Aleks Jakulin

Attribute Interactions in Machine Learning

Master's Thesis

Second Edition

Advisor: Prof. Dr. Ivan Bratko

17th February 2003

University of Ljubljana
Faculty of Computer and Information Science

Noni Angeli

Attribute Interactions in Machine Learning

Aleks Jakulin

Abstract

To make decisions, multiple data are used. It is preferred to decide on the basis of each datum separately, afterwards joining these decisions to take all data into consideration, for example by averaging. This approach is effective, but only correct when each datum is independent from all others. When this is not the case, there is an interaction between data. An interaction is true when there is synergism among the data: when the sum of all individual effects is smaller than the total effect. When the sum of individual effects is lower than the total effect, the interaction is false. The concept of an interaction is opposite to the concept of independence. An interaction is atomic and irreducible: it cannot be simplified or collapsed into a set of mutually independent simpler interactions.

In this text we present a survey of interactions through a variety of fields, from game theory to machine learning. We propose a method of automatic search for interactions, and demonstrate that results of such analysis can be presented visually to a human analyst. We suggest that instead of special tests for interactions, a pragmatic test of quality improvement of a classifier is sufficient and preferable. Using the framework of probabilistic classifier learning, we investigate how awareness of interactions improves the classification performance of machine learning algorithms. We provide preliminary evidence that resolving true and false interactions improves classification results obtained with the naïve Bayesian classifier, logistic regression, and support vector machines.

Keywords

- machine learning, data mining
- classification, pattern recognition
- interaction, dependence, dependency
- independence, independence assumption
- constructive induction, feature construction
- feature selection, attribute selection, myopic, information gain
- naive Bayes, simple Bayes
- naïve Bayesian classifier, simple Bayesian classifier
- information theory, entropy, relative entropy, mutual information

Acknowledgments

It was a true pleasure to work with my advisor, Professor Ivan Bratko. I am like a small child, fascinated with every tree along the path, distracted by every bird passing above. Ivan has a peerless feel for details, and he asks the kind of piercing questions that you can never answer with mere hand-waving, sometimes without questioning your paradigm. For example, the interaction gain formula would have never been explained, true and false interactions never told apart, and the definition of interactions never distinguished from a pragmatic test of interactions without his germane questions. He never let me down, in spite of obstacles and my failures. To say that I am merely grateful would be rude.

During the past months, my parents, Boris and Darja, and especially my grandmother, Angela, have virtually pampered me, alleviating me of everything they could do, running errands for me every day.

Janez Demšar is an exemplar colleague, an outstanding object-oriented designer, and a most witty writer. Many ideas in this work originated from conversations with Janez, and much of my work derives from and is built upon his. The Orange toolkit he co-authored with Blaž Zupan saved me much strain and programming effort. Blaž has introduced me both to machine learning research and to the problem of constructive induction, which he himself laid the cornerstones of with function decomposition. The breeding ground for the concept of interactions was function decomposition. My work was also affected by his fondness for visualization and his attentiveness towards the human analyst, the true user of machine learning tools.

Marko Robnik Šikonja patiently advised me on many occasions and provided much feedback and expert advice. Daniel Vladušič was always very friendly and helpful: most of the experiments were performed on his computer. Dorian Šuc reviewed drafts of this text, and suggested many improvements. I am grateful to Dr. T. Čufer and Dr. S. Borštner from the Institute of Oncology in Ljubljana, who have contributed the 'breast' data set, and to Doc. Dr. D. Smrke from the Department of Traumatology at the University Clinical Center in Ljubljana for the 'HHS' data set.

For many gratifying conversations, I would like to thank my colleagues, Aleksander Sadikov, Peter Juvan, Matjaž Bevk, Igor Kononenko, Ljupčo Todorovski, Marko Grobelnik, Janez Brank, Andrej Bauer. For many happy moments, I thank my friends, Mojca Miklavec, Miha Peternel, Jože Jazbec, Mark Sylvester, Jernej Starc, Aljoša Blažič, Janja Jereb, Matija Pajer, Iztok Bajec, and the idlas from #coders and #sezana.

Finally, this work would have never been performed without generosity of Slovenia's Ministry of Education, Science and Sport that supported me financially through the past 30 months. I am also grateful to Marcelo Weinberger, Gadiel Seroussi and Zvonko Fazarinc from Hewlett-Packard Labs, who introduced me to the world of research, with support from Hermes Softlab. I was strongly influenced by my inspiring secondary school teachers: Mark Sylvester, Manuel Fernandez and Bojan Kranjc.

Sežana, *Aleks Jakulin*
January 2003

CONTENTS

CHAPTER 1

Introduction

An engineer, a statistician, and a physicist went to the races one Saturday and laid their money down. Commiserating in the bar after the race, the engineer says, "I don't understand why I lost all my money. I measured all the horses and calculated their strength and mechanical advantage and figured out how fast they could run..."

The statistician interrupted him: "... but you didn't take individual variations into account. I did a statistical analysis of their previous performances and bet on the horses with the highest probability of winning..."

"... so if you're so hot why are you broke?" asked the engineer. But before the argument can grow, the physicist takes out his pipe and they get a glimpse of his well-fattened wallet. Obviously here was a man who knows something about horses. They both demanded to know his secret.

"Well," he says, between puffs on the pipe, "first I assumed all the horses were identical, spherical and in vacuum..."

Adapted from [Ver02]

When people try to understand data, they rarely view it as a whole. Instead, data is spliced, diced, cut, segmented, projected, partitioned and divided. Reductionism is the foundation of most machine learning algorithms. It works.

But there are pieces of knowledge and patterns of nature that spill and vanish if you slice them apart. One has to treat them holistically. But again, reductionism and simplification are crucial to our ability to generalize from the known to the undetermined. Why take blood samples if we know we can diagnose a flu by merely measuring the body temperature?

To resolve this eternal dilemma, a notion of *interactions* might be helpful. Interactions are those pieces of information which cannot be conquered by dividing them. As long as we do not cut into interactions, we are free to slash other data in any way we want.

Imagine a banker on Mars trying to classify customers in three basic classes: cheats, averages, and cash cows. The banker has a collection of the customer's attributes: age, profession, education, last year's earnings, this year's earnings, and debt.

The banker employs a number of subordinate analysts. He would like to assume that all these attributes are mutually independent, but dependent with the customer class. In the Caesarean style of "Divide et impera," the banker would assign each attribute to an individual analyst. Each analyst is an expert on the relationship between his attribute and the customer class, experienced from a number of past cases.

Once the analysts rush off with the data, they do not communicate with one another. Each analyst investigates his attribute, and on the basis of that attribute alone, he decides what class the customer is most likely to be in. Eventually, the banker calls all the analysts, and tells them to cast votes. Analysts who feel that they did not have enough information, may abstain from voting. He picks the class that got the most votes. If there is a tie, the banker picks assigns the customer to the worst class from those tied: it is better to treat a cash cow like a cheat, than a cheat like a cash cow, after all.

Unfortunately, there are two problems. First, several analysts may work with the same information. For example, once we know the profession, the education will not give us no additional information about the customer. That information becomes overrated in the voting procedure. We call these *false interactions*. False interactions indicate that the information about the label provided by the two attributes is overlapping: the same information is part of both attributes' deliveries. The sum of individual effects of falsely interacting attributes will exceed the true joint effect of all attributes. A concrete example of false interactions are correlated attributes.

Second, this year's earnings alone, and last year's earnings alone will not provide as much information as both earnings together. Namely, cheats tend to be those whose earnings suddenly drop, while they have to cope with retaining their former standard of living. We refer to these as *true interactions*. A truly interacting pair of attributes contains information about the label which can only get uncovered if both attributes are present. The most frequently used example is the exclusive-or (XOR) problem, where an instance is in class zero if the values of both binary attributes are identical, and in class one if the values are different. The sum of individual influences of truly interacting attributes is less than their joint influence. There is synergy among truly interacting attributes.

Therefore, we can describe interactions as situations in which the analysts should communicate with one another, as to improve the classification results. More realistically, the analyst who receives a pair of interacting attributes would derive a formula which unifies both attributes. For example, he would replace the two income figures with a new attribute describing the drop in income expressed as a percentage of last year's income. If we consider the value of that formula to be a new attribute, the analyst then forms his opinion merely on the basis of the relative reduction of income. The process is not much different with false interactions, where we try to filter out the hidden but relevant information shared by multiple attributes: for example by averaging multiple noisy measurements as to approach the true quantity measured.

Our example quite realistically describes the workings inside a computer when analyzing data, and arguably also inside the human brain when making decisions. The banker's procedure is similar to the naïve Bayesian classifier, and the source of its naïveté is in assuming that there are no interactions. Interactions are infrequent, so experts have long been fascinated by the surprisingly good performance of this simple method in face of sophisticated non-linear multi-dimensional hierarchical adaptively partitioning opposition. Namely, if there are no interactions, the naïve approach is optimal.

Our study will focus on the natural problem of identifying true and false interactions in a given classification problem. If we succeed, the banker will first invoke our procedure to decide which attributes are truly and which attributes are falsely interacting. With that information, he will be able to better divide up the work among the analysts. Our primary objective is therefore to visually present interactions to the human analyst, and assure they are insightful.

Such a procedure would also be useful to machine learning procedures. Once we discover what are the nasty interacting subproblems, we blast them away with the heavy multivariate artillery. But we only swat the simple subproblems with simple techniques. We will later see that simple techniques have advantages beyond their mere simplicity: because we take fewer assumptions and because the data is not as chopped up, we are able to obtain more reliable probability estimates. Therefore, our secondary objective will be to attempt improving the objective quality of probabilistic classifiers, as measured by evaluation functions.

In addition to that, we will briefly touch upon a large variety of endeavors that either cope with interactions, or might be benefitted by knowledge about them.

1.1 Contributions

- A framework for probabilistic machine learning, based on four elementary functions: estimation, projection, segmentation, and voting. A survey of evaluation methods of probabilistic classifiers.

- An interdisciplinary survey of interactions in machine learning, statistics, economics and game theory.

- A definition of an interaction in the above framework, using the notion of the segmentation function. A suggestion that an appropriate significance test of an interaction in machine learning should be associated with testing the significance of a classifier's improvement after accounting for the existence of an interaction, as estimated on unseen data.

- A classification of interactions into true and false interactions. Discussion of interactions in the context of supervised learning.

- An experimental survey of methods for detection of 3-interactions, with discussion of the relationship between an interaction and a taxonomy.

- A novel 3-way interaction probe, *interaction gain*, based on the concepts of information theory, which generalize the well-established notion of information gain.

- A proposal for visual presentation of true and false interactions, intended to provide insight to human analysts performing exploratory data analysis.

- An experimental study of algorithms for resolution of interactions, as applied to the naïve Bayesian classifier, logistic regression, and support vector machines.

1.2 Overview of the Text

In Chapter 2, we discuss a particular view of machine learning, based on uncertainty. We provide the mathematical skeleton onto which we later attach our contributions. Our survey of the concept of interactions is contained in Chapter 3. The reader may be intimidated by breadth, but it was our sly intention to demonstrate how fundamental the problem of interactions is.

Once we have run out of bonny distractions, we begin to chew the definition of interactions in Chapter 4. In accordance with the pragmatic Machiavellian philosophy ("Ends justify the means.") we propose that interactions are only significant if they provide an objective benefit. To be able to deal with objective benefit, we call upon a particular type of a learning algorithm, the naïve Bayesian classifier, and tie the notion of interaction with the notion of a segmentation function from our skeleton.

Since general k-way interactions are a tough nut to crack, we focus on interactions between three attributes in Ch. 5. We list a few traditional recipes for discovery of patterns which resemble interactions, and then provide our own, built on information theory. We conclude with a few schematic illustrations indicating how to visually present different types of interactions in the context of the set metaphor of information.

Our first stage of experiments is described in Chapter 6, where we focus on tying together all the definitions of interactions. The second batch of experiments in Chapter 7 explores how to present interactions in the domain to the user, and some practical benefits of being aware of interactions. Finally, we show in Chapter 8 that being aware of interactions in a domain helps improve classifier's performance. We make use of attribute reduction techniques as an improvement of Cartesian product for joining interacting attributes. We succeed in clogging the kitchen sink by listing a few unrelated notes and techniques in the Appendix A.

CHAPTER 2

Foundations

The question of whether computers can think is like the question of whether submarines can swim.

Edsger W. Dijkstra (1930–2002)

In this chapter, we will investigate the fundamentals of machine learning as applied to classification problems with nominal attributes. A strongly probabilistic approach to classification is endorsed. We explain how probabilities emerge, what they mean, and how we estimate them. We suggest a generalization of a probability in form of higher-order probabilities, where probabilities are assigned to probabilities. We list a number of important methodological guidelines.

Using the metaphors of uncertainty, gambling and decision-making, we show the usefulness of probabilistic classifiers. These metaphors provide us with foundations for evaluating probabilistic classifiers. We review a part of machine learning methodology and attempt to extract the few crucial procedures, the bricks most learning algorithms are built from. The reader should beware, because our view is both opinionated and obnoxiously abstract. This chapter requires some background in machine learning, statistics, and probability theory.

2.1 Machine Learning

One of the most active fields within machine learning is attribute-based supervised inductive learning. Given a set of *instances*, each of them described by a set of *attributes*, we try to predict the *label* of each instance. If the label is an element of a finite set of values, the problem is *classification*. If the label is a numeric quantity, the process is *regression*. In successive pages, we will focus on classification, but the concepts are also applicable to regression. The reader should note that when we mention 'machine learning' in this text, we normally mean attribute-based propositional supervised inductive learning.

The most frequently used measure of success of machine learning is classification accuracy, along with other objective measures of classification performance. Simplicity and understandability of knowledge are important features, should we try to help users understand their problem domain.

Simplicity is an intrinsic quality, in accordance with the Ockham's razor [Ock20] "Plurality should not be posited without necessity." Ockham's razor is predated by a Roman proverb "Simplicity is the hallmark of truth," which perhaps refers to the unlikely contraptions the liars need to construct to make their theories consistent. In his Physics, Aristotle wrote "For the more limited, if adequate, is always preferable," and "For if the consequences are the same, it is always better to assume the more limited antecedent" [Ell]. Finally, Albert Einstein highlighted the razor's link with truth as "Everything should be made as simple as possible, but not simpler."

Correctness is not caused by simplicity, it is merely correlated with it. The latent causes of both correctness and simplicity are inherent complexity of noise, and high prior probabilities of often used models and methods, resulting in their subsequently short descriptions: 'a' is a short frequent word, while the length of 'electroencephalographically' is only tolerable because the word is so rare; compare '+', 'ln' and 'arctan' in mathematics. Extensive and lengthy treatment of special cases is a hallmark of contrived theories.

Black box learning methods, such as neural networks, represent knowledge cryptically as a set of numeric weights. For that reason, they are referred to as subsymbolic learning algorithms. On the opposite side, symbolic learning methods, which arose from earlier work in artificial inteligence, focused on logical representation of knowledge. In parallel, many similar methods were developed in statistics, often predating similar ones developed in machine learning. Statistics has traditionally tried to describe knowledge to people, but its approach was more numeric than symbolic.

Initially, rules and classification trees were thought to provide the users with the most insightful view of the data. Symbolic learning was defended on the grounds of interpretability, as it often could not match the classification performance of subsymbolic and statistical techniques on the problem domains the methods were usually applied to.

Later, it was realized that simple numeric methods such as the naïve Bayesian classifier, based on additive effects and probabilities, are often preferred by users, especially if visualized with a nomogram. Visualization bridges the gap between symbolic and subsymbolic methods, providing insight through a powerful representation without giving up much classification performance.

2.2 Attributes and Labels

In this section we will examine how a typical classification problem is presented to a classifier. This representation appears to be narrow, especially from an idealistic artificial intelligence perspective, but it is useful and practically applicable.

Both learning and classification are performed on instances. Each instance has a number of attributes and a label. Each attribute may take a number of values. When an attribute is numerical, we refer to it either as a *discrete* or as a *continuous* numeric attribute. The set of values of a discrete numeric attribute is the set of integer numbers $\mathbb{Z}$, whereas the set of values of a continuous or real numeric attribute is the set of real numbers $\mathbb{R}$.

When an attribute takes only a discrete number of values, and these values are ordered, the attribute is *ordinal*. When an attribute's set of values is not ordered, we refer to it as a *nominal* attribute. Type of an attribute is not naturally derived from type of data, it is chosen. The type determines how an attribute will be treated, not what an attribute is like.

By form, the label appears to be another attribute, but its role distinguishes it. In fact, attributes are sometimes called non-categorical attributes, while the label is a categorical attribute, but we will not adopt these two terms: 'categorical' is sometimes synonymous with 'nominal'. Since we are dealing with classification, a label value will be called a *class*. The objective of classification is to assign an unlabeled instance to an appropriate class, having the knowledge of the instance's attributes. The objective of learning is to construct a classification procedure from labeled instances of the training set. In the training set, both an instance's attribute values and its class are known. It is important, however, that the classification procedure is applicable to previously unseen unlabeled instances from the test set: learning is more than rote.

In this text, we will focus on those learning problems where both the label and the attributes are nominal. We will assume that all the attribute values and classes are separate and independent from one another, and avoid any assumptions on how different attribute values and classes could be dependent on one other. Neither will we assume that attributes are in a hierarchy, in contrast to multilevel modeling. We assume no hierarchy of, or any other relationship between attribute values.

Assumption of dependence is often justified and normally takes the inconspicuous form of a value metric: body temperature of 37℃ is more similar to 37.1℃ than to 41℃. We cannot effectively treat continuous attributes without a value metric. Several authors claim that a metric should be learned and not assumed or chosen, e.g., [SW86, Min00, Bax97, KLMT00]. Recently, a lot of work has been invested in kernel-based learning methods. Kernels and metrics have a lot in common, and work was done kernel learning [LCB+02].

In formal treatment of these concepts, we will use the following denotation: A *classification problem* is a tuple $P = (\mathcal{A}, C)$, composed of a set of attributes $\mathcal{A}$ and a label C. An instance $\mathbf{i}$ is an element of a *world of instances* $\mathcal{I}$. Every attribute $A \in \mathcal{A}$ and the label C are maps. They map an instance I into an attribute value a: $A : \mathcal{I} \to \mathcal{D}_A$, where $A(\mathbf{i}) = v_{A,\mathbf{i}}$, $v_{A,\mathbf{i}} \in \mathcal{D}_A$. The *codomain* of the attribute A is $\mathcal{D}_A$, as is $\mathcal{D}_C$ the codomain of the label C.

Sometimes we refer to instances and attributes by their indices in respective sets $\mathcal{A}$ or $\mathcal{I}$, e.g., for an attribute value of a specific instance: $v_{i,j}$, $i = 1, 2, \ldots, |\mathcal{A}|$, $j = 1, 2, \ldots, |\mathcal{D}_{A_i}|$. The values of attributes X, Y for an instance $\mathbf{i}_j$ are $X(\mathbf{i}_j), Y(j)$. If we are interested in an attribute value regardless of an instance, we refer to it as x_k, $k = 1, \ldots, |\mathcal{D}_X|$, so k is an index of the value in the codomain, sometimes just as x. For an attribute A_i, its value with index k would also be $(a_i)_k$. Sometimes, the value of an attribute X may be undefined for an instance $\mathbf{i}_j$. We choose to assign it index 0 and then refer to it in several possible ways, depending on the context: $\{x_0, X(\mathbf{i}_j), v_{X,\mathbf{i}_j}, v_{X,j}, v_{i,j}\}$, if $X = A_i$.

When we refer to probabilistic concepts, we will use a slightly ambiguous but more compact notation, in which the attributes are $x_1, x_2, \ldots, x_n$, or represented all at once with a n-dimensional vector $\mathbf{x}$, while the label is y. When we use expressions with such notation, we refer to the properties of an idealized domain, without having to assume a particular set of instances.

There is no general consensus on terminology for these concepts [Sar94]. In neural networks, attributes are inputs, and labels are outputs. In statistics, attributes are independent, controlled or explanatory variables, predictors, regressors, sometimes attribute variables, while a label is a dependent or a predicted variable, an observed value, a response. The instances are sample values, training data is a sample or a contingency table, while an instance world is a population. In pattern recognition, attributes are features, instances of the training set are input vectors, and labels are outputs. In data mining, instances are records or rows, attributes are sometimes simply columns, and the set of instances is a database. Within artificial intelligence, a label is sometimes named a relation, or a goal predicate, the label is a classification, attributes may be properties, instances may be examples. In probabilistic approaches to learning, attributes are sometimes random variables. Numeric attributes are sometimes called real-valued or continuous. This tiny survey remains incomplete.

2.3 Classifiers

Although we have defined parameters to a classification problem, we have not yet defined the purpose and the form of learning. A deterministic *learning algorithm* is $L_D : (\mathcal{P}, \mathcal{W}) \to \mathcal{C}_D$, where $\mathcal{W}$ is a *universe of instance worlds* ($\mathcal{I} \in \mathcal{W}$), $\mathcal{P}$ is a classification problem as defined in Sect. 2.2. $d \in \mathcal{C}_D$ is a *classifier*, a target function, or a discriminant function, $d : \mathcal{I} \to \mathcal{D}_C$. $\mathcal{C}_D$ is the world of possible classifiers.

We learn by invoking a learning algorithm: $L_D(P, \mathcal{T}) = d$, $d \in \mathcal{C}$, where $\mathcal{T} \subseteq \mathcal{I}$ is a *training set*, a subset of the world of possible instances. A classifier maps an instance to its predicted class. With the term *knowledge* we will refer to the description of the classifier, while a classifier is the functional implementation.

The above formalization of learning is sometimes referred to as *discriminative learning*, while *probabilistic discriminative learning* refers to direct modeling of the posterior *probability distribution* of the label. Informative learning refers to modeling the label posteriors, roughly $P(\mathbf{i}|C(\mathbf{i})]$, and priors, $P(C(\mathbf{i})|\mathbf{i})$, with which we can arrive to the label posteriors via the Bayes rule.

Unsupervised learning, clustering, density estimation, hidden Markov models, and Bayesian networks are specific examples of *generative learning*. In generative learning there is no attribute that would have the distinguished role of the label. In association rule induction, we try to predict one set of attributes attributes from other attributes. In crisp clustering, we try to invent a new attribute which can be efficiently used to predicts all the others attributes. In Bayesian classification theory, generative learning refers to modeling of the joint probability distribution, $P(\mathbf{i})$.

2.4 Uncertainty

Now that we have defined classifiers, we will show why simple measurement of classification accuracy is often problematic, and why probabilistic approaches work better. We will also describe appropriate experimental methodology.

Instances of the *test set* or the evaluation set are $\mathcal{E} \subseteq \mathcal{I}$. It is not fair to evaluate a classifier on the data we trained it on, as it may simply remember all instances without gaining any ability to generalize its knowledge to unseen instances: we are not looking for trivial rote classifiers such as $d(\mathbf{i}) = C(\mathbf{i})$. Of course, an instance world may be deterministic, and we might have all the instances available and labeled. A rote classifier is then feasible, and in some sense optimal. Unfortunately, most domains are subject to uncertainty.

We should therefore evaluate a classifier on the test set $\mathcal{E}$, so that there is no overlap between the training set $\mathcal{T}$ and $\mathcal{E}$: $\mathcal{E} \cap \mathcal{T} = \emptyset$. In practice, we do not have a separate test set, and there are several good techniques for repeated splitting of the available set of instances into a training and a test set. The most frequently used method is the 10-fold cross-validation (10cv). Such techniques provide superior estimates of classifier quality in comparison with a single arbitrary training/test split. There are also non-empirical model-based methods, which assume that the data is generated in accordance with certain assumptions. But, as with metrics, we prefer to assume as little as possible.

The most frequent approach for evaluating a classifier is classification accuracy: on the test set, how many times out of total has classifier d correctly identified the class? Unfortunately, some domains' class distribution is unbalanced. For example, a classification problem may require deciding whether a particular transaction is fraudulent. But in our training data, only 1% of instances refer to fraudulent transactions. A dumb majority class classifier, which always predicts the most frequent class, will have 99% classification accuracy, but will miss all the frauds.

To avoid this, we introduce *cost-based learning* where mistakes may incur proportionally greater costs. We define a $|\mathcal{D}_C| \times |\mathcal{D}_C|$ *cost matrix* M. The cost of a particular act of classification of instance $\mathbf{i}$ is $\mathsf{M}(d(\mathbf{i}), C(\mathbf{i}))$. A learning algorithm attempts to minimize its classifier's cost on a test set, knowing the M. The trouble with this approach is that d depends on M, while we are sometimes not sure what M is. The simplest cost matrix is *0-1 loss*, where

$$\mathsf{M}(c_i, c_j) = \begin{cases} 0 & \text{if } i = j, \\ 1 & \text{if } i \neq j. \end{cases}$$

Minimizing zero-one loss is equivalent to maximizing classification accuracy. In a given transaction, we may have multiple costs and multiple benefits. Eventually, we either end up with a gain or with a loss.

2.4.1 Probabilities and Decision Theory

Instead of learning a set of classifiers d_{M}, for all imaginable M, we may view d stochastically. Instead of being interested solely in $d(\mathbf{i})$, we instead consider a *stochastic classifier*, which samples its deterministic responses randomly, according to some label *probability density function* (PDF). $\Pr\{d(\mathbf{i}) = c_k\}$, $k = 1, \ldots, |\mathcal{D}_C|$ can be seen as an approximation of the posterior probability distribution $P(C(\mathbf{i}) = c_k|\mathbf{i})$.

Later we will define *probabilistic classifiers* which explicitly present the probability density function to the decision-maker, instead of sampling their responses randomly. Stochastic and probabilistic classifiers are so similar that we will often pretend they are the same: a stochastic classifier is nothing else than a dice-throwing wrapper around a probabilistic classifier. Not to be too verbose, we will use the word 'classifier' instead of probabilistic classifier from now on. If a deterministic classifier accidentally enters the stage, we will pretend that it confidently assigns the probability of 1 to its one prediction. If a stochastic classifier comes in, we will pretend that its responses are sampled until a probability distribution of its responses for a given instance is obtained, and presented to the user.

Given a cost matrix M and the label probability distribution, we pick the class c_o which minimizes the predicted expected loss, or predicted *risk*:

$$c_o = \arg\min_{\tilde{c}\in\mathcal{D}_C} \sum_{\hat{c}\in\mathcal{D}_C} \Pr\{d(\mathbf{i}) = \hat{c}\}\mathsf{M}(\tilde{c}, \hat{c}).$$

This is similar to minimizing the *conditional risk* [DH73], or expected conditional loss, defined as $Risk(\tilde{c}|\mathbf{i}) = \sum_{\hat{c}\in\mathcal{D}_C} P(C(\mathbf{i}) = \hat{c}|\mathbf{i})\mathsf{M}(\tilde{c}, \hat{c})$. It is easy to see that results are optimal if the estimate of the probability distribution matches the true one. The optimal choice of class has the minimum conditional risk, and this risk is called *Bayes risk*.

We can compare the concept of risk to *utility*, a concept from decision theory [Mac01]: in an uncertain k-dimensional state of the world $\mathbf{x}$, the best action a_o will maximize the value of the utility function $U(\mathbf{x}, a)$:

$$a_o = \arg\max_a \int \mathrm{d}^k\mathbf{x} U(\mathbf{x}, a) P(\mathbf{x}|a).$$

Generally, utility is a subjective perception of quality. In the process of decision-making, we are trading benefits for costs, consequently ending up with net gains or losses, and evaluating these through positive or negative utility.

Similar computations are normally embedded inside cost-sensitive classification learners, whose classifications are adjusted to minimize risk for a specific M. If possible, it is better to separate these two operations into two independent modules that follow one another: one focusing on quality estimates of label posteriors, the other on deciding the least costly class.

Probabilistic knowledge is more complex than ordinary knowledge, but users prefer class probability to be shown than to be hidden. If simplicity is our goal, and if we know the probability cut-off, we can always extract a simpler representation of knowledge. Even without knowing M we can extract informative rules from knowledge, just like they they can be extracted from neural networks, should we fail to find suitable visual knowledge representations.

2.4.2 Gambling

We will use gambling examples to illustrate the preferences we might have for different estimates of a probability distribution, without assuming a particular cost matrix and without being excessively abstract. They also provide means of concretely computing the costs caused by erroneous posterior probability distribution estimates. And they are as ad hoc as anything else.

Assume a gambling game, with n possible outcomes. We estimate the probability of each outcome with p_i, $i = 1, 2, \ldots, n$. We place a bet of mr_i coins on each outcome. Once the outcome is known to be j, we get nr_j coins back. How should we distribute the coins, if our knowledge of the distribution is perfect?

We have m coins. Because we cannot incur a loss in this game if we play properly, it pays to use all of them. Let's bet m coins on the outcomes from $O = \{i : p_i = \max_j p_j\}$, so that $\forall j \notin O : r_j = 0$. If this was not an optimal betting, there would exist k coins that should be moved from an outcome $i \in O$ to an outcome $j \notin O$, so that we would earn more. We would then earn on average $nkp_j - nkp_i$ coins more. But since no $j \notin O$ has a larger or equal probability, we always make a loss. Such a bet is thus at least locally optimal, and optimal for $n = 2$. This is a *bold* strategy. The average profit made by the bold strategy in a game is $-m + nm \max_j p_j$. The expected *return on investment* (ROI) given a particular probability distribution is $n \max_j p_j \cdot 100\%$, which is maximum, so the bold strategy is max-optimal.

If we are ignorant about the probability distribution, we bet m/n coins on all outcomes, $r_i = 1/n$. Whatever the outcome, we never lose money, and our earnings in the worst case are n, implying ROI of 0%. Proportional betting is minimax-optimal, as we have a guaranteed bottom bound. In such proportional betting, we pay no attention to the probability distribution.

We could try a *timid* betting strategy, which takes the probability distribution into account: we bet on the proportion $r_i = p_i$ of m coins on outcome i. We thus spend all the coins, since the probabilities sum up to 1. The expected ROI is $n \sum_i p_i^2$.

We can see, that the bold strategy and the timid strategies ROI is a function of the probability distribution. We can thus judge potential profitability of these distributions. Later we will judge the cost of mistakes in estimating the probability distributions.

Let us try to find a strategy which will maximize our expected earnings in the long run, after several repetitions of the game. Our capital in $(k + 1)$-th game is what we obtained in the k-th. According to [Grü00], long-term capital is exponentially larger for a strategy whose $E_p[\ln r] = \sum_i p_i \ln r_i$ is maximized, in comparison with any other strategy, for a sufficiently large number of game repetitions. It is easy to see that $p = \arg\max_{\hat{r}} E_p[\ln \hat{r}]$. If we are unsure about p, merely knowing that $p \in \mathcal{P}$, we should pick the strategy r^* which maximizes

$$\min_{p \in \mathcal{P}} E_p[\ln r^*].$$

2.4.3 Probabilistic Evaluation

Let us now focus on evaluation of probabilistic classifiers. We evaluate the quality of a classifier with an *evaluation function* $q : (\mathcal{C}, \mathcal{W}) \to \mathbb{R}$, which provides an ideal evaluation of a classifier in some world, and whose value we want to maximize. In real life, we invoke the evaluation function on some instance set, usually the evaluation set $\mathcal{E}$, which we may stress by denoting the evaluation function as $q_{\mathcal{E}}$. In this definition, we have chosen to neglect the cost of learning, as well as the cost of obtaining instances, attribute values, and other pieces of information: these would only be subtracted from the above quality-as-reward. We will also leave out the world parameter, and refer to the evaluation function briefly as $q(d)$.

An evaluation function can be seen as a measure of returns actually obtained when using d to choose our actions in a decision theoretic framework. It is easy to perform such an evaluation of a classifier on unseen data, and most methods perform something similar. A classifier that outputs realistic probabilities rather than deterministic most-likely-class predictions will perform better as measured by the evaluation function in this situation.

We temporarily ignore any kind of comprehensiveness of the classifier: for that purpose we should use a wholly different evaluation function that rewards the amount of insight gained by the decider given the classifier.

2.4.4 Probability of a Probability

A *first order probability distribution* is $P(\Pr\{d(\mathbf{i}) = c_k\})$, $k = 1, \ldots, |\mathcal{D}_C|$. It is an estimate of the probability distribution of the zero-order approximation of posterior class probabilities by a stochastic classifier d. We can similarly define second-, third-, and so on, order probability distributions.

The zero-order estimate helped us pick the least costly outcome for a given cost matrix. A first order estimate would help us find the zero-order estimate which is on average least costly, or has another desirable property, like the minimum possible risk. We can work from the other direction and obtain the least costly cost matrix, in case the probability distribution of cost matrices is known.

Several methods resemble the one above: [DHS00] mentions minimax risk: from a set of permissible priors, we choose the prior for which the Bayes risk is maximum. MaxEnt [Grü98] selects in some sense the least costly zero-order probability distribution estimate given a certain range of acceptable zero-order estimates. Both methods avoid dealing with non-zero-order probability explicitly, implicitly assuming a first-order probability distribution estimate in which a certain subset of zero-order estimates are assumed to be equally likely.

Estimating first-order probability distributions is not hard conceptually: we learn from several samples of the training set, and examine the distribution of zero-order estimates. Examples of schemes for such estimates are cross-validation, and leave-one-out or jackknife.

We might relax our assumption that the training set is identically and independently sampled from the world. In such a case, we have to assume what else could be happening. For example, we can sample with replacement (bootstrap), pick samples of differing sizes, or introduce various kinds of counter-stratification in sampling. Such actions will probably increase the timidity of eventually obtained least costly zero- or higher-order probability estimates.

The sampling methods are time consuming, but sampling is not a necessary requirement. In estimating zero-order uncertainty we avoid resampling by fitting parametric probabilistic models directly to data. Similarly, if we accept the bias, we can use various maximum likelihood or maximum a posteriori parameter estimation methods with *higher-order probability distributions.*

Let us now provide a semi-formal coverage of the concept. We are already familiar with a 0-order PDF: $f_0^{\mathcal{X}} : \mathcal{X} \to [0,1]$, so that $f_0^{\mathcal{X}}(X) = P(X), X \in \mathcal{X}$. $\mathcal{X}$ is a set of mutually exclusive events, and we require probability of an event to be greater or equal to 0, $f_0^{\mathcal{X}}(X) \geq 0$, for all $X \in \mathcal{X}$. We also require that the probabilities for all events sum up to 1, $\sum_{X \in \mathcal{X}} f_0^{\mathcal{X}}(X) = 1$, in line with normalization and positivity conditions for density functions. Any discrete event E should be represented with a set $\mathcal{X}_E = \{E, \neg E\}$, to be appropriate for density functions.

A m-order PDF is $f_m^{\mathcal{X}} : \mathcal{X} \to \mathcal{F}_{m-1}^{\mathcal{X}}$. It maps an event into a $(m-1)$-order density function, and we keep these in $\mathcal{F}_{m-1}^{\mathcal{X}}$ for convenience. The intermediate k-order, $m > k > 0$, density function $f_k \in \mathcal{F}_k$ is a mere mediator: $f_k : [0,1] \to \mathcal{F}_{k-1}$. The final, 0-order density function maps to a real value interval $[0,1]$: $f_k : [0,1] \to [0,1]$, representing a concrete probability.

The 1-order density function for an event $E \in \mathcal{X}_E$ would thus map the base outcome into a density function f_0 which describes the density function of the probability itself. Thus $(f_1(E))(p) = \Pr\{P(E) = p\}$. The 2-order density function merely extends this to $((f_1(E))(p_1))(p_2) = \Pr\{\Pr\{P(E) = p_1\} = p_2\}$. We need not be intimidated: we are usually satisfied with 0-order probability functions. Now we have something more general. This system of high-order probability density functions can be seen as a possible formalization of the idea of imprecise probabilities.

2.4.5 Causes of Probability

We should distinguish *uncertainty*, *ignorance* and *unpredictability*. When we obtain a probability of an outcome, the probability may be there due to uncertainty, due to ignorance, or because of inherent unpredictability. Uncertainty is a subjective estimate, provided by a probabilistic classifier. Ignorance and unpredictability are objective properties. Ignorance is a consequence of limited information. Unpredictability is a consequence of inherent unpredictability and unknowability of the world, and a big philosophical dilemma: If we throw a dice, the outcome may either be due to inherent unpredictability of the dice, or it may be because of our ignorance of the weight, shape, position, speed, acceleration, etc., of the dice. To avoid dilemmas, we will only refer to ignorance while accepting that unpredictability may be a part of ignorance. All three can be represented with distribution functions and probabilities.

Our predicted probability of falling sick p can be seen as a measure of uncertainty: it is a measure of how serious our current health situation is. We have little way of knowing what will actually happen, and we are ignorant of all the details. Nevertheless, uncertainty is useful directly as information: far fewer unfavorable events are sufficient to push a $p = 0.6$ healthy person into disease than a $p = 0.9$ healthy person. Similarly, the expiration date on supermarket food signifies that from that date onwards the uncertainty that the food is spoiled is above, e.g., $p = 0.1$.

The objective of learning is to minimize probability due to uncertainty, taking advantage of attributes to reduce its ignorance. Noise is the ignorance that remains even after considering all the attributes. The lesser the uncertainty, the better the results with most decision problems: we earn more in betting games.

An ideal classifier's uncertainty will match its objective ignorance exactly. There are two dangers in learning, both are tied to subjective uncertainties mismatching objective ignorance: *overfitting* refers to underestimating the subjective uncertainty, below the actual level of objective ignorance; *underfitting* refers to a timid learner's uncertainty overestimating its objective ignorance.

2.4.6 No Free Lunch Theorem

If we assume nothing, we cannot learn [WM95, Wol96]. Not even timid predictions are acceptable: we have to be timid about our uncertainty: from all outcomes are equally likely, through all outcome distributions are equally likely, to all distributions of outcome distributions are equally likely, and so ad infinitum.

If learning is hard, it is hard for everyone. Nevertheless, some learners sometimes manage to learn better than others, and make better decisions. In an imperfect world, where only a number of possible situations occur, some learners will always be empirically better than others.

We could assume that a domain has a particular level of inherent ignorance, that it is generated by a particular model, and we could even assume that it is deterministic.

But we choose to only assume that we can generalize from a subset to the whole set: we investigate the *learning curves* to ascertain ourselves. In a learning curve, we plot the value of the evaluation function with respect to the proportion of data used to train the classifier. Should they not converge, we question our trust in the predictions, and represent the distrust with higher-order uncertainty. Lift charts and cumulative gains charts are largely synonymous to the learning curves.

2.5 Estimating Models

Earlier, we referred to a probabilistic learner indirectly: we studied the probability distribution of the stochastic classifier's predictions. Such an approach requires sampling at best, and is generally impractical. An alternative approach is to use models. *Models* are functions that map some representation of an instance $\mathbf{i}$ into the k-order probability density functions of the label given an instance $\mathbf{i}$. In this text, we only consider closed-form functions as models.

Models are a concrete form of Bayesian posteriors, and they can be used by a probabilistic classifier to provide predictions. Practical probabilistic classifiers should return density functions rather than behave stochastically. Probabilistic classifiers' predictions are density functions. For that reason we will denote the k-order density function, the codomain of model $\mathcal{H}_i^k$, as $f_k^{\mathcal{D}_C}(c) = M_k^{\mathcal{H}_i}(c|\mathbf{i}, \mathbf{w})$, where $c \in \mathcal{D}_C$ is a label value. As earlier, $\mathbf{i}$ is an instance whose class we want to know, and $\mathbf{w}$ are the parameters — results of fitting a model to the data. Our definition is intentionally lax, and not even a proper function, but we do not want to fix exact operations performed on data in the model definition itself.

Having the concept of a model, we can define a *probabilistic learner*, no longer having to resort to stochastic classifiers: $L : (\mathcal{P}, \mathcal{W}) \to \mathcal{C}$, where $\mathcal{W}$ is a universe of instance worlds ($\mathcal{I} \in \mathcal{W}$), $\mathcal{P}$ is a classification problem as defined in Sect. 2.2, and $\mathcal{C}$ is a set of probabilistic classifiers. $pc \in \mathcal{C}$ is a probabilistic classifier, $pc : \mathcal{I} \to \mathcal{F}^{\mathcal{D}_C}$, where $\mathcal{F}^{\mathcal{D}_C}$ is the world of label (C) distribution functions of some order.

Therefore, a probabilistic classifier is a map from an instance description to some probabilistic model. The probabilistic learner determines appropriate models, transforms the instance into parameters, estimates one or more models with respect to these parameters on the training data, and returns a label probability function. The details of these operations will be explored in the coming sections.

The first natural problem is adjusting model's parameters to fit data (estimation, or model fitting), and the second is choosing a model (model testing or model comparison). Estimation is the process of obtaining a concrete model or its parameters by estimating the label probability distribution. One can simply pick the most likely class, and assign it the probability of 1. With more sophistication, one can fit probability models via maximum likelihood or maximum a posteriori methods, or can perform less biased estimates via resampling, perhaps even modeling higher-order uncertainty.

We now focus on procedures for estimating models. Their task is to determine the parameters of a given parametric model function to fit the data.

2.5.1 Bayesian Estimation

Let us recall the Bayes rule:

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}} ,$$

where likelihood $= P(y|x)$, prior $= P(x)$, posterior $= P(x|y)$, and evidence $= P(y)$. Bayesian inference is built upon this rule and provides a framework for constructing models which provide zero-order probability estimates.

Should we assume a model $\mathcal{H}_i$ is true, we infer its parameters $\mathbf{w}$, given data D, by proceeding as follows:

$$P(\mathbf{w}|D, \mathcal{H}_i) = \frac{P(D|\mathbf{w}, \mathcal{H}_i)P(\mathbf{w}|\mathcal{H}_i)}{P(D|\mathcal{H}_i)} .$$

This step is called *model fitting*. The optimal $\mathbf{w}$ has maximum posterior, and can be obtained with gradient descent. The curvature of the posterior can be used to obtain the error bars of $\mathbf{w}$, but these error bars should not be mistaken for first-order posteriors. The *evidence* is often ignored, as it can merely be a normalizing factor. But MacKay [Mac91] names it 'evidence for model $\mathcal{H}_i$.'

2.5.2 Estimation by Sampling

Estimation can be seen as nothing other than frequency count gathering through sampling. From these counts we obtain probabilities, and use them in the classifier. We do not try to optimize anything in this process of estimation, as this would induce bias in our estimates. Reliability of estimates should be the burden of the learning algorithm, and estimation a module the learning algorithm uses like a black box.

With sampling, a model becomes nothing else than a non-parametric empirically-fitted distribution function, and we sample the data repeatedly to arrive to the distribution function from the gathered frequencies, as we described in Sect. 2.4.4. We assumed nothing, and our estimates of uncertainty are less biased. In reality, we will use several such models, joining their predictions. Furthermore, not all models are equally useful to the decision-maker: but this is a problem of learning, not of modeling. Our models may consequently be biased.

Model fitting by sampling is inefficient, but we can fit higher-order distribution functions quite easily. If we model higher-order uncertainty, we also avoid having to define ad hoc preferences for uncertainty distributions, as it is described in Sect. 2.4.5. Thus, if we successfully model uncertainty, we will be able to pick the optimal decision without having to burden ourselves with assuming detailed properties of risk and utility in the model itself.

With sampling we must especially mind *computational economy*: the benefit of better estimates may be obsoleted by changes through time, or may provide no gain once the costs of computation are subtracted. Finally, as we cannot possibly evaluate all models, we normally end up with imperfect ones. Many of these dangers should be managed by the decision-maker who has more information than the learner does. However, the learner should nevertheless be aware of uncertainty orders, computational economy, and be thorough with respect to supported models.

2.6 Classifier Evaluation

The learner should provide the best classifier it can create. We could choose the model functions randomly from some set, estimate their parameters, and if we had means of evaluating their worth, we would already have a working learning procedure.

