THE KLUWER INTERNATIONAL SERIES
IN ENGINEERING AND COMPUTER SCIENCE

FEATURE EXTRACTION, CONSTRUCTION AND SELECTION:

A Data Mining Perspective

edited by

Huan Liu
National University of Singapore
SINGAPORE

and

Hiroshi Motoda
Osaka University
Osaka, JAPAN

KLUWER ACADEMIC PUBLISHERS
Boston / Dordrecht / London

23 FEATURE TRANSFORMATION STRATEGIES FOR A ROBOT LEARNING PROBLEM

Luis Seabra Lopes[1] and Luis M. Camarinha-Matos[2]

[1]Departamento de Electrónica e Telecomunicações
Universidade de Aveiro
3810 Aveiro, Portugal
lsl@ua.pt

[2]Departamento de Engenharia Electrotécnica
Universidade Nova de Lisboa
2825 Monte da Caparica, Portugal
cam@uninova.pt

Abstract: This chapter illustrates, with a case study from the robotized assembly domain, the importance of feature transformation. The specific problem that is addressed is learning failure diagnosis models for a pick-and-place operation. Several feature transformation strategies are evaluated on flat as well as hierarchical learning problems. The SKIL learning algorithm, previously proposed by the authors, is used in most experiments. A comparison with an oblique tree learning algorithm is also included.

23.1 INTRODUCTION

The research described in this paper was motivated by the need to develop execution supervision functionalities for flexible assembly systems (Camarinha-Matos et al., 1996; Seabra Lopes and Camarinha-Matos, 1996).

Execution supervision is the activity of executing a plan while making sure that all its goals are achieved. Besides the lower (sub-symbolic/reactive) control levels, an architecture for intelligent supervision should possess high-level decision-making capabilities. This includes monitoring goal achievement during task execution, diagnosis when unforeseen situations are detected and planning both for nominal task execution and for failure recovery. One main problem in developing an intelligent supervisor is the acquisition of knowledge about the task and the environment to support monitoring, diagnosis and recovery.

This is especially true in the product assembly area, where task execution must achieve complex goals (like the relationships between components in the completed assembly) defined at a symbolic level. In order to alleviate this sort of knowledge acquisition bottleneck, the use of machine learning techniques was investigated. The general problem we have been investigating can be described as *robot learning at the task level* (Seabra Lopes, 1997). An important part of the work, concerned with the induction of structured knowledge for diagnosis, evolved in the framework of the B-LEARN II project (Kaiser et al., 1995).

During the research, the authors came across the problem of having to transform the space of features that characterize each training example, in order to improve the quality of the learned models. The applied feature transformation strategies and the improvements they produced are described in this chapter. In the initial sections, the programming by demonstration framework, the applied learning algorithm and the experimental situation will be described.

23.2 ROBOT PROGRAMMING BY DEMONSTRATION

In real execution, when the monitoring function detects a deviation in the behavior of the robot that may correspond to a failure, the diagnosis function is called to confirm and characterize the failure. Diagnosis is a decision process that requires a sophisticated model of the task, the system and the environment. Based on the failure description, recovery planning can be attempted.

The problem of building such sophisticated model is not easily solved. Even the best domain expert will have difficulty in specifying the necessary mappings between the available sensors on one side and the monitoring conditions, failure classifications, failure explanations and recovery strategies on the other. Also, a few less common errors will be forgotten.

The paradigm of Robot Programming by Demonstration (RPD), adopted in the B-LEARN II project, seems indicated to overcome this type of difficulties. According to this paradigm, complex systems are programmed by showing examples of their desired behavior. In our approach, interaction with the human, seen as a tutor is fundamental. Usually, emphasis is put on robots learning from their own perception of how humans perform certain tasks (Kuniyoshi et al., 1994). In our approach, RPD is broader and includes any interaction with the human that leads to improvement in future robot performance. An adequate user interface facilitates transfer of the human's knowledge to the robot and learning capabilities enable it to generate new knowledge.

The human will carry out an initial training phase for the nominal plan execution. The traces of all testable sensors will be collected in order to generate primitive skills and the corresponding monitoring knowledge. Also in the initial training phase, the human operator may decide to provoke typical failures and collect sensor data about them. Failure classification knowledge is subsequently generated by induction. When a new failure is detected during real execution of the assembly system, the human operator is called to classify and explain that failure and to provide a recovery strategy for the situation.

The classification phase of the diagnosis task can be performed based on knowledge generated automatically by inductive learning. A new algorithm, SKIL, was developed to perform this task. SKIL generates hierarchies of structured concepts. As far as diagnostic knowledge is concerned, the most important evaluation criterion is accuracy. Information on hierarchical problem decomposition, given to SKIL, leads to significant improvements in prediction accuracy. Another way to improve accuracy, emphasized in this chapter, is feature transformation.

The crucial step in diagnosis seems to be training the system to understand the meaning of sensor values and learn a qualitative and hierarchical model of the behavior of each operation. Programming such model would be nearly impossible. Since the human defines the "words" (discrimination features, classification attributes and respective values) used in the model, the human is capable of understanding the more or less detailed description that the model provides for each situation. It is then easier to hand-code explanations for the situations described in the model.

