CLASSIFICATION USING NORMALIZED COMPRESSION DISTANCE

Uri Shaham
M.Sc. thesis
Department of Industrial Engineering and Management
Ben-Gurion University of the Negev
Beer-Sheva, ISRAEL

Advisors:
Prof. Yael Edan
Dr. Joel Ratsaby

20/2/2009

Abstract

Classification algorithms that use the NCD (Normalized Compression Distance) as a similarity metric are proposed. This way of measuring similarity allows either skipping the feature selection and feature extraction phases or creating feature vectors without focusing on specific attributes of the data. Such an approach might be more objective than common feature extraction methods, is suitable to a wide range of classification problems and data types, and might be less biased than methods that are based on explicit extraction of features.

The thesis includes classification experiments of images of several types, embedding different challenges to the classifier, OCR (Optical Character Recognition), voice samples and time series of ECG signals. Two different NCD-based classification methods- support vector machines and k nearest neighbor were employed. In several cases, the performance of the NCD-based algorithms was compared to the performance of popular state-of-the-art classification techniques. Results are comparative to those achieved by more complicated, domain-specialized learning algorithms that are based on an integration between data processing, feature extraction and classification. A special emphasis was given to domains on which it is hard to extract explicit features that can be related to the labeling of the instances, such as in the tasks of emotion recognition through voice samples and pain detection through ECG signals. On such domains, the NCD-based algorithms, that define and extract features implicitly, have a great potential to succeed where other common feature extraction methods and classification algorithms fail.

Contents

List of Tables

List of Figures

Acknowledgement

The experiments with voice samples were performed in collaboration with Dr. Tal Sobol-Shikler and Ms. Einat Sukenik, Department of Industrial Engineering and Management, Ben Gurion University of the Negev. In particular, the emotion recognition experiments were inspired by Tal's work [42]. The ECG experiments were done in collaboration with Dr. Shai Tejman-Yarden (Soroka hospital), Mr. Alex Beyzarov from the Department of Biomedical Engineering and Mr. Eran Tomer from the Department of Computer Science, both at Ben Gurion University. The ECG recordings were taken in Soroka hospital's pain clinic. The olives classification experiments were done in collaboration with Dr. Shahar Laykin, Ben Gurion University, based on data from Shahar's work [26].

I would like to thank my advisors, Dr. Joel Ratsaby, for exposing me to the fascinating worlds of machine learning and Kolmogorov complexity and Prof. Yael Edan, for her kind and professional assistance and supervision. I am most pleased to thank more people who contributed to this work: Dr. Rudi Cilibrasi, University of Amsterdam, Holland, for his kind assistance in NCD-related issues, Prof. Eamonn Keogh, University of California, Riverside, for his assistance with compression based classification issues, Dr. Uri Kartoun, Ben Gurion University, for his help in designing the kNN division modification, Dr. Ofer Levi, Dr. Boaz Lerner, Dr. Liron Yedidsion, Dr. Dvir Shabtay, Mr. Amit Assa, Mr. Nir Naor and Mr. Amit Gil from the Department of Industrial Engineering and Management, and Mr. Dudi Cohen and Mr. Oded Judelman from the Department of Computer Science, all from Ben Gurion University.

Notations

1. $\{0,1\}^*$ refers to the set of all finite binary strings.

2. ε refers to the empty binary string.

3. $\lceil \cdot \rceil$ refers to the ceiling operation that returns the smallest integer that is equal to or greater than its argument.

4. For $x, y \in \mathbb{R}^n$, $\langle x, y \rangle$ refers to the inner (dot) product $\sum_{i=1}^{n} x_i \cdot y_i$

5. Consider a sample of n observations, each of k measurements. The term *instance* refers to an observation. The term *feature* refers to a measurement. A *dataset* in a classic form is a $n \times k$ matrix in which each row refers to an instance and each column to a feature. Each instance might get a *label (target value)*, which is a discrete numerical value from a finite set L. In a given dataset, the set of all instances with same label is called a *class*.

1 Introduction

This thesis focuses on incorporation of a relatively new similarity measure, the Normalized Compression Distance (NCD) defined by Cilibrasi et al. [12] in the classical task of classification. NCD-based learning algorithms are robust to very large scale of data types, and are feature-free (i.e., no prior knowledge of the input is required). To be more precise, NCD-based learning algorithms are based on similarity between objects, as many other learning algorithms. However, the similarities considered by the NCD are far from being trivial, where trivial similarities usually implies similar values of certain pre-defined features. Rather, the similarities considered by the NCD are those found by data compressors, using the fact that combining two files to a single conglomerate archive file prior to compression often achieves better compression than the compression of the two files separately [10]. One very important application area of NCD-based classification algorithms is where the definitions and roles of the features determining the target values of the instances are vague. This issue will be demonstrated in depth as a part of this thesis.

This thesis provides a thorough investigation of applying NCD-based classification algorithms to 7 different datasets and multiple types of data and classification tasks: face recognition, texture and fruit classification, optical character recognition, speaker identification, capturing emotion and pain detection. As far as the author knows, there is no such extensive NCD-based classification experimentation previously documented. This thesis aims to show the robustness of the NCD-based algorithms to many types of data inputs and its success in areas where it is hard to define potential relevant features of the data that effect the target values, such as in capturing emotion and pain detection. The experiments in this thesis are inspired by the interesting experimentations of [12], [24] and [16].

1.1 Classification and feature free algorithms

The field of pattern recognition consists of three main consecutive processes: data processing, in which manipulations are performed on the raw data in order to identify and remove unnecessary information and emphasize important details, dimensionality reduction, in which the data is converted to a more convenient representation, and learning, where new knowledge of the data is obtained. Classification is one of the typical learning tasks. Other such tasks include clustering, anomaly detection and more. Classification is the task of assigning labels to unknown test objects given a set of labeled training objects from a human expert. The goal is to try to learn the underlying patterns that the human expert is displaying in the choice of labeling shown in the training objects, and then to apply this understanding to the task of making predictions for unknown objects that are in some sense consistent with the given examples. Usually the problem is reduced to a combination of binary classification problems, where all target labels along a given dimension are either 0 or 1 [19], [7], [34], [22], [33], [2].

Most machine learning algorithms, and classification algorithms in particular, require input of feature vectors representing the instances to be studied. The basic features are usually first defined by a human expert. On that basis, more complex features can be later on created by feature extraction and selection techniques. The very first definition of the features usually requires prior knowledge about the domain from which the instances come from [34]. There are some major problems embedded in defining, selecting and extracting input features [24]. Ignoring important features may cause an algorithm to fail in finding the true patterns. Another problem is that the algorithm may report spurious patterns that do not really exist, or greatly overestimate the significance of the reported patterns. This is sometimes likely when the user fails to understand the role of defining features and of tuning parameters. A feature free algorithm would limit the ability to impose prejudices, expectations, and presumptions on the problem at hand, and would let the data itself speak to us. This is exactly the sense in which NCD-based algorithms are feature free: NCD-based algorithms require no prior knowledge of the instances or the domain they come from and do not analyze the data looking for particular features. More precisely, feature extraction does take place in NCD-based algorithms, however not by analyzing the data looking for particular features, but by data compressors. In addition, the NCD of each pair of instances might be determined by a different dominating feature in which they are most similar, as will be discussed further on.

1.2 Objectives

This research aims to demonstrate the robustness of NCD-based methods, as well as their capability to be applied to a large scale of classification problems, by performing NCD-based classification experiments of various types of data. In several cases the performance of NCD-based methods will be compared to other state-of-the-art domain specialized classification methods. In addition, a sensitivity analysis of the performance will be presented, in which the effects of changes in several experimental parameters will be demonstrated, and integration with several classifiers and dimensionality reduction methods will be conducted. An important goal of this work is to demonstrate the potential in applying NCD-based classification methods to domains where it is hard to identify and extract specific features of the instances that might determine their labeling. On one such domain, recognition of speaker's emotion through voice samples, a thorough analysis of the performance of the NCD-based algorithm will be provided.

1.3 Thesis structure

Chapter 2 contains the technical details and terminology needed for the classification methods, in particular the basic notions from the theory of Kolmogorov complexity, the Normalized Information Distance (NID) and the normalized compression distance, Support Vector Machines (SVM), and other works regarding compression-based learning. Chapter 3 describes the research phases and methods. Experimental results are summarized in Chapter 4. The thesis concludes with a discussion and recommendations for future research.

2 Literature overview

2.1 Classical information theory: entropy and coding

This subsection is based on [41], [4].

2.1.1 Prefix codes

Definition 1 *A* code *is defined by a decoding function D which is a mapping from a set $Y \subseteq \{0,1\}^*$ to some arbitrary set X. The elements of Y are called* code words*. The elements of X are called* source words*. if D^{-1} exists, it is called the* encoding function*.*

Definition 2 *A binary string y is a* proper prefix *of a binary string x if we can write $x = yz$ for $z \neq \varepsilon$. A set $\{x, y, ...\} \subseteq \{0,1\}^*$ is* prefix-free *if no element is a proper prefix of any other.*

Definition 3 *A* prefix code *is a code where the set of code words is prefix free.*

Example 1 *A set $X \subseteq \{0,1\}^*$ can be encoded to a prefix free set by encoding the string x of length n by $\overline{x} = 1^n0x$. This requires $2n+1$ bits to encode x. To get a more economic prefix encoding, one can encode x by first $\lceil \log n \rceil$ ones, then a 0, then the binary number n (which will require $\lceil \log n \rceil$ bits), then x literally. This will require only $n + 2\lceil \log n \rceil + 1$. One can repeat this to get $n + \lceil \log n \rceil + 2\lceil \log \log n \rceil + 1$, or as many logarithms as necessary.*

There is a precise constraint on the number of code words of given length in prefix codes. It is called the *Kraft Inequality* and is due to L.G Kraft [25]:

Lemma 1 *Let $l_1, l_2, ...$ be a finite or infinite sequence of natural numbers. There is a prefix code with this sequence as lengths of its binary code words iff*

$$\sum_n 2^{-l_n} \leq 1$$

2.1.2 Entropy and coding

A basic assumption of classical information theory is that each message belongs to a set of messages that is known both to the sender and receiver [41]. According to classical information theory, information is defined as the ability to choose a certain message from the set of possible messages. Intuitively, the information in an object is to what extent it "surprises you": the more probable messages the set contains, the more information it holds. The term *entropy* (in its information context), invented by Claude Shannon in his famous paper [41], refers to the amount of information in a given random variable:

Definition 4 *Let X be a discrete random variable over $\{x_1, ..., x_n\}$ that gets the value x_i with probability p_i. The* entropy *of X is:*

$$H(X) = -\sum_{i=1}^{n} p_i \log_2(p_i) \tag{1}$$

It can be seen that the entropy of a random variable is maximal when all its possible values are equiprobable. The entropy of a discrete random variable X equals to the minimal number of bits required for efficient description of each possible value of X.

Redundancy of a discrete random variable is defined as:

$$R(X) = -\sum_{i=1}^{n} \frac{1}{n} \log(\frac{1}{n}) - (-\sum_{i=1}^{n} p_i \log p_i)$$

That is, redundancy is the difference between current state and maximal entropy state. If the probabilities are not equal then there exist more probable messages, for whom we can supply shorter descriptions.

Example 2 *Let us consider a 4-letter alphabet with equal probabilities. The entropy will therefore be: $H = -4(0.25 \log 0.25) = 2$. It means that we'll need at least 2 bits for describing each symbol (it will be possible with the set of codewords $\{00, 01, 10, 11\}$ for instance). The redundancy will be 0 and no compression can be made. Now let us assume that the occurrence probabilities are not equal, but $0.49, 0.25, 0.25, 0.01$. We will get $H = 1.57$. If we use the above coding we'll get redundancy of $R = 2 - 1.57 = 0.43$, hence we'll look for a code with different codeword lengths. The code 1,01,000,001, for instance, has an average codeword length of 1.77 (while taking into account the probabilities) hence is more economic than the previous one for the given probabilities. Note that the above code is a prefix code. The Shannon-Fano code is a prefix code for a given set of occurrence probabilities [4]. It encodes a source word x by a code word of length $\left\lceil log\frac{1}{p(x)}\right\rceil$ therefore for an information source X, the expected transmitted codeword length is $\sum_x p(x) \log \frac{1}{p(x)} = H(X)$. It is optimal, (that is, has minimal average codeword length) by Shannon's noiseless coding theorem [41].*

2.2 Kolmogorov complexity

Many binary strings have shorter descriptions than their trivial ones. For example, one can describe the binary string 11...1 (10,000 1's) by the computer program defined in Algorithm 1:

Algorithm 1

```
step 1: for i:=1 to $10,000$
step 1.1: print 1
```

The size of the binary encoding of this program is $\lceil \log 10,000 \rceil + O(1)$ bits (the log term is needed for the binary representation of 10,000). Similarly, there are very short computer programs for the computation of the first n digits of infinitely long sequences like π or e. The theory of Kolmogorov complexity uses this fact to quantify the amount of information in a binary string. The following section brings the important notions and most relevant material from the theory of Kolmogorov complexity. Subsection 2.2.1 is based on [3]. Subsections 2.2.2, 2.2.3 are based on the very detailed description in [29].

2.2.1 Turing machines

A *Turing machine* is an abstract mathematical model of a computer, that was invented by Alan Turing in 1936. Turing machines are the common model used in computer science for analyzing time and space complexities of algorithms. It receives as its input a string of symbols, which may be thought of as a "program", and it outputs the result of running that program, which amounts to transforming the input using the given set of rules. A deterministic Turing machine consists of:

- A tape, which is divided into cells, one next to the other. Each cell contains a symbol from some finite alphabet. The alphabet contains a special blank symbol and one or more other symbols. The tape has a left end, marked with a special symbol and is assumed to be arbitrarily extendable to the right, i.e., the Turing machine is always supplied with as much tape as it needs for its computation. Cells that have not been written to before are assumed to be filled with the blank symbol. Initially, the cells are all filled with blanks, except for a binary string (the "program") written on the left end side of the tape.

- A reading/writing head that can read and write symbols on the tape and can move a single step to the left or right at a single time step.

- A rule table (transition function) of instructions (each rule is a 5-tuple) that, given the state the machine is currently in and the symbol it is reading on the tape tells the machine to do the following in sequence: (*i*) either erase or write a symbol, and then (*ii*) move the head one step in one of the possible two directions, and then (*iii*) assume the same or a new state as prescribed. This function defines the behavior of the Turing machine at each step, allowing it to perform simple actions and run a program on a tape just like a real computer but in a very mathematically simple way.

- A finite state register that stores the state of the Turing table. The number of different states is always finite and there is one special start state with which the state register is initialized. There are two terminating states (with which the machine halts): accepting state and rejecting state.

Some problems require a slight enhancement of the Turing machine structure, to a model with an input (read only) tape, an output (write only) tape, and a finite number of work tapes. It is a well known fact in computational complexity that such machines have an identical computation power as single tape machines, means that for a concrete program, the order of magnitude of the time and space needed for execution is identical on single and multiple tape machines. One can represent a Turing machine T by a function Φ, which is the mapping of all programs on which T halts to their outputs. We say that a Turing machine is *universal* if it can simulate any other Turing machine. When such universal Turing machine receives as input a pair $< x, y >$, where x is a formal specification of another Turing machine T_x, it outputs the same result as one would get if one would input the string y to the Turing machine T_x. It turns out that we can choose a particular set of state-transition rules such that the Turing machine becomes universal in this sense.

A *prefix Turing machine* is a Turing machine whose set of halting programs is a prefix set, that is, no halting program is a prefix of another halting program. Just as there are universal "ordinary" Turing Machines, there are also universal prefix machines that have identical computational power [29].

2.2.2 The Kolmogorov complexity function K

The Kolmogorov complexity of the binary string x, with reference to the concrete Turing machine T, represented by the function Φ is written $K_\Phi(x)$ and represents the size in bits of the shortest binary program for T with which T prints x and halts. In other words, $K_\Phi(x)$ represents the minimum program size $|z|$ over all programs z such that $\Phi(z) = x$. In practice, we would like to use a Turing machine that is as general as possible, therefore it is convenient to use a universal prefix Turing machine. Since all universal Turing machines can simulate each other, it does not matter which one we take. This implies that all variations of K are in some sense equivalent, because any two different variants of K given two different reference universal Turing machines will never differ by more than a fixed-size constant that depends only on the particular Turing machines chosen and not on the sequence. Therefore, one can simply use the short term $K(x)$. Intuitively, $K(x)$ is the minimal amount of information needed to create x by any reliable efficient computerized process.

Another form of the Kolmogorov complexity function K is the conditional form: $K(x \mid y)$ is defined as the size of the shortest binary program required to create x given a prefix free encoding $\overline{y}$ for y as an external input to the program. This model is supported by a model of a multiple tape Turing machine, with regular input, work and output tapes, and another limited functionality input tape, that supports only the operation "read next symbol". The idea is that if y gives a lot of information on x then $K(x \mid y) \ll K(x)$. On the other hand, if y gives almost no information on x then $K(x \mid y) \approx K(x)$. Using the conditional notation, we can consider $K(x)$ as $K(x \mid \varepsilon)$.

One more sophisticated form of K is $K(x, y)$, stands for the minimal length of program needed to output the binary string x followed by y, plus a way to distinguish between them. A convenient way is outputting $\overline{x}y$, where $\overline{x}$ is a prefix encoding of x.

Remark 1 *Following the example in section 2.1, one can immediately obtain the following upper bound:*

$$K(x) \leq n + \log n + O(\log \log n). \tag{2}$$

2.2.3 Incomputability of K

The major drawback in K is that it is incomputable. Assuming the computation of $K(x)$ for every binary string x is possible leads to solving the famous Halting Problem for every binary input, which is, of course, an undecidable problem. "The good news" though, are that K is upper semi-computable, namely can be approximated from above by a computable function $f(x, t)$, decreasing in t, such that

$$\lim_{t \to \infty} f(x, t) = K(x)$$

(one can consider t as number of computation steps).

2.2.4 H vs. K

The major difference between the two terms is that entropy is defined for random variables (or information sources), that is, the information in a message depends on the set of messages it belongs to. In Kolmogorov complexity theory, information in a binary string depends on the string itself. This property also helps when trying to approximate how random a string is. Nevertheless, there are some basic (in)equalities in the classical information theory that have their equivalent form in the Kolmogorov complexity theory, as shown in Table 1:

Table 1: Information theory vs. Kolmogorov complexity theory

Information theory	Kolmogorov complexity theory
$H(x,y) = H(x) + H(y \mid x)$	$K(x,y) = K(x) + K(y \mid x) + O(\log \max\{\lvert x \rvert, \lvert y \rvert\})$
$H(x,y) \leq H(x) + H(y)$	$K(x,y) \leq K(x) + K(y)$
$H(y \mid x) \leq H(y)$	$K(y \mid x) \leq K(y)$

2.3 Normalized Information Distance (NID)[30]

2.3.1 Information Distance

Let $x, y \in \{0,1\}^*$ and denote $K(x,y) - \min\{K(x), K(y)\}$ by $E(x,y)$. As shown above, $K(x,y)$ represents the information needed to create the pair (x,y) by an efficient computation process. Without loss of generality assume $K(x) \leq K(y)$, hence $E(x,y)$ stands for the amount of information needed for printing the pair (x,y) and does not exist in x, that is, the amount information in y that does not exist in x. In this manner, $E(x,y)$ represents the *information distance* between x and y. In other words, $K(x,y) - \min\{K(x), K(y)\}$ is, up to a negligible logarithmic term, the length of the shortest binary program in the reference universal computing system, that computes output y from input x and output x from input y [14].

Definition 5 *A distance function $D : \{0,1\}^* \times \{0,1\}^* \to \mathbb{N}$ is called an* admissible distance *if it is positive, symmetric, and computable, namely there is a prefix program that, given two strings x and y, has binary length that equals to the distance $D(x,y)$ between x and y [14] (in simple words, E is admissible distance if it can be easily computed for every two strings).*

An important property of $E(x,y)$ is its *universality* upon the class of admissible distances. It states that given any $x, y \in \{0,1\}^*$ and an admissible distance D,

$$E(x,y) \leq D(x,y) + c_D$$

where c_D is a constant depending on D but not on x, y ([6], theorem 4.2). In that case one can say that $E(x,y)$ *minorizes* $D(x,y)$ up to an additive constant. Then E is universal for the family of admissible distances because it minorizes every member of that family up to an additive constant [30]. In other words, $E(x,y)$ captures all computable similarities between x, y because if x, y are close according to a computable distance function D (which satisfies a reasonable density constraint, see Remark 2), they are at least that close according to E. Since every feature in which one can compare the two strings can be quantified in terms of distance, and every distance can be viewed as a difference in the value of a particular feature between the two strings, $E(x,y)$ minorizing the most dominant feature in which x and y are similar. Note that when considering more than two strings, the distance E may be based on a different dominating feature for each pair of strings.