There are two essential paradigms to classifier evaluation. First, we may assume that the model function is identical to some hypothetical 'function' that 'generated' the data. The second paradigm is pragmatic, and there we seek to maximize the benefit of the classifier's user, as measured by some evaluation function.

2.6.1 Generator Functions

We can assume that our knowledge and models are truly identical to ones that 'generate' data in real world and then estimate the probability — likelihood — that the particular sample was generated by a certain model. We effectively assume that $P(D|\mathcal{H}) = P(\mathcal{H}|D)$.

But the data might have been 'generated' by a set of 'functions'. We thus introduce expectations about this set of functions. For example, we may assign prior probabilities to individual models, which weigh the likelihood with the function prior. Or we may employ regularization, which penalizes certain parameter values: in the sense that certain parameters are more probable than others.

For Bayesian inference, MacKay estimates the *plausibility* of the model via

$$P(\mathcal{H}_i|D) \propto P(D|\mathcal{H}_i)P(\mathcal{H}_i).$$

We could either assume such evidence (subjective priors), or integrate evidence over all possible parameter values, thus penalizing the size of parameter space, in semblance of (but not equivalence to) VC dimension:

$$P(D|\mathcal{H}_i) = \int P(D|\mathbf{w}, \mathcal{H}_i)P(\mathbf{w}|\mathcal{H}_i)\mathrm{d}\mathbf{w},$$

or we could infer it from a number of diverse classification problems, just as the length of an expression in natural language is a result of natural language being applied to many real sentences in human life.

MacKay suggests the second possibility, and approximates it as evidence $\simeq$ best fit likelihood $\times$ *Ockham factor*:

$$P(D|\mathcal{H}_i) \simeq P(D|\mathbf{w}_{\mathrm{MP}}, \mathcal{H}_i)P(\mathbf{w}_{\mathrm{MP}}|\mathcal{H}_i)\Delta\mathbf{w},$$

where $\Delta\mathbf{w}$ is the posterior uncertainty in $\mathbf{w}$, a part of the Ockham factor. He suggests approximating it for a k-dimensional $\mathbf{w}$ with a Gaussian as:

$$\Delta\mathbf{w} \simeq (2\pi)^{k/2} \det{}^{-1/2}(-\Delta\Delta \log P(\mathbf{w}|D, \mathcal{H}_i)).$$

2.6.2 Evaluation Functions

Pragmatic learners instead try to find classifiers that would be evaluated as positively as possible by the decision-maker. The trouble is that we may achieve perfect performance on the training data, because it has already been seen. The evaluation methods should therefore examine the ability of the learner to generalize from a sample to the whole population. It is not easy, because the learner has no way of knowing how the data was collected, but it is equally hard for all algorithms.

A learning algorithm may try to explicitly maximize the value of some decision-theoretic evaluation function q, but it is only provided a sample of the instances and no insight about the decision-maker. Not only has it to approximate the instance label, it also has to approximate the decision-maker's evaluation function. We will not discuss this issue, and simply assume that an evaluation function q is given. Sect. 2.6.2 surveys a number of evaluation functions.

We employ two families of techniques. One group of approaches work with the seen data, use heuristic measures of classifier quality, and infer the the estimated classifier quality from the sample size and classifier complexity. With respect to the evaluation function, they also moderate the confidence of classifier's predictions with respect to the expected classifier performance.

Validation set methods simulate how a decision-maker would use a classifier: they evaluate the classifier on unseen data. Because all their data is essentially 'seen', they simulate leaving out a portion of the training data. Once they determine the properties of the classifier that worked best on portions of the training data, they train this winner with all the training data.

Evaluating Classifiers on Unseen Data

Because we cannot be given unseen data, we pretend that a part of the training data has not been seen. The training set $\mathcal{T}$ is thus split into a validation set $\mathcal{V} \subset \mathcal{T}$ and the remainder set $\mathcal{R} = \mathcal{T} \setminus \mathcal{V}$. The classifier is trained on the remainder set and evaluated on the validation set. Normally, multiple splits of the training set are performed (e.g., via 10-fold cross-validation) to obtain more reliable quality estimates.

The method is often called internal cross-validation, and we will refer to it as internal evaluation. *Wrapper methods* [Koh95] use the observations obtained in internal evaluation to adjust the parameters of the final classifier. The final classifier is is trained on the whole training set, but uses the same algorithm and the same parameters that were used in internal validation.

The core problem is that the method may be overly conservative since we generalize knowledge obtained from a portion of the training data to the whole training data. For example, if we are given exactly as many instances as there are needed to understand the domain, the splitting into the validation set will cause the estimates to underestimate the model, on average. On the other hand, we have no idea about how the model will be used, so extra fuss about this issue is unneeded.

The learning algorithm can, however, investigate the relationship between the label probability function with respect to differing size and differing sampling distribution inside the training data, in a meta-learning style. If the dependence does not disappear with larger and larger proportions of training data, meaning that the classifier does not appear to converge, the learning algorithm should wonder about whether there is enough data, and should increase its uncertainty estimates. With more sophistication, the dependence of uncertainty on the amount of training data can be estimated via learning curve extrapolation, as described in [Koh95, CJS$^+$94, Kad95]. Apart from resampling the training data, we can use background knowledge, which can result from meta-learning on a number of domains.

Once the amount of ignorance has been estimated, moderating procedures can be represented as an additional timid maximin gain-optimizing model. The final result of the classifier is obtained by voting between this timid model, weighted with the estimated amount of ignorance, and the trained reward-maximizing ignorance-minimizing model. Alternatively, we can skip the ignorance estimation stage, and represent this dependence in form of higher-order uncertainty, obtained by estimating the models on a number of instance samples drawn from the training set, as described in Sect. 2.4.4.

Evaluation Functions Surveyed

Because we do not know the true label probability distribution for a given instance, we pretend that the test instance's class is always deterministic, even if there are multiple class values for the same set of attribute values. For an instance $\mathbf{i}$, $\hat{P}$ is our approximation to the true probability distribution, and is defined as

$$\hat{P}(C(\mathbf{i}) = c) := \begin{cases} 1 & \text{if } C(\mathbf{i}) = c, \\ 0 & \text{if } C(\mathbf{i}) \neq c. \end{cases} \tag{2.1}$$

Gambling Assume that each test instance is a short-term betting game. Our capital's growth rate in a game on an instance $\mathbf{i}$ with deterministic class $C(\mathbf{i})$ is $\Pr\{d(\mathbf{i}) = C(\mathbf{i})\}$. If the class is also probabilistic, the growth rate is $\sum_{c \in \mathcal{D}_C} P(c|\mathbf{i})\Pr\{d(\mathbf{i}) = c\}$. The ideal classifier should try to maximize this growth rate. *Classification accuracy* or 0/1-loss is a special case of this measure when both the class and the classifier are deterministic. We can modify our heuristic if we have more information about the utility, or risk-averse preferences.

Description Length Data compression can be seen as an exercise in gambling. The decompression module predicts the distribution of symbols. When a symbol comes, it is compressed using the predicted distribution. If we use optimal arithmetic entropy coding, the symbol c_i will consume exactly $\log_2 \Pr\{d(\mathbf{i}) = c_i\}$ bits, and this is *description length*, the quantity we want to minimize. In data compression, there are further requirements: the decompressor originally has no information about the problem domain, or the classifier. We can thus either transfer the model beforehand (and incur a loss because of transferring the model), or we can transfer instances one by the other, having the decompressor infer the model. Although there are some similarities, especially with respect to general undesirability of overly confident predictions, data compression is not decision-making.

Relative Entropy In similarity to the reasoning from [Grü98], a classifier d is optimal when it minimizes *relative entropy* or *Kullback-Leibler divergence* [KL51], also known as KL distance, relative entropy, and cross entropy. KL divergence is measured between two probability distributions, the *actual* distribution of label $P = P(C(\mathbf{i})|\mathbf{i})$ and the *predicted* distribution of label $Q = \Pr\{d(\mathbf{i})\}$:

$$D(P||Q) := \sum_{c \in \mathcal{D}_C} P(C(\mathbf{i}) = c) \log \frac{P(C(\mathbf{i}) = c)}{\Pr\{d(\mathbf{i}) = c\}}. \tag{2.2}$$

Relative entropy is a heuristic that rewards both correctness and admittance of ignorance. It can also be used to choose a posterior distribution Q given a prior distribution P. KL divergence can be understood as an increase in description length incurred by the imperfect probability distribution estimate in comparison with the description length obtained by the actual probability distribution.

The logarithm of the probability can be seen as a logarithmic utility function. It was already Daniel Bernoulli who observed diminishing marginal utility in human decision-making, and proposed logarithmic utility as a model in 1738 [Ber38, FU]. In concrete terms, he observed that happiness a person increases only logarithmically with his earnings. Entropy can be seen as a particular utility function.

Because the real P is often unknown, we have to approximate it for each instance, for example with $\hat{P}$ (2.1). If there are n instances whose classes are distributed with P, and we try to sum the divergence for all instances as an output of our evaluation function, the result will not be $nD(P||Q)$. If there are k outcomes ($|\mathcal{D}_C| = k$), we will instead obtain have $nD(\hat{P}||Q) = \sum_{i=1}^{k} nP(c_i)D\left[P(c_i) = 1||Q\right]$. The deviation is $nD(P||Q) - nD(\hat{P}||Q) = n\sum_{i=1}^{k} P(c_i)\log P(c_i) = -nH(P)$. It is fortunate that this deviation is independent of Q, so a comparison between different classifiers on the same data will be fair.

Testing Goodness of Fit As a side note, examine the expression for Pearson's X^2 *statistic* from 1900, which compares an observed distribution with an empirical one, performed on N instances $\mathcal{V} = \{\mathbf{i}_1, \mathbf{i}_2, \ldots, \mathbf{i}_N\}$ [Agr90], rewritten with our symbols:

$$X^2 = N\sum_{i=1}^{N}\sum_{c\in\mathcal{D}_C}\frac{(P(C(\mathbf{i}_i) = c) - \Pr\{d(\mathbf{i}_i) = c\})^2}{\Pr\{d(\mathbf{i}_i) = c\}}.$$

It has approximately a χ^2 null distribution with degrees of freedom equal to $(N-1)(|\mathcal{D}_C|-1)$. Also compare KL divergence with Wilks's *likelihood ratio statistic* from 1935:

$$G^2 = 2N\sum_{i=1}^{N}\sum_{c\in\mathcal{D}_C} P(C(\mathbf{i}_i) = c)\log\frac{P(C(\mathbf{i}_i) = c)}{\Pr\{d(\mathbf{i}_i) = c\}}.$$

It is also distributed with χ^2 distribution with $df = (N-1)(|\mathcal{D}_C|-1)$. This way, we can also use KL divergence as the basic statistic for determining the significance of model differences.

X^2 usually converges more quickly than G^2, as G^2 fits poorly when $N/df < 5$, whereas X^2 can be decent even when $N/df > 1$, if frequencies are not very high or very low. The original form of both statistics refers to sample counts, not to probabilities. To obtain the count formula, remove the N multiplier from the beginning of both expressions, and use class- and instance-dependent counts everywhere else instead of probabilities.

There are several other goodness-of-fit tests that we did not mention. For example, Kolmogorov-Smirnov and Anderson-Darling tests are well-known, but are primarily used for comparing continuous distributions.

Information Score It has been suggested in [KB91] that an evaluation of a classifier should compare it to the timid classifier d_t, which ignores the attributes and offers merely the label probability distribution as its model: $\Pr\{d_t(\mathbf{i}) = c\} = P(c)$. A more complex classifier is only useful when it is better than the timid learner. If it is worse than the timid learner, its score should be negative. For a given instance $\mathbf{i}$ of deterministic class $C(\mathbf{i})$, the *information score* IS of a classifier d is defined as:

$$IS := \begin{cases} \log_2 \Pr\{d(\mathbf{i}) = C(\mathbf{i})\} - \log_2 P(C(\mathbf{i})) & \Pr\{d(\mathbf{i}) = C(\mathbf{i})\} \geq P(C(\mathbf{i})), \\ -\log_2 (1 - \Pr\{d(\mathbf{i}) = C(\mathbf{i})\}) + \log_2 (1 - P(C(\mathbf{i}))) & \Pr\{d(\mathbf{i}) = C(\mathbf{i})\} < P(C(\mathbf{i})). \end{cases} \tag{2.3}$$

Information score is closely related to Kullback-Leibler divergence. We are subtracting the divergence achieved by the classifier d from divergence achieved by the timid classifier d_t. If the classifier d is worse than the timid classifier, the score is negative. Kullback-Leibler divergence strongly penalizes underestimation of ignorance, unlike information score. Consequently, it may happen that a classifier that correctly estimates its ignorance will obtain a lower score than a classifier that incorrectly estimates its ignorance.

Regret Regret or *opportunity loss* is a concept used for evaluating decision-making in economics [LMM02]. It is the difference between what a decision-maker could have made, had the true class probability distribution and the true cost matrix been known, and what he actually made using the approximations. We take the optimal decision with respect to the classifier's probabilistic prediction, and study the opportunity loss caused by the classifier's ignorance:

$$\begin{aligned} L_o(d, \mathbf{i}) = \left(\sum_{c \in \mathcal{D}_C} P(C(\mathbf{i}) = c) \mathsf{M} \left[\left(\arg \min_{\tilde{c} \in \mathcal{D}_C} \sum_{\hat{c} \in \mathcal{D}_C} \Pr\{d(\mathbf{i}) = \hat{c}\} \hat{\mathsf{M}}(\tilde{c}, \hat{c}) \right), c \right] \right) - \\ \min_{\tilde{c} \in \mathcal{D}_C} \sum_{c \in \mathcal{D}_C} P(C(\mathbf{i}) = c) \mathsf{M}(\tilde{c}, c). \end{aligned} \tag{2.4}$$

Here the $\hat{\mathsf{M}}$ is our approximation to the cost matrix which we use to make a decision. It is not necessarily equal to the true (but possibly unknown) cost matrix. The formula appears complex partly because we assumed intrinsic ignorance about the class: we compute the expected loss over all possible outcomes. It would be easy to formulate the above expressions for minimax loss, or with utility or payoff instead of cost.

In case there is no unpredictability, merely ignorance, we can introduce the concept of the *expected value of perfect information* or EVPI:

$$\text{EVPI} = \mathsf{M}(d(\mathbf{i}), C(\mathbf{i})) - \mathsf{M}(C(\mathbf{i}), C(\mathbf{i})),$$

where $d(\mathbf{i})$ represents the optimal decision we have made with the available information and the available cost matrix, while $C(\mathbf{i})$ is the actual outcome.

Receiver Operating Characteristic Receiver Operating Characteristic (ROC) analysis [PF97] was originally intended to be a tool for analyzing the trade-off between hit rate and false alarm rate. ROC graphs are visualizations of classifier performance at different misclassification cost settings. Of course, if we have an universal probabilistic learner, we are not forced to re-train the classifier for each setting, although we have shown that this could pay off. ROC can be used to switch between multiple classifiers, depending on the desired cost function, thus achieving a characteristic approximately represented by the convex hull over all the classifiers' characteristics. Choosing a classifier using ROC can be viewed as a multiobjective optimization problem: we only retain Pareto optimal classifiers, and dispose of the others. For example, if one classifier's ROC is fully below another ROC, we can say that it is dominated. We have no need for dominated classifiers. For a k-class classification problem the ROC visualization is k-dimensional. Area under the ROC (aROC) is a univariate quantitative measure used for comparing classifiers, with the assumption of uniform distribution of cost functions.

Evaluating Classifiers on Seen Data

Two essential elements of training classifiers have been always kept in mind: bold pursuit of confident predictions, and timid evasion from overfitting. The former is achieved by impurity-minimizing partitioning or merging, and model-fitting. The latter is embodied in pruning, model priors, penalized complexity, cross-validation, hypothesis testing, regularization, moderation. There are many expressions for this notion, timidity = avoiding bias, boldness = avoiding variance; timidity = avoiding overfitting, boldness = avoiding underfitting; timidity = minimax profit, boldness = max profit; timidity = maximum uncertainty, boldness = minimum uncertainty.

Should we evaluate the models merely on training data, we have to balance boldness and timidity. Here we present evaluation functions that operate merely with uncertainties, without knowledge of the exact utility function. But we do have knowledge of the decider's strategy, as it can either be long-term or short-term.

It is easy to modify the measures below to account for utility: an error in estimating the probability of the action that will be chosen by the decider has greater importance than error in the probabilities of other outcomes. To phrase it more precisely, errors in those probabilities which are unlikely to change the decision are less important than those errors that would cause the decision to deviate from the optimal. Awareness of this somewhat justifies cost-based learning. However, our uncertainty-based approach reduces the problem in comparison with other measures such as classification accuracy that become very brittle with class-unbalanced data.

Boldness We might not want to expose the exact utility function and parameters of the decision strategy (e.g., the degree of risk aversion) to the learner. Instead, we might want to merely operate with uncertainties while learning, according to our above definition of L_P. We are encouraged by the conclusions of Sect. 2.4.2: the lower the measure of uncertainty, the greater the maximum returns.

Timidity However, sometimes we require the opposite - a classifier that would be least risky: We can use entropy in the following way [Jay88]: should there be several permissible probability functions, we should pick one that has the maximum entropy. This is the 'safest' probability distribution. *Entropy* [Sha48] is a measure of information content of an information source, and is defined as

$$H(\Pr\{d(\mathbf{i})\}) = -\sum_{\hat{c}\in\mathcal{D}_C} \Pr\{d(\mathbf{i} = \hat{c})\} \log \Pr\{d(\mathbf{i}) = \hat{c}\}. \tag{2.5}$$

It can be measured in bits, when 2 is the logarithmic base, or in nats when a natural logarithm is used.

When we are playing to maximize returns over the long run, we prefer entropy-like measures. When we play to maximize returns in a single run, we are interested in maximizing the probability of the most likely outcome.

The *MaxEnt principle* is a generalization of Laplace's prior. Of course, MaxEnt is merely a heuristic that embodies the preference for the most timid of the available equally likely distributions. It is interesting to observe the contrast between the MaxEnt heuristic, and that of seeking classifiers that yield minimum entropy (let us call them MinEnt).

MinEnt is the process of seeking most profitable classifiers, while MaxEnt is a procedure of seeking classifiers that minimize loss in long-term gambling. For short-term gambling, entropy is no longer desirable: if we were given several acceptable models, we would pick the model that maximizes $\max_{c \in \mathcal{D}_C} (1 - \Pr\{d(\mathbf{i} = c)\})$ instead of one that would maximize entropy.

2.7 Constructing Classifiers

Models are a powerful mechanism for managing uncertainty, and we know how to estimate them. We also know how to choose classifiers, but we do not yet know how to bridge the gap between the two. In this section, we will discuss how we construct classifiers from a set of robust model functions, and a set of utility functions that link different models.

It is futile for the learner to continuously serve unprofitable models. As general functions are a gigantic model space, we should try to be more specific. The learner's true objective is to provide models which get chosen by the decision-maker, both for their prediction performance as well as for their computational efficiency.

It is obvious that this creates a market of learners, where it is hard to point a finger on the best one. There are many niche players. Still, we can investigate the optimization process of the learner intelligently and actively trying to maximize its odds of getting chosen by the decision maker, rather than stupidly using and fitting the same model over and over again. We apply two concepts from good old-fashioned artificial intelligence: heuristics and search. Search helps us try several models, before we pick the best one, and heuristics help us efficiently search for the best one.

We thus seek useful families of models, those that are more likely to be chosen as optimal by the decision-maker, and only eventually those that are less likely to optimal. Should we interrupt the execution of the learner at some point, it would be ideal if it could immediately present a good model: a decision-maker interested in efficiency could use this interruption mechanism to maximize its computational economy criterion.

In this section, we only focus on well-known families of models that have proven to work well. In constructing classifiers, there are four most important building blocks: we have already discussed estimation of non-parametric and parametric models in Sect. 2.5, and now we present construction of descriptors from instances, segmentation of instances, and voting between models. By manipulating these blocks, we quickly build useful classifiers.

2.7.1 Building Blocks

Remember that the engine of uncertainty estimates are models. Models map instance *descriptors* to uncertainty estimates. Although an instance's attributes could be its descriptors as a model's domain, this is neither necessary nor desirable. The descriptors should be simpler than attributes. We prefer to apply multiple simple models rather than a single complex and unwieldy one with many descriptors.

The n descriptors to a model span n dimensions. High *dimensionality* is a tough problem. People constantly try to diminish the number of descriptors to the minimum feasible amount. Our ability of visualizing data is too limited by its dimensionality. Perception is two-dimensional, while imagination has difficulty even with full three dimensions. Our visual minds have evolved to solve problems in colorful two-and-a-half dimensional landscapes. We are able to reach higher only by finding intelligent ways of projecting problems to two dimensions. Although computers cope with more dimensions than people, the phrase *curse of dimensionality* implies that computers too get burdened by extra dimensions.

Most learning methods try to diminish the dimensionality of the problem by reducing the number of descriptors in a probabilistic model. The first phase in learning converts attributes into descriptors, and it precedes model estimation. Descriptors may be continuous numbers (projections), or subsets of the training set of instances (segmentation). There can be several models, each with its own set of descriptors and its own set of training instances. There is no curse of dimensionality if we treat the descriptors one by one, it only occurs when multiple descriptors are to be treated at once. In the simplest model, there are zero descriptors, and we may use multiple models.

In the second phase, we use estimation methods, which we already discussed in Sect. 2.5, to estimate the class distribution function given the descriptor values. For an instance, we only know its class and its descriptor values.

If multiple models were created, their predictions are joined. Sometimes models do not overlap: for a given instance only a single model is used, and its output is the output of the whole classifier. If multiple models provided predictions for the same instance, but with different descriptors, we may apply *voting* to unify those predictions. Voting is not necessarily a trivial issue. If one of the models is duplicated, its predictions would carry twice the previous weight.

There are several trade-offs with respect to these methods, and they emerge in the process of estimation. The severity of the curse of dimensionality rises exponentially with the number of descriptors used in estimating one model. Besides, the more descriptors we use, the greater the sparseness of the probability function, and the lower the effectiveness of estimation procedures. The more instances we use to estimate each model, the more reliable and representative are its estimates, but the lower is the reduction of our ignorance. Voting between overlapping models helps us address the above two trade-offs, but voting itself carries certain problems which will be discussed later.

We will now examine each of these three techniques in more detail.

Projection

A *projection* involves acquiring a small number of numeric descriptors $\hat{\mathbf{x}}$ for a given instance, by transforming attribute values. For any $\mathbf{i} \in \mathcal{I}$, we can compute the m-descriptor projection as $\hat{\mathbf{x}}_\mathbf{i} = W(\mathbf{i}), W : \mathcal{I} \to \mathbb{R}^m$. These models' codomains are the descriptor spaces. Descriptors are computed from an instance's attributes.

The training procedure should attempt to find the most informative descriptor spaces, while keeping them low-dimensional. A hyperplane in n dimensions is a linear projection of the whole attribute space onto a single informative descriptor dimension. The sole descriptor of an instance is its distance to the hyperplane.

The abstract notion of distance descriptor is converted to a zero-order probability distribution with a link function, for example a step/threshold/Heaviside function (linear discriminant), or with a logit link (logistic regression). As there are several possible hyperplanes, the choice of one is determined by desirable properties of the probability distribution's fit to the training data in logistic regression. This should be contrasted to Vapnik's maximum-margin criterion [Vap99] for linear discriminants. Instance-based learning and kernel-based learning can be seen as families of models where the descriptors are distance functions between pairs of instances.

Segmentation

Segmentation is the process of dividing the training data into multiple segments. These segments partition the whole instance world $\mathcal{I}$, so for any $\mathbf{i}$, even if previously unseen, we can determine the segment it belongs to. Segmentation can be seen as a special case of projection, where the descriptor is a discrete number identifying the segment of an instance. The set of segments is a finite set $\mathcal{S}$: $W_S : \mathcal{I} \to \mathcal{S}$, and we use a single zero-descriptor model for each segment. If we apply segmentation to descriptors themselves, we implement non-parametric estimation of models with continuous descriptors.

A timid learner, which considers no attributes, can be seen as using a single 0-order 0-descriptor model with a single segment containing all the instances. An almost natural approach to segmentation is present in the naïve Bayesian classifier: every attribute value is a separate segment with its own zero-descriptor zero-order probability model.

In classification trees and rules we recursively partition the data with hyperplanes, which are usually orthogonal or axis-aligned, obtaining a set of data segments. For each segment of the data, a zero-order zero-descriptor probability model is estimated. For a new instance, we can determine the segment to which the instance belongs, and that model's distribution function is offered as output of the classifier. We can imagine a classification tree as a 'gate' which, for a given instance, selects the prediction offered by the single predictor, chosen as the most informative depending on the instance's attribute values. The choice of the predictor depends on the attribute values.

Voting

When we end up with multiple constituent models for a given instance, we need to unify their proposed distribution functions in a single distribution function. A simple example is the naïve Bayesian classifier. We can imagine it as a set of zero-order zero-descriptor models, one for each attribute-value pair. The models corresponding to attribute values present in the instance are joined simply by multiplication of individual models' distribution functions: a fixed formula which only functions properly when the constituent models are independent.

Learning can be applied to the problem of voting, and this often takes the form of estimating the final model on the basis of descriptors or segments derived from constituent models' distribution functions. In a simple case, if we choose to use the product of distribution functions as the result of voting, we can weigh each distribution function. This alleviates the problem of model duplication, but it is uncertain how well it functions with other artifacts.

A lot of work has been done in the domain of ensemble learning [Die00]. An ensemble is a collection of separate classifiers, whose predictions are eventually joined by voting. Specific examples of ensemble learning methods are Bayesian voting, bagging, boosting, and others. In the above examples, the views were always disjunct, but it has been observed that relaxing this requirement improves results.

CHAPTER 3

Review

Mathematics is cheap: all it takes is paper, a pencil and a dustbin.
But philosophy is cheaper, you need no dustbin.

In this chapter, we provide an overview of topics from a multitude of fields that cover the issue of interactions. A reader should merely skim through these sections, and perhaps return later when seeking explanation of a certain concept. We do not attempt to define interactions, as there are many different definitions. It will be helpful to the reader to imagine interactions in the literal sense of the word, and perceive the differences between approaches. Our objective will be fulfilled if the reader will notice the great variety of human endeavors in which the concept of interactions appears.

3.1 Causality

To understand what abstract interactions could represent in real world, we might approach the issue from the viewpoint of *causality*. According to [JTW90], there are six types of relationships, illustrated on Fig. 3.1, that can occur in a causal model:

A direct causal relationship is one in which a variable, X, is a direct cause of another variable, Y.

An indirect causal relationship is one in which X exerts a causal impact on Y, but only through its impact on a third variable Z.

A spurious relationship is one in which X and Y are related, but only because of a common cause Z. There is no formal causal link between X and Y.

A bidirectional or **a reciprocal** causal relationship is one in which X has a causal influence on Y, which, in turn, has a causal impact on X.

An unanalyzed relationship is one in which X and Y are related, but the source of relationship is not specified.

A moderated causal relationship is one in which the relationship between X and Y is moderated by a third variable, Z. In other words, the nature of the relationship between X and Y varies, depending on the value of Z. We can say that X, Y and Z interact.

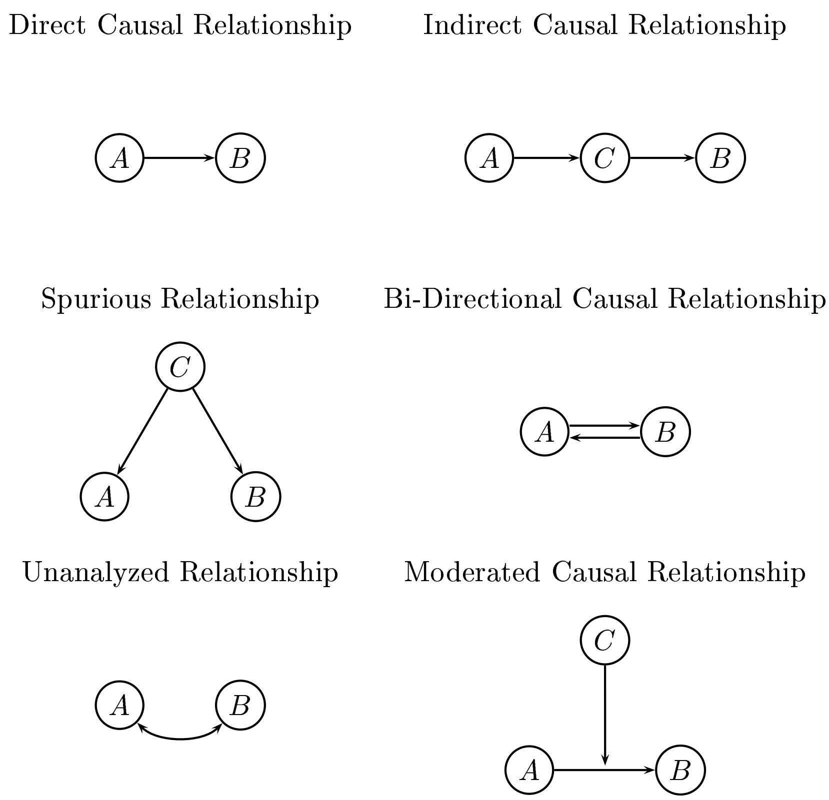


Figure 3.1: Six Types of Relationships

The moderated causal relationship is the one usually associated with interactions. It can be sometimes difficult to discern between the *moderator* and the *cause*. In moderated causal relationships, it is stressed that there is some interaction between attributes. There can be multiple dependent causes, yet it is not necessary that they are interacting with one another.

When the effects of pair of variables cannot be determined, we refer to them as *confounded* variables. Interacting variables cause unresolvable confounding effects. Moderation can be seen as a way of resolving confounding, although it can be ambiguous which of the variables is moderating and which is moderated.

3.2 Dependence and Independence

Our investigation of interactions may only proceed after we have fully understood the concepts of *association* and *dependence*. Association is a concept from categorical data analysis, while dependence is used in probability theory. The two concepts are largely synonymous. In probability theory, two events are independent iff the probability of their co-occurrence is $P(X,Y) = P(X)P(Y)$.

In categorical data analysis, to study attributes A and B, we introduce cross-tabulation of frequencies in a two-way *contingency table*.

E	a_1	a_2
b_1	5	10
b_2	12	8

For continuous numerical attributes, a *scatter plot* is most similar to a contingency table. We could compute approximate probabilities from contingency tables, dividing the field count by the total table count, but we do not do that in categorical data analysis: we always try deal only with frequencies.

The frequencies are obtained by counting instances in the training set that have a given pair of attribute values. For example, there are 5 instances in $\mathcal{S} = \{\mathbf{i} \in \mathcal{U} : A(\mathbf{i}) = a_1 \wedge B(\mathbf{i}) = b_1\}$, where $\mathcal{U}$ is the training set. A n-way contingency table is a multidimensional equivalent of the above, for n attributes. A 1-way contingency table is a simple frequency distribution of attribute values.

We may wonder whether two attributes are associated. Generally, they are not associated if we can predict the count in the two-way table from the two 1-way contingency tables for both attributes. The obvious and general approach is to investigate the dependence or independence of individual attribute value pairs as events: the corresponding measures of association for 2×2 tables are the odds ratio and relative risks. It works both for nominal and ordinal attributes.

Should we desire a test that will either confirm or refute whether two attributes are associated with a certain probability, we can use χ^2 statistic as a measure of *pairwise association* between attributes A and B:

$$E_{i,j} = \frac{N_{a_i} N_{b_j}}{N},$$

$$Q_P = \sum_{i \in \mathcal{D}_A, j \in \mathcal{D}_B} \frac{(N_{a_i,b_j} - E_{i,j})^2}{E_{i,j}}.$$

Here $E_{i,j}$ is the predicted count. For associations of multiple attributes, a similar formula is used. N is the number of instances, and $N_{\text{condition}}$ the number of instances fulfilling a particular condition.

We can invoke the χ^2 statistical test to verify association at the specific level of significance. The variables are associated if Q_P exceeds the tabulated value of χ^2 at the given level of significance with the degrees of freedom equal to $df = (|\mathcal{D}_A| - 1)(|\mathcal{D}_B| - 1)$. The inner workings of this test are based on checking the goodness of fit of the multiplicative prediction on the basis of marginal frequencies $E_{i,*}$ and $E_{*,b}$ to the actual $E_{i,j}$.

We could also use the continuity-adjusted χ^2 test, and the likelihood-ratio χ^2 test. Fisher's exact test can be used for 2×2 tables, while its generalization, Freeman-Halton statistic, has no such limitation and can be used for general $R \times C$ tables.

For a pair of numerical attributes, we might investigate the correlation coefficient between the two attributes. For associations between numerical and nominal attributes, we could use ANOVA.

There are several measures and tests of association intended specifically for pairs of ordinal attributes: Pearson's correlation coefficient, Spearman's rank correlation coefficient, Kendall's tau-b, Stuart's tau-c, Somers' $D(C|R)$ and $D(R|C)$, Mantel-Haenszel chi-square statistic, Cochran-Armitage trend test (for a pair of a bi-valued and an ordinal attribute), but we will focus solely on nominal attributes in this text. An interested reader may refer to [SAS98].

Measures of Association

Sometimes a binary decision of whether a pair of attributes are dependent or independent is insufficient. We may be interested in the *measure of association*, an example of which is the contingency coefficient:

$$P = \sqrt{\frac{Q_P}{Q_P + \min(|\mathcal{D}_A|, |\mathcal{D}_B|)}}.$$

The contingency coefficient P is equal to 0 when there is no association between variables. It is always less than 1, even when the variables are totally associated. It must be noted that the value of P depends on the size of value codomains for attributes, which complicates comparisons of associations between different attribute pairs. Cramer's V solves some of the problems of P for tables other than 2×2. Another modification of P is phi coefficient.

Other measures of association are gamma (based on concordant and discordant pairs of observations), asymmetric lambda $\lambda(C|R)$ (based on improvement in predicting column variable given knowledge of the row variable), symmetric lambda (average of both asymmetric $\lambda(C|R)$ and $\lambda(R|C)$), uncertainty coefficient $U(C|R)$ (proportion of entropy in column variable explained by the row variable), and uncertainty coefficient U (average of both uncertainty coefficients $U(C|R)$ and $U(R|C)$).

Furthermore, we can apply tests and *measures of agreement*, such as the McNemar's test, which specialize on 2×2 and multi-way $2 \times 2 \times \ldots$ tables. Cochran-Mantel-Haenszel statistic can be used for analyzing the relationship between a pair of attributes while controlling for the third attribute, but it becomes unreliable when the associations between the tested pair of attributes are of differing directions at different values of the controlled attribute, e.g., $\theta_{XY(1)} < 1$ and $\theta_{XY(2)} > 1$. Breslow-Day statistic is intended for testing homogeneous association in $2 \times 2 \times k$ tables, but cannot be generalized to arbitrary 3-way tables, and does not work well with small samples.

3.2.1 Marginal and Conditional Association

We will now focus on situations with three attributes. We can convert three-way contingency tables into ordinary two-way ones in by picking two *bound* attributes to represent the two dimensions in table, while the remaining attribute is considered to be *conditional*.

A *marginal table* results from averaging or summing over the uninvolved *free* attribute, we can imagine it as a projection onto the two bound attributes. On the other hand, a *conditional table*, sometimes also called a partial table, is a two-attribute cross-section where the condition attribute is kept at a constant value. In classification, the label will be the condition attribute, unless noted otherwise.

Table 3.1: Simpson's paradox: looking at location alone, without controlling for race, will give us results which are opposite to the actual.

marginal			
location	lived	died	p_{death}
New York	4758005	8878	0.19%
Richmond	127396	286	0.22%

white				non-white			
loc.	lived	died	p_{death}	loc.	lived	died	p_{death}
NY	4666809	8365	0.18%	NY	91196	513	0.56%
Rich.	80764	131	0.16%	Rich.	46578	155	0.33%

A pair of attributes A, B is *conditionally independent* with respect to the third condition attribute C if A and B are (marginally) independent at all values of C.

Conditional and marginal association are not necessarily correlated. In fact, there are several possibilities [And02]:

Marginal	Conditional	Comment
independence	independence	not interesting
independence	dependence	conditional dependence
dependence	independence	conditional independence
dependence	dependence	conditional dependence

Conditionally independent attributes are suitable for sub-problem decomposition and latent variable analysis (variable clustering, factor analysis, latent class analysis). Essentially, the attributes are correlated, but tell us nothing about the class. The only conclusion we can make is that some groups of attributes have something in common.

Conditional dependence with marginal independence is interesting, as the XOR problem ($C = A$ XOR B) is one such example. Myopic attribute selection would dispose of the attributes A and B.

The most complex is the fourth scenario: simultaneous marginal and conditional associations. There are several well-known possibilities:

Simpson's Paradox occurs when the marginal association is in the opposite direction to conditional association. An example from [FF99] notes the following tuberculosis example, with attributes P and R:

$$\begin{aligned} \mathcal{D}_P &= \{\mathrm{NewYork}, \mathrm{Richmond}\}, \\ \mathcal{D}_R &= \{\mathrm{white}, \mathrm{non-white}\}. \end{aligned} \tag{3.1}$$

The label identifies whether an inhabitant died of tuberculosis. We only consider the probability for death in Table 3.1. From Fig. 3.2, we see that by considering location alone, it would seem that New York health care is better. But if we also control for the influence of skin color, Richmond health care is better.

Homogeneous Association describes the situation with attributes A, B, and C, where the measure of association between any given pair of them is constant at all values

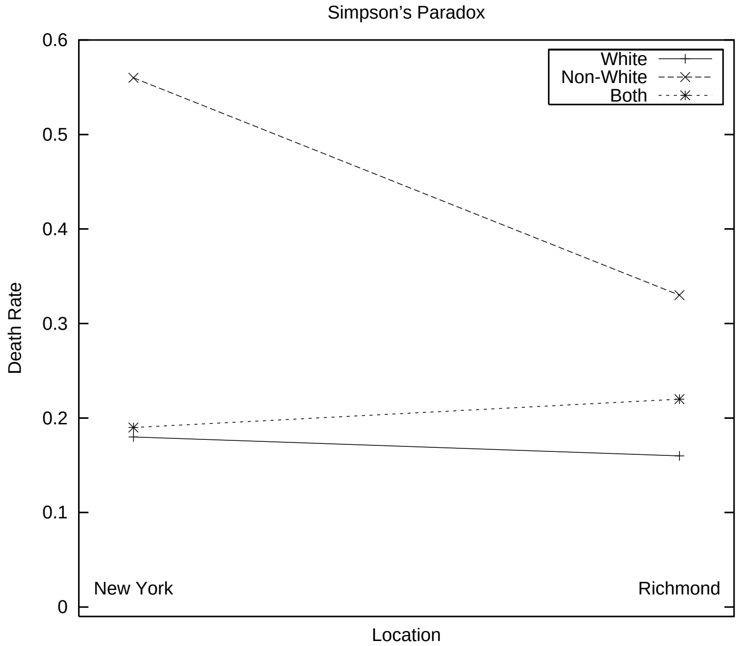


Figure 3.2: A graphical depiction of Simpson's paradox. The lines' gradient describes the relationship between location and death rate.

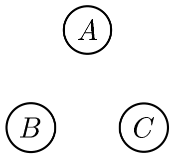
of the remaining one. From this we can conclude that there is no 2-way interaction between any pair of attributes, and that there is no 3-way interaction among the triple.

3.2.2 Graphical Models

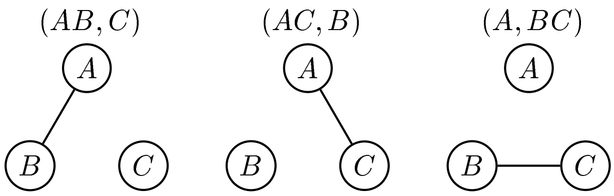
As in the previous section, we will base our discussion on [Agr90, And02]. Assume three attributes A, B, C. We want to visually present their marginal associations. If there is a marginal association between attributes A and B, but C is independent of both, we describe this symbolically with (AB, C). A graphical depiction is based on assigning a vertex to each attribute, and an edge to the possibility of each marginal association between the two vertices. Let us now survey all possibilities on three attributes:

Complete Independence: There are no interactions, everything is independent of everything else.

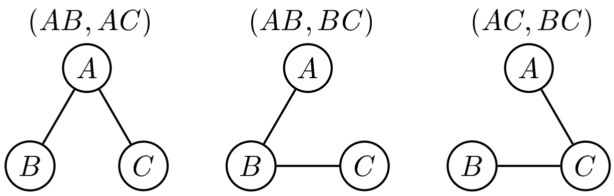
(A, B, C)



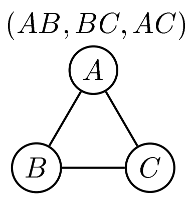
Joint Independence: Two variables are jointly independent of the third variable. In (AB, C), A and B are jointly independent of C, but A and B are associated.



Conditional Independence: Two variables are conditionally independent given the third variable. In (AB, AC), B and C are conditionally independent given C, although both A and B, and A and C are mutually associated.



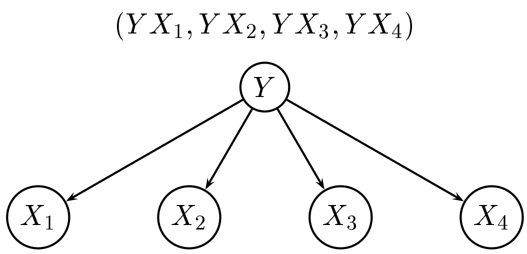
Homogeneous Association: Every pair of three variables is associated, but the association does not vary with respect to the value of the remaining variable.



But what about a 3-way association model, (ABC)? In practice, the homogeneous (AB, BC, AC) model is the one not illustrated, as a clique of n-nodes indicates a n-way association. Are these graphical models satisfying with respect to the difference between marginal and conditional association? For example, how do we represent an instance of Simpson's paradox with such a graph?

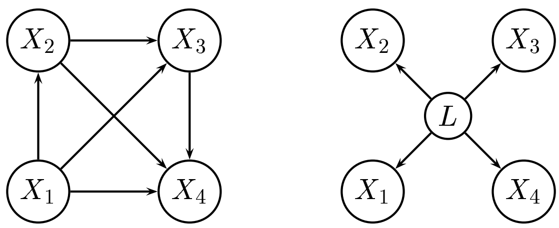
3.2.3 Bayesian Networks

In *Bayesian networks* [Pea88], the edges are directed. There are further requirements: the networks may have no cycles, for example. For a given vertex A, we compute the probability distribution of A's values using the probabilities of parent vertices in $\mathcal{P}_A = \{X : (X, A) \in E\}$, where E is the set of directed edges in the network. The Bayesian network model corresponding to a naïve Bayesian classifier for attributes X_1, X_2, X_3, X_4 and label Y would be as follows:



In Bayesian networks, an edge between two vertices is a marginal association. If we are learning Bayesian networks from data, we usually try to simplify the graphical model by eliminating a direct edge between A and B if there exists another path between the two vertices which explains the direct association away. The essence of learning here is pursuit of independence.

If a vertex A has several parents $\mathcal{P}_A$, we will assume that these parents are conditionally associated with A. Thus, the probability of each value of A is computed from a conditional n-way contingency table, where $n = |\mathcal{P}_A|$, the condition being that an instance must have that particular value of A to be included in a frequency computation. As n-way contingency tables may be very sparse, classification trees and rules are used to model frequencies and probabilities, for example: $P(A = \text{yes}|X_1 = \text{no}, X_2 = \text{no}, X_3 = *, X_4 = *)$. Latent variables, also referred to as hidden or unobserved, may be introduced to remove complex cliques or near-cliques from the network:



before with latent variable L

3.2.4 Generalized Association

In the previous subsections, we only considered domains with two attributes and a label. If there are three attributes and a label, the definition of marginal association does not change much, as we are still projecting all instances to the matrix of two attributes. Similarly, we might introduce a 3-way marginal association where only one attribute is removed. But the notion of conditional association involves three attributes and two roles for attributes: two attributes have been bound attributes, and one of them has been the conditional attribute. We should wonder what role should the fourth attribute have.

