JMLR: Workshop and Conference Proceedings 13: 127-143
2nd Asian Conference on Machine Learning (ACML2010), Tokyo, Japan, Nov. 8–10, 2010.

The Coding Divergence for Measuring the Complexity of Separating Two Sets

Mahito Sugiyama MAHITO@IIP.IST.I.KYOTO-U.AC.JP
Graduate School of Informatics, Kyoto University
Yoshida Honmachi, Sakyo-ku, Kyoto, 606-8501, Japan
Research Fellow of the Japan Society for the Promotion of Science

Akihiro Yamamoto AKIHIRO@I.KYOTO-U.AC.JP
Graduate School of Informatics, Kyoto University

Editor: Masashi Sugiyama and Qiang Yang

Abstract

In this paper we integrate two essential processes, *discretization* of continuous data and learning of a model that explains them, towards fully *computational* machine learning from continuous data. Discretization is fundamental for machine learning and data mining, since every continuous datum; *e.g.*, a real-valued datum obtained by observation in the real world, must be discretized and converted from analog (continuous) to digital (discrete) form to store in databases. However, most machine learning methods do not pay attention to the situation; *i.e.*, they use digital data in actual applications on a computer whereas assume analog data (usually vectors of real numbers) theoretically. To bridge the gap, we propose a novel measure of the difference between two sets of data, called the *coding divergence*, and unify two processes discretization and learning computationally. Discretization of continuous data is realized by a topological mapping (in the sense of mathematics) from the d-dimensional Euclidean space $\mathbb{R}^d$ into the Cantor space Σ^ω, and the simplest model is learned in the Cantor space, which corresponds to the minimum *open set* separating the given two sets of data. Furthermore, we construct a classifier using the divergence, and experimentally demonstrate robust performance of it. Our contribution is not only introducing a new measure from the computational point of view, but also triggering more interaction between experimental science and machine learning.

Keywords: Coding divergence, Discretization, Cantor space, Binary encoding, Computable Analysis, Computational Learning Theory

1. Introduction

The aim of this paper is giving a computational basis for machine learning and data mining from continuous data. We integrate two fundamental processes; *discretization* of continuous data and learning of a model that explains them, using theories of encoding of real numbers in Computable Analysis (Weihrauch, 2000) and those of learning from examples in Computational Learning Theory (Jain et al., 1999). We treat the problem of measuring the difference between two sets of continuous data by assuming that one contains positive data and the other contains negative data, since such a problem is basic for a lot of tasks in machine learning such as classification and clustering. We propose a novel measure, called the *coding divergence*, to measure similarity between such sets, and construct a classifier using it. The key concept of the divergence is the simple procedure *separation*:

The coding divergence measures the complexity of separating a positive data set from a negative data set, which corresponds to the height of a decision tree separating them intuitively.

1.1 Background and Problems in Machine Learning

With the recent rapid development of machine learning techniques, more and more data sets are stored in databases and distributed through computer networks, where every continuous datum such as a real-valued datum obtained by scientific activity; *e.g.*, observations or experiments in the real world, is *discretized* from analog to digital form. Thereby there is an unavoidable gap between raw continuous data and converted discrete data used for learning and mining on a computer. This means that whereas lots of machine learning methods assume continuous data (usually vectors of real numbers) ideally, actual applications on computers use discretized binary data.

However, to the best of our knowledge, existing theoretical bases for machine learning do not try to bridge the gap. For example, methods originated from the perceptron are based on the idea of regulating analog wiring (Rosenblatt, 1958), hence they take no notice of discretization. Furthermore, despite now there are several discretization techniques (Elomaa and Rousu, 2003; Fayyad and Irani, 1993; Friedman et al., 1998; Gama and Pinto, 2006; Kontkanen et al., 1997; Lin et al., 2003; Liu et al., 2002; Skubacz and Hollmén, 2009), they treat discretization as just the data preprocessing for improving accuracy or efficiency of the machine learning/data mining procedure, and the process discretization itself is not considered from computational point of view. Thus, roughly speaking, we have to build an "analog-to-digital (A/D) converter" into a machine learning method computationally to treat discretization appropriately.

1.2 Background and Problems in Experimental Science

In experimental science, a lot of experiments are designed including a *negative control* and a *positive control*. Positive controls confirm that the procedure is competent for observing the effect, and negative controls confirm that the procedure is not observing an unrelated effect. Testing the effect of a treatment group is the objective of such an experiment. A typical situation is testing the effect of a new drug, where the result of applying the new drug (treatment group) is compared against placebo (negative control).

The standard method for the above task is a *statistical hypothesis test*, which has developed from the works by Neyman and Pearson (Neyman and Pearson, 1928, 1933), and Fisher (Fisher, 1925, 1956). However, it is well known that there exist many fundamental problems in statistical hypothesis testing such as non-verifiable assumptions for populations and arbitrary p values (Johnson, 1999) when we apply such methods to actual data sets. As a result, even in the top journals such as *Nature*, *Science*, and *Cell*, we can easily find inappropriate usage of statistical hypothesis testing[1]. Thus an alternative method to give theoretical justification to experimental results is required.

All the scientists have to do is to judge which controls a treatment group belongs to and, obviously, this typical task in experimental science can be viewed as *classification* in machine learning, that is, a pair of negative and positive controls is given as a training data set, and a classifier labels a treatment group (corresponds to a test set) "negative" or "positive" using the training set. Note that labels of all elements in the test set are same. Therefore, comparing the similarity between the

1. *Nature* says "please help us" to deal with various statistical methods appropriately (`http://www.nature.com/nature/authors/gta/`).

negative control and the test set to the one between the positive control and the test set is a direct way, and the coding divergence, which will be introduced in this paper, can achieve such task.

Note that lots of existing classification methods based on statistics could not work well, since most of them have been proposed for *large* data sets, while typical data sets are *small* in experimental science such as physiology.

1.3 Solutions and Examples of Our Method

In this paper we focus on machine learning from real-valued data, and measure the difference between two sets of real-valued data by the complexity of separating the two sets. We call this novel measure the *coding divergence*. This divergence uses no statistics and no probabilistic models, and purely depends on the topological structure (in the sense of mathematics) of the *Cantor space* Σ^ω, which is known as the standard topological space of the set of infinite sequences (Weihrauch, 2000). Thus, for example, it has no assumptions for the probability distributions of data sets. This property enables us to develop a robust machine learning algorithm for various data sets.

We identify each real-valued datum with a vector of real numbers in the d-dimensional Euclidean space $\mathbb{R}^d$, and realize discretization of a real-valued datum through an topological mapping from $\mathbb{R}^d$ into the Cantor space Σ^ω. A topology uses *open sets* to axiomatize the notion of approximation, and it enables us to treat each discretized datum as a base element of an open set of the Cantor space.

