Inductive Confidence Machine for Pattern Recognition: is it the next step towards AI?

David Surkov
Royal Holloway University of London
2004

Abstract

Since the dawn of the computer era everyone has been fascinated about the idea of creating an Artificial Intelligence. In the early days it seemed like it was around the corner. Nevertheless, every advance in computer science has been pushing the creation of AI further and further away. This is mainly attributed to the fact that we are still not entirely united in defining the meaning and the essence of intelligence itself let alone the Artificial Intelligence.

Meanwhile another branch of computer science called Machine Learning has evolved. Researchers in this field have been concentrating on developing computer algorithms to learn data and to produce output when presented with new data from the same knowledge domain. This field turned out to be more practical and therefore useful in the real world. However, little attention has been drawn into how the advances in Machine Learning can lead to the AI creation.

In this thesis we overview the Machine Learning field to understand its current role and where it fits within the AI development. We analyse problems currently experienced in this field and identify what needs to be introduced in order to advance further in the direction of AI creation.

A typical machine learning algorithm is designed to produce bare predictions. We show that the next major step in the Machine Learning is the development of algorithms that produce not only a prediction but also quality measures whereby the prediction can be assessed rather than unconditionally accepted. In other terms, we argue that the machine learning algorithms can potentially resemble intelligence if they could produce qualified predictions.

Following the idea of algorithmic intelligence we introduce and analyse a way of obtaining qualified predictions based on Algorithmic Information Theory or Algorithmic Theory of Randomness in particular. By creating so called Confidence Machine that is capable of producing measures of confidence and credibility for each prediction along with the prediction itself, and by demonstrating its capabilities practically, we make philosophical speculations about further possible developments of the Artificial Intelligence.

Contents

Notations

$\Re$	set of real numbers
N	set of natural numbers
z	set of objects
z_i	i_{th} set of objects
z^i	i_{th} object in the set z
$\geq^+$	greater or equal up to an additive constant
$\leq^+$	less or equal up to an additive constant
$=^+$	equal up to an additive constant
α	individual strangeness measure
C	plain Kolmogorov complexity
K	prefix Kolmogorov complexity
KD	information distance
$d_A^p(x)$	p-typicalness deficiency of an element x of a set A
$\lambda_A^p(x)$	p-typicalness level of an element x of a set A
U	universal Turing machine
K	kernel function for Support Vector Machine
log	logarithm base 2
ln	natural logarithm (base e)
$p \subseteq q$	sequence p is a prefix of q

List of Figures

Preface

Since the dawn of the computer era everyone has been fascinated about the idea of creating an Artificial Intelligence. In the early days it seemed like it was around the corner. Nevertheless, every advance in computer science has been pushing the creation of AI further and further away. This is mainly attributed to the fact that we are still not entirely united in defining the meaning and the essence of intelligence itself let alone the Artificial Intelligence.

Meanwhile another branch of computer science called Machine Learning has evolved. Researchers in this field have been concentrating on developing computer algorithms to learn data and to produce output when presented with new data from the same knowledge domain. This field turned out to be more practical and therefore useful in the real world. However, little attention has been drawn into how the advances in Machine Learning can lead to the AI creation.

In this thesis we overview the Machine Learning field to understand its current role and where it fits within the AI development. We analyse problems currently experienced in this field and identify what needs to be introduced in order to advance further in the direction of AI creation.

A typical machine learning algorithm is designed to produce bare predictions. We show that the next major step in the Machine Learning is the development of algorithms that produce not only a prediction but also quality measures whereby the prediction can be assessed rather than unconditionally accepted. In other terms, we argue that the machine learning algorithms can potentially resemble intelligence if they could produce qualified predictions.

Following the idea of algorithmic intelligence we introduce and analyse a way of obtaining qualified predictions based on Algorithmic Information Theory or Algorithmic Theory of Randomness in particular. We start from explaining the Algorithmic Theory of Randomness and then move into the details of Kolmogorov complexity and randomness deficiency. Then we create so called Confidence Machine that is capable of producing measures of confidence and credibility for each prediction along with the prediction itself.

A transductive approach to Confidence Machine creation is described first. We demonstrate how such machine can be constructed theoretically using an ideal

approach employing randomness deficiency, followed by the way it can be practically constructed using typicalness approximation to randomness deficiency. Then we propose how the typicalness approximation can be performed using well known bare prediction algorithms. We also present invalid and valid approaches to constructing Transductive Confidence Machines for binary and multi-class cases.

While the Transductive Confidence Machine has been shown to be the universal confidence machine, it is very slow and cannot be used efficiently on real-world problems. Therefore we then move into describing the main feature of this thesis – Inductive Confidence Machine. We show how the same principles behind the Transductive Confidence Machine along with the inductive approach can be used to produce a valid Confidence Machine that is able to output real time predictions provided an initial training is completed.

After presenting algorithms for Transductive and Inductive confidence machines, we cover how the Confidence Machine system is actually designed and practically implemented with requirements to be unified and expandable.

Next we investigate the scope of applications for confidence machines, how and where they might be practically employed and for what purposes. Following that we document a multitude of experiments performed using TCM and ICM, compare their performances on various data sets and various environments, and make conclusions about the comparable ICM performance and about the confidence machines in general.

Finally we analyse all the results presented in this thesis and conclude where the Inductive Confidence Machine stands within the Machine Learning field. We identify its drawbacks and possible improvements followed by its potential in general purpose learning tasks and in making a further step towards the Artificial Intelligence.

Chapter 1

1. Introduction

This thesis is devoted to the study of a new idea in the Machine Learning field – Qualified Predictions in Real Time. Qualified predictions in real time means we are able not only to make a prediction but also able to produce a well-calibrated degree of accuracy for each of our predictions within a reasonable amount of time.

In this section we will give an introduction to the Machine Learning as a general subject, the problems currently experienced in this field and what improvements we would like to introduce in terms of giving measures of confidence to our predictions hence making qualified predictions rather than bare ones as done conventionally.

1.1 Artificial Intelligence

An idea of creating an Artificial Intelligence and fast developments in computer technology are probably the most likely reasons for the birth of the Machine Learning field. Therefore it is sensible to start this thesis with an introduction to AI to understand the scope of this idea and what role the Machine Learning plays in it.

Although there is no exact definition of intelligence, making Artificial Intelligence even more undefined, AI can be abstractly described as the attempt to build objects that think and act like humans, that are able to learn and to use their knowledge to solve problems on their own.

Likewise, according to Webopedia [15], *Artificial Intelligence is the branch of computer science concerned with making computers behave like humans*. The term was coined in 1956 by John McCarthy at the Massachusetts Institute of Technology.

Artificial Intelligence includes:

- **games playing:** programming computers to play games such as chess and checkers
- **expert systems:** programming computers to make decisions in real-life situations (for example, some expert systems help doctors diagnose diseases based on symptoms)
- **natural language:** programming computers to understand natural human languages
- **neural networks:** systems that simulate intelligence by attempting to reproduce the types of physical connections that occur in animal brains
- **robotics:** programming computers to see and hear and react to other sensory stimulators.

Presently, any attempt to define AI leads to a definition which is relative to and based on human behaviour. This is simply because humans are the only known creatures that are considered to be intelligent, so until the AI is actually built and understood on its own the only way to identify it would be to compare it to the human intelligence in a certain way. Naturally the well known test for intelligence that has been with us for more than fifty years, is the Turing Test [84], which can presumably tell whether a given interactive subject is intelligent or not. If this subject passes the Turing Test and it is not of human origin then it is definitely an example of Artificial Intelligence. The Turing Test was introduced in 1950 by English mathematician Alan Turing. The test is simple: a human interrogator is isolated and given the task of distinguishing between a human and an artificial subject based on their replies to questions that the interrogator poses. After a series of tests are performed, the interrogator attempts to determine which subject is human and which is not. The interactive subject's success at thinking can be quantified by its probability of being misidentified as the human subject.

Even though the Turing Test was introduced more than fifty years ago it is still considered by many people as the most appropriate test for intelligence. Nevertheless many people also disagree with this statement for various reasons. For example M. Humphrys [14] asks and answers the following set of questions about the Turing Test that lead to an interesting conclusion:

"*Could an artificial intelligence (or indeed any non-human intelligence) convincingly pass for a human?* No, of course not. Even if you are not allowed to see it, you can ask it where it is from, who its mother was, where it went to school. Talk about events in your childhood. Where did you live? Oh, did you know such and such? Did you ever go to this place? No one can lie forever. If the machine is not allowed to talk about its *real* life history, you'll soon find the cracks. And if you are forced to only talk about abstract topics, the stilted conversation will prove little. The goal of AI should not be to pass the Turing Test!

But how will we be able to tell it's intelligent then? When aliens discover us, how will they be able to tell we're intelligent? We won't be able to pass as convincing aliens.

But, says Turing, *You only think I'm intelligent because of my behaviour*. No, I don't. I know you're intelligent without even meeting you or hearing a word you say. I know you're intelligent because *I'm related to you*."

Other people like Dr David Fogel [61] even claim that the Turing Test is a very dangerous definition that practically halted the development of the Artificial Intelligence because it was misquoted and misinterpreted almost from day one. The Turing Test has a computer pretend to be human, and this paradigm became a signpost saying that the road to Artificial Intelligence is through mimicry of human behaviour. According to Fogel, however, that path leads only to an illusion of intelligence – for example the kind of wooden intelligence exhibited by Deep Blue. In his book "Blondie 24: Playing At The Edge of AI" Fogel states that the popularity of the Turing Test has been leading researchers in the wrong direction which will not bring us any closer to the AI creation. He says that the only way to build an intelligence is the evolutionary way. Therefore the new definition of intelligence is proposed.

Intelligence is the capacity of a decision-making system to adapt its behaviour to meet goals in a range of environments.

David Fogel dramatically demonstrates how evolutionary computation may in fact bring us to our goal of creating a thinking machine far more quickly than traditional artificial intelligence has been able to do. This claim is supported by his achievement in creating a checkers playing program which has no prior knowledge about any checkers game played by other players. Instead it can teach itself by playing against itself or other players. The program is based on

the Neural Networks learning algorithm. Everything starts from a population of neural networks each with a random selection of weights. Each neural network plays at least five games, and the neural network with the best scores lives on for the next stage, while the poorer performers are removed from the population. After each stage a variation is carried out by a program which duplicates the successful neural networks from the previous stage and applies some constrained random variation to their weights. Then the process is repeated continuously. The hope is that after many generations of variation and selection the best neural network will be an expert at checkers.

The program that emulated the basic Darwinian principles of evolution was named "Blondie 24" probably to attract players over the internet. After some time of playing with itself and other players the program managed to create its own system for deciding which moves to make. Even though it was running on Pentium II 400 (extremely slow machine compared to projects like Deep Blue designed to play chess) in a matter of months it managed to beat 99% of all human players. Fogel's example shows that an artificial intelligence can probably be created using an evolutionary process, the fact not denied even by Alan Turing who also at the end suggested to create a child brain first and then teach it to become an adult [84]. Still it only shows a possible way of achieving it – it does not give a definition which would allow us to identify the AI when encountered if we do not know its origin. The definition of intelligence proposed by Fogel seems to be slightly vague, it is not clear how to actually implement it as a practical test for intelligence.

Nevertheless, D. Dobrev [16] and others are less radical about the Turing Test idea stating that it is just too old and too abstract and propose a newer more up-to-date definition. The reason for considering the Turing Test non-adequate for identifying AI lies in the fact that Turing's definition suggests that an intellect is a person with knowledge gained through the years. If this is so then what about a newly born baby? Is it an intellect? The obvious answer is "yes" even though it probably will not pass the Turing Test. The more appropriate definition of an intellect should be something like: a thing that knows nothing but it can learn. To create a formal definition of AI, Dobrev [16] assumes that it is an object living in a kind of world. At each moment it receives information from the world and influences at the world by the information it works out. This assumption does not contradict anything because we are, as human intelligence, living in our world with the same sort of interactions. After these assumptions, *AI will be such an object, which in an arbitrary world will cope not worse than a human*.

Taking in account Humphrys' comments it seems that Dobrev's definition of AI is considerably fairer and more achievable than passing the Turing Test. Moreover, it seems to expand Fogel's definition of intelligence making it less abstract. In fact, it can underestimate the human intelligence because in some worlds, such as playing logical games, a machine with pre-defined algorithm can perform much better than any human being and can appear as super-intelligent. Hence Dobrev's definition should be modified to minimise the influence of such situations on the final test result. The more appropriate definition then should be: *AI will be such an object, which on average in an arbitrary set of worlds will cope not worse than the average of an arbitrary group of humans*.

Even though this definition seems accurate it still raises questions about the way of assessing the degree of coping, which in turn leads to a question about the meaning of life that actually defines who is coping better or worse. Obviously this can be cruelly approximated assuming "surviving" as the assessment factor. In theory the definition appears to be easier to achieve than to pass the Turing Test, however the test based on this definition could be much more difficult to implement to truly apply it. Human beings have natural access to many "worlds": we can see, touch, feel, use hands and legs etc. An artificial object in contrast by default has no interfaces at all. To test such an object using Dobrev's definition we need to build an appropriate interface for each "world." Each interface should be as functional as the human version otherwise it will have disadvantages that will make significant influences on the overall result. Vice versa if an artificial object's interfaces are more advanced than the human ones then human beings have considerable disadvantages an example of which can be seen in the "games playing worlds." The thoughts about the interfaces take us back to Humphry's comments above about the impossibility of testing artificial intelligence in a sense of the human one unless the tests are done on the abstract worlds, and both human beings and artificial objects have equal interfaces to and equal prior knowledge about those worlds.

1.2 Machine Learning

An Artificial Intelligence system, which will be able to pass the Turing, Fogel, Humphry, Dobrev or other proposed tests, whichever will be finally accepted as the most correct one, is the ultimate goal to be achieved.

Making a comparison with the human being, currently the only example of intelligent system, it seems natural to think that when finally created Artificial Intelligence will not be a single component system. This is especially true if the modified Dobrev's definition of AI will be used to test its intelligence. Human being consists of many components such as brain, heart, liver, lungs, ears, eyes, legs, hands and many more including a huge amount of various sensors and nerves passing the information in terms of signals around the body. Working together in a precise coordination all these components make up the intelligent system called human being. It is still not discovered what exactly makes us intelligent but it is clear that none of the components mentioned can function alone and show the sign of intelligence. Another example of a perceived and argued intelligent system is a group of wasps. An individual wasp is fixed in its behaviour. However, the species of wasps are considered to be intelligent as a group or system by many researchers.

Therefore, it is very likely that the Artificial Intelligence system will have to be a defined collection of a variety of artificial components. Exactly in the same manner as a human being is constructed, the AI system would have to be constructed from the set of artificial components designed to work together in order to achieve the goal of been able to function sufficiently enough for passing the accepted AI test. Due to many years of development in medical and biological sciences we now know all physical components that make up a human being. So, what would our guesses be when we try to think of the components of the Artificial Intelligence system?

Attempts to envisage the structure of the AI system resulted in two different approaches to the problem of AI creation. The first approach is based on the idea of copying the nature. As all of the physical components and their functions within biological organisms including human beings are known, the simplest approach seems to be to create all those components artificially and assemble them together in the precise structure. It is also perceived and almost 100% accepted for fact that the reason for human intelligence lies in the struc-

ture of our brain. The brain consists of neurons, which are apparently very simple (in terms of functionality) devices that can be easily created artificially. Therefore, in this approach to AI creation all the attention is concentrated on artificially creating neurons, either biologically or using the computer model, and then connecting them into a system, which supposedly can learn and apply the knowledge whenever required. Neural Networks learning algorithm is an example of such approach, which is modelled on the computer. Some positive results have been achieved using this approach, however our brain consists of billions of neurons, and there is a possibility for a huge number of possible interconnections one neuron can make with other neurons. Currently even the most advanced artificially created bio-neurons and neural networks can only produce the tiny fraction of these amounts of neurons and connections. Hence the abilities of such artificial systems are very limited and certainly not enough for the AI test. In principle, however if our technical and computational resources were not limited it might be possible to develop this approach further and eventually create an AI system. This system would be based on the same theory that is used by nature hence it would be very similar to the human in terms of the level of intelligence.

The second approach to the AI creation states that it is not necessary to copy the nature to create a system that is at least as intelligent as the human being. This approach heavily relies on our advances in mathematics, computer technology and computer programming. Observing the intelligent subject's behaviour and also thinking about the nature of the Turing and other tests, it is obvious that the intelligence is identified by the way of processing the incoming information and the way of reacting upon the new pieces of information. The fact that computers can process some kinds of information millions of times faster than the human gives us scary thoughts that using this approach an AI system could eventually be even more intelligent than the human. For the same reason this approach is developing much faster than the first approach described above. While the "nature copying" approach is suffering from the computational and technical difficulties, in this approach the results are constantly improved by enhancing existing and introducing new algorithms for information processing. In this approach we may assume that an AI system is a kind of system that can analyse and learn the incoming information, then react upon the new bits of information according to the learnt knowledge. Therefore the core component of such system would be a computer program that can learn the data and produce the response by applying the discovered knowledge to the new incoming

data. The main research in this approach is concentrated on developing algorithms for computers that can learn and predict. Hence the whole new field of science called Machine Learning has now emerged, whose main goal is researching and enhancing learning algorithms. This thesis is another research work in the Machine Learning field presenting another fundamental enhancement to the existing learning algorithms that could bring us even closer to the construction of the computer based AI system.

1.3 Qualified Predictions

Developing and enhancing machine learning algorithms for the purpose of eventually creating an AI system is a very interesting research topic. However, we are still quite a long way from the final destination even though we now have a few learning algorithms which have been tested and found to be performing well. It turns out that learning algorithms themselves even without the "intelligence" are also useful in many fields. Learning machines are now competing with and almost replace statistical methods of data analysis in various areas including medicine, stock markets, manufacturing etc. A variety of expert systems have been implemented helping professionals to make decisions in the fields where the knowledge from the vast amounts of data is necessary to make a correct decision. In such cases humans are likely to make a mistake by not been able to process the available information efficiently. In contrast, a learning machine is able to learn the presented data without human mistakes and apply the knowledge to the new data. Predictions on the new data are then used as help in order for a human operator to make a decision. There are also fully automated implementations of learning machines such as machines for images and hand writing recognition, speech recognition etc.

Examples of learning algorithms include Nearest Neighbours [33], Dual Ridge Regression [8], Bayesian Classifiers [59,60], Neural Networks [7], Support Vector Machines [6]. The main function of these machines is to learn the input data presented in the following format:

$$z = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n) \tag{1.3.1}$$

where $x \in \Re^k$ is an object represented as a vector of size k, y is its label assigned by a supervisor and $y \in \{Y_1, \ldots, Y_c\}$, where $c \in N$ and all $Y_i \in N$ for Pattern Recognition problems or $y \in \Re$ for Regression problems. When new object x_{new} arrives the machine using the knowledge learnt from the previously given data tries to predict the label y_{new} of this new object.

Many machine learning algorithms share the same limitation – they are designed to output just one number for each new unclassified example (we will call it *bare prediction*). The user then has to rely on the previous experience or if you are lucky on some relatively loose theoretical upper bounds on the probability of error to actually assess the quality of the prediction. This kind of quality assessment becomes very important if machine learning algorithms are used as a help tool for professionals in order to simplify their decision making process. This is especially vital when vast quantities of data are available that cannot be comprehended by any human being within a reasonable amount of time. In such cases bare prediction is simply not enough and does not really help anyone. It is also very important for the machine to be able to assess itself if it is to be used as a component in the AI system. As human beings we are given this ability by nature. In most of the cases we can assess our actions, we are given the ability to be certain or not in what we do or say. Based on the certainty level we either make a decision to do something, postpone it or consider other actions that we might find more certain. Ability to do this kind of assessment is regarded as an abstract notion of intuition but, if we are to think logically, there is no other explanation apart from the fact that this ability is coming from the knowledge and experience we have. This explains the simple fact why people with more experience and knowledge usually make better decisions. Obviously there are exceptions and sometimes people with less experience and knowledge manage to make a better decision or produce a more correct answer to a given problem, but it happens rarely, everyone accepts it just as a miracle and luckiness. Moreover this event can be explained by Von Mises's "Law of Excluded Gambling Strategy" which says that a gambler cannot consistently make more profit in the long run betting according to a system than from betting at random. This means that in our case sometimes a random answer could be better than the one derived from knowledge and experience especially when the presented problem does not have a known solution yet. This further shows that the origin of human intuition does in fact lie in our ability to assess various possible answers and produce the one that has the most certainty based on our knowledge.

If we are to construct an AI system based on machine learning algorithms it seems inevitable that we must "teach" learning algorithms to assess themselves. In other words we have to make them able to produce meaningful *Qualified Predictions*.

There have been number of attempts to overcome this limitation of the standard learning algorithms. All of these attempts are based on the modifications of the original algorithms in order to produce posterior probabilities for each output. For instance posterior probabilities can be learnt using cross-entropy and various other error functions on Neural Networks [9], and also probabilities can be assigned to the Support Vector Machine output as described in [10]. Well known Bayesian classifiers produce output probabilities by design. However, these approaches are specifically designed and limited to the underlying learning algorithm. Some such measures cannot be applied to individual test examples, others are not very useful in practice (PAC theory; see, e.g., Melluish et al. [35]) and Bayesian methods rely on strong underlying assumptions. Also the methods used to obtain the probabilities are fundamentally different for each algorithm making comparison of the results virtually impossible.

Another important fact is that in the case of multi-class pattern recognition the posterior probabilities do not really show us the level of confidence in each prediction. It seems that probability of a result is not a suitable measure in the quality assessment of the prediction. It can only express expectations of properties of the total set of outcomes for the given process under given distribution. In general real-world data we cannot assume that it comes from a particular known distribution, hence we make a very weak assumption that our data is independently and identically distributed. We are interested in the exact quality assessment based purely on the knowledge we have rather than on the probability of the event under the unknown distribution.

A simple example illustrating the limitation of the probabilistic approach can be seen in the multi-class pattern recognition problem. Increasing probability of one class results in the decreasing probabilities for other classes because all of them have to total up to 100%. The first problem is when a new example has nothing similar with any of the classes in the given data set e.g. we have a dataset of digits and the new example is a letter. In this case at the best we would have all classes predicted with the same probability $1/c$ where c is the number of classes. This does not mean a lot because to assess the situation truly in

this case we would like to have probability close to 0% for all classes, which is impossible under the definition of probability.

The probabilistic approach may give us some degree of confidence in our prediction however it cannot clearly discriminate between the prediction and other possible predictions and also it cannot tell us how typical our prediction is. Assuming we have the same situation as above, for some reason on a set of digit images we might try to predict a classification of an image of a letter. It will definitely be predicted as a digit (because these are the only classes known to the program) and it is quite possible that this letter will be assigned a reasonably high probability. To decide either to classify this image as predicted or not we need to ask another question for instance: "How typical is this image compared to all other images with the same classification as the predicted one?" Unfortunately, this question cannot be answered easily and hence we are unable to obtain a comprehensive assessment of the situation.

Another existing approach that seems to have a few things in common with qualified predictions does not actually involve learning algorithms. Along with the machine learning field there is another research field called Data Mining and Knowledge Discovery which has some overlap with the machine learning. Instead of performing some kind of learning, researchers in this field are concentrating on finding useful patterns, rules or facts in row data. They have two measures called Confidence and Support a combination of which can be used to distinguish between *powerful* and *weak* patterns. The confidence is calculated by dividing the number of rows that have a specific pattern by the number of rows that do not have this pattern. The support is in fact just a number of rows that contain this pattern. Overall the idea is that the higher the Confidence and Support levels, the more powerful is the pattern. The final result of data mining is a discovery of all facts and rules in a data, whereby rules and facts are the patterns that have confidence higher than 50%. Combining two measures for assessing the quality of a rule gives us possibility to make an assessment from two different angles, which is a natural thing to do and considered by everyone as common sense. In real-life situations we are always asked to evaluate the situation from different angles before making a decision, hence the AI algorithms and therefore the machine learning algorithms must be able to do it as well.

Data Mining can be considered as a way of constructing the AI because it produces rules and facts as output that can be used to deduce which pattern a new data belongs to and make some kind of prediction. Well known programming language for AI called PROLOG is based on the Data Mining approach and many expert systems have been built over the years. With regards to the qualified predictions, Data Mining only assigns quality assessment measures to the rules and facts hence the prediction quality of a new example will be the same as the confidence and support assigned to the associated rule.

Clearly a new approach to the area of qualified predictions is required. This approach must not have the limitation of the probabilistic ways of assessing the quality, and it should provide more than one view to the quality from different perspectives. Also this approach must not have the limitation of the quality assessment used in the Data Mining field, and it should provide the individual quality measures in the continuous space for each new example.

1.4 Confidence Machines

To understand the overall idea of the new prediction method let us consider the following task. We are given a sequence of numbers

2 2 3 3 2 2 3 3 2 ?

and asked to predict the number shown as the question mark. This kind of problems exists in every aptitude test and their role is to assess a person's logical thinking or sometimes they are referred even as "intelligence tests." Surprisingly the results of such tests play a very important role in our life because our position in life quite often depends on them. We may be accepted to an educational institution or not, employed or not and the choice of our profession also depends on the results of such tests.

The obvious answer to the task above will be **2** and it will be answered as such by most of the people. Moreover, we are able not only to give an answer, we are also able to crudely quantify the quality of each possible answer. In this particular task most of us would probably say that **2** is the most likely answer, **3** is a less likely answer and something like **5** is the least likely one.

How do we know this?

To come up with such an answer we do not compare possible answers to see which one is the closest (Nearest Neighbours approach), we do not try to find the probability of answers (Bayesian approach), we do not transform each number into a high-dimensional space and separate each possible answer from others (Support Vector Machine), in fact we do not use any of the logic used by existing machine learning algorithms to come up with an answer. However, the key to our ability to perform the task above could be more straightforward than expected.

It seems natural to think that when we search for an answer we try to find a combination that will give us the best pattern i.e., the combination that can be described by the shortest possible description. It could be that the reason why we say that the missing number should be **2** lies in a fact that this answer makes the whole sequence least random (most typical) compared to any other possible solution.

The natural approach described above gives us an idea for a new learning machine method. None of the existing machine learning algorithms, apart from the new one proposed by Vovk and Gammerman [18], employ this method to make predictions. In their work they introduce an alternative method called Confidence Machine for learning and prediction which is based on measuring randomness of a sequence of objects. This method also presents a totally different approach to solving the problem of a prediction's quality assessment, the first practical implementation of which was introduced in [11]. In this paper Transductive Confidence Machine (TCM) was described, which practically implements a confidence machine and introduces the novel quality assessment measures. The new measures were *confidence* and *credibility*. Confidence value shows us the level of confidence for the given prediction with respect to other possible predictions and the credibility value serves as an indicator of the reliability of the data upon which we make our prediction. Together these values give us a complete picture for each individual prediction and an appropriate decision can be made whenever required. Instead of using probability of a possible sequence of data, which does not mean much as it will be shown below, measures of confidence and credibility are derived from the statistical notion of p-values which in turn are obtained from measuring the randomness (or typi-

calness) of a particular sequence. To obtain the qualified prediction a set of sequences is constructed, each sequence representing a concatenation of the existing data and the new data with one possible outcome. Using any of the possible methods we measure the typicalness level of all the sequences in the set. Intuitively the prediction will be the outcome assigned to the sequence with the highest typicalness level. Furthermore, the *credibility* of the prediction will be the highest typicalness level itself and the *confidence* will be one minus the second highest typicalness level.

Use of typicalness (or randomness) for making and assessing decisions seems natural and adequate. Analysing how humans make decisions in everyday life, it is easy to discover that we make decisions by analysing how well a particular answer fits within the knowledge we have about the subject. In other words we do calculate the typicalness of a particular solution even without realising that this process takes place in our brain. If these assumptions are in fact true then an interesting conclusion can be made. As we already mentioned, aptitude tests are widely perceived to be the tests for "intelligence." These tests are shown to be the exercises in finding patterns or, in other words, measuring randomness because we also compare randomness levels amongst possible solutions to decide which one is the least random and the most appropriate. Hence, the intelligence level can be quantified by the ability to find patterns or by the ability to measure randomness correctly.

Measuring randomness, however, is another problem which presents a whole variety of challenges. Probabilistic approach fails to produce any meaningful measures of randomness. Nevertheless, the answer can be found in the algorithmic information theory and in Kolmogorov complexity. Using Kolmogorov complexity as a measure of randomness we can create an ideal procedure of predicting with confidence and credibility. In practice, however, the procedure has to be approximated as Kolmogorov complexity cannot be calculated. It turns out that the process of calculating randomness level can also be approximated by using existing learning algorithms such as Support Vector Machines and others. Thus in [11] it was shown how to practically create a Confidence Machine using existing learning algorithms and transductive approach. One drawback of the transductive approach to this problem is that for each new example TCM has to relearn all the existing data by constructing a new decision rule. For relatively small training and test sets this is not a problem, but in larger data sets constructing a decision rule may take a considerably longer time,

which is not convenient for practical use. For example, to run SV TCM on the USPS data set of hand-written digits, with 7291 training examples and 2007 test examples takes several weeks to complete. In this thesis we are following the idea of predicting with confidence and credibility and specifically, we study how to do it efficiently in the case of pattern recognition and relatively large data sets. We will analyze an inductive approach to approximating randomness and hence creating an Inductive Confidence Machine (ICM) with the purpose of speeding up the process of producing qualified predictions.

To assess the perfomance and compare various methods of approximating randomness we set six different performance criteria that can be used to compare confidence machines on measures other than error rate. This is necessary because they might produce exactly the same error rate, so there is a need for measures to assess their randomness approximation mechanism.

Chapter 2

2. Algorithmic theory of randomness

2.1 Introduction

Our method of producing qualified predictions is derived from the algorithmic theory of randomness. In this chapter we would like to give an introduction to the algorithmic theory of randomness. We will describe philosophical and intuitive ideas of randomness. This theory is based on the original Kolmogorov complexity theory, so we will explain the idea and theory behind the Kolmogorov complexity along with the reasons why we are interested in it. Then we will move into describing the algorithmic approach to defining randomness and how it is derived.

2.2 Randomness

The concept of randomness is historically a recent development started probably in the 1700's. Even more recently in the 20th century, randomness and probability have been embraced by quantum physics. The idea of existence of randomness in our world has not been left alone by any of the greatest scientists, once A. Einstein said: “God does not play dice with the Universe!” which shows his opinion on this matter.

Some people reject the notion of randomness on religious or philosophical grounds. They do not accept the idea that events in our world can occur randomly. One might want to consider this controversial idea. If you believe in the notion of “free will”, then you must believe that human behaviour is ultimately random! If you could know ahead of time what course of action a person would choose, then his/her choice is not completely “free.” If the choice is determined by past history, personality, or current forces, then how could the choice result from “free will”? The result is that freedom implies unpredictability. Because

the behavioural psychologist B.F. Skinner [13] believed that our behaviour was totally determined by our past history, genetic endowments and current forces, he denied the possibility of free will, or even freedom.

What exactly does the word "random" mean? Dictionary definitions emphasize the notion of "apparent absence of cause, planning or design," "lack of method or system," or "accidental, haphazard." Statistical definitions involve the inability to predict outcomes or to find any pattern in a series of outcomes. Some people use the word to mean that something can't be explained with current theory or they might mean that the causes can never be fully specified.

Nevertheless, everyone has an intuitive notion of what a random number (sequence) is. For example, following Chaitin's example [19], consider two sequences of binary digits:

01010101010101010101010101
10010011011000111011010000

The first is obviously constructed according to a simple rule - it consists of the pair 01 repeated thirteen times. If one asked to predict on how the sequence might continue, one could easily say that the next two digits would be 0 and 1. Analysis of the second sequence of digits would not reveal any comprehensive pattern that would give one a hint on how this sequence might continue. There is no obvious rule governing the formation of the second sequence and there is no rational way to guess the succeeding digits. The sequence appears to be a random assortment of 0's and 1's.

Unfortunately, intuitive notion is not an exact rule and may appear to work only in cases of short sequences. We need an exact definition of what a random number is and how to measure the randomness of a sequence.

Based on the classical probabilistic approach let's assume that the sequences above were generated by flipping a coin 26 times and writing a 1 if the outcome was heads and a 0 if it was tails. Tossing a coin is a classical procedure for producing a random number, and one might think at first that the provenance of the sequence alone would certify that it is random. This is not so. Tossing a coin 26 times can produce any one of 2^{26} binary sequences, and each of them has exactly the same probability 2^{-26}. If origin in a probabilistic event were made

the sole criterion of randomness, then both sequences would have to be considered random, and indeed so would all the others, since the same mechanism can generate all the possible sequences.

The fact that for all sequences above the classical probability theory assigns the same probability shows that it cannot express the typicalness or randomness of an individual sequence. It can only express expectations of properties of outcomes of random processes i.e. the expectations of properties of the total set of sequences under some distribution. Clearly a more sensible definition of randomness is required, one that does not contradict the intuitive concept of a "pattern less" sequence. Such definition has been found in the algorithmic information theory and in the idea of Kolmogorov complexity.

2.3 Kolmogorov complexity

Recent discoveries have unified the fields of computer science and information theory into the field of *algorithmic information theory*. This field is also known by its main result, Kolmogorov complexity. With the growing number of computers in our society and their ever increasing role in our lives we have already entered the information age where everything is described in terms of information processed by the computers. Information is used to describe the cultural structures of science, legal and market institutions, art, music, knowledge and beliefs. Information is also used in describing the structures and processes of biological phenomena and phenomena of the physical world. As information becomes more and more important and we are increasingly dependent on it, there is a definite need to understand the information and its properties exactly. Kolmogorov complexity gives us a new way to grasp the mathematics of information. It was introduced independently by Solomonoff [1,2], Kolmogorov [3] and Chaitin [4].

