Aalto University
School of Science and Technology
Faculty of Information and Natural Sciences
Degree Programme of Computer Science and Engineering

Antti Rauhala

Pair Expression:
Re-expression Based Machine Learning Technique

Master's Thesis

Helsinki, April 30, 2010

Supervisor: Professor Harri Lähdesmäki

Instructor: Matti Haavikko, M.Sc.

Aalto University
School of Science
and Technology

Aalto University School of Science and Technology Faculty of Information and Natural Sciences Degree programme of Computer Science and Engineering	ABSTRACT OF THE MASTER'S THESIS

Author: Antti Rauhala

Title: Pair Expression: Re-expression Driven Machine Learning Technique

Number of pages: 110	Date: April 30, 2010	Language: English
Professorship: Information and Computer Science		Code: T-61

Supervisor: Harri Lähdesmäki

Instructor(s): Matti Haavikko

Abstract:

This paper introduces a new technique for machine learning that is based on a brand new approach. Pair-expression attempts to find simpler and more dense expression for data so, that unknown variables becomes easier to predict and data is easier to compress. So in fact the technique is re-expression technique, but it has been designed for and it can be succesfully used for machine learning. Combined with naive bayesian predicting, it eliminates efficiently the bias resulting from naive assumption, and it can lead to even dramatic reduction in the error depending of the sample. As a result, naive bayesian no more functioned as a mere classifier, but as a predicter which provided (approximately) bialess probability estimates for unknown variables.

So as a difference to traditional machine learners, the technique attempst to optimize the information expression. In this problem setting, the aim is to find an optimal language L, that can be used for re-expressing the original data in form, where the redundancy between variables has been minimized and as a consequence the regularities present in the data are captured in the language's structure. During language construction, surprisingly common variable pairs are re-expressed by introducing new expression variables. The redundancy introduced by the new expresison variables is eliminated with a special technique called 'variable reduction'.

Technique's properties were examined with a thought play, where the initial independence assumption is equated with classical analysis (where problems are divided into subproblems) and re-expression is equated with classical synthesis (where solution are formed from subsolutions). In the method the 'synthesis' is targeted against regular subsystems, which is considered to reduce approximation error with ideally small price of complexity and training error.

Technique's performance was evaluated by teaching it seven samples from 1995 Statlog study and by comparing results against 22 machine learners' public results. In testing pair-expression produced superior results for data, which consisted mostly of discrete variables, and it was first or second in four of seven samples, but with purely numeric samples the results were mediocre. The results were interpreted as extremely good and promising, especially because there is still lot to develop in actual algorithms. Especially the predicted variables could not be included for re-expression, because problems with prediction algorithm, which limited the learning ability. Based on experiences of this study, expression driven learning appears as very fruitful grounds for future research.

Keywords: Re-expresssion, machine learning, compression, re-expression driven learning, language learning, pair expression, statistical learning, naive Bayesian, knowledge representation

Aalto-yliopisto
Teknillinen korkeakoulu

Aalto-yliopisto Teknillinen korkeakoulu Informaatio- ja luonnontieteiden tiedekunta Tietotekniikan tutkinto-ohjelma/koulutusohjelma	DIPLOMITYÖN TIIVISTELMÄ

Tekijä: Antti Rauhala

Työn nimi: Pari-ilmaisu: Uudelleenilmaisuun perustuva koneoppimistekniikka

Sivumäärä: 110	Päiväys: 30.4.2010	Julkaisukieli: Englanti

Professuuri: Tietojenkäsittelyekniikka	Professuurikoodi: T-61

Työn valvoja: Harri Lähdesmäki

Työn ohjaaja(t): Matti Haavikko

Tiivistelmä:

Tämä paperi esittelee uuden tekniikan koneoppimiseen, joka perustuu uudelle lähestymistavalle. Tekniikassa pyritään hakemaan yksinkertaisempaa ja tiiviimpää ilmaisua datalle siten, että tuntemattomat muuttujat on helpompi selvittää ja että data pakkaantuu pienempään tilaan. Varsinaisesti tekniikka on siis tiedon uudelleenilmaisemistekniikka, mutta se on suunniteltu ja sitä voidaan soveltaa menestyksekkäästi koneoppimiseen. Käytettynä naiivin Bayesialaisen ennustajan kanssa, se eliminoi tehokkaasti naiivista oletuksesta johtuvaa systemaattista harhaa, ja voi johtaa jopa dramaattiseen ennustusvirheen pienenemiseen riippuen otteesta. Seurauksena naiivi Bayesialainen ei toiminut niinkään luokittajana, vaan ennustajana, joka tarjosi (likimain) harhattomia todennäköisyysestimaateja tuntemattomille muuttujille.

Erona perinteisiin koneoppimismenetelmiin tekniikalla pyritään siis optimoimaan tiedon ilmaisua. Ongelman asettelussa pyritään löytämään optimaalinen kieli L, jolla alkuperäisen datan voidaan ilmaista uudelleen muodossa, jossa esityksen muuttujien välinen redundanssi on minimoitu ja seurauksena tieto datan säännönmukaisuuksista tallentuu kielen rakenteeseen. Kieltä muodostettaessa alkuperäiseen tiedon esitykseen lisätään yksitellen uusia pari-ilmaisumuuttujia ilmaisemaan yllättävän yleisiä tilapareja. Ilmaisujen lisäyksien jälkeen käytetään muuttujien vähennystekniikkaa, jolla eliminoidaan ilmaisumuuttujan järjestelmään tuoma redundanssi.

Tekniikan ominaisuuksia tarkasteltiin ajatusleikillä, missä pohjaoletuksena tehty riippumattomuusoletus rinnastettiin klassiseen analyysiin (jossa ongelma jaetaan osaongelmiin) ja uudelleenilmaisu rinnastetaan klassiseen synteesiin (jossa ratkaisu kootaan osaratkaisuista). Menetelmässä 'synteesi' on kohdistettu säännöllisiin osajärjestelmiin, minkä katsottiin vähentävän approksimointivirhettä jopa ideaalisen pienellä monimutkaisuuden ja opetusvirheen hinnalla.

Tekniikan suorituskykyä arvioitiin opettamalla sille seitsemän otetta 1995 Statlog tutkimuksesta ja vertaamalla tuloksia 22 koneoppijan julkisia tuloksia vastaan. Testauksessa pari-ilmaisu tuotti ylivertaisia tuloksia datalle, joka koostui pääasiassa diskreeteistä muuttujista ollen ensimmäinen tai toinen neljälle otteelle, mutta puhtaasti numeerisilla otteilla tulokset olivat keskinkertaisia. Tulokset tulkittiin erittäin hyviksi ja lupaaviksi, erityisesti koska itse algoritmeissä on vielä paljon kehitettävää. Erityisesti on mainittava, että ennustattavia arvoja ei otettu uudelleenilmaisuun mukaan johtuen ongelmista ennustamisalgoritmin kanssa, mikä rajoitti oppimiskykyä. Perustuen tutkimuksessa saatuun kokemukseen, uudelleen-ilmaisuun pohjautuva oppiminen vaikuttaa erittäin lupaavalta alueelta lisätutkimukselle.

Asiasanat: Uudelleenilmaisu, koneoppiminen, pakkaus, uudelleenilmaisuun perustuva oppiminen, pari-ilmaisu, naiivi bayesialainen, kielen oppiminen, tilastollinen oppiminen, tiedon esittäminen

Table of Contents

1 Introduction....6
2 Previous Study....12
2.1 Basics....13
2.2 Artificial Intelligence....14
2.3 Machine Learning....16
2.4 Formal Languages....18
2.5 Compression....20
2.6 Knowledge Representation....22
2.7 Considerations....24
3 Conceptual and Theoretical View on the Problem....27
3.1 Problem Setting....27
3.2 Notations....28
3.3 System Analysis And Synthesis....29
3.3.1 Justification for System Analysis....29
3.3.2 Limitations of System Analysis....30
3.3.3 System Complexity and the Curse of Dimensionality....32
3.3.4 Justification for System Synthesis....34
3.3.5 Re-Expression as a Method Of Synthesis....36
3.4 Pair Expression Based Statistical Learning Algorithm....38
3.4.1 Pair Expressions....39
3.4.1.1 The Theoretic Framework....39
3.4.1.2 Re-Expression And Interpretation Algorithms....43
3.4.1.3 Pair Expressions by Example....45
3.4.1.4 Pair Expression Notation....47
3.4.1.5 Pair Expressions And Regularities....48
3.4.2 Language Formation....49
3.4.2.1 Concepts And Design....49
3.4.2.2 Pair Expression Inclusion Logic....52
3.4.2.3 Overfitting And The Pair Expression Inclusion Logic....55
3.4.3 Predicting....58
3.4.3.1 Naive Bayesian....58
3.4.3.2 Naive Bayesian And Pair Expression....60
3.4.3.3 Limitations To Naivety....62
3.4.4 Pair Expression Mechanism Properties....68
4 Evaluation....70
4.1 Pair Expression Configuration....70
4.2 StatLog Comparison Samples....73
4.3.1 German Credit....74
4.3.2 Heart....79
4.3.3 Australian Credit....80
4.3.4 DNA....81

4.3.5 Vehicle....82
4.3.6 Image Segmentation....85
4.3.7 Shuttle....86
4.4 Secondary Properties And Performance....87
5 Conclusions....90
5.1 Learning and Predicting....90
5.2 Supporting Human Understanding....93
5.3 Performance....95
5.4 Compression and System Entropy....96
5.5 Pair Expression Configuration....97
5.6 Future Sights....99
5.7 Summary....99
6 References....102
APPENDIX....106
A. Promising Techniques Around Pair Expression....106
A.1 Against Overfitting....106
A.2 Symmetry and Re-expressing Predicted Variables....107
A.3 Interest-Driven Lossy Compression....109

1 Introduction

What is intelligence? This question has ever tempted the curiosity of human mind. What a wonder this gift of mind truly is; how unique and how powerful? What has the history of science proven for us, but that there seems to be no limit for human understanding. The wonders from the world, from the stars to the lightning to the mystery of life, even the greatest of the ancient secrets and mysteries have not escaped human understanding, but instead become trivia; everyday knowledge; food for the minds of our children. The ever deepening and improving view on the world reveals not only the incredible and enormous complexity that is in-build in this reality, but also the incredible ability of our limited minds to comprehend and understand something, that in its essence seems infinitely complex. The essence of this world is captured in words, shysapes, memories and thoughts and in our minds it is understandable and it is predictable; and the way how we know the world guides ours hands and decisions in our daily lives. It is like a city in a bottle, the infinite complexity of the world trapped in the small space behind our eyes. Yet we manage in our everyday lives. Yet the great wonders of the world appear understandable for us, except for the mystery that we encounter when taking the very first steps of our lives; either on the surface of the water or on a glass of a mirror; or when we turn inside to wonder the very essence of ourselves.

Of course this essence of our minds that is intelligence is to still remain a mystery. This even when the Greek devised formal tool of formal logic to express understanding and to derive and validate conclusion; a tool that become almost like a living mind on the paper; a tool so powerful that it became almost a synonym for reason. This even, when advanced mathematical theory was introduced with even more powerful logic for reasoning in the form of statistics and probabilities. This even, when the computers emerged and so did the many solutions of artificial intelligence, decision making and reasoning. This is even, when a wide scientific

front emerged, each section carefully specializing on different fields like brain study, problem solving, machine learning, knowledge representation, planning/decision making, learning, language processing, motion/manipulation, perception, social behavior, creativity and general intelligence. Centuries have passed and so much has been sacrificed and given, yet the problem remains.

This paper is another small attempt in the wide field of artificial intelligence, and it concerns topics around machine learning and knowledge representation. Machine learning concerns the ability to learn from observations and to make predictions of unknown variables based on patterns recognized in the data. Knowledge representation concerns expression of knowledge; of many things, let them be variable states or patterns or many things and on how to use this information to derive further knowledge. Now, one may question, why the topics of *both* machine learning and knowledge representation are present in this paper? After all, these are often viewed as different fields of study; machine learning concerning patterns, while knowledge representation concerns expression of knowledge. Shouldn't they be separate? One may even say, that the only connecting link is on how knowledge representation tries to express the patterns machine learner tries to reveal.

Now, other could argue, that these topics are - in fact – strongly interconnected and even inseparable. There is an idea, that optimal or even meaningful expression of knowledge is heavily dependent of the expressed data and the patterns present in this data. Just consider the world we live in. In its essence, in microscopic level, it reveals its complexity, where even the smallest shapes consist of billions and billions of particles like a sea of small wheels ever living and turning. Still, on whatever level we view the world, let it be through telescope, naked eye or microscope, simple shapes and patterns form before our eyes and the world reveals itself as understandable. Galaxies, planets, lakes, trees, microscopic beings and molecules; our language as well as our thinking reflects the shapes and patterns of the reality and this great regularity of the world - of how it repeats the same rules and shapes of the same behavior - is the sole reason why we can comprehend the world and why we

can express knowledge and form memories. We cannot remember the ocean of restless pixels of our vision, but instead even the simplest details in our minds are in their essence patterns like shapes in our eyes that have continuity in time, in space and in reality. At this singularity of infinite yet super regular complexity, where even the smallest details are patterns of millions, the representation of knowledge has to be based on the observed regularities and the problems of knowledge representation and pattern learning becomes inseparable and one. Together they become the problem of approximation, the problem of modeling, the problem of dealing with complexity that is the grand problem of understanding, that how a system with million wheels can be reduced to a comprehensible and predictable form.

This new paraphrasing of the problem does not only combine pattern recognition, prediction and knowledge representation problems, but also the problem of compression. This should not be considered as a surprise, because all compression is based on re-expression of knowledge, and similarly all compression is based on developing better estimates for probabilities for pieces of knowledge, so that better codewords can be developed. Typically the only way to develop better estimates for probabilities is by recognizing patterns in the data and through this mean develop an expression or even a language for more efficient and compact expression. Fundamentally the compression problem is a probability prediction and a language problem, equivalent with machine learning and knowledge representation problems. The similarity can be demonstrated by comparing compression of the photographs with the way we understand the reality. Both of these are based on the heavy regularities present in the photographs and in the world. Good algorithm for photographs may provide compression of ten or hundred fold. The way we capture the complexity of the world inside our small heads seems to provide compression rate that seems in practice infinite.

At this point, let's get back to the artificial intelligence problem for a moment. We have paraphrased a number of problems to a single grand problem of understanding, but what is the relationship of this problem to the artificial intelligence field? Let's

consider an agent acting in an environment resembling this reality. Ultimately the agent problem is a decision making problem; the agent holds purpose, which it seeks to advance, it receives information of the surrounding world and is entitled to actions that hold consequences on the environment. Greatly, the decision making problem is about understanding the consequences of the actions and understanding how these consequences further or undermine the raised purpose. In here, the decision making relies very heavily on the understanding and the knowledge extracted from the surrounding environment. Being able to predict the consequences of the actions is critical and to do so: knowledge of both details and patterns present in environment need to be utilized. Knowledge of patterns is needed for heuristics and to what we call common sense. Knowledge of details are needed for tasks like path finding, for familiarity with people, things and places and for most everyday activities. Often also the boundaries between details and patterns is fuzzy; one man's detail being another's pattern and vice versa; with even knowledge of details being impossible to form without utilizing the presence of regularities. Overall, it seems rather clear that decision making agent is very difficult to support with a machine learner or a database alone, but instead it would be best supported by further integrated mechanism that is the regularity based modeling; capable of both knowledge storing and prediction work; capable of dealing with all sorts of thing; big and small. In this context, this machinery's purpose would be no higher, but to provide the necessary understanding of the world for the artificial intelligence for it being able to make informed decisions.

Now, the problem of understanding can be approached through the problem of a language or expression. After all, all knowledge has to be expressed in some form; for men with words or letters, for apes with screams, for computers with bits. To make the expression more powerful one has to give it structure and so a language emerges. To determine, that how the bits express the modeled complexity, one has to control not only syntax, but also the semantic meaning of expression, and the problem transforms into a form of knowledge representation. We seek to describe knowledge of some entity that we can call a system. When put to words, the problem of patterns

and regularities becomes the problem of “What is the optimal language L for describing the system S?” (or even “What is the optimal language L for describing the system S to support purpose P?” that is more relevant for AI problem). The solutions for this problems serves both as a solution for knowledge representation problem and machine learning problem, because the language definition itself will reflect the patterns and regularities present in the system enabling learning and predicting, while the combination of the language definition and the re-expressed information captures the understanding and knowledge of the system. All this can be supported with statistical knowledge of the relationships between the concepts present in the language.

But for this paper, the attempt is not only to ask the grand question, but also to provide an answer for it; and this answer comes in the form of pair expression, which is in roots a tool for creating a language definition that can be used to re-express regular information in a heavily compressed form. The great trickery consist of using the language definition and statistical information to demonstrate understanding of the system in the way of making predictions for unknown variables and of acting as a powerful and well-performing solution for the machine learning problem.

So the research question for this paper is the question of understanding that is how to provide such an artificial construct, that it can captures the essence of examined system so that it can be used to re-express the information in a compressed form and it can be used for making predictions of unknown variables. It acts as an answer – even if as a limited one - for a number of problems that are the problems of learning, predicting, knowledge expression and compressing, and it stands in the middle ground for these fields of study. It learns the regularities and patterns present in the complexity to understand it and form a simplified expression for the knowledge. Yet compared to the human mind, it is still a baby step and rather limited in its ability of regularity based modeling. Unlike the modeling of mind or many compression techniques for photographs, it is unlike to drop complexity by magnitudes. Instead, greatly because the compression/simplification is lossless, the rates are much more

modest. Equally, it's ability to learn patterns and regularities are limited and instead similar to other machine learners, like neural networks or KNN. While it would appear to form classes or higher level concepts, this ability is also rather limited, even if it does perform very well when compared to a number of other machine learners as demonstrated in this study.

2 Previous Study

The topic of this paper concerns quite wide field of study; having an emphasis on machine learning with the algorithm's background in compression and formal language learning, while having very special relationship to knowledge representation in the sense, that the algorithm's output can be interpreted as a such. While the topic concerns quite a number of areas, it does not build heavily upon existing machine learning, compression or language learning algorithms, even when it is possible to discover similarities with such. In a sense, pair expression takes some basic concepts from formal languages, introduces these concepts into the world of machine learning, introduces some semi-new ideas of system analysis/synthesis approach, variable re-expression/reduction mechanisms and finally builds some algorithms around these concepts based on quite basic information theory, statistics and university math.

While this paper is not relaying heavily on existing learning machines (except for naive bayesian), compression techniques (except basic theory found in the Shannon's famous paper) or formal language algorithms, the philosophy and quite lot thought work is based on various ideas present in the field of machine learning and computing science as a whole. Also comparison with similar ideas and techniques gives interesting context and a point of comparison for the introduced technique.

Before moving forward, the reader should know that there exist also implementation of pair expression algorithm for dealing with symbol sequences like plain text. This algorithm produced simplistic grammar, yet it was mainly designed fo compression and it provided similar if slightly weaker results for plain text as the very popular zip compression. To perform the compression, it analyzed the text to create a formal grammar that was by its expressive power weaker than regular expression. This

simple grammar learner / compression algorithm inspired the pair expressiont version introduced in this paper, that functions to solve machine learning problems instead solving patterns behind symbol sequences. This unpublished algorithm is utilized in following chapters to help comparing pair expression with various formal languages and their learners.

2.1 Basics

This work relies heavily on the basic foundation of the information theory. The most important tools used when approaching problems are the basic ideas of entropy and information revealed in Shannon's famous paper “A Mathematical Theory for Communication”. While the basic approach is based on information theory, the information theory is again based on the wide foundation of probabilities and statistics, which are widely employed in this paper for both describing the binary variable systems and its properties and making predictions based on Bayes inference. Concept of 'naive assumption', the assumption of variable independence, is omitted from the context of the naive Bayes classifier, which is laid as an underlying assumption and corrected as dependencies are recognized. A major theme on this paper are considerations around Curse of Dimensionality coined by Richard Bellman, which emphasizes the problem of exponential growth in (state) complexity when amount of variables grows linearly. The complexity of problems is traditionally fought by a form of analysis, that is a process of dividing a complex problem into a number of smaller and less complex problems; an idea strongly present in Bellman's dynamic programming and an idea widely applied also in this paper. [1][2][8][9][10][11]

2.2 Artificial Intelligence

The artificial intelligence is the study of intelligent agents. It has rather a long history, the topic being subject to philosophical debate from rather ancient times. It can be seen predeceased by the fields of formal reasoning and automation, before the era of computing and before emerging as a science of its own during 60s and 70s. It has been a subject of interests for computer scientists beginning from Turing from the very dawn of computing. It has seen its eras of progress and optimism, as well as its dark winters, which typically raised from recognition of incredibly challenging and seemingly unsolvable problems. [15][16][24]

The first golden era of artificial intelligence witnessed the emerging of the basic AI philosophical ground, symbolic reasoning, searching as reasoning as well as experiments with logic, simple neurons (perceptrons) and languages. This raise was brought to end by the limitations in the computing power and problems with exponentially growing complexity that is typical for artificial intelligence problems. Another boom followed during 80s with the raise of expert systems, knowledge revolution and back-propagation in neural networks, but exaggerated expectation were followed by disappointments and cuts in funding. Still meanwhile, quite a number of technical solutions brought by AI research have shown to be useful in a number of application fields, like the use of machine learners for expert systems, data mining, speech recognition and banking.

