View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Helsingin yliopiston digitaalinen arkisto

Department of Computer Science
Series of Publications A
Report A-2008-1

# Modeling Efficient Classification as a Process of Confidence Assessment and Delegation

Ilkka Autio

Academic Dissertation

*To be presented, with the permission of the Faculty of Science of the University of Helsinki, for public criticism in Auditorium XIII, University Main Building, on February 1st, 2008, at 12 o'clock noon.*

University of Helsinki
Finland

Copyright © 2008 Ilkka Autio

ISSN 1238-8645
ISBN 978-952-10-4462-5 (paperback)
ISBN 978-952-10-4463-2 (PDF)
http://ethesis.helsinki.fi/

Computing Reviews (1998) Classification: I.5, I.5.1, I.5.2, I.4, I.4.7, I.2.6

Helsinki University Printing House
Helsinki, Jan 2008 (212 pages)

# Modeling Efficient Classification as a Process of Confidence Assessment and Delegation

Ilkka Autio
Department of Computer Science
P.O. Box 68, FI-00014 University of Helsinki, Finland
ilkka.autio@cs.helsinki.fi

## Abstract

In visual object detection and recognition, classifiers have two interesting characteristics: accuracy and speed. Accuracy depends on the sophistication and complexity of the image features and classifier decision surfaces. Speed depends on both the hardware and the computational effort required to use the features and decision surfaces. When attempts to increase accuracy lead to increases in complexity and effort, it is necessary to ask how much are we willing to pay for increased accuracy. For example, if increased computational effort implies quickly diminishing returns in accuracy, then those attempting to design inexpensive or embedded surveillance applications cannot aim for maximum accuracy at any cost. It becomes necessary to find satisfactory trade-offs between accuracy and effort.

We study efficient classification of images depicting real-world objects and scenes. Classification is efficient when a classifier can be controlled so that the desired trade-off between accuracy and effort (speed) is achieved and unnecessary computations are avoided on a per input basis. In studying efficient classification, it is desirable to have a framework in which similar classification problems can be understood and handled with relative ease. In this dissertation, a framework is proposed for understanding and modeling efficient classification of images. In the framework, classification is modeled as a tree-like process. In designing the framework, it is important to recognize what is essential for many problems and to try to avoid structures and techniques that are narrow in applicability. Earlier frameworks are lacking in this regard, e.g., in one particular framework, efficient classifiers have a cascade (degenerate tree) structure in which all nodes use the same kind of features.

The overall contribution of this dissertation is two-fold. First, the framework is presented, subjected to experiments, and shown to be satisfactory for the intended purpose. Second, certain unconventional approaches are experimented with. This allows the separation of the essential from the conventional, i.e., for the purpose of detecting what does not have to be and should not be required in the framework. For example, it is conventional to assume that the desired trade-off must be specified prior to training. It will be shown that this is unnecessary. To determine if the framework is satisfactory, and to determine its limitations, three categories of questions are identified: trade-off optimization, classifier tree organization, and rules for delegation and confidence modeling.

Related to trade-off optimization, we first address the problem of selecting suitable root node features. A hypothesis is proposed for feature selection. Using the hypothesis, it is possible to avoid computational bottlenecks that limit the available range of trade-offs. Two different experiments are contributed, both of which agree with the hypothesis. Second, it is claimed that if the framework is implemented properly, then accuracy versus effort (speed) trade-offs can be controlled after training, i.e., the desired trade-off can be changed without training the classifiers again. Empirical evidence is contributed and the evidence supports the claim.

Regarding classifier tree organization, we first consider the task of organizing a classifier tree in a problem-specific manner. An organization of the nodes is problem-specific if the classification problem at hand determines which node is connected to which. For finding the organization, an unconventional approach is designed and experimented with. In the approach, tree-like hierarchies of classes are created by using visual queries posed to human observers. The experimental results support the use of the approach, and it is demonstrated that the approach is compatible with the framework, i.e., with the goal of trade-off optimization. Second, we ask if problem-specific organization is strictly necessary. The experimental results support the claim that, at least sometimes, the framework does not require problem-specific organization. The key idea is that nodes can be formed from simpler modules when required.

Related to delegation and confidence modeling, we first ask if predictions can be combined efficiently over multiple views of objects under motion. The experimental results support an answer

in the positive, and it is demonstrated that the proposed delegation rules are not limited to classifying single views. Second, a different kind of multi-class delegation rules are subjected to theoretical analysis. These rules are the ones that are used for avoiding problem-specific organization. The analysis shows that the rules, which are based on monotonic confidence modeling, can exceed the accuracy of a simple baseline approach, and that excess accuracy can be traded for speed. Third, a non-monotonic confidence modeling approach is contributed and subjected to analysis. The approach is based on adaptive combination of large-margin classifiers. In the analysis, it seems that the combination operation is not especially dangerous because the risk of overfitting is not necessarily increased. Further, the non-monotonic approach is experimented with, and the results are compared to results from monotonic modeling.

Finally, we address one question that is not covered by the three categories. In contrast to the efficiency of classifying inputs, we examine the efficiency of feature selection and training, and ask if the framework is useful in improving the latter in addition to the former. The results support an answer in the positive. The key idea is to use a certain kind of modules in the root node to constrain the sampling of candidate features in other nodes.

*Computing Reviews (1998) Categories and Subject Descriptors:*

I.5 Pattern Recognition
I.5.1 Models
I.5.2 Design Methodology
I.4 Image Processing and Computer Vision
I.4.7 Feature Measurement
I.2.6 Learning

*General Terms:* Computer Vision, Experimentation

*Additional Key Words and Phrases:* Computer Vision, Pattern Recognition, Machine Learning, Efficient Methods for Visual Pattern Recognition

# Acknowledgements

I am grateful to my team of supervisors and mentors Esko Ukkonen, Tapio Elomaa, and Jyrki Kivinen. I would like to thank Esko for the guidance, encouragement, and the facilities and resources necessary for finishing this manuscript and the related works. I am grateful to Tapio for participating in my earliest works and for remaining interested in my progress even after circumstances led to his move to Tampere. I would like to thank Jyrki for his expertise and advice on all things related to machine learning. I am especially grateful to Juha Röning and Jorma Laaksonen for reviewing this manuscript and providing helpful insight and criticism.

I would like to thank the Helsinki Graduate School in Computer Science and Engineering (HeCSE) and the From Data To Knowledge (FDK) research unit for funding and the opportunity to conduct my research. The former organized many stimulating summer courses that opened my eyes to interesting topics of study that I would not have encountered otherwise. I am also grateful to the Department of Computer Science at the University of Helsinki for providing many essential services and for offering interesting courses.

I am very grateful to my co-workers, especially Jussi T. Lindgren whose enthusiasm for experimental work was admirable, and whose critical eye was often valuable in drawing the line between the feasible and the hopeless.

Last but not least, I am most grateful to my parents Sirkka and Kalevi, and my beloved girlfriend Natalia, without whom I would not have had the strength to keep going.

# Contents

CHAPTER 1

# Introduction

In visual object detection and recognition, we have classifiers that have two characteristics of interest: accuracy and speed. Accuracy depends on the suitability and sophistication of the extracted image features and the capability of the classifier to use accurate decision surfaces in the space of the features. Speed depends on both the hardware and the computational effort required to calculate the feature values and to use the decision surfaces.

To discuss the relationship between accuracy and speed, and the importance of both, it is best to begin with something that many readers can agree with: given a classification problem and many classifiers of equal accuracy, the fastest one is the best. We can now ask what it takes to increase the accuracy of the best solution to some desired level. One possibility is that the solution has to become more complex, increasing the computational effort of classifying inputs. For example, consider a perceptron or a support vector machine that takes raw pixel vectors as inputs that are classified effortlessly and with an accuracy that is significantly above chance level. The classifier, however, cannot approximate the correct decision surface(s) arbitrarily well. To increase the accuracy to the desired level, it may be necessary to allow more complex decision surfaces in the classifier, or to allow more sophisticated input features in the space of which the correct decision surface is easier to approximate. Both improvements increase the computational effort. The other possibility, that any desired accuracy can be achieved without increasing the complexity or effort inherent in the solution, is not universally realistic. Above, we already fo-

cused on the best solution of those with equal accuracy, i.e., badly programmed solutions were ruled out.

Based on the above, we suppose that for some classification problems there are thresholds at which increases in accuracy must lead to increases in complexity and computational effort. It now becomes necessary to ask how much we are willing to pay for increased accuracy. For example, if increased computational effort leads to quickly diminishing returns, then those who would seek to embed face recognition programs in inexpensive surveillance cameras should admit that they need a *trade-off* between accuracy and effort – not maximum accuracy at any cost.

In this dissertation, we study efficient classification, i.e., how to make trade-offs when accuracy and effort (speed) are in conflict. Related to the above motivation, it will be shown that in the parameter spaces of efficient classifiers there are clearly identifiable points of diminishing returns. The conflict arises when we consider real-world image classification problems that require speed and are associated with the use of complex features, multiple kinds of features, or complex preprocessing (possibly segmentation) to achieve good accuracy. Examples of such problems can be found in the literature [VJ01, UVNS02, VNU03, CDV03, BBFS00, OPS$^+$97, MKS00, GJLAMBGM05] relevant to surveillance applications and vehicle mounted applications, e.g., detection and recognition of faces, cars, pedestrians, and traffic signs. We examine problems in the context of supervised batch learning, and make one basic assumption in the methodology. We assume that efficient classifiers use conditional exclusion of computations on a per input basis, e.g., as assumed also in [VJ01]. If computations are not excluded on this basis, both easy and hard inputs take the same effort, which is hardly efficient.

Given that there are many classification problems in which the basic conflict appears to be present and the need for controlled trade-offs emerges, it seems wasteful to consider each problem in isolation from the others. It would be useful to have a framework for understanding and modeling efficient classification of images depicting real-world objects and scenes. In this dissertation, such a framework will be presented. Informally, a framework is a supporting structure within which similar problems can be handled with relative ease. A framework may include theory, structures for or-

ganizing information, and software. In designing a framework, it is necessary to recognize what is essential for a family of problems and avoid structures or techniques that are too narrow in applicability. For example, in designing the desired framework, it would be a mistake to assume that all efficient classifiers can use the cascade structure of Viola and Jones [VJ01] to achieve conditional exclusion, or that each node of the structure can use the same kind of features. Yet, Viola and Jones claim that they present a framework and not just a solution to one specific problem. In addition, designing a framework enables one to separate the essential from the conventional, i.e., to see what is widely accepted, but does not have to be.

The overall contribution of this dissertation is two-fold. First, the framework is presented, subjected to experiments (e.g., related to trade-offs), and shown to be satisfactory. In brief, the framework models the classification process as a tree, enables the combination of many kinds of image features in a manner that isolates incompatible features from each other, and allows freedom of choice when there is no unique solution to a subproblem, e.g., choosing the organization and placement of the nodes in the tree. Second, there are subproblems that allow the use of some unconventional approaches. These approaches are experimented with. It will be shown that certain conventional approaches have attractive alternatives, and hence, the conventions should be considered unnecessary. For example, it will be shown that trade-offs can be controlled after training by the use of simple parameters. In contrast, decision trees and typical cascades that allow trade-offs require that one specific trade-off is chosen prior to training.

## 1.1 Organization and contributions

The framework, along with some initial analysis, is presented in Chapter 2. To determine if the framework is satisfactory, and to determine its limitations, it is necessary to present answers related to three categories of questions.

1. *Trade-off optimization.* The framework must allow predictable control over trade-offs. The available range of trade-offs should be wide, e.g., high accuracy should be available to those who desire it and high speed (low classification effort) should be available to others. It is necessary to ask if and how sufficient control is achievable in practice.

2. *Classifier tree organization.* The framework has certain requirements related to the organization and structure of classifier trees. For example, computationally lighter nodes should precede heavier nodes and every node must be able to terminate classification. It is necessary to ask if there are practical means to satisfy the requirements. *A priori*, this is not obvious. For example, knowledge of ordinary decision trees does not provide the answers because the requirements are different. In decision trees, only leaf nodes can terminate and there is no concept of computational effort associated with nodes. Knowledge of cascades (degenerate trees) does not provide the answers either.

3. *Delegation and confidence.* The framework requires delegation rules and confidence modeling schemes. Prior to Chapter 2, it suffices to say that delegation rules decide the paths taken by inputs and confidence models decide if nodes predict or abstain. It is necessary to determine what kind of rules and models are available and what capabilities and limitations they have.

Related to *trade-off optimization*, the following is presented. *First*, in Chapter 3, we address the problem of selecting suitable root node features. A hypothesis for feature selection is contributed. The hypothesis makes claims about what kind of features should be used in root nodes. Basically, if certain preconditions are met, there is a good chance that global summations or other coarse statistics of

local features are suitable. Root nodes deserve special attention because these nodes can limit the available range of trade-offs so that the limitations cannot be overcome by delegation rules. In other words, unsuitable root nodes lead to insufficient control over trade-offs. The main experiment of Chapter 3 satisfies the preconditions and claims of the hypothesis. In Chapter 4, a related experiment is presented. The second experiment complements the first and also satisfies the preconditions and claims. Thus, two different experiments are contributed and both agree with the hypothesis.

*Second*, in Chapter 4 and in the first half of Chapter 5, we address the problem of controlled trade-offs. Experimental evidence related to the problem is contributed. The evidence supports the claim that if the framework is implemented properly, then speed versus accuracy trade-offs can be controlled (optimized) after training, i.e., the desired trade-off can be changed without training the classifiers again. This unconventional control is sufficient and is achieved by the use of simple parameters. Proper implementation involves details related to the root node, confidence modeling, and delegation rules. It is important to avoid the kind of input filtering that is done by cascades of Viola and Jones and decision trees. Such filtering is not compatible with the idea of changing the desired trade-off without training again.

Related to *classifier tree organization*, the following is presented. *First*, in Chapter 4, we ask where the organization of the base classifier nodes comes from. It is supposed that there are broad classes and narrow classes, and that confusing broad classes is relatively serious compared to confusing narrow classes. For finding the organization, an unconventional approach is presented and experimented with. An experimental procedure is designed for creating tree-like hierarchies of classes by using visual queries posed to human observers. One hierarchy is then used as the organization of the nodes. The contributed experimental evidence supports the use of the approach. The results show that the approach is compatible with the requirements of the framework and efficient classification in general, i.e., trade-off optimization works well.

*Second*, in the first half of Chapter 5, we ask if the framework requires problem-specific organization of the classifier nodes. An organization of the nodes is problem-specific if the classification problem at hand determines which node is connected to which. For

example, decision trees are problem-specific and it is conventional to assume problem-specificity. The contributed experimental evidence supports the claim that, at least sometimes, the framework does not require problem-specific organization. The evidence is based on the use of the *multi-class delegation rules* proposed in the first half of Chapter 5. The unconventional idea is that nodes are formed from simpler modules.

Related to *delegation and confidence*, the following is presented. *First*, in Chapter 4, we ask whether predictions can be combined efficiently over multiple views of objects under motion. In this task, efficiency is inversely proportional to the sum of total classifier losses over motion sequences. The contributed experimental evidence supports the positive answer, e.g., the delegation rules are not limited to classifying single views. Each prediction is represented as a ranked list of class labels. Simple voting-based combination of these lists is found sufficiently efficient, given that the classifier tree can do simple motion segmentation.

*Second*, in the first half of Chapter 5, the proposed *multi-class delegation rules* are subjected to theoretical analysis. These are the rules that are used for avoiding problem-specific organization. It is assumed that monotonic confidence modeling is appropriate. The analysis shows that the rules can exceed the accuracy of an alternative approach. The excess accuracy can then be traded for speed.

*Third*, in Chapter 2, a non-monotonic confidence modeling approach is contributed and subjected to analysis. The approach is based on adaptive combination of classifiers. In the analysis, it seems that combination by using this approach is not especially dangerous because increasing the number of classifiers does not necessarily increase the risk of overfitting. In the second half of Chapter 5, we ask if pre-made classifier modules can be combined adaptively to work together in (non-root) nodes. A module is pre-made if it has been made or trained prior to combining, i.e., it is not made *during* combining. As a counterexample, boosting does not allow pre-made modules. The contributed experimental evidence supports the claim that adaptive combination of pre-made modules works in practice. The proposed non-monotonic approach from Chapter 2 is compared to a non-adaptive monotonic alternative. The non-monotonic approach is found slightly better.

Finally, we address one question that is not covered by the three categories of questions that were explained above. We examine the efficiency of feature selection and training (in contrast to the efficiency of classifying inputs). In the first half of Chapter 5, we ask whether the framework is useful in improving the efficiency of feature selection and training. The contributed experimental evidence supports the positive answer. The efficiency of feature selection is improved by using modules in the root to constrain the sampling of candidate features in other nodes.

To summarize, using the contributed experimental results and analysis it will be shown that the proposed framework is satisfactory for understanding and modeling efficient classification and that there are common conventions that seem to be unnecessary or even counterproductive for efficient classification.

## 1.2 Related Publications

The chapters of this dissertation are based on peer-reviewed works that were published prior to this dissertation. Chapter 2 is exceptional in the sense that it has the least direct relationship with any single publication. It integrates information from all of the publications. Chapter 3 is based on early works that were originally presented in [AE03], [AEK01a], and [AEK01b]. Chapter 4 is based on [Aut06]. The first half of Chapter 5 is based on [AL04], and the second half is based on [ABI+05]. Not all of the related published works are included in this dissertation. An excluded work [AL06], examines the selection of computationally demanding features in the context of online-learning, and may be of related interest.

The chapters also extend the publications. For example, Chapters 4 and 5 extend the empirical results of [Aut06] and [AL04] by presenting many parameterized trade-offs whereas the original publications did not.

## 1.3 Summary of the datasets used in the experiments

A short summary of the datasets used in the experiments is presented below. To understand why an experiment requires a particular kind of dataset, it is necessary to understand the experiment in some detail. The chapters have separate introductions in which the empirical questions, the experiments, and the datasets are explained in sufficient detail. Below, the datasets are summarized to the extent that is possible here.

In the main experiment of Chapter 3, the data was collected by using a mobile robot. The data consisted of grayscale images depicting a typical office environment and people found there. The images were divided into four classes: doorways, signs, (close-up views of) people, and miscellaneous. The overall idea of the collection procedure was that the targets were perceived in their correct contexts, viewed from reasonable viewpoints. As required by the hypothesis that is examined in Chapter 3, the classes are human-recognizable in low resolution and seem to belong to distinct superordinate categories. The complete control of the data collection process also allowed experimentation with sampling bias. Sampling bias is given special attention in Chapter 3.

In Chapter 4, data from the set *ETH80* [1] was used. The dataset has 80 objects from 8 basic level categories (ordinary classes) that are labeled as apple, car, cow, cup, dog, horse, pear and tomato. The plain backgrounds were replaced by images depicting real-world environments. From the data, it is possible to discover superordinate categories (broad classes). The discovery of superordinate categories is essential to the proposed approach of finding the organization of the base classifier nodes. Also, the dataset allows the controlled rotation of objects. This is required when the problem of combining predictions over multiple views (motion) is examined.

In the first half of Chapter 5, the car dataset of Agarwal and Roth [AR02], the face dataset of Martinez and Benavente [MB98], and the BioID data of Jesorsky, Kirchberg and Frischholz [JKF01]

[1] `http://www.vision.ethz.ch/projects/categorization/eth80-cropped256.tgz`

were used. There were three classes: cars, faces, and non-objects (miscellaneous backgrounds). Experiments related to improving the efficiency of feature selection and training were performed. Hence, it was reasonable to use fragment features resembling those of Ullman, Vidal-Naquet, and Sali [UVNS02, VNU03] that can be characterized as inefficient but useful. Fragment features are suitable for rigid and semi-rigid objects. Faces and cars were thus appropriate.

In the second half of Chapter 5, images from the repository of the Diabetes Control and Complications Trial (DCCT) [The90] were used. The images depicted retinal fundus of patients with retinal microaneurysms. The microaneurysms were the class of interest. The data was suitable for investigating the adaptive combination of classifiers because a problem was required in which it is difficult to find suitably accurate feature spaces (efficient or not). This was required because if there exists a superior feature space (a set of compatible features), then there is less motivation for combination (although delegation is a different matter). There is less motivation because given the superior feature space, one can likely train a sufficiently good individual classifier.

As can be seen from the above, datasets depicting everyday objects and scenes were favored. In the data mentioned above, the object backgrounds were never trivial. As apparent, the datasets vary a lot in the sense that they are associated with different classification problems. This is appropriate because the current framework is of general interest, e.g., it is not a face detection framework nor a microaneurysm detection framework.

CHAPTER 2

# The delegation framework for efficient classification

In this chapter, the framework and some initial analysis are presented. The viewpoint here is entirely theoretical in contrast to the following chapters that are dedicated to practical questions and experiments. In short, the framework models the process of classification as a tree. Each node of the tree can be activated due to an input image. All input images activate the root node of the tree. When a node is activated, it gets a chance to do useful work. Depending on the current input, the node may fail, i.e., the work done by the node may turn out to be rather useless. Whether the node succeeds or fails, the activation results in a cost related to the use of computational resources (effort, time). The costs may vary significantly from node to node. The usefulness of the work done by an activated node is decided by a *confidence model*. The tree operates by the use of *delegation rules* that, for each input, choose the path to be taken by the input. Good delegation rules direct inputs to those nodes that can do useful work in predicting the class labels of the inputs. For example, if the confidence model does not trust in the prediction of the root node, the input is delegated to some other node.

The design of the framework is such that incompatible features are isolated from each other. In many classification problems, it is advantageous and efficient to use multiple kinds of features, but using one feature space to contain all the features may cause difficulties. For example, it may be difficult to handle a support vec-

tor machine that sees raw pixel values, Gabor filter responses, and binary template matching indicators at the same time. More importantly to us, features may be incompatible due to differences in the computational effort required to use them. For this reason, we *encapsulate* feature spaces within the nodes. Different nodes use different kinds of features. The system level, i.e., the level of the delegation rules, cannot access encapsulated features and, in turn, cannot transmit the values of encapsulated features between the nodes. For example, a node may contain a base classifier that uses its own feature space (say, template matching indicators only) and that space is not accessed outside the node. The use of encapsulation makes it simple to characterize the cost of activating a node.

We proceed as follows. We begin by discussing typical loss functions that are used in supervised learning of classifiers, and then proceed to efficiency-sensitive losses that are used to make resource (time, effort) consumption explicit in optimization criteria and analysis. We will then be able to analyze the tree model with respect to so-called zones of confidence and other related quantities, and can gain some insight into the behavior of the model. We then proceed to the topic of base classifier (node) design, which is closely related to the topic of confidence modeling. We note that the classification margins of large-margin classifiers seem to be suitable for modeling confidence. We examine monotonic and non-monotonic confidence models for large-margin classifiers, and develop one non-monotonic modeling approach that allows the combination of classifiers.

## 2.1 Efficient tree models

The tree-like model of classification, as presented in this chapter, does not describe a complete implementation of a classification system. The model, however, is an essential part of the framework that is used in the experiments. Before examining the model, we examine an overview of the necessary fundamentals.

### 2.1.1 Classifiers and loss functions

A classifier is a program that measures properties of objects, based on which it attempts to predict the class memberships of the objects. Let $f : \mathbb{R}^d \rightarrow \mathit{Classes}$ denote a classifier function mapping $d$-dimensional real-valued inputs (e.g., image rasters such that $\mathit{width} \times \mathit{height} = d$) to a finite set of class labels. If the class labels are not mutually exclusive, we may allow $f$ to predict multiple labels at once. In that case, the range of $f$ is the power set of *Classes*, i.e., $2^{\mathit{Classes}}$ using common notation. The outputs $f(\mathbf{x})$ of $f$ are called predictions from inputs $\mathbf{x}$.

In addition to the predictions, we have $y \in \mathit{Classes}$ denoting the correct class of the input $\mathbf{x} \in \mathbb{R}^d$. Note that in the general case $\mathbf{x}$ may contain insufficient information for determining $y$ reliably – even if $f$ is optimal. If $\mathbf{x}$ may belong to multiple classes at once, then $y$ is a set that is an element of the power set, i.e., $y \in 2^{\mathit{Classes}}$.

It is given that the designer has some preferences related to the relationship between the predictions and the true classes. The goal of classifier optimization is to create the preferred relationship. The optimization process is usually at least loosely grounded in utility theory [Fis70], probability and statistics [DS01].

Utility theory enforces consistency in encoding preferences. If a hypothetical designer creates a search procedure without precise preferences in mind (e.g., the designer wants to see what "emerges") and the procedure is not completely random, then there are implicit preferences for certain states over others even if the designer fails to appreciate these.

Probability theory and statistics provide well-established tools for dealing with the uncertainty over the interpretation of measurements and the weighting of the preferences. The classifiers we are interested in are connected to the physical world, and thus require tools for coping with the uncertainty resulting from our incomplete understanding of the world.

We assume *cardinal utilities* for encoding preferences, i.e., we have *loss functions* for mapping predictions $f(\mathbf{x})$, the input vectors $\mathbf{x}$ and the correct answers $y$ to numbers indicating how bad the predictions are, given the answers. Another, perhaps interesting, choice could be *ordinal utilities*. When using ordinal utilities, the prediction outcomes are ranked by preference instead of having ab-

solute numeric values. If there is uncertainty over preferences, then it may sometimes be easier to work with ordinal utilities.

**Definition** Let

$$L_{(1,1)} : \mathit{Classes} \times \mathit{Classes} \times \mathbb{R}^d \rightarrow \mathbb{R}, \tag{2.1}$$

$$L_{(0+,1)} : 2^{\mathit{Classes}} \times \mathit{Classes} \times \mathbb{R}^d \rightarrow \mathbb{R}, \tag{2.2}$$

$$L_{(1,0+)} : \mathit{Classes} \times 2^{\mathit{Classes}} \times \mathbb{R}^d \rightarrow \mathbb{R}, \tag{2.3}$$

$$L_{(0+,0+)} : 2^{\mathit{Classes}} \times 2^{\mathit{Classes}} \times \mathbb{R}^d \rightarrow \mathbb{R}. \tag{2.4}$$

Functions with the forms $L_{(1,1)}, L_{(0+,1)}, L_{(1,0+)}$ and $L_{(0+,0+)}$ are syntactically valid real-valued loss functions. The shorthand 0+ means "zero or more labels", and $\mathbb{R}^d$ is included to allow the possibility of the inputs $\mathbf{x} \in \mathbb{R}^d$ affecting the losses more directly than just through the predicted labels.

If we suppose that the inputs $\mathbf{x} \in \mathbb{R}^d$ have no direct effects on the losses, e.g., $L_{(1,1)}(f(\mathbf{x}), y, \mathbf{x}) \equiv L_{(1,1)}(f(\mathbf{x}), y)$, then losses of the form $L_{(1,1)}$ from above include the 0/1 loss and similar losses that assign different constant penalties for different types of prediction mistakes, possibly weighting some mistakes over others. These constant losses are simple to understand as encoding preferences and not tied to any particular classifier architectures or optimization algorithms. On the other hand, they provide limited guidance for the optimization process. For example, the 0/1 loss does not recognize if $\mathbf{x}$ is such that a large perturbation of $\mathbf{x}$ would be necessary to change $f(\mathbf{x})$ to the correct value, thus hinting that $f$ needs large adjustments to accommodate $\mathbf{x}$.

Losses based on the other $L$-forms are seen less often in machine learning applications, although they are quite appropriate for problems involving multiple non-exclusive classes. For instance, class labels may be hierarchical, e.g., an object may be both *an animal* and *a dog*. In such a problem, a classifier may predict just the most specific label wrong, or it may omit the most specific label altogether. We will have hierarchic classes for visual classification in Chapter 4 (see Figure 2.1). In general, class hierarchies seem to be popular in document classification [KS97, MRMN98, WWP99, DC00, RS02], but less so in classification of images.

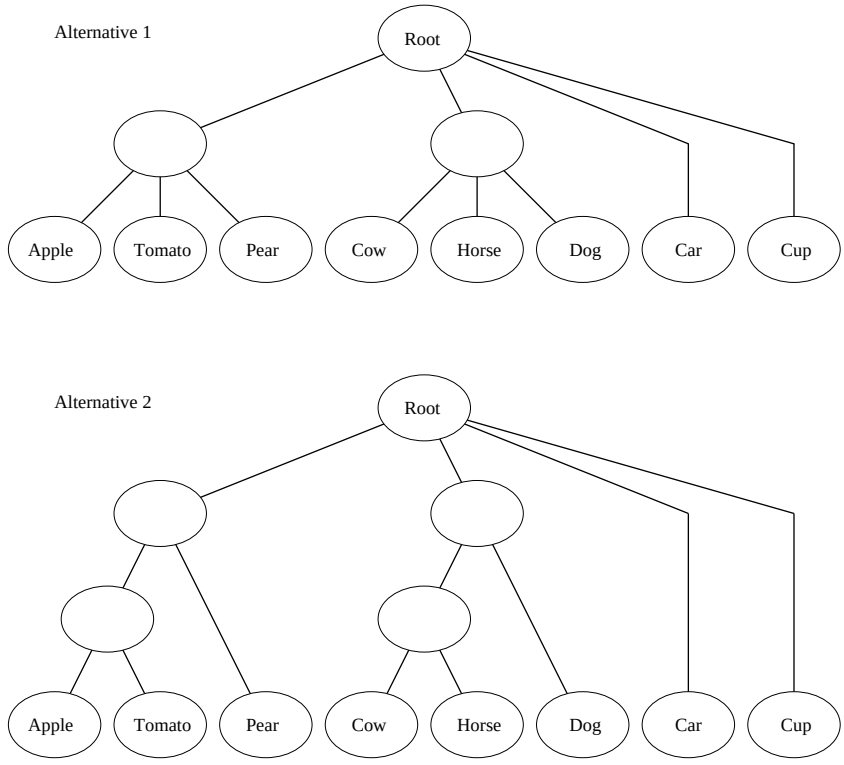



Figure 2.1: Two alternative class hierarchies extracted from query data. The unnamed parent node of *Dog* represents the superclass of animals.

It is often necessary to be flexible with the encoded preferences and choose surrogate loss functions such that the classifier becomes easier to optimize. For example, loss functions that are *differentiable* with respect to classifier parameters often allow the use of error-gradient-based optimization (e.g., classic neural network backpropagation [Hay99]).

When choosing surrogates, one important practical property is the convexity of the surrogate loss function. Convex optimization problems (problems based on convex loss) are, in general, easier to solve than non-convex ones [BV04]. For instance, the intuitive 0/1 loss is not convex, which often makes optimization difficult. Because the hinge loss [CBL06] is convex and also an upper bound on

the 0/1 loss, the hinge loss is often a promising surrogate for minimization. For a real-world example, the soft-margin SVM uses the hinge loss internally [CST00, SS01] even if the actual performance after optimization (learning) is typically measured using the 0/1 loss, i.e., counting mistakes.

To summarize, in what follows we will usually choose the form $L_{(1,1)}$ and treat the class labels as being mutually exclusive. When we examine the tree-like model, we view losses as encoding pure preferences, and do not discuss surrogates even if such are inevitable in practical implementations. To minimize expected losses, which is the *risk-neutral* choice, we also make certain typical assumptions about the distribution of the data.

### 2.1.2 Efficiency-sensitive loss functions

We will now consider losses of the form

$$L_{(1,1)} : \mathit{Classes} \times \mathit{Classes} \times \mathbb{R}^d \to \mathbb{R}$$

from Definition (2.1). Let $f : \mathbb{R}^d \to \mathit{Classes}$ denote a classifier function predicting $f(\mathbf{x}) \in \mathit{Classes}$ based on the input images $\mathbf{x} \in \mathbb{R}^d$. We denote the correct label by $y \in \mathit{Classes}$. Let $\mathit{prog}_f$ denote a program implementing $f$ for some reference computer. Clearly, multiple programs exist for most $f$ and some programs may be preferable over others. We extend the definition of loss by allowing $\mathit{prog}_f$ to appear in loss functions and define an additive loss

$$\mathit{Loss}(f(\mathbf{x}), y, \mathbf{x}, \mathit{prog}_f) = \mathit{ML}(f(\mathbf{x}), y) + \mathit{RL}(\mathbf{x}, \mathit{prog}_f), \qquad (2.5)$$

where $\mathit{ML}$ denotes *mistake* loss determined by the predicted and correct labels, and $\mathit{RL}$ denotes *resource consumption*-based losses. We make the common assumption that $\mathit{ML}(y, y) = 0$. The mistake losses do not have to be symmetric, i.e., we may have $\mathit{ML}(y, y') \neq \mathit{ML}(y', y)$ given $y \neq y'$. In what follows, we suppose $\mathit{RL}$ is determined by how much time $\mathit{prog}_f$ spends on $\mathbf{x}$.

The time spent by the program $\mathit{prog}_f$ on $\mathbf{x}$ is not necessarily proportional to the length of the program, and we are not interested in the difficult problem of discovering short programs implementing $f$ (see [LV93] for related issues). Finding the fastest program for $f$ is also a difficult problem. In principle, it is possible that for

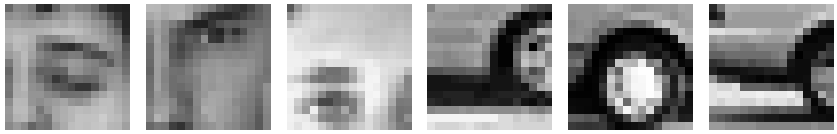


Figure 2.2: Partial image templates.

some $f$ there is no fastest program. For example, *Blum's Speed-Up Theorem* [Blu67] states that there exists a computable predicate $f(\mathbf{x})$ such that for any $prog_f$ with running time $t(\mathbf{x})$ there is another (larger) program for $f$ the running time of which is proportional to the logarithm of $t(\mathbf{x})$.

In the proposed model, $prog_f$ is constructed from *base classifier nodes* $prog_{f,k}$, $k \in \{1, \ldots, M\}$, that are activated using suitable rules. Efficiency-related optimization is limited to tuning the rules of how input images are delegated along the connections between the nodes. The programs $prog_{f,k}$ may be complex classifiers. The running times of the individual $prog_{f,k}$ are constants, $\gamma_k$, for each input image $\mathbf{x}$.

To understand why the running times of different $prog_{f,k}$ may vary significantly, consider two simple linear classifiers. The first classifier observes low resolution images $\mathbf{x}$ directly in terms of pixel values. The second classifier is not allowed to see pixels directly. It has a feature detector that detects the presence or absence of previously learned partial image templates (see Figure 2.2) using normalized cross-correlation to achieve limited position invariance. After feature detection, classification itself is computationally trivial. The differences in the running times of the classifiers are essentially determined by the features used, i.e., the size and the number of the partial templates used by the second classifier versus the number of pixels used by the first classifier. Based on the work in Chapter 5 it is reasonable to estimate that the number and sizes of the partial templates are such that the second classifier is significantly slower than the first one.

In principle, we seek to minimize the expected loss of the complete classifier program. The expected loss can be written as

$$\min_{prog_f \in \mathcal{P}} E[ML(f(\mathbf{x}), y) + RL(\mathbf{x}, prog_f)], \tag{2.6}$$

where $\mathcal{P}$ is the rather limited set of programs reachable by tuning the parameters of the delegation rules and the free parameters of the base classifiers. The above formula (2.6) defines the *Bayes strategy* or the *optimal Bayes classifier* [Mit97] for the problem.

### 2.1.3 The preferred model

To examine classifiers based on (2.6), we must have some constraints related to the underlying probability models and classifier structure. Here, we choose a tree-like structure for organizing the base classifier nodes.

We associate a random vector $\mathbf{X}$ with the observed image vectors $\mathbf{x} \in \mathbb{R}^d$, and when necessary, use subscripts $\mathbf{X}_i = \mathbf{x}_i$ to index the observations. Similarly, we associate a random variable $Y$ with the observed values $y \in \mathit{Classes}$, and use subscripts $Y_i = y_i$ to index the observations. We assume that the pairs $(\mathbf{X}_i, Y_i), (\mathbf{X}_j, Y_j)$, given $i \neq j$, are independent and identically distributed (i.i.d.) according to some unknown probability distribution. The i.i.d. assumption is commonplace in statistics and machine learning. For brevity, we denote $P(Y = y)$ by $P(y)$ when there is no risk of confusion. In addition, we abbreviate $\sum_{y \in \mathit{Classes} \setminus \{c\}}$ by $\sum_{y \neq c}$.

Each base classifier, $\mathit{prog}_{f,k}$ ($k \in \{1, \ldots, M\}$), is associated with two non-overlapping regions of the input space, $B_k$ and $\overline{B}_k$ such that

$$B_k \subset \mathbb{R}^d \text{ and} \tag{2.7}$$

$$\overline{B}_k = \mathbb{R}^d \setminus B_k, \tag{2.8}$$

where both regions consist of a finite number of continuous subregions each having non-zero volume, i.e., there are no isolated points. In practical use of the model, the regions could be specified using a few hyperplanes, for example. The region $B_k$ is interpreted as the set of inputs with which the $k$th base classifier is *confident*. The meaning of confidence is discussed later, but for now it suffices to know that the tree classifier makes decisions as if it trusted the predictions made by confident base classifiers. The confidence of a base classifier, however, does not necessarily mean that there is a valid reason to expect that the base classifier predictions are probably correct. Naturally, the base classifiers should be designed such that

this kind of confidence also indicates confidence in the well-defined statistical sense.

If a base classifier $prog_{f,k}$ is activated by an input $\mathbf{x}$, and $\mathbf{x} \in B_k$, then the base classifier is *confident* and outputs a proper class label (or labels, if the model is extended appropriately). Correspondingly, if a base classifier is activated and $\mathbf{x} \in \overline{B}_k$, then the base classifier *abstains* and may output a special label to indicate the fact. For convenience, we denote by $NaN$ the index of a special *overflow classifier* that is confident for all inputs, but always predicts wrong labels, if given the chance. Thus, $\overline{B}_{NaN}$ is the empty set.

Depending on the rules and connections by which the complete $prog_f$ operates, an individual base classifier is not necessarily activated for each input $\mathbf{x}$. Denoting the parameters of the ruleset by $\pi$, we define the subregions

$$A_{\pi,k} \subseteq B_k \text{ and} \tag{2.9}$$
$$\overline{A}_{\pi,k} \subseteq \overline{B}_k, \tag{2.10}$$

such that $\mathbf{x} \in A_{\pi,k}$ means that the $k$th base classifier is activated, confident, and outputs a proper class label $c \in \mathit{Classes}$. Correspondingly, $\mathbf{x} \in \overline{A}_{\pi,k}$ means that the $k$th base classifier is activated and abstains. Abstaining base classifiers do not make mistakes. The above subregions also consist of a finite number of continuous (sub)subregions each having non-zero volume. If $\mathbf{x} \in A_{\pi,k}$ and $prog_{f,k}$ predicts $c \in \mathit{Classes}$, we denote the event by $\mathbf{x} \in A_{\pi,k,c}$. The overflow event, $\mathbf{x} \in A_{\pi,NaN}$, is undesirable and it should be made unlikely.

The program $prog_f$ is organized like a tree or a cascade (a degenerate tree), and the inputs $\mathbf{x}$ propagate from a root node downward, following an input-specific unique path until $\mathbf{x} \in A_{\pi,k,c}$ for some $k$ or until there are no more nodes further down, resulting in the overflow classifier being activated. The propagation process is called delegation because the responsibility for handling the inputs propagates along with the inputs. For brevity, we denote by $crl(\pi, \mathbf{x})$ the cumulative resource losses of the path taken by $\mathbf{x}$. The overall arrangement is such that for each $\mathbf{x}$ there is exactly one $k$ (possibly $NaN$) such that $\mathbf{x} \in A_{\pi,k}$. Furthermore, $crl(\pi, \mathbf{x})$ is constant inside $A_{\pi,k}$.

The construct may be interpreted as a kind of decision tree (see [Qui93, Mit97]) such that each internal node is directly connected to at least one leaf node and a variable number of other internal nodes further down. In practice, the number of connections may be large. In Figure 2.1, we did not show all the connections used by the tree to be detailed in Chapter 4.

Our base classifier nodes are more complex than typical decision tree nodes, i.e., we have no simple splits to rectangular subregions based on individual components of $\mathbf{x}$. Further, optimization based on (2.6) may lead to solutions not considered by typical decision tree learners that only seek to maximize accuracy (i.e., 0/1 loss by surrogate). Based on (2.6), a fast node of mediocre accuracy may be preferable over an extremely slow node that has near perfect accuracy.

Beginning from (2.6), we examine the connections between the expected loss, the parameters $\pi$ of $prog_f$, and the regions (2.9, 2.10). We abbreviate $\min_{prog_f \in \mathcal{P}}$ by $\min_{prog_f}$ and $\max_{y,c}\{ML(c,y)\}$ by $MML$. We write

$$\min_{prog_f} E[ML(f(\mathbf{x}),y) + RL(\mathbf{x}, prog_f)] \tag{2.11}$$

$$= \min_{prog_f} \int_{\mathbf{x}} p(\mathbf{x}) \sum_y P(y \mid \mathbf{x}) \Big\{ ML(f(\mathbf{x}),y) + RL(\mathbf{x}, prog_f) \Big\} d\mathbf{x} \tag{2.12}$$

$$= \min_{prog_f} \Big\{ \int_{\mathbf{x}} \sum_y p(\mathbf{x}) P(y \mid \mathbf{x}) ML(f(\mathbf{x}),y) d\mathbf{x} + \int_{\mathbf{x}} p(\mathbf{x}) RL(\mathbf{x}, prog_f) [\sum_y P(y \mid \mathbf{x})] d\mathbf{x} \Big\} \tag{2.13}$$

$$= \min_{prog_f} \Big\{ \sum_y P(y) \int_{\mathbf{x}} p(\mathbf{x} \mid y) ML(f(\mathbf{x}),y) d\mathbf{x} + \sum_k P(\mathbf{x} \in A_{\pi,k}) crl(\pi, \mathbf{x}) \Big\} \tag{2.14}$$

$$= \min_{prog_f} \Big\{ \sum_y P(y) \sum_{k,c} P(\mathbf{x} \in A_{\pi,k,c} \mid y) ML(c,y) + \sum_k P(\mathbf{x} \in A_{\pi,k}) crl(\pi, \mathbf{x}) \Big\} \tag{2.15}$$

$$
\begin{aligned}
&\leq \min_{prog_f} \Big\{ \sum_{k \neq NaN, c} P(\mathbf{x} \in A_{\pi,k,c}) \sum_y P(y \mid \mathbf{x} \in A_{\pi,k,c}) ML(c,y) \\
&\quad + P(\mathbf{x} \in A_{\pi,NaN}) MML \\
&\quad + \sum_k P(\mathbf{x} \in A_{\pi,k}) crl(\pi, \mathbf{x}) \Big\} \qquad (2.16) \\
&\leq \min_{prog_f} \Big\{ \max_{k,c} \{P(Y \neq c \mid \mathbf{x} \in A_{\pi,k,c})\} MML \sum_{k \neq NaN, c} P(\mathbf{x} \in A_{\pi,k,c}) \\
&\quad + P(\mathbf{x} \in A_{\pi,NaN}) MML \\
&\quad + \sum_k P(\mathbf{x} \in A_{\pi,k}) crl(\pi, \mathbf{x}) \Big\} \qquad (2.17) \\
&= \min_{prog_f} \Big\{ \max_{k,c} \{P(error \mid \mathbf{x} \in A_{\pi,k,c})\} (1 - P(\mathbf{x} \in A_{\pi,NaN})) MML \\
&\quad + P(\mathbf{x} \in A_{\pi,NaN}) MML \\
&\quad + \sum_k P(\mathbf{x} \in A_{\pi,k}) crl(\pi, \mathbf{x}) \Big\}. \qquad (2.18)
\end{aligned}
$$

Above, (2.14) follows from cumulative resource losses being the same for all inputs with the same path. Step (2.15) follows from piecewise integration using pieces in each of which $ML(f(\mathbf{x}), y)$ is constant. Step (2.16) results from separating the special overflow classifier from the first sum and then estimating the losses of the overflow classifier upward. In (2.17), we take advantage of the fact that we may assume $ML(y, y) = 0$. If this was not true initially, we could normalize the losses to make it so.

For simplicity, abbreviate $P(\mathbf{x} \in A_{\pi,NaN}) = \rho(\pi)$. Now, examining the upper bound (2.18) allows some insight into the trade-offs inherent in the model. If it is possible to train the base classifiers so that the individual error probabilities are bounded,

$$P(error \mid \mathbf{x} \in A_{\pi,k,c}) \leq \epsilon(\pi)$$

for some $0 < \epsilon(\pi) < 1$ and for all $k, c$, i.e., the zones of confidence

($A_{\pi,k,c}$, $A_{\pi,k}$ and $B_k$) have reasonable interpretations, then we have

$$\min_{prog_f} E[ML(f(\mathbf{x}), y) + RL(\mathbf{x}, prog_f)] \tag{2.19}$$

$$\leq \min_{prog_f} \Big\{\epsilon(\pi)(1-\rho(\pi))MML + \rho(\pi)MML + \sum_k P(\mathbf{x} \in A_{\pi,k}) crl(\pi, \mathbf{x})\Big\} \tag{2.20}$$

$$= \min_{prog_f} \Big\{MML[\epsilon(\pi) - \epsilon(\pi)\rho(\pi) + \rho(\pi)] + \sum_k P(\mathbf{x} \in A_{\pi,k}) crl(\pi, \mathbf{x})\Big\}. \tag{2.21}$$

With typical base classifiers, it is reasonable to expect that when $\epsilon(\pi)$ is made smaller, the zones of confidence shrink, and thus the probability of overflow, $\rho(\pi)$, increases. Correspondingly, if we try to make overflows unlikely, then the zones of confidence should expand to cover more inputs, and tight bounds $\epsilon(\pi)$ on base classifier errors could become unachievable.

Examining mistake losses only, a classifier could in principle be a chain or a cascade of base classifiers activated in any arbitrary order with disjoint sets $B_k$. In this case, we would have $\forall k : B_k = A_{\pi,k}$ and the overflow probability would be

$$\rho(\pi) = P(\mathbf{x} \in A_{\pi,NaN}) = 1 - \sum_{k \neq NaN} P(\mathbf{x} \in B_k),$$

i.e., independent of any specific propagation rules $\pi$. This arrangement could result in a simple training process with some slight resemblance to boosting [MR03, Sch02]. The first base classifier would be trained on all the training data available, after which $B_1$ would become defined. The second base classifier would be trained on data from $\overline{B}_1$, $B_2$ would become defined, and the third base classifier would then be trained on data from $\overline{B}_1 \cap \overline{B}_2$ and so on. Quite attractively, this training process could make the overflow probability inversely proportional to the number of base classifiers trained.

Due to the arbitrary activation order of the base classifiers, the above cascade would likely be inefficient as measured by resource losses. While most inputs $\mathbf{x}$ could have exactly one base classifier confident in handling $\mathbf{x}$, the arbitrary rules $\pi$ could result in $\mathbf{x}$ going

through many unnecessary base classifiers not capable of confidence. Hence, it seems clear that not all rules $\pi$ are equal in efficiency, and it then becomes necessary to characterize what makes rules suitable.

In face detection cascades (e.g., [VJ04, VJ01]), it is possible to take advantage of class imbalances in designing efficient and fixed activation orderings that are input-independent except in that different inputs may need a different number of steps along the fixed path of activations. For these cascades, ordering the base classifiers by increasing resource losses is sensible. In general, there may be more than two classes and efficiency may require that the rules $\pi$ describe a proper tree such that each $\mathbf{x}$ has a delegation path that depends on the attributes of $\mathbf{x}$.

Now, if the zones $B_k$ were trustworthy and disjoint, then ideal efficiency-optimal delegation rules $\pi$ would immediately deliver each $\mathbf{x}$ to the single base classifier confident with respect to $\mathbf{x}$. Thus, with efficiency-optimal delegation rules, disjoint zones of confidence would be very desirable. In reality, we should expect the rules $\pi$ to be quite mistake-prone relative to the base classifier zones of confidence because the computations required for evaluating the rules should be quite simple. With flawed rules, we either have overflows or we append a suitable cascade to the end of each path. Hence, instead of disjoint zones it could be preferable that the base classifiers down the tree would *subsume* the classifiers above, i.e., $B_{above} \subset B_{below}$. The subsumption arrangement, in turn, makes sense only if the classifiers down the tree have larger resource losses than those above. Naturally, subsumption arrangements make little sense without the concept of resource losses because the subsumed classifiers are worthless otherwise.

Now consider subsumption. Down the tree some $B_k$ have to be large. As we said earlier, expanding zones of confidence could make tight $\epsilon(\pi)$ unachievable. To work around this problem, it is possible to try to use more computational resources down the tree. The use of sophisticated image features could result in base classifiers that are accurate within large zones of confidence.