The notion of Kolmogorov complexity has its roots in probability theory, information theory and philosophical notions of randomness. As shown in the previous chapter according to probability theory if we flip a coin hundred times the probability of each sequence of outcomes is equally likely 2^{-100}. This means that probability theory gives us no basis to challenge an outcome after it has happened. The satisfactory resolution for this problem was found by combining

notions of computability and statistics to express the complexity of a finite object. This complexity is the length of the shortest binary program from which the object can be effectively reconstructed. It may be called the algorithmic information content of the object. This quantity turns out to be an attribute of the object alone and absolute. It is the Kolmogorov complexity of the object. Formally

$$C(x) = \min\{|p| : U(p) = x\} \tag{2.3.1}$$

where $C(x)$ is the Kolmogorov complexity of object x, U is a universal Turing machine, and $U(p) = x$ means that p is a binary program which reproduces the x and terminates. Thus Kolmogorov complexity of the sequence shows its regularity - it will be much smaller than the length of the sequence for regular sequences and approximately equal to the length of the sequence in case of the truly random sequence. For our sequences above all three of them will have different Kolmogorov complexity hence different degrees of randomness.

2.3.1 Plain Kolmogorov complexity

The definition of the Kolmogorov complexity above and consequent results of this definition is a part of so called *plain Kolmogorov complexity* theory.

Let X be some ensemble of constructive objects, such as the set Q of all rational numbers or the set $\{0,1\}^*$ of all finite binary sequences. A partial function F from $\{0,1\}^*$ to X is called *mode of description* if it is computable. Intuitively, F is interpreted as a decoder: if $F(p) = x$ then p is regarded to be a description of the object x. The length of the shortest description

$$C_F(x) = \min\{|p| : F(p) = x\} \tag{2.3.2}$$

is called the $F-complexity$ of x. Obviously, the length of the shortest description for the object x depends on the choice of the mode of description F.

However, it has been shown (see, e.g., [5]) that there exists a *universal mode of description* such that

$$C_U(x) \leq^+ C_F(x),\ x \text{ ranging over } X.$$

The function $C_U(x)$ is called *plain Kolmogorov complexity* and it is abbreviated to $C(x)$. If U and V are two universal modes of description, then $C_U(x) =^+ C_V(x)$, therefore the natural accuracy with which the function $C(x)$ is defined is "up to an additive constant." As stated earlier, partial function F is called mode of description if it is computable; universal Turing machine can simulate any other machine and any computable function, hence the universal mode of description U is in fact the universal Turing machine. This results in obtaining the general definition of the Kolmogorov complexity (2.3.2) described earlier.

2.3.2 Prefix Kolmogorov complexity

The development of an algorithmic theory of complexity according to the standard definitions within plain Kolmogorov complexity theory resulted in many achievements, but for certain goals the mathematical framework is not yet satisfactory. The plain complexity has a number of inconvenient features e.g. it is not sub-additive, it has non-monotonicity over prefixes and others [5]. Therefore the *prefix Kolmogorov complexity* was introduced, which deals with such problems. This is the complexity induced by Turing machines with a set of programs in which no program is a proper prefix of another program.

Using the definitions from the previous section, a mode of description F is called *prefix-free* if the following holds:

$$\left.\begin{array}{l} p \subseteq q \\ F(p) \text{ is defined} \end{array}\right\} \Rightarrow \big(F(q) \text{ is undefined}\big) \qquad (2.3.3)$$

Another possible definition of the prefix Kolmogorov complexity is that a mode of description F is called prefix if the following takes place:

$$\left.\begin{array}{l} p \subseteq q \\ F(p) \text{ is defined} \end{array}\right\} \Rightarrow \big(F(q) \text{ is defined and } F(q) = F(p)\big) \qquad (2.3.4)$$

It can be shown (see, e.g., [5]) that, if U is a universal *prefix-free* mode of description and V is a universal *prefix* mode of description, then

$$C_U(x) =^+ C_V(x)$$

Therefore we fix a universal prefix-free and a universal prefix mode of description and call it *prefix Kolmogorov complexity* $K(x)$.

2.3.3 Conditional Kolmogorov complexity and Information Distance

The natural extension to identifying the complexity of an object is the ability to specify the complexity of an object when another object is already specified. Therefore the *conditional Kolmogorov complexity* is defined as

$$C(x \mid y) = \min\left\{|p| : F(y, p) = x\right\} \qquad (2.3.5)$$

for the plain Kolmogorov complexity, where F is a universal partial recursive function. The same definition

$$K(x \mid y) = \min\left\{|p| : F_k(y, p) = x\right\} \qquad (2.3.6)$$

can be given for the prefix Kolmogorov complexity but in this case F_k is a universal prefix partial recursive function.

Unconditional Kolmogorov complexity can be defined from the conditional one as $C(x) = C(x \mid \varepsilon)$ and $K(x) = K(x \mid \varepsilon)$ where ε is the empty string.

Kolmogorov complexity is a measure of absolute information content of an object. Conditional Kolmogorov complexity gives us the possibility to measure the information required to construct an object from another object. In effect there should be a possibility to measure the absolute information distance between

individual objects. Such a notion of universal information distance between two objects is the minimal quantity of information sufficient to translate between the objects x and y, generating either object from the other.

Intuitively, the minimal information distance between x and y is the length of the shortest program for a universal computer to compute x from y and vice versa. It might seem that the conditional Kolmogorov complexity $K(x \mid y)$ is exactly what is required; however, it is unsuitable because it is asymmetric. For example, $K(\varepsilon \mid x)$ is small for all x even though it is obvious that a long random string x is not similar to the empty string ε. One possible definition of the information distance using conditional complexity where the asymmetry disappears is $K(x \mid y) + K(y \mid x)$; however, it over estimates the information required to translate between objects because there might be some redundancy between the information required to translate from x to y and the information required to translate from y to x.

After investigating to what extent the information required to compute x from y can be made to overlap with that required to compute y from x, it was found [17] that almost complete overlap can be achieved in all cases and the measure of distance $KD(x, y)$ shown to be, up to a logarithmic additive term, equal to the maximum of the conditional Kolmogorov complexities between objects x and y:

$$KD(x, y) \approx \max\left\{K(x \mid y), K(y \mid x)\right\} \tag{2.3.7}$$

2.4 Algorithmic Randomness

We have shown that probability theory is unable to give us the definition of a random sequence. Even though the notion of randomness seems intuitive the answer has not been found in the classical mathematics. However, with the development of the algorithmic information theory we are now able to define randomness.

Finite sequences that cannot be effectively described in a significantly shorter description than their literal representation are called random. Infinite random sequences are the sequences where all initial finite segments are random.

Kolmogorov complexity shows the absolute information content of an object or the amount of information required to reconstruct the object. Effectively if the amount of information required to reconstruct an object is significantly smaller than the literal description of the object then this object can be compressed and obviously described in a shorter description. Therefore, Kolmogorov complexity is an indicator of randomness. Now we may say that finite sequences which have Kolmogorov complexity equal to the length of the sequence up to an additive constant are called random.

Algorithmic information theory and Kolmogorov complexity give us an ability to talk about the randomness like never before. Statistics and probability theory have only been able to identify what is not random and the definition of randomness has been beyond their abilities. Suddenly the notion of randomness becomes not only a speculation but a notion that has clear definition.

Kolmogorov complexity is a quantitative measure of the complexity of an object. This allows not only to define randomness but also to measure the level of randomness of a finite sequence. This can be achieved using the p- and i-randomness deficiency of an element x of a set A which can be defined as follows (see, e.g., Li and Vitanyi [5]):

$$d_A^p(x) = \log|A| - C(x \mid A) \tag{2.4.1}$$

$$d_A^i(x) = \log|A| - K(x \mid A) \tag{2.4.2}$$

where C is plain Kolmogorov complexity and K is the prefix complexity. This also means as shown in [5] that there is a universal randomness test

$$d_0(x) = l(x) - C\left(x \mid l(x)\right) \tag{2.4.3}$$

which tells us that x is random or incompressible if $d_0(x)$ is small with respect to $l(x)$. Otherwise $d_0(x)$ shows the randomness deficiency telling us how non-random x is.

Chapter 3

3. Confidence Machine

3.1 Introduction

Confidence Machine is a type of learning machine that produces qualified predictions. In this section we will go into the details of the confidence machine principals. The theory of producing qualified prediction is based on the algorithmic information theory, however practical implementations are based on the randomness approximation using well known learning algorithms. Hence we will start from describing the theory behind the confidence machine which represents an ideal approach to predicting with confidence and credibility. Then we will briefly recall the ideas behind the two well known bare prediction learning algorithms and finally how the bare prediction algorithms can turn the theory of qualified predictions into practical implementations.

3.2 Ideal Qualified Predictions

In the introduction we already mentioned the importance of producing qualified predictions and what kind of attempts have been done to add the ability of assessing the quality of predictions to existing learning algorithms. As already discussed these attempts are based on the probabilistic approach, this means that existing learning algorithms are modified to produce posterior probabilities for each of the possible prediction. However, due to the nature of probability such approaches suffer from certain limitations. For example most of the approaches involve applying Bayesian framework to existing learning algorithms to obtain confidence values, however they rely on unjustified priors and if priors are incorrect then these confidence values have no theoretical base. Moreover, the ways in which the probabilities are obtained are different from

one learning algorithm to another that makes it difficult to compare or even assess the overall performance.

In this section we will describe, following [18], an ideal approach to prediction with confidence and credibility which is not based on finding the posterior probabilities. This approach is based on the algorithmic information theory [19] which gives us an ability to speculate about the randomness of individual sequences.

All practical problems solved by standard learning algorithms are presented in the following format. Assume we have a sequence of data:

$$z = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n), \tag{2.6.1}$$

where $x \in \Re^k$ is an object represented as a vector of size k, y is its label assigned by a supervisor and $y \in \{Y_1, \ldots, Y_c\}$, where $c \in N$ and all $Y_i \in N$ for Pattern Recognition problems or $y \in \Re$ for Regression problems. When new object x_{new} arrives the machine using the knowledge learnt from the previously given data z tries to predict the label y_{new} of this new object.

Taking the algorithmic information theory route, theoretically we may consider computing Kolmogorov complexities of each sequence z_i (assuming each sequence z_i can be transformed into a binary sequence) in the following set of sequences:

$$z_1, z_2 \ldots, z_c, \text{ where}$$
$$z_i = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n), (x_{new}, Y_i), \; i = 1, \ldots, c. \tag{2.6.2}$$

Kolmogorov complexity strongly relates to the randomness as shown in Section 2.4, therefore potentially we can measure randomness of each sequence z_i. Naturally, the least random sequence z_i will correspond to the best postulated label Y_i for the new example x_{new} given the training data z. Moreover, quantifying the degree of randomness for each postulated label we could produce new measures that will give an accurate indication of the quality of prediction as it will be shown below. There are some complications with this approach,

first of all, Kolmogorov complexity is not computable. Secondly, in the traditional algorithmic information theory $z_i \in \{0,1\}^*$ i.e. z_i is a binary string, whereas in the typical practical applications z_i is a set of complex objects with their classifications. There is no trivial way to convert a set of complex objects into a binary string. Therefore, instead of using the Kolmogorov complexity directly, we must use another randomness test to deal with such data in practice.

As already mentioned, no empirical process can be tested against an idealized notion of randomness. In practice we cannot tell what randomness is, only what isn't. There has been a number of attempts to come up with randomness tests that can identify how random a sequence is. Examples of such tests are *p-typicalness* (named by Gammerman and Vovk [18]) from Martin-Löf [21], *i-typicalness* (named by Gammerman and Vovk [18]) from Levin [24], Lutz randomness [26] and Ko randomness [27]. For finite sequences (unlike the infinite ones) there is no sharp distinction between typical and atypical sequences, but one can define the *typicalness level*, which is close to 0 if a sequence is not typical. In practice we deal only with finite sequences, so all the definitions below are applicable to such sequences.

3.2.1 Martin-Löf p-typicalness

Martin-Löf's [21] definition of randomness using the notion of typicalness is called *p-typicalness* by Gammerman and Vovk [18]. It is known that a infinite sequence is Kolmogorov random if and only if it is Martin-Löf random (see, e.g., Levin [24]). It perfectly suits us in achieving our goals, hence it is used in our confidence machines to perform the randomness tests. In fact other typicalness tests such as i-typicalness can also be used as shown in [83], but we will be using p-typicalness only for our confidence machines because it has a nice feature of been able to be substituted by the statistical notion of p-values for practical implementations.

In a general case we consider a typicalness of elements of some constructive space Ω which is equipped with some computability structure allowing us to speak of computable functions on Ω. The word "computable" intuitively means "doable by a computer." As there is a huge number of different kinds of computers and of different models of computability, we will work with the universal Turing machine [20], which can simulate any computer. Hence, a partial function $f : A^* \to B^*$ is computable if there exists a Turing machine which, when fed with any string $(a_1,\ldots,a_n) \in A^*$, outputs either $f(a_1,\ldots,a_n)$ followed by "stop" symbol if $f(a_1,\ldots,a_n)$ is defined or nothing if the result is undefined.

Let P be a computable probability distribution in Ω. Then we say that a function $t : \Omega \to N$, where N is the set $\{0,1,\ldots\}$ of non-negative integers, is a *log-test for p-typicalness* with respect to P if:

$$\begin{aligned} &1.\ \text{for all } m \in N, P\{z \in \Omega : t(z) \geq m\} \leq 2^{-m}; \\ &2.\ t \text{ is lower semi-computable.} \end{aligned} \tag{2.6.3}$$

Intuitively, a log-test for p-typicalness is a device for finding unusual features in the element $z \in \Omega$. Item 1 in the definition says that the amount of unusual features is measured in bits, while item 2 says that the device should be realisable on a computer. Also, our tests for typicalness are allowed to work forever all the time finding new regularities in the data, hence, technically, function t is only required to be lower semi-computable rather than computable. Briefly, if a

real-valued function $t: X \to \Re$ is *lower semi-computable* then its proper subgraph $\{(x, y): t(x) > y\}$ is a constructively open set in the product $X \times \Re$ i.e., when given an oracle for a point $x \in X$, we can compute an increasing sequence $y_1 y_2 \ldots$ of real numbers converging to $t(x)$. For more detailed information about lower semi-computable functions, constructively open sets and other components of *constructive topology* see [18].

As we need to find all unusual patterns in the presented sequence z we are interested in the universal log-test for p-typicalness, which was shown in [18] to exist if P is computable. By fixing some universal log-test for p-typicalness d_P^p with respect to P we introduce *p-typicalness deficiency* $d_P^p(z)$ of z with respect to P, which is the ultimate intelligence that eventually finds out all patterns in the given data sequence that make it non-typical.

The standard notions of algorithmic theory of randomness use the logarithmic scale, hence the definitions of typicalness so far have been presented in terms of log-tests. In practical applications, however, it is more convenient to use the direct scale. Without any consequences we can reformulate Martin-Löf definition into a *test for p-typicalness*.

We say that a function $t: \Omega \to [0,1]$ is a *test for p-typicalness* with respect to P if:

$$\begin{aligned} &1.\ \text{for all } r \in [0,1], P\{z \in \Omega : t(z) \le r\} \le r; \\ &2.\ t \text{ is upper semi-computable.} \end{aligned} \qquad (2.6.4)$$

Similarly, it also makes sense to move from the log scale into the direct scale for the definition of p-typicalness deficiency. We will call the function $\lambda_P^p(z) = 2^{-d_P^p(z)}$ the *p-typicalness level* of z with respect to P. In the rest of this thesis we will be using the p-typicalness level only.

3.2.2 Confidence and Credibility

Our data is supposed to satisfy the *iid assumption* (all the examples are independent and identically distributed). This is a typical assumption in machine learning and it will be required below to make sure our data sequences are exchangeable. As we use typicalness to perform randomness tests, to be consistent with Kolmogorov theory [22] and non-parametric statistics (see, e.g., [23]) the typicalness level λ_{iid} with respect to this *iid model* will also be called the *randomness level* in the direct scale or *randomness deficiency* in the log scale.

Now *confidence* and *credibility* might be derived from the typicalness level λ_{iid} using the following procedure displayed on Figure 3.2.2.1 (assuming we have an oracle computing λ_{iid}):

Figure 3.2.2.1: Confidence and credibility using typicalness level.

1. Consider all possible values $Y_1,\ldots,Y_c \quad Y, c \in N$ for the label y_{new} . For example, in a case of hand-written digits the possible values are $0,1,\ldots,9$.

2. Compute the typicalness level λ_{iid} for all possible sequences

$$z_i = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n), (x_{new}, Y_i) \text{ , } i = 1, \ldots, c$$

3. An ideal prediction will be the Y_i which corresponds to the largest typicalness level $\lambda_{iid}^{\max}$.

4. *Credibility* of the prediction is the largest typicalness level $\lambda_{iid}^{\max}$ itself.

5. *Confidence* of the prediction is $1-\lambda_{iid}^{\max_2}$, where $\lambda_{iid}^{\max_2}$ is the second largest typicalness level obtained.

This procedure is very intuitive. Based on the typicalness level given by an oracle we find the most typical combination of the training examples, the new unlabelled example and a postulated label for it. Our prediction then will be the postulated label Y_i corresponding to the most typical combination. High (close to 1) *confidence* would mean that all alternative classifications are excluded (make the combination atypical), and high (significantly different from 0) *credibility* would mean that the new unlabelled example itself is typical with respect to other examples of the same class (if it is not, the classification problem is not well-posed).

The procedure described above represents an ideal approach to producing confidence and credibility provided we have an oracle computing λ_{iid}. This procedure is applied only in a case of pattern recognition, which is a type of learning researched in this thesis. In case of regression the procedure will be similar but different due to the fact that we have an infinite number of possible values for y_{new}. Detailed information about qualified predictions for regression can be found in [30].

Unfortunately, as mentioned in the previous section, it is impossible to calculate the typicalness level; it has long been known that it is computable only in the weakened sense of computability from above. To overcome this problem an interesting similarity has been found between the test for p-typicalness and well known in statistics p-values.

3.2.3 Statistical notion of p-values

The standard statistical theory (see, e.g., [28]) contains a definition of p-values from a measurable function $f : \Omega \to \Re$. Assuming that large values of f are significant, the *p-value* corresponding to a point $\omega_0 \in \Omega$ is defined to be

$$F(\omega_0) = P\{\omega : f(\omega) \geq f(\omega_0)\} \tag{2.6.5}$$

It is not difficult to see that this function $F(\omega_0)$ is a test for p-typicalness (2.6.4) assuming the second part of that definition is irrelevant, since tests of any interest in applications of statistics are always computable.

Hence, we may say that the test for p-typicalness is a universal version of the standard statistical notion of p-values. It turns out that we can calculate p-test for each sequence

$$z_i = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n), (x_{new}, Y_i), \; i = 1, \ldots, c$$

using the following function:

$$t(z_i) = \frac{\#\{j = 1, \ldots, n+1 : \alpha_j \geq \alpha_{n+1}\}}{n+1}, \tag{2.6.6}$$

where α_j is an individual strangeness measure of the example (x_j, y_j) and α_{n+1} is an individual strangeness measure of the new example (x_{new}, Y_i) with a postulated label Y_i. *Individual strangeness measure* intuitively is a kind of measure that shows how strange an example is with respect to the given sequence. It can be defined as follows (see, e.g., [29]).

A family of functions $\{A_l : l \in N\}$, where $A_l : Z^l \to \Re^l$ for all l, is called an *individual strangeness measure* if for any l:

a) any permutation $\pi : \{1, \ldots, l\} \to \{1, \ldots l\}$

b) any $(z_1, \ldots, z_l) \in Z^l$ where $z_k = (x_k, y_k)$ and $k = 1, \ldots, l$

c) any $(\alpha_1,\ldots,\alpha_l)\in\Re^l$

we have

$$(\alpha_1,\ldots,\alpha_l)=A_l(z_1,\ldots,z_l)\Rightarrow(\alpha_{\pi(1)},\ldots,\alpha_{\pi(l)})=A_l(z_{\pi(1)},\ldots,z_{\pi(l)}) \tag{2.6.7}$$

Once we have individual strangeness values obtained using any possible way, we can calculate a corresponding p-value for each postulated label of a new example that gives us a p-typicalness level, which in turn allows us to make a prediction and produce confidence and credibility as shown on Figure 3.2.2.1.

3.3 Learning Algorithms

There have been many bare prediction learning algorithms developed over the years. Some of them such as Least Squares and Ridge Regression are standard statistical tools that have been widely used even before the machine learning field existed. With the advances of the machine learning field these statistical methods have been adjusted to new requirements. For example, a dual version of Ridge Regression was introduced by Saunders et al. [57] in 1998 that allows to perform regression in a high-dimensional feature space that includes Least Squares as a special case. Other algorithms have been derived from other areas, such as Bayesian Networks come from the probability theory, Neural Networks from the nature observations etc.

Vapnik [6] defines three classes of computer learning problems: regression, pattern recognition and density estimation. Hence, all the learning algorithms are classified into one of these classes. Nevertheless, most of the algorithms can be adjusted to solve learning problems of more than one learning class. In this thesis we deal only with pattern recognition, therefore we are concentrating on two learning algorithms that are originally designed to work with this class of learning problems. These algorithms are Nearest-Neighbours and Support Vector Machines both of which are clear cut pattern recognition learning algorithms.

3.3.1 Nearest-Neighbours

Nearest-Neighbours learning algorithm is one of the oldest and most widely studied approaches to pattern recognition. Nearest-Neighbours classifier is so intuitive and naive that anyone could claim to be the inventor of this method. However, it seems like it was first described in 1951 by Fix and Hodges [32], then it was formally analysed in 1967 by Cover and Hart [33] and then investigated by many researches both theoretically and experimentally. Successful applications have been reported in a multitude of real-world domains (see e.g. Dasarathy [34] and the references therein).

The main and the original idea behind this method is as follows. Assuming we have a sequence of training data

$$z = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n),$$

where $x \in \Re^a$ is an object represented as a vector of size a, y is its label assigned by a supervisor and $y \in \{Y_1, \ldots, Y_c\}$ $Y_i, c \in N$ as we deal only with Pattern Recognition problems. When new object x_{new} arrives we calculate a set of distances between the new object and all objects in our training set using some distance function D such as $d_i = D(x_i, x_{new})$. The new object x_{new} will be assigned a label y_i which corresponds to the most similar object x_i according to the function D i.e. $d_i = d_{\min}$.

Euclidean distance is probably the most popular function D ever used:

$$D(x_1, x_2) = \sqrt{\sum_{j=1}^{a} (x_1^j - x_2^j)^2}, \qquad (2.7.1)$$

where x_1^j is an attribute j of the object x_1 and a is the total number of attributes in the given objects. In principal, however, any suitable function can be used to measure similarity between objects.

The original definition of the Nearest-Neighbours algorithm is almost never used. Real-world data usually contains a significant amount of noise which makes this method very unstable and prone to many errors. A simple modification significantly improves the algorithm. The improvement is achieved by considering k nearest neighbours that vote about the outcome. Instead of finding just one object with minimal distance to the new one, we obtain k such objects and then make the decision based on the most frequently occurred classification amongst these k examples. This algorithm is usually denoted as k-NN (k-Nearest Neighbours). Having analyzed k-NN under the theoretical assumptions of infinite numbers of training examples that populate the instance space with sufficient density, Cover and Hart [33] were able to prove that for large k, the k-NN classifier approaches the classification accuracy of idealized Bayesian classifiers.

3.3.2 Support Vector Machines

The Support Vector Machine (SVM) learning algorithm was invented by Vapnik and Chervonenkis [36] some time ago. However, only recently the idea has received the required attention and gathered the momentum for significant research. Many machine learning researchers started investigating this method after Vapnik published his book "The Nature of Statistical Learning Theory" in 1995. This book discusses the fundamental ideas which lie behind the statistical theory of learning and generalization. It considers learning as a general problem of function estimation based on empirical data and provides an in depth analysis of the new statistical learning algorithm called Support Vector Machine. The experimental results obtained and discussed in the book outperformed other known learning algorithms, which immediately attracted the attention of a very wide audience in this field.

Since then a considerable amount of theoretical and experimental work has been done showing the supreme performance of this method. For the pattern recognition case, SVMs have been used for isolated handwritten digit recognition (see e.g. Cortes and Vapnik [38]; Schölkopf, Burges and Vapnik [39]), object recognition (see, e.g., Blanz et al. [40]), speaker identification (see, e.g., Schmidt [41]), face detection in images (see, e.g., Osuna et al. [42]), text categorization (see, e.g., Joachims [43] and Lodhi et al. [44]) and even for biological tasks of protein classification and function prediction (see, e.g., Markowetz et al. [45] and Surkov et al. [46]). For the regression estimation case, SVMs have been compared on benchmark time series prediction tests (see, e.g., Muller et al. [47]; Mukherjee et al. [48]), the Boston housing problem (see, e.g., Drucker et al. [49]), and (on artificial data) on the PET operator inversion problem (Vapnik, Golowich and Smola [50]). In most of these cases, SVM generalization performance (i.e. error rates on test sets) either matches or is significantly better than that of competing methods. The use of SVMs for density estimation (Weston et al. [51]) and ANOVA decomposition (Stitson et al. [52]) has also been studied. The basic SVM contains no prior knowledge of the problem. For example, it would give the same results if the pixels were first permuted randomly with each image suffering the same permutation, an act of vandalism that would leave the best performing neural networks severely handicapped. This feature of the SVM could be good and bad at the same time. To research an extent of the significance of this feature some work has been done on incorporating prior knowledge into SVMs (see, e.g., Schölkopf et al. [53]; Burges [54]). Although

SVMs have good generalization performance, they can be very slow during the learning phase. This problem has also been addressed (see, e.g., Osuna and Girosi [55] and Stitson [56]).

The main idea behind the SVM learning algorithm is to learn data by constructing a hyper-plane, which separates the binary labelled training data with the maximum distance between the classes (so called "the maximal margin hyper-plane").

Let's assume we have a training data:

$$z = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n),$$

where $x \in \Re^k$ is an object represented as a vector of size k, y is its classification assigned by a supervisor and $y \in \{-1, +1\}$ in the simplest binary case. Our aim is to find a hyper-plane which separates this data such as all points for which $y = +1$ are on one side of the hyper-plane and all the points for which $y = -1$ are on the other side. Such hyper-plane must satisfy the conditions:

$$(\omega \cdot x_i) + b \geq 1, \qquad \text{if } y = +1$$

$$(\omega \cdot x_i) + b \leq -1, \qquad \text{if } y = -1$$

which is equivalent to

$$y_i \left[(\omega \cdot x_i) + b \right] \geq 1, \quad i = 1, \ldots, n. \tag{2.7.2}$$

However, there are a number of hyper-planes that might satisfy the condition (2.7.2). To find the optimal separating hyper-plane i.e. the maximal margin hyper-plane we need to find the hyper-plane that not only satisfies the condition (2.7.2) but also maximizes the minimum distance between the hyper-plane and any point in the training data. This distance is

$$\rho(\omega, b) = \min_{\{x_i | y_i = +1\}} \frac{\omega \cdot x_i + b}{\|\omega\|} - \max_{\{x_j | y_j = -1\}} \frac{\omega \cdot x_j + b}{\|\omega\|}. \tag{2.7.3}$$

From the equations (2.7.2) and (2.7.3) it follows that

$$\rho(\omega, b) = \frac{2}{\|\omega\|}. \tag{2.7.4}$$

Therefore to find the optimal separating hyper-plane we need to minimize

$$\frac{1}{2}\|\omega\| \text{ or } \frac{1}{2}\omega\cdot\omega$$

with respect to both the vector ω and scalar b subject to the constraint (2.7.2).

This optimization problem is hard to solve in practice, therefore another approach to solve it is to find the saddle point of the appropriate Lagrange functional

$$L(\omega,b,\alpha)=\frac{1}{2}\omega\cdot\omega-\sum_{i=1}^{n}\alpha_i\left[(\omega\cdot x_i+b)y_i-1\right], \qquad (2.7.5)$$

where α_i are Lagrange multipliers. The Lagrangian has to be minimized with respect to ω,b and maximized subject to $\alpha_i\geq 0$.

From the conditions, that must be satisfied by the solution in the saddle point, we obtain the following properties of the optimal separating hyper-plane (for more detailed derivation see, e.g., Vapnik [6] or Stitson [56])

1.

$$\sum_{i=1}^{n}\alpha_i y_i=0,\ \alpha_i\geq 0,\ i=1,\ldots,n \qquad (2.7.6)$$

2.

$$\omega=\sum_{i=1}^{n}\alpha_i y_i x_i,\ \alpha_i\geq 0,\ i=1,\ldots,n. \qquad (2.7.7)$$

Those vectors x_i for which the corresponding α_i are non-zero are called the Support Vectors, and they exactly identify the hyper-plane. By substituting equations (2.7.6) and (2.7.7) into the Lagrangian (2.7.5) we obtain the following functional

$$W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j). \qquad (2.7.8)$$

This functional has to be maximized subject to

$$\sum_{i=1}^{n} \alpha_i y_i = 0,\ \alpha_i \geq 0.$$

Once the solution to this problem is found in the form of a vector $\alpha^0 = (\alpha_1^0, \ldots, \alpha_n^0)$ the classification of a new example x_{new} can be found by

$$sign\left(\sum_{SVs} y_i \alpha_i^0 (x_i \cdot x_{new}) - b \right). \qquad (2.7.9)$$

Solving functional (2.7.8) allows to construct a linear decision surface in the original input space. However, it is often the case that the training data cannot be separated by the linear hyper-plane. To overcome this problem we map all the data points from the original input space X into some high-dimensional feature space H $\left(\mathrm{k} : X \to H\right)$

$$\mathrm{k}(x) = h,\ x \in X,\ h \in H$$

and find a linear optimal hyper-plane in the space H which corresponds to the non-linear decision surface in the input space X and therefore allows to separate the training data.

So, the functional (2.7.8) now becomes

$$W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \left(\mathrm{k}(x_i) \cdot \mathrm{k}(x_j)\right) \qquad (2.7.10)$$

and the decision function for the new example becomes

$$sign\left(\sum_{SVs} y_i \alpha_i^0 \left(\mathrm{k}(x_i)\cdot \mathrm{k}(x_{new})\right) - b\right) \tag{2.7.11}$$

Mapping data into a high-dimensional feature space by the function $k(x)$ and then finding the inner product in that space may be a very costly operation which growth with the dimensionality of the feature space. However, the nice thing about functionals (2.7.10) and (2.7.11) is that we do not even need to know the mapping function $\mathrm{k}(x)$ and do not need to translate each training vector into the feature space as long as we know

$$\mathrm{K}(x_i, x_j) = \mathrm{k}(x_i)\cdot \mathrm{k}(x_j) \tag{2.7.12}$$

This function is called Kernel and allows to find the inner product of two vectors in a feature space without actually performing the calculations in that space. Moreover, according to the Mercer Theorem any symmetric and non-negative function $\mathrm{K}(u,v)$ represents an inner product in some feature space hence can be used as a kernel.

3.4 Strangeness approximation

To move from the theoretical explanation of predicting with confidence and credibility to practical solutions the only problem left to solve is to find a way of obtaining individual strangeness measures. We are required to obtain an individual strangeness measure for each example in a given training set. A theoretical notion of individual strangeness measure was already described in Section 3.2.3. For practical purposes let us just say that this measure shows how strange each example is with respect to all other examples in a given set. By analysing existing bare prediction learning algorithms such as Nearest Neighbours and Support Vector Machines it becomes clear that they can be employed to produce such measures.

3.4.1 Using Nearest Neighbours

As we already described, the essence of the Nearest Neighbours algorithm is to find the closest example in the training set, take its label and assign it to the new example. Alternatively we find k such examples and assign the most frequent label amongst them to the new example. Assuming we have only two classes (binary case) it is easy to see that strangeness of an example can be quantified by comparing its average distance to examples of the same class with an average distance to examples of the opposite class. In fact an individual strangeness measure α_j of example (x_j, y_j) can be calculated as follows:

$$\alpha_j = \frac{\frac{1}{n^{y_j}} \sum_{i=1}^{n^{y_j}} D(x_j, x_i^{y_j})}{\frac{1}{n^{-y_j}} \sum_{i=1}^{n^{-y_j}} D(x_j, x_i^{-y_j})}, \qquad (2.8.1)$$

where $D(x_j, x_i^{y_j})$ is a distance between object x_j and the i_{th} object of same class y_j, n^{y_j} is the total number of examples of class y_j, n^{-y_j} is the total number of examples of the classes other than y_j, and $D(x_j, x_i^{-y_j})$ is the distance between object x_j and the i_{th} object of the classes other than y_j.

An interesting fact is that this is not the only way to calculate an individual strangeness measure using Nearest Neighbours. As well as using formula (2.8.1) we can use another one

$$\alpha_j = \frac{\sum_{i=1}^{k} D(x_j, x_i^{y_j})}{\sum_{i=1}^{k} D(x_j, x_i^{-y_j})}, \qquad (2.8.2)$$

where we find k nearest examples from both classes and find the ratio between their sums of distances. Both formulas are very natural because the strangeness of an example increases when the distance from the examples of the same class becomes bigger or when the distance from the examples of the opposite class becomes smaller. In fact using this idea there are possibilities for other ways of calculating strangeness using this learning algorithm.

3.4.2 Using Support Vector Machines

Following the idea of obtaining strangeness measures using the Nearest Neighbours algorithm it seems logical to see whether other learning algorithms can also be considered for this task. In Section 3.3.2 we already described Support Vector Machine learning algorithm. The essence of this algorithm is to map given examples into a high-dimensional space and then separate those examples in that space using a hyper plane (binary case again). The implementation of this algorithm involves solving an optimization problem the result of which is a set of Lagrange multipliers assigned to each example in the set. The resulting hyper plane is uniquely identified by these examples and by the Lagrange multipliers as shown in the formula (2.7.11) which is used to classify new examples.

