% chance that the test will incorrectly reject the hypothesis that the distributions are the same, assuming that the two samples do come from the same distribution. The probability of falsely claiming that they have the same distribution has to be minimized [15].

Static sampling runs the appropriate hypothesis on each of the fields. If it accepts all of the hypotheses, then it claims that the sample set does indeed come from the same distribution as the data set, and it reports the current sample as sufficient.

There are several shortcomings to the static sampling model. When running several hypothesis tests, the probability that at least one hypothesis is wrongly accepted increases with the number of tests. Static sampling is an attempt to answer the question "Is this sample good enough?", but first asking "What is the purpose of the sample set?" seems to be more appropriate. Static sampling takes no notice of this question. In other words, the tool which will be used after sampling (e.g., a learning algorithm) is ignored.

Static sampling is a sampling procedure which continues sampling the data set until a suitable sample set is obtained, and it is often unclear how the setting of the levels of the hypothesis tests will affect the size and the contents of a sample set [15][20]. These are the two main drawbacks of static sampling.

2.3 Dynamic Sampling

Sampling a database involves a decision about a tradeoff. The decision is how much to give up in accuracy to obtain a decrease in running time of, e.g., a learning algorithm. Dynamic sampling, sometimes referred to as active sampling, will address this decision directly, instead of indirectly looking at statistical properties of sample sets independent of how they will be used [15].

As opposed to static sampling, dynamic sampling takes the tool that will process the sample set into account. Determining whether the sample is good enough is based upon the purpose of the sample set (e.g., providing a maximum amount of information to a learning algorithm). Dynamic sampling uses advance knowledge on the behavior of the learning algorithm in order to choose a sample set. The test of whether or not a sample set is suitably representative depends on how the sample will be used. A static sampling method does not use this kind of information, and instead applies a fixed criterion to the sample to determine whether or not it is a suitable representation of the data set [7][15]. However, obtaining the advance knowledge is quite difficult due to the nonlinear behavior of most tools.

A sample set which is obtained by dynamically sampling a data set needs to be evaluated, e.g., by using the Probably Close Enough (PCE) criterion. If it is assumed that the performance of the learning algorithm on a sample is probably close enough to what it would be when the entire data set was applied, then the sample set is satisfactory. One would like to have the smallest sample set of size n such that

$$P(acc(N) - acc(n) > \epsilon) \leq \delta, \tag{2.1}$$

where $acc(n)$ refers to the accuracy of the output of the tool after applying a sample of size n to it, $acc(N)$ refers to the accuracy after applying the entire data set, ϵ is a parameter describing the phrase "close enough", and δ is a parameter describing the phrase "probably" [15].

Quantifying ϵ and δ is difficult, and it implies a sampling procedure that continues sampling the data set until a suitable sample set is obtained. It is often unclear how quantifying these parameters will affect the size and the contents of a sample set. These same problems occurred in case of static sampling.

Both static and dynamic sampling commonly use some type of random sampling technique to select records from the data set. Static and / or dynamic tests are performed after randomly selecting the records to determine whether or not the sample set is a suitable one.

2.4 Random Sampling

Random sampling is a fundamental operation having many applications in science and industry. In general, the problem is to draw a random sample set of size n without replacement from a file containing N records. The n records usually appear in the same order as they do in the data set. The sample set size n is generally

small relative to the data set size N [3][15][20][27]. In other words, simple random sampling involves, among others, forming a random sample set of n records from $\{1, 2, ..., N\}$.

Random sampling is typically used to support statistical analysis of a data set, either to estimate parameters of interest or for hypothesis testing. In the field of machine learning, random sampling is often used to create a subset of the original data set to extract specific data. It has proven to be a valuable tool, but random sampling generally confirms that a relatively large sample set is necessary to maximize accuracy [3].

The various types of random sampling can be classified according to the manner in which the sample size is determined [3][20]:

- Whether the sample is drawn with or without replacement;
- Whether the access pattern is random or sequential;
- Whether or not the size of the data set is known;
- Whether or not each record has a uniform inclusion probability.

The most common types of random sampling are [20]:

- Simple random sample without replacement (SRSWOR):

  Each record of the data set is equally likely to be included in the sample set. Duplicates are not allowed.

- Simple random sample with replacement (SRSWR):

  Equal to SRSWOR, but, due to the replacement, duplicates are allowed.

- Stratified random sample (SRS):

  The sample is obtained by partitioning the population into so-called strata (e.g., by gender), then taking a sample (usually SRSWOR) of specified size of each strata.

- Weighted random sample (WRS):

  The inclusion probabilities for each record of the data set are not uniform.

- Probability proportional to size (PPS):

  A weighted random sample without replacement in which the probability of inclusion of each record of the data set is proportional to the size of the record (e.g., age).

- Monetary Unit Sample (MUS):

  A weighted random sample generated by iteratively taking a sample of size 1 with inclusion probabilities proportional to the sizes of the records (typically monetary values) of the data set (the same as PPS, but duplicates are allowed now).

- Clustered sample (CS):

  Generated by first sampling a cluster unit of records (e.g., a disk page), and then sampling several records within the cluster unit.

- Systematic sample (SS):

  Obtained by taking every k-th element of a data set (the starting point is chosen at random).

A characteristic of random sampling is its dependence on a chance. After all, it is random.

Simple random sampling with replacement is part of the conducted field-test (see chapter four). This sampling technique consists of simple random samples (i.e., unweighted) with replacement drawn from a data set of known size stored on disk as fixed size records (i.e., fixed blocking). The sample set of size n can be obtained by generating uniformly distributed random numbers between 1 and N, and reading (random access) the corresponding records [20].

2.5 Compaction

Identical records which appear several times in a data set are redundant in the sense that only one instance is needed to store them into (or retrieve them from) the data set. There is no need to store identical records, but it then is necessary to store an extra variable per record, namely the weight. The weight indicates how often a specific record appears. Adding an extra weight to indicate how often a specific record appears is a commonly used technique to reduce the demand for the resource space [3].

A record comprises one or more fields. Some fields store a continuous value, others a discrete one. If a field of a record stores a continuous value, it hardly ever happens that two or more records are identical. Nevertheless, records which only store discrete values are often the same (particularly if the data set is large in comparison with the number of classes). Unfortunately, records often store both continuous and discrete values. This diminishes the possibility to use compaction in a general sense.

Compaction can be improved by not compacting records, but by compacting the fields of the records. Instead of thinking from left to right (compaction of records), one should think from top to bottom (compaction of fields). This approach 'splits' the fields into either discrete or continuous, and it therefore makes compaction independent of the different types of fields within a record.

2.6 Deterministic Sampling

A sampling algorithm selects records from the data set following some procedure. This procedure is either a random, deterministic, or a combination of a random and deterministic process. The sample set has to represent the data set as closely as possible, and, consequently, the relation between the common and uncommon data points must be preserved as accurately as possible.

Each data point is always part of at least one cluster, and the smallest possible cluster is a cluster of one data point. One would like to maximize the chance to select the correct records in such a manner that the different clusters are equally reduced.

An example will clarify the previous. A data set comprises two clusters of 5000 and 1000 data points respectively and 2000 entirely random data points. If the reduction of the data set is equal to 10, the sampling algorithm should select 500 data points from the first cluster, 100 from the second one, and 200 from the randomly distributed data points. To maximize the chance to properly select a suitable sample set, the sampling algorithm has to be deterministic. Sampling a data set randomly may produce a suitable data set, but the actual information contents of a specific sample set will deviate as contrasted to deterministic sampling.

The deterministic part of the deterministic sampling algorithm implies that some information on the data set is available or acquired. To decide whether or not a specific record is a suitable one is based on specific information. It is this decision that makes a sampling algorithm a deterministic sampling algorithm.

Since no advance knowledge is available, the information has to be acquired by investigating the data set. The investigation of the data set must provide information on the entire distribution. The process of acquiring information will increase the demand for time and space (the increase has to be linear for both resources, otherwise the problem is shifted from the learning algorithm to the sampling algorithm). The deterministic sampling algorithm should be able to perform the following tasks:

- Deciding whether or not a specific record is a suitable one (i.e., a useful addition to the sample set);
- Addressing specific records.

The first task is related to the resource time (reducing the training set in order to limit the training time), the latter to the resource space (addressing records to overcome the limited amount of main memory). Addressing a record can be based on the same information as deciding whether or not a record is a suitable one. As explained in the previous chapter, the resource related problems have a joint cause, and a joint cause implies a joint solution. Therefore, the underlying structure to perform these two tasks is (practically) identical.

Chapter 3 Deterministic Sampling

3.1 Introduction

The previous chapter introduced four subproblems:

- How to maximize the information contents of a sample set?
- How to minimize the redundancy of a sample set?
- When is a data point common?
- When is a data point uncommon?

These four subproblems imply that specific information on the data set is necessary to determine a suitable sample set. Information is obtained by either investigating the data set set or by asking an expert. The latter is not always possible, and the deterministic sampling algorithm should compensate the lack of advance knowledge. Therefore, it is assumed that no information is available in advance. If an expert provides valuable information, investigating the data set is not or only partially necessary, and it can therefore be entirely or partially skipped. This introduces the first part of the algorithm (see also Figure 3.1):

- Investigating the data set.

The entire data set is probably too large to fit it into main memory, and an accurate representation of it is necessary to determine a suitable sample set. A representation of the data set can be interpreted as a 'reflection' of the data set. Hence the name mirror image. Ideally, a mirror image contains a maximum amount of information, a minimum amount of redundancy, discriminates between a common and an uncommon data point, and uses a minimum amount of main memory. In fact, determining the mirror image is the most important part of the algorithm, and it constitutes the second part of the algorithm:

- Determining a representation of the data set, a mirror image.

Once the mirror image is available, it can be used used to determine a sample set in a completely deterministic manner. The mirror image provides a structure to ensure that the sample set represents the data set as accurately as possible. Therefore, the third and last part is:

- Determining a sample set.

The three parts of the algorithm should efficiently operate in linear time and space. Efficiently, because if it takes 10 hours and 10 MB of main memory to process a data set of 5 MB, then the process may be linear, but it certainly is not efficient.

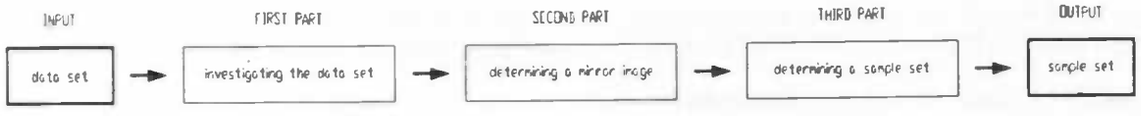


Figure 3.1 General picture of the deterministic sampling process.

This chapter provides an explanation and an overview of the three parts of the deterministic sampling algorithm. A running example will elucidate the theory.

3.2 Investigating the Data Set

3.2.1 Generalizing the Data Set

A data set comprises at least one record of at least one field. It may contain an arbitrary number of records, but each record must consist of the same number of fields. The data space defined by the data set is composed of one or more dimensions. A specific dimension of the data space is defined by a specific field of the data set. Referring to a dimension of the data space, is referring to a field of the data set. A field contains a specific

value which can be of any type (i.e., an integer, float, character, or a string of two or more characters). By adding the value of each character of a string, an integer based value is obtained (e.g., "male" produces 415).

The total number of records is denoted N and the total number of dimensions D. Let n be a specific record and d a specific dimension. A specific field is then addressed by (n,d), where $0 \leq n < N$ and $0 \leq d < D$. Figure 3.2 shows an example of a small data set.

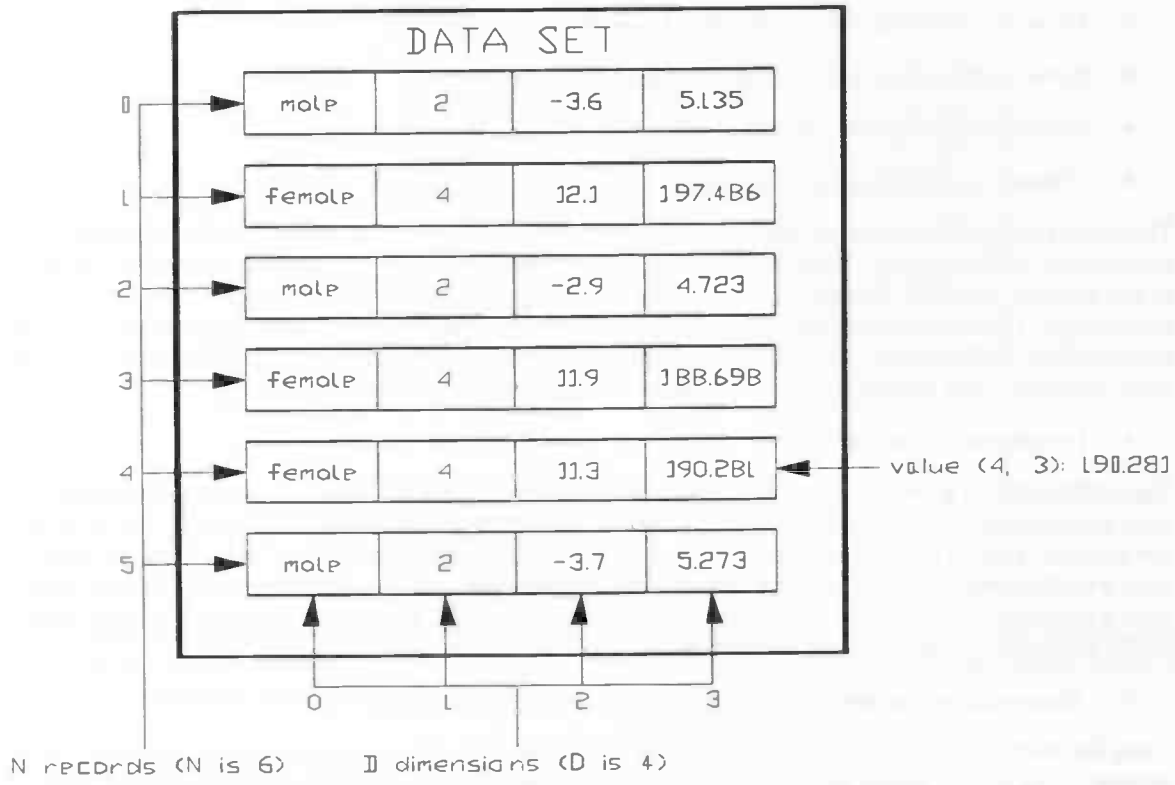


Figure 3.2 A data set which contains 6 records of 4 fields (dimensions) each.