## 2.2 Related hierarchical models

Based on the details presented in Section 2.1.3, we may now discuss related hierarchical models. We begin with degenerate trees (cascades) and then proceed to more generic models.

### 2.2.1 Cascades

When a tree of base classifiers is degenerate, i.e., simply a sequence of base classifiers, such that the rules $\pi$ of $prog_f$ do not allow the inputs $\mathbf{x}$ to skip nodes along the only available path, we may call $prog_f$ a *cascade* or a *pipeline*. Cascades are potentially simpler to build if there is a good efficient ordering of the nodes (base classifiers) available. The rules $\pi$ do not have to consider path-finding problems, as an input image may only propagate one step forward or not.

Cascades have become common in visual pattern recognition. Since Viola and Jones [VJ01, VJ04] presented their now famous method for face detection, methods based on similar principles have been appearing in the literature. In the Viola-Jones approach, the base classifiers of a cascade are learned using the *Adaboost* algorithm [SFBL97]. In the language of boosting, the base classifiers of the cascade are *strong* classifiers produced by Adaboost, and in turn consist of several *weak* classifiers, the number of which is related to the computational complexity of the strong classifier produced. The weak classifiers are essentially individual features selected from a large set. The Viola-Jones cascade does not use multiple kinds of features: all the features used by the weak classifiers are based on contrasts between rectangular subimages, somewhat resembling Haar wavelets [Chu92].

The Viola-Jones approach has been extended in several ways. In [FNSH04], the authors use a Viola-Jones-style boosted cascade for classifying laser scanner inputs common in robotics. Each cascade node uses weak classifiers based on edge, line, or center/surround features. The authors succeed in showing that the cascade approach works for inputs that are not produced by cameras in the traditional sense. In [ZLQH04], it is shown that boosted cascades based on Gabor filters [KB01, JP87] are feasible in face recognition. The authors use two kinds of Gabor-based features, thus having a greater va-

riety of features in an efficiency-sensitive cascade than what Viola and Jones had. In [LZ04], the authors show that suitably enhanced boosted cascades work with more complex object classes as well. The authors address multi-view face detection, meaning that the faces may be viewed from different directions (in [VJ01, VJ04] the views were frontal). Allowing multiple viewpoints increases the difficulty of the face detection problem due to increased within-class variation of the positive class.

More generally, cascades do not necessarily require boosting in building the (strong) base classifiers. In [EHOK02] the authors present a *maximal rejection classifier* that consists of a collection of hyperplanes. If the projection of an input vector is on the wrong side of a plane, the input is immediately rejected as a non-face. Otherwise the input is tested against the next hyperplane. Finally, if an input survives all the tests, it is classified as a face.

Both boosted and other cascades are, explicitly or implicitly, based on the notion that there is some *acceptable misclassification risk*, and that minimizing misclassification risk cannot be the only goal of a practical system. Assuming that losses are purely mistake-based and overfitting [GBD92, DHS00, Mit97, Bre01] is controlled, initially there seems to be no reason not to use all available tests or features for each input image. In principle, the class-related information provided by additional tests may never be negative (see [CT91] for basic properties of entropy). Thus, limiting the number of tests may seem like deliberately ignoring information based on which the misclassification risk could be decreased. In practice, both computational costs and mistakes resulting from the use of point estimates of information gain [Hut01, Hut02] imply that arbitrarily increasing the number of tests may be harmful. Suppose we define the acceptable misclassification risk. Once the actual risk is reduced below this limit, further tests may be omitted. For Viola-Jones cascades involving asymmetric class distributions, i.e., the number of negatives is vastly larger than the number of positives, further tests are omitted if an input fails to pass a test. The tests are designed to be such that almost 100% of positives pass, and thus if an input does not pass a test, then the risk of the input being positive is very small.

The notion of acceptable misclassification risk is a basic concept in *sequential analysis* [Wal47, Sch92], which is a subtopic of statis-

tics dealing with costs of measurements. Sequential analysis has been applied to the design of medical experiments, e.g., [BBC94]. In the typical example, a patient may have to be classified as having a particular condition or not, and the number of tests should be limited due to discomfort or monetary costs.

The connections between classic sequential analysis and cascades have been appreciated only recently [SM05]. In [SM05] Sochman and Matas acknowledge that the quality of a classifier is determined by both the error costs and the time required for making predictions. In principle, the tree model from Section 2.1.3 allows for maximum acceptable misclassification risk if there is a training procedure ensuring that the term

$$MML[\epsilon(\pi) - \epsilon(\pi)\rho(\pi) + \rho(\pi)] \tag{2.22}$$

from step (2.21) is bounded from above. Such bounds seem difficult to guarantee because of the $\rho$ term. The possibility of overflow problems (with probability $\rho(\pi)$) is not addressed by Wald's classic sequential probability ratio test (SPRT), which forms the basis of the approach of Sochman and Matas. Wald's test seems to assume that the supply of base classifiers is unlimited in the sense that for each input there is always some sequence of tests such that the risk can be reduced to an acceptable level.

### 2.2.2 Trees

Non-degenerate trees enable better control over the conditional exclusion of unnecessary computations. This enhanced level of control is desirable when there are more than two classes. For instance, the path that an input image follows through a tree may reflect narrowing down the set of candidate labels (see [BG05] for related, more general discussion). While traditional axis-parallel decision trees [Qui93, Mit97] and more general perceptron trees [Utg89, MKS94, BM92, BM94a, BM94b, BFOS84] tend to use fairly simple features and are somewhat brittle and prone to overfitting, these characteristics reflect the algorithms employed and not any fundamental problems with modeling the computational process of classification as a tree. Recently, the overfitting problems of perceptron trees have been examined in the light of margin maximization theory [BCSTW00].

Computational efficiency issues have been considered explicitly in the context of decision trees. Geman and Jedynak [GJ01] discuss a penalty term for expected evaluation depth that is added to a more traditional term related to misclassification loss. Expected depth is, of course, not a completely appropriate criterion if different tests consume different amounts of time (resources).

The difference between cascades and proper trees may be somewhat unclear because these types of structures may be combined and mixed in many ways. For example, a cascade node might contain another cascade (or a tree) responsible for the predictions the node makes. Alternatively, a tree might contain subtrees that are degenerate (cascades). For simplicity, we defined the base classifier nodes as units that require constant time (resources) per input (Section 2.1.2).

Viola and Jones [JV03] address the problem of multi-view face detection by dividing the class of faces into pose-specific subclasses. In the first stage, there is a simple decision tree which determines the pose of an input, i.e., "the pose class". The decision tree then activates a pose-specific cascade that makes the final prediction. The empirical results suggest that (boosted) cascades may be made subtrees of more traditional (entropy-based) decision trees, as long as the basic features are suitable.

More generally, tree-like classifiers were studied widely in visual pattern recognition before cascades began to attract attention. Amit, Geman and Wilder [AGW97] utilize decision trees for handwritten digit recognition. The system learns the trees and the features together, implementing embedded feature extraction. Multiple trees are evaluated for each input, leading to increased accuracy at least partly due to random choices in creating the trees (see random forests, e.g., [Bre01] for discussion of overfitting decision trees). Huang, Gutta and Wechsler [HGW96] managed to do rudimentary frontal face detection with a single decision tree utilizing very simple features.

In addition to decision trees in which an input always takes an unique path down, there are other tree-like hierarchical classification systems. Basic examples include mixtures of experts [JJNH91], and hierarchical mixtures of experts [JJ94]. These mixtures are not necessarily well-suited for efficiency optimization, as they resemble weighting schemes more than delegation schemes. In other words,

different paths do not represent mutually exclusive alternative computations.

Hierarchical classification systems do not often follow a *class hierarchy*, e.g., the subtrees of a decision tree do not necessarily correspond to meaningful groupings of classes. To name some exceptions, there are the pose classes of Viola and Jones [JV03] and the *superclasses* in Chapter 4.

Some visual classification systems have successfully used a class hierarchy in directing inputs toward the most specific classifier applicable [RMN+98, SKB+99]. Some others [LS03, NS98] recognize the existence of perceptual similarity-based class hierarchies in visual multi-class problems, but are not focused on cost-efficient classification. Still others [SK04, LZ04] use lower-level hierarchies for basically two-class problems, e.g., training separate classifiers for different poses in face detection.

The use of meaningful class hierarchies is interesting. For example, a path through a tree could represent a sequence of predictions beginning with the most generic superclass labels (e.g., "an animal") and ending with the most specific labels (e.g., "Whiskers the cat"). If such a sequence contained mistakes toward the end, e.g., a wrong subtree was chosen after some point, the more generic labels could still be correct and useful.

## 2.3 Designing classifiers for nodes

### 2.3.1 General issues

A classifier $prog_f$ that is organized like a tree should use base classifier nodes that satisfy certain criteria. As stated earlier, the running time of each base classifier $prog_{f,k}$ should be constant, $\gamma_k$, for every input image $\mathbf{x}$. The other, more complex criteria are related to controlling the mistake losses. Recalling steps (2.11) – (2.18), we have the term $P(error \mid \mathbf{x} \in A_{\pi,k,c})$ that is associated with the upper bound on the mistake loss component. The set $A_{\pi,k,c}$ was defined as the subset of $B_k$, the set of inputs with which the $k$th base classifier is confident (for valid reasons or not). The tree $prog_f$ trusts confident base classifiers and the first activated and confident base classifier is responsible for producing the output of $prog_f$. Hence, if

the base classifiers use some ill-founded heuristic determining confidence, the end result may be that the error probability of the tree classifier, $P(error \mid \mathbf{x} \in A_{\pi,k,c})$, becomes high.

Ultimately, we judge confidence determination mechanisms by the empirical results, whether the mechanisms are rigorous or heuristic. A mechanism is useful if it leads to some desired $\epsilon(\pi)$ such that

$$P(error \mid \mathbf{x} \in A_{\pi,k,c}) \leq \epsilon(\pi)$$

and hence

$$P(Y = c \mid \mathbf{x} \in A_{\pi,k,c}) > 1 - \epsilon(\pi).$$

A confidence determination mechanism is immediately available for base classifiers that use explicit statistical modeling of class probabilities. We assume that a statistical classifier always predicts the class that appears the most probable based on a (possibly imperfect) model and the input. The $k$th statistical base classifier has probability estimates $\hat{P}(y \mid S_k(\mathbf{x}))$, where $S_k(\mathbf{x})$ denotes the set of features that are extracted from inputs $\mathbf{x}$. Alternatively, we denote it $\mathbf{S}_k(\mathbf{x})$ when it is appropriate to assume that we have a feature vector. The implied confidence determination mechanism has the form

$$\textbf{if } \hat{P}(y \mid S_k(\mathbf{x})) > 1 - \hat{\epsilon} \textbf{ then } \text{confident}, \tag{2.23}$$

where $\hat{\epsilon} \in [0, 1]$ may even be class-specific. Note that the relation between $\hat{\epsilon}$ and $\epsilon(\pi)$ is not trivial because the former is a threshold for a possibly incorrect implemented model while the latter is a bound on the true error. The bound on the true error is affected by the organization of the tree, e.g., $\epsilon(\pi)$ may be small because certain inputs that would exceed $1 - \hat{\epsilon}$ *do not* reach the base classifier.

For example, if we wanted to imitate Wald's classic SPRT procedure from sequential analysis [Wal47, SM05], we would have to build a degenerate tree (cascade) of pre-ordered classifiers $(f_1, f_2, \ldots, f_M)$ such that $S_1(\mathbf{x}) \subset S_2(\mathbf{x}) \subset \ldots \subset S_M(\mathbf{x})$. Wald's model would be incorrect in practice if the individual features were not conditionally independent given the class.

Otherwise, it is easy to see that Wald's ratio test may be expressed in the form (2.23), i.e., the class-specific $\hat{\epsilon}$ is a simple function of Wald's thresholds and the class priors. The tree (encoded in $prog_f$ and $\pi$) would simply output the prediction of the first

confident base classifier, which would be the first passing the ratio test.

The models responsible for the estimates usually require some similarity metrics applied to the features because the number of different feature combinations $S_k(\mathbf{x})$ is large and class probability estimates must be available for inputs not seen during training.

If a base classifier does not use explicit statistical modeling of probability distributions, finding suitable confidence determination mechanisms may be non-trivial. A non-statistical classifier may sometimes be extended so that the extensions produce class probability estimates that may then be thresholded as in (2.23). Sometimes the extensions for determining the level of confidence may be non-probabilistic, in which case the thresholding of probability estimates is replaced by some other decision rule.

In the later chapters, we examine non-statistical classifiers with various extensions for determining confidence. In the rest of this section, we explore some classifier architectures and extensions.

### 2.3.2 Basic support vector machines

Support vector machines (SVMs) are a well-established class of machine learning algorithms [CV95, Vap98]. They fall under the category of kernel-based methods [CST00, Her01, SS01]. Characteristic properties of SVMs are that they lack local minima, have a sparse solution, and are dimension-independent. Taken together all this makes SVMs an attractive approach to use in applications such as machine vision.

Basic SVMs are designed for two-class problems only and the input features $\mathbf{S}(\mathbf{x})$ must be vectors of reals with some fixed dimensionality $d'$ that may differ from the dimensionality of the images $\mathbf{x} \in \mathbb{R}^d$. The labels should be encoded such that $y_i \in \{+1, -1\}$.

For brevity, we denote the input features $\mathbf{z} \in \mathbb{R}^{d'}$, $\mathbf{z} = \mathbf{S}(\mathbf{x})$. Correspondingly, $\mathbf{Z}$ is the random vector $\mathbf{S}(\mathbf{X})$, i.e., the input $\mathbf{X}$ is random and the function $\mathbf{S}$ is known and deterministic. The *machine input space* is the space containing the vectors $\mathbf{z}$. Sometimes it is useful to let the SVM kernel perform the feature mapping completely, in which case $\mathbf{z}$ is the flattened pixel raster, $\mathbf{z} = \mathbf{x}$, and the machine input space is the original input space.

Given a training set of $N$ samples of the form $(\mathbf{z}_i, y_i)$ that are

assumed independent and identically distributed (i.i.d.), the task of the SVM learning algorithm is to formulate a *hypothesis* on the basis of the samples for classifying further instances from the machine input space. The hypotheses are linear separators, i.e., hyperplanes in some space. In other words, these hypotheses must be interpreted as class membership indicator functions without immediately apparent mechanisms for determining classification confidence.

The classic algorithm for learning linear separators is the *perceptron* [Ros58], which is guaranteed to converge in a finite number of iterations provided that the input samples are linearly separable [Nov62]. The perceptron outputs a linear function of $\mathbf{z}$,

$$f(\mathbf{z}) = \mathbf{w}^T\mathbf{z} + b, \tag{2.24}$$

where $\mathbf{w}$ is a weight vector determining the orientation of the plane and $b$ is a scalar determining the displacement of the plane from the origin of the machine input space. The hypothesis is the sign of (2.24). The same plane may be specified in several ways of which the above (2.24) is called the *primal form*.

The plane $\mathbf{w}$ is learned from the $N$ samples and may be written as a linear combination of them,

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{z}_i. \tag{2.25}$$

The number of iterations required to learn the hypothesis depends on the geometric *margin* of the training set, which is the maximum Euclidean distance of the samples from any hyperplane.

Rosenblatt's [Ros58] on-line, mistake-driven procedure for training a perceptron works by adding misclassified positive training samples to or subtracting misclassified negative ones from an initial zero weight vector. Hence, once a sample has been fixed, the vector $\boldsymbol{\alpha}$ can be thought of as an alternative encoding of the hypothesis. When (2.25) is substituted for $\mathbf{w}$ in (2.24) we get the *dual form* of $f$.

An alternative learning scheme projects the data through a fixed non-linear mapping $\boldsymbol{\phi}$ to a *machine feature space*, instead of operating on the machine input space. The mapping $\boldsymbol{\phi}$ allows the use of non-linear separating surfaces. Although the hypothesis is a plane in the machine feature space, it does not have to be a plane in the

machine input space or the original input space. The mapping also typically increases the dimensionality of the samples. The corresponding primal form is:

$$f(\mathbf{z}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{z}) + b. \tag{2.26}$$

Handling $\mathbf{w}$ in a very high-dimensional space may become inefficient as the number of required multiplications leads to high costs in time. Expressed in dual form, (2.26) becomes

$$f(\mathbf{z}) = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{\phi}(\mathbf{z}_i)^T \boldsymbol{\phi}(\mathbf{z}) + b. \tag{2.27}$$

Computing $\boldsymbol{\phi}(\mathbf{z}_i)^T \boldsymbol{\phi}(\mathbf{z})$ may often be made efficient by using suitable *kernel functions*. For our purposes, a kernel $K$ is a function such that

$$K(\mathbf{z}, \mathbf{z}') = \boldsymbol{\phi}(\mathbf{z})^T \boldsymbol{\phi}(\mathbf{z}'), \tag{2.28}$$

for all $\mathbf{z}, \mathbf{z}'$ in the machine input space. For the detailed requirements of kernel functions we refer the reader to the SVM literature [CST00, Her01, SS01, Vap98]. A kernel may be efficient in our terms, if it has a program that evaluates $\boldsymbol{\phi}(\mathbf{z})^T \boldsymbol{\phi}(\mathbf{z}')$ without having to evaluate $\boldsymbol{\phi}(\mathbf{z})^T$ and $\boldsymbol{\phi}(\mathbf{z}')$ separately (e.g., the polynomial kernel).

Vapnik and Chervonenkis' [Vap98] theory of learning bounds the generalization error of linear machines in terms of the margin of the hypothesis with respect to the samples. This result does not depend on the dimensionality of the machine feature space. By enforcing conditions from optimization theory, the dual representation of the hypothesis is sparse and, hence, may produce efficient classifiers if the kernel itself is efficient. It is, however, entirely possible that the primal representation is sometimes more efficient.

Taken all together, the basis of the *maximal margin classifier* is in the following result from optimization theory. Given i.i.d samples $\{(\mathbf{z}_i, y_i)\}_{i=1}^N$ that are linearly separable in the machine feature space implicitly defined by kernel $K$, suppose that the vector of

parameters $\boldsymbol{\alpha}^*$ solves the quadratic optimization problem

$$\max_{\boldsymbol{\alpha}} \left[ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j K(\mathbf{z}_i, \mathbf{z}_j) \right] \tag{2.29}$$

$$\text{with constraints} \begin{cases} \sum_{i=1}^{N} y_i \alpha_i = 0, \\ \forall i \in \{1, \ldots, N\} : \alpha_i \geq 0. \end{cases}$$

Based on (2.25), the optimal $\mathbf{w}$ is

$$\mathbf{w}^* = \sum_{i=1}^{N} \alpha_i^* y_i \boldsymbol{\phi}(\mathbf{z}_i). \tag{2.30}$$

Only for inputs $\mathbf{z}_i$ which lie closest to the hyperplane are the corresponding $\alpha_i^*$ non-zero. They are called *support vectors*. Let $\mathcal{SV}$ denote the set of support vectors in the training set. Take one support vector $\mathbf{z}^{(SV)} \in \mathcal{SV}$ such that the support vector belongs to the positive class (+1). The positive support vector $\mathbf{z}^{(SV)}$ satisfies

$$\mathbf{w}^{*T} \boldsymbol{\phi}(\mathbf{z}^{(SV)}) + b^* = 1, \tag{2.31}$$

where $b^*$ is the optimal $b$. When (2.30) is substituted for $\mathbf{w}^*$ in (2.31) we can use the kernel to solve $b^*$. The decision rule given by the sign of the function

$$f(\mathbf{z}) = \sum_{i=1}^{N} y_i \alpha_i^* K(\mathbf{z}_i, \mathbf{z}) + b^* \tag{2.32}$$

is then equivalent to the maximal margin hyperplane implicitly defined by the kernel $K$. The maximal margin hyperplane has geometric margin $(\sum_{i \in \mathcal{SV}} \alpha_i^*)^{-1/2}$. In terms of view-based classification, (2.32) defines a prototype-based classifier in which the support vectors are the selected prototypes and the kernel is the similarity measure on the inputs.

Technical properties of kernels ensure that the optimization problem is convex, which in turn means that the maximal margin optimization problem has a unique solution that can be found efficiently [BV04]. On the other hand, maximal margin classification requires the data to be linearly separable, which is not usually the case in the real world. Therefore, the strict requirement of linear separability has to be relaxed. The theory behind such machines has also been worked out [CST00, Her01, SS01, Vap98].

Finally, we note that while basic support vector machines are strictly for two-class problems, there are methods for combining several machines so that the resulting combination is fit for multi-class problems. Practical combination schemes may use either *one-vs-rest* or *one-vs-one* training with voting for combining predictions [DK05, HL02, HT98]. In addition, multi-class problems may be decomposed into multiple binary problems by using *error correcting output codes* (ECOC) [ASS00, DB95, PW72]. In principle, SVMs can also be formulated for multi-class problems directly without having to combine binary machines [CS01], and there are formulations suitable for class taxonomies as well [THJA04]. In practice, training SVMs on large datasets may be difficult (see [Joa98]). Because datasets involving multiple classes tend to be large, i.e., one should have a reasonable amount of data from each class, the training problem is especially relevant in the multi-class setting. Hence, decomposition schemes such as one-vs-one training have their advantages.

In the one-vs-rest scheme, there is one dedicated machine per class. The dedicated machine of a class is trained so that inputs from the class are labeled $+1$ while inputs from the other classes are labeled $-1$. The scheme implies some difficulties. First, the number of machines evaluated per test input equals the number of classes. Second, because soft-margin SVMs minimize the hinge loss as a surrogate for the 0/1 loss, class priors have an effect on the results (see Section 2.1.1 for discussion on losses). For example, if we had 20 classes each with the prior probability of 0.05, then *all* the machines would be biased against predicting $+1$. The *winner takes all* strategy is popular in overcoming the second problem: the combined machinery predicts the class $c$ the dedicated machine of which has the largest raw output $f_c(\mathbf{z})$ from (2.32).

In the one-vs-one scheme, there is one dedicated machine per class pair, and thus $M(M-1)/2$ machines overall. A dedicated machine is trained using inputs from its pair of classes only. If a machine dedicated to the class pair $(c_i, c_j)$ predicts $+1$ then class $c_i$ gets one additional vote. Otherwise class $c_j$ gets the vote. When all machines have predicted, the combined prediction is the class that gained the most votes. The one-vs-one scheme has the advantage that the machines are not inappropriately biased if the class priors are approximately equal. The computational requirements,

however, are even worse than with the one-vs-rest scheme. A 20-class problem would require 190 machines, which may be rather inefficient computationally.

### 2.3.3 Monotonic confidence models for hyperplane classifiers

The basic SVM is evaluated by taking the sign of (2.32), which means there are no confidence determination mechanisms immediately available. If we choose to get the confidence information through probabilistic modeling *and* the SVM hypothesis, then we may build a class probability model, $\hat{P}(Y = y \mid \mathit{Features}(\mathbf{x}))$, on top of the SVM hypothesis with *Features* being measurable from the SVM and possibly the data.

One possible probability model may be found by Platt's procedure [Pla00] of fitting a sigmoid to the raw SVM outputs $f(\mathbf{z})$. Platt's procedure produces the probability estimator

$$\hat{P}(Y = +1 \mid f(\mathbf{z})) = \frac{1}{1 + \exp(\mathcal{A}_{Platt} f(\mathbf{z}) + \mathcal{B}_{Platt})}, \tag{2.33}$$

where the SVM-specific constants $\mathcal{A}_{Platt}$ and $\mathcal{B}_{Platt}$ are estimated by a model trust minimization algorithm [Pla00]. For estimating the constants, the algorithm should use a portion of the training set that is kept separate from the portion that is used for training the underlying SVM (i.e., $f$). This recommendation is based on the empirical work that will be presented later in Chapter 5. It was found that without separation the classification results were inferior.

Explained another way, Platt's procedure creates a very simple *distinct* classifier on top of the SVM. This simple classifier is probabilistic and based on exactly one scalar feature: the raw SVM output describing how far $\mathbf{z}$ is from the hyperplane and which side it is on. Platt's model assumes a particular simple *monotonic* relationship between the class probabilities and $f(\mathbf{z})$. The function (2.33) is strictly monotonically increasing and continuous. Platt argues that there is a strong prior in favor of monotonicity in the case of raw SVM outputs [Pla00].

In the context of the classification trees or cascades that we have been discussing, it is possible to ask if it is necessary to have a complete model such as (2.33) available. Recalling (2.23) and using

Platt's model as an example, we simply need the ability to evaluate whether

$$\hat{P}(Y = +1 \mid f(\mathbf{z})) > 1 - \hat{\epsilon}$$

holds for two suitable values $\hat{\epsilon}$. The two values define two threshold probabilities: a lower threshold such that when $\hat{P}$ is below the threshold, then the model is confident that the class is $-1$, and a higher threshold above which the model is confident that the class is $+1$. Assuming that the probabilities are monotonic, two threshold probabilities suffice.

Given that two threshold probabilities are enough, we may then ask if these thresholds indeed have to be probabilities, $1-\hat{\epsilon} \in [0, 1]$. The answer is no because strictly monotone continuous functions are bijections, and thus there must be some unique value of $f$ for each threshold. There is at least one $\mathbf{z}_{\hat{\epsilon}}$ such that

$$\hat{P}(Y = +1 \mid f(\mathbf{z})) > 1 - \hat{\epsilon} \Leftrightarrow f(\mathbf{z}) > f(\mathbf{z}_{\hat{\epsilon}}). \tag{2.34}$$

Hence, if the class probabilities are monotonic and continuous in the range of $f$, there is an equivalent non-probabilistic confidence determination rule that is also simple in the sense of requiring just two inputs as parameters. Further, the non-probabilistic rule may be compatible with multiple monotonic probability models: these different models may produce the same probabilities around the important thresholds while having different shapes at the tails (over- or underestimating the probabilities of extreme events).

The non-probabilistic rule may be more convenient to use if the tree is built using heuristic search, e.g., we select threshold inputs iteratively from the training data, train the classifiers and the tree, finally stopping when an evaluation set of data indicates that the loss (2.6) is sufficiently small.

Using a model like (2.33) in multi-class problems would require some considerations, the nature of which depends on the combination scheme and how many hyperplanes are involved. For instance, Duan and Keerthi [DK05] show how Platt's rule may help in modeling class probabilities in the context of one-vs-one training through pairwise coupling (see also [HT98]).

### 2.3.4 Beyond monotonic and probabilistic models of confidence

Instead of assuming a continuous monotonic relationship between the raw output of a hyperplane classifier $f$ and class probabilities $P(y \mid f(\mathbf{z}))$, it is possible to explore more complex non-monotonic relationships. We discuss one possible approach to this kind of exploration. In the current chapter, we focus on the theoretical aspects of the approach. The approach will be examined empirically in the second half of Chapter 5. In that chapter, there is also a characterization of the practical circumstances in which it makes sense to use this particular approach. In what follows, we assume that the outputs (predictions) of individual hyperplane classifiers are initially encoded in the usual manner, i.e., the set *Classes* is $\{+1, -1\}$.

In the approach, hyperplane classifiers are given *post-processors* that are programs capable of transforming the outputs of the classifiers. The outputs of a classifier are transformed so that the classifier becomes able to indicate lack of confidence. The classifier indicates lack of confidence by using the special output of 0. If the classifier is confident, the positive class is indicated by a positive output and the negative class is indicated by a negative output. Although the positive and negative outputs are numbers, they are not necessarily $+1$ and $-1$. A post-processor essentially replaces the sign function normally applied to the raw output $f(\mathbf{z})$ of a hyperplane classifier $f$ such as (2.32).

Define $g : \mathbb{R}^d \to O$, where $O$ is an *output set* that contains zero, one positive real number, and one negative real number. Let $g$ describe one complete chain of processing from input pixels to post-processor output,

$$g(\mathbf{x}) = (r \circ f \circ \mathbf{S})(\mathbf{x}), \tag{2.35}$$

where $r$ is a post-processor, $f$ is a hyperplane classifier (not taking the sign), and $\mathbf{S}$ is a feature extractor.

As a special case, the above definition (2.35) allows $r$ to use Platt's estimator (2.33) internally. In that case the estimated probability of the event $Y = +1$ is compared to two suitable threshold values. The comparisons then determine which value from the output set $O$ is chosen by $r$. More precisely, suppose the threshold

values are the probabilities $0.5 - a_{low}$ and $0.5 + a_{high}$, where $a_{low}$ and $a_{high}$ are positive real-valued constants smaller than 0.5. The post-processor $r$ maps the probability interval $[0.5-a_{low}, 0.5+a_{high}]$ to the output value of zero in the output set $O$. The interval $]0.5 + a_{high}, 1]$ is mapped to the positive value in the output set. Likewise, the interval $[0, 0.5-a_{low}[$ is mapped to the negative value in the output set. Also note that the threshold values do not have to be probabilities. Based on (2.34), it is possible to use some equivalent threshold values that can be compared directly to the raw values $f(\mathbf{z})$.

Let us assume a fixed output set $O$ that has a subset that can be mapped to class labels without ambiguity. For example, let $O = \{-0.33, 0, +1.2\}$ and let the classes be *apples* and *oranges* so that a negative value indicates apples and a positive value indicates oranges. In the general case, the definition (2.35) allows *all* interesting models of confidence that take values of $f(\mathbf{z})$ as input and choose output from $O$. In the scope of this work, interesting models are computable models that work according to the i.i.d. assumption of the inputs, e.g., the output depends on current $f(\mathbf{z})$, but not on the previous one. Both probabilistic and non-probabilistic models are allowed. Likewise, the models can be either monotonic or non-monotonic.

The proposed approach is non-monotonic and non-probabilistic. Non-monotonicity is chosen because we abandon the assumption that values of $f(\mathbf{z})$ should be directly proportional to confidence. For example, abnormally high absolute values of $f(\mathbf{z})$ may be a characteristic of outliers, inputs dissimilar to all inputs in the training set. We use several hyperplane classifiers within a node of a tree. More precisely, a node has a *combiner* that takes the post-processed outputs of the chosen hyperplane classifiers as inputs. The combined hyperplane classifiers operate in parallel and their post-processed outputs are interpreted as *votes*. The combiner then produces output based on the votes. The post-processors are trained to optimize the outputs of the combiner.

Suppose that a node has $M_{combiner} > 0$ hyperplane classifiers that are to be combined, i.e., we have $f_k$ and $\mathbf{S}_k$ for each $k$ such that $1 \leq k \leq M_{combiner}$. Note that in this context $k$ does not index base classifier nodes, but more elementary hyperplane classifiers (we do not want to use secondary subscripts). The functions $f_k$ and $\mathbf{S}_k$

are fixed, but their inputs are random. The vector $\mathbf{Z} = \mathbf{S}_k(\mathbf{X})$ is a random vector. Obviously, each $\mathbf{S}_k$ is associated with a *distinct* random vector $\mathbf{Z}$, but plain $\mathbf{Z}$ without subscripts or superscripts is now abused to abbreviate notation. The variable $f_k(\mathbf{Z})$ is a continuous random variable. Classification confidence is established through the discretization of the values of $f_k$. Based on the ordered set of the $N$ observed values

$$V_k = (f_k(\mathbf{z}_1), f_k(\mathbf{z}_2), \ldots, f_k(\mathbf{z}_N)) \tag{2.36}$$

from a training set, we generate $I + 2$ non-overlapping value intervals, or bins, the union of which equals $]-\infty, \infty[$. We require that $I \geq 1$ and $N \geq 2I$. A suitable number of intervals can be found experimentally, e.g., using cross-validation. In Chapter 5, we use a modest number of intervals (42, i.e., $I = 40$). The theoretical analysis, which will be presented in the next section, suggests that the larger the number $I$ is, the greater the risk of overfitting becomes. Perhaps surprisingly, the same analysis suggests that increasing $M_{combiner}$ does not necessarily increase the risk of overfitting.

Suppose that the ordered set $V_k$ from (2.36) has been sorted into ascending order and let $v_{k,j}$ denote the $j$th element of $V_k$. Hence, $\min\{f_k(\mathbf{z}_i)\}_{i=1}^N = f_k(\mathbf{z}_1) = v_{k,1}$. The first interval is $int_{k,1} = ]-\infty, v_{k,1}[$. The second interval is $int_{k,2} = [v_{k,1}, v_{k,\lfloor \frac{N}{I} \rfloor}[$. The $i$th $(2 < i < I + 2)$ interval is

$$int_{k,i} = [v_{k,\lfloor \frac{N}{I}(i-2) \rfloor}, v_{k,\lfloor \frac{N}{I}(i-1) \rfloor}[. \tag{2.37}$$

Finally, the last interval is $int_{k,I+2} = [v_{k,N}, \infty[$. Intuitively, each of the middle intervals contains $100/I\%$ of the observed values.

The intervals $int_{k,i}$ will be marked as either *standard* or *abstain* intervals as decided by the optimization procedure of the combiner. Inputs falling into abstain intervals map to the value 0 in the output set $O$. Inputs falling into standard intervals map to the nonzero values in $O$. More formally,

$$r_k(v) = \begin{cases} 0 & \text{if } v \in int_{k,i} \wedge \text{ abstain}(k,i) = 1 \\ signmap_k(v) & \text{if } v \in int_{k,i} \wedge \text{ abstain}(k,i) = 0, \end{cases} \tag{2.38}$$

where $signmap_k$ maps values $v$ to the nonzero values in $O$.

The optimization procedure responsible for choosing the *abstain* intervals of the post-processor $r_k$ requires an optimization criterion. We use the criterion of *maximal voting margins*. Here, voting refers to us interpreting the post-processed outputs of the combined classifiers as votes. Margin maximization in general is associated with certain generic upper bounds on classifier error [BM02] such that the larger the margin, the smaller the bound (assuming empirical error does not increase). The concept of margins has been useful in explaining the generalization ability of boosting algorithms [SFBL97].

The empirical voting margin of a combiner is defined as

$$\text{vmargin}(\{(\mathbf{x}_i, y_i)\}_{i=1}^N) = \sum_{i=1}^{N} y_i \sum_{k=1}^{M_{combiner}} g_k(\mathbf{x}_i), \tag{2.39}$$

where $N$ is the number of training inputs and $M_{combiner}$ is the number of combined hyperplane classifiers. The approach requires that the empirical voting margin (2.39) is maximized.

Abbreviating

$$f_{combiner}(\mathbf{x}) = \sum_{k=1}^{M_{combiner}} g_k(\mathbf{x}), \tag{2.40}$$

the output of the combiner is

$$g_{combiner}(\mathbf{x}) = \begin{cases} 0 & \text{if } f_{combiner}(\mathbf{x}) = 0 \\ +1 & \text{if } f_{combiner}(\mathbf{x}) > 0 \\ -1 & \text{if } f_{combiner}(\mathbf{x}) < 0. \end{cases} \tag{2.41}$$

The above Equation (2.41) allows the combiner to declare lack of confidence when $g_{combiner}(\mathbf{x}) = 0$. This may happen if the votes of the combined $g_k$ cancel each other out. It is, of course, also possible to define two thresholds so that small absolute values of $f_{combiner}(\mathbf{x})$ result in $g_{combiner}(\mathbf{x}) = 0$. Moreover, if the combiner is in a node from which delegation is not possible, then the case $f_{combiner}(\mathbf{x}) = 0$ may be mapped to $+1$ or $-1$.

In Chapter 5, we use $M_{combiner} = 15$, but it is possible to use $M_{combiner} = 1$ as well. Using a single hyperplane classifier may make sense when margins are increased through abstains, e.g., when high absolute values of $f_k(\mathbf{z})$ indicate outliers whose true labels are random.

If the classification problem is not binary, then it is possible to use multiple combiners. For example, combiners can themselves be combined using the one-vs-one training scheme. Each combiner is dedicated to some pair of classes and obviously the combined classifiers of the combiner have to be dedicated to the same pair. Given an input, each combiner then casts a vote for one of two classes or simply abstains from voting ($g_{combiner}(\mathbf{x}) = 0$).

### 2.3.5 A simple $0/1$ loss bound for a voting margin machine

We derive a simple loss bound that illustrates how the margins of $f_{combiner}$ are related to the true accuracy of $g_{combiner}$. In the derivation we need certain mathematical machinery and concepts. The concept of *Rademacher complexity* is necessary.

The Rademacher complexity of a family of functions $F$ w.r.t. $N$ independently drawn samples is defined as (see [BM02])

$$R_N(F) = E\left[\sup_{f\in F} |\frac{2}{N}\sum_{i=1}^{N}\sigma_i f(\mathbf{z}_i)|\right], \tag{2.42}$$

where the expectation $E$ is taken over the $N$ samples $\{\mathbf{z}_i\}_{i=1}^N$ and the $N$ independent uniformly distributed $\{+1,-1\}$-valued random variables $\{\sigma_i\}_{i=1}^N$. Intuitively, the Rademacher complexity is related to how well the family $F$ can adapt to random label noise.

To proceed, note that each post-processing (abstain-capable) component machine $g_k$ of $f_{combiner}$ in Equations (2.40) and (2.41) can be expressed in a different form. We write

$$\begin{aligned} g_k(\mathbf{x}) \quad = \quad & \frac{1}{2}\sum_{i=1}^{\Lambda_k}(\eta_{i,k}\text{sign}(\mathbf{w}_k^T\mathbf{S}_k(\mathbf{x}) + b_k + \kappa_{i,k}) + \\ & \nu_{i,k}\text{sign}(\mathbf{w}_k^T\mathbf{S}_k(\mathbf{x}) + b_k + \iota_{i,k})), \end{aligned} \tag{2.43}$$

where $i$ indexes the $\Lambda_k$ *non-abstaining* intervals, $(\mathbf{w}_k, b_k)$ is the underlying basic SVM $f_k$ (i.e., $\mathbf{z} = \mathbf{S}_k(\mathbf{x})$) and the parameters $\eta_{i,k}, \nu_{i,k} \in \{+1,-1\}$ and $\kappa_{i,k}, \iota_{i,k} \in \mathbb{R}$ are artifacts of the construction. The parameters are set so that each non-abstaining interval $i$ is covered using the basic SVM hyperplane twice: separate instances of the plane of $\mathbf{w}_k$ are placed to the opposite ends of the interval using parameters $\kappa_{i,k}$ and $\iota_{i,k}$ to control the placement. The

orientations of the instances are then controlled by $\eta_{i,k}$ and $\nu_{i,k}$. If the interval is for positive predictions, then the instances are oriented toward the center of the interval. For negative predictions, the instances are oriented away from the center. The net result is that the sum of each pair predicts $+2$ (positive intervals) or $-2$ (negative intervals) for each input within the interval. Outside the interval, the predictions of each pair sum to zero. By construction, the intervals do not overlap and the multiplier $1/2$ normalizes the final output to $\{+1, -1\}$. In other words, this analysis assumes the very basic output set $O = \{+1, -1, 0\}$.

Next, note that $\eta_{i,k}\text{sign}(\mathbf{w}_k^T\mathbf{z} + b_k + \kappa_{i,k})$ is a support vector machine regardless of the extra parameters. Denote the family of support vector machines by $F_k$ and the family of post-processing (abstain-capable) support vector machines of Equation (2.43) by $G_k$. Suppose that $\Lambda_k$ is maximal, i.e., all available intervals are non-abstaining intervals. Observing that the Rademacher complexity of $G_k$ cannot be greater than the complexity of the superset of $G_k$ that allows the $2\Lambda_k$ "machines" (i.e., artifacts of the analysis) within each $g_k$ to have independent weight vectors, we apply Theorem 12 from [BM02] to write

$$R_N(G_k) \leq R_N(\frac{1}{2}\sum_{i=1}^{\Lambda_k}(F_k + F_k)) = \Lambda_k R_N(F_k). \tag{2.44}$$

We can now bound the Rademacher complexity of

$$f_{combiner}(\mathbf{x}) = \sum_{k=1}^{M_{combiner}} g_k(\mathbf{x})$$

from Equation (2.40). Earlier, we assumed the worst case that $\Lambda_k$ is maximal, i.e., all available intervals are non-abstaining intervals. By construction, all our component machines have the same number of available intervals $(I + 2)$. Hence, we can abbreviate $\Lambda_k = \Lambda$. We apply Theorem 12 from [BM02] and Equation (2.44) to write

$$\begin{aligned} R_N(\frac{1}{M_{combiner}}\sum_{k=1}^{M_{combiner}} G_k) &\leq \frac{1}{M_{combiner}}\sum_{k=1}^{M_{combiner}} R_N(G_k) \\ &\leq \frac{\Lambda}{M_{combiner}}\sum_{k=1}^{M_{combiner}} R_N(F_k). \end{aligned} \tag{2.45}$$

Above, the complexity of the combiner is bounded by the average complexity of the component SVMs times the number of the discrete intervals available per machine. The complexity $R_N(F)$ of SVMs is a known quantity. The bound is loose because the derivation assumes many more degrees of freedom than what the post-processors actually have. On the other hand, the actual degrees of freedom are tied to the number of intervals available. Averaging implies that the overall complexity bound does not necessarily increase as more component machines are included.

Denote the true expected mistake-based risk, e.g., the 0/1 loss, of $f_{combiner}(\mathbf{x})$ by $\mathcal{L}_{combiner}$, and define the empirically measured 0/1 margin error as

$$\tilde{\mathcal{L}}^{\theta}_{combiner} = \frac{1}{N}\sum_{i=1}^{N} Ind(y_i f_{combiner}(\mathbf{x}_i) \leq \theta), \tag{2.46}$$

where $Ind$ is the indicator function and $\theta > 0$ is the required classification margin. Based on Theorem 3 from [MR03] and the upper bound of (2.45), every $f_{combiner}$ satisfies with probability $1-\delta$ that

$$\mathcal{L}_{combiner} \leq \tilde{\mathcal{L}}^{\theta}_{combiner} + \frac{4\Lambda}{\theta M_{combiner}} \sum_{k=1}^{M_{combiner}} R_N(F_k) + \sqrt{\frac{\log(2/\delta)}{2N}}. \tag{2.47}$$

If $y_i f_{combiner}(\mathbf{x}_i)$ tends to be large when $f_{combiner}$ is correct, then it may be possible to increase $\theta$ without increasing the empirical error (2.46) at all. Large $\theta$ is desirable because it makes the complexity term smaller. This is why the approach requires that the empirical voting margin of $f_{combiner}$ is maximized over the training set. Although the bound is loose, the possible trade-offs should be considered in practical applications. For example, using too many intervals probably results in large $\mathcal{L}_{combiner}$ even if the measured $\tilde{\mathcal{L}}^{\theta}_{combiner}$ is small. In other words, overfitting occurs.

## 2.4 Summary

In this chapter, the framework was presented and examined from a theoretical point of view. The framework has two essential characteristics. First, there is the tree-like organization of base classifier nodes that handles the conditional exclusion of unnecessary

computations on a per input basis. Second, there are rules for determining how inputs and responsibility are delegated between the nodes. The rules require that the nodes can assess their classification confidence.

The chapter began with a discussion of typical loss functions in classifier learning. Within the proper context, the discussion then proceeded to efficiency-sensitive losses. Through efficiency-sensitive losses, resource (time) consumption was made explicit in optimization criteria and subsequent analysis. Without explicit resource consumption, it is hard to analyze the behavior of efficient classifiers and some interesting things just cannot be seen properly. For example, organization resembling subsumption makes sense in certain circumstances if resource consumption is explicit. If resource consumption is not explicit, subsumption makes no sense on the formal level. An additive loss model was chosen, i.e., mistake losses and resource losses are added together (using desired coefficients). The chosen model allows straightforward analysis and is also quite intuitive. For example, one can specify that one mistake is worth two seconds of time. In discussing related research, it was noted that decision tree evaluation depth is not a good measure of resource consumption. Practical nodes are not equal in resource consumption, e.g., some image features may be trivial while others require extensive computation.

The tree-like model of classification was analyzed with respect to base classifier zones of confidence, overflow probabilities and bounds on base classifier errors. It was shown how the above three are related and how changing one affects the others. It was explained that so-called subsumption arrangements are useful in specific circumstances. In the subsequent survey of related research, it was noted that hard system-level error bounds, i.e., the maximum acceptable misclassification risk of sequential analysis, are difficult because of overflow problems. The SPRT of sequential analysis does not consider (non-zero) overflow probabilities. To summarize, the framework is viewed as a tool for examining resource versus mistake trade-offs instead of hard constraints.

The discussion proceeded to the topic of base classifier design, which was closely related to the topic of confidence models. Support vector machines (SVMs) were overviewed because they are practical large-margin classifiers and because they are used in the later

chapters. The classification margins of inputs appear to be related to classification confidence. After two-class SVMs, SVM combination methods were discussed. These methods allow multi-class classification. For reasons of efficiency, there is a tendency towards using variants of one-vs-rest training in the later chapters.

We examined monotonic and non-monotonic confidence models for large-margin classifiers (e.g., SVMs). It was explained how monotonic models (e.g., Platt) reduce to two thresholds for the purpose of making decisions. The thresholds may be represented using the $f$-values of suitable training inputs. The point was that a suitable pair of thresholds is always compatible with multiple probability models and that the choice between those models does not matter. More intuitively, nothing is gained by identifying the precisely correct probability model. The discussion then proceeded to non-monotonic modeling. One particular approach was contributed. The approach allows the combination of multiple classifiers. It was based on the use of voting margins and intervals of $f$-values. In the subsequent analysis, it seemed that increasing the number of intervals increases the risk of overfitting, but increasing the number of combined classifiers does not necessarily increase the risk. Hence, combining classifiers using the approach is worth an empirical study (to be presented in Chapter 5).

There are several important questions that were not addressed in this chapter. It was not discussed what kind of image features are required in the base classifier nodes. This is a practical question and the requirements vary from node to node. For example, the requirements of the root node are addressed in Chapter 3. Furthermore, it was not discussed how the organization of the tree is found. It was simply supposed that some organization exists. This question is addressed in Chapter 4. Moreover, there is the related question of the problem-specificity of organization, i.e., does each classification problem require a customized organization of the nodes. With ordinary decision trees, this is required. In Chapter 5 it is shown how problem-specific organizations may be avoided.

CHAPTER 3

# Finding the root of the problem

This chapter begins the empirical part of the dissertation. In the previous chapter the delegation framework was presented as a theoretical construct without much regard to practice. In practice there are design choices that have measurable consequences affecting delegation probabilities, node accuracies, and efficiency. The most open-ended choices are related to the feature spaces, i.e., what kind of features should be used in each node.

In this chapter we focus on the question of what kind of features can be used in the root node. The root node is special in that wrong choices may lead to efficiency bottlenecks that cannot be removed by changing or tuning *any* kind of delegation rules. In contrast, efficiency bottlenecks caused by other nodes can be affected by delegation rules, which is the intended purpose of these rules in the first place.

In this chapter, a *hypothesis* is contributed according to which certain kind of *global image features* enable the construction of root nodes that are likely to avoid delegation-rule-independent efficiency bottlenecks that would prevent rapid classification. For brevity, we will refer to this hypothesis as the *root feature selection hypothesis*. Alternatively, the hypothesis may be seen as a design heuristic that is specified as a hypothesis to encourage falsification attempts.

## 3.1 The root feature selection hypothesis

Before the root feature selection hypothesis can be described adequately, the nature of efficiency bottlenecks must be explained. For this purpose, the generic model and terminology from the previous chapter will be used.

### 3.1.1 Efficiency bottlenecks

Certain design choices may become obstacles to efficient classification. By obstacles we mean computational bottlenecks that prevent the use of the framework for the purpose of making desired trade-offs between accuracy and speed, as characterized by the mistake loss ($ML$) and resource loss ($RL$) in the minimization problem (2.6).

The obstacles may be expressed as lower bounds on the solution of the minimization problem (2.6),

$$\begin{aligned} & \min_{prog_f \in \mathcal{P}} E[ML(f(\mathbf{x}), y)] + LB(\mathcal{P}) \\ \leq \quad & \min_{prog_f \in \mathcal{P}} E[ML(f(\mathbf{x}), y) + RL(\mathbf{x}, prog_f)], \end{aligned} \tag{3.1}$$

where the parameter $\mathcal{P}$ of the resource loss lower bound $LB$ means that $LB$ depends on the set of programs considered. The set has to be severely constrained by design choices because otherwise the search for the minimum becomes impractical.

A lower bound $LB(\mathcal{P})$ can be simple to interpret. It may be assumed that the weighting between $ML$ and $RL$ is handled by a constant multiplier *within* the term $ML$. The term $RL(\mathbf{x}, prog_f)$ can be directly proportional to the number of seconds it takes to run the program $prog_f$ given $\mathbf{x}$ on a benchmark computer. A particular $LB(\mathcal{P})$ can then be interpreted as (proportional to) the number of seconds that *all* programs in the *constrained set* $\mathcal{P}$ require.