The essential idea of the Support Vector Machine method is that after the optimisation process only a few examples will be assigned Lagrange multipliers which are not 0. The rest of the examples will have 0 multiplier and thus become irrelevant. The examples that are assigned non-zero multiplier called Support Vectors and they determine the shape of the hyper-plane and therefore the classification of a new example. A logical analysis of the Lagrange multipliers tells us that they can be used to indicate the strangeness of an example. In fact examples with Lagrange multipliers equal to zero are very typical i.e. not

strange at all, and the examples with non-zero multiplier are as strange as the value of its Lagrange multiplier. Therefore, Lagrange multipliers α produced by Support Vector Machines are straightforward individual strangeness measures that can be used in our algorithm for predicting with confidence and credibility.

Lagrange multipliers may be the perfect individual strangeness measures, however, as with the Nearest Neighbours algorithms this is not the only possibility. SVM learning algorithm separates given examples in a high-dimensional space with a hyper plane which means that examples of one class end up on one side of the hyper plane and examples of another class on the opposite side. It is obvious that the closer an example lies to the hyper plane the stranger it is. Vice versa a good typical example should be as far from the hyper plane on the right side as possible. The result is that we can use a signed distance from the hyper plane as a reversed measure of strangeness. In this case we cannot use a signed distance directly because it is a reversed measure of strangeness. More accurately a way to calculate an individual strangeness measure would be

$$\alpha = -h,$$

where h is a signed distance from the hyper plane.

One problem with using signed distance from a hyper plane is that its large value does not necessarily mean an ever decreasing strangeness of an example. Examples that are similar to each other would be situated in a fairly close area in a high-dimensional space. An example with a very large distance from a hyper plane which is also very far from most of the other examples may be strange too, even if it is on the right side of the hyper plane. This indicates the limitation of this method which is not present when we use Lagrange multipliers as our strangeness measures because Lagrange multipliers cannot be less than zero by definition. The limitation, however, can be overcome by introducing some sort of penalties if a distance from a hyper plane significantly exceeds an average distance amongst other examples of the same class. Alternatively, there may be other monotone-decreasing functions of h that could produce better results. We are not considering other functions in this thesis however, this could be a subject for further research in order to improve performance.

3.4.3 Using Kolmogorov complexity approximation

In the previous two sections we described how standard bare prediction learning algorithms such as SVM and Nearest Neighbours can be used to produce individual strangeness measures which in turn are used to produce p-values followed by the confidence and credibility. According to Figure 3.2.2.1, if we could produce typicalness level of a sequence, then we could easily calculate prediction, confidence and credibility. Formula 2.4.3 in Section 2.4 shows how typicalness deficiency could be obtained straight from the Kolmogorov complexity without the need of p-values. Kolmogorov complexity itself is not computable however, it is possible that it could be approximated in some practical cases.

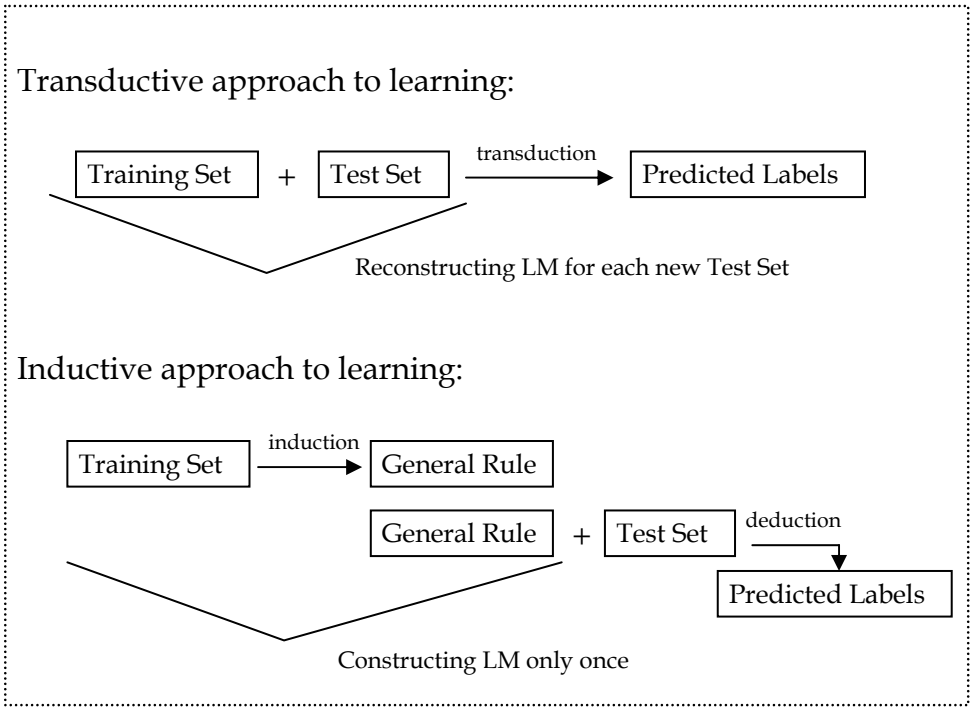
Kolmogorov complexity of an object is the length of a program on a universal Turing machine that reproduces this object. If our object is simply a symbol then our data is a sequence of symbols, and we can approximate its Kolmogorov complexity by finding the length of this sequence after it is compressed by any of the known compression algorithms. The idea is that the compressed sequence is effectively a program that can be used to reproduce the original sequence hence its length can be a measure of Kolmogorov complexity. Now we have a formula that can be used to produce typicalness deficiency of a sequence of symbols:

$$d(x) = l(x) - CL(x),$$

where $l(x)$ is the length of the original sequence x and $CL(x)$ is the length of the sequence x after compression.

This approach to calculating typicalness deficiency is the least complicated, however it is also least practical because most of the data cannot be easily represented in a sequence of symbols suitable for compression. Also compression algorithms are not exactly universal Turing machines and they provide only a very crude approximation to the Kolmogorov complexity. Nevertheless this approach can potentially be used successfully on some real-world problems.

Figure 3.5.1.1: Comparison between Inductive and Transductive approaches to learning.



3.5 Transductive Confidence Machine

Transductive Confidence Machine (TCM) is the first attempt to practically produce Qualified Predictions using the theory described in Section 3.2. The idea of transductive inference is shown on Figure 3.5.1.1 in the top half. We combine a given training set with a test set, and using a transductive method we predict labels for the training set without deriving an intermediate rule. This approach seems a natural way forward in a practical implementation of the ideal method of prediction with confidence and credibility described in Section 3.2. Saunders et al. [58] define TCM which uses Support Vector Machine to produce individual strangeness measures in a case of binary classification. In this section we will look even deeper into the idea of TCM by analysing how TCM can be constructed using Nearest Neighbour learning algorithm as well as Support Vector Machine or any other bare prediction learning machine. We will also investigate the validity issues that become apparent in a case of multi-class pattern recognition.

3.5.1 Binary TCM

As with any other supervised learning algorithm theory we start from the simplest case of binary pattern recognition. In this case the learning task is as follows. We are given a set of training examples

$$z_{train} = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n),$$

where $x \in \Re^k$ is an object represented as a vector of size k, y is its classification assigned by a supervisor and $y \in \{-1, +1\}$. The machine is given a set of test examples

$$z_{test} = (xt_1, \cdot), (xt_2, \cdot), \ldots, (xt_{nt}, \cdot),$$

where $x \in \Re^k$ is an object represented as a vector of size k, but its classification is unknown and required to be predicted.

Following the qualified predictions theory explained in Section 3.2 and using transductive approach we construct the following algorithm displayed on Figure 3.5.1.2.

Figure 3.5.1.2: Possible binary TCM procedure for multiple test examples (may be invalid).

1. Concatenate training and test sets into one set

$$z = (z_{train}\ z_{test}) = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n), (xt_1, Y_1), (xt_2, Y_2), \ldots, (xt_{nt}, Y_{nt})$$

2. Choose a combination of possible postulated labels:

$$Y_1, Y_2, \ldots, Y_{nt} \in \{-1, +1\}$$

3. Obtain an individual strangeness measure α_i for each example in the set z by applying an individual strangeness measure function A to the set z:

$$A(z) = (\alpha_1, \ldots, \alpha_{n+nt})$$

4. Calculate p-value for each test example using the following formula

$$\text{p-value}_{xt_i} = \frac{\#\left\{ j = 1, \ldots, n+nt : \alpha_j \geq \alpha_i \right\}}{n+nt},\ i = 1, \ldots, nt$$

5. Calculate the p-value for the whole test set as an average of p-values of all examples in the test set:

$$\text{p-value}_t = \frac{1}{nt} \sum_{i=1}^{nt} \text{p-value}_{xt_i}$$

6. Repeat the procedure from Step 2 until all combinations of postulated labels are tried and all p-value_t are found, where $t = 1, \ldots, 2^{nt}$ in our binary case.

7. Out of all found p-values identify the largest one

$$\text{p}_1 = \max\{\text{p-value}_t\}$$

and the second largest one

$$\text{p}_2 = \max\left(\{\text{p-value}_t\} \setminus \text{p}_1\right)$$

where $t = 1, \ldots, 2^{nt}$.

8. The prediction will be the combination of postulated labels that corresponds to the largest p-value p_1.

9. Confidence of the prediction will be $1 - \text{p}_2$.

10. Credibility of the prediction will be p_1.

The drawback of the procedure above is that it requires to perform exactly the same operation of calculating p-values 2^{nt} times which grows exponentially with the number of test examples. Also, the validity of this procedure is difficult to prove. In practice, however, it is much more efficient and probably better to consider each test example individually. In this case we add test examples one by one from the test set to the given training set, we skip Step 5 and we have only two possible combinations of the postulated label. Thus we only need to perform $2nt$ of such operations, which is much more favourable. Also we obtain measures of confidence and credibility for each individual example rather than for the whole test set which is much more useful as well. At last, by considering test examples one by one we make our predictions purely on the information we have within the training set only. This is in contrast to the procedure which uses not only the information within the training set but also the information within the test set which is not classified yet. The updated procedure for predicting with confidence and credibility based on individual test examples is shown on Figure 3.5.1.3.

Figure 3.5.1.3: Binary TCM procedure for individual test examples.

1. Concatenate the training set and new example into one set

$$z = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n), (x_{new}, Y_{new})$$

2. Assign a first possible postulated label for the new example

$$Y_{new} = -1$$

3. Obtain an individual strangeness measure α_i for each example in the set z by applying an individual strangeness measure function A to the set z:

$$A(z) = (\alpha_1, \ldots, \alpha_{n+1})$$

4. Calculate p-value for the new example using the following formula

$$\text{p-value}_t = \frac{\#\left\{ j = 1, \ldots, n+1 : \alpha_j \geq \alpha_{n+1} \right\}}{n+1}$$

5. Assign the second possible postulated label for the new example

$$Y_{new} = +1$$

and repeat the procedure once again from Step 3.

6. Out of two found p-values identify the larger one

$$\mathrm{p}_1 = \max\{\text{p-value}_t\}$$

and the second largest one (the other one in this case)

$$\mathrm{p}_2 = \max\left(\{\text{p-value}_t\} \setminus \mathrm{p}_1\right)$$

where $t = 1, 2$.

7. The prediction will be the postulated label that corresponds to the largest p-value p_1.
8. Confidence of the prediction will be $1 - \mathrm{p}_2$.
9. Credibility of the prediction will be p_1.

Both procedures above explain how predictions with confidence and credibility can be done in the binary case. The first procedure should be used if we are required to make predictions on the whole test set at once taking into account information contained in every test example for predicting other test examples. The second procedure should be used when we are allowed to make predictions without knowing the information contained in all other test examples. For most practical applications the second procedure is the most appropriate one in terms of the efficiency and logic. This is why all our algorithms will be implementing this approach rather than the first one.

To obtain a valid test for p-typicalness our test must satisfy condition one in the definition of Martin-Löf p-typicalness (2.6.4). It is proven that every function for finding p-values (2.6.6) obtained using a computable individual strangeness measure will satisfy the p-typicalness condition one (see Gammerman and Vovk [18]). In Section 3.4 we explained how Nearest Neighbours and Support Vector Machines can potentially be used to produce individual strangeness measures. However, we have not addressed the following question yet - combined with the procedure above for binary TCM would individual strangeness measures obtained from NN and SVM provide us with a valid test for p-typicalness?

To answer this question let us start from the case of using Nearest Neighbours learning algorithm. Assume we use the binary TCM procedure for single test example and the function (2.8.1) for calculating strangeness. According to the definition (2.8.1) the order of examples does not make any difference because to find strangeness α for postulated label +1 we take k nearest in terms of distance examples of class +1, then find the sum of distances between one example and the k nearest ones. The same sum found between the example and the k nearest ones from the opposite class. Regardless of the order of examples in the training set for each example we will always have exactly the same k nearest ones from the same and from the opposite classes, which will always result in the same strangeness value for each example. Even if we permute the dataset randomly and the last example will not be the new example, still every example will have exactly the same k nearest neighbours, which will always result in the same strangeness. Therefore, the procedure for binary TCM with a single test example and Nearest Neighbours learning algorithm produces a valid p-typicalness test.

In the case of SVM we train the learning machine with the set z two times (one for each postulated label for the new example). The result of the optimisation problem does not depend on the order of examples in the training set, hence each one of them will be assigned the same strangeness value regardless of the order of examples in the set z. When we permute the dataset randomly the last example in the set might have the classification different to the new example's postulated label. This would result in the classification invert for all examples before the optimization starts. However, because we have only two classes the hyper-plane will be exactly the same, and all examples would obtain exactly the same Lagrange multipliers serving as individual strangeness measures. Therefore, SVM will also produce valid p-typicalness test combined with the binary TCM procedure for a single test example.

3.5.2 Multi-class TCM

In the previous section we presented how Transductive Confidence Machine can be constructed in the simplest case when our data has only two classifications +1 and –1. In the real-world problems like this rarely take place. Most of the available data is multi-class for example images of digits, images of letters, patient illnesses data etc. Therefore, to apply TCM successfully to the real data it must be able to cope with the multi-class requirement.

The multi-class problem does not have a unique solution as it has been shown in the variety of multi-class implementations of bare prediction machine learning algorithms. For example, in the Support Vector Machine alone there are three major ways of solving the multi-class problem. However, all of these methods essentially begin by splitting the multi-class problem into a number of binary problems. Each of the binary problems is solved independently and then a voting takes place to decide which offers the best solution. The differences between multi-class approaches lie in the way the original problem is split into binary problems and in the voting mechanism. Examples of the most popular decomposition approaches are:

a) One-Against-All; where each binary problem contains examples of one class as having classification +1 and all other classes as –1.
b) One-Against-One; where each binary problem contains examples of one class as having classification +1 and one another class as –1. In this case we obtain many more binary problems compared to the previous approach.

Voting mechanisms also can be implemented in different ways. Mainly the voting is done by employing some functions for example Hamming error-correcting codes, VC dimension or other generalisation ability measures which can favour one solution against others. Voting also depends on the learning algorithm used. In Support Vector Machine we mainly use a distance from hyperplane, in *k*-Nearest Neighbours it is a distance to *k* nearest examples.

To build a Transductive Confidence Machine for multi-class data it is natural to employ the same kind of decomposition into the multiple binary problems. By assuming we deal with a whole set of test examples individually for each ex-

ample one by one, the procedure for multi-class TCM naïvely could be a slightly modified version of the binary TCM as shown in Figure 3.5.2.1.

However, if we analyse the validity of the naïve procedure shown on Figure 3.5.2.1 it becomes clear that the straightforward approach of adapting binary TCM to multi-class TCM is not valid. Experiments using this procedure produce graphs similar to Figure 3.5.2.2. This graph shows the quantity of p-values in ascending order on the X-axis and actual p-values on the Y-axis. In an ideal situation the graph should look like Figure 3.5.2.3. In all other cases the graph should be below the diagonal line. The fact that Figure 3.5.2.2 shows the obtained graph above the diagonal line means that the algorithm has a flaw somewhere because the Martin-Löf condition (2.6.4) is not satisfied.

Figure 3.5.2.1: An invalid multi-class TCM.

1. Assign a possible postulated label for the new example
$$Y_t = Y_1, \ldots, Y_c$$
where *c* is a number of classes in the given data.

2. Concatenate the training set and the new example into one set
$$z = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n), (x_{new}, Y_t).$$

2. Obtain new set z_t by changing classification of each example x_i in the set z to +1 if $y_i = Y_t$ and to -1 otherwise, $i = 1, \ldots, n+1$.

3. Obtain an individual strangeness measure α_i for each example in the set z_t.

4. Calculate p-value for the new example using the following formula
$$\text{p-value}_t = \frac{\#\left\{ j = 1, \ldots, n+1 : \alpha_j \geq \alpha_{n+1} \right\}}{n+1}$$

5. Repeat the procedure from Step 1 until all possible postulated labels are tried and we obtain *c* p-values, one for each postulated label.

6. Out of *c* found p-values identify the largest one
$$\text{p}_1 = \max\{\text{p-value}_t\},\ t = 1, \ldots, c$$
and the second largest one
$$\text{p}_2 = \max\left(\{\text{p-value}_t\} \setminus \text{p}_1\right),\ t = 1, \ldots, c$$

8. The prediction will be the postulated label that corresponds to the largest p-value p_1.

9. Confidence of the prediction will be $1 - \text{p}_2$.

10. Credibility of the prediction will be p_1.

Figure 3.5.2.2: P-values produced by an invalid TCM.

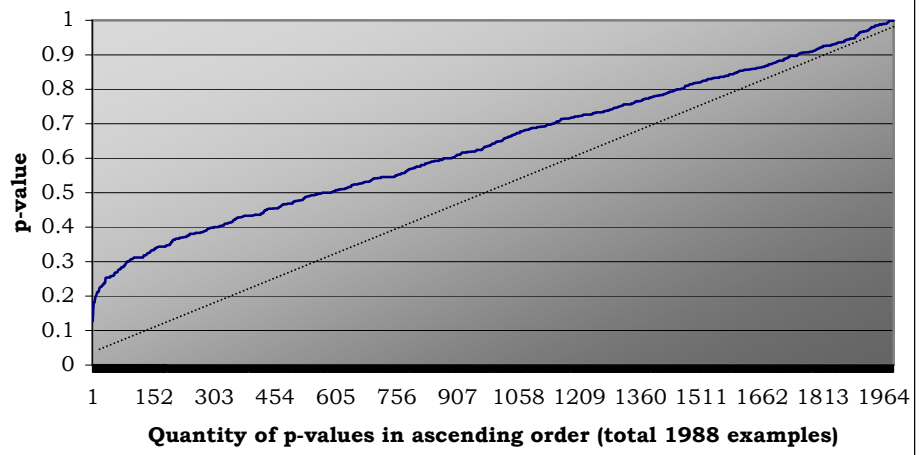
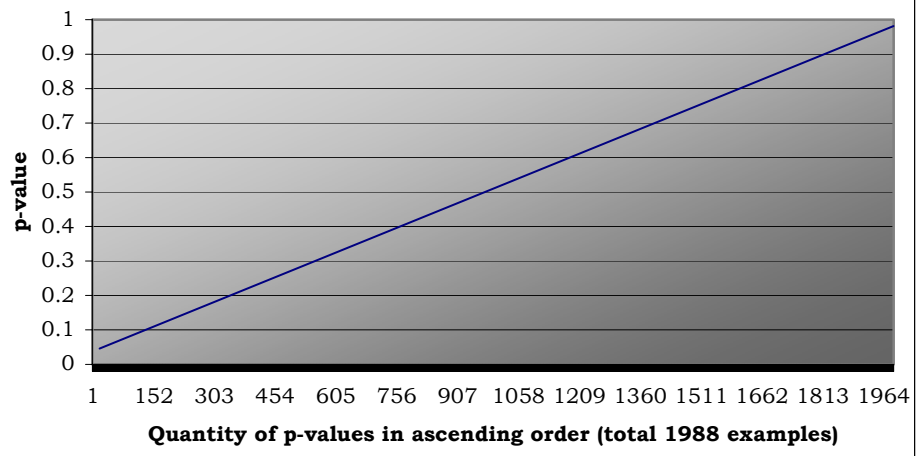


Figure 3.5.2.3: P-values from an ideal Confidence Machine.



To analyse where the potential flaw might be, we should look at Step 4 on Figure 3.5.2.1 where we obtain individual strangeness measures for each example. The definition of individual strangeness function (2.6.7) states that any permutation of the data produces exactly the same individual strangeness values. Following the procedure above we assign a postulated label to the last example (which is our new example) in the set, change every example's classification to +1 and –1 according to their real classification and then supposedly find an individual strangeness measure for every example. However, if we permute the data set, the last example could have different classification compared to the postulated classification of the new example which would result in a completely different set of strangeness measures. This contradicts the definition (2.6.7) and makes the algorithm invalid.

To produce valid individual strangeness measures the procedure for multi-class TCM should be modified considerably. A procedure displayed on Figure 3.5.2.4 is an example of such algorithm that constructs a valid individual strangeness measure function. This procedure implements one-against-all approach and voting is done using obtained p-values. The postulated label that produces the largest p-value wins. The algorithm assures that any permutation of the data produces exactly the same set of strangeness measures, however it comes at a significant cost. According to this algorithm multi-class TCM would require an individual strangeness measure function A to be applied c^2 times on a given data set, where c is a number of classes. Recalling that applying function A in practice means training one of the bare prediction learning algorithms on the given data, we may say that the algorithm is extremely inefficient compared to a binary TCM. Nevertheless, careful analysis of this procedure reveals a large number of redundant operations during the steps 3.1 to 3.5, therefore in practice we only need to perform $2c$ training operations. A simple optimization makes the procedure very efficient without any loss of quality.

One-against-all is the most popular way to solve multi-class problems because it gives us the best trade-off between the clarity, computational complexity and overall accuracy. However, one-against-one approach may also be possible by using the following procedure shown on Figure 3.5.2.5, but the validity of this procedure is not clear.

Figure 3.5.2.4: Proper (valid) multi-class TCM for one-against-all approach.

1. Assign a possible postulated label for the new example

   $Y_t = Y_1, \ldots, Y_c$, where *c* is a number of classes in the given data.

2. Concatenate the training set and the new example into one set

   $z_t = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n), (x_{new}, Y_t)$.

3. Obtain an individual strangeness measure α_i for each example in the set z_t using the following procedure:

   3.1 Select classification $Y_u = Y_1, \ldots, Y_c$

   3.2 Obtain new set $z_t^{(u)}$ by changing classification of each example x_i in the set z_t to +1 if $y_i = Y_u$ and to –1 otherwise, $i = 1, \ldots, n+1$.

   3.3 Apply individual strangeness measure function A to the set $z_t^{(u)}$

   $$A(z_t^{(u)}) = (\alpha_1^{(u)}, \ldots, \alpha_{n+1}^{(u)})$$

   3.4 Repeat the procedure from Step 3.1 until all classifications are tried.

   3.5 Individual strangeness measure α_i for example i, where $i = 1, \ldots, n+1$ is $\alpha_i^{(u)}$ obtained in Step 3.3 when $Y_u = y_i$ i.e. when the label selected in Step 3.1 is equal to the example's label.

4. Calculate p-value for the new example using the following formula

   $$\text{p-value}_t = \frac{\#\left\{ j = 1, \ldots, n+1 : \alpha_j \geq \alpha_{n+1} \right\}}{n+1}.$$

5. Repeat the procedure from Step 1 until all possible postulated labels are tried and we obtain *c* p-values, one for each postulated label.

6. Out of *c* found p-values identify the largest one

   $$\text{p}_1 = \max\{\text{p-value}_t\},\ t = 1, \ldots, c$$

   and the second largest one

   $$\text{p}_2 = \max\left(\{\text{p-value}_t\} \setminus \text{p}_1\right),\ t = 1, \ldots, c$$

7. The prediction will be the postulated label that corresponds to the largest p-value p_1.

8. Confidence of the prediction will be $1 - \text{p}_2$.

9. Credibility of the prediction will be p_1.

Figure 3.5.2.5-part 1: Multi-class TCM for one-against-one approach (may be invalid).

1. Assign a possible postulated label for the new example

   $Y_t = Y_1, \ldots, Y_c$, where *c* is a number of classes in the given data.

2. Concatenate the training set and new example into one set

   $z_t = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n), (x_{new}, Y_t)$.

3. Select a possible classification Y_m not equal to the Y_t selected in the Step 1 such as

   $$m = (1, \ldots, c) \setminus t$$

4. Obtain new set $z_t^{(m)}$ by taking from the set z_t only the examples with classification $y_i = Y_t$ or $y_i = Y_m$, making sure the new example is the last one in the new set $z_t^{(m)}$.

5. Obtain an individual strangeness measure α_i for each example in the set $z_t^{(m)}$ using the following procedure:

   5.1 Select classification $Y_u = Y_t, Y_m$.

   5.2 Obtain new set $z_t^{(m^u)}$ by changing classification of each example x_i in the set $z_t^{(m)}$ to +1 if $y_i = Y_u$ and to –1 otherwise, $i = 1, \ldots, \left| z_t^{(m)} \right|$.

   5.3 Apply individual strangeness measure function A to the set $z_t^{(m^u)}$

   $$A(z_t^{(m^u)}) = (\alpha_1^{(m^u)}, \ldots, \alpha_l^{(m^u)}),\ l = \left| z_t^{(m)} \right|.$$

   5.4 Repeat the procedure from Step 5.1 until both classifications are tried.

   5.5 Individual strangeness measure α_i for example x_i, where $i = 1, \ldots, \left| z_t^{(m)} \right|$ is $\alpha_i^{(m^u)}$ obtained in Step 5.3 when $Y_u = y_i$ i.e. when the label selected in Step 5.1 is equal to the example's label.

6. Calculate p-value for the new example using the following formula:

   $$\text{p-value}_m = \frac{\#\left\{ j = 1, \ldots, l : \alpha_j \geq \alpha_l \right\}}{l},\ l = \left| z_t^{(m)} \right|.$$

Figure 3.5.2.5-part 2: Multi-class TCM for one-against-one approach (may be invalid).

7. Repeat the procedure from Step 3 until all possible classifications Y_m are tried and we obtain $c-1$ p-values, one for each class against the postulated label.

8. Calculate an average p-value for a postulated label Y_t

$$\text{p-value}_t = \frac{\sum_m \text{p-value}_m}{c-1},\ m = (1,\ldots,c)\setminus t$$

9. Repeat the procedure from Step 1 until all possible postulated labels are tried and we obtain *c* average p-values, one for each postulated label.

10. Out of *c* found p-values identify the largest one

$$\text{p}_1 = \max\{\text{p-value}_t\},\ t = 1,\ldots,c$$

and the second largest one

$$\text{p}_2 = \max\left(\{\text{p-value}_t\}\setminus \text{p}_1\right),\ t = 1,\ldots,c$$

11. The prediction will be the postulated label that corresponds to the largest p-value p_1.

12. Confidence of the prediction will be $1-\text{p}_2$.

13. Credibility of the prediction will be p_1.

As seen from the procedure on Figure 3.5.2.5, one-against-one approach deals with smaller data sets each time it solves a binary sub-problem. However, it requires many more repeating operations. Depending on the size of the data and the complexity of the single binary problem, it might require less or more computational resources compared to the one-against-all approach. Overall, this approach is less commonly used and in our experiments we will be using the one-against-all method only; also because it is not clear whether the procedure shown on Figure 3.5.2.5 is valid or not.

3.5.3 Regression TCM

In this thesis we deal only with pattern recognition cases. Nevertheless, it is quite possible to extend the idea of Transductive Confidence Machine into regression problems. The difference between pattern recognition and regression lies in the fact that we have an infinite number of possible classifications in a case of regression. We still have a sequence of data

$$z = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n),$$

where $x \in \Re^k$ is an object represented as a vector of size k, y is its classification assigned by a supervisor, but now we have $y \in \Re$ rather than $y \in \{Y_1, \ldots, Y_c\}$ $Y_i, c \in N$.

Unfortunately, we cannot try all possible postulated labels for a new example like it is done in the Pattern Recognition TCM. Therefore, another approach is required. An example of the regression TCM is presented by Nouretdinov et al. [62]. In this paper they build a Regression Confidence Machine using Dual Ridge Regression learning algorithm as the basis. By fixing a significance level they obtain predictive intervals as the output.

3.6 Inductive Confidence Machine

Transductive Confidence Machine does produce a prediction along with measures of confidence and credibility for each prediction. We described the exact procedures for TCM above which tell us that TCM has a significant drawback. The drawback is that we have to obtain a complete set of individual strangeness measures for all examples in the training set every time we get new example. The procedure of obtaining individual strangeness measures in fact is the procedure of learning because we use bare prediction learning algorithms for this task. In a case of multi-class problems we have to perform as many as $2c$ learning operations on the whole training set just to classify one new example. Commonly, learning algorithms are quite slow during the learning phase, hence classification of a new example using TCM might take some time. If we have more than one example to predict then the whole procedure has to be repeated as many times as there are examples. This means that the overall procedure to predict real data may take so long that it would be inappropriate for any practical use. The main purpose of Confidence Machine is to help people to predict with confidence and credibility and be able to perform this task on large data files within a reasonable amount of time. Therefore another type of confidence machine is required.

The key process within the TCM that makes it slow is the process of obtaining individual strangeness measures. To construct a faster machine, a natural idea is to create a system which can obtain individual strangeness measures without re-learning the training data every time a new example arrives. The possibility of achieving this can be found in using an inductive approach rather than transductive. The difference between them can be seen on Figure 3.5.1.1. Inductive approach implies creating a general rule which can then be used to classify new examples without re-learning. A similar concept can be used within the confidence machine which results in another type of confidence machine called Inductive Confidence Machine. It is also based on obtaining individual strangeness measures, however, in this case we can obtain them in advance along with a general rule to calculate individual strangeness measure of a new example. This would allow us to classify new examples without re-learning the data each time.

3.6.1 Binary ICM

Following the same pattern used for describing Transductive Confidence Machine, we start introducing Inductive Confidence Machine from the simplest binary case. Exploring the idea of inductive learning a naïve approach to ICM would seem to be as shown on Figure 3.6.1.1.

This algorithm seems like an inductive confidence machine, however, it will produce invalid p-values, therefore it cannot be used. The reason why it would produce invalid p-values is the same as described in Section 3.5.2 when we investigated a naïve approach to multi-class Transductive Confidence Machine. If we randomly permute the set

$$z = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n), (x_{new}, y_{new})$$

then our last example would not necessarily be our new example, therefore the general rules D^+ and D^- would have to be recalculated. This would result in completely different individual strangeness measures that would contradict the validity requirement.

To create a valid Inductive Confidence Machine we need to overcome the problem of individual strangeness measures dependency on the original training set. One possible solution is to split a given training set into two subsets before proceeding further as shown on Figure 3.6.1.2. After this initial step is completed the following algorithm displayed on Figure 3.6.1.3 could be used to produce a valid binary Inductive Confidence Machine. This procedure is valid because we use a general decision rule which is independent of the data to produce individual strangeness measures from. The data to produce individual strangeness measures from is obtained by concatenating a calibration set with the new example. No matter how many times we randomly permute this data, the decision rules are still the same and they will produce exactly the same individual strangeness measures that would make this algorithm a valid one.

Figure 3.6.1.1: Naïve (invalid) procedure for binary ICM.

1. Assume we have a training set:
$$z = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n).$$

2. Obtain an individual strangeness measure $\alpha_i^{(+)}$ for each example x_i in the set z by applying an individual strangeness measure function A^+ to the set z for identifying examples with $y_i = +1$:
$$A^+(z) = (\alpha_1^{(+)}, \ldots, \alpha_n^{(+)})$$
such as it also produces a general rule D^+ which can re-produce individual strangeness measure for any example:
$$\alpha_i^{(+)} = D^+(x_i),\ i = 1, \ldots, n.$$

3. Repeat Step 2 with A^- to obtain individual strangeness measures $\alpha_i^{(-)}$ and a general rule D^-, $i = 1, \ldots, n$.

4. When new example x_{new} arrives find p-values using the following procedure:

   4.1 Apply D^+ to x_{new} in order to obtain $\alpha_{new}^{(+)}$: $\alpha_{new}^{(+)} = D^+(x_{new})$.

   4.2 Apply D^- to x_{new} in order to obtain $\alpha_{new}^{(-)}$: $\alpha_{new}^{(-)} = D^-(x_{new})$.

   4.3 $$\text{p-value}_+ = \frac{\#\left\{ j = 1, \ldots, n+1 : \alpha_j^{(+)} \geq \alpha_{new}^{(+)} \right\}}{n+1}$$

   4.4 $$\text{p-value}_- = \frac{\#\left\{ j = 1, \ldots, n+1 : \alpha_j^{(-)} \geq \alpha_{new}^{(-)} \right\}}{n+1}$$

5. Out of two found p-values identify the largest one
$$\text{p}_1 = \max\{\text{p-value}_+, \text{p-value}_-\}$$
and the second largest one
$$\text{p}_2 = \max\left(\{\text{p-value}_+, \text{p-value}_-\} \setminus \text{p}_1\right).$$

6. The prediction will be the postulated label that corresponds to the largest p-value p_1.

7. Confidence of the prediction will be $1 - \text{p}_2$.

8. Credibility of the prediction will be p_1.

Figure 3.6.1.2: Splitting the original training data into the Proper Training Set and Calibration Set for ICM.

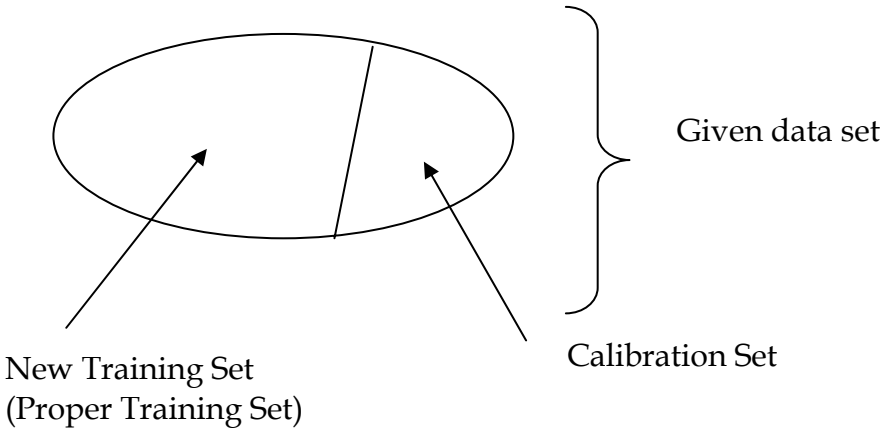


Figure 3.6.1.3-part 1: Proper (valid) procedure for binary ICM.