The simplest model that is consistent with the given two sets is learned in the Cantor space with assuming that one set is a set of positive examples and the other set is a set of negative examples; *i.e.*, the learned model explains all positive examples and does not explain any negative examples. This model corresponds to the *minimum open set* in the Cantor space Σ^ω. The coding divergence is obtained from the length of the code that encodes the learned model in the Cantor space.

Figure 1 shows two examples of computing the *binary-coding divergence* (the coding divergence with the binary encoding) where each datum is a vector in $\mathbb{R}^2$. The unit cube $\mathcal{I} = [0,1] \times [0,1]$ is partitioned recursively, and each partitioned cube is encoded by the binary encoding (see Section 3.2 and Figure 3). In the left panel (Figure 1(a)), to separate one set from the other set, we need only one partition in each dimension, and need $2/10$ symbols par datum. In contrast in the right panel (Figure 1(b)), three recursive partitions are needed in each dimension, and we need $26/10$ symbols par datum. The binary-coding divergence is determined directly from these numbers of symbols, that is, $2/10 + 2/10 = 0.4$ in Figure 1(a) and $26/10 + 26/10 = 5.2$ in Figure 1(b).

1.4 Related Works

Liu *et al*. (Liu et al., 2008) used the similar idea of our divergence to anomaly detection. They constructed decision trees by partitioning intervals randomly and recursively, and used the height of them to measure the complexity of separating each datum. Comparing to the method, our contribution is as follows: We treat the way of partition as the process of discretization and formulate it as a mapping (encoding) of $\mathbb{R}^d$ into the Cantor space Σ^ω, and show that the height of a decision tree corresponds to the size of an open set in the topological space Σ^ω. This gives the theoretical justification for the process "partition". Furthermore, this is the first time to perform classification using such idea.

Discretization is also used to construct decision trees from continuous data. For example, C4.5 (Quinlan, 1996) is one of famous algorithms to develop decision trees together with discretization.

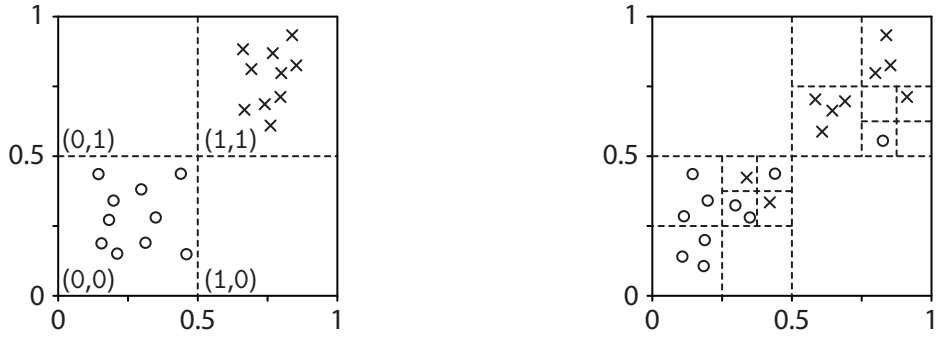

Figure 1: Two examples of computing the binary-coding divergence. Each $\circ$ and $\times$ denotes a real-valued datum, and each cube is encoded by the binary encoding (same as the base-2 embedding in Definition 5). In the left panel (a), we need a pair of codes $(0, 0)$ (encoding $[0, 1/2] \times [0, 1/2]$) to separate $\circ$ from $\times$, and $(1, 1)$ (encoding $[1/2, 1] \times [1/2, 1]$) to separate $\times$ from $\circ$, hence we can separate them with $2/10$ symbols par datum, and the binary-coding divergence is $2/10 + 2/10 = 0.4$ (For each pair of codes, we ignore symbols "(", ",", and ")", and actually wrap into one code. See the equation (1)). In contrast in the right panel (b), we need codes $(00, 00)$, $(00, 01)$, $(010, 010)$, $(011, 011)$, and $(110, 100)$ to separate $\circ$ from $\times$, and $(10, 10)$, $(11, 11)$, $(111, 101)$, $(010, 011)$, and $(011, 010)$ to separate $\times$ from $\circ$. Thus they are separated by $(4 + 4 + 6 + 6 + 6)/10$ symbols par datum, and the divergence is $26/10 + 26/10 = 5.2$.

Our approach is different from them since we realize discretization process by methods of encoding real numbers, give theoretical support by modern mathematical theories in Computable Analysis, and integrate the process into computational learning.

Kernel methods including Support Vector Machines (SVMs) are known to be one of the most famous and powerful classification methods (Bishop, 2007), where any form of data sets, for example sequences, images, and graphs, are mapped into a high dimensional feature space, which is the d-dimensional Euclidean space $\mathbb{R}^d$, or more generally the infinite-dimensional Hilbert space, to measure the similarity between each pair of data. In contrast, our strategy is inverted: Every real-valued datum in $\mathbb{R}^d$ is mapped into the Cantor space Σ^ω. The "discrete" space Σ^ω might seem to be strange as a feature space, but is natural for machine learning from real-valued data with discretization.

Some studies use the Kolmogorov complexity for machine learning such as clustering (Cilibrasi and Vitányi, 2005; Li et al., 2003) by measuring the similarity between a pair of data. Comparing to the methods, our approach has mainly two advantages as follows: First, the coding divergence is

computable (Section 2.3) whereas the Kolmogorov complexity is not, hence they use actual compression algorithms such as `gzip` to obtain an approximate value of the Kolmogorov complexity, which result in a gap between theory and practice; second, our approach is more general than their ones, because we can treat any continuous objects through encoding of them, however they cannot since they do not consider encoding process of objects.

1.5 Organization of This Paper

This paper is organized as follows: Section 2 is the main part in this paper, where we introduce the coding divergence and a classifier using the divergence. We give mathematical background in Section 3, and analyze performance of our classifier experimentally in Section 4. Section 5 gives conclusion.

1.6 Notation

We assume that the reader is familiar with the basic concepts of analysis and the ordinary computability theory (Dieudonné, 1960; Hopcroft and Ullman, 1979). In the following $\mathbb{R}$ denotes the set of real numbers, and $\mathbb{R}^d$ denotes the d-dimensional Euclidean space. The unit interval $[0,1] \times \cdots \times [0,1]$ in $\mathbb{R}^d$ is denoted by $\mathcal{I}$. The set of all finite sequences and infinite sequences over an alphabet Σ is denoted by Σ^* and Σ^ω, respectively. The *length* of a finite sequence w is the number of positions for symbols in w, denoted by $|w|$, and the *empty string* is the string whose length is 0, denoted by λ. For a set of sequences W, the *size* of W is defined by $|W| := \sum_{w \in W} |w|$. For example, if $W = \{\mathtt{11}, \mathtt{0}, \mathtt{100}\}$, then $|W| = 2 + 1 + 3 = 6$.