For example, the instance-independent $RL$ of the root node is a lower bound on $E[RL(\mathbf{x}, prog_f)]$ of all trees which have, or are grown from that root. The $RL$ of the root is also the largest *delegation-rule-independent* lower bound on $E[RL(\mathbf{x}, prog_f)]$ of trees with that root. A bound of this sort may be an obstacle to efficient classification such that no tuning of the delegation rules can enable desired trade-offs between speed and accuracy.

Here, we care about bounds $LB$ that are caused by root nodes and are rule-independent. Ensuring that there are no large root bounds should be a precondition for examining practical delegation rules (in the next chapter). Inserting a state-of-the-art classification algorithm in the root is almost certainly the wrong choice. If the algorithm is accurate but too slow, inserting it in the root guarantees that the tree of nodes is no faster. If the algorithm is accurate and fast enough, then the current framework is unnecessary.

Instead of imitating state-of-the-art classifiers, root nodes can use naive features and modeling. Non-trivial segmentation and large collections of prototypes may be too expensive.

### 3.1.2 Statement of the root feature selection hypothesis

Because the hypothesis attempts to make claims of efficiency that are relevant to practice, the claims need to be connected to a certain level of hardware. As the empirical reference point, we take the typical general purpose computer circa 2006.

**The root feature selection hypothesis:** Assume that the input images belong to distinct non-overlapping classes of everyday scenes and objects, each class considered belongs to a distinct *superordinate category*, and the input images are downsampled (using low-pass filtering) so that they are represented in a very low resolution. Finally, assume that a typical human observer can solve the classification problem accurately when shown the *downsampled* images. For a significant number of problems satisfying the assumptions, there exists (can be found) a root node program that satisfies the following three claims.

1. The root node operates in (close to) real time for standard video frame rates, i.e., it can make predictions multiple times per second.

2. The root node uses only *global features* that are statistics of local feature responses over the low-resolution images such that spatial relations between local features are discarded.

3. The root node discriminates between the classes above chance level.

We will now examine the hypothesis in detail, e.g., why do we use a human observer as a reference and what is accurate human performance, what is very low resolution, what are superordinate categories, what do the claims mean and imply, how does prior research justify the hypothesis, and how could we approach falsification.

First note that we use an average human observer as a reference. The point is *not* to do cognitive science in a computer science dissertation. We have to ensure that the classification task is fair. If the classification task was arbitrary, then there would be no reason to assume that the images convey any information at all about the class labels. The human observer is used as a tool for checking that claims are not made for problems that are unsolvable in low resolution.

We say that a human observer is accurate if the observer can make predictions with at least 95% accuracy. This requirement is quite permissive for many tasks. For example, according to Renninger and Malik [RM04], human beings can identify basic level categories of everyday scenes with 90% to 95% accuracy given just 70 milliseconds per image. Renninger and Malik note that superordinate categories can be identified even quicker.

For determining if the resolution can be considered very low, the limit of $150 \times 150$ pixels seems roughly appropriate. This is somewhat ad hoc, but the magnitude is suitable considering both computing power and the intuitive notion of very low resolution. The threshold of $150 \times 150$ does not constrain the aspect ratio of the images. For example, the resolutions of $64 \times 48$ and $200 \times 100$ are acceptable because the total number of pixels is smaller than $150 \times 150$.

The concept of *superordinate categories* is borrowed from prototype theory and vision science (see [LS03] for related discussion). The concept is best described by comparison to *basic level categories*. Basic level categories are classes whose labels have intermediate generality in categorical hierarchies and represent the level at which most of our knowledge is organized. Note that if our knowledge is organized around basic level categories, then we may expect that in many supervised learning tasks, the target classes have the generality of basic level categories.

According to Rosch [Ros78] human subjects can easily associate a single visual representation (shape) with a single basic category, while superordinate categories have no such cognitive visual representations. For example, the labels "beach" and "dog" are basic, while "outdoors environment" and "animal" are superordinate. Lower level *subordinate categories* are more specific, and possibly require attention to subtle detail, e.g., "Mykonos Island Beach", "Bayrischer Gebirgsschweisshund". In the hypothesis, the intent is to pay attention to differences between classification problems and avoid making claims about problems the solution of which requires attention to subtle details. Note that not all subtle details can be associated with high-frequency content of images. For example, different dog breeds may have different ratios of body length to height, and this may be clearly visible in images from which the high-frequency content has been removed.

When we say that each class belongs to a distinct superordinate category, we mean that a class either *is* a superordinate-level category, *or* is a subcategory of such. In the latter case, no other class may be a subcategory of the same superordinate category.

In the scope of this chapter, it is suggested that the target classes should satisfy the following heuristic criterion for the lack of class groups: *if* there is a pair of classes such that the classes have clear similarities in terms of shape and there is some (other) pair of classes that lacks these similarities, *then* there are class groups. *If* there are no apparent class groups, *then* the classes belong to distinct superordinate categories. The somewhat imprecise nature of this heuristic is apparent. A more disciplined method for the extraction of human-perceived categorical hierarchies (hierarchical groupings) will be suggested in the next chapter. The use of human-perceived hierarchies and groupings is not in itself a problem. Using these, we get to draw on existing research investigating what kind of features humans use for rapid recognition.

Having examined the assumptions, we can now examine what was claimed. There were three claims, and it was postulated that there is a significant number of classification problems for which there is a root node such that the claims hold given the assumptions. For reasonable falsification, a representative set of interesting problems should be chosen, the criteria for interestingness should be explained, and large failure rates should be considered decisive

evidence against the hypothesis. For example, failure rates larger than 80% can be considered decisive. The experimental results in this dissertation satisfy the claims when the assumptions are satisfied, but it is not argued that the set of problems is large enough.

Regarding falsification, it is suggested that the assertion that a suitable root exists is interpreted to mean that a competent practitioner can design it without much effort, e.g., in a week or so. Existence disproofs are hard, and therefore it is suggested that persistent lack of positive evidence is, for practical reasons, interpreted as evidence against the hypothesis. It is not interesting to consider toy domains in which exhaustive search allows conclusive disproofs.

In the first claim, it was stated that the root node must be able to make several predictions each second, i.e., the root must operate at least at 2 Hertz. As stated earlier, a typical general purpose computer circa 2006 is used as the empirical reference point in the measurements that are reported in this dissertation. The first claim is connected to the theory (Section 3.1.1) in the following way: when $RL$ in (3.1) is a linear function of the seconds used by the program, then it is trivial to express the *equivalent* optimization problem where $RL$ *equals* the seconds used by the program. In other words, $RL$ bounds become tangible quantities and we can say which root-induced $RL$ bounds are small enough for a specific application and which are not. The hypothesis gives a chance that root-induced $RL$ bounds do not prevent the desired trade-offs if the root uses the suggested kind of low-resolution global features, the application requires at most 2 Hertz, resource losses are linear in seconds used, and there is evidence that there is class-related information in the low-resolution images.

The second claim states that the root uses global features that are statistics of local feature responses over the low-resolution images, and that the global features ignore spatial relations between local features. The claim can be justified *a priori* on the basis of vision science research that shows the proposed kind of features are sufficient for rapid recognition of everyday scenes by humans, especially when superordinate categories are involved.

It is known that humans are very quick at understanding and identifying scenes, even after minimal exposure on the order of 20 milliseconds [TFM96]. Human accuracy at identifying scenes also improves considerably when the exposure times are increased. So, it

appears that human observers do not always have the time to utilize their resources fully, but classification is still possible at a reduced accuracy. The question of which kind of feature representations suffice to explain this fast performance in humans has attracted interest.

Oliva and Torralba [OT06] define the *gist* of a scene as the amount of perceptual and semantic information that *human* observers can comprehend at a "glance", which is defined as about $200ms$ exposure to an image. According to Oliva and Torralba, and Rensink [Ren00], the gist usually includes the semantic label of a scene, i.e., the class label of an image in our terms.

Oliva and Torralba explore formal (computational) models that could explain the efficiency of this quick comprehension given its accuracy and extent. In particular, they ask what kind of *representation* would be sufficient for explaining the *efficiency*. Their conclusion is that it is *sufficient* to use low-dimensional vectors of global features, which themselves are summations of local feature values. The local features are weighted combinations of oriented filters, i.e., Gabor-like filters similar to those thought to exist in the V1 of the primate (including human) cortex. The weights are derived from principal components analysis [Hay99]. In particular, the global features are noticeably coarse and "imperfect" to ensure efficiency. Taking into account the earlier work of Torralba and Oliva [TO03], the overall result seems to be that at least *superordinate* categories of scenes can be discriminated using coarse global statistics of local oriented features.

Although discrimination on the basic category level is also (often) possible using oriented feature approaches of the above sort, one can note two things in [TO03]. First, on the superordinate category level, the feature signatures of Torralba and Oliva seem sufficient for discrimination. Second, when the feature signatures seem sufficient for discriminating between two basic categories, it appears that the basic categories belong to different superordinate categories, and the basic category signatures resemble their respective superordinate signatures. In other words, when the basic signatures seem clearly different, the explanation may well be on the superordinate level. For this reason, and after a scrutiny of the classes involved in our experiments, we tie the claims (especially the second) of our hypothesis to the requirement of the classes be-

ing associated with distinct superordinate categories.

For further justification of the connection between coarse global image features and the requirement that the classes belong to distinct superordinate categories, we examine the work of Renninger and Malik [RM04]. Renninger and Malik conduct human experiments in which the subjects are asked to identify scenes within $70ms$ exposure. There are ten basic categories of scenes, namely "beach", "mountain", "forest", "city", "farm", "street", "bathroom", "bedroom", "kitchen", and "living room". The basic categories can be placed into three superordinate categories: "natural/outdoor", "man-made/outdoor", and "man-made/indoor". Renninger and Malik observe that the subjects can get the gist with one fixation, accuracy is always above chance, and improves with exposure duration. They build a computational texture model using V1-like (Gabor-like) features and use histograms of these features to classify images. They then observe that the computational model leads to similar identifications and confusions that the human subjects make with limited processing time. More precisely, human performance is similar to their model at 37 milliseconds (of human exposure), but the subjects outperform the model when they are given more than $37ms$.

Importantly, *both* humans and the model can identify the superordinate categories before the basic-level categories are identified. Basic-level categories *within the same superordinate category* are confused by both humans and the model. For example, cities are heavily confused with streets (both are outdoors city scenes) and farms are confused with beaches (both are largely natural scenes).

To conclude the review of the claims, we can note some differences between the assumptions of the hypothesis and what was assumed in the vision science literature that was surveyed. First, in the hypothesis we do not distinguish between scene recognition and object recognition. In the literature that was surveyed, this distinction is made. It can be argued that the distinction between scene images and images where a single object dominates is a bit fuzzy. For example, Torralba and Oliva [TO03] consider portraits of people and images of large buildings as scenes. Hence, we do not make sharp distinctions between scenes and objects.

Second, objects in the real world are not independent of their context. For example, if a root node determines that an image be-

longs to the superordinate category of animals, and delegates to a specialist classifier capable of segmenting animals and discriminating between different species, then it hardly matters if the superordinate category was recognized because an animal was perceived directly or because a faintly animal-like blob was perceived in a strongly predictive context, e.g., on a farm-like background. On the other hand, the next chapter contains empirical evidence that shows that a texture-based root *does not* need contextual features to succeed in object recognition at the superordinate level.

## 3.2 Overview of the experiments

Here, we present an overview of our early experiments that are related to the root feature selection hypothesis. Related experiments are also found in the next chapter. These experiments predate the current formalization of the hypothesis. In general, so-called *post-hoc* hypotheses are allowed and often necessary in science, but their nature should be acknowledged.

For the main experiment of this chapter, a mobile robot was used to collect grayscale images that were categorized into four classes: *doorways*, *signs*, *people* (close-up views), and *miscellaneous* indoor scenes. The overall idea of the collection procedure was that the relevant objects were perceived in their correct contexts, viewed from reasonable viewpoints, and at appropriate scales. As by the hypothesis, the four classes are human-recognizable in low resolution and seem to belong to distinct superordinate categories.

We pay special attention to the effects of sampling bias. When a root node is trained with a limited set of data and the classes are rather broad, i.e., each class contains a large number of different object instances, different deformations, and varying lighting conditions, then the training samples may not be properly representative of the classes. We examine what effects this kind of sampling bias has on the classification results. In practical applications, it would be useful if root nodes could tolerate this bias. Then it could be possible to add new (sub)classes without having to retrain the root completely.

After the main experiment, additional experiments are examined. We examine the recognition of various characters and symbols

that may be printed on signs, thus becoming subclasses of signs. These experiments are based on a *specialized* segmentation method that can work within the broad class of signs *only*. The results with the characters and symbols are then contrasted with results from the recognition of more complex objects that are visible to the same segmentation method. Contrasting the results allows us to predict if specialist classifiers are able to correct potential delegation errors without having to pay the full resource loss.

In the next chapter, we will see more advanced experiments involving confidence assessment and delegation mechanisms. There, the setting allows investigating the hypothesis again, although the focus of that chapter is elsewhere. The interesting difference is that object contexts (image backgrounds) are made statistically independent of the objects.

## 3.3 The main experiment

### 3.3.1 Extracting global features

The lowest level of the recognition process is based on a set of *Gabor filters*, which are designed to extract useful structural information from the views. The basic principles that we use are close to those in [TS01]. Gabor filters are also known to be useful in texture segmentation [WHD96]. We take advantage of the fact that a filter can be tuned to respond to a specific texture-like property of an image. The underlying assumption of our approach is that indoor scenes can often be recognized by considering the relative quantities of different textures in an image. If the scenes contain difficult objects, the objects tend to contain non-homogeneous subregions within the object boundaries, and such regions might have distinctive combinations of textures. In the latter case, if an object occupies a large portion of the scene, it may be possible to detect the presence of an object without having to search for it within the scene (e.g., search by using a sliding window, the contents of which are classified).

After the initial filtering, we calculate certain statistics of the filter responses and then classify the derived feature vectors with a set of SVMs (see Section 2.3.2). SVMs are particularly suitable for this task mainly because of two reasons. First, it is bothersome to

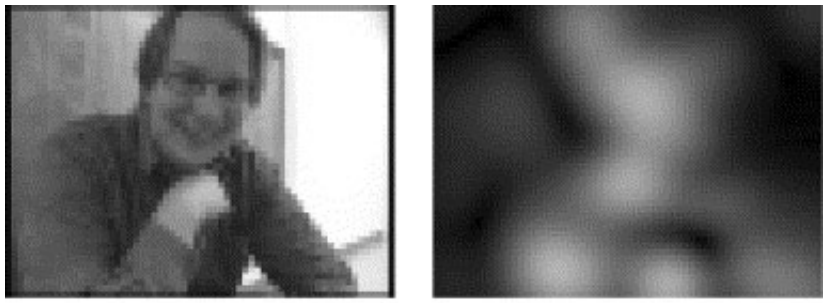



Figure 3.1: (a) A person and (b) the corresponding filter response.

add class labels to large sets of data and SVMs are known to cope well with small training sets. Second, we can use high-dimensional feature vectors without having to compute and store them explicitly.

In the SVMs, we use a second degree polynomial kernel to derive the final image features that are pairwise products of the input feature components. Consider the following toy example: we have several images of a forest clearing where there is a large pond. Imagine that we have a marker (a Gabor filter) that attaches to the particular texture of the pond. We also have a corresponding marker for the texture of the trees. Measuring the amount of both markers we have some information about the images. Measuring the products of the marker amounts gives us additional information related to the correlation of the amounts. In realistic applications, we could have dozens of interesting textures and their relationships. In that case, it becomes relevant to consider the representations of the product features as there are a lot of combinations to choose from. Note that with this representational choice, we do not have to decompose scenes into named and located objects. Figure 3.1 illustrates a rare case in which the filter response directly corresponds to a real-world entity.

### 3.3.2 The input feature space and classification

Gabor filters

The basic idea of the classifier we are building is that it is possible to select a set of Gabor filters whose combined responses carry a lot of information about the structure of image contents. This principle has been demonstrated in the past both experimentally and theoretically. For a short overview of the theory see [KB01], for its biological justification [JP87], for a texture segmentation viewpoint [WHD96], for hardware implementation issues [Shi99], and for related applications in scene recognition [CJR+98, SCB00].

For image plane coordinates $(x, y)$, the complex-valued impulse response of a Gabor filter centered at the origin of the plane is

$$H(x, y; \sigma, U, V) = G_\sigma(x, y) \exp(-i2\pi(Ux + Vy)), \tag{3.2}$$

where $i$ denotes the imaginary unit and $G$ is a Gaussian such as

$$G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right). \tag{3.3}$$

Thus, the filter has three external parameters: $U$, $V$, and $\sigma$. The complex sinusoid described by $H$ is centered at frequency $(U, V)$ and $\sigma$ determines the spatial extent of the Gaussian envelope $G$. Conceptually, one can find it convenient to replace $U$ and $V$ with the angle $\theta$ and spatial frequency $f$ so that $U = f\cos(\theta)$ and $V = f\sin(\theta)$. Hence,

$$H(x, y; \sigma, f, \theta) = G_\sigma(x, y) \exp(-i2\pi f(x\cos(\theta) + y\sin(\theta))). \tag{3.4}$$

From the above we see that $(x, y)$ is projected on the axis specified by the unit vector $(\cos(\theta), \sin(\theta))$, which identifies the direction toward which the complex sine wave of the spatial frequency $f$ evolves.

A source image $I$ is convolved with $H$ to produce a filtered image

$$m(x, y; \sigma, f, \theta) = |I(x, y) * H(x, y; \sigma, f, \theta)|. \tag{3.5}$$

Taking the pixel-wise absolute values transforms the complex convolved image into a real-valued image. The convolution itself is a

discrete approximation of

$$I(x,y)*H(x,y;\sigma,f,\theta) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(a,b)H(x-a,y-b;\sigma,f,\theta)\,\mathrm{d}a\,\mathrm{d}b, \tag{3.6}$$

which can be calculated faster in the frequency domain with the help of the discrete Fourier transform $\mathcal{F}$ and its inverse $\mathcal{F}^{-1}$:

$$m = |\mathcal{F}^{-1}\{\mathcal{F}\{I\} \otimes \mathcal{F}\{H\}\}|, \tag{3.7}$$

where $\otimes$ denotes element-wise multiplication in the frequency domain.

Let $m_{i,j}(x,y) = |I_i(x,y) * H_j(x,y)|$ denote the absolute value of the convolved image $I_i$ when using the $j$th filter of the chosen set of filters. The parameters $\sigma$, $f$, and $\theta$ are determined by $j$ and the chosen set. As the $j$th component of the feature vector $\mathbf{z}_i$ we select

$$z_{i,j} = \frac{s_{i,j} - \mu_j}{\sigma_j}, \tag{3.8}$$

where

$$s_{i,j} = \sum_x \sum_y m_{i,j}(x,y), \tag{3.9}$$

$$\mu_j = \frac{1}{M}\sum_{i=1}^{M} s_{i,j}, \tag{3.10}$$

$$\sigma_j = \sqrt{\frac{1}{M-1}\sum_{i}^{M}(s_{i,j} - \mu_j)}. \tag{3.11}$$

Above, $\mu_j$ and $\sigma_j$ are the mean and bias-corrected standard deviation of the sums $s_{\cdot,j}$, and $M$ is the number of images $I_i$ in a *training sequence* of images. The use of $\mu_j$ and $\sigma_j$ in the above manner is called *variance normalization* in elementary statistics. Because our filters $H_j$ can vary a lot in terms of response magnitude, it is reasonable to use variance normalization in order to make the feature components stand on equal ground. The normalization step is especially important in filter selection because principal component analysis is performed for sets of candidate features.

The sums $s_{\cdot,j}$ describe the amount of the $j$th kind of texture-like structure present in the image. For example, in Figure 3.2 we have

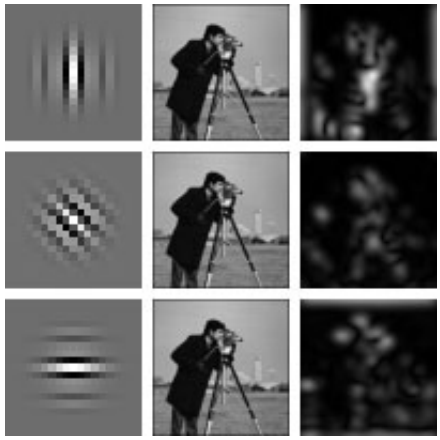


Figure 3.2: Three convolutions. The first column shows the real parts of three $19 \times 19$ Gabor filter masks with $\sigma = \sqrt{6}$, $f = 0.4$, and $\theta \in \{0, \pi/4, \pi/2\}$. These specific filter masks were used in the actual experiment as well. The filter masks are shown scaled so that the minimum of each mask is shown in black and the maximum is shown in white. The second column shows a $64 \times 64$ image that is convolved with each of the masks. The third column shows the pixel-wise absolute values of the convolution results. For illustrative purposes, the result images are scaled relative to each other so that the maximum value over *all* the results is shown in white.

convolved a low resolution image of a cameraman with three Gabor filters. Because the three filters differ only in $\theta$, normalization is not strictly necessary. Supposing that our set of filters consisted of the three filters shown, in the order from top to bottom, then we would have $s_{\cdot,1} > s_{\cdot,3} > s_{\cdot,2}$. Roughly, the first filter responds strongly to vertical bars such as that protruding from the bottom of the camera. The third filter responds to horizontal bars visible on the ground. The second filter has the weakest response due to the lack of diagonal bars in the image.

### Filter selection

The question remains how to select a suitable set of filters. Obviously, the feature vectors $\mathbf{z} = [z_1, \ldots, z_N]^T$ should be useful in clas-

sification, i.e., they should convey information about the classes. Furthermore, the size of the filter masks and the number of filters $N$ should be small to ensure fast computation of features. From the viewpoint of the root feature selection hypothesis, $\mathbf{z}$ should be computable several times a second with enough margin left for preprocessing and classification steps.

Because a root node should not care about subtle details, it is reasonable to use low-pass filtering on the images, as far as the images remain recognizable by human observers. The high-frequency content (including high-frequency noise) is attenuated in the process while the low-frequency content can be represented well using small low-resolution image matrices. The low-resolution images and small filter masks are both necessary because fast (discrete) convolutions are essential for the fast computation of features. The low-resolution images also allow small masks to represent filters that respond to structures that were relatively large in the original images.

Prior to preprocessing, the inputs are $640 \times 480$ *or* $320 \times 240$ grayscale images with 256 levels of intensity. The former input resolution is used most of the time, and in all offline experiments. The latter resolution is used *only* in online tests performed with a robot that is given fully-trained classifiers. In those tests, the robot initially captures $640 \times 480$ images, but it can access (focus on) $320 \times 240$ subimages of the captured images. At all times, however, the inputs (whole images or subimages) are downsampled to the resolution of $64 \times 48$ pixels prior to the use of Gabors and the computation of features. Downsampling to $64 \times 48$ is achieved by convolving the input images (or subimages) with a $11 \times 11$ low-pass filter with subsequent bilinear interpolation, i.e., exactly as the Matlab image processing function *imresize* does. In case of the robot experiments (without Matlab), the Matlab function was imitated exactly. The specific target resolution of $64 \times 48$ was chosen because it retains the aspect ratio of the original images and is close to the limit below which human recognizability of the images is not obvious.

The filter selection problem was resolved by the iterative visualization of data, where the data was projected in three dimensions with the help of principal component analysis [Hay99] that we now abbreviate PCA. The application of PCA creates an orthogonal ba-

sis of vectors that are linear combinations of the components of $\mathbf{z}$. The basis vectors (eigenvectors) have the useful characteristic that they indicate the directions in which the data varies the most and the least. By choosing the most important basis vectors (i.e., those with the largest eigenvalues) and projecting the data on the resulting basis, the projected data retains the significant variations of the data.

In the selection process, two distinct sets of data were used: a PCA training set and a visualization set. The PCA training set consisted of a video sequence captured with a robot moving in a room and a corridor. The visualization set consisted of labeled images from all the four target classes, i.e., *doorways*, *signs*, *people*, and *miscellaneous* indoor images (recall the overview of the experiments in Section 3.2). Both sets were collected with a robot using the collection procedure to be described in Section 3.3.3. These particular sets were used only in the filter selection stage.

Given these two sets of data, the following steps were done. First, a reasonable initial set of 24 Gabor filters $H_j$ was chosen. The ranges of the parameters $(f, \theta, \sigma)$ were reasonable given that the images (in both sets) were $64 \times 48$. For example, it was confirmed that the filters could be represented using small masks and that all orientations $\theta$ could be approximated. Second, the training images $I_i$ were convolved with the filters $H_j$, thus producing the intermediate results $m_{i,j}(x, y)$ for all training image pixels $I_i(x, y)$. Using the intermediate results, the unnormalized sums $s_{i,j}$ were computed. Third, the unnormalized sums were used to estimate $\mu_j$ and $\sigma_j$ for all $j$, after which the 24 feature values $z_{i,j}$ were computed for each of the training images $I_i$. Fourth, PCA was performed on the normalized feature vectors $\mathbf{z}_i$ of the PCA training set and the first three principal component axes were chosen as a basis. Fifth, the visualization set was projected on the basis and the distribution of the points was examined. After determining the apparent merits of a distribution, the ranges of the parameters $(f, \theta, \sigma)$ were tuned and the process was repeated from the second step to the fifth several dozen times. Finally, the set of ranges that produced the best set of filters was chosen.

In examining the distribution of the points from the visualization set, attention was paid to the emergence of clusters resembling the classes. For example, the visualization set contained images of

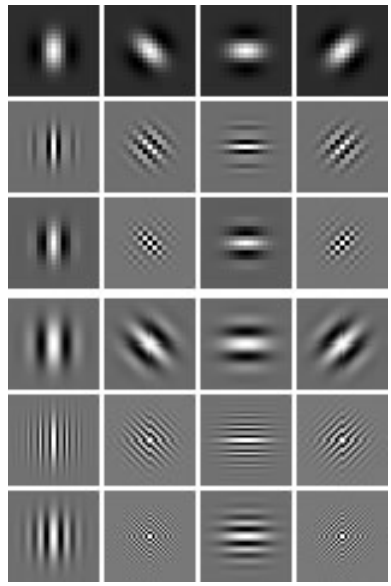


Figure 3.3: The real parts of the chosen filters. The $19 \times 19$ and $37 \times 37$ masks have been scaled to a larger size and are illustrated so that the minimum value of each mask is shown in black and the maximum in white. The columns show variations in $\theta$ while the rows show variations in $f$. The topmost 12 masks have $\sigma = \sqrt{6}$ while the rest have $\sigma = 6$.

people, and if the PCA projected feature vectors of such images were grouped together but separate from the transformed feature vectors of images without people, then the set of filters specified by the current ranges of parameters was given merit.

The end result was the set of 24 filters, the real parts of which are shown in Figure 3.3. The filter parameters had the ranges $\theta \in \{0, \pi/4, \pi/2, 3\pi/4\}$, $f \in \{0.1, 0.4, 0.8\}$, and $\sigma \in \{\sqrt{6}, 6\}$. All 24 combinations of the values were used in the classification experiments and all can be represented using either $19 \times 19$ ($\sigma = \sqrt{6}$) or $37 \times 37$ ($\sigma = 6$) complex-valued masks. Note that some of the parameter combinations result in filters whose usefulness may be questionable.

Roughly put, our overall approach in filter selection was that of *exploratory data analysis*, in which we tried to gain insight into

how the filter parameters affected the distribution of features and if any class-like clusterings could be found. The emphasis was *not* on discovering an optimally accurate set of filters or on creating automated processes and algorithms for feature extraction. We wanted to find out if a small and very crude set of filters operating on low-resolution images could potentially accomplish the classification task.

On one hand, we discovered that it was not possible to find a set of crude filters such that the resulting feature space would have made the classification step trivial. In the best of cases, the clusters were somewhat poor and overlapping. Using an obsolete 233MHz Pentium computer, the feature vectors $\mathbf{z}$ could be computed in about 2 seconds per image. Our more recent measurements showed that the same calculations took less than 0.4 seconds per image using a computer equipped with an AMD 3500 processor (and somewhat suboptimal code). We took into account all stages of preprocessing in these estimates, including the initial downsampling. The principal component projections were excluded from these estimates because they were used only in visualization. Using the projections in classification appeared to degrade the results. In other approaches pruning a set of filters using PCA has been claimed beneficial [TS01].

Finally, we note that the proposed kind of features are appropriate for examining the root feature selection hypothesis (Section 3.1.2). The features $\mathbf{z}$ are holistic texture descriptors and represent fairly basic global statistics of the responses of oriented features over individual images. Spatial relationships of feature responses are discarded when we compute the sums $s_{i,j}$ in (3.9).

Comparing our features to the features used by Renninger and Malik in their computational model for investigating the rapid scene recognition ability of human observers [RM04] (see also Section 3.1.2), we may note some similarities and differences. Renninger and Malik use histograms of oriented filter responses, which is similar to our approach in the sense the spatial relationships are discarded and that the relative amounts of different filter responses matter. The technical details, however, are quite different. We do not assign individual pixels to specific texton channels (e.g., based on which filter dominates) because we do not need descriptors that are normalized like histograms, and we are not interested in repro-

ducing biologically plausible or universal sets of filters or textons.

The hypothesis does not require that the root nodes for different problems use the same set of filters. Our emphasis is on the speed of *the computable model itself*, and the distinguishing characteristic of our experiments here is that we work with very low-resolution images and small filter masks.

### Classification

On the basis of the image preprocessing step, we now have feature vectors of the type $\mathbf{z} = [z_1, \ldots, z_{24}]^T$. Each feature component $z_j$ roughly measures the normalized amount of structure or texture type $j$ within an image.

The exploratory principal component analysis step showed that the classification problem was non-trivial. To some extent, filter selection and parameter modification could be used to adjust the feature space in order to get the desirable kind of nearest neighbor relationships. However, the results seemed clearly insufficient for enabling the reliable use of simple linear classifiers (hyperplanes) in the feature space of the vectors $\mathbf{z}$.

Instead of a direct correspondence between the feature components and filters, we perform classification on the basis of pairwise relationships of the components of $\mathbf{z}$. The space of vectors $\mathbf{z}$ becomes the *input space* of an SVM kernel (recall Section 2.3.2). The pairwise relationships of the components are captured implicitly by the second degree polynomial kernel,

$$K(\mathbf{z}, \mathbf{z}') = (\mathbf{z}^T \mathbf{z}')^2, \tag{3.12}$$

the values of which are simply dot products $\boldsymbol{\phi}(\mathbf{z})^T \boldsymbol{\phi}(\mathbf{z}')$ in a high-dimensional space. The high-dimensional vector $\boldsymbol{\phi}(\mathbf{z})$ has one component for each product of two components $\mathbf{z}$ has. Hence, given our 24 filters, the vector $\boldsymbol{\phi}(\mathbf{z})$ has more than five hundred components that are fortunately never computed explicitly.

### 3.3.3 Empirical evaluation

In the experiments, we had four broad categories that have uses in some robot navigation tasks. We had *doorways*, *signs*, *people*, and *miscellaneous* (corridor and room scenes) classes. Examples of the classes are shown in Figure 3.5. The classification problem of these classes is trivially solvable in low-resolution ($64 \times 48$) by human observers. The small number of the target classes of course helps in keeping the number of filters small and also helps in avoiding the emergence of class groups. The target classes can be characterized as follows:

- *Doorways* are precisely what the label indicates, i.e., rectangular door frames with the door itself completely open and not necessarily visible. The other side of the doorway, e.g., the adjoining room can be seen through the doorway. The images we used were close-up frontal views. In indoor navigation tasks, recognizing doorways may be useful because the navigation control program may "think" in terms of conduits that separate traversable open areas. In our experience, purely sonar-based approaches can have difficulties in discriminating miscellaneous environmental bottlenecks from real doorways.

- *People* are represented by frontal views of the head and upper torso of human participants. The frontal view requirement is not very precise, so people do not have to waste time in carefully positioning themselves in front of the robot. The size of faces ranges from 4% to 15% of total image area. It should not be assumed that people are detected based on faces and heads alone. The upper torso, including clothing, may be crucial.

- *Signs* are sheets of paper that have a word on them in very large font. As opposed to doorways and people, we do not assume that signs occur "naturally" in indoors environments. Instead, they may be artificial *landmarks* in robot navigation, marking special places or circumstances. Because the words are on otherwise empty sheets of paper, the individual letters are relatively easy to segment and classify. Hence, the class of signs allows intuitive examples of what may be expected of ideal delegation. After recognizing a sign, the root node of a

system may delegate the responsibility for finer distinctions to a node that uses specialized segmentation algorithms, i.e., segmentation algorithms that are useless in the general case, but useful in segmenting these landmarks.

- *Miscellaneous* images are views of rooms and corridors that do not belong to the other three classes. The class of miscellaneous images is a natural superordinate category containing many unnamed basic categories, e.g., walls.

As required by the preconditions (assumptions) of the root feature selection hypothesis, each of the four classes either belongs to a distinct superordinate category or is one (i.e., miscellaneous). Using the heuristic criterion for the lack of class groups (as in Section 3.1.2) we saw that one apparent similarity between the classes is that images of doorways and signs always contain prominent rectangles (or prominent vertical and horizontal lines). This similarity, however, is shared by all classes, because unsegmented indoor views tend to contain rectangles. Also, the class of miscellaneous corridor and room scenes has more variability than the other classes. There is a small subset of images in which structures resembling doorways are seen. Hence, this subset combined with the class of doorways could form a class group. Because the subset is small, doorways and miscellaneous scenes are not grouped together. Images from all the classes also tend to have walls and portions of the floor visible.

Prior to training any classifiers or selecting any filters (Section 3.3.2), we collected images from the four categories (or classes) using a mobile robot. The overall idea was that the relevant objects were perceived in their correct contexts, viewed from reasonable viewpoints, and at appropriate scales. The robot was a Nomad Super Scout II with vision (see Figure 3.4). The color video camera of the robot captures $640 \times 480$ pixel images at five Hertz. In the current experiment, we discarded the colors and downsampled the images to $64 \times 48$ pixels prior to computing features.

To capture miscellaneous images of rooms and corridors, the robot was placed in random initial locations after which random turns and movements were simulated. The camera on top of the robot was kept pointing to the front so that the $x$-axis of the captured $640 \times 480$ images was aligned with the horizon at all times. In addition to the camera, the robot had a ring of sonar range-finders

Figure 3.4: Nomad Super Scout II, a mobile robot equipped with a video camera.

providing information about the proximity of walls and other obstacles. To simulate reasonable viewpoints, extreme close-up views of obstacles were avoided. During these simulations, the robot had no image recognition abilities. The captured images were labeled afterwards. In capturing images of doorways, people, and signs, we simply placed the robot to random locations near an appropriate target and made the robot turn approximately towards the target. In some tests that are described below, we deviated from the above procedure. These deviations are always noted explicitly, and unless it is said otherwise, the reader should assume that the above description holds.

After initial data collection and feature selection, we performed three tests. The first two of the tests were done entirely offline on a desktop computer. In these, we measured different kinds of generalization ability (narrow vs. broad). As a result, we have numeric performance statistics that can be used for comparisons. The third test was not as formal as the preceding two. In it, we downloaded trained classifiers to the robot and let the robot make predictions when executing some simple navigation-related behaviors. We also measured the running times of the classifiers and were ready to note if the observed performance differed significantly from the offline experiments.

In this dissertation, the role of these tests is that we may show that the claims of the root feature selection hypothesis are satisfied when the assumptions are. Although the particular system that we build here likely *does not* allow state-of-the-art applications, it allows the study of the hypothesis. We can focus on speed issues

Table 3.1: Set sizes in basic holdout validation.

| Class | Train | Validation | Total |
|---|---|---|---|
| Doorways | 30 | 10 | 40 |
| People | 18 | 6 | 24 |
| Signs | 12 | 3 | 15 |
| Miscellaneous | 30 | 10 | 40 |
| Overall | 90 | 29 | 119 |

using a small and very crude set of features computed in very low resolution, which is not possible if one aims for maximal accuracy. By controlling and varying the data collection process (as detailed in the tests below), we can also investigate the effects of sampling bias in training classifiers for broad target classes. It is important that root nodes can tolerate this kind of bias.

### Basic Holdout Validation Test

In the first test we performed repeated holdout validation with randomized train and validation sets. The test is the least interesting of the tests, but it allows us to assess if the *chosen features and filters* are any good at all.

Holdout validation falls under the broad umbrella term of cross-validation, although strictly speaking the train and validation sets are chosen randomly and not crossed over. We repeat the holdout validation process several times and then examine the average and worst case results. The sizes of the randomized train and validation sets are shown in Table 3.1. For example, each random training set has 90 images, 30 of which are doorways, and each validation set has 29 images, 10 of which are doorways.

Some examples of the $64 \times 48$ pixel images are shown in Figure 3.5. In sampling the images for the first test we made some interesting restrictions. The images from the class of people represented only one particular subcategory (subclass) that was Person A (Figure 3.5 c) viewed from different angles and against different backgrounds. Likewise, the images from the class of signs represented the subcategory of signs with the text *WAIT* on them (Figure 3.5 d).

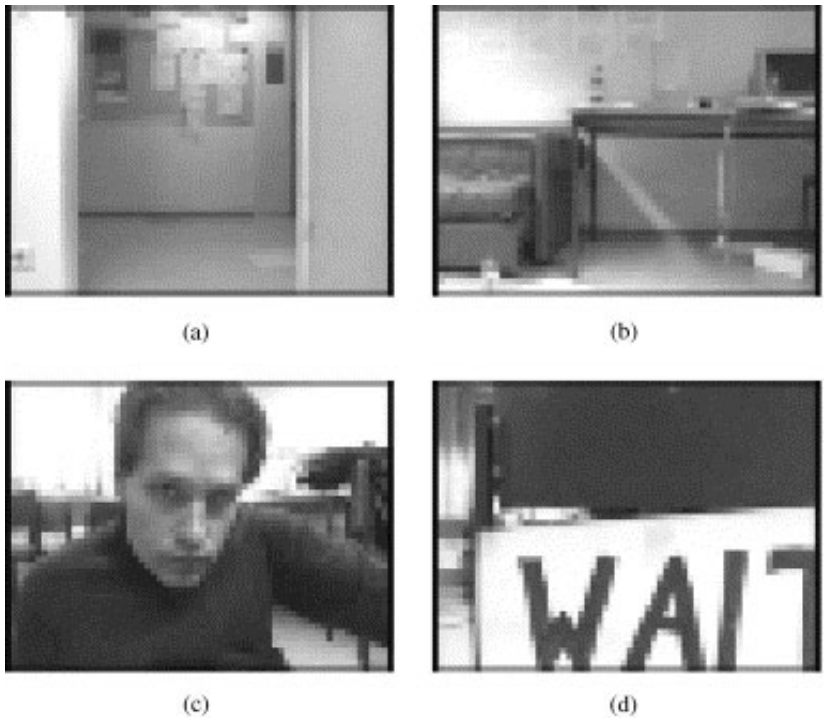



Figure 3.5: (a) A doorway, (b) a miscellaneous scene, (c) person A, and (d) a sign.

In the first test, the above restrictions imply that the two classes are simplified, classification may become easier or harder, and this may be reflected on the holdout validation results. In the second test, however, we lift these restrictions from the *validation* sets while keeping them for the *training* sets. Hence, we get to see how well training on subcategories generalizes to performance on the broader category level. Training on subcategories can be called biased sampling, and it is interesting to see if this can be allowed in training root nodes, as it could save human effort. *A priori*, it is not unreasonable to expect that the chosen kind of features allow this. If subcategories differ by somewhat subtle details, then the chosen kind of crude features may be blind to those details.

In the first test the validation procedure was as follows. First, from the set of images collected using the robot we randomly sam-

Table 3.2: Basic holdout validation results.

| | Doorways | People | Signs |
|---|---|---|---|
| Class prior | 1/3 | 1/5 | 2/15 |
| Minimum correctness (%) | 69 | 93 | 93 |
| Average correctness (%) | 88 | 99 | 99 |
| Correct/29 | 25.395 | 28.83 | 28.755 |
| Doorways/10 | 1.53 | 0.025 | 0.045 |
| Miscellaneous/10 | 1.66 | 0 | 0.005 |
| Humans/6 | 0.035 | 0.075 | 0 |
| Signs/3 | 0.38 | 0.075 | 0.195 |

pled 40 images of doorways, 24 images of Person A, 15 images of *WAIT* signs, and 40 miscellaneous images of rooms and corridors. Second, the holdout validation procedure was repeated 200 times as follows.

1. Choose the validation set by sampling from each class without replacement. Sample 10 images of doorways, 6 images of Person A, 3 signs, and 10 miscellaneous. The training set consists of the remaining 90 images.

2. For each target class, except the class of miscellaneous images, train a support vector machine using the *one-vs-rest* training scheme (Section 2.3.2). Hence, each of the three machines has 90 training images, the minority of which are labeled positives. For example, the machine for recognizing doorways has 30 positive inputs and 60 negative inputs.

3. Process the validation set of 29 images using the three machines and record the results for analysis.

Table 3.2 shows the results of the repeated holdout validation procedure. All rows, except the first and second, show average values over the 200 trials. The first row shows the prior probabilities of each class. The second row shows the minimum correctness of each of the class-specific machines over the 200 trials. Each of the minimum values is larger than what could be expected of baseline predictors that predicted according to class priors as estimated from the training sets. The baseline predictors could be expected to have

correctness rates of 2/3 (betting against doorways), 4/5 (betting against people), and 13/15 (betting against signs). The averaged correctness estimates on the third row are significantly larger than what could be expected of the baseline predictors. The fourth row shows how many inputs each machine classified correctly out of the 29 in each validation set. The remaining rows show the class-specific numbers of misclassified inputs for each machine (column).

From these results, we can see that the features based on the 24 chosen filters clearly convey class-related information and that class-specific machines can be trained using small training sets. Hence, it is reasonable to proceed to the more labor-intensive second test.

### Validation with altered sampling bias

In the first test, there are a few problems. The problems result from the idealized validation setting, limited data, and the simplified circumstances.

The first problem, from the viewpoint of the root feature selection hypothesis, is that we did not construct *full classifiers* capable of four output labels. We simply tested separate binary SVMs, each trained to discriminate one class from the rest. The separate machines were appropriate for testing the feasibility of the features, but are not appropriate for much else.

To overcome the problem, we now use a simple rule for combining the predictions of the binary SVMs. The rule is a simplified variant of the rule we use for combining SVMs in the root node of the experimental system to be presented in Chapter 5 (Section 5.3.1). Denoting the SVM trained with the class $c$ as the positive class by $\text{SVM}_c$, the simple rule is as follows:

1. Activate all class-specific binary SVMs.
2. If all three SVMs predict $-1$, then choose *prediction* $\leftarrow$ *miscellaneous*.
3. Else, if a single $\text{SVM}_c$ predicts $+1$, then choose *prediction* $\leftarrow c$.
4. Else, if more than one SVM predicts $+1$, then choose *prediction* $\leftarrow$ *miscellaneous*.

The above rule suffices here, but it should be noted that the full rule is necessary for delegation and the full rule in turn requires (modified) SVMs with three output values. One way to construct that kind of SVMs is to use Platt's procedure (see Section 2.3.3) and then discretize the resulting monotonic confidence model.

The second problem is that the data was partitioned *randomly* into training and validation sets. Hence, we cannot really see how the features and machines generalize when there are sampling biases in imaging conditions. For example, if the training set of images is captured during daylight hours, then how well do the trained machines classify images captured under artificial lighting conditions. If the shadows cast by objects are altered, then the edges and bars contained in images are altered as well. Given that we use Gabor filters, these kinds of alterations may affect the feature values. Because root nodes are expected to handle *broad* classes, the ability to tolerate various sampling biases in imaging conditions is certainly important. Sampling biases cannot be investigated using random partitions because subsets of the captured data get mixed. For example, if we have images captured both during daylight hours and under artificial lighting, then the training and validation sets will each have both types of images.

The third problem is similar to the second. Our sets of images may contain multiple views of some individual object instances, i.e., some individuals are overrepresented considering the number of individuals a broad class contains. Hence, due to random mixing the validation results may reflect generalization performance over *different views* of particular individuals instead of generalization over *different individuals* of the same class. Because root nodes are intended for broad classes that may contain a very large number of different individuals, or object instances, it is important to test that generalization over individuals occurs. To some extent, it is reasonable to expect that the use of crude features promotes the right kind of generalization, as subtle (e.g., fine-scale) individual differences may be lost in the feature extraction stage.

To alleviate the second and third problems of the first test, we altered the sampling bias of the validation sets. We sampled new validation data while keeping the data from the first test as training data. Naturally, the i.i.d. assumption of the SVMs was violated, but testing the effects of altered sampling bias on generalization

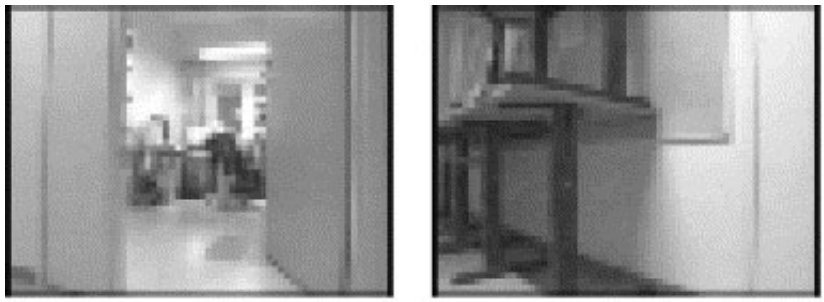
ability became possible. In real world applications, classifiers are used for decision making, and decisions (e.g., related to movement) affect the distribution of the inputs. Hence, classifiers should not be fragile w.r.t. the sampling bias.

In the new validation set of *doorways*, we used artificial lighting exclusively, resulting in somewhat darker images with different shading compared to the portion of the training set captured in daylight. In addition, the training and validation sets had no overlap in the sense of common object instances. The new set was acquired from different sections of the building. An example is shown in Figure 3.6(a).

In the new validation set of *miscellaneous* images, we discarded the simulation of random movements. Random movements make it practically difficult to sample images from *large* spaces so that certain locales are not overrepresented (e.g., near the initial locations) and others underrepresented. In the new set we changed the sampling bias so that larger spaces could be covered with no accidental duplicate images. In the new set we sampled systematically along a corridor. The images were captured from 10 different locations and we took 8 images from each. The distance between two consecutive locations was roughly 1.5 meters plus a small random factor (from 0 to about 30 centimeters forward or backward). The locations were roughly on the centerline of the corridor, as the robot was not allowed to drift far off the line. The eight images from each location were taken with a 45-degree rotation between each. An example is shown in Figure 3.6(b). Two of the systematically taken images happened to be doorways and were not counted as miscellaneous.

In the new validation set of *people*, we expanded the generality (breadth) of the class compared to the training set (recall that the training data inherited from the holdout test had images of a single person only), and also introduced a few deliberately distracting non-class images. The new set was as follows:

- 3 images of person A (the training person) wearing a lightly colored shirt as opposed to the dark shirt worn in the training instances,
- 3 images of person B in a lab and 3 images of him in his office,
- 3 images of person C in the lab and 3 images of him in his



(a) (b)

Figure 3.6: (a) A doorway and (b) a miscellaneous corridor scene.

(a) (b)

Figure 3.7: Persons C and D.

office,

- 3 images of person D in her office,
- 3 images of person E leaning on a corridor wall, and
- 6 control images of some of the situations above — the test person was replaced with an object (e.g., a chair) while the circumstances were otherwise unaltered.

Two examples from the new set are shown in Figure 3.7.

In the new validation set of *signs*, we expanded the generality of the class by introducing a new type of sign not present in the training data (recall that the inherited training data had *WAIT*

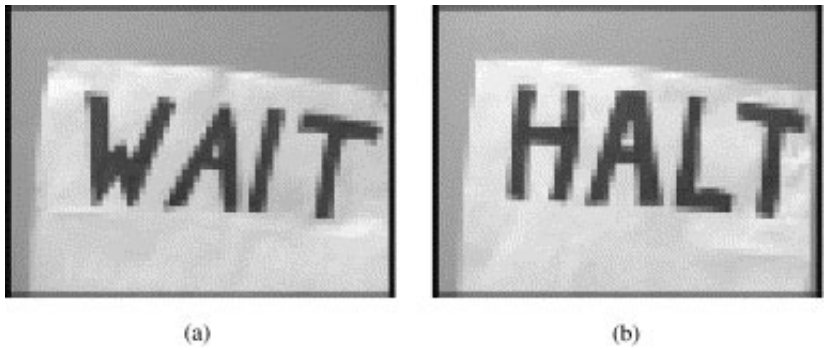



Figure 3.8: The pair of signs.