The field of artificial intelligence casts its shadow over most of topics covered in this paper. Artificial intelligence research has acted as a womb for fields like machine learning, knowledge representation and it is closely related to the sciences of formal languages and compression. Acting merely as an umbrella above the fields of more concrete and specialized sciences it provides interesting philosophical framework and proposes many questions that are relevant also for the topic of this paper. The early framework for the problem of artificial intelligence is presented in McCartney's 1969

paper “Some Philosophical Problems From the Standpoint of Artificial Intelligence”. This paper divides the problem of artificial intelligence into two parts, which are the heuristic part and the epistemological part, where the epistemological part captures the information of the world and the heuristic part uses the information to make necessary decisions to solve the problem setting. The questions raised in McCarthy's paper for epistemological part concern the kind of representation of the world, for its physical parts and for non-physical concepts like mathematics, goals and so forth, how observations are transferred into knowledge and how the system's knowledge is expressed. [16]

The problem of the epistemological part in its entirety is the artificial intelligence problem related to the subject of this thesis. A major part of the problem relates to the link between various concepts in the language and the observations and variables and states expressing the observed information. Another part relates to that how the concepts relate to the parts or shapes of reality. Pair expression learns patterns and shapes in the observations by creating expressions for abnormally common variable state combinations, which explains both how learning is done from observations and the relationship between learned concepts and observations. Now, considering that the observations are generated based on reality through some observation mechanism and the patterns, shapes and consistencies present in the world are expected to be translated as patterns, shapes and consistencies in the observation data. If the patterns present in the world become patterns in the data, the concept structure learned on the observations should ultimately reflect the structures present in the reality. This would explain the connection between expression concepts and the physical reality. After these considerations, the remaining problem is the encoding of knowledge as a sequence of symbols, which is somewhat trivia, because the representation/learning scheme is expression/language based. While the pair expression would appear to be somewhat complete answer to the epistemological problem in the sense that it provides some kind of answer for every aspect of the problem, it is not very powerful answer and its ability for pattern recognition is limited.

2.3 Machine Learning

The field of machine learning covers wide range of different models, algorithms and methods based on very different rationales and even very different views and definitions on learning. The different models draw their influence from various sources like quite basic mathematics, differential calculus, statistics, computation theory, information theory and from fields like evolutionary biology and neurology. Just for basic classification purposes a great number of different algorithms of different kinds can be identified. There are symbolic learners/classifiers associated with data mining as well as classical algorithms (K nearest neighbor, naive Bayes, linear discriminant, logistic regression) and more modern statistical algorithms (kernel density estimates) and numerous artificial neural networks (back-propagation, radial basis function) as well as many numerous other: Kohonen self organizing map, support vector machine, decision trees, bagged/boosted decision tree forests, different genetic algorithms, Bayesian networks and so forth. [4][6][7][24]

It is needless to say, that the range of applications areas is even wider, but the various application areas can be also used to provide common metrics for evaluating the vastly different machine learning systems. The measurements of machine learning systems performance on different application areas can be used for making meaningful comparison between the numerous and vastly diverse solutions. While under the hood the comparison between learning systems may appear like comparing apples and oranges, on the application level they provide very similar or even the very same services of making predictions and revealing patterns and dependencies among different items or variables. While the many learning algorithms' performance vary depending of application and problem field, success in one application field typically predicts success on other application fields and data sets. This consistency of success when facing highly varying random problems and data sets could be

rightfully described as a generic learning ability. While it is far from an absolute metric, its approximate, that is the average ability to learn given a diverse data sets, has been used when comparing and ranking the performance of various systems in the past. Still even stricter metrics for performance can be obtained by analytical solutions that set boundaries for errors. For example Vapnik–Chervonenkis dimension is a formal technique for measuring the learning ability of statistical classification algorithms. [33]

There has been few comprehensive machine learner comparisons including StatLog: Comparison of Classification Algorithms on Large Real-World Problems in 1995 and An Empirical Comparison of Supervised Learning Algorithms in 2006. Statlog provided very interesting data on that day's non-commercial algorithm's performance, and it also attempted to find similarities and categorize algorithms based on their performance on the many samples. The results demonstrated e.g. the strength of statistical algorithms with credit standing problems, the strenght of classic machine learner's on segmentation problems and revealed presence of numerous similar patterns. What the 2006 study did show, was the strength of the aggregated 'democratic' learning machine, where the given answer to a problem is picked by a poll by a high number of weaker learning machines. In the study, this kind of 'democratic' boosted or bagged decision tree forests emerged as best performing solutions. These decision tree forests, which in the test situation could contain up to thousand individual decision trees, have been rather new development on the field. The idea was introduced in Michael Kearns unpublished manuscript of 1988 and the technique saw emergence in the 1990s. Of the more traditional systems, support vector machines, K-nearest neighbor and neural networks performed decently in the study and simple decision trees and naive Bayes classifiers became last. While the older study (StatLog) did not contain newer boosted/bagged decision trees, it did measure performance of symbolic learners and statistic regression as an addition to other traditional methods. This older study did not appear to contradict the results of the newer study. [6][7]

The statistic/probabilistic solution in this paper is mainly a statistical algorithm and is closely related to naive Bayes classifier, in the sense that it attempts to improve naive Bayesian's results. Based on this knowledge, the algorithm can be expected to perform as well or better than naive Bayesian and have results that are similar to other statistical algorithms like Bayesian networks or logistic regression. Naive Bayesian in fact is not very well performing algorithm, but still it is worth noting that the main reason for the naive Bayes classifier's poor performance is the resulting bias, when the naive assumption does not really hold, and the solution seeks to fight this bias by modifying the information representation into a form, where the expressed variables are either exclusive or statistically independent.

2.4 Formal Languages

As this paper's topic is closely related to the expression of information, formal languages are naturally interesting. The study of formal languages came to life in 1956, when Chomsky formulated a mathematical model for grammar in connection of his study for natural languages. Shortly afterwards the connection between this formulation and programming languages was recognized, and the study of formal languages gave birth for syntax oriented compiling and techniques like compiler compilation. Also the connection between formal languages and various automata was recognized, and nowadays the languages and various automata are inseparable, like regular expression and state machine and stack automata. Also algorithms and Turing machine can be considered to be a pair of similar relationship. The complexity of the language is closely related to the computational power of the machinery validating it. [19]

Formal languages and machine learning have an old yet strengtening relationship. According to the introduction of Jurafsky's and Martin's 2009 book “Speech and

Language Processing", the huge acceleration in language processing and speech recognition research since 1990 can be greatly attributed to the new emphasis on language learning that is supported by various machine learning techniques. The many versions of Markov Model has been utilized for ages to construct probabilistic state machines to approximate the structure of various languages. Nowadays support vector machines, maximum entropy techniques, multinomial logistic regressions and graphical bayesian models have become basic practice in computational linguistics. This development is supported by the emerge of various unsupervised statistical machine learning approaches. [18]

The current state of the art language learning techniques seems to be able to learn a subgroup of context free grammars. The abilities of various techniques were tested in the 2004 Omphalos Context Grammar learning competition. The winning algorithm is described in Alexander Clark's 2005 paper "Learning deterministic context free grammars: The Omphalos competition.", which remarked the competition being perceived as extremely challenging. The algorithm used mutual information to construct the grammar, but was otherwise quite different compared to pair expression. It assumed text to follow context free grammar with a limiting non-terminally separated (NTS) property, and it was interested of surprisingly common symbols surrounding specific strings instead of surprising high co-occurrence of symbols or variable states as in pair expression. While Clark's algorithm provided the best performance in the competition, it is remarked in Clark's paper that the algorithm is able to learn only a subset of context free languages. [22][23]

When considering the significance of formal languages in relationship to pair expression; two connections between these things can be recognized. One connection is the connection between formal languages and the machine learners in general. Another connection is the background of pair expression mechanism in the formal languages. In fact, the first 'version' of pair expression was designed for the purpose of generating a formal grammar and for functioning as a simplistic compression algorithm and for this purpose it did work. While the focus of the mechanism has

changed from symbol sequence to binary variable system, quite many characteristics of formal languages were carried to this new re-expression / machine learner / compression algorithm. Still as a result of this metamorphosis, the pair expression language got loaded with semantic meaning in the sense, that its expressions carried information of the examined system. Despite application domain differences, the interpretation of pair expression as a formal language is interesting and somewhat revealing. Original pair expression was able to express very limited subset of regular expression; and its automata equivalent had been rather limited non-deterministic state machine. These 'limitations' in formal languages world still don't appear to be a handicap in the machine learning world, where the pair expressions perform well. One interpretation would be that neither other machine learners hold expressiveness of e.g. context-free grammar or computational power of e.g. stack automates, and they cannot have, because the data they are working on is not ordered/organized. Having organized data (e.g. binary variables set in sequential order or organized as a matrix) would perhaps allow machine learners with greater expressiveness that could be interpreted as more powerful automates. E.g. contexfulness could provide means for recognizing shapes (shape borders form NTS context) in a color matrix and the mindplay tempts interpretations of other language learner techniques as machine learners. If the data were ordered, could e.g. Clark's algorithm be generalized as a machine learner?

2.5 Compression

As people that are familiar with information theory know, compression is based on ability to assign accurate probabilities for pieces of knowledge (to get optimal codewords), making it very closely related to predicting. Naturally, compression is also related to re-expression in the sense that compression is re-expression; with an emphasis of reducing the re-expressed/encoded information size. In a away, both

predicting and compression are based on the assumption that examined system is somewhat predictable and regular. This also helps to understand why there is a strong connection between machine learning and compression to the point that machine learning solutions can be combined with simple compression algorithms for very impressive results. [1]

While machine learning techniques can be used for compressing information, the compression field itself has developed algorithms for rather generic lossless data compressions as well as specialized algorithms for specific purposes like audio and video compression. The main data compression algorithms are based on finding optimal codewords for symbols in the data and eliminating recurring sequences of symbols or simply words. Examples of codeword assignment methods includes various Huffman encoding schemas and arithmetic coding. Lossy audio and video compression techniques typically transform the data into waveform and then eliminates components that hold only marginal effect on the outcome. Still, the very traditional and common means for compression hold only little relevance for the topic. [8][26][27]

The combinations of machine learners and compression algorithms are somewhat more interesting. Machine learner is feed with some parameters, like previous bytes or words, and it is simply used to assign probabilities for the upcoming symbol's alternatives which are then transformed to code words. In the simplest scenarios, the predictions can be based on plain statistics as it is in PAQ compression algorithm, which achieved the Hutter prize (awarded for compressing 100 MB text corpus of human knowledge) baseline in August 2006. As a curiosity, newest versions of PAQ compression uses artificial neural network to combine predictions from various statistics. [22][23]

Similarly machine learners, which internals can be used for compression as such, are very interesting. An example of such is the Kohonen Self-Organizing Map, which neuron map 'stretches' to form an approximation of the trained variable space. Symbolic FP-tree, which is used to reveal association rules and which has

background in data-mining, is similarly able to both make predictions and compress information, all thought as a difference to SOM the compression is lossless. Compression interpretations can be done for the internals of other machine learners as well. [25][26][28]

Even more interesting are the compression techniques, that assume the data to follow language structure and will compress the text based on a grammar generated from the text. An example of this is the language compression program SNPR introduced in J. Gerard Wolff's 1982 paper “Language Acquisition, Data Compression and Generalization”. SNPR's output actually resembles heavily pair expression's symbol stream version's output, except that it apparently uses simple statistical rules instead of information theory for expression forming, all though it seems otherwise more sophisticated and more powerful. The main difference is that after picking out common symbol sequences (what pair expression also does), SNPR's also seeks to find shared contexts within the sequences to construct a context-free grammar. [21]

Also methods that construct probabilistic automates can be interpreted as a language-driven compressors. E.g. algorithm that reconstructs a Markov model can be seen language driven compressor; as the results that is the probabilistic finite state machine can be interpreted as a regular expression kind of formal language. Here again there are similarities with pair expression, which constructs a grammar that is similar yet weaker than regular expression. Still, one major difference between Markov model and pair expression is that the pair expression can be generalized also for unsupervised and supervised problem settings of the machine learners.[19]

2.6 Knowledge Representation

Knowledge representation has important role for this paper, solely for the reason that the pair expression can be interpreted as a method of expressing knowledge as such;

even if its abilities in expressing knowledge and patterns are somewhat limited. The field of knowledge representation studies expression of knowledge so that reasoning can be done, which basically means either validating or deriving new knowledge from the expressed knowledge base. Davis Shrobe and Szolovits article “What Is a Knowledge Representation?” provides five different aspects for knowledge representation, of which one concerns representation as 'replacement' of real world (author's comment: or raw data), a way of formulating problems, thinking and reasoning in the world and where remaining aspects consider the practical performance aspects and the human ability to understand the expression. [29]

In a sense, the McCartney's article of AI philosophy (discussed before) concerns quite wholesomely this topic and the epistemological problem of the paper is very closely related to the problem of knowledge representation. The main difference between the problems is that epistemological problem includes also the problem of machine learning (of how to learn from observations). For KR, while classes, patterns and rules function as targets of description, the aim is not to learn them from the raw data, but rather devise means to describe known instances, classes, data and regularities. One may even say that knowledge representation attempts to describe things that machine learners attempt to learn. Still, one can recognize a mismatch in the sense that the knowledge representation languages are able to describe far more powerful concepts and entities than any of the existing machines learners are able to ever learn. This mismatch is reflected by how the research around semantic networks and knowledge systems concern problems that are from rather different world compared to the problems relevant for this paper. [16]

From the pair expression point of view, the most interesting question is the expressive power of various notations, e.g. what are the expression power differences between hierarchical and network models, classification and description models, and that what kind of information requires symbolism or recursiveness. This would provide a point of comparison for the expression and help in setting new and interesting goal for future research. According to Nebel's 2000 article “On the Compilability and

Expressive Power of Propositional Planning Formalisms", there is a wide consensus around the expressive power of various language features, but uncertainty on how to measure the expression power in a formal way. Currently still, the one commonly accepted criteria for comparison seems to be the conciseness of translating/compiling expression from one language to another, even if this compilation is not always computationally possible. For example, if some expression in language A translates into infinite series of language B expressions, one may conclude that language A is stronger at least in this one respect (author's note: this approach would indicate, that the expressive power of knowledge representation is related to its ability to compress information and therefore to its ability to express regularities). [30]

It is apparent, that there is a number of formal tools for measuring the expressive power for a knowledge representation, even if there is uncertainty and the debate continues. Such formal tools are presented not only in Nebel's article, but also in 1996 paper from Cadoli et. al. and in Gogic et. al. 1995. Still, formal study of the pair expression's expressive power would apparently be a major task and perhaps deserve a paper of its own. However, such study would likely show the expressive power of pair expression to be rather limited. Still in the paper's solution's defense, the 'standard' problem setting as well as common corpuses for machine learning set their own strong limitations on how powerful patterns can be learned. Some kind of organization (sequential or matrix) or desription of the variable relations (e.g. x_{i+1} follows x_i , y_i is modified x_i) would allow more powerful learners.[31][32]

2.7 Considerations

While it is possible to draw connections between the work on this paper and numerous studies, the author could not find anything quite like the pair expression. The most similar topic of study seemed to be the field of learning automata, that

studies reconstruction of probabilistic state machines. Still, this learning is targeted towards approximation of stochastic processes and not towards prediction of unknown variables. [20]

While there are also a number of other algorithms, that are in a way or another similar with pair expression, the significance of the pair expression does seem to be in taking rather basic concepts and techniques from the field of formal languages and applying them for the purpose of machine learning and compression. In this paper it is demonstrated that these concepts of expression are actually very useful and powerful in solving machine learning problems. This is explained by the way how language expression parse, organize and simplify the modeled complexity in order to being able to make better predictions; or to form even more powerful expressions.

Another very important aspect of introducing expression into the world of machine learning is that while semantic meaning of language elements is not easily recovered from the text alone, the semantic meaning of observed variables in machine learning problem is often known or their significance is determined by their relationship to interesting variables. This means that the formed expressions hold automatically semantic meaning, because they express relationships between meaningful variables and they resolve to the states of meaningful variables. In this sense, they reflect the understanding of patterns in the observed system instead of reflecting artificial rules in examined language. In machine learning problems, the constructed expression is in this sense more complete as it is loaded with both structure and meaning. The structure present in the language is more than a grammar, but an essential tool in modeling the examined system and making predictions. These properties combined with the fact that the language can be used to store information in compressed and more informative form, does emphasize the idea that this form of expression is more than a grammar and instead a method of knowledge representation, a method of modeling and something what captures the understanding of the system. At the same time, the similarity between pair expression and formal languages is striking and underlined by the easiness, how pair expression can be applied to symbol sequences

and text of natural language.

3 Conceptual and Theoretical View on the Problem

In this section, we describe some used notations, the problem and the basic justification for the used solution and the solution. The main target of the study is the binary variable system and generally the aim of the study is to analyze large binary variable system to recognize patterns within variables, and to utilize the knowledge of these patterns for compressing individual samples of system's state and making predictions. This chapter does not cover the design or implementation details beside what is needed to describe the problem, the theory and the abstract solution.

3.1 Problem Setting

This paper concerns the generic problem of regularity solving in a binary variable system for the purposes of

1. Predicting
2. Compressing
3. Human analysis

where predicting includes both supervised and unsupervised classification problems. Still what unites all of these purposes, is that to serve them one has to first solve the problem of regularity solving / pattern recognition. This grand problem is approached in this paper with the tools of information theory, statistics and the mechanism of re-expression.

3.2 Notations

In this paper, the object of the study is the binary variable system consisting of a number of boolean variables. The binary variable system can be expressed as $S = X_1 \circ X_2 \circ X_3 \circ \ldots \circ X_n$, where S is denotes the system and $X_1, X_2, X_3, \ldots, X_n$ are boolean variables called in this context system variables as theirs states are used for expressing each individual state of the system $s = (x_1, x_2, x_3, \ldots, x_n) \in S$ where $x_1, x_2, x_3, \ldots, x_n$ are the boolean states $x_i \in 0, 1$ of the system variables.

The focus of this paper is in the statistical properties of the system and especially on the statistical dependencies between individual system variables. Because of this focus, the information entropy $H(S)$ is perhaps most interesting property of the system, which is known to be as less or equal to the summed entropies of the system variables:

$$H(S) \leq \sum_{\forall i : i \in [1,n]} H(X_i) = -\sum_{\forall i : i \in [1,n]} (p(X_i) \log(p(X_i)) + p(\neg X_i) \log(p(\neg X)_i)) \quad (1)$$

For the purpose of this paper, we choose to call the sum of system variable entropies the naive system entropy, because it is equivalent with system entropy, when the 'naive' assumption of statistical independence holds. This name is borrowed from the the context of naive Bayesian classification, where the statistical independence assumption is called naive, because it is systematically assumed for systems, for which it doesn't really apply for the sake of reducing complexity. We mark this naive system entropy with $H_{naive}(S)$. With this notation, the system mutual information can be defined as the difference $I(S) = H_{naive}(S) - H(S)$.

3.3 System Analysis And Synthesis

Before describing the pair expression mechanism, let's consider following the fundamental problem that is the curse of dimensionality. In this paper, system analysis/synthesis approach is suggested as a solution for curse of dimensionality (in this application field) and in the end of this section the mechanism of re-expression is interpreted as a method of such approach.

3.3.1 Justification for System Analysis

In this context, the meaning for 'analysis' is the traditional one, which is the re-expression of a complex problem in the form of numerous simpler problems. The leading idea behind splitting or 'analyzing' the problem is that the simpler problems are typically easier to solve and manage. Solving the subproblems and combining the drawn conclusions through process called synthesis is hoped to solve the original complex problem. In the case, that the smaller problems after the original split are still too complex for solving, the small problems can still be further analyzed into smaller problem until sufficient simplicity is achieved and the subproblems becomes solvable.

Similar kind of idea can be utilized also to binary variable systems, where the system can be split into smaller subsystems to fight the curse of dimensionality. When the system is split once, the system of 2^n states becomes two subsystem both having $2^{n/2}$ states. It is possible to continue splitting until all formed subsystems have less than some k variables and less than 2^k states. Effectively this makes possible to solve the possible dependencies within each subsystem with traditional state probability table of size 2^k. After the subsystems states have been traced and possible dependencies have been detected and expressed, these subsystem patterns

can be combined to provide a wider set of patterns applying to the entire system.

In this chapter we provide a justification for doing this sort of system analysis based on the definition of the entropy. Now, let's consider a system $S = X_1 \circ X_2 \circ X_3 \circ \ldots \circ X_n$. If we choose to express part of the variables in the system in the form of subsystem $S_1 = X_i \circ X_{i+1} \circ \ldots \circ X_j$, system can be re-expressed in the form of $S = X_1 \circ X_2 \circ \ldots \circ S_1 \circ \ldots \circ X_n$. Now we can se the upper boundary for the system so that:

$$H(S) \leq H(X_1) + H(X_2) + \ldots + H(S_1) + \ldots + H(X_n) \quad (2)$$

If dependencies were recognized revealing subsystem mutual information $I(S_1) = H_{naive}(S_1) - H(S_1)$ we can recognize the upper bound of mutual information for the system to decrease similarly.

$$H(S) \leq H_{naive}(S) - I(S_1) \quad (3)$$

For a number of non-overlapping subsystems $S_1, S_2, \ldots, S_n$ the equation becomes following:

$$H(S) \leq H_{naive}(S) - I(S_1) - I(S_2) - \ldots - I(S_n) \quad (4)$$

It means, that reduced entropy in the subsystem interprets as reduction of entropy in the entire system. In other words, the patterns present in subsystem can be applied to the entire system. This somewhat obvious result gives a mathematical justification for trying to solve the system regularity solving problem locally in the level of subsystems containing only a handful of variables. This approach avoids very effectively the curse of dimensionality by granting means for analyzing the system at any 'dimensionality' desired.