23.3 LEARNING STRUCTURED CONCEPTS

Having as motivation the automatic construction of the models required for the assembly supervisor, the idea of generating a concept hierarchy became attractive. The problem of learning at multiple levels of abstraction has not yet been adequately considered in the literature. In some approaches, a fixed decomposition of concepts is used, and learning is applied at each level (Koller and Sahami, 1997). However this is not flexible enough. Fixed decompositions have also been used for feature values. A new algorithm, SKIL (*structured knowledge generated by inductive learning* (Seabra Lopes and Camarinha-Matos, 1995)), was developed to perform this task.

The concepts in the hierarchy learned by SKIL are characterized by a set of symbolic classification attributes. At the lower levels of the hierarchy, concepts are described in more detail, i.e., more attribute values are specified. Moreover, in detailing or refining a concept, in which attributes take certain values, it may make sense to calculate other attributes. Therefore, the user may provide a set of attribute enabling statements of the form (A_i, a_{ij}, A_k), meaning that when the value of A_i is determined to be a_{ij}, then attribute A_k should be included in the set of attributes to consider in the continuation of the induction process. For example, when learning the behavior of a robotized *Transfer* operation, if a collision is found, it may make sense to determine some characteristics of the colliding object, such as its size. This could be expressed by the following attribute enabling statement: (*failure_type*, *collision*, *obj_size*).

The attribute values of the concepts in the hierarchy are determined inductively based on training data specified in terms of a set of discrimination features, which can be numerical or symbolic. Each example in the training

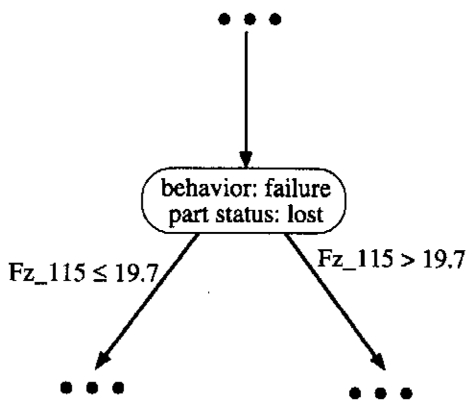

Figure 23.1 Example of a node in a concept hierarchy generated by SKIL

set is composed of the concept instance, represented as a list of attribute-value pairs, followed by a vector of feature values.

The algorithm is a recursive procedure that takes as parameters a list of examples, a list of classification attributes, a list of attribute enabling triples and a list of discrimination features. In each stage of the induction, the main goal is to predict the values of as many classification attributes as possible. For each attribute, the discrimination power of features is evaluated, using information theoretic measures. For continuous features, segmentation of feature values is done at each decision node, in a way that maximizes its discrimination power. The feature that, for some attribute, gives the highest discrimination power is selected to be test feature. Expansion stops when the values of all attributes have been predicted or when it is not possible to extract more information from the data.

The basic knowledge transmutation used by SKIL is, therefore, empirical inductive generalization, only that at multiple levels of abstraction. The generated knowledge structure is a hierarchy, where nodes represent structured concepts. A concept consists of a set of predictions. The number of predictions defines the abstraction level (concepts with less predictions are more abstract). The hierarchy is, simultaneously, a decision tree that can be used to recognize instances of the concepts. The formation of these concepts, guided by the attribute enabling statements, depends highly on the training data.

In Figure 23.1 an example of a node with two predictions and two child nodes is presented. The predictions identify a failure situation and assert that

the handled part was lost. In order to know which of the child nodes is the correct specialization of the displayed node, the feature Fz_115 must be tested.

As a special case, SKIL can also work as any traditional classification algorithm. This happens when only one classification attribute is defined. In such situations, SKIL performs equally well as (or even better than) some of the most popular classification algorithms, including CART (Breiman et al., 1984), ID3 (Quinlan, 1986), C4.5 (Quinlan, 1993) and OC1 (Murthy et al., 1993).

23.4 CONSTRUCTIVE INDUCTION

In robotics applications, information about the status of the system must often be obtained from complex sensors that provide numerical data difficult to analyze. From the supervision point of view, the results obtained in the initial experiments carried out within B-LEARN II were not satisfactory, due to the low classification accuracies on unseen examples. The most typical accuracy results were between 50% and 70%. One obvious explanation for these poor results was that the relevant features in each situation were not explicit in the sensor data, and, if the relevant features were somehow implicit in the data, the applied learning algorithms were not powerful enough to extract them.

One solution for this problem would be to use constructive induction. The problem of constructive induction, which aims to generate hypotheses described in terms of features that do not appear in the training data, has been raised by several authors, starting with Michalski (Michalski, 1983). In this case, a method should be capable of inventing new features, by applying known functions to the initial features, and evaluating the relevance of these new features to the learning task.