Inspired by [Dem02], let us introduce four disjunct sets, the bound set of attributes $\mathcal{B}$, the conditional set of attributes $\mathcal{C}$, the context set of attributes $\mathcal{K}$, and the marginal set of attributes $\mathcal{M}$ so that $\mathcal{B} \cup \mathcal{C} \cup \mathcal{K} \cup \mathcal{M} = \mathcal{A} \cup \{C\}$, where $\mathcal{A}$ is the set of attributes of a given domain, and C is the label. In [Dem02], the free set of attributes is $K \cup M$ and $\mathcal{C} = \{C\}$.

To investigate generalized association, we first disregard the attributes of the marginal set by computing the marginal contingency table of the remaining attributes. Then, for each value of $\bigotimes_{X \in \mathcal{C}} \mathcal{D}_X$, we investigate the corresponding contingency table of bound and context attributes.

Each of these $(|\mathcal{B}| + |\mathcal{C}|)$-way contingency tables is converted into a 2-way contingency table, so that every row corresponds to a tuple of bound attribute values, an element of the Cartesian product of bound attribute codomains $\bigotimes_{X \in \mathcal{B}} \mathcal{D}_X$, and every column to a similarly defined tuple of context attribute values. A 3-way partition matrix [Zup97] is obtained by a superimposing all these 2-way contingency tables, so that each field in the matrix carries the probability or frequency distribution of conditional set attribute value tuples.

When we investigate a particular generalized association, we are effectively studying the conditional association of bound attributes with respect to to the conditional attributes, while controlling for the context attributes. While the marginal set could always be empty, it is usually practical to include attributes into the marginal set in order to reduce the sparsity of the partition matrix.

The utility of generalized association was demonstrated in [Zup97] as means of losslessly eliminating groups of bound attributes and replacing them with a single new attribute, without affecting the classification performance of the classifier on the training set. The HINT algorithm always assumed that $\mathcal{C} = \{C\}$ and $\mathcal{M} = \emptyset$. Each new value of the attribute corresponds to some a subset of equivalent elements of a Cartesian product of the original attributes' codomains, given a equivalence relation. An appropriate equivalence relation is compatibility or indistinguishability of attribute values with respect to all other attributes and the label. If there are several possible bound sets, HINT picks the the set which yields an attribute with a minimal number of values, pursuing minimal complexity. If the method was repeatedly and recursively applied on the domain, there would eventually remain a single attribute whose codomain is the codomain of the label itself. HINT was intended for deterministic domains, but similar algorithms have been developed for noisy and non-deterministic problem domains.

Classification Association Marginal association is a special case of generalized association, where $\mathcal{K} = \mathcal{C} = \emptyset$. We define classification association for multi-way contingency tables as a special case of the generalized association: $\mathcal{K} = \emptyset$ and $\mathcal{C} = \{C\}$.

3.3 Interactions in Machine Learning

One of the first multivariate data analysis methods was Automatic Interaction Detector (AID) by Morgan and Sonquist [MS63]. AID was one of the first classification tree learning systems, according to [MST92] predated only by [Hun62].

The notion of interactions has been observed several times in machine learning, but with varying terminology. For example, J. R. Quinlan [Qui94] referred to the problem in such a way:

> We can think of a spectrum of classification tasks corresponding to this same distinction. At one extreme are *P-type tasks* where all the input variables are always relevant to the classification. Consider a n-dimensional description space and a yes-no concept represented by a general hyperplane decision surface in this space. To decide whether a particular point lies above or below the hyperplane, we must know all its coordinates, not just some of them. At the other extreme are the *S-type tasks* in which the relevance of a particular input variable depends on the values of other input variables. In a concept such as 'red and round, or yellow and hot', the shape of the object is relevant only if it is red and the temperature only if it is yellow.

He conjectured that classification trees are unsuitable for P-type tasks, and that connectionist back-propagation requires inordinate amounts of time to learn S-type tasks. By our terminology, P-type tasks indicate domains with independent attributes, while S-type tasks indicate domains with interacting attributes. However, in a single domain there may simultaneously be P-type subtasks and S-type subtasks.

Interactions are not an issue with most instance-based learning methods, based on computing proximities between instances. In fact, such methods are usually called upon to resolve the problem of interactions, assuming that interactions do not introduce non-Euclidean artifacts in the metric attribute space.

The notion of interactions in context of machine learning has been initially associated with hardness of learning in machine learning, e.g., though the example of learning parity [Ses89]. Sensitivity of *feature selection* algorithms to interactions was solved with algorithms such as Relief [KR92], recently surveyed and analyzed in [Šik02].

An important contribution to field was the work of Pérez and Rendell, who developed a method, *multidimensional relational projection* [Pér97], for discovering and unfolding complex n-way interactions in non-probabilistic classification problems. [PR96] is a comprehensive survey of attribute interactions. Pérez [Pér97] defined interactions as 'the joint effect of two or more factors on the dependent variable, independent of the separate effect of either factor', following [RH80].

More recently, Freitas reviewed the role of interactions in data mining in [Fre01], pointed out the relevance of interactions to rule interestingness, their relation with myopia of greediness, and constructive induction. In [FF99], they scanned for examples of Simpson's paradox in the UCI repository domains.

Perhaps the most important event in pattern recognition with regards to interactions was the book by Minsky and Papert [MP69], where they proved that the perceptron cannot learn linearly inseparable problems, such as the XOR function [RR96]. XOR is an example of a domain with interactions. The hidden layer in neural networks allows learning of three-way interactions, while $n-1$ hidden layers are required for n-way interactions. Here we refer to 'ordinary' linear neurons: other types of neurons may not be as sensitive to interactions.

3.4 Interactions in Regression Analysis

According to [Bla69], interactions can be defined as: 'A first-order interaction of two independent variables X_1 and X_2 on a dependent variable Y occurs when the relation between either of the X's and Y (as measured by the linear regression slope) is not constant for all values of the other independent variable.' Other expressions for interactions are moderator effects and moderating effects, but mediation refers to something different. In this section, we summarize [JTW90]. In statistical study of an interacting pair of variables, the moderator variable is often the weaker predictor of the two.

The significance of the interaction effect with dichotomous variables is estimated by the F test. The strength of the effect can be measured in a variety of ways, one of which is the η^2 index, defined as the proportion of variance in the dependent variable that is attributable to the interaction effect in the sample data. However, η^2 is a positively biased estimator of the effect size. Main effects are the effects of individual variables, while interaction effects are the contributions of variable interactions.

When working with many dichotomous variables, Bonferroni procedure, adjusted Bonferroni procedure, or Scheffe-like methods are recommended to control for experiment-wise errors and thus prevent discovering accidental interactions.

With continuous variables X_1, X_2 affecting a dependent variable Y, three methods are possible:

- dichotomization of both X_1 and X_2;
- dichotomization of the moderator variable (X_2), while the slope of X_1 and Y is studied independently for each of the values of the moderator variable; this way we can study also the nature of the interaction;
- the use of multiple regression, introducing multiplicative interaction terms ($X_1 X_2$). Here, an interaction is deemed significant if the difference between the R^2 values (squared sample multiple correlation coefficients) for the expanded model with (R_2^2) and the original model (R_1^2) without the interaction term is itself significant (just like higher-order terms of power polynomials in multiple regression) by testing the significance of the following statistic:
$$F = \frac{(R_2^2 - R_1^2)/(k_2 - k_1)}{(1 - R_2^2)/(N - k_2 - 1)},$$
where N is the total sample size, and k denotes the number of predictors in each model. The resulting F is distributed with $k_2 - k_1$ and $N - k_2 - 1$ degrees of freedom. It must be noted that this method induces multicollinearity in the model, because the variables are correlated with the interaction terms, introducing inflated standard errors for the regression coefficients. One recommendation is to center X_1 and X_2 prior to introducing the interaction term.

Interactions may be ordinal or disordinal. With disordinal or crossover interaction effects, the regression lines for different groups may intersect; for ordinal interactions this is not the case. With disordinal interactions, there may be a region of nonsignificance which is a range of values of X_1 where the value of the moderator variable X_2 has no effect.

3.4.1 Interactions and Correlations

Sometimes we want to verify if correlation between X and Y is constant at all levels of the moderator variable. The procedure for evaluating this null hypothesis when there is a single moderator variable is as follows:

We transform each correlation to Fisher's Z:
$$Z = \frac{1}{2}\left(\ln(1+r) - \ln(1-r)\right),$$
where r is the correlation between X and Y in a given group. The various values of Z are combined by means of the following formula:
$$Q = \sum_j (n_j - 3)(Z_j - Z')^2,$$
where n_j is the number of observations for group j, and
$$Z = \sum_j Z_j \frac{n_j - 3}{\sum_j n_j - 3}.$$
Q is distributed approximately as a χ^2 with $k - 1$ degrees of freedom.

3.4.2 Problems with Interaction Effects

When the interaction effect or a moderated relationship is the result of an irrelevant factor, we term this a false moderator, or a false interaction effect. Some possible causes are group differences in:

- range restrictions (i.e., less variability in X or in Y) due to arbitrary sampling decisions;
- reliability of the predictor variables;
- criterion contamination;
- predictor and criterion variable metric: it may make sense to transform the criterion variable to eliminate the false ordinal interaction effects.

We might not detect interactions because of small sample sizes. The lower the R values of independent variables, the larger the sample sizes need to be in order to obtain interaction effects with sufficient power. Power refers to the probability of correctly rejecting the null hypothesis. Without centering, problems with multicollinearity are likelier. When the interaction is not bilinear (in the sense that the slope of X_1 and Y changes as a linear function of the moderator variable X_2), the traditional cross product term is not appropriate for evaluating the interaction effect, and the interaction might go undetected. In fact, there are infinitely many functional forms of moderated relationships between continuous variables.

3.5 Ceteris Paribus

One way of explaining the concept of interactions is via a well-known concept in economics: *ceteris paribus.* The expression is normally used in the following sense, as explained by [Joh00]:

> *Ceteris Paribus*: Latin expression for 'other things being equal.' The term is used in economic analysis when the analyst wants to focus on explaining the effect of changes in one (independent) variable on changes in another (dependent) variable without having to worry about the possible offsetting effects of still other independent variables on the dependent variable under examination. For example, 'an increase in the price of beef will result, ceteris paribus, in less beef being sold to consumers.' [Putting aside the possibility that the prices of chicken, pork, fish and lamb simultaneously increased by even larger percentages, or that consumer incomes have also jumped sharply, or that CBS News has just announced that beef prevents AIDS, etc. — an increase in the price of beef will result in less beef being sold to consumers.]

If we state an influence of X on Y under the ceteris paribus assumption, we explain Y from X while all other variables are kept constant. If X and Y interact with some Z, we would not be able to plot a graph of X versus Y without controlling for Z. Therefore, we may only use the ceteris paribus assumption when there are no interactions of X and Y with other variables.

Most useful statements in economics are usually of qualitative nature ('the lower the price, the greater the quantity of goods sold'), so we can relax the interpretation, formulating it rather as: there is no attribute Z which would reverse the qualitative association between X and Y. There may be interactions, as long as they do not invalidate the qualitative statements.

3.6 Game Theory

An interesting question is that of estimating the *value* of a game to a given *player*. A colorful application of interaction indices is in political analysis of coalition voting 'games', where players are voters, e.g., political parties in a parliament, or a court jury. In such a case, value is the ability to affect the outcome of the vote, and *power indices* are its measures [DS79, MM00]. Two well-known examples of power indices are Shapley-Shubik and Benzhaf-Coleman values, the former arising from game theory, and the latter emerging from legislative practice. Power index is an estimate of the actual voting power of a voter in a given coalition: not all voters have the same power, even though they may have the same weight. For that reason, votes in courts are now sometimes weighted using the knowledge of power indices.

However, players may interact: cooperate or compete. This may twist the power indices, which place different assumptions about the coalitions. For example, Benzhaf-Coleman index assumes that all coalitions are equiprobable, while for Shapley-Shubik index it is assumed that the decision of each voter is uniformly random and independent of other voters. As a possible solution, according to [Mar99] first mentioned in [Owe72], we may instead calculate the value for a coalition of a pair of players. *Interaction index* is the measure of coalition value. Interaction index was axiomatized for arbitrary groups of k players in [GR99]. Interaction index is defined with respect to some value estimate, e.g., Benzhaf or Shapley value, which carries the assumptions about coalitions.

For the simple case of a 2-way interaction index, the interaction index for players i and j, some coalition S, and a value function v is

$$v(S \cup \{i,j\}) - v(S \cup \{i\}) - v(S \cup \{j\}) + v(S).$$

Interaction index may be positive or negative. If it is positive, the players should cooperate in a *positive interaction*, else they should compete in a *negative interaction*. If interaction index is zero, the players can act independently. However, if we are not given a coalition, it is often worth studying the interaction index over multiple possible coalitions, and [GR99] suggests averaging over all possible coalitions.

In economics, concretizations of the term player are diverse, but the value is almost always utility or a monetary quantity. The players may be companies, and we study whether these companies are competitors or complementors. They are competitors when the interaction index is negative, and they are complementors when the interaction index is positive. Players may also be goods from the viewpoint of a consumer. A matching skirt and a blouse are complementary, while two pairs of similar boots are substitutable.

Recently, these concepts have also been applied to studying interactions between attributes in the rough set approach to data analysis [GD02], simply by placing the classifier evaluation function in the place of value of a game, and an attribute in the place of a player. Furthermore, there are interesting applications of interaction indices in fuzzy vote aggregation [Mar99].

CHAPTER 4

Interactions

A theory is something nobody believes, except the person who made it.
An experiment is something everybody believes, except the person who made it.

In this chapter, we provide our own definition of interactions. Although we have used the concept of interactions and have listed many possible definitions, we have neglected to decide upon the definition we will pursue ourselves. Our definition will be built upon the concepts from Chaps. 2 and 3, applied to the naïve Bayesian classifier (NBC) as the fundamental learning algorithm. We will first investigate the deficiencies of NBC, and focus on interactions as one cause of the deficiencies. In the remainder of our work, we will address interactions, how to find them and how to deal with them.

4.1 Naïve Bayesian Classifier

We have mentioned Bayes rule earlier in Sect. 2.5.1. A *naïve Bayesian classifier* (NBC) is its concrete form as applied to classification. Our derivation will follow [Kon97]:

We start with the Bayes rule:

$$P(c_k|\mathbf{i}) = P(c_k)\frac{P(\mathbf{i}|c_k)}{P(\mathbf{i})},$$

and assume independence of attributes $A_1, \ldots, A_n \in \mathcal{A}$, given class c_k, meaning that:

$$P(A_1(\mathbf{i}), A_2(\mathbf{i}), \ldots, A_n(\mathbf{i})|c_k) = \prod_{i=1}^{n} P(A_i(\mathbf{i})|c_k). \tag{4.1}$$

We then obtain:

$$P(c_k|\mathbf{i}) = \frac{P(c_k)}{P(\mathbf{i})}\prod_{i=1}^{n} P(A_i(\mathbf{i})|c_k).$$

After another application of Bayes rule:

$$P(A_i(\mathbf{i})|c_k) = P(A_i(\mathbf{i}))\frac{P(c_k|A_i(\mathbf{i}))}{P(c_k)},$$

we obtain

$$P(c_k|\mathbf{i}) = P(c_k)\frac{\prod_{i=1}^{n} P(A_i(\mathbf{i}))}{P(\mathbf{i})}\prod_{i=1}^{n}\frac{P(c_k|A_i(\mathbf{i}))}{P(c_k)}.$$

Since the factor $(\prod_{i=1}^{n} P(A_i(\mathbf{i})))/P(\mathbf{i})$ is independent of the class, we leave it out and obtain the final formula:

$$P(c_k|\mathbf{i}) = P(c_k)\prod_{i=1}^{n}\frac{P(c_k|A_i(\mathbf{i}))}{P(c_k)}.$$

The objective of a learning algorithm is to approximate the probabilities on the right-hand side of the equation. The knowledge of the NBC is a table of approximations of a priori class probabilities $P(c_k)$, $k = 1, \ldots, |\mathcal{D}_C|$, and a table of conditional probabilities of class c_k given a value $(a_i)_j$ of attribute a_i, $i = 1, \ldots, |\mathcal{A}|$, $j = 1, \ldots, |\mathcal{D}_{A_i}|$: $P(c_k|(a_i)_j)$.

The NBC formula yields a factor, proportional to the probability that the instance $\mathbf{i}$ is of class c_k. Instance $\mathbf{i}$ is described with the values of attributes, $A(\mathbf{i})$, where A is one of the attributes $A \in \mathcal{A}$:

$$\Pr\{d_{NB}(\mathbf{i}) = c_k\} \propto f_{NB}(\mathbf{i}, c_k) = P(c_k)\prod_{A\in\mathcal{A}} P(A(\mathbf{i})|c_k). \tag{4.2}$$

The zero-order label probability distribution obtained by joined votes is normalized, to guarantee that the probabilities for all the classes sum up to 1 for a given instance $\mathbf{i}$:

$$\Pr\{d_{NB}(\mathbf{i}) = c_k\} = \frac{f_{NB}(\mathbf{i}, c_k)}{\sum_{l=1}^{|\mathcal{D}_C|} f_{NB}(\mathbf{i}, c_l)}. \tag{4.3}$$

4.1.1 Naïve Linear Regression

Multiplication of zero-order probability distributions in classification could be roughly compared to summation in a particularly simple form of linear regression, which we will call naïve linear regression (NLR). Modern multiple regression procedures solve some of the deficiencies of NLR by using least-squares fitting instead of averaging. Assumptions are similar: both NLR and NBC are linear and assume attribute independence. The analogy between NLR and NBC can be seen as a survey of deficiencies in NBC.

In NBC, we compute conditional probabilities of classes given an attribute value, and the classes' prior probabilities. Using these probabilities, we predict the label probability distribution. It is possible to use several ad hoc techniques for artificially 'smoothing' or 'moderating' the probability distribution, often with good results, e.g. in the m-error estimation [Ces90].

In NLR, we compute the effect f of each attribute x_i, $i = 1, 2, \ldots, k$ on the label value y on n instances in the familiar way with univariate least-squares fitting:

$$f(x_i) = \frac{\sigma_{x_i y}}{\sigma_i^2}(x_i - \bar{x}_i),$$

$$\sigma_i^2 = \frac{1}{n-1}\sum_{j=1}^{n}(x_{i,j} - \bar{x}_i)^2,$$

$$\sigma_{x_i y} = \frac{1}{n-1}\sum_{j=1}^{n}(x_{i,j} - \bar{x}_i)(y_j - \bar{y}).$$

σ_i^2 is the sample variance of attribute x_i, and $\sigma_{x_i y}$ is the sample covariance. We arrive to y from x_i by $y = f(x_i) + \bar{y}$. Finally, we naïvely average all the effects, and obtain the following multivariate linear regression model:

$$f(\mathbf{x}) = \bar{y} + \frac{1}{k}\sum_{i=1}^{k}\frac{\sigma_{x_i y}}{\sigma_i^2}(x_i - \bar{x}_i).$$

When attributes are correlated the independence assumption is violated. One solution to NLR is multivariate least-squares fitting, which is performed on all attributes at once, and the resulting 'ordinary' LS regression is the most frequently used procedure. However, correlations still influence the significance tests (which assume independence).

Neither NLR nor NBC are resistant to uninformative attributes. Techniques for NBC like *feature selection* have an analogous set of techniques in regression called best-subset regression. For training NBC, a wrapper approach is based on adding attributes to the model one after another, or removing one after another, until the quality of the model is maximized. In regression, this is called the *step-wise method.*

Both, NLR and NBC tolerate errors in attributes, but assume the label information is perfect. In regression, orthogonal least squares (OLS) fitting assumes a measurement error also in the label values, fitting correspondingly. Another group of methods, based on robust statistics, accept that certain instances might be outliers and not subject to the model, and prevent them from influencing it, thus achieve a better fit on the model-conforming data.

Furthermore, missing attribute values are problematic in many ways. This is usually solved by leaving out the vote of the missing attribute, or by using specific imputation methods. Although it is usually assumed that values are missing at random, it is likelier that they do not miss at random. It can be more profitable to represent the missing value with a special value of the attribute.

4.1.2 NBC as a Discriminative Learner

We will now show how the naïve Bayesian classifier (NBC) can be understood as a linear discriminator. Its optimality is subject to linear separability as has been observed already in [DH73]. We will discuss the form of NBC which only applies for ordinal attributes, which is the form normally used in the machine learning community. Continuous attributes should be discretized prior to applying NBC. Sometimes, e.g., in [RH97], NBC is often formulated differently, assuming a p-dimensional real observation vector and a model (e.g., Gaussian) for class densities. In this section, we will only be concerned with non-probabilistic discriminative learning.

In the two-class discrimination problem $\mathcal{D}_C = \{c_1, c_2\}$, the objective is to correctly determine the most likely class, rather than to estimate the probability distribution. Thus, instance $\mathbf{i}$ is predicted to be of class c_1 if:

$$P(c_1)\prod_{A\in\mathcal{A}}P(A(\mathbf{i})|c_1) > P(c_2)\prod_{A\in\mathcal{A}}P(A(\mathbf{i})|c_2).$$

We can take the logarithm of both sides in the inequality. Because of monotonicity of logarithm, this is without loss. We then rearrange the terms:

$$\log\frac{P(c_1)}{P(c_2)} + \sum_{A\in\mathcal{A}}\log\frac{P(A(\mathbf{i})|c_1)}{P(A(\mathbf{i})|c_2)} > 0.$$

NBC allows each attribute value to separately affect the class distribution. Therefore, not each attribute but each attribute value has its own dimension. We illustrate this with *dummy coding*. If an instance has a particular attribute value, the dummy value for that dimension becomes 1, else 0. So each n-valued attribute A, $n = |\mathcal{D}_A|$ is replaced with n dummy attributes. We will treat them as Boolean factors. Thus, the class boundary is a hyperplane in as many dimensions as there are attribute-value pairs. An attribute A has n_A values: $a_1,\ldots,a_{n_A}$.

$$\log\frac{P(c_1)}{P(c_2)} + \sum_{A\in\mathcal{A}}\sum_{l=1}^{n_A}(A(\mathbf{i}) = a_l)\log\frac{P(a_l|c_1)}{P(a_l|c_2)} > 0,$$

where the $n_C\sum_{A\in\mathcal{A}}n_A$ probabilities $P(v_{A,l}|c_k)$, $l = 1,\ldots,n_A$ have been calculated on the training set.

With such a representation of NBC, it is quite easy to see that NBC cannot be always zero-one loss optimal even when the domain is linearly separable in attribute-value space. Conjunctions and disjunctions are linearly separable in attribute-value space, and NBC correctly captures them according to [DP97]. But most Boolean m-of-n concepts cannot be learned with NBC, although they are linearly separable [ZLZ00]. We need not stress that classification accuracy is less stringent than accuracy of label probability distributions in probabilistic classifiers that we pursue.

4.2 Improving NBC

In previous sections we used the terminology normally used in the context of the naïve Bayesian classifier. We will now show how NBC fits in the formalisms of Chap. 2. NBC is based on $\sum_{A\in\mathcal{A}}|\mathcal{D}_A|$ zero-order zero-descriptor models, one for each attribute value. However, for a particular instance (assuming no probabilistic attribute values), only $|\mathcal{A}|$ models will be used, because we apply segmentation for every attribute along its values. The voting method is a simple multiplication of probability functions followed by normalization in order to fulfill the requirement that all probabilities should sum up to 1.

The NBC learning algorithm returns a classifier which, for a problem definition $P = (\mathcal{A}, C), \mathcal{A} = \{A_1, A_2, A_3\}$ and an instance world $\mathcal{I}$, a subset of which we are given as $\mathcal{T}\subseteq\mathcal{I}$, takes the following form:

$$L_{NB}((\{A_1,A_2,A_3\},C),\mathcal{T}) = IBC(A_1,A_2,A_3) = V\begin{pmatrix}E[\mathcal{T},C,S(A_1)]\\E[\mathcal{T},C,S(A_2)]\\E[\mathcal{T},C,S(A_3)]\end{pmatrix}.$$

Here, E is the estimation function, V is the voting function, and S is the segmentation function.

There are many possibilities for improving the naïve Bayesian classifier. We now discuss replacing each of the constituent elements of the NBC, voting, projection, estimation, and finally, segmentation.

Voting

There has been little discussion about validity of (4.2), but normalization in (4.3) has been subject to several proposals. Feature selection can be seen as a simplistic optimizing voting algorithm, which includes and excludes attributes in order to maximize value of the evaluation function.

Feature selection is an obvious example of weighting, where an attribute either has weight of 1, or weight of 0. The voting function V_{fs} performs optimal feature selection, which precedes voting according to (4.3). Thus, if V_{fs} is given a set of probability functions $\mathcal{M}$, it will select a subset $\mathcal{M}' \subset \mathcal{M}$, so that the value of the evaluation function is maximized.

If we abandon the rigorous probability distribution estimation methods, we could assign weights to individual attributes [WP98], or even to individual attribute values [FDH01]. For example, each of a pair of identical attributes would only get half the weight of other attributes. If the weights are always greater or equal to zero, the simple normalization in (4.3) is not affected.

Estimation-Voting

We can view the function f_{NB} in (4.2) as a projection, the result of which is a continuous descriptor. This way we replace voting with estimation. We may consequently estimate the label probability distribution with respect to the descriptor, somewhat like:

$$E\left[\mathcal{T}, C, f_{NB}\begin{pmatrix} E[\mathcal{T}, C, S(A_1)] \\ E[\mathcal{T}, C, S(A_2)] \\ E[\mathcal{T}, C, S(A_3)] \end{pmatrix}\right].$$

One approach to estimation is *binning*, suggested in [ZE01], where we segment the values of f_{NB} and estimate the class distribution for each segment: $E(\mathcal{T}, C, S(f_{NB}(\cdot)))$. Alternatively, we may assume that the label is logistically distributed with respect to f_{NB}, and obtain the parameters of the logistic distribution, with $E_{\text{ln}}(\mathcal{T}, C, f_{NB}(\cdot))$. An approach similar to that is described in [Pla99]. For both approaches, we may split the training set into the validation and remainder set, as described in Sect. 2.6.2. Then we use the remainder set to train f_{NB}, while the validation set is used to obtain the parameters of the logistic distribution, or the bin frequency counts.

Estimation

If our evaluation function is classification accuracy, which does not appreciate probabilistic classifiers, we may retain the closed-form voting function, but replace the estimation algorithm E with one that explicitly maximizes the classification accuracy [RH97]. The voting approach in [WP98] is also based on optimizing the weights in order to maximize

0-1 loss. Similar approaches work for other algorithms too: classification accuracy of 0-1 loss-minimizing (rather than conditional likelihood-maximizing) logistic regression is better than classification accuracy of NBC when there is plenty of training data [NJ01].

Projection

NBC can be viewed as a simple closed-form projection function, where the descriptor that it provides is computed with associated closed-form functions. The descriptor computed from probability estimates, obliviously to the evaluation function, may be rather suboptimal, as discussed in Sect. 4.1.1.

Alternatively, we could introduce a parameter vector $\mathbf{w} \in \mathbb{R}^{\sum_{A \in \mathcal{A}} |A|}$, where its individual dimensions are $\mathbf{w}_0, \mathbf{w}_{1,1}, \ldots, \mathbf{w}_{|\mathcal{A}|,n_{|\mathcal{A}|}}$. Each of them corresponds to an attribute-value pair. The parameters appear in the following generalized linear function:

$$f_{GL}(\mathbf{i}, c_1) > f_{GL}(\mathbf{i}, c_2) \Leftrightarrow w_0 + \sum_{i=1}^{|\mathcal{A}|} \sum_{l=1}^{n_i} w_{i,l}(v_{i,\mathbf{i}} = (a_j)_l) > 0, \tag{4.4}$$

$$(v_{i,\mathbf{i}} = (a_j)_l) = \begin{cases} 0 & \text{if } v_{i,\mathbf{i}} \neq (a_j)_l \\ 1 & \text{if } v_{i,\mathbf{i}} = (a_j)_l. \end{cases}$$

We now have an arbitrary linear projection function. In this form, the model closely resembles the one of a perceptron, the one-layer neural network. This analogy is discussed with more detail in [Kon90, Hol97]. The parameter $\mathbf{w}$ can be fitted with a variety of methods. We may apply special purpose Bayesian neural network training algorithms [Kon90]. We may even dispose of any connection with probability and employ a general purpose Rosenblatt's perceptron learning algorithm, or margin-maximization algorithms based on quadratic programming [Vap99].

The change of the description did not affect comprehensibility or complexity of the classifier. The classifier can still be easily visualized with a nomogram or a similarly simple alternative. The only difference lies in flexibility of choosing algorithms for parameter fitting.

Unfortunately, we have abandoned the probabilistic nature of the original naïve Bayesian classifier. The continuous value f_{GL} in (4.4) can be seen as a projection of an instance into a continuous descriptor, and we may then apply estimation using binning or parametric estimation, as in Sect. 4.2, to obtain label probability distribution.

Segmentation

A segmentation function $S(A)$ maps an attribute A to a parameter function $s_A : \mathcal{I} \to \mathcal{D}_A$, $s_A(\mathbf{i}) = A(\mathbf{i})$. It maps an instance to a discrete set of values. In NBC the V function is a simple voting function that joins multiple models by multiplying the probability functions they submit. Function $E(\mathcal{T}, C, s_A)$ estimates a zero-order model with one discrete descriptor: $M_0^1(C(\mathbf{i})|s_A(\mathbf{i}))$, for all $\mathbf{i} \in \mathcal{T}$. We thus have $|\mathcal{D}_A|$ zero-order M_0^0 models, and the model is chosen according to the value of $s_A(\mathbf{i})$. The discrete descriptor has as many values as the there are values in the range of the parameter function s_A.

Some concepts, e.g., those discussed in [ZLZ00], cannot be learned with NBC. They cannot be learned even if we replace the voting function and even if adopt the generalized linear model. The problem lies in the inability of the segmentation algorithm to distinguish different groups of instances which have discernible class distributions. The only way lies in increasing the level of segmentation. Since individual attributes are already maximally segmented, the only way of increasing the level of segmentation can be achieved by inter-attribute segmentation. The new segments are now defined by conjunction of attribute values of two or more attributes. Learning by segmentation is exemplified by decision tree algorithms, $L(P, \mathcal{T}) = E(\mathcal{T}, C, S(A_1, A_2, A_3))$. If there are only three attributes, the voting function has been trivialized.

Inter-attribute segmentation is achieved by joining attributes into new attributes, as first suggested in [Kon91], and later in [Paz96]. We will now investigate the situations where attribute joining is required in detail, focusing solely on NBC, because there are too many possible generalizations.

The function S may try to optimize the profitability of the classifier by *attribute reduction*: creating fewer segments as is possibly could: this is performed using heuristics, either traditionally top-down by stopping the process of segmentation at some point, or by bottom-up merging of segments for as long as it pays, which can be observed in constructive induction [Dem02]. From the viewpoint of constructive induction, we can interpret the value of the segmentation function $S(A_1, A_2, A_3)$ applied on three attributes as a new constructed attribute which joins A_1, A_2, and A_3. Its value is the value of the segmentation function.

4.3 Interactions Defined

Earlier, we have referred to interactions as situations when attributes should not be considered independently. In the context of the naïve Bayesian classifier, the segmentation functions provide us with the sole means of joint treatment of multiple attributes. *An interaction is a feature of the problem domain which cannot be learned by means other than joint segmentation of two or more attributes.* An interaction is *resolved* by joint segmentation of two or more attributes.

We will name interactions between two attributes and the label as three-way interactions, and interactions between $n-1$ attributes and the label as n-way interactions. We note that an association is a 2-way interaction, but we will not refer to is as an interaction. XOR is an example of a 3-way interaction, and parity on n bits is often used as an example of a $(n+1)$-way interaction. In the context of classification, we are not interested in interactions that do not involve the label.

An important requirement for a general k-way interaction, $k > 2$, is *collapsibility*: by the above definition, it must be impossible for an interaction to be collapsed into any combination of interactions of lower order. Three attributes may be truly be dependent, but if this dependence occurs merely because only two attributes are interacting, we cannot say that there is a 3-way interaction! If it is true that $P(A, B) \neq P(A)P(B)$ it is also true that $P(A, B, C, D) \neq P(A)P(B)P(C)P(D)$, but in spite of dependence among $\{A, B, C, D\}$, an interaction among them is not necessary, because we only have evidence for a 2-way interaction between A and B. There might be another, but separate interaction between C and D. Therefore, if we can learn some feature of the problem domain with several interactions, neither of which is k-way or more, there is no k-way interaction in the domain. This problem is still simple enough with 3-way interactions, but becomes tedious with interactions of higher order.

Our dimensionality-averse definition tries to be aligned with traditional definitions of interactions, e.g., in loglinear models. In many respects, it could be preferable to minimize simplicity of the knowledge, rather than to minimize dimensionality, as we do.

In a more general context, we may define interactions with respect to limitations of the projection functions, instead of basing them on segmentation functions, as we did. Even if the concept is simple, a formalization is tedious, and will be left for some other occasion. In a limited context, if we adopt a generalized linear model as a projection function, and a discriminative classification problem, interactions involve linearly inseparable domains. The concept of collapsibility helps us identify the minimal set of attributes and dimensions needed to resolve inseparability.

4.3.1 Interaction-Resistant Bayesian Classifier

We will now introduce an interaction-resistant naïve Bayesian classifier learner (IBC). Instead of a flat set of attributes $\mathcal{A}$, IBC uses $\mathcal{S}$, a set of subsets of $\mathcal{A}$, for example, $\mathcal{S} = \{\{A_1, A_2\}, \{A_3\}\}$. Each element $\mathcal{S}_i \in \mathcal{S}$, indicates an interaction, and each interaction is resolved independently by applying a segmentation function non-myopically on all attributes $A \in \mathcal{S}_i$. In our example, the segmentation function attempts to resolve the interaction between A_1 and A_2. A briefer expression is $IBC(A_1A_2, A_3)$, following Sect. 3.2.2. After the segmentation functions do their job, we estimate and vote, as usual.

A careful reader surely wonders if there is any difference between existence of interactions, and violations of the assumption of independence in NBC. As mentioned earlier, for NBC, attributes are assumed to be conditionally independent given the class. There is no difference, but the concept of interactions requires the violation of independence to be clearly defined. Furthermore, interactions may not be collapsible into simpler interactions, which requires us to only seek islands of impenetrable dependence.

We will distinguish several types of interactions. The first group of interactions are true interactions: pairing up truly interacting attributes yields us more information than we would expect considering each attribute separately. Truly interacting attributes are synergistic. The second group are false interactions, where several attributes provide us the same information. Conditional interactions indicate complex situations where the nature and existence of another interaction depends on the value of some attribute. The types are discussed in more detail in Sect. 4.4.

Notes on the Definition

In categorical data analysis [Jac01], a 2-way interaction indicates an interaction between two variables *with respect to* population frequency counts. Often, the independent variable is not mentioned, but several models are built, one for each value of the independent variable. This indicates that the third variable is dependent and controlled for among the models! However, a dependence between 3 variables is a 3-way interaction, by definitions from 3.2.

We will understand the problem if we notice that in multiple regression [JTW90] a 2-way interaction indicates an interaction between two dependent variables *with respect to* the dependent variable. In regression, as well as in classification, we are not interested in any interactions that do not involve the label. But the order of the interactions is tied with the notion of the number of variables, and by this logic, what is called a 2-way interaction with respect to the dependent variable in regression should instead be called simply a 3-way interaction.

Since the terminology of interactions has not been established in machine learning, we have chosen to deviate from the terminology of interactions in regression. In classification the label is the third attribute, and we include it into consideration. A 2-way interaction in classification is obvious and trivial: it merely indicates that the attribute interacts with the label. In the context of generative learning, it is also interesting to study interactions, but we only have equal attributes, without distinguishing any of them with the role of the label. Our egalitarian 3-way interactions without respect to anything correspond to 2-way interactions with respect to the label.

4.3.2 A Pragmatic Interaction Test

How do we check whether a certain interaction exists in the domain, given a set of data? We know how to assume an interaction, and what this assumption means, but we now want to check for it. From the pragmatic perspective, effective decision-making is the primary purpose of knowledge and learning. We are sometimes better off disregarding some interactions if the training set $\mathcal{T}$ is too small, even if they exist in the instance world $\mathcal{I}$, from which $\mathcal{T}$ is sampled.

If we test the classifier on training data, the classifier which resolves an interaction will normally perform better than one that does not. But if the classifier is tested on unseen data, we need sufficient training data to be able to take advantage of the interaction. Otherwise, the classifier which assumes independence will perform better, even if there is an interaction. Namely, the segmentation function fragments data.

Philosophically, any property of the data is only worth considering if it helps our cause, and gives us a tangible benefit. This is the ultimate test for interactions: interactions are significant if they provide a benefit. This can be contrasted to performing statistical tests of whether interaction effects have been significant or not at some p-value.

A 3-way interaction between attributes A_1 and A_2 is *significant* given the NBC learning algorithm, a training set $\mathcal{T}$ and a test set $\mathcal{E}$ of instances, when:

$$q_{\mathcal{E}}\left(V\begin{pmatrix}E[\mathcal{T},C,S(A_1)]\\E[\mathcal{T},C,S(A_2)]\\E[\mathcal{T},C,S(A_3)]\end{pmatrix}\right) < q_{\mathcal{E}}\left(V\begin{pmatrix}E[\mathcal{T},C,S(A_1,A_2)]\\E[\mathcal{T},C,S(A_3)]\end{pmatrix}\right),$$

as measured by some evaluation function q. We will assume that this should be valid regardless of the choice of the voting function V, even if we cannot practically check the model with all these possibilities.

It is more difficult to test the significance of a k-way, $k > 3$, interaction between attributes and the class $\mathcal{A}' \subseteq \mathcal{A}$, where $|\mathcal{A}' \cup \{C\}| = k$. It is only significant when the quality cannot be matched with any classifier which uses m-way interactions, where $m < k$. If we are learning a classifier $IBC(\mathcal{S})$, where $\mathcal{S}_k \in \mathcal{S}, |\mathcal{S}_k \cup \{C\}| = k, \hat{\mathcal{S}} = \mathcal{S} \setminus \{\mathcal{S}_k\}$, there is a significant k-way interaction between attributes in $\mathcal{S}_k$ and the label iff:

$$q(IBC(\mathcal{S})) > \max_{S' \subseteq (\mathcal{P}(\mathcal{S}_k) \setminus \{\mathcal{S}_k\})} q(IBC(\hat{\mathcal{S}} \cup \mathcal{S}')), \tag{4.5}$$

where $\mathcal{P}(\mathcal{S}_k)$ is the power set of $\mathcal{S}_k$, containing all its subsets. Sometimes a different expression for power set is used: $\mathcal{P}(\mathcal{S}) = 2^{\mathcal{S}}$.

The definition in (4.5) is complex because it tries to account for the fact that subtracting models from the voting function may sometimes improve results. Instead of assuming an optimizing voting function, we admit that the interaction among all attributes of $\mathcal{A}'$ is significant only when no classifier which performs segmentation only on any proper cover of any subset of $\mathcal{A}'$ matches or exceeds the performance of a classifier which resolves the interaction by using the segmentation function on the whole $\mathcal{A}' : S(\mathcal{A}')$.

If we assume a feature-selecting voting function V_{fs}, and an IBC_{fs} based on it, a k)-way interaction between attributes in $\mathcal{S}_k$ and the class is significant iff:

$$q(IBC_{\text{fs}}(\mathcal{S})) > q(IBC_{\text{fs}}(\mathcal{S} \setminus \{\mathcal{S}_k\})).$$

If the reader finds the pragmatic test repugnant, we will evade a proper justification by depending on a vast amount material in statistics, where interactions are searched for primarily by model testing. Very few attributes in real data are truly conditionally independent, and tests of significance of attribute dependence are not much better than classifier evaluation functions. However, the reader is referred to Sect. 5.4 to see a heuristic assessment of an interaction between three attributes, based on entropy.

4.4 Types of Interactions

4.4.1 True Interactions

Most references to interactions in machine learning have dealt with the type of interaction exemplified by the exclusive OR (XOR) function $c := a \neq b$:

A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

Observe that the values of attributes A and B are completely independent, because we assumed that they are sampled randomly and independently: $P(a, b) = P(a)P(b)$. However, they are not conditionally independent: $P(a, b|c) \neq P(a|c)P(b|c)$. XOR is a standard problem for feature selection algorithms because $IBC(AB)$ yields a perfect classifier, even if $q(IBC(A)) = q(IBC(B)) = q(IBC())$. Because of this property, we refer to XOR as an example of a perfectly true interaction: the attributes are only useful after the interaction is resolved.

A generalization of XOR is the n-XOR, or the parity problem, where the binary class is a sum of n binary attributes modulo 2. There exists no k-way interaction on these attributes, $k < n$. Note that in 3-XOR problem, $d := (a + b + c) \pmod 2$, a and b are conditionally independent given d: $P(ab|d) = P(a|d)P(b|d)$, but this still violates the assumption of the naïve Bayesian classifier! Remember that the assumption in (4.1) is $P(a, b, c|d) = P(a|d)P(b|d)P(c|d)$.

For that reason, perfectly true interactions are a hard problem for most forward-searching algorithms that probe for interactions. The issue is partly remedied with backward-searching attribute-subtracting algorithms which start with, e.g., $IBC(WXYZ)$, and work backwards from there, removing independent attributes. An example of a backward-searching algorithm is MRP (Multidimensional Relational Projection), described in [Pér97].

Most true interactions are not as spectacular as the XOR. Both attributes yield some benefit, but their full predictive potential cannot be unleashed without resolving the interaction.

A	B	$P(c_0)$	$P(c_1)$
0	0	0	1
0	1	1/5	4/5
1	0	1/2	1/2
1	1	1	0

We note that $P(c_0|a_0) = \frac{1}{10}$, $P(c_0|b_0) = 1/4$, however $P(c_0|a_0, b_0) = 0$, while $P(c_0|a_0)$ $P(c_0|b_0) = 1/40$. We normalize our vote, knowing that $P(c_1|a_0)P(c_1|b_0) = 27/40$, so the final outcome is $\Pr\{IBC(A, B)([a_0, b_0]) = c_0\} = 1/28$.

We conclude that there is an interaction between A and B, because we derive better a prediction by resolving the interaction. Most interactions in true data are such.

We will now examine the example of OR, where $c := a \vee b$:

A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

We note that $P(c_0|a_0) = 1/2, P(c_0|b_0) = 1/2$, however $P(c_0|a_0, b_0) = 1$, while $P(c_0|a_0)$ $P(c_0|b_0) = 1/4$! Our voting is normalizing, and since $P(c_1|a_0)P(c_1|b_0) = 1/4$, the final outcome is $\Pr\{IBC(A, B)([a_0, b_0]) = c_0\} = 1/2$. We conclude that there is an interaction between A and B, even if the OR domain is linearly separable: this is because we are working with probabilistic classifiers and not merely with discriminative classifiers.

4.4.2 False Interactions

Situations where attributes do not have any synergistic effect and provide us with overlapping information will be referred to as false interactions. Just as true interactions, false interactions can too be found to be significant, and resolving them improves the classifier quality. The purpose of resolution in such a case is to correctly weigh the duplicated information, and such interactions are too found to be significant. However, although resolution may involve joint segmentation, at least some false interactions can be resolved without joint segmentation, but, for example, by an adaptive voting function. Therefore we want to distinguish them from synergistic true interactions, and will refer to them as false interactions.

Let us now consider a three-attribute domain, where attribute A is duplicated in A_1 and A_2. These two attributes are dependent, while A and B are conditionally independent.