3.3.2 Limitations of System Analysis

When performing system analysis the size of the observed subsystems can be rightfully referred as 'variable window' or 'entropy window' as it determines that how many variables are 'seen' at any time, when looking for patterns. Now, throught big windows one can typically detect 'bigger' patterns than when using smaller windows and not all patterns can be recognized with window of any size. While the usage of e.g. two variable window may expose many of the system regularities, complex systems may contain patterns, which are impossible to spot with small windows, similarly as it is difficult to recognize figures in the pictures, if only small part of the picture is revealed at a time. An example of such patterns is the relation, where a variable is the result of XOR:ing two other variables. If all variable probabilities equal 0.5, the window must be at least 3 variable wide to be able to recognize the mutual information/dependency between the variables.

The reason, why the window could also be referred as entropy window is that the entropy summed of all variables fitting the window seems to be related to the entropy of the variables forming the patterns. Even if there are patterns, which contain a great number of the variables, if naive/real entropy of the system is low, traces of the pattern or even the entire pattern is possible to detect with much smaller entropy windows. For example consider a system of n random variables and one 'oddness' variable, which is true only if odd amount of other variables are true. If this system's variables' probabilities are 0.5, n+1-sized window is required to spot the mutual information. If the same variables probabilities were e.g. 0.01 a degree of mutual information around the oddness variable could be detected with variable windows much smaller than n. This because, if non-oddness variable X is true, it is very unlikely that any other non-oddness variable would be true at the same time, and the relationship between X and oddness variable Y approximates implication X->Y that is easy to detect with 2 sized window.

This same phenomenon can be described in a more formal sense. Let's have a sequence of n variables $X_1, X_2, X_3, \ldots, X_n$ and measurement for mutual information for some subset of size w of these variables $I(X_{i_1}; X_{i_2}; \ldots; X_{i_w})$;

where this subsystem acts in fact as w-sized window. Now, let's consider few meters for describing, that how big windows are needed to extract certain amount of information from the system. One measure is the maximum amount of information that can be extracted with a single w-size window:

$$i_w = max_{j_1, j_2, \ldots, j_w \in [1,n]} I(X_{j_1}; X_{j_2}; \ldots; X_{j_w}) \quad (6)$$

This value can be used to limit the maximum mutual information i_w^k that can be extracted with an amount k of w variable sized windows, where this measurement is limited by $i_w \leq i_w^k \leq k\, i_w$ and ultimately by the system mutual information $i_w^k \leq i$. The actual value of i_w^k still depends of the properties of the system. What is noticeable still, that if the properties of the n-variable system allow i_w^k to approximate system entropy i so that $i_w^k + \epsilon = i$ with moderately growing window size $w \ll n$, and with moderately growing window amount $k \ll e^n$ the analysis approach can be applied to the system very effectively.

3.3.3 System Complexity and the Curse of Dimensionality

In the previous chapter it was underlined that the limitations of the system analysis approach are determined by the system's own nature and properties. Naturally the success of the analysis approach in general is determined by that what kind of systems are most often encountered. If the examined systems often resemble the example systems with XOR or oddness relationships, the approach is not very powerful, as the window needs to reach over the entire system.

On the other hand the remarkable property of the XOR and oddness relationship is that the system's mutual information is limited to 1. This reveals a pattern, where low relative mutual informations may be difficult to spot, but for such systems, the mutual information reward of solving the system remains very limited. In other words it is

difficult to spot regularities in systems, which are not very regular. Similarly systems, which are heavily regular are much easier to approach with system analysis and fortunately it often makes more sense to search regularities in systems, which are - in fact – regular, predicting that on average the examined systems are also so.

But what kind of systems can be encountered in the first place? The system's natural 'vulnerability' for analysis based techniques is revealed by the number of states that need to be traced to reveal a degree of system's mutual information. As demonstrated in the previous chapter, this traced state count and the system state count (that behaves exponentially) are different things, and depending of the system properties the ideal traced state count's growth may behave e.g. linearly.

Now as a mind play, let's consider a physical systems under a limited period of time t. Now, the amount of space affected by single disruptions is limited by kt^3, where k may be the speed of light, sound or some other constant. If we concentrate on otherwise static system, except for point-like random disruptions and their consequences within time period t, it means that for each system variable X it is possible to limit a subsystem S having all variables, which disruptions will hold effect on variable X. This again suggests that under these conditions it is always possible to find a window size w so that the entire system's mutual information can be determined through w sized windows and that the system is not in reality touched by the curse of dimensionality. While the state complexity still raises exponentially, the price of solving the system does not do so, but instead the (ideal) price raises linearly, even if this ideal price may not reflect the real price, because the limited set of (sub)system states that needs to be traced is not known.

For systems having the previously described property, there always exists window size w and factor a, so that the mutual entropy of n-variable system can be determined with k entropy windows of size w, where $k \leq an$. The amount of traced states is therefore limited from up by $ae^w n$, where ae^w is constant. Because the price of solving such system grows linearly, such systems can be called linearly complex systems and the system property can be called linear complexity.

For systems, which are touched by the curse of dimensionality in the sense that the amount of states that need to be traced grows exponentially, is given name exponential complexity systems. Similarly, if the amount of tracked states grows logarithmically, the system is called logarithmic complexity system. We pick the notion for expressing system complexity from the computer science world with the notion for exponential complexity systems being $O(S)=e^n$, for linear complexity systems $O(S)=n$, for constant $O(S)=1$ and so forth. Still, while we may set these kinds of ideal properties for systems, many systems are likely hard to fit into such categories and for a system e.g. only half of system mutual information may be possible to solve for a linear price, while the remaining information may be 'cursed by dimensionality'.

While this examination may not appear most concrete or useful as such, the hidden promise of this chapter is that if not most, then great amount of the systems encountered in the field of engineering are not in reality affected by the curse of dimensionality at all. Instead systems exist, which internal patterns and regularities may be expressed in linear or even logarithmic size (when compared to system variable count) and found out with a good performance in a (somewhat) limited time.

3.3.4 Justification for System Synthesis

In previous chapter the form of analysis called windowing was described as a way of fighting away exponential complexity. Now the problem with the constant size windowing is that only so much of mutual information can be detected with a window of certain size. The problem in growing the window size is the exponential cost or in other word the curse of dimensionality. In this chapter we introduce a form of synthesis that utilizes recognized mutual information as a mean of increasing window size *without* exponential cost. This method works in a situation, where the

system in reality is not exponentially complex, but instead contains internal regularities that can be recognized with windows of limited size. Following this idea, system analysis and system synthesis can be combined to fight even more effectively the curse of dimensionality without arbitrary limitations on that how big patterns can be recognized, all thought there are no strong guarantees that analysis/synthesis approach can fully solve patterns bigger than the starting window size.

In the context of the paper, similarly as the analysis is closely related to traditional idea of analysis, where a complex problem is split into less complex subproblems, the synthesis is strongly connected to the classic idea of combining subproblem answers to form a combined answer to the original complex problem.

Let's consider a subsystem $S_i \subset S$ having high internal mutual information $I(S_i)$. What needs to be recognized for systems with reduced entropy that the cost of windowing operations is equally reduced. To be exact the complexity of windowing $S_i \circ S_j$ is in the magnitude of:

$$H(S_i)+H(S_j)=(H_{naive}(S_i)-I(S_i))+(H_{naive}(S_j)-I(S_j)) \quad (7)$$

This is based on the simple assumption, that while the amount of traced states may not be lowered, the frequency information that needs to be stored for each state is all together lowered. For example in cases, where some of the states in either S_i or S_j are missing, naturally no information need to be stored for any state combination $s_i \circ s_j$ containing the missing state. This effectively reduces the amount of information present in the state trace and is helpful as such. Simplification can be brought even further and knowledge of subsystems internal states can be used to ignore state combinations that have marginal probabilities or lowered probabilities and concentrate only on surprisingly common state patterns.

For example let's consider a situation with hundred variables and window size 2. If we recognize two variables x_i and x_k to be equivalent, it is completely free (in the sense of complexity) to replace all 2 sized windows referring either x_i or x_k

with 3 sized window containing both of these variables. In here the '3-sized' window need in fact to trace only 4 different states (instead of 8) and it has only the price of 2 sized window. If all other variables were windowed with the equivalent variables, we can replace 197 2-sized windows with 98 3-sized windows thus gaining the benefit of more accurate knowledge of the probabilities. This is not only free, but it actually reduces complexity by removing frequency trace for almost 400 states (the state combinations between e.g. x_k and the other 99 variables).

It is also worth noting, that this window growing mechanism helps with the problem concerning the window size and the sample count. When having sized w window and sample count n, average system state occurs in the sample $n/2^w$ times. If this frequency is too low, relative error rates for state probability estimates become elevated and begin to drive error into the results. The presence of heavy regularities and the uneven distribution of state probabilities is a good thing considering the entropy measurement error, because it applies that the more probable and informative the state is, the smaller its relative error and the greater its impact in determining the total error rate. If the state probability distribution is biased enough, the surprisingly probable yet low-error states are common enough to dominate the entropy error rate and drive the error down. So, the more regular the system is, the smaller the error and this pattern of reduced error for regular (low entropy) system also applies to the errors of statistical prediction and compression.

The observations mean, that the curse of dimensionality should be possible to fight effectively by first performing analysis with small window size and after that by growing the window size above subsystems, which are regular, because the price of complexity (state frequence trace) and error associated with window size is lowered. The benefit depends of the system's nature, but it can be dramatic.

3.3.5 Re-Expression as a Method Of Synthesis

In the previous chapters, it was recognized that for certain patterns only a degree or no mutual information can be extracted given w sized variable window. As an alternative to increasing the window size there exist a mechanism of variable re-expression. In more specific, the mentioned limitation can be avoided by re-expressing a number of existing variables through a greater number of lower entropy variables.

For example 3 variable XOR system, where C = A XOR B, can be recognized with 2-variable window if A and B are re-expressed with variables $E_1 = A \wedge B$, $E_2 = \neg A \wedge B$, $E_3 = A \wedge \neg B$ and $E_4 = \neg A \wedge \neg B$. In such situation, the exclusiveness between C and E_1 and C and E_4 and implication from E_2 and E_3 to C are easy to detect with the 2-variable window. In a sense, the re-expression did raise the effective window size to 3, and similar kind of operations can be used to grow the effective window size to any arbitrary value and therefore to detect patterns of arbitrary size. In this sense re-expression can be used to increase traced window sizes.

So, re-expresssion mechanism can be used to grow effective window size. Now, the previous chapter described a method of smart synthesis, in which the window size is grown around regular subsystems (where regualarity is revealed by elevated subsystem mutual information). This means, that if an algorithm performs re-expression (equivalent with growing the window size) around regular subsystems it is in fact performing smart synthesis. In following chapter we introduce an algorithm, which targets surprisingly state combinations for re-expression. Such algorithm fulfills very strictly the requirements of 'smart synthesis', because if a subsystem state combination is surprisingly frequent (more frequent than naive assumption would allow), the subsystem mutual information must be lowered (meaning that the system is regular).

3.4 Pair Expression Based Statistical Learning Algorithm

This chapter describes a statistical machine learning system that is based on technique called pair expression. The pair expression is an artificial variable used to express a state combination of two other variables. This expression mechanism is supported by technique called variation reduction, which 'reduces' expressed variables by making them undefined, when their state can be determined based on other variables. The algorithm uses 2 sized windows (as defined in the system analysis section) for system analysis and it implements the concept of system synthesis by using pair expressions for re-expressing the original data as lower-entropy variables.

Now, let's consider system S and naive predictor P doing naive Bayesian predictions and naive compressor C that utilizes only variable state probabilities $p(v)$ for encoding. The question is that is it possible to find translation L from system S to translated system $S' = L(S)$ so that naive predictor P error is minimized and naive compressor's C compression rate is maximized, when system S' is used instead of S. In this sense, we choose to express both compression problems and prediction problems as a re-expression problem so, that we do not seek to optimize prediction/compression algorithms (which remain as constant P and C), but the language learning algorithm Λ in order to optimize the language $L = \Lambda(S)$ and the re-expressed system $S' = L(S)$.

In the following chapters we introduce pair expression mechanism as a mean for performing this translation. Algorithm does the translation in order to minimize following property for all system variable states (v_i', v_j'):

$$p(v_i' \wedge v_j') \log \frac{p(v_i' \wedge v_j')}{p(v_i')\, p(v_j')} \quad (8)$$

Minimizing this value will effectively 1) drive the estimated maximum system entropy down thus optimizing system for naive compression and 2) drive the system variables statistically more independent (in a weighted manner) thus optimizing the system for naive predicting. While the algorithm doesn't minimize statistical dependency, but its 'weighted cousin', it still holds that the more common the state combination (v_i', v_j') is, the more statistically independent it will be. Also, the more common the combination is the more accurate the dependency estimate will be and the more significant the combination is for the total prediction error.

3.4.1 Pair Expressions

3.4.1.1 The Theoretic Framework

Pair expressions are used to express some state combination of two variables. In the technique, singular states of variables $v_i \in V_i, v_j \in V_j$ are expressed with expression E_k, which is true on condition $E_k \Leftrightarrow (V_i = v_i \wedge V_j = v_j)$. Variables V_i, V_j can be either original binary variables or pair expressions. The benefit of pair expression comes from greater granularity, when examining the statistical properties of variable combinations. In other words, it is possible to provide better compression rate and predictions, when the probabilities $p(\neg V_i, \neg V_j)$, $p(\neg V_i, V_j)$, $p(V_i, \neg V_j)$ and $p(V_i, V_j)$ are known, compared to knowing only $p(V_i)$ and $p(V_j)$. Typically greater granularity leads to decrease in the system maximum entropy thus increasing our understanding of the system.

Still, while introduction of expressions may increase granularity, it adds new variables at the same time as it drives more dependencies and redundancy into the system. Increase in the variable count may paradoxically drive the observed 'naive'

system entropy up, if the added dependencies are not taken into account. Now, if there is some known subset of variable states, it may become possible to determine the states of some otherwise 'unknown' variables based on these dependencies. Knowledge of such determination operations can be used to eliminate added redundancy from both entropy calculations and encoding.

The simplest example of the described kind of determination is the mechanism of expression. For system state s, if the state $E_k = v_i \wedge v_j$ is known to be true, the states of original variables V_i, V_j don't need to be explicitly declared, because their combined states are – of course - v_i and v_j. A more complex situation raises, when E_k is false, while V_i is known to be v_i. In this case, we can reason that V_j must be $\neg v_j$ as $v_i \wedge v_j$ is known to be false. Equally implication $(\neg E_k \wedge v_j) \rightarrow \neg v_i$ applies. Similarly, when we have expressions $E_1 = (\neg v_i \wedge \neg v_j)$, $E_2 = (\neg v_i \wedge v_j)$, $E_3 = (v_i \wedge \neg v_j)$ and $E_4 = (v_i \wedge v_j)$ it is known that all of these expressions are in fact exclusive. If any of the expressions is known to be true, it is implicitly known that all other expression are false. Same mechanism applies when have exclusive expressions $E_1 = (a \wedge b)$ and $E_2 = (\neg b \wedge c)$ for variables A, B and C.

So the problem what we encounter is that the introduction of the pair expression E to the system S also introduces redundancy to the produced system $S \circ E$. The elimination of this redundancy leads us to the concepts of explicitly and implicitly known variables. Now, let's consider a situation, where translated system is written and read in a bit format. The bits need to be written and read in some order to be able to know, that which variable's state they declare. Bit for first variable needs to be always explicitly read (and declared in data), but depending of its state, a state of a number of remaining variables may become known. For example, first bit may declare expression $E = A \wedge B$ to be true, and because the states of A and B becomes implicitly known, we no more need to explicitly declare and read states of A and B. Similarly, every explicitly read bit thereafter may reveal states of following variables, making them implicitly known. As the process is deterministic,

given some variable ordering, it is possible to unambiguously determine for each system state s the implicitly known variables (called the implicit variables for s) and explicitly declared variables (called explicit variables for s).

Now, each variable V_i in this new encoding is explicitly declared only under some condition C_i called context. When encoding this variable, its ideal codeword size is determined by its entropy, which is determined by its probability. Because its state needs to be encoded only under condition C_i , the entropy should be based on probability $p(V_i|C_i)$ instead of $p(V_i)$. For entropy calculations and encoding, we are actually only and strictly interested on conditional variable $V_i|C_i$ and we can ignore variable V_i and its states whenever C_i is false. Similarly, when predicting variable Y and knowing that variable V_i was both true and explicitly declared, it makes more sense to use condition probability $p(Y|V_i \wedge C_i)$ instead of $p(Y|V_i)$ because it utilizes more information and the prediction will be more accurate. This means that for all purposes - after introduction of pair expressions $E_{0,}E_{1,}E_{2,}\ldots,E_n$ - we are only interested of conditional variables $V_i|C_i$ and the original variables V_i don't hold much relevance. Because variables V_i are 'transformed' into form $V_i|C_i$ of reduced context, this mechanism is called *variable reduction mechanism*.

Now, we may consider the introduction of expression E_1 as a translation of existing system $S=(V_i,V_{2,}\ldots,V_n)$ into a new system of form $S^1=(V_1|C^1_{1,}V_2|C^1_{2,}\ldots,V_n|C^1_n,E_1)$, which includes both pair expression and the interesting conditional variables, while the non-interesting and non-conditional original variables $V_i,V_{2,}\ldots,V_n$ are not included as separate variables. This model of system definition means that one cannot introduce a pair expression without fundamentally changing the system and its properties. By this it is meant, that no two pair expressions are ever applied to the exactly same system. So, if we seek to introduce another pair expression E_2 , the expression will be applied to the modified system S^1 instead of the original system S . When introducing a

sequence of pair expressions $E_{1,}E_{2,}\ldots,E_m$, the expressions are applied to systems $S^0,S^1,\ldots,S^{m-1}$, which form series. If we denote the function, which converts an earlier 'generation' system S^{i-1} into later generation system S^i , with symbol E^i (note the upper index 'i'), we can express each system in the form $S^i=E^i(E^{i-1}(\ldots(E^1(S))\ldots))$ or recursively $S^i=E^i(S^{i-1})$. The sequence of pair expressions forms a pair expression language $L=E^m\circ E^{m-1}\circ\ldots\circ E^1$ that can be used for translating system $S'=L(S)$ and its reverse function can be used for interpreting the pair expressed system $S=L^{-1}(S')$.

Now, at this point we have the basic theoretical framework for the pair expressions. We have the idea, that we can introduce pair expressions of form $E=v_i\wedge v_j$ and use them to extend a system S . We have the idea, that introduction of such expression produces a system $S\circ E$ with additional redundancy between added expression and other variables. We can eliminate the redundancy by developing rules for identifying redundantly declared variable states, mark them implicit, omit them from encoding and reformulating the system to have conditional variable 'defined' only when their state is explicitly declared, so that re-expressed system is of form $S^1=(V_1|C^1_{1,}V_2|C^1_{2,}\ldots,V_n|C^1_n,E_1)$. This framework can be used for deriving translated system $S^{i+1}=E^{i+1}(S^i)$. Then there is the idea, that series of pair expressions $E^{1,}E^{2,}\ldots,E^m$ are applied to the series of the systems $S^0,S^1,\ldots,S^{m-1}$ so that final translated system is $S^i=E^i(E^{i-1}(\ldots(E^1(S))\ldots))$, where the series of pair expressions form a pair expression language $L=E^m\circ E^{m-1}\circ\ldots\circ E^1$.

While previously the ordering of the variables for encoding was not explicitly defined, it is worth noting, that for each system S^i there is exactly one variable, which we can assert to never be expressed by other variables or made implicit by $\neg E\wedge v_1\rightarrow\neg v_2$ mechanism and that is the newest pair expression E^i , which state should be read and encoded first. After reading bit holding E^i state, we can transform system S^i bits b^i to system S^{i-1} bits b^{i-1} by dropping E^i 's bit

and by inserting the bits determined by expression E^i state to b^i (guaranteeing that E^{i-1} is explicit). Now, the same conclusion of E^{i-1} encoding priority can be repeated for system S^{i-1} bit sequence b^{i-1} and all the systems and bit sequences until the original system's bit sequence is recovered. This means, that the natural encoding order is from newest expression to the oldest expression and then to the original variables.

3.4.1.2 Re-Expression And Interpretation Algorithms

While the previous chapter introduced a general theoretic framework around pair expressions, this chapter introduces a simplistic algorithm for performing re-expression and interpretation. Now, let's start from the basic mechanisms, that mark variable states to be implicitly known. These mechanism are known as variable determination rules and they are

1. Expression rule. If expression E is true, the expressed variable states are implicitly known to be whatever E expresses.

2. Expressed state exclusion rule: $\forall E = v_1 \wedge v_2 : \neg E \wedge v_1 \rightarrow \neg v_2$. If E is not true, but one variable is in state that E expresses, we know implicitly that other variable must be in negation of the state E expresses.