In general, a disk is used for storing a data set. A disk contains concentric circles, the tracks. Each track stores the same amount of information, and is divided into disk blocks or pages. The block size is fixed during formatting, and cannot be changed dynamically. Typical disk block sizes range from 512 to 32768 bytes. For a read command, a block from disk is copied into a buffer, and for a write command the contents of the buffer are copied into the disk block. Sometimes several contiguous disk blocks, called a cluster, may be transferred as a unit [9]. By reading a specific record, a disk block storing a collection of records, among which the requested record, is copied into main memory. An exception occurs when the size of a record exceeds the size of a disk block, then the disk blocks which simultaneously store the record are copied into the buffer.

3.2.2 Establishing the Extremes

The records are addressed one by one, and each field of each record is read in order (from left to right and from top to bottom). The lowest and highest value of each dimension d are determined on the fly to produce $\min_d$ and $\max_d$. The range of each dimension d is denoted

$$R_d = \max_d - \min_d. \tag{3.1}$$

A specific number of bins, namely $\#bins_d$ is attributed to each dimension (the number of bins per dimension do not have to be equal). A range R_d is divided into subranges or intervals, denoted $range_{dj}$, where $0 \leq range_{dj} \leq R_d$. The indication j, $0 \leq j < \#bins_d$, specifies a subrange of dimension d. The resolution of the subranges of dimension d is directly related to the number of bins attributed to this specific dimension (the range R_d consists of $\#bins_d$ bins).

3.2.3 Mapping of Data Points

D arrays, A_0 through A_{D-1}, of length $\#bins_d$ are introduced. The k–th bin of dimension d is denoted by bin_{dk} and $k = 0, 1, \cdots, \#bins_d - 1$. A counter cnt_{dk} is introduced and added to each bin. The value of k indicates the k–th counter of dimension d.

By reading the data set for the second time, the field values of each dimension are linearly scaled (normalized) to a range of 0 to $\#bins_d$, and rounded off downwards to the nearest integer value, producing

$$i_d = \left\lfloor \frac{val_{nd} - \min_d}{\max_d - \min_d} * \#bin_d \right\rfloor ; \qquad i_d = 0, 1, \cdots, \#bins_d - 1, \tag{3.2}$$

where val_{nd} is the value of the n–th record of the d–th dimension to be scaled, and i_d an array index of array A_0 (it indicates the i_d–th bin of dimension d).

The array index is used to increase the i_d–th counter cnt_{di_d} of dimension d. Both the index of an array element and the element itself provide information on a specific subrange or bin. Figure 3.3 illustrates an example of a small data set after processing it according to the previous.

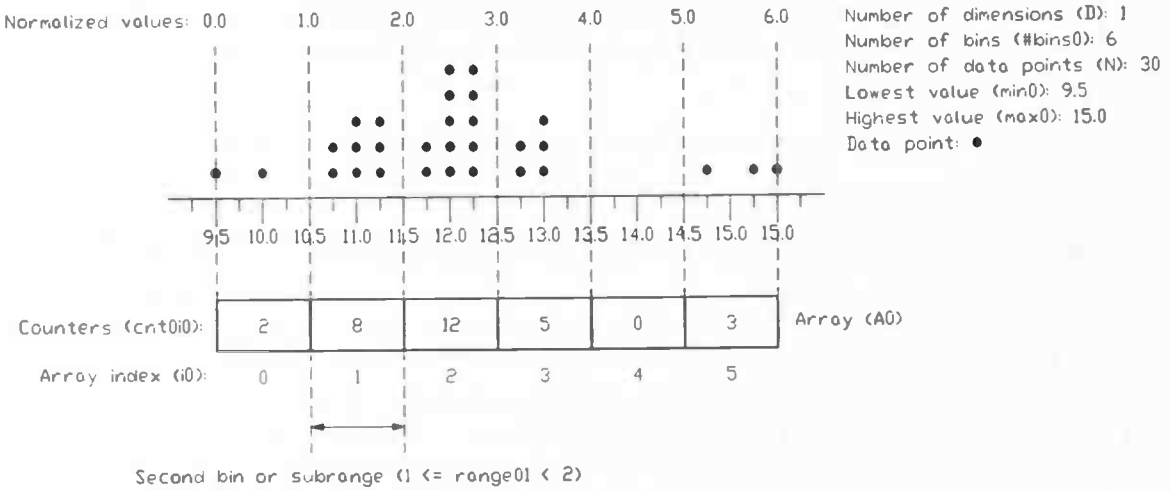


Figure 3.3 The index of an element and its contents defines a specific subrange or bin.

The value of each counter cnt_{di_d} of array A_d provides information on the distribution and clustering of dimension d of the data set. It does not provide information on other dimensions.

3.2.4 Examining the Bin Contents

Data points are practically always clustered. Without clustering, the data points are equally spreaded over the data space, providing no information other than the fact it is not clustered. By investigating the counters of both arrays in Figure 3.4, the large cluster can be recognized (the horizontal axis in Figure 3.4 is identical to the axis in Figure 3.3). The hatched area is the data space covered by the arrays A_0 and A_1. The values of the counters of A_0 indicate a cluster stretched from the second bin through the fourth bin. The counters of A_1 show that virtually all data points are positioned at the top of the data space. The values of the counters also discriminate between common and uncommon data points. Data points located at the top of the data space appear to be common, and the five data points at the bottom appear to be uncommon with respect to the cluster.

Each array provides a unique view on the data set. They describe the same data points, but from another perspective. The value of a counter cnt_{di_d} is related to the size of a cluster possibly present in the interval $range_{di_d}$. The value of the counter cnt_{12_1} of array A_1 in Figure 3.4 is 25, but there are only 30 data points. Therefore, at least one cluster is positioned somewhere in the interval of the last bin of A_1. The values of cnt_{01_0}, cnt_{02_0}, and cnt_{03_0} of array A_0 also imply a cluster. More exactly, they imply the same cluster. The only difference is the perception, because the cluster is observed from another dimension.

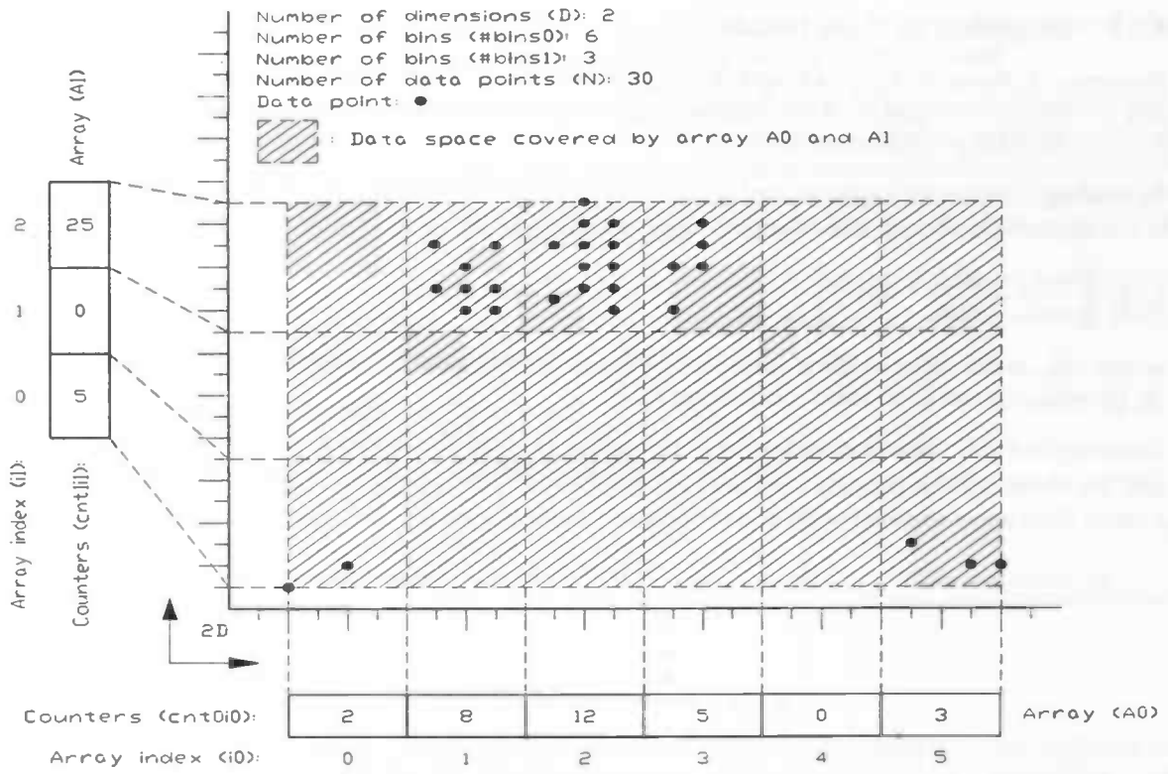


Figure 3.4 Counters indicate, to some degree, clustering and common / uncommon data points.

Each dimension is considered separately, but it seems that at least some information on the combination of dimensions is already present. By actually combining the arrays A_0 and A_1, the available amount of information will grow (information on the 'depth' of the data space becomes available).

3.3 Determining a Mirror Image

3.3.1 Histogram

A way to summarize a data distribution, one that has a long history in statistics, is to partition the range of the data into several intervals of equal length, count the number of data points in each interval, and plot the counts as bar lengths. The result is a histogram. The histogram is widely used, and familiar even to most non-technical people and without extensive explanation [5]. This is convenient, because each cnt_{di_d} of array A_d represents the i_d–thbar length in a histogram of dimension d. Therefore, a histogram is the underlying foundation of the mirror image. The reflection or histogram of a dimension does not produce a mirror image, it is a building block to create one.

3.3.2 Nonlinear Data Space

A highly dimensional data set indicates a practically empty data space. Linearly increasing the number of dimensions will nonlinearly enlarge the data space. The data space

$$S = R_0 * R_1 * \ldots * R_{D-1} \tag{3.3}$$

defines an enclosed space in which data points occur, if any point exists. Adding a dimension does not affect the number of data points. The chance that a value of a counter of a specific bin is unequal to 0 (i.e., one or more data points are located in the interval of the bin) approaches 0% when the number of dimensions is increasing. Figure 3.5 illustrates this behavior.

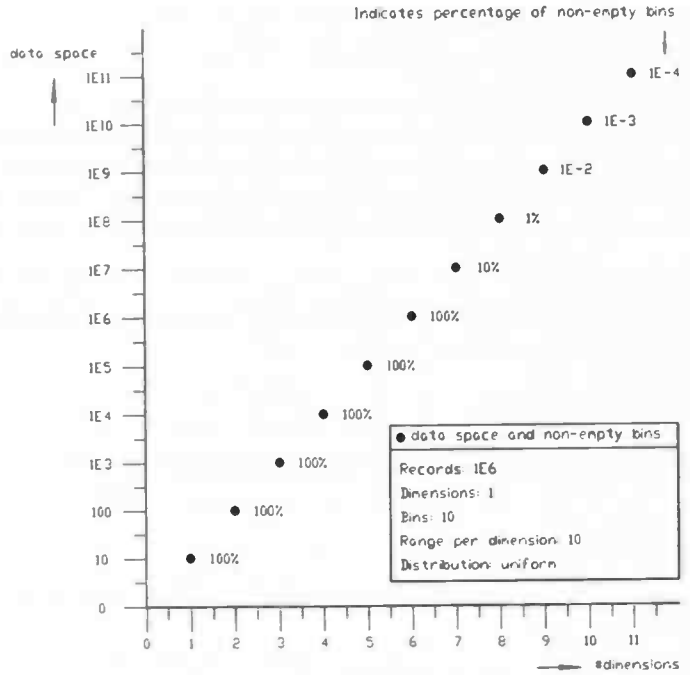


Figure 3.5 Nonlinear growth of the data space, and nonlinear decline of non–empty bins.

Empty bins do not provide information other than there are no data points in the specific interval. They also waste time and space. Utilizing empty bins may provide extra information on the data set (i.e., details). It is preferred to use the bins more efficiently, and, in order to do so, a new problem is introduced, namely maximizing the efficiency of the bins.

3.3.3 Bin Equalizing

Low contrast is in the field of digital image processing a common problem. A narrow shaped histogram (only a few bins are non–empty) indicates little dynamic range, and it corresponds to a low contrast image. A histogram with significant spread corresponds to an image with high contrast. A common way of manipulating histograms to increase the contrast of an image is histogram equalization (also known as histogram linearization) [13]. An image consists of pixels, and a luminance value is attributed to each pixel which indicates the intensity of the pixel at issue. If each luminance value occurs as frequently as any other luminance value (a uniform distribution), the histogram of the image is equalized. Consequently, the entropy or uncertainty is maximized, and a maximum amount of information is therefore presented to the observer [19].

Whilst equalizing the histogram of an image, the luminance values of the pixels are adjusted. The data points of a data set do not have a luminance value, they only exist in some data space. One or more fields of a record may indicate a value which corresponds with a luminance value (e.g., age). However, this would imply advance knowledge which is not available, but by looking at the matter from a different angle, it is possible to circumvent this problem.

Digital image processing is about adjusting an image to display it to an observer, e.g., the combination of the human eye and brain. The actual information contents of an image will not increase (a decrease is much more likely). The ability of the observer to process visual information is limited, and digital image processing provides preprocessing techniques to obviate this limitation. However, the 'observer' in this thesis is the deterministic sampling algorithm. This algorithm does not lack preprocessing techniques, it lacks information due to the empty bins.

Plain histogram equalization is not possible, but one would like to make use of the advantages of histogram equalization without taking the limited observer into account. The answer is not to equalize the histogram, but to equalize the horizontal axis of the histogram. In other words, by properly adjusting the range of the bins according to the distribution of the data points, each bin should contain the same number of data points. This will indirectly result in an equalized histogram. The purpose remained the same (an equalized histo-

gram), but the premise changed from adjusting the field values of a record (i.e., the luminance value of a pixel) to adjusting the interval of the bins (i.e., adjusting the width of the bars of the histogram).

Bin equalizing is not identical to histogram equalizing, but is closely related to it. Bin equalizing actually adds information to the representation of a dimension as opposed to histogram equalizing. After bin equalizing, each bin is maximally utilized, resulting in more information on the distribution of the data points (more bins become available to represent the distribution). Bin equalizing is a dynamic adjustment which solely depends on the data set. However, bin equalizing does require more preparatory work than histogram equalizing.