A_1	A_2	B	$\Pr\{IBC(A_1, A_2, B) = c_0\}$	$P(c_0)$	$P(c_1)$
0	0	0	1/13	1/7	6/7
0	0	1	3/7	3/5	2/5
1	1	0	4/7	2/5	3/5
1	1	1	12/13	6/7	1/7

The domain was engineered so that NBC would yield perfect results if either A_1 or A_2 was dropped from the attribute set. However, the learning algorithm might not account for that possibility. An ideal interaction detector operating on an infinite sample from the above domain would detect interaction $A_1 A_2$. After resolving it, the classifier becomes perfect.

Our example is the most extreme example of dependent attributes. Similar effects may appear whenever there is a certain degree of dependency among attributes. This type of interactions may sometimes disappear if we use a feature-selecting voting function V_{fs}. However, false interactions are not covered by several definitions of interactions mentioned earlier. Perhaps we should call a falsely interacting pair of attributes instead a pair of attributes that have interacted with a latent attribute.

Mutually Exclusive Events: Perfectly False Interactions

We modify the above example, but A_1 and A_2 are now mutually exclusive events: when $v_{A_1} = 1$, event A_1 occurred, when $v_{A_1} = 0$, event A_1 did not occur. Interaction between A_1 and A_2 will be found, and can we interpret the value of $S(A_1, A_2)$ as a value of a new, less redundant two-value latent attribute A_L, which resolves the interaction between A_1 and A_2. It might be desirable to provide each input attribute value as a separate attribute, and rely on the mechanism for resolving false interactions to create multi-valued attributes.

A_1	A_2	B	$P(c_0)$
0	1	0	1/7
0	1	1	3/5
1	0	0	2/5
1	0	1	6/7

Multiple Noisy Measurements: Ordinary False Interactions

Consider a case when A_1 and A_2 are separate, but noisy measurements of unobserved A_L. Attribute selection operating merely from the perspective of dependence might dispose of one of them. On the other hand, using both of them in predictions would improve the results, assuming that the noise in A_1 is independent of noise in A_2.

Resolving False Interactions

While resolving true interactions is efficiently performed with joint segmentation, segmentation is only one of the solutions for resolving false interactions. There are several approaches:

Explicit Dependence: A tree-augmented Bayesian classifier (TAN) [FG96] would discover a dependence between attributes A_1 and A_2, decide that A_2 is a consequence to A_1, and consequently it would approximate the joint probability distribution as $P(A_1, B_2, C) = P(C)P(A_1|C)P(A_2|C, A_1)$, in contrast to the NBC, which is based on $P(A_1, A_2, C) = P(C)P(A_1|C)P(A_2|C)$.

Latent Attributes: With segmentation we would create a new attribute A_L replacing both A_1 and A_2. Most segmentation functions are more appropriate for resolving proper interactions, not false interactions, because they use crisp mapping to segments. For example, if A_L is the unobserved cause of A_1 and A_2, we could be uncertain about the exact value of A_L. This paradigm has been discussed by the authors of TAN in [ELFK00]. Taking the Cartesian product of both attributes is a very simplistic form of the latent attribute approach. For example, a group of falsely interacting attributes are better left alone, without segmenting them jointly, even if they violate the NBC assumptions, as we will see in Sect. 6.2.1. A particularly simple latent attribute strategy is feature selection, where a group of attributes is replaced with a single one. Feature selection improves results both because random attributes confuse a learning algorithm, and because false interactions bias the results.

Do-Nothing: It is observed in [RHJ01] that naïve Bayesian classifier in the discriminative context of 0-1 loss works optimally in two cases: when all the attributes are truly independent (as it is assumed), and when all the attributes are perfectly dependent. Therefore, if all the attributes were perfectly falsely interacting, we might leave them alone, and the discriminative classification performance would not be affected. On the other hand, the discriminative classification performance would not be affected if we only picked a single attribute, since each one carries all the information. But for probabilistic classification, we also care about the accuracy of the predicted probability distribution, and replication of an attribute worsens the results, because the probability estimates tend towards extremes in the presence of replicated attributes. Finally, there may be several independent groups of falsely interacting attributes, and splitting them into subgroups would make sense. We can conclude that impassivity pays.

4.4.3 Conditional Interactions

Conditional interactions refer to situations where a multi-valued attribute A_1 interacts with a set of attributes $\mathcal{B}$ for some of its values, but does not interact or interacts falsely for its other values. Let us conjure up a domain to illustrate this situation, where $\mathcal{B} = \{A_2, A_3\}$. Note that the interaction between A_2 and A_3 is also dependent on the value of A_1:

A_1	A_2	A_3	$P(c_0)$	A_1	A_2	A_3	$P(c_0)$
0	0	0	1	2	0	0	1/25
0	0	1	0	2	0	1	1/7
0	1	0	0	2	1	0	3/11
0	1	1	1	2	1	1	3/5
1	0	0	0	3	0	0	2/5
1	0	1	1	3	0	1	8/11
1	1	0	1	3	1	0	6/7
1	1	1	0	3	1	1	24/25

For values $v_{A_1} \in \{0, 1\}$, A_1, A_2 and A_3 are truly interacting in a 3-XOR or 3-parity domain. However, for values $v_{A_1} \in \{2, 3\}$, attributes A_1, A_2 and A_3 are perfectly independent. It depends on the frequency of the values of A_1 whether this triple will be found to be a part of an interaction or not. We thus refer to attribute A_1 as a trigger attribute.

One way of resolving this issue is to split A_1 into two attributes, one being a binary $b_1 := a_1 \geq 2$, and the other a binary $b_2 := a_1 \pmod 2$. Then we can use the classifier

$$D(S(B_1), \{E[S(B_2, A_2, A_3)], V[E(S(B_2)), E(S(A_2)), E(S(A_3))]\}).$$

The switch function $D(s, \mathcal{S})$ chooses a model from $\mathcal{S}$, depending on the value of parameter function s. One can imagine the switch function as a simple conditional voting function. A learning algorithm which creates similar models is Kohavi's NBTree [Koh96]. The one-descriptor model $E(\mathcal{T}, C, S(A))$ is nothing but $D(S(A), \{E(\mathcal{T}, C)\} \times |\mathcal{D}_A|)$: if we do not use projection, we no longer need one-descriptor estimation, merely zero-descriptor estimation and the switch function.

Another variant of the conditional interaction is perhaps more frequent in real data: relevance (rather than dependence) of a certain attribute depends on the value of the trigger attribute. The second atribute affects the label only if the the trigger attribute has a certain value. This case is resolved with ordinary classification trees. With an NBTree, we may use a feature-selecting voting function, which would remove irrelevant attributes from certain models.

4.5 Instance-Sensitive Evaluation

If we limit ourselves to discriminative classifiers, the embodiment of interactions may take a simpler form. One attribute may *substitute* or *complement* another, even if do not study the interaction. Adding a complementary attribute improves the results, whereas a substitutable attribute does not provide additional help. We only need one from a set of mutually substitutable attributes.

We estimate their complementarity with a simple procedure for a domain with attributes $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$, and the label C. We construct n discriminative classifiers $d_1, d_2, \ldots, d_n$, where d_i is learned only with attributes $\mathcal{A} \setminus \{A_i\}$. We test their performance on the test set and construct the following contingency table for every pair of two discriminative classifiers d_i, d_j, each cell of which contains the number of instances in the test set corresponding to a given form of agreement between d_i and d_j:

Such a contingency table is also used for the McNemar's test, as described in [Die98, Eve77]. We can conceptually categorize the relations between attributes according to this list:

	d_j wrong	d_j correct
d_i wrong	n_{00}	n_{01}
d_i correct	n_{10}	n_{11}

- A *substitutable pair* of attributes provide the same information the corresponding contingency table is diagonal or close to diagonal, such as:

$$\begin{pmatrix} 10 & 0 \\ 0 & 9 \end{pmatrix}.$$

- In an *ordered pair* of attributes, the better attribute provides all the information that the worse attribute provides, while the worse attribute provides nothing extra. The contingency table is close to triangular:

$$\begin{pmatrix} 10 & 6 \\ 0 & 9 \end{pmatrix}.$$

- A *complementary* pair of attributes provides more information together than either attribute separately. The contingency table is neither diagonal nor triangular:

$$\begin{pmatrix} 8 & 5 \\ 4 & 9 \end{pmatrix}.$$

While we removed individual attributes to compute the contingency table, we could have approached the problem from another direction: we could have compared n classifiers, each of which was trained with its corresponding attribute alone. In such a case, we would probably not be able to correctly evaluate those examples which can only be understood in combination with other attributes.

This method obtains a considerable amount of information about the relationship between a pair of attributes by simply avoiding the customary averaging over all the instances in the test set. Instead, the method investigates the dependencies between individual attributes and instances. Although this method may appear similar to searching for interactions, it is unable to discover true interactions, because they require presence of both attributes. It can be easily extended for that purpose, but we will not investigate such extensions in this text.

Our classification of relationships between attributes is a valid method for comparing different classifiers. This is useful when we consider whether and how to integrate them in an ensemble, for example, it is beneficial to join the votes of two complementary classifiers. A number of other numerical measures of classifier diversity exist [KW01].

CHAPTER 5

Finding 3-Way Interactions

Engineers think equations approximate reality,
physicists think that reality approximates equations,
but mathematicians find no connection.

We could generate a massive number of classifiers, having some assume interactions and others not assume them, and eventually choose the best performing model. But we would desire a more sophisticated and less brute-force classifier selection algorithm which will examine the interactions selectively and efficiently by searching intelligently in the classifier space. Furthermore, the brute force method will not inform us about interactions in the domain.

Probes are heuristic methods which evaluate groups of attributes and estimate the level and type of their interaction as to uncover the interactions in a given domain. In *forward probing* we start with the assumption of no interactions, and iteratively build groups of interacting attributes.

Most interaction probes are based on simple predictors which only use a pair of attributes. The interaction effect is estimated with the improvement of the classifier resulting from segmentation operating on all attributes simultaneously. The predictors' quality can be evaluated either on the test or on the training set. We will refer to these as model probes. They provide reliable results, but one needs to assume the segmentation method, the base classification method and the evaluation function beforehand.

Association probes are based on statistical tests of conditional independence. These tests estimate the strength of an association, and then compute the likelihood of dependence given the strength and the number of instances. Association probes are not directly concerned with classifier quality. We warn the reader that the presence of conditional dependence does not necessarily indicate presence of a significant interaction.

Somewhat midway between wrapper probes and association probes, we may define an information-theoretic probe, which approximates the actual evaluation functions.

Several constructive induction algorithms [Dem02, Zup97] evaluate the benefit of applying an optimizing segmentation function to subsets of attributes to determine whether constructing a new attribute using these attributes is justified. The new attribute is supposed to replace and simplify the constituent attributes, but not relinquish information. These algorithms can be considered to be another family of interaction probes with several interesting properties. They can also be seen as optimizing segmentation functions.

The objective of the remainder of this chapter will be to examine and compare these probes. We will not attempt to evaluate the probes. Rather, we will examine similarities between probes, their sensitivity to various artifacts, and the methodology of using them. Evaluation is left for coming chapters.

5.1 Wrapper Probes

Wrapper probes attempt to predict a classifier's quality by evaluating various variants of the classifier trained on the portion of the training set of instances. Hopefully, the conclusions will help us determine the best variant of the classifier for the test set. Generally, wrapper probes are greedy search algorithms built around the pragmatic interaction test, and a certain resolution function.

One variant joints two attributes X and Y into a new attribute $\overline{XY}$ whose domain is a Cartesian product of constituent attribute domains $\mathcal{D}_{\overline{XY}} = \mathcal{D}_X \times \mathcal{D}_Y$. The probe's estimate is the quality gain of a naïve Bayesian classifier with the joined attribute over the default NBC with separate attributes. We define it as

$$QG := q(NBC(\overline{XY}, Z, W)) - q(NBC(X, Y, Z, W)),$$

for some evaluation function q. No effort is made to optimize segmentation of the joint attribute $\overline{XY}$.

Pazzani [Paz96] built two algorithms around such a probe, referring to the first as 'forward sequential selection and joining,' or FSSJ. FSSJ starts with a classifier trained using an empty set of attributes. In each iteration, it considers adding a new attribute to the set, or joining an attribute with one existing attribute already in the set. The chosen operation is the one that maximizes either the interaction or the attribute gain, if only the gain is positive. In case joining was chosen, it replaces the two constituent attributes with the new joint attribute.

He notes another algorithm, "backward sequential elimination and joining," which starts with the basic naïve Bayesian model, and for each attribute considers two operations: deleting the attribute, or joining two used attributes. Generally, his results with BSEJ are better than those with FSSJ, although BSEJ does not dominate FSSJ.

There are several variants of wrapper probes. For example, instead of the holistic quality gain, which includes other attributes into consideration $q(NBC(\overline{XY}, Z, W)) - q(NBC(X, Y, Z, W))$, we can simply myopically focus on $q(NBC(\overline{XY})) - q(NBC(X, Y))$. We may relax the wrapper methodology, and improve performance by evaluating the classifier on the test set. Although joining attributes will always improve performance, we may assume that large gains in performance truly indicate an interaction.

5.2 Constructive Induction

Function decomposition [Zup97] comes in two flavors: the noise-tolerant minimal-error (MinErr) decomposition, and the determinism-assuming minimal-complexity (MinComplexity) decomposition. Recursive application of function decomposition suffices to construct classifiers. The fundamental feature of function decomposition is the merging of attribute value pairs, which we call *attribute reduction* and is a type of a segmentation function. The core idea of attribute reduction is to merge similar attribute values, while distinguishing dissimilar attribute values. In addition to the attribute reduction mechanism, we use a heuristic estimate, a *partition measure*, to determine which pairs of attributes to join in a Cartesian product before reducing it.

Minimal-complexity attribute reduction considers more than just the label distributions: we can merge those value pairs which are not necessary to perfectly discern the class. This means that if we can discern the class of all the instances covered by a particular duo of value pairs with other attributes alone, these two value pairs can be merged. Our next objective is to maximize the number of such mergers, and we achieve it with graph coloring. The new attribute's values corresponds to colors. The fewer the colors, the better the constructed attribute.

Minimal-error attribute reduction is similar to clustering in sense that merging is performed on the basis of label distributions, greedily merging the closest pair of attribute value pairs. However, clustering performs multiple merges on the basis of the same dissimilarity matrix, whereas minimal-error decomposition performs only a single nearest-pair merge and updates the matrix after that. m-error estimate determines the number of mergers to perform, and that determines the number of clusters: we perform only those merges that reduce the m-error estimate. The value of m is usually determined by means of wrapper methods, e.g., with internal (training set) cross validation.

Although minimal-error decomposition is somewhat similar to Kramer's clustering algorithm [Kra94], it must be noted that Kramer's algorithm is computing a single label distribution for a attribute value pair. On the other hand, minimal-error decomposition computes a label distribution for every value tuple of all the other attributes. As with minimal-complexity attribute reduction, we estimate the similarity of label distributions *given* the values of all other attributes, which we refer to as *context* attributes. This way, we prevent destroying information, which could be useful in later resolutions. Decomposition is thus able to handle multi-way interactions while only resolving a small number of attributes at once. However, context is a handicap in domains with falsely interacting attributes, as Demšar observed in [Dem02].

Although function decomposition algorithms can operate with tuples of bound attributes, and not merely with pairs, the combinatorial complexity of testing the quality of all the possible 3-attribute constructions is excessive. The method thus uses heuristic probes to pick the pair of attributes that yielded the best constructed attribute. This attribute substitutes the original pair of attributes. The procedure terminates when only a single feature remains, and not when there are no more interactions.

The probe values for minimal-error decomposition is the total error reduction obtained with value merging, if the expected error is estimated with the m-probability estimate. For minimal-complexity decomposition, [Zup97] notes several possible heuristics, but the most frequently chosen one is based on the total number of segments obtained when losslessly decomposing a group of attributes. We will later investigate whether the value of these probes has anything to do with interactions.

5.3 Association Probes

Another possible definition of interactions is based on the notion of dependence and independence. This may be more appealing as we seem not to be tied to a classifier evaluation function or to a particular type of classifiers. In Sect. 3.2.1, we noted that there is no interaction if the attributes are conditionally independent. Generally, a k-way interaction exists if there is a dependency between k attributes which cannot be broken down completely into multiple dependencies, each of which would contain fewer than k attributes. One of the attributes is the label.

5.3.1 Cochran-Mantel-Haenszel Statistic

We can perform a Cochran-Mantel-Haenszel χ^2 test of the null hypothesis that two nominal variables are conditionally independent in each class, assuming that there is no 4-way (or higher) interaction. The details of the generalized Cochran-Mantel-Haenszel (CMH) statistic are intricate, and we refer the reader to [Agr90]. To prevent singular matrices in computations, we initially fill the contingency table, which measures the number of instances in the training data with those attribute values, with a small value (10^{-6}), as recommended in [Agr90]. Our implementation was derived from the library 'ctest' by Kurt Hornik, P. Dalgaard, and T. Hothorn, a part of the R statistical system [IG96].

As the output of this probe, we used the p-value of the statistic, which can be understood as a means of normalizing as to remove the influence of the number of attribute values that determine the number of degrees of freedom. It must be noted that p-value should not be equated with probability of the hypothesis. For many statistical tests, p-values are informative only when they are very low, else they may even be even randomly distributed.

5.3.2 Semi-Naïve Bayes

Semi-naïve Bayesian classifier [Kon91] attempts to merge those pairs of attribute value pairs that have similar label probability distributions. In contrast to Pazzani's algorithm, which joins attributes, SNB only joins attribute value pairs. It considers merging all pairs of attribute values, without restricting itself to a particular pair of attributes.

The theorem of Chebyshev gives the lower bound on the probability that the relative frequency f of an event after n trials differs from the factual prior probability p less than ϵ:

$$P(|f-p| \leq \epsilon) > 1 - \frac{p(1-p)}{\epsilon^2 n}$$

SNB recommends merging of value j_i of attribute J and value k_l of attribute K if:

$$1 - \frac{1}{N_{j_i k_l}\epsilon^2} \geq \frac{1}{2}$$

$$\epsilon = \sum_{j=1}^{m} P(c_j) \left| P(c_j|j_i k_l) - \frac{P(c_j|j_i)P(c_j|k_l)}{P(c_j)} \right|$$

Here, $N_{j_i k_l}$ is the number of instances in the training set, for which $J(\mathbf{i}) = j_i \wedge K(\mathbf{i}) = k_l$.

The squared part of the equation measures the difference between the label probability distributions of j_i and k_l. The factual probability is taken to be equal to 0.5. In SNB, m-probability estimation is used for estimating conditional and class probabilities.

Although SNB was designed as a feature constructor, it can also be used as a probe for estimating the level of interaction between whole attributes. For this, we compute the sum of merging probabilities for a pair of attributes over all their value pairs, normalized by the number of attribute value pairs that actually appear in the training data:

$$\mathrm{I}_{\mathrm{SNB}}(J,K) = 1 - \frac{\sum_{i\in\mathcal{D}_J}\sum_{l\in\mathcal{D}_K} 1 - 1/N_{j_ik_l}\epsilon^2}{|\mathcal{D}_J||\mathcal{D}_K|}$$

Although the formula appears complex, most of the complexity emerges because the values are re-scaled and normalized several times. However, this re-scaling proves to be useful, because the uniformity of the scores improves the clarity of results obtained with clustering and other methods of analysis.

5.4 Information-Theoretic Probes

We will focus on the information-theoretic notion of entropy, for which there are rigorous mathematical tools and bounds. However, like wrapper probes, we retain the concepts of a segmentation function and an evaluation function, even if they are somewhat camouflaged.

This way, we will be able to arrive at relatively simple, efficient, and illuminating closed-form probes for interactions. They involve evaluating on the training set with KL divergence as the evaluation function, and learning the joint probability distribution rather than merely the label probability distribution. The segmentation function is the trivial Cartesian product.

One can notice that all the formulae in this section are also appropriate for generative learning. Although it appears we do, we in fact do not set any attribute apart as the label. But since all our formulae are based on Kullback-Leibler divergence, we could easily use another non-generative evaluation function instead, perhaps giving up some of the elegance and efficiency.

5.4.1 3-Way Interaction Gain

Information gain of a single attribute A with the label C [HMS66], also known as *mutual information* between A and C [CT91], is defined this way:

$$\begin{aligned}
\mathrm{Gain}_C(A) &= I(A;C) \\
&= \sum_{a,c} P(a,c)\log\frac{P(a,c)}{P(a)P(c)} \\
&= H(C) + H(A) - H(\overline{AC}) \\
&= H(A) - H(A|C).
\end{aligned} \tag{5.1}$$

Mutual information is a special case of KL divergence in evaluating the approximation of the joint probability with the product of marginal probabilities: $I(X;Y) = I(Y;X) = D(P(x,y)||P(x)P(y))$. The lower the KL divergence, the better the two attributes can be modeled with the assumption of independence. Therefore, mutual information should not be seen as anything else than a particular type of evaluation function (KL divergence) of a generative model (predicting both the attribute and the class) with the assumption of independence. Evaluation takes place on the training set, rather than on the test set, so it is not obvious whether the result is significant or not. Furthermore, multiplication ($\times$) in $P(x)P(y) = P(x) \times P(y)$ is just one specific function we can use for approximating $P(x, y)$. There are other simple functions. We see that information theory is also based on assumptions about evaluation functions and about model functions.

Conditional mutual information [CBL97, WW89] of a group of attributes is computed by subtracting the entropy of individual attributes from the entropy of Cartesian product of the values of all the attributes. For attributes A, B and a label C, we can use the following formula:

$$\begin{aligned} I(A;B|C) &= \sum_{a,b,c} P(a,b|c) \log \frac{P(a,b|c)}{P(a|c)P(b|c)} \\ &= H(A|C) + H(B|C) - H(\overline{AB}|C), \end{aligned} \tag{5.2}$$

where $H(X)$ is entropy, and $\overline{AB}$ is the Cartesian product of values of attributes A and B. Each attribute itself can be evaluated by its quality as a predictor, and the joint entropy approach tries to separate the actual contribution of an interaction over independent contributions of separate attributes. We can also express conditional mutual information through KL divergence: $I(X;Y|C) = D(P(x,y|c)||P(x|c)P(y|c))$. Again, a greater value will indicate a greater deviation from independence.

Interaction gain for 3-way interactions can be defined as:

$$\begin{aligned} IG_3(ABC) &:= I(\overline{AB};C) - I(A;C) - I(B;C) \\ &= \mathrm{Gain}_C(\overline{AB}) - \mathrm{Gain}_C(A) - \mathrm{Gain}_C(B), \end{aligned} \tag{5.3}$$

and can be understood as the decrease in entropy caused by joining the attributes A and B into a Cartesian product. The higher the interaction gain, the more information we gained by joining the attributes in a Cartesian product. Interaction gain can be negative, if both A and B carry the same information. Information gain is a 2-way interaction gain of an attribute and the label: $IG_2(A, C) = \mathrm{Gain}_C(A)$, just as dependence between two attributes is nothing else than a 2-way interaction.

We can transform (5.2) by abolishing conditional probabilities and conditional entropy into:

$$\begin{aligned} I(A;B|C) &= H(A|C) + H(B|C) - H(\overline{AB}|C) \\ &= (H(\overline{AC}) - H(C)) + (H(\overline{BC}) - H(C)) - (H(\overline{ABC}) - H(C)) \\ &= H(\overline{AC}) + H(\overline{BC}) - H(C) - H(\overline{ABC}). \end{aligned}$$

We can also work backwards from the definition of interaction gain, rearranging the terms:

$$\begin{aligned} IG_3(ABC) &= (H(\overline{AB}) + H(C) - H(\overline{ABC})) \\ &\qquad - (H(A) + H(C) - H(\overline{AC})) - (H(B) + H(C) - H(\overline{BC})) \\ &= H(\overline{AB}) + H(\overline{AC}) + H(\overline{BC}) \\ &\qquad - H(\overline{ABC}) - H(A) - H(B) - H(C). \end{aligned} \tag{5.4}$$

The value of interaction gain is the same if we substitute the label with one of the attributes. Therefore we neglect distinguishing the label. Earlier, we only investigated interactions which included the label: such interactions are most interesting when studying classification problems.

$$\begin{aligned} IG_3(ABC) &= H(\overline{AB}) - H(A) - H(B) + I(A;B|C) \\ &= I(A;B|C) - I(A;B) \\ &= D(P(a,b|c)||P(a|c)P(b|c)) - D(P(a,b)||P(a)P(b)). \end{aligned} \tag{5.5}$$

If we nevertheless focus on one attribute (C) and then investigate the interplay between the attributes (A, B), we notice two parts, *dependency* measured by $I(A;B)$, and *interaction* $I(A;B|C)$. Both dependency and interaction are always zero or positive. *When the level of interaction exceeds the level of dependency, the interaction is true. When the opposite is true, the interaction is false.* Of course, a pair of attributes may have a bit of both, an example of which are the conditional interactions, and this is avoided by breaking multi-valued attributes into dummy one-valued attributes.

When there are only two attributes (A, B) with a label (C), and if we assume that both attributes are relevant, there are only two possible decompositions of the joint probability distribution: (ABC) and (AC, BC). Comparison between (ABC) and (AC, BC) with the assumption of independence of (AC, BC) is the essence of our (5.5). We view $I(A;B)$, a measure of sharing between A and B, but also as a measure of sharing between (AC) and (BC), even if this practice could be disputed. Although there are several possible approaches, we will not try to separate individual attributes' contributions to accuracy.

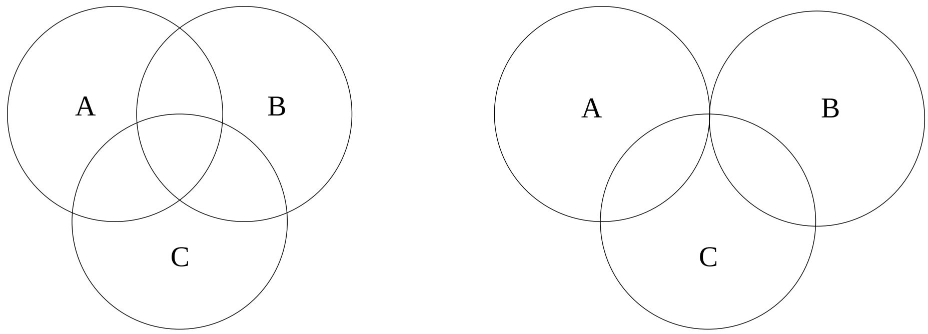
A true interaction exists merely if the conditional mutual information $I(A;B|C)$ is greater than what we would expect from $I(A;B)$. If $IBC(A)$ and $IBC(B)$ contain the same information, interaction gain is negative. This indicates the possibility of a false interaction according to the pragmatic criterion. For a perfectly true interaction we know that $I(A;B) = I(B;C) = I(A;C) = 0$, and a positive interaction gain clearly indicates the presence of a true interaction.

Generalizing Interaction Gain

It is possible to construct a multitude of interaction gain generalizations by varying the learning mode, the evaluation function, and the predictive model. It is certain that some of such generalizations will sometimes be better for some selection of data. Interaction gain is based on generative learning, the Kullback-Leibler divergence computed on the training data, and probabilistic prediction with and without the assumption of conditional independence. As such, it should be seen as a heuristic interaction probe. It has an important ability of distinguishing true from false interactions.

We could use other measures of association, several of which were mentioned in Sect. 3.2. A type of measures of association are *attribute impurity measures*, intended for feature selection in the context of classification, most of which are not generative. Several non-myopic attribute measures consider attributes other than the one evaluated. For example, Relief-like measures [Šik02] will reduce the worth of duplicated attributes. This often helps improve classification accuracy when the measure is used for feature selection, but such measures are inappropriate for evaluating interactions.

Some attribute impurity measures, e.g., the gain ratio, consider the number of attribute values, reducing the worth of an attribute proportionally to the number of its values. When



(a) (b)

Figure 5.1: A Venn diagram of three interacting attributes (a), and of two conditionally independent attributes plus a label (b).

we apply such a measure on an attribute obtained by resolving an interaction with the Cartesian product, the results will be strongly distorted: the complex joint attribute's worth will be excessively reduced.

5.4.2 Visualizing Interactions

The simplest way of explaining the significance of interactions is via the cardinality of the set metaphor. The definition for information gain from (5.1) is $I(A;B) = H(A)+H(B)-H(AB)$. This is similar to the cardinality of the set as computed using the Bernolli's inclusion-exclusion principle [Wei02]: $|A\cap B| = |A|+|B|-|A\cup B|$. The total information content of attributes A and B together is $-H(AB)$, of A alone it is $-H(A)$, and of B, $-H(B)$. Thus, $I(A;B)$ corresponds to the intersection between A and B. Note the sign reversal because of entropy.

To draw the analogy further, interaction gain, as defined in (5.5) and drawn in (a) on the left of Fig. 5.1, corresponds to $|A\cap B\cap C| = |A| + |B| + |C| - |A\cup B| - |B\cup C| - |A\cup C| + |A\cup B\cup C|$. Essentially, cardinality of an attributes' intersection corresponds to their interaction gain. Cardinality so computed may be negative, as noticed by [Ved02]. We have suggested and will show that this negativity provides useful information even if the pretty metaphor is ruined. Unfortunately, merely extending this idea to four sets no longer provides a useful heuristic.

As a side note, the domain as assumed by the naïve Bayesian classifier is Fig. 5.1(b). The entropy of C, as estimated by the naïve Bayesian classifier is $H[P(A|C)P(B|C)] = H(AC) + H(BC) - 2H(C)$, as compared with $H[P(AB|C)] = H(ABC) - H(C)$. An alternative way of defining a concept similar to interaction gain is by comparing $H(ABC)$ with $H(AC)+H(BC)-H(C)$ (we added $H(C)$ to both expressions). Such an approach might open a better course to generalizing interaction gain to arbitrary k-way interactions, but it requires assigning a special role to one of the attributes.

One of the pleasant properties of the set-theoretic metaphor is that it is independent from any notion of conditional probability. Therefore, we assume no ordering of attributes, and we do not separate causes from effects. Causality could only be an expression of temporal ordering of events, in sense that causes temporally precede the effects. We could

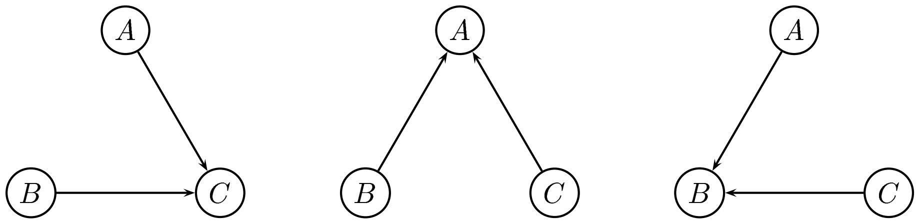


Figure 5.2: Three equivalent Bayesian networks for the XOR domain.

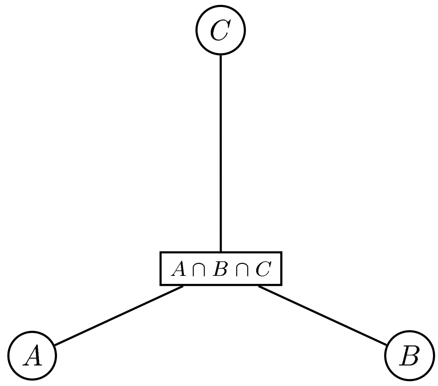


Figure 5.3: An interaction diagram for the perfect interaction between A, B and C, e.g., in the XOR domain.

pretend that effects are better predictable than causes, but quality of predictions may be irrelevant: in the information-theoretic framework it is always symmetric.

For the XOR domain $c := a \neq b$, viewed as a generative learning problem attempting to approximate $P(a, b, c)$, there are three appropriate Bayesian networks, as illustrated in Fig. 5.2. Although all these models correctly describe the joint probability distribution, the direction of edges is meaningless and the edges may be misleading because there are no dependencies between either pair of vertices.

We can use hypergraphs $G = (V, E)$ for visualizing the interactions in the domain, where a hyperedge $H = \{A, B, C\}$ exists iff $IG_3(ABC) > 0$. There are many ways of visualizing hypergraphs, either by using a different color for each hyperedge, or by using polygons in place of hyperedges, or by drawing polygons around vertices connected by a hyperedge. Or, instead of hypergraphs, we may use ordinary graphs with special 'interaction' vertices for each hyperedge or interaction, which are created when using edges alone would be ambiguous. We mark interactions either with dots, or with labeled rectangular nodes. Figs. 5.3–5.6 present graphical depictions of various types of interactions. Further variants of interaction visualizations appear in Ch. 7.

5.4.3 Related Work

Quantum Information Theory

In the field of quantum information theory, Vedral [Ved02] uses relative entropy as a quantification of distinguishability of physical states. He proposes a generalization of rel-

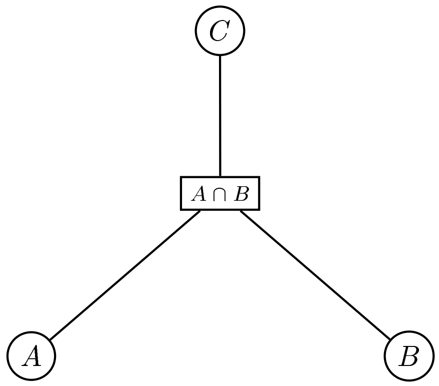


Figure 5.4: An interaction diagram rendered for a false interaction between A and B.

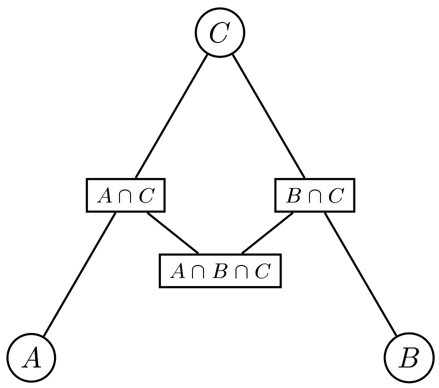


Figure 5.5: An interaction diagram rendered for an ordinary interaction with independent attributes A and B.

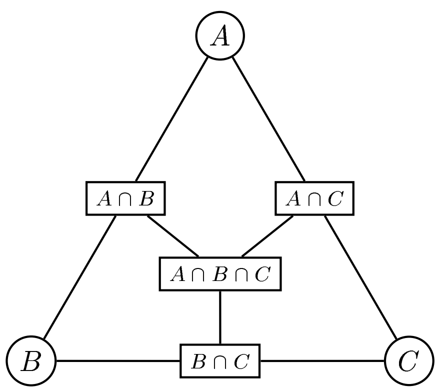


Figure 5.6: A full interaction diagram rendered for three sets, with tagged intersections and all possible interactions. This is the most complex situation.

ative entropy that measures the divergence between the joint probability distribution and its approximation based on the assumption of independence between all attributes, e.g., $I(A;B;C) := D(P(a,b,c)||P(a)P(b)P(c))$. He argues that the interaction gain, although a natural generalization inspired by the Bernoulli's inclusion-exclusion principle for computing a union of sets, is inappropriate because it may be negative. We now know that it is negative when the interaction is false. His definition of generalized relative entropy can never be negative. We can represent his generalization with entropy:

$$\begin{aligned} D(P(a,b,c)||P(a)P(b)P(c)) &= \sum_{a,b,c} P(a,b,c) \log \frac{P(a,b,c)}{P(a)P(b)P(c)} = \\ & -\sum_{a,b,c} P(a,b,c) \log P(a,b,c) + \sum_{a} \sum_{b,c} P(a,b,c) \log P(a) \\ & + \sum_{b} \sum_{a,c} P(a,b,c) \log P(b) + \sum_{c} \sum_{a,b} P(a,b,c) \log P(c) \\ & = H(A) + H(B) + H(C) - H(\overline{ABC}) \end{aligned} \tag{5.6}$$

Game Theory

As an aside, we may now define interaction index we first mentioned in Sect. 3.6, as described by [GMR00, GMR99]. The original definitions from [GR99] differ in a minor way, using $\subset$ instead of $\subseteq$. There are two interaction indices for a coalition $\mathcal{S}$, $I_S^v(\mathcal{S})$ for the Shapley value, and $I_B^v(\mathcal{S})$ for the Banzhaf value. The set $\mathcal{N}$ contains all the players, while the set $\mathcal{T}$ acts as an iterator for averaging over all possible coalitions.

$$I_S^v(\mathcal{S}) := \sum_{\mathcal{T} \subseteq \mathcal{N} \setminus \mathcal{S}} \frac{(|\mathcal{N}| - |\mathcal{T}| - |\mathcal{S}|)!|\mathcal{T}|!}{(|\mathcal{N}| - |\mathcal{S}| + 1)!} \sum_{\mathcal{L} \subseteq \mathcal{S}} (-1)^{|\mathcal{S}|-|\mathcal{L}|} v(\mathcal{L} \cup \mathcal{T}) \tag{5.7}$$

$$I_B^v(\mathcal{S}) := 2^{|\mathcal{S}|-|\mathcal{N}|} \sum_{\mathcal{T} \subseteq \mathcal{N} \setminus \mathcal{S}} \left(\sum_{\mathcal{L} \subseteq \mathcal{S}} (-1)^{|\mathcal{S}|-|\mathcal{L}|} v(\mathcal{L} \cup \mathcal{T}) \right). \tag{5.8}$$

We may adopt negative value of entropy as the value function v. As we remember from Sect. 2.6.2, the greater the entropy, the lower the expected earnings from a betting game. If we also assume that $\mathcal{N} = \mathcal{S}$, the Banzhaf interaction index simplifies to:

$$I_B^H(\mathcal{S}) = -\sum_{\mathcal{L} \subseteq \mathcal{S}} (-1)^{|\mathcal{S}|-|\mathcal{L}|} H(\mathcal{L}).$$

For $S = \{A, B\}$, $I_B^H(AB) = H(A) + H(B) - H(\overline{AB}) = IG_2(AB)$, while for $S = \{A, B, C\}$, $I_B^H(ABC) = H(\overline{AB}) + H(\overline{AC}) + H(\overline{BC}) - H(A) - H(B) - H(C) - H(\overline{ABC}) = IG_3(ABC)$. Unfortunately, using I_B^H for 4-way interaction gain may no longer be useful, as preliminary experiments indicate that other than 3-parity, correlated attributes may also yield positive values, in spite of the correlated attributes being conceptually an epitome of false interactions.

The final is the chaining interaction index [MR99], defined for a set of *maximal chains* $C(\mathcal{N})$, where a maximal chain $\mathcal{M}$ of a Hasse diagram $H(\mathcal{N})$ is an ordered collection of $|\mathcal{N}| + 1$ nested distinct coalitions:

$$\mathcal{M} = \left(\emptyset = \mathcal{M}_0 \subsetneq \mathcal{M}_1 \subsetneq \mathcal{M}_2 \subsetneq \cdots \subsetneq \mathcal{M}_{|\mathcal{N}|-1} \subsetneq \mathcal{M}_{|\mathcal{N}|} = \mathcal{N} \right).$$

Each maximal chain corresponds to a permutation of elements of $\mathcal{N}$. Let $\mathcal{M}^{\mathcal{S}}$ be the the minimal coalition belonging to $\mathcal{M}$ that contains $\mathcal{S}$. The chaining interaction index I_R^v is then an average value of a chain over all the maximal chains:

$$I_R^v(\mathcal{S}) = \frac{1}{|\mathcal{N}|!} \sum_{\mathcal{M} \subseteq C(\mathcal{N})} \delta_{\mathcal{S}} v(\mathcal{M}^{\mathcal{S}}), \qquad \emptyset \neq \mathcal{S} \subseteq \mathcal{N}. \tag{5.9}$$

Here $\delta_{\mathcal{S}} v(\mathcal{T})$ is the $\mathcal{S}$-derivative of v at $\mathcal{T}$, defined recursively as $\delta_{\mathcal{S}} v(\mathcal{T}) = v(\mathcal{T}) - v(\mathcal{T} \setminus \mathcal{S})$. It can be shown that

$$I_R^v(\mathcal{S}) = \sum_{\mathcal{T} \subseteq \mathcal{N} \setminus \mathcal{S}} \left(|\mathcal{S}| \frac{(|\mathcal{N}| - |\mathcal{S}| - |\mathcal{T}|)!(|\mathcal{S}| + |\mathcal{T}| - 1)!}{|\mathcal{N}|!} \right) (v(\mathcal{T} \cup \mathcal{S}) - v(\mathcal{T})). \tag{5.10}$$

We may again use negative entropy as a value function. Because conditional entropy is calculated as $H(A|C) = H(\overline{AC}) - H(C)$, we can express $H(\mathcal{T}) - H(\mathcal{T} \cup \mathcal{S}) = -H(\mathcal{S}|\mathcal{T})$. Therefore, the more dependence there is on average between $\mathcal{S}$ and other players, the higher value will I_R^H achieve.

CHAPTER 6

Practical Search for Interactions

Why think? Why not try the experiment?

John Hunter

Now that we have defined interactions theoretically, we will focus on investigating the nature of interactions in true data sets. The objective of this chapter will be to study probing techniques for discovering interactions in data. We also explore the relationship between interactions and domain structure, as designed by humans. We investigate the interplay between the information-theoretic interaction probes and the pragmatic definition of interactions.

Instead of performing superficial statistics on a large set of domains, we have chosen only four characteristic domains, but our analysis will be thorough. Two domains are noiseless uniform samples of manually constructed hierarchical DECMAC decision models, developed with the DEX software [BR90]: 'car' [BR88], and 'nursery' [OBR89]. We used a real medical data set 'HHS', contributed by Dr. D. Smrke from the University Clinical Center in Ljubljana, on the base of which an attribute structure was constructed by a domain expert, and described in [ZDS$^+$01]. Finally, we created an artificial domain which integrates the concepts from Chap. 4. All the attribute structures are illustrated in Fig. 6.1.

The DECMAC methodology is based on constructing higher-level features from primitive attributes, this way reducing the number of attributes in the domain. Ultimately, we create a function from a small number of higher-level attributes to the label value.

Our artificial domain was constructed as to contain all the types of true and false interactions: a 3-XOR, an ordinary interaction (OR), two noisy measurements, two mutually exclusive events, and a trigger attribute in two 3-way conditional interactions, one with conditioned dependence and one with conditioned relevance. We added two random attributes. The class value is stochastically sampled from the probability distribution obtained by assuming that all these binary sub-problems are independent.