3. Expression exclusion rule, which states that no two expressions can express same variable at the same time. If expression expressing state of variable V is true, all other expressions expressing any state of variable V are implicitly known to be false. Also more complicated chains of expressions obey this rule. If E_2 expresses E_1 , which expresses V , while E_4 expresses E_3 expressing V , then also E_2 and E_4 are exclusive.

These rules can also be reformulated to form a framework for a simple expression

algorithm. Let's consider original system state s with variable states $x_1, x_2, \ldots, x_n$ and language L with expressions $E_1, E_2, \ldots, E_m$. The language function can be defined through expression functions so that $L = E^1 \circ E^2 \circ \ldots E^m$. In this case, the algorithm ought to be defined through expression function E^i, which translates older generation system state into a newer generation one $s^{i+1} = E^{i+1}(s^i)$. For each system state s^i and variable V_j we have both variable states and variable's explicitness/implicitness stored.

Now how the algorithm of the operation $s^{i+1} = E^{i+1}(s^i)$ functions is following:

1. Considering system state s^i and expression E_{i+1}, we declare expression E_{i+1} state to be true, if and only if the expressed variables V_i and V_j have the expressed states v_i and v_j and are explicit.

2. *If* the expression E_{i+1} is set true, the expressed variables V_i, V_j and older excluded expressions E_j, $j < i+1$ (which are known to be false) are marked implicit.

3. *If* the expression E_{i+1} is not true, but expressed variable V_j with higher priority has expressed state, the lower priority variable V_j is marked implicit as its state is known to be $\neg v_j$

After applying the first expression, second expression can be applied and re-expression can be continued until the entire language has been applied. After entire re-expression, the implicit variable states can be forgotten without permanent loss of information.

Now, the interpretation function $L^{-1} = E^{-m} \circ E^{-(m-1)} \circ \ldots \circ E^{-1}$ does the opposite. Having s^i and expression E_i, there is need to translate from s^i to s^{i-1}. To perform this translation, we examine the expression E_i state and:

1. If the state is true, we mark the expressed variables V_i, V_j to have the

expressed states v_i , v_j and the excluded expressions E_j with $j<i$ to have false state. All these variables are also marked explicit.

2. If the state is false, while the higher priority expressed variable has the expressed state, the state of the lower priority variable is known to be complement of the state expressed by E_i . It is worth recognizing, that this operation may be non-trivia, because state of higher priority variable may not be explicitly known when E_i is resolved, and should be determined via complex state determination rules or lazily at the moment when the higher priority variable's state becomes explicitly known.

3.4.1.3 Pair Expressions by Example

But how does pair expression work in practice? Let's consider a simple system S consisting of binary variables A , B and C with expressions E_1 attempting to express true states of A and B and E_2 attempting to express true states of B and C . First consider that because the way languages are defined the oldest expressions will hold priority over newer expressions. It means that in case system state $s=A\wedge B\wedge C$, it will be expression E_1 that is allowed to be true, expressing true A and true B, while E_2 is determined false, because B is already expressed by E_1 . This is convenient, because when re-expressing, we always process oldest expression first and newest expression last (remember that $S'=E^n(E^{n-1}(\ldots(E^1(S))\ldots))$).

Now, the aim is to better understand translated system $S^2=E^2(E^1(S))$, but before that let's first consider introduction of $E_1=A\wedge B$ and the resulting intermediate system $S^1=(A|C_A^1,B|C_B^1,C|C_C^1,E_1)$. The great unknowns for this system are the context variables of form C_i^1 . Because E_1 expresses A and B, following applies

$E_1 \rightarrow \neg C_A^1 \wedge \neg C_B^1$ and B's context resolves to $C_B^1 = \neg E$. Now based on state exclusion rule: whenever E_1 is false, but B is true, A is implicitly known to be false thus reducing conditional A's context to $C_A^1 = \neg E_1 \wedge \neg(B \wedge \neg E_1) = \neg B$. Knowing all this, the intermediate system resolves to $S^1 = (A|\neg B, B|\neg E_{1,} C, E_1)$.

Following table demonstrates relationships between different original system states and intermediate system states.

S	A	B	C	E_1	S^1
000	0	0	0	0	0000
001	0	0	1	0	0010
010	Implicit 0	1	0	0	100
011	Implicit 0	1	1	0	110
100	1	0	0	0	1000
101	1	0	1	0	1010
110	Implicit 1	Implicit 1	0	1	01
111	Implicit 1	Implicit 1	1	1	11

Table 1. Relationship between system S and intermediate system S^1

On the extreme right under column S^1 one can see the binary form of system S^1 states. In the binary format, the system states are encoded from newest expression to the oldest up to the original variables, and all implicitly known variable states (bits) are left out.

Now, introducing the second expression E_2 will modify the system S^1 further to form $S^2 = (A|C_A^2, B|C_B^2, C|C_C^2, E_1|C_{2,}^2 E_2)$. In here, the second expression will have form $E_2 = (B|\neg E_1) \wedge C$ with somewhat 'creative' logical syntax, which interprets as $E_2 = B \wedge \neg E_1 \wedge C = B \wedge \neg(B \wedge A) \wedge C = B \wedge C \wedge \neg A$. So what we know now is that if E_2 is true, B and C are implicitly true, A is implicitly false and E_1 is implicitly false, as E_1 and E_2 are exclusive. We can further specify the system variable contexts that are $S^2 = (A|\neg B, B|\neg E_1 \wedge \neg C, C|\neg E_{2,} E_1|\neg E_{2,} E_2)$. Similarly, we can provide a table describing how the original system states are

transformed to translated states:

S	A	B	C	E_1	E_2	S^2
000	0	0	0	0	0	00000
001	0	Implicit 0	1	0	0	0100
010	Implicit 0	1	0	0	0	1000
011	Implicit 0	Implicit 1	Implicit 1	Implicit 0	1	1
100	1	0	0	0	0	10000
101	1	Implicit 0	1	0	0	1100
110	Implicit 1	Implicit 1	0	1	0	010
111	Implicit 1	Implicit 1	1	1	0	110

Table 2. Relationship between system S and translated system S^2

Note that even while expressions E_1 and E_2 are exclusive, E_2 is not marked implicit, when E_1 is true. This is because, having 'right-to-left' reading, when ever the state of E_1 is read, the state of E_2 is already known and thus true state of E_1 won't provide great help in determining expression E_2 state.

3.4.1.4 Pair Expression Notation

It is worth noting, that most of the handled variables are conditional variables of form $V|C$, meaning that they are defined only under definition context C. For convenience, all variables are assumed to have definition context C so that for original system variables of form X the definition context is simply defined as true $p(C_X)=1$. It is sometimes convenient to mark reduced variable with notation $V'=V|C$ with implicit context, so that $p(V')=p(V|C)$, $p(V'\wedge Y)=p(V\wedge Y|C)$, $p(Y|V')=p(Y|V\wedge C)$ and $p(V'|Y)=p(V|Y\wedge C)$. In this situations e.g. pair expression definition becomes shorter $E_k=v_i'\wedge v_j'$ instead of $E_k=v_i\wedge v_j\wedge C_i\wedge C_j$ as we don't need to explicitly mark the contexts. When knowledge of the context is needed, it can be

referred through context function C, e.g. $C_V = C(V)$. In this paper, V' syntax is used only, when the variable is reduced because re-expression (reduction could also be caused by expression exclusion mechanism). If variable gets re-expressed a number of times, syntaxes V'', V''' and so forth are used to mark this.

As an addition to notation $E_k = v_i \wedge v_j$ notation $E_k = < v_i , v_j >$ is used to emphasize that the question is of pair expressions and not of any arbitrary conjugation. After all, pair expressions are not so much logical statements as they are expressions, that are used to re-express systems with very wide consequences to the system variables through variable determination and variable reduction mechanisms. Using this notation, for example $< A, B >$, $< A, \neg B >$ and $< \neg A, \neg B >$ are pair expressions. Another benefit of special notation is to emphasize the structured nature of pair expressions, e.g. $<< A, B >, C >$, $<< A, B >, \neg < C, D >>$ with the logical conjugation of form $E_k = A \wedge B \wedge C$ being ambiguous if interpreted as a pair expression. Also there is the important emphasis on the fact, that pair expressions are applied to different generations of system. This means, that $<< A, B >, C > \neq < A, < B, C >>$, while $((A \wedge B) \wedge C) = (A \wedge (B \wedge C))$. In this notation, nested structures are also easier to express than in traditional language notation, e.g. $<< A, B >, C >$ compared to $E_1 \rightarrow A B$, $E_2 \rightarrow E_1 C$.

3.4.1.5 Pair Expressions And Regularities

This chapter demonstrates use of pair expressions for compression with an overly simplified example. Let's consider a heavily regular system with three states: $\neg A \wedge \neg B \wedge \neg C$, $A \wedge B \wedge \neg C$ and $\neg A \wedge B \wedge C$. Consider pair expressions $E_1 = < A, B >$ and $E_2 = < B', C >$ and the system states. The state expressions can be visualized with following logical table (green for E_1, orange for E_2):

S	A	B	C

000	0	0	0
110	1	1	0
011	0	1	1

Table 3. Pair expression and compression illustrated

The system S can be reinterpreted by using the pair expressions E_1 and E_2 so, that the variables A, B and C in their conditional reduced form no more vary. Instead the reduced A', B" and C' are always zero and have zero entropy. This effectively turns 3 variable systems with 8 possible states into 2 variable system with 4 possible states and because it is known that E_1 and E_2 exclude each other the amount of possible states can be reduced to 3. This state reduction results in a heavy compression rate for this heavily regular system. In fact, when bits of E_1 and E_2 are encoded with ideal codewords size of their entropy, the encoded system state size is ideal and is equal with the system entropy.

S	A'	B"	C'	E_1	E_2	S'
000	Constant 0	Constant 0	Constant 0	0	0	00
110	Implicit 1	Implicit 1	Constant 0	1	0	10
011	Constant 0	Implicit 1	Implicit 1	Implicit 0	1	1

Table 4. Relationship between S and S'. Note reduced binary format size

3.4.2 Language Formation

3.4.2.1 Concepts And Design

This chapter introduces the logic that is used to formulate pair expression language for estimated system. In this chapter we introduce following concepts:

1. Pair Expression Language. Pair Expression Language does not only contain a number of ordered pair expressions and starved variables, but also knowledge of all variables present in the system. Given language L, we can transform any sample $s' = L(s)$ in original system S into translated form s' in translated system S'. Equally, the language provides us the means to translate system S' state s' into original form $s = L^{-1}(s')$.

2. Pair Expressed Data. The data contains samples in either translated form or in an intermediate form, which is easy to convert to both translated and original formats.

3. Pair Expression Statistics. Statistics are gathered from the pair expressed data for enabling pair expression formation, predicting and compression.

4. Pair Expression Learner. Pair Expression Learner is the exact method, which determines that which pair expressions are formed and which are not.

Typically, how pair expression learning process happens, is that we have initial language $L^0 : R(S^0) \to S^0$, which doesn't hold any pair expressions, and for which it applies that $L^0(s) = s$. In this context we understand expression as a functions from initial system state s^i to translated system state s^{i+1} , so that expression is $E^i : R(S^i) \to S^{i+1}$. Languages are developed incrementally by introducing expressions one by one so that generation i language L^i can be defined as $L^i = E^i \circ E^{i+1} \circ \ldots \circ E^1 \circ L^0$ or recursively as $L^i = L^{i-1} \circ E^i$. We mark different re-expressed systems and system states following similar notation, so that system state s can be expressed into a translated form of any generation i so that $s^i = L^i(s)$, which is equal with $s^i = E^i(E^{i-1}(\ldots E^1(L^0(s))\ldots))$.

We also have some sample of system states $\{s_1, s_2, \ldots, s_n\} \subset S$, which we can express as a 'system' $S_D = \{s_1, s_2, \ldots, s_n\}$ that is essentially a subset of the original system's state set $S_D \subset S$. Equally we have translated $S_D^i = L_D^i(S)$, which is what we call pair expressed data or pair expressed sample.

For translated sample S_D^g special statistics are generated and maintained $ß^g = ß(S_D^g)$. The statistics contain for each translated system variable $V_i^g \in S_D^g$ defined in context C_i^g following values: $n(V_i^g \wedge C_i^g)$ describing that in how many samples V_i^g is both defined and true and $n(C_i^g)$ describing that in how many samples V_i^g is defined. Also, for all variables V_i^g and V_j^g with contexts C_i^g and C_j^g, the values $n(V_i^g \wedge C_i^g \wedge C_j^g)$, $n(V_i^g \wedge C_j^g \wedge C_j^g)$, $n(V_i^g \wedge V_j^g \wedge C_i^g \wedge C_j^g)$ and $n(C_i^g \wedge C_j^g)$ calculated and maintained. This numerical information can be used to calculate for each V_i^g and V_j^g probabilities $p(V_i^g|C_i)$, $p(V_j^g|C_j)$, $p(V_i^g|C_i^g \wedge C_j^g)$, $p(V_j^g|C_i^g \wedge C_j^g)$ and $p(V_i^g \wedge V_j^g|C_i^g \wedge C_j^g)$. This is all knowledge that is needed for pair expression language forming.

Pair Expression Learner is a function Λ, which accepts language L^g and statistics $ß^g$ and gives back pair expression $E^{i+1} = \Lambda(L^i, ß^i)$, which can be used to form another generation of the language $L^{i+1} = E^{i+1} \circ L^i$. After forming next generation of language, another generation of the system data $S_D^{i+1} = L^{i+1}(S_D^i)$ and statistics $ß^{i+1} = ß(S_D^{i+1})$ can be formed. One model for pair expression learner function is the one of greedy search, where the benefit of adding each potential pair expression E is measured with benefit function $b(E, ß)$, and the best possible pair expression is provided, if benefit value is above a certain threshold t. If there are no pair expression above threshold t the expression adding stops and the language is complete. So if we define $E_{pos}(L)$ as the set of all possible pair expressions for language L, we can define the pair expression learner function as:

$$\Lambda(L, ß) = \{E \in E_{pos}(L) | b(E, ß) > t \wedge \forall D \in E_{pos}(L), D \neq E : b(E, ß) \geq b(D, ß)\} \quad (9)$$

Having this model, the definition of language learner Λ simplifies into defining the benefit function $b(E, ß)$ and threshold t. In the later chapters we refer to

benefit function simply as $b(E)$ assuming implicitly that the used probabilities are always based on the statistics $ß^g$ that again are calculated based on examined language L^g and data S_D^g .

3.4.2.2 Pair Expression Inclusion Logic

Now, the real focus of this chapter is the pair expression learning or inclusion mechanism Λ , which minimizes the re-expressed system S^g 's naive entropy and forms the optimal language-statistics pair $(L^g, ß^g)$ for the prediction work. In practice, these aims support each other to the level that they are fundamentally just different sides of the same goal. Before moving forward, it needs to be emphasized that the ultimate goal is to model and approximate the system S and not the system sample $S_D \subset S$ as modeling S_D instead of S will end up overfitting in both compressing and predicting. After all, even when S_D may have similar statistical properties with system S , there properties are not the same, and of all properties the true system entropy is typically much smaller with S_D compared to the approximated system S . Even more, there is always the possibility that the random variation present in S_D will be misinterpreted as patterns, which naturally won't apply to system S . Still, this is not a concern, if pair expression is used to specifically to compress the system sample S_D . After all, the difference in 'solving' system S_D and solving system S is actually the only difference there is between the realms of compressing and probability predicting. For compressing, S_D and S are always the same, while with predicting such thing can never be assumed, but instead entropy of sample S_D is typically only a fraction of real system S 's equal.

Also another difference is that for compression problem the binary size of serialized language definition is also relevant, while for expressing it can be ignored. The

serialized language definition size can be estimated and it can be included in the pair expression inclusion logic, but as the emphasis on this paper is in predicting the actual formula is not introduced here.

Now, the similarity with both of compressing and predicting problems is that both goals are aided by the reduction in the naive system entropy, yet any arbitrary amount of pair expressions cannot be applied. When compressing, unlimited introduction of expressions will bloat serialized language definition horribly, while the consequence for predicting is overfitting.

Mainly out of curiosity, let's first consider the compression problem. When a pair expression $<v_i, v_j>$ is applied to the system, the naive entropy is reduced by a negative delta $\Delta_S(E) = -I(V_i; V_j)$, and the language definition will also increase by some $p_e(E_k)$ and to decrease the entire database (serialized data+language) it has to apply that:

$$I(V_i; V_j) > \frac{p_e(E_k)}{n} \quad (10)$$

If this equation is used for pair expression inclusion logic, it is worth noting, that when the amount of samples approaches infinity the right hand threshold reduces to zero and the effect $\Delta_H(E_k)$ on the system entropy starts to dominate the pair expression inclusion.

Still, there is the problem related to the equation (10), that it doesn't put a preference over a pair expressed combination state. Now, it is worth noting, that when we examine individual state combinations, statistical dependency of the state combination will inevitably predict mutual information $I(V_i; V_j)$ between the variables simply, because mutual information peaks, when for each state combination $(v_i, v_j) \in V_i \circ V_j$ it applies $p(v_i \wedge v_j) = p(v_i)\, p(v_j)$. So in fact, when finding out elevated mutual information it is sufficient to concentrate on the deviation in the dependency value:

$$d(v_i; v_j) = \frac{p(v_i \wedge v_j)}{p(v_i)\, p(v_j)} \quad (11)$$

If this value is above of below one for any state combinations, variables V_i, V_j are dependent. The problem with the dependency value is that the relevance of the state combination is heavily dependent of the state combination probability, in the way, that the bigger the state combination, the more relevant the state combination is for mutual information and actually for both compression and prediction work all together. For this reason a better indicator is bit amount ideally saved by re-expressing the state combination that is

$$p(v_i \wedge v_j) \log \frac{p(v_i \wedge v_j)}{p(v_i) \wedge p(v_j)} = p(v_i \wedge v_j) \log d(v_i; v_j) \quad (12)$$

In this case, we compare the bit amount, when encoding the state with two codewords with one for V_i and another for V_j against encoding the state with a codeword of length $\log p(v_i \wedge v_j)$. The state combination is true on probability thus $p(v_i \wedge v_j)$ resulting in the equation. When compressing, given n samples, the amount of information saved by the state re-expression in total can be compared to the price of adding a pair expression entry into the language definition. When ignoring, that the addition of the pair expression will also change encoding of the remaining states, the entity consisting of pair expressed data and language is compressed if it applies that

$$p(v_i \wedge v_j) \log \frac{p(v_i \wedge v_j)}{p(v_i)\, p(v_j)} \geq \frac{p_e(E)}{n} \quad , (13)$$

where the $p_e(E)$ is the entropy of the pair expression entry or in other words how much the language definition grows in bits. With this function, when sample amount n is set to approach infinity, the right side of the equation will approach zero, and what is interesting is that for all variable pairs it always applies that there is a state combination that will fulfill this condition and therefore will cause a pair expression to be formed. After the pair expression is introduced, the same condition will again

apply to the starved variable pairs, as well as pairs of other variables and the formed pair expressions. Given this condition, what will happen is that all original variables are entirely eliminated by starvation and for each system state there will be a pair expression, which is true only for the system state and false for all other states. In this sense with infinite samples, for each system state there will be equivalent pair expression expressing it; the naive system entropy becomes equal with true system entropy (of the sample S_D) and the encoding transforms into system state encoding thus producing optimal encoding.

This improved equation, with the strong if not necessarily practical promise of 'eventually' finding the optimal expression (for the sample S_D), is chosen as the main pair expression inclusion logic for the task of compression, while accepting that there may be other functional models as well. For the equation the benefit function can be defined as

$$b(<v_i, v_j>) = n\, p(v_i \wedge v_j) \log \frac{p(v_i \wedge v_j)}{p(v_i)\, p(v_j)} - p_e(<v_i, v_j>) \quad , (14)$$

while the threshold is defined as $t = 0$.

3.4.2.3 Overfitting And The Pair Expression Inclusion Logic

It is worth noting that in the case only a small sample of states $\{s_1, s_2, \ldots, s_n\} \subset S$ in the original system are known, the inclusion logic presented in the previous chapter will not work to provide optimal expression for S, but for system S_D containing only a small sample $\{s_1, s_2, \ldots, s_n\} = S_D \subset S$. There are no practical guarantees that an expression designed for S_D will provide good compression rate, when applied to S. Quite in the opposite it is likely, especially if the sample is small, that the random variation present in S_D will be 'recognized' as patterns and additional

expression will be formed to reflect the random noise. Similarly, if the pair expression language formed for S_D is used for making predictions with S (as described later in this paper), it is likely that the pair expression language will suffer of overfitting, which will drive error to predictions.

There are different ways to try to avoid this kind of error. One simplistic method is simply introducing a threshold $t>0$ so that it is required for pair expression to have $b(E)\geq t$.