Remark 2 *If D is an admissible distance, then for every $x \in \{0,1\}^*$ the set $\{D(x,y) : y \in \{0,1\}^*\}$ is the length set of a prefix code, hence according to Lemma 1 satisfies density condition derived by the* Kraft inequality

$$\sum_y 2^{-D(x,y)} \leq 1.$$

2.3.2 Normalization

$E(\cdot,\cdot)$ is an absolute distance. However, large strings that differ by some fixed amount of information are intuitively much closer than short strings that differ by same amount of information, therefore one would like to normalize $E(\cdot,\cdot)$ into a relative information distance.

Based on [6], Li et al. [30] define the *Normalized Information Distance (NID)* for every two binary strings x, y by

$$NID(x,y) = \frac{K(x,y) - \min\{K(x), K(y)\}}{\max\{K(x), K(y)\}}. \tag{3}$$

Recall from Table 1 that $K(x,y) - K(x) \leq K(y)$, hence the denominator is a normalization of $E(\cdot,\cdot)$ to the range $[0,1]$. In order to comprehend the behavior of the NID, let us take a closer look at the two edge cases:

- Assume $x = y$: then up to $O(\frac{1}{K(x)})$ term $K(x,y) = K(x) = K(y)$ because no information is needed to create a copy of x when x is given, and $K(x,y) - \min\{K(x), K(y)\} = 0$, hence we get $NID(x,y) = 0$.
- Assume the information needed to create x has no overlap with the information for y: then $K(x,y) = K(x) + K(y)$. Without loss of generality $K(x) \leq K(y)$, and we get

$$NID(x,y) = \frac{K(y)}{K(y)} = 1$$

From its definition, the NID is symmetric, and positive. In addition, up to an additive precision $O\left(\frac{1}{K}\right)$ where K is the maximal Kolmogorov complexity of the strings involved, the NID satisfies also the triangle inequality, namely $NID(x,z) \leq NID(x,y) + NID(y,z)$ for every binary strings x, y, z. Therefore, together with the first case above, the NID satisfies the metric (in)equalities up to the above precision.

2.3.3 Universality

Similarly to the distance E, also the NID has the important *universality* property, that justifies calling it *the* similarity metric. This time, the universality refers to the class of computable normalized distances:

Definition 6 *A function* $d : \{0,1\}^* \times \{0,1\}^* \to [0,1]$ *is a* normalized distance *if it is symmetric and for every* $x \in \{0,1\}^*$ *and constant* $e \in [0,1]$*,*

$$|\{y : d(x,y) \leq e \leq 1\}| < 2^{eK(x)+1}.$$

This density constraint is implied by distances $d : \{0,1\}^* \times \{0,1\}^* \to [0,1]$ *that obey a normalized version of the* Kraft inequality

$$\sum_y 2^{-d(x,y)K(x)} \leq 1.$$

The universality property of the NID states that for a normalized computable distance d and every $x, y \in \{0,1\}^*$

$$NID(x,y) \leq d(x,y) + O\left(\frac{1}{\min\{K(x),K(y)\}}\right)$$

In Appendix 7.1 we bring the proof of the universality property of the NID.

2.4 Normalized Compression Distance (NCD)

The NCD, the main focus of this work, was introduced and analyzed by Cilibrasi et al. [12], [10].

2.4.1 Approximating K by compressors

The basic idea in data compression is to encode a string from a biased information source (that is, one which is not uniform distribution) to a minimum number of bits, in a way that for some strings the output is shorter than the original string. If one can also decompress the compressed version and obtain the full original string, then the compression is called *lossless*. As the Kolmogorov complexity of a binary string is not computable, a fruitful approach has been to apply Kolmogorov complexity by approximating it with data compressors. Like the shortest program that creates object x, also the ideal compression of x is such that will take advantage of every redundancy in x. Hence, it is natural to approximate the incomputable Kolmogorov complexity using compressors. It can be said that an ideal compressor will compress every string x to $K(x)$. In the real word, ideal compression does not exist [38]. However, as every compression algorithm defines a computable function from binary strings to the lengths of the compressed versions of those strings, we can use it to approximate the incomputable Kolmogorov complexity: let C be a lossless compressor, $x \in \{0,1\}^*$, and denote by $C(x)$ the length in bits of the C-compressed version of x. Then $C(x)$ is the length of prefix program that computes x, therefore is an upper bound on $K(x)$, up to an additive constant term, depending on C but not on x [14].

2.4.2 Normalized Compression Distance

As Kolmogorov complexity is incomputable, so is the NID. However, when approximating Kolmogorov complexity by lossless compressors, we arrive to the *Normalized Compression Distance (NCD)*, a computable approximation of the NID (3) as follows [12]: given a lossless compressor C, the denominator can be trivially approximated by $\max\{C(x), C(y)\}$. The numerator is more tricky: $K(x,y)$ can be approximated by $\min\{C(xy), C(yx)\}$, where xy is a simple concatenation of x and y. However, most popular compressors show only small deviations from symmetry, therefore $C(xy)$ will do for this purpose. This leads us to defining the NCD by

$$NCD(x,y) = \frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}}. \tag{4}$$

The significant innovation in the NCD is that it enables one to get for every two computer files a number in range $[0, 1+\varepsilon]$ expressing how similar these files are, without indicating specific feature(s) with which the similarity will be measured. Allegedly, this might make NCD based classification methods 'feature free' methods. We will analyze this 'feature free-ness' concept in the discussion.

Remark 3 *As every computer file is basically a finite binary string, one can compute the NCD for every two computer files, or binary strings.*

Remark 4 *While better compression of a binary string (that is, with shorter length of the compressed version) will always approximate its Kolmogorov complexity better, this is not necessarily the case for the NCD: due to the subtraction and division, it is theoretically possible that while all the elements get compressed better (but not equally), the NCD moves away from the NID. However, in our experiments, as in those of [10] and [12], this behavior was not observed in a noticeable way.*

Remark 5 *As the Kolmogorov complexity is not computable, it is in principal impossible to know how far the NCD is from the NID.*

Remark 6 *In practice, most of NCD values got in our experiments are in range* $[0.85-1]$ *for different objects and* $[0.2-0.3]$ *for identical objects. In a few cases we got* $NCD > 1$*, usually by a very small term.*

2.5 Data compressors used in this work

2.5.1 PPM [5]

PPM stands for Prediction by Partial Matching. It belongs to the family of statistical compressors- compressors that use the occurrence probabilities of the symbols, to achieve good compression. The main idea in statistical encoding (like Huffman codes, for example) is to encode each character in relation to its context and frequency in the text [38]. As a character is more probable to occur, it gets a shorter encoding. PPM usually achieves the best performance of any real compressor yet is also usually the slowest and most memory intensive [10].

The coding scheme of PPM uses a Markov model which conditions the probability that a particular symbol will occur on the sequence of characters which immediately precede the symbol. The order of the Markov models is the number of characters in the context used for prediction. For example [5]: suppose the current symbol sequence is: *"...suppose#the#current#symbol#sequence#i"* (here # stands for a space). If the order is 2, the next character φ is predicted on the basis of occurrences of trigrams "# i φ" earlier in the message. φ="e", "j", "h", "i", for instance, are unlikely, and therefore will have small probabilities, hence will be encoded here by relatively large number of bits. φ="s", will have reasonable high probability in this context, therefore will be encoded shortly.

PPM uses adaptive encoding, namely the coding scheme use a dynamic order: both encoder and decoder recognize predictions on the basis of the longest string match between the present context and previously seen ones. For example, if φ="s" occurs in the context "# i φ" for the first time, the prediction will be based on the 1-order "i φ", thus if the string "is" has occurred previously in the text (even without a preceding space- as in "history" for example) the coding of the character "s" will be based on this foreshortened context. If the string had not occurred previously (that might occur frequently especially in the beginning of the text), the context will be further shortened to the empty string. In this case, the character will be predicted on the basis of its frequency so far in the text. The encoder uses "escape mechanisms" to inform the decoder when the order is changed.

2.5.2 bzip2 [9]

bzip2 is a relatively new compressor using the blocksort algorithm. It provides good compression and an expanded window of 900 kilobytes allowing for longer-range patterns to be detected, yet the size of the input block must be large (a few kilobytes) to achieve good compression. It is also reasonably fast. The algorithm works by applying a reversible transformation to a block of input text. The transformation does not itself compress the data, but reorders the data in a way that the probability of finding a character close to another instance of the same character is increased substantially. Text of this kind can be compressed very efficiently using fast compression algorithms. The algorithm transforms a string S of n characters by forming the n rotations (cyclic shifts) of S, sorting them lexicographically and extracting the last character of each of the rotations. A string L is formed from these characters, where the i^{th} character of L is the last character of the i^{th}sorted rotation. In addition to L, the algorithm computes the index I of the original string S in the sorted list of rotations. There is an efficient algorithm to compute the original string S given only L and I. The string L is easy for compression by locally adaptive compression algorithms.

Example 3 *[9]: let the original string S be* "abraca"*. Then the sorted list of rotations is:*

aabrac
abraca
acaabr
bracaa
caabra
racaab

therefore the string L is "caraab"*.*

The sorting leads to good compression of L because any localized region in L is likely to contain a large number of few distinct characters. For example, consider the letter *'t'* in the word *"the"* and assume the original string contains many occurrences of *"the"*. After the sorting, all rotations starting with *"he"* will be adjacent in the sorted list. Many of them are likely to end in *'t'*, therefore one region of L will contain a disproportionately large number of *'t'*s. The overall effect is that the probability to find character c in a given point in L is relatively high if c occurs near that point in L. Move-to-front coder, a statistical compressor, needs exactly this property for efficient coding.

Remark 7 *It should be mentioned that although PPM and bzip2 are basically text compressors, they achieve very good compression rates and are commonly used to compress other data types as well. In this work we use them to calculate NCD between images, voice samples and signals.*

2.6 Previous works in Kolmogorov complexity-based learning

Cilibrasi & Vitanyi developed a new method for hierarchical clustering using ternary trees [13]. The method is powerful because it can handle data from many different domains and does not require the number of clusters, nor any other prior knowledge about the particular subject area. The only input for the clustering algorithm is the NCD matrix, namely pairwise distances, of the objects being clustered. It is based on a randomized algorithm aims to find a tree that reflects the pairwise distance matrix well (that is, pairs of objects with relatively small NCD are expected to be closer on the tree, than pairs with higher NCD values). However, the performance of the method deteriorates significantly when dealing with more than 40-50 objects [12]. Using this method they cluster DNA sequences to build phylogeny trees and test some hypotheses in biology, languages, viruses, music pieces (also in [11]), Russian texts, OCR (Optical Character Recognition) images and astronomical X-ray observations [12]. Two more applications of this method include clustering of fetal heart rate tracings [16] and analysis of network traffic and clustering of computer worms and viruses [45]. In [24] Keogh et al. approximate the NID by a compression based computable function, the CDM (Compression-based Dissimilarity Measure), similar to the NCD (however not a metric) to perform clustering experiments of time series and text, anomaly detection in signals and classification of time series. They compare the compression based method to leading methods in data mining and show that the compression based approach is competitive or superior to the state-of-the-art methods in anomaly detection and clustering heterogenous data.

In addition, there are several successful experiments using other approximations of the incomputable NID: along with their NCD experiments, Li et al. [30] perform hierarchical clustering of genomes, when for a string x, they replace $K(x)$ by $N(x)$, number of distinct k-length strings in x (they use $k = O(\log x)$). In [14] Cilibrasi & Vitanyi define the NGD (Normalized Google Distance) in which they use the functions $G(x)$, number of web pages found by Google to contain the word x and $G(x, y)$, the number of web pages that contain both the words x and y, to approximate the NID. The new distance is then used to acquire knowledge from the world wide web by a very interesting series of clustering and classification experiments.

2.7 Classifiers

This section presents the basics of linear classification and lazy learning methods, and the two main classifiers used in this thesis: the weighted k Nearest Neighbor (kNN) and Support Vector Machines (SVM). In addition, other classifiers used in the emotion recognition experiments are shortly presented as well.

2.7.1 Linear trainable classifiers

A classifier is a computation process aim to pick up relationships between input and output [34]. The input for most classifiers in machine learning is feature vectors representing instances from a certain domain and the output is a set of target values, also called classes. The set of initial features according to whom the classification is being performed is defined by human expert. Later on, a new set of features can replace the initial one, either by selecting a subset of features, a process called feature selection, or by extracting new features out of the initial ones (feature extraction).

In two-class classification, we seek to estimate a separating function $f : X \to \{\pm 1\}$ based on input-output training data. We assume that the data were generated independently from some unknown (but fixed) probability distribution $P(x, y)$ [2]. Our goal is to learn a function that will correctly classify unseen examples (x, y), i.e. we want $f(x) = y$ for examples (x, y) that were also generated from $P(x, y)$. Given a training set $\{(x_1, y_1), .., (x_n, y_n)\}$, it is easy to see that for any function f and any test set $\{(x'_1, y'_1), .., (x'_m, y'_m)\}$ that doesn't intersect with the training set, there exists a function f^* such that $f(x_i) = f^*(x_i)$ for all $i = 1, .., n$ yet $f(x'_i) \neq f^*(x'_i)$ for all $i = 1, .., m$, therefore a good separating function cannot be found only by minimizing the empirical training error

$$R_{emp}[f] = \frac{1}{n} \sum_{i=1}^{n} |f(x_i) - y_i| .$$

We will also have to seek for relatively "simple" separating functions. Simple, in this case, is usually interpreted as linear: given a training set $\{(x_1, y_1), ..., (x_n, y_n)\}$, a classifier aims to learn/output a continuous function M mapping d-dimensional input vectors to the one dimensional reals. Such a function can be transformed into learning algorithms for binary classification, as the function f above, which for a test vector x can be defined by

$$f(x) = \left\{ \begin{array}{c} 1, \text{ if } M(x) \geq 0 \\ -1 \text{ if } M(x) < 0 \end{array} \right\} = sign(M(x))$$

The sign of $M(x)$ represents x's location in the d-dimensional space in relation to some hyperplane of the form $\langle w, x \rangle + b$, defined by M.

While trainable classifier systems output functions with a discrete range (the set of classes), as the function f above, some of the most successful ones are built on top of continuous learning algorithms [19] (like the function M above). The continuous learners are a broad and important class of algorithms in machine learning. There are at least two good choices for linear trainable learner components for use with NCD: these are Artificial Neural Network (ANN) and the Support Vector Machine (SVM) [10] that will both be described below. Both of these techniques take as input the set of labeled training data as well as some specialized model parameters that must be set through some means [2], [39]. Usually the specialized parameters are set by a human expert or set through an automatic procedure using cross-validation based parameter scanning. Both learner systems produce a single label out of the set L as a result given any input d-dimensional test vector. Each have many strengths and weaknesses and they each give unique performance profiles. They should both be considered when deciding to do classification using NCD, although all experiments in this thesis are based on support vector machines, due to the fact that in the basic structure of ANN, there are more parameters to be set, as will be explained in subsection 2.7.4, and there is no simple rule to guide how to choose these parameters because there are all sorts of bad behavior possible from wrong choices. Since as far as the author knows, no prior knowledge is documented regarding parameter setting for NCD based experiments, and acquiring such knowledge is not included in the scope of this work, SVMs are much more convenient for our purpose due to fewer parameters to be set, as will be explained right below.

2.7.2 Support Vector Machines

This subsection provides a summary of relevant mathematical theory surrounding Support Vector Machines, based on details in [35], [39].

Suppose we have a training set $\{(x_i, y_i)\}_{i=1}^{n}$ where $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, +1\}$ for $i = 1, .., n$. Consider the set of hyperplanes

$$\langle w, x\rangle + b = 0,\ w \in \mathbb{R}^d,\ b \in \mathbb{R}$$

corresponding for the decision functions

$$f(x) = sign(\langle w, x\rangle + b).$$

An optimal hyperplane will be such with the largest margin of separation between the two classes:

$$\max_{w,b} \min\{\|x - x_i\| : w \in \mathbb{R}^d,\ \ \langle w, x\rangle + b = 0,\ i = 1, .., n\}$$

this leads to solving the optimization problem

$$\begin{array}{ll} \min\limits_{w,b} \frac{1}{2}\|w\|^2 & \\ \text{subject to:} & y_i \cdot (\langle w, x_i\rangle + b) \geq 1,\ i = 1, .., n \end{array}$$

which can be solved by the Lagrangian dual

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i (y_i \cdot (\langle w, x_i\rangle + b) - 1).$$

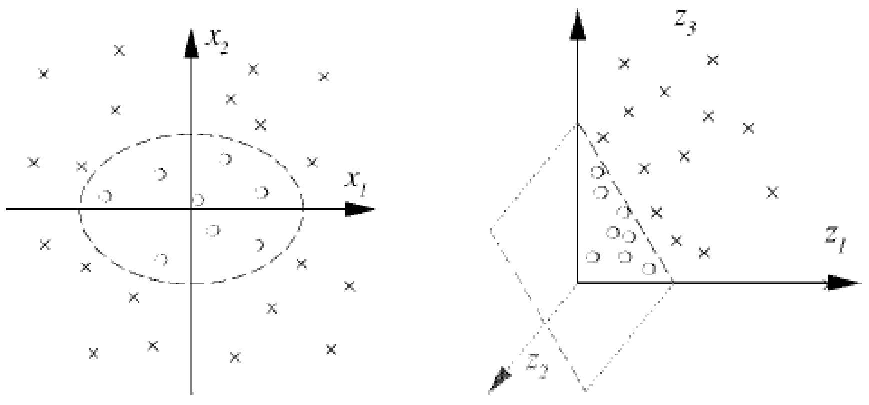


Figure 1: Two dimensional classification example: (right) Using mapping to 3 dimensional feature space, the problem is separable in the feature space by linear hyperplane. (left) In the input space, this hyperplane corresponds to a nonlinear elipsoidal decision boundary (figure from [40])

where α_i are the Lagrange multipliers. After some simplification, we are left with the convex optimization problem

$$\max_{\alpha \in \mathbb{R}^n} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_i \langle x_i, x_i \rangle \tag{5}$$

$$\text{subject to} \quad : \quad \alpha_i \geq 0 \text{ for all } i, \ \sum_{i=1}^{n} \alpha_i y_i = 0$$

that can be solved with several techniques [35]. It happens to be that $\alpha_i > 0$ only for the instances from each class that are closest to the decision boundary. These instances are called the *support vectors*. The solution to (5) defines a separating hyperplane

$$f(x) = sign\left(\sum_{i=1}^{n} \alpha_i y_i \langle x, x_i \rangle + b\right)$$

which is linear in the input space. Since in the real world, linear separation rarely exists, one needs to be able to solve also non linear separable problems, such as the famous *exclusive or* problem [20]. To solve problems of this type one needs to transform the instances by some mapping $\Phi : \mathbb{R}^d \rightarrow F$ to a space of a higher dimension, in which the problem can be solved linearly, as in Figure 1. As F might be very high dimensional, it is impossible to perform computations directly in F. This is where kernel functions are used. A kernel function $k(x, x') = \langle \Phi(x), \Phi(x') \rangle$ is the result of the dot product of the instances x, x' in F. Due to the fact that all instances in (5) occur only within dot products, one can use the kernel function to get the result of the calculations in the feature space F, without actually transforming the instances to F. Using kernel function the decision function gets the more general form

$$f(x) = sign\left(\sum_{i=1}^{n} \alpha_i y_i k(x, x_i) + b\right)$$

that can bend the decision surface around training points in the least stressful way and actually approximate any continuous function up to an arbitrary degree of accuracy given enough training data. In addition, to be capable of dealing with noisy data, SVM uses also slack variables in the optimization problem, allowing instances of a certain class be in the wrong side of the decision boundary, with the cost of a fine for each such instance. There are several common kernel functions used with SVM. According to Mercer's theorem [35], any positive definite symmetric function can be used as a kernel function, where a function $K(x, z)$ is *positive definite* if $\int K(x, z)f(x)f(z)dxdz \geq 0$ for every $f \in L_2$. A kernel function $K(x, z)$ is a measure of the similarity between the instances x, z (for example, the polynomial kernel function is $\gamma(\langle x, z\rangle + r)^d$, with $\gamma, r \in \mathbb{R}$, $d \in \mathbb{N}$, and dot product of vectors is actually a measure of correlation). Among the widely used kernel functions are polynomial kernel, linear kernel (same formula as polynomial, with $d \equiv 1$), and RBF (Radial Basis Function) kernel, which has the form $k(x, z) = \exp\left(-\gamma \|x - z\|^2\right)$.