Table 3.3: Set sizes in validation with altered sampling bias.

| Type | Size | Characteristics |
|---|---|---|
| Doorways | 19 | no natural light sources, completely separate physical instances |
| Mostly Miscellaneous | 78+2 | systematic sampling |
| Mostly People | 21+6 | different people, more varying backgrounds |
| Signs | 15+15 | another type of sign involved |

signs only). The new type of signs had the text *HALT* on them. The new set had 15 pairs of the signs *WAIT* and *HALT* placed against various backgrounds. Circumstances were otherwise controlled so that the letters on both types were of equal size, thickness, spacing, and relative coverage of the background. A pair is shown in Figure 3.8.

The new validation sets are summarized in Table 3.3. The rows correspond to class-specific validation sets. The second column shows the size of each set, and the third summarizes the set-specific details explained above.

From the viewpoint of training the classifiers, the SVMs responsible for recognizing people and signs were trained with subcategories of positive inputs. The SVM for people was trained with images of one person while the validation set had images of five

people. Similarly, the SVM for signs was trained with one kind of sign while the validation set had two kinds of signs. In addition, we limited the availability of negative training inputs so that the machines were trained with subcategories of negatives. The availability of negatives was limited as follows:

- From the training set of the SVM for doorways, we removed all images of people. The remaining negative inputs were signs and miscellaneous.
- From the training set of the SVM for people, we removed doorways and signs, but added 10 more miscellaneous images for balance. The negative inputs were all miscellaneous.
- From the training set of the SVM for signs, we removed doorways. The remaining negatives were people and miscellaneous.

As can be seen in the above, the removals were not symmetric, i.e., there are no two machines such that their training sets would be identical if the signs of the labels were reversed. In the case of the SVM for doorways, we found out that we could not remove signs from the negative inputs without noticeable effects on accuracy.

Based on the fact that these reduced training sets worked (as apparent from the results further below), it can be speculated that if the classes form clusters in the high-dimensional feature space, then the class of miscellaneous images could be in the "middle" of the other clusters. For example, a hyperplane that separates people from miscellaneous also separates people from all clusters that are on the opposing side of the hypothetical cluster of miscellaneous images. Hence, it can be speculated that it may be possible to add new classes without having to retrain the existing machines. The whole set of new and old classes would have to "surround" the miscellaneous class, distant from each other but close to the nearest representatives of miscellaneous.

With the new validation sets, we got the results shown in Tables 3.4, 3.5, 3.6, and 3.7. The last row in each table represents the *worst* result that could be possible using the proposed combination rule for combining the outputs of the binary SVMs. We use these worst-case estimates, because we did not use the rule when the original experiments were made, and it is not possible to find out

Table 3.4: 19 doorway images.

| SVM | Mistakes | Description |
|---|---|---|
| Doorways | 3 | false negatives, 84.21% correct |
| People | 0 | no false positives |
| Signs | 0 | no false positives |
| Combiner | 3 | 84.21% correct |

Table 3.5: 78 systematically photographed miscellaneous scenes and 2 doorways.

| SVM | Mistakes | Description |
|---|---|---|
| Doorways | 29 | false positives, 63.75% correct |
| People | 0 | no false positives |
| Signs | 4 | false positives, 95% correct |
| Combiner | $\leq 33$ | at least 58.75% correct |

Table 3.6: 21 images of people and 6 control images.

| SVM | Mistakes | Description |
|---|---|---|
| Doorways | 2 | false positives from 2 control images, 92.59% correct |
| People | 4 | 1 false positive, 3 false negatives, 85.19% correct |
| Signs | 0 | no false positives |
| Combiner | $\leq 6$ | at least 77.78% correct |

Table 3.7: 30 images of signs in 15 pairs.

| SVM | Mistakes | Description |
|---|---|---|
| Doorways | 1 | a false positive, 96.67% correctness |
| People | 0 | no false positives |
| Signs | 3 | false negatives, 90% correctness |
| Combiner | $\leq 4$ | at least 86.67% correct |

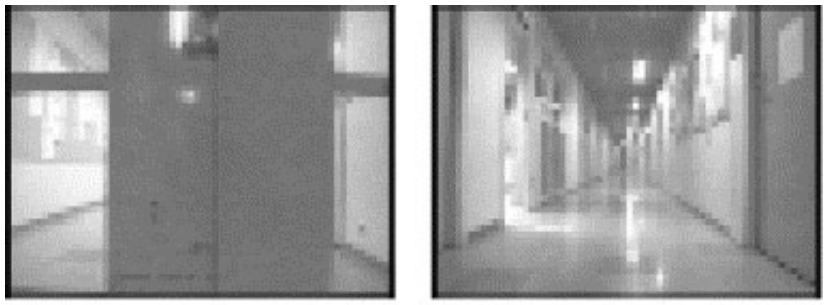



Figure 3.9: Two false doorways.

which individual SVM errors were actually simultaneous. The worst of these worst-case estimates shows 58.75% correctness. Hence, the average correctness of the combiner over the sets is higher. Note that the average can be taken because the SVMs (and thus the combiner) were the same in each test. In the new validation sets the most probable class is *miscellaneous*, the probability of which is 53.85%. Hence, the combiner appears to be better than a fixed (chance-based) classifier that predicts the class with the largest prior (in the validation set).

We analyzed the mistakes the doorway SVM made on miscellaneous images (Table 3.5). We found out that 4 of the mistakes resulted from a situation that might be difficult to handle without sophisticated (expensive) features and large training sets. These four mistakes resulted from images that had a tall window on a wall (for an example, see Figure 3.9 a).

Another 7 of the mistakes had the camera pointed toward an end of the corridor (there being 20 such images in the set). The cause of these mistakes might be that there are doorway-like frames on the walls (for an example, see Figure 3.9 b).

Some of the remaining mistakes resulted from images that had a closed door depicted in them. This is understandable because it is likely that doorways are recognized by emphasizing the filters that respond to door frames. The scenes that are visible through open doorways do not have much in common, except that they are not empty.

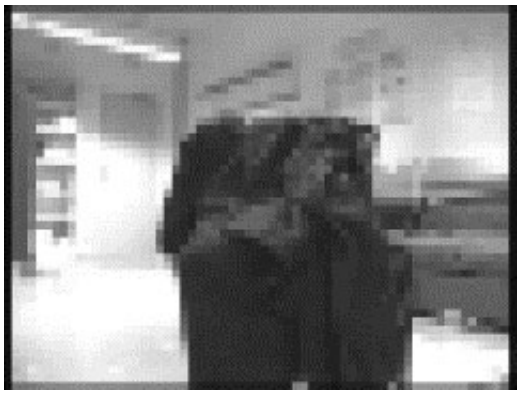


Figure 3.10: A falsely detected person.

The four mistakes made by the SVM detecting people (Table 3.6) are revealing. The false positive resulted from a control image that had an overcoat resting on top of a chair (see Figure 3.10). This might result from the SVM learning to emphasize filters that respond to textures of clothing, i.e., there is no reason to assume that detecting people works by detecting heads or faces exclusively. Two of the false negatives occurred with person E. It is possible that the problem could be corrected by expanding the training set (recall that to investigate sampling bias we trained the SVM using images of a single person only). The remaining false negative had person C in the image, but quite far away from the camera — much further away than the single person in the images of the training set ever was.

The false negatives of signs (Table 3.7) all had the text *HALT* on them. This is not surprising because the SVM was trained with just *WAIT* signs as positive inputs. It is possible that the mistakes are a consequence of the changes of sampling bias.

### Validation with the robot

Encouraged by the results of the second test that indicated the SVMs were tolerant of changes of the sampling bias, we performed the third test with the robot. Tests with a robot would be somewhat uninteresting, if the classifiers could not affect decisions re-

lated to movement. If, however, the predictions of the classifiers affect movement, then it is likely that the distribution of the images and labels changes. The pairs are no longer independent and identically distributed over time as assumed by the SVMs. Equivalently, we could say that we are still sampling from some universal distribution, but the sampling bias is affected by the movements. For example, if a navigation program ran the robot against a wall and kept the robot there, then the distribution of the inputs would be changed drastically compared to the distribution the training set was drawn from.

We downloaded the trained SVMs of the second test to the robot and experimented with some simple behaviors. Since Matlab was no longer involved, it made sense to begin with efficiency measurements. Using the now outdated computer (Pentium 233MHz) on board the robot, each image took about 2 seconds to classify (including all stages, such as the initial downsampling of the images).

Of the required time per image, the portion required by the SVMs was negligible. Hence, as long as the set of SVMs uses the same feature space and the number of classes is not huge, scalability would not seem to be a problem from the viewpoint of efficiency. Of course, having one SVM per class in a node results in the resource losses of the node being roughly linear in the number of classes. Because the constant part of the loss (preprocessing and features) is relatively large, however, the effect of adding or subtracting half a dozen classes is insignificant, unless the number of support vectors per machine changes significantly.

One of the simple behaviors we tested was *approaching* a seen target. As we mentioned in Section 3.3.2, the original inputs can be either $640 \times 480$ or $320 \times 240$ pixels prior to downsampling them to $64 \times 48$ pixels. Because the SVMs were trained to recognize targets of a certain size, e.g., people whose presence dominates the $64 \times 48$ pixel images, it is necessary to process subimages of the $640 \times 480$ pixel images to detect targets that are far away.

To detect a target that is far away, the robot may sample subimages of the images it sees. The subimages are transformed to $64 \times 48$ pixels and then classified. If a subimage seems to contain a target, the target can be approached. To avoid some false positives, several overlapping subimages may be classified and subjected to voting. When approaching a target, the presence of the target may

be checked again.

We noted that additional sonar inputs could make behaviors such as approaching, or moving through doorways more efficient. Because each SVM was trained to recognize images in which the target dominates, it is useless to classify images in which there can be no dominating targets. For example, if the sonars indicate that there is nothing in front of the camera, then the robot cannot be close to a doorway and it is useless to have the SVM responsible for detecting doorways to process the *whole* input image.

Overall, the third test demonstrated that the accuracy of the classifiers was not radically different from what was seen in the second test, even if the sampling bias was different, more unpredictable, and harder to characterize.

## 3.4 The additional experiments

In this section, we present additional experiments that complement the main experiment of this chapter. In the main experiment we showed that broad classes could be discriminated from each other efficiently even if the classification machinery was trained with a wrong sampling bias. One interesting form of bias was that machines were trained with subcategories of inputs. For example, at the training stage the machines were exposed to one kind of sign (*WAIT* signs) while testing involved two kinds of signs (*WAIT* and *HALT*).

In the additional experiments presented here, we examine what can be done *after* an image has been given a broad kind of a class label by the root, e.g., it has been labeled as a sign. After the broad category has been identified, subcategory level classification can be attempted. This subcategory level classification can be either harder or easier than classification on the broad level. Here we focus on the easier kind of classification. Examples of the harder kind follow in the next chapter.

Certain kinds of clear similarities between subcategories may enable easier discrimination between these subcategories. For example, if the objects of the subcategories are of similar color, this similarity may enable adequate and fast segmentation, which in turn may enable fast shape-based discrimination between the sub-

categories. In other words, similarities of one kind may enable easier perception of dissimilarities of another kind.

In the experiments below, we use SVMs to recognize artificial landmarks that are essentially like the signs from the main experiment of this chapter. The blank backgrounds and the uniform foreground colors of these landmarks enable fast segmentation that has sufficient quality for recognition. Further below, we increase the difficulty of the landmarks by removing the requirement of blank backgrounds and by relaxing the requirement of uniform object colors. Thus, we can examine the accuracy of the SVMs when the segmentation method is pushed to the breaking point.

### 3.4.1 Simple segmentation for specialists

Above, we noted that some similarities between subcategories may enable fast segmentation with sufficient quality for discriminating between the subcategories. Sufficient quality, of course, depends on the classifiers used.

Given objects like the signs (artificial landmarks) that have uniformly colored foregrounds and backgrounds, it is reasonable to make the tentative assumption that adjacent pixels of the same color can be joined into segments that correspond quite directly to whole projections of whole objects. In other words, segmentation may be based on simple and well-known region-growing methods [SHB99] such as that of Bruce, Balch and Veloso [BBV00].

In contrast, segmentation of various kinds of objects originating from dissimilar higher-level categories may be expected to require much more generic segmentation algorithms. Achieving generic, reasonably fast segmentation of adequate quality is difficult because, in general, the segmentation problem cannot be reduced to any simple criteria that could be evaluated separately for every disjoint part of the input image [SM00, Pal99]. The decisions related to each pixel, i.e., whether it is a background or object pixel, may be intertwined with the corresponding decisions of all the other pixels. As a result, somewhat generic segmentation algorithms tend to be slow. For example, the famous *normalized cuts* approach of Shi and Malik [SM00] takes segmentation as a graph-cutting problem in which every pixel is considered in parallel. Solving the cutting problem is then reduced to solving a generalized eigensystem, which

is slow if the input image is not very small. Even if there is enough time to run a supposedly generic segmentation algorithm, there seems to be no guarantees that individual segments correspond to projections of whole objects.

In a sense, the framework of efficient classification discussed in this dissertation is opposed to the classic view of segmentation always preceding recognition. Segmentation stages can be interleaved with classification stages, and there is no reason why segmentation algorithms should reside in the root node.

We use a trivial color-based segmentation method in which neighboring pixels of similar color are joined into segments. The method takes $640 \times 480$-pixel RGB images as input. The pixels that approximately match the known object foreground color are marked. Using the marked images, adjacent marked pixels are joined into segments. The initial segments are not refined in any manner – if two segments do not touch, they remain separate segments. From the list of segments, all segments consisting of fewer than 100 pixels are discarded. Finally, the method takes the largest segment found, fits a bounding box around the segment and then resizes the contents of the bounding box to a $32 \times 32$ matrix of bits. A bit gets the value of one if the corresponding position is within the segment and zero otherwise. Hence, pixel intensities and colors are discarded and only the coarse shape of the largest segment remains to represent the input image. Before classification, the $32 \times 32$ binary matrices are flattened into 1024-dimensional binary vectors that become SVM inputs.

The segmentation method is certainly not sophisticated, but it is fast and the coarseness of the selected segments allows us to see how well SVMs perform when given less than perfect inputs.

### 3.4.2 Experiments with landmarks

In the first part of these experiments we collected images of different flat landmarks which had simple uniform foregrounds and backgrounds. The landmarks were created by cutting different letters from red paper and then attaching them on sheets of blank white paper. The sheets were attached to walls, doors, chairs, and miscellaneous indoor surfaces. Images of these landmarks were captured under varying lighting conditions and from several viewing angles.

Figure 3.11: Representative segments from the classes *8* and *A*

For capturing the images, we used the robot mentioned in the description of the main experiment of this chapter. We divided the landmarks into nine classes.

In the preprocessing stage, all images were segmented as described in the previous subsection. We gave each image a class label, and the largest segment of each image inherited the label. After preprocessing, we got labeled binary matrices each representing one segment of one input image. Some examples of typical segments are shown in Figure 3.11. As illustrated, the largest segments could often represent the whole foreground.

Each of the nine classes was given a dedicated SVM that used the second-degree polynomial kernel given in Equation (3.12). Hence, the features subjected to linear classification correspond to products of all pairs of the 1024 bits. In the kernel-induced feature space, the feature vectors have approximately one million bits with a specific bit getting the value of one if and only if the bit corresponds to a pair of ones in the original bit matrix of the segment.

As in the main experiment of this chapter, each dedicated SVM was trained and tested using one class as the source of positive inputs and the rest of the classes as sources of negatives. As an additional source of negatives, we used binary matrices of miscellaneous segments originating from images that did not belong to any of the proper classes. Below, these negatives are referred to as *no class* inputs.

The classification and generalization ability of the SVMs was measured using repeated five-fold cross-validation. Each SVM was validated separately. Overall, we had about 20 to 40 images from each class. The images from each class were randomly partitioned into five folds, each containing equal numbers of images from the class. During each of the five iterations (per machine) of the validation process, four folds were used for training and one for validation.

Table 3.8: Cross-validation results for single letter landmarks.

| | A | B | C | D | O | P | R | S | 8 | – |
|---|---|---|---|---|---|---|---|---|---|---|
| A | 97 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
| B | 0 | 94 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| C | 0 | 0 | 100 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| D | 0 | 0 | 0 | 98 | 0 | 0 | 0 | 0 | 0 | 0 |
| O | 0 | 0 | 0 | 0 | 99 | 0 | 0 | 0 | 0 | 0 |
| P | 0 | 0 | 0 | 0 | 0 | 95 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | 0 | 0 | 0 | 99 | 0 | 0 | 8 |
| S | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 97 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 92 | 0 |

Hence, each training set had about 16 to 32 training inputs per class and each validation set had about 4 to 8 inputs per class. Therefore, for each machine the positive inputs were always in the minority. The process of five-fold cross-validation was then repeated 20 times and the results were averaged over repetitions and folds. The SVMs were built using the publicly available *SvmFu* implementation (`http://fpn.mit.edu/SvmFu/#getting-download`).

The averaged validation results are shown in Table 3.8. The rows correspond to the class-specific SVMs and the columns correspond to classes. The symbol – denotes the *no class* class, i.e., the set of miscellaneous negative inputs. Each number indicates the averaged percentage of inputs from a particular class getting a positive (+1) response from a machine. For example, the first row shows that, on the average, the machine trained to recognize $A$ recognizes 97% of inputs of class $A$ and falsely recognizes 2% of non-class inputs as $A$. In contrast, inputs of class $B$ are not mistaken for inputs of class $A$. As can be seen in the table, the results were very good. On the other hand, the objects were quite easy.

In the second set of experiments, we increased the difficulty of the landmarks. We used three-dimensional objects that were pictured on non-uniform backgrounds (e.g., backgrounds such as that shown in Figure 3.5 (b), the miscellaneous scene). The new objects had large regions of uniform color, but because the objects were not flat (except one), shading and different viewpoints had a larger impact on the apparent colors and shapes of the segments. The new objects

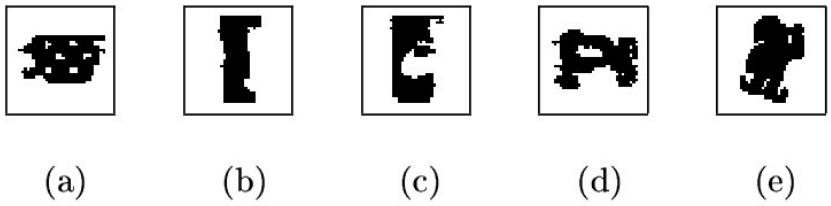



Figure 3.12: Segmentations of a) coffee cup, b) large can (side view), c) small can (side view), d) football, and e) christmas elf (flat).

Figure 3.13: Segmentations of the football.

were harder for the simple segmentation method, resulting in more spurious segments, i.e., the results looked more random.

In the new data, we had images of five objects: a dotted coffee cup, a football, two different beverage cans of different sizes, and one red christmas elf (flat). Some of the largest segments associated with these objects are shown in Figures 3.12 and 3.13. The randomness of the segmentation results is apparent in Figure 3.13, where we show the largest segments of different images of the same football.

The cross-validation results with the new objects are shown in Table 3.9. The validation process was identical to the one above,

Table 3.9: Cross-validation results for more difficult landmarks.

| | cup | football | large can | small can | elf | – |
|---|---|---|---|---|---|---|
| cup | 95 | 2 | 0 | 1 | 0 | 8 |
| football | 4 | 74 | 0 | 2 | 0 | 7 |
| large can | 0 | 0 | 95 | 0 | 0 | 8 |
| small can | 0 | 2 | 0 | 95 | 0 | 0 |
| elf | 0 | 0 | 0 | 0 | 97 | 0 |

i.e., we had 20 repetitions, 5 folds, and about 20 to 40 images from each class per fold. The table is interpreted like Table 3.8 above. For example, 8% of the *no class* inputs caused the cup SVM to predict a false positive.

The latter results (Table 3.9) are clearly worse than the earlier ones (Table 3.8). Together, the results illustrate that the segmentation method starts to break down when the objects are no longer flat. In Table 3.9, it can be seen that the SVMs dedicated to the christmas elf (a flat object) outperform the others. The SVMs dedicated to side-views of cylindrical objects (cup, cans) seem moderately successful, while the SVMs dedicated to the spherical object (football) perform the worst.

From the viewpoint of the overall framework, we can discuss what these results imply regarding the delegation process. When a root node delegates an input image to a specialist, it is reasonable to ask if the specialist classifier can reject the input as inappropriate, i.e., outside the specialty of the specialist. In other words, it is reasonable to ask what happens after an error in delegation.

The simple segmentation method seems to lead to good results when the objects are flat signs, but the property of flatness does not seem to be efficiently and reliably detectable from the appearance of the segments. Although the average number of segments per image was larger for some non-flat objects, using the number of segments as a threshold to determine rejection seemed implausible. When the number of (large) segments was larger than one, the object could well be flat or not. With a threshold of one segment, unnecessary rejections would result. Moreover, other superficial (efficiently recognizable) characteristics of the segments, such as size, did not seem to yield any better criteria for rejection.

Hence, at least in the current case, it seems that characteristics of the segments cannot efficiently predict the reliability of segment-based classification prior to executing the classification code. In the general case it should not be assumed that rejection is possible with anything less than the full resource loss of the node.

## 3.5 Summary

In this chapter we began to bridge the gap between theory and practice. The classification framework that we discuss involves many design choices not covered by the theory from the previous chapter. For example, given a classification problem, what kind of features should be used in each node. In this chapter, we began to consider what kind of features should be used in the *root* node.

A hypothesis was proposed, called the root feature selection hypothesis. The hypothesis makes claims about what kind of features should be used in a root node to avoid efficiency bottlenecks. The claims have certain preconditions (assumptions), the purpose of which is to limit the claims to a particular sort of classification problems.

The most complex of these preconditions is that the target classes have to be broad categories of everyday scenes or objects such that no classes can be grouped together based on similarities of shape. The most intuitive interpretation of that precondition is that the classes simply have to look very different even in low resolution, i.e., the classification problem must not be about subtle differences of shape.

If the preconditions are met and the problem requires that the root can process several images each second, the hypothesis claims that there is a good chance of a solution based on global features that are summations or other coarse statistics of local, possibly oriented features. For example, a subtype of solutions may use features resembling texture descriptors.

The hypothesis was inspired in part by recent research on the ability of humans to recognize everyday scenes very rapidly. Some of the research suggested that this ability could be explained by models using global statistics (e.g., histograms) of local oriented features (e.g., Gabor-filter responses). The evidence seemed to indicate that at least broad (superordinate) categories of scenes could be discriminated well.

The main experiment of this chapter satisfied the assumptions and claims of the hypothesis. The root could classify about 2.5 images per second given inexpensive hardware. A small collection of crude Gabor filters was used to compute global statistics (normalized sums of local filter responses) of images. Feature vectors

containing these statistics were then classified with SVMs and the results were measured.

Our training sets were small, but the classes were quite broad in the sense of having large variability of content. In hierarchic classification, root nodes are expected to see this kind of broad classes. Due to this, we paid special attention to the ability of the SVMs to tolerate changes of sampling bias after training. This kind of tolerance is also important when the classifiers can make decisions (movement) affecting the sampling bias. We changed the sampling bias in various ways, e.g., by changing lighting conditions, by training SVMs with subclass data only, and by omitting whole classes of negative inputs in training. The results were still good and satisfied the assumptions and claims of the hypothesis, suggesting that root nodes may not have to be trained with perfectly representative training sets given the current kind of crude features. In other words, the crude features may be incapable of noticing certain changes of sampling bias. Yet they work on the broad-class level because the classes are very distinct.

In the additional experiments we took a preliminary look at what can be done after a root node determines that a target belongs to a broad class. We examined the recognition of flat targets that can be seen as subclasses of signs that consist of lone letters glued on blank sheets of paper. The use of a rather elementary segmentation method, able to work within the broad class of these targets, allowed the targets to be recognized well.

We then examined a more difficult set of targets that were not flat. This time the classification results were not good. We contrasted the different results gained from the two sets of targets. The contrasted results and the characteristics of the extracted segments suggested that it would be hard to undo delegation errors without paying the full resource loss of a mistakenly selected specialist node.

CHAPTER 4

# Organizing delegation

In the previous chapter, root nodes were examined in isolation. The main question was how to prevent root nodes from becoming computational bottlenecks that could prevent a delegation system from achieving interesting trade-offs between accuracy and speed, i.e., trade-offs enabling several images to be classified per second. From now on, we take the system-level view, and do not examine nodes in isolation.

## 4.1 The main questions

In this chapter we ask two questions, the first of which is essential to determining how useful the delegation framework is in practice. The second question is less essential, but interesting for those who are willing to postpone classification until several images of an object are acquired. Answers to both questions are contributed. In addition, it will be shown that a particular test system satisfies the assumptions and claims of the root feature selection hypothesis (3.1.2).

### 4.1.1 The first question

The *first question* is that if one intends to build a classifier according to the framework, then where the organization of the base classifier nodes comes from. The organization of the nodes is the result of choices such as how many base classifiers are involved, whether

some classes are grouped together, and which classifier delegates to which. The organization was simply assumed to exist in Chapter 2. In Chapter 3 we examined what kind of features should be used in the root of the organization, and the additional experiments (Section 3.4) suggested that correcting delegation errors is hard.

### Background

It can be supposed that the difficulty of the question is proportional to the number of classes involved. With two-class problems, prior research justifies the assumption that *cascades* (see Chapter 2) are adequate. The assumption works for interesting efficiency-sensitive problems such as face detection, as seen by Viola and Jones [VJ01] and Elad, Hel-Or and Keshet [EHOK02]. In cascades, delegation may occur in the order determined by the complexities of the nodes.

When there are more than two classes, cascades are *not* adequate. Cascades do not readily allow the use of *specialist* nodes that are good at discriminating between classes only within some limited group or subset of classes. We use the terms *subset* and *group* somewhat interchangeably, because while the former term is correct technically, the latter conveys the idea that a subset is created according to some *grouping criterion*. The existence of specialists may result from the use of fairly specialized features or segmentation algorithms, such as those seen in the end of the previous chapter. In a cascade, a specialist would be positioned in the head (root), in the tail, or in the middle. The head may be excluded immediately, and the tail may be excluded if we consider the possibility of at least two distinct specialists. In the middle, specialists would be given inputs outside their specialty. Recalling that rejection seems unlikely to be cheap (Section 3.4), it seems that in the best of cases these inputs would be forwarded after the full resource loss of the specialist was paid.

When the number of classes increases, it may be expected that the difficulty of organization increases as well because the number of potential hierarchic class groups and trees is larger. For now, we take the approach that a tree reflects both a hierarchy of class groups and the runtime process of gradual exclusion of classes until predictions can be made confidently.

Based on the above, it seems that the question of organizing a

tree of classifier nodes in the context of the current framework is not simple, but is essential and worthy of study. Previous research has established that cascades work efficiently for problems involving two classes, but the results do not generalize to problems involving more classes.

### Two approaches to organization

Given that the question of organization is worthy of study, it is natural to ask *when* an organization of nodes should be designed, created, or learned from training data. More precisely, which comes first: the overall set of allowed features and image processing primitives, or the organization of the nodes?

If the features and image processing primitives are known first, then the requirement of encapsulation dictates that the features are divided into groups that become distinct feature spaces. These feature spaces may then be roughly ordered by complexity, i.e., how much time it takes to map an input image to a point in a feature space. This ordering should clearly offer some constraints for organizing a tree of nodes, each of which has to use at least one feature space. It seems possible in principle that the remaining unknowns of the organization could be learned from data using a *multi-cue decision tree learning* algorithm resembling that of Leibe and Schiele [LS03]. The algorithm would have to be modified, however, because their version optimizes accuracy only, and thus has no apparent means to avoid generating inefficient delegation paths that do not make sense in the current framework.

For brevity, we say that the process of Leibe and Schiele characterizes *the feature-centered approach*, in which an algorithm is given a set of features and image processing primitives plus a set of labeled images. The algorithm then generates the full classifier, including the nodes and their organization, without any additional input or human assistance. All mistakes are assumed equal.

Here in this chapter we take the approach that the organization of the nodes comes first, i.e., we attempt to build a tree-like hierarchy of nodes prior to determining what feature spaces and classification algorithms the nodes are allowed to use. For brevity, we call this *the organization-centered approach*. The term is *not* intended to mean that an optimal organization of the nodes somehow trivial-

izes the feature selection and node classifier learning problems. The approach, however, leads to constraints that are imposed on classifier learning, and these constraints are stricter than the constraints imposed by the feature-centered approach. In the learning of classifiers, constraints that limit the capacity of the learning algorithm to adapt to inputs may lead to a smaller risk of overfitting, and generally lead to smaller complexity terms in theoretical mistake loss bounds (see Equation (2.47) in Section 2.3.5 for an example of simple loss bounds).

### Organization and hierarchic mistake losses

The organization-centered approach may allow the organization of the nodes to reflect hierarchic mistake losses inherent to a classification problem. For example, it may be preferable to confuse a horse with a cow rather than with an apple because horses and cows are animals and have similar shapes from the point of view of the user, i.e., the point of view that should define the losses. Hierarchic mistake losses have been used in document classification problems, in which the number of classes is often large. For example, Dekel, Keshet and Singer [DKS04] use a tree of classes and path lengths within the tree to encode mistake losses and derive loss bounds.

Recall that in the delegation framework all inputs have a path through the tree of nodes. The possibility of backtracking is excluded because we do not expect benefits from that, e.g., the results in Section 3.4 suggested that efficient backtracking is difficult. Each non-leaf node is the root of a subtree, which is in turn associated with the subset of labels that the subtree can predict. When we follow a path, we essentially take subsets of these subsets until a single label remains and that label determines the most precise prediction. If hierarchic mistake losses are modeled, then the whole path should be considered to be the prediction. From the path we can take the smallest subset that contains the correct label, and the loss can be proportional to the size and content of that subset. In terms of the background theory, the subset is evaluated using a loss function that has the form $L_{(0+,1)}$ from Definition (2.2). For example, if the correct answer is *horse*, then the subset $\{horse, cow\}$ could be preferred over $\{horse, cow, dog\}$ and $\{horse, apple\}$. The subsets may, of course, be given descriptive names, e.g., *animals*.

With the organization-centered approach we have freedom to encode hierarchic losses in the organization of the tree prior to learning classifiers and choosing features. With the feature-centered approach, the learning algorithm responsible for creating the tree must be able to use the desired kind of hierarchic mistake loss from the onset. Otherwise, there is little reason to expect that the learned tree is good in terms of the loss. Typical algorithms for learning trees assume that all mistakes are equal, e.g., like the algorithm used by Leibe and Schiele, and are thus unable to use hierarchic losses.

It can also be claimed that if the learning algorithm begins to take hierarchic losses into account, the resulting approach is not purely feature-centered or organization-centered. A hierarchic loss essentially encodes a tree-like structure that can be made explicit by iteratively grouping together classes the confusion of which costs the least. Hence there is an organization that exists prior to and independent of feature selection, and that organization must determine the organization of the nodes to a large degree.

### Coarse-to-fine organization of classification

When one takes the approach that organization of the nodes is determined prior to feature selection, there are multiple ways to proceed. For example, if hierarchic mistake losses are known by numeric value, then one might create internal nodes by grouping together classes according to how much confusions cost within a group. The desired result would be that the internal nodes closest to the root distinguish between class groups the confusion of which costs the most, and that the nodes furthest from the root distinguish between smaller class groups the confusion of which costs the least. In other words, the desired result could be called *coarse-to-fine classification*.

The desirability of this result, *in the current framework*, is based on one assumption, which is that the mistake loss for confusing two classes is inversely proportional to the computational cost of a node that can distinguish them reliably. If the assumption holds, then the coarse-to-fine strategy of classification coincides with delegation proceeding from computationally lighter nodes to heavier nodes. Recall that this order of proceeding is desirable because heavy nodes

close to the root increase the classification times of *many* inputs.

The assumption is reasonable when the purpose of the system is to imitate human vision in everyday circumstances, i.e., minimize the number of errors that human users would find extremely stupid. In other words, the mistake losses would be smaller for "understandable" errors a normal person could make and larger for errors that tend to ruin the user experience. For example, if an image retrieval application retrieves an image of a parrot when asked to retrieve images of cars, then the user may abandon the application. The assumption is reasonable because it seems that classes that look "obviously" and utterly different to human observers can be distinguished with computationally cheap features. For example, in Chapter 3 we discussed research according to which the human ability to recognize broad classes of scenes could be explained with models that use coarse and quickly computable global features. On the other hand, if the differences between classes *within some subset* look subtle even to a human observer, then algorithmic discrimination may require careful segmentation for shape extraction and a large collection of prototype inputs for classification. It is *inefficient* to apply this careful segmentation if it is *unlikely* that the target belongs to such a subset. Segmentation may also fail without the appearance constraints implied by the subset membership (see Section 3.4).

In the experiments of this chapter we do not assume that the numeric values of hierarchic mistake losses are known *a priori* or that the classifiers we employ are capable of handling arbitrary loss functions. Rather, a hierarchy (tree) of classes is extracted using visual queries posed to human observers. The replies are subjected to statistical analysis that results in a tree being built. The tree represents a coarse-to-fine organization of classes, and the distance of two nodes in the tree is intended to be proportional to the perceived dissimilarity of the classes that the nodes represent. Hence, if we assume that confusing highly dissimilar classes implies larger loss, i.e., the error seems especially stupid to a human user, then the tree also represents a loss function that was not known *a priori*.

With the tree of classes organized in a coarse-to-fine manner, it can then be tested whether features can be selected and computational nodes can be placed on the tree so that delegation works.

Delegation is considered to work if the delegation mechanism enables reasonable trade-offs between accuracy and speed and there are no significant bottlenecks. Regarding root node bottlenecks, we also get to see that the results satisfy the assumptions and claims of the root feature selection hypothesis from Chapter 3.

The experimental results provide an answer to the first question of this chapter, i.e., where the organization of the nodes comes from. The results allow one to argue in favor of the *organization-centered approach*, i.e., organization is extracted prior to features. More precisely, it is demonstrated that the coarse-to-fine organization of classes works efficiently with the delegation framework, and such an organization can be extracted using elementary similarity queries that capture the insight of human observers.

### 4.1.2 The second question

The *second question* investigated in this chapter is whether predictions can be combined efficiently over multiple views of objects under motion. By combination over multiple views we mean that the classifier is shown a sequence of at least two different images of the same object, and the classifier then outputs a single prediction for the whole sequence. The classifier has to know, or deduce, whether the images in a sequence represent the same object, and it should be required that the sequence-specific predictions are better than individual predictions. The sequence-specific prediction is considered better than a sequence of individual predictions if the total loss over the sequence is smaller when the individually predicted labels are replaced with the predicted label of the sequence. The total loss includes both mistakes and resource losses.

It is necessary to emphasize that when multiple views are combined the resource losses *cannot* be directly proportional to real time measured from the instant the first image in the sequence is acquired to the instant the last image in the sequence has been processed. We cannot speed up the motion of objects and it would be pointless to acquire a sequence of nearly identical images at high frame rates. The resource losses can, however, be directly proportional to the time spent on *the classification task*. We assume that the time spent on classifying a sequence is distributed evenly over the time intervals between consecutive images of the sequence.

Hence, the less time is spent, the more time there is for the CPU to do something else between images. What else the CPU does is, of course, application dependent. For example, in robotic applications the CPU could concentrate on collision avoidance using sonar data.

The reason why the combination problem is interesting is that some individual views may be unrecognizable, or corrupted by noise. Because there are views of different quality, it is reasonable to investigate if there are combination strategies that reduce the total loss. In terms of efficiency, it seems best to prefer combination strategies that distribute the computational effort evenly over the time intervals between images. This minimizes the wait between acquiring the last image of the sequence and getting the prediction.

We investigate simple heuristic voting strategies for combining individual predictions. An individual prediction is represented as a ranked list of class labels. Each image is at first classified separately, and the resulting sequence of ranked lists is mapped to a single class label. This approach ensures that the classification effort is distributed evenly over time, but that each individual prediction provides more information than just a single label. Because the voting heuristics are computationally trivial compared to the classification of the individual images, the resource loss of voting may be considered zero. In addition, because individual predictions exist as lists of class labels, nothing prevents the classification of subsequences.

The empirical investigation provides an answer to the second question posed in this chapter, i.e., can predictions be combined efficiently over multiple views. The results allow one to argue that individual predictions can be combined efficiently using simple voting heuristics, the input of which comes from classifiers that use coarse-to-fine organization of classes and produce ranked lists of labels.

## 4.2 Overview of the experiments

The questions posed in this chapter should be answered empirically. The reason is that satisfactory theoretical answers are not apparent. To get empirical results, a test system is required.

We proceed in two stages. First we extract the organization, i.e.,

the tree of nodes. The tree is the basis of our delegation rules $\pi$ in terms of Section 2.1. Second, after the organization of the tree is extracted, the nodes are assigned features and classifiers. After training, the system and nodes behave as the generic tree model of Chapter 2, Section 2.1.3. Classification proceeds in a coarse-to-fine manner, narrowing down the subset of classes the target object is likely to belong to. Any node may, if confident enough, narrow the subset down to the size of one and terminate the delegation process. Hence, *both* internal nodes and leaves can terminate in contrast to decision trees.

The subsets of classes that the internal nodes represent are called *superclasses* and are given their own descriptive labels. These superclasses are analogous to *superordinate categories* that were discussed in Chapter 3 in the context of the root feature selection hypothesis. The reason is that the organization of the tree is extracted from queries measuring human similarity judgments. It is also demonstrated that the root node of the test system satisfies the assumptions and claims of the hypothesis, and the system has negligible bottlenecks.

In terms of Chapter 2, the fully trained test system gives a heuristic solution to *many* problems that are formulated as in Equations (2.6) and (2.11). The test system can do this because the realized trade-off between mistake losses ($ML$) and resource losses ($RL$) is controlled by a simple parameter that can be changed *after* training the system. Hence, if we change the problem by changing the desired weighting of $ML$ versus $RL$, we can change the solution accordingly without training again. The parameter affects the tendency of the node classifiers to overestimate their confidence, e.g., deliberate overconfidence may be chosen to decrease the consumption of time in exchange for an increased risk of mistakes.

Because the test system fits the generic tree model, for which related research was already overviewed in Section 2.2.2 of Chapter 2, we do not repeat that overview here. For example, axis-parallel decision trees [Qui93, Mit97, LS03] and perceptron trees [Utg89, MKS94, BM92, BM94a, BM94b, BFOS84] resemble the test system because they use delegation, but hierarchical mixtures of experts models [JJ94] do not because they do not use delegation. It was mentioned that Viola and Jones [JV03] have addressed the problem of multi-view face detection by dividing the class of

faces into pose-specific subclasses, which amounts to taking the *organization-centered* approach in terms of the current chapter, but their approach does not reach superordinate categories.

The test system requires a classification problem and data for learning and performance measurements. The first question requires a multi-class problem, and the larger the number of classes the better. The second question requires that object-specific sequences of views are available. In this study, a classification problem of eight (basic) classes was chosen, which seems to be large enough a number for non-trivial organizations to appear.

The publicly available dataset *ETH80* [1] was chosen. The dataset was used by Leibe and Schiele in their multi-cue decision tree experiments presented in [LS03]. The dataset allows the controlled rotation of objects, which means that sequences of views are available as required. By choosing this dataset we also allow *accuracy* comparisons between our results and those of Leibe and Schiele. Because they do not measure classification speed and because we did not duplicate their system, we cannot compare speeds. It is not clear if and how that system can be modified to make parameterized trade-offs between accuracy and speed.

Although the dataset is mostly satisfactory, it has one problem worth correcting. In the original dataset the problem of segmentation is abstracted away by the object images having almost featureless backgrounds and having perfect segmentation masks available *a priori*. The assumption that segmentation and recognition can be treated as independent subtasks is questionable. In Section 3.4.1 of Chapter 3 we touched on the subject of generic versus specialist segmentation and noted that the latter kind should be faster and more reliable. Even if we assumed that treating segmentation and recognition as independent problems was acceptable in the context of research focused on classification accuracy only, it is unacceptable in the context of efficient classification. Good segmentation costs time.

To correct the problem, a modified version of the dataset was created. In the modified version, the plain backgrounds were replaced by images depicting real-world environments. The *a priori*

[1] `http://www.vision.ethz.ch/projects/categorization/eth80-cropped256.tgz`

segmentation masks were then made unavailable to the algorithms. The classification results are reported for both the modified and the original datasets. In the case of the original dataset, we still refuse to consider the use of the segmentation masks in classification, fully aware that this could put us at a disadvantage when our results (accuracy) are compared to those of Leibe and Schiele.

The modifications also have a beneficial effect from the point of view of the root feature selection hypothesis. The replacement background of each object image is independent of the object class and object appearance. Hence, it is clear that the root node is recognizing objects *in spite of* the distracting backgrounds – not with the help of the backgrounds. In the previous chapter the hypothesis was examined using data in which the backgrounds may have helped. Therefore, the experiments in the current chapter complement those of the previous one. It can be demonstrated that helpful backgrounds are not necessary, i.e., the root does not need contextual features to succeed in object recognition at the superordinate level. In the real world, of course, objects are not independent of their backgrounds. The characteristic backgrounds may sometimes provide strong contextual cues for recognizing an object, or at least for excluding improbable objects.

## 4.3 Extracting hierarchic class relationships using visual queries

### 4.3.1 Class similarities and hierarchies

When there are more than two classes, some classes may appear to be more similar than others. More precisely, we may take an image (view) from class $A$ and note that a very similar image can be found in class $B$. If this happens with many images taken from class $A$, it can be argued that $A$ overall seems to be similar to some subset of $B$. If the relationship is symmetric, then the classes may be considered similar. Classes $A$ and $B$ are assumed mutually exclusive by definition, e.g., we may be comparing apples to oranges. These perceived similarities depend on the measure used. On one hand, while the measures used by human observers are unknown, they should not be assumed arbitrary, random or constantly changing.

On the other hand, it is unreasonable to assume that human beings use some unique and fixed measure [Pal99]. Here, we try to see what kind of similarity judgments can be extracted from typical human observers and we try to see if those judgments are consistent.

Because some pairs of classes may be more similar than other pairs, it is possible to construct groups and hierarchies based on similarity. For the sake of usefulness, some care should be taken regarding what kind of perceived similarities should be allowed. Perceived similarity is a blend of physically based *visual* similarity of the objects and other factors, such as associating the objects by *semantics* [Pal99, MR01]. By semantics we mean things that depend on the context, the known function of the objects, and details that are known but not visible in the images.

For example, suppose that a human observer is shown a few coarse but recognizable images of a bus, a sports car, and a train. Next, suppose that *after* the images are withdrawn from sight, we ask which pair was more similar – the bus and the sports car or the bus and the train. The observer may answer that the bus and the sports car were more similar, even if they had few visible features in common and the bus and the train shared an almost identical box-like profile. Potential reasons may include that the observer is comparing the objects (not necessarily the image instances) in memory, cars and buses are commonly seen traveling on roads (context and function), and both vehicles stand on rubber wheels (known detail, but not distinguishable in the images).

Trying to limit the impact of semantics seems reasonable when the primary purpose of the constructed hierarchy is to make classification efficient on a computer. *Visually* similar classes are grouped together because they may be assumed hard to tell apart using generic and fast features, i.e., the coarse shapes, colors, and textures may be the same. Hence, discriminating between visually similar classes may require specialist procedures, features, and more time. The specialist procedures may take advantage of the common within-group characteristics, i.e., specialized segmentation methods may be used for more precise segmentation that allows subtle aspects of shape to be used for discrimination. If the hierarchy is based on semantics, we may end up grouping together classes that are not at all difficult to tell apart using generic and fast features, and the resulting hierarchy cannot serve in efficient classification.

For the sake of intuitiveness, we assign labels to the class groups, i.e., subsets of classes that are extracted. These labels are like class labels with the difference that the group labels are not present in the original dataset. Given the labels the groups can be considered new classes that are not mutually exclusive with the original classes. The new classes are called *superclasses*.

A class is directly associated with at most one superclass, and superclasses may themselves have other superclasses. If a class is associated (linked) with a superclass, we say that the superclass *covers* the class. Covering is considered transitive. Covered classes may also be called subclasses. In a complete hierarchy (tree) superclasses become internal nodes. When the internal nodes are assigned classifiers, the superclasses are assigned possibly specialized procedures and features that are responsible for discriminating between the covered alternatives. Classification proceeds, in a coarse-to-fine manner, from predicting superclass labels to predicting precise labels.

When the basic classes, which are present in the original dataset, correspond to *basic level categories* of objects or scenes (recall Chapter 3), the superclasses correspond to *superordinate-level categories*. These terms are appropriate and the correspondences hold as long as the hierarchy is based on human similarity judgments. If a hierarchy is not based on human similarity judgments, classes and superclasses should be kept separate from basic and superordinate categories in order to avoid confusion with cognitive science.

The basic idea of our approach may be contrasted with previous research. Because the hierarchies are tree-like, there is a certain resemblance to decision trees [Qui93, Mit97]. The hierarchy of decision trees, however, is learned entirely from labeled multi-class data based on (surrogates of) 0/1 loss. The learning process assumes that a finite set of features is available *a priori*. The image pixels cannot serve as (trivial) features, because typical top-down learning of decision trees greedily assigns a single feature to a node, i.e., the resulting tree would simply test as many pixels as there are nodes on a path. With pixel features, the tree would have to be enormously complex to learn any image transformation invariances. Our tree, in contrast, is intended to be extracted from query responses *prior to* there being a set of non-trivial features available. The tree then represents *domain knowledge* that is used to

limit the degrees of freedom in classifier learning. Using the terms from the introduction of this chapter, we can say that decision trees take the *feature-centered approach* while our experiments take the *organization-centered approach*. In other words, a decision tree defines a solution to a classification problem, while our tree defines the structure of the problem which is the first step towards a solution.

### 4.3.2 A procedure for discovering a class hierarchy

At first, it could seem straightforward to simply dictate the superclasses and their labels based on the labels known to be present in the dataset. This amounts to abstract introspection, i.e., asking the question "What would I see in the data?" instead of "What do I see in the data?". On second thought, the classifiers will be trained using one portion of a specific dataset and evaluated using separate but similar data. It seems better to create superclasses that depend on the data, because these superclasses will have the potential to significantly affect the data-dependent learning process and subsequent evaluation. Our intent is to see if human similarity judgments can be used to form a hierarchy useful in classification – not to see if self-predicted similarity judgments coincide with the actual given a specific dataset.

Given the above, actual data has to be shown to test subjects. Because there are different views of different individual objects in different poses, showing a lot of data to several subjects seems preferable. Again, it could seem straightforward to simply ask the subjects to narrate what kind of similarities they remember seeing, or even ask them to draw a class hierarchy on paper. As noted earlier, this would amount to the subjects comparing the objects *in memory*, i.e., comparisons would not necessarily be limited to the views seen. We would also be unable to see details, such as the consistency of the judgments over different object instances and views of the same class. Further, we would not see if the judgments changed in the course of the experiment.

A specific procedure was designed for constructing class hierarchies. In executing the procedure, the test subjects are given visual similarity queries, each of which involves simple multiple choice. First, the procedure is designed to limit the influence of semantics. Each choice is made based on the views visible on a computer

screen, which should minimize the need to compare objects in memory. Second, the procedure is capable of revealing inconsistencies and changes in judgment. Each choice, from the first to the last, is recorded for analysis. Third, if mistake losses are inversely proportional to visual similarity, then the procedure may be seen as a means for extracting a reasonable (mistake) loss function from the test subjects, i.e., the relative severity of different errors.

In each query, the subject is presented with *two pairs of images* representing objects of interest from a dataset. The object instances are drawn randomly from *three* distinct random classes present in the original dataset. We denote a random triplet of classes by the labels of the classes, i.e., $(c_i, c_j, c_k)$ where $i \neq j$, $j \neq k$ and $i \neq k$. Using a random triplet of classes, we take one random object instance and view from each class of the triplet. We get a triplet of images, denoted $(I_{c_i,m}, I_{c_j,n}, I_{c_k,l})$, where $I_{c_i,m}$ denotes the $m$th image from class $c_i$ and the subscripts of the two other images $I$ are interpreted correspondingly. Two pairs, $(I_{c_i,m}, I_{c_j,n})$ and $(I_{c_j,n}, I_{c_k,l})$ are shown to the test subject. The two pairs have one image, $I_{c_j,n}$, in common to make visual comparisons meaningful.