Only recently, constructive induction started to grow steadily and it is certainly a good promise for the future. It is unclear if the existing techniques are already mature for complex real-world problems. Experience within B-LEARN II indicates that they are not. In the project activities, the SMART+ system (Botta and Giordana, 1993), was applied to problems from the assembly application domain. SMART+ is a sophisticated learner, capable of combining different types of inference, such as inductive, deductive and abductive inference, and able to handle numerical features and to explore background knowledge. The concept description language used in SMART+ is an extended first order Horn clause language. The predicates that make up the body of a clause can be operational (those defined by the teacher) or non-operational (invented by the learner). The possibility of the teacher defining operational predicates enables the algorithm to perform a shift of bias, i.e. enables the generation of hypotheses expressed in terms of predicates not used in the training data. The application of these predicates to the examples in the training set establishes new features potentially more discriminative with respect to the learning problem. With this idea in mind, SMART+ was applied to diagnostic problems in the assembly domain. Several serious disadvantages became apparent. To start with, the system was too complex to parameterize. Some of the learned rules were equally complex. The system was not able to handle problems with many features. Finally, the system took a long time, typically many hours, to produce the rule set.

In numerical feature spaces, *oblique decision trees* are an alternative method of automatically deriving new features. The CART system (Breiman et al., 1984) was extended in this direction. During tree growth and in each step of the recursive partitioning procedure, instead of evaluating only the features provided in the training set, linear combinations of these features are evaluated.

The linear feature combination can be regarded as a new feature and therefore this approach can be thought of performing constructive induction. While tests on single features are equivalent to partitioning the input space by an axis-parallel hyperplane, linear feature combinations define oblique hyperplanes.

The vector of coefficients is initialized with random values and then a hill-climbing style search procedure will determine the vector of coefficients that maximizes class discrimination. The major drawback of this approach is the combinatorial explosion of possibilities that it must face. For a training set with N examples described in terms of p features, there are $\binom{N}{p}$ different splitting regions. This means that the number of alternative hyperplanes grows exponentially with the size of the training set. CART has the additional drawback that it does not necessarily find the best vector of coefficients because the search procedure can get stuck in local maxima. The more recent OC1 algorithm (Murthy et al., 1993), that will be used for comparison later in this chapter, uses randomization to escape local maxima. The specific strategy of OC1 makes it polynomial.

23.5 EXPERIMENTAL SITUATION: THE PICK AND PLACE TASK

A part pick-and-place macro-operation has been chosen as case study. For the experiments, three of the primitives involved in the operation were selected:

1. approach to grasp position (*Approach-Grasp*),
2. *Transfer* (of a part), and
3. approach to the final position (*Approach-Ungrasp*).

During the training phase, each of the selected operations was executed many times and several external exceptions were simulated. In most cases an object was placed, either in motion or stopped, in the robot arm motion path. A trace of forces and torques, covering the course of events from failure detection to stabilization of the system, was collected and a detailed failure description was assigned to the trace by the operator.

The force/torque sensor provides the features that will enable the learner to discriminate among alternative failure descriptions. Each time the sensor is consulted, the values of six variables, namely the forces and torques in the X, Y and Z directions, are obtained. This information constitutes a *sensor sample*. During the training phase, when a failure is detected, sensor samples are collected at regular time increments until the state of the system stabilizes.

During the observation window (a time interval covering the typically encountered failures) 15 sensor samples are collected. The values of a certain sensor variable (a force or a torque) in the successive samples in a failure trace make up a *profile* of that sensor variable, which is an element of $\Re^{15}$. Each failure trace is therefore composed of three force profiles and three torque profiles, providing a total of $(3+3) \times 15 = 90$ features to the learning process.

The failure description appended by the teacher to each example is the dependent variable in the learning process. When a similar failure is encountered, the robot should consider a similar failure description for further decision-making. However, these descriptions are not simple labels, as is customary in most learning systems, but structured descriptions, containing considerable detail. In the style of the previously described SKIL algorithm, each failure description consists of a set of attribute-value pairs. These attributes correspond to the classification attributes of SKIL and should not be confused with discrimination features.

In this way, 88 examples were collected for the *Approach-Grasp* operation, 47 examples for the *Transfer* operation and 117 examples for the *Approach-Ungrasp* operation. Based on these examples, it is possible to formulate a variety of learning problems.

23.6 FEATURE TRANSFORMATION STRATEGIES

The alternative followed in the experiments described in this chapter is to apply, in a pre-processing phase, a set of pre-defined feature transformation strategies. This is a highly domain-dependent approach, that remains an art and has not received enough attention from the machine learning researchers, traditionally more concerned with feature selection only. These strategies involve applying complex functions, including the discrete Fourier transform, to the raw features in order to obtain new features, hopefully more discriminative with respect to the learning problems. Most of these complex functions would hardly ever be discovered by existing constructive induction algorithms. The basic elements manipulated by feature construction functions are the profiles of the sensor variables, i.e. elements $\overline{p} = (p_1, ..., p_{15}) \in \Re^{15}$. Each feature transformation strategy receives the profiles of forces and torques and produces a feature vector, which results of joining together (or appending) the feature vectors returned by the applied feature construction functions.

In what follows, appending two feature vectors, $\overline{f} \in \Re^m$, and $\overline{g} \in \Re^n$, denoted by $(\overline{f}, \overline{g})$, is an element $\overline{h} \in \Re^{m+n}$ such that $h_i = f_i$ for $i = 1, ..., m$ and $h_i = g_{i-m}$ for $i = m+1, ..., m+n$.