1. Assume we are given a training set:

$$z = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n).$$

2. The given training set z is randomly split into two subsets as shown on the Figure 3.6.1.2:
   a) the *proper training set* of size l_1 to construct a general decision rule from;
   b) the *calibration set* of size l_2;

   the only information used at this stage should be the classifications $y_1, \ldots, y_n$ of the elements of the training set.

3. Assuming we have only two classes +1 and –1 (binary case), we construct two decision rules D^+ and D^- using strangeness function A and the proper training set.

4. Apply both rules D^+ and D^- to all corresponding examples from the calibration set to obtain two sets of individual strangeness measures:

$$\alpha_1^{(+)}, \ldots, \alpha_{l_2^{(+)}}^{(+)} \text{ and } \alpha_1^{(-)}, \ldots, \alpha_{l_2^{(-)}}^{(-)}$$

   where $l_2^{(+)}$ and $l_2^{(-)}$ are the number of examples with $y_i = +1$ and $y_i = -1$ correspondingly in the calibration set, $i = 1, \ldots, l_2$.

Figure 3.6.1.3-part 2: Proper (valid) procedure for binary ICM.

5. When new example x_{new} arrives find p-values using the following procedure:

   5.1 Apply D^+ to x_{new} in order to obtain $\alpha_{new}^{(+)}$: $\alpha_{new}^{(+)} = D^+(x_{new})$.

   5.2 Apply D^- to x_{new} in order to obtain $\alpha_{new}^{(-)}$: $\alpha_{new}^{(-)} = D^-(x_{new})$.

   5.3 Compute $\text{p-value}_+ = \dfrac{\#\left\{ j = 1,\ldots, l_2^{(+)} + 1 : \alpha_j^{(+)} \geq \alpha_{new}^{(+)} \right\}}{l_2^{(+)} + 1}$.

   5.4 Compute $\text{p-value}_- = \dfrac{\#\left\{ j = 1,\ldots, l_2^{(-)} + 1 : \alpha_j^{(-)} \geq \alpha_{new}^{(-)} \right\}}{l_2^{(-)} + 1}$.

6. Out of two found p-values identify the largest one

   $$\text{p}_1 = \max\{\text{p-value}_+, \text{p-value}_-\}$$

   and the second largest one

   $$\text{p}_2 = \max\left(\{\text{p-value}_+, \text{p-value}_-\} \setminus \text{p}_1\right).$$

7. The prediction will be the postulated label that corresponds to the largest p-value p_1.

8. Confidence of the prediction will be $1 - \text{p}_2$.

9. Credibility of the prediction will be p_1.

10. Repeat the procedure from Step 5 for all new examples in a test set.

3.6.2 Multi-class ICM

We have already mentioned that most of the real-world problems are not binary ones. In Section 3.5.2 we explained how the multi-class Transductive Confidence Machine can be constructed. The same ability to deal with multi-class problems is required from the Inductive Confidence Machine. Fortunately, a transition from the binary ICM to multi-class ICM is much simpler than it is for TCM. The procedure shown in Figure 3.6.1.3 needs only a slight modification to obtain a multi-class ICM as presented on Figure 3.6.2.1.

Figure 3.6.2.1-part 1: Proper (valid) procedure for multi-class ICM.

1. Assume we are given a training set:

$$z = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$$

2. The given training set z is randomly split into two subsets as shown on the Figure 3.6.1.2:
   a) the *proper training set* z_1 of size l_1 to construct a general decision rule from;
   b) the *calibration set* z_2 of size l_2;

   the only information used at this stage should be the classifications $y_1, \ldots, y_n$ of the elements of the training set.

3. Assuming we have c classes, $Y_t = Y_1, \ldots, Y_c$, we construct a decision rule D^t for each classification $t = 1, \ldots, c$ using strangeness function A and the proper training set.

4. Apply each rule D^t to all examples with label $y_i = Y_t$ from the calibration set to obtain c sets of individual strangeness measures:

$$D^t(z_2^{(t)}) = \alpha_1^{(t)}, \ldots, \alpha_{l_2^{(t)}}^{(t)}, \; t = 1, \ldots, c,$$

   where $z_2^{(t)}$ is a subset of the calibration set that contains all the examples with labels Y_t and $l_2^{(t)}$ is its cardinality $l_2^{(t)} = \left| z_2^{(t)} \right|$.

Figure 3.6.2.1-part 2: Proper (valid) procedure for multi-class ICM.

5. When new example x_{new} arrives find p-values using the following procedure:

   5.1 Apply D^t to x_{new} in order to obtain $\alpha_{new}^{(t)}$: $\alpha_{new}^{(t)} = D^t(x_{new})$.

   5.2 Compute $\text{p-value}_t = \dfrac{\#\left\{ j = 1,\ldots,l_2^{(t)}+1 : \alpha_j^{(t)} \geq \alpha_{new}^{(t)} \right\}}{l_2^{(t)}+1}$.

   5.3 Repeat from Step 5.1 until all c p-values are obtained, $t = 1,\ldots,c$.

6. Out of c found p-values identify the largest one

$$p_1 = \max\{\text{p-value}_t\}$$

   and the second largest one

$$p_2 = \max\left(\{\text{p-value}_t\} \setminus p_1\right).$$

7. The prediction will be the label that corresponds to the largest p-value p_1.

8. Confidence of the prediction will be $1 - p_2$.

9. Credibility of the prediction will be p_1.

10. Repeat the procedure from Step 5 for all new examples in a test set.

3.6.3 Regression ICM

In the same way as Transductive Confidence Machine can be extended to deal with the regression, Inductive Confidence Machine can also be adjusted for these kind of problems. We do not deal with regression in this thesis, however the details about the ICM for regression can be found in Papadopoulos et al. [63]. Regression ICM employs the same initial step of splitting a given training set into a proper training set and a calibration set. The rest is similar to the Regression TCM whereby we specify a confidence level C (sometimes it is also represented as a significance level = 1 – confidence level) and then find a predictive region such that one can be $C\%$ confident that the label of the new example will be covered by that predictive region.

3.7 Comparing performance of Confidence Machines

Comparing learning algorithms that only produce bare predictions is an easy task because the prediction error rate is the most natural performance measure. So far all the machine learning algorithms have been compared by running them on exactly the same training and test data sets and then comparing the number of misclassifications they produce on these data sets. Misclassifications on the training set are called training errors, and misclassifications on the test set are called test errors. Quite often researchers also use a slight variation of this method by measuring and comparing true positives, false negatives and their ratio. This is essentially the same method however, the measures are applied not to the whole data set but instead onto specific classes to see how they perform.

In the case of machines that predict with confidence and credibility we somehow have to be able to compare their predictive performance with respect to the confidence and credibility values as well as the error rate. On particular given data sets confidence machines might produce exactly the same error rate. This does not necessarily mean that these confidence machines are equivalent. The key procedure within the confidence machine is the procedure of typicalness approximation. Even if both machines produce exactly the same error rate on some data sets their typicalness approximation mechanisms could be different and one could be better than another. The better one obviously would produce more credible results in the long run on unseeing yet examples that they might be presented with in the future. So, it is important to compare and identify which confidence machine is better in terms of its typicalness approximation performance. In this section we propose six possible performance measures that can potentially be used to do exactly this. Our new performance measures are not based on training or test errors hence they do not have analogs in the bare prediction learning algorithms and may be difficult to understand at first.

Suppose we are given a training set

$$Z = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n),$$

a test set

$$T = (xt_1, yt_1), (xt_2, yt_2), \ldots, (xt_l, yt_l),$$

and the set of all possible classifications is $Y = Y_1, \ldots, Y_c$, where $x_i, xt_j \in \Re^k$, $i = 1, \ldots, n$, $j = 1, \ldots, l$ and $Y_t \in N$, $t = 1, \ldots, c$. Then the performance measures are:

Performance 1.

$$\gamma_1 = \sum_{\Omega \in T \times Y} \frac{-\ln p_\Omega}{|T \times Y|},$$

where p_Ω is the p-value obtained for the pair Ω consisting of a test example and its postulated label, and $|T \times Y|$ is a size of the Cartesian product $T \times Y$. Essentially this measure shows a minus logarithm of the typical p-value obtained on the overall test set. Smaller typical p-value means better algorithm, which corresponds to the higher value of this performance measure. The meaning of this measure can be explained by the fact that in reality we would like to have one label with very high p-value (this label will be our prediction), and all other labels should have very low p-values (so all other possible labels are excluded). Hence, the overall typical p-value should be lower for better algorithms.

Performance 2.

In the Performance 1 typical p-value may be distorted by the large p-values that are usually obtained for true classification. In this measure we try to exclude such values to obtain a clearer picture. This measure is very similar to the previous performance measure but does not take into account the p-values for the true labels:

$$\gamma_2 = \sum_{t \in T} \sum_{y \in Y \setminus \{y_t\}} \frac{-\ln p_{(t,y)}}{|T|\left(|Y| - 1\right)},$$

where y_t is the true label for the test example t, $p_{(t,y)}$ is the p-value obtained for the test example t and the "wrongly" postulated label y.

Similar to the previous performance, this one also shows a minus logarithm of the typical p-value obtained on the overall test set while ignoring potentially large p-values for true classifications. Smaller typical p-value means better algorithm, which corresponds to the higher value of this performance measure. Intuitively, the meaning of this measure is the same as for the previous one however, this one should be more accurate because we ignore large p-values that correspond to true labels.

Performance 3.

In the previous two measures we dealt with most of the p-values to get overall statistics for multiple postulated classifications. In this measure, in contrast, we will deal only with one p-value for each example – this is the second largest p-value. This p-value identifies the confidence given to our predictions. By measuring the typical second largest p-value we may judge the typical confidence assigned to examples in the test set:

$$\gamma_3 = \sum_{t \in T} \frac{-\ln p_t^{(2)}}{|T|},$$

where $p_t^{(2)}$ is the second largest (over the postulated labels) p-value for the example t in the test set.

This measure shows a minus logarithm of the typical second largest p-value obtained for each example in the test set. Smaller typical p-value means higher typical confidence and hence better algorithm that corresponds to the higher value of this performance measure. The meaning of this measure can be explained by the desire to have confident predictions (predictions with high confidence). It is possible that algorithms that produce more confident predictions could be better.

Performance under specified significance levels.

The previous three measures are concerned solely with p-values without taking into account the results of predictions and their meaning. The following meas-

ures are concerned with making predictions under various standard significance levels. The importance of this can be illustrated by the following example. Let us say the machine has made a prediction however, its credibility is 9% and its confidence is 93% - do we really want to make this prediction or not?

There are standard significance levels from statistics that are 1%, 5% and 10%. They are used to indicate the significance of a particular solution. For example, we might setup 10% significance level, which means that we are unable to make a decision if credibility (largest p-value) for a particular prediction is less than 90%. Under the same significance level we might say that we are also unable to make a decision if more than one potential classification is predicted with more than 90% credibility and so on including other standard significance levels. Therefore, it is important to assess the performance of a confidence machine on a particular data set given standard significance levels and the prediction constraints as described above.

Let δ be a *significance level* (1%, 5% or 10% in our experiments). We will call the complementary value $1-\delta$ the *confidence level*. Then the following performance measures can be constructed.

Performance 4.

$$\gamma_4^{\delta} = \frac{\#\left\{ p_t^{(\max)} : p_t^{(\max)} < 1-\delta \right\}}{\left|T\right|} \cdot 100\%,\ t \in T,$$

where $p_t^{(\max)}$ is the largest (over the postulated labels) p-value for the example t in the test set.

This performance measure essentially shows a percentage of time the program is unable to predict an example because its credibility is not exceeding the given confidence level $1-\delta$. The meaning of this measure can be explained by the fact that we would like to maximise the percentage of confident predictions under given credibility threshold. Naturally, if we reject some predictions because their credibility does not meet the required level the better algorithm should produce fewer such cases.

Performance 5.

$$\gamma_5^{\delta} = \frac{\#\left\{ p_t^{(2)} : p_t^{(2)} \geq 1-\delta \right\}}{|T|} \cdot 100\%$$

where $t \in T$, $p_t^{(2)}$ is the second largest (over the postulated labels) p-values for the example t in the test set.

In contrast to the previous performance measure, this one shows a percentage of time the program can potentially predict an example with more than one classification because more than two postulated labels produce p-values exceeding the given confidence level $1-\delta$. The meaning of this measure is similar to the previous one however, now we would like minimise occurrences of predictions when we cannot effectively make a conclusive prediction because two or more labels could be assigned to the same example under the specified credibility threshold.

Performance 6.

$$\gamma_6^{\delta} = \frac{\#\left\{ p_t^{(\max)}, p_t^{(2)} : \left(p_t^{(\max)} \geq 1-\delta \right) \& \left(p_t^{(2)} < 1-\delta \right) \right\}}{|T|} \cdot 100\%$$

where $t \in T$, $p_t^{(\max)}$ is the largest and $p_t^{(2)}$ is the second largest (over the postulated labels) p-values for the example t in the test set.

Complementing the ideas behind two previous performance measures, this one shows a percentage of time the program is absolutely certain about its prediction because only one postulated label exactly produces a p-value exceeding the given confidence level $1-\delta$.

Chapter 4

4. Implementation of the Confidence Machine

4.1 Introduction

One of the significant parts of this thesis is the practical design and implementation of the confidence machine system. In this chapter we will go into the details of the actual implementation of the confidence machine system, which incorporates TCM and ICM in one system with common interface and output, such as both systems can be compared. We will give detailed diagrams of the system internals, how it is designed to work with any underlying learning algorithm and how common parts are shared to make the system expandable.

4.2 Confidence Machine system objectives

During the life time of the machine learning field many machine learning algorithms have been developed. Even though most of the algorithms solve exactly the same type of problems, almost everyone has their own implementation of such algorithms. This results in having different formats for data sets, different interfaces for inputting the data, different parameters and more importantly different ways and formats for presenting the results. It is a major logistical issue for anyone trying to perform experiments using various learning algorithms in order to analyse and compare their relative performances because all of the issues explained above have to be addressed. Confidence Machine adds another layer of functionality and complexity to the existing learning algorithms that make the potential task of comparison even harder to achieve. In order to complete all the experiments required for this thesis and conclusively

show the performance of Inductive Confidence Machine from all angles, we need to create a system that has to deal with all the incompatibility and multi-format issues described above. Moreover, Confidence Machine uses bare prediction learning algorithms as strangeness approximation functions, so we have to deal with all the interface issues in order to successfully incorporate such learning algorithms into the Confidence Machine system. There are two types of confidence machines: Transductive and Inductive. Along with possible bare prediction learning algorithms the system has to be able to implement both of these types of confidence machines. Overall the system should have a unified format for inputting the data, a single point of parameters entry and a unified output format that can be used for performance comparisons. Only after all the described requirements are met we can perform experiments with various parameters and compare the performances.

The following list is a summary of requirements for the desired Confidence Machine system:

a) single format for inputting given training and test data;
b) unified interface for plugging in the bare prediction algorithms;
c) unified interface for entering system parameters;
d) unified output format that allows the results comparison;
e) modulated design for easy expandability.

4.3 System design

4.3.1 Input data format

Machine learning algorithms deal with data represented in the following format:

$$z = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n),$$

where $x \in \Re^k$ is an object represented as a vector of size k, y is its classification assigned by a supervisor and $y \in \{Y_1, \ldots, Y_c\}$ $Y_i, c \in N$ for Pattern Recognition problems or $y \in \Re$ for Regression problems.

Vast majority of real-world problems solved by machine learning algorithms can be represented as data sets in the format above. Each object is converted into a vector of some attributes relative to the object, and all given objects have the same number of such attributes. Of course, some of the real-world problems such as learning DNA sequences or texts cannot be represented in this format directly, however most such problems are solved by pre-processing the data in order to convert it into the format above. Data sets are entered into machine learning systems from text files, therefore the natural format for data files would be as follows:

$$\begin{array}{l} n \\ a \\ c \\ v_{11}, v_{12}, \ldots, v_{1a}, y_{11}, \ldots, y_{1c} \\ \vdots \\ v_{n1}, v_{n2}, \ldots, v_{na}, y_{n1}, \ldots, y_{nc} \end{array}$$

where n is a number of examples (objects) in the given data set, a is a number of attributes in each vector, c is a number of classifications assigned to each object, v_{na} is an attribute a of the object n, and y_{nc} is a label number c assigned to the object n.

This file format is suitable for both training and test sets and therefore it will be used for feeding the data into the Confidence Machine.

4.3.2 Interface for bare prediction learning algorithms

From the first glance at the algorithms for TCM and ICM shown on Figures 3.5.2.4 and 3.6.1.3 it is not clear whether there is a unified interface to bare prediction learning algorithms that can be used in both approaches. Detailed analysis and procedural optimization, however, show that a bare prediction learning algorithm should have an ability to implement only two methods in order to be seamlessly connected to both of the confidence machines. Moreover, both of these methods should be able to deal only with binary data sets that makes the adaptation of almost any existing learning algorithm possible.

Both Confidence Machine algorithms at the lowest level come down to the multiple repetitions of applying a strangeness function A to a binary data set

$$z = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n),$$

where $x \in \Re^k$ is an object represented as a vector of size k, y is its classification assigned by a supervisor and $y \in \{+1, -1\}$. The only difference between TCM and ICM at this level is that a strangeness function A is obtained directly from the set z in a case of TCM, while in a case of ICM a strangeness function A is obtained from a different data set in advance and then applied to the set z to obtain individual strangeness measures.

As we have described already in the previous chapters, we can use bare prediction learning algorithms as approximations to the strangeness functions. Therefore each algorithm has to be able to produce individual strangeness measures in two different ways. The first method should be as follows:

$$(D, \alpha) = TrainBinary(z),$$

where z is a binary data set, α is a set of individual strangeness measures for each example in the set z and D is a derived rule that can be used to produce strangeness measures afterwards $\alpha = D(z)$.

The second method is the one that applies the rule derived using the first method to a data set to obtain individual strangeness measures:

$$\alpha = ClassifyBinary(D, z).$$

Any learning algorithm that can implement these two methods can be used within the Confidence Machine system for strangeness approximation purposes.

In this thesis we are dealing with two bare prediction learning algorithms, so let us analyse how they can implement the methods above. Starting from the Nearest Neighbours algorithms it is clear that the $(D, \alpha) = TrainBinary(z)$ method can be implemented by using the formula (2.8.1) to obtain individual strangeness measures. Rule D is in fact a data set z itself. Due to the nature of the Nearest Neighbours algorithm where there is no training as such, the other method $\alpha = ClassifyBinary(D, z)$ is identical to the first one and also uses the formula (2.8.1) to obtain individual strangeness measures. Hence, Nearest Neighbours algorithm can be employed within the Confidence Machine.

In the Support Vector machine algorithm $(D, \alpha) = TrainBinary(z)$ method is essentially an optimization procedure that creates a hyper-plane. Resulting Lagrange multipliers are taken as individual strangeness measures in case of TCM and $\alpha = -h$ in case of ICM. The rule D is the hyper-plane itself identified by all the support vectors with their Lagrange multipliers. The other method $\alpha = ClassifyBinary(D, z)$ is a procedure that reconstructs a hyper-plane learned by the first method and then finds a distance from the hyper-plane for each example in the set z.

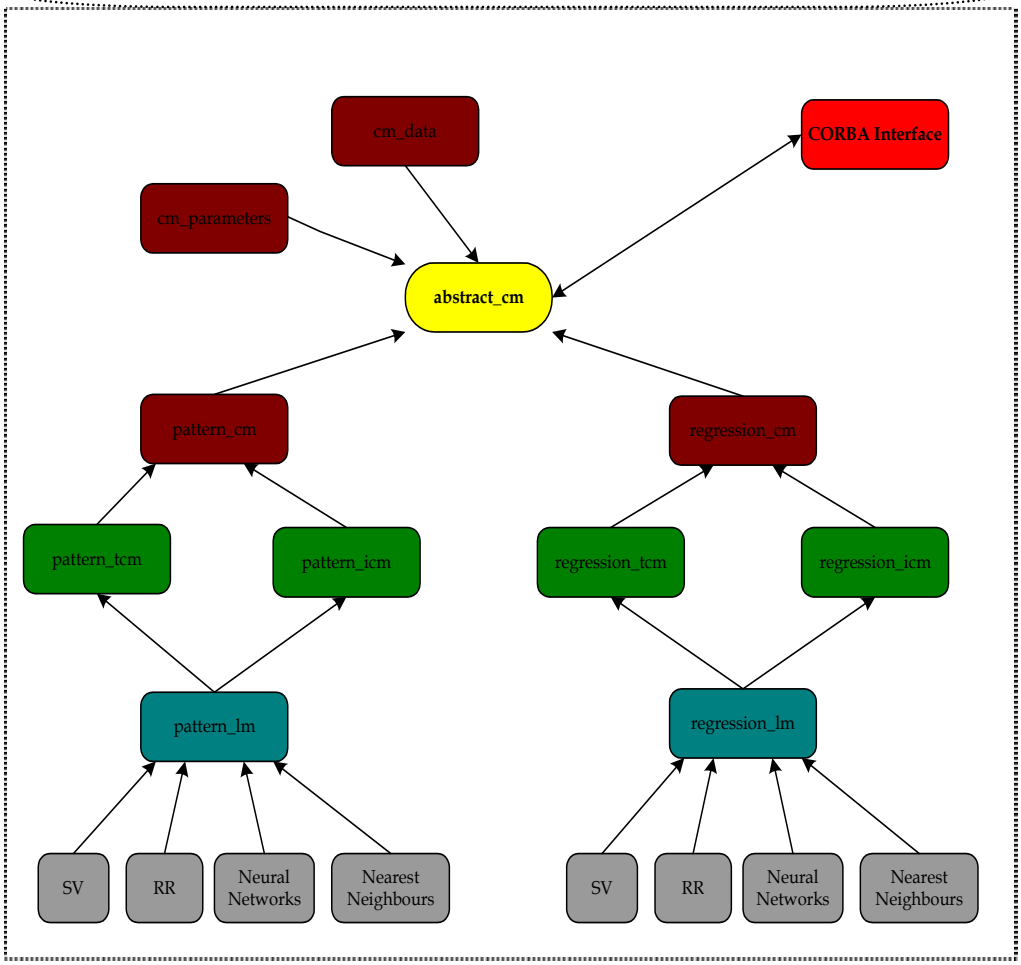
4.3.3 Internal diagram

One of the learning algorithms used in our Confidence Machine is the Support Vector Machine. We use the SVM implementation by Stitson [56] which is a large library designed and implemented using Object Oriented Programming. Following their idea, it makes sense to design our Confidence Machine using the OOP principles as well. Breaking down the system into individual objects (classes) we modulate the system by specifying exact interface for each class. This allows to develop a system that is easily expandable in any direction which is one of the system's requirements. Each class performs a specified operation and more complex operations are constructed by combining these classes in a precise manner. This allows to use the same classes in different places and avoid duplication of code that could make the system unstable and impossible to maintain.

The object diagram of the Confidence Machine system is shown on Figure 4.3.3.1.

In the centre of the system we have an abstract class **abstract_cm** which performs the highest level operations within the system. These operations do not depend on the type of confidence machine in use and include reading the data, reading the parameters, calculating predicted classifications along with confidence and credibility from the p-values obtained in the other classes. This class also has interface with CORBA that allows to control and execute the system from remote applications either over the network or from applications written in a different programming language such as Java.

Figure 4.3.3.1: Internal Diagram of the Confidence Machine System



At the next level of the operations hierarchy we have two classes **pattern_cm** and **regression_cm.** These classes implement functions that are common for both TCM and ICM within the pattern recognition or regression scopes. The functions include procedures such as calculating p-values from the sets of strangeness measures.

At the next level down the hierarchy we have four classes **pattern_tcm, pattern_icm, regression_tcm** and **regression_icm.** Again these classes perform common operations for each type of the confidence machine, which are independent of the bare prediction learning algorithms. These classes implement the main functions that discriminate one type of confidence machine from another. The functions implemented within these classes include all the procedures for correct formulating of confidence machine problems, correct concatenation of the training data and test examples, correct procedures for multi-class predictions and so on.

Next level down is occupied by another two classes **pattern_lm** and **regression_lm.** These classes implement the methods that are used by the classes above to access the underlying bare predictions learning algorithms. These classes implement methods like TrainBinary and ClassifyBinary described in the previous section and methods for interfacing learning algorithms thus shielding the classes above from the complexities of the bare prediction learning algorithms.

At last at the bottom level we have classes that implement bare prediction learning algorithms with the methods described in the previous section to interface with the Confidence Machine System.

4.3.4 Execution procedures

The whole Confidence Machine system consists of just one executable that combines all the machines and all the operations required to run the learning algorithms. For underlying algorithms like Support Vector Machine we also need to use another program called ***paragen*** for creating SV parameter file. Algorithms like Nearest Neighbours do not require any additional parameters hence we can skip the first step of creating parameters for an underlying bare prediction learning algorithm in such cases.

For example, if we need to use ICM or TCM based on SV, first we need to create a parameter file as shown on Figure 4.3.4.1. This figure shows a typical combination of parameters that can be used for most of the problems. The only parameter that must always be the same for pattern recognition problems is "SV Machine Type" which must be set to Pattern Recognition. All other parameters can be changed as required. Parameter file is created by executing ***paragen*** and then saving to ***sv.prm***. For more information about the SV parameters see, e.g., Stitson [56].

The next step is to prepare training and test files in a format described in Section 4.3.1. Then we can run our confidence machine by executing ***./coma_main*** with a set of required parameters as shown on Figure 4.3.4.2. Parameters in <> brackets shown on this figure are mandatory and the ones in [ ] brackets are optional. Combining these parameters in different variations we can perform all the experiments we need.

Figure 4.3.4.1: Creating SV parameters for TCM and ICM based on Support Vector machine.

```
./paragen sv.prm

SV Machine parameters
    ======================
    Pattern Recognition

    Full polynomial

    Alphas bounded up to 1e+06.
    Input values will not be scaled.
    Training data will not be posh* chunked.
    Training data will not be sporty* chunked.
        Optimizer: 1
        SV zero threshold: 1e-16
        Margin threshold: 0.1
        Objective zero tolerance: 1e-07

    Degree of polynomial: 3.
    Kernel Scale Factor: 256.
    Kernel Threshold: 0.

     1. Set the SV Machine type
     2. Set the Kernel type
     3. Set general parameters
     4. Set kernel specific parameters
     5. Set expert parameters

     0. Exit

    Please enter your choice:
```

* SVM chunking types and other parameters defined in [56]

Figure 4.3.4.2: Executing Confidence Machine System.

```
./coma_main    <-dm DecisionMachine> <-lm LearningMachine>
               <-lmt LMType> <-of OutputFile> [-crossval]
               [-train TrainingFile] [-test TestFile]
               [-sp SVM Parameters]
               [-c Calibration File | -cre NE -crp NP]
               [-ct TrainedICMFile]

<-dm DecisionMachine> - TCM or ICM
<-lm LearningMachine> - SV, RR, NRST_NBR, NRL_NKS
<-lmt LMType> - P for Pattern Recognition or R for Regression
<-of OutputFile> - path to the file for output of p-values and statistics

[-crossval] - optional, leave-one-out prediction will be performed if present
[-train TrainingFile] - optional path to the file with training data
[-test TestFile> - optional path to the file with test data

[-sp SVMParameters] - optional path to the parameters file for SV

[-c CalibrationFile] – optional, either path to the file with calibration data for
                       ICM or you need to specify how to create a calibra-
                       tion set by specifying the following parameter:

[-cre NE -crp NP] - NE: number of examples for each class required in the
                       calibration set,
                    NP: number of times to randomly permute the given
                       Training Set before splitting it into the proper
                       training and calibration sets

[-ct TrainedICMFile] -  optional path to the file where Trained ICM will be
                       saved which can be used later for classification, if
                       Calibration Set is not specified at all then Classifica-
                       tion Mode is assumed for the ICM.
```

4.3.5 Output format

After an execution is completed the confidence machine system produces three output files. The file names are constructed by adding an explanation to the end of the name specified in the parameter <-of OutputFile> which is described in the previous section.

The first output file is ***<OutputFile>_pvalues*** (e.g. sv-tcm.results_pvalues). This file contains p-values, real classification, predicted classification, confidence and credibility for each test example. Each line represents one test example in the following format:

$p_0 \quad p_1 \quad \ldots \quad p_c$ Real Class Predicted Class Confidence Credibility

where p_t is a p-value for a postulated classification $t = 1, \ldots, c$ and c is a number of possible classifications. For instance, one line in this file for a digit 7 predicted as 1 could look like the following:

0.1 0.65 0.3 0.31 0.45 0.21 0.32 0.5 0.11 0.3 7 1 0.5 0.65

The second output file is ***<OutputFile>_stats*** (e.g. sv-tcm.results_stats). This file contains all calculated performance measures and various statistics about an experiment as described in Section 3.7 above. It shows the information not only for all examples overall but also all the performance measures are given for individual classes. Results are written into this file in a format shown on Figure 4.3.5.1. This figure shows a format of the second output file with some sample data.

Finally, the third output file is ***<OutputFile>_stats_line*** (e.g. sv-tcm.results_stats_line). This file is a cut down version of the previous one. It only contains the performance measures for all test examples and not for individual classes. Also, all the information is displayed in one line. The purpose of this file is to be able to import results easily into programs such as MS Excel for analysis and charting.

Figure 4.3.5.1: Output produced in <*OutputFile*>_*stats* file.

Performance\Class	**0**	**1**	**2**	**.....**	**8**	**9**	**All**
Perf. 4 (1%)	357.00	260.00	196.00		164.00	176.00	1991.00
Perf. 4 (5%)	344.00	249.00	192.00		161.00	169.00	1915.00
Perf. 4 (10%)	318.00	232.00	187.00		158.00	159.00	1820.00
Perf. 5 (1%)	0.00	0.00	0.00		0.00	0.00	0.00
Perf. 5 (5%)	0.00	0.00	0.00		0.00	0.00	0.00
Perf. 5 (10%)	0.00	0.00	0.00		0.00	0.00	0.00
Perf. 6 (1%)	2.00	4.00	2.00		2.00	1.00	16.00
Perf. 6 (5%)	15.00	15.00	6.00		5.00	8.00	92.00
Perf. 6 (10%)	41.00	32.00	11.00		8.00	18.00	187.00
No. of Examples	359.00	264.00	198.00		166.00	177.00	2007.00
No. of Errors	7.00	13.00	13.00		9.00	6.00	101.00
No. of Errors (%)	1.95	4.92	6.57		5.42	3.39	5.03
Perf. 1							5.06617
Perf. 2							5.50127
Perf. 3							4.68846

Chapter 5

5. Confidence Machine Applications

5.1 Introduction

Machine learning algorithms have already found many practical applications. These applications include handwritten text recognition for scanners, speech recognition in various human-machine interfaces, image recognition in ordinary and surveillance cameras and so on. Confidence Machine is another learning algorithm that improves current machine learning algorithms by providing assessment measures for each of its output as well as the output itself. This means that it can be practically used for any current machine learning application. However, it also opens up new opportunities that have not been available before. In this chapter we try to envisage the scope of applications and show the advantages of using the confidence machine rather than bare prediction learning algorithms.

5.2 Biological Sequences and Texts recognition

Repeating what we already stated, a typical machine learning algorithm deals with the following kind of problems. We are given a set of data:

$$z = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n),$$

where $x \in \Re^k$ is an object represented as a vector of size k, y is its classification assigned by a supervisor and $y \in \{Y_1, \ldots, Y_c\}$ $Y_i, c \in N$ for Pattern Recognition problems or $y \in \Re$ for Regression problems. When new object x_{new} arrives the machine using the knowledge learnt from the previously given data z tries to predict the classification y_{new} of this new object.

This means that any real-world data needs to be converted into the format above before it can be fed into a learning machine. While some data is already in this format or can easily be represented in such format, most of the real problems cannot be represented as such trivially. Examples of such data include texts where we are given a set of different types of text (each has different length) and asked to predict the type of other fragments of text. This is one the most popular and essential problems to be solved. Solving this problem would allow Internet search engines to work much better than currently achievable. Another problem that currently cannot be solved easily comes from the bioinformatics field. Presently we have a vast amount of protein and DNA sequences collected from various organisms. Obviously there are many problems currently unsolved such as identifying functions and classes of proteins, identifying protein secondary and tertiary structures, locating coding regions such as promoters in DNA etc. from these sequences. These problems potentially can be solved efficiently with learning machines however, we have the same obstacle as with texts – all sequences or their parts have different lengths therefore they cannot be trivially converted into the required format.

The beauty of machine learning algorithms such as Support Vector Machines lies with its ability to learn an Independently and Identically Distributed data. Moreover, we can learn and predict the data successfully without knowing any of its domain specific properties. For example with image recognition we do not have to know anything about image properties such as orientation, brightness, contrast etc. However, all these become essential when we try to apply learning algorithms to the data such as protein and DNA sequences or texts. There are two main approaches to applying learning algorithms to such data.

The first one is called sequence alignment. Essentially we count a number of occurrences of specified combinations of symbols starting from the most basic ones. Then we create a vector of such numbers, and because we specify exactly which combinations to count every vector would have exactly the same length. These vectors are then presented to machine learning algorithms. There is a large number of variations to these methods. Each method gets more and more complicated with decreasing execution speed and results are not improved significantly.

The second method is even more complicated but more successful. It is based on our knowledge of the domain the data belongs to. It is also concerned with

creating vectors of the same size however, the attributes of these vectors are very specific properties of an original sequence. These properties come from the existing knowledge of experts. For example, we might know that if a particular combination of DNA symbols lies within a specific interval from another combination then there is a significant chance of having a promoter within a certain region. Hence, one of the attributes could be the interval itself. Following the same methodology experts can come up with many parameters that identify a specific type of a sequence. These parameters are organised as a vector and the obtained data can easily be learnt by a learning machine.