2.7.3 Lazy learning methods and the k-nearest neighbor algorithm

SVMs belong to a group of classification methods that involve parameter tuning and provide a general explicit description of the target function (that determines the labels of the instances) once training examples are provided. Among other members in this group are Artificial Neural Networks and Decision Trees, that will be described below. A different group of classification methods contains instance based methods ("lazy" methods), in which the training examples are simply stored in the computer, and the generalization (the learning) is postponed until a new instance must be classified. A key advantage of lazy classification is that instead of estimating the target function once for the entire instance space, these methods can estimate it locally and differently for each new instance to be classified [34]. The k-Nearest Neighbor algorithm (kNN), which is used in this work, is a lazy learning method. This 40 year old algorithm has many versions and is widely used and analyzed (for example, [47]). This relatively naive algorithm appears to work well on several datasets, as will be shown in the analysis chapter. The classification rule of the kNN algorithm is rather simple: whenever a new test instance x arrives (which its label needs to be determined), its distance from all instances with known labels is calculated. Then, considering only the set of k nearest neighbors of x, each potential labeling gets a grade according to how close the test instance is to instances from same class. Then x is classified according to the labeling with highest grade. The kNN algorithm can use any distance function for the distance calculation. In this work, we use the NCD for that purpose. In subsection 3.5 we provide the pseudo code of the NCD-based kNN used in this thesis.

2.7.4 Other classifiers used in this work

The following material is a short glimpse at artificial neural networks, based on [2], decision trees, based on [34] and the simple naive nearest centroid classifier.

Artificial Neural Networks (ANN) A neural network is inspired by the human brain. The network consists of single neurons, which are computation units called perceptrons. Each perceptron gets several numerical inputs and calculates a weighted sum of which. The result of the summing function is inputted to an activation function, that computes the neuron's output. The network itself is based on the connections between the neurons, where the neurons are divided to several layers and all the neurons in the i'th layer are connected to all the neurons in the $i + 1$'s layer. Each such connection has a weight, which is updated in every case of misclassification during the training phase. The update of the weights is commonly done by the backpropagation algorithm. The first layer is the input layer, hence always consists of as many neurons as the number of features. The next layers are called hidden layers. It is known that two such layers suffice for good approximation of any smooth function (and for many classification problems one hidden layer will be enough). The last layer is an output layer, and approximates the posterior probabilities, i.e., the probabilities that the instance belongs to each of the classes. This layer classifies the instance to the class with the highest posterior probability. When working with ANN, one has to choose the activation function (common choices are transcendental sigmoid functions like arctangent, there are also several others in use). Another issue that should be considered in all types of ANNs is the architecture of the network, namely the number of hidden layers, and the number of neurons in each hidden layer. In addition, one should define the criterion for stopping the training epochs (commonly used options are fixed number of epochs, or fixed accuracy), where one should avoid the case of over training of the network, and learning rate and momentum parameters, that can also affect the network's gross behavior and overall success or failure. Altogether this implies at least four hard choices before one can train a neural network.

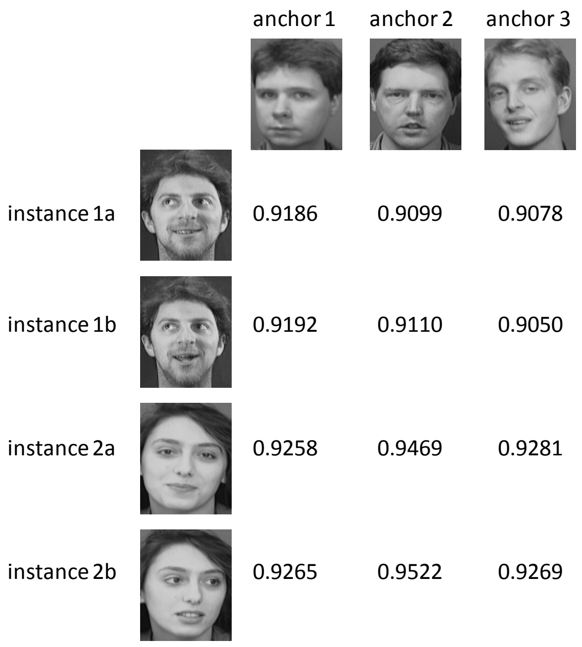
Decision Trees (DT) A decision tree has on each node a feature and a decision rule. The tree is built iteratively, where in each iteration a feature is chosen, as well as a decision rule that divides the scale of the feature to two or more segments. Each segment is then represented by a subtree. A common method to choose appropriate feature in each iteration is by minimization of the entropy of the dataset [34]. As one might understand, it is easy to create for a given training set, a tree with 100% accuracy. Such a tree will probably demonstrate poor generalization ability. Hence, when creating a DT, one will probably not want a tree with maximal accuracy on the training set, but a tree that has similar accuracy on the training and test sets (that implies for the tree's generalization ability). This is usually done by defining the pruning severity and minimal number of instances per branch.

The *inductive bias* (a basic notion in [34]) of a decision tree is our preference for trees of shorter depths, deriving from our belief that such trees will enable us to generalize better than deeper trees.

Nearest centroid classifier This is another version of the classic kNN classifier: for each class we create an average feature vector (centroid) in which each feature value is the average of the values of the same feature among all training instances from the given class. When a new test instance arrives, its Euclidian distance to each centroid is calculated and the instance is classified according to the nearest centroid. To neutralize the bias caused by differences in the scales of different features, it is always recommended to normalize the values of each feature prior to the calculations. Note if one normalizes the features so each of them has zero mean and unit standard deviation, and the features are not correlated, then the algorithm computes the Mahalanobis distance [18].

2.8 The anchor method

Most trainable classifiers require the input of feature vectors of the instances to be classified in order to build the classification model and classify new instances. This was also the case in all SVM experiments in this work. The anchor method, presented in [10] and [14], is a feature extraction method, meaning it gets as an input the instances in their raw representation (image/text/voice files etc...) and for each such instance outputs a feature vector, which will be used by the classifier. The method works as follows: for $x_1, .., x_n$ the instances to be classified, we define anchors $a_1, ..., a_k$ which are objects from the same world. We then create for instance x_i the feature vector $\underline{u_i}$, in which the j^{th} entry is $NCD(x_i, a_j)$. As one can understand, the feature vectors represent the location of instance x_i relatively to the anchors (this idea might be dangerous, because as previously mentioned in Section 2.3.3, every pair of objects might have a different dominant shared feature with which they are most similar, however our results, as well as those of [14] show that it does work quite well). An example for feature vectors created by the anchor method can be seen in Figure 2. As was explained in Section 2.7, the original set of features of the instances to be classified is usually defined



		anchor 1	anchor 2	anchor 3
instance 1a		0.9186	0.9099	0.9078
instance 1b		0.9192	0.9110	0.9050
instance 2a		0.9258	0.9469	0.9281
instance 2b		0.9265	0.9522	0.9269

Figure 2: An example of the anchor method on AT&T faces dataset: feature vectors built for 2 instances from each of 2 classes by 3 anchors. Value in the i^{th} row and j^{th} column is the NCD between the i^{th} instance and the j^{th} anchor. Note that for each feature, instances from same class have similar values, and instances from different classes have greater difference in values.

by a human expert. Based on this set, a new set of features can be created using feature selection and feature extraction methods. One of the basic problems in defining, selecting and extracting features occurs when the features ignore true patterns in the instances, or report spurious patterns that do not really exist. This may cause a severe bias in the classification model built by the classifier. The anchor method is also a feature extraction method, however as described here, the features are extracted implicitly, by the compressor used for computing the NCD values. The features created by the anchor method have only a slight physical meaning- as the location of the anchors in the space is not known, and the dominating feature determining each NCD value is not known either, the feature vector of each instance cannot enable us conclude anything about absolute properties of the instances. However, by leaving these terms in vague, one can reduce the bias caused by a well established feature definition.

For the rest of the paper, let the short term *SVM-anc* refer to SVM using the anchor method. In [14] the authors use the anchor method for 2 class problems only, here we apply it also to multiclass problems, with up to 78 anchors.

Remark 8 *The issue of how many anchors to use, and how to choose the anchors will be addressed in Subsection 3.6.*

Remark 9 *The anchor method can use any distance function, that can be used to quantify distance between computer files, or binary strings. In one of our experiments we will compare the results using the NCD to those when using Hamming distance instead.*

Remark 10 *Feature selection and extraction techniques can be applied to the original feature set created by the anchor method. We will use some of these techniques in our experiments in voice samples and analyze their performance.*

2.9 Input dimensionality reduction

The term "*the curse of dimensionality*" refers to the problem caused by the exponential increase in volume associated with adding extra dimensions to a (mathematical) space [36]. For example, one will need only 100 evenly-spaced sample points to sample a unit interval with no more than 0.01 distance between points, however, an equivalent sampling of a 10-dimensional unit hypercube with a lattice with a spacing of 0.01 between adjacent points would require 10^{20} sample points. In machine learning, one aims to learn a state-of-nature (maybe infinite distribution) from a finite (low) number of instances, hence a severe problem occurs if the instances belong to high dimensional space [34]. In other words, roughly speaking, when performing a classification experiment, the minimal number of instances needed is exponential in the dimension of the input, (except for cases of *sparsity*, i.e., where only few features do vary over the instances). As one might understand, the dimension of the input doesn't have to be too high to make it impossible to collect as many instances as needed to train the learning system, hence one will always be interested in reducing the dimension of the input. This can not explain a greater percentage of the variance of the data, however, reducing the dimension of the input can actually increase the classification accuracy [34]. In addition, from a statistical point of view, it adds degrees of freedom, and in case that new uncorrelated features are extracted, it can improve model's accuracy (such as in linear regression, for example). There is a lot of literature about methods to reduce the input dimension. In our experiments in emotion recognition, the dimensionality of the instances was reduced using several state-of-the-art feature selection and extraction methods which we will be briefly described now (for more information, please see [21]).

2.9.1 Principal Component Analysis (PCA)

PCA is mathematically defined as an orthogonal linear transformation that transforms the data to a new orthogonal coordinate system such that the greatest variance by any projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on. The input for the PCA is the original set of features and the output is an orthogonal set of new features (principal components), which are linear combinations of the original features. Formally, given n k-dimensional instances (the j^{th} instance is $x_{j1}, ..., x_{jk}$) in $X = \begin{pmatrix} x_{11} & ... & x_{1k} \\ : & . & : \\ x_{n1} & ... & x_{nk} \end{pmatrix}$, the output is $Y = XA$, where A is the matrix of the eigenvectors of the correlation (or covariance) matrix of X, with the eigenvector corresponding to the largest eigenvalue in the first column, to second large eigenvalue in the second column and so fourth. PCA is basically an algebraic method that due to the fact that each component explains as much as possible of the remaining variance of the data, enables us to use less variables than in the original set, yet explain most of the variance. In addition, the new set of variables is orthogonal, hence they are uncorrelated in pairs. To neutralize the effect of different scales and units in different variables, it is often recommended to use correlation matrices, rather than covariance matrices (or simply standardize the variables prior to the components extraction). Theoretically, PCA is the optimum transform of the data, in least squares terms. It is a widely used tool in exploratory data analysis.

2.9.2 Discriminant Analysis (DA)

While PCA focuses on the global variance of the data, with no relation to the different classes in the dataset, discriminant analysis methods aim to separate between the different classes involved. Similar to PCA, the Fisher Discriminant Analysis (FDA) outputs a new set of features by a linear transformation of the original set, using eigenvectors of the matrix $S_B \cdot S_W^{-1}$, where S_B is the covariance matrix of the class centroids (average vectors), therefore corresponds to between class variance and S_W is a weighted average of the covariance matrices of each class in the dataset. This corresponds to Fisher criterion which states that a good separation is such that has a high between class variance and low within class variances. The number of new features obtained by FDA is at most the number of classes minus 1. Linear Discriminant Analysis (LDA) assumes normal distributions of the classes density functions and equality of covariance matrices of each class and for a c-class classification problem outputs c discrimination functions $g_1, ..., g_c$, according to which a classification can be made by classifying instance x to class i if $g_i(x) \geq g_j(x)$ for all $j \neq i$, $i, j \in \{1, ..., c\}$. In the special case of 2-class problem, a single discrimination function g can be obtained by $g(x) = g_1(x) - g_2(x)$. An example of the differences between the features extracted by PCA and FDA is shown in Figure 3.

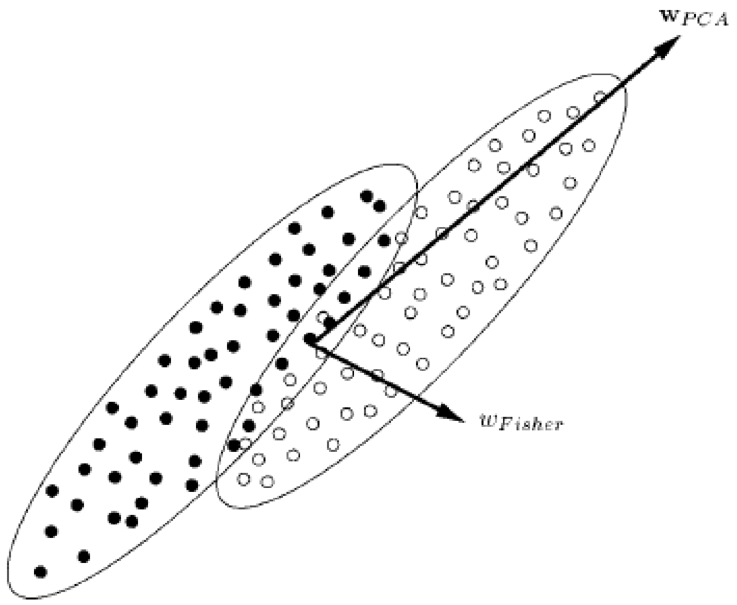


Figure 3: Illustration of the projection of the primary principal component of PCA and this of Fisher's discriminant for a toy 2-class data set. It is clearly seen that PCA is purely descriptive, whereas the Fisher projection is discriminative (figure from [35]).

2.9.3 Factor Analysis (FA)

Factor analysis is a statistical method used to explain variability among observed variables in terms of fewer unobserved variables called factors. The observed variables are modeled as linear combinations of the factors, plus "error" terms. The information gained about the interdependencies can be used later to reduce the set of variables in a dataset. FA has three main phases: determining the number of factors, extraction (PCA is often used for this purpose) and rotation. FA is especially used when it is of interest to understand the underlying factors generating the variables in the experiment. It might enable one to group together different variables that might correspond to same factor, and reorganize one's understanding of one's variables. Though this is not our goal in this work, we use it as a feature extraction method.

3 Methodology

3.1 Databases and main phases

The NCD-based kNN and SVM-anc were applied to 7 different datasets and several types of classification problems:

1. Face recognition: the input here is images of faces of several people, each image has a label corresponding to the person in it. Based on a labeled training set, the classifier is supposed to label images in the unlabeled test set correctly. The face recognition experiments involve two separate datasets: the popular AT&T ORL faces dataset and Yale faces dataset, in which images also vary by illumination (links for both datasets are given in the analysis section). The experiments on both datasets are performed using a NCD-based weighted k-nearest neighbor algorithm. Performance analysis includes accuracy using different training set sizes, comparison of results on the AT&T ORL dataset to those of several state-of-the-art techniques taken from [23] and [31], and a slight modification of the kNN algorithm to improve classification accuracy.

2. Object and texture classification: Classification experiments of images of home objects and textures were performed. On the home objects dataset, the capability of the NCD-based algorithms to deal with rotation of the objects was tested. On the texture dataset, the results were compared to results when using polynomial kernel function for the SVM, rather than RBF kernel. Results on both datasets were compared to those of other state-of-the-art methods from the literature.

3. Optical character recognition: The robustness of NCD-based methods to deal with OCR tasks was tested and compared to popular technique in OCR. In addition, results when replacing the NCD with Hamming distance, for quantifying dissimilarity between the images, were analyzed.

4. Olive image classification: Rather than the above experiments, this one concerns 'real world' data. The dataset contains images of olives from several species and different quality degrees. The images were classified according to the species and to the quality degree. The results were then compared to results using explicit feature extraction and decision trees, taken from [26].

5. Experiments with voice samples: The dataset for this series of experiments contains $2-3$ second voice samples of several speakers. A speaker identification experiment was first performed, with an analysis of classification accuracy when increasing the number of speakers. In a second experiment, emotion of the speaker was captured, regardless of

its identity, through voice samples. In this experiment, the performance of the SVM was compared to 4 other classifiers. In addition, several methods for reducing the dimension of the input were applied to improve classification accuracy. The emotion recognition experiment is an example for a domain on which NCD-based learning algorithms can outperform other classification techniques, due to implicit feature extraction.

6. Detection of pain using time series of ECG signals: This is another example for a domain on which regular feature extraction techniques fail to create the correct features that affect the target value. Time series of ECG signals of two patients were classified according to one of 3 pain levels reported by the patients when recording the signals.

In general, there are two main types of classification experiments in this work. The first consists of our image classification and speaker identification experiments, both are widely researched areas. In these experiments we measure the performance of the NCD-based classification methods, test their robustness to some challenges in images classification (differences in illumination, rotation, real world data, etc.) and compare their performance to the performance of several state-of-the-art methods, specializing in the specific domains on which the classification is conducted. The second type of experiments regards domains in which traditional methods face great challenges due to difficulties in definition, extraction and selection of features that might effect the labeling of the instances being classified. This part contains our experiments in emotion recognition and pain detection. On these domains we think that NCD-based methods might have an advantage over traditional methods, due to the fact that they use global similarities and not specific attributes of the instances. The goal of this part is to test the capability of NCD-based methods to face these classification challenges. In the emotion recognition experimental section, we provide a thorough analysis, including analysis of error types, integration of the anchor method with dimensionality reduction techniques, testing the model's generalization capability and expansion of number of speakers and emotions. The ECG experiment was conducted on a very small number of patients, therefore can not be statistically valid, however on this domain we aim to check feasibility of NCD based methods to detect pain in patients by automatic means, something which does not exist, up today, in the medical community, as shall be explained in subsection 4.3.

3.2 Preliminary manipulation of the data

An important element in preparing the raw data for NCD-based experiments is de-noising [12], [16], [24]. The main idea is that in order for the compressor to capture similarities better (by better compression) it should ignore delicate differences between the two files compressed each time. For example, when playing a perfect sinus wave from a computer speaker into a microphone and digitizing the result to a computer file. Assume the file contains 5000 samples of 8 bit each. When inputting the file into a lossless compressor, it will probably compress well. However, if the file contains only 50 samples instead of 5000, it is unlikely to find a significant compression, because of small amount of noise due to imperfections in the microphone pickup, for instance. This can be repaired if we reduce the size of each sample to, say, 4 bit. By doing it, one might have a good chance to see significant compression again, due to better substring matching. From same reasons, for two strings x, y that are relatively similar, namely such that $K(x, y) \ll K(x) + K(y)$ if we don't use denoising, we might get $C(x, y) \approx C(x) + C(y)$. However, after denoising, the similarity might be captured by the compressor C better, and we will get $C(x, y) < C(x) + C(y)$, as we would expect. Though this is a heuristic, we found that denoising does improve the classification accuracy significantly. In our image experiments, the denoising was performed by resizing the images to a smaller size than their original one (exact sizes in each experiment will be detailed in the analysis chapter), and by reducing the pixel size to one byte, therefore limiting the number of colors to 256. In the ECG experiment, all header data was first removed from the files, to be left only with the signal itself, then the data was rescaled to one byte per sample (therefore each sample has one of 256 possible values). In our experiments with voice samples, the samples are relatively clean due to a very good recording, and we did not perform any preliminary denoising.