For $v \in \Sigma^*$ and $p \in \Sigma^* \cup \Sigma^\omega$, if $p = vq$ for some $q \in \Sigma^* \cup \Sigma^\omega$, then v is a *prefix* of p, denoted by $v \sqsubseteq p$. The ith symbol of a sequence p is denoted by $p(i)$, and the prefix $p(0) \dots p(n)$ of the length $n+1$ of p is denoted by $p[n]$. The set $\{p \in \Sigma^\omega \mid w \sqsubseteq p\}$ is denoted by $w\Sigma^\omega$, and the set $\{p \in \Sigma^\omega \mid w \sqsubseteq p \text{ for some } w \in W\}$ by $W\Sigma^\omega$.

2. The Coding Divergence

In this section we give a novel measure of the difference between sets in the d-dimensional Euclidean space $\mathbb{R}^d$, called the *coding divergence*, with assuming that an element in $\mathbb{R}^d$ corresponds to a real-valued datum. This divergence is the main contribution in this paper. We design a classifier using it in 2.2, and construct a learner that learns the divergence in 2.3.

2.1 Definition and Properties

Let X and Y be a pair of nonempty finite sets in the unit interval $\mathcal{I} \subset \mathbb{R}^d$. We encode them by a mapping γ from $\mathcal{I}$ to the Cantor space Σ^ω, where γ is called an *embedding*, and its formal definition will be given in Section 3.2. Thus $\gamma(X)$ and $\gamma(Y)$ are sets of infinite sequences.

We set a *model* of given data as an *open set* in the Cantor space, since this property "open" is desirable for machine learning as follows: For a set P, if P is open, then there exists a set of finite sequences W such that $W\Sigma^\omega = P$ (remember that $W\Sigma^\omega = \{p \in \Sigma^\omega \mid w \sqsubseteq p \text{ for some } w \in W\}$). This means that P is *finitely observable* (Smyth, 1992), that is, for any infinite sequence p, we can decide whether or not $p \in P$ by observing just some prefix of p. Thus from sets of infinite sequences $\gamma(X)$ and $\gamma(Y)$, we can obtain an open set as a model of them *in finite time* (see Section 3.1 for its formal definition).

We say that a set of infinite sequences $P \subseteq \Sigma^\omega$ is *consistent* with an ordered pair $(\gamma(X), \gamma(Y))$ if $P \supseteq \gamma(X)$ and $P \cap \gamma(Y) = \emptyset$, hence if we see $\gamma(X)$ and $\gamma(Y)$ as positive and negative examples, respectively, then the set P "explains" all elements in $\gamma(X)$ and does not explain any elements in $\gamma(Y)$.

Definition 1 Given an embedding $\gamma : \mathcal{I} \to \Sigma^\omega$. For a pair of nonempty finite sets $X, Y \subset \mathcal{I}$, define the *coding divergence*[1] *with respect to* γ by

$$C_\gamma(X, Y) := \begin{cases} \infty & \text{if } X \cap Y \neq \emptyset, \\ D_\gamma(X; Y) + D_\gamma(Y; X) & \text{otherwise,} \end{cases}$$

where D_γ is the *directed coding divergence with respect to* γ defined by

$$D_\gamma(X; Y) := \frac{1}{\|X\|} \min\{\, |O| \mid O \text{ is open, and consistent with } (\gamma(X), \gamma(Y)) \,\}$$

($\|X\|$ is the cardinality of X).

The standard embedding for the coding divergence is the base-β embedding γ_β that will be given in Definition 5 and, especially, the coding divergence with respect to γ_β is written by $C_\beta(X, Y)$. If $\beta = 2$, then we call $C_2(X, Y)$ the *binary-coding divergence*. Informally, the base-β embedding encodes each real number by dividing each interval into β same width intervals recursively (see Figure 3).

Intuitively, the procedure to obtain the directed coding divergence is as follows: We encode given sets X, Y into the Cantor space Σ^ω by an embedding γ, find the simplest model (minimum open set) O that is consistent with an ordered pair of sets of infinite sequences $(\gamma(X), \gamma(Y))$, and measure the size $|O|$. Thus the directed coding divergence $D_\gamma(X; Y)$ can be viewed as the result of *consistent learning* from positive examples $\gamma(X)$ and negative examples $\gamma(Y)$ in Computational Learning Theory context (Jain et al., 1999).

Example 1 Suppose that $X = \{\sqrt{0.2}, 0.8\}$ and $Y = \{\pi/10\}$. Then in the binary embedding γ_2 (Definition 5), these sets are encoded into infinite sequences as follows: $\gamma_2(X) = \{\mathtt{011}\ldots, \mathtt{111}\ldots\}$ and $\gamma_2(Y) = \{\mathtt{010}\ldots\}$. Thus for an open set $V\Sigma^\omega$ with $V = \{\mathtt{011}, \mathtt{1}\}$, $V\Sigma^\omega$ is minimum and consistent with $(\gamma_2(X), \gamma_2(Y))$. Therefore we have $D_2(X; Y) = (3 + 1)/2 = 2$. Similarly, $W\Sigma^\omega$ with $W = \{\mathtt{010}\}$ is minimum and consistent with $(\gamma_2(Y), \gamma_2(X))$, hence $D_2(Y; X) = 3/1 = 3$. Consequently, $C_2(X, Y) = D_2(X; Y) + D_2(Y; X) = 5$.

It is trivial that the coding divergence is not a metric since $C_\gamma(X, Y) \neq 0$ for all nonempty finite sets $X, Y \subset \mathcal{I}$.

Lemma 2 *The coding divergence satisfies the following conditions:*

1. $C_\gamma(X, Y) > 0$.
2. $C_\gamma(X, Y) = C_\gamma(Y, X)$.

Furthermore, it does not satisfy the triangle inequality.

1. The usage of the word "divergence" follows the original definition of the Kullback-Leibler divergence (Kullback and Leibler, 1951).

Proof The conditions $C_\gamma(X,Y) > 0$ and $C_\gamma(X,Y) = C_\gamma(Y,X)$ are proved directly from the definition. We can easily find an example, where the triangle inequality does not hold. For example, let $X = \{0.1\}$, $Y = \{0.8\}$, and $Z = \{0.2\}$, and assume that we use the binary embedding. We have $C_2(X,Y) = 2$, $C_2(Y,Z) = 2$, and $C_2(Z,X) = 6$, thus $C_2(X,Y) + C_2(Y,Z) < C_2(Z,X)$. ■

2.2 Classification Using the Coding Divergence

We construct a lazy learner that classifies a given data set using the coding divergence. It classifies a test set by measuring its similarity (coding divergence) to a training data set. It is quite simple and has no parameters, hence we can apply it to any type of real-valued data sets without any consideration. We will show its robustness in Section 4.