Examples of various methods for converting data into the format required by learning machines can be found in [77], [78] and [79].

With the invention of the Confidence Machine there is now an alternative approach to learning such data. The theory of the Confidence Machine using an ideal approach described in Section 3.2 has no requirements for operations between two examples. Therefore, in theory there is no requirement for data to have examples represented as vectors of the same size. If we decide to use the Kolmogorov complexity approximation as our method of constructing Confidence Machine as shown in Section 3.4.3, then potentially we can construct a learning machine which does not have this restriction on data format. To achieve this we may use the procedure shown on Figure 3.2.2.1. In Step 2 on this figure we calculate typicalness level of all possible sequences constructed using training data and the new example. Firstly, we convert all our data examples into the binary sequences, which is usually not a great problem for texts and bio-sequences. Then we construct a sequence z_t by concatenating all training sequences of class t and the new sequence to be predicted. We find typicalness deficiency $d(z_t)$ by compressing the sequence z_t according to our Kolmogorov complexity approximation assumption and using the following formula:

$$d(z_t) = l(z_t) - CL(z_t)$$

where $l(z_t)$ is the length of the sequence z_t and $CL(z_t)$ is the length of the sequence z_t after compression. Then we convert the typicalness deficiency into

the typicalness level $\lambda(z_t) = 2^{-d(z_t)}$ and continue from Step 3 on Figure 3.2.2.1 and find our prediction, confidence and credibility.

In the approach above we employed Kolmogorov complexity approximation right in the core of the procedure for an ideal prediction with confidence and credibility. There is also an alternative, whereby we can continue to approximate strangeness with bare prediction learning algorithms and we use Kolmogorov complexity approximation within the SVM Kernel. Kernel is one of the key ideas within the Support Vector Machine. As described in Section 3.3.2 it allows to calculate a dot product between two vectors in a high dimensional space without transforming these vectors into that space. SVM requires a matrix of dot products or kernels between all training and test examples. Normally kernel is a standard function between two vectors which is a primary reason why SVM requires all data examples to be represented as vectors of a certain size. Nevertheless, a kernel potentially can be constructed using the same idea of Kolmogorov complexity approximation.

The main idea of such kernel is that we consider the measure of information distance defined in Section 2.3.3 to be the real distance between two objects in some unknown high dimensional Hilbert space. Then the kernel is derived as follows.

A dot product between vectors X and Y can be represented as

$$X \cdot Y = \frac{1}{2}\left(\|X\|^2 + \|Y\|^2 - D(X,Y)^2\right),$$

where $\|X\|$ is the length of vector X (distance from the origin to the point X), $\|Y\|$ is the length of vector Y and $D(X,Y)$ is the distance between the points $\|X\|$ and $\|Y\|$. Kernel is a dot product in some feature space:

$$\mathrm{K}(X,Y) = X' \cdot Y',$$

where X' and Y' are the corresponding points in the feature space. Hence,

$$\mathrm{K}(X,Y) = \frac{1}{2}\left(\|X'\|^2 + \|Y'\|^2 - D(X',Y')^2\right),$$

where $\|X'\|$, $\|Y'\|$ and $D(X',Y')$ are the lengths of vectors and distance between the points in the feature space.

This shows that if we know a distance from the origin for both points and a distance between the points in the feature space we can easily find the kernel. Now we have a possibility to have points in the original space of different dimensions assuming there exists a transformation into one feature space for both points. We assume that Kolmogorov information distance described in Section 2.3.3 between points X and Y is a real distance between them in some unknown feature space for which there exists the required transformation. Then a distance from the origin in that feature space would be

$$KD(X,\varepsilon) = \max\{K(X \mid \varepsilon), K(\varepsilon \mid X)\} = K(X)$$

for point X and $K(Y)$ for point Y, where ε is the empty string representing the origin. The kernel for our points X and Y will be

$$\mathrm{K}(X,Y) = \frac{1}{2}\left(K(X)^2 + K(Y)^2 - KD(X,Y)^2\right).$$

As assumed earlier Kolmogorov complexity of a string can be approximated by the length of this string after compression $K(X) \approx CL(X)$. The distance between our objects in the high dimensional feature space may now be approximated as follows:

$$KD(X,Y) = \max\{CL([YX]) - CL(Y), CL([XY]) - CL(X)\}$$

assuming that the conditional prefix complexity can be approximated by

$$K(X \mid Y) = CL([YX]) - CL(Y),$$

where $CL(X)$ is the length of the sequence X after compression and $CL([YX])$ is the length after compression of a sequence obtained by concatenating Y and X.

Combining the last two equations we obtain the final formula for our kernel:

$$\mathrm{K}(X,Y)=\frac{1}{2}\left(\begin{array}{l} CL(X)^2+CL(Y)^2- \\ \left(\max\left\{CL([YX])-CL(Y),CL([XY])-CL(X)\right\}\right)^2 \end{array}\right).$$

Now we can construct a Confidence Machine which uses SVM as the strangeness approximation mechanism and works with a variable dimensionality data such as texts and bio-sequences. In both approaches we not only get a prediction but also measures of confidence and credibility to make sure the machine or human operator can make intelligent decisions.

In Section 6.2.4 below we have experimental results of using the technique above. The results show that this method outperforms other previously employed methods in identifying protein fold classes from their sequences.

5.3 Medical Applications

Automated or computer assisted diagnosis of illnesses based on symptoms is one of the most promising areas for the machine learning applications. Currently though it is not very developed and we are still relying on human experts to do the diagnosis. It is a very well known fact that in the UK there is a major shortage of qualified doctors that can assess patients' situation correctly. Due to the complexity of the subject it is getting more and more difficult to perform the diagnosis. This is mainly because new viruses penetrate our lives, new and unknown symptoms appear and in many cases doctors just do not have enough time to study every single situation thoroughly. This is obviously a very critical issue and machine learning algorithms could help significantly. Potentially they could analyse all the existing data and provide the diagnosis based on given symptoms. Many human mistakes can be avoided especially on issues that have a very large database to analyse that cannot be comprehended by any human being within any reasonable amount of time.

The answer is obvious – there is a definite need for an automated system that can analyse all available data and advise a doctor on possible diagnosis in each individual case. Machine Learning algorithms can already be used for such tasks however, they did not get much popularity probably because bare predictions could make more damage than good. As we already stated, when a human being makes a decision he or she usually looks at the problem from different perspectives. Various different possibilities are considered and the most sound one is taken as a decision or decision is not made due to uncertainties in all possibilities. Bare prediction learning algorithms produce just one answer without any indications of how good it is and whether it is worth considering or not. This is probably why machine learning algorithms have not been accepted by the medical community. It is understandable that in their view an answer is just not enough because it could well be a mistake. What they require is an intelligent answer, the one that looks at the problem from different angles and gives measures of certainty for each possibility.

This is where the Confidence Machine can show its strength compared with bare prediction. For each possible answer confidence machine produces quality measures called confidence and credibility. Confidence indicates the certainty that all other possibilities are excluded. For example, low confidence could tell the operator that along with the prediction there are possibilities of other pre-

dictions and the level of it exactly quantifies this possibility. Credibility on the other hand shows how typical the prediction is within the database of records with the same prediction. For example, based on given abdominal pain symptoms the machine predicted Appendicitis for one of the patients. Credibility would show how this instance of Appendicitis compares with all known instances, thus quantifying the typicality of the prediction. Moreover all other possibilities are also given the same measure of typicality within their scope.

Analysing these quality measures for each prediction a doctor can clearly see the situation according to the database knowledge and hence one could confirm their assumptions if confidence and credibility are high or consider reanalysing the case if it is otherwise.

Figures 6.2.3.4, 6.2.3.5 and 6.2.3.6 in the experiments Section 6.2.3 with ABDO dataset below show what would a doctor see. These are standard charts that can be produced from the Confidence Machine's output. They clearly show a diagnosis for a particular patient. A doctor can clearly identify the possibility of each possible diagnosis, how typical each one of them is according to an existing knowledge base and a decision can be made relatively easy.

5.4 Model Selection

All machine learning algorithms require various parameters to be specified before even beginning training and classifying examples. Some algorithms require only a few parameters while others might need a dozen of them. For example, Support Vector Machine needs to be supplied minimum with the following values: bound on alphas, multi-class type, scaling type, chunking type, optimiser, various thresholds and kernel. In turn some of these values require more parameters such as a kernel might need degree, scale factor, threshold etc. Each possible combination of parameters produce very different results. The obvious question is how can one select the best combination of parameters to work with? Quite often there is no simple answer to this question because different data requires different parameters and the task of identifying such parameters becomes the most consuming one.

Model selection is a set of methods for finding a combination of parameters (a model) that will perform the best on yet unseen data. Training error rate is the simplest, the most common and yet the least reliable way of selecting a model. In this method we apply a trained machine on the training data and select the model which produces the least amount of errors on the data it was trained on. Most of the time a machine learning algorithm can learn the data entirely without any errors. This gives no difference between one model and another. In many cases a real-world data contains a significant amount of noise which should be identified and ignored or, in other words, the data should be learned with errors to provide better performance on yet unseen data. So, quite often the model selection based on training errors does not produce a good model. Nevertheless, the model selection problem is one of the most important ones in the machine learning field because any practical use of learning algorithms will be preceded with identifying the best parameters in each particular case. Inductive Confidence Machine, the main topic of this thesis, requires a calibration set selection which is another parameter that influences the result significantly but it is not clear how to actually select one apart from choosing at random.

The importance of this problem attracted many machine learning researchers. There have been many different suggestions that depend on algorithm or data. For example one of the general ideas is to measure a complexity of a model and then apply a principle that the simplest model is the best one. Even though it can be implemented in some particular cases most of the time it faces questions

like: how to measure complexity of a model or what is the lower bound on complexity to make sure that the model is not too simple? There is no definite answer to these questions. Another idea that has been researched widely concerned with introducing a loss function rather than just counting errors. It is a kind of an extension of training error model selection however, it is more flexible because training error rate is adjusted according to some properties found in the data during training. The loss function in such cases is a function of training errors and some carefully selected data properties. Usually these properties are different from one data to another and each particular domain requires new research into obtaining a suitable loss function. These examples are just a few basic ones and there are many more other approaches to this problem. More information about various model selection methods can be found in [80], [81] and references therein.

During the training phase a confidence machine produces p-values as described above in Section 3.2.3. These p-values indicate typicalness of each possible solution. Performance measures γ_1, γ_2 and γ_3 defined in Section 3.7 are based on the obtained p-values and assumed to indicate some measures of performance of a particular confidence machine. It turns out that these performance measures can be used for model selection independently of training error rate. To prove this statement empirically we perform the following experiment trying to obtain the best ICM calibration set from randomly chosen ones.

a) Obtain a random selection of proper training and calibration sets from a given training set. All other parameters are specified and fixed throughout this experiment.
b) Train an Inductive Confidence Machine on the sets obtained in the previous step.
c) Apply the trained ICM on the originally given training set to obtain the performance measures γ_1, γ_2 and γ_3 and the overall training error rate.
d) Apply the trained ICM on the specified and fixed test set to obtain the test error rate.
e) Repeat 50 times from step a) and record all the results.
f) Analyse dependencies between the values of performance measures, training error rate and test error rate to find out if performance measures are better indicators of future performance (on yet unseen test data) compared to training error rate.

If the analysis of this experiment would show that better values of the performance measures on the training set correspond to lower error rates on the test set then we could use these measures as a model selection criteria.

Figures 6.2.1.20 and 6.2.3.6 in the experiments section below show charts where we can observe possible dependencies between the test error rate and the performance measures on the training set during learning. The graphs prove that the training error rate is a very unreliable measure for model selection. At the same time it is shown that there is a weak but conclusive dependency between the confidence machine performance measures on the training set and the testing error rate on yet unseen data. The dependency does not seem to be very strong and, for some reason, the best performance measures correspond to the 4^{th} and 9^{th} best testing error rate on USPS and ABDO data sets respectively. However, the tendency is clear and perhaps this could be explained by the limited number of repetitions executed in the experiment. Obviously this experiment is only an approximation to proving the model selection fact because to truly prove it we need to perform an infinite number of repetitions with all possible testing data sets. This cannot be done empirically, hence any experiment trying to prove the model selection capabilities would be an approximation.

5.5 Other applications

In the introduction of this chapter we stated that Confidence Machine is another learning algorithm therefore it can be employed at least everywhere where current machine learning algorithms are used. So far machine learning algorithms have not achieved as much as they could have done or as have been expected. The main reason for this could probably be the fact that answers from these algorithms are not considered to be intelligent and therefore useless in the real-world situations. We had already pointed this out when we talked about medical applications above. However, the problem extends to any other field. From the moment of computer birth mankind has been expecting intelligent answers from machine learning systems – the dream that has not been delivered. The original idea of Confidence Machine as presented by Prof. Vovk and Prof. Gammerman is to produce a machine learning algorithm which can provide more than just an answer but also some measures of quality for this answer. The idea turned out to be more than just that. As we described at the beginning of this thesis quality measures produced by the confidence machine are not just measures – they have a deeper meaning. It seems like they quantify a machine's ability to approximate randomness of a presented situation. By comparing this with well known aptitude tests we drew a conclusion that these measures provide a mechanism in order for a machine learning algorithm to produce intelligent answers. Suddenly we make a qualitative jump from raw execution to intelligent decisions. Moreover, we actually produce a definition of intelligence which is not based on standard dictionary definitions that limit it to a simple comparison with human behaviour. From our discussions in Section 1.4 we conclude that *intelligence is an ability to find the least random situation out of all possible ones.* This is exactly how confidence machine comes up with a decision along with quantitative measures of randomness for each possible solution. This is why we may say that confidence machine is the first machine learning algorithm resembling intelligence.

If our assumptions about the essence of intelligence are in fact true then we would have infinite possibilities for confidence machine applications. Any real-world problem, any industrial process, any decision making system that require intelligent answers may employ confidence machine as a decision engine. Independent confidence machines can be grouped to make a final decision when one machine is unable to make one. For example, a confidence machine produces an answer that has low confidence or credibility. Another confidence

machine can analyse all such cases and make a decision based on this secondary knowledge. If this is still not satisfactory then a third confidence machine can analyse the output of two previous machines with or without the original input and so on until a confident decision can be made. This process sounds dangerously similar to a process of human thinking and reasoning. Now we have a machine that is assumed to produce intelligent answers and which can be grouped to simulate thinking - isn't it the idea behind the Artificial Intelligence?

Chapter 6

6. Experiments and comparisons

6.1 Introduction

Experimental results and their comparison is one of the most important parts in researching new machine learning algorithms. However, this part is often overlooked and not sufficiently covered. Traditionally, there are certain benchmark data sets that consist of a training set and a test set. Researchers run their learning algorithms on a training set and then apply learnt knowledge to a test set. Standard measures for assessing the quality are error rate, false positives/negatives, square error, average error etc. The main problem with this approach is that a fact that one learning algorithm gives lower error rate on a given training and test sets does not mean that this algorithm is generally better. Benchmark data sets can be obtained in a variety of ways and in each particular case the data sets selection plays a very important role in the performance of an algorithm. Some algorithms work better if data comes from a particular known distribution, and others work better if data is independently and identically distributed. It is impossible to say something about comparative performance of two algorithms simply by comparing their error rates on a given set of data. To enhance the comparison other measures are often introduced. These measures reflect other performance attributes of a learning algorithm for example number of false positives and true negatives, generalisation performance, VC dimension and so on. Combining two or more measures in theory should lead to a more decisive answer, however in practice it is even more difficult to compare because the measures could be contradicting each other. Now we have a situation whereby we need to compare learning algorithms somehow, we need to use more than one performance measure and at the same time we do not want to be bound by the given benchmark data splits.

It seems that to achieve a more or less objective comparison we should not be limited to a single experiment with given data splits. Instead, for each performance measure we need to create a large number of our own random splits and

perform experiments with each learning algorithm under question. Then average results can be compared and some meaningful conclusions can be made. Therefore, experiments in this thesis are performed using the following plan:

a) Select a list of benchmark data sets to participate in our experiments.

b) Apply SVM and Nearest Neighbours bare prediction learning algorithms to the benchmark data splits to obtain the optimal parameters required to achieve the best published results.

c) For each benchmark data set:

    1) Merge the training and test sets and produce 50 random splits.
    2) Using the optimal parameters discovered in step b) apply SVM, NN, TCM-SVM, TCM-NN, ICM-SVM and ICM-NN to each random split obtained in the previous step to produce a set of performance measures.
    3) Draw corresponding graphs and calculate average performance measures over 50 splits.

d) Compare graphs and average results individually by the data sets, learning algorithms and performance criteria.

6.2 Results

6.2.1 USPS

USPS is a dataset of handwritten digits, created by the US Postal Service. It is one of the most popular pattern recognition datasets and widely used as a benchmark for experiments with learning machines. It consists of 7291 training examples and 2007 test examples. USPS database contains a few obvious errors that make the overall performance degraded hence many researches have tried to modify the original data set slightly to improve the recognition (see, e.g., [64], [65], [66]). We will be working with the original data set for which Figure 6.2.1.1 shows the results achieved using various algorithms.

Figure 6.2.1.1: Previous results on USPS data set.

Algorithm	**Errors (%)**
Optimal margin Classifier [67]	4.6
SVM [68]	4.0
Virtual SVM [70]	3.2
Virtual SVM, local kernel [69]	3.0
Human error rate [71]	2.5
Human error rate [72]	1.51

We will be using the original Support Vector Machine as a bare prediction learning algorithm for TCM and ICM. SVM requires many parameters such as kernel, bound on alphas and working set size adjusted to obtain optimum performance on a particular data. We have performed a number of experiments with SVM on the original USPS data with various parameters and found the following parameters that achieve the error rate of 4%:

Kernel: $\mathrm{K}(x, y) = \frac{(x \cdot y)^5}{256}$

Bound on alphas (c): 175

Working set size: 400

Hence, the parameters above will be used in our experiments with USPS data set.

Following the plan setup in the previous section, we need to create 50 random splits of combined USPS training and test sets and then run our selection of

learning algorithms on these splits. We deal with pattern recognition and use Support Vector Machine and Nearest Neighbours bare prediction learning algorithms. Hence, we need to run the following algorithms on each split of the data:

a) Support Vectors Machine
b) Support Vector Transductive Confidence Machine
c) Support Vector Inductive Confidence Machine
d) Nearest Neighbours
e) Nearest Neighbours Transductive Confidence Machine
f) Nearest Neighbours Inductive Confidence Machine

The main problem we face when running the algorithms above is the speed of the Transductive Confidence Machine. As shown in Section 3.5.2 multi-class TCM requires $2c$ learning iterations for each test example. In our case it means $2 \cdot 10 \cdot 2007 = 40140$ learning iterations to classify all test examples. Unfortunately, this will take an unreasonable amount of time even on a fast computer. Hence, we will need to cut the size of training and test sets to a manageable size. The data sets for the first experiment are obtained by randomly selecting 50 examples for the test set and 1000 examples for the training set. For the calibration and proper training sets we further split the obtained training set randomly into 490 and 510 examples respectively. Figure 6.2.1.2 shows the amalgamated average results for this experiment.

Figure 6.2.1.2: Average results on USPS data set over 50 random splits with 1000 training (incl. 490 calibration) and 50 test examples.

Algorithm	**Errors %**
SV	2.96
SV TCM	3.28
SV ICM	4.88
NN	5.56
NN TCM	5.56
NN ICM	7.44

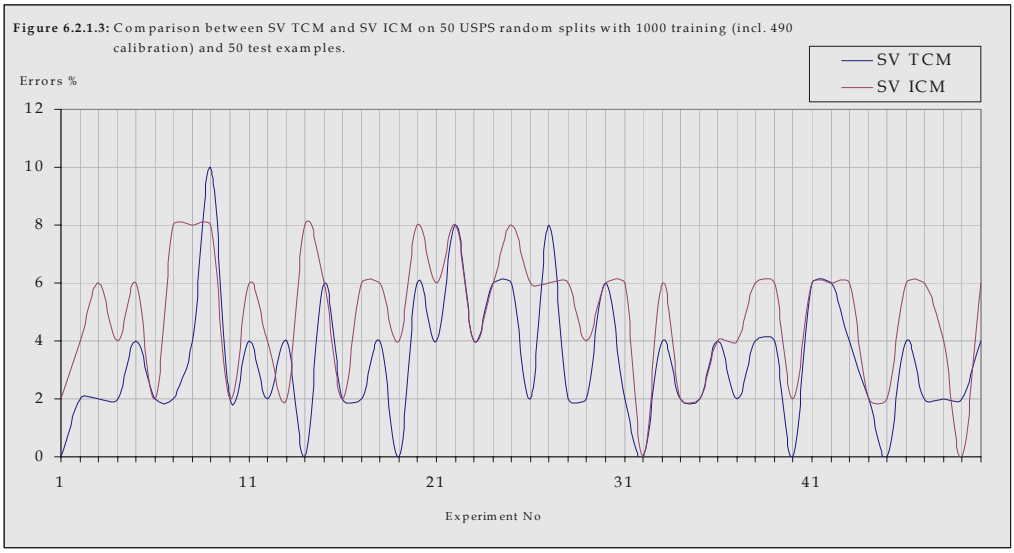


Figure 6.2.1.3: Comparison between SV TCM and SV ICM on 50 USPS random splits with 1000 training (incl. 490 calibration) and 50 test examples.

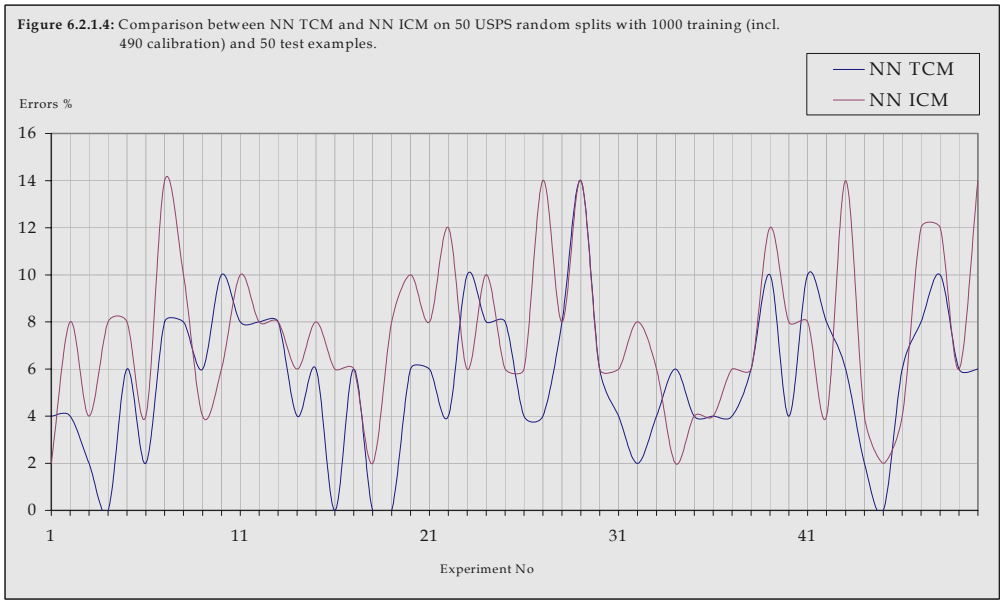


Figure 6.2.1.4: Comparison between NN TCM and NN ICM on 50 USPS random splits with 1000 training (incl. 490 calibration) and 50 test examples.

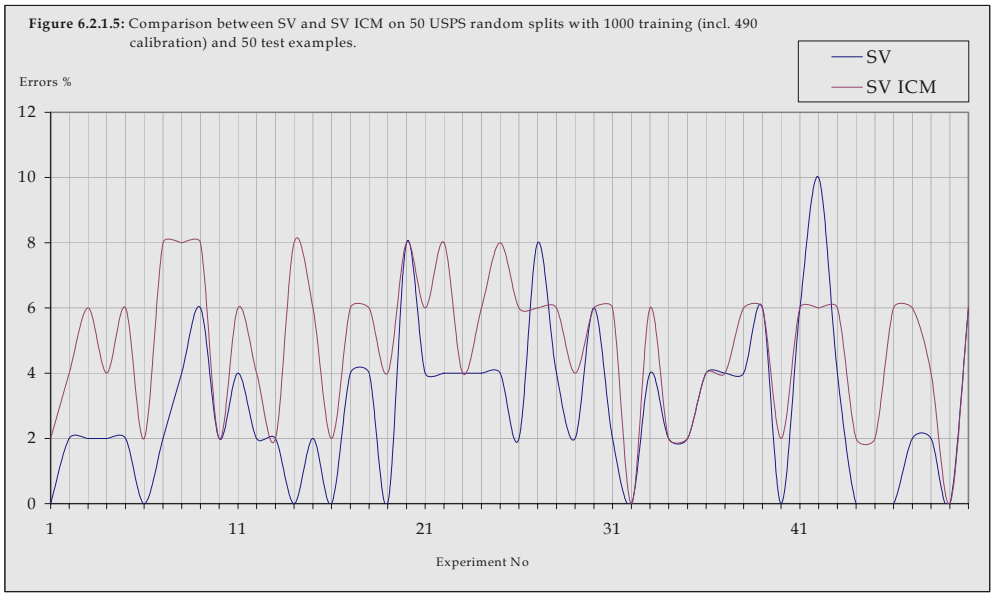


Figure 6.2.1.5: Comparison between SV and SV ICM on 50 USPS random splits with 1000 training (incl. 490 calibration) and 50 test examples.

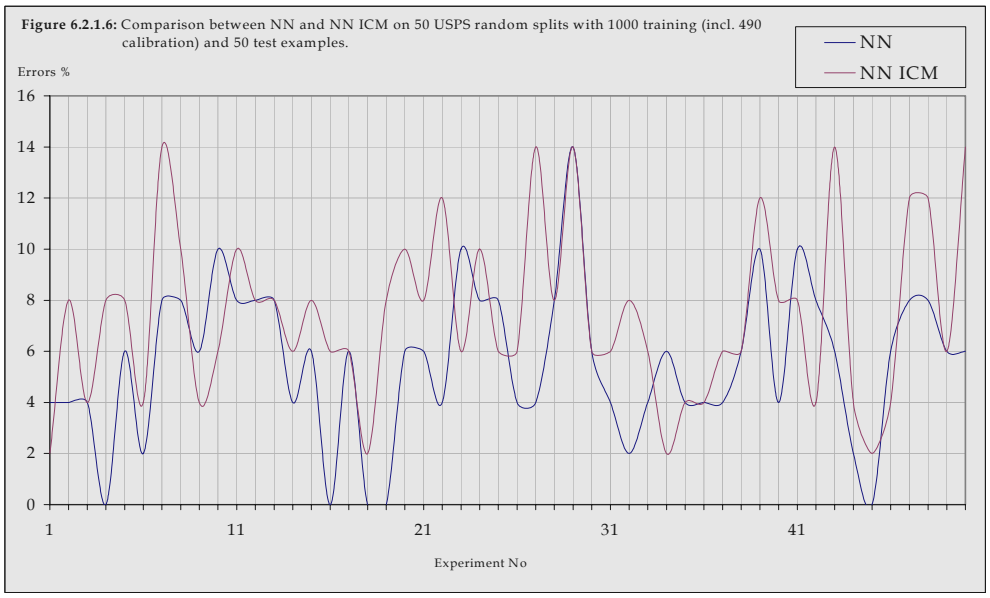


Figure 6.2.1.6: Comparison between NN and NN ICM on 50 USPS random splits with 1000 training (incl. 490 calibration) and 50 test examples.

Figure 6.2.1.7: Average results including performance measures on USPS data set over 50 random splits with 1000 training (incl. 490 calibration) and 50 test examples.													
Algorithm	**Errors %**	**Perf 1**	**Perf 2**	**Perf 3**	**Perf 6 (10%)**	**Perf 6 (5%)**	**Perf 6 (1%)**	**Perf 5 (10%)**	**Perf 5 (5%)**	**Perf 5 (1%)**	**Perf 4 (10%)**	**Perf 4 (5%)**	**Perf 4 (1%)**
SV	2.96												
SV TCM	3.28	3.87	4.22	4.00	60.36	60.36	60.36	0	0	0	39.64	39.64	39.64
SV ICM	4.88	3.47	3.76	3.17	13.04	6.84	2.68	0	0	0	86.96	93.16	97.32
NN	5.56												
NN TCM	5.56	5.69	6.21	5.45	10.84	6.08	1.24	0	0	0	89.16	93.92	98.76
NN ICM	7.44	3.50	3.79	3.38	13.04	6.64	2.84	0	0	0	86.96	93.36	97.16

Performance 1. $\gamma_1 = \sum_{\Omega \in T \times Y} \frac{-\ln p_\Omega}{|T \times Y|}$, minus logarithm of the typical p-value obtained on the overall test set.

Performance 2. $\gamma_2 = \sum_{x \in T} \sum_{y \in Y \setminus \{y_i\}} \frac{-\ln p_{(x,y)}}{|T|(|Y|-1)}$, minus logarithm of the typical p-value on the test set while ignoring potentially large p-values for true classifications.

Performance 3. $\gamma_3 = \sum_{i \in T} \frac{-\ln p_i^{(2)}}{|T|}$, minus logarithm of the typical second largest p-value obtained for each example in the test set.

Performance 4. $\gamma_4^\delta = \frac{\#\{i : p_i^{(y_i)} < 1-\delta\}}{|T|} \cdot 100\%$, percentage of time the program is unable to predict an example because its credibility is not exceeding the confidence level $1-\delta$.

Performance 5. $\gamma_5^\delta = \frac{\#\{i : p_i^{(y)} \geq 1-\delta\}}{|T|} \cdot 100\%$, more than one classification because at least two p-values exceed the confidence level $1-\delta$.

Performance 6. γ_6^δ - absolutely certain prediction because only one p-value exceeds the confidence level $1-\delta$.

For the detailed information about these performance measures see Section 3.6.

Figures 6.2.1.3, 6.2.1.4, 6.2.1.5 and 6.2.1.6 show the split by split comparative performance graphs. Figure 6.2.1.7 is an extended version of the Figure 6.2.1.2 which shows average results for all performance measures for each algorithm on the same splits.

Analysis of these figures allows us to make the following preliminary conclusions about the experiment. On USPS data, using exactly the same parameters for bare prediction learning algorithms in all implementations, having relatively small training and test data sets (1000 and 50 examples), having proper training and calibration sets of almost the same small size (510 and 490 examples):

a) SV and NN TCM are slightly worse than raw implementations of SV and NN.
b) SV and NN ICM are worse than their TCM implementation.
c) TCM produces lower typical p-values compared to ICM (this is expected simply because more examples involved in calculating p-value in a case of TCM).
d) Typical p-values produced by SV ICM are only slightly better than the ones produced by NN ICM. However, they are much better in the case of TCM.
e) In the case of SV as an underlying bare prediction learning algorithm, TCM is significantly better at producing absolutely certain predictions under standard significance levels (see Performance 6). However, ICM is actually slightly better than TCM in the case of NN.
f) Similar to the previous point, in the case of SV, TCM produces significantly lower number of predictions when application is unable to make a prediction under standard significance levels (see Performance 4). However, again ICM is actually slightly better than TCM in the case of NN.
g) TCM and ICM (both SV and NN) are equivalent when it comes to the number of at least two possible predictions for test examples.
h) All algorithms based on SV are considerably better than the ones based on NN.

The previous set of experiments dealt with classifying a given set of test examples and the statistics was mainly concentrated on the overall error rate on all such examples. In the real world, however, it is almost never the case that we can produce a certain answer for every question. Even in the simplest case

when we are given a list of handwritten images of digits, quite often different people might assign different classes to some of the images. This happens because some of the images could have poor quality or they could be written carelessly. For example, Figure 6.2.1.1 shows that even a human operator has 2.5% of mistakes when trying to identify digits from the USPS test data set. It is worth noting that the fact of making such mistakes is actually a good thing. To be more precise, a human operator was probably not sure about some of the images during the identification process. However, to be compared with bare prediction learning algorithms they had no choice but to give an answer from 0 to 9 even though they might not wish to. With the invention of Confidence Machines we now have a completely different situation. For every answer machine produces two quality measures that can tell us a quality of the answer. Similar to the human operator, machine now has an alternative option of rejecting any possible answer due to the lack of knowledge, which is quite an intelligent thing to do in such situations. Using the same splits of data as in the previous set of experiments we may specify a certain threshold for confidence and credibility. We agree that if both confidence and credibility for an answer exceed this threshold then this answer is certain and worth considering. If, however, it is otherwise then we skip the answer. Figures 6.2.1.8, 6.2.1.9, 6.2.1.10 and 6.2.1.11 show how the acceptance of answers and the test error rate depend on various thresholds. Analysis of these graphs shows that on these particular splits of the USPS data set we can get a considerable improvement in the error rate by skipping 5-10% of uncertain answers identified by using a threshold in a region of 0.1-0.15. Bare prediction learning algorithms do not even give a chance for doing this kind of things. They have to produce an answer regardless of whether it is sensible or not. In contrast, Confidence Machines can eliminate answers that do not make sense for one reason or another and hence perform better on the real-world problems.

To illustrate the situation with non-sensible predictions we have taken a few test examples that have been incorrectly predicted by the SV TCM and SV ICM with confidence and credibility not exceeding a threshold of 15%. Figures 6.2.1.12 and 6.2.1.13 show such examples. On these figures we can see how these digits actually look and the p-values obtained for each postulated label. Even grown up human beings with many years of experiencing written digits would have trouble identifying what is written in these cases! Whatever answer we prefer, it would not be certain and in this case we would normally refrain from the definite answer simply stating that we do not know what these digits

are because they look like none of the known to me digits. This seems to be the only intelligent answer in this case and the output of the confidence machines suggests exactly this kind of answer with an explanation as well. Figures 6.2.1.12 and 6.2.1.13 show how close the given example is to every known digit thus explaining its decision to refrain from the answer. In contrast, bare prediction learning algorithm would not be able to explain its answer and produce an intelligent decision in difficult situations such as these two.