In addition, because ideal compressors (that is, compressors that take advantage of all existing redundancies in the input strings, therefore compress a string x to $K(x)$) do not exist, we should find a representation of the input files for which it will be easier for the compressor to find similarities between every two files. We use this idea in our experiments in image classification, where we first turn the images to bmp (bitmap) files, that might have larger size than other representations, however give the most natural description of the images. One could fairly argue that this choice of image representation essentially equals to choosing the better features, which goes against the claimed advantage of the NCD being a 'feature-free' method, however such choice is still a weaker kind of feature selection compared to what one would have to do when using standard parametric methods such as ANN or SVM where one has to explicitly choose every dimension and specify what the attribute is.

3.3 Compressor selection

When working with general purpose compressors, such as the PPM family, Lempel-Ziv compressors like gzip and bzip2, the choice of compressor is mostly an outcome of the size of the files to be compressed. In all experiments with images and ECG, all files involved had sizes up to 40kb, and the compressor used was the PPM. The relatively large window of the bzip2

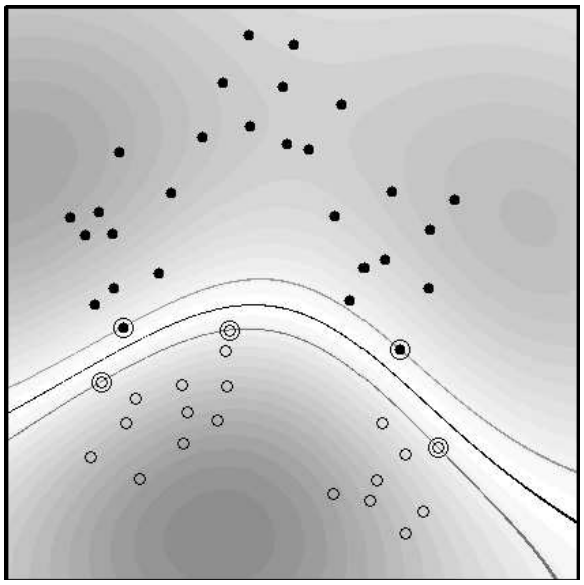


Figure 4: Example of a decision boundary found by RBF kernel. Both coordinate axes range from -1 to +1. Circles and disks are two classes of training examples; the middle line is the decision surface; note that the support vectors found by the algorithm (marked by extra circles) are not centers of clusters, but examples which are critical for the given classification task (figure from [39])

makes it suitable to larger files, as in our experiments in voice samples, in which most of the files had sizes above 100kb, where we indeed used bzip2.

3.4 Kernel function selection

Recall from subsection 2.7.2 that kernel functions express similarity. As $NCD(\cdot,\cdot)$ is a measure of dissimilarity, $1 - NCD(\cdot,\cdot)$ and $\frac{1}{NCD(\cdot,\cdot)}$ both express similarity. Further, As $NCD(\cdot,\cdot)$ is positive and bounded, $1-NCD(\cdot,\cdot)$ and $\frac{1}{NCD(\cdot,\cdot)}$ are also positive and bounded hence positive definite. In addition, they are also very close to be symmetric, hence might be used as kernel functions for SVM. This is theoretically true, however in our early experiments (see example in subsection 4.1.4), classification results using the $1-NCD(\cdot,\cdot)$ as a kernel function are mostly far behind results using other kernel functions (especially polynomial and RBF kernels), and so is the case using $\frac{1}{NCD(\cdot,\cdot)}$. Based on [10], as well as on our empirical data, the best one to be used with NCD is the RBF kernel, defined by

$$k(x,y) = \exp\left(-\gamma \left\|x-y\right\|^2\right).$$

An example to a decision boundary found using RBF kernel is shown in Figure 4. The parameter γ represents the kernel width and determines the rate of exponential decay around each of the training points. The second parameter that the SVM uses is c, the cost of misclassification, namely training points that fall on the wrong side of the decision boundary. Adjusting c can compensate for noisy or mislabeled training data. Both c and γ needs to be fixed before the classification starts. As there are only two parameters to be set, and they are both continuous, a common way to do it, is to tune them using a cross validation process [40], either by using simple two dimensional grid search or more sophisticated search methods like gradient descent.

3.5 Weighted k Nearest Neighbor algorithm (kNN)

We use The weighted k-nearest neighbor algorithm as follows:

Algorithm 2 *weighted k-nearest neighbor*

```
  input: training set $R$, test set $S$, # of distinct classes $m$, kNN parameter $k$
  Step 1: for each $x \in S$:
      Step 1.1: compute $NCD(x, y)$ for all $y \in R$.
      Step 1.2: find $y_1, .., y_k$ the nearest $k$ neighbors of $x$
      Step 1.3: initialize: $a_i = 0$, $i = 1, ..m$
      Step 1.4: for $j = 1$ to $k$
          Step 1.4.1: $a_l = a_l + \frac{1}{NCD(x,y_j)^2}$ for $l =label(y_j)$
      Step 1.5: return: the class of $x$ is $c = \arg\max\{a_i\}$.
```

An example of applying the kNN algorithm to a face recognition task is shown in Figure 5. Note that as explained in section 2.4, the NCD is only an approximation of the incomputable NID, and it is impossible to conclude how far is the NCD from the NID. Absolute NCD values are in general high in range $[0, 1+\varepsilon]$ (in this example all values are greater than 0.84), however this algorithm relies on relation between different NCD values, rather than on single absolute values, making it a more than reasonable NCD-based classification algorithm.

Remark 11 *Very roughly speaking, based on our image classification empirical results, we can note that the weighted kNN has a reasonable accuracy as long as the differences between different classes are simple enough to be easily captured by human eye, as in Figure 5.*

3.6 Anchor selection

The anchors may be chosen by a human expert or be chosen randomly from a large pool of training objects [10], belonging to the same dataset as the instances to be classified. Intuitively, it may seem very important which objects are chosen as anchors. While this is

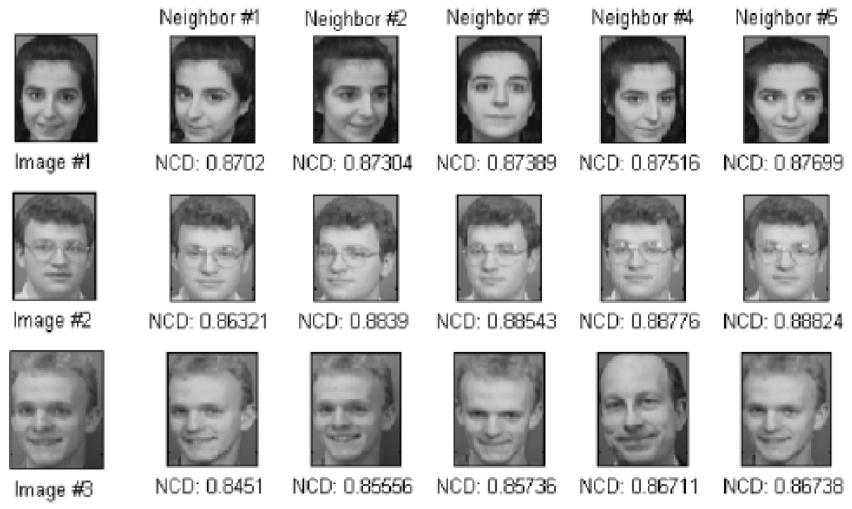


Figure 5: Illustration of the kNN algorithm on the AT&T ORL dataset: 5 nearest neighbors for each of 3 test instances, and NCD values between each test instance and its neighbors

sometimes the case, it seems like more often it is not. Our empirical results correspond to those of [10], showing that random anchors usually work well, as long as one picks at least one anchor from each significant category under consideration. Doing so is advisable to avoid falling into low-accuracy performance due to insufficient variation in the anchor set. Non formally, this way to choose anchors makes the set of anchors a spanning set of the input space, in a sense. Still, the number of anchors to use seem to depend on the specific dataset studied. Addition of anchors always provides more information about the instances, however, it also increases their dimension. Although It is hard to evaluate exact numbers of anchors needed for each classification task, one could generally argue that the more difficult the classification task is, the more anchors will be needed. Since a too high dimension might cause an overfitting problem (due to the curse of dimensionality), it seems natural to assume that there is an optimal number of anchors for every dataset. This number can be found using a cross validation process, where one simply looks for the number with which he achieves best accuracy on the training set. In this work, however, we do not look for an optimal number of anchors and use a fixed number of anchors, determined arbitrarily for each experiment with SVM-anc.

Remark 12 *One can use a relatively large number of anchors, and apply a dimensionality reduction such as PCA or DA, in order to control the dimension of the instances. In this work, we use this approach in our experiment in emotion recognition, where 78 anchors are used.*

Remark 13 *Though the anchors can be chosen randomly, it is important to exclude them* *from the training and test sets, to avoid misleading of the classifier ($NCD(x,x)$ will always be much smaller than $NCD(x,y)$ for every x,y in a real world (that is, not synthetic) data. Hence, a feature vector of an instance x that is also used as the j^{th} anchor will probably be significantly different from feature vectors of instances y belonging to the same class, in the j^{th} entry.*

3.7 Cross validation

The SVM parameters (c and γ) do not usually need to be exact; instead one can simply do a stepwise search along a grid of points in log space; thus it is fine to try, for instance, just 64 points for c ranging from 2^{-31} to 2^{32} doubling each time. A similar procedure works also to choose γ. This way we can define a search space S for each of the above parameters, and get the search space $S_\gamma \times S_c$. One may use m-fold cross-validation to estimate the accuracy of the given parameter setting and then just choose the maximum at the end. m-fold cross validation is a popular procedure that tries to estimate how well a model (of any type) that is trained on a set of training data will perform on unknown testing data. The basic assumption is simply that the training data and the testing data are similar. In order to estimate the testing accuracy using only the training data, we first need to divide the training set to m distinct folds, numbered 1 through m. Next, for each combination of $(\gamma, c) \in S_\gamma \times S_c$ we perform m iterations, in each iteration we train the classifier on $m-1$ folds and test it on the remaining one. We then calculate the average accuracy of these m iterations and assign it to (γ, c). After evaluating all optional pairs of values in $S_\gamma \times S_c$ we choose the pair with the highest average accuracy. This way we can find a good setting of (γ, c) using just the training data. In this work we use contour maps to get graphical illustration of the search space, along with exact accuracy calculations, as shown in Figure 6.

Finding the optimal value of k for the kNN classifier is an easier task, as the search space is much smaller and usually contains up to 10-15 options.

3.8 Experiments structure and statistics

Each of the SVM classification experiments was conducted in the following way:

1. Random division of the dataset into training and test sets, according to a given split percentage.
2. A 5-fold cross validation process was performed on the training set, outputting a chosen pair (γ, c) for the SVM.

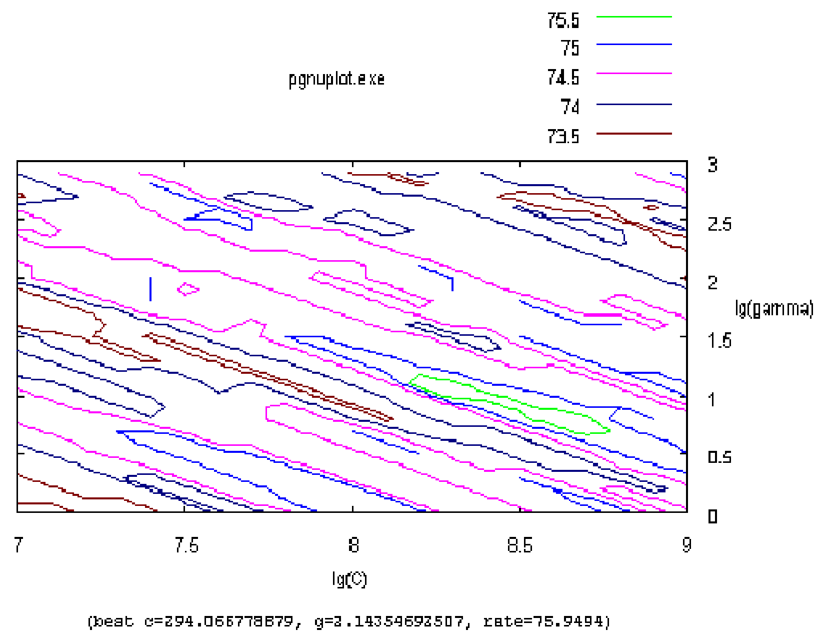


Figure 6: Contour map of the cross validation process for the SVM. Different colors correspond to different accuracy levels. The horizontal and vertical axes correspond to c and γ, respectively, in log scale.

3. SVM model creation (training) using the training set and the pair (γ, c) chosen in the cross validation process.

4. Label prediction for each of the instances in the test set, based on the SVM model created in phase 3. Then, accuracy on the entire test set was calculated by

$$\frac{\#\text{ of correct classifications}}{\#\text{ of test instances}} \cdot 100$$

When applying dimensionality reduction methods (PCA and discriminant analysis) to create new features out of those created by the anchor method, the principal components (or linear discriminants) where computed using only the training instances, and based on them the new features were computed both for the training and test instances, between phases 1 and 2. In case of the kNN algorithm, we used the following simpler procedure:

1. As for the SVM

2. Label prediction for each test instance based on Algorithm 2. The labels were predicted for several values of k. Accuracy was calculated as for the SVM.

The above phases ($1-4$ for the SVM, $1-2$ for the kNN) were repeated 5 times in order to compute average accuracy, standard deviation and worst performance. In addition, in the emotion recognition experiment, the confusion matrix, that counts the number of misclassification errors of each type (for example, for the 2 problem experiments the error types are false positive and true negative) is presented and analyzed.

4 Analysis

This section comprises the experimental part of the thesis: technical details of the datasets, as well as description and results of all experiments.

4.1 Image classification

4.1.1 Face recognition

Face recognition is a popular classification task in the field of machine learning (see [44], [49] for example). The input for face recognition experiments is a set of faces images, belonging to several people, and the task is to classify each image according to the person it belongs to. In this thesis, the face recognition experiments were conducted on two separate datasets, as will be explained below.

AT&T ORL faces dataset This popular dataset (*http://www.uk.research.att.com/ face-database.html*) contains 10 face image files of each of 40 different people (men and women). Each subject is upright in front of a dark homogeneous background. The images for some subjects vary by pose, illumination, facial expression, and whether they are wearing glasses. This dataset is relatively noiseless and intuitively has low variance within each class and high variance between the different classes, as can be seen in Figure 7. Each image is a 92×112 pgm file. Based on preliminary analysis, detailed in Appendix 7.2.1, a minor preliminary manipulation on the dataset here was to turn the image to a bmp file, of same size and 256 colors. The algorithm used for the classification is the NCD-based kNN. Table 2 summarizes the performance of the kNN on the ORL dataset, as well as a sensitivity analysis, in which the effect of different number of training images of each of the 40 persons in the dataset was analyzed. (detailed results are in Appendix 7.2.2). We can see that k=1 appears to be the

Table 2: Average performance on the ORL dataset and effect of different training set sizes

Training images	Best k	kNN average performance (%)	Worst performance
1	1	57.77	56.3
3	1	82.5	80
5	1	92.5	90
7	1	95.8	93.3
9	1	98.75	97.5

best among all values of k in all experiments. In addition, as one might expect, the average accuracy grows with the number of training images. Five training images per person suffice to obtain accuracy above 90% on this dataset. As a reference, Table 3 shows the NCD-based

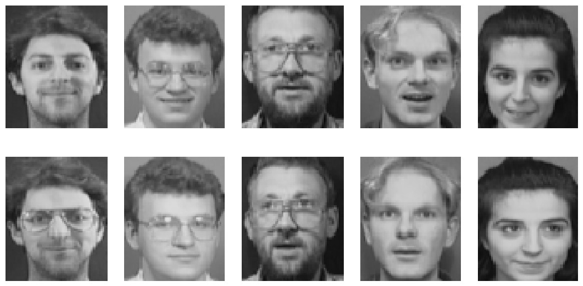


Figure 7: AT&T ORL dataset: images in each column are instances of same class, different columns correspond to separate classes.

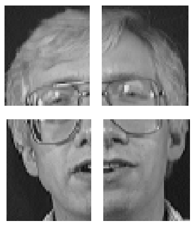


Figure 8: Dividing each picture to 4 independent parts and classifying each part separately

kNN results, comparing to results of several methods from [31] and [23] on the same dataset, all using 5 training images per person. Though not best, it can be seen that the NCD-based

Table 3: Performance on ORL dataset

Method	Average performance	Worst performance
PCA [31]	94.5	89.0
LDA [23]	$91 - 95$	–
orthogonal centroid [23]	$89.5 - 93.5$	–
NCD-based kNN	92.5	90

kNN's results on the ORL dataset are comparative to those of [31] and [23].

Improving the results To improve the results, the weighted kNN algorithm was modified as follows: 4 new datasets were created, where each dataset contains only one part of each image (upper left, upper right, lower left, lower right), as can be seen in Figure 8, The experiment was conducted using $k = 1$ on each dataset separately, using same random division for training and test sets in all datasets (with 7 training images per person). We then gave each answer the weight of $(\frac{1}{NCD \text{ to nearest neighbor}})^2$ and for each testing image, we combined the 4 answers according to these weights to get a single answer (answer with highest grade). This modification of the kNN algorithm improved the results to average accuracy of 99%,

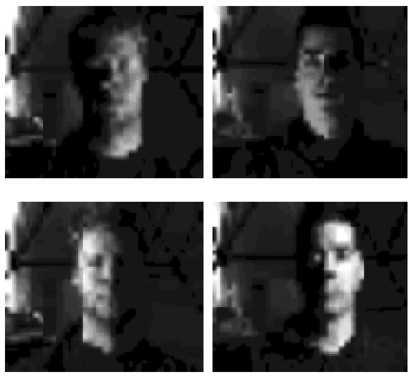


Figure 9: Yale dataset: images in each column are instances of same class, different columns correspond to separate classes. Note that in each class, in one of the images the person is illuminated from left and in the other one from right.

standard deviation of 0.93 and worst performance of 98.3, comparing to average accuracy of 95.8%, without the modification (as in Table 2). The results of each of 5 runs are detailed in Appendix 7.2.3.

Yale faces dataset: different forms of illumination This dataset (*http://cvc.yale.edu*) contains 585 30×40 pixel images of each of 10 people, with different forms (levels and directions) of illumination and different facial expression, as shown in Figure 9.

Classification experiments were conducted on a subset of the large dataset, consisting of the first 30 images of each person. The forms of illumination varied within each class but were identical among all classes, making the within class variance greater and the between class variance smaller, hence the classification task more difficult. The images were first transformed to size of 150×112 and 8 bit per pixel. 75% of the images (meaning 23 images of each person) were used for training. Table 4 summarizes the results of the kNN algorithm on the Yale dataset (detailed results can be found in Appendix 7.2.4). As can be seen, the

Table 4: Performance on Yale dataset.

k	1	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Average	98.54	98.54	98.84	99.12	99.1	98.82	98.82	98.82	99.12	99.42	99.42	99.12	99.12	98.82	98.82
SD	1.75	1.75	1.87	1.30	0.82	1.21	1.21	1.21	1.30	1.29	1.29	1.30	1.30	1.21	1.21
Worst	95.7	95.7	95.7	97.1	98.5	97.1	97.1	97.1	97.1	97.1	97.1	97.1	97.1	97.1	97.1

best results were obtained with $k = 6, 11$, and 12. The results indicate that the NCD-based

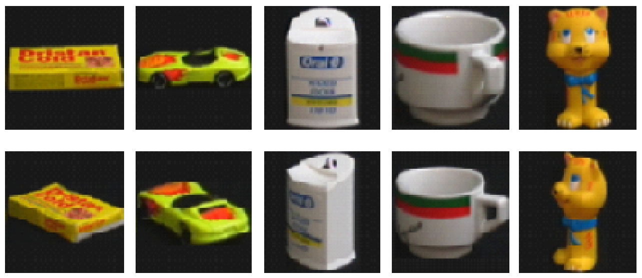


Figure 10: Coil100 dataset: images in each column are instances of same class, different columns correspond to separate classes.

kNN algorithm can be robust to differences in illumination, despite the noisy data. Both the average classification accuracy and the standard deviation on the Yale dataset are better than on the simpler, in terms of within class and between class variances, ORL dataset, however in cost of smaller number of classes (10 classes in the Yale dataset comparing to 40 in the ORL) and more training images (23 here and less than 10 in the ORL experiment). The main result though, is the NCD's capability of performing highly accurate classification in noisy conditions as demonstrated here.