The difference between our 'artificial' domain and the DECMAC structures is that our domain is probabilistic, whereas the DECMAC structures are deterministic. Whereas

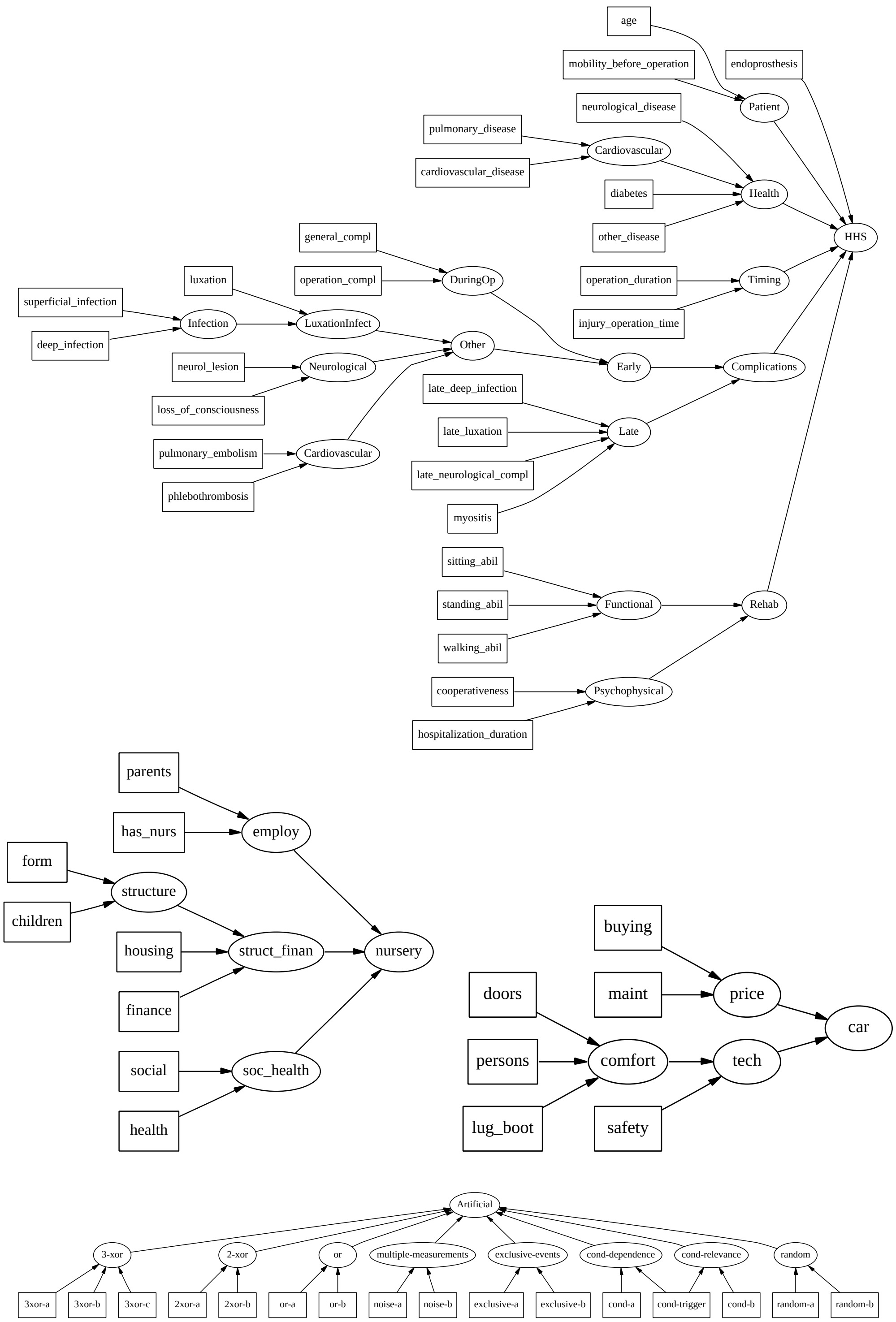


Figure 6.1: Hand-made attribute structures for the 'car' (middle-right), 'nursery' (middle-left), 'HHS' (top), and 'Artificial' (bottom) domains. Basic attributes are in rectangles, constructed attributes are ellipses.

non-primitive attributes in DECMAC structures are perfectly described by a function from primitive attributes, in 'artificial' the function involves randomness. Finally, we have no knowledge about the nature of 'HHS'.

For the 'artificial' and 'HHS' domain, we used the usual 10-fold cross-validation. On the 'car' and 'nursery' domains, we trained our classifiers on a randomly sampled proportion of the data, and tested the classifiers on the rest. Because the HINT classifier would achieve perfect classification accuracy on 90% of the data, we instead took smaller samples. For 'car' the proportion was 20% with 20 repetitions (HINT: ~95%, C4.5: ~90%), and for 'nursery' the proportion was 8% with 25 repetitions (HINT: ~98%, C4.5: ~95%). At this proportion C4.5 achieved worse results than HINT.

We have selected the negative value of the venerable KL divergence $D(\hat{P}||Q)$, defined and explained in Sect. 2.6.2 to play the role of our evaluation function, where $\hat{P}$ is an approximation to the true probability for an instance, and Q is the probabilistic classifier's output. When we test a classifier on a set of instances, we compute the divergence for each instance separately, and average the divergence over all the instances. In the graphs the top and right edges of the graphs consistently indicate positive qualities, and because low divergence is good and high divergence is bad, we negate it before plotting.

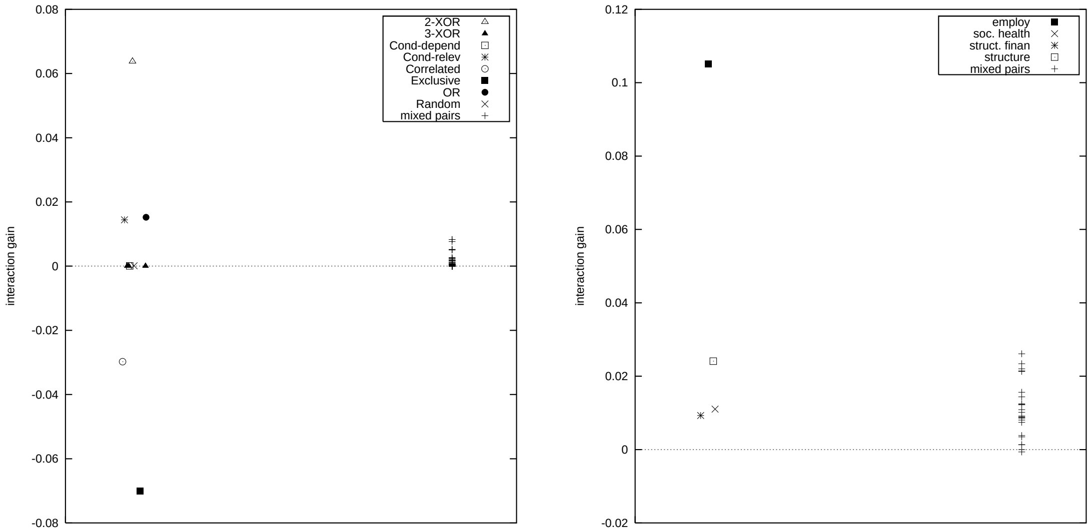
KL was chosen primarily because it offers greater sensitivity than classification accuracy: KL will distinguish between the classifier which predicted the actual class with $p = 0.9$ and one that predicted it with 0.51, while classification accuracy will not. Also, KL will more fairly penalize a classifier that assigned the actual class $p = 0.49$, while classification accuracy will penalize it as much as if it offered $p = 0$. The benefits of greater sensitivity in evaluation functions are discussed in [LZ02], although they refer to area under ROC (aROC). In contrast to aROC, Kullback-Leibler divergence is simpler, and is otherwise the most frequently used distance measure between probability distributions.

6.1 True and False Interactions

Our first exploration will focus on distinguishing true from false interactions. A heuristic measure which is especially suitable for this purpose is interaction gain from Sect. 5.4. In this chapter we will only use 3-way interaction gain, $IG_3(ABC)$, defined and explained in Sect. 5.4.1.

The ability of this heuristic to distinguish between true and false interactions is examined on domains 'artificial' and 'nursery' in Fig. 6.2. In 'artificial', neighborhood of attributes is associated with them being in an interaction of known type, whereas it is unknown whether the neighborhood in the human-designed structure of 'nursery' has anything to do with interactions.

For the 'artificial' domain, it can be seen that interaction gain properly determines whether pairs of attributes 3-interact with the label, and how. Most of the non-interacting attribute pairs' interaction gain is close to 0, as well as for the random pair of attributes. 3-way interaction gain is unable to detect the conditional relevance interaction, where the dependence between two attributes only occurs at certain condition attribute values. There is no improvement in joining pairs attributes that participate in the 4-way parity interaction with the label. However, the clearest aspect of this figure illustrates the ability of interaction gain to distinguish between true and false interactions: true interactions yield positive interaction gain, and false interactions yield negative interaction gain.



'artificial' 'nursery'

Figure 6.2: Analysis of interaction gain: on the left in each graph there are pairs of attributes neighboring in the attribute structure, jittered horizontally to prevent overdraw. On the right there are the non-neighboring attributes. On the vertical axis, we represent interaction gain.

In the 'nursery' domain, the interaction between attributes of only one concept is standing out. For other concepts, interaction gain is non-zero but not large. It is striking that there are virtually no false interactions, and this is because random sampling from the artificially generated nursery domain prevents them from showing up, even if they existed in the natural data from which the domain was constructed. The 'car' domain proved to be quite similar to the 'nursery'.

The most interesting aspect of the 'HHS' domain is that instances are natural rather than generated from the attribute structure. We divide the attribute pairs in two groups, the attributes that are part of the same concept, and attribute pairs where individual attributes belong to different concepts. The result is visualized in Fig. 6.3. It can be noticed that with respect to interactions, the structure is either not far from being arbitrary, or the interaction gain is inappropriate as a probe in this domain. There are also virtually no true interactions, but there are many false interactions.

6.2 Classifier Performance and Interactions

6.2.1 Replacing and Adding Attributes

We now focus on the effect of joining attributes to classification accuracy. First of all, we will investigate the relationship between the joint attribute replacing or complementing the original attributes. With NBC, it is customary to replace attributes [Paz96, Kon91], while for loglinear models [Agr90] we leave the original attributes in place. The fitting algorithms for loglinear models are optimizing, rather than estimating, but it is generally considered a bad practice to add additional dependence in NBC. For loglinear models, it is

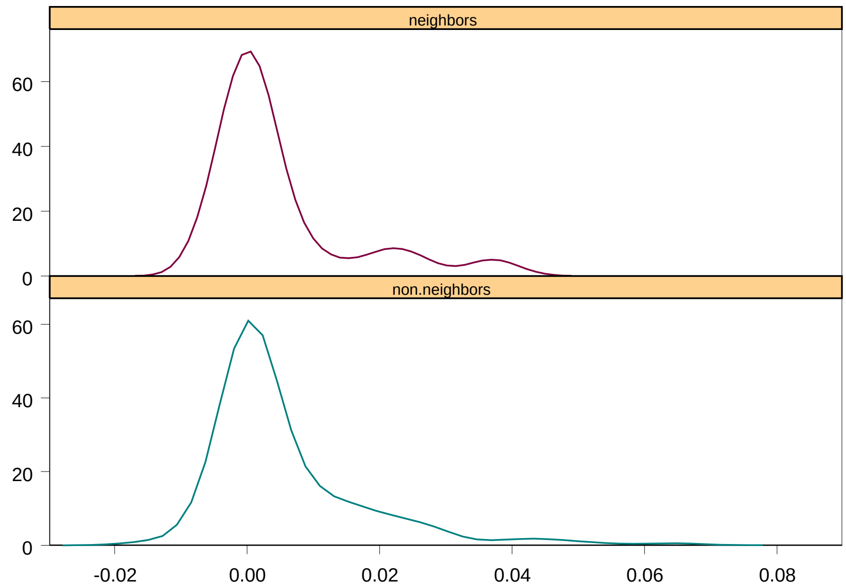


Figure 6.3: Interaction gain frequency distribution for the 'HHS' domain. The top distribution refers to the interaction gain for pairs of attributes which belong to the same concept, and the bottom distribution refers to pairs of attributes not belonging to the same concept.

very interesting to look at the individual contribution of an attribute to the classification, separate from its contribution which is a part of the interaction.

We will review the changes in evaluation function with respect to both strategies with joined attributes. If the base classifier is $IBC(X, Y, Z)$, the joint domain with replacement is $IBC(XY, Z)$, and with addition it is $IBC(XY, X, Y, Z)$. The what interests are the improvements in classification performance. For replacement, it is $r_r^{XY} = q(IBC(XY, Z)) - q(IBC(X, Y, Z))$ and for addition, it is $r_a^{XY} = q(IBC(XY, X, Y, Z)) - q(IBC(X, Y, Z))$.

The results can be seen in Fig. 6.4. Most concepts are not dependent, and joining them worsens the classifier quality. It is perhaps not surprising that there are relatively few significant interactions. Also, it is not a surprise that r_a rarely exceeds r_r, except for the 'nursery' and 'car', which do not have any correlated attributes because the domain is sampled.

On the 'artificial' domain, both true interactions are found to be useful. Certain complex interactions (3-XOR, conditional dependence) cannot be captured with the limited device of joining two attributes. A most interesting observation is that resolving false interactions may either yield an improvement or a deterioration: joining the exclusive pair of attributes improves the results, while joining the correlated but noisy pair of attributes worsens the results. This implies that a special method for handling false interactions could be useful.

Especially interesting is that, with several exceptions, joining attributes which are part of human-designed concepts did not improve the classification results. It appears that the attribute neighborhood in the attribute structure does not coincide with significant interactions, and we will now focus on the nature of attribute structure.

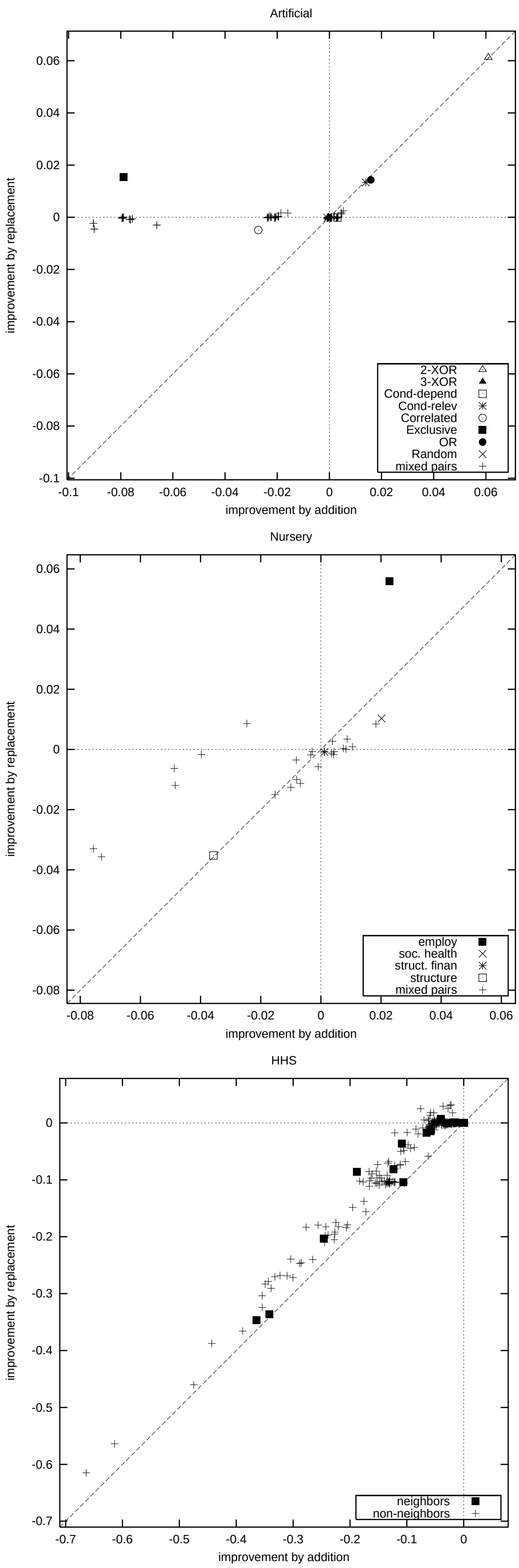


Figure 6.4: These graphs visualize the classifier quality change that occurs by joining pairs of attributes. The vertical axis represents r_r, effect of resolution by replacement, and the horizontal axis r_a, effect of resolution by addition.

6.2.2 Intermezzo: Making of the Attribute Structure

Attribute structure in DEX is based on joining primitive attributes into higher-level concepts. The true purpose pursued by a human designer of the structure is to maintain a small set of relevant attributes whenever making a decision. Capturing interactions is not an explicit objective in this methodology. Examples of motivations to join attributes into a concept are:

Taxonomic aggregation: We aggregate associated attributes (attributes about the cardiovascular system are all joined into a single concept; attributes associated with functional rehabilitation are too joined: sitting ability, standing ability, walking ability).

Taxonomic division: Trying to organize a large number of attributes, we divide them into groups, sometimes arbitrarily (medical complications can be divided into early complications and late complications).

Similarity: Several attributes may be similar or correlated, often they are all consequences of an unobserved attribute. For that purpose, the concept is defined to match the unobserved attribute, and its value is deductively derived from its consequences.

Interactions: The concept cannot be reduced to independent sub-problems. It cannot be fully understood without considering all attributes at once (deciding about car's comfort: the number of car doors, and the number of family members; presence of arithmetic operations: car price and maintenance price).

We have already discussed similarity: it is an example of a false interaction. But the functional relation between taxonomic relatedness and an interaction is only our hope that interactions between unrelated attributes are unlikely.

We do not claim that taxonomic relatedness and ontologies in general are harmful: aggregating multiple noisy measurements is generally beneficial, but representing the aggregation with a segmentation function is often less appropriate than representing it with a voting function. There are several automatic methods intended to perform a similar deed, often presented under the common term *variable clustering* [SAS98]. The voting function in the naïve Bayesian classifier is not burdened with the number of simultaneously present attributes, as long as they only 2-interact with the class: for a machine it is not as important to reduce the number of attributes in a group as it is for a human analyst.

6.2.3 Predicting the Quality Gain

After we have shown that the attribute structure is not necessarily a good predictor of existence and type of interactions, we will focus on evaluating various automatic heuristics for predicting the r_r: improvement or deterioration of classifier quality achieved by replacing interacting attributes X, Y with their Cartesian product $\overline{XY}$, *improvement by replacement* or simply quality gain. Quality gain is a non-binary quantification of the pragmatic interaction test from Sect. 4.3.2. We will no longer burden ourselves with addition of the joined attribute, as results from previous section demonstrate that it is quite consistently inferior to replacement.

Wrapper Heuristic

We will start with the most obvious approach. Using 10-fold cross-validation on the training set, we perform evaluation of the same classifier which will be later evaluated on the test set, e.g., $IBC(XY, Z)$, and average the results over those 10 folds. The results are drawn on Fig. 6.5, and appear satisfactory for 'nursery' and 'artificial'. For domain 'car', the size of the training set is quite limited, and the wrapper estimate of improvement is hence underestimating the actual improvement. For domain 'HHS', with relatively few instances, the errors are larger, but unbiased.

6.2.4 Myopic Quality Gain

Can we can simplify the wrapper heuristic? One approach is by focusing only on a pair of attributes, and ignoring all the others. Specifically, if we wonder about the interaction between attributes X and Y, we evaluate myopic improvement by replacement through $\hat{r}_r = (IBC(XY)) - q(IBC(X,Y))$, ignoring attribute Z. Judging by Fig. 6.6, this simplification did not affect the results much. The source of myopia lies in disregarding all other attributes while the interaction between a pair of them is investigated.

Desiring further simplification, we try to avoid using internal cross-validation, and just evaluate the improvement by replacement myopically on the test set. The results are presented in Fig. 6.7. Although true interactions do yield larger improvement by replacement estimates, all the estimates are positive. It is not obvious where the break-even point is, but if we have to use wrapper-like cross-validation to estimate that break-even point, we might as well use unambiguous wrappers everywhere.

6.3 Non-Wrapper Heuristics

6.3.1 Interaction Gain

We have previously used interaction gain to evaluate the types of interactions. We will now examine whether interaction gain is connected with the quality gain by replacement. The relationship does exist, even if not particularly strong, and is illustrated on Fig. 6.8. The important conclusion, however, is that quality gain by replacement can be understood as a test of significance of an interaction. Only strong false interactions and strong true interactions result in positive quality gain. But only interaction gain is able to classify the interaction type, we do not obtain this information from quality gain.

There is an is an indisputable similarity between interaction gain and myopic wrapper estimate of improvement, and this correlation is sketched in Fig. 6.9.

6.3.2 Cochran-Mantel-Haenszel Statistic

Earlier, we have mentioned the problem of test-set heuristics, where it is difficult to determine whether an estimate of improvement is significant or now. Cochran-Mantel-Haenszel χ^2 test is used for testing the null hypothesis that two attributes are conditionally independent with respect to the label. The null hypothesis is that two attributes are conditionally independent in each class, assuming that there is no four-way (or higher) interaction. The p-value is close to 0 when the null hypothesis is very unlikely, but if it is not very unlikely,

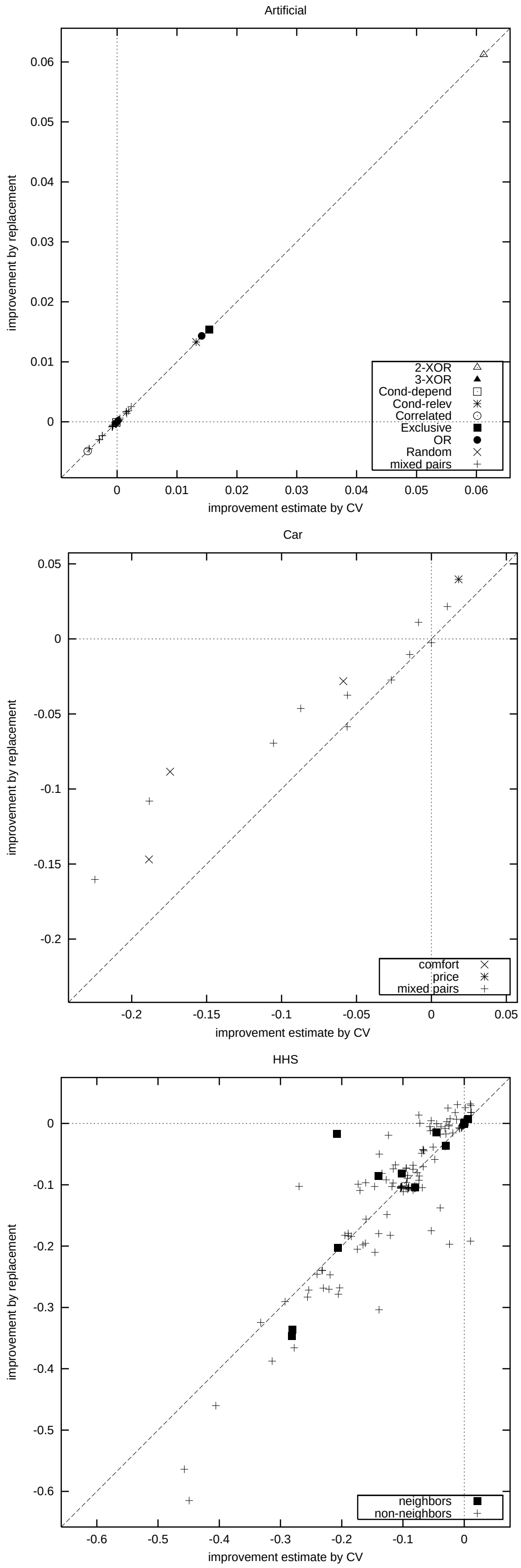


Figure 6.5: Wrapper estimates of improvement by internal 10-fold cross-validation on the training set in comparison with the external 10-fold cross-validation.

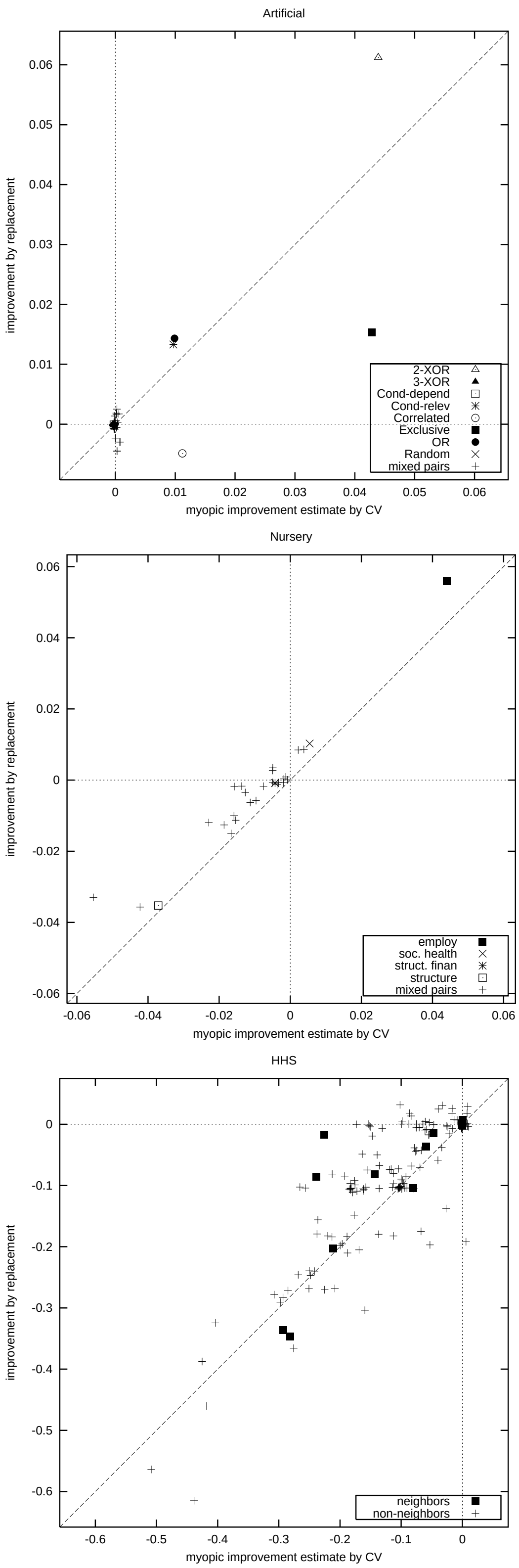


Figure 6.6: Wrapper estimates of improvement by replacement by myopic internal 10-fold cross-validation on the training set.

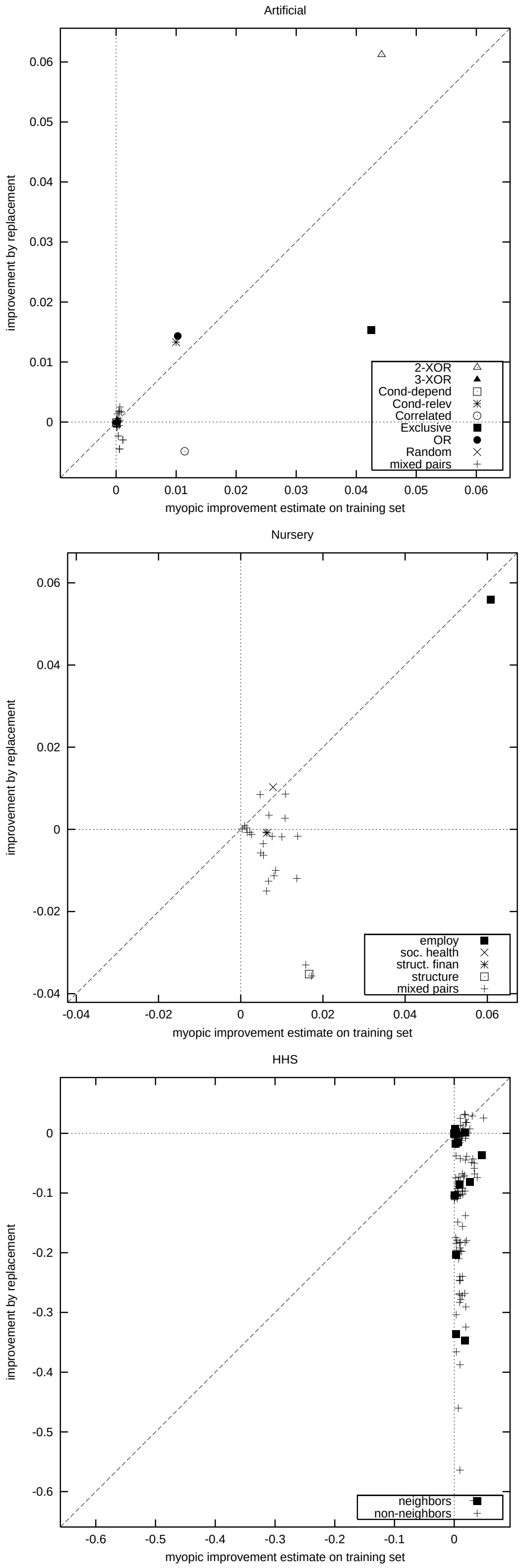


Figure 6.7: Wrapper estimates of improvement by replacement by myopic evaluation on training set.

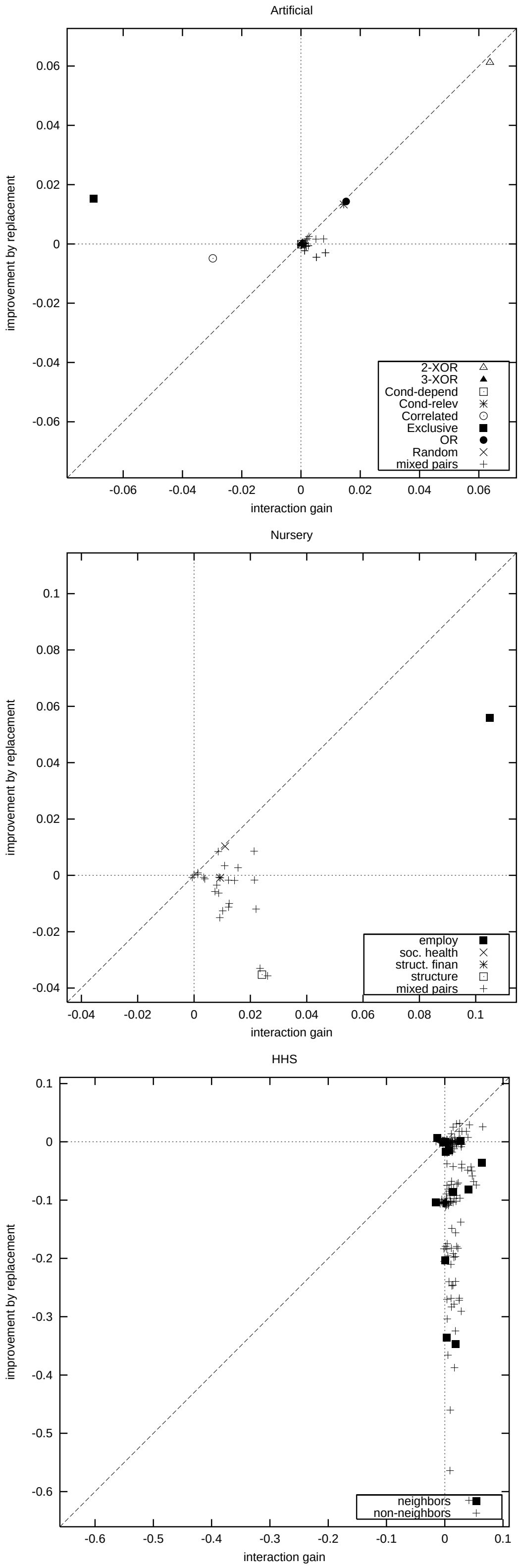


Figure 6.8: The relation between the interaction gain and quality gain by replacement.

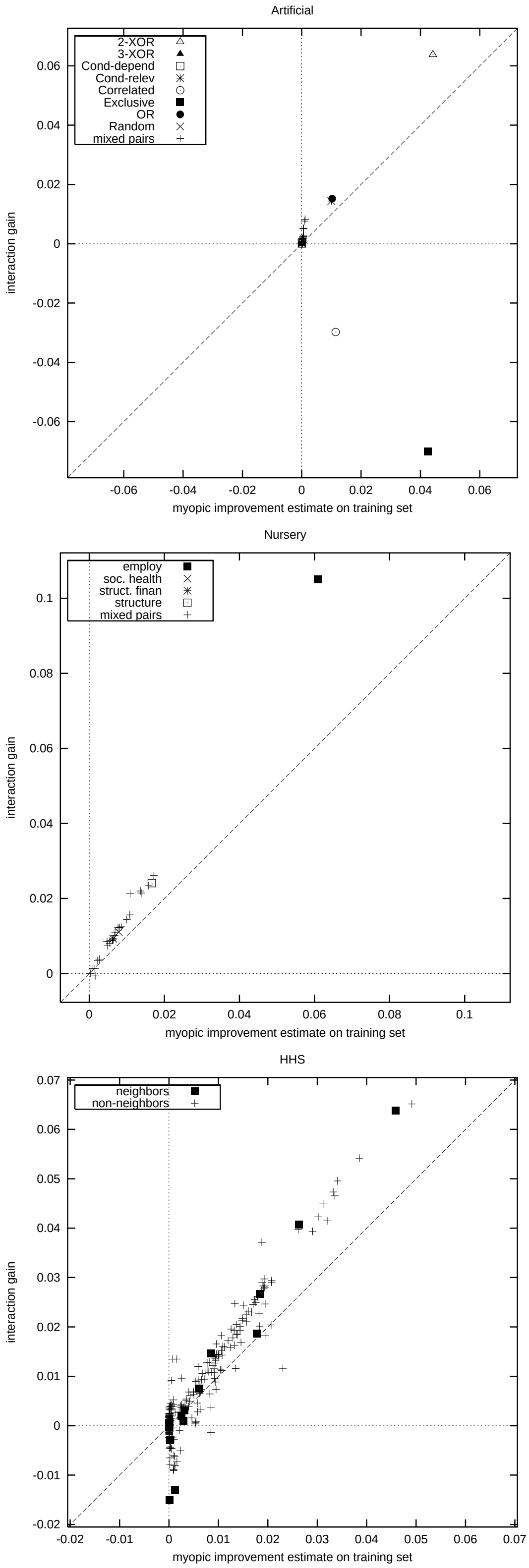


Figure 6.9: Comparison of interaction gain with myopic wrapper estimate of improvement on the training set.

the p-value may be randomly distributed. If we are disturbed by the nature of the p-value, we should use a measure of association, not a statistic.

Judging from Fig. 6.10, resolving the very likely dependencies, plotted on the left side of the graph, tends to cause a positive quality gain, especially in the 'nursery' domain. Many interactions go undetected by the CMH test, and many likely dependencies cause a deterioration, especially in 'HHS'. There are many other statistical tests of dependence, surveyed in Sect. 3.2. We only tested CMH because it appears to be the most frequently used and is sufficiently general, unlike many tests limited to $2 \times 2 \times 2$ contingency tables.

6.4 Heuristics from Constructive Induction

In this section, we will focus on non-myopic heuristics described in Sect. 5.2 and in [Zup97, Dem02]. We intend to compare these heuristics with interaction gain. We have used the Orange framework [DZ02] to conduct our experiments.

We also conducted experiments with the SNB and mutual information, but they did not provide much insight. Experimentally, SNB and mutual conditional entropy are closely related. They are especially sensitive to attribute dependencies in the form of positive $I(A;B)$.

6.4.1 Complexity of the Joint Concept

In our previous experiments, we expended no effort for trying to simplify the joint attribute, which was always a simple Cartesian product. In reality, simplifying that attribute would result in superior performance, as the estimation for each segment would be performed on more data. Of course, we should not simplify excessively, and we only merge those attribute value pairs which are compatible, in sense that the examples having those value pairs can still be correctly classified provided the values of other attributes. In this manner, the segmentation is non-myopic, and allows us to join attributes which could later prove to be a part of multi-way interactions.

One possible heuristic's value is the number of segments thus obtained. The results are illustrated in Fig. 6.11. We can notice that the heuristic excels at finding the human-designed concepts, even when these concepts are not immediately useful in the NBC context, in all domains, except for 'artificial' — especially interesting is its performance on the natural 'HHS' domain. For 'artificial', only the `Exclusive` concept has been discovered, along with several irrelevant concepts, while several useful concepts have been estimated as bad.

It is important to understand that when this heuristic is used in the context of function decomposition, only the best concept is chosen, and the creation of a new attribute modifies the domain, and consequently the heuristic values in subsequent iterations of the algorithm. So our comparison should be understood in a proper context.

This heuristic does not look for interactions. It is simplifying away irrelevant groups of attribute values, starting from the least useful attributes. Eventually remain only a few powerful rules, the decisive attribute values. For example, if you have a bleeding torn neck artery, an infected blister is largely irrelevant to your health status. Function decomposition will thus simplify the attributes related to the details of the blister. But would we not achieve a similar effect by increasing the weight of attributes relating to bleeding wounds?

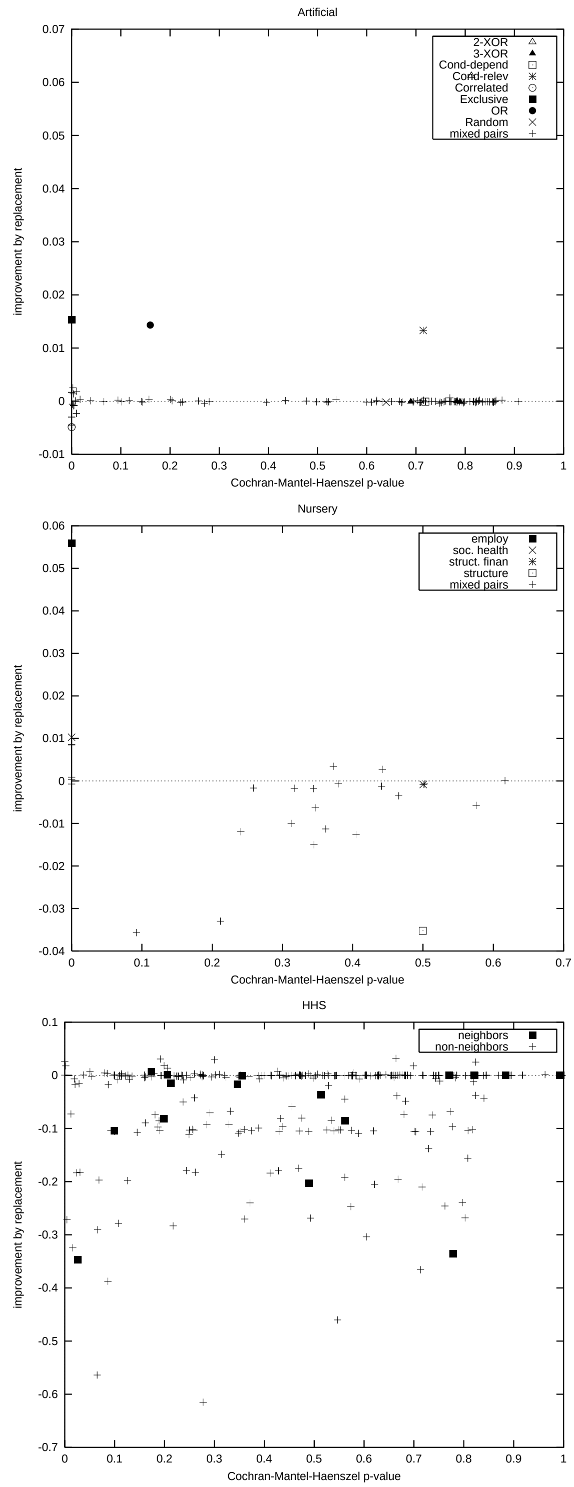


Figure 6.10: The horizontal coordinate of each point is the p-value of the null hypothesis that two attributes are conditionally independent in each class, assuming that there is no three-way interaction. On the left, there are the likely dependencies, elsewhere there are the less certain dependencies. The vertical coordinate is the actual improvement gained by replacing that pair of attributes.

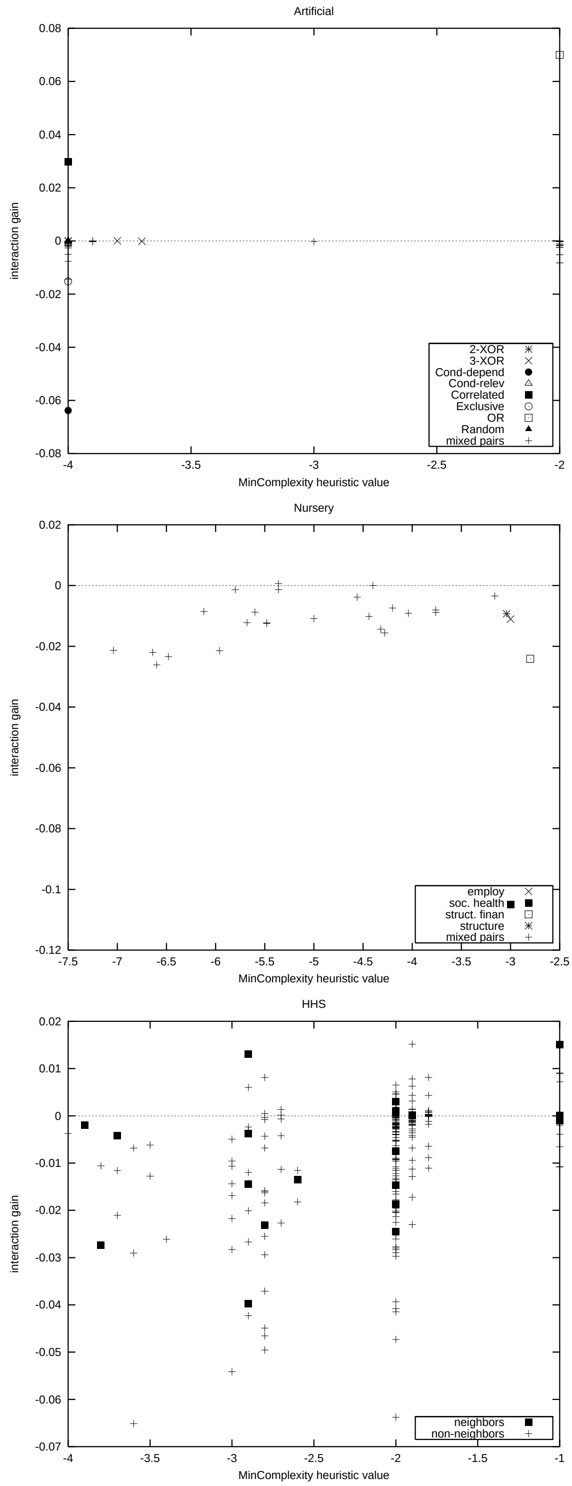


Figure 6.11: The horizontal coordinate is a heuristic estimate of joint attribute complexity after lossless segmentation. The vertical coordinate is interaction gain.

6.4.2 Reduction in Error achieved by Joining

In non-deterministic domains, residual ignorance is unavoidable. There crisp compatibility-based segmentation rules and the associated complexity-based heuristic from previous paragraphs are no longer appropriate. The class is rarely consistent even for instances with identical attribute values, and thus it is hard to find any compatibility whatsoever between attribute value pairs.

Instead, but in a similar style, we merge those attribute values which increase the purity of the domain, not myopically but with respect to other attributes. As merging never truly reduces the purity, possibly only increases it, we introduce m-error estimate [Ces90], an improved embodiment of the Laplacean prior that penalizes small instance sets. We set the value of m to 3, which works well on most domains.

Similarly, we merge away irrelevant attribute values. For example, for the OR concept in 'artificial' on Fig. 6.12, three of four attribute value pairs are indistinguishable from each other, and this heuristic will reduce the four attribute value pairs into merely two, without any loss in true purity. This way, minimization of error achieves stellar performance on the 'artificial' domain. Most importantly, it discovered the 3-way interactions (3-XOR), and only dismissed random attributes. Therefore, minimization of error appears not to be myopic. Unfortunately, on all other domains, it was not found useful for detecting either useful concepts or pairs of attributes with high interaction gain.

6.5 Experimental Summary

We tried to search for interactions. Our study was based around comparing different heuristics. We found out that interaction gain is a useful estimate of the interaction type, unlike most other known measures. It is worth to replace pairs of attributes which truly interact, or strongly falsely interact with a new attribute. Moderately falsely interacting attributes were better off left alone, given no suitable alternatives.

Our 3-way interaction gain is myopic in sense that it is unable to discover perfect 4-way interactions. Human-designed attribute structures do not distinguish between true and false interactions, so they are of limited applicability in resolving interactions on natural data.

A wrapper estimate of improvement of a naïve Bayesian classifier after joining the pair of attributes was found to be a robust test of significance of an interaction given only the training set of instances, but this conclusion is somewhat tautological. A useful simplification of the wrapper estimate was the myopic wrapper estimate, in which only the investigated pair of attributes was used to construct the naïve Bayesian classifier, while all other attributes were neglected.