Assume that there are two classes A and B, and a pair (X,Y) is given as a training data set (X,Y are nonempty finite sets in $\mathbb{R}^d$), where X (resp. Y) belongs to A (resp. B). Moreover, suppose that we have a data set $Z \subset \mathbb{R}^d$ in which every datum is unlabeled, and we just know that all labels of elements in Z are same.

The classifier performs as follows: First, it computes $C_\gamma(X,Z)$ and $C_\gamma(Y,Z)$, and next, judges

$$Z \text{ belongs to the class } \begin{cases} A & \text{if } C_\gamma(X,Z) > C_\gamma(Y,Z), \\ B & \text{otherwise.} \end{cases}$$

Generally, the computability of $C_\gamma(X,Z)$ and $C_\gamma(Y,Z)$ is not guaranteed. However, if we use the base-β embedding γ_β such as the binary embedding (Definition 5), then it is effectively computable. We give the learner ψ that learns the coding divergence with respect to γ_β in the next subsection.

2.3 Learning of the Coding Divergence w.r.t. γ_β

Here we integrate discretization of encoded real-valued data (*i.e.*, infinite sequences) and learning of the coding divergence with respect to γ_β, and construct a learner that learns the divergence $C_\beta(X,Y)$ from $\gamma_\beta(X)$ and $\gamma_\beta(Y)$.

Procedure 1 shows the learner ψ that learns the coding divergence $C_\beta(X,Y)$ from a pair of given sets of encoded infinite sequences $\gamma_\beta(X)$ and $\gamma_\beta(Y)$. It continues to discretize each sequence, obtain longer and longer prefixes, and generate an approximate value of the $C_\beta(X,Y)$. For inputs $\gamma_\beta(X)$ and $\gamma_\beta(Y)$, the output of ψ (finite or infinite sequence) is denoted by $\psi(\gamma_\beta(X),\gamma_\beta(Y))$, hence for example, $\psi(\gamma_\beta(X),\gamma_\beta(Y))(0)$ is the first output, $\psi(\gamma_\beta(X),\gamma_\beta(Y))(1)$ is the second output, and so on.

To learn the coding divergence, a learner has to judge whether or not an open set O is consistent with given sets $(\gamma_\beta(X),\gamma_\beta(Y))$, and we show that it is decidable in finite time.

Lemma 3 *For every open set O and every $P,Q \subseteq \Sigma^\omega$, it is decidable that whether or not O is consistent with (P,Q) in finite time.*

Proof Let $O = W\Sigma^\omega$ and $k = \max_{w\in W}|w|$, and define $P[k] := \{p[k] \mid p \in P\}$ and $Q[k] := \{q[k] \mid q \in Q\}$. We show that we can check the consistency only using W, $P[k]$, and $Q[k]$ (thus it is decidable). The condition $O \supseteq P$ holds if and only if for all $x \in P[k]$, there exists $w \in W$ such that $w \sqsubseteq x$. Moreover, $O \cap Q = \emptyset$ if and only if $y \not\sqsubseteq w$ and $w \not\sqsubseteq y$ for all $w \in O$ and $y \in Q[k]$. ■

Procedure 1 The learner ψ that learns $C_\beta(X, Y)$

Input: a pair $(\gamma_\beta(X), \gamma_\beta(Y))$ (both X and Y are nonempty finite sets in $\mathcal{I}$)
Output: a finite or infinite sequence converging to $C_\beta(X, Y)$

function MAIN($\gamma_\beta(X)$, $\gamma_\beta(Y)$)
1: LEARNING($\gamma_\beta(X)$, $\gamma_\beta(Y)$, $\emptyset$, $\emptyset$, $\|X\|$, $\|Y\|$, 0)

function LEARNING(P, Q, H_1, H_2, m, n, k)
1: $V \leftarrow$ DISCRETIZE(P, k)
2: $W \leftarrow$ DISCRETIZE(Q, k)
3: $H_1 \leftarrow H_1 \cup \{v \in V \mid v \notin W\}$
4: $H_2 \leftarrow H_2 \cup \{w \in W \mid w \notin V\}$
5: $P \leftarrow \{p \in P \mid p \notin H_1\Sigma^\omega\}$
6: $Q \leftarrow \{q \in Q \mid q \notin H_2\Sigma^\omega\}$
7: output $m^{-1}\sum_{v\in H_1}|v| + n^{-1}\sum_{w\in H_2}|w|$
8: **if** $P = \emptyset$ and $Q = \emptyset$ **then** halt
9: **else return** LEARNING(P, Q, H_1, H_2, m, n, $k+1$)

function DISCRETIZE(P, k)
1: **return** $\{p[n] \mid p \in P\}$, where $n = (k+1)d - 1$

If $X \cap Y = \emptyset$, then ψ halts in finite time, and the last output is exactly the same as $C_\beta(X, Y)$. Otherwise if $X \cap Y \neq \emptyset$, then ψ continues to output approximate values of the $C_\beta(X, Y) = \infty$ forever. Here, we can easily prove that for all $i, j \in \mathbb{N}$ with $i < j$,

$$\psi(\gamma_\beta(X), \gamma_\beta(Y))(i) \leqslant \psi(\gamma_\beta(X), \gamma_\beta(Y))(j), \text{ and } \lim_{i\to\infty} \psi(\gamma_\beta(X), \gamma_\beta(Y))(i) = \infty.$$

This means that we can obtain more and more accurate values of the coding divergence, even though we cannot obtain the exact value in finite time. This property corresponds to the *effective* computability realized by the *Type-2 machine*, where while a computer reads more and more precise information (longer and longer prefixes) of the input, it produces more and more accurate approximations of the result. This machine is a natural extension of the usual Turing machine, and used to introduce computability of continuous objects (Weihrauch, 2000). Furthermore, this property of computation is an effective version of the *consistency* in statistical context.

Example 2 Let us consider the case in Figure 1, and assume that X and Y consist of all $\circ$ and $\times$, respectively. Let $P = \gamma_2(X)$ and $Q = \gamma_2(Y)$. Every pair of codes is wrapped into one code by the wrapping function φ that will be defined in (1). In Figure 1(a), DISCRETIZE$(P, 0)$ returns $\{\mathtt{00}\}$ (corresponds to $(0, 0)$) and DISCRETIZE$(Q, 0)$ returns $\{\mathtt{11}\}$ (corresponds to $(1, 1)$). Thus ψ outputs $2/10 + 2/10 = 0.4$ and halts. In Figure 1(b), both DISCRETIZE$(P, 0)$ and DISCRETIZE$(Q, 0)$ return $\{\mathtt{00}, \mathtt{11}\}$, and DISCRETIZE$(P, 1)$ returns $\{\mathtt{0000}, \mathtt{0001}, \mathtt{0011}, \mathtt{1110}\}$, and DISCRETIZE$(Q, 1)$ returns $\{\mathtt{1100}, \mathtt{1111}, \mathtt{0011}, \mathtt{1110}\}$. Thus $H_1 = \{\mathtt{0000}, \mathtt{0001}\}$ and $H_2 = \{\mathtt{1100}, \mathtt{1111}\}$. Furthermore, for the rest of data, DISCRETIZE performs similarly, and finite sequences `001100`, `001111`, `111000` and `111011`, `001110`, `001101` are added to H_1 and H_2, respectively. Thus ψ outputs $26/10 + 26/10 = 5.2$ and halts.