In general, in a numerical domain, a feature transformation strategy is a function $f : \Re^d \to \Re^k$ which transforms d-dimensional feature vectors into k-dimensional feature vectors.

It is desirable that the execution supervisor reasons about the evolution of force and torque values, not in terms of the individual numerical values, but in terms of its overall characteristics, in short, as humans do. A human, making a qualitative description of such behavior, would probably divide it into intervals, and would mention roughly how long these intervals were, which were the average values in each interval, as well as the average derivatives. Dealing with time intervals is not an easy task, mainly when the goal is to apply feature-based learning algorithms to generate new knowledge. In the field of qualitative physics, the representations for numbers, frequently proposed, include signs, inequalities and orders of magnitude. Qualitative formalisms to describe changes in a system have also been proposed. Nevertheless, the feature transformation strategies that will be used do not produce qualitative features. The constructed features are all numeric, but they were designed to summarize potentially important aspects of the physical phenomenon under consideration. The numerical to symbolic conversion is a job for the learning algorithm itself.

One of the feature construction functions calculates the average of a certain sensor variable between two instants, m and n ($1 \leq m \leq n \leq 15$), in its profile:

$$A(\overline{p}, m, n) = \frac{\sum_{i=m}^{n} p_i}{n - m + 1} \tag{23.1}$$

Another function calculates the average derivative or rate of change between two instants in a given profile:

$$D(\overline{p}, m, n) = \frac{p_n - p_m}{n - m} \tag{23.2}$$

Another function tries to assess the degree of monotonicity of a profile. In this case, the general trend of the profile must be identified. The function $trend(\overline{p})$ determines if the general trend displayed by the profile is *ascending* (1) or *descending* (-1):

$$trend(\overline{p}) = \begin{cases} 1 & \text{if } A(\overline{p}, 1, 5) < A(\overline{p}, 11, 15), \\ -1 & \text{otherwise.} \end{cases} \tag{23.3}$$

The monotonicity of a profile $\overline{p}$ is then defined as the number of increases in consecutive instants of the profile, if the general trend is ascending, or the number of decreases in consecutive instants, if the general trend is descending. Monotonicity of $\overline{p}$ is computed by the following function:

$$M(\overline{p}) = \sum_{i=1}^{14} mon(\overline{p}, i) \tag{23.4}$$

where the function $mon(\overline{p}, i)$, that checks if the general trend of $\overline{p}$ is followed between instants i and $i+1$, is given by

$$mon(\overline{p}, i) = \begin{cases} 1 & \text{if } p_i \cdot trend(\overline{p}) < p_{i+1} \cdot trend(\overline{p}), \\ 0 & \text{otherwise.} \end{cases} \tag{23.5}$$

These three measures are summary measures. They are an approximation to the kind of features humans would extract from the profiles.

The measurable sensor variables are axis-parallel forces and torques. However, the resulting forces in the orthogonal planes and in 3D space might emphasize some important aspects of the physical process. For instance, the amplitude of the resulting torque in the XY plane might be more relevant to learning about vertical arm motion than the individual torques in the X and Y directions. In some of the feature transformation strategies that were experimented, two profile combination functions were applied. In both cases, the result is a profile of amplitudes of the resulting forces or torques.

The first of these functions, R_{2D}, takes as inputs two profiles, $\overline{a}, \overline{b} \in \Re^{15}$, and returns the amplitude profile $\overline{r} \in \Re^{15}$:

$$\overline{r} = R_{2D}(\overline{a}, \overline{b}) \Leftrightarrow r_i = \sqrt{a_i^2 + b_i^2}, \quad \text{for } i = 1, .., 15 \tag{23.6}$$

The other function, R_{3D}, computes a similar profile in 3D. It takes three profiles $\overline{a}, \overline{b}, \overline{c} \in \Re^{15}$ as input and returns the amplitude profile $\overline{r} \in \Re^{15}$:

$$\overline{r} = R_{3D}(\overline{a}, \overline{b}, \overline{c}) \Leftrightarrow r_i = \sqrt{a_i^2 + b_i^2 + c_i^2}, \quad \text{for } i = 1, .., 15 \tag{23.7}$$

In the selected case-study, most of the failure situations to be learned are related to different types of collisions of the robot arm with its environment. When a collision occurs, the arm enters a state of abnormal vibration that virtually disappears after a short time interval. In addition to the study in the time domain, it might be interesting to study the force and torque profiles in the frequency domain. A function of time for which an analytical expression is known can be transposed to the frequency domain by applying the *Fourier transform*. Waveforms are mathematical entities that describe important real-world phenomena (in optics, electricity, acoustics, etc.) as functions of time. In the meantime, spectra, that are the Fourier transformations of waveforms, also find physical correspondence. In several branches of science, it often happens that analysis of a given phenomenon is much simpler in the frequency domain.

There are several mathematical conditions that a function must meet in order for the Fourier transform to be applicable, but physical possibility is a valid sufficient condition for the existence of a transform.

In this case-study, the analytical expressions of the time functions represented by the profiles are not known. However, the measurements in each profile have been obtained at regular time intervals which enables the application of the *discrete Fourier transform.* For a given profile, what is obtained is a certain number of points of the Fourier transform of the underlying time function.