4.1.2 Home objects classification: rotated images

The next dataset is a subset of the very large coil100 (Columbia Object Image Library, *http://www.cs.columbia.edu/CAVE)* dataset, of 50 classes, where each class consists of 10 3-D images of a certain object. The goal here is to test the NCD-based kNN's robustness to rotation: in each class, each image differs in pose of 10 degrees from the next image. An example of the images in the dataset is shown in Figure 10. A preliminary manipulation of the images included turning the images to 90×90 bmp files. The kNN algorithm was used, with 5 random images of each object used for training and 5 more for testing. Table 5 summarizes the performance of the kNN on the coil100 dataset, for 8 and 24 bit per pixel images (detailed results are in Appendix 7.2.5).

Table 5: Performance on coil100 dataset

Bit per pixel \\k	Best k	Average	Sd	Worst
8	1	97.2	0.7	96
24	1	99.2	0.4	98.8

As one might notice, the results with 24 bit per pixel images are better than with 8 bit per pixel images. In both cases, however, the kNN achieves relatively high accuracy, and small standard deviation. Table 6 contains a reference of some classification results from [31] on the

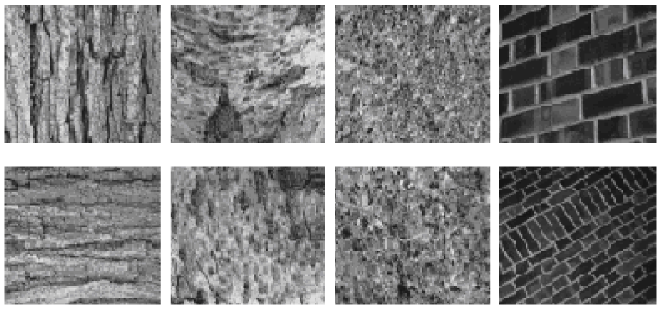


Figure 11: Texture dataset: images in each column are instances of same class, different columns correspond to separate classes.

same dataset. As can be seen, the number of training instances that we used is less than the

Table 6: [9]'s results on coil100 dataset

Method	8 training images	18 training images	10 degrees rotation
PCA	87.2	96.7	83.2
ICA	87.9	95.5	82.6
SHFDA	94.4	98.9	95.9

number used in [31] although our results are better than those of [31].

To summarize the coil100 experiment, our method turns out to be robust to little rotations in the point of view (that is, images taken from different angles), a result that is sometimes challenging for more complicated learning methods.

4.1.3 Texture Classification

The Ponce research group texture dataset (*http://www-cvr.ai.uiuc.edu/ponce*) consists of 40 images of each of 25 classes, in jpeg format. This dataset is considered to be quite difficult for classification experiments, due to its relatively high within class variance, as one can visualy note in Figure 11. The experiment was conducted on a subset of 15 classes, chosen randomly, with 75% of the images used for training. The images were first transformed to 128×96 bmp files with 8 bit per pixel. Both NCD-based classifiers were applied to this dataset. In addition, to analyze the sensitivity of the SVM-anc to its kernel function, the SVM was ran also with polynomial kernel function instead of RBF kernel. The two kernel functions seem to achieve similar accuracy, however as the RBF kernel uses a single parameter, and the polynomial kernel uses 3 parameters, we find RBF kernel more convenient for use. As a reference to our results, we attach results from [27] on the same dataset, using state-of-the-art sparse texture representation method. Results are summarized in Table 7 (detailed results, as well as SVM

Table 7: Texture classification performance

	kNN	SVM-anc, RBF kernel	SVM-anc, poly. kernel	[27]'s results
Avg performance	73.2	84.13	83.99	79.2 – 92.6
Sd	4.66	4.69	4.98	–
Worst	68.1	78	77.33	–

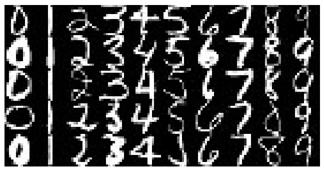


Figure 12: The USPS dataset (figure from [48])

parameters can be found in Appendix 7.2.6). Among the two classifiers, the SVM-anc performs significantly better than the kNN on this dataset, with higher accuracy and similar standard deviation. The two kernel functions achieve very similar results, with no more than a slim difference, in favour of the RBF kernel. [27]'s methods outperform ours, with p-value of 0.073, however one should keep in mind that the methods in [27] specialize in texture representation (and will probably not suit other types of data), however the NCD-based methods turn out to perform reasonably well also in texture classification, with no change of the classification algorithms.

4.1.4 Failure: Optical Character Recognition (OCR)

The applicability of the NCD-based methods to OCR was evaluated using the USPS dataset. This dataset (*http://cervisia.org/machine_learning_data.php*) contains 9298 handwritten digits (7291 for training, 2007 for testing), collected from mail envelopes in Buffalo [28]. Each digit is a 16×16 image. It is known that the USPS test set is rather difficult: the human error rate is 2.5% [8]. A collection of random test samples is shown in Figure 12. As the original size of the images is too small for compression, we had to enlarge the images, to allow good compression, thus we resized each image to 50×50 pixels. The experiment was conducted on a subset of this large dataset, containing 200 training and 200 testing instances. Both methods fail to give reasonable accuracy on this dataset, with average performance of 40.0% for the kNN , 56.4% for the kNN after the division modification (see subsection 4.1.1) and 62.4% for the SVM-anc (with 20 randomly chosen anchors). As a reference, we used a common feature extraction method for small images, in which each feature corresponds to a pixel in the image and the feature's value is simply the value of the pixel. When creating such feature vectors, the SVM (with polynomial kernel function) achieved average accuracy of 84.3% on the subset used for our experiment (and 94.41% on the entire USPS dataset). To summarize the OCR experience, in spite of the successful clustering experiments of [12] with OCR, the methods employed did not reach a good performance in all our OCR experiments.

Two additional experiments were conducted as follows:

- Using NCD-based kernel: recall that $\frac{1}{NCD(\cdot,\cdot)}$ is a positive definite function (see subsection 3.4), hence can use as a kernel function. In addition, it's a function that express similarity between its two arguments, therefore might naturally be a good kernel. However, when using it as a SVM kernel on a random subset (of 200 training samples and 200 test samples) we only reached classification accuracy of 41%, worse than when using the SVM-anc (that is, with RBF kernel).

- Using the anchor method, however when replacing the NCD by another distance function: as the anchor method can use any distance function (not just NCD), let us see what accuracy will we get if we replace NCD in the Hamming distance, for example.

  Definition 7 *The* normalized Hamming distance *between two equal sized binary strings* $x_1, ..., x_n$*,* $y_1, ..., y_n$ *is* $\frac{1}{n}\sum_{i=1}^{n} |x_i - y_i|$.

  Both the hamming distance and NCD of two identical strings will be small (to the matter of fact, the Hamming distance will be zero, and the NCD will probably be a small positive number, in range $0.2 - 0.4$, as we noticed in our experiments). However, when considering some simple cases, such as two images where one is a shift of the other, the NCD is supposed to capture the similarity relatively easily rather than the Hamming distance, that might be quite high. Therefore, we would expect the results with the NCD to be better than those using the Hamming distance. Surprisingly, while the results of using the NCD were all below 63%, replacing the NCD by the naive Hamming distance improved the average accuracy to 80.7%. This surprising result encourages our early impression that NCD does not perform well in OCR tasks. One optional explanation to the fair OCR results might be that the representation we chose for the digits (converting to bmp files of size 50×50) distorted the similarities found by the compressor. Recall from Section 3 that this representation can be considered as a weak form of feature extraction. These results might be an example to a case where unsuccessful feature extraction can insert large amount of bias into the data make the classifier fail in finding true patterns.

All detailed results on the USPS dataset are given in Appendix 7.2.7.

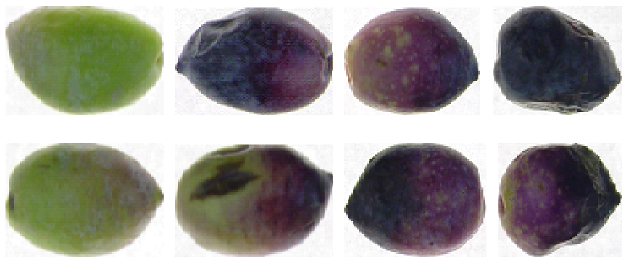


Figure 13: Olives from left to right columns: type 1 quality deg. 1, type 1 quality deg. 3, type 2 quality deg. 3, type 2 quality deg. 4

4.1.5 Real world botanical data: olive classification

The next dataset contained images of olives of various species, taken from [26]. In each species the olives are manually labeled according to their degree of quality, $2-3$ degrees for each species. Though impossible to quantify, it seems that the task of classification in this dataset is much more difficult than in the previous datasets, as the variance within each class is greater, while the variance between the different classes of each type is smaller, as can be seen in Figure 13. As earlier, a preliminary manipulation of the images was conducted, in which the dimensions of the images were reduced from 450×230 pixels to 200×100 pixels, and to 256 colors (each pixel gets a 8 bit value).

Each of the first 5 experiments on this dataset deals with olives of a single species, where each class refers to a certain degree of quality, and the task was to label the olives correctly according to the quality class. Results of the first 5 experiments for both the kNN algorithm and the SVM using the anchor method, as well as reference from classification using decision trees (70%-80% training set in our methods, 50% at most in DT) can be shown in Table 8 (detailed results in Appendix 7.2.8).

Table 8: Olives quality identification performance

Species\Classifier	kNN avg	kNN sd	SVM-anc avg	SVM-anc sd	DT avg
1	93.3	4.24	80.45	1.32	97.8
2	88.8	1.42	92.17	2.13	99.5
3	78.6	11.63	82.60	5.54	95.5
4	93.3	5.16	92.85	4.34	97.5
5	81.0	1.31	80.60	3.60	95.3

It can be seen that in our methods the performance of classification has a high amplitude, from 93.3 in the easiest species (1,4) to 78.6 in the hardest (species 3). In addition, it would be interesting to note that the results of each of the NCD-based classifiers have higher correlation with those of the DT classifier (0.73 for the SVM, 0.66 for the kNN) and lower correlation with the other NCD-based classifier (0.46). The two NCD-based classifiers seem to have similar overall average performance and standard deviations, with the kNN being better on 3 species and the SVM-anc on the other two. In all 5 cases, the DT classifier significantly outperforms both our methods, however, one should be aware of the fact that the input to the DT algorithm is feature vectors of length 30, created by a human expert after close examination of each image [26]. In this case, the feature extraction seems to work quite well, though the extraction requires quite a lot of effort and is time consuming, comparing to the NCD-based 'natural' methods. Here again, this is an example to feature extraction that specializes in this specific input (olives) and classification task (identification of quality degree), in contrast to the NCD-based methods, that almost do not depend on the specific input.

We then built 3 new datasets. In each dataset, each class represents olives of a single species. The first two datasets deal with olives of one degree of quality (1 and 3) with the task of identifying the species of each olive. In the last dataset each class represented, as in the former two datasets, a certain species of olives, but contained olives from all degrees of quality. Results of the kNN algorithm are shown in Table 9 (detailed results in Appendix 7.2.8).

Table 9: Olives type identification performance

Dataset	k	Average	SD
mixed types degree 3	1	91.1	3.11
mixed types degree 1	all	100	0
mixed types all degrees	3	82.8	3.12

As can be seen, the results for the first two datasets are relatively good, with no errors on the second dataset, probably due to the high variance between the classes (differences between each species, that can be easily captured where the olives are of a good quality). The results deteriorate for the third dataset, because of two main reasons: first, the within class variance is now greater, as each class contains olives of several degrees of quality rather than a single one, as in the former two datasets, and second, because of adding degree 4 to the dataset. Olives of degree 4 are of the worse quality. These olives look quite similar, regardless of their type, and this fact inserts a large amount of noise to the dataset, by reducing the between class variance.

4.2 Voices classification

The next series of experiments deals with voice samples. We shall analyze the performance of the NCD-based classification methods in two different tasks: speaker identification and recognition of speaker's emotion. The latter case is an example to a task where feature based methods face great problems, due to difficulties in defining and extracting features that might affect the target values. This part of the work contains a deeper analysis, including comparison between different classifiers and several state-of-the-art methods for dimension reduction. An analysis of the classification errors is also provided.

The parent dataset used for these experiments is taken from [42]. Each instance in the dataset is a relatively clean $2-3$ second voice sample, in which one speaker says one sentence in English. Each instance corresponds to a specific emotion, such as: thinking, enthusiastic, angry, etc. It is important to mention that the samples differ both in text and intonation, however not in volume.

4.2.1 Speaker identification

The first experiment with voice samples deals with the task of speaker identification, that is, to associate each unlabeled test instance with the appropriate speaker. Commonly, speaker identification experiments are done using versions of GMMs (Gaussian Mixture Models) [37]. The kNN and SVM-anc methods were used, without any preliminary manipulation of the voice samples. The speaker identification experiment results (for 4-class and 10-class experiments) are shown in Table 10 (technical details and detailed results are in Appendix 7.2.9). One

Table 10: Speaker identification performance

Difficulty	4 speakers		10 speakers	
Method	SVM-anc	kNN	SVM-anc	kNN
Avg	92.47	80.0	85.2	65.4
Sd	2.67	3.01	2.87	3.51

can notice that the SVM-anc significantly outperforms the kNN in both experiments, with the kNN's performance deteriorating severely in the 10-class experiments. The author is well aware of the fact that GMM classifiers perform well on almost noiseless speaker identification experiments, even for much larger number of speakers, however one should notice that no pre-processing of the input is required here, as well as no modifications of the algorithms, to fit this type of data. Generally speaking, working with NCD has some potential in voice samples as well.

4.2.2 Emotion recognition

Recognition of human emotions through voice samples is considered to be a relatively complicated task and requires a complex analysis of the voice samples [42]. The main reason is that it seems hard to identify features in the voice sample (that consists from several continuous signals) such that might be related to the different emotions. On the other hand, to human ear the task is quite trivial, and generally one can claim that humans manage with it successfully. As one of the machine learning goals is to develop systems that are able to imitate or even outperform human decisions (such as in face recognition, OCR etc.) [34], it seems natural to deal with this task using machine learning algorithms. Approaching it using NCD-based methods might have a great potential, due to the fact that the NCD considers each object as global, rather than as a set of features and extracts features implicitly, (recall subsection 2.4), hence many of the feature identification, definition and extraction difficulties can be 'bypassed'.

The dataset for the emotion recognition experiments consisted of 395 voice samples, belonging to two classes, of positive (i.e. happy, content, festive, delighted, grateful, rewarded, ...) and negative (humiliating, blaming, scolding, correcting, punishing, contradictory,...) emotions. Altogether, the dataset consisted of 187 positive instances and 208 negative instances, of 3 adult female speakers. As no other results are available on the subset we used, and the labeling of the samples ("positive" or "negative") can not be absolute, we first measured the human classification performance, as a reference to our results to come. The human performance was measured in a 7 participant experiment. As we wouldn't like the results to be biased due to the differences in text, rather than in the intonation, 4 of the participants were children below the age of 10, that do not speak English. The human accuracy measured was 81.3% for all participants and 76.6% for the non English speakers.

Fifty three randomly chosen anchors (from the entire parent dataset, including samples of male and children speakers) were used by the anchor method to create the feature vector for each voice sample to be classified. 75% of the instances used for training. We then made a comparison between 4 different classifiers, SVM, ANN, DT and Nearest Centroid (see subsection 2.7.4 for details), of which the results are summarized in Table 11 (detailed results and technical details in Appendix 7.2.9). As can be seen, every two classifiers do not differ

Table 11: Emotion recognition performance

classifier	Avg	Sd	Worst
SVM-anc	73.10	1.67	70.58
ANN-anc	73.69	3.00	70.0
DT-anc	72.31	2.77	68.46
NC-anc	73.10	2.38	69.74

significantly (statistically) in performance. In addition, the NCD-based classification results are surprisingly close to the non-English speakers human performance. In view of the above results, we will continue our experiments using the SVM.

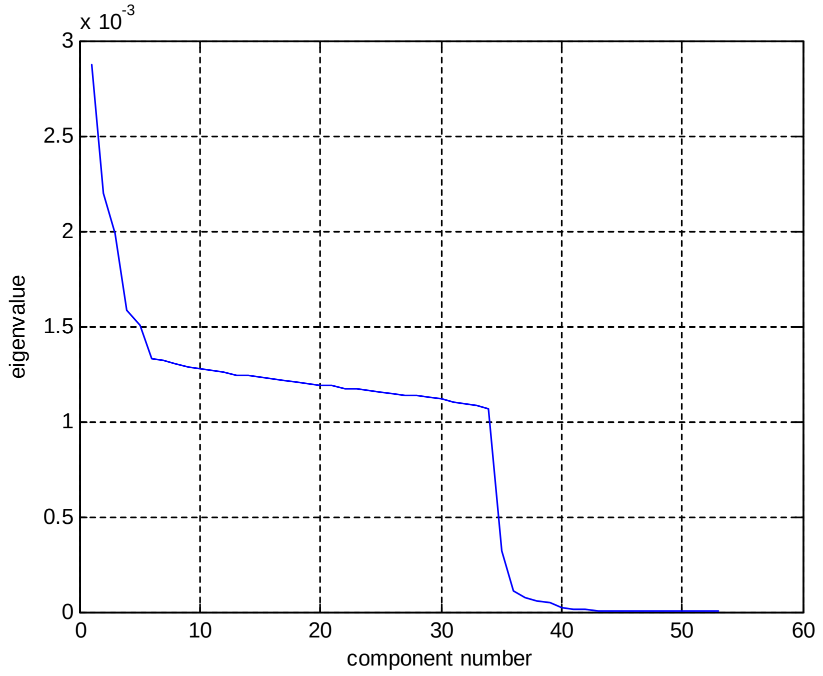


Figure 14: PCA scree plot: eigenvalues of the covariance matrix. A sharp decrease appears after the 34^{Th} component.

Reducing the dimensionality of the voice samples Let us see if we can improve the classification results using dimensionality reduction. The original dimension of the instances here is 53 (as the number of anchors used in the anchor method). We conducted the dimensionality reduction using 5 state-of-art-techniques (separately): Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), Fisher Discriminant Analysis (FDA), Factor Analysis (FA) and t tests. The first four techniques are feature extraction methods, and are briefly described in subsection 2.9. The latter is a feature selection method, in which for each feature we simply perform a t-test to compare its means on the 2 classes, and we withdraw all features for which the mean differences is not significant in 5% level.

- PCA: After extracting the principal components, one has to determine how many principal components should be used for the classification. Figure 14 shows the scree plot of the eigenvalues correspond to the principal components. Table 12 and Figure 15 show the cumulative variance in the data that is explained by the principal components. Based on these results, we decided to use the first 34 principal components, which explain altogether 98.4% of the variance of the data.

- DA: Figure 16 shows the distributions of the 2 classes under the discrimination function found by the LDA. Figure 17 shows scatter plots of the projections of the entire data

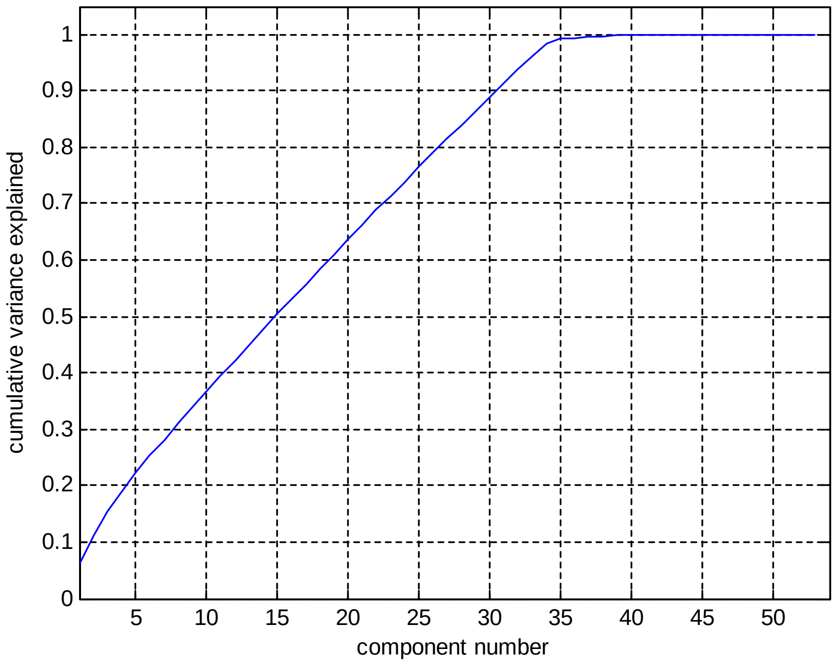


Figure 15: Cumulative variance explained by the principal components

Table 12: Cumulative variance explaind by principal components

Principal components	1	5	10	15	20	25	30
Cumulative variance explained	0.18	0.22	0.36	0.50	0.63	0.76	0.88

35	40	45	50	53
0.991	0.998	9.9997	9.9999	1

(all 395 samples) found by the LDA and FDA. The function computed by the LDA separates correctly 79.2% of the instances. The one computed by the FDA separates 74% of the instances.