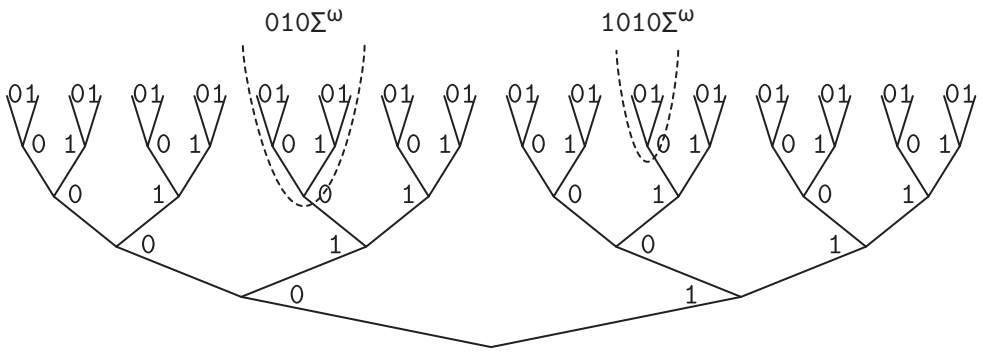

Figure 2: The tree representation of the Cantor space over $\Sigma = \{0, 1\}$. The subtrees $010\Sigma^\omega$ and $1010\Sigma^\omega$ are base elements, and $010\Sigma^\omega \cup 1010\Sigma^\omega$ is an open set.

3. Mathematical Background

In this section we introduce our mathematical background using the framework of Type-2 Theory of Effectivity (TTE) studied in the area of Computable Analysis (Weihrauch, 2000).

3.1 The Cantor Space

First, we introduce the *Cantor topology* into the set of infinite sequences Σ^ω, which is known as the standard topology on it (Weihrauch, 2000). Remember that $w\Sigma^\omega = \{p \in \Sigma^\omega \mid w \sqsubseteq p\}$ and $W\Sigma^\omega = \{p \in \Sigma^\omega \mid w \sqsubseteq p \text{ for some } w \in W\}$.

Definition 4 Define $\tau_{\Sigma^\omega} := \{W\Sigma^\omega \mid W \subseteq \Sigma^*\}$. We say that τ_{Σ^ω} is the *Cantor topology* over Σ, and the topological space $(\Sigma^\omega, \tau_{\Sigma^\omega})$ is the *Cantor space*.

We abbreviate $(\Sigma^\omega, \tau_{\Sigma^\omega})$ as Σ^ω if τ_{Σ^ω} is understood from context. The set $\{w\Sigma^\omega \mid w \in \Sigma^*\}$ becomes a base of the topology τ_{Σ^ω}.

Example 3 Let an alphabet $\Sigma = \{0, 1\}$. For example, the set $00\Sigma^\omega = \{p \in \Sigma^\omega \mid 00 \sqsubseteq p\}$ is a base element of the Cantor space Σ^ω, and $10\Sigma^\omega$ is also a base element. Of course, both sets $00\Sigma^\omega$ and $10\Sigma^\omega$ are open sets in the space Σ^ω; *i.e.*, $00\Sigma^\omega \in \tau_{\Sigma^\omega}$ and $10\Sigma^\omega \in \tau_{\Sigma^\omega}$. Assume $W = \{00, 10\}$. Then the set $W\Sigma^\omega = \{p \in \Sigma^\omega \mid 00 \sqsubseteq p \text{ or } 10 \sqsubseteq p\}$ is an open set. Note that $W\Sigma^\omega = 00\Sigma^\omega \cup 10\Sigma^\omega$.

The Cantor space Σ^ω can be visualized by a tree, where each infinite sequence $p \in \Sigma^\omega$ corresponds to an infinite descending path from the root (Figure 2). Then a base element $w\Sigma^\omega$ is a full subtree and an open set is a union of full subtrees, and the length $|w|$ corresponds to the depth of the root of the subtree.

3.2 Embeddings of the Euclidean Space into the Cantor Space

Let us approach the d-dimensional Euclidean space $\mathbb{R}^d$ through the Cantor space Σ^ω using topological mappings between $\mathbb{R}^d$ and Σ^ω.

We say that a surjective function $\rho :\subseteq \Sigma^\omega \to \mathbb{R}^d$ is a *representation* of $\mathbb{R}^d$, and an injective function $\gamma :\subseteq \mathbb{R}^d \to \Sigma^\omega$ is an *embedding* of $\mathbb{R}^d$, meaning that representations and embeddings are dual concepts. In the field of Computable Analysis, computability of $\mathbb{R}^d$ or other continuous objects (*i.e.*, uncountable sets) are treated through representations with Σ^ω (Schröder, 2002; Weihrauch, 2000), and in this study we apply this theoretical framework to machine learning. For simplicity, we turn an embedding γ into a bijective function by replacing the codomain Σ^ω by its actual image $\gamma(\mathbb{R}^d)$, and assume that a representation $\rho = \gamma^{-1}$.

When we identify an element in $\mathbb{R}^d$ with an object with analog quantity, an embedding γ can be viewed as a mathematical realization of an encoding method or an actual A/D converter, where a continuous signal (analog quantity) is converted to a sequence of bits (digital quantity). The true value x of the analog quantity is encoded as an infinite sequence $\gamma(x)$ by the embedding γ, and any prefix w of $\gamma(x)$ is a discretized digital value. The prefix w tells us x is in the interval $\gamma^{-1}(w\Sigma^\omega)$, and the width of the interval

$$\left|\, \gamma^{-1}(w\Sigma^\omega) \,\right| = \sup \left\{\, d(x,y) \,\middle|\, x, y \in \gamma^{-1}(w\Sigma^\omega) \,\right\}$$

(d is the Euclidean metric) corresponds to an error of w. This modeling reflects the fundamental property of observation together with discretization: Every (discretized) real-valued datum must have some error intrinsically (Baird, 1994).

We now define the standard embedding, the base-β embedding.