Analysis of Figures 6.2.1.8, 6.2.1.9, 6.2.1.10, 6.2.1.11, 6.2.1.12 and 6.2.1.13 allows us to say that a confidence machine is able to produce intelligent answers in real-world situations by rejecting some of the answers due to its lack of knowledge or due to an inconsistency in the test data. Inconsistency in the test data could be caused by having examples from the different domain compared to the training data learnt by the confidence machine. In some sense inconsistency in the test data is the same as the lack of knowledge in the domain where the unknown test example comes from.

Figure 6.2.1.8: Test errors depending on the acceptability threshold for SV TCM on 50 USPS random splits with 1000 training (incl. 490 calibration) and 50 test examples.

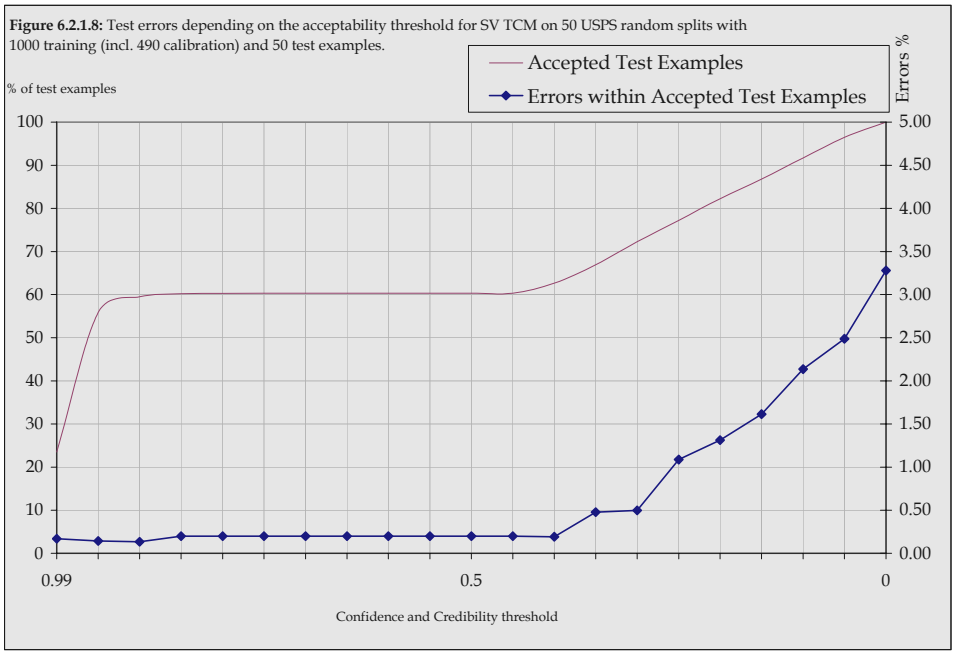


Figure 6.2.1.9: Test errors depending on the acceptability threshold for SV ICM on 50 USPS random splits with 1000 training (incl. 490 calibration) and 50 test examples.

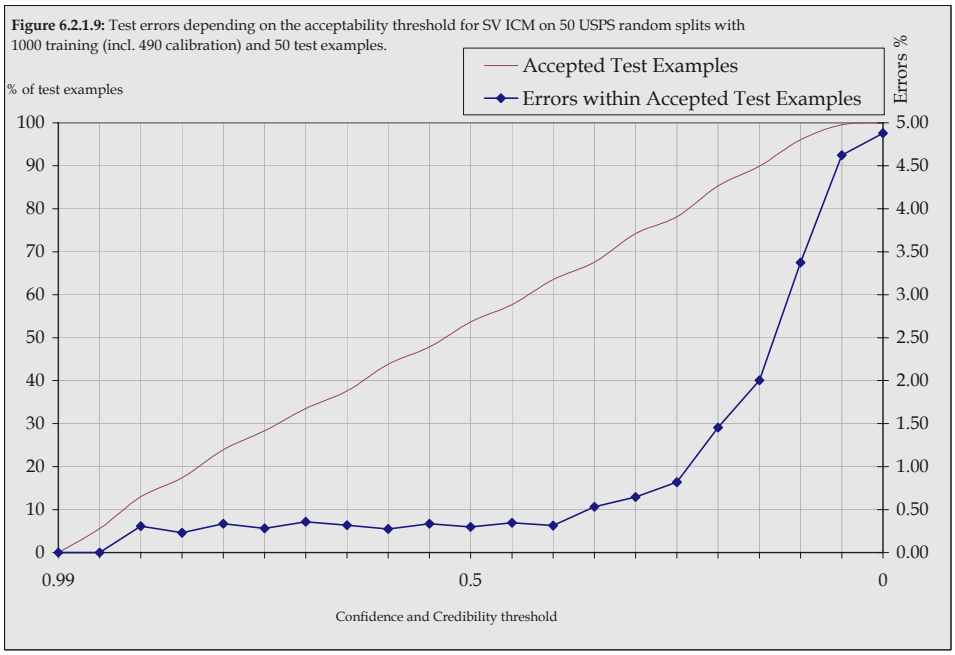


Figure 6.2.1.10: Test errors depending on the acceptability threshold for NN TCM on 50 USPS random splits with 1000 training (incl. 490 calibration) and 50 test examples.

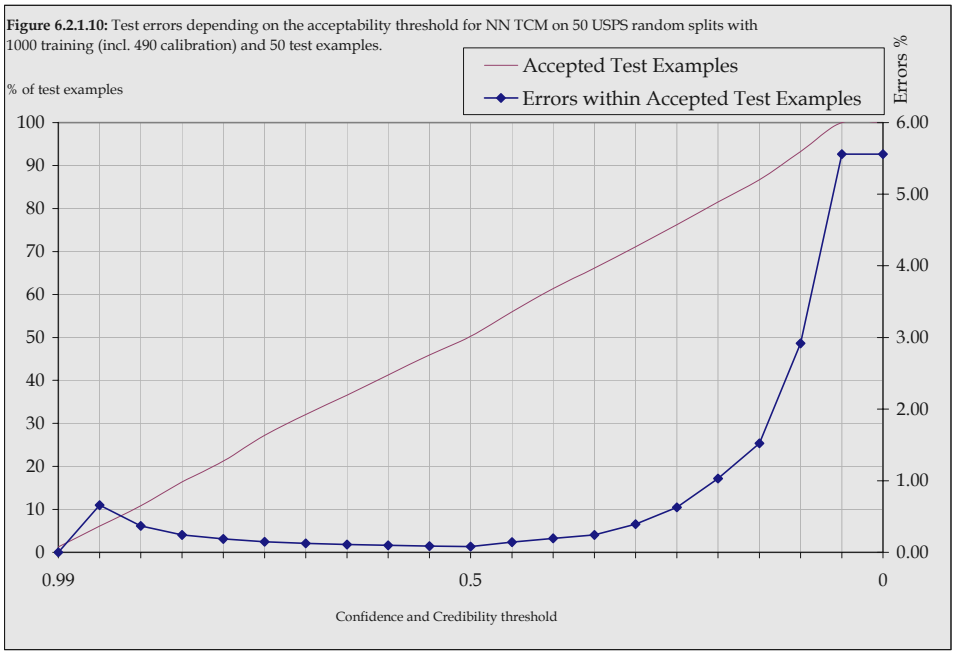


Figure 6.2.1.11: Test errors depending on the acceptability threshold for NN ICM on 50 USPS random splits with 1000 training (incl. 490 calibration) and 50 test examples.

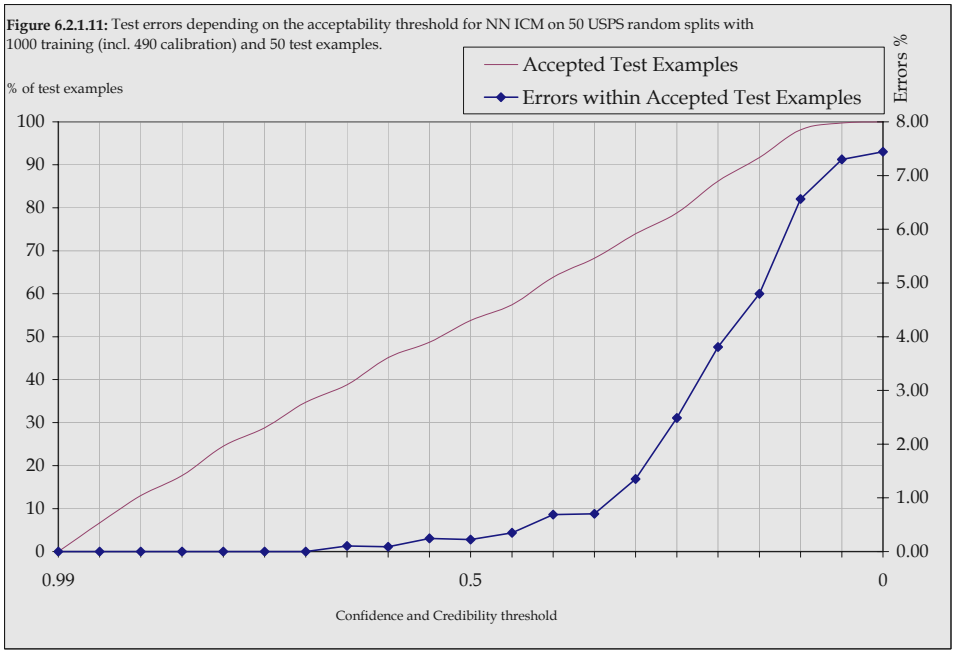


Figure 6.2.1.12: P-values for an uncertain (Confidence and Credibility are no greater than 15%) test example with true label '0'.

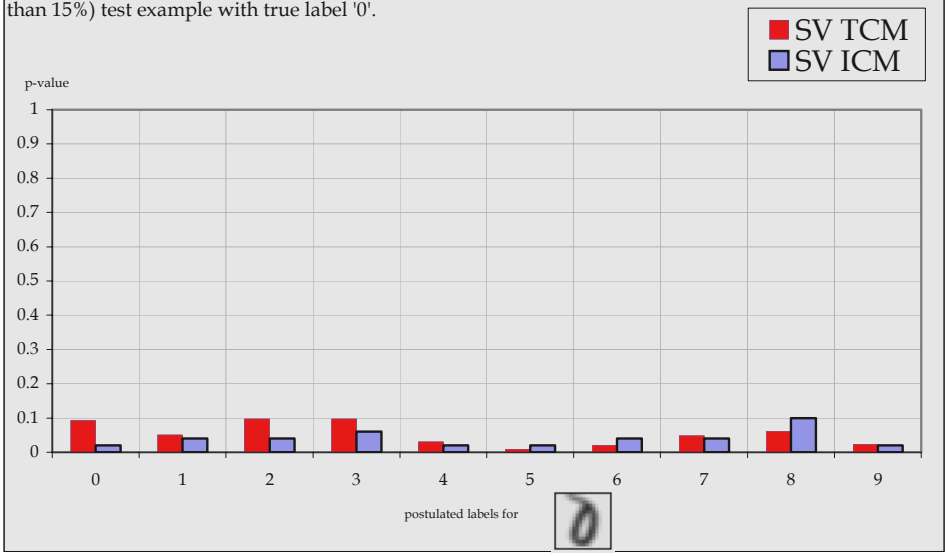
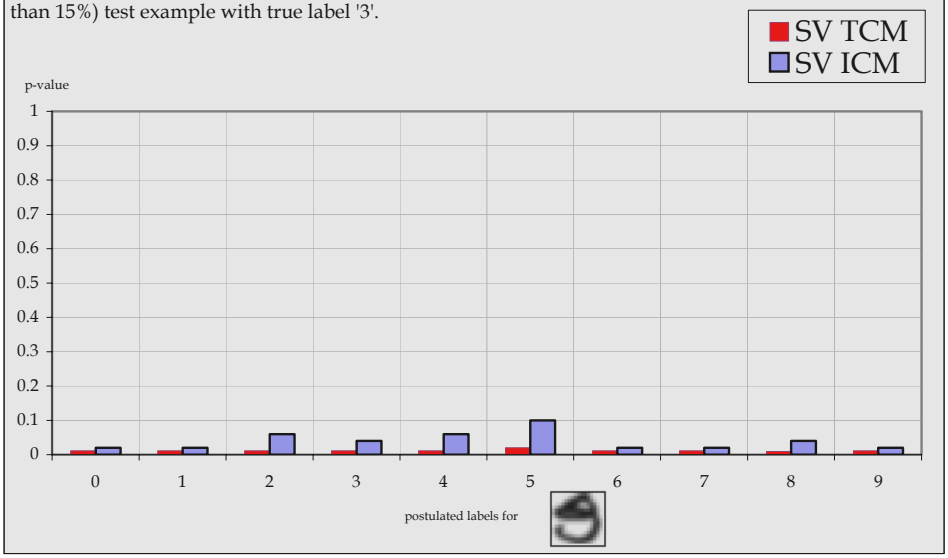


Figure 6.2.1.13: P-values for an uncertain (Confidence and Credibility are no greater than 15%) test example with true label '3'.



The previous set of experiments was done by obtaining random splits of data with 50 examples for the test set and 1000 examples for the training set. For the calibration and proper training sets we further split the obtained training set randomly into 490 and 510 examples respectively. These sets are very small especially when we obtain the calibration and the proper training sets. We only have 5 test examples of each class, 49 examples in the calibration and 51 examples in the proper training set respectively in each class. Obviously the amount of information that can be learnt from this data is limited but we are bound by the TCM training time hence we are forced to use such small sets. In the next experiment we increase the size of the training data to see how it affect the performance. The bare prediction learning algorithms parameters will remain the same as in the previous experiment.

In this experiment we also have 50 test examples, however we have 990 examples in a calibration set and 1000 examples in a proper training set. This gives us about twice as many examples to learn from. Figure 6.2.1.15 shows the average results of this experiment using the same format as Figure 6.2.1.7. Analysis of these figures allows us to make the following preliminary conclusions about the experiment. On USPS data, using exactly the same parameters for bare prediction learning algorithms in all implementations, having slightly larger training and test data sets (1990 and 50 examples), having proper training and calibration sets of almost the same but larger size (1000 and 990 examples) compared to the previous experiment:

a) Results are improved considerably (by ~30%) compared to the previous exactly the same experiment when having 510 and 490 examples in the proper training and the calibration sets respectively.
b) Error rate difference between SV ICM and SV TCM is reduced from 33% to 23%.
c) Error rate difference between NN ICM and NN TCM is reduced from 25% to 22%
d) NN and NN TCM are again equivalent in terms of error rate.
e) Error rate difference between SV TCM and SV is reduced from 23% to 19.4%.
f) In terms of other performance measures we have mixed results. All algorithms produced lower typical p-values however, while SV TCM is more certain in its predictions SV ICM, NN TCM and NN ICM are less

certain in their predictions during this experiment compared to the previous one.

Figure 6.2.1.14: Average results on USPS data set over 50 random splits with 1990 training (incl. 990 calibration) and 50 test examples.

Algorithm	**Errors %**
SV	2.16
SV TCM	2.68
SV ICM	3.48
NN	4.44
NN TCM	4.44
NN ICM	5.68

Figure 6.2.1.15: Average results including performance measures on USPS data set over 50 random splits with 1990 training (incl. 990 calibration) and 50 test examples.													
Algorithm	**Errors %**	**Perf 1**	**Perf 2**	**Perf 3**	**Perf 6 (10%)**	**Perf 6 (5%)**	**Perf 6 (1%)**	**Perf 5 (10%)**	**Perf 5 (5%)**	**Perf 5 (1%)**	**Perf 4 (10%)**	**Perf 4 (5%)**	**Perf 4 (1%)**
SV	2.16												
SV TCM	2.68	3.96	4.33	4.16	67.84	67.84	67.84	0.00	0.00	0.00	32.16	32.16	32.16
SV ICM	3.48	4.05	4.40	3.67	11.12	4.92	0.88	0.00	0.00	0.00	88.88	95.08	99.12
NN	4.44												
NN TCM	4.44	6.15	6.72	5.93	10.08	5.40	1.00	0.00	0.00	0.00	89.92	94.60	99.00
NN ICM	5.68	4.12	4.48	4.03	10.60	5.36	0.88	0.00	0.00	0.00	89.40	94.64	99.12

Performance 1. $\gamma_1 = \sum_{\Omega \in T \times Y} \frac{-\ln p_\Omega}{|T \times Y|}$, minus logarithm of the typical p-value obtained on the overall test set.

Performance 2. $\gamma_2 = \sum_{t \in T} \sum_{y \in Y \setminus \{y_t\}} \frac{-\ln p_{(t,y)}}{|T|(|Y|-1)}$, minus logarithm of the typical p-value on the test set while ignoring potentially large p-values for true classifications.

Performance 3. $\gamma_3 = \sum_{t \in T} \frac{-\ln p_t^{(2)}}{|T|}$, minus logarithm of the typical second largest p-value obtained for each example in the test set.

Performance 4. $\gamma_4^\delta = \frac{\#\{t \in T : p_t^{(2)} < 1-\delta\}}{|T|} \cdot 100\%$, percentage of time the program is unable to predict an example because its credibility is not exceeding the confidence level $1-\delta$.

Performance 5. $\gamma_5^\delta = \frac{\#\{t \in T : p_t^{(2)} \geq 1-\delta\}}{|T|} \cdot 100\%$, more than one classification because at least two p-values exceed the confidence level $1-\delta$.

Performance 6. γ_6^δ - absolutely certain prediction because only one p-value exceeds the confidence level $1-\delta$.

For the detailed information about these performance measures see Section 3.6.

Inductive Confidence Machine has one tricky moment that separates it from Transductive Confidence Machine and other learning algorithms. As seen from the experiments above we start from obtaining random training and test sets, however ICM also requires a calibration set which is randomly obtained from the previously obtained training set. The fact is that for every pair of training and test sets the end result of ICM is not deterministic, and in reality it is a matter of chance. This is a clear disadvantage over other learning algorithms when we try to compare their performances like for like. Figures 6.2.1.3, 6.2.1.4, 6.2.1.5 and 6.2.1.6 clearly show us how widely results vary from one selection of training and test sets to another, which implies that a random selection of the calibration set also significantly influences the prediction performance of ICM. This is shown on Figure 6.2.1.16.

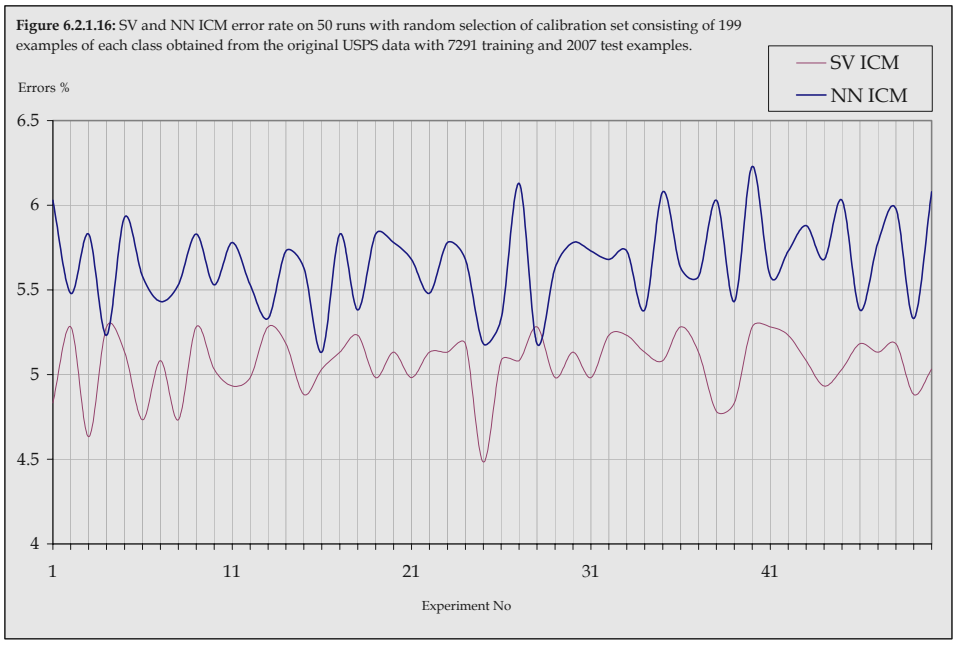


Figure 6.2.1.16: SV and NN ICM error rate on 50 runs with random selection of calibration set consisting of 199 examples of each class obtained from the original USPS data with 7291 training and 2007 test examples.

We have run SV and NN ICM 50 times on the full original USPS data set consisting of 7291 training examples and 2007 test. Each time we randomly selected a calibration set consisting of 199 examples of each class from the original training set leaving the rest to be the proper training set. We have tried various parameters such as different size of calibration set, different SV parameters and found that the best average error rate is obtained by having 199 examples of each class in the calibration set and the same SV parameters as explained in the beginning of this section. Figures 6.2.1.16 and 6.2.1.17 show the results of 50 runs with the best combination of parameters. It is easily noticeable that the er-

ror rate fluctuates from 4.48% to 5.28% in this particular case. However, during all the experiments the error rate spread was reaching 6.87% high. For practical use and fair comparison we obviously need to find a way of obtaining an optimum selection of calibration set that provides the best result.

The simplest and the least efficient way of achieving this is to produce a large number of random selections and then chose the best one. It may work for laboratory experiments however it might not be useful in practice because we might not have a sample test set to try them on or due to time requirements.

An idea about another possible way of achieving this comes from analysing how predictions are made in ICM. Figure 3.6.2.1 above shows the algorithmic procedure for ICM. P-value for a postulated label t is obtained by essentially comparing strangeness of the new example with strangeness values of each example in the calibration set with the same classification t. The rank of the new example's strangeness value within calibration set examples determines its p-value. P-value itself identifies how typical the new example with the postulated label t is compared to examples with classification t in the calibration set. It is natural therefore to assume that to produce a fair p-value we should have a calibration set that covers the knowledge domain given in the training set entirely. In other words, in the calibration set we should have enough examples to represent all possible variations within each class. By analogy, to predict a digit out of all possible ones we include examples of all possible digits in the calibration set. Therefore, to predict a single digit we need to include all possible variations of this digit. This makes our logic sound.

Practically we can run an algorithm such as SV or NN on the training set alone and obtain strangeness values for each example in this set. Then we sort examples of each class in order of their strangeness and evenly select a required number of examples to be moved into the calibration set. By evenly selecting examples we make sure that examples in the calibration set broadly cover the entire strangeness spread observed in the given training set. In the following experiment we select a calibration set using this algorithm and immediately obtain the performance better than the average of multiple random repeats. Figure 6.2.1.18 shows the results of this experiment.

Figure 6.2.1.17: Average results including performance measures on 50 runs with random selection of calibration set consisting of 199 examples of each class obtained from the original USPS data with 7291 training and 2007 test examples.													
Algorithm	**Errors %**	**Perf 1**	**Perf 2**	**Perf 3**	**Perf 6 (10%)**	**Perf 6 (5%)**	**Perf 6 (1%)**	**Perf 5 (10%)**	**Perf 5 (5%)**	**Perf 5 (1%)**	**Perf 4 (10%)**	**Perf 4 (5%)**	**Perf 4 (1%)**
SV ICM	5.06	4.78	5.19	4.72	9.70	4.96	1.13	0.00	0.00	0.00	90.30	95.04	98.87
NN ICM	5.66	4.77	5.17	4.72	10.29	5.15	1.13	0.00	0.00	0.00	89.71	94.85	98.87

Performance 1. $\gamma_1 = \sum_{\Omega \in T \times Y} \frac{-\ln p_\Omega}{|T \times Y|}$, minus logarithm of the typical p-value obtained on the overall test set.

Performance 2. $\gamma_2 = \sum_{t \in T} \sum_{y \in Y \setminus \{y_t\}} \frac{-\ln p_{(t,y)}}{|T|(|Y|-1)}$, minus logarithm of the typical p-value on the test set while ignoring potentially large p-values for true classifications.

Performance 3. $\gamma_3 = \sum_{t \in T} \frac{-\ln p_t^{(2)}}{|T|}$, minus logarithm of the typical second largest p-value obtained for each example in the test set.

Performance 4. $\gamma_4^\delta = \frac{\#\{t \in T : p_t^{(\max)} < 1-\delta\}}{|T|} \cdot 100\%$, percentage of time the program is unable to predict an example because its credibility is not exceeding the confidence level $1-\delta$.

Performance 5. $\gamma_5^\delta = \frac{\#\{t \in T : p_t^{(2)} \geq 1-\delta\}}{|T|} \cdot 100\%$, more than one classification because at least two p-values exceed the confidence level $1-\delta$.

Performance 6. γ_6^δ - absolutely certain prediction because only one p-value exceeds the confidence level $1-\delta$.

For the detailed information about these performance measures see Section 3.6.

Figure 6.2.1.18: Results including performance measures on the original USPS data with 7291 training and 2007 test examples with deterministic selection of calibration set consisting of 199 examples.													
Algorithm	**Errors %**	**Perf 1**	**Perf 2**	**Perf 3**	**Perf 6 (10%)**	**Perf 6 (5%)**	**Perf 6 (1%)**	**Perf 5 (10%)**	**Perf 5 (5%)**	**Perf 5 (1%)**	**Perf 4 (10%)**	**Perf 4 (5%)**	**Perf 4 (1%)**
SV ICM	4.83	4.75	5.14	4.53	10.06	4.48	1.20	0.00	0.00	0.00	89.94	95.52	98.80
NN ICM	5.63	4.73	5.12	4.57	11.06	5.73	1.20	0.00	0.00	0.00	88.94	94.27	98.80

Performance 1. $\gamma_1 = \sum_{\Omega \in T \times Y} \frac{-\ln p_\Omega}{|T \times Y|}$, minus logarithm of the typical p-value obtained on the overall test set.

Performance 2. $\gamma_2 = \sum_{t \in T} \sum_{y \in Y \setminus \{y_t\}} \frac{-\ln p_{(t,y)}}{|T|(|Y|-1)}$, minus logarithm of the typical p-value on the test set while ignoring potentially large p-values for true classifications.

Performance 3. $\gamma_3 = \sum_{t \in T} \frac{-\ln p_t^{(2)}}{|T|}$, minus logarithm of the typical second largest p-value obtained for each example in the test set.

Performance 4. $\gamma_4^\delta = \frac{\#\{t \in T : p_t^{(\max)} < 1-\delta\}}{|T|} \cdot 100\%$, percentage of time the program is unable to predict an example because its credibility is not exceeding the confidence level $1-\delta$.

Performance 5. $\gamma_5^\delta = \frac{\#\{t \in T : p_t^{(2)} \geq 1-\delta\}}{|T|} \cdot 100\%$, more than one classification because at least two p-values exceed the confidence level $1-\delta$.

Performance 6. γ_6^δ - absolutely certain prediction because only one p-value exceeds the confidence level $1-\delta$.

For the detailed information about these performance measures see Section 3.6.

Instead of trying to find an optimum split of the original training data set using some algorithm, we can always produce a sufficient number of splits and then select the best one. In the previous experiments a selection of calibration set was a matter of chance for each combination of training and test sets with only one random attempt at obtaining the calibration set. To produce a fair comparison the best selection of calibration set should take place for each selection of training and test sets. In the following experiment we achieve the ultimate comparison between our algorithms. For each of the 50 random splits of training and test sets we also produce 50 random splits of proper training and calibration sets. Then we calculate and compare performance between the best, average and algorithmically selected calibration set selections. Figure 6.2.1.19 shows the amalgamated results for this experiment. Analysis of this experiment allows us to make the following conclusions:

a) Confirming the previously found results with ICM on full USPS data, for each selection of training and test sets results of ICM vary significantly and depend on the calibration set selection.
b) SV TCM performs slightly worse than SV and NN TCM again performs similar to NN.
c) Selecting a calibration set using the algorithm described above produces slightly worse result than the average of 50 random calibration sets in both SV ICM and NN ICM.
d) The most important result here is that the best out of 50 random calibration sets outperforms raw SV, SV TCM, raw NN and NN TCM significantly. This means that for every selection of training and test sets we can always find a calibration set for ICM that outperforms any other learning algorithms. Moreover, the improvement in error rate is very significant reaching up to 48% in the case of SV and 63% in the case of NN.

Figure 6.2.1.19: Average results including performance measures on USPS data set over 50 random splits with 1990 training (incl. 990 calibration) and 50 test examples.

Algorithm	Errors %	Perf 1	Perf 2	Perf 3	Perf 6 (10%)	Perf 6 (5%)	Perf 6 (1%)	Perf 5 (10%)	Perf 5 (5%)	Perf 5 (1%)	Perf 4 (10%)	Perf 4 (5%)	Perf 4 (1%)
SV	2.92												
SV TCM	3.28	3.94	4.30	4.13	67.96	67.96	67.96	0.00	0.00	0.00	32.04	32.04	32.04
SV ICM Sel Cal	4.80	4.07	4.42	3.75	10.88	5.04	1.12	0.00	0.00	0.00	89.12	94.96	98.88
SV ICM Rand Average	4.74	4.06	4.40	3.71	11.12	5.09	1.05	0.00	0.00	0.00	88.88	94.91	98.95
SV ICM Rand Best Cal	**1.52**	**4.11**	**4.47**	**3.88**	**10.72**	**4.56**	**1.08**	**0.00**	**0.00**	**0.00**	**89.28**	**95.44**	**98.92**
NN	4.92												
NN TCM	4.92	6.14	6.71	5.92	9.72	4.44	0.80	0.00	0.00	0.00	90.28	95.56	99.20
NN ICM Sel Cal	6.72	4.10	4.45	3.93	10.88	4.64	1.08	0.00	0.00	0.00	89.12	95.36	98.92
NN ICM Rand Average	6.45	4.11	4.46	3.99	10.39	4.68	0.93	0.00	0.00	0.00	89.61	95.32	99.07
NN ICM Rand Best Cal	**1.84**	**4.13**	**4.49**	**4.08**	**10.12**	**4.36**	**1.04**	**0.00**	**0.00**	**0.00**	**89.88**	**95.64**	**98.96**

Performance 1. $\gamma_1 = \sum_{\omega \in T \times Y} \frac{-\ln p_\omega}{|T \times Y|}$, minus logarithm of the typical p-value obtained on the overall test set.

Performance 2. $\gamma_2 = \sum_{t \in T} \sum_{y \in Y \setminus \{y_t\}} \frac{-\ln p_{(t,y)}}{|T|(|Y|-1)}$, minus logarithm of the typical p-value on the test set while ignoring potentially large p-values for true classifications.

Performance 3. $\gamma_3 = \sum_{t \in T} \frac{-\ln p_t^{(2)}}{|T|}$, minus logarithm of the typical second largest p-value obtained for each example in the test set.

Performance 4 . γ_4^δ - percentage of time the program is unable to predict an example because its credibility is not exceeding the confidence level $1-\delta$.

Performance 5. γ_5^δ - more than one classification because at least two p-values exceed the confidence level $1-\delta$.

Performance 6. γ_6^δ - absolutely certain prediction because only one p-value exceeds the confidence level $1-\delta$.

For the detailed information about these performance measures see Section 3.6.

The experiments above have shown that results vary significantly depending on parameters used for our algorithms as well as on how a calibration set is selected for ICM. In Section 5.4 we talked about the possibility of model selection based on confidence machine performance measures defined in Section 3.7. We described a procedure that can be used to perform a model selection and a procedure how to show that it works. In the following experiment we follow this procedure to identify whether model selection can be achieved in the case of USPS data set. Figure 6.2.1.20 shows the results of this experiment.

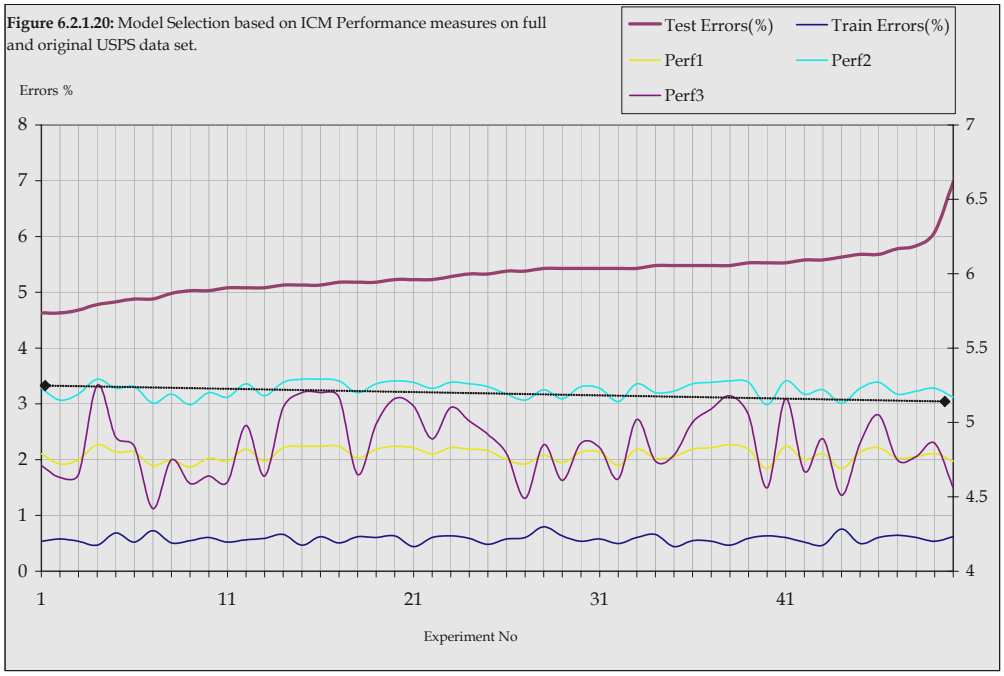


Figure 6.2.1.20: Model Selection based on ICM Performance measures on full and original USPS data set.