It very interesting that only strongly false interactions and strongly true interactions, as measured by the interaction gain, yielded a positive quality gain, as measured by the Kullback-Leibler divergence. Other probes did not provide much useful information, although we note that minimal-error probe appears to be able to pinpoint multi-way interactions.

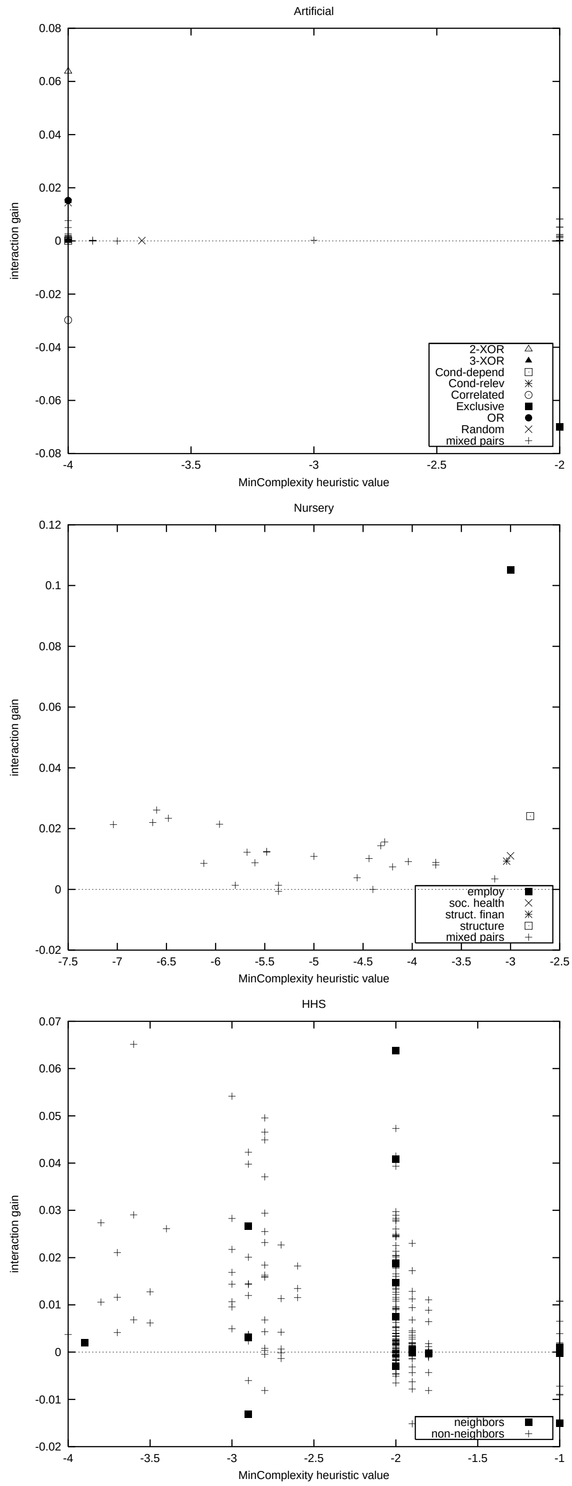


Figure 6.12: The horizontal coordinate is a heuristic estimate of expected reduction in error achieved by noise-proof segmentation. The vertical coordinate is interaction gain.

CHAPTER 7

Interaction Analysis and Significance

Statistics are like a bikini. What they reveal is suggestive, but what they conceal is vital.

Aaron Levenstein

In this chapter, we will attempt to demonstrate that being informed about interactions provides benefit to the data analyst. For that reason, we try to present the false and true interactions in the domain in a comprehensive visual way.

As is our custom, we will detailedly investigate three natural domains, a very frequent benchmark, the 'adult' or 'census' data set [HB99], and the natural domain 'HHS' we already used in Ch. 6, which contains relatively few instances. Furthermore, we explored a new medical data set 'breast' with many instances, contributed by Dr. T. Čufer and Dr. S. Borštner from the Institute of Oncology in Ljubljana.

Because 'adult' and 'breast' data sets contain numerical attributes, we used the Fayyad and Irani entropy-based algorithm to discretize them, as implemented in [DZ02], except when it would cause an attribute to be collapsed in a single value. In such a case, we used equal-frequency discretization with two intervals, with the median value being the interval boundary.

Missing values exist in the 'adult' and 'breast' data sets, and we represented them with a special attribute value. We could have assumed that the values are missing at random, but they rarely miss at random. It might be beneficial to introduce such an assumption in small data sets, if this were our focus (it was not).

Analyzing a domain with respect to interactions between attributes provides a useful representation to a human analyst. We will distinguish true from false interactions, as different visualizations suit each type. For example, the false interactions tend to be transitive, whereas true interactions tend not to be. Therefore, a hierarchical presentation of the false interactions captures their mutual similarities best, whereas a graph presents the few true interactions that may exist.

In this chapter, we will be concerned solely with 3-way interactions between two attributes and the label. Therefore, when an interaction between two attributes is mentioned, we really mean a 3-interaction between the two attributes and the label.

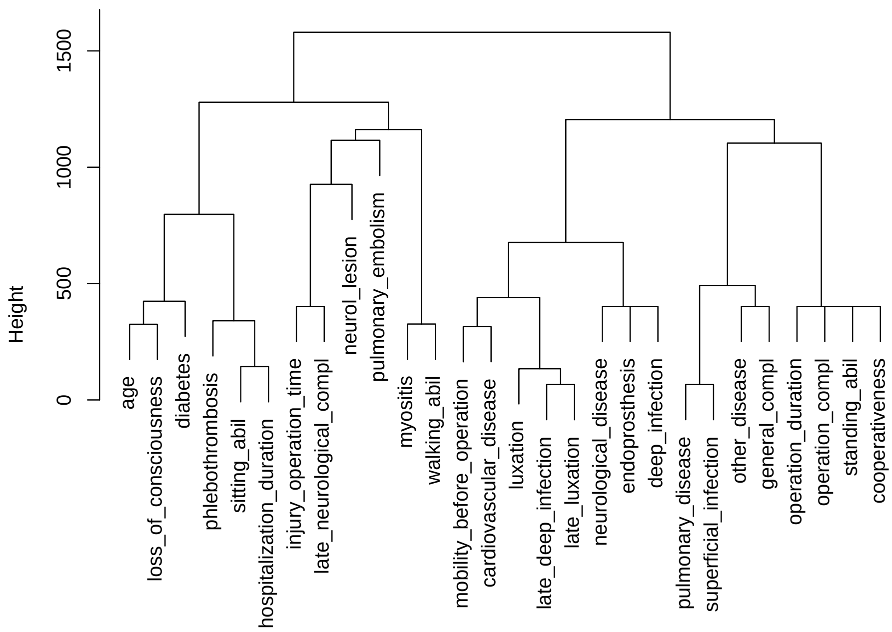


Figure 7.1: False interaction dendrogram analysis on domain ‘HHS’.

7.1 False Interactions

Attributes that interact falsely with the label should appear close to one another, while those which do not should be placed further apart. False interactions are transitive, so either *clustering* or multidimensional scaling are appropriate presentation methods. We used the hierarchical clustering method ‘agnes’ [KR90, SHR97], as implemented in the ‘cluster’ library for the R environment [IG96]. The results were obtained with Ward’s method, described in more detail in Sect. A.1. The dissimilarity function, which we express as a matrix D, was obtained with the following formula:

$$\mathsf{D}(A,B) = \begin{cases} NA & \text{if } IG(ABC) > 0.001, \\ 1000 & \text{if } |IG(ABC)| < 0.001, \\ -1/IG(ABC) & \text{if } IG(ABC) < -0.001. \end{cases} \tag{7.1}$$

The significance of (7.1) is that the dissimilarity is low when the interaction gain is negative, therefore the attributes are close. On the other hand, when the value of interaction gain is close to zero, they appear distant: independence pushes attributes apart. For true interactions, we cannot say anything about their proximity or remoteness, and we therefore assign the value of NA (not available), trying not to affect placement. Because the value of NA is not supported by the clustering algorithm, we replace it with the average dissimilarity in that domain. In summary, groups of dependent attributes will be clustered close together, while independent attributes will lie apart.

The results from the clustering algorithm are visualized as dendrograms in Figs. 7.3-7.2. The height of a merger of a pair of attributes in a dendrogram is also an indicator of their

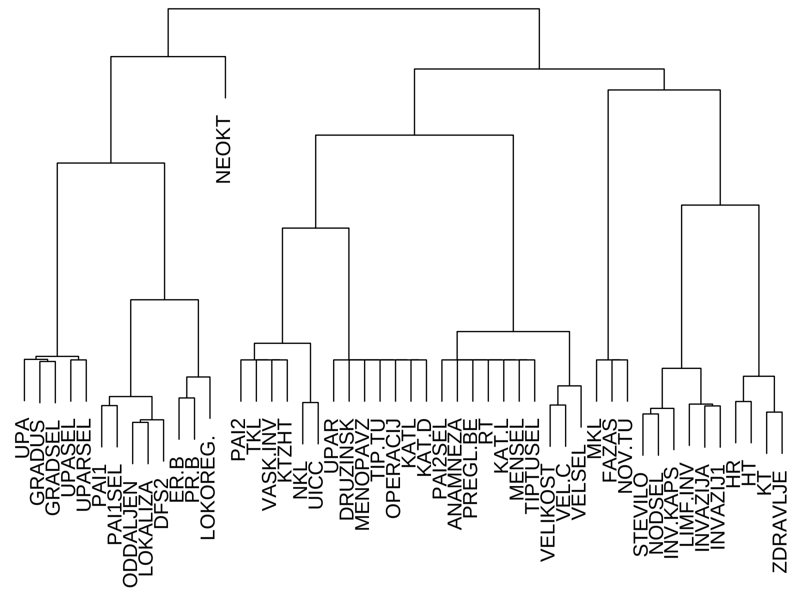


Figure 7.2: False interaction dendrogram for domain 'breast'.

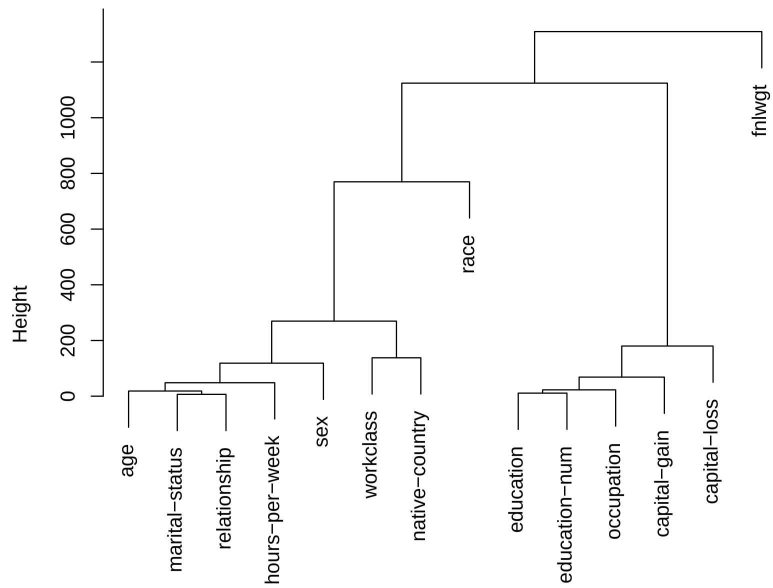


Figure 7.3: False interaction dendrogram for domain 'adult'.

proximity. The lower the merger in the tree, the closer the pair is. For example, in Fig. 7.3, the most falsely interacting pairs of attributes are `martial_status-relationship`, and `education-education_num`.

In Fig. 7.3, attributes `age`, `martial_status` and `relationship` give us virtually the same information about the label (`earnings`). The attribute `race` appears to provide novel information. On the other hand, attribute `fnlwgt` provides either completely unique information, or no information at all. These dendrograms do not provide any information about informativeness of individual attributes, merely about the similarities between informativenesses of attributes. Feature selection which disregards true interactions could be based simply on picking the most informative attribute from a given cluster.

One could contrast our method to the well-known variable clustering approach, as described in, e.g., [SAS98]. However, we do not compute dissimilarity merely on the basis of the similarity of the attribute values. We instead compare attributes with respect to the similarity of information they provide about the class. We also disregard true interactions.

It is surprising that the false interaction dendrograms appear to create meaningful clusterings without any background knowledge whatsoever. All they take into account is the sharing of information about the label in the attributes.

7.2 True Interactions

We may display true interactions in a graph, where vertices correspond to attributes and edges indicate the existence of 3-way interactions with respect to the class. We used the *dot* software for rendering the graphs [KN].

We identify true interactions by a positive value of interaction gain. It was noticeable already in Ch. 6 that most interactions are weak and may only be artifacts of noise. For that reason, we only pick the strongest interactions, with interaction gain above some cut-off point. Figs. 7.8–7.10 contain renderings of true interactions, as estimated by interaction gain. The edges are labeled with interaction gain, expressed as the percentage of the largest interaction gain in the domain. The opacity of the edge is adjusted with respect to that percentage, for better clarity.

The cut-off point was set at the 'knee-point' in the sorted series of interaction gains. For example, the series for 'HHS' is

$$[100, 97, 83, 76, 72, 71, 68, 64, 63, 62, 61, 60, 56, 45, 45, 44, 44, 43, 43, 42, 42, 42, 40, 40, \dots],$$

and its knee-point appears to be a discontinuity around 50. Beyond 50, the interaction gains start being densely distributed, and are likely to be sampled from a normal distribution centered around 0, also visible in between the two humps in Fig. 6.3. The large hump are the 'random' interaction gains, whereas the small hump are the true interactions. The distribution of interaction gain in most domains has such a shape.

7.2.1 Applicability of True Interactions

It is interesting to compare the interaction graph in Fig. 7.9 with the classification tree induced by the C4.5 classification tree learning algorithm [Qui93] for domain 'breast' in Fig. 7.7. These same two attributes are the interaction that performed best. This classification tree yielded perfect classification accuracy. Therefore, interaction gain could possibly be a non-myopic heuristic split selection criterion.

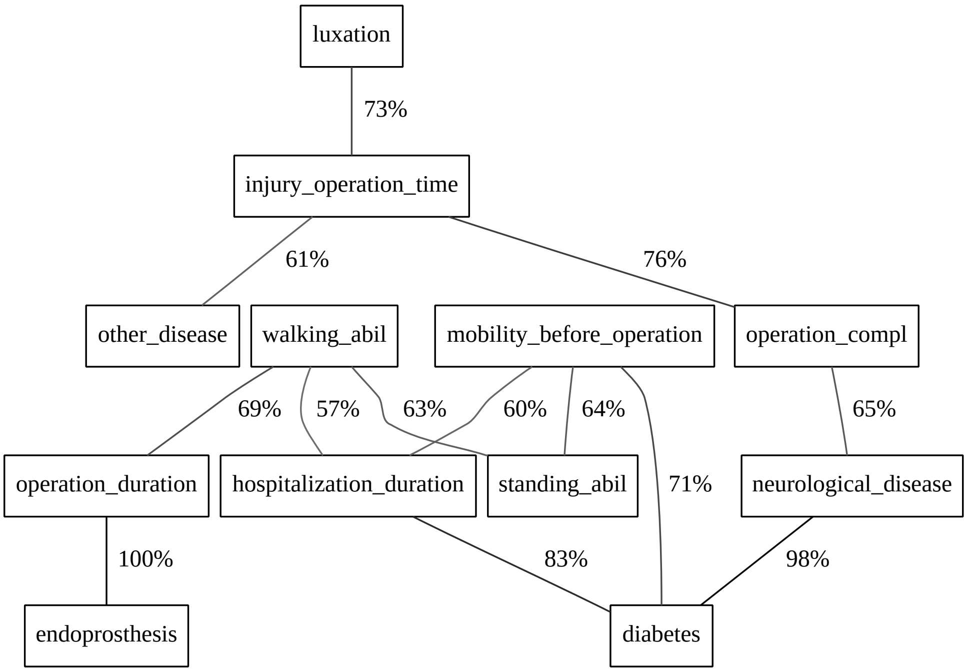


Figure 7.4: True interaction graph of domain 'HHS'.

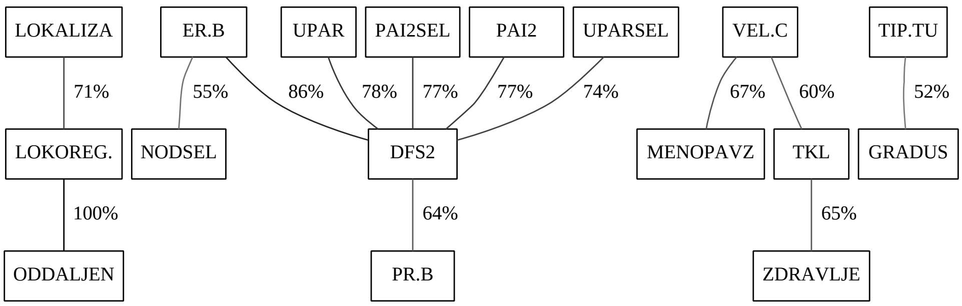


Figure 7.5: True interaction graph of domain 'breast'.

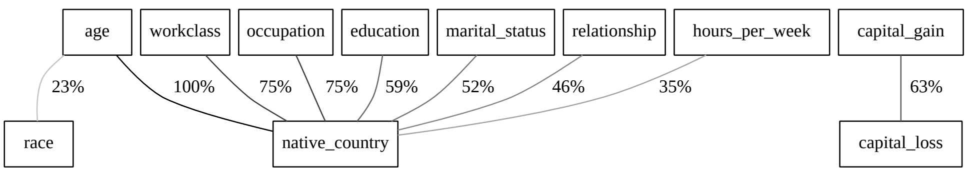


Figure 7.6: True interaction graph of domain 'adult'.

```
ODDALJEN > 0: y (119.0)
ODDALJEN <= 0:
:...LOKOREG. <= 0: n (506.9)
    LOKOREG. > 0: y (17.1/0.1)
```

Figure 7.7: A perfect tree classifier for the 'breast' domain, learned by C4.5.

If we mentioned the utility of false interactions to feature selection, we may now mention the utility of true interactions to discretization: most procedures for discretization are univariate and only discretize one attribute at a time. When there are true interactions, additional values in either of the attributes may prove uninformative and could be neglected. For that reason it would be sensible to discretize truly interacting groups of attributes together, in a multivariate fashion. Such a procedure was suggested in [Bay00]. It seems that for falsely interacting groups of attributes, multivariate discretization is not necessary, but such claim should be tested.

7.2.2 Significant and Insignificant Interactions

Rather than by adjusting the threshold, we may measure the magnitude of an interaction through its performance gain, as measured by the wrapper estimate of improvement after joining the attributes in a Cartesian product, described in Sect. 6.2.3. This is the same as performing the pragmatic interaction test from Sect. 4.3.2. Regardless of positive interaction gain, we disregard those interactions that do not also yield an improvement in classification performance. It is easy to see that only a small number of interactions are truly significant.

The edges in our visualizations of true interaction graphs in Figs. 7.8–7.10 are labeled with the quality gain, expressed as percentages of the best-performing interaction. The most significant interaction in the domain is marked with 100%.

In the visualization of domain 'adult' in Fig. 7.10, there is a single interaction worth mentioning: between `capital gain` and `capital loss`. The `fnl_weight` attribute is likely to be noise, and noise has a tendency of overfitting the data better when there are more values. No wonder that this domain has been used so frequently.

7.3 Experimental Summary

We can present false interactions to a human analyst in the form of a dendrogram, created with a hierarchical clustering algorithm. In the dendrogram, falsely interacting attributes appear close together, while independent attributes appear far from one another.

We illustrate true interactions in an interaction graph, where edges indicate the existence of a true 3-way interaction between the pair of attributes, denoted as vertices, with the label.

Although there are many candidates for true interactions, only a small number of them are truly important. We present a significance testing method, based on the pragmatic interaction test. We evaluate the improvement of classifier's performance when a pair of attributes is replaced with their Cartesian product. We can confirm that a true interaction

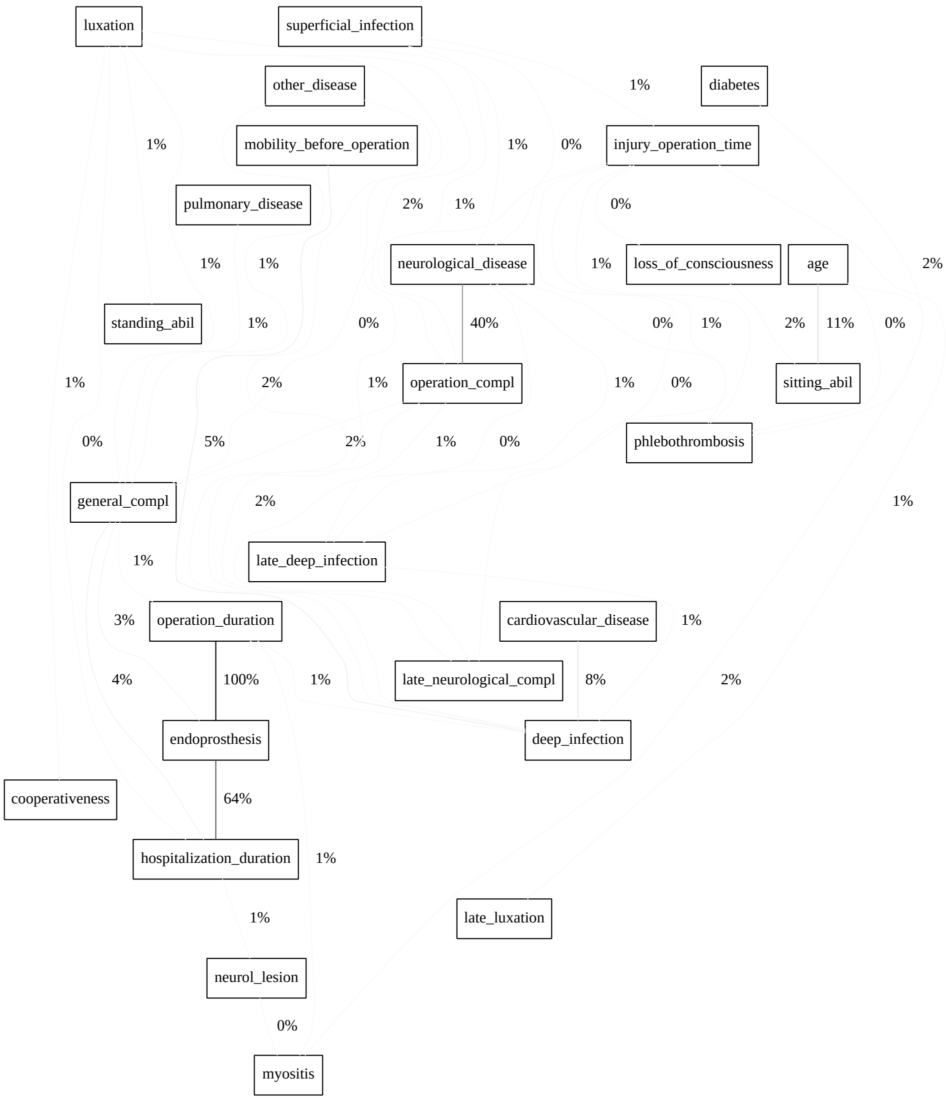


Figure 7.8: True interaction analysis on domain 'HHS' with performance gain cut-off.

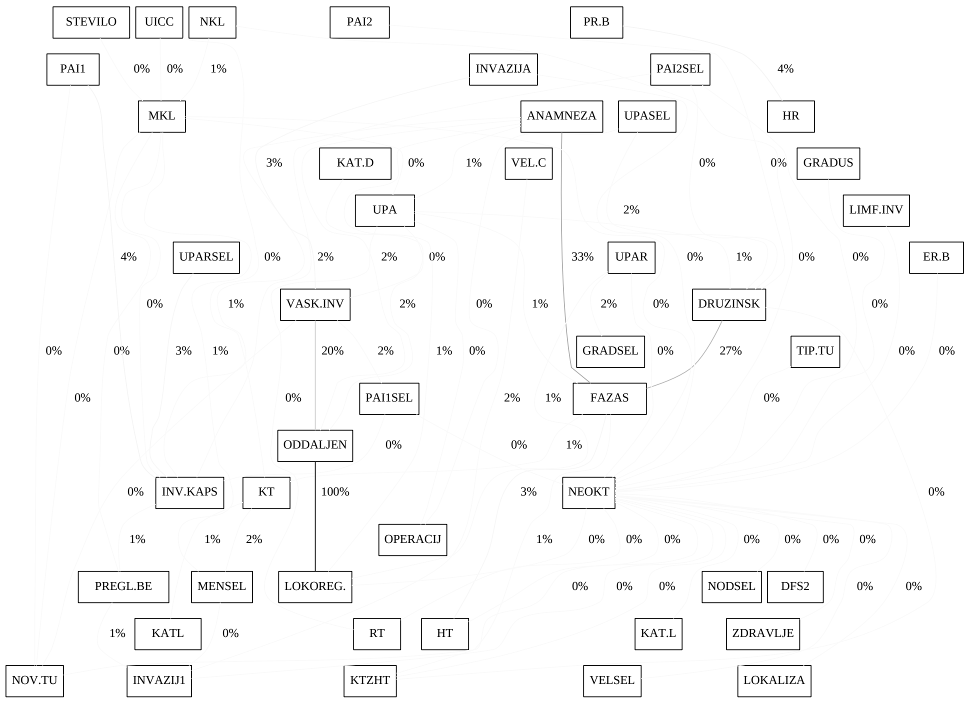


Figure 7.9: True interaction analysis on domain ‘breast’ with performance gain cut-off.

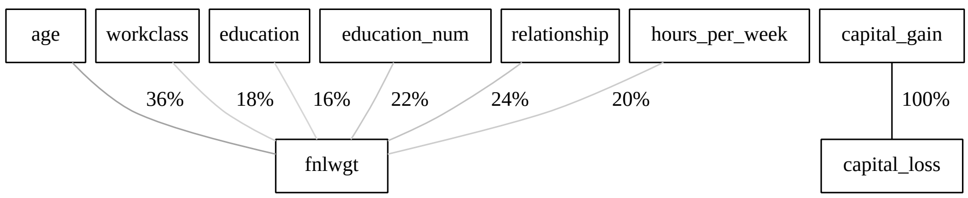


Figure 7.10: True interaction analysis on domain ‘adult’ with performance gain cut-off.

can only be significant if there is a lot of data, as it was already observed in [MJ93]. However, there are usually many false interactions.

We suggest that being informed about true interactions may be useful to non-myopic split selection in construction of classification trees, and may provide a starting point for non-myopic multivariate discretization. On the other hand, false interactions could be useful for feature selection.

Feature selection is meaningful for two reasons: removing irrelevant noisy attributes and removing duplicated attributes. On the true side, feature selection which does not consider interactions might dispose of attributes which may initially appear noisy, but disclose information inside a true interaction. On the false side, feature selection which simply disposes of correlated attributes is not doing its job well.

CHAPTER 8

Better Classification by Resolving Interactions

Torture the data long enough and they will confess to anything.

Anonymous

Although interaction dendrograms and interaction graphs are pretty, their usefulness is subjective. In this chapter we will show how knowledge of interactions can improve the objective performance of machine learning algorithms. We use the same data sets as in Ch. 7, identically processed.

The core idea for improving classification performance with knowledge of interactions is *interaction resolution*. If a simple learning algorithm receives a particular set of attributes, it assumes that different attributes are not interacting in complex ways with respect to the class. Our initial example was the naïve Bayesian classifier, but there are other algorithms that take a similar assumption, for example logistic regression (LR) [PPS01], and optimal separating hyperplanes [Vap99] (also see Sec. A.2), a type of support vector machines (SVM) with a linear kernel function. Both are built around a projection function that finds an informative hyperplane in the attribute space, and are designed for domains with two classes. Because they are linear, they are also sensitive to interactions.

Logistic regression determines this hyperplane with gradient descent or some other numerical optimization procedure as to maximize some statistical criterion, usually likelihood of the training data given the hyperplane and an estimated parameter of the logistic distribution. Apart from the hyperplane, which determines the points of equiprobability of both classes, there is another parameter to the logistic distribution, which defines the 'slant' of the logistic distribution function, or its scale parameter, and is estimated from the data. When the scale is zero, logistic regression behaves like a linear discriminant.

Optimal separating hyperplanes are discriminative learning algorithms, where the hyperplane is placed as to maximize the distance from the nearest instances of either class. Usually, quadratic programming is used for this purpose. Label's probability distribution is a simple threshold function of the distance to the hyperplane, unless we instead apply an estimation function. In our experiments, we used univariate logistic distribution instead, and estimated both of its parameters, the scale and the mean.

8.1 Implementation Notes

Our experiments were performed with the Orange machine learning toolkit [DZ02]. It implements the naïve Bayesian classifier (NBC), function decomposition (HINT), and classification trees (orngTree). Since our data is non-deterministic, we used minimal-error function decomposition (HINT-ME). Orange also contains the C4.5 tree induction algorithm [Qui93]. We used extensions to Orange [Jak02], which implement support vector machines (SVM) [CL01], and logistic regression (LR) [Mil92]. It is important to know that neither logistic regression nor support vector machines are standardized. There are many algorithms and implementations with differing performance.

Both logistic regression and SVM classifiers are designed for classification in binary classification problems with two classes. For problems with n classes, we create n binary classification problems, and the task is to separate instances of one class from those of different class. For fusing all these probabilities in a single probability distribution we used the algorithm described in [Zad02] and implemented in [Jak02].

In SVM, we used a separate feature dimension for each attribute-value pair, even for binary attributes. For logistic regression, a single variable for each binary attribute proved to be more effective. In our experiments, the bivalent representation $(-1, 1)$ of attributes worked well for SVM, while the binary representation $(0, 1)$ proved suitable to logistic regression. We used dummy coding of nominal attributes in LR and SVM: a 'dummy' binary attribute is created for every multi-valued nominal attribute value. The two classification tree learning algorithms had the advantage of using non-discretized data in the 'adult' and 'breast' data sets.

Although SVM is a powerful method, able to capture non-linear relationships between attributes and class, we used only the simplest, linear SVM kernel, with the option $C = 1000$. The complexity of such a classifier is comparable to the one of NBC. We discuss the definition slightly more detailedly in Sect. A.2. To obtain probabilistic classification using the SVM, we used 10-fold internal cross-validation to create a data set containing the distance to hyperplane as the sole descriptor. We then estimated the two parameters of the logistic distribution to build the probabilistic model.

We compared the different techniques with the unarguably simple method of classification accuracy, and beside it, with the more sensitive Kullback-Leibler divergence. If a classifier estimates the probability of the correct class to be zero, the KL divergence would reach infinity. To avoid penalizing such overly bold classifiers too much, we add $\epsilon = 10^{-5}$ to all the probabilities before computing the logarithm. Natural logarithms were used to compute KL divergence. For some instance $\mathbf{i}$, the KL divergence is computed as $\ln(1+\epsilon) - \ln(\Pr\{d(\mathbf{i}) = C(\mathbf{i})\} + \epsilon)$. This way, the KL divergence will never be more than 11.6 for any instance. This correction factor only affects the value of evaluation function near zero, elsewhere the influence is imperceptible.

We have not attempted to use sophisticated techniques for evaluation, such as information score, area under ROC, or the McNemar's test, because the differences between algorithms are ample enough. We used simple 10-fold cross-validation in all cases, and averaged the value of evaluation function over all the instances. We list the standard errors

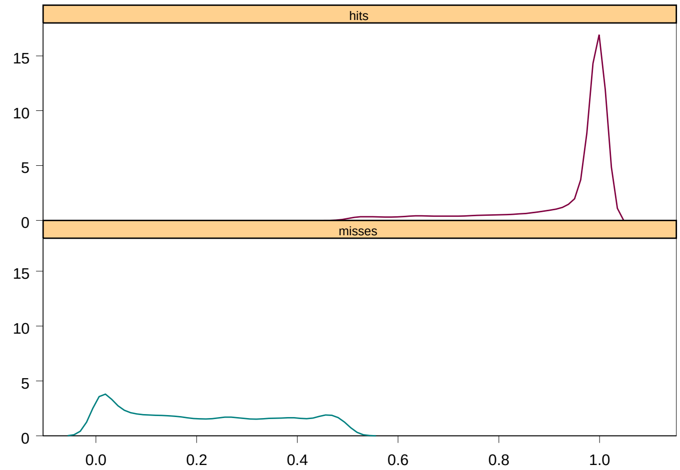


Figure 8.1: Distribution of true class probabilities as provided by the naïve Bayesian classifier for unseen data in the 'adult' domain. Above is the frequency distribution of probabilities for correct classifications, and below is the frequency distribution of probabilities for mistaken classifications.

next to each result. The standard error is estimated across the 10 folds of cross-validation, both for KL divergence and the error rate. For that reason, it should be viewed only as an illustration of result stability with respect to folds, and not as an instrument for judging the significance of result improvement.

Some tests were not executed, purely because of inefficient implementations of certain algorithms. For example, our implementation of SVM was not able to handle the 'adult' data set, as the performance of SVM drops rapidly with a rising number of training instances, even if it is extremely effective with a large number of attributes.

8.2 Baseline Results

The base classification results are presented in Table 8.1. It is easy to see that the SVM wins in the 'HHS' domain, classification trees and logistic regression in the 'breast' domain, and NBC in the 'adult' domain. In the comparison we included the timid learner which ignores all the attributes, and merely offers the estimated label probability distribution as its sole model.

The performance of the naïve Bayesian classifier on the 'adult' domain is interesting: it has the worst error rate, yet the best KL divergence. This indicates that it is able to timidly but reliably estimate the class probabilities, while logistic regression tends to be overly confident. Judging by Fig. 8.1, when the NBC estimated the probability to be different from 1, it was a lot likelier that it was a miss than a hit. On relatively few occasions was NBC confidently wrong.

'adult'	Kullback-Leibler	Error Rate
NBC	0.416 ± 0.007	16.45 ± 0.28
LR	1.562 ± 0.023	13.57 ± 0.20
C4.5	0.619 ± 0.015	15.62 ± 0.24
Timid	0.552 ± 0.001	24.08 ± 0.13

'HHS'	Kullback-Leibler	Error Rate
NBC	2.184 ± 0.400	56.25 ± 3.51
LR	1.296 ± 0.106	56.25 ± 2.72
Linear SVM	1.083 ± 0.022	55.36 ± 4.52
SVM: RBF Kernel	1.103 ± 0.025	59.82 ± 4.52
SVM: Poly Kernel	1.116 ± 0.023	63.39 ± 4.72
HINT-ME	1.408 ± 0.116	60.71 ± 6.04
orngTree	6.822 ± 0.699	62.50 ± 5.62
C4.5	3.835 ± 0.470	58.93 ± 4.98
Timid	1.112 ± 0.013	61.61 ± 4.89

'breast'	Kullback-Leibler	Error Rate
NBC	0.262 ± 0.086	2.80 ± 0.72
LR	0.016 ± 0.016	0.14 ± 0.14
Linear SVM	0.032 ± 0.021	0.28 ± 0.19
SVM: RBF Kernel	0.032 ± 0.021	0.28 ± 0.19
SVM: Poly Kernel	0.151 ± 0.049	1.54 ± 0.44
orngTree	0.081 ± 0.027	0.70 ± 0.23
C4.5	0.000 ± 0.000	0.00 ± 0.00
Timid	0.517 ± 0.019	21.12 ± 1.40

Table 8.1: Base classification results without resolving interactions.

$f \leftarrow 0$ {Number of failures}
$\mathcal{H} \leftarrow \{A_1, A_2, \ldots, A_n\}$
$b \leftarrow q(L(\mathcal{H}))$ {Base performance rate}
$\mathcal{I} \leftarrow \mathcal{H} \times \mathcal{H}$ {All attribute pairs}
while $f < N \wedge \mathcal{I} \neq \emptyset$ **do**
  $\langle A, B\rangle \leftarrow \arg\max_{I \in \mathcal{I}} IG_3(I, C)$
  $\mathcal{I} \leftarrow \mathcal{I} \setminus \{\langle A, B\rangle\}$ {Eliminate this interaction}
  $\hat{\mathcal{H}} \leftarrow R(\mathcal{H}, \langle A, B\rangle)$
  $\hat{b} \leftarrow \hat{q}(L, \hat{\mathcal{H}})$
  if $\hat{b} > b$ **then** {Is the new attribute set superior?}
    $f \leftarrow 0$
    $\mathcal{H} \leftarrow \hat{\mathcal{H}}$
    $b \leftarrow \hat{b}$
  else
    $f \leftarrow f + 1$
  end if
end while
return $L(\mathcal{H})$

Figure 8.2: General framework of an interaction resolution algorithm.

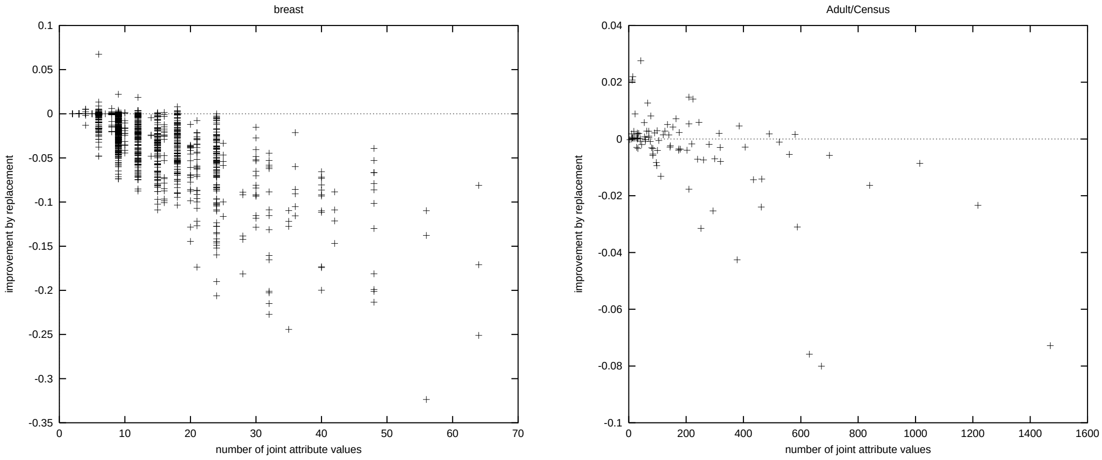
8.3 Resolution of Interactions

If there is an interaction between a pair of attributes, we resolve it using a segmentation function which considers both attributes and creates a new nominal joint attribute. The new attribute can be seen as a range of some segmentation function. The simplest segmentation function is the Cartesian product, but we also mentioned in Sect. 8.4 that we can apply the tool of attribute reduction to reduce the number of joint attribute values.

We are given some simple learning algorithm L, for example the naïve Bayesian classifier, logistic regression, or optimal separating hyperplane. We are given an evaluation function, such as the classification accuracy, or Kullback-Leibler divergence. Furthermore, we are given a resolution function R maps from an interaction and a set of attributes into a new set of attributes where that interaction no longer exists.

Our algorithm, presented in Fig. 8.2, uses interaction gain to guide the model search. We use a failure counter to determine when to stop the search, and we do that after N consecutive failures. To determine the worth of a particular model, we use a wrapper evaluation function $\hat{q}$, which trains a given learning algorithm with a given attribute set on the remainder set, and tests it on the validation set, for a number of remainter/validation set splits. Throughout the algorithm, the training data set is used, so we do not mention it explicitly as a parameter.

We will now address different choices of the learning algorithm, and the resolution function. Furthermore, we will distinguish resolution of false and true interactions.



'breast' 'adult'

Figure 8.3: Relationship between joint attribute complexity and quality gain after resolving the attribute pair.

8.4 Attribute Reduction

The resolution method, in our case the Cartesian product, is a parameter to the wrapper test of usefulness or significance of a particular interaction. Given a better resolution method, such as one that resolves the interaction with fewer attribute values than there are in the Cartesian product, the test could become more sensitive.

Is this needed? Several attributes in the 'adult' data set have a very large number of values, which interfere with the computations of improvement. They do not interfere with computations of interaction gain, however. As illustrated in Fig. 8.3, attributes with very many values indeed cannot improve classification accuracy, regardless of the instance numerousness in 'adult'. It may thus make sense to simplify the Cartesian products before further processing with some attribute reduction algorithm.

For that purpose, we conducted an experiment using the minimal-error attribute reduction in the place of a joint segmentation function [Zup97], briefly described in Sect. 5.2. Starting with the joint Cartesian product attribute, we keep merging pairs of values which are similar with respect to the class, as long as the estimated error keeps dropping. Although the algorithm originally also considers similarity of the class with respect to other attribute values, we disregard all other attributes, as it is not our intention to resolve other interactions. The process of value merging continues for as long as the classification performance is expected to rise, using m-error estimate with $m = 3$.

As to remove the potentially positive influence of minimal-error reduction in individual attributes, we reduce individuals attributes first, without context. We perform the reduction of the joint attribute from original attributes, as individual reduction may discard information which is only useful when both attributes are present. We compute the quality gain by subtracting the quality of the domain with a NBC voting between the two independently reduced attributes with the quality of the NBC with the reduced joint attribute. As earlier, we use interaction gain computed with the Cartesian product to

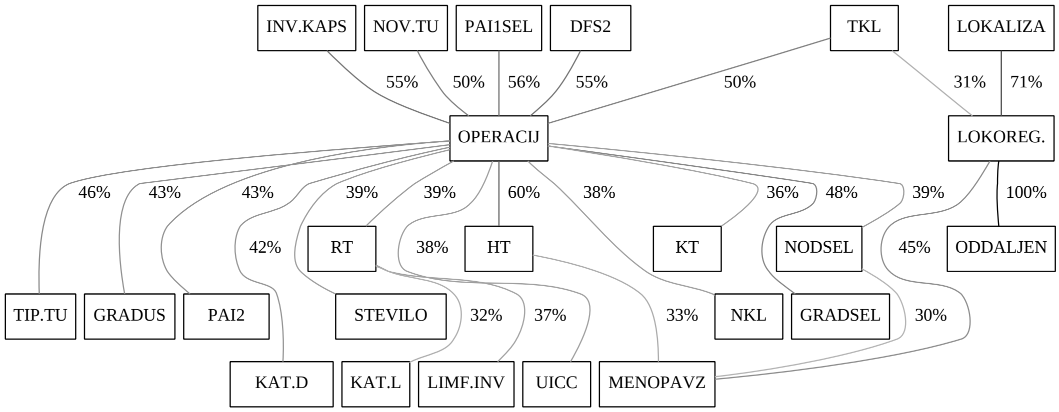


Figure 8.4: True interaction graph with MinErr resolution on domain 'breast'.

classify the interaction.

The results are illustrated in Figs. 8.4–8.5. The first observation is that many more interactions are now significant, also demonstrated by Figs. 8.6–8.8. Several interesting interactions appear, for example the broad moderating influence of the native country in 'adult'. However, the improvement is not consistent, and interactions which were found significant with the Cartesian product attribute disappear after minimal-error reduction, for example the `capital_gain/loss` interaction in 'adult'.

8.5 Resolving False Interactions

Feature selection can be seen as a simple form of attribute reduction where only a single attribute survives the reduction. We tried to do better than that. We used the aforementioned minimal-error attribute reduction. No context was used in attribute reduction, as suggested in [Dem02]. It is obvious that the pairs of attributes with the *lowest* interaction gain should be tried first, in contrast to the interaction resolution algorithm intended for true interactions in Fig. 8.2, which seeks maximal interaction gains.

Once we resolve a false interaction, we replace the original pair of attributes with the new attribute. If in some successive step we try to resolve an attribute which has already been resolved, we simply use its remaining descendant instead. We perform no checking for true interactions among falsely interacting attributes, relying on the wrapper evaluation function to prevent missteps.