When shown two pairs of images, the subject is queried and then rapidly chooses which one of the pairs is more similar. If uncertain, the subject can abstain. More precisely, a query has three possible outcomes, $sim(c_i, c_j) > sim(c_j, c_k)$, $sim(c_i, c_j) < sim(c_j, c_k)$, and *uncertain*, where $sim(c_i, c_j) > sim(c_j, c_k)$ denotes that an image of $c_j$ is more similar to an image of $c_i$ than to an image of $c_k$. We assume that $sim(c_i, c_j) = sim(c_j, c_i)$ for all $i$ and $j$. When the query process is repeated using different random triplets $(I_{c_i,m}, I_{c_j,n}, I_{c_k,l})$, we eventually get a frequency distribution of the three possible outcomes for each unordered pair of unordered class pairs $\{\{c_i, c_j\}, \{c_j, c_k\}\}$.

For each pair $\{\{c_i, c_j\}, \{c_j, c_k\}\}$, the distribution of the three outcomes is evaluated against the null hypothesis that the subject did not consistently favor any choice over the others. We use the $\chi^2$-score:

$$\chi^2 = \sum_{p=1}^{3} \frac{(f_p - e_p)^2}{e_p}, \tag{4.1}$$

where $f_p$ are the observed frequencies and $e_p$ are the expected frequencies of the outcomes. The expected frequencies are calculated

by assuming that the null hypothesis is true, i.e., the values $e_p$ are all equal because the uniform distribution is assumed. Note that the frequency distributions contain the unnormalized frequency counts. Sample size affects the significance of the score.

If the score (4.1) is significant at the 95% level of confidence, the null hypothesis is rejected, and the pair $\{\{c_i, c_j\}, \{c_j, c_k\}\}$ along with its distribution of outcomes are eligible for further processing. If a pair is eligible, we first check which one of the three outcomes dominates. If the outcome *uncertain* dominates, i.e., the frequency of this outcome is the largest of the three, then the pair is eliminated.

After elimination we have pairs of the type $\{\{c_i, c_j\}, \{c_j, c_k\}\}$ such that the null hypothesis was rejected for each pair and either $sim(c_i, c_j) > sim(c_j, c_k)$ or $sim(c_i, c_j) < sim(c_j, c_k)$ dominated the distribution of each pair. Each remaining distribution is then subjected to a binomial test of confidence, the purpose of which is to see if the dominating outcome is significant at the 95% level. The inequality relations of the dominating and significant outcomes are collected and used to create a set $\mathcal{S}$. If $sim(c_i, c_j) > sim(c_j, c_k)$ is dominating and significant, we put the ordered pair $(\{c_i, c_j\}, \{c_j, c_k\})$ in $\mathcal{S}$. If $sim(c_i, c_j) < sim(c_j, c_k)$ is dominating and significant, we put the ordered pair $(\{c_k, c_j\}, \{c_j, c_i\})$ in $\mathcal{S}$.

Having the set $\mathcal{S}$, we use the ordered pairs to build a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The vertices in $\mathcal{V}$ are the unordered pairs of classes, e.g., $\{c_i, c_j\}$, from $\mathcal{S}$. Each of the directed edges in $\mathcal{E}$ points from a more similar pair of classes towards a less similar pair of classes. More precisely,

$$\mathcal{V} = \{\{c_i, c_j\} \in \mathit{left}(\mathcal{S}) \cup \mathit{right}(\mathcal{S})\}, \tag{4.2}$$

$$\mathcal{E} = \{(\{c_i, c_j\}, \{c_j, c_k\}) \in \mathcal{S}\}, \tag{4.3}$$

where the sets $\mathit{left}(\mathcal{S})$, $\mathit{right}(\mathcal{S})$ denote the sets of unordered pairs appearing on the left- and right-hand sides of the ordered pairs in $\mathcal{S}$. For example, if $(\{c_i, c_j\}, \{c_j, c_k\}) \in \mathcal{S}$, then $\{c_i, c_j\} \in \mathit{left}(\mathcal{S})$ and $\{c_j, c_k\} \in \mathit{right}(\mathcal{S})$. Note that an edge always connects two pairs of classes such that the pairs have one class in common.

Using $\mathcal{G}$ as a constraint, we estimate a simple similarity mapping,

$$sim : \mathcal{V} \to \mathbf{R} \cup \{NaN\},$$

such that *sim* indicates class similarity in a manner that is consistent with the significant outcomes of the queries. Here, the term $NaN$ (Not a Number) is included to indicate the possibility that a proper numeric value has not yet been assigned to some pair $v = \{c_i, c_j\} \in \mathcal{V}$. The mapping *sim* is constructed by an iterative algorithm. In the algorithm, we use the shorthand $Parents(v)$, which is defined as $Parents(v) = \{w \in \mathcal{V} | (w, v) \in E\}$. Initially, we assign $sim(v) = NaN$ for every $v \in \mathcal{V}$. The algorithm is as follows:

1. Select one vertex $v \in \mathcal{V}$ such that currently $sim(v) = NaN$ and all $w \in Parents(v)$ have already been assigned a value, i.e., $sim(w) \neq NaN$. If no such $v$ exists, then stop.

2. If $Parents(v)$ is not empty, assign

$$sim(v) \leftarrow \min\{sim(w) | w \in Parents(v)\} - 1.$$

   Else, assign $sim(v) \leftarrow |\mathcal{V}| - 1$.

3. Repeat from step 1.

It is easy to see that the above algorithm stops if $\mathcal{G}$ has no directed cycles. The key observation is that every finite directed acyclic graph has at least one *source*, i.e., a vertex that does not have incoming edges. If the algorithm could stop before assigning each vertex a value, then $\mathcal{G}$ would have a proper non-empty subgraph such that each vertex in the subgraph would have $sim(v) = NaN$ and at least one parent in the same subgraph. When cut off from $\mathcal{G}$, the subgraph would be a directed acyclic graph without any sources – a contradiction. If $\mathcal{G}$ is acyclic, the algorithm stops after each $v \in \mathcal{V}$ has been assigned a number. If directed cycles are detected in $\mathcal{G}$, then the algorithm should not be run in the first place, because the queried subjects clearly have non-random but inconsistent views of similarity. Possible causes may include that the subjects have revised their perceptions of similarity in the course of the experiment.

After stopping, the values of $sim(v)$ are normalized to the interval $[0, 1]$. Finally, an undirected graph $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$ is constructed so that

$$\mathcal{V}_2 = \{c_i \in Classes\}, \tag{4.4}$$

$$\mathcal{E}_2 = \{\{c_i, c_j\} | sim(c_i, c_j) > Th\}, \tag{4.5}$$

where *Classes* denotes the original classes and the connected components of the graph define the superclasses for the given threshold value of *Th*. Two vertices are in the same connected component if and only if there is a path of edges between them.

By increasing the threshold *Th*, the connected components become smaller and fewer original classes are covered by the superclasses. Hence, a tree-like hierarchy is discovered by starting from a small *Th* and then increasing *Th* until the connected components have just one vertex each, at which point the leaves of the tree are discovered. Examples related to the discovery process can be found in Section 4.5.2 and Figure 4.6.

## 4.4 Classification

After a hierarchy has been extracted and encoded as a tree, we attach a classification system to the tree. The nodes of the tree must be assigned features and classifiers, and the delegation rules must be specified.

In terms of Chapter 2, the classification system $prog_f$ implementing the classification function $f$ consists of base classifiers $prog_{f,k}$, $k \in \{1, \ldots, M\}$, delegation rules $\pi$, and regions $B_k$ of the image space. From (2.7), recall that the $k$th base classifier declares it is confident in classifying the input image $\mathbf{x}$ if and only if $\mathbf{x} \in B_k$. Internally, a base classifier may use any suitable feature mappings and similarity metrics.

Each input image $\mathbf{x}$ goes first to the root base classifier ($prog_{f,1}$) of the system. If $\mathbf{x} \notin B_1$, the input is delegated down the tree to specialist base classifiers. Each specialist $prog_{f,k}$ (non-root, $k \neq 1$) activated by the input $\mathbf{x}$ may in turn declare lack of confidence (if $\mathbf{x} \notin B_k$), in which case $\mathbf{x}$ is delegated further down the tree to sub-specialists. In principle, inputs may be delegated to *any* sub-specialist that is further down – not just those that are immediately below the currently activated base classifier. Delegation is irreversible, i.e., inputs are never delegated up towards the root, because delegation errors are considered hard to detect and correct (recall Section 3.4).

### 4.4.1 Assigning classifiers to nodes

The root and each internal node of the tree are assigned base classifiers, each of which includes a feature space and a classifier. Recall (Chapter 2) that a base classifier does its own preprocessing and the computational cost, i.e., the resource loss, of a base classifier includes the cost of computing the feature values as well as classification. Each such node also corresponds directly to a superclass that covers at least two distinct classes present in the unmodified problem data, i.e., classes that correspond to leaf nodes. Base classifiers are not assigned to leaf nodes because the classes that correspond to leaf nodes have no subclasses.

For brevity, we call the internal nodes near the root *the top nodes*. Correspondingly, the internal nodes near the leaves are called *the bottom nodes*. In general, delegation proceeds from the computationally lighter top nodes towards the heavier bottom nodes. This order of proceeding is desirable because heavy top nodes would increase the classification times of many inputs. Because classification proceeds in a coarse-to-fine manner, excluding classes that are not covered by the superclass that corresponds to the activated node, the bottom nodes discriminate between fewer classes than the top nodes. In other words, the bottom nodes specialize on narrow subproblems.

Because the bottom nodes are allowed to consume more time and they solve subproblems, it is reasonable to expect that the bottom nodes should make fewer mistakes than the top nodes. In addition, if the organization of the tree reflects hierarchic mistake losses in the manner of [DKS04], the mistakes of bottom nodes may be less serious.

In our model, each base classifier is trained separately using a sufficient subset of the training set. The sufficient subset is determined by the superclass to which the node of the base classifier corresponds to. An input in the training set is in the sufficient subset if and only if the class of the input is covered by the superclass. For example, the root node superclass covers every class transitively and the sufficient subset of the root is the whole training set. Each input in a sufficient subset is augmented so that the labels of the covered superclasses of the input are explicitly present in training and may be predicted after training. For example, an input $(\mathbf{x}, dog)$

in the sufficient subset of the root becomes $(\mathbf{x}, \{dog, animal\})$ if *animal* is the only superclass of *dog* covered by the root. For later use, we say that the labels in the sufficient subset are *visible* to the base classifier.

Let the $k$th base classifier $prog_{f,k}$ compute the function

$$sign(f_k(\mathbf{S}_k(\mathbf{x}), l)), \tag{4.6}$$

where $\mathbf{S}_k(\mathbf{x})$ is a node-specific feature extraction function of possibly high computational complexity, and $l$ denotes a class label. If the sign is positive, then $\mathbf{x}$ is predicted to have the label $l$. Because explicit superclass predictions are used in guiding delegation, $\mathbf{x}$ may be predicted to have multiple labels that are not mutually exclusive.

Abbreviating $\mathbf{z} = \mathbf{S}_k(\mathbf{x})$, note that $f_k(\mathbf{z}, l)$ is slightly more general than the form $f_k(\mathbf{z})$ that was common in Chapter 2. Because not all labels are mutually exclusive, each base classifier must return a set of labels instead of a single label. The values of $sign(f_k(\mathbf{z}, l))$ are interpreted as label indicators, i.e., the set of predicted labels consists of the labels $l$ for which $sign(f_k(\mathbf{z}, l)) = +1$. Our approach also requires that the predicted labels in the set can be ordered by confidence.

It is interesting to consider whether a collection of binary (two-class) classifiers can be combined into a base classifier that can return a set of labels as a prediction. Recall that the classifier combination methods from Section 2.3.2 can combine binary classifiers into a single multi-class classifier, i.e., $f_k(\mathbf{z}, l)$ over different $l$ can be combined. Using the winner-takes-all scheme or the voting scheme over one-vs-one tournaments does not, however, lead to the desired kind of multi-class classifier. These schemes are designed for mutually exclusive classes only, i.e., in the winner-takes-all scheme there is a single winning label. The schemes would require non-trivial modifications.

When the classes cannot be considered mutually exclusive, it is possible to use one-vs-rest training to produce one classifier for each $l$ such that the classifier calculates the $l$th indicator $sign(f_k(\mathbf{z}, l))$ as if it was independent of other indicators. Because our approach requires that the predicted labels can be ordered by confidence, the values $f_k(\mathbf{z}, l)$ over different $l$ would have to be mapped to values the comparison of which would be meaningful. Mappings can be

created using confidence models. Earlier, we have discussed both monotonic and non-monotonic confidence models.

Some binary classifier combination methods can be used to produce two-class classifiers that have confidence models, i.e., different $f_k(\mathbf{z}, l)$ with fixed $l$ can be combined. For example, if non-monotonic confidence models are desired, the maximum voting margin classifier from Section 2.3.4 can be tried. Each of the combined component classifiers could be trained using one-vs-rest training, i.e., the class $l$ should be pitted against the incompatible classes. Different maximum voting margin classifiers could then be combined over $l$.

For simplicity, we assume monotonic confidence models (Section 2.3.3) in the current chapter. More precisely, we assume that confidence is proportional to the value of $|f_k(\mathbf{z}, l)|$. Also note that both monotonic and non-monotonic models discussed here are compatible with large-margin classifiers that are simpler than SVMs. For additional simplicity, boosted stumps [Sch02] are used in the root. A large-margin interpretation of boosting can be found in [SFBL97].

### 4.4.2 Prediction and delegation

After the nodes have been assigned base classifiers capable of predicting ordered sets of labels, system-level prediction and delegation follows simple rules. Here, we present how the $k$th base classifier node operates in relation to the system.

Assume $L^{(k)} = (l_1, \ldots, l_{n_k})$ is the confidence-ordered list of the $n_k$ labels visible to the $k$th node such that

$$f_k(\mathbf{z}, l_j) \geq f_k(\mathbf{z}, l_{j+1}) > 0.$$

Now, beginning with $k = 1$ (the root), the system makes predictions and delegates inputs using Algorithm 1.

First, note that Algorithm 1 takes a *fixed* threshold value $T$ as a parameter. The parameter $T$ is a system-level parameter that is not seen by the base classifiers during their training, i.e., changing the value of $T$ does not affect training results. In practical applications a fixed value of $T$ could be problematic as finding the optimal value could require trial and error. We, however, are interested in examining how efficiently and accurately the extracted coarse-to-fine organization of classes works with *varying* values of $T$. For

**Algorithm 1** Node $k$

**Require:** list $L^{(k)}$, positive threshold $T$ (e.g., 0.8)
  take $l_1$, the head of $L^{(k)}$ ($l_1 = null$, if the list is empty);
  **if** $l_1 = null$ **then**
    $choice \leftarrow none$;
  **else if** $l_1$ has no subclasses **then**
    $choice \leftarrow l_1$;
  **else**
    $l_1$ has subclasses; let $L'$ be the sublist of $L^{(k)}$ such that we take from $L^{(k)}$ only the classes covered (directly or indirectly) by $l_1$ in the same order they appear in $L^{(k)}$;
    let $l'_1$ denote the head of $L'$ ($l'_1 = null$, if the list is empty);
    **if** $l'_1 = null$ **then**
      $choice \leftarrow l_1$;
    **else**
      let $R = f_k(\mathbf{z}, l'_1)/f_k(\mathbf{z}, l_1)$;
      **if** $R > T$ **then**
        $choice \leftarrow l'_1$;
      **else**
        $choice \leftarrow l_1$;
      **end if**
    **end if**
  **end if**
  **if** *choice* has no subclasses **then**
    predict *choice*;
  **else**
    direct the current input to the specialist node of *choice*, i.e., $f_{k'}$, and delegate responsibility for the prediction to that specialist;
    restart this algorithm with the list $L^{(k')}$;
  **end if**

example, we want to know if changing $T$ allows satisfactory control of speed versus accuracy trade-offs. If it does not, asking for optimal $T$ is premature. Finding an optimal $T$, given some specific user preferences, is not our primary concern here. Still, it should be noted that changing the value of $T$ does not require much effort. Because the value does not affect the training of base classifiers, the value can be changed without retraining the classifiers.

To understand Algorithm 1, first suppose that $T = 1$, which is the maximally conservative threshold with respect to classification accuracy. Because $l'_1$, if it exists, is not the head of the list $L^{(k)}$ ($l_1$ is), it follows that $R \leq 1$, and we choose $l_1$, unless $L^{(k)}$ is empty. The above process reduces to the node always choosing the label with the largest positive magnitude. Such labels tend to be superclass labels, because the whole purpose of the visible superclasses is to group together alternatives that are hard to distinguish from each other by this node. Hence, the end result is that we must activate a specialist unless a particular input is so easy that confidence of a superclass does not exceed the confidence of the subclasses.

Now consider what happens when $0 < T < 1$. The node becomes *optimistic* in the sense that the specialists further down are assumed increasingly unnecessary the lower threshold $T$ is. Suppose that $L'$ is not empty and $l'_1$ is the head. Being careful, we choose $l_1$ and delegate to the specialist $k'$ of $l_1$ immediately, like we did in the case $T = 1$, and examine list $L^{(k')}$ of the specialist node. If this specialist $k'$ is completely unnecessary, it does not disagree with $L'$ and we observe that $L' = L^{(k')}$. Supposing that the specialist is conservative ($T = 1$), it chooses the head of list $L^{(k')}$, which is $l'_1$. Hence, delegation to the specialist was unnecessary. Supposing the node takes the superclass $l_1$ for granted, accepting a low $R$ amounts to "jumping to conclusions" about the subclass. Even if the subclass turns out to be wrong, the correctness of the superclass is not affected.

### 4.4.3 Multiple views of objects under motion

Object motion is in many ways an important source of information. Studies of human perception in infants indicate that motion cues have a crucial role in perceiving object unity in early stages of development [JCMJ03].

In the tests we have in mind, we have complex objects on complex backgrounds, and segmentation seems a formidable challenge. Further, from certain viewpoints, objects belonging to different mutually exclusive classes may seem indistinguishable. For example, viewing an apple and a pear from below, both have similar shape as the elongated form of the pear is not visible. To address these issues, we extend our approach to the case in which a moving object can be tracked to provide a sequence of views.

Let $tr = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ be a *visual trace* of an object under apparent motion. For simplicity, we assume that the individual images $\mathbf{x}_i$ are known to depict the same individual object. To enable simple motion segmentation (e.g., difference imaging [SHB99]), it is preferable to have fairly rigid objects and short time intervals between the consecutive images. This is the approach we take.

First, each still image $\mathbf{x}_i$ is classified by the tree of nodes as in Section 4.4.2. However, this time we pay special attention to the ordered lists $L^{(k)}$ calculated by the activated nodes $k$. Let $L_i^{(1)}$ denote the ordered list of the root computed for the view $\mathbf{x}_i$. The list is re-ordered using Algorithm 2.

**Algorithm 2** Re-order

**Require:** $i$ and lists $L_i^{(k)}$ of the nodes $k$ activated by $\mathbf{x}_i$
  move the choice of the root node to the head of $L_i^{(1)}$
  (if $T < 1$, the choice may not be there initially)
  **if** specialist nodes below the root were activated **then**
    **for all** activated nodes $k$ in the order of activation **do**
      let $L_i^{(k)}$ be the ordered list of activated node $k$;
      move the choice of the $k$th node to the head of $L_i^{(k)}$;
      move the labels of $L_i^{(k)}$ to the head of $L_i^{(1)}$ in the order specified by $L_i^{(k)}$ (labels already present in $L_i^{(1)}$ lose their original places in $L_i^{(1)}$);
    **end for**
    remove superclass labels from $L_i^{(1)}$, leaving only labels of classes that have no subclasses;
  **end if**

Algorithm 2 may be used to enable *rank-order voting* over $\mathbf{x}_i$ in a visual trace (see [Arr70] for a treatise on voting schemes). The final

re-ordered list $L_i^{(1)}$ at the root has the following desirable properties:

1. The final prediction of the tree for $\mathbf{x}_i$ is at the head of the list.

2. The alternatives ranked by a specialist are preferred over others originally ranked below the specialist superclass label. Specialist preferences overrule more general preferences in a consistent manner.

The visual trace $tr$ of an object can now be classified as a whole if the lists $L_i^{(1)}$ for different values of $i$ are transformed to votes, which are then subjected to a voting system. The choice of a voting system is best settled experimentally.

In our experiments (detailed later), we examined two simple alternative voting systems. In *plurality voting*, each view $\mathbf{x}_i$ casts a single vote in favor of the head of list $L_i^{(1)}$. The trace $tr$ gets the label with the most votes over $i$. In *rank-order voting*, each view $\mathbf{x}_i$ casts several votes. Given a fixed number of alternatives $A$, the label at the $j$th position in list $L_i^{(1)}$ gets $A - j + 1$ votes (not negative) from $\mathbf{x}_i$. Finally, votes are counted over $i$ and $tr$ gets the label with the most votes.

## 4.5 Experiments

Having presented the overall design, we now present specific experiments using a particular set of data and a test system. First, we discuss the details of the data used in the experiments. Second, we show what kind of a class tree or trees could be extracted from the data and visual queries using the procedure from Section 4.3.2. Third, we show what kind of base classifiers could be assigned to the nodes so that the test system works as designed in Sections 4.4.1 and 4.4.2. Fourth, we present the results of the experiment.

### 4.5.1 The data

For the experiments we chose the publicly available dataset *ETH80* [2]. The dataset was used previously by Leibe and Schiele [LS03] in their experiments related to multi-cue decision tree learning.

The dataset contains color images of 80 objects from 8 basic level categories. The categories are *apple*, *car*, *cow*, *cup*, *dog*, *horse*, *pear* and *tomato*. From each basic level category there are 10 different objects. Each object is represented by 41 different views. The views of an object are distributed uniformly over all viewing angles such that the observer is not below the object. Hence, the views allow the controlled rotation of the objects. Figure 4.1 shows the 41 views of one particular object instance from the category labeled *dog*. On the average the categories had high within-category variation of content. Figure 4.2 shows ten different horses viewed from one viewing angle. As can be seen, the surface colors, textures, and body poses (shape) vary within the category of horses.

Although the dataset is mostly satisfactory, e.g., there are many categories (classes) and the variation between categories and within categories is high, there is one problem worth correcting. In the data the problem of segmentation is abstracted away by the images having pre-made segmentation masks available. The segmentation masks are accurate and flawless. Even if the segmentation masks were not available, the simple and rather uniform backgrounds (Figures 4.1 and 4.2) could make efficient segmentation relatively easy. If we used the pre-made masks or other simplifying assumptions in classification, we would likely get efficient solutions to toy problems.

We replaced the image backgrounds. After replacing the image backgrounds, we discarded the pre-made segmentation masks. As replacement backgrounds we used random patches cropped from images in the Benchathlon 2001 set [3]. The Benchathlon set is meant for benchmarking content-based image retrieval programs and contains realistic images depicting everyday environments. For each object, we randomized one background image, i.e., the 41 views of an object each got the same background, but different objects

[2] `http://www.vision.ethz.ch/projects/categorization/eth80-cropped256.tgz`

[3] `http://www.benchathlon.net/img/todo/index.html`

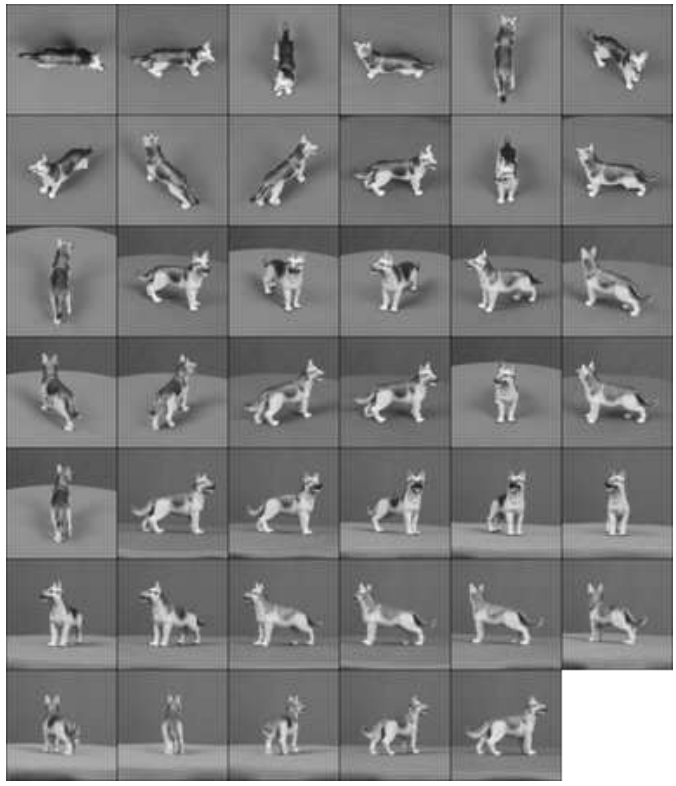


Figure 4.1: A dog viewed from 41 different angles. The backgrounds are from the original data. In this image, the colors have been discarded and the individual views have been downsampled to $64 \times 64$ pixels to allow them to be printed here.

got different backgrounds. The results of this replacement were examined to ensure that no artificial contours were created. Figure 4.3 shows the ten horses from Figure 4.2 on their replacement backgrounds. Figure 4.4 shows additional examples of the modified data we used.

The replacement process had the interesting characteristic that it made the objects statistically independent of their new backgrounds. The *leave-one-object-out* cross-validation process that we used (detailed later) ensured that the test sets did not contain backgrounds present in the training sets. In testing, the image backgrounds were both novel and independent of the object class.

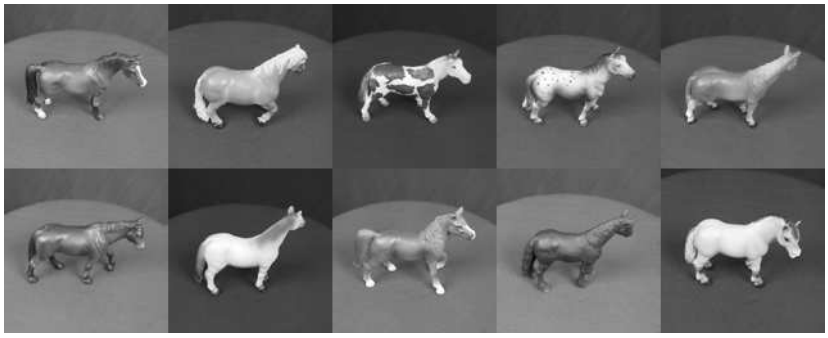


Figure 4.2: Ten different horses viewed from one viewing angle. The individual views have been downsampled to $128 \times 128$ pixels.

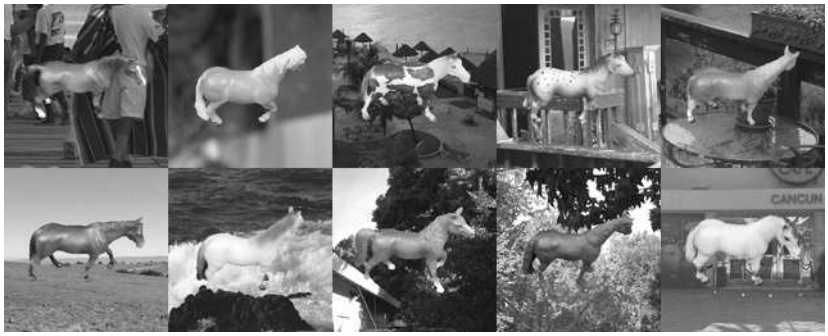


Figure 4.3: Ten different horses against replacement backgrounds.

Hence, the classifiers cannot successfully use image backgrounds to predict object class. It follows that the success of a particular classifier demonstrates that the classifier does not need helpful backgrounds to work. This is especially interesting in the context of the root feature selection hypothesis.

Because the views of an object can be ordered as if the camera was circling the object, we can extract visual traces, i.e., sequences of views, that look like the object was rotating on top of a real-world environment.

Figure 4.4: Examples of the modified data. In the rightmost subimage it is easy to see why segmentation is hard.

### 4.5.2 The hierarchy

We applied the discovery procedure (Section 4.3.2) combining the answers of two subjects. Figure 4.5 shows an example of a visual similarity query. The queries were done with the original data, i.e., the object backgrounds were simple.

The results indicated that there were two class pairs of maximal similarity: (*apple*, *tomato*) and (*cow*, *horse*). Only three other class pairs had large similarity ratings: (*apple*, *pear*), (*cow*, *dog*) and (*horse*, *dog*). The similarity between *car* and *cup* was minimal, suggesting that their shared non-organic look and the property of being man-made items did not matter.

The results were mostly compatible with the idea that similarity judgments are as if determined by the basic shape of the objects. For example, when viewed from similar angles, cows, horses and dogs all have roughly the same shape although precise bodily proportions vary. The results were not completely compatible, however. Apples and pears had high similarity although from most viewing angles the former are round and the latter are conical. Somewhat surprisingly, tomatoes and pears were not similar, even if tomatoes and apples were. One possible explanation is that the similarity between apples and pears was of semantic origin, e.g., both are fruits. Yet another possible explanation is that perceived visual similarity relations do not have to be transitive, e.g., if apples and pears are similar and apples and tomatoes are similar, then tomatoes and pears do not have to be similar.

Figure 4.6 shows two alternative hierarchies from using the discovery process explained in Section 4.3.2. Favoring simplicity, we

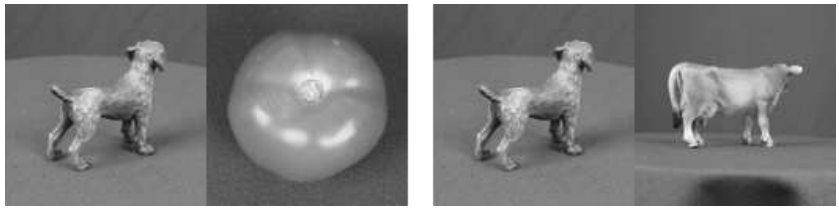


Figure 4.5: A similarity query between two pairs (simple backgrounds).

chose Alternative 1 that was found by increasing the parameter *Th* in Equation (4.5) in large steps. Alternative 2 was found by taking smaller steps. We named the superclasses of Alternative 1 *fruits and vegetables* and *animals*. If we had more different types of fruits and animals, then these superclasses could possibly become too abstract. In that case we would have to use a deeper hierarchy possibly resembling Alternative 2.

### 4.5.3 The classifier nodes

#### The root classifier node

The results of the hierarchy discovery process were compatible with the idea that similarity was mostly determined by the basic shape of the objects. In this, the results could be interpreted to suggest that the root node should measure and classify the coarse overall shape of the objects. If the coarse shape was consistent with a particular superclass, then a specialist classifier could be activated to do precise classification.

The above scheme has the problem that classifying the overall shape, however coarse, in the root node seems to require something resembling segmentation at a stage when nothing is known about the class of the input, e.g., inefficient generic segmentation could be required. We took the pragmatic approach and avoided shape extraction in the root node.

Our root node uses global features that are statistics of local feature responses over low-resolution input images. The spatial relations between local features are discarded. More precisely, the root uses *rotation-invariant uniform local binary patterns* (LBP)

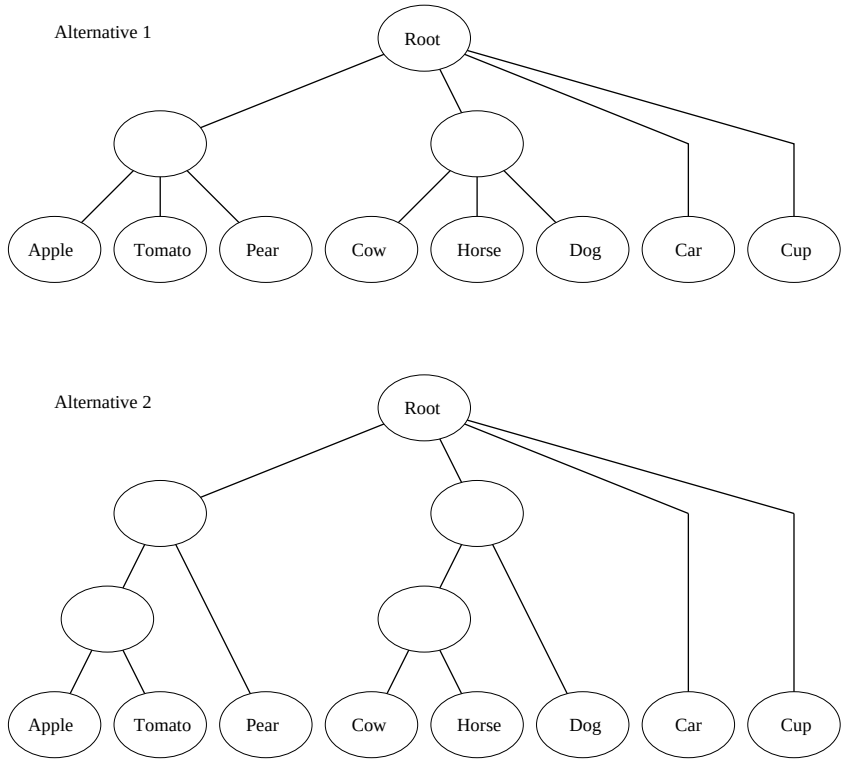



Figure 4.6: Two alternative class relationship hierarchies.

and *color histograms*.

Local binary patterns [OPM02, OPM01] are a computationally efficient family of features designed for fast extraction. Each pattern measures the local brightness differences between a center pixel and the surrounding pixels at the radius $R$ from the center. The patterns are binary, because only the sign of the differences is retained. Rotation invariance is achieved by rotating each observed pattern to an orientation determined by the contents of the pattern, but not affected by the original orientation. Uniform patterns are fundamental patterns of special significance that function as templates for structures such as bright spots, flat areas, dark spots, and edges of varying curvature.

The root uses the operator $LBP^{riu2}_{8,1}$, meaning the 8 neighbors at the radius $R = 1$ are used for computing the differences for every

center pixel. The operator produces $8 + 2$ distinct output values as explained in [OPM02]. A histogram of the 10 output values is calculated over the focus region of attention. The root computes separate histograms for three image resolutions: $128 \times 128$, $64 \times 64$, and $32 \times 32$ pixels. We also limit the focus to regions of apparent motion, as detected by difference imaging. Difference imaging is applied to the visual traces of the objects we have available. This allows much of the image background to be excluded without using real segmentation, i.e., object boundaries do not resemble the border of the excluded background.

Figure 4.7 shows the LBP histograms of three objects from different superclasses. The three histograms of each object (input) have been concatenated, i.e., bins 1 to 10 correspond to the resolution of $128 \times 128$ pixels, bins 11 to 20 correspond to the resolution of $64 \times 64$ pixels, and bins 21 to 30 correspond to the resolution of $32 \times 32$ pixels. In each case we used side-views of the objects as inputs. One particular pattern, visible at three scales (bars 9,19, and 29), was often prominent. This was not caused by the replacement of image backgrounds or the use of focus regions of attention. We observed that the same pattern was equally prominent if the backgrounds were not replaced and the focus regions were not used. In the figure, the apple (from the superclass of fruits and vegetables) is clearly distinguishable from the car (root superclass) and the horse (from the superclass of animals). Given the histograms of the specific side-views, the car and the horse do not seem to be that clearly distinguishable. Later, we show that the features do well statistically.

In addition to the LBP histograms, the root computes coarse color histograms of the inputs. There are 10 bins for each color channel (red, green, and blue). The resolution of the inputs for this operation is $128 \times 128$ pixels. Pixels outside the focus region of attention are ignored. The focus region is the same that is used with LBPs.

The local binary patterns may be seen as sensible substitutes for Gabor filters. Recalling Section 3.3.2, suppose that an image is convolved with a Gabor filter. The value of a pixel in the convolved image is the weighted sum of a neighborhood of pixels from the original image. Because pixel values are bounded and the filters use Gaussian envelopes, the weights of pixels beyond a certain radius

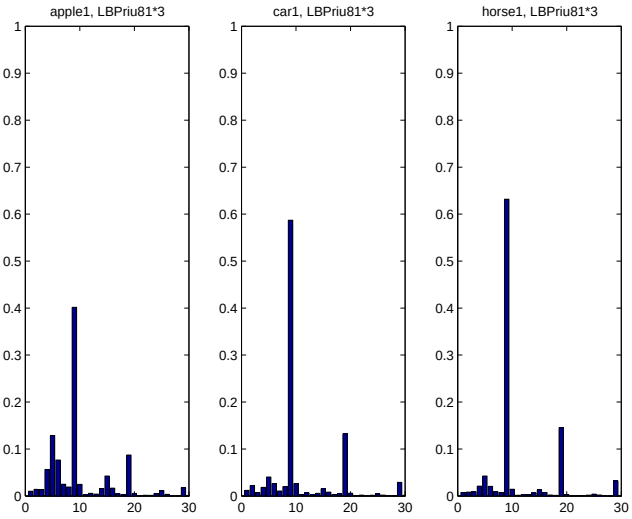



Figure 4.7: Examples of LBP histograms. Here, each visible histogram is the concatenation of three histograms that are computed at different scales.

approach zero. Where the weights are significant, the positive and negative weights form a wave-like pattern in some orientation. The output value of a filter is high when the image matches the pattern locally. The filter measures local intensity contrasts and the outputs may be thresholded if we prefer binary pattern matching. The LBPs also measure contrasts in local neighborhoods, but there is less freedom because the contrasts are taken with respect to the center and the local neighborhoods must be small (or the image resolution must be decreased). The restrictions lead to ease of use because the small number of possible patterns allows the exhaustive enumeration of the patterns that make sense as templates, i.e., the uniform patterns mentioned above.

The root classifier is given the concatenated histograms as 60-dimensional feature vectors, i.e., the three LBP histograms and the color histogram are not kept separate. The root classifier satisfies the assumptions of the root feature selection hypothesis (Section 3.1.2). The inputs are in low resolution, i.e., less than $150 \times 150$ pixels, when the features are computed. The classification problem, with or without superclasses, is trivially solvable by ordinary hu-

man observers. The root can make multiple predictions per second (detailed later), uses the right kind of global features, and discriminates between classes (from different superclasses) at above change level (detailed later).

The 60-dimensional feature vectors are used in boosting decision stumps. To do this we use *real AdaboostMH*, which is a multi-label boosting variant invented by Schapire and Singer [SS00, SS99]. It satisfies the confidence-estimation requirements elaborated earlier, i.e., the classifier output magnitudes are related to confidence.

The boosted hypothesis has the form

$$f(\mathbf{x}, l) = \sum_{t=1}^{Tmax} \alpha_t h_t(\mathbf{S}(\mathbf{x}), l), \tag{4.7}$$

where $h_t$ are the decision stumps that take the input $\mathbf{x}$ and the label $l$ and produce real-valued outputs. The choice of the weights $\alpha_t$ is related to minimizing the empirical *Hamming loss* of $f$ (see [SS00, SS99] for details). The function $\mathbf{S}$ is the feature extraction function that maps the input $\mathbf{x}$ into the 60-dimensional feature vector. We used $Tmax = 200$ as the number of stumps, although a smaller number of stumps could possibly suffice. Taking the sign of $f$, we get the proper form compatible with definition (4.6) in Section 4.4.1.

### The specialist nodes

Assigning a base classifier to the node specializing in the classification of different animals was a difficult subproblem. The different animals had roughly the same body shape. Details of the body surface, e.g., colors and textures, did not seem promising for the kind of accurate classification that specialist nodes do. For example, Figure 4.2 shows that the within-class variation of the details is high. Of the features considered, reasonably detailed measures of body shape seemed the most promising.

In the implementation, the specialist classifier of animals first uses the focus regions of apparent motion to extract the outer contours of the objects. The contours are then matched to known prototypes that are collected from training data.

The focus regions of apparent motion are used as follows. First, edges are detected in the focus region, after which morphological

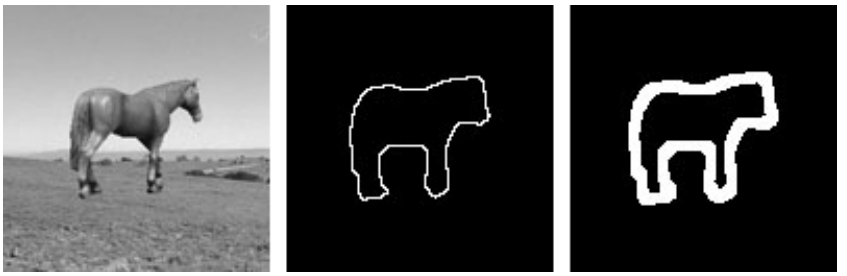


Figure 4.8: The contours of a horse, as seen by the classifier specializing in discriminating animals.

closing [SHB99] is used to gap the discontinuities of the outer contour. In essence, we do segmentation within the focus regions. Second, having the refined shape of the object at hand, the specialist uses Hausdorff-fraction-based nearest-neighbor search to find the closest matching animal shape from the training data prototypes (see [AM02] for the definition of the fractions and modern refinements). Because sub-specialists were not needed, the specialist simply selects the single best class. If sub-specialists were involved, a ranked list of preferences could be formed by ordering the alternatives by match-value.

Figure 4.8 shows the contours of an animal as seen by the specialist. The middle image shows the basic contour, which is of relatively good quality. The outline is not broken, and the shape and bodily proportions are recognizable. Many imperfections are visible as well. In the contour, the concave (inward curving) parts have been partially lost. For example, the neck of the animal has thickened, the stomach has grown downwards, and the middle of the back has become straight. The rightmost image shows the dilated version of the middle image. When Hausdorff-fractions are used to match shapes, the basic contour (middle) of one input is matched against the dilated contour (right) of some other input. The distance between inputs is inversely proportional to the fraction of contour pixels of one input falling within the dilated region of the other input. For symmetry, the measurement can be repeated after the roles of the inputs have been switched. For each pair of inputs we take the maximum of the two distance measurements.

Designing a specialist for fruits and vegetables was rather simple. Simple color-based segmentation yielded the shapes and dominant colors, based on which simple heuristic rules provided near-perfect results. For example, only tomatoes have an orange hue, and only pears have elongated shapes.

### 4.5.4 Classification results

The test system was evaluated using *leave-one-object-out* cross-validation. As noted by Leibe and Schiele [LS03], this type of cross-validation is preferable when we have to evaluate how well the system generalizes to new object instances – not just new views of objects in the training data.

The node classifiers were trained using views of 79 of the 80 objects after which we tested classification performance using the 41 views of the one object that was left out. This process was repeated 80 times, using each object in turn as the test object. The statistics shown in Table 4.1 are the averages over the 80 repetitions.

The process makes sure that the test object is novel for the classifiers that have seen only the training data. Further, the replacement backgrounds of the test objects were novel and independent of the objects. The common background of the 41 test views was not used in the training views. Therefore, any learner trying to exploit the backgrounds should be worse off than learners that do not.

In addition to the statistics from the use of the modified data, we provide comparable statistics that were calculated using the original data (i.e., using the original image backgrounds). The latter statistics are shown in Table 4.2. With the original data, the difference imaging technique for getting the focus regions of attention was a bit excessive and somewhat inappropriate (e.g., the objects do not rotate on top of the backgrounds). Hence, we enabled a trivial alternative for getting focus regions. We simply filter out of focus homogeneous parts of the image periphery. The alternative filtering technique is like difference imaging in the sense that it is not intended as an estimator of object boundaries.

In the Tables 4.1 and 4.2, the first two rows show the relative usage frequencies of the specialists. We use *fv* to denote the fruits and vegetables specialist, and *ani* to denote animals specialist. The relative usage frequency of a specialist is the relative frequency of

activating the correct specialist given that the true class is covered by the superclass that the specialist corresponds to. For example, if *fv use* is 0.0496, then about five percent of the different fruits and vegetables were classified using the corresponding specialist, while the rest were classified directly by the root or misclassified by another specialist.

From the third row down, the rows of the tables summarize classification accuracies at the level of the most specific labels. For example, the row *ani acc* shows the average accuracy of recognizing *cows*, *dogs*, and *horses* correctly using these specific labels. All columns, except the last two, show averages over views. The first column shows the case of disabled specialists – the root node classifies everything and delegates nothing. The middle columns show the results from the specialists enabled with varying values of $T$ (Section 4.4.2). Recall that $T$ controls the tendency of avoiding specialist use in favor of speed, and that the value of $T$ can be changed without retraining the classifiers (e.g., specialists). The last two columns show the averaged results from the use of voting procedures in classifying visual traces instead of single views. In testing voting procedures, we set $T = 0.8$. In rank-order voting, a small constant bias was added to Equation (4.7) to allow some votes to be given even to unlikely alternatives.

Having explained how Tables 4.1 and 4.2 should be read, we will now interpret the numbers. This is mostly about the accuracy of the classifiers. After dealing with classifier accuracy, we will present measured results related to speed and then interpret those results as well.

Examining the first two columns of Table 4.1, we see that using the specialists increases the overall mean accuracy even if $T$ has a small value. Increasing $T$ increases the mean accuracy simultaneously increasing the usage frequencies of the specialists, which in turn increases the expected classification time (resource loss). Comparing the mean accuracy at $T = 0.8$ to the best multi-cue results of Leibe and Schiele [LS03], we observe that when classifying single views, our results are within two percent of the cited results (0.9302). This is satisfactory, given that we did not use the perfect segmentation masks that Leibe and Schiele used. The lack of pre-made segmentation masks, and the use of complex replacement backgrounds, made the efficient use of contours rather difficult for

Table 4.1: Classification results.

| | dis | T=.05 | T=0.1 | T=0.2 | T=0.4 | T=0.8 | plur | rank |
|---|---|---|---|---|---|---|---|---|
| fv use | 0 | 0.0496 | 0.0553 | 0.0748 | 0.1398 | 0.4390 | – | – |
| ani use | 0 | 0.3455 | 0.3902 | 0.4854 | 0.6439 | 0.8098 | – | – |
| fv acc | 0.9724 | 0.9886 | 0.9886 | 0.9886 | 0.9846 | 0.9846 | – | – |
| ani acc | 0.5659 | 0.7431 | 0.7626 | 0.8008 | 0.8358 | 0.8610 | – | – |
| rest acc | 0.8963 | 0.8768 | 0.8768 | 0.8768 | 0.8768 | 0.8768 | – | – |
| mean acc | 0.8009 | 0.8686 | 0.8759 | 0.8902 | 0.9018 | 0.9113 | 0.9375 | 0.9750 |

Table 4.2: Classification results (original data).

| | dis | T=.05 | T=0.1 | T=0.2 | T=0.4 | T=0.8 | plur | rank |
|---|---|---|---|---|---|---|---|---|
| fv use | 0 | 0.0423 | 0.0528 | 0.0748 | 0.1618 | 0.5553 | – | – |
| ani use | 0 | 0.3000 | 0.3724 | 0.5089 | 0.7228 | 0.8943 | – | – |
| fv acc | 0.9748 | 0.9927 | 0.9935 | 0.9943 | 0.9935 | 0.9894 | – | – |
| ani acc | 0.6512 | 0.7854 | 0.7967 | 0.8301 | 0.8423 | 0.8374 | – | – |
| rest acc | 0.9146 | 0.8988 | 0.8988 | 0.8988 | 0.8988 | 0.8988 | – | – |
| mean acc | 0.8384 | 0.8915 | 0.8960 | 0.9088 | 0.9131 | 0.9098 | 0.9375 | 0.9500 |

us. As noted by Leibe and Schiele, the best multi-cue results they got were largely due to the use of contours. Hence, they succeeded in exploiting the masks they made available.

Enabling the voting procedures lets us exceed the cited results by a reasonable margin. The main reason seems to be that multi-view integration of predictions is necessary in avoiding mistakes resulting from bad viewing angles. For example, discriminating between the rear ends of a horse and a cow may be rather difficult even to a discerning observer.

Looking at Table 4.2, observe that when classifying single views, we are again within two percent of the best results of Leibe and Schiele (column $T = 0.4$). It might be possible to close the two-percent gap by fine-tuning our test system or by starting to use the segmentation masks instead of extracting contours the hard way. Neither of the alternatives to closing the gap is interesting.

Comparing Table 4.2 to Table 4.1, it can be seen that the accuracy statistics change less when $T$ is varied within the given range. From the first column, it is still apparent that disabling the specialists results in significant loss of accuracy. Comparing the tables it also seems that when the specialists are disabled, the root node benefits from improved focus region quality that the simple backgrounds allow. Although accuracy with animals is still poor, it is better than earlier.

Having dealt with classifier accuracy, we now present measured results related to speed. In Figure 4.9 we show the usage frequency of the animal specialist versus the average time spent classifying images of animals. The $x$-axis shows the frequency in the range $[0.0, 1.0]$ and the $y$-axis shows the time spent in seconds. For example, when the frequency was 0.3, classification took 0.5 seconds on the average. The circles denote the sample points of the plot, i.e., the number of sample points equals the number of observed animal specialist usage frequencies in Table 4.1 plus Table 4.2. Two of the sample points coincide when the frequency is zero. The averages represented by the sample points were calculated using images of animals that were classified either directly by the root or by the animal specialist. In other words, images of animals that were delegated to the wrong specialist are not factored in the averages. The times were measured using a fairly average AMD 3500 computer.