- FA: We used PCA for the extraction of the initial factors and varimax rotation with Kaiser normalization. The KMO statistic of sampling adequacy was 0.836, meaning the dataset is factorable, which also was concluded from the relatively high on-diagonal values and low off-diagonal values in the anti image matrix (which was too large to appear in the appendix). Nine factors were extracted and rotated, explaining 48% of the total variance. The factors were then reconstructed using regression.

- *t* tests: 32 features were found to have a significant mean difference over the 2 classes in 5% level. The remaining 23 features were not selected for classification. Figure 18 presents a visualization of the distributions of example of one feature of each kind over

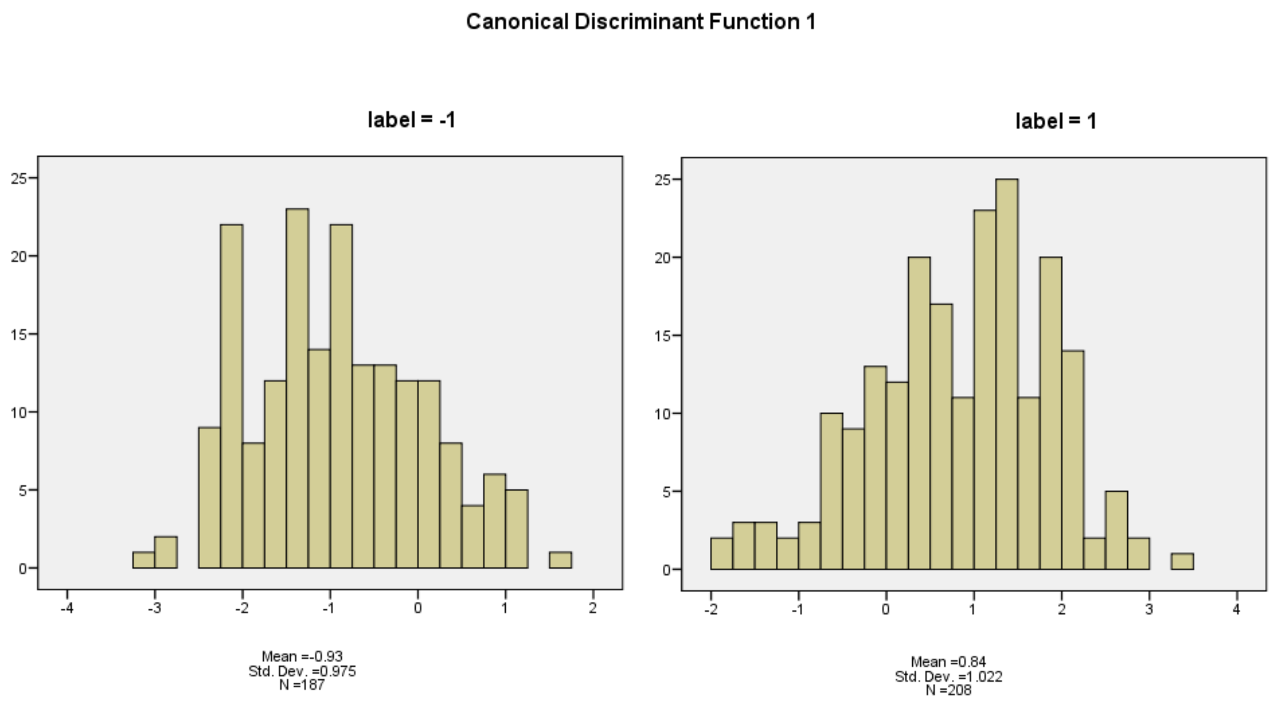


Figure 16: Class distributions under the LDA discriminant function. Label -1 refers to positive emotion samples.

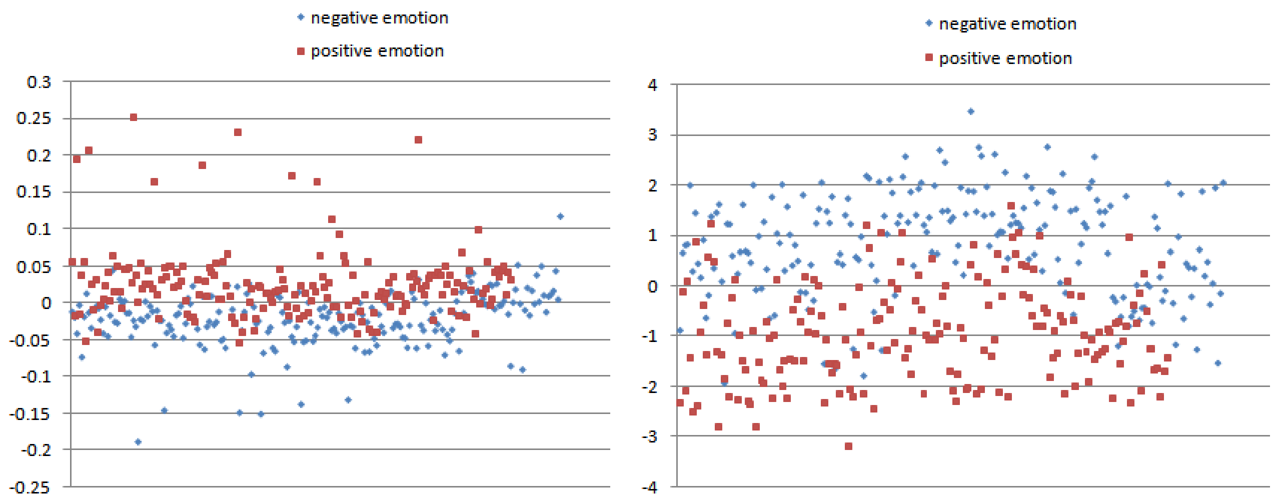


Figure 17: The projection found by the FDA (left) and LDA (right). The horizontal axis refers to the (random) number of the instance. The vertical axis refers to the value of each instance under the discrimination function (projection). Both the class means are closer and the within class variances is smaller under the FDA discriminant function than under that of the LDA.

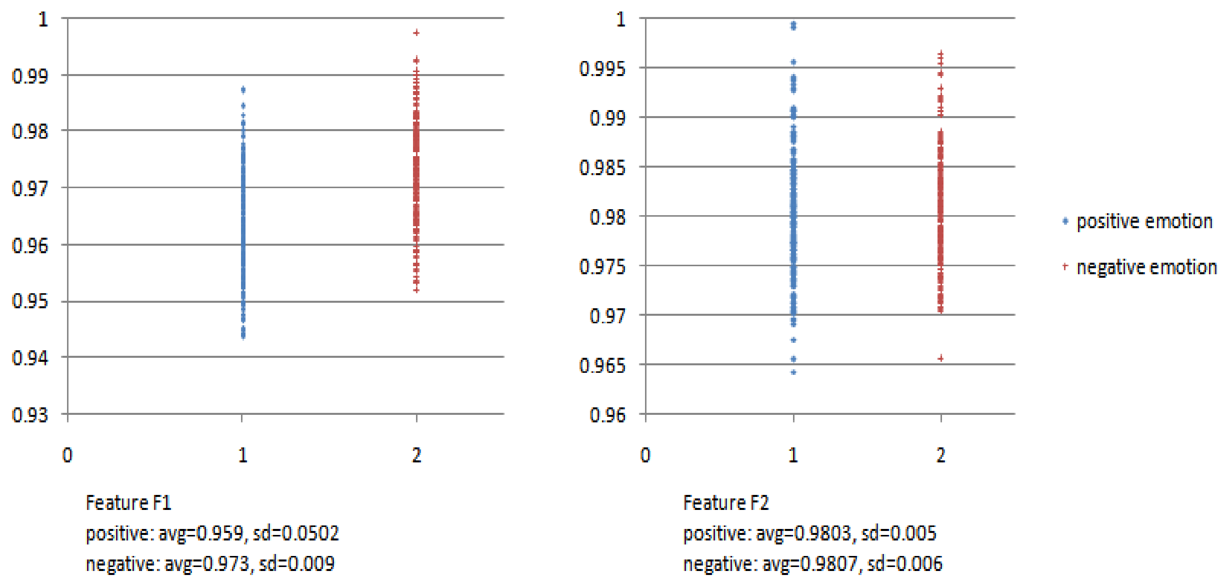


Figure 18: Visualization of the distributions of two features over the 2 classes: the vertical axis refer to the feature values (NCD values). Feature $F1$ has different distributions on the 2 classes and was found to have a significant mean difference in the t tests. Feature $F2$ has similar distribution over the 2 classes, had no significant mean differenc and was omitted.

the 2 classes.

Classification results of the SVM for all above methods are given in Table 13 (detailed results in Appendix 7.2.9). The PCA, LDA and the t tests selection method achieved better

Table 13: SVM emotion recognition results after applying dimensionality reduction techniques

Method	Avg	Sd	Worst
PCA	78.15	1.57	76.47
LDA	75.84	4.16	70
FDA	71.08	2.10	68.06
FA	70.58	4.52	64.7
t tests	74.95	4.30	69.74
reference: original 53 features	73.10	1.67	70.58

performance than using the original features. PCA yielded the best results, even above the non English speaker human performance. FDA and FA achieved worse accuracy than with the original dimensionality, however not significantly. It might be of interest to mention that in both the PCA, LDA and FDA we got significantly better results without standardizing the original features.

Confusion and types of errors Table 14 shows a confusion analysis of a typical run, using PCA extraction (119 test instances, overall accuracy of 76.4%). This implies the following

Table 14: Confusion matrix of a typical run using PCA

true emotion\classified as	positive	negative
positive	42	15
negative	13	49

probabilities:

1. $p_1 = p$(classifying positive emotion sample as positive emotion sample)$= 42/57 = 0.736$
2. $p_2 = p$(classifying positive emotion sample as negative emotion sample)$= 15/57 = 0.263$
3. $p_3 = p$(classifying negative emotion sample as positive emotion sample)$= 13/62 = 0.209$
4. $p_4 = p$(classifying negative emotion sample as negative emotion sample)$= 49/62 = 0.79$
5. $p_5 = p$(a sample that was classified as positive emotion is truly positive)$= 42/55 = 0.763$
6. $p_6 = p$(a sample that was classified as positive emotion is truly negative)$= 13/55 = 0.236$
7. $p_7 = p$(a sample that was classified as negative emotion is truly positive)$= 15/64 = 0.234$
8. $p_8 = p$(a sample that was classified as negative emotion is truly negative)$= 49/64 = 0.765$

As can be seen, the accuracy for negative emotion samples is slightly better than on good emotion samples, however the differences are not significant at 95% confidence level (the confidence interval for the differences of correct classification probabilities between positive and negative samples, $(p_4 - p_1)$ is $[-0.098, 0.206]$, containing zero, calculations in Appendix 7.2.9). The probability that a classification is correct (p_5, p_8) is almost identical for both classes. Overall, as we might want, the probabilities are sufficiently symmetric, hence the differences are acceptable.

Generalization In order to test the model's capability to generalize, the dataset was randomly divided to training and test sets and 4 separate runs were performed: in the first run the training set consisted of samples of all 3 speakers (denoted as $p6$, $r6$, $z6$). In each of the other runs, a different speaker was omitted from the training set. The test set was unchanged in all 4 runs and consisted of samples of all 3 speakers: 35 $p6$ samples, 27 $r6$ samples and 24 $z6$ samples. In each run we analyzed the accuracy rate for each speaker. Table 15 shows the accuracy (%) achieved for each speaker in each of the 4 runs. Note that in all runs the accu-

Table 15: Emotion recognition generalization performance

training speakers	$p6$ (1)	$r6$ (2)	$z6$ (3)	total
all speakers (a)	77.1	96.2	66.6	80.1
$p6$ omitted (b)	77.1	92.5	58.3	76.68
$r6$ omitted (c)	71.4	92.5	58.3	74.36
$z6$ omitted (d)	82.8	92.5	54.1	77.83

racy has some major differences on different speakers, from performance of over 90% for $r6$ to much worse accuracy for $z6$. One can also see that the overall accuracy slightly deteriorates whenever a speaker is omitted from the training set. It is interesting to note that the omission of $z6$ improves the accuracy on $p6$ samples. In addition, the omission of $p6$ deteriorated the classification results of $r6$ and $z6$, but not those of $p6$. However, our interest now is to check whether there are dependencies between the omission from the dataset and the errors for each speaker. Formally, we would like to perform the hypothesis test:

$$\begin{aligned} H_o &: \forall ij,\ p_{ij} = p_i p_j \\ H_1 &: \exists ij,\ p_{ij} \neq p_i p_j \end{aligned}$$

for $i \in \{a, b, c, d\}$, $j \in \{1, 2, 3\}$. The chi square statistic is $0.459 < 19.67 = \chi^2_{11,0.95}$, meaning we will not reject H_0 and say that there are no interactions between the speaker omission and the errors for each speaker. Similar result was obtained when comparing every proportion from one of the bottom 3 rows in Table 14 to the equivalent proportion in the first row (all 95% confidence intervals of the absolute proportion differences contained zero). Calculations for all statistical tests can be found in Appendix 7.2.9).

4.2.3 Expansion to more speakers and more emotions

With the above results, let us now expand our emotion recognition experimentation in the following manner: firstly, we enlarge the number of speakers from 3 to 10, among which are 3 adult men, 3 adult women, 2 teenagers and 2 children (one of each sex). Secondly, we perform a series of 5 2-class experiments, where instead of positive and negative classes, we will now try to separate between classes, differing in a more gentle way. The class definitions, as all samples, are taken from [42]. The classes and particular specification of each class in our new series of experiment are as follows:

- **absorbed**: absorbed, engaged, committed, concentrating, focused, thorough, involved, altogether 34 instances
- **thinking**: fantasizing, thinking, thoughtful, brooding, choosing, deciding, wool-gathering,

calculating, comprehending, altogether 80 instances

- **disagree**: argumentative, confrontational, contradictory, contrary, disagreeing, disapproving, disinclined, altogether 40 instances
- **excited**: alert, dynamic, lively, exhilarated, excited, inspired, invigorated, adventurous, altogether 47 instances
- **interested**: asking, curious, fascinated, probing, questioning, quizzical, scrutinizing, interested, altogether 47 instances
- **sure**: adamant, assertive, confident, sure, convinced, decided, knowing, determined, resolved, altogether 59 instances
- **unsure**: confused, clueless, faltering, hesitant, indecisive, unsure, undecided, insecure, ambivalent, puzzled, baffled, considering, debating, altogether 76 instances

We used 78 anchors, chosen randomly, and FDA to reduce the dimension of the input. The results of the 5 experiments are given in Table 16. In all experiments, the average accuracy

Table 16: Emotion recognition performance on 10 speakers and several emotions

Experiment	average	sd	worst
absorbed - disagree	79.09	9.42	68.18
absorbed - thinking	73.71	4.23	68.57
sure - unsure	75.234	3.98	69.04
interested - excited	71.72	5.66	65.51
absorbed - excited	77.694	6.32	69.23

was in between 70% and 80%. It is interesting to note that where the 2 classes reflect emotions that differ significantly, such as: absorbed - disagree, sure - unsure, absorbed - excited, the accuracy is higher than where the difference between the emotions is much more gentle, such as for absorbed - thinking and interested - excited. In terms of variance, the accuracy differences may derive from differences in the between class variance: it would be trivial to assume that the between class variance in the absorbed - excited experiment, for example, is much higher than in the interested - excited experiment, Thus, the results make some sense.

Summary of the emotion recognition experiments Emotion recognition experiments are a classic example to a domain in which it is difficult to define features that might affect the labeling of the samples. We showed that using the NCD and the anchor method one can classify voice samples according to emotions with similar accuracy to human performance. We got similar classification accuracy using 4 different classifiers. Accuracy was improved

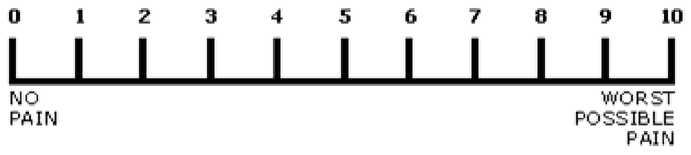


Figure 19: The VAS pain ruller, uses for a patient to define the level of pain he feels.

when applying dimensionality reduction methods. In case of the PCA, we got better results than those of the non English speakers in our human experiment. An analysis of the errors showed that both types of errors occur evenly. In the generalization experiments, we showed that testing the model on speakers for which the SVM was not trained, does not affect the accuracy for each speaker. A minor deterioration in the performance was noticed whenever a speaker is omitted however the statistical results encourage us to assume that this may not be the case if the dataset would contain more than 3 speakers, as in the above emotion recognition experiments. The experiment was then expanded to include more speakers, and more emotions. The results remain in [70% − 80%] range, with higher accuracy when the emotions differ more keenly.

4.3 Pain detection through ECG signals

Pain is an individual subjective experience that is ascribed to chemical or structural change in tissue, that is translated to a feeling of discomfort by the central neuro-system, with the intention to warn the body from the damage caused to that tissue. The formation mechanism of pain is assumed to be very complex. Pain gains importance in the medical community, however in contrast to other measures that can be objectively measured such as blood pressure, heartbeat, fever and breathing, pain, by its definition, is a subjective feeling that can not be quantified in objective means [17]. There are some tools aiming to create uniform evaluation of pain by patients and medical staff [15]. The most commonly used is the Virtual Analog Score (VAS), that is based on the patient's self evaluation of the pain it suffers from, by the time of his examination, as shown in Figure 19. This is a simple tool, that gives no more than a superficial evaluation of the patient's state. More complex tools, such as the Pain Management Index (PMI)and Brief Pain Inventory (BPI), are considered to be too awkward and are not in vast use by medical staff [15].

It is known that pain causes an increase in heart rate, and also in the heart rate variability [43]. In the past years, several spectral analysis methods and other mathematical methods for signal processing have been applied to ECG (Electrocardiogram) signals, which describes the electrical activity in ones heart, in order to investigate the possibility of detection of pain

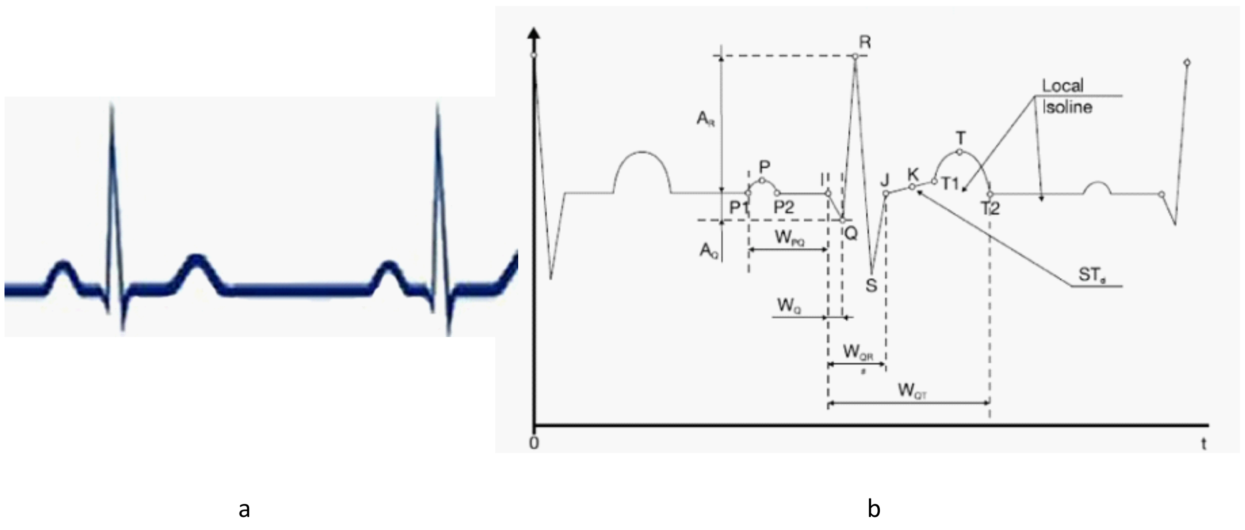


Figure 20: An example of ECG signal (a) and its fiducial measurements that use for extracting features (b). No subset of the commonly extracted features can capture existence of pain well

in patients. A common approach is to analyze the R-R interval, which reflects the heart rate variability (HRV). Some of the fiducial commonly extracted features of the ECG signal are shown in Figure 20. Although traditional methods such as Fourier and wavelets transforms can be efficient tools for denoising and identifying principal components and patterns in the signal, the common understanding is that the above methods can give only limited results, due to the fact that the ECG signal might embed sharp changes in very short time intervals [1]. As far as the author knows, in the field of biomedics there is no automatic way in clinical use that is capable of detecting the existence and type of pain that patients feel. This difficulty to that specialized methods for ECG analysis face in detecting pain, due to the fact that it is hard to identify ECG features that may explain the existence of pain, might be an advantage to NCD-based methods, just to spite their generality: considering the ECG signals globally and performing implicit or even no feature extraction.