The index i_d is used to indicate the subrange $range_{di_d}(bin_{di_d})$, but after bin equalizing it does not always provide the correct index. It is necessary to introduce a new variable to compensate for the lost information. The extra variable is a limit $limit_{di_d}$, where $0 \leq limit_{di_d} \leq R_d$.

3.3.4 Accumulating Overflow

Ideally, the values of the counters cnt_{di_d} of dimension d should be equal to

$$mean_d = \frac{N}{\#bins_d}. \tag{3.4}$$

If the contents of bin bin_{di_d} deviates from $mean_d$, it must be equalized. The equalizing procedure starts by calculating the overflow

$$overflow_{di_d} = cnt_{di_d} - mean_d; \qquad -mean_d \leq overflow_{di_d} \leq N - mean_d, \tag{3.5}$$

of each i_d–th bin of each dimension d. The overflow is positive, negative or 0. A positive value indicates a surplus of data points in the interval of the specific bin, a negative value a shortage, and 0 indicates a number of data points equal to $mean_d$.

The starting point of the equalizing procedure is not yet available. The starting point indicates the first bin of a sequence of bins (one bin or more) which represents the largest cluster of data points. Three variables are introduced, $start_d$, $temp$ and $oldTemp$ ($oldTemp$ and $start_d$ are initially 0). Each array A_d is traversed from the first through the last bin (from bin_{d0} through $bin_{d\,bin_d-1}$). On the fly, the value of $overflow_{di_d}$ is added to $temp$. If $temp \geq oldTemp$, the value of $temp$ is copied into the variable $oldTemp$, and the index of the starting point is copied into $start_d$. Otherwise this part of the procedure is skipped. The same process is repeated, but it now starts at the second bin. The procedure terminates when the starting point of the traverse becomes the last bin plus 1 ($\#bins_d$). The value of $start_d$ will then indicate the starting point of the equalizing procedure for dimension d. Figure 3.6 illustrates an example.

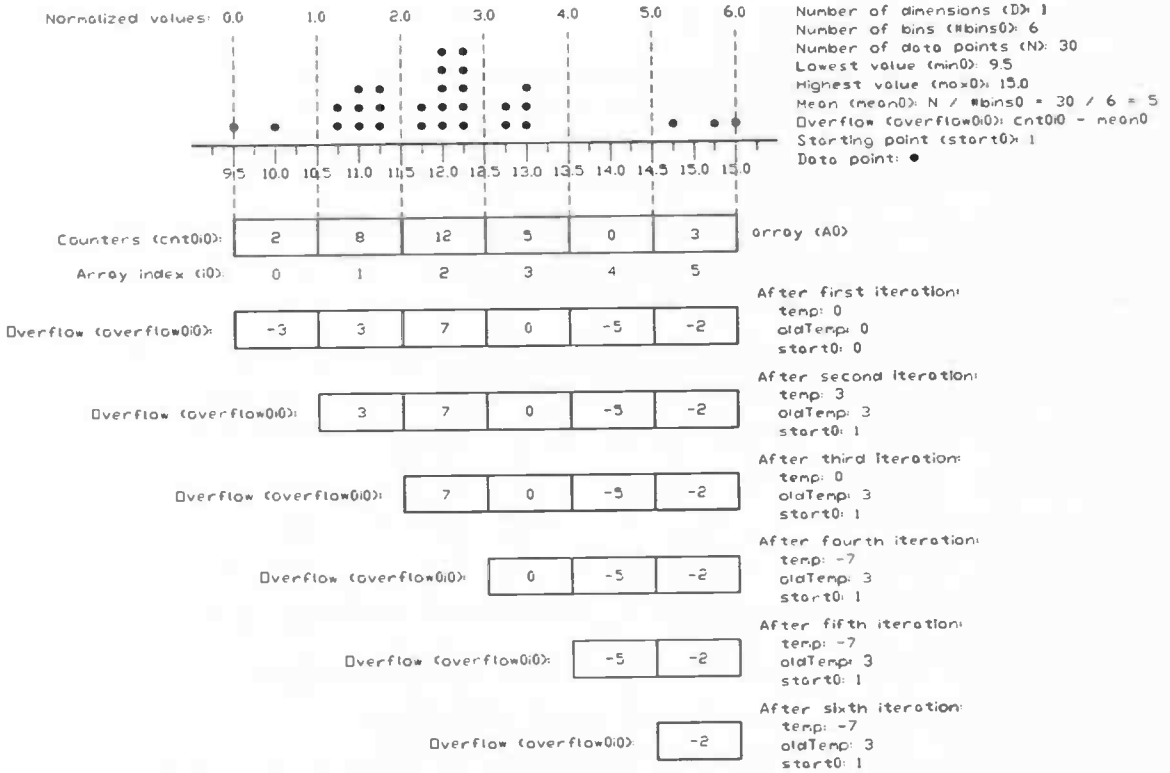


Figure 3.6 Calculating the starting point of the equalizing procedure ($start_d$ becomes 1).

The surplus of a bin will be smeared in such a manner that one or more succeeding bins will dissipate the surplus. The dissipation is denoted

$$dis_{di_d} = \textit{if } (overflow_{di_d} < 0) \textit{ then } overflow_{di_d} \textit{ else } 0. \tag{3.6}$$

A sequence of overflowing and / or underflowing bins may accumulate a positive or a negative surplus (or neither if the surplus is equal to 0). The accumulation of the positive surplus of preceding bins is denoted

$$acc_{di_d} = acc_{di_d} + acc_{di_d - 1}. \tag{3.7}$$

As soon as acc_{di} becomes smaller than 0, its value is replaced by 0 in an on the fly manner. The calculation of the accumulated overflow starts at bin $start_d$ and it terminates after processing bin $start_d - 1$ (array A_d is circular). Figure 3.7 will clarify the previous.

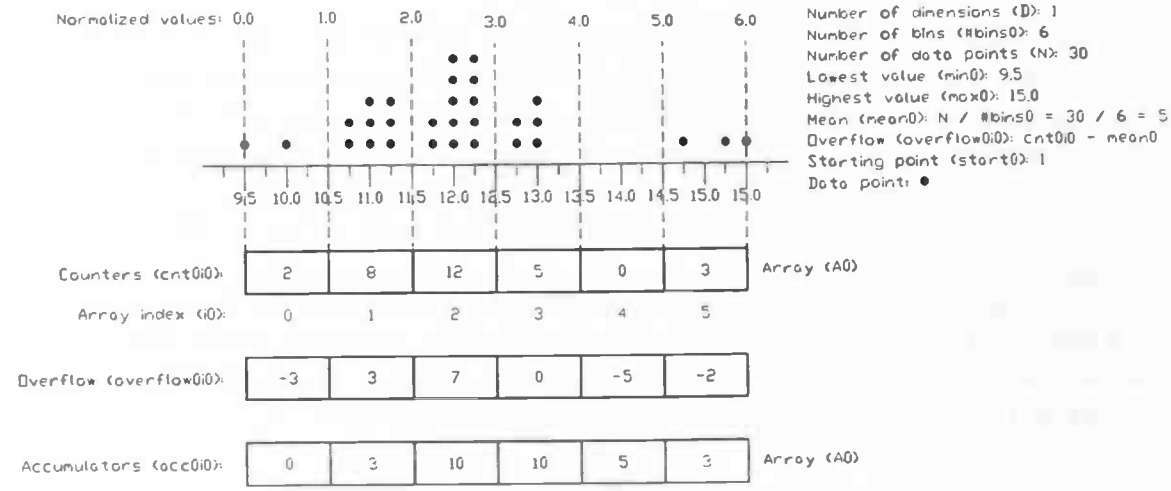


Figure 3.7 Calculating the accumulated overflow per bin (acc_{di_d} of bin $start_d - 1$ always becomes 0).

3.3.5 Overflow Processing

For each bin of each array, an initially empty binary search tree is introduced. Due to its balancing capabilities, a red-black tree is preferable. A node of the tree contains two variables, *key* and *keyCnt*. The first variable is used to store the field value val_{nd}, and the second variable is a counter denoting how often the value *key* occurs. The variable *keyCnt* comes under compaction of field values as described in chapter two, and is initially 0.

The data set is read for the third time. Each value val_{nd} is mapped on the corresponding bin bin_{di_d}. If acc_{di_d} is greater than 0, the value is stored into main memory by means of the binary tree, and the value of *keyCnt* is increased by one. The value of acc_{di_d} is decreased by one.

Quite often, the value of acc_{di_d} is 0. If so, val_{nd} is compared with the lowest value of *key*. If val_{nd} is greater than the lowest value of *key*, val_{nd} is copied into the tree. If the value of *keyCnt* of the node which contains the lowest value of *key* is greater than 1, then *keyCnt* is decreased by 1. Otherwise the node is removed from the tree. See also Figure 3.8.

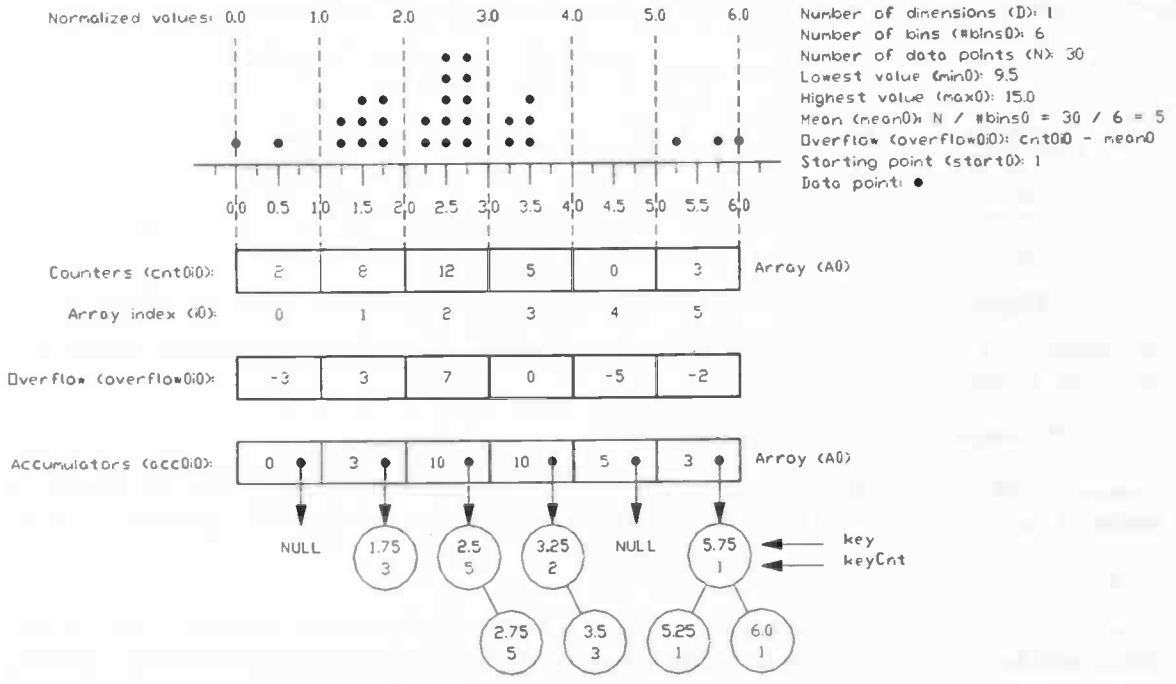


Figure 3.8 Building trees to determine the limit $limit_{di_d}$ per bin.

The value $limit_{di_d}$ of each bin bin_{di_d} is either stored as a *key* value, or is equal to a value in the range of $0, 1, \cdots, \#bins_d$. If a bin does not dissipate a surplus of previous bins, the value of $limit_{di_d}$ becomes the value of the index indicating the bin which stores $limit_{di_d}$. Otherwise starting at $start_d$, the *key* values of the trees are spreaded over the bins. A sequence of values of acc_{di_d} is said to be continuous as long as acc_{di_d} is greater than 0. Each tree which occurs within a continuous sequence is connected to the tree positioned at the right (the successor of the rightmost node of a tree is the root of the tree positioned at the right).

The value of *keyCnt* of a node is used to determine the order of the value of *key*, e.g., the node containing the value 3.5 in Figure 3.8 is the 16-th, 17-th and the 18-th value of the overflow. If $start_d$ is unequal to 0, array A_d is considered to be circular, and an offset is added to $limit_{di_d}$ for each i_d smaller than $start_d$. Determining the limit $limit_{di_d}$ involves a lot of different (exception) states, and it needs to be carefully programmed to prevent errors. Equalizing the bins of the running example results in Figure 3.9.

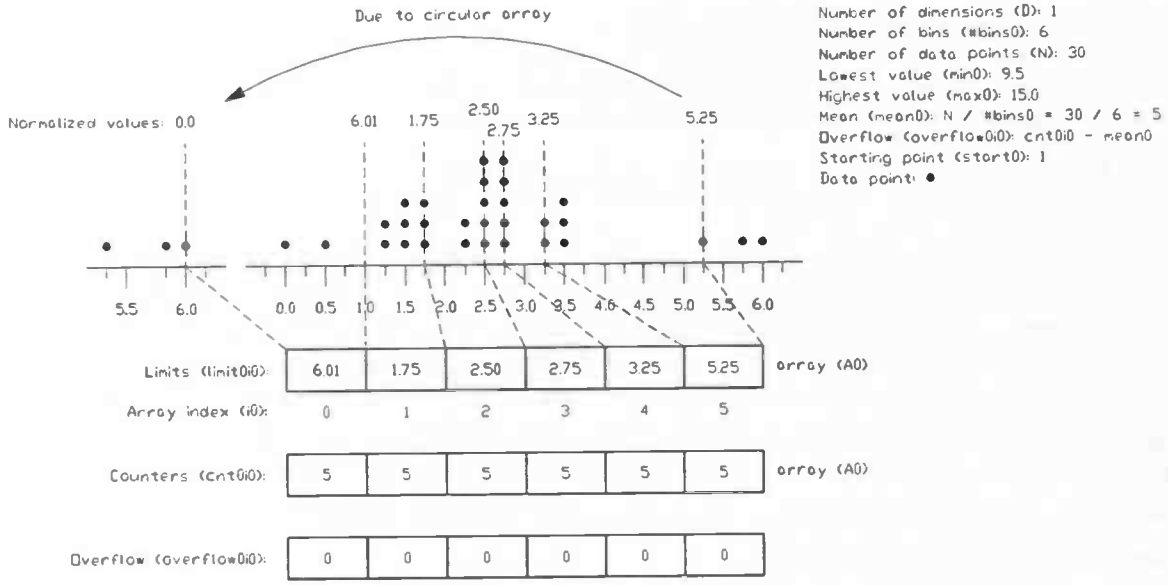


Figure 3.9 Equalizing the bins of array A_0 by means of the limits $limit_{di_d}$.