Another way is to try to minimize potential error in the predictions. If it appears, that the rate $p(a\wedge b)/p(a)\,p(b)$ is elevated, it will cause systematic bias of size $e_b=p(a\wedge b)-p(a)\,p(b)$ in the calculation. In prediction situation, it is beneficial to use approximate $p(a\wedge b)$ instead $p(a)\,p(b)$, in the situation, when the measurement error e_N of $p(a)\,p(b)$ plus systematic error e_b are both smaller than measurement error e_S of $p(a\wedge b)$. This results in a comparison

$$e_S<e_N+e_b \quad , (15)$$

which can be transformed into form

$$e_{AB}<e_A\,\hat{p}_B+e_B\,\hat{p}_A+e_Ae_B+(\hat{p}_{AB}-\hat{p}_A\hat{p}_B) \quad , (16)$$

where all errors e_X are of form

$$e_X=\sqrt{\frac{\hat{p}_X(1-\hat{p}_X)}{n}} \quad , (17)$$

in case the approximate $\hat{p}$ is simply the calculated average of n samples of binary variable X following Bernoulli equation. In case make assumption that probabilities p are equally likely, for n samples, where in k sample X is true, we gain a distribution

$$f_{P_X,K,N}(p_X,n,k)=(n+1)\binom{k}{n}p_X{}^k(1-p_X)^{n-k} \quad (18)$$

which expectation value, when assuming flat priori distribution f $f_{P_X}(p_X)=1, p_X \in [0,1]$, can be used as the estimate $\hat{p}$

$$\hat{p}=E(p|N(X)=k, N=n)=\frac{k+1}{n+2} \quad (19)$$

The distribution's standard deviation for can p be used as the error.

$$e_X=\sqrt{\frac{k+1}{n+2}\left(\frac{k+2}{n+3}-\frac{k+1}{n+2}\right)} \quad (20)$$

It is worth noting, that the pair expression inclusion equation based on prediction error will meet the system entropy based equation with big enough samples. When n approaches infinity, all errors e_{AB}, e_A, e_B will reduce to zero and equation will gain form $b>0$, which equals with $p(A\wedge B)>p(A)p(B)$ and $p(A\wedge B)/p(A)p(B)>1$ and $\log(p(A\wedge B)/p(A)p(B))>0$ and

$$p(A\wedge B)\log(\frac{p(A\wedge B)}{p(A)p(B)})>0 \quad (21)$$

Third way to approach the problem is to have naive assumption of statistical independence as the base assumption, and use pair expression only in the situation, when the likelihood, that variables are not independent is significantly higher than the likelihood that variables are independent. This can be expressed through following equation:

$$\frac{p(K=k, N=n|p(A\wedge B)=p(A\wedge B))}{p(K=k\wedge N=n|p(A\wedge B)=P(A)p(B))}>t \quad (22),$$

where t is threshold, and where the equation can be derived into form

$$\frac{p(A\wedge B)^k(1-p(A\wedge B))^{(n-k)}}{(p(A)p(B))^k(1-p(A)p(B))^{(n-k)}}>t \quad (23)$$

If variables are independent, left side of the equation equals 1. Putting the equation inside a logarithm, approximating $p(A\wedge B)=k/n, 1-p(A\wedge B)=(n-k)/n$, and applying

$$-H(A\wedge B)=p(A\wedge B)\log p(A\wedge B)+(1-\log p(A\wedge B))\log(1-\log p(A\wedge B))$$

(24)

provides the equation

$$n(-H(A\wedge B)-p(A\wedge B)\log p(A)\,p(B)-(1-p(A\wedge B))\log(1-p(A)\,p(B)))>\log t$$

(25)

The equation is true, if re-expression of state $s=A\wedge B$ will save all together over $\log t$ bits. The equation is very similar to the one introduced in chapter 3, except that it also contains saved bits for state's negation's information $\neg s=\neg(A\wedge B)$

$$n(-p(A\wedge B)\log\frac{p(A\wedge B)}{p(A)\,p(B)}-(1-p(A\wedge B))\log\frac{1-p(A\wedge B)}{1-p(A)\,p(B)})>t_0 \quad , (26)$$

where $t_0=\log t$. Another difference with this equation is that it may apply even in the situations, where following dependency value is below one

$$d(A;B)=\frac{p(A\wedge B)}{p(A)\,p(B)} \quad (27)$$

3.4.3 Predicting

3.4.3.1 Naive Bayesian

Predicting can usually be expressed in requesting the probability for variable Y , when the states $x_1, x_2, \ldots, x_n$ of variables $X_1, X_2, \ldots, X_n$ are known. This can be expressed as a conditional probability $p(Y|x_1\wedge x_2\wedge\ldots\wedge x_n)$, which can be derived into form

$$p(Y|x_1\wedge x_2\wedge\ldots\wedge x_n)=\frac{p(Y)\,p(x_1\wedge x_2\wedge\ldots\wedge x_n|Y)}{p(x_1\wedge x_2\wedge\ldots\wedge x_n)} \quad (28)$$

where the numerator is equivalent with $p(Y\wedge x_1\wedge x_2\wedge\ldots\wedge x_n)$, which can be

derived using conditional probability into form.

$$p(Y \wedge x_1 \wedge x_2 \wedge \ldots \wedge x_n) = p(Y)\, p(x_1|Y)\, p(x_2|Y \wedge x_1) \ldots p(x_n|Y \wedge x_1 \wedge x_2 \wedge \ldots \wedge x_{n-1})$$

(29)

If the naive independence assumption is made, conditional probabilities can be marked $p(x_i|Y \wedge x_1 \wedge x_2 \wedge \ldots) = p(x_i|Y)$ giving the original equation form:

$$p(Y|x_1 \wedge x_2 \wedge \ldots \wedge x_n) = \frac{p(x_1|Y)\, p(x_2|Y) \ldots p(x_n|Y)\, p(Y)}{p(x_1 \wedge x_2 \wedge \ldots x_n)} \quad , (30)$$

Unfortunately, the naive assumption rarely holds, which results causing systematic bias in the equation, which magnitude corresponds into the level of dependency between variables X_i . In fact, if the assumption does not hold, the result of the previous equation may be over 1 making it invalid probability value.

Because the previous equation does not always provide valid probability values and because $p(x_1 \wedge x_2 \wedge \ldots x_n)$ may be difficult to estimate another form of the equation is used. By utilizing the knowledge that $p(Y) + p(\neg Y) = 1$ and replacing context variables with $i = x_1 \wedge x_2 \wedge \ldots x_n$ the original bayesian equation can be turned into following form

$$p(Y|i) = \frac{p(Y|i)}{p(Y|i) + p(\neg Y|i)} \quad , (31)$$

resulting in equation

$$p(Y|i) = \frac{p(i|Y)\, p(Y)}{p(i)} \frac{p(i)}{p(i|\neg Y)\, p(\neg Y) + p(i|Y)\, p(Y)} = \frac{p(i|Y)\, p(Y)}{p(i|\neg Y)\, p(\neg Y) + p(i|Y)\, p(Y)}$$

(32)

Combining previous equations results in

$$\hat{p}(Y|x_1 \wedge x_2 \wedge \ldots \wedge x_n) = \frac{\rho(Y, x_{1,} x_{2,} \ldots)}{\rho(Y, x_{1,} x_{2,} \ldots) + \rho(\neg Y, x_{1,} x_{2,} \ldots)} \quad ,(33)$$

where

$$\rho(Y, x_{1,} x_{2,} \ldots) = p(Y)\, p(x_1|Y)\, p(x_2|Y)\, p(x_3|Y) \wedge \ldots \wedge p(x_n|Y) \quad (34)$$

and equivalently

$$\rho(\neg Y, X_{1,} X_{2,} \ldots) = p(\neg Y)\, p(X_1|\neg Y)\, p(x_2|\neg Y)\, p(x_3|\neg Y) \wedge \ldots \wedge p(x_n|\neg Y) \quad (35)$$

This series of equations provides approximation $\hat{p}(Y|x_1 \wedge x_2 \wedge \ldots \wedge x_n)$, that is accurate under naive assumption and when the naive assumption does not old, it is well behaving in the sense, that it's value is always inside the range [0, 1].

3.4.3.2 Naive Bayesian And Pair Expression

The entire benefit of using pair expressions for predicting raises from the original system re-expression $S' = L(S)$ using the pair expression language function L. In the system $S' = S^g = (X_1|C_1^g, X_2|C_2^g, \ldots, X_n|C_n^g, E_1^g|C_{E1}^g, \ldots, E_g)$, where the conditional variables can also be referred with notation $V'_i = V_i|C_1^g$, it applies that its conditional variables are either exclusive $i > j, V_i' \rightarrow \neg C(V_j')$ or for each two conditional variable state v_i', v_j' the statistical dependency $d(v_i', v_j')$ is constrained to be close to 1. Both of these conditions reduce the systematic error resulting from naive assumption. The mechanism, what limits the statistical dependency $d(v_i', v_j')$ is clarified later. In this chapter we simply describe a simple mechanism for calculating the semi-naive bayesian predictions.

In typical prediction situation some information i is given, which describes the state of some set of variables in system S. The aim is to know the probability of unknown variable Y under the information i. In a very simplistic way of making the prediction,

given the original system S, there is usually some set of states $\{s_i, s_j, \ldots, s_k\} \subset S$ for which the stated informatio applies and which can be re-expressed using the language function so that $S_i' = \{s_i', s_j', \ldots, s_k'\} = \{L(s_i), L(s_j), \ldots, L(s_k)\} \subset S'$. A more complicated and sophisticated method would in finding re-expressed information i' that matches the information i for system S' , but this conversion has not been studied in this paper. After getting the matching states, in principle, the calculation of probability $p(Y|i)$ should be simple in the sense that:

$$p(Y|i) = \frac{1}{p(i)} \sum_{s' \in S_i'} p(Y \wedge s') \quad (36)$$

where the sigma sums up all the probabilities of all re-expressed system states, where Y is true. The problem with this model is that the state probabilities $p(s')$ are difficult to approximate. In here we replace $p(Y \wedge s')$ with $p(s'|Y)\,p(Y)$, which can be approximated under naive assumption with

$$\hat{p}(s'|Y) = p(v_i'|Y)\,p(v_j'|Y) \ldots p(v_k'|Y) \quad , (37)$$

where all $v_i', v_j', \ldots, v_k'$ are explicit variable states for system state s' . In case the state of all other variables are known, except for predicted variable Y, there will be in fact only two different re-expressed states that are $s'_Y = L(i \wedge Y)$ and $s'_{\neg Y} = L(i \wedge \neg Y)$ and the estimate $\hat{p}(Y|i)$ becomes

$$\hat{p}(Y|C) = \frac{\hat{p}(s'_Y|Y)\,p(Y)}{\hat{p}(s'_Y|Y)\,p(Y) + \hat{p}(s'_{\neg Y}|\neg Y)\,p(\neg Y)} \quad (38)$$

Now it is good to remind, that variables of form V' are simply used as a notation for conditional variables of form $V|C^g$. It means that when this notation is removed, previous naive approximate equation changes its form to

$$\hat{p}(s'|Y) = p(V_i|C_i^g \wedge Y)\,p(V_j|C_j^g \wedge Y) \ldots p(V_k'|C_k^g \wedge Y) \quad , (39)$$

where it is known that all variable contexts of form C_i^g are true for state s' , because variables with C_i^g false are excluded from the equation.

Now there is the question that how these conditional variables work in practice. Let's consider very simple system with variables Y, A and B. If the system is re-expressed with expression $<A,B>$, the system gains form $S'=(Y,A|\neg B,B|\neg E,E)$. Now, if we are predicting Y, when both A and B are true the naive equation turns into form $\hat{p}(s'|Y)=p(E|Y)=p(A\wedge B|Y)$, which in fact doesn't have systematic error so that $\hat{p}(s'|Y)=p(s'|Y)$. Similarly for $\neg A\wedge B$ the estimate is $\hat{p}(s'|Y)=p(B|\neg E\wedge Y)\,p(\neg E|Y)$ that can be turned to following form by remembering that $B\wedge\neg E=B\wedge\neg A$:

$$\hat{p}(s'|Y)=\frac{p(B\wedge\neg E\wedge Y)}{p(\neg E\wedge Y)}\frac{p(\neg E\wedge Y)}{p(Y)}=\frac{p(B\wedge\neg A\wedge Y)}{p(Y)}=p(s|Y) \quad , (40)$$

meaning that it doesn't incorporate error. Finally the last estimate for $a\wedge\neg B$, where variable state a can be A or $\neg A$, is

$$\hat{p}(s'|Y)=p(a|\neg B\wedge Y)\,p(\neg B|\neg E\wedge Y)\,p(\neg E\wedge Y) \quad . (41)$$

Remembering that $\neg B\wedge\neg E\wedge Y=\neg B\wedge Y$, it is easy to prove that also the last kind of estimate is also biasless:

$$\hat{p}(s'|Y)=\frac{p(a\wedge\neg B\wedge Y)}{p(\neg B\wedge Y)}\frac{p(\neg B\wedge Y)}{p(\neg E\wedge Y)}\frac{p(\neg E\wedge Y)}{p(Y)}=p(a\wedge\neg B|Y)=p(s'|Y) \quad . (42)$$

This means that the pair expression mechanism with the conditional variables work to remove bias driven into the predictions by the naive assumption.

3.4.3.3 Limitations To Naivety

First, let's consider conditional probability $p(x_i|Y\wedge x_j\wedge x_k\wedge\ldots\wedge x_l)$ approximated by $p(x_i|Y)$. The probability can be derived into form

$$\frac{p(x_i \wedge Y \wedge x_j \wedge x_k \wedge \ldots \wedge x_l)}{p(Y \wedge x_j \wedge x_k \wedge \ldots \wedge x_l)} \quad , (43)$$

where the upper part can be expressed following the equation

$$p(a_1 \wedge a_2 \wedge \ldots \wedge a_l) = \sum p(a_i) - \sum p(a_i \vee a_j) + \sum p(a_i \vee a_j \vee a_k) - \ldots \quad , (44)$$

which can be derived into form

$$p(a_1 \wedge a_2 \wedge \ldots \wedge a_l) = \sum p(a_i) - \sum (1 - p(\neg a_i \wedge \neg a_j)) + \sum (1 - p(\neg a_i \wedge \neg a_j \wedge \neg a_k)) - \ldots$$

(45)

Under independence assumption this probability is approximated as

$$\hat{p}(a_1 \wedge a_2 \wedge \ldots \wedge a_l) = p(a_1)\, p(a_2) \ldots p(a_l) \quad , (46)$$

which is equal with

$$\hat{p}(a_1 \wedge a_2 \wedge \ldots \wedge a_l) = \sum p(a_i) - \sum (1 - p(\neg a_i)\, p(\neg a_j)) + \sum (1 - p(\neg a_i)\, p(\neg a_j)\, p(\neg a_k)) - \ldots$$

(47)

Now by using the difference $p(\ldots) - \hat{p}(\ldots)$ between the real value and the approximation, we can determine the inherent approximation's systematic bias

$$-\sum (p(\neg a_i)\, p(\neg a_j) - p(\neg a_i \wedge \neg a_j)) + \sum (p(\neg a_i)\, p(\neg a_j)\, p(\neg a_k) - p(\neg a_i \wedge \neg a_j \wedge \neg a_k)) + \ldots$$

(48)

which we mark with $e_b(\ldots)$. Modifying original equation we gain following form for $p(x_i | Y \wedge x_j \wedge x_k \wedge \ldots \wedge x_l)$

$$\frac{(\hat{p}(x_i \wedge x_j \wedge x_k \wedge \ldots \wedge x_l | Y) + e_b(\ldots | Y))\, p(x_i \wedge Y)}{p(x_i | Y)(\hat{p}(x_j \wedge x_k \wedge \ldots \wedge x_l | Y) + e_b(\ldots | Y))\, p(Y)} \quad , (49)$$

from where we can separate the relative bias component by replacing the context

variables with $c = x_j \wedge x_k \wedge \ldots x_l$ so that

$$p(x_i|Y \wedge c) = e \frac{p(x_i \wedge Y)}{p(Y)} = e\, p(x_i|Y) \quad , (50)$$

where the relative error component e is

$$e = \frac{\hat{p}(c|Y) + e_b(x_i \wedge c|Y) / p(x_i|Y)}{\hat{p}(c|Y) + e_b(c|Y)} \quad , (51)$$

which can also be expressed as

$$e = \frac{1 + e_b(x_i \wedge c|Y) / (p(x_i|Y)\, \hat{p}(c|Y))}{1 + e_b(c|Y) / \hat{p}(c|Y)} \quad (52)$$

Next, let's recognize that the relative bias is proportional to following values:

$$e \propto 1 + \frac{e_b(x_i \wedge c|Y)}{p(x_i|y)\, p(c|Y)} \quad (53) \quad e^{-1} \propto 1 + \frac{e_b(x_i|Y)}{p(c|Y)} \quad (54)$$

meaning that driving $e_b(\ldots)$ to zero will 'neutralize' relative bias by driving it to 1 and that $e_b(\ldots)$ can be re-expressed as

$$\begin{aligned} & -\sum p(\neg a_i)\, p(\neg a_j) \left(1 - \frac{p(\neg a_i \wedge \neg a_j)}{p(\neg a_i)\, p(\neg a_j)}\right) \\ & + \sum p(\neg a_i)\, p(\neg a_j)\, p(\neg a_j) \left(1 - \frac{p(\neg a_i \wedge \neg a_j \wedge \neg a_j)}{p(\neg a_i)\, p(\neg a_j)\, p(\neg a_j)}\right) \\ & + \ldots \end{aligned} \quad (55)$$

This means, that the relative error can be further 'neutralized' by driving equations of form

$$p(\neg a_i)\, p(\neg a_j) \ldots p(\neg a_k) \left(1 - \frac{p(\neg a_i \wedge \neg a_j \wedge \ldots \wedge \neg a_k)}{p(\neg a_i)\, p(\neg a_j) \ldots p(\neg a_k)}\right) \quad (56)$$

to zero. We can recognize a familiar component in the equation, that is the measure for statistical dependency

$$d(\neg a_1;\neg a_2;\ldots;\neg a_n)=\frac{p(\neg a_1\wedge\neg a_2\wedge\ldots\wedge\neg a_n)}{p(\neg a_1)\,p(\neg a_2)\ldots p(\neg a_n)} \quad (57)$$

How much this value differ from 1 determines greatly the error of the naive bayesian prediction. The carrying argument of this chapter is that after pair expression forming all statistical dependency values should be close to 1 (or exactly 0).

Let's consider a situation, where probability $p(Y|x_1\wedge x_2\wedge\ldots\wedge x_n)$ is replaced by its expressed form $p(Y|v_1\wedge v_2\wedge\ldots\wedge v_m)$ where each v_i may be either pair expression, variable or starved variable. Because that all variables, for which

$$p(v_i\wedge v_j)\log d(v_i;v_j)\geq\frac{t_0}{n} \quad , (58)$$

where left side is the benefit function and t_0 is threshold, have been turned into starved variables and pair expressions, for each two variables their statistical dependency is limited from top by

$$d(v_i;v_j)<e^{\frac{t_0}{n\,p(v_i\wedge v_j)}} \quad , (59)$$

where the right side is always above 1 and will approach 1 from positive side, when n approaches infinity. This gives a guarantee against greater than one dependency values between two variables. Now, there is also a guarantee against below-one dependency values, that consist of two parts:

1. Below-one dependency value for state combination v_i, v_j predicts above-one dependency values in one of the $d(\neg v_i;v_j)$, $d(v_i;\neg v_j)$ and $d(\neg v_i;\neg v_j)$. This is not weak indication as it can be proved (e.g. by using definition of entropy), that below-one dependencies must be balanced by above-one dependencies in other variable states. Let w_i, w_j be a 'balancing' state with the highest above-one dependency value. When n approaches infinity, the balancing state dependency value will be limited from up by 1, meaning that the balancing positive state is eventually

eliminated by pair expression formation. When the 'balancing' positive state is removed, the statistical properties of the variable changes, thus bringing the original below-one state dependency value closer to 1. As long the state's dependency value remains below-one, the variable combination will be a subject to starvation as n approaches infinity until there are no state combinations with either below-one or above-one dependency values left (instead the expression reduces for expressing individual system states with exclusive individual expressions). It is also worth noting, that the lower the original below-one dependency value is, the higher the balancing state combination's dependency value will be and the earlier the below-one dependency value will be neutralized by pair expression formation.

2. Second observation is simply that the variable states combinations with below-one dependency values are very rare; and the lower the dependency value; the lower the probability that both states co-occur. E.g. for dependency value 0, the probability that both states are present equals zero. Now, let's consider situation, where the need for knowing probability $p(x|c)$ can be approximated with $p(c)$. This is the situation for example for heuristic function $p(\mathit{will\ win}|\mathit{game\ situation})$, where only the game situations that actually happen (or may happen) in the game need to be evaluated. In this situation the expectation value for the square error is $H(e(p(x|c))) = \int (p(x|c) - \hat{p}(x|c))^2 p(c)\,dc$ and the effect on the average error that is caused by bias in $p(x|c)$ is proportional to $p(c)$. The more negative the dependency value is, the less likely $p(c)$ becomes, and the more meaningless the resulting bias becomes. While this observation doesn't affect the actual bias, it means (with given assumptions) that the bigger the bias (for below-one dependencies) is, the less relevant it actually becomes.

Now, we have approached two-variable dependency values, but how about multi-variable dependency values of form $d(a_1; a_2; \ldots; a_n)$? Given a dependency value

for over two variables, is it possible to make strong guarantees, that eventually also these values will all approach 1? The short answer is no. The longer answer contains limitations either to the bias caused by such values or recognition of probabilistic mechanisms, which – while not giving strong guarantees – will probabilisticly drive the multi-variable dependency value closer to one. The weak 'guarantees' of the long answer are:

1. The observation, that given multivariable dependency value $d(a_1;a_2;\ldots;a_n)$, its below-one or above-one value predicts below-one or above-one dependency values $d(a_i,a_j)$ in state combinations (a_i,a_j) .