According to the measurements, the root node required about

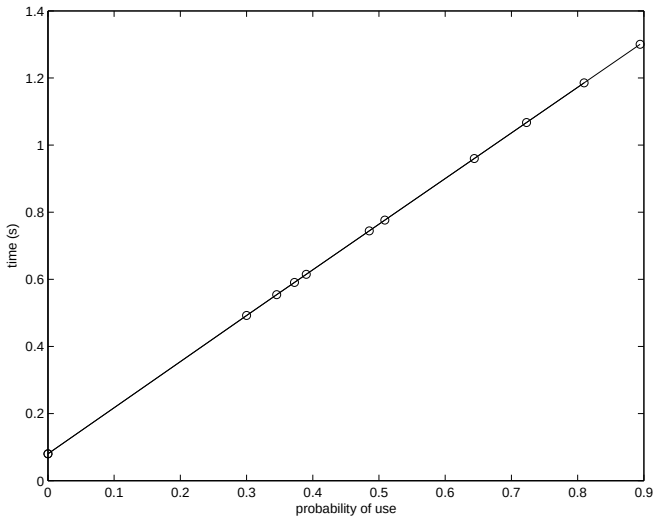



Figure 4.9: Animal specialist usage frequency versus average classification time.

0.08 seconds per input image, and could therefore process and classify images at 12.5 Hertz. This seems considerably faster than what we could achieve with the small set of Gabor filters in the previous chapter, even if we take the different downsampling requirements into account. Compared to the root and the other specialist, the animal specialist operated on a different timescale, i.e., the time consumption of the animal specialist made the other nodes seem insignificant. The animal specialist required about 1.36 seconds per activation. Average classification times were in practice mostly determined by how frequently the animal specialist is activated. If we assume (for simplicity) that the probability of being delegated to a wrong specialist is (close to) zero, then the empirical mean classification time $t_{animal}$ of animals is

$$t_{animal} = t_{root} + P(specialist)t_{specialist} \approx P(specialist) \times 1.5s, \tag{4.8}$$

where $t_{root}$ denotes the time used by the root, $P(specialist)$ denotes the probability of delegating to the animal specialist, and $t_{specialist}$ denotes the time used by the animal specialist.

Above, we presented results related to accuracy and speed sep-

arately. We can now tie the results together. In Figure 4.10 we show mean accuracy (from Table 4.1) as the function of mean classification time. The marked points denote different values of $T$. Mean classification times were measured separately for each value of $T$. To do this we followed each test input and recorded which specialists the input activated and how much time this took. We then calculated the mean time over the inputs. It can be seen that speed can be traded for accuracy. At about 0.35 seconds there is a point of diminishing returns. The curve shows no indication of undesirable behavior, e.g., there are no points at which increasing time would result in decreasing accuracy.

In Figure 4.11 we show both the mean time and mean accuracy as functions of the parameter $T$. Both time and accuracy seem to be roughly logarithmic in $T$. Comparing Figures 4.10 and 4.11 it can be seen that to get approximately linear trade-offs we must resort to doubling $T$. In other words, we can define a more intuitive control parameter $T'$ such that

$$T = 2^{T'}/5. \tag{4.9}$$

In any case, the consequences of changing $T$ (or $T'$) are quite predictable. Hence, using $T$ to control delegation allows *reasonable control* of speed versus accuracy trade-offs. Furthermore, there are no significant bottlenecks. Setting $T = 0$ (column *dis* in 4.1) prevents delegation. At $T = 0$ the test system has non-trivial accuracy and classifies inputs at 12.5 Hertz.

The results also indicate that predictions can be combined efficiently over multiple views. First note that Tables 4.1 and 4.2 show that rank-order voting is accurate. When the mistake losses are divided over views, rank-order voting results in smaller loss per image compared to classifying individual views. The time (resource) loss per view, however, is exactly the same as when classifying individual views. Hence, the sum of total losses is smaller. Of course, if inputs arrive fast and have to be processed quickly, e.g., at 3 Hertz, then spending over 0.5 seconds per view ($T = 0.8$) results in huge resource losses even if combining views is still more efficient. When necessary, the additional accuracy from voting should be traded for speed.

Last, we can examine what can be seen if mistakes are not considered equal. Table 4.1 shows that when delegation is disabled

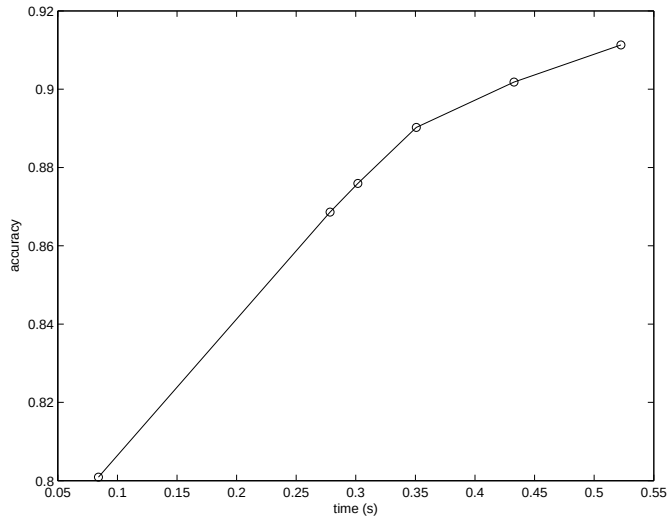



Figure 4.10: Mean time versus mean accuracy.

($T = 0$), mean accuracy is 0.8009. Using the confusion matrices generated during the tests, we calculated that the mean accuracy of the root node was 0.9445 when within-superclass mistakes were ignored completely. For example, confusing a cow with a dog was ignored, but confusing a cow with a pear was not. Similarly, we calculated that the mean accuracy of the root was 0.8727 when within-superclass mistakes were discounted by 50%. If we used *any* hierarchic mistake loss such that within-superclass mistakes would cost less, then the test system would appear to be better than Table 4.1 indicates. Most of the mistakes were within-superclass mistakes.

Ignoring the within-superclass mistakes showed that the root can discriminate well between classes that belong to different superclasses. Given that the root can also process multiple inputs per second, the results satisfy the claims of the root feature selection hypothesis. Earlier, we explained that the assumptions were also satisfied.

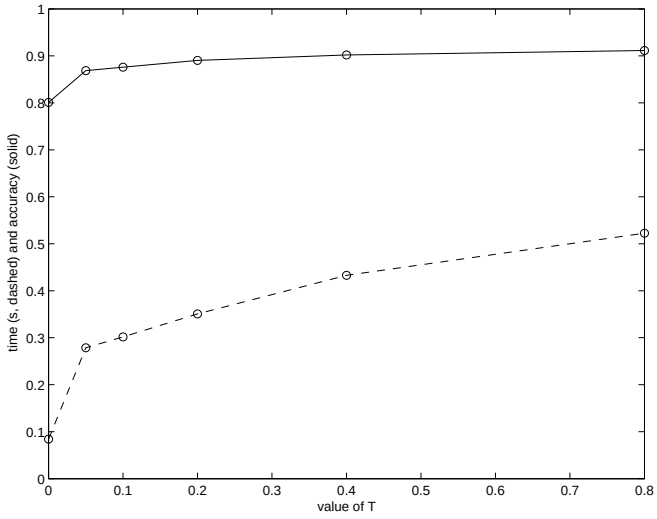



Figure 4.11: Parameter $T$ versus mean time and mean accuracy.

## 4.6 Summary

In this chapter we investigated two questions. The first question asked where the organization of the base classifier nodes comes from. The second question asked whether predictions can be combined efficiently over multiple views of objects under motion. The questions were investigated empirically using a test system.

In the task of finding the organization of the base classifiers, the *organization-centered* approach was tried. This approach could be contrasted with the alternative that we called *feature-centered*. In the organization-centered approach, the organization (tree) of classifiers is determined prior to feature selection and classifier learning. In contrast, in the feature-centered approach the features are defined first and the organization is learned using the computed feature values from labeled training data.

On a more detailed level, the approach that we called *coarse-to-fine* organization of classes was tried, with explicit superclasses and subclasses of objects. In coarse-to-fine classification, it is assumed that confusing broad (super) classes is inherently more harmful than confusing narrow (sub) classes. Further, classification and delegation proceed from the root towards the leaves so that the nodes

closest to the root distinguish between broad classes and nodes closest to the leaves distinguish between narrow classes.

An experimental procedure was designed. The procedure could produce coarse-to-fine trees of classes using visual queries posed to human observers. The idea was that by the controlled measurement of what classes the observers find similar, we can construct a tree in which the distance of (any) two nodes is proportional to perceived dissimilarity, which in turn can be assumed proportional to the cost of mistakes. Users are not likely to tolerate the confusion of objects that seem to be very dissimilar to them.

For the test system, delegation rules were designed. The delegation rules were designed so that a specific parameter $T$ controls the tendency to terminate delegation early. In contrast to typical decision trees, any node may terminate and make the final prediction. In addition, rules were devised for combining predictions. The purpose of these rules was to enable the investigation of the second question that was posed, i.e., can predictions be combined efficiently over multiple views resulting from motion? The procedure of visual queries was used, the nodes of the resulting tree were assigned features and base classifiers, and measurements of performance were taken.

The results imply that the organization-centered approach, and more precisely, the coarse-to-fine approach led to efficient classification of single views and multiple views.

First, in classifying single views the results indicated that avoiding early termination of delegation increased prediction accuracy as measured on the level of narrow classes, i.e., the specialists behaved as specialists should. In contrast, favoring early termination increased speed. Increasing accuracy decreased speed and vice versa. Although there was a point of diminishing returns, spending more time always resulted in better accuracy. No efficiency bottlenecks were found and the delegation parameter $T$ allowed reasonable control of speed versus accuracy trade-offs. Using $T$, the test system was made to go to different extremes. Given a high value, the system became accurate. The best accuracies were close to the published results of Leibe and Schiele [LS03]. That result was found satisfactory given that the perfect segmentation masks of Leibe and Schiele were not used by the test system in classification. Given a low value of $T$, the system became quite fast. Twelve images could

be classified each second. This range of allowed trade-offs cannot be considered restrictive. Because the value of $T$ can be changed without retraining the classifiers, very different preferences could be satisfied without much effort.

Second, in classifying multiple views the results indicated that predictions could be combined efficiently. Voting rules resulted in increased accuracy with no adverse effects on time spent in computing. Hence, voting was efficient in the sense of making the sum of total losses over sequences smaller.

Third, the results indicated that the test system would have been competent with hierarchic mistake losses. Most mistakes were within-superclass mistakes. If it is accepted that the procedure of visual queries extracts a tree such that membership in a superclass implies high similarity and high similarity implies that confusing the precise class label is tolerated, then most mistakes would have been tolerated.

Fourth, the results indicated that the root node, which satisfied the assumptions of the root feature selection hypothesis, also satisfied the claims of the hypothesis. This was interesting because the object backgrounds were unhelpful in contrast to the data used in the previous chapter. Hence, the results complemented those from the previous chapter.

Overall, the success of the test system can be counted as evidence in favor of using the organization-centered coarse-to-fine approach for finding the organization of the base classifiers and for combining predictions efficiently via voting. Because very different speed versus accuracy preferences could be satisfied by changing the value of $T$, it cannot be claimed that the evidence is trivial, i.e., specific to few preferences. Also, the data was non-trivial and one cannot just take an arbitrary approach and expect to succeed.

Finally, we note potentially interesting topics that we did not address. First, there is the topic of scaling. In the test problem, a shallow hierarchy of classes was sufficient. It would require larger problems with considerably more classes to test how well the chosen delegation and voting subsystems scale up. Some datasets, such as COIL100, advertise a high number of classes, but these classes are too narrow, e.g., each object instance is considered a separate class.

Second, there is the topic of comparing the organization-centered and feature-centered approaches. This should be considered to-

gether with the topic of scaling. Because Leibe and Schiele did not measure speeds and we did not implement their program, we could not compare speeds. It is not clear how their program could be modified to allow the control of trade-offs with or without retraining. Hence, using their design to represent the feature-centered approach in systematic comparisons could be difficult. If scaling is considered, the feature-centered approach may also become difficult. Difficulties arise if the learning procedure has to do exhaustive search over the possible features (or sets of features) when a node is created. It would be interesting to see if this search could be made less exhaustive by using speed-related constraints, i.e., requiring that computationally light nodes delegate to heavier nodes.

CHAPTER 5

# Questions of modularity

In this chapter we ask three questions related to the delegation framework. The questions address some of the assumptions that were present in the introduction of the previous chapter. Answers to all the questions are contributed.

In the introduction of the previous chapter, we contrasted two alternative approaches to organizing base classifiers so that a tree is formed. The two alternatives were called *the feature-centered approach* and *the organization-centered approach*. The alternatives had the common characteristic that the end result, i.e., the hierarchy expressed as a tree, was *problem-specific*. In other words, a hierarchy created for one classification problem is expected to be useless given some other problem. This problem-specificity is an assumption that may be accepted without much thought. After all, delegation trees are reminiscent of decision trees, which are in general constructed so that the hierarchy and connections of the nodes are problem-specific.

## 5.1 The main questions

### 5.1.1 The first question

As the *first question* we ask if the delegation framework requires problem-specific organization of the nodes. The question is clearly worthy of study because if organization does not have to be customized for each problem, then using the framework becomes sim-

pler. Recall that in the previous chapter we had *specialist* nodes each of which specialized in discriminating between classes in some subset of all classes. Some subsets were *privileged* in the sense that they were given specialists while other subsets were ignored. The approach of Leibe and Schiele [LS03] was similar to ours in that each node was responsible for a specific subset of classes and most subsets were ignored. If an approach to organization is such that some minority of subsets becomes privileged, then it is reasonable to expect that the selection of the minority, and thus organization, depends on the problem.

To answer the first question, we design, analyze, implement, and test an alternative approach to organization in which *all* non-empty subsets of object classes are privileged and given specialists. In this approach all problems that have $m$ object classes get the same organization, i.e., one root node and $2^m - 1$ specialists. Each specialist is directly connected to the root and there are no connections between specialists. In what follows, the approach is called *A2 multi-class delegation rules*. Obviously, we want to keep some of the interesting characteristics seen in the previous chapter, e.g., the possibility of adjusting speed versus accuracy trade-offs without retraining classifiers.

Recalling that in the previous chapter each specialist was customized and trained individually, the idea of using $2^m - 1$ specialists may seem impractical. The A2 multi-class delegation rules bypass this problem by forming the specialists dynamically at runtime. When an input requires delegation to a specialist, the specialist is formed from a set of simpler *modules*. After the input has been processed, the specialist is discarded. Given $m$ object classes, there are $m$ modules capable of participating. The modules are called *robust detection modules*. A set of modules that defines a specialist has from 1 to $m$ modules. The $m$ modules are trained individually, but the specialists are not.

### 5.1.2 The second question

As the *second question* we ask if the framework is limited to controlling the efficiency of classification, or if it is also useful in improving the efficiency of feature selection and training. More precisely, we ask if the *root* can improve the efficiency of feature selection and

training of *specialists*. As usual, efficiency means less computation and saving time.

The question is not as narrow as it may first seem. The apparently obvious way to increase the efficiency of the training process would be to filter the training inputs. Because feature selection and machine learning algorithms tend to require an amount of time proportional to the number of training inputs, it may seem that filtering out the inessential inputs is the appropriate solution. Basically, the root node could perhaps filter the training inputs of the modules that form the specialists. This process would resemble what decision trees and the cascades of Viola and Jones [VJ01] do, i.e, subsequent nodes are trained using those inputs that pass through earlier nodes. The idea of filtering, however, is not fully compatible with one goal that we have already stated. The goal was that we should be able to adjust speed versus accuracy trade-offs without retraining classifiers. The adjustments determine which inputs pass through which nodes. Hence, during training we cannot really decide that a particular input will never pass through the root.

Given the above problem with filtering, it seems that alternative means for increasing the efficiency of training are worthy of study. We focus on the use of spatial constraints in feature selection. This is relevant when the features measure local image properties. The idea is that spatial constraints improve efficiency by predicting where a feature selector *should not* look for candidate features. For example, input images may contain background in addition to objects, and it may make sense to avoid considering features characteristic of the background. More precisely, the root node contains modules that produce constraints for the feature selectors of the modules that form the specialists.

### 5.1.3 The third question

As the *third question* we ask if pre-made and weak classifier modules can be combined *adaptively* to work together in specialist nodes. By a weak classifier, we mean a classifier that is not sufficiently accurate when used alone. The lack of sufficient accuracy justifies combination to improve accuracy. By adaptive combination, we mean that classifiers in some set are combined to form a better classifier

in a manner that involves learning or task-specific optimization as opposed to using some fixed rules. Fixed rules, e.g., the A2 multi-class delegation rules examined in the first section of this chapter, may be simpler to analyze, but lack the ability to detect poorly functioning modules and adjust the influence of such modules on decisions. In contrast to the commonly used boosting approach, we focus on the use of pre-made modules.

In boosting [SFBL97], there is a central algorithm that builds a set of base classifiers to be combined. The algorithm builds the base classifiers sequentially, so that the $n$th base classifier is built to complement the abilities of the $n-1$ base classifiers built earlier. In boosting, it is not trivially possible to use pre-made classifiers that are not built by the central algorithm. In reality, however, one may have pre-made classifiers that should not be discarded as irrelevant or worthless. For example, several researchers may work independently and generate a set of insufficiently accurate classifiers. Hence, the problem of combining pre-made classifiers is worthy of study.

To answer the third question, we design, implement, and test a method with which pre-made classifier modules are combined adaptively to create specialist nodes. The adaptive mechanism uses a non-monotonic model of confidence. Non-monotonic models were discussed earlier in Section 2.3.4 of Chapter 2. Because such models are not in common use, they may raise some doubt. Hence, we perform experiments with both monotonic and non-monotonic models to compare the results.

## 5.2 Overview of the experiments

The three questions above basically ask whether certain specific things can be done in the context of the framework. Because our intention is to show that these things can be done, we must show how and measure the degree of success, i.e., experiments and test data are required.

In the context of the first and second questions, we use a certain kind of local features, called *image fragments*. This kind of features are known to lead to accurate classification results, as shown by Ullman, Vidal-Naquet, and Sali [UVNS02, VNU03]. Unfortunately,

fragment features are also very inefficient to use – both during classification and feature selection. Therefore, it is interesting to see if the use of fragments can be made more efficient by using the delegation framework. Because fragments are learned from training data and the number of potential candidates is enormous, the use of fragments allows that different specialist nodes use different sets of features as is characteristic of the delegation framework.

Although fragments are interesting, they are not, at least in their current form, suitable for all classification problems. In short, they are suitable for detecting or recognizing semi-rigid objects. Fortunately, there are interesting problems and datasets that have such objects. In our experiments related to the first and second questions, we measure performance on tasks related to the detection of cars and faces versus miscellaneous backgrounds. In the case of multi-class detection, we have two object classes, i.e., the cars and faces, and one *non-object* class for representing the miscellaneous backgrounds. The details and the reasons for choosing particular datasets are presented later.

In the context of the third question, we use an entirely different kind of data. To understand why, it is important to understand the question. The question emerges when there are several pre-made classifiers available and they are found to be too weak individually. For example, one may build several tentative solutions to a problem and find none of the solutions especially promising. Because current classifier algorithms tend to be quite capable, the reason for the weakness is likely the insufficient quality of the input features. The features are of insufficient quality when the designer, given a classification problem, is unable to come up with more suitable features (computationally efficient or not). Hence, the car and face detection problems and datasets mentioned above are incompatible with the third question.

To present a compelling case, it is necessary to present a classification problem and an associated dataset such that the problem is important and finding sufficiently good features is known to be hard. We attack the problem of detecting retinal microaneurysms from retinal fundus images. Because microaneurysms are indicators of a disease that is both common and causes blindness, the problem cannot be dismissed as contrived or unimportant. The details of the chosen dataset and the relevant background literature

(e.g., the nature of microaneurysms and the disease) are examined in the latter half of the current chapter that is dedicated to the third question.

## 5.3 Attention-driven object detection

### 5.3.1 Test system design

In this study we examine the first two questions raised in the introduction of the current chapter. In the context of the questions, we analyze and test two kinds of delegation rules. We begin with rules for two-class problems and then proceed to rules for multi-class problems. Both require the implementation of test systems for measuring performance. In the two-class case the systems are plain two-stage cascades. In the multi-class case we have a two-level tree, the structure of which resembles the tree that was used in Chapter 4, i.e., we have a root node and specialist nodes to which inputs can be delegated. If the root cannot classify an input directly, it selects a subset of classes and activates a specialist to perform finer discrimination. As earlier, there are threshold parameters that may be used to control speed versus accuracy trade-offs without re-training the classifiers.

In the multi-class case, there are also some noteworthy differences compared to the earlier test system. Because demonstrating the basic technique does not require a large number of classes, we use just three – two object classes and one *no object* class. There is one specialist for each non-empty subset of *object* classes. Hence, in our experiments there are three specialists. Each specialist specializes in one or two object classes plus the *no object* class. In contrast to earlier design, the specialists are not created *a priori*, but are formed dynamically by combining simpler *modules* when the root sees an input that requires delegation.

The nodes of each cascade and tree are formed from linear two-class SVM modules that are, when necessary, given monotonic confidence models (recall Section 2.3.3 in Chapter 2). Each module encapsulates a feature space in addition to the SVM. We use two kinds of modules, *attention controller modules* (ACMs) and *robust detection modules* (RDMs). The division is essentially based on the

type of features used. As may be guessed from the name, RDMs use quite strong (robust) features to compensate the limitations of linear SVMs with respect to accuracy. In contrast, the ACMs use very weak features as inputs to linear SVMs, which would obviously result in poor accuracy if the ACMs were used in isolation. The ACMs, however, are faster than RDMs by orders of magnitude. Their purpose is to control the *attention* of RDMs.

When we say that ACMs control attention, we mean that they are involved in two tasks. First, they are used for determining which RDMs participate in forming a specialist when an input requires delegation to a specialist, i.e., the ACMs choose which RDMs pay attention to the input. The root node consists of ACMs only. Second, each ACM is paired with one RDM for the purpose of selecting the features which the RDM uses. The ACM limits the attention of the RDM feature selector to specific image locations. Because there are plenty of potential RDM features and evaluating the suitability of each is expensive, such limitations are useful.

In addition to the above, we examine the role of low-frequency (LF) information in classifying images. In the current study, we were inspired by the recent work of Bar [Bar03] that examines the question of how biological vision systems can classify familiar scenes and objects very fast. Bar argues that it is possible for a higher-level visual component to receive a low-frequency (LF) representation of the image (i.e. a blurred image) to use in fast decision making. We blur the input images using different filters and then test the effects on prediction accuracy. In this dissertation, these limited LF experiments are superseded by the root feature selection hypothesis (Section 3.1.2). The reason is that the limited LF experiments are quite problem-specific because they depend on the presence of rather inflexible spatial relations between local features.

Robust detection modules (RDM)

The RDMs use grayscale templates of image parts as features. These features are called *image fragments* or simply *fragments*. Given an input image, a feature takes the value of *one* if a matching operation finds the image part from the input. If the image part is not found, the feature takes the value of *zero*. The matching operation allows *limited shift-invariance*, i.e., the image part does not have to be found in some precise location of an input. The image part is found if the *highest match value*, in some *neighborhood* of locations, exceeds a *threshold value*. The neighborhoods and thresholds are feature-specific. The matching operation (program) is shared. Hence, each feature is fully defined by a triplet that specifies the template, the neighborhood, and the threshold. The templates depict somewhat complex structures, e.g., structures that cannot be represented by *one* Gabor-mask or PCA basis image. In feature selection, the templates are cropped from labeled image data.

Our feature selector is a modification of the mechanism first proposed by Ullman, Vidal-Naquet, and Sali [UVNS02, VNU03]. There, the mechanism is primarily intended to demonstrate, as a proof of concept, that intermediate complexity features are better suited to visual discrimination than very simple features (e.g., simple wavelets) and very complex features (e.g., whole object templates).

By complexity, Ullman, Vidal-Naquet, and Sali refer to the complexity of the structures that the templates match. Simple templates such as center-surround filters, Gabor filters (Section 3.3.2), and LBPs (Section 4.5.3) match, and thus detect, the presence of simple structures such as blobs, oriented lines, and very simple local configurations of pixel intensities. Complex templates, on the other hand, match visually complex structures such as the printed image of this sentence or the face of a person. Intermediate complexity templates fall in between the above.

If simple templates are used, then individual features are not very informative and complex models or classifier functions may have to be used to capture the essential relationships between features. If, on the other hand, the target objects are not completely rigid, or if we have a class of non-identical target objects, then the

use of too complex (and specific) templates leads to generalization difficulties. We either have many false negatives (missed detections) or a huge dictionary of templates. Because template-matching operations are computationally expensive, matching huge dictionaries is not efficient. In our experiments we use $16 \times 16$ pixel templates cropped from $100 \times 40$ and $200 \times 40$ pixel images. Given the scale of the objects, the templates can represent object parts such as car tires, noses, eyes, and other structures that are detailed enough to be recognizable by human observers. Some examples are shown in Figure 5.5.

From the viewpoint of computationally efficient classification, fragments and the selection mechanism of Ullman and Vidal-Naquet have two deficiencies. First, fragment features are expensive to use in classification regardless of the selection mechanism. At the classification stage, it is necessary to perform dozens of template matching operations per each input to get the feature vectors that can be classified. Hence, it is reasonable to ask if *all* inputs require fragment-based classification or if delegation techniques can make classification more efficient. Second, it is expensive to select or learn features of sufficient quality. The choice of suitable fragment features is quite problem-specific. In the basic approach, fragments are chosen sequentially from a large pool of candidates so that each candidate is tested against training data. It makes sense to ask if unsuitable candidates can be eliminated with small computational effort.

Our RDM feature selector finds $N_{frag} \in \mathbb{N}$ raster templates of object parts from a subset $\mathcal{T}_p$ of class-labeled training data $\mathcal{T}$. The subscript $p$ of $\mathcal{T}_p$ indicates that there is one object class, called the *positive* class, that is the source of all the inputs in $\mathcal{T}_p$. We assume that there exists a special *no object* class that represents miscellaneous scenery. This special class is never the source of positive inputs. The inputs from the special class are denoted by $\mathcal{T}_n \subseteq \mathcal{T} \setminus \mathcal{T}_p$. If $\mathcal{T}$ contains inputs from several *object* classes (in addition to the *no object* class), then exactly one of the object classes is the source of positive inputs for one RDM feature selector. In multi-class problems we have one RDM for each object class and each object class is the source for one RDM.

During the selection each template is given a threshold value that is used in template matching. For matching we use *normal-*

*ized cross-correlation.* The image neighborhoods, in which cross-correlation is applied, are rectangles that are approximately 2.6 times the size of a fragment. For example, a $16 \times 16$ pixel fragment is still found if it shifts 5 pixels left and 5 down from the source location, i.e., the location from which the fragment is originally cropped.

Fragments are selected for informativeness, as measured by class-conditional entropy. The first fragment $Fr^{(1)}$ is selected to maximize

$$H(C) - H(C \mid Fr), \tag{5.1}$$

where $H(C)$ denotes the Shannon entropy of the class variable and $H(C \mid Fr)$ is the class entropy conditioned on that the status of the fragment feature $Fr$ is known, i.e., present or absent. The fragments $Fr^{(t)}$, $t > 1$, are selected to maximize the additional information

$$\min_{j<t}(H(C \mid Fr^{(j)}) - H(C \mid Fr^{(j)}, Fr^{(t)})). \tag{5.2}$$

In the context of fragment selection, the class variable $C$ is always *binary* regardless of the actual number of object classes. One of the object classes, the positive class, is the source of all the inputs in $\mathcal{T}_p$ for a particular RDM. In principle, the negative class lumps together all inputs in $\mathcal{T} \setminus \mathcal{T}_p$. In practice, we use just $\mathcal{T}_p$ and $\mathcal{T}_n$ in training each RDM, including the calculation of entropies. This allows us to experiment with learning from limited data in which inputs from some classes are not visible to the RDM at the training stage.

To determine whether a fragment is present or absent, suitable match thresholds have to be discovered so that the feature $Fr$ becomes a binary variable. Given a template sampled from $\mathcal{T}_p$, we cross-correlate the template with visible training inputs. The cross-correlation scores are then sorted from lowest to highest. In the sorted array, we mark the intervals where a negative input is followed by a positive input (that has a higher cross-correlation score due to sorting). The midpoints of the marked intervals become *split candidates.* Given a split candidate $sc$, a template $\mathbf{FrT}$, an input $\mathbf{x}$, and a matching operator *match*, the value of the binary variable $Fr$ is 1 iff

$$match(\mathbf{FrT}, \mathbf{x}) \geq sc.$$

Otherwise, $Fr = 0$. As discussed earlier, *match* takes the maximum value found in a specific neighborhood within $\mathbf{x}$. For each **FrT** we choose *sc* that minimizes $H(C \mid Fr)$ as measured from the visible training data.

Suppose that the ACM has been trained to choose a set $Iset$ of promising image locations, called *seed locations*, from which candidate fragments are sampled (the procedure is detailed later). The locations in $Iset$ and the inputs in the subset $\mathcal{T}_p$ of training data determine the pixel templates **FrT** that are available for selection. For example, we may take image $\mathbf{x} \in \mathcal{T}_p$ and location $\mathbf{l} \in Iset$, and then crop an image part **FrT** from $\mathbf{x}$ centered at the coordinates $\mathbf{l}$. After cropping enough patches, we determine the binary variables $Fr$, including the best split *sc* for each, and use Equations (5.1) and (5.2) for feature selection. More precisely, we use Algorithm 3 for greedy feature learning.

In the algorithm, the set $FTset$ denotes the set of templates **FrT** prior to split discovery and binarization. The set $Fset$ denotes the set of binary variables that are fully formed fragment feature candidates. The occurrence table $OT$ and the class labels from the visible training data contain all the information that is available for evaluating Equations (5.1) and (5.2). The gain table $GT$ stores gain values,

$$GT(i, j) = H(C \mid Fr_i) - H(C \mid Fr_j, Fr_i), \tag{5.3}$$

for feature pairs $(Fr_j, Fr_i)$. The stored gain values are used to avoid re-evaluations of Equation (5.2) for known pairs.

Comparing Algorithm 3 to the approach of Ullman, Vidal-Naquet, and Sali [UVNS02, VNU03], some differences are apparent: first, we have replaced the brute-force search with the use of an attention mechanism (ACM) that provides the set *Iset* of good seed locations applied to $\mathcal{T}_p$. Second, we introduced $GT$ that is used as a cache for speeding up the selection loop. The attention mechanism lets us concentrate on fewer candidate fragments than would otherwise be possible. Hence, a simple dense matrix may serve as $GT$. Otherwise, it would be best to store just the column minima of $GT$: in terms of Equation (5.2), we would then keep track of the "worst opponent" of each candidate, making the algorithm a bit more complex.

**Algorithm 3** Select Features

*Iset* ← image locations chosen by the ACM;
*FTset* ← image parts sampled from various $\mathbf{x} \in \mathcal{T}_p$ and $\mathbf{l} \in Iset$;
*Fset* ← ∅;
**for all** $\mathbf{FrT}_i \in FTset$ **do**
  $Fr_i \leftarrow$ the binary feature determined by the best split *sc* of the match scores of $\mathbf{FrT}_i$ over the visible training data;
  $Fset \leftarrow Fset \cup \{Fr_i\}$;
**end for**
build binary occurrence table *OT* s.t. $OT(i, j) = 1$ iff $Fr_i = 1$ in the $j$th image of the visible training data;
allocate empty information gain table *GT*;
**for all** $t \in \{1, \ldots, N_{frag}\}$ **do**
  choose $Fr^{(t)}$ using *GT* and Equations (5.1) and (5.2)
  **if** new gains were computed for some pairs $(Fr_j, Fr_i)$ **then**
    let $GT(i, j) \leftarrow H(C \mid Fr_i) - H(C \mid Fr_j, Fr_i)$ for each such pair;
  **end if**
**end for**

Our experimental results (presented later) indicate that we can sample fewer fragments if sampling is concentrated on certain locations using *Iset*, e.g., subregions of the object foreground. If sampling is not concentrated, e.g., all image locations have the same probability of being chosen, then many of the fragments represent object backgrounds that may not be predictive of object class. Therefore, more fragments have to be sampled to match the classification accuracy that results from the concentrated sampling.

Evaluating Equation (5.2) is expensive, having a computational cost proportional to the number of inputs in the visible training set. Hence, it is intuitively simple to understand that caching is useful. Because the use of the cache *GT* cannot affect the outcomes of decisions, the benefits can be estimated analytically.

Let $|Fset|$ be the initial number of candidate features seen by Algorithm 3 and let $N_{frag}$ be the number of features desired ($|Fset| \gg N_{frag}$). After the first iteration ($t > 1$), features are selected based on Equation (5.2), which requires that the remaining $(|Fset|-(t-1))$ candidates are tested against each of the $t - 1$ previously selected

ones. The overall number of pairwise evaluations is

$$k_1 = \sum_{t=2}^{N_{frag}} (|Fset| - (t-1))(t-1) \tag{5.4}$$

$$= |Fset|\frac{N_{frag}-1}{2}N_{frag} - \sum_{i=1}^{N_{frag}-1} i^2 \tag{5.5}$$

$$\approx |Fset|\frac{N_{frag}-1}{2}N_{frag} - \int_1^{N_{frag}-1} t^2 dt. \tag{5.6}$$

Using a reasonable caching scheme, such as our table $GT$, no pair has to be evaluated twice, and the number of such evaluations becomes

$$k_2 = \sum_{t=2}^{N_{frag}} (|Fset| - (t-1)) \tag{5.7}$$

$$= |Fset|(N_{frag}-1) - \frac{N_{frag}-1}{2}N_{frag}. \tag{5.8}$$

It is easy to see that typically $k_1 \gg k_2$,

$$k_1 - k_2 \approx (|Fset|+1)\frac{N_{frag}-1}{2}N_{frag} - \frac{(N_{frag}-1)^3}{3} - |Fset|(N_{frag}-1) + \frac{1}{3} \tag{5.9}$$

$$> |Fset|\frac{N_{frag}-1}{2}N_{frag} - \frac{N_{frag}-1}{2}N_{frag}^2 - |Fset|(N_{frag}-1) \tag{5.10}$$

$$= \frac{|Fset|N_{frag}}{2}(N_{frag}-1) - \frac{N_{frag}^2}{2}(N_{frag}-1) - |Fset|(N_{frag}-1), \tag{5.11}$$

which is much larger than zero given reasonable values for $|Fset|$ and $N_{frag}$. For example, if we select $N_{frag} = 50$ features out of $|Fset| = 300$, as in our experiments, we get $k_1 - k_2 > 291550$ while $k_1 \approx 328284$. In relative terms, $k_2$ is no more than about 11% of $k_1$.

After the $N_{frag}$ features have been selected, our classification model assumes that the characteristic spatial arrangements of the

fragments found in an input determine the class. Clearly, the model is best suited for semi-rigid objects (e.g., faces) that have some fairly rigid characteristic parts (e.g., eyes). The current matching method does not allow the compact representation of rotating parts. If such exist, each would have to be represented by several fragments. We also note that the matching method is not suitable for objects, the parts of which have large variation in surface textures, e.g., the spotted cows from the previous chapter.

The classification model is implemented using a linear SVM. The inputs to the SVM are encoded as binary vectors that have $N_{frag}$ bits – one per fragment feature. For training, the SVM is given the visible subset of $\mathcal{T}$. After training, the SVMs participate in forming specialists when test inputs require delegation to specialists.

### Attention controller module (ACM)

An *attention controller module* (ACM) serves two purposes: it assists in training the corresponding RDM by providing the seed locations, *Iset* of Algorithm 3, and controls the selective activation of RDMs via the delegation rules. An ACM and the corresponding RDM are trained using the same training set but different features.

The ACMs use whole images instead of fragments. They are linear SVMs such that the input features, $\mathbf{z} = S_{ACM}(\mathbf{x}) = \mathbf{x}$, are simply pixel intensities from the flattened image $\mathbf{x}$. For the ACMs we chose a monotonic model of confidence. From Section 2.3.3, recall that such a model may be encoded with two threshold values (per SVM) that do not have to be probabilities, even if we assume that the correct model is probabilistic.

Also recall (Section 2.3.2) that a linear SVM has the dual form

$$f(\mathbf{x}) = sign(\mathbf{w}^T\mathbf{x} + b) = sign(\sum_i y_i\alpha_i\mathbf{x}_i^T\mathbf{x} + b). \tag{5.12}$$

Here, the dual form illustrates that the ACM takes example views as prototypes that are then matched to new object views $\mathbf{x}$ using the simple dot product of images as the similarity measure. As seen in Equation (5.12), the prototypes are not equal in importance: they have individual weights, $y_i\alpha_i$. The important point is that in the equivalent primal form, a single weight vector $\mathbf{w} \in \mathbb{R}^d$ and $b$ incorporate the same information from possibly all the given prototypes

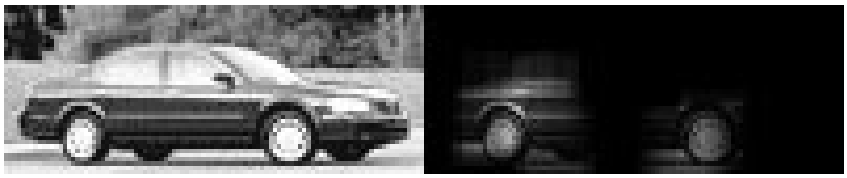


Figure 5.1: A side view car image and the highlighted regions around the pixel locations in *Iset*, as given by the most significant coefficients of $\mathbf{w}$.

$\mathbf{x}_i$ and their relative weights, but the primal form requires just a single dot product to be evaluated per prediction. Thus, while the linear kernel is obviously limited [MP88], the linear machine may be evaluated very quickly unlike machines based on some other kernels.

Denoting the primal form of an ACM by $(\mathbf{w}_{ACM}, b_{ACM})$, the set *Iset* required by Algorithm 3 contains the pixel locations of the most significant coefficients of $\mathbf{w}_{ACM}$. Empirically, these locations work well and seem to make sense as illustrated in Figure 5.1.

Further, the distance of each input from the separating hyperplane can be calculated efficiently and compared against the thresholds of the monotonic confidence model given that the latter are also encoded directly as distances. As illustrated in Figure 5.2, which shows the distribution of input distances from the learned hyperplane in a car detection problem, the monotonic model appears to be close enough to being correct for that particular dataset. Similar behavior was observed in a face detection problem.

We chose to use the medians of the distances of the positive and negative inputs in each SVM training set as the ACM threshold values. Note that the ACM threshold values have nothing to do with the fragment feature threshold values of the RDMs. The former control speed versus accuracy trade-offs while the latter have no influence on such trade-offs.

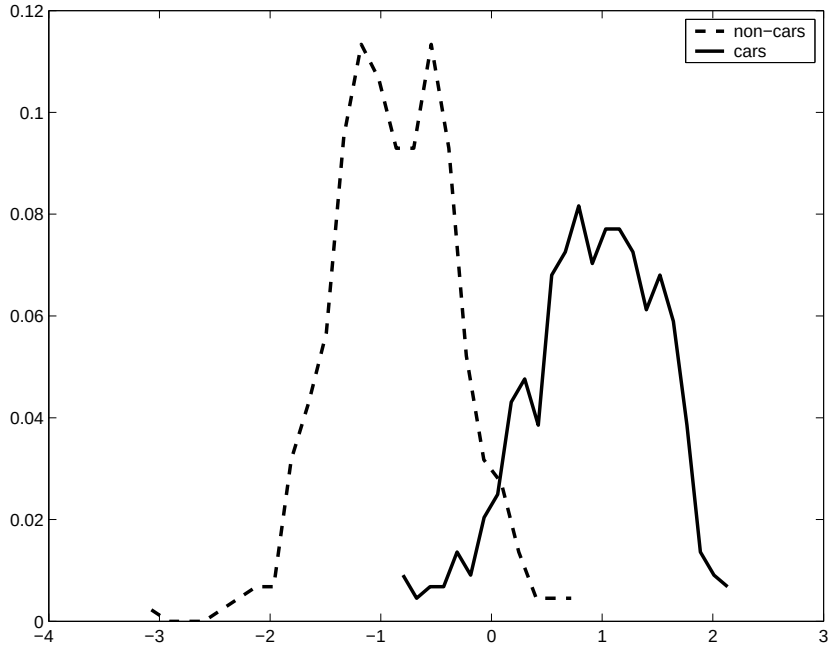



Figure 5.2: Distribution of distances from the discriminating view-based hyperplane $(\mathbf{w}, b)$ in the car detection problem.

### Delegation rules for two-class problems

The ACM and RDM modules should be used in a disciplined way that allows controlled trade-offs between accuracy and speed. As earlier, the delegation rules determine which base classifiers are activated on a per input basis. In the current chapter, a single base classifier may consist of several ACMs or RDMs, but may not have both types of modules.

First, we propose a simple delegation rule for two-class problems. The base classifiers have one-to-one correspondence with the modules. There is one ACM and one RDM. Like the rules in Chapter 4, the simple rule can, in principle, be tuned after training, i.e., making different trade-offs should not require re-training the ACM and RDM modules. After analyzing the rule, we will proceed to multi-class problems that require more complex rules and analysis. The classifier program $prog_f$ with the proposed rule for two-class

problems is the following:

1. The ACM (root base classifier) sees an input image and makes a prediction.

2. If the ACM is not confident (the input is too close to the hyperplane), it abstains and the input is delegated to the RDM, which chooses the final prediction. Else, the ACM is confident and makes the prediction.

Let $\mathbf{X}$ denote the random vector of pixels that is the original input and let $Y$ denote the random variable that is the class of the input. Because we now have two classes, we say that $Y$ is either $+1$ or $-1$. The classifier program $prog_f$ that implements the classification function $f$, including the ACM and RDM modules, is fixed and deterministic, i.e., we analyze the situation after feature selection and training the modules.

The ACM is a linear classifier denoted as

$$d(\mathbf{x}) = \mathbf{w}_{ACM}^T \mathbf{x} + b_{ACM}. \tag{5.13}$$

The learned parameters $\mathbf{w}_{ACM}$ and $b_{ACM}$ in Equation (5.13) are as $\mathbf{w}$ and $b$ in Equation (2.24), Section 2.3.2. For notational simplicity, we assume that the confidence regions of the ACM can be represented by a single threshold value $T \in \mathbb{R}^+$, i.e., the ACM is confident iff $|d(\mathbf{x})| > T$.

Correspondingly, the RDM is

$$D(\mathbf{x}) = \mathbf{w}_{RDM}^T \mathbf{S}(\mathbf{x}) + b_{RDM}, \tag{5.14}$$

where $\mathbf{S}$ maps the original input $\mathbf{x}$ to the binary vector $\mathbf{S}(\mathbf{x})$ in which each bit indicates the presence (1) or absence (0) of a selected fragment feature. Hence, the dimensionality of $\mathbf{w}_{RDM}$ and $\mathbf{S}(\mathbf{x})$ in Equation (5.14) is the number of fragments $N_{frag}$ in Algorithm 3. The RDM is always confident.

We use the shorthand $P(ok)$ to denote the probability of correct classification, $P(f(\mathbf{X}) = Y)$, and write

$$
\begin{aligned}
& P(ok) \\
= & \sum_{y \in \{+1,-1\}} P(Y = y) P(f(\mathbf{X}) = y \mid Y = y) && (5.15) \\
= & P(Y = +1) \Big[ P(d(\mathbf{X}) > T \mid Y = +1) + \\
& P(|d(\mathbf{X})| \leq T \mid Y = +1) P(D(\mathbf{X}) > 0 \mid Y = +1, |d(\mathbf{X})| \leq T) \Big] + \\
& P(Y = -1) \Big[ P(d(\mathbf{X}) < -T \mid Y = -1) + \\
& P(|d(\mathbf{X})| \leq T \mid Y = -1) P(D(\mathbf{X}) < 0 \mid Y = -1, |d(\mathbf{X})| \leq T) \Big] && (5.16) \\
= & \sum_{y} P(Y = y) \Big[ P(yd(\mathbf{X}) > T \mid Y = y) + \\
& P(|d(\mathbf{X})| \leq T \mid Y = y) P(yD(\mathbf{X}) > 0 \mid Y = y, |d(\mathbf{X})| \leq T) \Big] && (5.17) \\
= & P(Yd(\mathbf{X}) > T) + \\
& P(|d(\mathbf{X})| \leq T) P(YD(\mathbf{X}) > 0 \mid |d(\mathbf{X})| \leq T) && (5.18) \\
= & P(|d(\mathbf{X})| > T) P(Yd(\mathbf{X}) > T \mid |d(\mathbf{X})| > T) + \\
& P(|d(\mathbf{X})| \leq T) P(YD(\mathbf{X}) > 0 \mid |d(\mathbf{X})| \leq T). && (5.19)
\end{aligned}
$$

Above, (5.16) follows when we eliminate terms that are equal to zero. Step (5.19) follows because the event $yd(\mathbf{x}) > T$ means that an input is classified both correctly and by the ACM.

It is reasonable to assume that the event $|d(\mathbf{x})| \leq T$ conveys very little information about the event $yD(\mathbf{x}) > 0$. Hence, we assume that

$$
P(YD(\mathbf{X}) > 0 \mid |d(\mathbf{X})| \leq T) = P(YD(\mathbf{X}) > 0). \tag{5.20}
$$

The assumption may, of course, be checked empirically given data to be sampled, $d$, $D$, and a range of values for $T$. Given the assumption, we define

$$
\begin{aligned}
\beta_d(T) &= P(|d(\mathbf{X})| > T), && (5.21) \\
\gamma_d(T) &= P(Yd(\mathbf{X}) > T \mid |d(\mathbf{X})| > T) \text{ and} && (5.22) \\
\delta_D &= P(YD(\mathbf{X}) > 0). && (5.23)
\end{aligned}
$$

Note that because the mapping $D$ is fixed after training, the probability $P(YD(\mathbf{X}) > 0)$ in (5.23) is constant. Using (5.19 – 5.23) we write $P(ok)$ as a function of $T$,

$$P(ok) = acc_{d,D}(T) = \beta_d(T)\gamma_d(T) + (1 - \beta_d(T))\delta_D. \tag{5.24}$$

Note that when each mistake causes the mistake loss of one, $ML(+1,-1) = ML(-1,+1) = 1$, we have $E[ML(f(\mathbf{X}), Y)] = 1 - acc_{d,D}(T)$.

The expected resource loss of the hybrid is

$$E[RL(\mathbf{X}, prog_f)] = \beta_d(T)RL_{ACM} + (1-\beta_d(T))(RL_{ACM} + RL_{RDM}), \tag{5.25}$$

where $RL_{ACM}$ and $RL_{RDM}$ denote the resource losses of the ACM and RDM. Note that $E[RL(\mathbf{X}, prog_f)] < RL_{RDM}$ if and only if

$$\frac{RL_{ACM}}{RL_{RDM}} < \beta_d(T). \tag{5.26}$$

For example, if the ACM is a hundred times faster than the RDM, then the ACM has to be confident with slightly more than one percent of the data.

When $T$ grows in Equation (5.24), the event $|d(\mathbf{x})| > T$ becomes less likely and $\beta_d(T)$ diminishes. Furthermore, if monotonic confidence models are appropriate (recall Section 2.3.3), then $\gamma_d(T)$ grows with $T$. Figure 5.2 shows that monotonic confidence models are appropriate for the current data. We observe that:

1. If there exists $T$ such that $\gamma_d(T) > \delta_D\gamma_d(0)$, and $\beta_d(T) > 0$, then the hybrid accuracy exceeds both the RDM and ACM, i.e., $acc_{d,D}(T) > \delta_D$ and $acc_{d,D}(T) > \gamma_d(0)$. The ACM becomes an *expert* of a subset of inputs. Hence, if $RL_{ACM} \ll RL_{RDM}$, then $\beta_d(T)$ does not have to be much larger than zero to satisfy Equation (5.26) and thus allow the hybrid to be *both* more accurate and faster than the RDM in isolation. In other words, the delegation framework does not always force accuracy and speed to be conflicting goals.

2. With smaller $T$, we may have $\gamma_d(T) = \delta_D$, and a larger $\beta_d(T)$, thus gaining speed without losing accuracy.