3.3.6 Selecting the Correct Bin

Each field value is represented by a specific bin, and the counter of that bin has to be increased to represent that field value. To do so, a field value val_{nd} is addressed by (n,d), and is linearly scaled to a range from 0 through $\#bins - 1$. The result is rounded off downwards to produce an integer based value i_d. The value of i_d indicates a specific bin bin_{di_d} of array A_d (i_d is an index to indicate array A_d). Bin bin_{di_d} stores the values $limit_{di_d}$ and cnt_{di_d} the latter is initially 0. The value i_d is compared with $limit_{di_d}$. If $i_d < limit_{di_d}$ the correct bin has been found. Otherwise the test $i_d < limit_{di_d+1}$ is performed (bear in mind that array A_d is circular). This process continuous until the correct bin is selected. If the value i_d maps on a specific bin, the counter of that bin is increased by one. The worst case scenario comprises $\#bins_d$ tests.

The ultimate result of the running example, the mirror image, is shown in Figure 3.10. Only the combination of array A_0 through A_{D-1} is a mirror image. Since the running example is one-dimensional, the mirror image is provided by a single array, namely A_0.

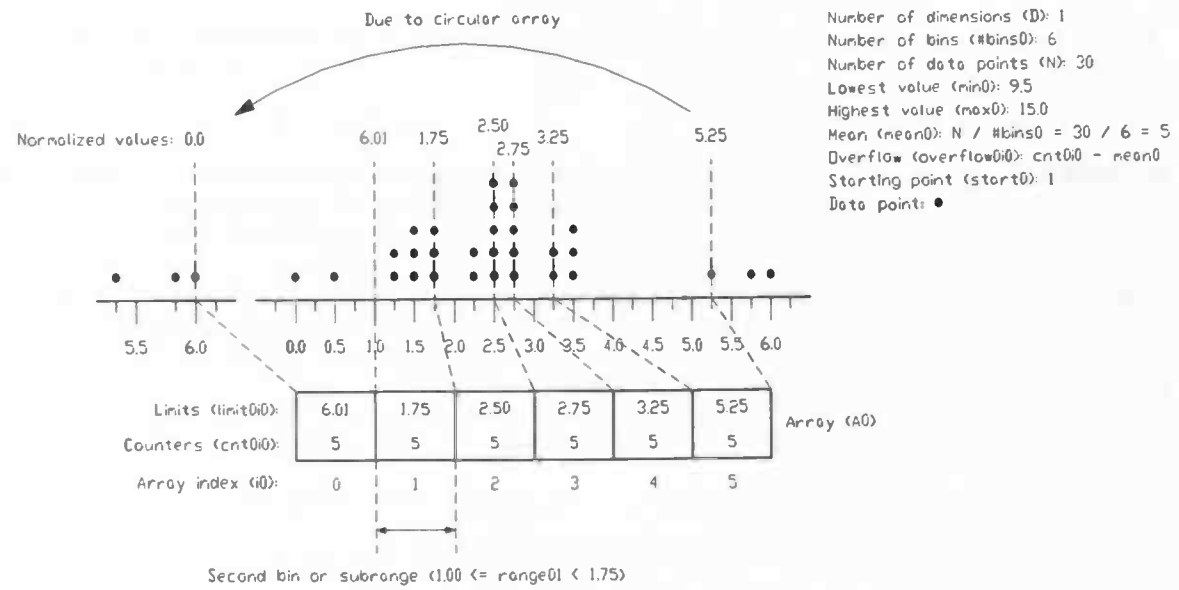


Figure 3.10 Mirror image of the data set constituted by array A_0.

3.4 Determining a Sample Set

A sample set is always smaller than the data set (otherwise there is no use to sample a data set). The ratio between the data set and the sample set is called the reduction

$$R = \frac{\#records\ sample\ set}{\#records\ data\ set}; \qquad 1 < R < \#records\ data\ set. \tag{3.8}$$

Determining a sample set is straight forward. The counter cnt_{di_d} of each bin bin_{di_d} is divided (reduced) by R. The data set is read for the fourth time, and each field value val_{nd} is normalized to produce the index i_d. The correct bin bin_{di_d} of array A_d is selected by means of index i_d. Then the counter cnt_{di_d} is examined (not increased or decreased). If each counter of each bin selected by the d values of the n-th record is greater than 0, then the n-th record is suitable record, and is copied into the sample set. The counters cnt_{0i_0} through $cnt_{\#bins-1\,i_{\#bins-1}}$ are then decreased by one. If the record does not suffice, i.e., at least one of the counters examined is equal to 0, it is rejected and no further adjustments are made. This process is entirely deterministic, and it therefore produces a deterministic sample set. The data set of the running example is reduced by $1\frac{2}{3}$, and the resulting sample set is shown in Figure 3.11.

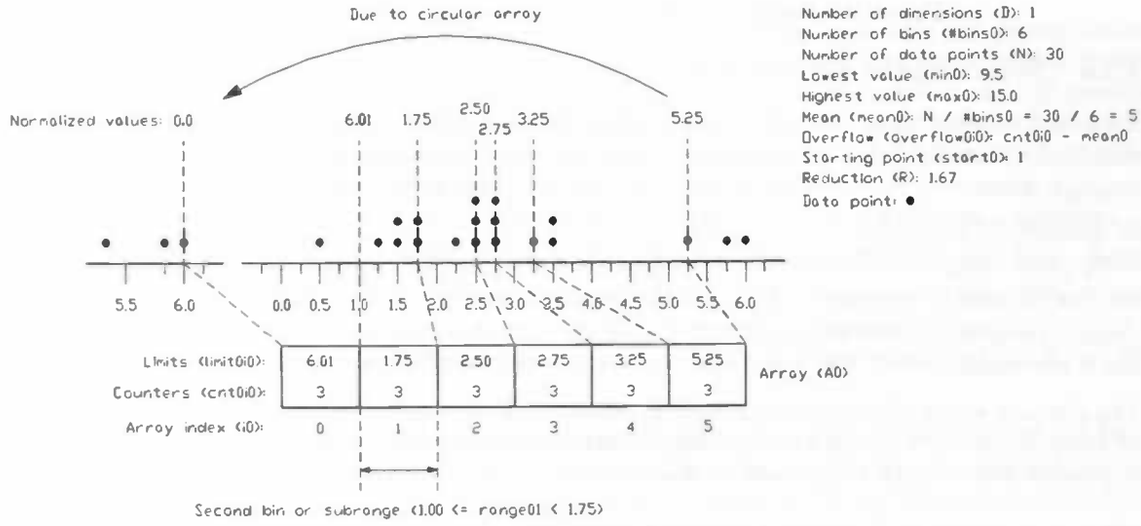


Figure 3.11 The resulting sample set after applying the data set to the deterministic sampling algorithm.

3.5 Loss of Information

The mirror image reflects the data set, but not all information provided by the data set is always preserved. The mirror image is based on a specific number of bins per dimension. Consequently, the resolution of the mirror image is limited. This limitation can be minimized by increasing the number of bins.

Some information on the combination of the dimensions is sometimes lost, because the mirror image is a linear representation of a nonlinear data space. The mirror image does not always represent the entire relation between the dimensions (reflecting the 'depth' of the data set is not always entirely possible). Figure 3.12 illustrates an example.

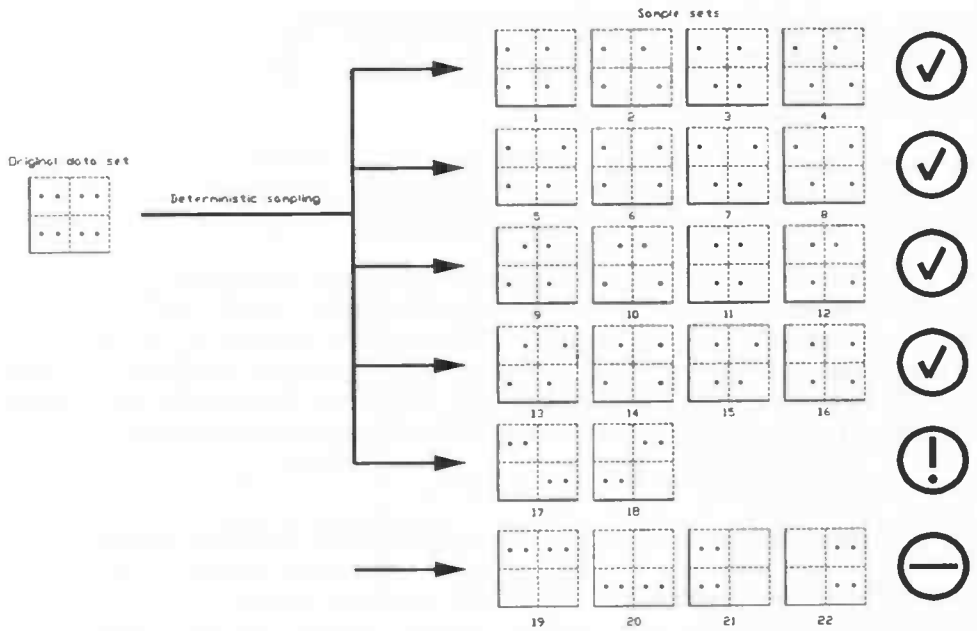


Figure 3.12 Loss of information due to a linear representation of a nonlinear data space.

The selection of a specific deterministic sample set depends on the order of the records in the data set. There are eighteen possible sample sets (not 22), two of which are less desirable (number 17 and 18). Adding extra dimensions will decrease the number of less desirable sample sets, because an extra dimension provides an extra view on the data set (i.e., it adds an extra layer to the mirror image). In other words, more information on the relation between the dimensions becomes available. Whilst increasing the number of dimensions, the chance to select a less desirable sample set will nonlinearly decline. The chance to select a less desirable sample set is already quite small in case of two dimensions, e.g., $\frac{2}{18} * 100\% = 11.1\%$ in Figure 3.12. The worst case scenario comprises two dimensions (not one, because one dimension is entirely flat) and a uniform distribution of the data points. The chance to select a less desirable sample set (number 17 and 18) is multiple times smaller than the chance to select a proper sample set (number 1 through 16). The phrase "less desirable" is used to emphasize that even a less desirable sample set is an important improvement in comparison with the worst possible sample sets. The deterministic sampling algorithm will never select the worst possible sample sets 19 through 22. Random sampling does not exclude these sample sets.

3.6 Addressing

3.6.1 Deterministic Sampling

A record defines a data point by providing a coordinate for each dimension of the data space. Each coordinate is a property of that record, but the combination of coordinates does also yield a specific property, e.g., in case of 3 dimensions *A*, *B* and *C*, there are 7 specific properties (i.e., *A*, *B*, *C*, *AB*, *AC*, *BC*, and *ABC*). Combining all dimensions into one property (e.g., *ABC*) will produce the most specific property, because it defines a point in a data space which comprises each available dimension. For the sake of clarity, *ABC* is not distinct from *ACB*, *BAC*, *BCA*, *CAB*, and *CBA*.

The mirror image provides the properties of the records that have to be copied into the sample set. If the properties of a record correspond with the required properties as defined by the mirror image, it is a useful record, and it can therefore be copied into the sample set. The process of selecting a record is based on three steps:

- Determine the properties of the record;
- Compare the properties with the desired properties provided by the mirror image;
- If the properties correspond, copy the record into the sample set.

In other words, the properties of a record are available and verified. Then a decision is made based on this verification. It is also possible also to address the records of a data set by reversing this process.

3.6.2 Deterministic Addressing

The mirror image can be used to address specific records in a data set. In other words, it is possible to read specific records from the data set without using a standard data structure (e.g., B–trees). An advantage of the mirror image is the lack of space consuming pointers, as opposed to most standard addressing techniques.

The difference between deterministic sampling and deterministic addressing is subtle. Whilst sampling, the record is available, and the required properties are provided by the record itself. However, whilst addressing, the properties of a desired record are already available, but the (address of the) record is not. Producing an address of a record can be interpreted as a decision, the record is or is not available. If it is available, the properties are suitable. Otherwise they lack suitability, and, as a result, the record is not available. The process of addressing a record is closely related to the process of selecting a record, namely:

- Determine the properties of the desired record;
- Compare the properties with the properties provided by the mirror image;
- If the properties correspond, read the record.

There is one important difference between deterministic sampling and deterministic addressing. In case of sampling, the required properties are always valid, because they are supplied by a single record. However, in case of addressing, it is possible to combine non–related properties of several records, and still produce an address (e.g., a combination of the field values of the first and last record). A 'link' between the bins of the dimensions of the mirror image is missing (i.e., the record itself), and some extra information is required to obviate this underlying relation between the bins.

An example will clarify the previous. Determining whether or not a record is suitable is like finding the entrance of a labyrinth by starting at the center of it. Going left or right depends on the properties supplied by the record. However, determining the address of a record is like finding the center of a labyrinth by starting at the entrance of it. Going left or right cannot be based on the properties, because they are not derived from an actual record, i.e., they are the result of an unknown process. Such an unknown process is, e.g., a learning algorithm.

A new variable $bitPattern_{di_d}$ is introduced, and added to each bin bin_{di_d}. The pattern is composed of b bits. The number of bits is equal to the number of bins of the next dimension ($\#bins_{d+1}$), except for the last dimension. The number of bits of the bit patterns of the last dimension $D-1$ depends on, e.g., the number of disk blocks or the number of records. The b–th bit provides information on the b–th bin of dimension $d+1$. If the b–th bit of bit pattern $bitPattern_{di_d}$ in bin bin_{di_d} of dimension d is equal to 1, then the b–th bin of dimension $d+1$ is a valid successor of bin bin_{di_d}. Otherwise it is not a valid successor.