Let's consider the general case of a profile of a time function with an even number N of measurements taken at regular time intervals ($\overline{p} \in \Re^N$). By definition, the discrete Fourier transform of $\overline{p}$ is (Bracewell, 1986):

$$dft(\overline{p}, \nu) = \frac{1}{N} \sum_{\tau=1}^{N} e^{-j2\pi(\nu/N)\tau} \cdot p_\tau \tag{23.8}$$

In this expression, ν/N is an analog of frequency. The function $dft(\overline{p},\nu)$ is a periodic function of ν with period N. Furthermore, half of the produced values are complex conjugates of the other half. This means that, for a profile with N values, the discrete Fourier transform produces $N/2$ different amplitudes.

The feature transformation strategies, that have been used in the experiments described below, make use of the function $F(\overline{p})$, which returns the vector of amplitudes produced by the discrete Fourier transform for the profile $\overline{p} \in \Re^{15}$. Since the length of the measured profiles is 15, the period of the transform will be N=16 and the last value of the measured profile is repeated in position 16. $F(\overline{p})$ therefore returns an element $\overline{s} \in \Re^8$:

$$F(\overline{p}) = \overline{s} \Leftrightarrow s_\nu = \|dft((\overline{p}, p_{15}), \nu)\|, \quad \text{for } \nu = 1, .., 8 \tag{23.9}$$

A computer implementation of the fast Fourier transform (FFT), a method of calculating the discrete Fourier transform, has been used to obtain the amplitudes.

It is now possible to describe rigorously the feature transformation strategies that were applied. The starting point for feature transformation are the measured profiles, namely the force profiles $\overline{f}_x$, $\overline{f}_y$ and $\overline{f}_z$ and the torque profiles $\overline{t}_x$, $\overline{t}_y$ and $\overline{t}_z$. The applied feature transformation strategies are the following (for simplicity, each strategy is identified by a code):

Strategy S1: Raw Features. The measured force and torque values constitute the features in this strategy. No feature transformation is performed.

$$S1 \equiv (\overline{f}_x, \overline{f}_y, \overline{f}_z, \overline{t}_x, \overline{t}_y, \overline{t}_z) \tag{23.10}$$

In total, 6*15=90 features are considered.

Strategy S2: Resulting Forces and Torques. In this case, the profiles of the amplitudes of forces and torques in the orthogonal planes and in 3D space are considered:

$$\begin{array}{ll} \overline{f}_{xy} = R_{2D}(\overline{f}_x, \overline{f}_y) & \overline{t}_{xy} = R_{2D}(\overline{t}_x, \overline{t}_y) \\ \overline{f}_{xz} = R_{2D}(\overline{f}_x, \overline{f}_z) & \overline{t}_{xz} = R_{2D}(\overline{t}_x, \overline{t}_z) \\ \overline{f}_{yz} = R_{2D}(\overline{f}_y, \overline{f}_z) & \overline{t}_{yz} = R_{2D}(\overline{t}_y, \overline{t}_z) \\ \overline{f}_{xyz} = R_{3D}(\overline{f}_x, \overline{f}_y, \overline{f}_z) & \overline{t}_{xyz} = R_{3D}(\overline{t}_x, \overline{t}_y, \overline{t}_z) \end{array} \tag{23.11}$$

These profiles together with the measured profiles define this strategy:

$$S2 \equiv (\overline{f}_x, \overline{f}_y, \overline{f}_z, \overline{f}_{xy}, \overline{f}_{xz}, \overline{f}_{yz}, \overline{f}_{xyz}, \overline{t}_x, \overline{t}_y, \overline{t}_z, \overline{t}_{xy}, \overline{t}_{xz}, \overline{t}_{yz}, \overline{t}_{xyz}) \tag{23.12}$$

The number of features is in this case $14 \times 15 = 210$.

Strategy S3: Summary Features. This strategy consists of extracting the summary features of each profile. First of all, a long list of averages will be calculated for each profile $\overline{p}$ by

$$\begin{aligned} averages(\overline{p}) = {} & (A(\overline{p},1,3), A(\overline{p},4,6), A(\overline{p},7,9), A(\overline{p},10,12), A(\overline{p},13,15), \\ & A(\overline{p},1,5), A(\overline{p},6,10), A(\overline{p},11,15), A(\overline{p},1,15)) \end{aligned} \tag{23.13}$$

where $A(\overline{p},m,n)$ was defined in Equation 23.1. These sets of averages covers the profile $\overline{p}$ at different granularities (3 instants, 5 instants and 15 instants). The goal is to capture global aspects of the profile as well as more localized incidents. The summary features that will be extracted from each profile are given by the following function:

$$S(\overline{p}) = (averages(\overline{p}), D(\overline{p},1,8), D(\overline{p},8,15), M(\overline{p})) \tag{23.14}$$

where $D(\overline{p},m,n)$ and $M(\overline{p})$ are the derivative and monotonicity functions defined in Equations 23.2 and 23.4. Finally, the summary features extracted from each example are:

$$S3 \equiv (S(\overline{f}_x), S(\overline{f}_y), S(\overline{f}_z), S(\overline{t}_x), S(\overline{t}_y), S(\overline{t}_z)) \tag{23.15}$$

The number of features is in this case $6 \times 12 = 72$.