Definition 5 Assume $d = 1$. The *base-β embedding* of the unit interval $\mathcal{I} = [0,1]$ is a mapping $\gamma_\beta : \mathcal{I} \to \Sigma^\omega$ that maps x to an infinite sequence p composed as follows: For all $i \in \mathbb{N}$, $x = 0$ implies $p(i) = \mathtt{0}$, and $x \neq 0$ implies

$$p(i) = \begin{cases} \mathtt{0} & \text{if } z < x \leqslant z + \beta^{-(i+1)}, \\ \mathtt{1} & \text{if } z + \beta^{-(i+1)} < x \leqslant z + \beta^{-(i+1)+1}, \\ \ldots & \\ \beta - 1 & \text{if } z + \beta^{-(i+1)+(\beta-2)} < x \leqslant z + \beta^{-(i+1)+(\beta-1)}, \end{cases}$$

where

$$z = \sum_{j=0}^{i-1} p(j)\beta^{-(j+1)}$$

if $i \neq 0$, and $z = 0$ otherwise.

Especially, we call the base-2 embedding the *binary embedding*. Figure 3 denotes the binary embedding γ_2, where each horizontal line means that the corresponding position is `1` on the line and `0` otherwise. For example, let $p = \gamma_2(0.3)$. Then $p(0) = \mathtt{0}$ since the position 0 is not on the line, and $p(1) = \mathtt{1}$ since the position 1 is on the line, and so on.

For infinite sequences $p_1, \ldots, p_d$, define the *wrapping function*

$$\varphi(p_1, \ldots, p_d) := p_1(0) \ldots p_d(0) p_1(1) \ldots p_d(1) p_1(2) \ldots p_d(2) \ldots . \tag{1}$$

With this function, define the *d-dimensional base-β embedding* γ_β^d from $\mathcal{I} \subset \mathbb{R}^d$ to Σ^ω by

$$\gamma_\beta^d(x_1, \ldots, x_d) := \varphi(\gamma_\beta(x_1), \ldots, \gamma_\beta(x_d)). \tag{2}$$

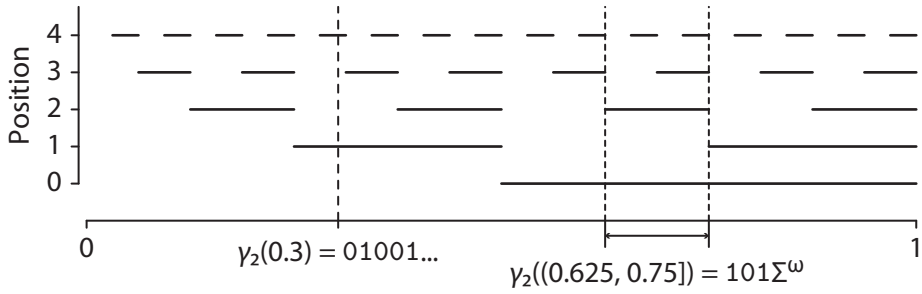

Figure 3: The (1-dimensional) binary embedding γ_2. The position i is 1 if it is on the line, and 0 otherwise.

Example 4 For the binary embedding γ_2, we have $\gamma_2(0.3) = 010\ldots$ and $\gamma_2(0.5) = 011\ldots$. Thus $\gamma_2^2(0.3, 0.5) = 001101\ldots$.

We abbreviate d of γ_β^d if it is understood from the context.

4. Empirical Experiments

We evaluate the performance of the classifier given in Section 2.2 by experiments to analyze the coding divergence empirically. We compared accuracy of classification performed by the classifier obtained from sensitivity and specificity to those of other classification methods.

Theoretically, our classifier can be applied to actual analog data such as (raw) real-valued data and continuous signals. However, it is difficult to collect such type of data sets and apply our classifier to them directly. Thereby in the following we use just discretized real-valued data stored in databases.

4.1 Materials and Methods

First, for empirical experiments we construct the learning algorithm **M** that computes an approximate value of the coding divergence with respect to the base-β embedding and always halts in finite time (Algorithm 1). The algorithm **M** does not receive infinite sequences, but receives just finite sequences.

Define $\gamma_\beta(X)[i] := \{w \mid |w| = i \text{ and } p \in w\Sigma^\omega \text{ for some } p \in \gamma_\beta(X)\}$. The algorithm **M** is slightly different from the learner ψ, since it receives only finite sequences $\gamma_\beta(X)[i]$ and $\gamma_\beta(Y)[j]$. We can get the exact coding divergence (*i.e.*, $\mathbf{M}(\gamma_\beta(X)[i], \gamma_\beta(Y)[j]) = C_\beta(X, Y)$) in the usual situation where $\gamma_\beta(X)[i] \cap \gamma_\beta(Y)[j] = \emptyset$ holds. Moreover, the output of **M** (denoted by $\mathbf{M}(\gamma_\beta(X)[i], \gamma_\beta(Y)[j])$) has the *monotonicity* with respect to i and j: For all pairs of i, j and i', j'

Algorithm 1 The learning algorithm **M** that learns $C_\beta(X, Y)$

Input: a pair $(\gamma_\beta(X)[i], \gamma_\beta(Y)[j])$ $(X, Y \subset \mathcal{I}$ and $i, j \in \mathbb{N})$
Output: an approximate value of $C_\beta(X, Y)$

function MAIN(V, W)
1: $(D_1, D_2) \leftarrow$ LEARNING(V, W, 0, 0, 0, $\min\{i, j\}$)
2: **return** $D_1 \,/\, \|V\| + D_2 \,/\, \|W\|$

function LEARNING(V, W, D_1, D_2, k, $k_{\max}$)
1: $V_{\text{dis}} \leftarrow$ DISCRETIZE(V, k)
2: $W_{\text{dis}} \leftarrow$ DISCRETIZE(W, k)
3: $V_{\text{sep}} \leftarrow \{v \in V_{\text{dis}} \mid v \notin W_{\text{dis}}\}$
4: $W_{\text{sep}} \leftarrow \{w \in W_{\text{dis}} \mid w \notin V_{\text{dis}}\}$
5: $D_1 \leftarrow D_1 + \sum_{v \in V_{\text{sep}}} |v|$
6: $D_2 \leftarrow D_2 + \sum_{w \in W_{\text{sep}}} |w|$
7: $V \leftarrow \{v \in V \mid v \notin V_{\text{sep}}\Sigma^\omega\}$
8: $W \leftarrow \{w \in W \mid w \notin W_{\text{sep}}\Sigma^\omega\}$
9: **if** $V = \emptyset$ and $W = \emptyset$ **then return** (D_1, D_2)
10: **else if** $k = k_{\max}$ **then return** $(D_1 + n\|V\|, D_2 + n\|W\|)$ $(n = (k+1)d - 1)$
11: **else return** LEARNING(P, Q, D_1, D_2, $k+1$, $k_{\max}$)

function DISCRETIZE(V, k) /* $V \subset \Sigma^*$ */
1: **return** $\{v[n] \mid v \in V\}$ $(n = (k+1)d - 1)$

with $i \leqslant i'$ and $j \leqslant j'$, we have

$$|\, C_\beta(X, Y) - \mathbf{M}(\gamma_\beta(X)[i], \gamma_\beta(Y)[j]) \,| \geqslant \left|\, C_\beta(X, Y) - \mathbf{M}(\gamma_\beta(X)[i'], \gamma_\beta(Y)[j']) \,\right|,$$

and $\lim_{i,j \to \infty} |\, C_\beta(X, Y) - \mathbf{M}(\gamma_\beta(X)[i], \gamma_\beta(Y)[j]) \,| = 0$. Thus, intuitively, if we obtain more and more accurate data (longer and longer sequences), then approximation of the coding divergence becomes better and better, meaning that **M** is an *effective* algorithm.