On this figure we show training and testing errors for 50 random selections of the calibration set obtained from the original USPS training set. Performance measures calculated on the proper training set are also shown. The graph is sorted in the ascending testing error rate which should allow us to see whether there is any dependency between the testing error rate and the performance measures. Analysis of this graph tells us that there is no conclusive dependency between the training and the testing error rates. This confirms what we have already said about using the training error rate as a quality measure in deciding which algorithm or parameter selection is better. However, there seems to be a weak (the correlation coefficient $r = -0.1$) but conclusive dependency between the testing error rate and the Performance 3. The Performance 3 reflects the second largest p-value and the confidence of a prediction. It is higher when this p-

value is lower. The highest obtained Performance 3 corresponds to the 4th best testing error rate. Two other performance measures are also the best at this point. This means that our performance measures 1, 2 and 3 do indicate the quality of training and show how well a confidence machine will perform on future unknown examples. This is exactly what is required to be achieved by the model selection.

6.2.2 MNIST

MNIST is another popular database of handwritten digits built by LeCun [73] research group at AT&T lab. This database is significantly larger than USPS. It contains 60,000 examples in the training set and 10,000 examples in the test set. Each example is represented as a vector of pixels with 784 attributes that describe an image of 28 by 28 pixels. Figure 6.2.2.1 shows previous results on this dataset.

Figure 6.2.2.1: Previous results on MNIST data set.

Algorithm	**Errors (%)**
3-NN [74]	2.4
LeNet1 [74]	1.7
400-300-10 network [74]	1.6
Polynomial SVM [75]	1.4
Tangent Distance [74]	1.1
LeNet4 [74]	1.1
LeNet5 [74]	0.9
Virtual polynomial SVM [76]	0.8
Boosted LeNet4 [74]	0.7

Based on the experience from the experiments with USPS described in the previous section, we would run the following experiment with MNIST:

a) Randomly obtain 1990 training and 50 test examples from the combined original MNIST training and test sets.
b) Run SV, SV TCM, NN and NN TCM on these sets.
c) Obtain 50 random splits of 1000 and 990 examples for proper training and calibration sets respectively. Run SV ICM and NN ICM on these sets and select the best obtained results.
d) Obtain the "best calibration set" using the methodology explained in the previous section. Run SV ICM and NN ICM using this split.
e) Repeat from Step a) 50 times.
f) Produce overall graphs and average performance measures for these 50 runs.

We have performed a number of experiments with SVM on the original MNIST data with various parameters and found the following parameters that achieve the error rate of 1.64%:

Kernel: $\mathrm{K}(x, y) = \frac{(x \cdot y)^7}{784}$

Bound on alphas (C): 10

Working set size: 1500

Hence, the parameters above will be used in our experiments with MNIST data set.

As concluded in the previous section, this experiment will give us an exact comparison between algorithms on MNIST data. Results of this experiment are shown on Figures 6.2.2.2, 6.2.2.3 and 6.2.2.4. Analysis of these figures allows us to make the following conclusions. On MNIST data, using exactly the same parameters for bare prediction learning algorithms in all implementations, having 1990 training and 50 test examples, having proper training and calibration sets of similar size (1000 and 990 examples):

a) SV TCM and NN TCM are slightly worse than SV and NN respectively.
b) Selecting a calibration set using the algorithm described above produces slightly better result than the average of 50 random calibration sets for SV ICM and other way around for NN ICM.
c) The most important result here is that again the best out of 50 random calibration sets outperforms raw SV, SV TCM, raw NN and NN TCM significantly. This is exactly the same result as in the case of USPS data already described. In this case the improvement in error rate is also very significant reaching up to 50% in the case of SV and 47% in the case of NN.
d) We are observing the same pattern of results as we had in the USPS experiment confirming the ICM performance characteristics independent of data.

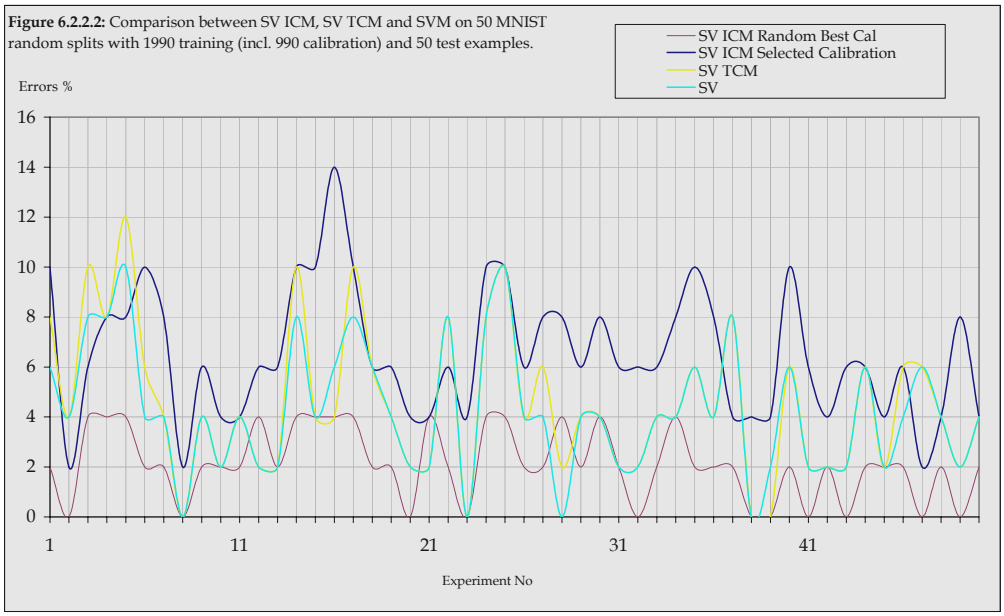


Figure 6.2.2.2: Comparison between SV ICM, SV TCM and SVM on 50 MNIST random splits with 1990 training (incl. 990 calibration) and 50 test examples.

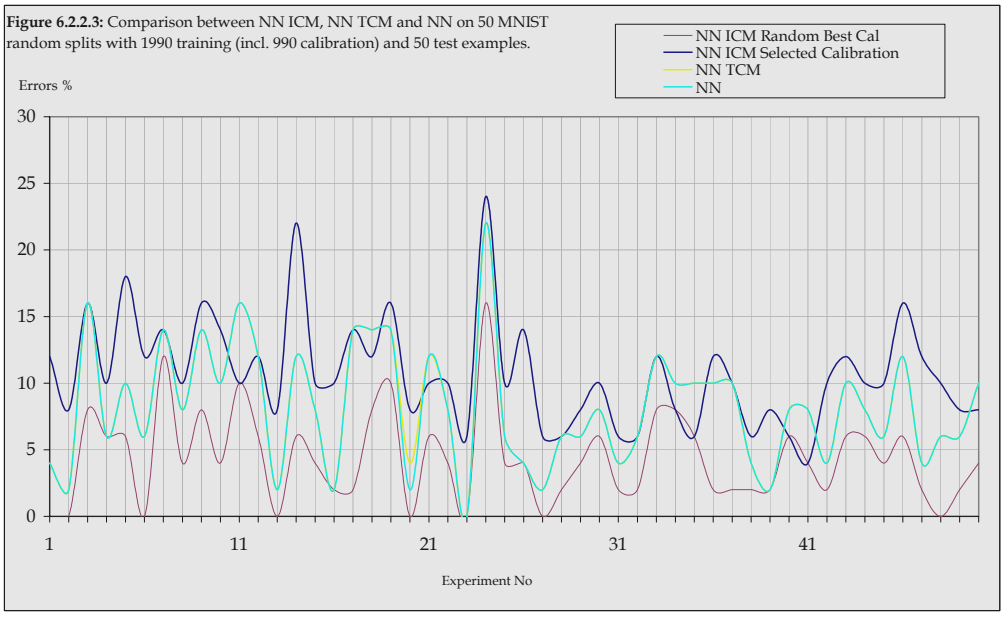


Figure 6.2.2.3: Comparison between NN ICM, NN TCM and NN on 50 MNIST random splits with 1990 training (incl. 990 calibration) and 50 test examples.

Figure 6.2.2.4: Average results including performance measures on MNIST data set over 50 random splits with 1990 training (incl. 990 calibration) and 50 test examples.													
Algorithm	**Errors %**	**Perf 1**	**Perf 2**	**Perf 3**	**Perf 6 (10%)**	**Perf 6 (5%)**	**Perf 6 (1%)**	**Perf 5 (10%)**	**Perf 5 (5%)**	**Perf 5 (1%)**	**Perf 4 (10%)**	**Perf 4 (5%)**	**Perf 4 (1%)**
SV	4.24												
SV TCM	4.52	4.94	5.40	4.70	46.92	46.92	46.92	0.00	0.00	0.00	53.08	53.08	53.08
SV ICM Sel Cal	6.52	3.98	4.32	3.34	11.56	5.16	0.96	0.00	0.00	0.00	88.44	94.84	99.04
SV ICM Rand Average	6.65	3.97	4.31	3.31	11.42	5.17	1.07	0.00	0.00	0.00	88.58	94.83	98.93
SV ICM Rand Best Cal	**2.12**	**4.01**	**4.35**	**3.40**	**11.40**	**5.28**	**1.16**	**0.00**	**0.00**	**0.00**	**88.60**	**94.72**	**98.84**
NN	8.20												
NN TCM	8.24	6.01	6.57	5.37	10.32	5.80	1.00	0.00	0.00	0.00	89.68	94.20	99.00
NN ICM Sel Cal	10.72	3.99	4.33	3.66	10.92	5.36	1.00	0.00	0.00	0.00	89.08	94.64	99.00
NN ICM Rand Average	10.35	3.99	4.33	3.67	11.80	5.25	0.92	0.00	0.00	0.00	88.20	94.75	99.08
NN ICM Rand Best Cal	**4.36**	**4.02**	**4.37**	**3.73**	**11.84**	**5.36**	**0.88**	**0.00**	**0.00**	**0.00**	**88.16**	**94.64**	**99.12**

Performance 1. $\gamma_1 = \sum_{\Omega \in T \times Y} \frac{-\ln p_\Omega}{|T \times Y|}$, minus logarithm of the typical p-value obtained on the overall test set.

Performance 2. $\gamma_2 = \sum_{t \in T} \sum_{y \in Y \setminus \{y_t\}} \frac{-\ln p_{(t,y)}}{|T|(|Y|-1)}$, minus logarithm of the typical p-value on the test set while ignoring potentially large p-values for true classifications.

Performance 3. $\gamma_3 = \sum_{t \in T} \frac{-\ln p_t^{(2)}}{|T|}$, minus logarithm of the typical second largest p-value obtained for each example in the test set.

Performance 4 . γ_4^δ - percentage of time the program is unable to predict an example because its credibility is not exceeding the confidence level $1-\delta$.

Performance 5. γ_5^δ - more than one classification because at least two p-values exceed the confidence level $1-\delta$.

Performance 6. γ_6^δ - absolutely certain prediction because only one p-value exceeds the confidence level $1-\delta$.

For the detailed information about these performance measures see Section 3.6.

The only other experiment that needs to be done is to run SV ICM on the whole MNIST data with various selections of calibration set. This experiment will be done using the following scenario:

a) Run the procedure from the Step b) using calibration sets of size 199, 999 and 1999 examples in each class.
b) Obtain a random selection of calibration and proper training set from the original MNIST training set.
c) Run SV ICM on this split and the original MNIST test set.
d) Repeat from Step b) 50 times.

This experiment will show ICM performance on the whole MNIST data set. Figure 6.2.2.5 shows the results of this experiment.

Figure 6.2.2.5: SV ICM results including performance measures on the original MNIST data set over 50 random splits with 199, 999 and 1999 examples of each class in the calibration sets.

Algorithm	Errors %	Perf 1	Perf 2	Perf 3	Perf 6 (10%)	Perf 6 (5%)	Perf 6 (1%)	Perf 5 (10%)	Perf 5 (5%)	Perf 5 (1%)	Perf 4 (10%)	Perf 4 (5%)	Perf 4 (1%)
SV ICM Average (Cal 199)	2.11	4.81	5.23	4.86	12.09	5.72	1.10	0.00	0.00	0.00	87.91	94.28	98.90
SV ICM Best (Cal 199)	1.63	4.84	5.27	5.11	12.20	5.71	1.25	0.00	0.00	0.00	87.80	94.29	98.75
SV ICM Average (Cal 999)	1.68	6.11	6.68	5.74	11.38	5.61	1.04	0.00	0.00	0.00	88.62	94.39	98.96
SV ICM Best (Cal 999)	1.56	6.20	6.78	6.10	11.30	5.67	0.90	0.00	0.00	0.00	88.70	94.33	99.10
SV ICM Average (Cal 1999)	1.70	6.61	7.24	6.00	11.41	5.63	1.07	0.00	0.00	0.00	88.59	94.37	98.94
SV ICM Best (Cal 1999)	1.51	6.66	7.30	6.10	11.60	5.70	1.20	0.00	0.00	0.00	88.40	94.30	98.80

Performance 1. $\gamma_1 = \sum_{\Omega \in T \times Y} \frac{-\ln p_\Omega}{|T \times Y|}$, minus logarithm of the typical p-value obtained on the overall test set.

Performance 2. $\gamma_2 = \sum_{t \in T} \sum_{y \in Y \setminus \{y_t\}} \frac{-\ln p_{(t,y)}}{|T|(|Y|-1)}$, minus logarithm of the typical p-value on the test set while ignoring potentially large p-values for true classifications.

Performance 3. $\gamma_3 = \sum_{t \in T} \frac{-\ln p_t^{(2)}}{|T|}$, minus logarithm of the typical second largest p-value obtained for each example in the test set.

Performance 4 . γ_4^δ **-** percentage of time the program is unable to predict an example because its credibility is not exceeding the confidence level $1-\delta$.

Performance 5. γ_5^δ **-** more than one classification because at least two p-values exceed the confidence level $1-\delta$.

Performance 6. γ_6^δ **-** absolutely certain prediction because only one p-value exceeds the confidence level $1-\delta$.

For the detailed information about these performance measures see Section 3.6.

6.2.3 ABDO

ABDO is a database of patients diagnosed with one out of nine abdominal pain causes. The records had been collected by Dr. Gunn of the Bangour Hospital in Roxburgh and then passed to us by Dr. Nixon of the Western General Hospital. The database contains patient records on Appendicitis, Diverticulitis, Perforated Peptic Ulcers, Non-specific Abdominal Pain, Cholecystitis, Intestinal Obstructions, Pancreatitis, Renal Colic and Dyspepsia. Each patient is diagnosed by 33 symptoms whereby each symptom could be a combination of values. For example, symptom 4 called *Pain-site present* can take the following values at the same time: left upper quadrant, right lower quadrant and left half. To record the data in a format suitable for our learning machines each symptom is represented as a binary sequence whereby 1 means symptom/value is present and 0 means otherwise. This representations results in 135 attributes data vector for each patient. More information about this dataset and previous results can be found in [56].

The following SVM parameters are used for experiments with this data set as they found to be the best performing:

Kernel: $\mathrm{K}(x, y) = (x \cdot y)^5$

Bound on alphas (c): 150

The data set itself is not very good quality, it contains a lot of noise, missing data and large differences between quantities of examples in each class. Therefore to perform a fair comparison we use the following procedure for this experiment:

a) Randomly obtain 810 training and 50 test examples from the combined original ABDO training and test sets.
b) Run SV, SV TCM, NN and NN TCM on these sets.
c) Obtain 50 random splits of 549 and 261 examples for proper training and calibration sets respectively. Run SV ICM and NN ICM on these sets and select the best obtained results.
d) Obtain the "best calibration set" using the methodology explained in the previous section. Run SV ICM and NN ICM using this split.
e) Repeat from Step a) 50 times.
f) Produce overall graphs and average performance measures for these 50 runs.

Results of this experiment are shown on Figures 6.2.3.1, 6.2.3.2 and 6.2.3.3.

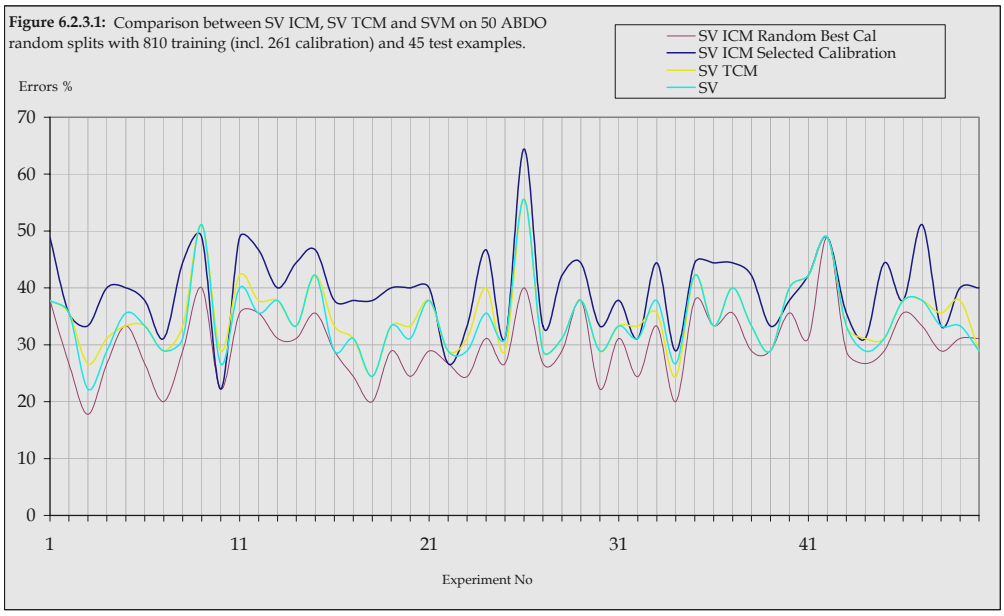


Figure 6.2.3.1: Comparison between SV ICM, SV TCM and SVM on 50 ABDO random splits with 810 training (incl. 261 calibration) and 45 test examples.

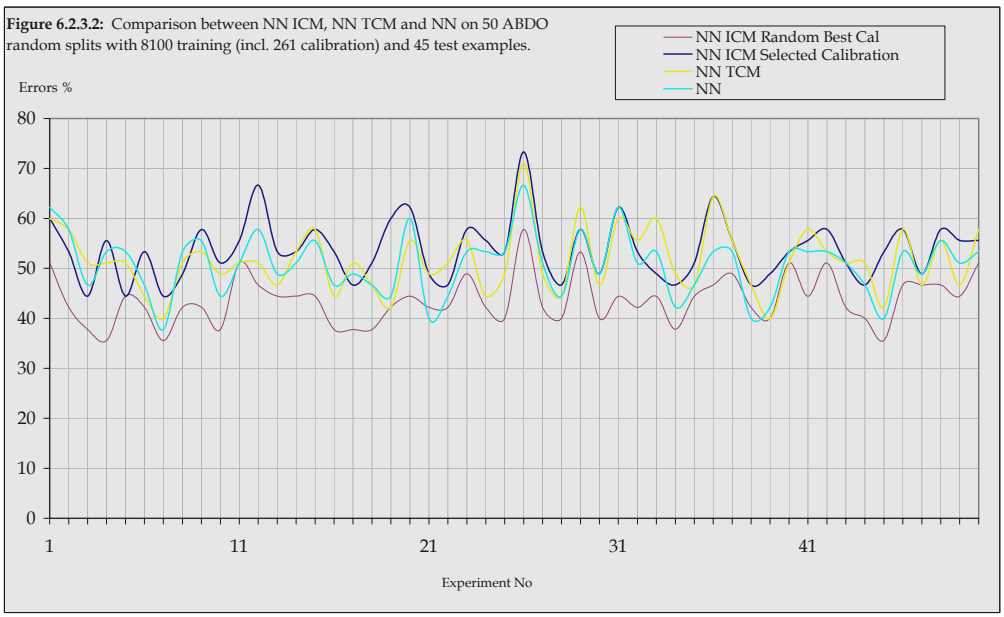


Figure 6.2.3.2: Comparison between NN ICM, NN TCM and NN on 50 ABDO random splits with 8100 training (incl. 261 calibration) and 45 test examples.

Figure 6.2.3.3: Average results including performance measures on ABDO data set over 50 random splits with 810 training (incl. 261 calibration) and 45 test examples.

Algorithm	Errors %	Perf 1	Perf 2	Perf 3	Perf 6 (10%)	Perf 6 (5%)	Perf 6 (1%)	Perf 5 (10%)	Perf 5 (5%)	Perf 5 (1%)	Perf 4 (10%)	Perf 4 (5%)	Perf 4 (1%)
SV	34.31												
SV TCM	34.93	2.07	2.21	1.56	11.25	5.55	4.62	0.00	0.00	0.00	88.75	94.45	95.38
SV ICM Sel Cal	39.81	2.30	2.48	1.25	16.31	8.04	3.91	0.00	0.00	0.00	83.69	91.96	96.09
SV ICM Rand Average	39.52	2.32	2.49	1.24	14.99	7.20	3.42	0.02	0.00	0.00	84.99	92.80	96.58
SV ICM Rand Best Cal	**30.05**	**2.37**	**2.55**	**1.29**	**14.62**	**7.11**	**3.46**	**0.00**	**0.00**	**0.00**	**85.38**	**92.89**	**96.54**
NN	50.80												
NN TCM	51.60	2.41	2.59	1.40	13.91	6.44	1.60	0.00	0.00	0.00	86.09	93.56	98.40
NN ICM Sel Cal	53.74	1.99	2.13	1.17	20.45	9.07	4.80	0.00	0.00	0.00	79.55	90.93	95.20
NN ICM Rand Average	54.00	1.99	2.13	1.16	20.62	9.69	4.80	0.00	0.00	0.00	79.38	90.31	95.20
NN ICM Rand Best Cal	**43.64**	**2.00**	**2.14**	**1.19**	**21.64**	**9.73**	**4.97**	**0.00**	**0.00**	**0.00**	**78.36**	**90.27**	**95.03**

Performance 1. $\gamma_1 = \sum_{\Omega \in T \times Y} \frac{-\ln p_\Omega}{|T \times Y|}$, minus logarithm of the typical p-value obtained on the overall test set.

Performance 2. $\gamma_2 = \sum_{t \in T} \sum_{y \in Y \setminus \{y_t\}} \frac{-\ln p_{(t,y)}}{|T|(|Y|-1)}$, minus logarithm of the typical p-value on the test set while ignoring potentially large p-values for true classifications.

Performance 3. $\gamma_3 = \sum_{t \in T} \frac{-\ln p_t^{(2)}}{|T|}$, minus logarithm of the typical second largest p-value obtained for each example in the test set.

Performance 4 . γ_4^δ - percentage of time the program is unable to predict an example because its credibility is not exceeding the confidence level $1-\delta$.

Performance 5. γ_5^δ - more than one classification because at least two p-values exceed the confidence level $1-\delta$.

Performance 6. γ_6^δ - absolutely certain prediction because only one p-value exceeds the confidence level $1-\delta$.

For the detailed information about these performance measures see Section 3.6.

Analysis of the figures above allows us to make the following conclusions. On ABDO data, using exactly the same parameters for bare prediction learning algorithms in all implementations, having 810 training and 45 test examples, having proper training and calibration sets of similar size (549 and 261 examples):

a) SV TCM and NN TCM are slightly worse than SV and NN respectively.
b) Selecting a calibration set using the algorithm described above produces slightly worse result than the average of 50 random calibration sets for SV ICM and other way around for NN ICM.
c) The most important result here is that again the best out of 50 random calibration sets outperforms raw SV, SV TCM, raw NN and NN TCM. This is exactly the same result as in the case of USPS and MNIST data already described.
d) We are observing the same pattern of results as we had in the USPS and MNIST experiments confirming the ICM performance characteristics independent of data.

If the confidence machine to be used as a tool for medical diagnosis then the output will be presented as shown on Figures 6.2.3.4, 6.2.3.5 and 6.2.3.6. On these figures we see two examples of uncertain prediction and an example of a certain one. According to the ABDO experiment above both uncertain examples were predicted incorrectly and the certain one was a correct diagnosis. Each new diagnosis would be presented with such chart and a doctor can clearly see the picture from all angles. We can clearly observe how credible our predicted diagnosis is and what are the possibilities of other diagnosis in this case. If a prediction clearly stands out from all other possible ones i.e. its p-value is more than 90% and all other ones are less than 10% then a doctor can confidently consider the suggested diagnosis. If, on the other hand, all predictions have very low p-values as shown on Figure 6.2.3.4 or a prediction has a high p-value but one or more other diagnosis also have high p-values as shown on Figure 6.2.3.6 then a doctor might need to re-evaluate the case to find out more information, consult with others and so on.

Figure 6.2.3.4: p-values for an uncertain ABDO test example with true classification 'Renal Colic' and predicted as 'Non-specific abdominal pain'.

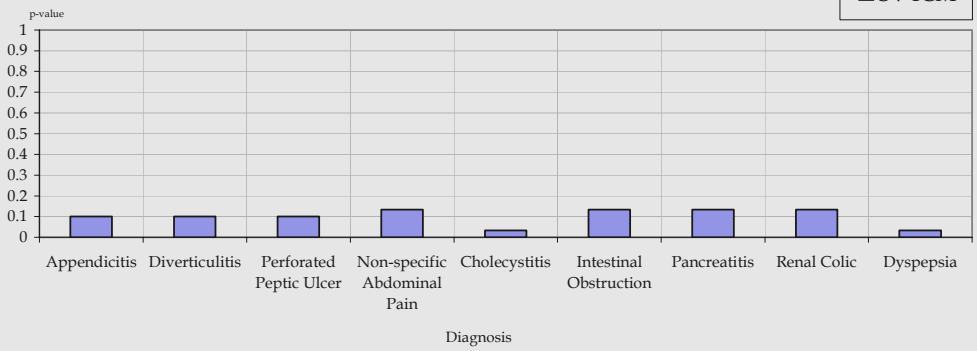


Figure 6.2.3.5: p-values for a certain ABDO test example with true classification 'Appendicitis' and predicted as 'Appendicitis'.

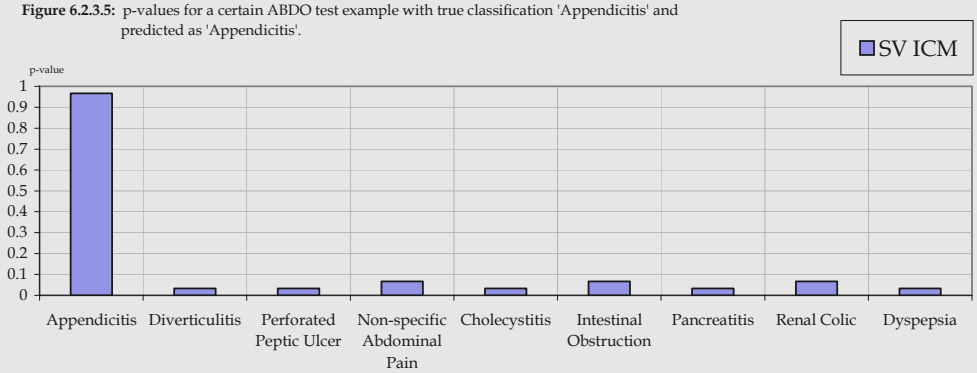
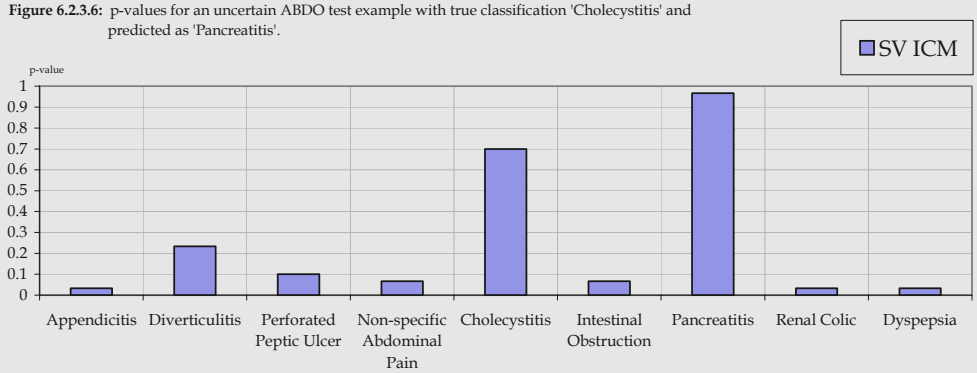


Figure 6.2.3.6: p-values for an uncertain ABDO test example with true classification 'Cholecystitis' and predicted as 'Pancreatitis'.



When a machine learning algorithm is used for such an important task as medical diagnosis it is very critical that parameters are tuned to the best possible performance on yet unseen cases. Therefore, before the confidence machine can practically be used with the data such as ABDO we have to find the best model to be used in terms of the best selection of a calibration set and the best SV parameters. The following experiment is destined to find out whether the model selection idea presented in Section 5.4 holds itself in the case of the ABDO data set. The experiment is done using the same procedure described in Section 5.4 already performed above in the case of USPS. Figure 6.2.3.7 shows the results of this experiment.

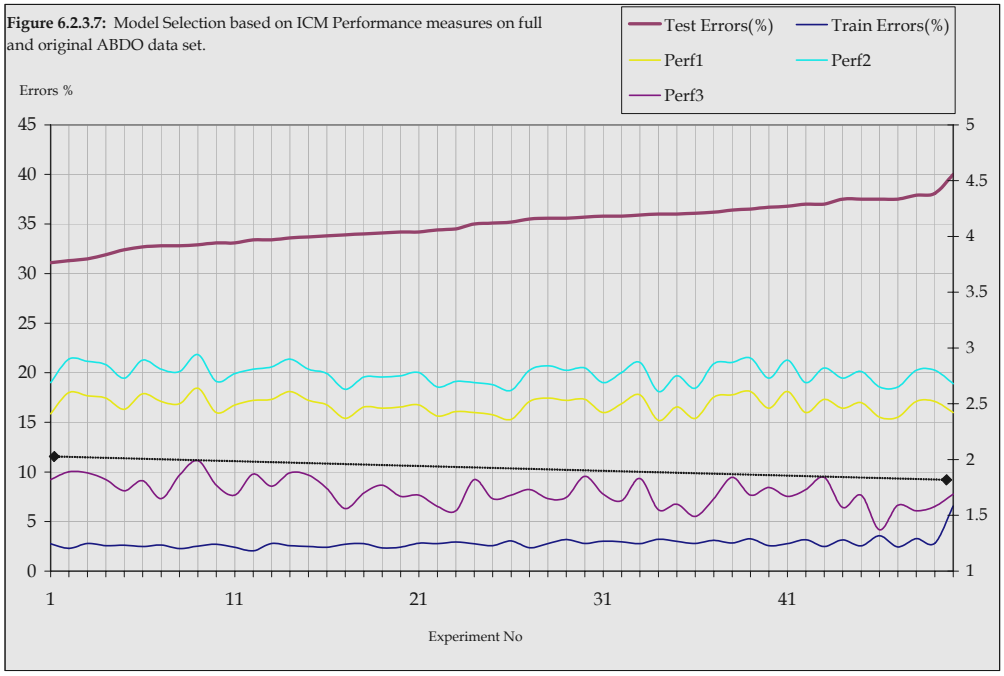


Figure 6.2.3.7: Model Selection based on ICM Performance measures on full and original ABDO data set.

Analysis of this graph indicates a similarity with Figure 6.2.1.20 further confirming the fact that there is a dependency between the performance measures on training data and the testing error rate on yet unseen data. Compared to the same experiment with USPS data (Figure 6.2.1.20) the dependency in this experiment is much stronger (correlation coefficient $r = -0.51$ against $r = -0.1$ in the case of USPS) making it even more conclusive and allowing us to employ the performance measures in selecting the best combination of parameters for practical use.

6.2.4 Protein Fold Classification

Knowledge of the 3D structure of a protein is essential for describing and understanding its function. There is a large number of already sequenced proteins facing a much smaller number of proteins whose structure has been identified. The process of determining the 3D structure involves growing a protein in a very specific environment which is very time consuming and expensive. Theoretically, the 3D structure is encoded in the protein's sequence hence there should be a way to predict the structure based on an existing empirical knowledge. Therefore the machine learning algorithms could potentially be used for this task. Unfortunately, all protein sequences have different length and cannot be trivially entered into a machine learning algorithm.

In Section 5.2 above we described how machine learning researchers are dealing with this problem, and how the confidence machine ideas can be employed to deal with such kind of data too. In this section we compare these methods on a data previously analysed in Markowetz et al. [77] and Grassmann et al. [82]. The data set contains total of 263 sequences whereby 143 of which are used as a training set and 125 as a test set. The data was provided by the authors already separated into the training and test sets, so we used exactly the same split to compare the results like for like. The total number of fold-classes in this set is 42. We employ the second method of constructing a learning machine for such kind of data as described in Section 5.2. As a compression algorithm we use a well known Dynamic Markov Compression, which is mainly a text compression algorithm. Each protein sequence is entered into our algorithm as it is i.e. as a string of letters each representing one out of the twenty possible amino-acids. Amount of data in this data set is very limited making it impossible to run ICM. Hence, we will run SV and SV TCM only and compare it with the results previously obtained in [77]. Results of this experiment are shown on Figure 6.2.4.1 and the comparison is shown on Figure 6.2.4.2.

Figure 6.2.4.1: Results of the Protein Fold Classification using the Kolmogorov complexity approximation kernel.													
Algorithm	**Errors %**	**Perf 1**	**Perf 2**	**Perf 3**	**Perf 6 (10%)**	**Perf 6 (5%)**	**Perf 6 (1%)**	**Perf 5 (10%)**	**Perf 5 (5%)**	**Perf 5 (1%)**	**Perf 4 (10%)**	**Perf 4 (5%)**	**Perf 4 (1%)**
SV(K)	20.00												
SV TCM(K)	21.60	2.51	2.56	2.24	7.2	7.2	7.2	0.00	0.00	0.00	92.8	92.8	92.8

Performance 1. $\gamma_1 = \sum_{\Omega \in T \times Y} \frac{-\ln p_\Omega}{|T \times Y|}$, minus logarithm of the typical p-value obtained on the overall test set.