Strategy S4: Fourier Features. Extract the amplitudes of the harmonics generated by the fast Fourier transform for each profile:

$$S4 \equiv (F(\overline{f}_x), F(\overline{f}_y), F(\overline{f}_z), F(\overline{t}_x), F(\overline{t}_y), F(\overline{t}_z)) \tag{23.16}$$

where the function $F(\overline{p})$ was defined in Equation 23.9. This strategy produces 6*8=48 features.

Strategy S5: All Features. Finally, all features used in the previous strategies are joined together, producing a large feature vector. The features extracted from each profile include all measurements in that profile as well as summary features and Fourier features.

The function

$$B(\overline{p}) = (\overline{p}, S(\overline{p}), F(\overline{p})) \tag{23.17}$$

returns such features for a profile $\overline{p} \in \Re^{15}$ (functions $S(\overline{p})$ and $F(\overline{p})$ were defined in Equations 23.14 and 23.9). Strategy S5 is given by

$$\begin{aligned} S5 \equiv {} & (B(\overline{f}_x), B(\overline{f}_y), B(\overline{f}_z), B(\overline{f}_{xy}), B(\overline{f}_{xz}), B(\overline{f}_{yz}), B(\overline{f}_{xyz}), \\ & B(\overline{t}_x), B(\overline{t}_y), B(\overline{t}_z), B(\overline{t}_{xy}), B(\overline{t}_{xz}), B(\overline{t}_{yz}), B(\overline{t}_{xyz})) \end{aligned} \tag{23.18}$$

This strategy produces $14 \times (15+12+8) = 490$ features. The goal of proposing and evaluating this strategy, defined simply as the reunion/combination of the previous strategies, is to confirm (or not) the general rule that the more features are provided to the learner the better will be the result.

23.7 EXPERIMENTAL RESULTS

The experimental results that will now be presented illustrate the importance of feature transformation in improving the quality of the learned concept hierarchies. The first results were obtained on five classification problems that were designed, based on the training data collected in the experimental setup and previously described. In each of them, the goal is to learn to recognize some relevant aspect of an execution failure, maximizing accuracy and compactness of the learned knowledge. That relevant aspect that is to be learned in each case can be captured by a single classification attribute. These learning problems could, therefore, be addressed by classical concept learning algorithms, such as ID3 or CART. The presented results were obtained with SKIL and OC1. For simplicity of presentation, the learning problems are identified by a code:

- **LP-1: failures in *approach-grasp*.** Four classes of system behavior, that will be learned, are considered: *normal*, *collision*, *front collision* and *obstruction*; 88 training examples are available.
- **LP-2: failures in *transfer* of a part.** Five classes of system behavior are considered: *normal*, *front collision*, *back collision*, *collision to the right* and *collision to the left*; 47 examples are available.
- **LP-3: failures in *transfer* of a part.** Four situations of the handled part are considered: *ok*, *slightly moved*, *moved* and *lost*; the same 47 cases were used.
- **LP-4: failures in *approach-ungrasp*.** The classes of failures to be learned are: *normal*, *collision* and *obstruction*; 117 examples are available.
- **LP-5: failures in motion with part.** Five classes of system behavior are considered: *normal*, *bottom collision*, *bottom obstruction*, *collision in part* and *collision in tool*; 164 examples are available.

The results, in terms of classification accuracy (measured experimentally following a leave-one-out scheme) and size of the learned classification trees, are presented in Tables 23.1 and 23.2. In general, the more discriminative the available features are, the more accurate and compact will be the learned hypothesis. As expected, the results obtained with strategy S1, in which the sensor measurements (raw features) are directly used, are clearly the worst results. It is also clear that problems LP-2 and LP-5 are very difficult, even after feature transformation.

In terms of accuracy, strategy S3 (summary features) delivered the best results in problems LP-1, LP-3 and LP-4. In all the three cases, the improvements were substantial. In LP-1, summary features enabled an accuracy of 96%, which is 18% above the accuracy obtained with the raw features. In problem LP-4, the summary features produced an accuracy of 95%, representing an improvement of 30%. A major improvement (38%) was enabled by summary features in problem LP-3. In this case, the accuracy raised from the 49% obtained with the first strategy to 87% obtained with summary features. The best result in problem LP-2 was obtained with strategy S4 (Fourier features). In this case, accuracy was improved from 45% with the raw features to 68% with the Fourier features. Finally, problem LP-5, which turned out to be the least sensitive to feature transformation, got the best result with strategy S5 (all features).

Table 23.1 Classification accuracy for different feature transformation strategies

Learning problem	*S1*	*S2*	*S3*	*S4*	*S5*
LP-1	78	80	96	85	89
LP-2	45	57	51	68	64
LP-3	49	75	87	85	83
LP-4	65	60	95	77	83
LP-5	69	63	72	49	77
Average	61	67	80	79	79

In average terms, considering all the learning problems, strategy S3 was also the most successful, leading to an average accuracy improvement of 19%, immediately followed by strategy S5 with an average improvement of 18%. Strategy S4 was not so successful, producing an average accuracy improvement of 11.6%. Strategy S2 (measured forces and torques plus the resulting forces and torques in 2D and 3D) produced the smallest average improvement, only 5.8%. The fact that strategy S5, which includes all features used in strategy S3 and many more (490 features in total), has performed a little worse than strategy S3 can only be a contingency. It is not the general rule. In fact, our experience shows that the more features are provided to the algorithm the greater is the probability of obtaining a good result.