The computational complexity of **M** is $O(mn)$, where m and n are the cardinality of X and Y, respectively, since in each level k, updating V and W (lines 7 and 8 in Algorithm 1) takes $O(mn)$.

To treat data that are not in the unit interval $\mathcal{I}$, we use the so-called min-max normalization that maps a value x to x', where

$$x' = \frac{x - \min\{X \cup Y \cup Z\}}{\max\{X \cup Y \cup Z\} - \min\{X \cup Y \cup Z\}}.$$

The classifier with the above algorithm **M** was implemented by the R language version 2.10.1 (R Development Core Team, 2009). We tested the performance of it by evaluating accuracy, which is the standard error measure of classification (Han and Kamber, 2006).

We chose ten data sets from UCI Machine Learning Repository (Frank and Asuncion, 2010): `abalon`, `transfusion`, `sonar`, `glass`, `segmentation`, `ionosphere`, `madelon`, `magic`, `waveform`, and `yeast`. Every datum in each data set is real-valued type and belongs to one of two classes (if there are more than two classes, we picked up just two classes). We used the size of each data set 10

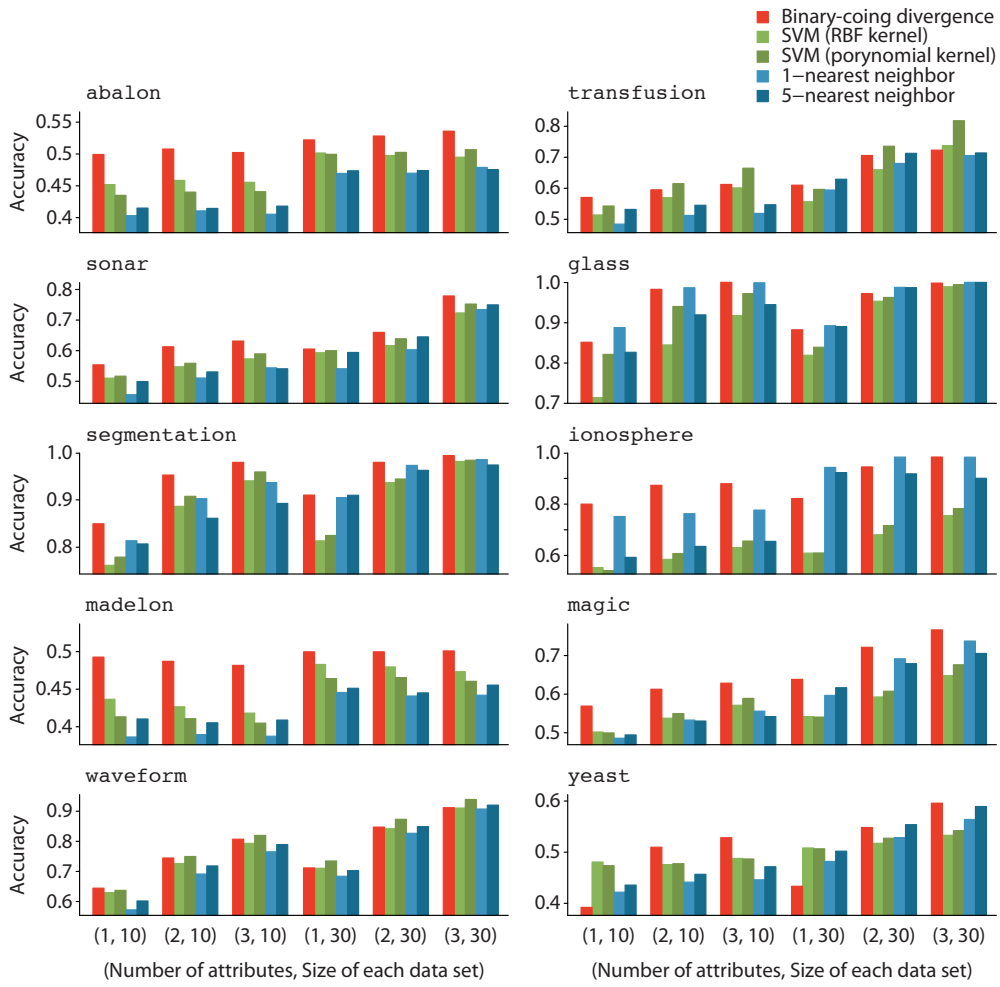

Figure 4: Experimental results of accuracy using real-valued data sets collected from UCI repository. We applied our classifier (using the binary-coding divergence) and other four classification methods: SVM with the RBF kernel, SVM with the polynomial kernel, k-nearest-neighbor classifiers ($k = 1, 5$), to ten real-valued data sets: `abalon`, `transfusion`, `sonar`, `glass`, `segmentation`, `ionosphere`, `madelon`, `magic`, `waveform`, and `yeast`. We examined six cases: The number of used attributes are 1, 2, and 3, and the size of each sampled training/test data sets are 10 and 30.

and 30, since one of main applications is controlled experiments in life science, where these small sizes are typical.

We repeated the following procedure 10,000 times: 1) Sample d attributes (d is 1, 2, or 3), 2) collect n data for a training set X and for a test set T_+ from one class by independent random sampling without replacement, and Y and T_- from the other class, where $n = 10$ or 30, and 3) using X and Y, classify test sets T_+ and T_- by our method and other classification methods. The binary-coding divergence was used through experiments.

Let t_{pos} be the number of true positives, that is, the number of the case in which T_+ is classified correctly, and t_{neg} be the number of true negatives. We calculated the accuracy by $(t_{\text{pos}} + t_{\text{neg}})/20000$, since the sensitivity is $t_{\text{pos}}/10000$ and the specificity is $t_{\text{neg}}/10000$. For reference, we used SVM with the RBF kernel, SVM with the polynomial kernel, and k-nearest neighbor classifiers ($k = 1$ and 5). We used the function `ksvm` in the "kernlab" package for SVM (Karatzoglou et al., 2004), and the function `knn` in the "class" package for the k-nearest neighbor classifiers, which have been implemented in R. Note that these methods classifies each element in a test set, thereby we classified T_+ (or T_-) to the class in which greater number of elements are classified.