To test our hypothesis we recorded a dataset that consisted of ECG signals of two patients. The signals were recorded in the pain clinic of Soroka hospital. We recorded signals both when a patient was in rest, and with one of his hands inside a container of ice-cold water, a popular method for generating pain [32], [46]. The recorded signals were then smoothed by rescaling each data point to an integer value in range to $[0, 255]$. The instances for the experiment were 10 seconds pieces of the recorded signals (with no intersection between every two pieces). The instances were labeled according to the level of pain the patient was feeling by the time of recording the signals: no_pain, increasing_pain and stable_pain. We conducted 2 and 3-class experiments, each experiment is conducted on samples from a single patient. 75% of the samples used for training. Technical data and results of both our methods on two patients are shown in Table 17 (detailed results in Appendix 7.2.10). Due to technical difficulties in

Table 17: Pain detection data and performance

patient	$p1$		$p2$	
total # of instances	34		30	
# of classes	2		3	
classifier	kNN	SVM-anc	kNN	SVM-anc
average	93.75	95.2	93.33	97.5
sd	5.59	6.5	9.12	5.5

collecting the data, only a relatively small number of instances could be recorded. However, despite the small number of training instances, in all cases results seem to be surprisingly high, though also the standard deviations seem to be quite high. This experiment was conducted on two patients only- an extremely small extent, hence can not be statistically valid, of course, and much more intensive experimentation is needed to validate these preliminary findings. However, we find these results most important, because samples in which the patients felt pain were practically separated from samples in which the patients did not feel pain, and the feasibility of NCD-based methods to detect pain through ECG signals in automatic means has been proved. A lot of work still needs to be done to validate the results, however, as for now, results are quite promising and the ECG experiments might therefore be considered as this thesis' "killer application".

5 Summary

5.1 Discussion

The NCD enables one to classify objects without extracting features. This unique method has received only limited attention in the literature, although featureless representation of objects is a promising area of research in pattern recognition and data mining. However, are the NCD-based classification methods really feature-free? Allegedly, no choice of specific features of the objects with which the comparison will be done was made, and this fact might introduce smaller bias to the classification algorithm in comparison to cases where a specific set of features is defined and chosen. In other words, no explicit features of the objects were used in our classification methods: the kNN algorithm did not make any use of feature vectors and the features extracted by the anchor method express compression distance to some fixed anchors and have no physical sense about the object itself. In addition, the NID and NCD of a pair of objects reflect their distance under the computable feature under which the two objects are most (in case of NID) or very (in case of NCD) similar, however, the uniqueness of these metrics is that this feature remains unknown and might be different for different pairs of objects. Let us give a closer examination at the so-called featurelessness and unbiasness of the NCD-based methods: recall from subsection 2.2.2 that the Kolmogorov complexity of a binary string x is defined in relation to a specific computation model, i.e., universal Turing machine, $K_{\Phi}(x)$. Similarly, the NCD of a pair of objects depends on the specific compressor that is used for the computation (normalized compression distances of a given pair of objects computed by different compressors will probably not be the same), thus the choice of compressor itself introduces some bias: every practically usable compressor will be blind to some possibly relevant features of the data. If that feature occurs in the data, it makes the data highly regular and compressible (low Kolmogorov complexity) but the compressor that is used will not compress it (large C-complexity). Different compressors will thus miss out on different features. Thus, implicitly, the choice of compressor amounts to the choice of a certain set of features. Still, this set of features is very large, so it is true that no features must be manually selected. However, the method is still not totally "feature free". A similar argument is valid regarding the choice of file representation, for example, our choice of using image files in bmp format. This choice is equivalent to a kind of feature extraction. This indeed occurs in a weak way, comparing to methods that require explicit specification of every attribute, however, it also turns the feature free-ness of the NCD under some reservations.

Feature free methods might be less biased in the sense that they would limit our ability to impose our prejudices, expectations, and presumptions on the problems and hand, and would let the raw data manifest itself. In addition, feature free methods might prevent the problem of reporting spurious patterns that do not really occur in the data, that can happen, for example, when generating new features out of original inputs by feature extraction methods. Indeed, this sounds promising. However, this concept represents a "data mining" view that is quite controversial in the machine learning community. In fact, many machine learning researchers might acknowledge that nothing can be learned without introducing some bias; this quite opposite view, is made most explicit in the subjective Bayesian school, but it is presented also in other paradigms. That said, you might do better job by making more explicit what sort of bias is acceptable and what is not. Putting it differently, the lack of parameters, while avoiding prejudices, also avoids the possibility to use domain knowledge in a beneficial way. NCD-based methods are truly general, which is a positive thing, however they might tend to fail to achieve the state-of-the-art in any given domain, much less to improve it, so one can see only limited consolation in the fact the NCD is feature free and relatively easy to apply. It was indeed the case in our olives, OCR and speaker identification experiments, where the general NCD-based methods achieved reasonable accuracy though were not comparative to the domain-specialized methods. However, we haven't noticed it in the Yale and Coil100 experiments, for instance, where the NCD-based methods achieved high accuracy and outperformed several specialized techniques. Nevertheless, a major exception for this approach is the case where it is rather difficult to define and extract features that might explain the target values (labeling), such as in the domains of emotion recognition and pain detection, where NCD-based methods might have an advantage over specialized methods, just to spite their being such general methods.

At the technical level, some issues should be taken under consideration. First, when using the anchor method, it is not obvious how many anchors should be used, and which specific choices of anchors can be made to increase the classification accuracy. We did not obtain a deep insight about which anchors should be used, however our results, as those of [10] show that random anchors work well most of the time. Determining the number of anchors can be ignored, if one uses dimensionality reduction methods such as PCA, where one can initially choose a large number of anchors and then choose as few principal components as is needed to explain the desired percentage of overall variance, or DA. Second, in image classification experiment, the question of how to select 256 colors in the denoising process has several answers. In the image experimentation, we used an automatic choice made by the Fotobatch software, however, one should be aware of the fact that there are many ways for color selection, and depending on the specific dataset one is working with, some of them might probably be better for classification. Third, the division modification of the kNN algorithm for images improved the classification accuracy in both the ORL and OCR experiments. A possible explanation might be that by dividing each image to 4 parts and working with 4 independent datasets, one reduces the within class variance, and increases the between-class variance, hence eases the classification. However, as more NCD calculations are made, this modification increases the running time significantly, making it not practical for large datasets, or large sized images. In addition, this modification was found to improve classification results, however, there are domains, such as our olive dataset, on which this modification might not work properly. Running time should generally be an important issue when running NCD-based algorithms, as compression is a time consuming computation process. Fourth, in our pain detection experiment, despite our success to separate ECG samples in which the patient felt different types of pain from each other, one can not yet guarantee that such method could *identify* pain in ECG samples, and separate it from other factors, like excitement of the patient, for example. However, one can use it to detect changes relatively to patients rest state.

The main criticism of the NCD approach is that in spite of the impressive experimental results, such as those in [12], [30], [16] and [24], beyond the basic theory (about universality of the incomputable NID distance), there is basically only experimental work. The research lacks analysis of why the method works or fails, and when one may expect whichever outcome. This difficulty appears also in this work: the present study contributes to the knowledge base by providing positive and negative examples, but despite our results, it seems hard to provide a more analytic discussion about the findings, as in the case of choosing the anchors, for example, and investigate to what extent the current general purpose compression tools can be of use in machine learning tasks of a much narrower scope.

5.2 Conclusions and further research

This thesis applies the normalized compression distance similarity metric, implemented using general-purpose compression tools, to classification problems in a variety of domains. We attempt to demonstrate how powerful the concept is, and its ability to succeed where explicit feature extraction methods fail. As far as the author knows, no such extensive NCD-based classification experimentation is documented in the current literature. The experimental results are reasonably satisfying and show a great potential on a wide range of problems, especially if one takes into account the small amount of preprocessing compared to other methods i.e., no explicit feature selection and extraction. The performance of the NCD based methods was often found to be competitive to the performance of highly specialized, tailored methods that often rely on feature extraction subroutines. In the face recognition experiments the kNN algorithm performed competitively to several state-of-the-art face recognition methods, and achieved high accuracy also where the illumination differs among the images. In addition, a sensitivity analysis of different training set sizes was given and a modification of the kNN algorithm that improved the accuracy in the ORL experiment was presented. In the experiment on the coil100 dataset, kNN algorithm outperformed several image classification methods and showed robustness to slight rotation of the objects in the image. The capability of the NCD-based methods to classify textures was examined and the SVM-anc was found to perform comparatively to a specialized texture classification method. The texture dataset was also used to compare the performance of polynomial kernel function and RBF kernel when using features extracted by the anchor method, and found that they both achieved similar results, however as RBF kernel uses less parameters, we find it more convenient for use. In the OCR experiment, the NCD-based methods failed to achieve reasonable performance. Additional results on the OCR dataset showed that the performance of a NCD-based kernel function for SVM achieves worse performance than using RBF kernel and the anchor method, and that the naive Hamming distance surprisingly achieves better classification accuracy than the NCD, as a measure of dissimilarity in OCR. An optional explanation to the fair results in our OCR experiments was the prior manipulation in which the images size was significantly enlarged. This manipulation may had a destructive effect on the capability of the NCD to track true similarities between the images. The performance of the NCD-based methods on real world biological data was examined, where they achieved reasonable performance, however worse than using pre-defined and manually extracted features. In the voice samples experimental section, results of the speaker identification experiments showed that NCD based methods can be successfully applied for this task. In addition, the potential of using NCD-based classification methods on domains in which it is difficult to define and extract features that might be related to the labeling of the instances was demonstrated. In such cases, the NCD can be used in discovering unknown features in which the data can be similar. Two such domains are emotion recognition through voice samples and pain detection through ECG signals. In the emotion recognition experiments, the SVM-anc method achieved similar performance to human performance, and even outperformed human accuracy of non English speakers. A comparison between several classifiers to be used with the anchor method was performed and all achieved similar performance, however the SVM was found to be the most convenient one to use. Further, we integrated the NCD-methods with several dimensionality reduction methods and found that PCA turns to be the most effective in the emotion recognition experiment. In addition, error percentages on the classes were examined and found to be similar and the model built in the experiment showed a good generalization ability, i.e., it can classify samples of a speaker it wasn't trained on with similar accuracy to samples of speakers on which it was trained. The experiment was then expanded to comprise more speakers and more emotions. The results revealed that NCD-based classification can be applied also to greater number of speakers, and more emotions, with accuracy similar to the positive-negative experiment, however depending on how sharply do the two emotions differ. In the ECG experiment we showed that NCD-based methods can be used to classify ECG samples according to the type of pain the patient was feeling during recording the signal. This result is important due to the fact that there is no automatic tool in biomedicine that can tell whether a patient feels pain.

Except for their being feature free, in a sense, another major advantage of NCD-based methods is the very little, if any at all, manipulation of the input that is necessary. Among the 2 classification algorithms applied in this work, the SVM, incorporated with the anchor method, appears to be more robust and achieves better performance than the simpler kNN. It might be interesting to note that several results imply that NCD-based classifiers have some sort of "human capabilities": this was the case in the emotion recognition experiment, where the results of the SVM-anc classifier were very close to the human performance, and where the classification accuracy in the expanded experiment deteriorated as the two compared emotions were intuitively closer. Another example is taken from the image classification experiments, where the kNN's performance was reasonable, as long as the class differences could also be captured by human eye, see Remark 11, excluding the OCR results.

Though the ideas here are few years old, the documented experimentation is not abundant. We believe that there is great potential embedded in NCD-based learning methods, due to the robustness of the methods, as well as they being general and less biased than methods that require explicit feature selection and extraction.

We believe that this empirical work can be a step toward a more global understanding of when NCD-based methods work or fail. An ongoing research is being conducted in the emotion recognition area, in which we expand the classification experiment to 9 distinct emotions. Another ongoing research is being done in the pain detection area, in which we aim to found our results by collecting more samples, from more patients. Among optional areas for future research in NCD-based learning methods are developing a predictive ability to the success of NCD-based classification and clustering, together with ways to optimize the performance of NCD-based methods, for example by an analysis of an intelligent choice of anchors (apart from dimensionality reduction methods), or finding the best (known) compressors to be used for NCD calculations, depending on the type of data. Running time considerations, that were not taken in this work, have also a major significance in NCD experiments and minimizing running time is also an important practical goal.

References

[1] M. Aharon, M. Elad, and A.M. Bruckstein, "On the Uniqueness of Overcomplete Dictionaries, and a Practical Way to Retrieve Them", *Journal of Linear Algebra and Applications*, vol. 416, 2006, pp. 48-67.

[2] M. Anthony, and P. L. Bartlett, *Neural Network Learning: Theoretical Foundations*, Cambridge University Press, 1999.

[3] S. Arora, and B. Barak, *Complexity Theory: A Modern Approach*, To be published by Cambridge University Press, around March 2009.

[4] R. Ash, *Information Theory*, Dover Publications, New York, 1990.

[5] T. Bell, J. Cleary, and I. Witten, "Data Compression Using Adaptive Coding and Partial String Matching", *IEEE Transactions on Communications*, vol. 32, no. 4, 1984, pp. 396–402.

[6] C.H. Bennet, P. Gacs, M. Li, P.M.B. Vitanyi, and W. Zurek, "Information Distance", *IEEE Transactions on Information Theory*, vol. 44, no. 4, 1998, pp. 1407-1423.

[7] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[8] J. Bromley, and E. Sackinger, *Neural Network and k-Nearest Neighbor Classifiers.*, Technical Report 11359-910819-16TM, AT&T, 1991.

[9] M. Burrows, and D.J. Wheeler, "A Block-Sorting Lossless Data Compression Algorithm", Technical Report 124, Digital Systems Research Center, May 1994.

[10] R. Cilibrasi, *Statistical Inference Through Data Compression*, Ph.D. thesis, Institute for Logic, Language and Computation, Universiteit van Amsterdam, Holland, 2007.

[11] R. Cilibrasi, R. de Wolf, and P.M.B. Vitanyi, "Algorithmic Clustering of Music Based on String Compression", *Computer Music Journal*, vol. 28, no. 4, 2004, pp. 49-67.

[12] R. Cilibrasi, and P.M.B. Vitanyi, "Clustering by Compression", corrected version of *IEEE Transactions on Information Theory*, vol. 51, no. 4, 2005, pp. 1523-1545.

[13] R. Cilibrasi, and P. Vitanyi, "A New Quartet Tree Heuristic for Hierarchical Clustering", http://arxiv.org/abs/cs/0606048.

[14] R.L. Cilibrasi, and P.M.B Vitanyi, "The Google Similarity Distance", *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 3, 2007, pp. 370-383.

[15] C.S. Cleeland, "Assessment of Pain in Cancer Measurement Issues", *Advances in Pain Research Therapy.* vol. 16, 1990, pp. 47-55.

[16] C. Costa Santos, J. Bernardes, P.M.B. Vitanyi, and L. Antunes, "Clustering Fetal Heart Rate Tacings by Compression", Proc. 19th *IEEE Symp. Computer-Based Medical Systems*, 2006, pp. 685-690.

[17] M.P. Davis, and D. Walsh, "Cancer Pain: How to Measure the Fifth Vital Sign", *Cleveland Clinic Journal of Medicine* 2004, vol 71, no.8, pp. 625-632.

[18] R. De Maesschalck, D. Jouan-Rimbaud and D. L. Massart, "The Mahalanobis Distance" ,*Chemometrics and Intelligent Laboratory Systems*, vol. 50, no. 1, 2000, pp. 1-18.

[19] R. O. Duda, P.E. Hart, and D.G Stork, *Pattern Classification,* John Wiley and sons, New York, 2001.

[20] C. L. Giles, and T. Maxwell, "Learning, Invariance, and Generalization in High-Order Neural Networks", *Applied Optics*, vol. 26, 1987, pp. 4972-4978.

[21] J.F. Hair, W.C. Black, B.J. Babin, R.E. Anderson, and R.L. Tatham, *Multivariate Data Analysis*, Pearson Prentice Hall, New Jersey, 6th edition 2006.

[22] J. Han, and M. Kamber, *Data Mining: Concepts and Techniques.* Morgan Kaufmann, 2006.

[23] P. Howland, J. Wang, and H .Park, "Solving the Small Sample Size Problem in Face Recognition using Generalized Discriminant Analysis", *Pattern Recognition*, vol. 39, 2006, pp. 277 – 287.

[24] E. Keogh, S. Lonardi, and C.A. Rtanamahatana, "Towards Parameter-Free Data Mining", In: Proc. 10th ACM SIGKDD *Int'l Conf. Knowledge discovery and data mining*, Seattle, Washington, USA, August 22-25, 2004, pp. 206-215.

[25] L.G. Kraft, *A device for quantizing, grouping and coding amplitude modulated pulses*, Master's thesis, Dept. of Electrical Engineering, M.I.T., Cambridge, Mass., 1949.

[26] S. Laykin, *On-Line Multi-Stage Classifier for Agricultural Sorting Systems*, Ph.D. Thesis, Dept. of Industrial Engineering & Management, Ben Gurion University of the Negev, Israel, 2006.

[27] S. Lazebnik, C. Schmid, and J. Ponce. "A Sparse Texture Representation using Local Affine Regions", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, 2005, pp. 1265-1278.

[28] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation Applied to Handwritten Zip Code Recognition", *Neural Computation*, vol. 1, no. 4, 1989, pp. 541–551.

[29] M. Li, and P.M.B. Vitanyi, *An introduction to Kolmogorov Complexity and its Applications*, Springer-Verlag, New York, 2nd Edition 1997.

[30] M. Li, X. Chen, X. Li, B. Ma, and P.M.B. Vitanyi, "The Similarity Metric", *IEEE Transactions on Information Theory*, vol. 50, no.12, 2004, pp. 3250-3264.

[31] X. Liua, A. Srivastavab, and D. Wang, "Intrinsic Generalization Analysis of Low Dimensional Representations", *Neural Networks,* vol. 16, no. 5-6, 2003, pp. 537–545.

[32] M.S. Menkes, K.A. Matthews, D.S. Krantz, U. Lundberg, L.A. Mead, B .Qaqish, K.Y. Liang, C.B. Thomas, and T.A. Pearson, "Cardiovascular Reactivity to the Cold Pressor Test as a Predictor of Hypertension", *Hypertension*, vol. 14, 1989, pp. 524-530.

[33] D. Michie, D.J. Spiegelhalter, C.C. Taylor, and J. Campbell , *Machine Learning, Neural and Statistical Classification*, Ellis Horwood Upper Saddle River, New Jersey, 1995.

[34] T. M. Mitchell, *Machine Learning.* McGraw-Hill, New York, 1997.

[35] K-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf, "An Introduction to Kernel-Based Learning Algorithms", *IEEE Transactions on Neural Networks*, vol. 12, no. 2, 2001 pp.181-201.