2. Now it is worth reminding, that for each state combination (a_i,a_j) with $d(a_i,a_j)>1$ there is always sample amount n, which after it will be re-expressed by a pair expression. Equally for each state combination (a_i,a_j) with $d(a_i,a_j)<1$, there is always sample amount n, which after the state combination's dependency is neutralized through starvation. Overall all multi-variable systems, which have two-state combinations with non-one dependency values, would eventually be re-expressed in a form, where all variables are either statistically independent or exclusive. This means, that as n grows larger, all multivariable systems, which state combinations' dependency values reveal dependency, will become increasingly rare, until they disappear as n approaches infinity.

3. The more variables in the multivariable system, the lower the probability $p(\neg a_1)\,p(\neg a_2)\ldots p(\neg a_n)$ in the equation 56 and therefore the lower the effect on the error.

Overall, the bias resulting from naive assumption is considered to be limited, because two-variable dependencies are limited with strong guarantees (as n grows large) and 3+ variable dependencies (of original system) are limited with weak guarantees. This limits the error of naive predicting.

3.4.4 Pair Expression Mechanism Properties

First, let's have few thoughts around how pair expression behaves with samples of varying sizes from the perspective of approximation error and measurement error. Typically, there is a competition between approximation error, which relates to that how fine grained or clumsy models or rules are constructed to express the knowledge, and measurement error, which raises from the errors in measuments and from the limited supply of observations. Approximation error can be reduced by increasing granularity of modeling, but this typically leads to greater measurement errors.

Now, given very small samples, pair expression behaves like naive bayesian and it tries minimize measurement error (that dominates small sample problems) simply by not having expressions and by having high approximation error. As sample count increases, the granularity is increased around state combinations, which are surprisingly common, thus reducing approximation error with minimal cost to measurement error (because measurement error is reduced for surprisingly common state combinations). As sample count approaches infinite, pair expression forms one expression for each existing system state thus reducing to ordinary system state trace thus minizing approximation error. In this sense the algorithm is well behaving, as it works to find a proper compromise between the two errors.

As a second topic, there are few blind spots, which are difficult for pair expression style pattern recognition and predicting (which both are essentially the same thing). Perhaps the most significant one is the 3-variable xor system, where each variable's probability is 0.5. Given variables $X_1 = X_2 \, xor \, X_3$ the 3-variable dependency values are e.g. $d(X_1, X_2, X_3) = 0$ or $d(X_1, X_2, \neg X_3) = 8$, which will all force very significant bias in the result, if these variables are not re-expressed via pair

expression. As as for each x_i, x_j the two variable dependency value is always $d(x_i, x_j) = 1$ it will stay below the pair formation threshold for any n meaning that the variables will not be re-expressed.

The inability to recognize the 0.5 probability 3-variable xor problem raises from the use of 2 variable windowing and its limitations. Generalizing, the same observations, that were recognized in the system analysis chapter apply here. Because 2 variable window is used, patterns requiring bigger windows may go unnoticed by pair expression formation or their patterns may be only partially solved. If pair expression fails the task of pattern recognition, systematic bias will inevitably raise in predictions. Fortunately even rather large patterns are often possible to treat by first recognizing their sub-patterns and by splitting the system variables into a greater number of lower entropy expressions, which are again more vulnerable to 2 variable window based analysis. Also in the real-world examples variable probabilities are rarely exactly 0.5 and despite this 'limitation' the algorithm itself demonstrated very powerful ability to learn and predict in the actual testing.

4 Evaluation

To be able to make a comparative evaluation of pair expression ability to learn and perform, the method is tested with pre-existing samples, which have been tested and evaluated with a number of other learning methods. Two different studies with public data sets and algorithm results were considered; a newer study (2006, Caruana, Niculescu-Mizil) and the older study (1994, StatLog). The performance of machine learners in the newer study were evaluated with 8 different metrics and evalution of pair expression with all 8 metrics had been a significant task of its own. For this reason data sets from 1994 study StatLog were used. [6][7]

4.1 Pair Expression Configuration

For proof-of-concept and for being able to test the learning machine design, an implementation of Pair Expression Learning Machine was implemented in Java. This chapter describes some solutions and configurations used in the implementation and also introduces a special notations for different configurations. In following chapters, various configurations are identified with this special notation.

First it must be emphasize that none of the ran configurations did try to re-express the predicted variables (~classes). Intuitively, increasing window size (and thus re-expressing) should be most beneficial to do around the predicted variables, but it was difficult to formulate accurate and efficient prediction mechanism, which would allow it. This means that the pair expression mechanism should perform badly for example on problems (e.g. binary sum), where the parameters are independent, but

outcome is a result of complicated formula. If the re-expression of the predicted variables was allowed, these kind of systems could be solved easily by starting the re-expressing from output-input variable pairs.

Now, the learning machine used following benefit function by default

$$b(E)=-n\,p(v_i\wedge v_j)\log\frac{p(v_i\wedge v_j)}{p(v_i)\,p(v_j)} \quad .(60),$$

while some configurations used benefit function

$$b(E)=-n\,p(v_i\wedge v_j)\log\frac{p(v_i\wedge v_j)}{p(v_i)\,p(v_j)}-n(1-p(\neg v_i\wedge v_j))\log\frac{(1-p(v_i\wedge v_j))}{(1-p(v_i)\,p(v_j))} \quad (61)$$

Configurations using this function are marked with b_2 .

For both benefit functions configurable threshold t was used. Threshold was adjusted manually for each sample to gain optimal performance. This was done by performing testing with various tresholds and using the threshold, which consistely provided best results. Threshold is strongly related to that how representative given sample S_D is of approximated binary variable system S . The better the sample S_D represents S the smaller the optimal threshold is. The errors behaved rather consistently in the sense, that lowering threshold below optimal threhold increased data as measured from train sample and increased error as measured in test sample. Increasing threshold lead to increase in both train error and testing error. Still, it appeared that there were sometimes few local optimums very close to the approximated optimal threshold.

In configuration notation $t=X$ is used to mark the threshold parameter. The predicted variables were not included in the pair expressing, because difficulties in finding a prediction formula, which would consistently improve results, if predicted variables were subjected to expression. There was another configuration, where also the predicted variables could be re-expressed, but because problems in the prediction algorithm both processing speed and accuracy of predictions were suboptimal and

they are not provided with these results.

For estimating variable probabilities $\hat{p}(V)$ and $\hat{p}(V|Y)$ there were two different models, where the other was the calculated average so that

$$\hat{p}(V)=\frac{k}{n} \quad , (62)$$

where n was amount of samples where variable V was defined and k was the amount of samples where variable V was both defined and true. This probability was used by default. Another probability was based on the probability distribution for p, when as priori we assume that all probabilities p are equal. This probability distribution is

$$f_{P_X,K,N}(p_X,n,k)=(n+1)\binom{k}{n}p_X{}^k(1-p_X)^{n-k} \quad (63)$$

which expectation value for p (when p has flat priori distribution $f_P(p)=1$) can be used as the estimate $\hat{p}$

$$\hat{p}=E(p|N(X)=k,N=n)=\frac{k+1}{n+2} \quad (64)$$

This estimate was called in this paper priori based estimate. Because probability values were also needed for variables (e.g. reduced conditional variables, pair expressions), which priori probability expectation value was not 0.5 (e.g. in case of pair expressions, where priori probability expectation was often 0.25), this priori based formula was modified into form:

$$\hat{p}=\frac{2p_0}{n+2} \quad , (65)$$

where p_0 is the priori expectation value for probability p . The approximate is not based on proof or solid math, but in a number of samples use of this approximate helped against overfitting and improved predictions. The configuration using priori based approximate was marked with 'Priori'.

It is worth noting, that no expressions need to be added to the language. In this case

predictions are based on the empty language L^0 for which $S = L^0(S)$. In this case, because the naive bayesian predictions are based on the original system S instead of expressed system $S' = L^g(S)$, naive bayesian learning is very simple to demonstrate with the Pair Expression implementation. To measure that how much benefit predicting on re-expressed system provides compared predicting based on original system, also the zero-expression naive bayesian predictions are presented for the test samples. The configuration acting as a naive bayesian predictor was marked with 'Naive'

Configuration Syntax	*Description*
'Naive'	Empty language having no pair expressions. Systems is acting in naive bayesian mode.
t=X	Threshold is X
'Priori'	Uses priori based approximates for probabilities. If configuration is not marked with 'Priori', average based approximates are used.
b_2	Alternative benefit function is used. If configuration is not marked with b_2 , main benefit function is used.

Table 5. The pair expression run's configuration syntax

4.2 StatLog Comparison Samples

The evaluation is performed by testing following data sets from StatLog:

1. German Credit data using a cost matrix
2. Heart data using a cost matrix
3. Australian Credit
4. DNA-primate splice junction gene sequences

5. Vehicle
6. Image segmentation
7. Shuttle

All data sets are available in the UCI Machine Learning Repository. The results for various learning machines can be found from both StatLog web pages. The results, their analysis and analysis on testing conditions and learning machines can be found in the StatLog book and in a paper done of StatLog results. [7][12][13][14]

According to StatLog project, Heart and German Credit data favor strongly statistical methods, while neural networks and statistical methods perform well on Australian Credit, DNA and Vehicle data sets, while Image Segmentation and Shuttle favor machine learning methods. Quite lot of data was numeric, and as pair expression deals with binary variables, the data had to be converted into binary variable format before processing. The test arrangements of StatLog project were followed as well as possible to provide comparable results.

Various configurations for Pair Expression implementation were tested and the best performing configuration was used. Configuration contained the way, how the data is turned into binary format, pair expression threshold and whether probability approximates were priori based. A separate measurements were collected for the pair expression predictor configuration, which had no pair expressions added and was essentially a naive bayesian predictor.

4.3.1 German Credit

The purpose in the german credit sample is to evaluate the customer's credit standing as 'good' or 'bad'. The sample requires the use of cost matrix, which assigns the cost

of misclassification. If good customer is classified as bad, the cost is 1, while if bad customer is classified as good, the cost is 5. This raises the threshold to classify customers as good, so that for classification the probability for customer to be good needs to be over 80%. Following StatLog introductions, the average costs for both train and test data were gathered, and the testing was performed with 10-fold cross validation for the 1000 observation sample. All available 20 categorial and numerical attributes were used for predicting. These variables contained for example status of existing current account, sex, marital status, age of applicant, duration of loan, how much money he was borrowing and job status.

The numerical attributes (loan duration and borrowed money) that seemed to follow negative exponential distribution were normalized by putting the attributes to logarithm. After normalization the normalized variables were divided into n different ranges of the same size and a bit was allocated to represent each of these ranges. If the number was within the associated range, the bit was true and false otherwise. Also a variant was tested where each bit was true, if the number was within its range or greater than its range. One to five bits were allocated to represent each scalar variable. In case of categories, one bit was reserved to represent each category. This transformation turned 20 categorial / numerical variables into 80 binary variables plus the predicted “good/bad” binary variable.

*Configuration**	*Expressions*	*Train Cost*	*Test Cost*	*StatLog Rank*
Naive, Priori, FillBits	0	.497	.534	(1.)
t=95, Priori, FillBits	~47	.477	.499	(1.)
t=95, Priori, FillBits, b_2	~16	.498	.532	(1.)
Naive, Priori, ExclBits	0	.477	.512	(1.)
t=95, Priori, ExclBits	~47	.472	.508	(1.)
t=95, Priori, ExclBits, b_2	~14	.474	.501	(1.)
Discrim (Statlog)	-	.509	.535	1.
NaiveBay (StatLog)	-	.600	.703	12.
Default (Statlog)	-	.700	.700	11.

Table 6. Results for German Credit sample. (*Configuration syntax is described in table 5)

What surprises in the calculation is the sharp difference in the StatLog naive bayesian performance and the naive bayesian performance used in my measurements. The difference may be explained by the fact, that StatLog used naive bayesian as classifier (where Y_i with highest $p(Y_i)\,p(s|Y_i)$ is selected), while naive bayesian of this study used priori based approximates and normalized the probabilities (by comparing $p(Y_i)$ against $p(\neg Y_i)$) . Also this study's and StatLog's naive bayesian likely handled sample's attributes differently.

How ever, for the german credit sample the pair expression provided far superior results (.499 at best) compared to the best performer Discrim (.535) in the StatLog study. For the sample, pair expression had the benefit of most data being in categorial format as categorial data is very easy to process in binary format. Also the fact, pair expression aims to provide biasless probability estimates for predicted variables, is very beneficial when there are costs involved. Because the output consist of biasless estimates, the expectation values are easy calculate, and the optimasl decision borders are easy to draw based on biasless expectation value estimates.

Following print out describes the re-expressed system variables, that are most informative in predicting the credit standing of the customer. The syntax is in form “I(good|system variable)=mutual information, d(...)=log(dependency value), p(defined)=variable context probability”. Mutual information is very powerful tool in revealing dependencies between variables. In this case mutual information in magnitude of .7. The logarithmic dependency value is simply $\log(p(A\wedge B)/\,p(A)\,p(B))$ and positive values mean that the true states of system variable predicts good customer, while negative values mean that positive true of system variable predicts bad customer. Remember that re-expressed system variables have definition context C. All values (like mutual information and dependency) are calculated within this context. The context's probability is marked following syntax “p(defined)=X”.

```
i(good[1]|<!account[no], account[empty]>)=0.119/0.837, d(...)=-0.485, p(defined)=0.667
i(good[1]|<!savings[0-100], savings[unknown]>)=0.018/0.898, d(...)=0.249, p(defined)=0.788
i(good[1]|<creditHistory[noLoansOrGood], !
creditHistory[goodTillNowForThisBank]''''''>)=0.017/0.878, d(...)=-1.000, p(defined)=0.941
i(good[1]|<!duration[X≤37.07], <!duration[X≤19.82], !creditAmount[X≤3793.56]>>)=0.016/0.879,
d(...)=-0.619, p(defined)=1.000
i(good[1]|<duration[X≤7.10], duration[X≤11.31]>)=0.016/0.879, d(...)=0.358, p(defined)=1.000
i(good[1]|<!creditInBank[X≤1.75], <creditHistory[vip], !
creditHistory[goodTillNowForThisBank]>>)=0.013/0.831, d(...)=0.176, p(defined)=0.821
i(good[1]|<creditHistory[goodForThisBank], !
creditHistory[goodTillNowForThisBank]''''>)=0.013/0.857, d(...)=-0.711, p(defined)=0.900
i(good[1]|<age[X≤26.00], age[X≤33.00]>)=0.013/0.885, d(...)=-0.270, p(defined)=0.869
i(good[1]|creditInBank[X≤1.75]'')=0.012/0.937, d(...)=0.056, p(defined)=0.622
i(good[1]|<!account[no]'', account[0<X<200DM]>)=0.012/0.888, d(...)=-0.215, p(defined)=0.938
i(good[1]|<!purpose[forRadioTelevision], purpose[forNewCar]>)=0.010/0.877, d(...)=-0.192,
p(defined)=0.818
i(good[1]|<property[realEstate], !property[carOrOther]>)=0.010/0.840, d(...)=0.107,
p(defined)=0.616
i(good[1]|purpose[forUsedCar])=0.008/0.879, d(...)=0.253, p(defined)=1.000
i(good[1]|<creditHistory[vip], !creditHistory[goodTillNowForThisBank]>')=0.008/0.878,
d(...)=0.242, p(defined)=0.601
i(good[1]|duration[X≤19.82]')=0.007/0.820, d(...)=0.056, p(defined)=0.727
i(good[1]|<employement[0y-1y], !employement[1y-4y]''''>)=0.007/0.872, d(...)=-0.212,
p(defined)=0.938
i(good[1]|duration[X≤11.31]')=0.007/0.901, d(...)=0.251, p(defined)=0.911
i(good[1]|purpose[forRadioTelevision]'')=0.006/0.828, d(...)=0.082, p(defined)=0.583
i(good[1]|<home[rent], !home[own]>)=0.006/0.863, d(...)=-0.184, p(defined)=0.891
i(good[1]|<relationship[femaled&divorced/separated/married], !
relationship[male&single]>)=0.006/0.876, d(...)=-0.113, p(defined)=0.861
```

Output 1. The variables of re-expressed system S' that best predict good customer

What can be read from the data is that having account empty is the most informative sign of a 'bad' customer. It predicts strongly 'badness' as revealed by -0.48 logarithmic dependency value, and it is apparently quite common signal to be able to score so high mutual information value. Quite lot of the re-expressed system variables are simply original system variables bundled with some alternative category value like the ones predicting good for short loan durations and bad for people renting their apartment . Different category values are exclusive, which is a form of dependency and something pair expression mechanism is prone to eliminate. Still, there are more complex expressions including variable states from a number of different input parameters. For example a very informative positive signal comes from expressions “<!creditInBank[X≤1.75], <creditHistory[vip], ! creditHistory[goodTillNowForThisBank]>>” describing 'critical customer' with no more than one existing loan in the bank and very informative negative signal comes from expression “<!duration[X≤37.07], <!duration[X≤19.82], ! creditAmount[X≤3793.56]>>” describing long (over 37 months) and big loan (over 3800 Deutch mark).

It is worth noting, that variable reduction mechanism may raise counter-intuitive sounding results. For example, over 7 year employment as such is a positive signal. On the other hand, 7 year employment is strongly connected with critical account thus calling for pair expression. Now, if all critical account 7+ year old employed are pair expressed, the remaining non-critical account and over 7 year old employed are represented by a reduced variable, where this reduced variable – in fact - sends 'counter intuitively' a negative signal. This simply means that the people with 7+ year employment, who have been rejected this critical status, tend to be in fact worse than average customers, but this rather complicated meaning is very difficult to see from the 'dependency' print out and one may be left wondering why long employment of any form should be associated with bad credit standing. Overall, the hidden complexity of reduced variables calls for carefulness, when trying to interpret the meaning of any reduced variable of forms X', X'', X''', ... These kinds of traps are easier to avoid, when viewing the ordered language definition, where it is easy to see that which expressions do reduce which variables.

Also to demonstrate, how manual threshold was chosen, following diagram describes train and test error rates for various threshold, when priori is used and when data is expressed in form, where bit vectors are filled. The graph demonstrates classic symptoms of overfitting for threshold below 95 in the sense, that test errors increase, while train errors reduce. Also, it is worth noting that the graph is not smooth. This is explained by the idea that, while the formation of a pair expression does reduce entropy and is typically helpful, it is not guaranteed that a specific pair expression is helpful for predicting the specific output variable.

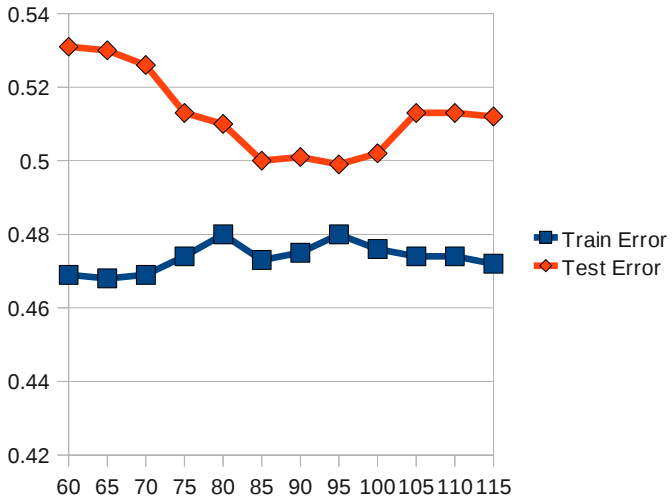

Diagram 1. Train and test errors for various thresholds for German Credit.

4.3.2 Heart

In the heart sample, there are 13 attributes that are used to predict the presence of the heart disease. The sample requires use of cost matrix with cost 1 for false positive (diagnosed healthy patient with heart disease) and 5 for false negative (failed to recognize heart disease). There are 270 observations and the testing is done with 9 fold cross validation. Sample's 13+1 variables were encoded in 75 bits. All thought all of the parameters were encoded as floating point numbers, most of the attributes were categorial.

The amount of samples was relatively low, and it was recognized that the average costs varied heavily from run to another. For this reason the average costs were

averaged from 40 separate test runs (each having 9 rounds of cross validation).

*Configuration**	*Expressions*	*Train Cost*	*Test Cost*	*StatLog Rank*
Naive, Priori	0	.343	.407	(shared 4.)
t=40, Priori	15.0	.326	.378	(2.)
t=40, Priori, b_2	6.9	.351	.386	(2.)
NaiveBay (StatLog)	-	.351	.374	1.
Default (StatLog)	-	.560	.560	15.

Table 7. Results for heart sample. (*Configuration syntax is described in table 5)

Naive Bayesian used in study could not reach the performance of the StatLog's naive bayesian. Still adding pair expressions did improve results significantly, but not enough to reach StatLog's naive bayesian predictor's performance. It is possible that the difference between the Naive Bayesians is in the way the attributes are handled. For example the StatLog Bayesian may be more able in processing numeric information.

4.3.3 Australian Credit

In the australian credit sample, there are 14 attributes that are used to predict the presence of the credit standing of customers. Details of what attributes describe are not revealed to maintain customer confidentiality. There are 690 observations and the testing is done with 10 fold cross validation. Sample's 14+1 variables were encoded in 80 bits. Because there was a degree of variation in each run's results, the errors were calculate based on averages of 20 runs with each run containing 10 cross validation rounds.