Performance 2. $\gamma_2 = \sum_{t \in T} \sum_{y \in Y \setminus \{y_t\}} \frac{-\ln p_{(t,y)}}{|T|(|Y|-1)}$, minus logarithm of the typical p-value on the test set while ignoring potentially large p-values for true classifications.

Performance 3. $\gamma_3 = \sum_{t \in T} \frac{-\ln p_t^{(2)}}{|T|}$, minus logarithm of the typical second largest p-value obtained for each example in the test set.

Performance 4 . γ_4^δ - percentage of time the program is unable to predict an example because its credibility is not exceeding the confidence level $1-\delta$.

Performance 5. γ_5^δ - more than one classification because at least two p-values exceed the confidence level $1-\delta$.

Performance 6. γ_6^δ - absolutely certain prediction because only one p-value exceeds the confidence level $1-\delta$.

For the detailed information about these performance measures see Section 3.6.

Figure 6.2.4.2: Comparison of Protein Fold Classification results between standard techniques [77] and the Kolmogorov complexity approximation kernel.

Algorithm	**Errors %**
SV (Kolmogorov)	20.0
SV TCM (Kolmogorov)	21.6
Linear Discriminant Analysis	34.4
k-Nearest-Neighbours	33.6
Neural Networks	28.8
SV (rbf)	23.2

The results show that the Support Vector machine and SV Transductive Confidence Machine comfortably outperform the best previously obtained results. If we had enough data then the Inductive Confidence Machine could also be tried. Following the conclusions made from the results with USPS, NIST and ABDO experiments we may assume that ICM would probably perform even better.

6.2.5 E-mail spam filtering (Ling-Spam)

Unsolicited bulk e-mail (spam) posted to millions of recipients is becoming such a burden that most of the working time is actually spent on analyzing and deleting unwanted messages before reading the legitimate ones. Moreover, most of the time such messages contain computer viruses and unaware recipients get infected. This costs industries a lot of money in repairing the damages and in lost productivity. While many anti-spam filtering solutions are based on filtering e-mails using a list of known keywords there are also solutions based on Bayesian classifiers. Machine learning algorithms should be much better in identifying spam compared to keyword filtering because spammers constantly adjust their techniques to avoid using known patterns. However, machine learning algorithms face the same problem as in the case above with protein sequences. All e-mail messages have different length and therefore have to be transformed before they can be analysed by a learning machine. Even Bayesian classifiers require each message to be presented as a vector of constant size. There are many possibilities in obtaining such vector from a message. The most effective approach described in Androutsopoulos et al. [79] is based on identifying 50 most frequent words in a training set, then constructing a vector with 50 binary attributes indicating the presence of these words for each message. The obtained vectors are then fed into the Bayesian learning algorithms which produces an answer for a new message. Again, even though we use a machine learning algorithm still we rely on counting the most frequent words in order to be able to execute the algorithms. The results are better compared to a simple keyword filtering but the essence is the same and therefore the potential is limited.

With the invention of the confidence machine and the ideas presented in Section 5.2 we may now have a better way of identifying spam. Not only we can deal with the messages directly without worrying about its wording content, but also each new e-mail message will be given measures of confidence and credibility allowing to make an intelligent decision about this message. By varying a confidence and credibility threshold a system can be tuned according to user requirements. Some users may prefer a very secure environment whereby each message must have a very high credibility and confidence to pass through. Other users may wish to allow some spam to make sure that uncommon e-mails get through.

The data set called Ling-Spam is given to us by Androutsopoulos et al. [79]. It contains:

> 2,412 Linguist messages obtained by randomly downloading digests from archives.
>
> 481 spam messages received by the authors of [79] whereby attachments, HTML tags and duplicate spam messages received on the same day were not included.

The first experiment with this data will be done in the same manner as the ones done for USPS, MNIST and ABDO data already described. To obtain a clear picture about comparative performance of our algorithms on this data set we use the following procedure:

a) From the whole data set randomly obtain 2,793 training and 100 test examples.
b) Run SV and SV TCM on these sets.
c) Obtain 50 random splits of 2,595 and 198 examples for proper training and calibration sets respectively from the training set obtained in Step a). Run SV ICM on these sets and select the best and average obtained results.
d) Repeat from Step a) 50 times.
e) Produce overall graphs and average performance measures for these 50 runs.

As a kernel we use the same kernel used for protein fold classification experiments above. It is based on Kolmogorov complexity approximation using Dynamic Markov compression algorithm as described in Section 5.2.

Results of this experiment are shown on Figures 6.2.5.1 and 6.2.5.2. Analysis of these figures tells us that we are observing the same pattern of results as with USPS, MNIST and ABDO data sets. In this case again SV ICM with the best random selection of calibration set is consistently and significantly outperforms SV and SV TCM.

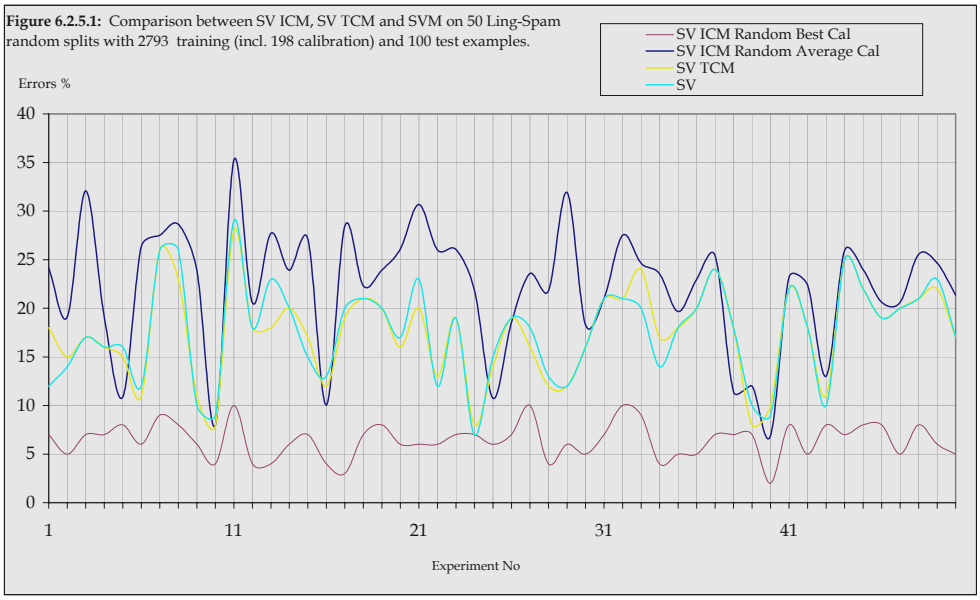


Figure 6.2.5.1: Comparison between SV ICM, SV TCM and SVM on 50 Ling-Spam random splits with 2793 training (incl. 198 calibration) and 100 test examples.

The results show that ideas presented in Section 5.2 work well for this type of data too confirming our assumptions about using Kolmogorov complexity approximation for practical construction of confidence machines and SV kernels. This is the same result as in the previous section on protein fold data. In both cases we dealt with the multi-dimensionality data very easily without knowing or extracting any information from the data before running the confidence machine. Moreover, in both cases we obtained better than previously known results.

Figure 6.2.5.2: Average results including performance measures on Ling-Spam data set over 50 random splits with 2793 training (incl. 198 calibration) and 100 test examples.													
Algorithm	**Errors %**	**Perf 1**	**Perf 2**	**Perf 3**	**Perf 6 (10%)**	**Perf 6 (5%)**	**Perf 6 (1%)**	**Perf 5 (10%)**	**Perf 5 (5%)**	**Perf 5 (1%)**	**Perf 4 (10%)**	**Perf 4 (5%)**	**Perf 4 (1%)**
SV	17.60												
SV TCM	17.52	1.26	2.07	2.48	98.20	98.20	98.20	0.50	0.50	0.50	1.30	1.30	1.30
SV ICM Rand Average	22.22	2.19	3.41	3.68	11.32	5.12	1.03	0.00	0.00	0.00	88.68	94.88	98.97
SV ICM Rand Best Cal	**6.42**	**2.41**	**3.92**	**4.02**	**11.54**	**4.80**	**0.66**	**0.00**	**0.00**	**0.00**	**88.46**	**95.20**	**99.34**

Performance 1. $\gamma_1 = \sum_{\Omega \in T \times Y} \frac{-\ln p_\Omega}{|T \times Y|}$, minus logarithm of the typical p-value obtained on the overall test set.

Performance 2. $\gamma_2 = \sum_{t \in T} \sum_{y \in Y \setminus \{y_t\}} \frac{-\ln p_{(t,y)}}{|T|(|Y|-1)}$, minus logarithm of the typical p-value on the test set while ignoring potentially large p-values for true classifications.

Performance 3. $\gamma_3 = \sum_{t \in T} \frac{-\ln p_t^{(2)}}{|T|}$, minus logarithm of the typical second largest p-value obtained for each example in the test set.

Performance 4 . γ_4^δ - percentage of time the program is unable to predict an example because its credibility is not exceeding the confidence level $1-\delta$.

Performance 5. γ_5^δ - more than one classification because at least two p-values exceed the confidence level $1-\delta$.

Performance 6. γ_6^δ - absolutely certain prediction because only one p-value exceeds the confidence level $1-\delta$.

For the detailed information about these performance measures see Section 3.6.

Chapter 7

7. Conclusions

This thesis was started for the purpose of researching a new pattern recognition machine learning algorithm called Inductive Confidence Machine. ICM is another way of constructing a valid confidence machine based on the original idea by Gammerman and Vovk to produce qualified predictions rather than bare ones as most of the standard learning algorithms are designed to do. Instead of producing just an answer such as "4" when identifying written digits, confidence machine also produces two new measures of quality called Confidence and Credibility. These measures indicate the quality of a prediction from two different perspectives. Credibility shows how well our prediction fits within all known examples with the same classification as the prediction. Confidence indicates impossibility of other predictions. Also each possible prediction is given a p-value indicating its possibility. These additional measures are expected to be very useful in real-world applications because they provide more information about each prediction allowing to make more correct decisions. The first practical implementation of the confidence machine was done using a transductive approach which resulted in a very slow algorithm difficult to use on real data. Inductive approach is a typical way in machine learning field to be able to run slow algorithms much faster, hence it is a basis of the Inductive Confidence Machine.

Before explaining the details of the Inductive Confidence Machine we investigated the whole field of machine learning trying to identify its origin, current state and further development routes. We concluded that machine learning field is probably a result of trying to make computers behave like humans i.e. creating an Artificial Intelligence. However, as this goal has not been feasible machine learning has emerged as a research field concentrating on developing algorithms that can analyse given data and produce output when new data is presented based on information extracted from the original data. Some time later this process was called learning and prediction stating the fact that a computer has to learn some information embedded in a given data and then use the knowledge to make sensible answers on new data from the same knowledge

domain. Analysis of existing machine learning algorithms showed that they are mainly designed to produce just one number and most of the performance comparisons have been based on a prediction error rate. It seems that the current state of machine learning is a rush to come up with more learning algorithms or modify existing ones that could make the error rate lower on some particular data. Learning algorithms such as Support Vector Machine have already achieved very impressive prediction error rates often comparable to human experts. This means that there is not much else that can be done on this front and any error rate improvements are not significant nor practically viable.

We discussed that even though current machine learning algorithms achieve impressive prediction error rate they are still not widely used in industries probably because they cannot justify their predictions. Human operators expect more than just an answer – they expect an intelligent answer. After analysing how human beings make decisions we proposed that they analyse a situation from different angles/perspectives once at a time or in parallel and then select the one that sounds better. Moreover, we seem to be able to quantify all possibilities and then explain by how much each possibility is better or worse compared to others. Then we touched well known aptitude tests which are very commonly used when we need to assess one's logical thinking and abilities. We discussed the role and meaning of these tests making a conclusion that we actually use these tests to measure a subject's intelligence without realising this fact. Moreover, analysis of these tests showed that they are mainly based on identifying patterns and selecting a pattern that fits the situation in the best possible way. In other words it seems like the goal of these tests is to find a least random combination. The one who finds a combination less random than someone else does is considered to be more clever or even more intelligent. This simple example allowed us to make assumptions or even definitions of intelligence outside standard ambiguous definitions. We proposed that intelligence is an ability to find the least random situation out of all possible ones.

Then we moved into describing the idea behind the confidence machine. The original idea was to add quality measures for each prediction produced by standard learning algorithms such as Support Vector Machine. However, during the development of this idea it has grown into a completely new algorithm independent of the existing ones. Trying to come up with a way of producing qualified predictions Vovk and Gammerman turned into the Algorithmic In-

formation Theory and Kolmogorov complexity. By measuring randomness of a sequence which is constructed by concatenating training examples and the new example with all postulated labels we can identify which postulated label results in the least random sequence. Naturally, the postulated label that corresponds to such sequence should be the prediction. Moreover, by quantifying randomness for each postulated label we obtain new measures that indicate quality of our predictions. Following this we described a history behind randomness, how many mathematicians have tried and failed to come up with the precise non-contradictory definition of randomness until the invention of Kolmogorov complexity. Therefore we described ideas about Kolmogorov complexity and how it relates to randomness making it a precise notion rather than just an obscure one. With an ability to define randomness and theoretically calculate it the confidence machine was born in theory.

Comparing the idea of making a prediction by calculating randomness of all possible solutions and the conclusions we made about aptitude tests and human intelligence, we came up with an interesting conclusion that confidence machine is the first machine learning algorithm that could be said to produce intelligent predictions. Firstly because its predictions are obtained in a similar way as human beings make their decisions (according to our assumptions). Secondly because it provides quality measures for all prediction possibilities allowing to look at the problem from different perspectives – the fact, we think, can be recognised as a basis of intelligence.

After theoretical talks about intelligence, randomness and confidence machines we moved into describing the first attempt to build the confidence machine. A few different techniques were described that allow to perform a practical randomness approximation either using standard bare prediction learning algorithms or compression algorithms for approximating Kolmogorov complexity directly. We described in detail how Transductive Confidence Machine was constructed and why it was the first one to be implemented. We investigated the transductive approach in details explaining how to construct a valid confidence machine in binary and multi-class cases. We pointed out the invalid ways also to clearly show the meaning of the confidence machine validity. Transductive Confidence Machine procedures for binary and multi-class cases showed that each new example would require twice as many learning operations as there are classes. Moreover, each new example is present in the learning process meaning the procedure has to be repeated for every single test example

one after another. We use standard learning algorithms for the purpose of randomness approximation and they are usually slow during the learning phase. This makes the whole process very computationally expensive resulting in limited practical use. Transductive approach is a direct move from confidence machine theory into a practical implementation hence it was the first one done. It would be very difficult to improve the speed of learning significantly therefore an inductive approach should be tried for building a faster confidence machine. Next we described how an Inductive Confidence Machine can be constructed whereby we only perform the learning phase once and then classify new examples without re-learning. We started from a naïve implementation to show that it would lead to an invalid confidence machine and then we described valid ways of building a confidence machine for binary and multi-class cases. It was achieved by splitting the original training set into a proper training and calibration sets. All learning is done on the proper training set and then classification is done by approximating randomness of the calibration set combined with the new example using the learnt rules. This way we achieved an enormous speed increase because the learning phase needs to be done only once for any number of test examples. Even if we have only one test example still ICM would be twice as fast compared to the TCM because it only requires the same number of learning operations as there are classes.

The fact that an original training set is split into a proper training and calibration sets before learning and the learning is done only on the proper training set raised many questions about the performance of ICM. It is natural to think that lower number of examples presented for training would result in lower performance. However, it could be a possibility that it might be compensated by the calibration set. To produce a fair comparison between TCM, ICM and bare prediction learning algorithms we have designed specific procedures for our experiments that would clearly show us the comparison. Typically machine learning researchers take a set of data considered to be a standard, run their algorithms on exactly the same split as given to them and compare error rate with previously published results. In some sense it is useful however, when we need to produce a fair comparison between various algorithms we cannot rely on just one split of data. Usually it is not exactly clear how and from where the data was obtained resulting in very different results. In this thesis we performed all experiments independently of previously published results and we made sure that all algorithms were executed under the same conditions and parameters. Also we repeated each experiment 50 or 100 times, each time ob-

taining a new set of data by randomly splitting the original data into required training, proper training, calibration and test sets. Moreover, all experiments were repeated for different types of data such as USPS, MNIST, ABDO, Protein Fold and Ling-Spam to make sure the results are truly domain independent. To prove that the results are properties of the confidence machine rather than of an underlying bare prediction learning algorithm, we also repeated all experiments having Support Vector Machine and Nearest Neighbours as our underlying learning machines. Analysis of all results showed that regardless of data and underlying learning algorithm TCM is slightly worse than the underlying bare algorithm. However, it seems like it is always possible to obtain a proper training and calibration set selection to make sure that ICM significantly outperforms both TCM and the underlying bare prediction learning algorithm. This is a positive result that would allow a wide use of confidence machines in practical applications. Not only ICM performs better than TCM and bare prediction algorithms but it is also significantly faster.

Testing error rate is the primary way to compare standard bare prediction learning algorithms. Confidence machines now have an additional output such as confidence, credibility and p-values requiring other means of comparison between them. We defined six performance measures that could potentially be used for confidence machine comparisons. These measures are based on obtained p-values and on machine's ability to make predictions given some thresholds for confidence and credibility. All presented results show these performance measures that allowed us to make various conclusions about different confidence machine configurations. It was noticed that TCM usually produces lower typical p-values and confidence and credibility are usually higher compared to ICM. It also turned out that our new performance measures could be used for model selection. We performed experiments with USPS and ABDO data sets showing that there is a weak but clear dependency between the performance measures on a training set and the error rate on yet unseen data. This would allow us to use these measure to select an optimum calibration set and other parameters when we need to tune the system to the maximum expected performance on future data. Model selection has always been a difficult area in the machine learning field. Quite often it is data or algorithm specific. It now looks like there is a new way of achieving the model selection that is data independent and can be used in many situations.

Experiments showed that ICM is faster and better compared to TCM and bare prediction learning algorithms. So we looked at the possible applications for confidence machine in general. It is obvious that confidence machine can be used everywhere where existing learning machines are currently used, but also more opportunities are now opened. We looked at how confidence machine would benefit the society in highly skilled applications such as medical diagnosis, whereby it could be used as a very useful tool for doctors or nurses in making diagnosis decisions. We also looked at how confidence machine ideas can be used to deal with data such as texts and biological sequences – the data that cannot be trivially fed into a standard learning machine algorithm. Two new ways of dealing with such data were described allowing to run a learning machine without any knowledge about the data simplifying the process significantly. Experiments also confirmed that the new methods are at least as good as the previously known results obtained using very complicated and data specific techniques.

When talking about the confidence machine applications we also continued discussing our ideas about the essence of intelligence and further routes to creating an artificial intelligence. We looked again at our proposed definition of intelligence, compared it with how confidence machine works and suggested a way of combining confidence machines to simulate a process of reasoning or even thinking. Then we made an interesting conclusion that perhaps the confidence machine is a way forward for machine learning field towards creating a computer system that resembles intelligence or, in other well known words, creating an Artificial Intelligence.

Summarising everything said in this thesis we would like to say that it is clear that Inductive Confidence Machine does outperform existing algorithms in many real-world cases, provides new ways of assessing quality of predictions, allows to perform model selection and gives new ways of dealing with multi-dimension data. We hope that we demonstrated theoretically and proved these facts empirically. Finally, new ideas about machine intelligence were proposed that could be used as a basis for further research in the direction of Artificial Intelligence.

Bibliography

[1] Solomonoff, R. J. (1964). A formal theory of inductive inference. *Information and Control,* 7:1-22 and 224-254, 1964.

[2] Solomonoff, R. J. (1997). A preliminary report on a general theory of inductive inference. Technical Report, ZTB-138, Zator Company, Cambridge, MA, November 1997.

[3] Kolmogorov, A. N. (1965). Three approaches to the quantitive definition of information. *Problems Inform Transmission, 1:1-7, 1965.*

[4] Chaitin, G. J. (1969). On the length of programs for computing finite binary sequences: statistical considerations. *J Assoc. Comput. Mach, 16:145-159, 1969*

[5] Li, M., Vitanyi, P. (1997). *An Introduction to Kolmogorov Complexity and Its Applications.* Springer-Verlag New York, Inc.

[6] Vapnik, V. (1998). *Statistical Learning Theory*. New York: Wiley.

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

[8] Saunders, C., Gammerman, A. and Vovk, V. (1998). Ridge Regression Learning Algorithm in Dual Variable. *Proceedings of the 15th International Conference on Machine Learning (pp. 515-521).*

[9] Finke, M., and Muller, K.-R. (1994). Estimating a-posteriori probabilities using stochastic network models. In *Mozer, Smolensky, Touretzky, Elman, & Weigend, eds., Proceedings of the 1993 Connectionist Models Summer School, Hillsdale, NJ: Lawrence Erlbaum Associates, pp. 324-331*

[10] Platt, J.C. (2000). *Probabilities for Support Vector Machines*. Advances in Large Margin Classifiers. MIT Press.

[11] Saunders, C., Gammerman, A. and Vovk, V. (1999). Transduction with confidence and credibility. *Proceedings of the 16th International Joint Conference on Artificial Intelligence (pp. 722--726).*

[12] Kolmogorov, A. (1956). *Foundations of the theory of probability.* Chelsea publishing company, New York.

[13] Skinner, B.F. (1971). *Beyond Freedom & Dignity.* New York: Bantam.

[14] Humphrys, M. (2003). http://www.compapp.dcu.ie/~humphrys/philosophy.html#turing.test

[15] Webopedia Dictionary. http://www.webopedia.com

[16] Dobrev, D. (2000). http://www.dobrev.com/AI/definition.html

[17] Bennet, C. H., Gacs, P., Li, M., Vitanyi, P., Zurek, W. (1998). Information Distance. *IEEE Transactions on Information Theory (44), 4, 1998, pp.1407-1423.*

[18] Gammerman, A. and Vovk, V. (1999). Algorithmic theory of randomness and its applications in computer learning. Royal Holloway University of London.

[19] Chaitin, G. (1982). Algorithmic information theory. *Encyclopedia of Statistical Sciences, volume 1, pages 38-41.* Wiley, New-York.

[20] Turing, A. (1936). On computable numbers, with an application to Entscheidungsproblem. In *Proceedings of the London Mathematical Society, 42(3-4):230-265.*

[21] Martin-Löf, P. (1966). The definition of random sequences. *Inform. And Control,* 9:602-619.

[22] Kolmogorov, A. (1963). On tables of random numbers. *Sankhya, The Indian Journal of Statistics, pages 369-376.*

[23] Fraser, D. A. S. (1957). *Non-parametric methods in statistics*. Wiley, New-York.

[24] Levin, L. (1973). On the notion of a random sequence. *Soviet Mathematics Doklady*, 14:1413.

[25] Wang, Y. (1996). Randomness and Complexity. *University of Heidelberg, PhD Thesis.*

[26] Lutz, J.H. (1992). Almost everywhere high non-uniform complexity. *J. Computer System Science, 44:220-258.*

[27] Ko, K. (1986). On the notion of infinite pseudorandom sequences. *Theoretical Computer Science, 48:9-33.*

[28] Cox, D.R. and Hinkley, D.V. (1974). *Theoretical statistics*. Chapman and Hall, London.

[29] Melluish, T., Saunders, C., Nouretdinov, I., Vovk, V. (2001). The typicalness framework: a comparison with the Bayesian approach. Technical report, CLRC-TR-01-05, Royal Holloway University of London.

[30] Nouretdinov, I., Melluish, T., Vovk, V. (2001). Ridge Regression Confidence Machine. *Proc. 18th International Conf. On Machine Learning.*

[31] Papadopoulos, H., Proedrou, K., Vovk, V., Gammerman, A. (2001). Inductive Confidence Machines for Regression. In *Proc. ECML-2002.*

[32] Fix, E. and Hodges, J.L. (1951). Discriminatory analysis, non-parametric discrimination. *USAF School of Aviation Medicine, Randolph Field, TX, Project 21-49-004, Report 4, Contract AF41(128)-3.*

[33] Cover, T.M., and Hart, P.E. (1967). Nearest Neighbour Pattern Classification. *IEEE Transactions on Information Theory, IT-13, pp. 21-27.*

[34] Dasarathy, B.V. (1991). Nearest-Neighbour Classification Techniques. *IEEE Computer Society Press, Los Alomitos, CA.*

[35] Melluish, T., Saunders, C., Nouretdinov, I., Vovk, V. (2001). Comparing the Bayes and typicalness frameworks. In *Proceedings of ECML-2001.*

[36] Vapnik, V. (1979). *Estimation of Dependencies Based on Empirical Data.* Nauka, Moscow (English translation: Springer Verlag, New York, 1982).

[37] Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag, New York.

[38] Cortes, C. and Vapnik, V. (1995). Support vector networks. *Machine Learning, 20:273-297.*

[39] Schölkopf, B., Burges, C. and Vapnik, V. (1995). Extracting support data for a given task. *Proceedings, First International Conference in Knowledge Discovery and Data Mining. AAAI Press, Menlo Park, CA.*

[40] Blanz, V., Schölkopf, B., Bulthoff, H., Burges, C., Vapnik, V. and Vetter, T. (1996). Comparison of view-based object recognition algorithms using realistic 3d models. *Artificial Neural Networks - ICANN'96, pages 251-256, Berlin. Springer Lecture Notes in Computer Science, Vol. 1112.*

[41] Schmidt, M. (1996). Identifying speaker with support vector networks. In *Interface 1996 Proceedings, Sydney.*

[42] Osuna, E., Freund, R. and Girosi, F. (1997). Training support vector machines: an application to face detection. In *IEEE Conference on Computer Vision and Pattern Recognition, pages 130-136.*

[43] Joachims, T. (1997). Text categorization with support vector machines. Technical report, LS VIII Number 23, University of Dortmund.

[44] Lodhi, H., Saunders, C., Shaw-Taylor, J., Cristianini, N. and Watkins, C. (2002). Text Classification using String Kernels. *Journal of Machine Learning Research 2.*

[45] Markowetz, F., Edler, L., Vingron, M. (2001). Support Vector Machines for Protein Fold Class Prediction. In *Proc. Mount Holyoke Conference: Statistics in Molecular Biology and Genetics.*

[46] Surkov, D., Chervonenkis, A., Gammerman, A. (2001). Kernel for protein sequences classification. Royal Holloway University of London Technical Report CLRC-TR-01-08.

[47] Muller, K. R., Smola, A., Ratsch, G., Schölkopf, B., Kohlmorgen, J. and Vapnik, V. (1997). Predicting time series with support vector machines. In *Proceedings, International Conference on Artificial Neural Networks, page 999. Springer Lecture Notes in Computer Science.*

[48] Mukherjee, S., Osuna, E. and Girosi, F. (1997). Nonlinear prediction of chaotic time series using a support vector machine. In *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing 7, pages 511-519, Amelia Island, FL.*

[49] Drucker, H., Burges, C., Kaufman, L., Smola, A. and Vapnik, V. (1997). Support vector regression machines. *Advances in Neural Information Processing Systems, 9:155-161.*

[50] Vapnik, V., Golowich, S. and Smola, A. (1996). Support vector method for function approximation, regression estimation, and signal processing. *Advances in Neural Information Processing Systems, 9:281-287.*

[51] Weston, J., Gammerman, A., Stitson, M., Vapnik, V., Vovk, V. and Watkins, C. (1997). Density estimation using support vector machines. Technical report, Royal Holloway College, Report number CSD-TR-97-23.

[52] Stitson, M., Gammerman, A., Vapnik, V., Vovk, V., Watkins, C. and Weston, J. (1997). Support vector ANOVA decomposition. Technical report, Royal Holloway College, Report number CSD-TR-97-22.

[53] Schölkopf, B., Burges, C. and Vapnik, V. (1996). Incorporating invariances in support vector learning machines. *Artificial Neural Networks - ICANN'96, pages 47-52, Berlin. Springer Lecture Notes in Computer Science, Vol. 1112.*

[54] Burges, C. (1997). Building locally invariant kernels. In *Proceedings of the 1997 NIPS Workshop on Support Vector Machines.*

[55] Osuna, E. and Girosi, F. (1998). Reducing the run-time complexity of support vector machines. In *Proceedings of International Conference on Pattern Recognition.*

[56] Stitson, M. (1999). Design, Implementation and Applications of the Support Vector Method Learning Algorithm. Royal Holloway University of London, PhD Thesis.

[57] Saunders, C., Gammerman, A., Vovk, V. (1998). Ridge regression learning algorithm in dual variables. In *ICML 1998. Proceedings of the 15th International Conference on Machine Learning, pages 515-521. Morgan Kaufmann.*

[58] Saunders, C., Gammerman, A., Vovk, V. (1998). Transduction with Confidence and Credibility. *Proceedings of the 16th International Joint Conference on Artificial Intelligence, pages 722-726.*

[59] Heckerman, D. (1997). Bayesian networks for data mining. *Data Mining and Knowledge Discovery, (1):79-119.*

[60] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

[61] Fogel, D., B. (2001). *Blondie 24: Playing At The Edge of AI.* Morgan Kaufmann; ISBN: 1558607838.

[62] Nouretdinov, I., Melluish, T. and Vovk, V. (2001). Ridge Regression confidence machine. In *Proc. 18th International Conf. on Machine Learning.*

[63] Papadopoulos, H., Proedrou, K., Vovk, V. and Gammerman, A. (2001). Inductive Confidence Machine for Regression. In *European Conference on Machine Learning, Lecture Notes in Artificial Intelligence,* pp.345-356, 2002.

[64] Simard, P., LeCun, Y. and Denker, J. (1993). Tangent prop - a formalism for specifying selected invariance in an adaptive network. In *Advances in Neural Information Processing Systems*, J.E. Moody, S.J. Hanson, and R. P. Lippmann, Eds., vol. 4, Morgan Kaufmann, San Mateo, CA.

[65] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R.E., Hubbard, W. and Jackel, L. J. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, vol. 1, pp. 541—551.

[66] Bottou, L. and Vapnik, V. (1992). Local Learning algorithm. *Neural Computation,* Vol. 4, no. 6, pp. 888—901.

[67] Boser, B.E., Guyon, I.M. and Vapnik, V.N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, D. Haussler, Ed., pp. 144--152, ACM Press, Pittsburg, PA.

[68] Scholkopf, B., Burges, C.J.C. and Vapnik, V. (1995). Extracting support data for a given task. In *Proceedings, First International Conference on Knowledge Discovery and Data Mining*, U.M. Fayyad and R. Uthurusamy, Eds., pp. 252--257, AAAI Press, Menlo Park, CA.

[69] Scholkopf, B. (1997). Support Vector Learning. *Ph.D. thesis*, R. Oldenbourg Verlag, Munchen., Technical University of Berlin.

[70] Scholkopf, B., Burges, C.J.C. and Vapnik, V. (1996). Incorporating invariances in support vector learning machines. In *Artificial Neural Network-ICANN'96*, C. von der malsburg, W. von Seelen, J.C. Vorbruggen, and B. Sendhoff, Eds., vol. 1112, pp. 47--52, Springer Lecture notes in Computer Science, Berlin.

[71] Bromley, J. and Sackinger, E. (1991). Neural-network and k-nearest-neighbor classifiers. Tech. Rep. 11359--910819-16TM, AT&T.

[72] Dong, J. (2001). Statistical Results of Human Performance on USPS database. *Report. CENPARMI*, Concordia University, October 2001.

[73] LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998). Gradient-based learning applied to document recognition. In *Proceedings of the IEEE, vol. 86, No. 11, pp. 2278--2324, November 1998.*

[74] LeCun, Y., Jackel, L.D., Bottou, L., Denker, J.S., Drucker, H., Guyon, I., Muller, U.A., Sackinger, E., Simard, P., and Vapnik, V.N. (1995). Comparison of learning algorithms for handwritten digit recognition. In *Proc. Int'l Conf. Artificial Neural Networks*, pp. 53--60, Paris, 1995.

[75] Burges, C.J.C. and Scholkopf, B. (1997). Improving the accuracy and speed of support vector learning machines. In *Advances in Neural Information Processing*

Systems, M. Mozer, M. Jordan, and T. Petsche, Eds., vol. 9, pp. 375--381, MIT Press, Cambridge, MA, 1997.

[76] Scholkopf, B., Simard, P., and Vapnik, V. (1998). Prior knowledge in support vector kernels. In *Advances in Neural Information Processing Systems*, M. Jordan, M. Kearns, and S. Solla, Eds., vol. 10, pp. 640--646, MIT Press, Cambridge, MA, 1998.

[77] Markowetz, F., Edler, L., Vingron, M. (2001). Support Vector Machines for Protein Fold Class Prediction. In *Mount Holyoke Conference: Statistics in Molecular Biology and Genetics.*

[78] Surkov, D., Chervonenkis, A., Gammerman, A. (2001). Kernel for protein sequences classification. Royal Holloway University of London Technical Report CLRC-TR-01-08.

[79] Androutsopoulos, I., Koutsias, J., Chandrinos, K.V., Paliouras, G. and Spyropoulos, C.D. (2000). An Evaluation of Naive Bayesian Anti-Spam Filtering. *Proceedings of the workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning, Barcelona, Spain, pp. 9-17, 2000.*

[80] Burnham, K. P., Anderson, D. R. (1998). *Model selection and inference. A practical information-theoretic approach.* ISBN 0-387-98504-2 Springer-Verlag, New York.

[81] Chapelle, O., Vapnik, V., Bengio, Y. (2001). Model selection for small sample regression. AT&T Research Labs Technical Report.

[82] Grassmann, J., Reczko, M., Suhai, S. and Edler, L. (1999). Protein Fold Class Prediction – New Methods of Statistical Classification. In *Proceedings: Seventh International Conference on Intelligent Systems for Molecular Bialogy (ISMB), pages 106-112.*

[83] Gammerman, A., Vapnik, V. and Vovk, V. (1998). Learning by transduction. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence,* 1998, pages 148-156, San Francisco, CA, Morgan Kaufmann.

[84] Turing, A. (1950). Computing machinery and intelligence. *Mind (Vol. LIX, Oct. 1950).*