After adding a bit pattern to each bin, a bin will contain three variables, namely cnt_{di_d} of 32 bits, $limit_{di_d}$ of 32 bits, and $bitPattern_{di_d}$ of $\#bins_{d+1}$ bits. Using the mirror image to address a record is quite efficient. The data set which was used to perform the field–test described in the next chapter comprises 14742 records of 78 dimensions each. Suppose that each dimension is represented by 100 bins, and that the records have to be addressed in blocks of 10 records (i.e., $\lceil \frac{14742}{10} \rceil$ produces a total number of 1475 blocks). The total demand for main memory then becomes 77 * 100 * (32 + 32 + 100) + 100 * (32 + 32 + 1475) = 1416700 bits (approximately 0.18 MB). Increasing the number of bins per dimension will increase the resolution of the mirror image. Nevertheless, tests have shown that even 25 bins per dimension will suffice (see chapter four).

Setting the bits of $bitPattern_{di_d}$ can be done during the investigation of the data set. All bits are initially 0. Each time when bin i_{d+1} of dimension $d+1$ is processed, the i–th bit of $bitPattern_{di_d}$ of dimension d is set to a value of 1 (j is the preceding bin which was processed in the previous step). When the last dimension $D-1$ is processed, the n–th bit of $bitPattern_{D-1 i_{D-1}}$ is set to a value of 1. The variable n denotes, e.g., the n–th record or the disk block which stores the record at issue. If a bit was already equal to 1 (regardless of the dimension processed), then its value does not have to be altered. Figure 3.13 shows an example of the previous.

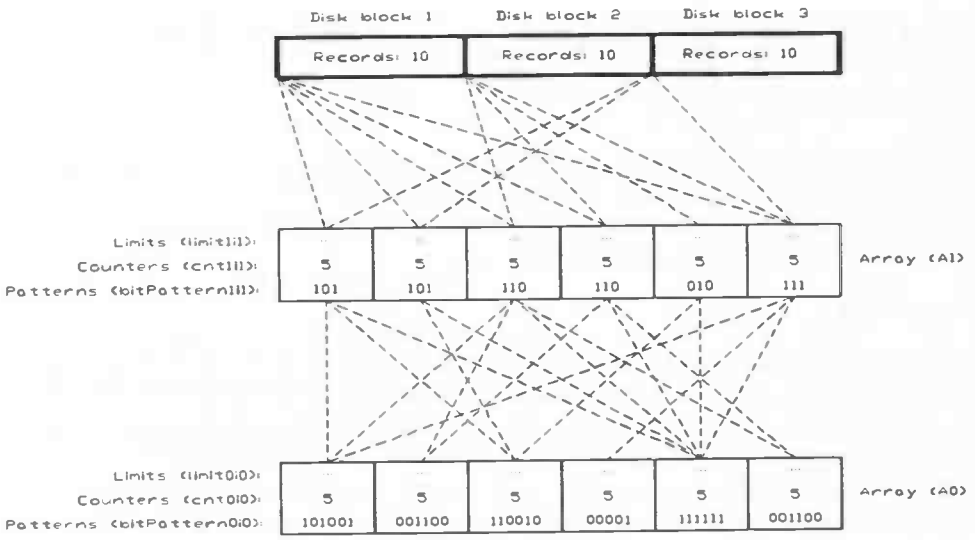


Figure 3.13 Addressing a record within a disk block by means of the mirror image. A dashed line indicates a valid path from one bin to another.

The premise of the next chapter is deterministic sampling. However, the chapter does also apply to deterministic addressing, because deterministic sampling and deterministic addressing are entirely interchangeable.

3.7 Summary

A data set comprises at least one record of at least one field. The number of records and fields is arbitrary, but each record must contain the same number of fields. The fields can be of any type. The data set defines a data space, an enclosed space in which data points occur, if any point exists. The data space is composed of dimensions, and linearly increasing the number of dimensions will nonlinearly enlarge the data space. The result is a virtually empty data space. The dimensions of the data space are defined by the fields of the data set.

Each dimension is represented by a specific number of bins. A bin partitions the range of a specific dimension into intervals of equal length. The number of data points in the interval of the bin are counted and the outcome is stored into the bin. The result is a histogram of each dimension. By utilizing the empty bins (or almost empty bins) by means of bin equalizing, the efficiency of the bins is improved. Bin equalizing is closely related to histogram equalizing, it provides the same advantages, but, unlike histogram equalizing, bin equalizing actually adds information to the histogram.

By combining the histograms into a single structure, a mirror image of the data set is obtained. The counter of each bin of each dimension is divided by the desired reduction of the data set The records in the data set are mapped on the mirror image in a one by one manner. Each field value of a record indicates a specific bin of the dimension at issue. If the counters of the bins selected are unequal to 0, then this specific record is a useful addition to the sample set, and the counters are decreased by one. Otherwise the record is not suitable, and no adjustments are made.

In other words, the data space of the data set is divided into areas, and each area (approximately) contains the same number of data points. The size of such an area depends on the number of bins per dimension and on the total number of data points available. By dividing the number of data points per area by the desired reduction, a related reflection of a suitable sample set is found. Except for the number of data points, the reflection of the sample set is identical to the reflection of the data set. Mapping each record separately on the data space, and examining the number of data points of the related areas, will reveal whether or not a specific record is a useful addition to the sample set.

By changing the premise from sampling to addressing, and by adding an extra variable to each bin of the mirror image, the mirror image can be used to retrieve a specific record from the data set. The demand for space is minimal due to the lack of actual pointers.

Deterministic sampling does not differ from deterministic addressing. It is the same operation observed from a different perspective. After all, simplicity is the hallmark of truth.

Chapter 4 Behavior and Field-Test

4.1 Introduction

Experiments are necessary, because the methods employed and the data used are generally too complex for a complete formal treatment. Of course, there is a vast literature in statistics, computational (learning) theory, and the like, but the last word is always spoken by an empirical check, an experiment, as in any other science which needs empirical evaluation of its theories [12].

4.1.1 Intuition

As explained, building an equalized histogram of each dimension is the underlying foundation of the mirror image. Bin equalizing is closely related to histogram equalizing, but the behavior of the entire deterministic sampling algorithm does not immediately agree with intuition. The mirror image itself is a linear structure, but it represents a nonlinear data space. This appears to be a contradiction and an insurmountable problem, but it mainly depends on the interpretation of linear and nonlinear. To substantiate this interpretation, the behavior of the algorithm will be examined before setting out the actual field-test.

4.1.2 Standard Input Set

The number of possible data sets is infinite, and the behavior of the algorithm is directly related to the distribution of the data points, resulting in an infinite number of behavior patterns. Examining all behavior patterns is therefore impossible. To get round this difficulty, a standard input set is defined. The standard input set provides a worst case scenario to emphasize the specific behavior patterns which have to be examined.

The number of data points is arbitrarily set to a value of $2.5 * 10^6$, but any value which is large in comparison with the number of training patterns commonly applied to the learning algorithm at issue will suffice.

If a sample of the first $\frac{N}{R}$ occurring data points is drawn from a sorted data set (i.e., one ore more fields were used to sort the records), then an in order sample set is acquired. If the same process is repeated in case of random storage, the sample set will be random. An in order sample set will contain precise information on the beginning of the data set, but a random sample set will probably contain some information on the entire data set. If the records are stored in order, the dependence on the mirror image is maximized. The algorithm has to delay the selection of a suitable record whilst traversing the data set. The delay is based on the resolution of the mirror image (i.e., the number of bins). The sorting rate of the data set will influence the quality of the sample set. If the number of bins is relatively small, an unsorted data set is preferred to a sorted data set. Table 4.1 illustrates this.

Number of bins	Contents of the sample set indicated by the indexes of the records
1	1 ... N / 2 worst sample
2	1 ... N / 4 and $2N$ / 4 ... $3N$ / 4
3	1 ... N / 6 and $2N$ / 6 ... $3N$ / 6 and $4N$ / 6 ... $5N$ / 6
4	1 ... N / 8 and $2N$ / 8 ... $3N$ / 8 and $4N$ / 8 ... $5N$ / 8 and $6N$ / 8 ... $7N$ / 8
...	...
N / 2	1, 3, 5, ..., N best sample

Table 4.1 A sample set is drawn from a one-dimensional data set which stores N data points in order $(1 \cdots N)$. The reduction R is equal to 2.

The total number of records divided by the reduction seems to yield an optimal number of bins, namely

$$\#bins_d = \frac{N}{C} \tag{4.1}$$

The standard input set is either one-dimensional or two-dimensional, and the data points form part of one cluster or two clusters of equal size (both depend on the examined behavior.)

The worst kind of distribution is a uniform one, because the maximum spread of the data points causes the mirror image to reflect the entire data space. The opposite of the previous is a cluster of data points positioned on top of each other. In this case the mirror image will reflect a data space as small as one data point (the mirror image does not represent empty data space).

According to the previous, the following assumptions are made to obtain a standard input set:

- The number of data points is equal to $2.5 * 10^6$;
- The data set stores the data points in order;
- The number of dimensions is equal to one or two;
- The number of clusters is equal to one or two;
- In case of one cluster, the distribution of the data points is uniform (one large entirely spreaded cluster);
- In case of two clusters, the distribution of the data points within the clusters is uniform (the clusters are separated by an area of empty data space).

The examined behavior is based on the standard input set. It provides an indication of the behavior to support the comprehension of the algorithm in general. However, the standard input set provides a worst case scenario to emphasize specific properties of the algorithm, and the results of applying the standard input set to the algorithm have to be considered accordingly.

4.2 Behavior

4.2.1 Accuracy

The accuracy is expressed as the ratio between the variance of the sample set and the variance of the data set (or as the ratio between the mean of the sample set and the mean of the data set). The accuracy is related to the reduction and the number of bins, as shown in Figure 4.1. The reduction is maximized for each specific number of bins to obtain worst case behavior, e.g., $2.5 * 10^6$ records and 25 bins yields a maximum reduction of $\frac{2.5 * 10^6}{25} = 100 * 10^3$ (i.e., the counter of each bin of the mirror image of the sample set becomes 1).

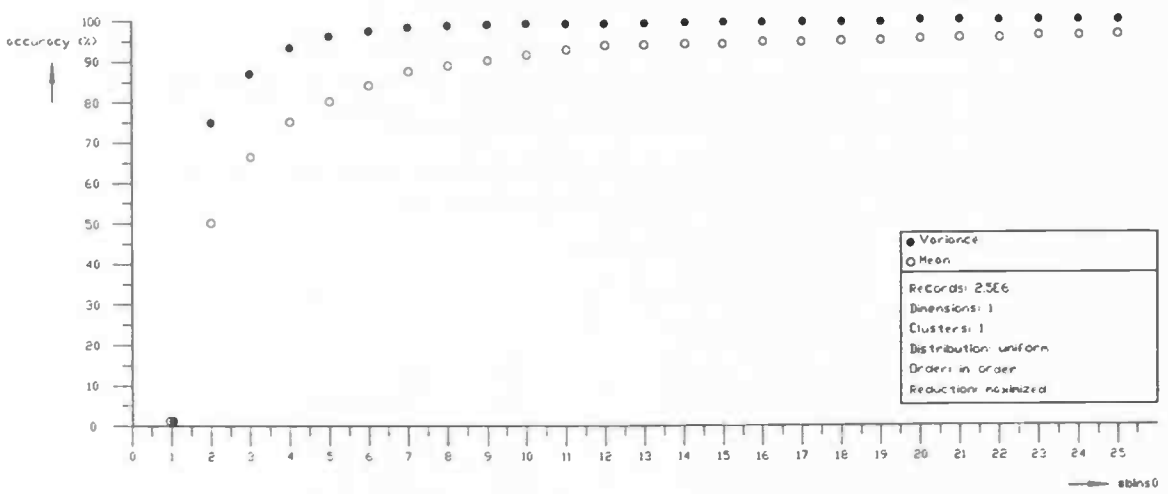


Figure 4.1 The ratio of the variance and mean of the sample set with respect to the data set (the accuracy is equal to $\frac{variance_{sample\ set}}{variance_{data\ set}} * 100\%$ or $\frac{mean_{sample\ set}}{mean_{data\ set}} * 100\%$).

4.2.2 Maximum Reduction

The maximum reduction is related to the number of bins, because adding more bars (i.e., bins) to an equalized histogram decreases the average height of the bars (i.e., the average value of the counters). If the (average) maximum reduction

$$R_{\max_d} = \frac{N}{\#bins_d}, \tag{4.2}$$

equals $mean_d$, then its value is an optimized one. If the value of cnt_{di_d} is smaller than $R_{\max_d}$ (i.e., the histogram is not entirely flat), then cnt_{di_d} is the maximum reduction of that specific bin (e.g., if cnt_{di_d} is 25 and $R_{\max_d}$ is 30, then the maximum reduction of bin_{di_d} will be 25 instead of 30). The maximum reduction is nonlinearly related to the number of bins. This relation is plotted in Figure 4.2.

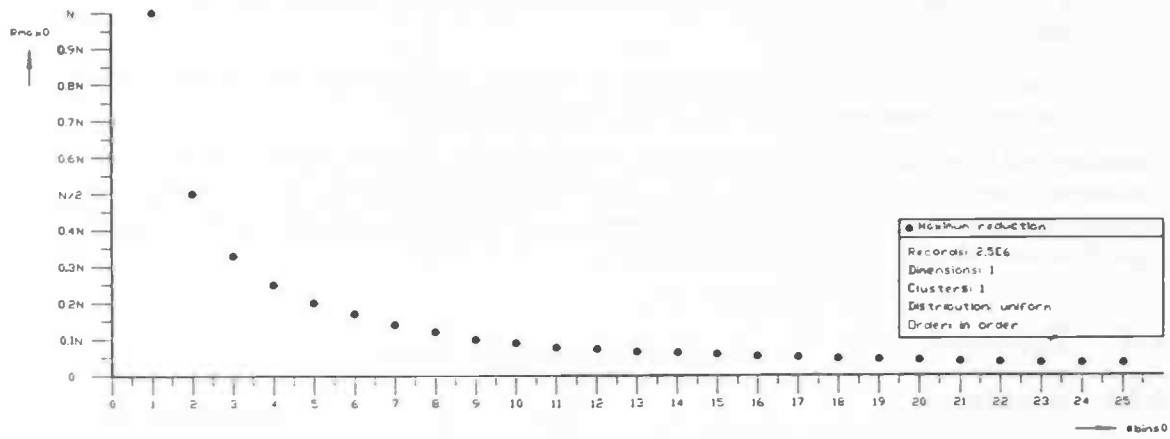


Figure 4.2 The maximum possible reduction of a bin.