4.2 Results and Discussions

The experimental result of accuracy is shown in Figure 4. Let us compare the result using the binary-coding divergence to those of other methods. In most cases, the accuracy of our classifier is the highest value, and only in some cases other methods are more accurate than our method (*e.g.*, low dimensional data sets in `yeast`). This result shows the robustness of our classifier for various data sets. The simple process "separation by encoding" of our theoretical background might cause this robustness.

Moreover, results of other methods highly depend on kinds of data sets. For example, 1- and 5-nearest neighbor methods are better than SVMs in `ionosphere`, but are worse in `abalon`. This means that these methods require adjustment of parameters depending on data sets. However, our method has no parameters and does not need any adjustment. Thus we can apply our method effectively without special background knowledge about data sets.

5. Conclusion

In this paper, we have proposed a novel measure of the difference between sets, called the coding divergence, and integrated discretization of analog data and learning of models that explains given data through computational learning of it, where the extracted model corresponds to the minimum open set in the Cantor space Σ^ω. The key idea is *separation* of intervals, which corresponds to realization of an encoding process of analog data by a mathematical embedding of the Euclidean space into the Cantor space. Furthermore, we have constructed a classifier, the lazy learner using the coding divergence, and shown the robust performance of it by empirical experiments.

Our mathematical framework is general and has possibility to develop further in the field of machine learning and data mining, since any type of data sets can be handled through an appropriate embedding from such objects to the Cantor space. This approach is new and, intuitively, opposite to the one using kernel functions.

Furthermore, since our proposed measure is quite simple, it can be applied to various tasks in machine learning and data mining. For example, we can measure the difference between clusters by the coding divergence. This means that we can perform hierarchical clustering using the coding divergence directly. Thus applying our measure to such tasks is a future work.

Another future work is application for other actual data sets. For example, image matching is an important topic in computer vision, where each image is assumed to be a set in $\mathbb{R}^3$. One of well-known distances is the Hausdorff metric, which is a distance between compact sets, and some metrics based on the Hausdorff metric have been proposed (Huttenlocher et al., 1993; Zhao et al., 2005). Trivially our method can be applied to such topics. In preliminary experiments, we have checked that the accuracy of classification using the coding divergence is better than that using the Hausdorff metric. Thus our method might be a better classifier than that with the Hausdorff metric.

Acknowledgments

We would like to thank Professor Takashi Washio for his helpful comments. This work was partly supported by Grant-in-Aid for Scientific Research from the Japan Society for the Promotion of Science.

References

D.C. Baird. *Experimentation: An Introduction to Measurement Theory and Experiment Design*. Benjamin Cummings, 3 edition, 1994.

C.M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2007.

R. Cilibrasi and P.M.B. Vitányi. Clustering by compression. *IEEE Transactions on Information theory*, 51(4):1523–1545, 2005.

J. Dieudonné. *Foundations of Modern Analysis*. Academic Press, 1960.

T. Elomaa and J. Rousu. Necessary and sufficient pre-processing in numerical range discretization. *Knowledge and Information Systems*, 5(2):162–182, 2003.

U.M. Fayyad and K.B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1022–1029, 1993.

R.A. Fisher. *Statistical Methods for Research Workers*. Oliver and Boyd, Edinburgh, 1925.

R.A. Fisher. *Statistical Methods and Scientific Inference*. Oliver and Boyd, Edinburgh, 1956.

A. Frank and A. Asuncion. UCI machine learning repository, 2010. URL `http://archive.ics.uci.edu/ml`.

N. Friedman, M. Goldszmidt, and T.J. Lee. Bayesian network classification with continuous attributes: Getting the best of both discretization and parametric fitting. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 179–187, 1998.

J. Gama and C. Pinto. Discretization from data streams: applications to histograms and data mining. In *Proceedings of the 2006 ACM symposium on Applied computing*, pages 23–27, 2006.

J. Han and M. Kamber. *Data Mining*. Morgan Kaufmann, 2 edition, 2006.

J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley Publishing Company, 1979.

D.P. Huttenlocher, G.A. Klanderman, and W.A. Rucklidge. Comparing images using the Hausdorff distance. *IEEE Transactions on pattern analysis and machine intelligence*, pages 850–863, 1993.

S. Jain, D. Osherson, J.S. Royer, and A. Sharma. *Systems That Learn*. The MIT Press, 2 edition, 1999.

D.H. Johnson. The insignificance of statistical significance testing. *The journal of wildlife management*, 63(3):763–772, 1999.

A. Karatzoglou, A. Smola, K. Hornik, and A. Zeileis. kernlab–an S4 package for kernel methods in R. *Journal of Statistical Software*, 11(9):1–20, 2004.

P. Kontkanen, P. Myllymaki, T. Silander, and H. Tirri. A bayesian approach to discretization. In *Proceedings of the European Symposium on Intelligent Techniques*, pages 265–268, 1997.

S. Kullback and R.A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

M. Li, X. Chen, X. Li, B. Ma, and P. Vitányi. The similarity metric. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 863–872. Society for Industrial and Applied Mathematics Philadelphia, PA, USA, 2003.

J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, page 11, 2003.

F.T. Liu, K.M. Ting, and Z.H. Zhou. Isolation forest. In *Eighth IEEE International Conference on Data Mining*, pages 413–422, 2008.

H. Liu, F. Hussain, C.L. Tan, and M. Dash. Discretization: An enabling technique. *Data Mining and Knowledge Discovery*, 6(4):393–423, 2002.

J. Neyman and E.S. Pearson. On the use and interpretation of certain test criteria for purposes of statistical inference: Part I. *Biometrika*, 20(1):175–240, 1928.

J. Neyman and E.S. Pearson. On the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 231:289–337, 1933.

J.R. Quinlan. Improved use of continuous attributes in C4.5. *Journal of Artificial Intelligence Research*, 4:77–90, 1996.

R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2009. URL `http://www.R-project.org`.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386–408, 1958.

M. Schröder. Extended admissibility. *Theoretical Computer Science*, 284(2):519–538, 2002.

M. Skubacz and J. Hollmén. Quantization of continuous input variables for binary classification. In *Intelligent Data Engineering and Automated Learning—IDEAL 2000. Data Mining, Financial Engineering, and Intelligent Agents*, pages 101–115. Springer, 2009.

M.B. Smyth. Topology. In S Abramsky, D.M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 1, pages 641–761. Oxford University Press, 1992.

K. Weihrauch. *Computable Analysis : An Introduction*. Springer, November 2000.

C. Zhao, W. Shi, and Y. Deng. A new hausdorff distance for image matching. *Pattern Recognition Letters*, 26(5):581–586, 2005.