As it can be seen in Table 8.2, the results are consistently good, especially for the NBC. In two of three cases, the best classifier's result improved, and in the remaining case nothing changed. Only in 'HHS' there was some result deterioration in KL divergence scores, while classification accuracy sometimes even improved. Perhaps the default parameter of $m = 3.0$ should be adjusted to match the large amount of ignorance in the 'HHS' domain. Also, internal cross-validation used by the wrapper evaluation function is not reliable on small data sets: a leave-one-out approach would be feasible performance-wise in this case. Nevertheless, on 'HHS' the SVM classifier's baseline performance, which was best prior to resolution, even improved after resolving false interactions.

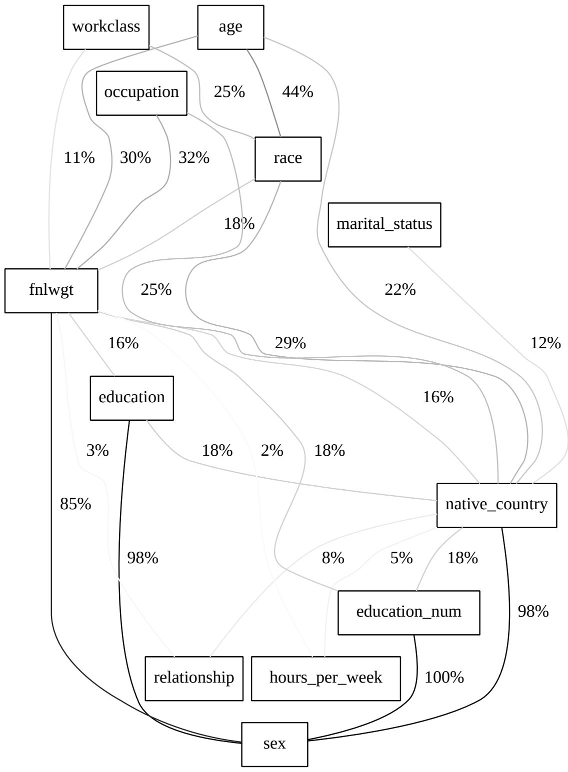


Figure 8.5: True interaction graph with MinErr resolution on domain ‘adult’.

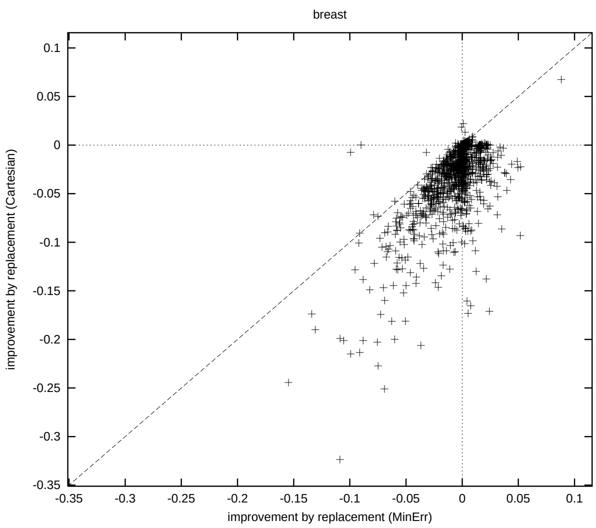


Figure 8.6: Comparing the improvement in resolution with Cartesian product and the MinErr procedure on the ‘breast’ domain.

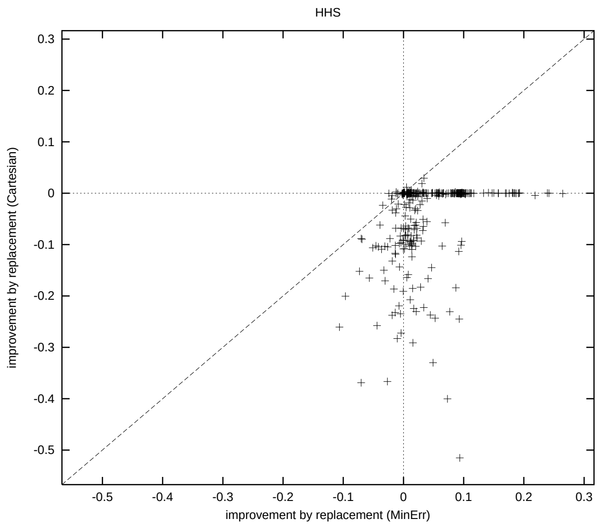


Figure 8.7: Comparing the improvement in resolution with Cartesian product and the MinErr procedure on 'HHS'.

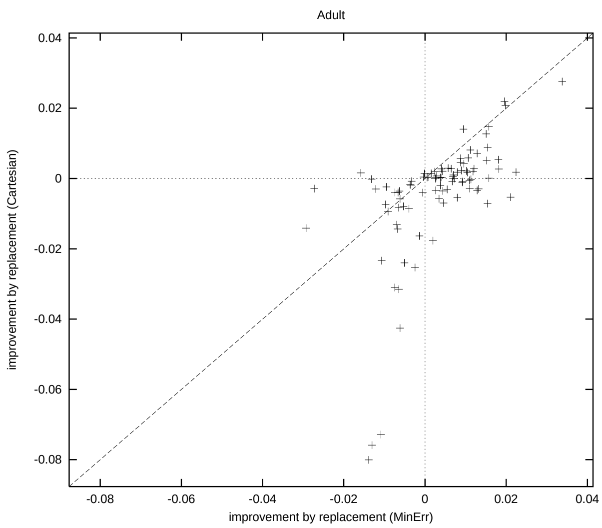


Figure 8.8: Comparing the improvement in resolution with Cartesian product and the MinErr procedure on 'adult'.

'adult'	Kullback-Leibler Div.		Error Rate (%)	
	Baseline	False + ME-R	Baseline	False + ME-R
NBC	0.416 ± 0.007	0.352 ± 0.006	16.45 ± 0.28	15.00 ± 0.27
LR	1.562 ± 0.023	0.418 ± 0.124	13.57 ± 0.20	13.24 ± 0.27
Linear SVM	—	—	—	—

'HHS'	Kullback-Leibler Div.		Error Rate (%)	
	Baseline	False + ME-R	Baseline	False + ME-R
NBC	2.184 ± 0.400	2.238 ± 0.394	56.25 ± 3.51	50.89 ± 4.08
LR	1.296 ± 0.106	1.352 ± 0.093	56.25 ± 2.72	58.04 ± 2.40
Linear SVM	1.083 ± 0.022	1.081 ± 0.020	55.36 ± 4.52	52.68 ± 6.21

'breast'	Kullback-Leibler Div.		Error Rate (%)	
	Baseline	False + ME-R	Baseline	False + ME-R
NBC	0.262 ± 0.086	0.187 ± 0.073	2.80 ± 0.72	1.40 ± 0.46
LR	0.016 ± 0.016	0.016 ± 0.016	0.14 ± 0.14	0.14 ± 0.14
Linear SVM	0.032 ± 0.021	0.032 ± 0.021	0.28 ± 0.19	0.28 ± 0.19

Table 8.2: Comparison of baseline results with those obtained after resolving false interactions using minimal-error attribute reduction.

In the 'breast' domain, false interaction resolution did not yield any improvement to SVM and LR. Probably feature weighting, which is inherent in these two methods, successfully eliminated the effects of false interactions in this domain.

In all cases, there were relatively few false interactions resolved. Hence, the complexity of the classifier did not increase significantly. On the other hand, we can view resolution of false interactions with attribute reduction as simplification of the classifier, and not vice versa. In fact, a human analyst would quite easily understand which attributes were joined from the interaction dendrogram.

8.6 Resolving True Interactions

We again used the minimal-error attribute reduction algorithm, but this time for resolving true interactions. As with false interaction resolution, we may have already eliminated an attribute when a new interaction involving that attribute is suggested. In such a case, we resolve its descendants, if only they have not been merged into a single attribute already.

An important concept in resolving true interactions is context. Assume A interacts with B, and B interacts with C. If we first resolved A and B with attribute reduction, we might have disposed of values which will prove useful when resolving $\overline{AB}$ with C. For that reason, we include all the attributes that significantly interact with either of the two attributes, whose interaction we are resolving, in the context. As a significance criterion for this purpose, we used the pragmatic test of interaction with the Cartesian product resolution method, reasoning that it is appropriate because the context attributes in the minimal-error reduction algorithm are not reduced either.

'adult'	Kullback-Leibler Div.		Error Rate (%)	
	Baseline	True + ME-R	Baseline	True + ME-R
NBC	0.416 ± 0.007	0.392 ± 0.007	16.45 ± 0.28	15.61 ± 0.25
LR	1.562 ± 0.023	1.564 ± 0.024	13.57 ± 0.20	13.58 ± 0.21
Linear SVM	—	—	—	—

'HHS'	Kullback-Leibler Div.		Error Rate (%)	
	Baseline	True + ME-R	Baseline	True + ME-R
NBC	2.184 ± 0.400	2.411 ± 0.379	56.25 ± 3.51	56.25 ± 3.51
LR	1.296 ± 0.106	1.319 ± 0.107	56.25 ± 2.72	57.14 ± 3.49
Linear SVM	1.083 ± 0.022	1.124 ± 0.038	55.36 ± 4.52	58.04 ± 3.57

'breast'	Kullback-Leibler Div.		Error Rate (%)	
	Baseline	True + ME-R	Baseline	True + ME-R
NBC	0.262 ± 0.086	0.171 ± 0.086	2.80 ± 0.72	1.40 ± 0.75
LR	0.016 ± 0.016	0.016 ± 0.016	0.14 ± 0.14	0.14 ± 0.14
Linear SVM	0.032 ± 0.021	0.016 ± 0.016	0.28 ± 0.19	0.14 ± 0.14

Table 8.3: Comparison of baseline results with those obtained after resolving true interactions using minimal-error attribute reduction.

The importance of resolving true interactions is lower than that of resolving false interactions in our domains. Still, the results improved, except in the 'HHS' domain. Apparently, there are either insufficient training instances, or simply no truly significant true interactions in this domain. The results worsened because a wrapper evaluation function requires a sufficient number of instances to be reliable at validating model variants. Perhaps the standard error in wrapper estimates should be considered, and only significant improvements put into effect.

In Table 8.4 we compare the results obtained with the minimal-error attribute reduction and those without attribute reduction. Attribute reduction always helped the naïve Bayesian classifier, only in one case the results worsened: in the domain 'HHS' for the SVM, where the wrapper estimation has problems correctly validating models. On 'adult', resolution without reduction created too many attribute values for LR to even function properly.

8.7 Experimental Summary

Our significance testing method is biased against attributes with many values, so we also examined the results by using the minimal-error attribute reduction. Minimal-error attribute reduction simplifies the joint Cartesian product attributes, which causes many more interactions to become significant.

The use of attribute reduction generally improved results for both false and true interactions. Although resolution is indeed required for resolving true interactions, we thought that false interactions could be better resolved by other, simpler means, such as attribute

'adult'	Kullback-Leibler Div.		Error Rate (%)	
	True + CART	True + ME-R	True + CART	True + ME-R
NBC	0.414 ± 0.007	0.392 ± 0.007	16.39 ± 0.26	15.61 ± 0.25
LR	—	1.564 ± 0.024	—	13.58 ± 0.21
Linear SVM	—	—	—	—

'HHS'	Kullback-Leibler Div.		Error Rate (%)	
	True + CART	True + ME-R	True + CART	True + ME-R
NBC	2.879 ± 0.482	2.411 ± 0.379	55.36 ± 4.07	56.25 ± 3.51
LR	1.467 ± 0.146	1.319 ± 0.107	57.14 ± 2.46	57.14 ± 3.49
Linear SVM	1.100 ± 0.023	1.124 ± 0.038	55.36 ± 5.02	58.04 ± 3.57

'breast'	Kullback-Leibler Div.		Error Rate (%)	
	True + CART	True + ME-R	True + CART	True + ME-R
NBC	0.229 ± 0.086	0.171 ± 0.086	0.84 ± 0.37	1.40 ± 0.75
LR	0.016 ± 0.016	0.016 ± 0.016	0.14 ± 0.14	0.14 ± 0.14
Linear SVM	0.016 ± 0.016	0.016 ± 0.016	0.14 ± 0.14	0.14 ± 0.14

Table 8.4: Comparison of true interaction resolution with attribute reduction (ME-R) and without attribute reduction (CART).

selection and weighting, inherent in LR and SVM. It was a surprise to us that false interaction resolution improved results also for these two methods.

We can confirm previous observations, e.g., in [MJ93], that significant true interactions are relatively rare, and can only be supported by a considerable amount of data. Such support is required both to improve classification performance by resolving them, and to demonstrate their significance. By our understanding, improving classification performance and demonstrating significance is indistinguishable.

Support vector machines and logistic regression performed well, but were not particularly robust. They often demonstrated excessive confidence. If they were made more robust, they would have a good chance of rarely losing in competing with the naïve Bayesian classifier. On the other hand, we can notice that sophisticated methods were in only one case ('breast') better than simple methods.

Our results should be viewed as preliminary, but encouraging. There are many possible extensions, and improvements that could be worth considering:

- The use of interaction gain as the guiding principle might be not ideal: we could, instead or in addition to it, use the quality gain with the appropriate attribute reduction method for resolution, which we used for pragmatic interaction significance testing.

- We could speed up the procedure by resolving all the significant interactions, rather than performing the time-consuming model search iteratively though the list of candidate interactions.

- We did not use the classification tree learning algorithms as a attribute resolution algorithm, to replace minimal-error attribute reduction. If we did that, we would check that the thus obtained segmentations achieve good quality with respect to probabilistic evaluation functions.

- If two continuous attributes interact, the segmentation obtained with a classification tree learning algorithm may be a good multivariate discretization of both attributes.

- It would be interesting to compare feature selection and feature weighting with resolution of false interactions. We included logistic regression and SVM to be able to investigate whether feature weighting, inherent in these two procedures, obsoletes the false interaction resolution, but this was apparently not the case.

- We did not perform any parameter tuning, and this could affect the results. However, we tried to be fair by using recommended parameter values in all cases, and not tuning any of them.

CHAPTER 9

Conclusion

A true skeptic is skeptical about his skepticism.

In Ch. 2 we started with a framework for probabilistic machine learning. We have shown that probabilistic classifiers have many useful properties. They are able to estimate the precision of their predictions which assures cost-based decision making without requiring to know the utility function or a cost matrix prior to learning. We briefly explored the problem of uncertainty and ignorance, and suggested that the classifier's estimate of uncertainty should match its actual ignorance on unseen data. It is sometimes useful to even estimate our ignorance about our ignorance, and for that purpose we proposed the notion of higher-order uncertainty, sketching its possible formalization.

As classification accuracy is unfair to probabilistic classifiers, we surveyed a number of possible evaluation functions. We have presented some decision-theoretic measures, but they require the knowledge of the utility function. Collapsing a receiver operating characteristic into a single number by computing the area under it involves assuming that all cost matrices are equally likely, besides the analysis is time-consuming. We have presented an analogy with gambling, where a proper measure between boldness and timidity determines long-term success, and have decided to use the Kullback-Leibler divergence as the evaluation function. We strongly stressed that an classifier should only be evaluated on the instances that were not used in the process of learning.

We then tried to unify a number of probabilistic machine learning methods in a single framework, and the result are four fundamental functions. Given a set of instances, an zero-descriptor estimation function attempts to capture the probability distribution of label values with no information beyond knowing the class of each instance. Classifier's output is such a distribution, which we formally denote with the concept of a model.

A segmentation function divides instances on the basis of their attribute values into groups. For each group separately, the estimation function creates a model. A voting function is able to join multiple models into a single one, without considering the attribute values.

Estimation can also consider instance descriptors. Here, the estimation function receives one or more descriptors of each instance in addition to its class. The descriptors need not correspond to attributes, instead a projection function should find a small number of informative projections from the attribute space into some new descriptor space. The most frequently used projection is linear, where the descriptor is the distance of the instance from some hyperplane in the attribute space.

Before creating our own definition of interactions, we surveyed a number of related fields in Ch. 3. The concept of interactions appears in statistics, in categorical data analysis, in probability theory, in machine learning, in pattern recognition, in game theory, in law, in economics, and in the study of causality. The causal explanation of an interaction is based on the concept of a moderator: a moderator attribute is moderating the influence the cause has on the effect. The probabilistic interpretation of an interaction is that it is a dependence between a number of attributes. In machine learning there is a vague notion that considering one attribute at a time is myopic, and fully justified only when there are no interactions.

From this heritage, and using the framework of Ch. 2, along with the specific example of the naïve Bayesian classifier, we present a view of an interaction that is based on intrinsic limitations of the segmentation function in Ch. 4. To be able to improve classification performance with the knowledge of interactions, it is required to resolve it. Although we admitted that there are better resolution methods, and address some of them later in this work, we initially focused on the Cartesian product of a pair of attributes. Replacing the original attributes with their Cartesian product is the step which enables a naïve Bayesian classifier to take advantage of an interaction. Of additional interest may be an assessment the naïve Bayesian classifier's limitations: some of them can be solved without resolving interactions, e.g., by attribute selection and weighting.

Instead of introducing special tests for interactions, we suggested that an interaction is significant only when resolving it improves the classification performance. We admitted that this definition is dependent upon the learning algorithm, the evaluation function, and the quantity of data, but it is completely sensible if we choose to pursue classification performance. We stressed that only those interactions that involve the label as one of the attributes are interesting for classification problems.

The main four types of interactions are true, false, problematic, and non-existent interactions. A pair of falsely interacting attributes will provide us with the same information about the class. A pair of truly interacting attributes provides information about the label which is not visible without the presence of both attributes. Finally, the problematic interactions are those cases, when the type of an interaction is dependent on some attribute value. One approach is to create several non-problematic attributes from a single problematic one, introducing a new binary attribute for an attribute value. Non-existent interactions the assumption of most simple machine learning algorithms. We briefly touched upon methods for resolving false interactions, since latent variable analysis and feature weighting may be preferable to resolving with the Cartesian product.

In Ch. 5, we listed the existing methods from machine learning which have been used for searching patters in data that resemble interactions. We also proposed our own approach, based on information theory. Our interaction gain can be seen as a generalization of information gain to from two to three attributes. We showed in Ch. 6 that interaction gain is useful for identifying pairs of attributes that do not interact, interact falsely or truly. Furthermore, we presented a set-theoretic explanation of information content in attributes which might illuminate the problem. We pointed out several similarities between our approach and respective approaches in game theory and in quantum information theory.

Other experiments in Ch. 6 focused on the relation between interaction gain and the pragmatic interaction test. We found out that strong false interactions and strong true interactions yield a larger improvement with respect to the pragmatic test. If we search for interactions with the intention of improving the classification performance, we have shown that internal cross-validation provides reliable results. With respect to the naïve Bayesian classifier, it is usually more beneficial to replace attributes in resolution, rather than add the resolved attributes to the initial set of attributes. Furthermore, if we desire simplifying and speeding up the pragmatic test, we can exclude other attributes while we focus on a specific pair.

We tried to illuminate the relationship between ontological attribute structures that people use to organize attributes in the domain. We found that the neighborhood of attributes structures does not always imply either false or true interactions. However, since these structures represent the background knowledge, the classifier could either use them to speed up search for interactions by first considering attributes neighboring in the ontology. On the other hand, for detailed analysis, a user would probably be more surprised by an unexpected interaction between distant attributes than by expected interactions among neighbors.

A machine learning system should not pursue mere precision of its predictions, but should also try to provide a human analyst with insight about the characteristics of the problem domain. For that reason, we have investigated interaction analysis in Ch. 7. We suggested visualization of true interactions in an interaction graph, whereas the false interactions are more informatively visualized in an interaction dendrogram. We have performed some experiments with the pragmatic test of significance of an interaction and found out that only a small number of true interactions are significant.

Finally, in Ch. 8 we resolved both true and false interactions. This improved classification performance of the naïve Bayesian classifier, logistic regression, and of support vector machines. We found that, in contrast to the interaction gain, the resolution with a Cartesian product is dependent on the number of attribute values. We proposed using an attribute reduction algorithm, such as the minimal-error attribute reduction from the field of function decomposition, which was used to resolve interactions in the classification performance experiments.

Extended Abstract in Slovene Language

POGLAVJE 10

Interakcije med atributi v strojnem učenju

Povzetek

Za odločanje o nekem problemu imamo po navadi na voljo več podatkov. Hkrati bi si želeli obravnavati le tiste, ki so med seboj res povezani. Formalizacija te povezanosti so interakcije. Neka skupina podatkov je med seboj v interakciji, če njihovih medsebojnih povezanosti ne moremo več popolnoma razumeti, ko odstranimo kateregakoli od podatkov. Interakcije ločimo na sodejavnosti in soodvisnosti. Pri sodejavnostih se nam nekateri vzorci v podatkih odkrijejo le, če imamo na voljo tudi ostale podatke. Pri soodvisnostih pa ugotovimo, da nam več podatkov poda iste informacije, zaradi česar moramo paziti, da jim ne damo prevelike teže. Interakcije so po definiciji nepoenostavljive: ne moremo jih razbiti na več ločenih interakcij. Če to lahko naredimo, to niso interakcije.

V tem magistrskem delu preučimo več problemov povezanih z interakcijami. To zahteva interdisciplinaren pristop, saj so interakcije temeljni problem na več področjih, od strojnega učenja, statistike do teorije iger in kvantne fizike. Preučimo obstoječe metode za odkrivanje interakcij in predlagamo izračun interakcijskega prispevka, s sposobnostjo razlikovanja med sodejavnimi, soodvisnimi ter neodvisnimi skupinami treh atributov. Ta izračun je posplošitev informacijskega prispevka oziroma medsebojne informacije. Predlagamo pragmatični test pomembnosti interakcij: upoštevanje interakcije prispeva k boljšim rezultatom neke družine algoritmov strojnega učenja le, če je ta interakcija pomembna. Take so le izrazite sodejavnosti in soodvisnosti. Prikažemo, kako lahko uporabniku na vizualen način predstavimo interakcije v danem klasifikacijskem problemu in kako lahko nekatere najpopularnejše algoritme strojnega učenja z upoštevanjem interakcij izboljšamo.

Ključne besede

- strojno učenje
- klasifikacija, razpoznavanje vzorcev, uvrščanje
- interakcija, soodvisnost, sodejavnost, odvisnost, neodvisnost
- konstruktivna indukcija
- mere nečistoče, informacijski prispevek atributa, ocenjevanje kvalitete atributov
- Bayesov klasifikator, naivni Bayesov klasifikator, delno naivni Bayesov klasifikator
- teorija o informacijah, entropija, relativna entropija, medsebojna informacija

10.1 Uvod

Ko poskušamo ljudje razumeti podatke, jih ne obravnavamo v celoti. Raje jih razbijemo na manjše, bolj obvladljive koščke. To deljenje problemov na podprobleme je osnova večine postopkov strojnega učenja. Čeprav je redukcionističen, deluje.

A obstajajo delčki znanja in vzorci v naravi, ki izginejo, če jih poskušamo razrezati. Moramo jih obravnavati kot celoto. Po drugi strani pa spet ne moremo vsega obravnavati kot celoto, saj je poenostavljanje ključno za zmožnost posploševanja. Čemu bi jemali krvne vzorce, če vendar lahko gripo diagnosticiramo le z merjenjem telesne temperature?

Da bi prerezali ta gordijski vozel, vpeljimo koncept *interakcij*. Interakcije so tisti vzorci, ki jih ne moremo razumeti po koščkih, le v celoti. Problem lahko prosto razbijamo na koščke, če le ne razbijemo interakcij.

Predstavljajmo si marsovskega bankirja, ki bi rad stranke razdelil v tri skupine: goljufe, povprečneže in molzne krave. Bankir ima na voljo množico atributov, ki stranko opisujejo: starost, poklic, izobrazbo, lanskoletne dohodke, letošnje dohodke in dolgove.

Bankir zaposluje več analitikov. Najraje bi predpostavil, da so vsi atributi med seboj neodvisni, a hkrati tudi vsi povezani z razredom. Potem bi lahko vsakemu analitiku predal v študij le po en atribut. Vsak analitik je strokovnjak o odnosu med svojim atributom in razredom, izkušnje pa je pridobil na velikem številu primerov, ki jih je že preučeval. Ko analitiki odhitijo s podatki, med seboj ne komunicirajo: samo na podlagi svojega atributa se poskušajo odločiti, v katerem razredu je nova stranka.

Bankir čez nekaj časa skliče vse analitike in jim pove, naj glasujejo za posamezen razred. Če nek analitik čuti, da nima dovolj podatkov, mu je dovoljeno, da se vzdrži glasovanja. Bankir izbere razred, ki je dobil največ glasov. V primeru, da je takih razredov več, izbere najslabšega: vsekakor je boljše, da obravnava molzno kravo kot goljufa, kot pa da bi klečeplazil pred goljufom.

Žal sta tu dve težavi. Več analitikov lahko preučuje iste informacije. Na primer, ko enkrat poznamo strankin poklic, nam njena izobrazba ne bo povedala kaj bistveno novega. Zato bo ta plat stranke dobila preveliko težo pri glasovanju. Takim atributom pravimo, da so *soodvisni*.

Druga težava je v tem, da nam lanskoletni dohodki in letošnji dohodki ne povejo toliko, kot bi nam povedali, če bi namesto tega vedeli, kako so se dohodki spremenili. Na primer, stranke se včasih spreobrnejo v goljufe, če se jim dohodki na hitro zmanjšajo. Takim atributom pravimo, da so *sodejavni.*

Interakcije so pojem, ki združuje sodejavnosti in soodvisnosti. Ko imamo interakcije, se splača, da analitiki med seboj sodelujejo, da bi dosegli boljše rezultate. Malo bolj realistično: en sam analitik naj obdeluje več atributov in jih združi v eni sami formuli. Na primer, dva atributa o dohodkih zamenjamo z indeksom padca dohodkov, kar je novi atribut, na podlagi katerega analitik sprejme svojo odločitev.

Naš primer precej realistično opisuje delovanje računalnika, ko ta preučuje podatke, mogoče pa tudi naše možgane, ko sprejemajo odločitve. Bankirjev pristop je precej podoben znanemu *naivnemu Bayesovemu klasifikatorju*, katerega glavna omejitev je ravno ta, da predpostavlja, da interakcij ni. Resda interakcije, še posebej sodejavnosti, niso najbolj pogoste, zato so se strokovnjaki dolgo čudili solidnim rezultatom, ki jih je tako enostavna metoda dosegla v primerjavi z veliko bolj zapletenimi alternativami.

Naše delo se bo osredotočilo na naravni problem iskanja sodejavnosti in soodvisnosti v podatkih za dan klasifikacijski problem. Če nam bo uspelo, bo bankir najprej uporabil naš postopek, da ugotovi, kateri atributi so sodejavni in kateri soodvisni. Potem bo lahko delo bolje razdelil med svoje analitike. Zato je naš prvi cilj čimbolj razumljivo prikazati interakcije v domeni človeku, ki podatke preučuje, po možnosti kar grafično.

Tak postopek bi tudi bil koristen postopkom strojnega učenja. Ti bi lahko z njegovo pomočjo ugotovili, kje so zapleteni podproblemi, te razrešili z zapletenimi postopki, ki hkrati obravnavajo več atributov. Tam, kjer pa ni komplikacij, bi uporabili enostavne postopke, na primer naivnega Bayesovega klasifikatorja. Videli bomo, da imajo enostavni postopki prednosti, ki niso vezane na samo enostavnost: ker predpostavimo manj in ker ne drobimo podatkov, jih lahko bolj zanesljivo opisujemo in merimo. Naš drugi cilj je zato izboljšati objektivno kvaliteto algoritmov strojnega učenja, kot jih merimo s funkcijami cenilkami.

10.2 Negotovost v strojnem učenju

Večji del strojnega učenja temelji na predstavitvi učnih primerov z *atributi*, primere pa uvrščamo v več *razredov.* Naloga učenja je naučiti se uvrščati učne primere v razrede na podlagi njihovih atributov. Temu pravimo uvrščanje ali *klasifikacija.* Rezulat učenja pa je, očitno, *znanje*, ki ga tu predstavimo v obliki *klasifikatorja.*

Nalogi učenja sta dve: po eni strani želimo doseči, da bi naš klasifikator pravilno uvrstil vse primere, še posebej tiste, na katerih se ni učil. Po drugi strani pa bi si želeli, da nam zgradba klasifikatorja pove nekaj koristnega o naravi klasifikacijskega problema.

Sprva se je mislilo, da pravila in klasifikacijska drevesa nudijo ljudem najboljši vpogled v problem, saj temeljijo na logiki in pravilih, ki jih dobro poznamo iz jezika. Potem se je izkazalo, da ljudje velikokrat raje vidijo znanje v obliki vplivov in verjetnosti, ki ga zajame npr. naivni Bayesov klasifikator, še posebej če ga na vizualen način predstavimo v nomogramu. Vizualizacija je način, s katerim tudi znanje numeričnih ali subsimbolnih postopkov predstavimo človeku na razumljiv način, ob tem pa nismo vkalupljeni v omejeni jezik pravil.

Poznamo več vrst atributov. Po vlogi ločimo *navadne atribute* in *razredni atribut.* Razredni atribut je po obliki tak kot navadni atributi, loči ga le vloga. Vsak atribut ima lahko več vrednosti. Če so vrednosti števila, takim atributom pravimo *številski* ali numerični, ki jih potem delimo na *števne* ali diskretne ter *zvezne* atribute, v odvisnosti od množice števil, ki jo uporabljamo. Če so vrednosti elementi neke urejene končne množice, atributom pravimo *urejeni* ali ordinalni atributi; če končna množica vrednosti ni urejena, so atributi *neurejeni* ali nominalni.

V tem besedilu se bomo osredotočili na neurejene atribute, saj je definicija urejenosti hud oreh. Namreč, urejenost lahko izhaja iz števil, lahko pa jo tudi definiramo po svoje, na primer preko vpliva na razred. Pravzaprav, če jo definiramo po svoje, lahko precej pridobimo.

Pri klasifikacijskih problemih so vrednosti razrednega atributa elementi končne množice, pri regresijskih problemih pa so vrednosti razrednega atributa elementi neke množice števil. Vrednosti razrednega atributa pri klasifikacijskih problemih so razredi.

Učni algoritem je funkcija, ki preslika nek *klasifikacijski problem* v klasifikator. Klasifikacijski problem je neke množica učnih primerov ter opisi atributov. Atributi so funkcije, ki nam učni primer preslikajo v vrednosti atributov, kot so opisane zgoraj. Klasifikator pa je funkcija, ki preslika učni primer v razred. Tu bomo ločili *diskriminativne*, *stohastične* in *verjetnostne* klasifikatorje. Diskriminativni preslikajo učne primere v točno določen razred. Stohastični lahko za isti učni primer ob različnih prilikah vrnejo različne razrede: v skladu z neko verjetnostno porazdelitvijo. Verjetnostni klasifikatorji nam vrnejo kar verjetnostno porazdelitev samo in te imamo najraje.

10.2.1 Negotovost

Nesmiselno je, da bi ucenjevali klasifikatorje po njihovi uspešnosti uvrščanja primerov, ki so jih že videli: saj bi si jih lahko vendar dobesedno zapomnili. Izziv je klasificirati primere, ki jih učni algoritem še ni videl. Če smo se učili na *učni množici* primerov, svoje znanje preverjamo na *testni množici*. Čeprav je to težko, je enako težko za vse klasifikatorje, zato jih lahko med seboj primerjamo.

Težava se pojavi takrat, ko za rešitev nekega klasifikacijskega problema nimamo dovolj primerov, ali pa ko ta problem ni determinističen. Čeprav bi lahko diskriminativni klasifikator vedno predložil najbolj verjeten razred, bi si bolj želeli verjetnostnega, ki bi opisal možnosti pojavitve določenega razreda z verjetnostmi.

Koncept verjetnosti uporabljamo v več situacijah. Prva je *negotovost* in ta je subjektivne narave, saj izraža našo negotovost o tem, kaj je. Drugi dve sta *neznanje* in *nepredvidljivost*, ki sta objektivni lastnosti. Neznanje je neobhodna posledica našega nepopolnega poznavanja resničnosti. Nepredvidljivost se nanaša na to, da tudi če bi imeli vse podatke, ne bi mogli nečesa predvideti. Ker je filozofsko sporna, bomo govorili le o neznanju.

Cilj klasifikatorja je, da se njegova negotovost kot ocena lastnega neznanja ujame z dejanskim neznanjem. Primeru, ko je negotovost 'manjša' od neznanja, pravimo preveliko prileganje podatkom ali *bahavost* (angl. overfitting). Ko je negotovost 'večja' od neznanja, gre za premajhno prileganje podatkom ali *plašnost* (angl. underfitting).

10.2.2 Vrednotenje klasifikatorjev

Najpopularnejša metoda za vrednotenje klasifikatorjev, *klasifikacijska točnost*, ne nagrajuje klasifikatorjev, ki pravilno ocenijo svoje neznanje, saj je bila zamišljena za diskriminativne ali bahave klasifikatorje. Zato bi si želeli metod, ki bi to upoštevale.

Težavo bi lahko rešili tako, da definiramo *funkcijo koristnosti* (angl. *utility function*), ki oceni odgovor nekega klasifikatorja in ga primerja s pravilnim odgovorom. Če verjetnostni klasifikator funkcijo koristnost pozna, ji lahko svoj odgovor prilagodi, da bo čimbolj koristen. Na primer, hujša napaka je, da odpustimo bolnega pacienta, kot pa da pregledamo zdravega. Recimo, da funkcijo koristnosti predstavimo s cenovno matriko M: $\mathsf{M}(d(\mathbf{i}), C(\mathbf{i}))$ je cena, ki jo mora klasifikator d plačati pri primeru $\mathbf{i}$, če je pravilni razred $C(\mathbf{i})$. Za klasifikacijsko točnost gre takrat, ko je

$$\mathsf{M}(c_i, c_j) = \begin{cases} 0 & \text{če } i = j, \\ 1 & \text{če } i \neq j. \end{cases}$$

Verjetnostni klasifikator lahko izbere optimalni odgovor za poljubno M po naslednji formuli:

$$c_o = \arg\min_{\tilde{c} \in \mathcal{D}_C} \sum_{\hat{c} \in \mathcal{D}_C} \Pr\{d(\mathbf{i}) = \hat{c}\} \mathsf{M}(\tilde{c}, \hat{c}).$$

Tu je $\mathcal{D}_C$ množica vrednosti razrednega atributa C. Vseeno pa uporabnik vsekakor raje vidi napovedi klasifikatorja v obliki ocene negotovosti kot v obliki enega samega odgovora, četudi za tega računalnik pravi, da je optimalen.

Žal pa klasifikator ponavadi ne ve, kakšna je funkcija koristnosti. Zato bi si želeli neke *funkcije cenilke*, ki bi ocenila odstopanje negotovosti od neznanja. Tu ima lepe lastnosti *relativna entropija* ali *Kullback-Leiblerjeva divergenca* [KL51]. KL divergenco merimo med dvema verjetnostnima porazdelitvama razredov, med dejansko $P = P(C(\mathbf{i})|\mathbf{i})$ in predvideno $Q = \Pr\{d(\mathbf{i})\}$:

$$D(P||Q) = \sum_{c \in \mathcal{D}_C} P(C(\mathbf{i}) = c) \log \frac{P(C(\mathbf{i}) = c)}{\Pr\{d(\mathbf{i}) = c\}}.$$

Relativna entropija je hevristika, ki nagradi oboje, natančnost in priznanje neznanja. Logaritem lahko razumemo kot logaritmično funkcijo koristnosti. To je omenjal že Daniel Bernoulli, ki je opazil, da je sreča ljudi približno logaritmična funkcija zaslužka, in zato predlagal logaritmično funkcijo koristnosti že leta 1738 [Ber38, FU].

10.2.3 Gradnja klasifikatorjev

Obstaja nekaj bistvenih funkcij, iz katerih gradimo klasifikatorje:

Ocenjevanje Klasifikator lahko predstavimo kot funkcijo, ki preslika atributni opis primera v nek *model*, model pa ni nič drugega kot funkcija, ki slika iz vrednosti razrednega atributa v verjetnosti. Modeli niso nič posebnega, so le verjetnostne porazdelitve. Te so lahko parametrične, kot sta na primer Gaußova in logistična, ali pa neparametrične, kot je na primer histogram. Za določanje parametrov modela uporabimo le obstoječe postopke ocenjevanja na podlagi pretvarjanja frekvenc v verjetnosti.

Projekcija Na podlagi vrednosti atributov določimo neko novo zvezno vrednost, ki služi kot spremenljivka, in iz te model slika v verjetnostno porazdelitev razrednega atributa. Na primer, ko imamo dva razreda, logistična regresija slika vhodne atribute v oddaljenost od neke hiperravnine, ki poskuša ločiti primere enega razreda od drugega. To oddaljenost pa povežemo z verjetnostjo posameznega razreda z logistično porazdelitvijo. V primeru, da je hiperravnina uspela pravilno ločiti vse primere, je ta poradelitev stopničasta.

Delitev Primere razdelimo na več skupin, za vsako skupino pa ločeno ocenjujemo model. Na primer, klasifikacijsko drevo razdeli vse primere na nekaj skupin, vsaki skupini pa pripiše nek neparametrični model. Delitev si lahko predstavljamo tudi kot diskretno projekcijo, kjer projeciramo primere v neko novo diskretno spremenljivko, glede na katero z neparametričnim modelom ocenjujemo porazdelitev razrednega atributa. Klasifikacijsko drevo je konkreten primer delitvene funkcije.

Glasovanje Če imamo večje število modelov, jim lahko omogočimo, da med seboj glasujejo in s tem proizvedejo nov model, ne da bi ob tem uporabljali kakršenkoli atribut. Enostaven primer je naivni Bayesov klasifikator, kjer enakopravno glasujejo modeli, vsak od katerih pripada svojemu atributu. Vsak atribut pa ni nič drugega kot segmentator, kjer se vsaki vrednosti atributa priredi neparametrični model. Ta model šteje koliko primerov posameznega razreda ima določeno vrednost atributa.

10.3 Interakcije

10.3.1 Vzročnost

Najenostavneje si interakcijo predstavljamo v kontekstu vzročnosti. Na sliki 10.1, prirejeni iz [JTW90], vidimo različne vrste povezav med atributi A,B in C. Pri tem si lahko B predstavljamo kot razredni, A in C pa kot navadna atributa. Z interakcijami ima največ veze *nadzorovana povezava*. Tu je C nadzornik in nadzoruje vpliv, ki ga ima vzrok A na posledico B. Seveda je velikokrat težko ločiti vzrok od nadzornika, zato je velikokrat bolje, da ne poskušamo ločiti vlog atributov, ki so v interakciji.

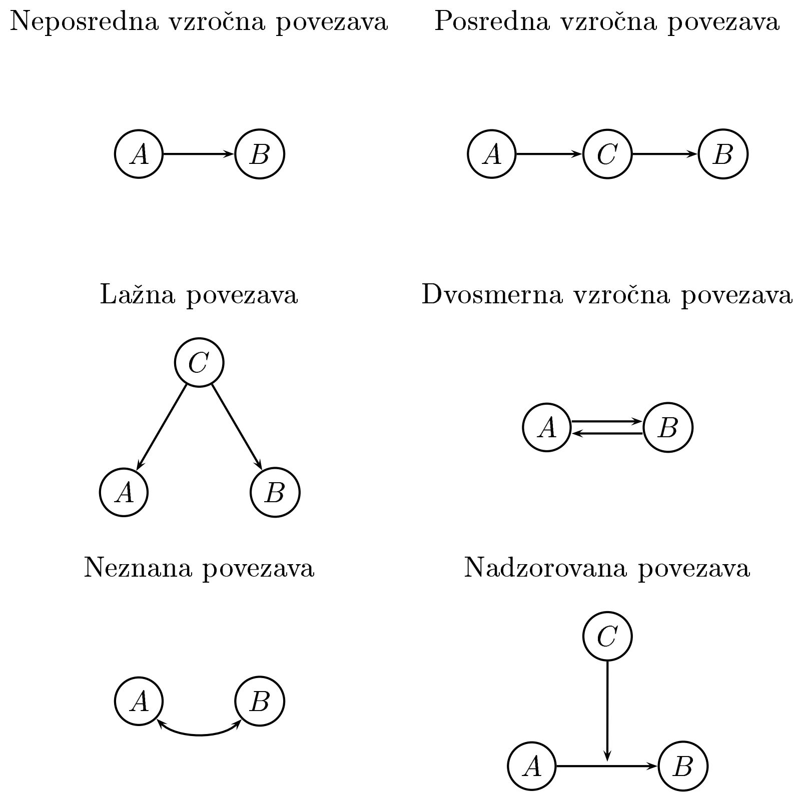
10.3.2 Odvisnost

Če so atributi med seboj neodvisni, med njimi ni interakcij. Pri dogodkih A, B in C to velja takrat, ko je $P(A,B,C) = P(A)P(B)P(C)$. Tu moramo paziti, lahko velja $P(A,B) = P(A)P(B)$, ne velja pa več $P(A,B|C) = P(A|C)P(B|C)$. Rečemo lahko, da čeprav sta atributa A in B neodvisna, nista neodvisna glede na C, razen če $P(A,B|C) = P(A|C)P(B|C)$. Težava s to definicijo je, da skoraj nikdar ne moremo natančno opisati skupne verjetnosti kot produkta posamičnih, zato se moramo zateči k različnim hevristikam in statističnim testom.

Znan primer napak zaradi neupoštevanja interakcij v tem kontekstu je *Simpsonov paradoks*, do katerega pride, ko dobimo nasprotne rezultate pravilnim, če ne upoštevamo tretjega atributa. Omenimo primer iz [FF99] o tuberkulozi. Zanima nas primerjava zdravstvenih sistemov v mestih New York and Richmond, kar merimo s smrtnostjo zaradi tuberkuloze. Imamo naslednjo tabelo:

kraj	živeli	umrli	p_{smrt}
New York	4758005	8878	0.19%
Richmond	127396	286	0.22%

Zdi se nam, da je zdravstveni sistem v mestu Richmond slabši. A poglejmo, kaj se zgodi, ko v obravnavo vključimo še podatke o barvi kože:



Slika 10.1: Šest tipov vzročnih povezav

belci				ne-belci			
kraj	živeli	umrli	p_{smrt}	kraj	živeli	umrli	p_{smrt}
NY	4666809	8365	0.18%	NY	91196	513	0.56%
Rich.	80764	131	0.16%	Rich.	46578	155	0.33%

Torej je zdravstveni sistem v mestu Richmond boljši kot v mestu New York, za oba barvna odtenka. Razlika se mogoče pojavi zaradi različnega razmerja med barvnimi odtenki v teh mestih.

10.3.3 Omejitve klasifikatorjev

Naivni Bayesov klasifikator predpostavi, da lahko zapišemo verjetnost $P(X,Y,Z|C)$ z glasovanjem med $P(X|C)$,$P(Y|C)$ in $P(Z|C)$. Čisto pragmatično lahko definiramo interakcijo med dvema atributoma in razredom kot primer, ko bi dosegli boljše rezultate z združenim obravnavanjem dveh atributov. Torej, če je med X,Y in C interakcija, bi potem zapisali $P(X,Y|C)P(Z|C)$.

Zdi se čudno, a čeprav poskuša metodologija Bayesovih mrež [Pea88] predstaviti neko verjetnostno porazdelitev na veliko atributih kot produkt verjetnostnih porazdelitev na manj atributih, njena predstavitev z grafi ne more ločiti med $P(X,Y|C)P(Z|C)$ in $P(X,Y,Z|C)$, razen če ne uvedemo novega atributa X,Y. Uporabljena je le druga možnost.