Configuration	*Expressions*	*Train Error*	*Test Error*	*StatLog Rank*
Naive, Priori	0	.124	.135	(2.)
t=150, Priori	9.0	.122	.133	(2.)

t=50, Priori, b_2	9.4	.118	.131	(shared 1.)
Cal5 (StatLog)	-	.132	.131	1.
NaiveBay (StatLog)	-	.136	.151	9.
Default (StatLog)	-	.440	.440	22

Table 7. Results for Australian credit. (*Configuration syntax is described in table 5)

Again the naive version of the learning machine is functioning differently compared to StatLog version in the sense, that it performed much better. Again the difference is likely in the way the attributes are handled. All non-categorial variables in the data set appeared to follow negative exponential distribution and they were normalized by using logarithm. Another important aspect is the amount of ranges/bits, that determines the granularity the learning machine can deal with attributes. The selected granularity contains a compromise between that how much information is lost in discretization and the error raising from potential overfitting. Overfitting may happen, if there are too few samples having numeric values in individual ranges.

4.3.4 DNA

In the DNA sample, there are 60 attributes representing DNA nucleotides that are used to predict a property of given DNA. There are three predicted classes and 3186 observations of which several samples contains missing nucleotides and were ignored for this study. The result was 3175 observations, of which 2000 were randomly picked as the train data and 1175 as the sample for testing. Sample's 60+3 variables were encoded in 243 bits. Because there was a degree of variation in each run's results, the errors were calculate based on averages of 10 runs.

*Configuration**	*Expressions*	*Train Error*	*Test Error*	*StatLog Rank*
Naive	0	.000	.001	(1.)
t=100	~210	.000	.002	(1.)

t=100, b_2	12.9	.000	.001	(1.)
Naive, Priori	0	.011	.012	(1.)
t=150, Priori	192.7	.005	.005	(1.)
t=80, Priori, b_2	18.0	.010	.011	(1.)
RBF (StatLog)	-	.015	.041	1.
NaiveBay (StatLog)	-	.052	.068	9.
Default (StatLog)	-	.475	.492	22

Table 8. Results for DNA sample (*Configuration syntax is described in table 5)

Again, the naive bayesian used in this study outperformed StatLog's equivalent as well as all the other algorithms. What is significant is that runs, which were using average based estimates instead of priori based estimates, were performing better. Priori's based approximation typically provides protection against overfitting and in most cases, priori based approximates are needed for the pair expression to provide any kind of meaningful results. When priori is used, the error rates are much higher to begin with, but introduction of expressions does provide significant improvement (.012 -> .005). When priori is used (estimates are averages), the error rates are low before re-expressing, but adding pair expressions increases errors apparently as a result of overfitting. For the sample, it seems to be beneficial to draw strong conclusions from patterns based small amount of examples (e.g. X seem to imply Y, but is X true only in 3 samples) and because priori based estimation downplays this kind of rules it leads to higher error rates.

4.3.5 Vehicle

In Vehicle sample, there are 18 numeric attributes that were generated based on vehicle silhouettes. The aim is to predict whether the vehicle described by silhouette properties is either Opel, Saab, Bus or Van. There are 846 observations and the learning system performance is measured with 9 fold cross-validation. Sample's 18+4

variables were normalized and encoded in 184 bits. Because there was a degree of variation in each run's results, the errors were calculate based on averages of 20 runs.

*Configuration**	*Expressions*	*Train Error*	*Test Error*	*StatLog Rank*
Naive, Priori	0	.404	.439	(21.)
Naive	0	.379	.423	(21.)
t=36, Priori	228.2	.208	.298	(shared 16.)
t=36, Priori, b_2	168.2	.207	.299	-18
Quadisc (StatLog)	-	.015	.150	1.
NaiveBay (StatLog)	-	.519	.558	23.
Default (StatLog)	-	.750	.750	24.

Table 9. Results for Vehicle sample (*Configuration syntax is described in table 5)

This the first sample, where it is very difficult to get the naive bayesian to provide good results. It is also worth noting, that Statlog version of Naive Bayesian is the worst performer for the data set. Under these kind of conditions, great amount of pair expressions can be added, and adding pair expressions provide great improvement in the prediction accuracy, when compared to naive predictor. Still, the resulting performance is below the average given the test data. Most competing algorithm perform quite similarly, in the sense that in StatLog ranks from 10 to 17 perform in the .275-.300 range. It is also worth noting, that unlike in the previous samples, Vehicle does not contain discrete categorial information. Instead all parameters are numerical and most of them are non-linear, which may in its own explain mediocre performance.

To demonstrate that how pair expression works in practice, following print of parameter-class dependencies is provided. The syntax in the print is of form "i(class[type] | variable[X≤4]) = 0.123 / 0.456". The class variable is on the left side following syntax class[bus]. On the right side there is a binary variable from the re-expressed system S'. The value " 0.123 / 0.456" describes, that how much of class'es variation is explained by re-expressed system variable. Number of left side is mutual information and the value on right side is the class variable entropy. Having e.g

0.5/0.5 would mean that the variables are equivalent and 0.0/0.5 and that they are independent. In syntax “d(...)=X” X is the dependency value in logarithm, so $\log d\,(class\,;\,var)$, while “p(defined)=X”, describes the probability X of the definition context, where the re-expressed system variable is explicit/defined and in which all mutual informations, entropies and dependency values are calculated. The fact that the context varies also explains, why class entropies vary from comparison to another.

```
i(class[bus]|maxLengthAspectRatio[X≤7.13]')=0.149/0.947, d(...)=0.507, p(defined)=0.684
i(class[van]|scaledVarMajorAxis[X≤173.32]'')=0.119/0.459, d(...)=2.880, p(defined)=0.576
i(class[van]|<!radiusRatio[X≤166.45], <!scaledVarMajorAxis[X≤185.42], <longatedness[X≤41.91], !
scaledVarMinorAxis[X≤390.92]>>>''')=0.071/0.965, d(...)=-Infinity, p(defined)=0.556
i(class[bus]|<<scaledVarMajorAxis[X≤173.32], <<prAxisRectangularity[X≤19.06],
prAxisRectangularity[X≤19.64]>, <scatterRatio[X≤153.73], scaledVarMinorAxis[X≤340.58]>>>'',
<distanceCircularity[X≤72.73], distanceCircularity[X≤79.27]>''>)=0.070/0.769, d(...)=1.890,
p(defined)=0.739
i(class[van]|<<scaledVarMinorAxis[X≤273.66], <!longatedness[X≤48.27], <scatterRatio[X≤139.82],
<scaledVarMinorAxis[X≤302.49], <prAxisRectangularity[X≤18.24],
prAxisRectangularity[X≤18.60]>>>>>, <scaledVarMajorAxis[X≤163.86], <!longatedness[X≤45.09],
<scaledVarMajorAxis[X≤173.32], <<prAxisRectangularity[X≤19.06], prAxisRectangularity[X≤19.64]>,
<scatterRatio[X≤153.73], scaledVarMinorAxis[X≤340.58]>>>>>>')=0.069/0.678, d(...)=2.329,
p(defined)=0.727
i(class[van]|compactness[X≤93.91]''')=0.068/0.812, d(...)=0.907, p(defined)=0.472
i(class[van]|<!prAxisAspectRatio[X≤63.60], <!radiusRatio[X≤208.09], <!radiusRatio[X≤187.27], <!
radiusRatio[X≤166.45], <!scaledVarMajorAxis[X≤185.42], <longatedness[X≤41.91], !
scaledVarMinorAxis[X≤390.92]>>>>>>'')=0.065/0.920, d(...)=-Infinity, p(defined)=0.648
i(class[van]|<!maxLengthRectangularity[X≤156.18]''', <!maxLengthRectangularity[X≤149.82]'', <!
maxLengthAspectRatio[X≤7.13], !maxLengthAspectRatio[X≤9.81]>>>)=0.064/0.821, d(...)=1.964,
p(defined)=0.846
i(class[bus]|<hollowsRatio[X≤183.73], <!skewnessMajorAxis[X≤80.04],
<kurtosisMinorAxis[X≤181.45], <<hollowsRatio[X≤186.45], <hollowsRatio[X≤189.18],
hollowsRatio[X≤191.91]>>, <<!skewnessMajorAxis[X≤74.26], kurtosisMinorAxis[X≤184.18]>,
<kurtosisMinorAxis[X≤186.91], hollowsRatio[X≤194.64]>>>>>>')=0.053/0.716, d(...)=1.897,
p(defined)=0.777
i(class[bus]|distanceCircularity[X≤79.27]''')=0.052/0.681, d(...)=1.699, p(defined)=0.493
i(class[van]|<scaledVarMajorAxis[X≤173.32]'', <<!skewnessMajorAxis[X≤74.26],
kurtosisMinorAxis[X≤184.18]>, <kurtosisMinorAxis[X≤186.91],
hollowsRatio[X≤194.64]>>'''>)=0.043/0.796, d(...)=2.055, p(defined)=0.967
i(class[opel]|<<!maxLengthAspectRatio[X≤7.13], !maxLengthAspectRatio[X≤9.81]>''''', <!
compactness[X≤98.09]'', <!distanceCircularity[X≤98.91], <!distanceCircularity[X≤85.82], !
distanceCircularity[X≤92.36]>>''>>)=0.043/0.821, d(...)=1.519, p(defined)=0.819
...
```

Output 2. The variables of re-expressed system S' that best predict vehicle classes.

While some original system variables are still present in their reduced form, one can recognize that most items in this ranking are pair expressions. Some pair expressions seem simple, while many of the shown pair expressions are in fact very complicated. The simplest ones at the top reveal some basic powerful characteristics of busses and vans like length aspect ration and compactness. Around the more complex expressions, it is a bit less clear what they do. It is possible that for example the expression associated with Opel (the last one in listing) describes the properties of

some very specific Opel model or family of Opel models. Despite the expression's complexity, the expression has to be relatively common to be able to score as high as 0.043 for mutual information with Opel. In many ways, the pair expression works to form its own internal classes of parameters that can be then associated with given bus/van/Saab/Open classes.

4.3.6 Image Segmentation

In Image Segmentation sample, 7 outdoor images were hand-drawn segmented to create classification for each pixels as one of brickface, sky, foliage, cement, window, path and grass. Each 3x3 region was associated with 19 numeric attributes, like different colors and location in the picture. The aim was to predict the pixel class based on these 19 attributes. The 19+7 system variables were converted into 79 bits. The testing was based on 10 fold cross-validation following the StatLog example. Because there were variation in the results, the results are based on 10 run averages, where each of rounds contain 10 rounds of cross validation.

*Configuration**	*Expressions*	*Train Error*	*Test Error*	*StatLog Rank*
Naive, Priori	0	.149	.154	(20.)
Naive	0	.124	.129	(20.)
t=65, Priori	293.9	.058	.072	(16.)
t=65, Priori, b_2	232.9	.065	.078	-17
ALLOC80 (StatLog)	-	.033	.030	1
NaiveBay (StatLog)	-	.265	.265	21
Default (StatLog)	-	.760	.760	23

Table 10. Results for image segmentation. (*Configuration syntax is described in table 5)

The sample resembles vehicle sample in the sense that both StatLog version and this study's version of naive Bayesians are both performing very badly to begin with. For both of these two samples there is the pattern that one can add lots and lots of pair

expressions to get a dramatic improvement (halves the original error rate), but even with dramatic improvement the performance is mediocre, when compared with other learning machines. According to STATLOG, the sample favors traditional machine learners (e.g. KNN) as compared to statistical methods and neural networks.

4.3.7 Shuttle

In the shuttle sample, the aim is to pick correct class (out of 7 possible) for each observation based on 9 numerical attributes. The test runs is based on 43700 train observations and 14500 test observations. Because, there was no random element involved in sampling, it made no sense to try make several runs and average results. Sample's 9+7 variables were normalized and stored in 79 bits.

*Configuration**	*Expressions*	*Train Error %*	*Test Error %*	*StatLog Rank*
Naive, Priori	0	3.53%	3.76%	(17.)
Naive	0	3.48%	3.62%	(17.)
t=75	879	0.08%	0.14%	8.
NewID (StatLog)	-	0.00%	0.01%	1.
NaiveBay (StatLog)	-	3.60%	4.50%	19.
Default (StatLog)	-	21.59%	20.84%	22

Table 11. Results for shuttle sample (*Configuration syntax is described in table 5)

Because huge amount of samples, there was typically no fear of overfitting. For this reason, the only real boundary for adding more pair expressions seemed to be the size of array, where co-occurrence information $p(V_1 \wedge V_2)$ is stored and the time pair expression adding took. Both of these things have O(v^2) component, where v is the amount of expressions and variables, meaning that when amount of expressions got close to thousand, the processing time magnitude got to million. There was a strong pattern of that the more pair expression were added, the better the results became, and the time limitations was main reason, why smaller thresholds were not used.

4.4 Secondary Properties And Performance

This chapter provides some information about processing times and some other secondary properties like the reduction in the parameter system entropy, which describes heavily the real entropy and regularities within the parameters:

Sample	*Train Samples*	*Binary Variables*	*t*	*Expressions*	*Naive entropy change*	*Train time (s)*	*Single prediction time (ms)*
German	900	80+1	95	45	46 -> 32	0.7	-
Heart	240	74+1	40	6.9	39 -> 34	0.05	0.05
Australian	621	79+1	150	9.0	41 -> 35	0.09	0.04
DNA	2000	240+3	150	192	195 -> 120	8-9	-
Vehicle	752	180+4	35	232	105 -> 48	7.3	0.56
Segment	2079	167+7	65	296	97 -> 37	12.4	0.99
Shuttle	43500	72+7	75	879	39-> 13	181.1	0.93

Table 11. Samples, pair expression runs and secondary properties (entropy, execution times)

In the table, the train binary variable amount follows notation P+C, where P is the amount of binary variables representing parameters, while C is the count of bits representing classes. *t* stands for threshold. Naive entropy change column describes the naive entropy of original parameters and naive entropy of the re-expressed parameters (class variables were not included in expression forming), so that the original system entropy is before "->" mark. All other samples, except for German and DNA, were running utilizing special bit converter package and a small framework to support testing. German and DNA had their specialized testing codes, which were written before creation of the special tools. For this reason for German and DNA samples, the train times are inaccurate and prediction times are missing. For reference, the processing was done in Java 6.0 desktop runtime environment in MacBook with 2.2 Ghz Intel Core 2 Duo processor.

To provide some way to measure the processing times compared to naive bayesian learner, following statistics were measured and now provided. Before looking at the data, it needs to be warned, that the values are based on single runs and are likely very error-prone to both measurement errors and systematic biases. Especially Java HotSpot mechanism tends to mess measurements, because the code gets optimized run time, and for this reason it tends to runs faster the longer it runs. This turns the execution times into a random variable dependent of time. The results should be considered only preliminary and directional, yet they provide some information about the magnitude. The most interesting values in following table are relative train time and relative prediction times, which describe that how much slower pair expression based system is compared to plain naive Bayesian learner.

Sample	*PairExp Train time (s)*	*PairExp Single prediction time (ms)*	*Naive Train time (s)*	*Naive Single prediction time (ms)*	*Relative Train Time*	*Relative Prediction Time*
Heart	0.05	0.05	0.02	0.05	2.5x	1x
Australian	0.09	0.04	0.05	0.03	1.8x	1.3x
Vehicle	7.3	0.56	0.3	0.09	23x	6x
Segment	12.4	0.99	0.7	0.14	18x	7x
Shuttle	181.1	0.93	4	0.06	45x	16x
Average	-	-	-	-	18x	6x

Table 12. Samples and pair expression performance comparison with naive bayesian

What the test data indicates is that re-expression based learner is easily around a magnitude slower than a naive Bayesian predictor in both training and predicting. For 5 tested samples, pair expression was on average 18 times slower than naive bayesian predictor in training and on average 6 times slower in predicting. The data would suggest rather intuitive rule that the more expressions are added, the slower both training and predicting becomes. Simply based on the implemented algorithm, the learning complexity should be $O(train\,time) = E((E+V)^2+N)$, where E is the

amount of added expressions, V is the amount of system variables and N is the amount observations in the sample. Based on learning algorithm, single prediction complexity should be $O(\mathit{prediction\,time}) = E(E+V)+K$, where $E(E+V)$ describes system state re-expression times and K is the amount of explicit re-expressed variables. K is limited between 1 and $E+V$ and depends of that how regular the system is and that how many variables are actually needed to express the information in the system.

5 Conclusions

This chapter provides the conclusions made as a result of the study. It contains conclusions that can be drawn from the various measurements and also conclusions that can be drawn based on theory and on the understanding accumulated during this study. The conclusions consider the different practical properties of pair expression, its measured / predicted cons and benefits and its potential promises.

5.1 Learning and Predicting

Plainly considering how pair expression performed on various samples, the measurement results suggest following:

1. The system produces better than average results, when compared to other learning systems in StatLog. Average rank for tested samples was 6^{th} or 7^{th} out of 22 plus default.

2. The system did not demonstrate bad performance for any of the samples. Even the worst case performances were not far below the median performance in comparison.

3. Re-expression improves naive Bayesian based predictions significantly in most cases and dramatically in some cases.

4. The more inter-dependent the parameters are, the more pair expressions can be applied and the greater the benefit from re-expression.

5. There appears to be association with statistical independence of parameters

and good results. This is explained by the naive assumption that is assumed for parameters by default (until proven false by statistical evidence).

6. There seems to be association with categorial parameters and good performance. This seems natural as the system 'runs on' binary variables.

7. There seems to be association with numerical parameters and below average performance, especially when compared to systems 'running on' numerical variables.

The fact that pair expression based learning machine did perform very well compared to other learning machines in StatLog is rather easy to base on the samples. In 4 cases out of 7, the performance was excellent (first or second) all thought for all these samples also the naive bayesian working on same binary variables provided quite similar results. In fact for DNA, best performance was reached in the 'naive mode' without the slightest re-expression. Worst performance was experienced in samples consisting entirely of numerical variables (image segmentation, vehicle) and having relatively few observations. Still even in these cases the performance was not far below the median performance of competing systems. While shuttle had solely numerical variables it also had large amount of observations (especially when compared to the parameter count) and the relative performance was good.

Compared to the naive bayesian learner it seems, that on data sets, where naive bayesian learner provides good results, the system provides slightly better results, all thought this is not guaranteed. On data sets, where naive bayesian performed poorly, the improvement was dramatic. Still even dramatic improvement did not guarantee good results, when compared to other learning methods in StatLog, but instead the errors for both vehicle and image segmentation were below average.

Still, while naive Bayesian's obvious weaknesses are heavily cross dependent parameters, it is suspected that for the pair expression based system the main obstacle were the numerical parameters. This especially, when compared to systems 'running on' numbers like various discriminant based methods, which performed very well for

vehicle, and certain kinds of decision trees, which performed very well on image segmentation. There are likely two sources of the performance difference, where the first one is related to the assumptions that can be made of scalars and that systems 'running on' numbers can utilize. These assumptions are for example closeness/distance (e.g. which numbers or categories are close to each other), orderliness and linear properties of numbers. The systems 'running on' numbers may also have in-build support to deal well with a number of scalar operations like sums and multiplications or even with non-linear operations like in the case of quadratic discriminants. Ability to make assumptions of data á priori is extremely beneficial especially with limited samples (as long the assumptions apply). The main benefit of these assumption is anyway the ability to 'fill out' the gaps in the sample data; figuratively connect the dots; that is to interpolate and also to extrapolate. The benefit comes mainly from the assumption of continuity of linear (or more complex) regularities outside the observed points. With large samples the gaps in the data are smaller and they get easily eliminated in discretization. This reduces the need for interpolation and extrapolation thus reducing the benefit of these assumptions for large samples.

The other source to the difficulties with numbers is likely the process, which turns scalars into binary presentation. For example, consider situation, where X is divided into three ranges that are $X<-1$, $-1<X<1$ and $X>1$ so that one bit is assigned to represent each of these classes. If there is a border separating classes A and B at X=0, there may be significant number of items belonging to middle category $-1<X<1$, which cannot be accurately assigned to any of two categories. If e.g. third of the items belong to this category, the error rate driven by this inaccuracy may easily end up being around 20%, while both discriminant and decision tree based systems can easily draw very accurate decision borders of this form thus giving them a significant advantage.

These two problems are related to each other in the way, that the most obvious fix for the problem with inaccuracy is the increase of categories and thus of granularity, but

this easily leads to overfitting or even to categories having no hits for the train data. If there are too few category hits for each category, individual hits and noise in measurements may start to dominate predictions leading to harmful overfitting. This phenomenon makes selection of 'right granularity' essentially a compromise between the error raising from discretization inaccuracy and the error caused by overfitting. Overfitting and empty categories could be fought with various assumptions (like closeness and continuity) around numeric variables, if these assumptions could be somehow in-build into the pair expression algorithms.

5.2 Supporting Human Understanding

Now, the sole idea of re-expression is to form expression that would be more informative than the old expression. The entire process is based on recognition of pattern and regularities in the original data and creating a language to reflect these regularities. So the pair expression language (when supported by statistics) reveals...

1. ...more informative expression for the data
2. ...patterns and regularities in the data

As such these properties would appear very beneficial, but to be able to support human understanding, it should be possible to interpret these mechanisms in human terms; with concepts of natural language. The basic re-expression mechanism of pair expression is very simple to understand as it is basically classification. For example, everyone knows, what it means if expression E resolves as 'yellow bird, which quacks and walks funnily'. It means that E expresses a duck.