3. With even smaller $T$ such that $\delta_D > \gamma_d(T) > \gamma_d(0)$, we can trade accuracy for speed.

### Delegation rules for multi-class problems

In the multi-class setting we have $m \geq 2$ mutually exclusive object classes and a single *no object* class. Each of the $m$ object classes has an ACM and an RDM trained to discriminate the object class from the no object class. Being two-class classifiers, the ACMs and the RDMs do not have one-to-one correspondence with the base classifiers of the classifier tree. For example, the root node of the classifier tree must discriminate between at least three classes $(m + 1)$ and no single ACM or RDM can do that.

We examine a set of rules the purpose of which is to combine ACMs and RDMs so that a delegating classifier tree exists, i.e., specialists are formed from modules when necessary. In addition, the rules have parameters such that accuracy versus speed trade-offs can be controlled without re-training the individual ACMs and RDMs.

In terms of the tree model from Chapter 2, the root node of the tree consists of *all* of the ACMs. For the root node to work efficiently, the object classes have to be distinct enough to allow coarse discrimination on the basis of low-frequency information. This requirement is similar to the requirement of distinct superordinate categories that was stated in the root feature selection hypothesis (Section 3.1.2, Chapter 3). The current experimental setting is easier and less generic than the settings the hypothesis is meant for. In the current setting, the objects are quite rigid in shape and the viewing angles are severely restricted, e.g, we have frontal upright views of faces. Hence, efficiency is not pursued as in the hypothesis, i.e., by ignoring the spatial relations between local features. The weight vector of an ACM is a template of where the local features (pixel intensities) have to be. Such templates obviously do not work when the data is as complex as it was in Chapter 4. With complex data, efficiency may have to be pursued by ignoring spatial relations. Note, however, that our theoretical analysis further below does not require or imply the specific kind of features or templates that we use in the current study.

A typical RDM in the current study behaves like a subprogram of a specialist base classifier that searches for highly class-specific object parts, such as car tires. The tree implements gradual exclusion of classes. The root delegates to base classifiers each of which

is a set of 1 to $m$ RDMs. If the set defining a base classifier has $m' < m$ RDMs, then $m - m' > 0$ classes are excluded and $m' + 1$ remain before the base classifier is activated. This can be seen in the delegation rules presented below. Further, in the current study the use of highly class-specific object parts allows us to skip class-vs-class training. With more generic features, such as wavelets or Gabor filter responses, we might have to use some class-competitive training scheme.

Each of the $m$ ACMs selects an answer from $\{yes, abstain, no\}$. Each *activated* RDM selects from $\{yes, no\}$. The delegation rules are the following:

1. Activate the root base classifier that consists of $m$ ACMs.
2. If all $m$ ACMs of the root say *no*, then choose *prediction* $\leftarrow$ *no object*.
3. Else, if a single $\text{ACM}_j$ says *yes*, then choose *prediction* $\leftarrow j$.
4. Else, if more than one ACM says *yes*, then choose the prediction randomly from the positive matches.
5. Else, the root is *not confident* and delegates. Direct the input to the base classifier that consists of the RDMs of the classes whose ACMs abstained. Denote the set of such RDMs by $A$.
6. For each $RDM_i \in A$, activate $RDM_i$. This results in $|A|$ predictions, each of which is *yes* or *no*. Choose as follows:
   (a) If all $|A|$ predictions are *no*, then choose *prediction* $\leftarrow$ *no object*.
   (b) Else, if exactly one prediction, by $RDM_i$, is *yes*, then choose *prediction* $\leftarrow i$.
   (c) Else, more than one prediction is *yes*. Choose the prediction randomly from the positive matches.

First, note that if RDMs could abstain and the tree was made deeper, containing even more specialized base classifiers, the above rules could easily be formulated as a recursive program much like

the rules in Chapter 4 were formulated. The latter steps for handling the $|A|$ predictions of the activated RDMs essentially repeat the earlier steps for ACMs. The single exception is that the set of $|A|$ RDM predictions cannot contain a mixture of *no* and *abstain* responses, and hence the delegation step is omitted.

Second, note that the root can predict any class immediately without activating additional base classifiers (RDM sets). This ability was also used in the delegation rules in Chapter 4. If the rules here were formulated as a recursive program, then any activated base classifier would have the same ability.

Third, the above rules are somewhat more careful than the rules in Chapter 4. At the end of Chapter 3 we argued that it is difficult for a base classifier to detect if it has been activated to process an inappropriate input outside its specialization. If gradual exclusion of classes is implemented, it is also pointless to detect inappropriate activation because delegation is irreversible and the activated base classifier cannot predict the labels of excluded classes. What can be done, however, is that greater care may be taken in gradual exclusion. In Chapter 4 exclusion was not very fine-grained. For example, activating the animal specialist caused all non-animal classes to be excluded immediately. In contrast, the above rules are fine-grained and delegation does not necessarily exclude *any* classes, i.e, the root may delegate to a base classifier that consists of *all* RDMs. Delegation results in an irrecoverable error only if the ACM of the correct class says *no*.

The multi-class delegation rules can be analyzed much like the two-class rules. Denote the object classes by $1, \ldots, m$, and *no object* by $-1$. The class priors are $P(Y = i) > 0$ for all classes $i$. The priors are abbreviated as P(i). The event $d_i(\mathbf{X}) > T_i$ means $\text{ACM}_i$ predicts *yes*, and $d_i(\mathbf{X}) < -T_i$ means *no*. Else, $\text{ACM}_i$ *abstains*. The events $D_i(\mathbf{X}) \geq 0$ and $D_i(\mathbf{X}) < 0$ for the RDMs are interpreted the same way.

It is assumed that *events describing classifier output inequalities are mutually independent if the class is known*. For example,

$$P(d_j(\mathbf{X}) > T_j \mid Y = i, d_k(\mathbf{X}) < -T_k) = P(d_j(\mathbf{X}) > T_j \mid Y = i). \tag{5.27}$$

First, note that typical thresholds satisfy $T_j \neq 0$ and *less* than full

conditional independence, e.g.,

$$P(d_j(\mathbf{X}) > T_j \mid Y = i, d_k(\mathbf{X}) = v) = P(d_j(\mathbf{X}) > T_j \mid Y = i), \tag{5.28}$$

is assumed. For example, the precise value $v$ of $d_k(\mathbf{X})$ may easily convey more information about $\mathbf{X}$ than the inequality $d_k(\mathbf{X}) < -T_k$. Given $v$, $\mathbf{x}$ must occupy one specific hyperplane while the inequality allows an infinite number of parallel hyperplanes. Second, note that the current assumption is more general than (5.20) and implies (5.20). We have

$$\begin{aligned} & P(YD(\mathbf{X}) > 0 \mid |d(\mathbf{X})| \leq T) \\ = & \sum_y P(y \mid |d(\mathbf{X})| \leq T) P(yD(\mathbf{X}) > 0 \mid y, |d(\mathbf{X})| \leq T) \end{aligned} \tag{5.29}$$

$$= \sum_y P(y) P(yD(\mathbf{X}) > 0 \mid y) \tag{5.30}$$

$$= P(YD(\mathbf{X}) > 0). \tag{5.31}$$

Above, in step (5.30) $P(y)$ results from $|d(\mathbf{X})| \leq T$ conveying no information about the class and both $P(D(\mathbf{X}) > 0 \mid Y = +1)$ and $P(-D(\mathbf{X}) > 0 \mid Y = -1)$ result from the current assumption that inequality events are independent if the class is known. Third, note that inequality events must be assumed dependent if the class is not known.

To simplify the analysis, we assume that when a randomized prediction is made, it is always wrong. An input is classified correctly iff one of the following conditions is true.

1. An object of class $i$ is detected *fast*, if

   $$d_i(\mathbf{X}) > T_i$$

   and

   $$\forall j \neq i : d_j(\mathbf{X}) \leq T_j.$$

2. An object of class $i$ is detected *slowly*, if first

   $$|d_i(\mathbf{X})| \leq T_i,$$

   and then

   $$D_i(\mathbf{X}) \geq 0$$

and

$$\forall j \neq i : d_j(\mathbf{X}) \leq T_j$$

and

$$\forall j \neq i : RDM_j activated \Rightarrow D_j(\mathbf{X}) < 0.$$

3. A non-object (class $-1$) is dismissed, i.e., no ACM or RDM says *yes*.

Abbreviating probabilities $P(\cdot|\text{condition})$ as $P_{\text{condition}}(\cdot)$, we get

$$P(ok) = \sum_{i>0} P(Y=i)P_{Y=i}(ok) + P(Y=-1)P_{Y=-1}(ok) \tag{5.32}$$

$$P_{Y=i}(ok) = P_{Y=i}(d_i(\mathbf{X}) > T_i)R_i + P_{Y=i}(|d_i(\mathbf{X})| \leq T_i)P_{Y=i}(D_i(\mathbf{X}) \geq 0)Dis_i \tag{5.33}$$

$$P_{Y=-1}(ok) = \prod_{j>0} P_{Y=-1}(dis_j) \tag{5.34}$$

$$R_i = \prod_{j>0:j\neq i} P_{Y=i}(d_j(\mathbf{X}) \leq T_j) \tag{5.35}$$

$$Dis_i = \prod_{j>0:j\neq i} P_{Y=i}(dis_j) \tag{5.36}$$

$$P_{Y=i}(dis_j) = 1 - P_{Y=i}(d_j(\mathbf{X}) > T_j) - P_{Y=i}(|d_j(\mathbf{X})| \leq T_j)P_{Y=i}(D_j(\mathbf{X}) \geq 0), \tag{5.37}$$

where $dis_j$ (dismissal) means there is no positive response (detection) from the $j$th ACM or RDM.

Next, we prove that the hybrid system may, in principle, exceed the accuracy of a pure RDM committee. This hybrid accuracy may then be traded for speed. In a pure RDM committee all RDMs are activated once per input. If exactly one $RDM_i$ predicts *yes* and other RDMs predict *no*, then the committee predicts class $i$. Else, if all RDMs predict *no*, then the committee predicts *no object*. Else, the committee predicts a random class.

We make some reasonable assumptions about the thresholds. When $i \neq j$, well-behaved ACMs have

$$P_{Y=i}(d_j(\mathbf{X}) > T_j) \ll P_{Y=i}(d_j(\mathbf{X}) < -T_j), \tag{5.38}$$

and

$$P_{Y=i}(d_j(\mathbf{X}) < -T_j) > 0. \tag{5.39}$$

The former assumption states that fast false positives are much less likely than fast and correct rejections. The latter assumption states

that fast and correct rejections are possible. In addition, we assume that the value of $T_j$ is sufficiently large so that

$$i \neq j \Rightarrow P_{Y=i}(d_j(\mathbf{X}) > T_j) < P_{Y=i}(D_j(\mathbf{X}) \geq 0). \tag{5.40}$$

Note that if the monotonic model of confidence is true, then such $T_j$ must exist.

Let $\epsilon > 0$ be a very small constant and let $j \neq i$. For later convenience, choose $\epsilon < P_{Y=i}(d_j(\mathbf{X}) < -T_j)$. If the accuracies of the RDMs are limited by

$$P_{Y=i}(D_i(\mathbf{X}) \geq 0) \quad < \quad \frac{P_{Y=i}(d_i(\mathbf{X}) > T_i)}{P_{Y=i}(d_i(\mathbf{X}) < -T_i) + P_{Y=i}(d_i(\mathbf{X}) > T_i)} \tag{5.41}$$

$$P_{Y=i}(D_j(\mathbf{X}) \geq 0) \quad > \quad \frac{P_{Y=i}(d_j(\mathbf{X}) > T_j)}{P_{Y=i}(d_j(\mathbf{X}) < -T_j) - \epsilon}, \tag{5.42}$$

then the hybrid exceeds the accuracy of the committee of RDMs. The inequalities do not contradict each other, i.e., there are probability assignments that satisfy both. The first inequality (5.41) essentially means that if the RDMs are very good at detecting the classes they are dedicated to, then the corresponding ACMs must be very good at avoiding confident (fast) false negatives. The second inequality (5.42) essentially means that if the RDMs are very good at avoiding false positives, then the corresponding ACMs must be very good at avoiding confident (fast) false positives and making confident (fast) rejections.

To prove the claim, we first take the trivially true inequality (again $i \neq j$)

$$\begin{aligned} & P_{Y=i}(D_j(\mathbf{X}) \geq 0) \\ > \quad & \left[P_{Y=i}(|d_j(\mathbf{X})| \leq T_j) + P_{Y=i}(d_j(\mathbf{X}) < -T_j) - \epsilon\right] \\ & P_{Y=i}(D_j(\mathbf{X}) \geq 0). \end{aligned} \tag{5.43}$$

Multiplying (5.43) by $-1$, adding 1 to both sides, substituting bound (5.42), and using the fact that we chose

$$\epsilon < P_{Y=i}(d_j(\mathbf{X}) < -T_j)$$

yields

$$P_{Y=i}(dis_j) > P_{Y=i}(D_j(\mathbf{X}) < 0). \tag{5.44}$$

It immediately follows from (5.44) that

$$\begin{aligned} Dis_i &= \prod_{j>0:j\neq i} P_{Y=i}(dis_j) \\ &> \prod_{j>0:j\neq i} P_{Y=i}(D_j(\mathbf{X}) < 0) = Dis_i^*. \end{aligned} \tag{5.45}$$

The new term $Dis_i^*$ is the probability that the individual machines of a pure RDM committee do not make false detections.

Next, performing simple algebraic manipulation of the assumption (5.40) yields

$$P_{Y=i}(d_j(\mathbf{X}) \leq T_j) > P_{Y=i}(D_j(\mathbf{X}) < 0). \tag{5.46}$$

Using (5.46), we immediately see that

$$\begin{aligned} R_i &= \prod_{j>0:j\neq i} P_{Y=i}(d_j(\mathbf{X}) \leq T_j) \\ &> \prod_{j>0:j\neq i} P_{Y=i}(D_j(\mathbf{X}) < 0) = Dis_i^*. \end{aligned} \tag{5.47}$$

Substituting the inequalities (5.45) and (5.47) into (5.33) yields

$$\begin{aligned} P_{Y=i}(ok) &> \Big[P_{Y=i}(d_i(\mathbf{X}) > T_i) + \\ &\quad P_{Y=i}(|d_i(\mathbf{X})| \leq T_i)P_{Y=i}(D_i(\mathbf{X}) \geq 0)\Big] Dis_i^* \end{aligned} \tag{5.48}$$

$$\begin{aligned} &> \Big[P_{Y=i}(|d_i(\mathbf{X})| > T_i) + \\ &\quad P_{Y=i}(|d_i(\mathbf{X})| \leq T_i)\Big] P_{Y=i}(D_i(\mathbf{X}) \geq 0) Dis_i^* \end{aligned} \tag{5.49}$$

$$= P_{Y=i}(D_i(\mathbf{X}) \geq 0) Dis_i^* \tag{5.50}$$

$$= P_{Y=i}^*(ok), \tag{5.51}$$

where (5.49) resulted from substituting the bound (5.41) into (5.48). The new term $P_i^*(ok)$ is the probability that a pure RDM committee predicts correctly if the true class is $i > 0$. Similarly, substituting (5.44) into (5.34) gives the corresponding result for the *no object* class

$$P_{Y=-1}(ok) > \quad P_{Y=-1}^*(ok). \tag{5.52}$$

Finally, substituting (5.51) and (5.52) into (5.32) shows that if the bounds (5.41) and (5.42) hold, then the hybrid is more accurate than a pure RDM committee. If the bounds are relaxed, we can trade accuracy for speed. If randomized choices (i.e., provoked by multiple *yes* answers) are taken into account, the pure committee still has no advantage.

### 5.3.2 Experimental design

We designed experiments to evaluate the performance of the delegation rules for two-class and multi-class problems. In these experiments we used *detection problems* in which one or two classes represented objects and the additional *no object* class represented miscellaneous scenery with no objects of interest to be seen. The two-class rules were applied when the detection problem was such that there was one object class plus the *no object* class. The multi-class rules were applied when there were two classes plus the *no object* class. The object classes were cars and faces.

We used the car dataset from Agarwal and Roth [AR02]. The data had gray-scale side views of various cars depicted in mostly urban environments, e.g., parked on streets. The negative data (non-cars) consisted of mostly urban scenery without cars, but tended to depict places where one could drive a car. This ensured that trained detectors did not detect by context, e.g., detected streets (common contexts) instead of cars. The dataset had about 1000 gray-scale images in $100 \times 40$ resolution. Some examples of cars and non-cars are shown in Figure 5.3.

The generation of our face dataset was more complicated. We sampled random frontal face images from the AR dataset of Martinez and Benavente [MB98]. Because the AR image backgrounds were small and completely trivial, we embedded the faces on larger backgrounds sampled from the BioID database of Jesorsky, Kirchberg and Frischholz [JKF01]. We made the backgrounds very large compared to the size of the faces to increase the number of unsuitable image fragments, i.e., to see how image fragment selection copes. Backgrounds sampled from the same source were also used as negative data (non-faces). Our face dataset had about 1900 gray-scale images in $200 \times 40$ resolution. Some examples of faces and non-faces are shown in Figure 5.4. We note that we did not use

Figure 5.3: Five cars (left) and non-cars (right).

the BioId database as a source of faces because the AR set has more individuals and greater variation in skin colors, facial expressions, and facial hair.

To evaluate how well the classifiers cope with even coarser information than what is available in the $100 \times 40$ and $200 \times 40$ pixel inputs, we performed two experiments in which we convolved the input images of the ACMs with Gaussian filters. In the first experiment, we used a $7 \times 7$ mask with standard deviation $\sigma = 1$. In the second experiment, we used a $9 \times 9$ mask with $\sigma = 2$ for heavier blurring. The blurring operations discard higher frequencies of image content. Unlike the ACMs, the RDMs were allowed to use the original data.

In the following, we use some abbreviations. A1 stands for the variant of the hybrid approach where the ACMs are used *only* for the selection of fragments in the learning phase. A2 stands for the main variant where the ACMs are used for both the selection of fragments and delegation. It is A2 that was the focus of the introductory Section 5.3.1. RP stands for pure RDM classifiers each of which selects fragments from a random set of candidates sampled from random locations in object data. When RP is compared to a variant of the hybrid (A1 or A2), the random set size equals the size of the *Fset* used by the hybrid. SVM1 and SVM2 are linear and second-order polynomial soft-margin SVMs, respectively. RF80

Figure 5.4: Five faces (left) and non-faces (right).

denotes Random Forests [Bre01] using 80 full-grown decision trees. The SVMs and random forests can be considered state-of-the-art machine learning algorithms that, in the current experiments, see the raw image data directly.

For all RDMs, the candidate fragment size was fixed to $16 \times 16$ pixels. The image neighborhoods, in which cross-correlation was applied, were $26 \times 26$ pixels in size. Each fragment had a neighborhood the center of which was the source location of the fragment, i.e., the location from which the fragment was originally cropped. The number of selected fragments, $N_{frag}$, was set to 50 in Algorithm 3. Each $\mathbf{w}_{ACM}$ had either $100 \times 40$ or $200 \times 40$ coefficients depending on the inputs. For each RDM-specific set of seed locations *Iset*, we chose the locations of the 60 most significant coefficients of $\mathbf{w}_{ACM}$ of the corresponding ACM. Each RDM sampled 5 fragment candidates from each of its 60 seed locations, i.e., each RDM chose 5 positive input images randomly, and from each image chosen, selected one fragment centered on a seed location. Hence, the number of candidate fragments was $|Fset| = 300$.

For the ACM thresholds $T_j^+$ and $T_j^-$, we initially chose the medians of the sets $\{d_j(\mathbf{x}_k) : y_k = +1\}$ and $\{d_j(\mathbf{x}_k) : y_k = -1\}$ calculated from the visible training inputs $\mathbf{x}_k$. Note that in the analysis of the delegation rules the superscripts $+$ and $-$ of $T_j$ were omitted for simplicity. In practice we got values such as $T_j^+ = 1.0003$ and

$T_j^- = -1.0002$. Hence, $T_j = T_j^+$ and $-T_j = -T_j^+ \approx T_j^-$.

The reported results describe performance on *inverted* 10-fold cross-validation. Instead of using $(n-1)/n$ of the data for training (including feature selection) and $1/n$ for testing as per iteration in the ordinary kind of $n$-fold cross-validation, we inverted the roles of the training and testing sets to investigate learning from small training samples.

In the two-class car detection problem the balanced dataset had 490 images of cars and 490 images of non-cars. Each fold contained 49 positives (cars) and 49 negatives (non-cars). During the $i$th iteration of inverted cross-validation, the $i$th fold became the training set for feature selection and classifiers. After training, the inputs outside the training fold were used for testing. Hence, during each iteration 441 positives and 441 negatives were used for testing. In the two-class face detection problem the balanced dataset had $460 + 460$ images, $46 + 46$ in each training set, and $414 + 414$ in each test set.

In the multi-class detection problem we had three classes: the car class, the face class, and the *no object* class. The previous datasets were merged after the data from the face detection problem was cropped to the same resolution as the car data, i.e., $50 \times 40$ pixels were removed from the left and right borders of the face detection data. Each fold contained 49 cars, 46 faces, and $49 + 46$ non-objects. The non-objects were the 49 non-cars and 46 non-faces from the two-class experiments. For the A2 multi-class delegation rules, training was done as in the two-class problems above, i.e., object-class-specific ACMs and RDMs saw *only* objects and non-objects from the corresponding two-class problem. For example, the $\mathcal{T}_p$ of a car-specific RDM consisted of images of cars and the $\mathcal{T}_n$ of the RDM consisted of images of non-cars from the two-class car detection problem. After training, the 441 cars, 414 faces, and $441 + 414$ non-objects outside the fold became test data for the rules. Hence, class-specific ACMs and RDMs were exposed to *novel kinds* of negatives. For example, the car ACM was exposed to faces and non-objects that did not resemble the original non-cars. The approach is similar to the tests in Section 3.3.3 of Chapter 3. Those tests indicated that at least some SVMs were tolerant of changes of sampling bias, e.g., training with subcategories of inputs and testing with full categories (supposing simplified variant of the A2

multi-class rules).

Exposing the ACMs and RDMs to novel kinds of negatives allowed us to see if classifiers trained for two-class problems could become reusable modules in A2 multi-class delegation rules of Section 5.3.1. The multi-class results of the other methods were obtained using multiple binary classifiers and a *one-vs-rest* wrapper. Hence, the binary classifiers of the other methods did not have to see novel kinds of negatives during testing. Also recall that in Section 2.3.2 of Chapter 2 we already discussed the disadvantages of one-vs-rest multi-class training, e.g., what kind of trouble is caused by the class priors when participating binary classifiers minimize (surrogates of) the 0/1 loss.

### 5.3.3 Results

The accuracies of the various methods are summarized in Table 5.1. The numbers denote averages over cross-validation folds. The first two rows correspond to the two-class detection problems, i.e., car detection and face detection. The third row (*multi*) corresponds to the three-class detection problem involving cars, faces, and non-objects.

In the car problem, A1 using the attentive mechanism has similar accuracy to RP, the pure parts-based method. In the face problem, A1 seems to have an advantage over RP. Because both A1 and RP used 300 candidate fragments to select 50, it is reasonable to infer that A1 had better candidate fragments. It seems likely that the candidates of A1 were better because A1 used the significant coefficients of the ACM to select the seed locations while RP used random seed locations. In other words, RP would have to sample more candidates from various locations to compete with A1. Because the faces were small compared to the backgrounds (Figure 5.4), it is understandable why this advantage was apparent in the face detection problem, but not necessarily in the car detection problem (Figure 5.3). The precise framing of the selected parts does not seem to matter much (see Figure 5.5), e.g., RP was reasonably good because even random seed locations inevitably produce some fragments that overlap with important facial parts. If precise framing mattered, RP could be severely disadvantaged. Overall, A1 seems to demonstrate that the delegation framework is not limited

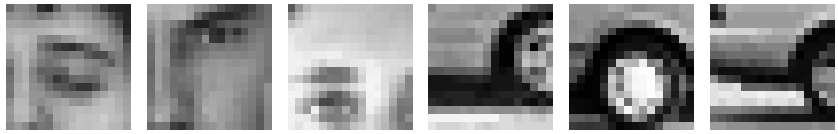


Figure 5.5: Some example fragments A2 extracted from faces and cars.

Table 5.1: Inverted 10-fold cross-validation accuracies.

| | A1 | A2 | RP | SVM1 | SVM2 | RF80 |
|---|---|---|---|---|---|---|
| Cars | 0.9681 | 0.9702 | 0.9506 | 0.9263 | 0.9378 | 0.9247 |
| Faces | 0.9679 | 0.9691 | 0.9303 | 0.9019 | 0.8787 | 0.9224 |
| Multi | – | 0.9483 | – | 0.9030 | 0.9009 | 0.9190 |
| $C_{\sigma=1}$ | – | 0.9664 | – | 0.9194 | 0.9349 | 0.9239 |
| $C_{\sigma=2}$ | – | 0.9605 | – | 0.9090 | 0.9272 | 0.9167 |
| $F_{\sigma=1}$ | – | 0.9616 | – | 0.8912 | 0.8710 | 0.9161 |
| $F_{\sigma=2}$ | – | 0.9545 | – | 0.8738 | 0.8568 | 0.9066 |

to improving the efficiency of classification after training. Learning may be made more efficient as well, because better candidate fragments mean that fewer have to be given to the selection algorithm.

Comparing A1 to A2, the variant that increases classification speed, shows that using the simple linear machine in preliminary classification did not degrade the accuracies. In the car problem, the ACM of A2 was about 770 times faster than the RDM, and in the face problem, about 340 times faster. We denote the constant evaluation times of the modules by $t_{ACM}$ and $t_{RDM}$. Supposing that resource losses are linearly proportional to time, i.e., $\exists l > 0 : RL_{ACM} = lt_{ACM}$ and $RL_{RDM} = lt_{RDM}$, we may approximate Equation (5.25) as

$$\begin{aligned} E[RL(\mathbf{X}, prog_f)] &= \beta_d(T) l t_{ACM} + (1 - \beta_d(T)) l (t_{ACM} + t_{RDM}) \\ &\approx (1 - \beta_d(T)) l t_{RDM}, \end{aligned} \tag{5.53}$$

because $RL_{RDM}$ is orders of magnitude larger than $RL_{ACM}$. In car detection $\beta_d(T)$ was about 0.44 and in face detection it was about 0.42. Hence, Equation (5.53) shows that car detection using A2 costs just 56% of the pure RDM cost in resources because 44%

of the cars are classified instantly using the ACM only. Further, Inequality (5.26) shows that given the values of $\beta_d(T)$, A2 would have had a resource loss advantage even if the ACM had been just about 2.5 times faster than the RDM.

In the first and second rows of Table 5.1 it can be seen that the black-box methods, SVM1, SVM2, and RF80, were less accurate than A1 and A2. The differences were larger in face detection, which was likely due to the large backgrounds confusing the black-box methods. The second-order polynomial machine, SVM2, seemed to be affected the most. SVM2 also has larger resource losses than SVM1 because SVM2 cannot be evaluated in primal form. Given little training data and non-trivial backgrounds, simple ACMs based on linear machines (such as SVM1) may thus be better for the hybrid delegation approach than more complex non-linear machines.

In the third row of Table 5.1 it can be seen that the A2 multi-class delegation rules predicted more accurately than the wrapped black-box methods. Recall that this is in spite of the wrappers using one-vs-rest training, which ensures that the binary machines see all kinds of inputs during training, e.g., cars, faces, and two kinds of non-objects (non-cars and non-faces). In contrast, the ACMs and RDMs of A2 were trained using subsets of training data, and were thus exposed to novel kinds of inputs during the testing.

The runtime behavior of A2 multi-class delegation rules is shown in Figures 5.6 and 5.7. In Figure 5.6 we show mean accuracy as the function of mean classification time (in seconds). In Figure 5.7 we show mean time and accuracy as functions of $T_j$. The circles denote different values of $T_j$. As explained earlier, during the training the values of $T_j$ were set to medians over the training sets. To create the figures, we simply changed the values after the training and no retraining was necessary. Both ACMs were always given the same value, e.g., $T_1 = T_2 = 1.0$ when the mean time was 0.164 seconds. The measurements were taken with a computer equipped with an AMD 3500 processor. A pure RDM committee required about 0.41 seconds per input, i.e., two RDMs were always activated for each test input.

Figures 5.6 and 5.7 show that when $T_j$ were low, the accuracy of A2 multi-class delegation rules was worse than the accuracies of SVM1, SVM2 and RF80 (third row of Table 5.1). As noted

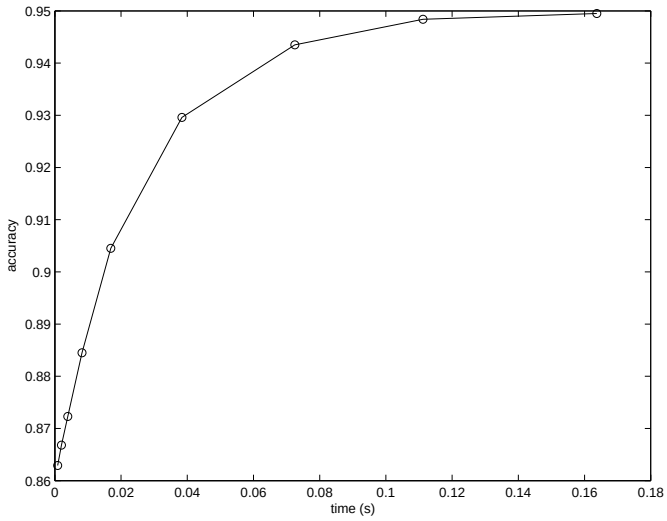



Figure 5.6: Mean time versus mean accuracy.

earlier, the multi-class results of the latter three were obtained using multiple binary classifiers and a *one-vs-rest* wrapper. Thus, unlike A2 the binary classifiers of the latter three methods did not have to see novel kinds of inputs during the testing. When $T_j$ were set to 0.01, A2 essentially became a committee of linear SVMs and the measured accuracy was 0.8629. SVM1, an another committee of linear SVMs, had the accuracy of 0.9030. Since both committees used raw pixels as features, the likely explanation for the accuracy gap is the presence of novel inputs in A2.

The figures show that speed can be traded for accuracy. As in Figure 4.10 of the previous chapter, there is clearly a point of diminishing returns. In Figure 5.6 the point is at about 0.08 seconds. The consequences of changing $T_j$ seem to be predictable and well-behaved. Increases always lead to better accuracy and worse speed. In addition there are no bottlenecks, i.e., extreme trade-offs can be made if desired. Hence, we can say that the parameters $T_j$ allow reasonable control over trade-offs.

Finally, we note the results of the blurring experiments. The results are shown in the lower half of Table 5.1. The fourth and fifth rows correspond to the two-class car detection problem, and the sixth and seventh rows correspond to the two-class face detec-

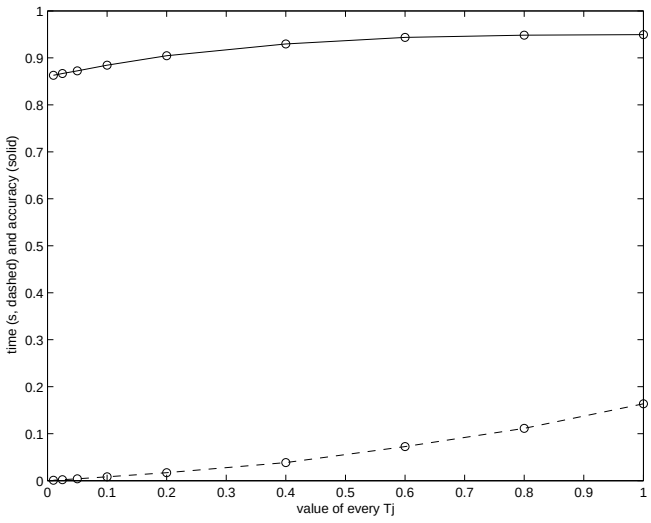



Figure 5.7: $T_j$ versus mean time and mean accuracy.

tion problem. It can be seen that blurring the data caused only graceful degradation of accuracy for all of the methods. Hence, the results show that basic categories of objects may be recognizable using only very low frequencies of image content. Of course, compared to normal high-resolution photographs the raw $100 \times 40$ and $200 \times 40$ pixel inputs had already lost the highest frequencies, i.e., downsampling had involved low-pass filtering to avoid aliasing.

### 5.3.4 Summary

In the above first half of the chapter, we examined two questions. The first question asked if the framework requires problem-specific organization of the classifier nodes. The second question asked if the framework is limited to controlling the efficiency of classification, or if it is also useful in controlling the efficiency of feature selection and training. Answers to these questions were sought empirically.

Based on the empirical results, especially from the use of *A2 multi-class delegation rules*, we can answer the first question and say that the framework does not require that a problem-specific organization of the classifier nodes is used.

We demonstrated that *all* non-empty subsets of object classes

can be given specialists, each of which is formed from a set of simpler modules (RDMs) at runtime. The number of these modules was smaller than the number of specialists. In the general case the difference can be very large.

To make the demonstration interesting, the test systems and modules had certain capabilities and properties. First, the threshold parameters $T_j$ could be used to control speed versus accuracy trade-offs without retraining the classifiers (RDM and ACM modules). This capability is clearly useful and was also present in Chapter 4. Here and in Chapter 4 we showed that the parameters allowed reasonable control over these trade-offs, i.e., changing parameter values resulted in predictable changes of behavior and there were no bottlenecks preventing extreme trade-offs. Second, the individual modules were trained as if they were meant for two-class detection problems. Hence, the modules were exposed to novel kinds of inputs after training. While this may seem demanding, it is desirable because of modularity and reusability. In other words, a module used in multi-class detection is not very modular if retraining is necessary when the number of object classes $m$ is increased to $m + 1$.

We complemented the demonstration with a theoretical analysis of the delegation rules. In the simplified case, we derived inequalities that describe which conditions and threshold parameter values lead to speed versus accuracy trade-offs and which lead to increased speed without loss of accuracy. In the case of A2 multi-class delegation rules, we proved that the rules could, in principle, exceed the accuracy of a pure RDM committee. We noted that this accuracy could then be traded for speed. The basic point of the proof was, quite simply, that the rules are not arbitrary or inherently limited to poor performance.

Based on the empirical results, we can also answer the second question and say that the framework can be useful in improving the efficiency of feature selection and training. For the demonstration we used fragment features that were selected from training data. The fragment features were a good choice for the demonstration because they are known to produce accurate classifiers, but their selection is computationally expensive. We made the selection process more efficient. The more interesting efficiency improvements resulted from the use of ACMs (in the root node) to

constrain the sampling of candidate features in RDMs (in the specialist nodes). More precisely, each ACM focused the attention of the corresponding RDM to a set $Iset$ of image locations from which the ACM predicted that promising features could be found.

The experimental results showed that when the image backgrounds were large compared to the objects, the ACM-focused attention led to more accurate classifiers than randomly focused attention, given that both were used to select $|Fset|$ candidate fragments. Hence, ACM-focused attention is more efficient as it allows a smaller $|Fset|$ to be used. A smaller $|Fset|$ is preferable because selecting the features from the set $Fset$ is quite expensive even with caching – every feature template in the set must be cross-correlated with every input image in the training set.

In addition to the main results above, there were a few additional results worthy of summarization. In the experiments, we used inverted cross-validation. The immediate consequence was that the training sets were somewhat small, e.g., we had just 49 images of cars in a training fold. The advantage of small training sets was that the cost of evaluating the candidate fragments was reduced. As stated above, every fragment template in $Fset$ must be cross-correlated with every training image. The disadvantage of small training sets is that some classifiers may be prone to overfitting if the inputs are high-dimensional. In addition, overfitting could be made worse if the training sets are not representative of the whole problem, e.g., certain kinds of inputs are deliberately omitted, as we did with A2 multi-class rules.

The variants of A2 and A1 did not seem to suffer from overfitting. The other methods that were trained with wrappers did not have to deal with omitted inputs, but they had high-dimensional inputs. The other methods seemed to be somewhat confused by the large backgrounds in face detection, e.g., they may have overfitted to model the background pixels.

Fragment features have the attractive property that if they are suitable for a problem, then the classifier on top of the fragments can be simpler than a classifier of similar accuracy built on top of simpler features. In other words, suitable image patches are much more informative than simpler measured properties of images. For example, we got accurate results with linear support vector machines on top of fragment features. Linear support vector machines

(perceptrons) are known to be too simple for the recognition of non-trivial image classes when the input features are trivial (e.g., pixels).

Fragments also have their disadvantages. In their current form, i.e., using cross-correlation, they are not well-suited for non-rigid objects, scenes, or even rigid objects that have highly variable surface textures. If objects may be viewed from any arbitrary angle and there is no preprocessor for recovering rotations, then the lack of efficient means for achieving rotation invariance hurts the performance of fragment models.

Finally, we summarize the results of the blurring experiments. We saw that blurring the input images with Gaussian filters caused only graceful degradation of accuracy for all of the methods. Hence, it seems that basic categories of objects may be recognizable using only very low frequencies of image content.

## 5.4 Adaptive voting margin combiners

### 5.4.1 Test system design

In this study we examine the third question raised in the introduction of the current chapter. As explained in the introduction, the question emerges when there are several pre-made weak classifier modules available. As earlier, a module consists of features and a classifier that takes the features as input. In contrast to the earlier, we call the modules *component classifiers* to emphasize their incomplete nature.

We take a set of component classifiers and use an adaptive, i.e., learning, combination method to create a specialist node. The method that we designed is based on optimizing *voting margins* and uses non-monotonic modeling of component classifier confidence. The related theory was presented earlier in Sections 2.3.4 and 2.3.5 of Chapter 2. In what follows, the non-monotonic model that we use is called $M2$. Basically, each component classifier is associated with one of several feature spaces and the components are given post-processors that are optimized to maximize the voting margins of a combiner in a specialist node.

The test system in our experiments is minimalistic. Nothing

inessential is included. The test system is a cascade of two stages (nodes). The first stage, i.e., the root base classifier node, handles the subset of inputs that is considered easy to classify. It is based on a fairly traditional approach to the classification problem at hand. The second stage, i.e., the specialist node, handles the subset of inputs the first stage is unable to handle. The combination experiments are focused on the second stage specialist only. Because the question of interest is about combining classifiers *within* nodes, we consider it sufficient to use one suitable node.

The classification problem is a two-class detection problem from the medical image processing domain. The objects of interest are called microaneurysms. The non-objects consist of miscellaneous and varying background as found on images of human retina. Because the problem is unlike those examined earlier in this dissertation, we present a detailed overview below.

### 5.4.2 The application domain and related details

To explain the classification problem, we begin with background information related to the application domain. Diabetic retinopathy (DR) is the most common cause of blindness in individuals between the ages of 20 and 65. DR appears because the small capillary vessels of the retina are damaged by poor blood glucose control [KOA+04]. The presence of *microaneurysms* (MAs), small dilations in the retinal capillary vessels, is the earliest indicator of DR. Microaneurysm detection cannot be achieved without specialized equipment and labor-intensive methods, and requires the participation of the diabetic population in life-long screening programs. The standard procedure involves the examination of retinal fundus photographs for MAs by a medical professional – a tedious task that we seek to automate.

The microaneurysm detection problem has, of course, attracted considerable attention. A significant portion of the previous work [LBK83, SPSF92, SOM+96, COM+98] is based on the use of *angiograms*, retinal images enhanced by introducing fluorescent substances to the bloodstream before photography. However allergic responses to contrast-enhancing substances are not uncommon in patients and protocols avoiding angiographic techniques are simpler.

Previous proposals to solve this problem use matched filters [SPSF92, WK02b] or morphological constructs [WK02a] because typical MAs appear as roundish blobs whether one uses regular photographs or angiograms. In the first stage of our system (the root), we use an approach similar to [WK02a] to decide which test inputs will be delegated to the second stage (the specialist).

The two-staged cascade we present next may be contrasted with the two-class delegation rules from Section 5.3. In the cascade we are about to present, the root may only predict negatives directly (it is never confident that an input belongs to the positive class), and the features are different. The fragment features from the first half of this chapter were found almost useless in the MA detection problem. MAs clearly do not have the necessary rigidity of form for fragments to work well.

### 5.4.3 The first stage (root)

The primary purpose of the first stage (root) is to detect and exclude image regions that clearly contain no microaneurysms. The secondary purpose is preprocessing the remaining image regions: the input images, each of which depicts a large portion of the retina, are cut into smaller image patches suitable for the second stage.

The first-stage data consists of regular images of the retinas of patients. In training, there is also annotation that pinpoints the locations of the MAs. Each regular image may have many MAs in many locations. The first stage takes each regular retinal image and transforms it into a sequence of image patches. The image patches are then delegated to the second stage to be classified individually. The length of a sequence depends on the regular image that is being processed. For example, the first stage may determine that there are no potential MAs present in an image and produce an empty sequence.

The images used in the first stage are digitalizations from the photographic repository of the Diabetes Control and Complications Trial (DCCT) [The90], a comprehensive study spanning over a decade of diabetes research. In particular, our experiments use 71 images taken from 11 different patients. Image acquisition was done by using a Nikon Coolscan 4000 color film scanner at maximal resolution (4000 dpi) on the original slides produced at the

DCCT and obtaining 24-bit color samples. Image annotation was performed by qualified ophthalmologists, Dr. Ilkka Immonen and Dr. Petri Jalli.

The root stage has several sub-stages or preprocessing steps. First, in the RGB color model, the red and blue channels of the raw image are discarded (typically, the former is saturated while the latter is underexposed). Taking the green channel as a grayscale image, a Gaussian lowpass filter is applied for noise removal.

Then, after downsampling the image by a factor of three, local histogram normalization is performed by applying a linear mapping (as in the local contrast enhancement operator proposed by Walter and Klein [WK02a]). The extreme values of the output range interval are chosen so that the resulting histogram has 66% of the histogram width of the original retinal image. We denote the result of this initial preprocessing as $InitImg$.

The root then proceeds to segment the regions of interest in $InitImg$. A list of the coordinates of local intensity minima in all neighborhoods of 11 pixels is constructed. All minima under a certain threshold are taken. The threshold is 0.8 times the maximum intensity of $InitImg$. Then, a binary image is constructed by setting to one all the pixels corresponding to the locations in the list. The resulting binary image is called $MinImg$.

Next, we proceed by applying the morphological MA detection technique of Walter, Klein, Massin, and Zana [WKMZ00]. We construct a set of bottom hat image transforms using a sequence of straight structuring elements whose length is slightly larger than the diameter of the microaneurysms, and whose orientation spans from 0 to 170 degrees in steps of 10 degrees. Each of the 18 structuring elements is used to process the image $InitImg$ once. The resulting 18 images are then thresholded using an empirical estimate of the average response of microaneurysms. After thresholding, the set of the 18 binary images is denoted $\{BImg_1, \ldots, BImg_{18}\}$.

By applying the logical AND over the whole set of images

$$\{MinImg, BImg_1, \ldots, BImg_{18}\},$$

we get a binary image $Bselected$ that indicates the center points of the round and dark objects that are potential MAs. The bits that have the value of one in $Bselected$ become the centers of the square image patches that are delegated to the second stage of the

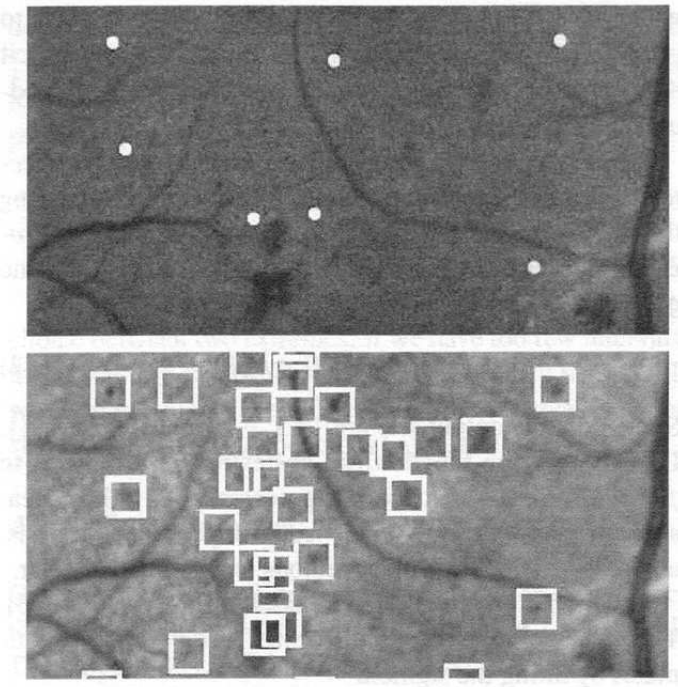


Figure 5.8: Retinal images. An original green channel subimage is shown on the top with the superimposed white dots indicating microaneurysm annotation. The results of the first stage of the cascade are shown on the bottom with the squares indicating the image patches that are selected for delegation.

cascade. The bits that have the value of zero in *Bselected* become the centers of the patches that are classified as negatives by the first stage and not delegated. In Figure 5.8, we show the delegated image patches of one particular retinal image.

Once the centers have been determined, the image patches to be delegated are cropped from the original green channel retinal image. The size of the cropped patches is large enough to allow the modeling of the immediate environment (context) of the potential microaneurysms.

### 5.4.4 Second-stage feature spaces

In the first stage, varying the values of the different parameters affects the quality of the results. The parameter settings that allow the capture of all of the positive patches (MAs) necessarily result in the capture of many negatives as well. Because MAs are uncommon and tiny compared to the size of the retinal images, the number of captured negatives is actually larger than the number of captured true positives. The class distribution of the delegated image patches is biased towards the negative class, i.e., the number of negatives is roughly ten times the number of true positives.

The second stage is given image patches each of which is classified individually. In what follows, image patches are called inputs. The inputs are scaled to the resolution of $60 \times 60$ pixels with the typical MA fitting within the central area of $20 \times 20$ pixels which we call the *focus area* (see Figure 5.9).

The grayscale inputs, interpreted as vectors $\mathbf{x} \in \mathbb{R}^d$ (with $d = 3600$), could, in principle, be used directly as in the ACMs in Section 5.3. In the current application domain, however, ordinary SVM kernels (linear, polynomial) with direct access to pixels did not work well. We can choose better sets of features for which simple dot products are useful measures of similarity. Each distinct subset of features with similar types and scales is associated with a distinct feature space within which dot products are calculated.

We use five distinct feature spaces. Let

$$\mathbf{S}_k : \mathbb{R}^d \to I_k, k \in \{1, \ldots, 5\},$$

such that $I_k$ is an inner product space of features. For our purposes, $I_k \subseteq \mathbb{R}^d$ suffices if equipped with the ordinary dot product. Similarity in the $k$th space is measured by $sim(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{S}_k(\mathbf{x}_i))^T \mathbf{S}_k(\mathbf{x}_j)$.

In practice, the microaneurysm detection problem is compatible with shape-based measures of similarity invariant to small translations and rotations. Intuitively, the perception of shapes is easier if object borders can be distinguished. We extract the shapes and borders by using naive graylevel thresholding segmentation similar to the rudimentary algorithms in Chapter 5.1 of [SHB99]. In that way, fewer assumptions are made than by the specialized methods in the first stage of the cascade. For increased robustness and representational power, we vary the threshold computing several segmentations per image.

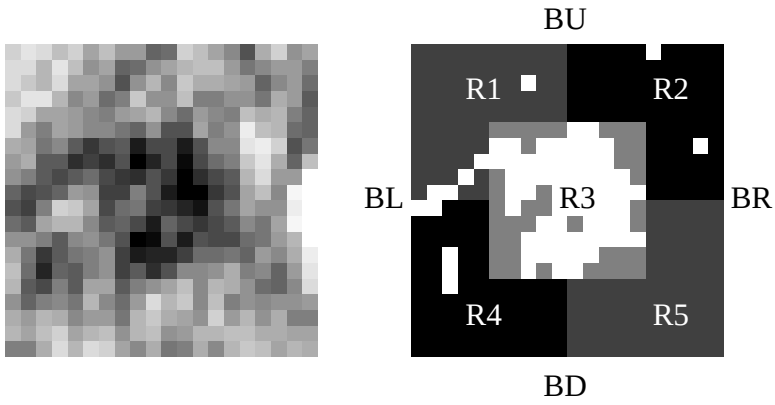



Figure 5.9: A microaneurysm (focus area, enhanced). One of the segmentations is shown on the right superimposed with an outline of the five regions of interest and the borders.

The feature spaces and the corresponding extractors are defined as follows:

1. *Raw segments.* The $20 \times 20$ gray values of pixels in the central focus area of an image patch are sorted into ascending order. We take the 3.125th, 6.25th, and 12.5th percentiles of the intensities as thresholds and segment the focus into foreground and background using each threshold. The segmentation matrices are then reshaped into a 1200-dimensional binary vector.