By turning Figure 4.1 upside down, the progress of the curve is almost identical to the curve in Figure 4.2. Therefore, an accurate variance and / or mean implies more bins, but a large value of the reduction implies less bins. This is a contradiction, because one would like to have a sample set which is accurate and small.

4.2.3 Accuracy and Reduction

If the actual reduction R exceeds the value of cnt_{di_d} then cnt_{di_d} will become 0 after dividing it by R, and all data points in the interval of bin_{di_d} are excluded. As a result, the accuracy will decrease instantaneously and becomes 0. The upright line in Figure 4.3 illustrates this behavior.

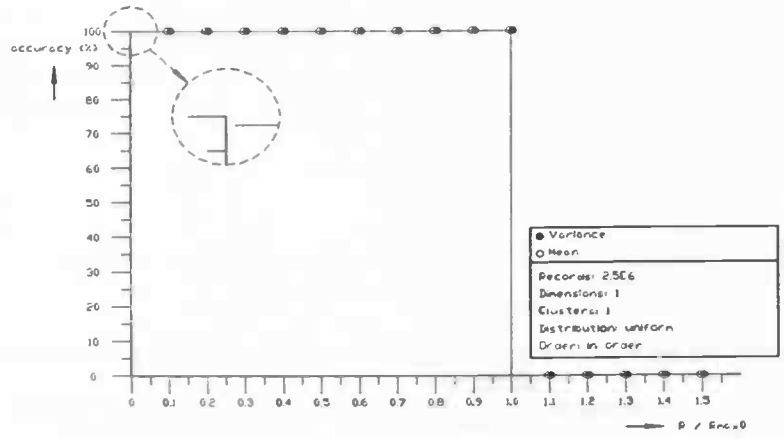


Figure 4.3 Instantaneous decrease of the accuracy when the reduction exceeds the maximum reduction.

When the reduction exceeds the maximum value of the counter of a bin which represents an entire cluster, the entire cluster will be excluded from the sample set. In case of multiple clusters, the result will be a stepwise

reduction of the accuracy (i.e., each step is the result of excluding a cluster). The decline of the accuracy in Figure 4.3 is composed of just one step, because all data points were located in the interval of a single bin.

4.2.4 Cluster Ratio

The bins represent the data points, and at least one bin is necessary to represent a cluster. A bin may represent more than one cluster, but then the mirror image cannot distinguish between the different clusters. The minimum number of bins which are required to properly sample the data set depends on the number of data points of the smallest cluster. In Figure 4.4, the cluster ratio

$$CR = \frac{\#records\ smallest\ cluster}{\#records\ largest\ cluster} \tag{4.3}$$

divides the figure into two areas A and B.

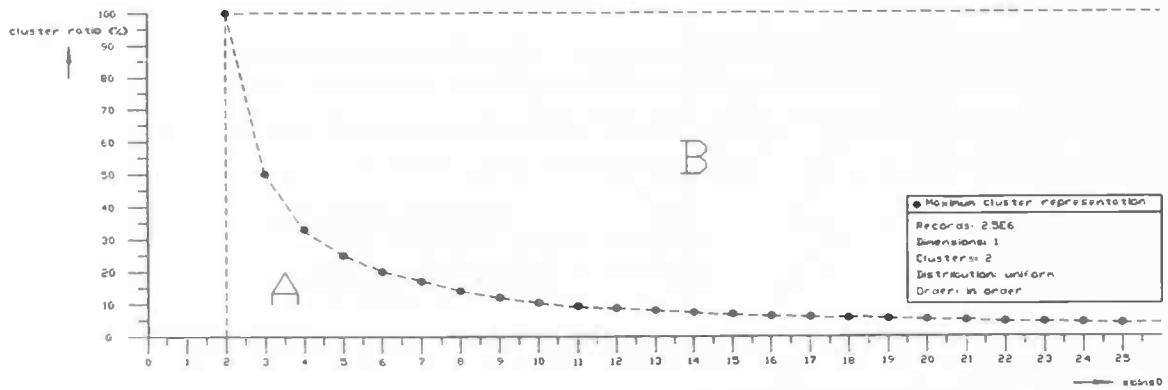


Figure 4.4 The minimum number of bins required to properly represent the smallest cluster. In area A, both clusters are properly represented, but in area B, the two clusters are (partially) merged.

Recapitulating the previous, a specific number of bins implies a certain accuracy (Figure 4.1), a maximum reduction (Figure 4.2 and Figure 4.3), and a minimum size of the smallest cluster (Figure 4.4).

4.2.5 Cluster Representation

Representing two or more clusters by a single bin will affect the accuracy of the sample set. The reflection of the smallest cluster, i.e., its mirror image

$$MI = \frac{\#records\ smallest\ cluster}{N} * \#bins_{di} \qquad 0 < MI \leq \#bins_{di} \tag{4.4}$$

is a measure of the available bin space which actually reflects the cluster. Figure 4.5 illustrates the previous.

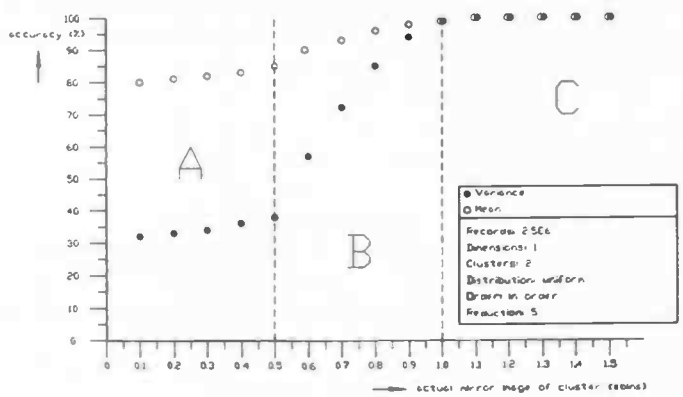


Figure 4.5 The mirror image MI of a cluster must be at least 1.0 (i.e., a single bin). In area A, two clusters are entirely located in the interval of a single bin, two clusters are partially located in the interval of a single bin in area B, and each cluster is reflected by at least one bin in area C.

4.2.6 Resource Time

The building block of the mirror image is a histogram of each dimension d. A histogram is an array of length $\#bins_d$ and the total number of arrays is equal to the number of dimensions, namely D. For each record in the data set, D arrays are processed a fixed number of times. The complexity (i.e., the demand for time) of the processes is related to the length of an array (the worst case scenario comprises one traverse of the array). Therefore, the complexity is linearly related to the number of records, the number of dimensions, and the number bins. The complexity is not related to the distribution or the reduction of the data set. The demand for time is plotted in Figure 4.6.

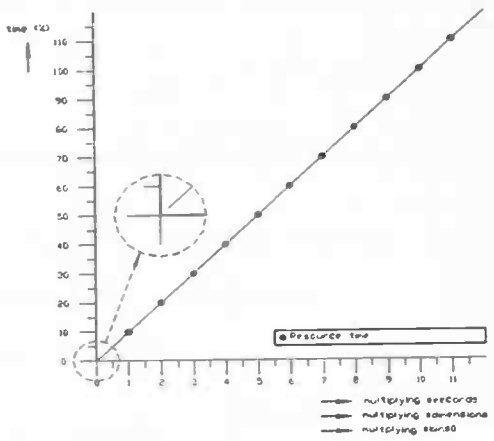


Figure 4.6 Linear demand for time.

4.2.7 Resource Space

The demand for the resource space depends on the number of records, the number of dimensions, the number of bins, and the distribution. In case of a uniform distribution, a negligible amount of main memory is required (i.e., the demand for space solely depends on array A_0 through A_{D-1}). The same amount is needed when each data point is positioned on top of another data point. Both cases are exactly the opposite of each other, but they both have the same demand for space.

When multiple bins are used, and the data points are located in the interval of just one bin, but not on top of each other, the demand for space is maximized. A maximum number of data points must be relocated to empty bins. The magnitude of such a relocation depends on the number of bins in a linear fashion. An example will clarify this. Two bins are used to represent a one-dimensional data set. A dense cluster of 100 different data points is located in the interval of the first bin. Ideally, 50 data points should be located in the interval of each bin. In order to relocate 50 data points from the first bin to the second bin, 50 data points have to be stored into main memory (i.e., into trees). In case of three bins, 67 data points have to be stored into main memory, etc. As a result, the demand for space (mainly) depends on the number of bins. The demand for space will linearly increase in case of an invariable distribution and a linear increase of the number of records or dimensions. See also Figure 4.7.

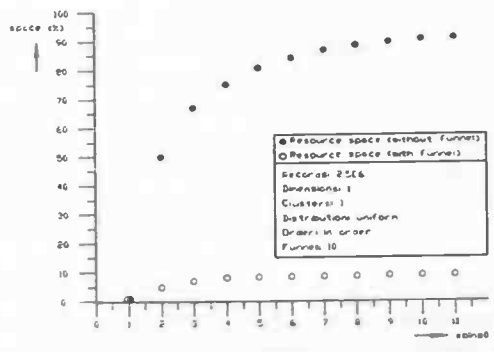


Figure 4.7 Linear demand for space.

The funnel mentioned in Figure 4.7 is a simple but very effective improvement of the algorithm. By using a funnel to store data points into the tree (i.e., each node has its own funnel), main memory usage can be drastically decreased. A funnel of 10 will funnel 10 separate values into a single node. A funnel is actually a range, and the value of the node is located somewhere in this range. An extra variable *keyLimit*, is added to each node to indicate the range of a node. Whilst determining the mirror image, the range of a funnel is determined in an on the fly manner.

A funnel dynamically divides bins into smaller bins, the sub–bins. The number of values that can be stored into a node is restricted to a maximum (10 in Figure 4.7). This maximum ensures an equalized histogram of the sub–bins, a sub–histogram. A sub–histogram is the histogram of one of the bars (i.e., bins) of the histogram of the entire dimension.

A funnel has the advantage of combining all values which differ slightly into a single node (e.g., the difference between 1.23456789 and 1.23456788 is negligible if the range R_d is substantially greater than this difference). A funnel is a generalized interpretation of compaction.

When very dense clusters are represented by many bins, a funnel is necessary to limit the demand for space. Funnelling 0.1% or less of the total number of values of a dimension into a node will hardly influence the accuracy of the resulting sample set. The funnel in Figure 4.7 funnels $\frac{10}{2.5 * 10^6} * 100\% = 0.0004\%$ of the total number of field values into a single node (0.1% equals 2500 field values).

4.2.8 Chance to Retrieve a Cluster

In case of simple random sampling with replacement, the chance

$$P = \frac{\#records\ smallest\ cluster}{\#records} \tag{4.5}$$

to retrieve a data point from the smallest cluster is relatively small. In case of deterministic sampling, P is maximized, and it practically becomes 100% (not theoretically due to round of errors and the possibility of some lost information as described in chapter three). The difference between random and deterministic sampling is clearly shown in Figure 4.8.

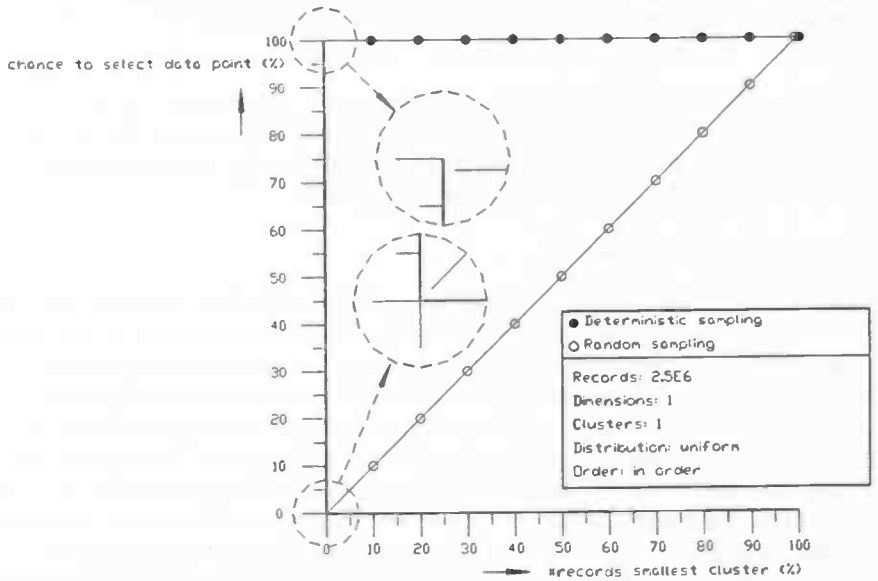


Figure 4.8 Chance to retrieve a data point from the smallest cluster.

4.3 Field-Test

4.3.1 Introduction

A field-test is conducted by sampling a large realistic data set. The sample set is applied to a common class of NNs, namely a classifying multilayer perceptron (MLP). The size of the sample set should become as small as possible, but the classifying capabilities of the resulting network should not substantially differ from a network trained with the entire data set. To provide a frame of reference, the deterministic sample algorithm is compared with simple random sampling with replacement.

The field-test is categorized by the two parameters of the deterministic sampling algorithm:

- Reduction;
- Number of bins.

These two parameters are the starting point of each test conducted. Adjustments of these parameters will produce specific results to examine:

- Accuracy (i.e., the performance of the MLP after training);
- Demand for time (i.e., sampling and training time);
- Demand for space (i.e., main memory).

4.3.2 Data Set

The data set employed is generally used to train NNs to classify the characters and numbers of registration plates of cars. It comprises specific features of the characters and numbers, and is extracted from the raw data. The raw data is obtained by taking photos of passing cars.

The specific features are not important. Regardless of the data set, the deterministic sampling algorithm should perform at least as well as random sampling. The task of the deterministic sampling algorithm is to preserve the features and the underlying relations of the data set. Simulations have indicated that making use of high ordered concentrated features to train a NN will yield very high accuracy rates of the categorization [16]. In other words, the clustering of data points is increased. The previous has shown that clustering of data points is advantageous when one wants to select a deterministic sample set.

The data set comprises 14742 records of 78 dimensions each. The number of inputs is 42, and the number of outputs (or targets) is 36. There is no reason to make a distinction between the input and output, because they represent dimensions which have to be generalized by the algorithm, i.e., each input combined with its output defines a single data point. The type of each field is a double.