The variable reduction mechanism of pair expression is somewhat more challenging, but it can be understood as raising from the priority of saying things (from abstract to details) and from the avoidance of redundancy. It means that there is always a

guarantee that the duckishness is always declared before yellowness, and the subject is never redundantly described as yellow, if it is already known to be a duck. This also means that subject is explicitly described as yellow only if it is not a duck. To some degree, this reflects a way of natural communication between humans. People typically try to use the most informative words for describing things, and only when they fail to find a suitable high level concept to describe the subject they dwell into the details. For example, when asking 'describe it', one typically says 'well – it has wheels' only when the person is puzzled, and the described item does not fit to the typical categories of cars, bikes, trucks and such. In this way, men sometimes use similar kind of 'reduced words', where the level of detail in the description already reveal that there is no pre-existing easy class/concept/word to describe the subject.

While one side of analysis is certainly that how easy it is to describe the expressions and mechanisms of pair expression in a human understandable form, another side is actually taking hands on approach and trying to analyze and interpret the output produced by the pair expression mechanism. Overall, pair expression was very useful in revealing patterns in the data. In the DNA sample, especially if the classes were also included in the re-expression, language was very useful in demonstrating the patterns and structures present in the DNA. For heart, German credit and Australian credit quite lot of the patterns 'solved' by pair expression were in fact redundancy caused by the binary presentation formation. While in remaining samples, the high amount of bits in number representation made the language definitions very large and complex, the list of best predicting expressions remained informative and somewhat easy to interpret. E.g. for vehicle the main problem appeared to be interpretation of original parameters (e.g. what longatedness and scatter ration mean?).

As a third topic, pair expression deals with binary variables and there is a number of helpful visual representations for the visualizing binary variable systems. In one visual representation, the binary variables are plotted to the space so, that the distance between binary variables describes the variables' dependency so, that variables, which distance is 0 are equivalent and the greater the distance, the less

dependent they are. This mechanism is very powerful in clustering the dependent binary variables in visual representation and demonstrating the variable dependencies. It can be further enhanced by drawing a color coded line between each variable to demonstrate relationship (e.g. red for exclusiveness, green for equivalence and such) and even by adding arrows to mark implications and their directions.

Overall, the pair expression mechanism seemed to provide useful and somewhat easy to interpret information of the data. Main problems were the complexity raising from high number of the bits used to represent numerical variables and the risk for misinterpretation because the complex variable reduction mechanism. It is believed, that the use of pair expression for analysis work could be greatly aided by graphical visualization.

5.3 Performance

Based on the results, the pair expression mechanism seemed to be around a magnitude (ten times) slower in both learning and predicting compared to plain naive bayesian, yet scaling worse (getting relatively slower as the variable, expression and sample count increased). Considering that naive bayesian is one of the fastest algorithms to teach and run, this cannot be considered as a bad result, but a good one. In StatLog, a number of algorithms were two or even three magnitudes slower than naive bayesian for most samples.

In case of pair expression based algorithm, there is lot of space for speed optimization in case of both learning and predicting. In learning work, one of the potential bottle necks for performance is the algorithm, that searches for the optimal potential pair expression, which is currently implemented in O(N^2) manner. It should be possible to replace with a sorted list, which is updated based on changes in the statistics, to reach either O(N) or O(N log N) performance. In the prediction work

typically quite many of the parameters could be ignored in exchange for a marginal error. Also in the prediction work, it may be possible to use specialized graphical or vector processing units to accelerate the statistical calculations.

5.4 Compression and System Entropy

For the test data, pair expression mechanism seemed to be useful for simplifying the original system. The reduction in the system entropy varied from around ten percent to up to two thirds. While likely part of the redundancy was driven into the system by the number discretization mechanism, e.g. 67% (ideal) lossless compression rate can be considered good for a system, which primary function is predicting.

Now, according to the information theory, it is always possible to find encoding scheme with average encoded system state bit length equaling the system entropy. Still, only the reduction in the system entropy was measured and encoding the system states with optimal bit length was not attempted and in fact a formulation of the encoding scheme seemed challenging. Encoding could be provided by finding optimal code words for each re-expressed system state, but in this case the definition for encoding becomes easily enormous and possessed by the curse of dimensionality. For a number of encoding schemes, the challenge was mainly that the re-expressed system variables are ordered and conditional based on this order and the encoding scheme should likely take into account the complex dependencies between re-expressed system variables. Even when pair expression does eliminate system's naive entropy, pair expression cannot be used for compression as such before suitable encoding scheme has been devised.

As described in the introduction, it would be highly beneficial, if re-expression could provide very heavily compressed expression in order to provide also memories or store details for environments that are best described as 'too complex, yet heavily

regular'. While the peek 67% entropy reduction may seem impressive, it is not even nearly enough for scenarios, where the need for compression is not described by percentages but by magnitudes. Still, in the 'Future Sights' chapter in the appendix there is a special chapter introducing an idea for interest-driven lossy compression perhaps providing a step in this direction.

5.5 Pair Expression Configuration

This chapter examines different configurations for pair expression and their usability for the predicting / learning purposes. Few configurations were tested for the samples. The main configuration variables used were

1. Threshold controlling the degree of re-expression
2. Priori based probability estimate against average based probability estimates
3. Type of benefit function

Around thresholds, the threshold controls that how much entropy gets eliminated via pattern recognition. The threshold is used to make the compromise between approximation error (resulting from lack of granularity) and measurement error (caused by too small window size), and it is naturally very sensitive to the entropy of noise $H(S)-H(S_D)$ present in the sample S_D from system S. If $H(S_D) \ll H(S)$ the threshold should be high, while if $H(S_D)=H(S)$ the threshold can be 0. Most of mutual information recognized beyond the system mutual information $H_{naive}(S)-H(S)$ can be expected to be noise and lead to overfitting, while some overfitting likely happens also before reaching this limit. Based on testing, e.g constant threshold of 70 may provide tolerably good results in most cases. Beyond these simple rules of thumb, it is very challenging to formulate any kind of equation for determining the correct threshold based on known values

like system naive entropy or sample count.

Still, if entropy $H(S)$ of original system would be known, the mechanism of learning could be devised into form, where pair expressions are added until the naive entropy $H_{naive}(S')$ of the re-expressed system reaches $H(S)$. In this approach the problem is naturally that $H(S)$ is typically not known. Another approach could be based on the biases and measurement error, where the systematic bias $e_b = p(A \wedge B) - p(A)\,p(B)$ for each pair expression can be compared to the measurement error $\sqrt{(1-p(A \wedge B))\,p(A \wedge B)/n}$. If the systematic bias is estimated to be bigger than the measurement error, pair expressions are formed, until no potential pair expressions exist fulfilling this condition. Both of these mechanisms would get entirely rid of the threshold and it is likely, that the best way to solve the threshold problem is not providing estimate for optimal threshold, but to eliminate the variable entirely. For these reasons, author does not attempt to provide any formula for determining the optimal threshold.

Next topic is the comparison of priori based probability estimating against average based probability estimating. Based on both the samples and understanding on how the system functions, it seems clear that pair expression mechanism could not function for most data without using some kind of priori based probability estimates. Both pair expression and variable reduction mechanism tend to drive variable probabilities into extremes (e.g. by turning variables into constants) and drive down the sample count, on which the probabilities are based on. This makes the system quite vulnerable for overfitting, which gets exaggerated when using average based estimates. While the fuzzier version of the estimates appear to provide superior results, it is worth remembering that the estimate formula is not based on analytical solution, and even better results could be achieved by using estimates based on more solid math.

Around benefit functions, what the testing demonstrated was that better results were gathered using the main benefit function (which was based on how many bits are

saved) instead of the alternative benefit function (which was based that on the probability that the variables are statistically dependent). Reason for this is unclear.

5.6 Future Sights

While the pair expression mechanism does already provide good predictions, it has also unleashed potential. For example it is expected that errors raising from overfitting could be reduced and better managed. Prediction mechanism that would allow the predicted variables to be re-expressed, would increase the learning ability and allow symmetry in predictions thus making the system more flexible. Developing a functioning encoding for pair expression and developing techniques like interest-driven lossy compression could provide a mechanism for very powerful compression techniques for e.g. storing meaningful memories or for modeling some decision making agent's environment. Combination of these techniques could make the pair expression very promising for integrating it strongly with a decision making agent. These topics are further discussed in the appendix A.

Even further there is a great potential in techniques, which would make the mechanism scale better and run even faster. Also the main two problems await solving that are the presence of threshold and the difficulties with numbers. The threshold configuration variables should be entirely eliminated from the inclusion logic. This could be done e.g. by basing inclusion on comparison of expected bias against the increase in measurement error.

5.7 Summary

Overall, the pair expression can be considered as a success. It is completely new kind of learning mechanism following a new kind of philosophy (expression/language driven learning), it is fast and it provided good results in measurements. Solely based on the testing, pair expression has already shown to be a useful tool for predicting and for human based analysis. It has a solid theoretical foundation in the probabilities, statistics and in information theory and properties that appear most interesting like smart windowing/synthesis, expression power, biasless probability estimation and the mixture of predicting and compressing. While the testing and usage demonstrated certain weak points like with numerical variables and the presence of 'manually configurable' magical value (threshold), it also shows great potential for significant improvement in the future, and it should be fruitful ground for further research.

Simply based on the results it appears plausible, that pair expression could be used to solve real practical problems already as it is. Especially it seems promising to use pair expression for categorial or binary information, for which it demonstrated great or even superior performance. Still for data that consist mostly or entirely of numeric variables, it is likely that learning systems, which internal logic is build around numeric variables, provide better results. Because pair expression mechanism provides biasless estimates for probabilities instead of classification, the mechanism seems to be especially useful, when alternatives are associated with prices or benefits, and biasless expectation values and decision borders need to be known. These conclusions shape the application area that would be most promising for pair expression, which could be e.g. in dealing with language data (like spam filtering), in decision making (e.g. game AIs) or in credit standing and health related predictions, where the testing did show superior results.

The theoretical properties of pair expression seem promising. It provides very smart windowing (which should minimize approximation error at a cost of minimal measurement error), powerful expression for regularities (no artificial limitations), biasless estimates, compression for information and a philosophy, which follows the

idea of regularity based modeling. It stands on the borderline of compression / expression and prediction work. It promises all these properties, while in its core it is based on extremely simplistic model of re-expression.

In the future, the compression rates can possibly be greatly increased by interest-driven lossy compression, and predictions can be made more powerful, accurate and symmetric with better prediction algorithms. The encouraging results in this study also tempts the imagination on what other language or re-expression based techniques could be adapted or invented for the purpose of machine learning.

6 References

(1) C.E. Shannon, "A Mathematical Theory of Communication", Bell System Technical Journal, vol. 27, pp. 379-423, 623-656, July, October, 1948

(2) Bellman, R.E. 1957. Dynamic Programming. Princeton University Press, Princeton, NJ.

(3) T Mitchell, B Buchanan, G DeJong, T Dietterich, P Rosenbloom, and A Waibel, Machine Learning, Annual Review of Computer Science Vol. 4 417-433, June 1990

(4) T.G. Dietterich. Machine learning research: Four current directions. AI Magazine, 18(4):97--136, 1997.

(5) Fisher, R. A., Theory of Statistical Estimation. Mathematical Proceedings of the Cambridge Philosophical Society (1925), 22:700-725

(6) Rich Caruana, Alexandru Niculescu-Mizil. An Empirical Comparison of Supervised Learning Algorithms. Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA, 2006.

(7) R. D. King, C. Feng, A. Sutherland, StatLog: Comparison of Classification Algorithms on Large Real-World Problems. Applied Artificial Intelligence: An International Journal, 1087-6545, Volume 9, Issue 3, 1995, Pages 289 – 333.

(8) D. A. Huffman: "A method for the construction of minimum redundancy codes," in Proc. IRE, vol. 40, no. 9, pp. 1098-1101, (1952).

(9) Thomas Bayes, 1765, An Essay Towards Solving a Problem in the Doctrine of Chances

(10) Minsky, M. (1961). "Steps toward Artificial Intelligence." Proceedings of the IRE 49(1):8-30.

(11) Lewis D David, Naive Bayes At Forty: The Independence Assumption in Information Retrieval

(12) UCI Machine Learning Repository. http://archive.ics.uci.edu/ml/

(13) StatLog Data Sets: comparison of results. http://www.is.umk.pl/projects/datasets-stat.html

(14) D. Michie, D.J. Spiegelhalter, C.C. Taylor, Machine Learning, Neural and Statistical Classification. StatLog Project Book. http://www.amsta.leeds.ac.uk/~charles/statlog/

(15) John McCarthy, 2007, What is Artificial Intelligence. http://www-formal.stanford.edu/jmc

(16) John McCarthy, 1969, Some Philosophical Problems From the Standpoint of Artificial Intelligence.

(17) Alan Turing, 1950, Computing Machinery and Intelligence

(18) Daniel Jurafsky, James Martin, Speech and Language, Processing. Second edition. Chapter 1.

(19) John E. Hopcroft, Jeffrey D. Ullman, 1969, Formal Languages and Their Relationship to Automata.

(20) Dana Ron, 1995, doctorate thesis for Hebrew University, Automata Learning and Its Applications

(21) J. Gerhard Wolff, Language & Communication, Vol.2, No. 1, pp. 57-89,

1982, Language Acquisition, Data Compression and Generalization.

(22) The Omphalos Context-Free Language Learning Competition, 2004, http://wwww.irisa.fr/Omphalos

(23) Alexander Clark, Springer Science + Business Media 2006, Learning deterministic context free grammars, The Omphalos competition

(24) Stuart Russell, Peter Norvig. Artificial Intelligence: A Modern Approach. 2nd Edition. ISBN 0-13-080302-2

(25) Simon Haykin, Neural Network, A Comprehensive Foundation, Second Edition. ISBN 0-13-908385-5

(26) Rafael C. Gonzales, Richard E. Woods, Digital Image Processing. ISBN: 0-130-94650-8. Chapter 8, Image Compression.

(27) J. Rissanen, IBM Journal of Research and Development, 1976, Generalized Kraft Inequality and Arithmetic Coding

(28) Jiawei Han, Jian Pen and Yiwen Yin, School of Computing Science, Simon Fraser University, 2000, Mining Frequent Patterns without Candidate Generation.

(29) Randall Davis, Howard Shrobe and Peter Szolovits, AI Maganize Volume 14, Number 1, 1993, “What Is a Knowledge Representation?”. http://groups.csail.mit.edu/medg/ftp/psz/k-rep.html

(30) Bernhard Nebel, Journal For Artificial Intelligence Research 12, 2000, On the Compilability and Expressive Power of Propositional Planning Formalisms.

(31) Cadoli, M., Donini, F. M., Liberatore, P., & Schaerf, M.,1996, Comparing space efficiency of propositional knowledge representation formalism. Principles of Knowledge Representation and Reasoning:

Proceedings of the 5th International Conference (KR-96), pp. 364–373

(32) Gogic, G., Kautz, H. A., Papadimitriou, C. H., & Selman, B., 1995. The comparative linguistics of knowledge representation. In Proceedings of the 14th International Joint Conference on Artificial Intelligence, pp. 862–869

(33) Anselm Blumer, Andrzej Ehrenfeucht, David Haussler and Mafred K. Warmuth, Learnability and the Vapnik-Chervonenkis Dimension, 1989, Journal of the ACM, Volume 36, Issue 4.

APPENDIX

A. Promising Techniques Around Pair Expression

A.1 Against Overfitting

In the testing, it was possible to find a threshold for every sample except for shuttle, which below the results got undermined by overfitting. The underlying cause behind the overfitting error is that the random noise in the sample starts to affect or even dominate the predictions. It happens rather easily, if variables, which probability is rather extreme (near zero or one), are used for making predictions. In this case, there are easily very marginal amount of surprising states (e.g. ones for very rare variable). When tracing the relationship between two of such variables, the dependency estimate may very easily get determined by very few co-occurrences of surprising states and the dependency may become drawn very flimsily based on very few samples thus leading to dramatic measurement errors.

Now, for pair expression overfitting happens easily, when using too big 'window' or in other words too many pair expressions. Expression variables tend to have significantly smaller probabilities than the original expressed variables and every step of re-expressing tend to increase measurement error at the same time as systematic bias or approximation error decreases. In any pair expression forming situation, there are two competing estimates, where the other estimate is affected by bias (the estimate $\hat{p}_{naive}(A \wedge B) = \hat{p}(A)\,\hat{p}(B)$), while the other estimate has greater measurement error (e.g. average based estimate $\hat{p}(A \wedge B) = n(A \wedge B)/n$). Now, to provide best possible estimate there are different strategies, 1) either pick the estimate with smaller error rate or 2) combine the estimates using formula:

$$\hat{X} = \frac{\sigma_2^2 \hat{X}_1 + \sigma_1^2 \hat{X}_2}{\sigma_2^2 + \sigma_1^2} \quad ,$$

where σ_1 is the standard deviation of estimate $\hat{X}_1$ and σ_2 is the standard deviation of estimate $\hat{X}_2$. It can be proven, that by combining the estimates by using the formula provides betsyer results than using any individual estimate. The improved estimate could be produced by inserting the error estimates into this formula. It may be possible, that by developing this mechanism further, such method for pair expression based predictions could be devised, that the addition of pair expressions would not result to worsening of the estimate under any condition. As such, it would eliminate the negative effect of overfitting entirely. Instead the re-expressed system error would converge towards some limit (likely close to the error gained now with optimal threshold) as more and more pair expression are introduced.

A.2 Symmetry and Re-expressing Predicted Variables

It seems plausible, that the predicted variables could be included for re-expression without losing accuracy in the predictions. This kind of mechanism would make the predictions entirely symmetric, meaning that after re-expression is done, any variables could act as predicted variables, while the rest of variables would act as known parameters. This kind of symmetry would provide great flexibility for the prediction work, in the sense that any set of information could be used for predicting the probability for any variable or condition. This kind of flexibility would likely be very beneficial for many sophisticated problems.

In fact, some means for providing predictions in entirely symmetric conditions were developed. One way was a method of state based predicting, where the probability of $p(Y|C)$ was transformed into form

$$p(Y|C) = \frac{p(Y \wedge C)}{p(Y \wedge C) + p(\neg Y \wedge C)}$$

Again the probabilities for any re-expressed system state s were calculated based on formula:

$$\hat{p}(s') = \hat{p}(v'_1 \wedge v'_2 \wedge \ldots \wedge v'_n) = \hat{p}(v'_1)\,\hat{p}(v'_2) \ldots \hat{p}(v'_n) = \prod \hat{p}(v'_i)$$

This provided rather inaccurate estimates, mainly because it didn't utilize the very sophisticated dependency values $d(v_i; v_j)$ available in the statistics.

For this reason, another estimate was used, which can be derived from the same naive assumption used for deriving the naive bayesian formula:

$$\hat{p}(s') = \hat{p}(v'_1) \ldots \hat{p}(v'_n)\hat{d}(v'_1; v'_2) \ldots \hat{d}(v_{n-1}; v_n) = \prod \hat{p}(v'_i) \prod_{\forall i,j : i<j} \hat{d}(v'_i; v'_j)$$

The problem with this formula were both performance and error, which appeared to be a result from overfitting. Pair expressing mechanism provided expression variables, which probabilities were very low. When dependency values were calculated for combinations of very rare expression variable states, it become rather likely that the variables never co-occurred or the frequency was otherwise dominated by noise from random variation. This lead to inaccurate dependency values for expression pairs and to major error in the formula. Performance problems were on the other hand a result of calculating approximately $O(N^2)$ dependency values.

Still, it is likely, that a number of performance and overfitting problems can be avoided with some rather basic means. Most dependency values disappear in any case in divisions of form: $p(Y \wedge C)/(p(Y \wedge C) + p(\neg Y \wedge C))$. The most basic problem to overfitting problem is to simply exclude or otherwise reduce the significance of low entropy variable combinations.

A.3 Interest-Driven Lossy Compression

In certain application areas, one of the main problems is the ability to deal with enormous complexity. One example is the decision making in an environment, that resembles the real world. In such situations, there is typically massive amount of information coming from various senses, yet the environment is extremely regular and good decisions can be made on rather limited amount of highly abstracted information. In these cases, great part of the problem raises from the way how the information from the environment is processed and transformed into high level abstractions. Part of this process is simply recognizing commons shapes and patterns in the surroundings and creating higher level concepts to reflect these. Another part of the process is just ignoring great of irrelevant information. Part of irrelevant information is simply noise and small variation what is left after high level concepts are extracted from the input data. Part of the irrelevant information may be high level concepts, which may be informative in the sense that they cover and explain lot of incoming information, but irrelevant in the sense that they provide no benefit or only marginal benefit in the decision making.

Pair expression does its small bit in compressing the incoming data and in trying to express the information in more compact, simplified and informative form. But after re-expression this compression can be taken even further. When we have re-expressed system variables on the other side and the interesting predicted variables on the other side, it is always possible to simply calculate mutual information between parameters and predicted variables and mark in all the parameter variables, which hold more mutual information with predicted variables than some threshold. Variables having mutual information below this threshold can be ignored for the price of some marginal error, that can be controlled by adjusting the threshold. All re-expressed system variables variables, that are left unmarked, can be just filtered out from the

expression and left out when compressing the information or when making predictions. Combining the compressing power of pair expression with the mutual information based filtering could be very powerful way for compressing data to retain only most interesting information. Here the term *interesting information* refers to information which is beneficial for some meaningful prediction work. We may think that some variables (e.g. credit standing) are interesting as such while often unknown, while some other known variables (e.g. employment) are interesting only as they help to determine the interesting variable probabilities. This formula could be e.g. used for providing highly compressed memories for decision makers.