Table 23.2 Tree size for different feature transformation strategies

Learning problem	*S1*	*S2*	*S3*	*S4*	*S5*
LP-1	9	9	6	9	8
LP-2	18	12	12	11	12
LP-3	14	10	6	10	8
LP-4	11	10	9	8	7
LP-5	28	26	14	20	11
Average	16	13	9	12	9

As far as the compactness of the learned tree is concerned (Table 23.2), strategy S5 was, actually, the most successful on the average of all learning problems, immediately followed by strategy S3. The two major improvements in compactness were precisely obtained with these strategies. Strategy S3 enabled the generation of a tree with 6 nodes for problem LP-3, less than half the size of the tree obtained with the raw features. Strategy S5 enabled the generation of a tree with 11 nodes for problem LP-5, much smaller than the 28 nodes tree obtained with the raw features.

In Table 23.3, a comparison with OC1 (Murthy et al., 1993) is presented. This algorithm can be used to generate oblique trees, whose tests are based on linear combinations of features, as well as classical axis-parallel trees. The linear feature combinations can be seen as automatically learned features. It is, therefore, interesting to compare this automated approach to the approach followed in this chapter, based on pre-defined feature transformation strategies. From the results obtained, we see that the performances of OC1 (axis-parallel and oblique) and SKIL on the five learning problems previously presented are equivalent: the average accuracy is of 61% in the three cases. Therefore, the constructive nature of OC1-oblique didn't bring the expected improvements. This is even more disappointing considering the long running time required by OC1. In fact, to generate oblique trees and perform the leave-one-out test for the five learning problems, OC1 took more than 30 hours. To generate its hierarchies, SKIL needed around one hour only. In contrast with the lack of success of oblique trees, the feature transformation strategies that were applied enabled significant improvements in accuracy. As can be seen from Table 23.3 it was not difficult to get improvements of 20% or more.

Table 23.3 Comparison of various approaches

Approach	*LP-1*	*LP-2*	*LP-3*	*LP-4*	*LP-5*	*Average*
OC1 (axis-parallel)	65	43	70	70	58	61
OC1 (oblique)	65	49	57	80	53	61
SKIL (no feature transf.)	78	45	49	65	69	61
SKIL (2nd best results)	89	64	85	83	72	79
SKIL (best results)	96	68	87	95	77	85

Another experiment was conducted, in this case with a structured concept learning problem. This is the kind of problem that classical concept learning algorithms cannot handle. The training data used in this experiment consists of 88 examples of failures in the *approach_grasp* operation.

Ten classification attributes were considered:

- *behavior*: generic information about the operation behavior; it can be *normal* or *failure*; in the second case, the attributes *failure type, affected body, object size, object hardness* and *object weight* become enabled.
- *phase*: phase of the failed operation; it can be *initial, middle* or *terminal*.
- *failure type*: a classification or identification of the failure, for instance *collision, front collision* or *obstruction*.

- *affected body*: what was involved in or more affected by the failure: it can be the *tool*, the *tool tubes* or the *fingers*; if it is *tool* then the attribute *tool region* becomes enabled and if it is *fingers* then the attribute *fingers region* becomes enabled.
- *tool region*: region of the tool that was affected: it can be *front, left, right* and *back* sides, and *bottom*; in the last case the attribute *bottom subregion* becomes enabled.
- *bottom subregion*: if the failure affected the bottom part of the tool, where exactly was it; it may be *front, middle* or *back*.
- *fingers region*: region of the fingers that was affected, including *left finger, right finger* and *both fingers*.
- *object size*: size of object causing failure: *small, large*.
- *object hardness*: can be *soft* or *hard*.
- *object weight*: can be *low* or *high*.

Since not all attributes are enabled at the same time, the maximum size of a concept is of 8 predictions. The average concept size in the leaf nodes of a fully grown hierarchy is of 4.7 predictions (the method of calculating this figure is omitted here). The goal of this experiment is to evaluate the impact of feature transformation, not only in terms of the accuracy and compactness of the learned hierarchy, but also in terms of its *coverage* of the conceptual space. Coverage is defined as the ratio between the average concept size in the leaf nodes of the generated hierarchy and the average concept size in the leaf nodes of a fully grown tree. For instance, if the average concept size in the leaf nodes of the hierarchy generated for the problem above is 2.35, its coverage will be $2.35/4.7 = 50\%$.