4.3.3 Multilayer Perceptron

The MLP initially is a 42-10-36 network. The inputs and outputs of the network are determined by the number of inputs and outputs as defined by the data set. The number of neurons of the hidden layer is commonly used to train NNs with the entire data set to properly classify the registration plates. Since the sample set is small in comparison with the data set, the number of hidden neurons probably contains a surplus. Due to the surplus, a network tends to store the sample set instead of storing a generalization of it. In other words, it overfits due to an oversized neural network or a too small training set. To improve the generalization ability, either the network has to be reduced, or additional records have to be added to the training set [4]. The latter is not preferable, because the purpose is to minimize the training set. Therefore, the number of hidden neurons is empirically reduced to 7 neurons. Sampling will reduce the size of the training set, and, as a result of a small sized training set, the size of the network (i.e., the number of neurons of the hidden layer) can also be reduced. The number of connections of the MLP is reduced from 780 to 546. Such a reduction will especially limit the demand for time.

In order to obtain a straight comparison between the classifying capabilities of a network trained with the sample set and a network trained with the entire data set, the remaining parameters (particularly the learning rate and the momentum) are not adjusted. A well designed algorithm (or electronic circuit) requires a minimum of adjustments, regardless of the environment.

The performance of a network after applying a test set to it (i.e., the mean squared error of the test set) is examined after performing 100 epochs. The generalization capabilities of a network will improve after performing, e.g., 500 epochs, but tests have shown that the progress of the curve hardly changes after 100 epochs. Due to the number of networks that had to be trained (approximately 250), the number of epochs was limited to finalize the field-test in time. The test set consists of all available patterns, including the patterns of the train set. The latter is not a problem, because the size of the sample set is negligible in comparison with the size of the entire data set.

4.3.4 Field-Test

4.3.4.1 Introduction

The results of the field-test always indicate an average. The same test is conducted at least five times, and the classifying capabilities of the resulting networks are averaged after omitting the best and the worst network. This approach is necessary to obtain plausible data, because training a network twice with the same training set will roughly yield the same result in general. However, deviating extremes may occur. They are inherent in the learning algorithm, not in the sampling algorithm. Simply omitting the extremes will increase the feasibility of the field-test.

Randomly obtained sample sets provide a frame of reference to interpret the results of the deterministic sampling algorithm. Simple random sampling with replacement or simple random sampling without replacement are the two possibilities to choose from, because these random sampling techniques are commonly used at the department to select the sample sets to train NNs. Besides that, virtually all remaining random sampling techniques directly or indirectly use advance knowledge (e.g., by attributing a weight to the records). Since the deterministic sampling algorithm does not use advance knowledge, these techniques do not provide a proper frame of reference.

Sampling with or without replacement will not yield substantially different results, because the sample set is very small in comparison with the data set. In other words, the chance to select the same record twice is negligible (e.g., if 250 records have to be selected to constitute the sample set, then the chance to select the same record twice is $\frac{250}{14742} * \frac{250}{14742} * 100\% = 0.029\%$). Simple random sampling with replacement will suffice.

Each dimension is represented by the same number of bins to keep the field-test surveyable. An improvement of the results can be obtained by separately adjusting the number of bins per dimension.

4.3.4.2 Learning Curves

Figure 4.9 shows the typical difference between the learning curve of a network trained with a deterministic sample set, and the learning curve of a network trained with a random sample set. If the reduction is increased, the deviation D between the learning curves will also increase and vice versa.

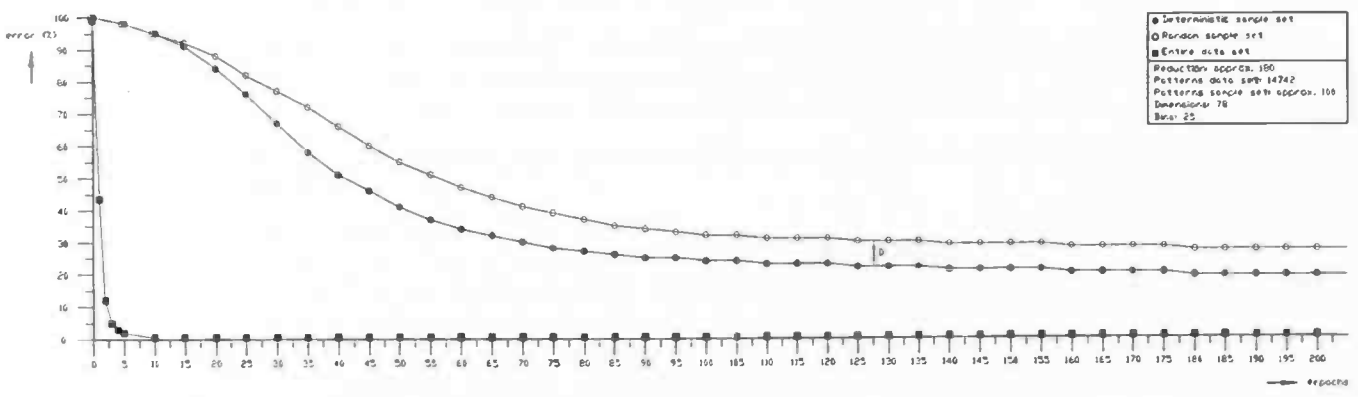


Figure 4.9 Learning curves of the MLP (obtained after performing 200 instead of 100 epochs).

A deterministic sample set generally contains more information than a random sample set. Generally, because the results are average results, and random sampling may incidentally produce a better sample set. The deviation D between the learning curves is an average measure of the difference of information between the sample sets.

The progress of both learning curves are roughly the same. This implies that when a certain reduction R is used to produce a random sample set to obtain a certain recognition rate, the recognition rate can also be obtained by applying a deterministic sample set of smaller size (i.e., R is increased). In other words, deterministic sampling increases the chance to find a suitable sample set in comparison with random sampling. This does not guarantee a low error of, e.g., 0.5%, it states that, for a given recognition rate and reduction, a deterministic sample set is smaller than a random sample set. However, this does not hold when the sample set is relatively large in comparison with the data set (e.g., the size of the sample set is half the size of the data set).

4.3.4.3 Deviation of the Mean Squared Error

The deviation D depends on the reduction R. Whilst increasing the reduction, the deviation between the two curves in Figure 4.10 will grow.

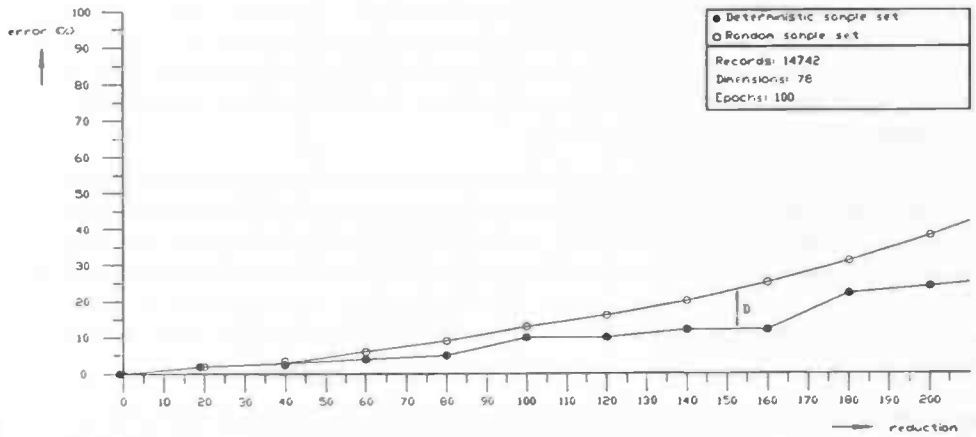


Figure 4.10 Increase of the deviation of the mean squared error. The stepwise progress of the deterministic sampling curve is typical for deterministic sampling (see also Figure 4.11).

4.3.4.4 Bin Dependency

Figure 4.11 illustrates the relation between the mean squared error and the reduction with respect to various numbers of bins. The deterministic sampling curve in Figure 4.10 is the result of taking the best results plotted in this figure.

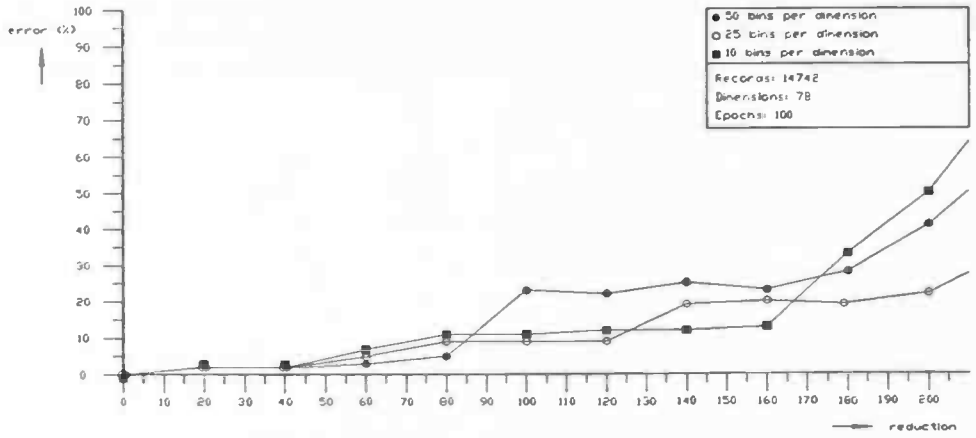


Figure 4.11 Mean squared error for various numbers of bins.

Each curve curves away at a specific point (the 50 bins curve at 80, the 25 bins curve at 120, and the 10 bins curve at 160). Curving away is the result of excluding a cluster. This is due to a reduction R which exceeds the value of a counter of some bin (i.e., the average value C_{max_d} is exceeded). The process will repeat itself until each cluster is excluded (e.g., each curve starts curving away at 40). The same behavior is shown in Figure 4.3. Comparing the random sampling curve in Figure 4.10 with the worst values of Figure 4.11 will reveal that deterministic sampling performs at least as well as random sampling.

4.3.4.5 Resource Time: Sampling Time

The deterministic sampling algorithm will always require a constant amount of time to verify the suitability of a record, because a number of counters equal to the number of dimensions is examined. In case of random sampling, the selection of a specific record by means of a randomly generated index takes a small and constant amount of time. The demand for time to select a deterministic sample set depends on the number of records in the data set, but in case of random sampling, it depends on the number of records in the sample set. Therefore, random sampling is faster than deterministic sampling.

As opposed to random sampling, deterministic sampling requires a constant amount of extra time to determine the mirror image. Once it is determined, it can be stored into a file to, e.g., determine another sample set of a different size. When the number of bins of a dimension is altered, the mirror image has to be determined again. The average time to determine a mirror image which reflects the entire data set of 14742 patterns is approximately 5 minutes. When the distribution remains the same, the time to determine the mirror image is linearly related to the number of records, the number of dimensions, and the number of bins.

When the mirror image is available, the data set is traversed a single time to determine the sample set, regardless of the size of the sample set. This results in a constant sampling time which is indicated by the upright line in Figure 4.12.

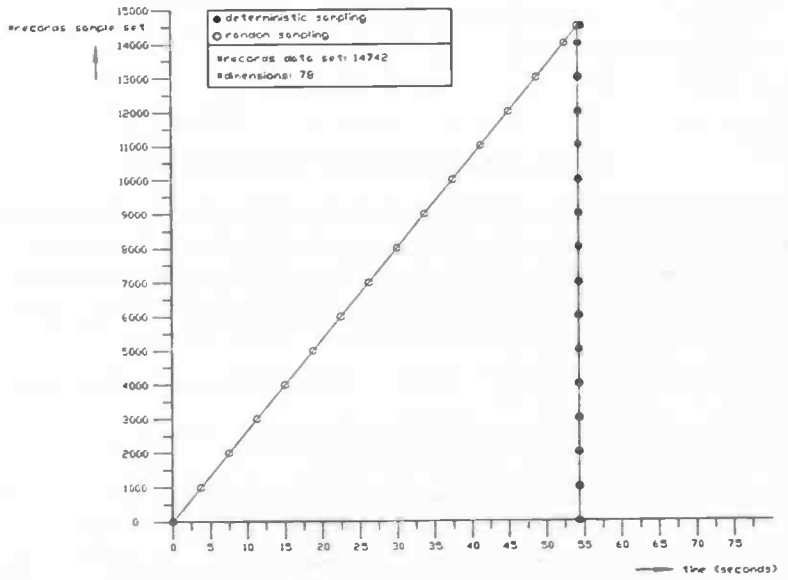


Figure 4.12 Sampling time (the mirror image is already available).

Random sampling is faster than deterministic sampling, particularly when selecting a small sample set. Improving the deterministic sampling algorithm is possible by quitting the traverse of the data set as soon as a complete sample set is collected. However, this does not change the worst case scenario.

With respect to the demand for training time, the demand for sampling time is negligible for both deterministic and random sampling.

4.3.4.6 Resource Time: Training Time

The demand for training time depends on the size of the training set. Figure 4.13 illustrates the growth of the demand for training time whilst increasing the size of the sample set. Comparing this figure with Figure 4.9, Figure 4.10, and Figure 4.11 will support the following observation. It is possible, within certain limits of the reduction, to make use of a smaller training set to train a NN without making concessions to the accuracy of the resulting network.

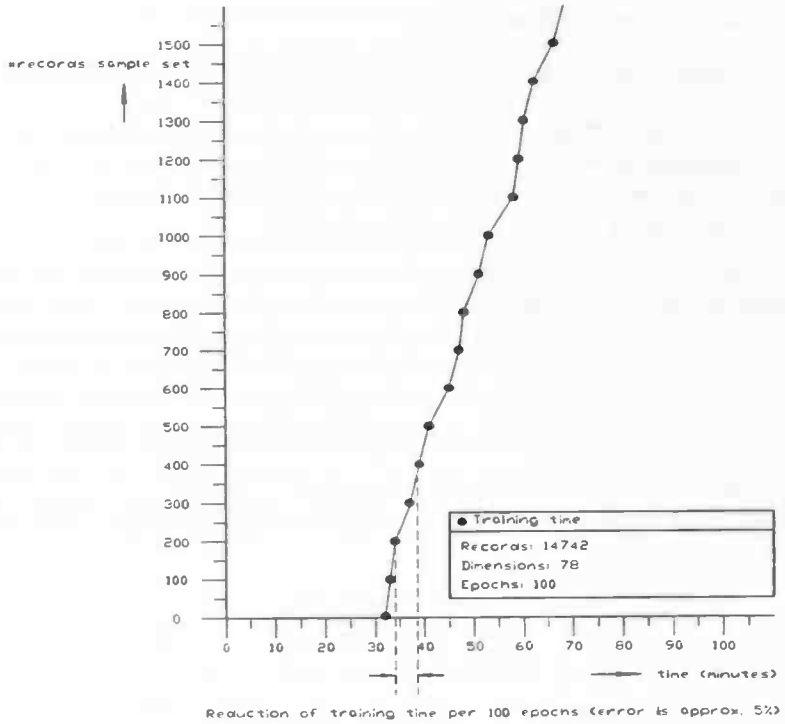


Figure 4.13 Training time of the MLP.