[36] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, Wiley-Interscience, 2007.

[37] D. A. Reynolds, "Speaker Identification and Verification using Gaussian Mixture Speaker Models", *Speech Communication*, vol. 17, no. 1-2, 1995, pp. 91-98.

[38] D. Salomon, *Data Compression: the Complete Reference*, Springer, New York, 2000.

[39] B. Schölkopf, *Support Vector Learning*, Oldenbourg-Verlag, Munich, 1997.

[40] B. Schölkopf, and A. J. Smola, *Learning with Kernels*, MIT Press, 2002.

[41] C. E. Shannon, "A mathematical Theory of Communication", *Bell Systems Technical Journal*, vol. 27, 1948, pp. 379–423 and 623–656.

[42] T. S. Shikler, *Le Ton Fait la Musique: Analysis of Expressions in Speech*, Ph.D. dissertation, Computer Laboratory, University of Cambridge, United Kingdom, 2007.

[43] R. J. Storella, Y. Shi, D. M. O'Connor, G. H. Pharo, J. T. Abrams, and J. Levitt, "Relief of Chronic Pain May Be Accompanied by an Increase in a Measure of Heart Rate Variability", *Anesthesia Analgesia*, vol. 89, 1999, pp. 448–450.

[44] M. Turk, and A. Pentland, "Eigenfaces for Recognition", *Journal Of Cognitive Neurosciences*, vol. 3, no. 1, pp. 1991, 71-86.

[45] S. Wehner, "Analyzing Network Traffic and Worms using Compression", *Journal of Computer Security*, IOS Press, vol. 15, no. 3, pp. 303-320.

[46] D.L. Wood, S.G. Sheps, L.R. Elveback, and A. Schirger, "Cold Pressor Test as a Predictor of Hypertension", *Hypertension*, vol. 6, 1984, pp. 301-306.

[47] B. Zhang, and S. N. Srihari, "Fast k-Nearest Neighbor Classification Using Cluster-Based Trees", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 4, 2004, pp 525-528.

[48] H. Zhang, A. C. Berg, M. Maire and J. Malik, "SVM-kNN: Discriminative Nearest Neighbor Classification for Visual Category Recognition", *IEEE Computer Society Conf. Computer Vision and Pattern Recognition*, vol. 2, 2006, pp. 2126-2136.

[49] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "*Face Recognition: A Literature Survey", ACM Computing Surveys (CSUR)*, vol. 35 , issue 4, 2003, pp. 399-458.

6 Papers by U. Shaham

1. U. Shaham and Y. Edan, "Classification using Normalized Compression Distance", Proc. *Int'l Conf. Artificial Intelligence and Pattern Recognition*, Orlando, Florida, USA, July 7-10, 2008, pp.63-69.

7 Appendices

7.1 Universality of the NID [30]

The NID is universal in the sense that if objects are similar according to a particular computable feature, then they are at least that similar under the NID. The proof shows that the NCD is at least as small as any distance in the wide class of computable normalized distances (recall the normalization condition from 2.3.3). This class is so wide that it will capture everything that can be remotely of interest.

Theorem 1 : *The NID minorizes every computable normalized distance $f(x,y)$ by*

$$NID(x,y) \leq f(x,v) + O\left(\frac{1}{K}\right)$$

where $K = \min\{K(x), K(y)\}$.

Proof. : Let x, y be a pair of objects and let f be a computable normalized distance. Let $f(x,y) = e$. Without loss of generality assume that $K(x) \leq K(y)$. Then, given x we can recursively enumerate the pairs (x,v) such that $f(x,v) \leq e$. Note that the enumeration contains (x,y). By the normalization condition, the number of pairs enumerated is less than $2^{eK(x)+1}$. Every such pair, in particular (x,y), can be described by its index of length $eK(x)+1$ in this enumeration. Since the Kolmogorov complexity is the length of the shortest effective description, given x, the binary length of the index plus an $O(1)$ bit program to perform the recovery of y must at least be as large as the Kolmogorov complexity, which yields

$$K(y \mid x) \leq eK(x) + O(1)$$

$K(x) \leq K(y)$, hence

$$NID(x,y) = \frac{K(y \mid x)}{K(y)}$$

and substitution gives

$$NID(x,y) = \frac{K(y \mid x)}{K(y)} \leq \frac{eK(x) + O(1)}{K(x)} \leq f(x,y) + O\left(\frac{1}{K(x)}\right)$$

■

7.2 Detailed results

7.2.1 ORL dataset: Effect of changing image size and number of colors

Table 18 shows the average performance of the kNN algorithm on the AT&T ORL dataset, using 70% of the images for training set, for different values of k. The original image size is 92×112. One can notice here that $k = 1$ had the best results among all values of k on this

Table 18: Effect of number of colors and image size

Image size	Bit per pixel	1NN	3NN	4NN	5NN	6NN	7NN
41×50	8	93.3	91.6	90.8	90	90.8	90
41×50	32	92.5	89.1	91.6	88.3	85	82.5
57×70	8	95	91.6	91.6	90.8	90.8	90
57×70	32	95	95	95.8	92.5	91.6	90
74×90	8	95.8	91.6	90	90	87.5	87.5
74×90	32	96.6	96.6	93.3	89.1	89.1	84.1
98×120	8	98.3	91.6	90	88.3	86.6	82.5
98×120	32	98.3	95.8	95.8	90.8	90.8	86.6

dataset. When analyzing only the results for $k = 1$, we can further see that the results are generally better as the size of the image size is closer to the original size, meaning that some meaningful information is lost when reducing the size of the image. For a given image size, the accuracy using true color (32 bit per pixel) is slightly better than using 256 colors (8 bit per pixel), however, as it is only a slight difference, and using true color images increases the size of the image file significantly, making its compression very time consuming, we decided to continue our experiments on this dataset using images of the original sizes, and 256 colors.

7.2.2 ORL dataset: Effect of different numbers of training images

Table 19 shows the performance of the kNN algorithm on the ORL dataset, for different sizes of training set. One can see that here again, $k = 1$ gives the best accuracy. In addition, the algorithm reaches its best accuracy only for above 50% training set (meaning 5 images of each person).

7.2.3 ORL dataset: kNN division modification

Table 20 shows the results of 5 runs using the modified kNN on the ORL dataset, with 7 training images per person.

Table 19: Effect different values of k and training set percentage

Training images	1NN	3NN	4NN	5NN	6NN	7NN	8NN	9NN
1	56.3	–	–	–	–	–	–	–
1	57.77	–	–	–	–	–	–	–
3	85	80	–	–	–	–	–	–
3	80	76.07	–	–	–	–	–	–
5	94	88	86.5	83	–	–	–	–
5	90	87	85.5	82.5	–	–	–	–
5	92.5	89.5	87	83.5	–	–	–	–
5	93.5	91.5	91	86	–	–	–	–
7	98.3	91.6	90	88.3	86.6	82.5	–	–
7	93.3	89.1	86.6	86.6	85	80.8	–	–
9	1	97.5	95	92.5	90	90	90	80
9	97.5	95	95	95	97.5	97.5	95	85

Table 20: Performance of the modified kNN on the ORL dataset

Run 1	Run 2	Run 3	Run 4	Run 5	Average	Sd	Worst
98.3	98.3	100	98.3	100	99.0	0.93	98.3

7.2.4 Yale dataset classification results

Table 21 shows the results of 5 runs using the kNN algorithm on the Yale dataset, with 23 training images per person.

Table 21: Performance of the kNN algorithm on the Yale dataset

Run \\k	1	3	4	5	6	7	8	9	10	11	12	13	14	15	16
run 1	100	100	100	100	98.5	98.5	98.5	98.5	100	100	100	100	100	98.5	98.5
run 2	95.7	95.7	95.7	97.1	98.5	97.1	97.1	97.1	97.1	97.1	97.1	97.1	97.1	97.1	97.1
run 3	98.5	98.5	98.5	98.5	98.5	98.5	98.5	98.5	98.5	100	100	100	100	100	100
run 4	100	100	100	100	100	100	100	100	100	100	100	98.5	98.5	98.5	98.5
run 5	98.5	98.5	100	100	100	100	100	100	100	100	100	100	100	100	100

7.2.5 Coil100 dataset classification results

Table 22 shows the results of 5 runs using the kNN algorithm on the coil100 dataset, with 5 training images per each of the 50 objects, with 8 and 24 bit per pixel images.

Table 22: Performance of the kNN algorithm on the coil100 dataset

	8 bit per pixel				24 bit per pixel			
Run \\k	1	3	4	5	1	3	4	5
Run 1	96.8	96.8	96.4	94.8	98.8	98.4	98.8	96
Run 2	98	95.6	96.8	96	98.8	98.4	98.4	96.8
Run 3	96	96.4	96.8	96.8	99.6	97.2	97.2	96.8
Run 4	97.2	97.2	96.8	94.4	99.2	98	98.4	97.2
Run 5	97.6	98	98	97.6	99.6	98.4	99.2	97.2

7.2.6 Texture classification results

Table 23 shows the results of 5 runs using the kNN (for its best value, $k = 1$, 11 class experiment) and the SVM using the anchor method (15 class experiment) on the texture dataset, with 30 training images per each class. The SVM-anc used feature vectors created by 20 randomly chosen anchors. The parameters for the RBF kernel function (found by cross validation) were $\gamma = 2.82$, $c = 1024$, and for the polynomial kernel function ($\gamma(\underline{u} \cdot \underline{v} + r)^d$) were $\gamma = d = 4$, $r = 1024$, $c = 2048$.

Table 23: Performance of the kNN and SVM-anc on the texture dataset

Classifier\\Accuracy	Run 1	Run 2	Run 3	Run 4	Run 5
1NN	76.3	68.1	73.6	79	69
SVM-anc, RBF kernel	86	82	84	78	90.66
SVM-anc, polynomial Kernel	84.66	80.66	88.66	82	88.66

7.2.7 OCR classification results

Table 24 shows the results of 5 runs using the kNN (for its best value, $k = 1$) and the SVM using the anchor method on the USPS dataset, with 200 training images per each class. The SVM-anc used feature vectors created by 20 randomly chosen anchors. The parameters for the RBF kernel function (found by cross validation) were $\gamma = 1.41$, $c = 1024$. The parameters for the SVM-anc with the Hamming distance were $\gamma = 1$, $c = 45.25$.

Table 24: Performance on the USPS dataset dataset

experiment	Run 1	Run 2	Run 3	Run 4	Run 5	Avg
kNN	35.6	42.3	47.2	36.1	38.8	40.0
kNN, division modification	50.3	52.4	60.8	61.3	57.2	56.4
SVM-anc	60.1	65.2	59.3	63.4	64	62.4
SVM, $\frac{1}{NCD(\cdot,\cdot)}$ kernel	35.1	45.2	43.6	44.4	37.7	41.2
SVM-anc, Hamming distance	79.5	84.5	83.5	77.66	78.5	80.7

7.2.8 Olives classification results

Table 25 shows the technical details of the experiments of the quality degree identification and results of 5 runs using the kNN (for its best value, $k = 1$) and the SVM using the anchor method.

Table 25: Performance of the kNN and SVM-anc classifiers on the olives dataset

	Technical details		1NN						
Species	# of classes	Images per class	Run 1	Run 2	Run 3	Run 4	Run 5	Avg	Sd
1	3	23	85.7	95.2	95.2	95.2	95.2	93.3	4.2
2	2	185	88.3	90.6	87.2	87.9	89.9	88.8	1.4
3	3	20	75.0	66.6	83.3	100	75.5	80.1	11.6
4	2	71	95.2	85.7	97.6	90.4	97.6	93.3	5.2
5	3	106	82.8	81.2	79.1	80.7	81.1	81.0	1.3

	SVM-anc						
Species	# of anchors	Run 1	Run 2	Run 3	Run 4	Avg	Sd
1	12	81.6	79.31	79.31	81.6	80.45	1.32
2	9	93.04	94.78	90.43	90.43	92.17	2.13
3	13	84.78	73.91	82.6	89.13	82.6	5.54
4	12	90.47	88.09	95.23	97.61	92.85	4.34
5	13	80.11	78.5	83.17	76.63	80.60	3.60

There seems to be no clear effect of the number of anchors on the accuracy, neither a dependence of the accuracy in the size of the classes. The differences in the accuracy over different species seem to result from the within and between class variance embedded in each species.

Table 26 shows the technical details and results of the experiments of the species identification. The latter experiment has the worst results, however note that its conditions (# of classes) are more difficult than these of the first two experiments.

Table 26: The kNN's performance in the olives type identification experiment

Dataset	# of classes	Images per class	k
mixed species deg. 3	5	42	1
mixed species deg. 1	3	22	all
mixed species all degs	7	30	3

dataset	run 1	run 2	run 3	run 4	run 5	avg	sd
mixed species deg. 3	86.2	93.8	90.8	93.8	90.8	91.1	3.1
mixed species deg. 1	100	100	100	100	100	100	0
mixed species all degs	80.9	80.9	80.9	83.3	88.1	82.8	3.12

7.2.9 Voice samples classification results

Speaker identification results Table 27 contains the details of the speaker identification experiments. Altogether there were 179 voice samples in the 4-class experiment and 250 in the 10 class experiment. All experiments were done using 75% of the instances for training. The SVM-anc used 15 anchors.

Table 27: Speaker identification experiments

	Classifier	Parameters	Run 1	Run 2	Run 3	Run 4	Run 5	Avg	Sd
4 speakers	kNN	$k = 1$	90.2	93.5	89.1	95.2	94.35	92.47	2.67
4 speakers	SVM-anc	$c = 1024, \gamma = 2.82$	78.5	82.3	77.8	84.1	77.3	80	3.01
10 speakers	kNN	$k = 1$	87.4	81.2	84.5	81.3	88.6	85.2	2.87
10 speakers	SVM-anc	$c = 2048, \gamma = 2.82$	61.3	66.4	68.3	62.1	68.9	65.4	3.51

Emotion recognition results Table 28 contains the results of the 53 feature experiment. After comparison between different possible architectures for the ANN, we conducted the experiment with the best one found, of single hidden layer with 15 neurons. In addition, we used 200 training epochs and learning rate of 0.1. The decision tree used was of type $C5$. We used boosting and limited the number of trials to 10. In addition we used pruning severity of 75 and minimal number of 10 instances per branch. The SVM parameters were $\gamma = 1.41$, $, c = 477$. Table 29 shows the performance of the feature selection and extraction methods

Table 28: Emotion recognition results with 53 anchors

Classifier	Run 1	Run 2	Run 3	Run 4	Run 5	Avg	Sd
SVM-anc	74.78	70.58	73.94	73.94	72.26	73.10	1.67
ANN-anc	70	77.69	74.62	74.62	71.54	73.69	2.99
DT-anc	68.46	73.08	76.15	72.31	71.54	72.30	2.77
Centroids-anc	73.1	73.1	76.47	73.1	69.74	73.10	2.37

used in the emotion recognition experiment.

Table 29: Emotion recognition results using dimensionality reduction

Method	Run 1	Run 2	Run 3	Run 4	Run 5	Avg	Sd
PCA	74.47	79.83	76.47	78.99	78.99	78.15	1.57
LDA	80.77	73.85	78.46	76.15	70	75.846	4.16
FDA	73.1	72.26	69.74	68.06	72.26	71.08	2.10
FA	73.1	70.58	68.06	76.47	64.7	70.58	4.52
t tests	69.74	73.1	79.83	78.99	73.1	74.95	4.30
PCA with standardization	59.66	59.66	62.18	57.98	60.5	59.99	1.52
FDA with standardization	66.38	65.54	68.9	69.74	66.38	67.33	2.45

Table 30: Detailes and performance of the 10 speakers emotion recognition experiment

Experiment	γ, c	Run 1	Run 2	Run 3	Run 4	Run 5	Average	Sd	Worst
absorbed - disagree	$1, 1024$	77.27	68.18	86.36	72.73	90.91	79.09	9.42	68.18
absorbed - thinking	$2.82, 256$	80	71.42	74.28	68.57	74.28	73.71	4.23	68.57
sure - unsure	$11.31, 16$	76.19	69.04	73.80	78.57	78.57	75.234	3.98	69.04
interested - excited	$1.41, 128$	68.96	75.86	68.96	65.51	79.31	71.72	5.66	65.51
absorbed - excited	$1.41, 256$	80.77	80.77	73.08	84.62	69.23	77.694	6.32	69.23

Statistical tests

- Confidence interval for $(p_4 - p_1)$: we use a formula for $1-\alpha$ confidence interval for difference between proportions:

$$\widehat{p_1} - \widehat{p_2} \pm z_{1-\frac{\alpha}{2}} \sqrt{\frac{\widehat{p_1}(1-\widehat{p_1})}{n_1} + \frac{\widehat{p_2}(1-\widehat{p_2})}{n_2}} \tag{6}$$

  and get:

$$0.79 - 0.736 \pm 1.96\sqrt{\frac{0.79(1-0.79)}{62} + \frac{0.736(1-0.736)}{57}}$$

giving

$$(p_4 - p_1) \in [-0.098, 0.206]$$

hence we conclude that the proportions difference is not significant in 95% level.

- Chi square test for independency: Table 31 shows the observed and expected numbers of correct classifications for each speaker in each of the 4 runs performed. The χ^2 statistic is $\sum_i \frac{(\text{observed}_i\text{-expected}_i)^2}{\text{expected}_i} = 0.459 < \chi^2_{11,0.95} = 19.67$, meaning the probabilities are independent indeed.

- probabilities differences confidence intervals: we use the formula in (6) to obtain the following 95% intervals for the difference between each proportion in one of the bottom 3 rows in Table 14 to the equivalent probability in the top row. The confidence intervals are given in Table 32.

Table 31: Chi square test for independency

Training speakers		Errors on $p6$	Errors on $r6$	Errors on $z6$	Total
all speakers (a)	observed	27	26	16	69
	expected	28.02	26.20	14.79	
$p6$ omitted (b)	observed	27	25	14	66
	expected	26.80	25.06	14.14	
$r6$ omitted (c)	observed	25	25	14	64
	expected	25.98	24.30	13.71	
$z6$ omitted (d)	observed	29	25	13	67
	expected	27.20	25.44	14.36	
total		108	101	57	266

Table 32: Confidence intervals for probabilities differences

Training speakers	Errors on $p6$	Errors on $r6$	Errors on $z6$	Total
$p6$ omitted (b)	$(-0.16, 0.16)$	$(-0.06, 0.13)$	$(-0.10, 0.27)$	$(-0.13, 0.19)$
$r6$ omitted (c)	$(-0.16, 0.27)$	$(-0.08, 0.15)$	$(-0.16, 0.32)$	$(-0.18, 0.23)$
$z6$ omitted (d)	$(-0.27, 0.16)$	$(-0.09, 0.16)$	$(-0.14, 0.39)$	$(-0.18, 0.17)$

7.2.10 Pain detection results

Table 33 shows the performance of the kNN and SVM in the pain detection experiment. For patient $p1$ we used 17 anchors for the SVM-anc and 15 anchors were used for patient $p2$. All anchors were chosen randomly.

Table 33: Pain detection performance

Patient	$p1$								$p2$							
Run	1	2	3	4	5	Avg	Sd	Parameters	1	2	3	4	5	Avg	Sd	Parameters
kNN	100	100	100	100	87.5	97.5	5.5	$k = 3$	83.3	100	83.3	100	100	93.3	9.1	$k = 3$
SVM-anc	88	100	100	88	100	95.2	6.5	$c = 128$, $\gamma = 2.82$	100	100	100	100	87.5	97.5	5.5	$c = 256$, $\gamma = 2.82$

7.3 Software, raw data and code written in this work

The SVM classification experiments were performed using the libsvm 2.85 software (*http:// www.csie.ntu.edu.tw/~cjlin/libsvm*). bzip2 compression was done using Rudi Cilibrasi's complearn software (*http://www.complearn.org/*). Image denoising and manipulations were done using Fotobach software (*http:// www.keksoft.com/index2.htm*). The LDA was performed using SPSS. Code for PCA and FDA was written in Matlab, as well as a code for converting a csv file for a convenient format for the libsvm format and a code for Hamming distance computation of squared images. The ppm compression, kNN, the kNN's division modification and nearest centroid classifier codes were written in Java. All datasets used in this work, as well as the code written as part of the thesis can be found in the attached CD, as well as a readme file with explanations of the code files.