Coverage may be much less than unity when the training set does not contain enough information. In that case, expansion of the tree is stopped before predictions have been determined for all attributes. The particular stopping criterion that was used is to discard every prediction for which the anticipated accuracy is not above 70%. The results obtained for this structured learning problem are shown in Table 23.4. This time, strategy S5 (all features) was consistently the most successful, followed by strategies S3 (summary features) and S4 (Fourier features). Since the minimum allowed accuracy for a prediction is of 70%, the average accuracy is high, even in the case of strategy S1. What varies more clearly is coverage of the conceptual space. While using only the raw features the coverage is only 46%, using the Fourier features (strategy S4) or all features (strategy S5) raises coverage to 66%.

Typically, the predictions added in the lower levels of a concept hierarchy have lower accuracy. This means that, if the stopping criterion is very permissive, the hierarchy will be large but not very accurate. On the other end, if the stopping criterion is very severe, the hierarchy will be more compact and accurate, but less informative (less predictions, smaller coverage). In the meantime, if the number and the discrimination power of the features used in describing the examples increase, it will be possible to accept more predictions and create more nodes. But, simultaneously, it will be possible to optimize the size of the hierarchy. For this reason, there is no clear rule to explain the impact of features in the size of the hierarchy. Accuracy increases (between 3% and 10% in Table 23.4) when better features are provided. This is the obvious effect of providing more information for learning while keeping the threshold for prediction acceptance. In structured problems, coverage is the measure with the more direct dependence on the quality of discrimination features.

Table 23.4 Impact of feature transformation on a structured problem

	S1	*S2*	*S3*	*S4*	*S5*
Experimental accuracy (%)	82	85	90	87	92
Coverage (%)	46	59	52	66	66
Size (nodes)	18	25	21	19	22

23.8 CONCLUSIONS

This chapter described the application of feature transformation to the problem of learning diagnosis knowledge about execution failures in robotics. The original features are time dependent and the transformation strategies take that into account.

Some experiments with constructive induction algorithms were performed (and only partially described above). Since no good results were obtained, we tried the alternative approach of using pre-programmed feature transformation strategies. These strategies were developed specifically for our application. The size of the feature vectors after feature transformation varies between 48 and 490 features. Summary features, like averages and slopes, extracted from sensor traces, were quite successful in improving the accuracy and compactness of the learned hierarchies. Frequencies, extracted by the fast Fourier transform, did also contribute to improve accuracy, although to a lesser extent. In some learning problems, feature transformation led to accuracy improvements of 30% or more and to reductions in tree size to half the size of the trees generated with raw features only.

It must be noted that feature transformation is a highly domain-dependent topic. The transformation strategies that we developed for force-based diagnosis in assembly can give inspiration to develop methods for other domains, but probably won't give the best results if directly applied. What was described is no more than a case study. We believe that future developments in feature transformation should be concerned in identifying major classes of objects and major feature transformation strategies for those objects. One example of such

objects in our particular case study are the force/torque profiles. Profiles appear in many domains and, independently of the particular variables whose evolution they describe, there are a certain number of features that may make sense to extract. It seems, therefore, that a good methodology for feature transformation should look at examples not as feature vectors but as collections of standard objects. Precisely, in our case study an example was a collection of six profiles.

References

Botta, M. and Giordana, A. (1993). Smart+: A multi-strategy learning tool. In *Proc. International Joint Conference on Artificial Intelligence*, pages 937–943.

Bracewell, R. (1986). *The Fourier Transform and its Applications.* McGraw-Hill, Singapore.

Breiman, L., Friedman, J., Oleshen, R., and Stone, C. (1984). *Classification and Regression Trees.* Wadsforth International Group.

Camarinha-Matos, L., Seabra Lopes, L., and Barata, J. (1996). Integration and learning in supervision of flexible assembly systems. *IEEE Transactions on Robotics and Automation*, 12:202–219.

Kaiser, M., Giordana, A., Nuttin, M., and Seabra Lopes, L. (1995). B-learn ii: Combining sensing and action (final project report: Short version). Technical report, ESPRIT BRA 7274 B-LEARN II.

Koller, D. and Sahami, M. (1997). Hierarchically classifying documents using very few words. In *Proceedings of the International Conference on Machine Learning*, pages 170–178.

Kuniyoshi, Y., Inaba, M., and Inoue, H. (1994). Learning by watching: Extracting reusable task knowledge from visual observation of human performance. *IEEE Transactions on Robotics and Automation*, 10:799–822.

Michalski, R. (1983). A theory and methodology of inductive learning. *Artificial Intelligence*, 20:111–161.

Murthy, S., Kasif, S., Salzberg, S., and Beigel, R. (1993). Oc1: Randomized induction of oblique decision trees. In *Proceedings of the Eleventh National Conference on Artificial Intelligence.*

Quinlan, J. (1986). Induction of decision trees. *Machine Learning*, 1:81–106.

Quinlan, J. (1993). *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers, San Francisco, California.

Seabra Lopes, L. (1997). *Robot Learning at the Task Level. A Case Study in the Assembly Domain (Ph.D. Thesis).* Universidade Nova de Lisboa, Portugal.

Seabra Lopes, L. and Camarinha-Matos, L. (1995). Inductive generation of diagnostic knowledge for autonomous assembly. In *Proc. IEEE International Conference on Robotics and Automation*, pages 2545–2545.

Seabra Lopes, L. and Camarinha-Matos, L. (1996). Learning failure recovery knowledge for mechanical assembly. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems.*