Training a MLP with the entire data set of 14742 patterns will approximately take 12 hours. A deterministic sample set of 250 patterns will produce a suitable network. The recognition rate of the test set is then approximately 98.0%. To attain the same accuracy by applying a random sample set, the size of the sample set has to be approximately 350 records. Figure 4.13 illustrates that the reduction of training time is about 5 minutes per 100 epochs (i.e., the training time is reduced from 39 to 34 minutes). The reduction of training time in comparison with random sampling is $\frac{5}{39} * 100\% \approx 13\%$, and in comparison with a training set which comprises the entire data set, the reduction is $\frac{12 * 60 - 34}{12 * 60} * 100\% \approx 95\%$.

Given a mean squared error (i.e., a fixed recognition rate), the size of a deterministic and random sample set will differ. Applying both sample sets to a MLP will result in different training times. The difference between the training times is nonlinearly related to the mean squared error, as shown in Figure 4.14.

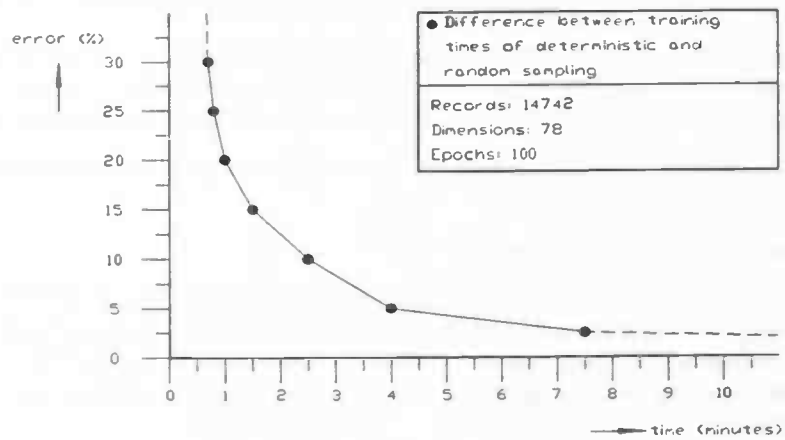


Figure 4.14 Difference between the training times of a MLP trained with a deterministic and a random sample set respectively.

4.3.4.7 Resource Space

Whilst sampling, the demand for space particularly depends on the number of bins. Figure 4.15 shows the relation between the demand for space and the number of bins.

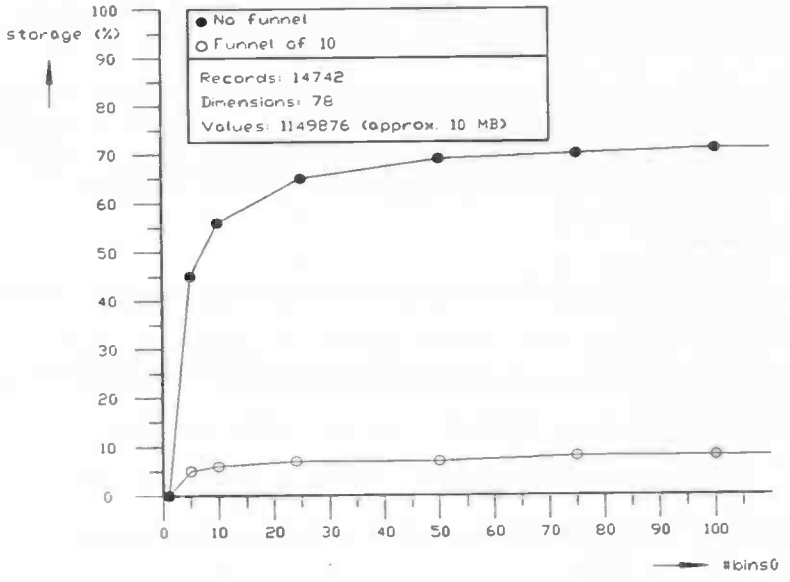


Figure 4.15 Occupied main memory.

A funnel drastically decreases the demand for space. A narrow funnel does not noticeably influence the selection of the sample set, but the algorithm runs faster (i.e., less tree nodes to visit), and makes use of less space (i.e., less tree nodes to store into main memory). The funnel in Figure 4.15 funnels 10 values into a single node ($\frac{10}{14742} * 100 = 0.068\%$ of the available values of a dimension are stored into a node).

4.3.5 Data Set Acknowledgement

Suppose that a cluster of 5 data points exists in some data space, and that the total number of data points in that data space is equal to 10. The chance to randomly select one of the 5 data point is 50%, but if the total number of data points becomes 1000, the chance is decreased to 0.5%. When the chance to select a data point within a specific cluster is small, deterministic sampling is the best candidate to select a sample set. The field-test is based on a relatively small data set which comprises one large and dense cluster (the progress of both curves in Figure 4.11 implies that the data set contains one large and dense cluster). Due to the lack of relatively small clusters, this data set cannot maximally show the benefits of the deterministic sampling algorithm. Nevertheless, the adverse effect of the data set cannot prevent that random sampling is still outperformed.

4.4 Summary

4.4.1 General Summary

The limited amount of the resources time and space will generate problems when a too large data file is applied to an algorithm (e.g., a machine learning algorithm). The joint cause of these problems (i.e., a too large data set) implies a single solution.

A common approach to overcome resource related problems is to sample the original data set to produce a smaller sample set. Sample sets are usually obtained by random sampling, but this technique has the inability to guarantee that the proportion of the relations between the clusters is preserved. Ideally, expressing this guarantee as a percentage should at least yield an approximation of 100%. This can only be attained by changing the nature of the algorithm, it should be deterministic instead of random.

Standard addressing techniques (e.g., B-trees) require a specific structure which is independent of the remaining part of the (learning) algorithm. This is not preferable, because it increases the demand for the resources

time and space. As said, both problems have the same cause, and, consequently, a deterministic sampling algorithm has to have the ability to retrieve records from disk or the like. In other words, the data structure which is used to sample the data set should be (reversely) used to retrieve a specific record.

The deterministic sampling algorithm is based on a representation of the data set, a so–called mirror image. On the principle that the mirror image is an efficient reflection of the data set, decisions are made to determine whether or not a record meets the requirements of the sample set. This process is entirely deterministic.

The deterministic sampling algorithm makes uses of the mirror image to select a sample set by verifying the suitability of a record. By changing the premise of the mirror image, it can be used to retrieve a specific record from the data set. Deterministic sampling and deterministic addressing are similar processes. As a result, the mirror image provides a feasible solution to both resource related problems.

4.4.2 Conclusions

- Deterministic sampling provides a feasible solution to the problem related to the resource time (e.g., training time);
- Deterministic addressing provides a feasible solution to the problem related to the resource space (e.g., main memory).

Chapter 5 Conclusions

The limited amount of the resources time and space will generate problems when a too large data file is applied to an algorithm (e.g., a machine learning algorithm). A feasible solution to these problems is provided by deterministic sampling (related to the resource time) and deterministic addressing (related to the resource space).

Deterministic sampling and deterministic addressing both use the same representation of the data set, a so-called mirror image. Except for their premise, deterministic sampling and deterministic addressing are identical, because both resource related problems have a joint cause (i.e., a too large data set), and a joint cause implies a joint solution.

5.1 Advantages

- A deterministic sample set reduces the training time of a NN, making no concessions with respect to the accuracy within certain limits of the reduction;
- The deterministic sampling and addressing algorithm both use linear time and space;
- A deterministic sample set contains a constant amount of information;
- Deterministic sampling performs at least as well as random sampling, but it generally outperforms random sampling;
- The mirror image can be used for both sampling and addressing, both processes are the same;
- The training set or a portion of it does not have to be loaded into main memory, loading one record (or disk block) a time will suffice;
- The mirror image requires a small amount of main memory, particularly in comparison with commonly used data structures;
- The mirror image can provide specific information on the data set (e.g., statistics);
- Increasing the number of records, the number of dimensions, and / or the reduction will increase the efficiency of the mirror image.

5.2 Disadvantages

- The number of bins to represent a specific dimension is an important parameter which must be carefully adjusted to prevent an inaccurate mirror image;
- The number of bins is inversely related to the reduction (i.e., a small sample set implies less bins, and an accurate sample set implies more bins);
- Random sampling is faster than deterministic sampling, and it requires less main memory;
- Both deterministic and random sampling perform equally well when the sample set is relatively large in comparison with the data set;
- The mirror image cannot preserve the entire information contents of the data set;
- The mirror image provides a linear representation of a nonlinear data space which does not immediately agree with intuition.

References

[1] H.J.C. Berendsen, *Information Theory.* Rijksuniversiteit Groningen, Groningen, The Netherlands, 1993.

[2] A. van den Bosch, T. Weijters and J. van den Herik, "Scaling Effects with Greedy and Lazy Machine-Learning Algorithms," in *Proceedings of the 7th Dutch Conference on Artificial Intelligence,* pp. 211-218, 1995.

[3] J. Catlett, *Megainduction: machine learning on very large databases.* Ph.D. Thesis, University of Sydney, Sydney, Australia, June 1991.

[4] S. Cho and K. Cha, "Evolution of Neural Network Training Set through Addition of Virtual Samples," in *Proceedings of the 1996 IEEE International Conference on Evolutionary Computing,* pp. 685-688, Nagoya, Japan, May 1996.

[5] J.M. Chambers, W.S. Cleveland, B. Kleiner and P.A. Tukey, *Graphical Methods for Data Analysis.* Duxbury Press, Boston, 1983.

[6] M.W. Craven and J.W. Shavlik,"Using Neural Networks for Data Mining," *Future Generation Computer Systems,* vol. 13, pp. 211-219, 1997.

[7] T.G. Dietterich, "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms," *Neural Computation,* December 1997.

[8] W. Duch, "Scaling Properties of neural classifiers," *Third Conference on Neural Networks and Their Applications,* pp. 189-194, October 1997.

[9] R. Elmasri and S.B. Navathe, *Fundamentals of Database Systems.* Benjamin/Cummings, 1994.

[10] S.E. Fahlman, "An Empirical Study of Learning Speed in Back-Propagation Networks," CMU-CS-88-162,Carnegie Mellon University, USA, September 1988.

[11] D.H. Fisher and K.B. McKusick, "An Experimental Comparison of ID3 and Back-propagation," in *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence,* Detroit, USA, pp. 778-793, 1989.

[12] A. Flexer, "Statistical Evaluation of Neural Network Experiments: Minimum Requirements and Current Practice," in *Proceedings of the 13th European Meeting on Cybernetics and Systems Research,* pp. 1005-1008, Vienna, Austria, 1996.

[13] R.C. Gonzalez and R.E. Woods, *Digital Image Processing.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1993.

[14] S. Haykin, *Neural Networks, A Comprehensive Foundation.* New Jersey: Prentice Hall, 1994.

[15] G.H. John and P. Langley, "Static Versus Dynamic Sampling for Data Mining," in *Proceedings of the Second International Conference on Knowledge Discovery in Databases and Data Mining,* pp. 367-370, 1996.

[16] N. Kadaba, K.E. Nygard, P.L. Juell and L.J. Kangas, "Modular Back-Propagation Neural Networks For Large Domain Pattern Classification," in *Proceedings of the International Joint Conference on Neural Networks,* vol. II, pp. 607, June 1989.

[17] H. Lu, R. Setiono and H. Liu, "Effective Data Mining Using Neural Networks," *IEEE Transactions on Knowledge and Data Engineering,* vol. 8, no. 6, pp. 957-961, December 1996.

[18] M. Misra, "Parallel Environments for Implementing Neural Networks," *Neural Computing Surveys,* vol. 1, pp. 48-80, 1997.

[19] R. Moddemeijer, *Beeldbewerking.* Rijksuniversiteit Groningen, Groningen, The Netherlands, August 1997.

[20] F. Olken, *Random Sampling from Databases.* Ph.D. Thesis, University of California, Berkeley, USA, April 1993.

[21] L. Prechelt, *Proben 1 - A Set of Neural Networks Benchmark Problems and Benchmarking Rules.* Technical Report 21/94, Universität Karlsruhe, Germany, September 1994.

[22] F.J. Provost and D.N. Hennessy, "Scaling Up: Distributed Machine Learning with Cooperation," in *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, Portland, OR, USA, pp. 74–79, 1996.

[23] E. Radeke and M.H. Scholl, "Federation and Stepwise Reduction of Database Systems," in *Applications of Databases, Proceedings of the First International Conference*, ADB–94, pp. 381–399, June 1994.

[24] J. Shafer, R. Agrawal and M. Mehta, "SPRINT: A Scalable Parallel Classifier for Data Mining," in *Proceedings of the 22nd VLDB Conference*, Bombay, India, pp. 544–555, 1996.

[25] P.K. Simpson, "Foundations of Neural Networks," *Artificial Neural Networks, paradigms, applications, and hardware implementations*, IEEE Press: New York, pp. 3–24, 1992.

[26] H.–H. Song and S.–W. Lee, "A Self–Organizing Neural Tree for Large–Set Pattern Classification," *IEEE Transactions on Neural Networks*, vol. 9, no. 3, pp. 369–380, May 1998.

[27] J.S. Vitter, "An Efficient Algorithm for Sequential Random Sampling," *ACM Transactions on Mathematical Software*, vol. 13, no. 1, pp. 58–67, March 1987.

[28] B. Verma, "Fast Training of Multilayer Perceptrons," *IEEE Transactions on Neural Networks*, vol. 8, no. 6, November 1997.

[29] J.–L. Yuan and T.L. Fine, "Neural–Network Design for Small Training Sets of High Dimension," *IEEE Transactions on Neural Networks*, vol. 9, no. 2, pp. 266–280, March 1998.

[30] G. Zhou, "Advanced Neural–Network Training Algorithm with Reduced Complexity Based on Jacobian Deficiency," *IEEE Transactions on Neural Networks*, vol. 9, no. 3, pp. 448–453, May 1998.