Namesto statističnih testov signifikantnosti tu enostavno uporabimo funkcijo cenilko (angl. *evaluation function*). Če klasifikator z interakcijo deluje boljše kot brez nje, interakcijo priznamo, sicer pa je ne. Seveda je ta definicija odvisna od osnovnega klasifikatorja, ki smo ga uporabili. Smiselna je za enostavne klasifikatorje, kot so recimo naivni Bayesov klasifikator, na kateroga smo se opirali pri zgornji razlagi, linearna ali pa logistična regresija ter podobne. Lahko se zgodi, da bodo trije atributi v interakciji pri naivnem Bayesovem klasifikatorju, ne pa tudi pri logistični regresiji.

Čeprav se ne bomo spustili v podrobnosti, lahko definicijo posplošimo na neko družino klasifikatorjev, ki so zgrajeni s principi iz razdelka 10.2.3. Take pragmatične interakcije se pojavijo natanko takrat, ko moramo neko skupino atributov obravnavati skupaj znotraj neke delitvene funkcije S ali projekcijske funkcije F. Z delitvenimi funkcijami lahko dosežemo boljše rezultate edino tako, da namesto $S(A)$ in $S(B)$ uporabimo $S(A, B)$.

10.3.4 Teorija informacije

Navkljub smiselnosti pragmatične definicije bi si želeli definicijo interakcij, ki bi bila do neke mere neodvisna od klasifikatorja in bi nam nudila nek vpogled v interakcije. Poglejmo si *informacijski prispevek* atributa A k razredu C iz [HMS66]:

$$\mathrm{Gain}_C(A) = H(C) + H(A) - H(\overline{AC}) = \mathrm{Gain}_A(C). \tag{10.1}$$

Upoštevajmo, da ta definicija ne razlikuje med vlogo posameznih atributov: prispevek A k C je enak kot prispevek C k A. Tu je $\overline{AC}$ kartezični produkt atributov A in C. Opazimo lahko tudi, da je informacijski prispevek $\mathrm{Gain}_C(A)$ identičen *medsebojni informaciji* $I(A; C)$ med A in C.

Tu uporabljamo koncept *entropije*, ki je mera informacijske vsebine nekega vira informacij [Sha48]. Ker je atribut A vir informacij, jo zanj definiramo kot:

$$H(A) = -\sum_{a \in \mathcal{D}_A} P(a) \log P(a).$$

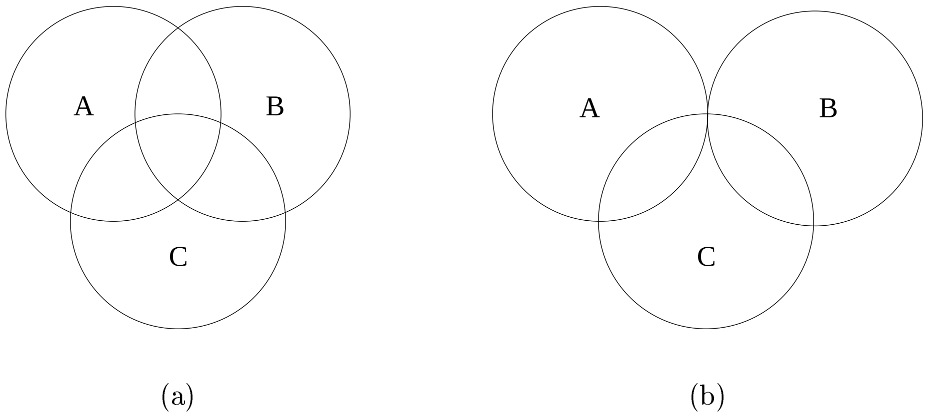
Če uporabimo dvojiški logaritem, entropijo merimo v bitih, za naravni logaritem pa v natih.

Če jemljemo informacijski prispevek kot oceno 2-interakcije, lahko potem na podoben način definiramo oceno 3-interakcije, ki mu bomo rekli *interakcijski prispevek*:

$$\begin{aligned} IG_3(A, B, C) &:= I(\overline{AB}; C) - I(A; C) - I(B; C) \\ &= \mathrm{Gain}_C(\overline{AB}) - \mathrm{Gain}_C(A) - \mathrm{Gain}_C(B) \\ &= H(\overline{AB}) + H(\overline{AC}) + H(\overline{BC}) \\ &\quad - H(\overline{ABC}) - H(A) - H(B) - H(C). \end{aligned} \tag{10.2}$$

Podobno kot prej, razredni atribut nima posebne vloge. Na interakcijski prispevek lahko gledamo kot na nekakšno posplošitev pojma medsebojne informacije na tri informacijske vire ter informacijskega prispevka na tri atribute.

Interakcijski prispevek najlažje razložimo z metaforo števila elementov množic. Recimo, da je informacija vsebina nekega atributa A množica, informacijsko vsebino pa merimo z $-H(A)$: manjša kot je atributova entropija, več informacij nam da. Nikoli pa



Slika 10.2: Vennov diagram treh atributov v interakciji (a), in dveh pogojno neodvisnih atributov glede za razred (b).

se ne more zgoditi, da bi potem vedeli manj kot prej. Dva atributa skupaj v svoji uniji vsebujeta $-H(\overline{AB})$, kar je smiselno, saj vemo, da je entropija dveh virov vedno manjša ali enaka vsoti obeh. Informacijski prispevek na nek način meri presek med množicama, ravno tako kot $|A \cap B| = |A| + |B| - |A \cup B|$. Interakcijski prispevek pa lahko primerjamo z $|A \cap B \cap C| = |A| + |B| + |C| - |A \cup B| - |B \cup C| - |A \cup C| + |A \cup B \cup C|$, glej sliko 10.2(a). Paziti pa moramo, saj je lahko interakcijski prispevek tudi negativen [Ved02]. To bomo razložili v naslednjem razdelku. Mimogrede, naivni Bayesov klasifikator predpostavi, da atributa A in B prispevata informacijo o C nekako takole kot v sliki 5.1(b).

Paziti moramo, ker nas lahko interakcijski prispevek še vedno zavede, če imamo več kot tri atribute. Spomnimo se Simpsonovega paradoksa. Informacijski prispevek, kot smo ga definirali, je tudi primeren le za ocenjevanje 3-interakcij.

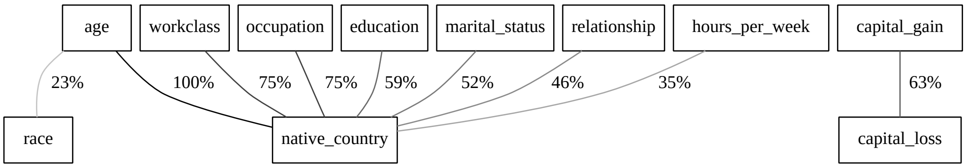
10.4 Vrste interakcij

10.4.1 Sodejavnosti

Pri sodejavnostih velja, da nam atributa A in B skupaj povesta več o C, kot bi nam, če bi le glasovala. Primer popolne sodejavnosti je znani problem ekskluzivnega ALI $c := a \neq b$:

A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

A in B sta vsak zase popoloma neodvisna od C in zato popolnoma neuporabna kot glasovalca, a ko jih damo skupaj, bosta C napovedala pravilno. Ni pa nujno, da morata za sodejavnost biti oba atributa sprva neuporabna. Poglejmo si primer navadnega OR, kjer je $c := a \vee b$:



Slika 10.3: Interakcijski graf atributov v domeni 'adult'.

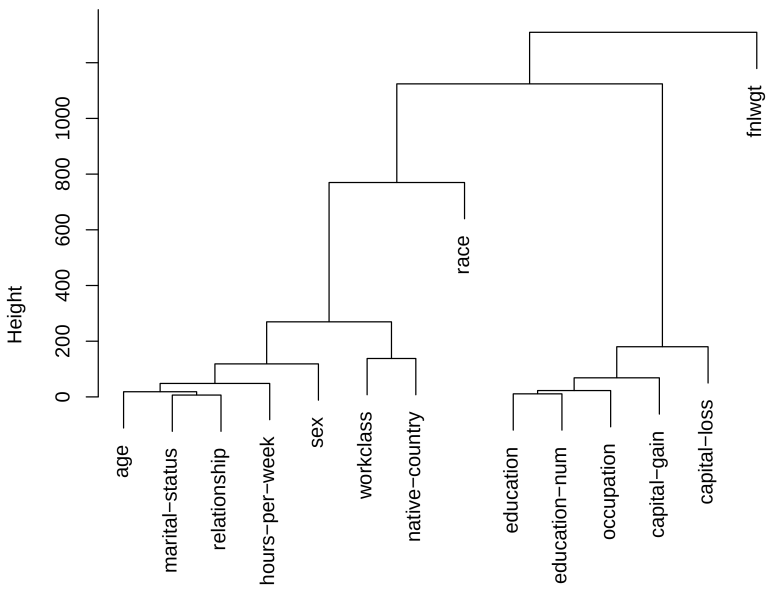
A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

Četudi nam A in B pomagati pri napovedovanju C, bi z glasovanjem naivni Bayesov klasifikator za primer $a = 0, b = 0$ ocenil verjetnost razreda kot $P(c_0) = 1/2$, a pravilno je $P(c_0) = 0$. Od števila učnih primerov je odvisno, ali nam bo upoštevanje take sodejavnosti prineslo oprijemljivo korist.

Pri sodejavnih atributih ima informacijski prispevek pozitivno vrednost. Večja kot je, bolj izrazita je sodejavnost. Da bi rezultate take analize prikazali uporabniku, lahko uporabimo graf, kjer vozlišča označujejo atribute, povezave pa sodejavnosti. Da bi razlikovali močnejše sodejavnosti od šibkejših, povezave označimo z indeksom moči ter z barvo. Povrh tega pa prikažemo le najmočnejše sodejavnosti, saj so neodvisni pari atributov v končnih množicah primerov porazdeljeni okrog ničle in ne nujno na njej, zaradi česar jih ima polovica in več pozitiven interakcijski prispevek. Na sliki 10.3 je prikazana interakcijska analiza domene 'adult' z repozitorija UCI [HB99]. Kot kaže, nam podatek, s katere države je prišel posameznik, pove veliko v kombinaciji z drugimi atributi. V kombinaciji nam povedo ti povezani pari atributov nepričakovano več o zaslužku posameznika, kot bi nam kot bi nam posamično z glasovanjem.

Sodejavnosti so primer pri katerem se splača tvoriti nove atribute, ki nadomestijo osnovne, kar je bil že precej raziskovan problem [Kon91, Paz96]. Obstajata pa še dve uporabi, ki do sedaj nista bili veliko omenjani: pri ločeni diskretizaciji atributov, ki so sodejavni, lahko zapravimo podatke, saj ju podcenjujemo. Zato je za diskretizacijo sodejavnih atributov bolje uporabiti nekratkovidne metode diskretizacije, recimo tisto v [Bay00], ali pa kar hkratno delitev s klasifikacijskimi drevesi.

Druga povezava se nanaša na gradnjo klasifikacijskih dreves. Zanimivo pa je, da lahko vidimo sodejavnosti kot atribute, ki se morajo pojavljati v odločitvenem drevesu skupaj. Na primer, v domeni 'breast' doseže popolne rezultate drevo, ki je zgrajeno le iz tistih dveh atributov, ki sta v najmočnejši interakciji. V zadnjem času je učinkovita in popularna metoda klasifikacijskih gozdov, kjer več klasifikacijskih dreves med seboj glasuje. Trenutni pristopi gradijo ta drevesa naključno, na primer naključni gozdovi (angl. *random forests*) [Bre99]. Ključno pa je le, da zgradimo drevo iz atributov, ki so medsebojno povezani v interakciji.



Slika 10.4: Interakcijski dendrogram za domeno 'adult'.

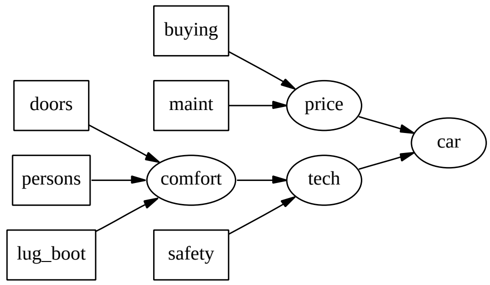
10.4.2 Soodvisnosti

Pri soodvisnostih velja, da nam atributa A in B podata deloma iste informacije, ki nam skupaj povejo manj o C, kot bi pričakovali, če bi kar sešteli obseg informacij, ki jih poda vsak posamezni atribut. Posledica je, da postanemo bolj gotovi, kot bi smeli biti.

Soodvisnosti razkriva negativni interakcijski prispevek. Soodvisnosti v domeni prikažemo z interakcijskim dendrogramom, ki je rezultat postopka hierarhičnega razvrščanja (angl. *hierarchical clustering*) [KR90, SHR97]. Pri tem uporabimo tole definicijo funkcije razdalje med atributoma A in B glede na razred:

$$\mathsf{D}(A,B) = \begin{cases} NA & \text{if } IG(ABC) > 0.001, \\ 1000 & \text{if } |IG(ABC)| < 0.001, \\ -1/IG(ABC) & \text{if } IG(ABC) < -0.001. \end{cases} \tag{10.3}$$

Na ta način bodo soodvisni atributi blizu, neodvisni pa daleč. Sodejavnosti na razdaljo ne bodo vplivale. Soodvisnosti je v domenah polno. Rezultat take analize za domeno 'adult' je na sliki 10.4. Na primer, število let izobrazbe nam ne pove kaj bistveno novega o zaslužku posameznika, če že vemo, kakšno izobrazbo ima. Taka analiza soodvisnosti nam pomaga zmanjšati število atributov.



Slika 10.5: Ročno izdelana struktura atributov za domeno, ki se ukvarja s problemom odločanja o nakupu avtomobila. Osnovni atributi so v pravokotnikih, izpeljani pa v elipsah.

10.5 Uporaba interakcij

10.5.1 Pomembnost interakcij

Med vsemi preizkušenimi postopki je edino interakcijski prispevek pravilno ugotovil tip interakcije. Seveda pa nam interakcijski prispevek pove le za tip in moč interakcije, ne pove pa nam kaj dosti o tem, ali je v pragmatičnem smislu za klasifikacijsko točnost to interakcijo smiselno upoštevati. Zato predlagamo, da se namesto posebnega testa pomembnosti ali signifikantnosti interakcije uporabi kar teste signifikantnosti izboljšanja klasifikatorja. Interakcija je pomembna, če dosega klasifkator, ki jo upošteva, pomembno boljše rezultate kot pa klasifikator, ki je ne upošteva. To velja za oboje, sodejavnosti in soodvisnosti.

Kar zanesljive rezultate dobimo s prečnim preverjanjem (angl. *cross-validation*) na učni množici, saj na ta način simuliramo situacijo, da imamo primere, ki jih še nismo videli. Klasifikator je namreč nesmiselno preverjati na primerih, na podlagi katerih je bil naučen, saj bi si jih lahko le zapomnil, ne da bi jih tudi razumel.

10.5.2 Interakcije in struktura atributov

Človek v svoji analizi domene atribute organizira v drevesno strukturo [BR90], na primer na sliki 10.5. Vprašamo se lahko, ali so sosedni atributi v interakciji z razrednim atributom ali niso. Izkaže se, da niso, saj ljudje organizirajo strukturo z namenom združevati atribute, ki so na nek način povezani med seboj, lahko zaradi soodvisnosti ali sodejavnosti, ali pa tudi ne. Povrh tega take strukture ne ločijo med obema vrstama interakcij. Človeku struktura služi, da lahko predpostavi neodvisnost med daljnimi sosedami. Človek tudi išče sodejavnosti in soodvisnosti le med bližnjimi sosedami.

Pomen avtomatskega odkrivanja interakcij naj zato uporabnikom ne omenja nujno interakcij, ki so že zajete v drevesni strukturi, saj so te pričakovane. Človeka bi zanimala le odstopanja, tako v smislu nepričakovanih interakcij, kot tudi nepričakovanih neodvisnosti. Po drugi strani pa se v naglici splača sprva preveriti sosede v drevesni strukturi, saj sosednost izraža človekovo predznanje o domeni.

10.5.3 Odpravljanje interakcij

Nekateri algoritmi strojnega učenja ne zmorejo obravnavati interakcij v podatkih. Najbolj znana je ta pomanjkljivost pri naivnem Bayesovevm klasifikatorju. Zato smo razvili postopka za odpravljanje interakcij, ki s prečnim preverjanjem na učni množici ugotovita, katere interakcije je smiselno odpraviti. Izkaže se, da je bolje kot s kartezičnim produktom odpravljati interakcije z uporabo krčenja prostora atributov [Dem02], sicer postopkom iz področja funkcijske dekompozicije [Zup97]. Za krčenje prostora atributov smo uporabili metodo minimizacije napake, ki uspešno obvladuje nedeterministične domene.

Postopka smo preizkusili z naivnim Bayesovim klasifikatorjem, pa tudi z logistično regresijo ter s klasifikatorji s podpornimi vektorji (angl. *support vector machines*). Pri vseh je prišlo v povprečju do izboljšanja, če je le bilo v domeni dovolj učnih primerov, da je prečno preverjanje na učni množici pravilno ocenilo kvaliteto variant klasifikatorja. Izboljšanje je bilo predvsem dobro pri odpravljanju soodvisnosti, tudi pri logistični regresiji in metodah podpornih vektorjev, četudi smo pričakovali, da bi obteževanje atributov, ki je v teh metodah že vgrajeno, opravilo boljše delo. Kot kaže, je za odpravljanje sodejavnosti potrebnih veliko učnih primerov, pomembne sodejavnosti pa so tudi dokaj redke.

Izjava

Izjavljam, da sem magistrsko nalogo izdelal samostojno pod mentorstvom prof. dr. Ivana Bratka. Ostale sodelavce, ki so mi pri nalogi pomagali, sem navedel v razdelku Acknowledgments na strani iv.

Sežana, *Aleks Jakulin*
17. februar 2003

APPENDIX A

Additional Materials

Yellow cat, black cat, as long as it catches mice, it is a good cat.

Deng Xiaoping

A.1 Clustering

There are three kinds of clustering algorithms: partitioning, hierarchical, and algorithms that assign a probability of membership of a given instance to a given cluster.

Partitioning algorithms take the number of clusters as a parameter to the algorithm and attempt to minimize an objective function. The function attempts to evaluate the quality of the clustering, for example, the distance of elements of a cluster to the cluster center.

Hierarchical algorithms are greedy and of two kinds: *agglomerative* algorithms join the closest pair of elements into a new cluster, and in subsequent operators consider joining the new cluster with another element, two other elements, or other clusters. The final result is a tree. *Divisive* algorithms operate similarly, but by finding the best division into two clusters. In succeeding iterations the new clusters are divided further, as long as only clusters of one element remain. Hierarchical algorithms do not presuppose the number of clusters, as the clustering for all possible number of clusters are present in the tree. This assures that the process is computationally quite efficient.

Density-based algorithms define clusters as dense regions separated by sparse regions. The density estimation process can be performed in a variety of ways. Some algorithms assume specific probability distributions, for example the Gaussian probability distribution.

Fuzzy clustering algorithms assign a cluster membership vector to each element. An element may belong to multiple clusters, each with a certain probability. The algorithms described earlier are *crisp*, where each element is a member of only a single cluster, with unitary probability. Most of the above algorithms, especially the density-based algorithms, can be adjusted to work with membership vectors.

We base our description of example algorithms in the following subsections on [KR90, SHR97].

A.1.1 Partitioning Algorithms

The *pam* algorithm is based on search for k representative objects or medoids among the observations of the data set. After finding a set of k medoids, k clusters are constructed by assigning each observation to the nearest medoid. The goal is to find k representative objects which minimize the sum of the dissimilarities of the observations to their closest representative object. The algorithm first looks for a good initial set of medoids in the build phase. Then it finds a local minimum for the objective function, that is, a solution such that there is no single switch of an observation with a medoid that will decrease the objective (this is called the swap phase).

A.1.2 Hierarchical Algorithms

The agglomerative nesting *agnes* algorithm constructs a hierarchy of clusterings. At first, each observation is a small cluster by itself. Clusters are merged until only one large cluster remains which contains all the observations. At each stage the two nearest clusters are combined to form one larger cluster.

Different linkage methods are applicable to hierarchical clustering. In particular, hierarchical clustering is based on $n-1$ fusion steps for n elements. In each fusion step, an object or cluster is merged with another, so that the quality of the merger is best, as determined by the linkage method.

Average linkage method attempts to minimize the average distance between all pairs of members of two clusters. If P and Q are clusters, the distance between them is defined as

$$d(P,Q) = \frac{1}{|P||Q|} \sum_{i \in R, j \in Q} d(i,j)$$

Single linkage method is based on minimizing the distance between the closest neighbors in the two clusters. In this case, the generated clustering tree can be derived from the minimum spanning tree:

$$d(P,Q) = \min_{i \in R, j \in Q} d(i,j)$$

Complete linkage method is based on minimizing the distance between the furthest neighbors:

$$d(P,Q) = \max_{i \in R, j \in Q} d(i,j)$$

Ward's minimum variance linkage method attempts to minimize the increase in the total sum of squared deviations from the mean of a cluster.

Weighted linkage method is a derivative of average linkage method, but both clusters are weighted equally in order to remove the influence of different cluster size.

A.1.3 Fuzzy Algorithms

In a fuzzy *fanny* clustering, each observation is 'spread out' over the various clusters. Denote by $u_{i,v}$ the membership of observation i to cluster v. The memberships are non-negative, and for a fixed observation i they sum to 1. Fanny is robust to the spherical cluster assumption.

Fanny aims to minimize the objective function:

$$\sum_{v}^{k} \frac{\sum_{i}^{n} \sum_{j}^{n} u_{i,v}^{2} u_{j,v}^{2} d_{i,j}}{2 \sum_{j}^{n} u_{j,v}^{2}}$$

where n is the number of observations, k is the number of clusters and $d_{i,j}$ is the dissimilarity between observations i and j. The number of clusters k must comply with $1 \leq k \leq \frac{n}{2} - 1$.

A.1.4 Evaluating the Quality of Clustering

Silhouettes are one of the heuristic measures of cluster quality. Averaged over all the clusters, the average silhouette width is a measure of quality of the whole clustering. Similarly, the agglomerative coefficient is a measure of how successful has been the clustering of a certain data set.

The silhouette width is computed as follows: Put a_i = average dissimilarity between i and all other points of the cluster to which i belongs. For all clusters C, put $d(i, C)$ = average dissimilarity of i to all points of C. The smallest of these $d(i, C)$ is denoted as b_i, and can be seen as the dissimilarity between i and its neighbor cluster. Finally, put $s_i = (b_i - a_i)/\max(a_i, b_i)$. The overall average silhouette width is then simply the average of s_i over all points i.

The agglomerative coefficient measures the clustering structure of the data set. For each data item i, denote by m_i its dissimilarity to the first cluster it is merged with, divided by the dissimilarity of the merger in the final step of the algorithm. The *ac* is the average of all $1 - m_i$. Because *ac* grows with the number of observations, this measure should not be used to compare data sets of much differing size.

A.2 Optimal Separating Hyperplanes

As there can be many separating hyperplanes which are consistent with all the training instances, one can question which of them is optimal. Vapnik's [Vap99] notion of an optimal separating hyperplane is based on attempting to place it so that it will be as far as possible from the nearest instance of either class.

In contrast, the 'traditional' approach to linear discriminant analysis is based on placing the separating hyperplane as far as possible from the *means* of both classes. Such a classifier is ideal or Bayes optimal if each class is normally distributed, while all classes share the covariance matrix, but any discussion of such conditional optimality gives a faulty sense of security, as we should assume too much about the nature of data.

As it is often impossible to find a consistent separating hyperplane, one can relax the assumptions. We will try to apply soft-margin separating hyperplanes as described in

[Vap99]. The soft-margin hyperplane (also called the generalized optimal hyperplane) is determined by the vector $\mathbf{w}$ which minimizes the functional

$$\Phi(\mathbf{w}, \boldsymbol{\xi}) = \frac{1}{2}(\mathbf{w} \cdot \mathbf{w}) + C\left(\sum_{i=1}^{\ell} \xi_i\right)$$

(here C is a *given* value) subject to constraint

$$y_i((\mathbf{w} \cdot \mathbf{x}_i) - \mathbf{b}) \geq 1 - \xi_i, \qquad i = 1, 2, \ldots, \ell$$

To find the coefficients of the generalized optimal (or maximal margin) hyperplane

$$\mathbf{w} = \sum_{i=1}^{\ell} \alpha_i y_i \mathbf{x}_i$$

one has to find the parameters $\alpha_i, i = 1, \ldots, \ell$, that maximize the quadratic form

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j)$$

with the constraint

$$0 \leq \alpha_i \leq C, \quad i = 1, \ldots, \ell$$

$$\sum_{i=1}^{\ell} \alpha_i y_i = 0$$

Only some of the coefficients $\alpha_i, i = 1, \ldots, \ell$, will differ from zero. They determine the support vectors.

However, unlike the support vector machines, we perform no nonlinear mapping of input features.

Quadratic programming tools expect the QP problem to be represented somewhat differently. Following [Joa98], we can define matrix Q as $Q_{ij} = y_i y_j \left(\mathbf{x}_i \cdot \mathbf{x}_j\right)$, and reformulate the above form as:

$$\begin{aligned} \text{minimize}: \quad & W(\boldsymbol{\alpha}) = -\boldsymbol{\alpha}^T \mathbf{1} + \tfrac{1}{2} \boldsymbol{\alpha}^T Q \boldsymbol{\alpha} \\ \text{subject to}: \quad & \boldsymbol{\alpha}^T \mathbf{y} = 0 \\ & \mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1} \end{aligned}$$

Choosing the value of C is very important, and this is rarely mentioned in SVM literature. For example, in an unbalanced linearly separable domain, the above QP optimization will not arrive to a correct separating hyperplane for a binary AND problem! Even worse, with some QP algorithms, the whole process may fall in an infinite loop. If the value of C is increased, the solution will be obtained. There are several more issues which may result in the above QP optimization not arrive at a correct solution. Therefore, although the methods seem conceptually simple, there are many traps, unlike with the foolproof naïve Bayesian classifier.

REFERENCES

[Agr90] A. Agresti. *Categorical data analysis.* Wiley, 1990.

[And02] C. J. Anderson. Applied categorical data analysis lecture notes. University of Illinois, Urbana-Champaign, 2002.

[Bax97] J. Baxter. The canonical distortion measure for vector quantization and approximation. In *Proc. 14th International Conference on Machine Learning*, pages 39–47. Morgan Kaufmann, 1997.

[Bay00] S. D. Bay. Multivariate discretization of continuous variables for set mining. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000.

[Ber38] D. Bernoulli. Specimen theoriae novae de mensura sortis. In *Commentarii Academiae Scientiarum Imperialis Petropolitanae*, volume 5, pages 175–192, 1738.

[Bla69] H. M. Blalock, Jr. *Theory Construction; from verbal to mathematical formulations.* Prentice-Hall, Inc., Englewoods Cliffs, New Jersey, USA, 1969.

[BR88] M. Bohanec and V. Rajkovič. Knowledge acquisition and explanation for multi-attribute decision making. In *8th Intl Workshop on Expert Systems and their Applications*, pages 59–78, Avignon, France, 1988.

[BR90] M. Bohanec and V. Rajkovič. DEX: An expert system shell for decision support. *Sistemica*, 1(1):145–157, 1990.

[Bre99] L. Breiman. Random forests – random features. Technical Report 567, University of California, Statistics Department, Berkeley, 1999.

[CBL97] J. Cheng, D. Bell, and W. Liu. Learning Bayesian networks from data: An efficient approach based on information theory. In *Proceeding of the 6th ACM International Conference on Information and Knowledge Management*, 1997.

[Ces90] B. Cestnik. Estimating probabilities: A crucial task in machine learning. In *Proc. 9th European Conference on Artificial Intelligence*, pages 147–149, 1990.

[CJS$^+$94] C. Cortes, L. D. Jackel, S. A. Solla, V. Vapnik, and J. S. Denker. Learning curves: Asymptotic values and rate of convergence. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 327–334. Morgan Kaufmann Publishers, Inc., 1994.

[CL01] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[CT91] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley Series in Telecommunications. John Wiley & Sons, 1991.

[Dem02] J. Demšar. *Constructive Induction by Attribute Space Reduction*. PhD thesis, University of Ljubljana, Faculty of Computer and Information Science, 2002.

[DH73] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, New York, USA, 1973.

[DHS00] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley, 2nd edition, October 2000.

[Die98] T. G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924, 1998.

[Die00] T. G. Dietterich. Ensemble methods in machine learning. In J. Kittler and F. Roli, editors, *First International Workshop on Multiple Classifier Systems*, Lecture Notes in Computer Science, pages 1–15, New York, 2000. Springer Verlag.

[DP97] P. Domingos and M. Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29:103–130, 1997.

[DS79] P. Dubey and L. Shapley. Mathematical properties of the banzhaf power index. *Mathematics of Operations Research*, 4(2):99–131, 1979.

[DZ02] J. Demšar and B. Zupan. Orange: a data mining framework. `http://magix.fri.uni-lj.si/orange`, 2002.

[ELFK00] G. Elidan, N. Lotner, N. Friedman, and D. Koller. Discovering hidden variables: A structure-based approach. In *Proceeding of the Neural Information Processing Systems conference*, 2000.

[Ell] M. Ellison. `http://www.csse.monash.edu.au/~lloyd/tildeMML/`.

[Eve77] B. Everitt. The analysis of contingency tables, 1977.

[FDH01] J. T. A. S. Ferreira, D. G. T. Denison, and D. J. Hand. Weighted naive Bayes modelling for data mining. Technical report, Dept. of Mathematics, Imperial College, London, UK, May 2001.

[FF99] C. C. Fabris and A. A. Freitas. Discovering surprising patterns by detecting occurrences of simpson's paradox. In *Research and Development in Intelligent Systems XVI (Proc. ES99, The 19th SGES Int. Conf. on Knowledge-Based Systems and Applied Artificial Intelligence)*, pages 148–160. Springer-Verlag, 1999.

[FG96] N. Friedman and M. Goldszmidt. Building classifiers using Bayesian networks. In *Proc. National Conference on Artificial Intelligence*, pages 1277–1284, Menlo Park, CA, 1996. AAAI Press.

[Fre01] A. A. Freitas. Understanding the crucial role of attribute interaction in data mining. *Artificial Intelligence Review*, 16(3):177–199, November 2001.

[FU] G. L. Fonseca and L. Ussher. The history of economic thought website. `http://cepa.newschool.edu/het/home.htm`.

[GD02] G. Gediga and I. Düntsch. On model evaluation, indices of importance, and interaction values in rough set analysis. In S. K. Pal, L. Polkowski, and A. Skowron, editors, *Rough-Neuro Computing: A way for computing with words*, Heidelberg, 2002. Physica Verlag.

[GMR99] M. Grabisch, J.-L. Marichal, and M. Roubens. Equivalent representations of a set function with applications to game theory and multicriteria decision making. In *Proc. of the Int. Conf. on Logic, Game theory and Social choice (LGS'99)*, pages 184–198, Oisterwijk, the Netherlands, May 1999.

[GMR00] M. Grabisch, J.-L. Marichal, and M. Roubens. Equivalent representations of set functions. *Mathematics of Operations Research*, 25(2):157–178, 2000.

[GR99] M. Grabisch and M. Roubens. An axiomatic approach to the concept of interaction among players in cooperative games. *International Journal of Game Theory*, 28(4):547–565, 1999.

[Grü98] P. Grünwald. *The Minimum Description Length Principle and Reasoning Under Uncertainty.* PhD dissertation, Universiteit van Amsterdam, Institute for Logic, Language, and Computation, 1998.

[Grü00] P. Grünwald. Maximum entropy and the glasses you are looking through. In *Proceedings of the Sixteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI 2000)*, Stanford, CA, USA, July 2000.

[HB99] S. Hettich and S. D. Bay. The UCI KDD archive `http://kdd.ics.uci.edu`. Irvine, CA: University of California, Department of Information and Computer Science, 1999.

[HMS66] E. B. Hunt, J. Martin, and P. Stone. *Experiments in Induction.* Academic Press, New York, 1966.

[Hol97] A. Holst. *The Use of a Bayesian Neural Network Model for Classification Tasks.* PhD thesis, Royal Institute of Technology, Sweden, September 1997.

[Hun62] E. B. Hunt. *Concept Learning: An Information Processing Problem.* Wiley, 1962.

[IG96] R. Ihaka and R. Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314, 1996.

[Jac01] J. Jaccard. *Interaction Effects in Logistic Regression*, volume 07–135 of *Sage University Papers. Quantitative Applications in the Social Sciences.* Sage, 2001.

[Jak02] A. Jakulin. Extensions to the Orange data mining framework. `http://ai.fri.uni-lj.si/aleks/orng`, 2002.

[Jay88] E. T. Jaynes. The relation of Bayesian and maximum entropy methods. In *Maximum-Entropy and Bayesian Methods in Science and Engineering*, volume 1, pages 25–29. Kluwer Academic Publishers, 1988.

[Joa98] T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning.* MIT Press, Cambridge, USA, 1998.

[Joh00] P. M. Johnson. A glossary of political economy terms. Dept. of Political Science, Auburn University, 1994–2000.

[JTW90] J. Jaccard, R. Turrisi, and C. K. Wan. *Interaction Effects in Multiple Regression*, volume 72 of *Sage University Papers. Quantitative Applications in the Social Sciences.* Sage, 1990.

[Kad95] C. M. Kadie. *Seer: Maximum Likelihood Regression for Learning-Speed Curves.* PhD thesis, University of Illinois at Urbana-Champaign, 1995.

[KB91] I. Kononenko and I. Bratko. Information based evaluation criterion for classifier's performance. *Machine Learning*, 6:67–80, 1991.

[KL51] S. Kullback and R. Leibler. On information and sufficiency. *Ann. Math. Stat.*, 22:76–86, 1951.

[KLMT00] P. Kontkanen, J. Lahtinen, P. Myllymäki, and H. Tirri. An unsupervised Bayesian distance measure. In E. Blanzieri and L. Portinale, editors, *EWCBR 2000 LNAI 1898*, pages 148–160. Springer-Verlag Berlin Heidelberg, 2000.

[KN] E. Koutsofios and S. C. North. *Drawing Graphs with dot.* Available on `research.att.com` in `dist/drawdag/dotguide.ps.Z`.

[Koh95] R. Kohavi. *Wrappers for Performance Enhancement and Oblivious Decision Graphs.* PhD dissertation, Stanford University, September 1995.

[Koh96] R. Kohavi. Scaling up the accuracy of naive-Bayes classifiers: a decision-tree hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 202–207, 1996.

[Kon90] I. Kononenko. *Bayesovske nevronske mreže.* PhD thesis, Univerza v Ljubljani, Slovenija, 1990.

[Kon91] I. Kononenko. Semi-naive Bayesian classifier. In Y. Kodratoff, editor, *European Working Session on Learning - EWSL91*, volume 482 of *LNAI*. Springer Verlag, 1991.

[Kon97] I. Kononenko. *Strojno učenje.* Fakulteta za računalništvo in informatiko, Ljubljana, Slovenija, 1997.

[KR90] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis.* Wiley, New York, USA, 1990.

[KR92] K. Kira and L. A. Rendell. A practical approach to feature selection. In D. Sleeman and P. Edwards, editors, *Machine Learning: Proceedings of the International Conference (ICML'92)*, pages 249–256. Morgan Kaufmann, 1992.

[Kra94] S. Kramer. CN2-MCI: A two-step method for constructive induction. In *Proc. ML-COLT'94 Workshop on Constructive Induction and Change of Representation*, New Brunswick, New Jersey, USA, 1994.

[KW01] L. I. Kuncheva and C. J. Whittaker. Measures of diversity in classifier ensembles and their relationship with ensemble accuracy. *Machine Learning*, forthcoming, 2001.

[LCB+02] G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan. Learning the kernel matrix with semi-definite programming. In C. Sammut and A. Hoffmann, editors, *Proceedings of the 19th International Conference on Machine Learning*, Sydney, Australia, 2002. Morgan Kaufmann.

[LMM02] D. A. Lind, W. G. Marchal, and R. D. Mason. *Statistical Techniques in Business and Economics.* McGraw-Hill/Irwin, 11/e edition, 2002.

[LZ02] C. X. Ling and H. Zhang. Toward Bayesian classifiers with accurate probabilities. In *Proceedings of the Sixth Pacific-Asia Conference on KDD.* Springer, 2002.

[Mac91] D. MacKay. *Bayesian Methods for Adaptive Models.* PhD thesis, California Institute of Technology, 1991.

[Mac01] D. MacKay. Decision theory – a simple example. `http://www.inference.phy.cam.ac.uk/mackay/Decision.html`, August 2001.

[Mar99] J.-L. Marichal. *Aggregation Operators for Multicriteria Decision Aid.* PhD thesis, University of Liège, Department of Management, 1999.

[Mil92] A. J. Miller. Algorithm AS 274: Least squares routines to supplement those of Gentleman. *Appl. Statist.*, 41(2):458–478, 1992.

[Min00] T. P. Minka. Distance measures as prior probabilities, 2000. `http://www.stat.cmu.edu/~minka/papers/metric.html`.

[MJ93] G. H. McClelland and C. M. Judd. Statistical difficulties of detecting interactions and moderator effects. *Psychological Bulletin*, 114:376–390, 1993.

[MM00] T. Matsui and Y. Matsui. A survey of algorithms for calculating power indices of weighted majority games. *Journal of Operations Research Society of Japan*, 43(1):71–86, 2000.

[MP69] M. L. Minsky and S. A. Papert. *Perceptrons.* MIT Press, Cambridge, MA, expanded 1990 edition, 1969.

[MR99] J.-L. Marichal and M. Roubens. The chaining interaction index among players in cooperative games. In N. Meskens and M. Roubens, editors, *Advances in Decision Analysis*, volume 4 of *Mathematical Modelling - Theory and Applications*, pages 69–85. Kluwer, Dordrecht, 1999.

[MS63] J. N. Morgan and J. A. Sonquist. Problems in the analysis of survey data, and a proposal. *Journal of the American Statistical Association*, 58:415–435, 1963.

[MST92] D. Michie, D. J. Spiegelhalter, and C. C. Taylor, editors. *Machine Learning, Neural and Statistical Classification.* Ellis Horwood, London, UK, 1992.

[NJ01] A. Y. Ng and M. I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In *NIPS'01*, 2001.

[OBR89] M. Olave, M. Bohanec, and V. Rajkovič. An application for admission in public school systems. In I. T. M. Snellen, W. B. H. J. van de Donk, and J.-P. Baquiast, editors, *Expert Systems in Public Administration*, pages 145–160. Elsevier Science Publishers (North Holland), 1989.

[Ock20] W. of Ockham. Quodlibeta septem. scriptum in librum primum sententiarum. In *Opera Theologica*, volume I, page 74. 1320.

[Owe72] G. Owen. Multilinear extensions of games. *Management Sciences*, 18:64–79, 1972.

[Paz96] M. J. Pazzani. Searching for dependencies in Bayesian classifiers. In *Learning from Data: AI and Statistics V.* Springer-Verlag, 1996.

[Pea88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann, San Francisco, CA, USA, 1988.

[Pér97] E. Pérez. *Learning Despite Complex Attribute Interaction: An Approach Based on Relational Operators.* PhD dissertation, University of Illinois at Urbana-Champaign, 1997.

[PF97] F. Provost and T. Fawcett. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In *Proc. of KDD'97.* AAAI, 1997.

[Pla99] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In A. J. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers.* MIT Press, 1999.

[PPS01] C. Perlich, F. Provost, and J. S. Simonoff. Tree induction vs logistic regression: A learning-curve analysis. CeDER Working Paper IS-01-02, Stern School of Business, New York University, NY, Fall 2001.

[PR96] E. Pérez and L. A. Rendell. Statistical variable interaction: Focusing multiobjective optimization in machine learning. In *Proceedings of the First International Workshop on Machine Learning, Forecasting and Optimization (MALFO'96)*, Leganès, Madrid, Spain, 1996. Universidad Carlos III de Madrid, Spain.

[Qui93] J. R. Quinlan. *C4.5: programs for machine learning.* Morgan Kaufmann Publishers Inc., San Francisco, CA, 1993.

[Qui94] J. R. Quinlan. *Comparing connectionist and symbolic learning methods*, volume I: Constraints and Prospects, pages 445–456. MIT Press, 1994.

[RH80] R. P. Runyon and A. Haber. *Fundamentals of Behavioral Statistics.* Addison Wesley Publishing Company, Inc., Philipines, 4th edition, 1980.

[RH97] Y. D. Rubinstein and T. Hastie. Discriminative vs informative learning. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 49–53. AAAI Press, August 1997.

[RHJ01] I. Rish, J. Hellerstein, and T.S. Jayram. An analysis of data characteristics that affect naive Bayes performance. Technical Report RC21993, IBM, 2001.

[RR96] V. Rao and H. Rao. *C++ Neural Networks and Fuzzy Logic.* BPB Publications, New Delhi, India, 1996.

[Sar94] W. S. Sarle. Neural networks and statistical models. In *Proceedings of the 19th Annual SASUG International Conference*, April 1994.

[SAS98] *SAS/STAT User's Guide.* SAS Institute Inc., Cary, NC, USA, 1998.

[Ses89] R. Seshu. Solving the parity problem. In *Proc. of the Fourth EWSL on Learning*, pages 263–271, Montpellier, France, 1989.

[Sha48] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[SHR97] A. Struyf, M. Hubert, and P. J. Rousseeuw. Integrating robust clustering techniques in S-PLUS. *Computational Statistics and Data Analysis*, 26:17–37, 1997.

[Šik02] M. Robnik Šikonja. Theoretical and empirical analysis of ReliefF and RReliefF. *Machine Learning Journal*, forthcoming, 2002.

[SW86] C. Stanfill and D. Waltz. Towards memory-based reasoning. *Communications of the ACM*, 29(12):1213–1228, 1986.

[Vap99] V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag New York, 2nd edition, 1999.

[Ved02] V. Vedral. The role of relative entropy in quantum information theory. *Reviews of Modern Physics*, 74, January 2002.

[Ver02] J. Verhagen, editor. Science jokes. `http://www.xs4all.nl/~jcdverha/scijokes/`, August 2002.

[Wei02] E. W. Weisstein. *Eric Weisstein's World of Mathematics.* `http://mathworld.wolfram.com/`, 2002.

[WM95] D. W. Wolpert and W. G. Macready. No free lunch theorems for search. Technical Report SFI-TR-05-010, Santa Fe Institute, 1995.

[Wol96] D. W. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8:1341–1390, 1996.

[WP98] G. I. Webb and M. J. Pazzani. Adjusted probability naive Bayesian induction. In *Proceedings of the Tenth Australian Joint Conference on Artificial Intelligence*, pages 285–295, Brisbane, Australia, July 1998. Springer Berlin.

[WW89] S. J. Wan and S. K. M. Wong. A measure for concept dissimilarity and its application in machine learning. In *Proceedings of the International Conference on Computing and Information*, pages 267–273, Toronto, Canada, 1989. North-Holland.

[Zad02] B. Zadrozny. Reducing multiclass to binary by coupling probability estimates. In *Advances in Neural Information Processing Systems 14 (NIPS*2001)*, June 2002.

[ZDS+01] B. Zupan, J. Demšar, D. Smrke, K. Božikov, V. Stankovski, I. Bratko, and J. R. Beck. Predicting patient's long term clinical status after hip arthroplasty using hierarchical decision modeling and data mining. *Methods of Information in Medicine*, 40:25–31, 2001.

[ZE01] B. Zadrozny and C. Elkan. Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML'01)*, pages 609–616, Williams College, Massachussetts, June 2001. Morgan Kaufmann.

[ZLZ00] H. Zhang, C. X. Ling, and Z. Zhao. The learnability of naive Bayes. In *Proceedings of Canadian Artificial Intelligence Conference*, pages 432–441. Springer, 2000.

[Zup97] B. Zupan. *Machine Learning Based on Function Decomposition*. PhD thesis, University of Ljubljana, Faculty of Computer and Information Science, 1997.

INDEX