2. *Contrasts.* Segmenting as above, we compute the proportion of the foreground pixels in each of the five regions shown in Figure 5.9. Of the four peripheral regions, we mark by ones those with at least ten times the number of foreground pixels that the central region has (after normalizing for region size). For example, if 90% of the pixels of a peripheral region are foreground pixels and 9% of the pixels of the central region are foreground pixels, then the peripheral region is marked. The results of the four comparisons per segmentation are written into a 12-dimensional binary vector.

3. *Region counts.* Segmenting as above and using the same regions, the number of distinct connected components of fore-

ground pixels is computed within each region, and the results are written into a 15-dimensional vector. In the example of Figure 5.9, the shown segmentation produces the component counts of $2, 2, 1, 2$, and 0 for the regions.

4. *Intensity.* For each of the five regions, we compute the mean and standard deviation of the intensity of the pixels within the region. The results are written into a 10-dimensional feature vector.

5. *Connections.* For each of the three segmentations, we take the segment closest to the focus center and measure its reach. The focus area borders (BL, BU, BD, and BR) touched by the center segment are marked. The results go to a 12-dimensional binary vector.

### 5.4.5 Classification in the second stage

The second stage of the cascade uses linear SVMs equipped with the non-monotonic model of confidence from Section 2.3.4. Each SVM receives input from exactly one of the extractors $\mathbf{S}_k$ specified above. Denote $\mathbf{z} = \mathbf{S}_k(\mathbf{x})$. Let $Train = \{(\mathbf{z}_1, y_1), (\mathbf{z}_2, y_2), \ldots, (\mathbf{z}_N, y_N)\}$ denote the training set, where $y_i \in \{+1, -1\}$ denote class memberships (MA or not). The linear SVM learns a separating hyperplane in the space $I_k$ specified by the extractor $\mathbf{S}_k$. Recall that linear SVMs in primal form are extremely fast to evaluate after the feature values are known.

Recall that according to Equation (2.35) in Section 2.3.4, the $k$th SVM equipped with the non-monotonic model has three distinct parts: the feature extractor $\mathbf{S}_k$, the hyperplane classifier $f_k$ (without taking the sign), and the post-processor $r_k$ mapping values of $f_k$ to a discrete output set. If $r_k$ is the sign function, we have a standard SVM. In this study, the output set is $O_k = O = \{+1, 0, -1\}$ for all $k$, where 0 indicates that the SVM abstains from voting. For empirical comparisons between monotonic and non-monotonic models of confidence, we examine one of each type.

The monotonic model, $M1$, is based on Platt's rule [Pla00]: the probability estimates from Equation (2.33) are mapped to $O$ by mapping $f_k(\mathbf{z}) \in [0.5 - a_{low}, 0.5 + a_{high}]$ to $0 \in O$ using suitable thresholds $a_{low}$ and $a_{high}$. The values outside the interval are

mapped to the label of the most probable class. Note that Platt's rule was derived for a single SVM and it is not intended for optimizing how multiple component SVMs perform as a group.

The non-monotonic model, $M2$, is exactly as in Section 2.3.4. We use 42 intervals for each SVM. The middle intervals are as defined in Equation (2.37).

Initially, there is a set of pre-made component SVMs available. The components $(\mathbf{S}_k, f_k)$ are then given the post-processors $r_k$. Each post-processor needs the locations of 42 intervals and the mapping of each interval to the output set $O = \{+1, 0, -1\}$. The locations and mappings are estimated by running training data through the SVMs to get the values $f_k(\mathbf{z}_i)$ for each $\mathbf{z}_i$ in the training data. These values are required for the estimates. The locations are estimated as in Equation (2.37). Each of the 40 middle intervals contains 2.5% of the data. The mappings are estimated using an optimization procedure that is described further below in the context of the combiner rule. We use the *sign* function as the *signmap* function that is required in Equation (2.38), i.e., positive values are mapped to $+1$ and negative values are mapped to $-1$. As the result of the above process, the SVMs are adapted to work together in a problem-specific manner.

### The second stage combiner

Our combiner is a majority voting rule on top of a collection of SVMs equipped with a model of confidence ($M1$ or $M2$). The rule is shown in Equation (2.41), but for convenience we repeat it here in an appropriate form. Given $M$ SVMs, the rule is

$$Comb(\mathbf{x}) = sign(\sum_{k=1}^{M} g_k(\mathbf{x})), \tag{5.54}$$

where $g_k(\mathbf{x}) = (r_k \circ f_k \circ \mathbf{S}_k)(\mathbf{x})$. Comparing Equation (5.54) to Equation (2.41), note that the former is incapable of zero output. The reason is that there is no need to let *Comb* indicate lack of confidence. In the current application, the combiner is in the tail of the cascade and inputs cannot be delegated further.

The second model of confidence ($M2$) requires the optimization of the post-processors $r_k$, where the degrees of freedom are limited

to the tagging of certain intervals as abstain intervals (2.38). Having defined the rule *Comb* (5.54), we can now describe the optimization procedure. The optimization criterion is based on *voting margins*. The empirical voting margin over a training set is given in Equation (2.39).

Denote the parameters of the optimization problem by the matrix $\boldsymbol{\tau} = [\boldsymbol{\tau}_1, \boldsymbol{\tau}_2, \ldots, \boldsymbol{\tau}_M]$, where each $\boldsymbol{\tau}_k$ is a vector of bits tagging the abstain intervals of the $k$th post-processor. The optimization problem is defined as

$$\boldsymbol{\tau}^* = \arg\max_{\boldsymbol{\tau}} \sum_{k=1}^{M} \left( \sum_{i=1}^{N} y_i g_k(\mathbf{x}_i; \boldsymbol{\tau}_k) \right). \tag{5.55}$$

From (5.55), two properties of this simple solution are seen: the votes of relatively poor classifiers can be censored, and new component classifiers can be added afterwards without retraining of the existing component classifiers or post-processors.

In practice the class priors and the misclassification costs are unbalanced, and we must control classifier *sensitivity* versus *specificity*. Sensitivity is the fraction of positives detected as such, and specificity is the corresponding fraction of negatives detected as such. Let $Id^+$ and $Id^-$ denote the indices of the positive and negative inputs in the training set. We modify the optimization problem (5.55) by including weights $w^+, w^- \in \mathbb{R}$ for tuning sensitivity against specificity. The modified problem is

$$\max_{\boldsymbol{\tau}} \left( \frac{w^+}{|Id^+|} \sum_{i \in Id^+} y_i \sum_{k=1}^{M} g_k(\mathbf{x}_i; \boldsymbol{\tau}_k) + \frac{w^-}{|Id^-|} \sum_{i \in Id^-} y_i \sum_{k=1}^{M} g_k(\mathbf{x}_i; \boldsymbol{\tau}_k) \right). \tag{5.56}$$

The solution is again denoted by $\boldsymbol{\tau}^*$. In the current application, we used the values $w^+ = 1$ and $w^- = 1.2$ to emphasize the margins of the negatives.

Optimization is easy because the parameters of the intervals can be optimized independently of each other. An interval $int_{k,j}$ can be set to abstain if the choice improves the portion of the score (5.56) that depends on the $k$th component classifier and the training data falling in the $j$th interval of that classifier.

Our tests using 15 SVMs indicated that empirical voting margins correlated positively with the generalization accuracy of the

combiner. In Section 2.3.5 we also presented a simple bound that demonstrated how the relationship between empirical voting margins and true accuracy may be examined theoretically. It may be assumed that the basic design of the combiner is sound.

Based on the tests, the abstain intervals of the non-monotonic model $M2$ were concentrated close to the hyperplanes of the component SVMs. For example, one machine had four consecutive non-empty abstain intervals near the hyperplane of the machine. Yet some of the more unreliable SVMs had abstain intervals associated with high absolute values of $f_k(\mathbf{z})$, i.e., far from the hyperplanes. These abstain intervals that are associated with high absolute values can be called *outlier intervals*. We noticed that there were non-abstaining intervals in between abstain intervals containing the hyperplanes and outlier intervals. Thus, genuinely non-monotonic models were found for the unreliable SVMs.

### 5.4.6 Experimental results

To estimate the generalization ability of the cascade, we used standard 10-fold cross-validation [RH95]. Our dataset consisted of 710 positive and 6930 negative inputs (image patches) which we randomly partitioned into 10 folds each having the same relative class frequencies as before partitioning.

The inputs were sampled from the original retinal images with the help of the first stage that discarded most of the obviously negative patches from consideration (e.g., featureless, or "blank" patches) without discarding any of the positive patches.

During each of the 10 iterations, 9 folds were used for training and 1 for testing. The training process constructed the second stage of the cascade that consisted of the following:

1. A set of 15 component SVMs, each with the simple output set $\{+1, 0, -1\}$. Using the 5 feature spaces described in Section 5.4.4, we trained 3 SVMs per space.

2. A combiner. Both of the models $M1$ and $M2$ were tested with the combiner. For $M2$ we used the optimization procedure described in the previous subsection.

Each SVM was trained with the full set of positive inputs available in the current training folds. Because the relative class fre-

Table 5.2: The averaged results of cross-validation for models $M1$, $M2$ and the best individuals.

| | 2nd stage sensitivity | 2nd stage specificity | system sensitivity | system specificity |
|---|---|---|---|---|
| $M1$ | 88% | 78% | 88% | 96% |
| $M2$ | 88% | 81% | 88% | 96% |
| *individual* | 87% | 74% | 87% | 95% |

quencies were quite unbalanced (639 to 6237), we did not train with all the available negative inputs. Given $n$ positive training inputs, we sampled $n$ random negative training inputs from the available ones. This sampling was repeated for each SVM. Hence, the 15 component SVMs within the combiner were trained with different but overlapping training data. In this sense, the setting had some resemblance to bagging of classifiers [Bre96].

We estimated (and tested) that the above process could help in reducing the effects of mislabeled negative inputs. Sampling reduces the ratio of mislabeled negatives to true positives in each SVM-specific training set. Although the data was labeled by medical professionals, the task of labeling is tedious, and it is possible that sometimes genuine positives are missed. In our case, examining the evolution of microaneurysms in patient-specific time series of retinal images revealed that mislabeled negatives existed. Sometimes microaneurysms had the correct label at times $t$ and $t + 2$ while lacking the label at the time $t + 1$.

With model $M1$ we used the abstain interval of $[0.475, 0.525]$, which was a fairly conservative choice downplaying the overall role of abstains. The parameters $\mathcal{A}_{Platt}$ and $\mathcal{B}_{Platt}$ required by $M1$ (see Section 2.3.3) were estimated by withholding 30% of the available training data for this purpose only.

The averaged cross-validation results of models $M1$, $M2$ and the best individual classifiers are shown in Table 5.2. The best individual classifier was found on a per fold basis, e.g., during the first iteration of cross-validation the best could be a machine using the second feature space and during the second iteration it could be a machine using the fifth feature space. No machine or feature space won the majority of the folds, and thus there is no overall

best feature space.

The first and second columns show the second stage (combiner) mean sensitivity and specificity statistics from cross-validation. The remaining columns show the estimated statistics of the whole system including both stages of the cascade. The estimates are based on 80% of the negatives getting caught by the first stage, while 100% of the positives pass.

Clearly, the better the first stage is at removing image regions that contain no positives, the smaller the number of inputs reaching the second stage and the smaller the average classification times are for negatives. For correctly detected positives, classification times are never reduced. Because positives are in the minority (in the test data), the classification times of positives have little impact overall.

From the viewpoint of the third question posed in the introduction of this chapter, i.e., can pre-made components be combined adaptively, only the first two columns of Table 5.2 matter. The second stage sees the difficult inputs while the first stage (i.e., the traditional MA detector) classifies large amounts of easy inputs. The first two columns suffice to show that, given the input distribution that the second stage sees, the adaptive combiner works better than the best individual classifiers. Both the non-adaptive monotonic combiner $M1$ and the adaptive non-monotonic combiner $M2$ beat the individual classifiers. Especially $M2$ is noteworthy. It is considerably better than the individuals and slightly better than $M1$.

If the estimated effects of the first stage are taken into account, i.e., as in the third and fourth columns, the differences between $M1$ and $M2$ are not visible given the numeric precision of the table.

### 5.4.7 Summary

In this section we examined the third question raised in the introduction of the current chapter. The question asked whether premade classifier modules or components, which are too weak individually, can be combined adaptively to work together in specialist nodes. As explained in the introduction, adaptive combination involves learning in contrast to using fixed rules. We presented the combiner $M2$ that uses non-monotonic confidence modeling and learns at the level of SVM post-processors. The approach differs

from boosting in the sense that boosting algorithms do not combine pre-made classifiers.

Based on the empirical results, we can answer the question and say that adaptive combination of pre-made components does work when $M2$ is used. In addition, our earlier analysis in Section 2.3.5 led to a simple bound that demonstrated how the relationship between empirical voting margins and true accuracy may be examined theoretically in the context of $M2$. The basic design of the combiner $M2$ is sound.

In the experiments, we compared $M2$ to $M1$, the non-adaptive monotonic combiner, and to best individual classifiers. Including $M1$ was illustrative because monotonic modeling of SVM outputs has been established in the literature (e.g., by Platt [Pla00]). The experimental results indicated that using any combiner ($M2$ or $M1$) led to an accuracy advantage over using any one component classifier (or feature space) individually. The combiner $M2$ was considerably better than any individual classifier and slightly better than $M1$.

To see why $M2$ should be more accurate than $M1$, it is useful to consider some of the experimental observations that we made earlier. First note that abnormally high absolute values of SVM responses (before taking the sign or using a post-processor) may indicate outliers, i.e., inputs that are dissimilar to any in the training set. In $M1$ such outliers would be classified with great confidence. Examining the results of voting margin maximization of $M2$, we noticed that there were outlier intervals, i.e., intervals that were earlier defined as abstain intervals associated with high absolute values. We noticed that there were non-abstaining intervals in between abstain intervals containing the hyperplanes and outlier intervals. Hence, genuinely non-monotonic modeling was supported by the data.

In contrast to some of the other studies in this dissertation, we did not provide speed versus accuracy plots in this study. In our opinion, such plots were unnecessary given the question that we sought to answer here.

CHAPTER 6

# Conclusions

The proposed framework uses tree-like organization of base classifiers to achieve conditional exclusion of computations on a per input basis. This conditional exclusion leads to efficient classification of inputs. Efficiency means the ability to make controlled trade-offs between accuracy and computational effort (speed). An input image is delegated down the tree along a unique path until it reaches a base classifier node that is confident and predicts. In contrast to decision trees, any node may terminate delegation. The primary function of the proposed trees is to exclude the computations associated with the paths that are not taken by the input. Delegation involves the subproblems of determining node confidence and where the input goes next. The general idea of organizing the classification process as a tree was, of course, not novel. For example, in decision trees the path of an input reflects maximum information gain about the class. In [BG05] the path of an input reflects the narrowing down of the set of classes.

In Chapter 2, resource consumption (loss) was made explicit in the notation and analysis. Without explicit resource consumption, there are properties of organization that cannot be seen properly (e.g., subsumption). Resource consumption is measured precisely, e.g., in seconds. Imprecise and more indirect measures, e.g., the expected evaluation depth penalty of Geman and Jedynak [GJ01], are not realistic because base classifiers are not equal in resource consumption. When multiple kinds of features are used, the resource consumption of activated base classifiers may vary significantly. Like Geman and Jedynak, we chose additive losses. Addi-

tive losses are easy to understand intuitively, e.g., one may state preferences such as that one classification mistake is worth two seconds of time.

In base classifier design, we require classifiers that offer something beyond the predicted class labels as output – something that can be interpreted as related to prediction confidence. Large-margin classifiers, e.g., support vector machines and the boosted root of Chapter 4, can offer the classification margins. The classification margins can be interpreted by confidence models. In Chapter 2, we examined monotonic and non-monotonic models. Platt's monotonic model was a well-known example of the former, while a contributed model served as an example of the latter. It was explained that, in the current framework, each monotonic model reduces to two threshold values. Correspondingly, any two thresholds are compatible with multiple probabilistic models of confidence, and we gain nothing by identifying the precisely correct probabilistic model. We choose and use thresholds directly without mapping to probability values. For example, the ACMs of Chapter 5 have directly chosen thresholds. Related to large-margin classifiers (SVMs), it was also noted that two-class classifiers can be combined in several ways to yield multi-class capable classifiers. Some combination schemes (e.g., one-vs-rest) are more efficient than others (e.g., one-vs-one). In the preceding chapters, there was a tendency towards the use of schemes that resembled one-vs-rest.

In the introduction, it was stated that the first goal of this dissertation was to show that the proposed framework is satisfactory for the purpose of understanding and modeling efficient classification of images depicting real-world objects and scenes. The second goal was to detect unnecessary conventions related to certain subproblems of efficient classification. Three categories of questions were identified, each of which includes vital questions related to how satisfactory the framework is. The categories were *trade-off optimization*, *classifier tree organization*, and *delegation and confidence*.

Regarding *trade-off optimization*, the following was presented. *First*, the problem of selecting root node features was addressed. From Chapter 3, recall that the available range of trade-offs is easily limited by root node bottlenecks. The root is special in that the limitations of the root cannot be overcome by any delegation rules.

In other words, unsuitable root nodes lead to insufficient control over trade-offs. In Chapter 3, a hypothesis was contributed. The hypothesis makes claims about what kind of features should be used in root nodes to avoid root bottlenecks. The hypothesis was inspired by recent research on the ability of humans to recognize scenes very rapidly. Some preconditions were stated. The preconditions limit the claims to a particular sort of classification problems. The preconditions are compatible with coarse-to-fine organization of classes, e.g., the use of superclasses. If the preconditions are met, the hypothesis claims that there is a good chance of a solution based on global features that are summations or other coarse statistics of local, possibly oriented features. The main experiment of Chapter 3 satisfied the preconditions and claims of the hypothesis. A small set of crude Gabor filters was used to compute global statistics of images. The statistics were normalized sums of local filter responses. A related experiment was done in Chapter 4. That experiment also satisfied the preconditions and claims of the hypothesis. In that case, local binary pattern (LBP) features were used. Local binary pattern features are impressively efficient. In contrast to the first experiment, the image backgrounds were not capable of helping in the classification task. Hence, two different experiments were contributed and the results agreed with the hypothesis.

*Second*, in Chapter 4 and in the first half of Chapter 5, more experimental evidence was contributed. The evidence supported the claim that if the framework is implemented properly, then speed versus accuracy trade-offs can be controlled after training by the use of simple parameters ($T$ and $T_j$). Proper implementation means that there are no root bottlenecks, confidence modeling works, and that the delegation rules allow any node to terminate. It is especially important to avoid the kind of input filtering that is done by the cascades of Viola and Jones and decision trees, i.e., nodes are trained using inputs that pass through earlier nodes. That kind of filtering is not compatible with the idea of controlling trade-offs after training. In the experiments, the available range of trade-offs was large and the chosen trade-off could be changed at will. Based on the evidence, it seems that it is unnecessary to target a specific trade-off in the classifier training stage.

Related to *classifier tree organization*, the following was discovered. *First*, in Chapter 4, we asked where the organization of

the base classifier nodes comes from (supposing the organization is problem-specific). Experiments related to the problem were conducted. In the experiments, the *organization-centered* approach was used. On a more detailed level, coarse-to-fine organization with explicit superclasses was used. It was assumed that confusing broad (super)classes is more serious than confusing narrow classes. An experimental procedure was designed. The procedure could produce coarse-to-fine trees of classes by using visual queries posed to human observers. The idea was that by measuring which classes the observers find similar, the result of the procedure is a tree in which the distance between nodes (classes) is proportional to (hierarchic) mistake loss. The contributed experimental evidence supported the use of the coarse-to-fine organization-centered approach. The results showed that the approach was compatible with the requirements of the framework and led to efficient classification (see *trade-off optimization* above). The classifier tree would have been competent with formal hierarchic mistake losses because most mistakes were within-superclass mistakes.

*Second*, in the first half of Chapter 5, we asked whether the framework requires problem-specific organization of the classifier nodes. Related experimental evidence was contributed. The evidence supported the claim that, at least sometimes, the framework does not require problem-specific organization. This evidence was based on the use of the proposed *A2 multi-class delegation rules*. All non-empty subsets of object classes were given specialist nodes. Each specialist node was formed from a set of simpler modules (RDMs) at runtime when an input required that particular specialist node. The nodes did not exist as individual memory-consuming units. It was also demonstrated that the modules were able to handle novel kinds of negative inputs. If this ability was missing, then there would be certain dependencies between the modules at the training stage. These dependencies could be as undesirable as allowing each node to exist individually in memory. In general, non-modular specialist node implementations, such as those from Chapter 4, are not compatible with the idea of problem-independent organization.

Further, the following results related to *delegation and confidence* were presented. *First*, in Chapter 4, we asked whether predictions can be combined efficiently over multiple views of objects under

motion. Efficiency in this task was defined as being inversely proportional to the sum of total losses over motion sequences (since motion itself cannot be made faster). Experimental evidence was contributed. The evidence supported the affirmative answer, e.g., the proposed delegation rules were not limited to classifying single views. Each individual prediction was a ranked list of class labels. Simple voting-based combination of ranked lists was sufficient, given that the underlying classifier tree was capable of simple motion segmentation.

*Second*, in the first half of Chapter 5, a theoretical analysis of the proposed *A2 multi-class delegation rules* was contributed. In the analysis, it was assumed that monotonic models of confidence are appropriate. It was proved that the rules can, in principle, exceed the accuracy of pure RDM committees. This accuracy can then be traded for speed. Intuitively, the point was that although the rules might seem ad hoc at first sight, they can be understood in detail and that they are not inherently flawed or limited to poor performance.

*Third*, in Chapter 2, a non-monotonic confidence modeling approach was contributed and analyzed. The approach was based on adaptive combination of classifiers and maximization of empirical voting margins. In the analysis, it seemed that combining classifiers by using the approach is not especially dangerous because increasing the number of combined classifiers does not necessarily increase the risk of overfitting. The approach was later called *M2*. In the second half of Chapter 5, we asked whether pre-made classifier modules can be combined adaptively to work together in specialist nodes. Recall that in the first half of the chapter combination was not adaptive. It was explained why boosting does not allow *pre-made* modules. Related experimental evidence was contributed. The evidence supported the claim that adaptive combination of pre-made modules works in practice. The approach *M2* was compared to *M1*, the non-adaptive monotonic combiner. Comparisons were appropriate because monotonic modeling is well-established, e.g., by Platt [Pla00]. It was found that *M2* was slightly better than *M1*. Based on the data, non-monotonic modeling was appropriate.

Finally, we addressed one question that was not covered by the three categories above. In the first half of Chapter 5, we asked if the framework is useful in improving the efficiency of feature selec-

tion and training. Related experimental evidence was contributed. The evidence supported the positive answer. The efficiency of feature selection and training was improved when ACMs (in the root node) were used to constrain the sampling of candidate features in RDMs (in the specialist nodes). For the demonstration of improved efficiency, Ullman's image fragments were used as RDM features. Fragments were a practical choice because they are known to produce accurate classifiers, but their selection is computationally expensive.

Based on the above results, the proposed framework seems to be satisfactory. Related to *trade-off optimization*, the framework allows control over trade-offs, the available range of trade-offs is wide, and there are ways to avoid root node bottlenecks. Related to *classifier tree organization*, the requirements (e.g., lighter nodes precede heavier nodes, every node can terminate) can be satisfied in practice. Organization can be problem-specific or not. In addition, the idea of hierarchic mistake losses seems to be compatible with the framework. Related to *delegation and confidence*, it was demonstrated that compatible and simple delegation rules and confidence models are readily available – even when it is necessary to predict from multiple views (under motion). The rules and models can be analyzed and understood in detail. In addition to being satisfactory for the intended purpose, the framework seems to allow improving the efficiency of feature selection and training.

In the process of experimentation, it was found that some common conventions appear to be unnecessary, or even counterproductive. First, it was found that speed versus accuracy trade-offs can be controlled after training, i.e., changed without further training. In this, it is vital to avoid the conventional input filtering done by cascades and decision trees. Second, coarse-to-fine trees of classes were produced by using visual queries posed to human observers. Through this, it seems possible to get a tree in which the distances between nodes encode information about the cost of confusing pairs of class labels. For conventional feature-centered approaches, this is not possible. Hence, one should not consider feature-centered approaches as the obvious choice. Third, it was demonstrated that the convention assumption of problem-specific tree classifier organization is not always justified.

Regarding future directions, a few interesting topics were noted

in the earlier chapter summaries. For an example related to classifier tree organization, no experiments involving a very large number of classes were contributed. It would certainly be interesting to see if scaling issues emerge. However, instead of rather specific topics related to particular methods, it is more interesting to consider broader and more urgent lines of development.

While delegation operates in a sequential manner, there is room for parallel processing within the activated nodes. Hence, parallelized combiners that operate within nodes are likely important in allowing more computational effort to be spent per second. It should be investigated if the framework could offer expanded support for understanding and creating this kind of combiners. Furthermore, it would be interesting to see if delegation could be replaced by some other mechanism that has better potential for parallelization. However, the idea of conditional exclusion of unnecessary computations implies that some sequential processing is necessary for obtaining savings in the resource losses.

# References


[ABI+05] I. Autio, J. Borras, I. Immonen, P. Jalli and E. Ukkonen. A Voting Margin Approach for the Detection of Retinal Microaneurysms. In *Proceedings of the IASTED International Conference on Visualization, Imaging, and Image Processing*, pages 511–517. Benidorm, Spain, 2005.

[AE03] I. Autio and T. Elomaa. Flexible view recognition for indoor navigation based on Gabor filters and support vector machines. *Pattern Recognition*, 36(12):2769–2779, 2003.

[AEK01a] I. Autio, T. Elomaa and T. Kurppa. Robot landmark learning with SVMs. In *Proceedings of the 7th Scandinavian Conference on Artificial Intelligence (SCAI01)*, pages 157–158. IOS Press, 2001.

[AEK01b] I. Autio, T. Elomaa and T. Kurppa. Support vector learning of landmarks for a mobile robot. In *Proceedings of the 2001 International Conference on Artificial Intelligence (IC-AI01)*, pages 151–157. CSREA Press, 2001.

[AGW97] Y. Amit, D. Geman and K. Wilder. Joint Induction of Shape Features and Tree Classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11):1300–1305, 1997.

[AL04] I. Autio and J. Lindgren. Attention-driven parts-based object detection. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI)*, pages 917–921. Valencia, Spain, 2004.

[AL06] I. Autio and J. Lindgren. Online learning of Discriminative Patterns from Unlimited Sequences of Candidates. In *Proceedings of the 18th International Conference on Pattern Recognition (ICPR06)*, pages 437–440. Hong Kong, China, 2006.

[AM02] H. Alhichri and M.Kamel. Multi-resolution image registration using multi-class Hausdorff fraction. *Pattern Recognition Letters*, 23(1):279–286, 2002.

[AR02] S. Agarwal and D. Roth. Learning a sparse representation for object detection. In *Proceedings of the 7th European Conference on Computer Vision (ECCV)*, pages 113–130. Springer-Verlag, Copenhagen, Denmark, 2002.

[Arr70] K. Arrow. *Social Choice and Individual Values (2nd edition)*. Yale University Press, New Haven, CT, USA, 1970.

[ASS00] E. Allwein, R. Schapire and Y. Singer. Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers. *Journal of Machine Learning Research*, 1:113–141, 2000.

[Aut06] I. Autio. Using natural class hierarchies in multi-class visual classification. *Pattern Recognition*, 39(7):1290–1299, 2006.

[Bar03] M. Bar. A cortical mechanism for triggering top-down facilitation in visual object recognition. *Journal of Cognitive Neuroscience*, 15(4):600–609, 2003.

[BBC94] E. Bellissant, J. Benichou and C. Chastang. A Microcomputer Program for the Design and Analysis of Phase 11 Cancer Clinical Trials with Two Group Sequential Methods, the Sequential Probability Ratio Test, and the Triangular Test. *Computers and Biomedical Research*, 27(1):13–26, 1994.

[BBFS00] A. Broggi, M. Bertozzi, A. Fascioli and M. Sechi. Shape-based Pedestrian Detection. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 215–220. Detroit, MI, USA, 2000.

[BBV00] J. Bruce, T. Balch and M. Veloso. Fast and inexpensive color image segmentation for interactive robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2061–2066. Takamatsu, Japan, 2000.

[BCSTW00] K. Bennett, N. Cristianini, J. Shawe-Taylor and D. Wu. Enlarging the Margins in Perceptron Decision Trees. *Machine Learning*, 41(3):295–313, 2000.

[BFOS84] L. Breiman, J. Friedman, R. Olshen and C. Stone. *Classification and Regression Trees*. Chapman and Hall, New York, NY, USA, 1984.

[BG05] G. Blanchard and D. Geman. Hierarchical Testing Designs for Pattern Recognition. *The Annals of Statistics*, 33(3):1155–1202, 2005.

[Blu67] M. Blum. A Machine-Independent Theory of the Complexity of Recursive Functions. *Journal of ACM*, 14(2):322–336, 1967.

[BM92] K. Bennett and O. Mangasarian. Robust Linear Programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1:23–34, 1992.

[BM94a] K. Bennett and O. Mangasarian. Multicategory discrimination via linear programming. *Optimization Methods and Software*, 3:29–39, 1994.

[BM94b] K. Bennett and O. Mangasarian. Serial and parallel multicategory discrimination. *SIAM Journal on Optimization*, 4(4):722–734, 1994.

[BM02] P. Bartlett and S. Mendelson. Rademacher and Gaussian Complexities: Risk Bounds and Structural Results. *Journal of Machine Learning Research*, 3:463–482, 2002.

[Bre96] L. Breiman. Bagging Predictors. *Machine Learning*, 24(2):123–140, 1996.

[Bre01] L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

[BV04] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK, 2004.

[CBL06] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, Cambridge, UK, 2006.

[CDV03] M. Coimbra, M. Davies and S. Velastin. Pedestrian detection using MPEG-2 motion vectors. In *Proceedings of the International Workshop on Image Analysis for Multimedia Interactive Services*, pages 164–169. London, UK, 2003.

[Chu92] C. Chui. *An Introduction to Wavelets*. Academic Press, London, UK, 1992.

[CJR+98] M. Carreira, J.Orwell, R.Turnes et al. Perceptual grouping from Gabor filter responses. In *Proceedings of the Ninth British Machine Vision Conference*. Southampton, UK, 1998.

[COM+98] M. Cree, J. Olson, K. McHardy, P. Sharp and J. Forrester. A fully automated comparative

microaneurysm digital detection system. *Eye*, 11:622–628, 1998.

[CS01] K. Crammer and Y. Singer. On the Algorithmic Implementation of Multi-class SVMs. *Journal of Machine Learning Research*, 2:265–292, 2001.

[CST00] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Methods*. Cambridge University Press, Cambridge, UK, 2000.

[CT91] T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley & Sons, West Sussex, UK, 1991.

[CV95] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20(3):273–297, 1995.

[DB95] T. G. Dietterich and G. Bakiri. Solving Multiclass Learning Problems via Error-Correcting Output Codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[DC00] S. Dumais and H. Chen. Hierarchical classification of Web content. In *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval*, pages 256–263. ACM Press, New York, US, Athens, GR, 2000.

[DHS00] R. Duda, P. Hart and D. Stork. *Pattern Classification*. John Wiley & Sons, West Sussex, UK, 2000.

[DK05] K. Duan and S. Keerthi. Which Is the Best Multiclass SVM Method? An Empirical Study. In *6th International Workshop on Multiple Classifier Systems (MCS2005)*, pages 278–285. Seaside, CA, USA, 2005.

[DKS04] O. Dekel, J. Keshet and Y. Singer. Large margin hierarchical classification. In *Proceedings*

*of the Twenty-first International Conference on Machine Learning (ICML2004)*, pages 209–216. Banff, Alberta, Canada, 2004.

[DS01] M. DeGroot and M. Schervish. *Probability and Statistics (3rd Edition).* Addison-Wesley, Boston, MA, USA, 2001.

[EHOK02] M. Elad, Y. Hel-Or and R. Keshet. Rejection based classifier for face detection. *Pattern Recognition Letters*, 23(12):1459–1471, 2002.

[Fis70] P. Fishburn. *Utility Theory for Decision-Making.* John Wiley & Sons, West Sussex, UK, 1970.

[FNSH04] S. Frintrop, A. Nüchter, H. Surmann and J. Hertzberg. Saliency-based Object Recognition in 3D Data. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'04)*, pages 2167–2172. Sendai, Japan, 2004.

[GBD92] S. Geman, E. Bienenstock and R. Doursat. Neural Networks and the Bias/Variance Dilemma. *Neural Computation*, 4(1):1–58, 1992.

[GJ01] D. Geman and B. Jedynak. Model-based classification trees. *IEEE Transactions of Information Theory*, 47(3), 2001.

[GJLAMBGM05] P. Gil-Jimenez, S. Lafuente-Arroyo, S. Maldonado-Bascon and H. Gomez-Moreno. *Shape Classification Algorithm Using Support Vector Machines for Traffic Sign Recognition*, vol. 3512 of *Lecture Notes in Computer Science*, pages 873–880. Springer-Verlag, New York, NY, USA, 2005.

[Hay99] S. Haykin. *Neural Networks: A Comprehensive Foundation.* Prentice Hall, New Jersey, USA, 1999.

[Her01] R. Herbrich. *Learning Kernel Classifiers: Theory and Algorithms*. MIT Press, Cambridge, MA, USA, 2001.

[HGW96] J. Huang, S. Gutta and H. Wechsler. Detection of human faces using decision trees. In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pages 248–252, 1996.

[HL02] C. Hsu and C. Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13:415–425, 2002.

[HT98] T. Hastie and R. Tibshirani. Classification by Pairwise Coupling. In *Advances in Neural Information Processing Systems*, vol. 10. MIT Press, Cambridge, MA, USA, 1998.

[Hut01] M. Hutter. Distribution of Mutual Information. In *Advances in Neural Information Processing Systems*, vol. 14, pages 399–406. MIT Press, Cambridge, MA, USA, 2001.

[Hut02] M. Hutter. Robust Feature Selection using Distributions of Mutual Information. In *Proceedings of the 18th International Conference on Uncertainty in Artificial Intelligence (UAI-2002)*, pages 577–584. Morgan Kaufmann, San Francisco, CA., 2002.

[JCMJ03] S. Johnson, L. Cohen, K. Marks and K. Johnson. Young infants' perception of object unity in rotation displays. *Infancy 4*, 4:285–295, 2003.

[JJ94] M. Jordan and R. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6(2):181–214, 1994.

[JJNH91] R. Jacobs, M. Jordan, S. Nowlan and G. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

[JKF01] O. Jesorsky, K. Kirchberg and R. Frischholz. Robust Face Detection Using the Hausdorff Distance. In *Proceedings of the 3rd International Conference on Audio and Video based Person Authentication*, pages 90–95. Halmstad, Sweden, 2001.

[Joa98] T. Joachims. Making large-scale support vector machine learning practical. In B. Schölkopf, C. Burges and A. Smola, eds., *Advances in Kernel Methods: Support Vector Machines*, pages 169–184. MIT Press, Cambridge, MA, USA, 1998.

[JP87] J. Jones and L. Palmer. An evaluation of the two-dimensional Gabor filter model of simple receptive fields in cat striate cortex. *J. Neurophysiology*, 58(6):1233–1258, 1987.

[JV03] M. Jones and P. Viola. Fast Multi-view Face Detection. Tech. Rep. TR2003-096, Mitsubishi Electric Research Laboratories, 2003.

[KB01] W. Kropatsch and H. Bischof. *Digital Image Analysis: Selected Techniques and Applications*. Springer-Verlag, New York, NY, USA, 2001.

[KOA+04] T. Kawasaki, N. Ogata, H. Akanuma et al. Postprandial plasma fructose level is associated with retinopathy in patients with type 2 diabetes. *Metabolism*, 5(53):583–588, 2004.

[KS97] D. Koller and M. Sahami. Hierarchically classifying documents using very few words. In *Proceedings of the Fourteenth International Conference on Machine Learning (ICML-97).*, pages 171–178. Nashville, Tennessee, USA, 1997.

[LBK83] B. Lay, C. Baudoin and J. Klein. Automatic detection of microaneurysms in retinopathy fluoroangiogram. In *Proceedings of SPIE*, vol. 432, pages 165–173, 1983.

[LS03] B. Leibe and B. Schiele. Analyzing Appearance and Contour Based Methods for Object Categorization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'03)*, pages 409–415. Madison, WI, USA, 2003.

[LV93] M. Li and P. Vitanyi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer-Verlag, New York, NY, USA, 1993.

[LZ04] S. Li and Z. Zhang. FloatBoost Learning and Statistical Face Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1112–1123, 2004.

[MB98] A. Martinez and R. Benavente. The AR Face Database. Tech. rep., CVC Technical Report 24, 1998.

[Mit97] T. Mitchell. *Machine Learning*. McGraw-Hill, New York, NY, USA, 1997.

[MKS94] S. Murthy, S. Kasif and S. Salzberg. A System for Induction of Oblique Decision Trees. *Journal of Artificial Intelligence Research*, 2:1–32, 1994.

[MKS00] J. Miura, T. Kanda and Y. Shirai. An active vision system for real-time traffic sign recognition. In *Proceedings of the IEEE Intelligent Transportation Systems*, pages 52–57. Dearborn, MI, USA, 2000.

[MP88] M. Minsky and S. Papert. *Perceptrons: An introduction to Computational Geometry (2nd edition)*. MIT Press, Cambridge, MA, USA, 1988.

[MR01] A. Mojsilovic and B. Rogowitz. Capturing image semantics with low-level descriptors. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, pages 18–21. Thessaloniki, Greece, 2001.

[MR03] R. Meir and G. Rätsch. *An Introduction to Boosting and Leveraging*, pages 119–184. No. 2600 in Lecture Notes in Artificial Intelligence. Springer-Verlag, New York, NY, USA, January 2003.

[MRMN98] A. McCallum, R. Rosenfeld, T. Mitchell and A. Ng. Improving text classification by shrinkage in a hierarchy of classes. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML-98).*, pages 359–367. Madison, Wisconsin, USA, 1998.

[Nov62] A. Novikoff. On convergence proofs on perceptrons. In *Symposium on Mathematical Theory of Automata*, vol. 12, pages 615–622. Polytechnical Institute of Brooklyn, 1962.

[NS98] R. Nelson and A. Selinger. Large-Scale Tests of a Keyed, Appearance-Based 3D Object Recognition System. *Vision Research, special issue on computational vision*, 38(15-16), August 1998.

[OPM01] T. Ojala, M. Pietikäinen and T. Mäenpää. A generalized Local Binary Pattern operator for multiresolution gray scale and rotation invariant texture classification. In *International Conference on Advances in Pattern Recognition*, pages 397–406. Rio de Janeiro, Brazil, 2001.

[OPM02] T. Ojala, M. Pietikäinen and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with Local Binary Patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002.

[OPS+97] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna and T. Poggio. Pedestrian detection using wavelet templates. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, pages 193–199. San Juan, PR, USA, 1997.

[OT06] A. Oliva and A. Torralba. *Building the Gist of a Scene: The Role of Global Image Features in Recognition*, vol. 155 of *Progress in Brain Research*, pages 23–36. Elsevier, Oxford, UK, UK, 2006.

[Pal99] S. Palmer. *Vision Science*. MIT Press, Cambridge, MA, USA, 1999.

[Pla00] J. Platt. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In A. Smola, P. Bartlett, B. Schölkopf and D. Schuurmans, eds., *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, Cambridge, MA, USA, 2000.

[PW72] W. Peterson and E. Weldon. *Error Correcting Codes*. MIT Press, Cambridge, MA, USA, 1972.

[Qui93] J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[Ren00] R. Rensink. The Dynamic Representation of Scenes. *Visual Cognition*, 7(1–3):17–42, 2000.

[RH95] B. Ripley and N. Hjort. *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge, UK, 1995.

[RM04] L. Renninger and J. Malik. When is scene identification just texture recognition? *Vision Research*, 44(19):2301–2311, 2004.

[RMN+98] C. Rodriquez, J. Muguerza, M. Navarro et al. A two-stage classifier for broken and blurred digits in forms. In *Proceedings of the 14th International Conference on Pattern Recognition (ICPR)*, pages 1101–1105. Los Alamitos, USA, 1998.

[Ros58] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization

in the brain. *Psychological Review*, 65(6):386–408, 1958.

[Ros78] E. Rosch. *Principles of Categorization*, pages 27–48. John Wiley & Sons, 1978.

[RS02] M. Ruiz and P. Srinivasan. Hierarchical Text Categorization Using Neural Networks. *Information Retrieval*, 5(1):87–118, 2002.

[SCB00] F. Smeraldi, O. Carmona and J. Bigün. Saccadic search with Gabor features applied to eye detection and real-time head tracking. *Image and Vision Computing*, 18(4):323–329, 2000.

[Sch92] N. Schmitz. *Optimal Sequentially Planned Decision Procedures*. Springer-Verlag, New York, NY, USA, 1992.

[Sch02] R. Schapire. The boosting approach to machine learning: An overview. In *MSRI Workshop on Nonlinear Estimation and Classification*. Berkeley, CA, USA, 2002.

[SFBL97] R. Schapire, Y. Freund, P. Bartlett and W. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. In *Proceedings of the 14th International Conference on Machine Learning*, pages 322–330. Morgan Kaufmann, 1997.

[SHB99] M. Sonka, V. Hlavac and R. Boyle. *Image Processing, Analysis, and Machine Vision*. Brooks/Cole Publishing Company, Pacific Grove, CA, 1999.

[Shi99] B. Shi. A one-dimensional CMOS focal plane array for Gabor-type image filtering. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 46(2):323–326, 1999.

[SK04] H. Schneiderman and T. Kanade. Object Detection Using the Statistics of Parts. *International Journal of Computer Vision*, 56(3):151–177, 2004.

[SKB+99] S. Simon, H. Kestler, A. Baune, F. Schwenker and G. Palm. Object classification with simple visual attention and hierarchical neural network for subsymbolic-symbolic coupling. In *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 244–249. Monterey, CA, USA, 1999.

[SM00] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[SM05] J. Sochman and J. Matas. WaldBoost - Learning for Time Constrained Sequential Detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, pages 150–156. San Diego, CA, USA, 2005.

[SOM+96] T. Spencer, J. Olson, K. McHardy, P. Sharp and J. Forrester. An image processing strategy for the segmentation and quantification of microaneurysms in fluorescein angiograms of the ocularfundus. *Computers in Biomedical Research*, 29:284–302, 1996.

[SPSF92] T. Spencer, R. Phillips, P. Sharp and J. Forrester. Automated detection and quantification of microaneurysms in fluorescein angiograms. *Grafe's Archive of Clinical and Experimental Ophthalmology*, 230:36–41, 1992.

[SS99] R. Schapire and Y. Singer. Improved Boosting Algorithms Using Confidence-rated Predictions. *Machine Learning*, 37(3):297–336, 1999.

[SS00] R. Schapire and Y. Singer. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39(2):135–168, 2000.

[SS01] B. Schölkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[TFM96] S. Thorpe, D. Fize and C. Marlot. Speed of processing in the human visual system. *Nature*, 381(6582):520–522, 1996.

[The90] The DCCT Research Group. The Diabetes Control and Complications Trial (DCCT): Update. *Diabetes Care*, pages 427–433, 1990.

[THJA04] I. Tsochantaridis, T. Hofmann, T. Joachims and Y. Altun. Support Vector Learning for Interdependent and Structured Output Spaces. In *Proceedings of the Twenty-first International Conference on Machine Learning (ICML2004)*. Banff, Alberta, Canada, 2004.

[TO03] A. Torralba and A. Oliva. Statistics of Natural Image Categories. *Network: Computation in Neural Systems*, 14(3):391–412, 2003.

[TS01] A. Torralba and P. Sinha. Recognizing indoor scenes. Tech. Rep. AI Memo 2001-015, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, July 2001.

[Utg89] P. Utgoff. Perceptron Trees: A Case Study in Hybrid Concept Representations. *Connection Science*, 1:377–391, 1989.

[UVNS02] S. Ullman, M. Vidal-Naquet and E. Sali. Visual features of intermediate complexity and their use in classification. *Nature Neuroscience*, 5(7):682–687, 2002.

[Vap98] V. Vapnik. *Statistical Learning Theory.* John Wiley & Sons, West Sussex, UK, 1998.

[VJ01] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR01)*, pages 511–518. Hawaii, USA, 2001.

[VJ04] P. Viola and M. Jones. Robust Real-Time Face Detection. *International Journal of Computer Vision*, 57(2):137 – 154, 2004.

[VNU03] M. Vidal-Naquet and S. Ullman. Object Recognition with Informative Features and Linear Classifcation. In *Proceedings of the 9th International Conference on Computer Vision (ICCV)*, pages 281–288. Nice, France, 2003.

[Wal47] A. Wald. *Sequential Analysis.* Dover, NY, USA, 1947.

[WHD96] T. Weldon, W. Higgins and D. Dunn. Efficient Gabor filter design for texture segmentation. *Pattern Recognition*, 29(12):2005–2015, 1996.

[WK02a] T. Walter and J.-C. Klein. Automatic Detection of Microaneyrysms in Color fundus Images of the Human Retina by Means of the Bounding Box Closing. In *Third International Symposium on Medical Data Analysis (ISMDA)*, pages 210–220. Rome, Italy, 2002.

[WK02b] T. Walter and J.-C. Klein. A Computational Approach to Diagnosis of Diabetic Retinopathy. In *Proceedings of the 6th Conference on Systemics, Cybernetics and Informatics (SCI)*, pages 521–526. Orlando, Florida, July 2002.

[WKMZ00] T. Walter, J.-C. Klein, P. Massin and F. Zana. Automatic segmentation and registration of retinal fluorescein angiographies – Application to diabetic retinopathy. In *First International Workshop on Computer Assisted Fundus Image Analysis (CAFIA)*. Copenhagen, Denmark, 2000.

[WWP99] A. Weigend, E. Wiener and J. Pedersen. Exploiting hierarchy in text categorization. *Information Retrieval*, 1(3):193–216, 1999.

[ZLQH04] L. Zhang, S. Li, Z. Qu and X. Huang. Boosting Local Feature Based Classifiers for Face Recognition. In *Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'04)*, page 87. Washington, DC, USA, 2004.

TIETOJENKÄSITTELYTIETEEN LAITOS
PL 68 (Gustaf Hällströmin katu 2 b)
00014 Helsingin yliopisto

DEPARTMENT OF COMPUTER SCIENCE
P.O. Box 68 (Gustaf Hällströmin katu 2 b)
FIN-00014 University of Helsinki, FINLAND

JULKAISUSARJA **A**

SERIES OF PUBLICATIONS **A**

Reports may be ordered from: Kumpula Science Library, P.O. Box 64, FIN-00014 University of Helsinki, FINLAND.

A-1998-3 E. Sutinen: Approximate pattern matching with the q-gram family. 116 pp. (Ph.D. thesis).

A-1999-1 M. Klemettinen: A knowledge discovery methodology for telecommunication network alarm databases. 137 pp. (Ph.D. thesis).

A-1999-2 J. Puustjärvi: Transactional workflows. 104 pp. (Ph.D. thesis).

A-1999-3 G. Lindén & E. Ukkonen (eds.): Department of Computer Science: annual report 1998. 55 pp.

A-1999-4 J. Kärkkäinen: Repetition-based text indexes. 106 pp. (Ph.D. thesis).

A-2000-1 P. Moen: Attribute, event sequence, and event type similarity notions for data mining. 190+9 pp. (Ph.D. thesis).

A-2000-2 B. Heikkinen: Generalization of document structures and document assembly. 179 pp. (Ph.D. thesis).

A-2000-3 P. Kähkipuro: Performance modeling framework for CORBA based distributed systems. 151+15 pp. (Ph.D. thesis).

A-2000-4 K. Lemström: String matching techniques for music retrieval. 56+56 pp. (Ph.D.Thesis).

A-2000-5 T. Karvi: Partially defined Lotos specifications and their refinement relations. 157 pp. (Ph.D.Thesis).

A-2001-1 J. Rousu: Efficient range partitioning in classification learning. 68+74 pp. (Ph.D. thesis)

A-2001-2 M. Salmenkivi: Computational methods for intensity models. 145 pp. (Ph.D. thesis)

A-2001-3 K. Fredriksson: Rotation invariant template matching. 138 pp. (Ph.D. thesis)

A-2002-1 A.-P. Tuovinen: Object-oriented engineering of visual languages. 185 pp. (Ph.D. thesis)

A-2002-2 V. Ollikainen: Simulation techniques for disease gene localization in isolated populations. 149+5 pp. (Ph.D. thesis)

A-2002-3 J. Vilo: Discovery from biosequences. 149 pp. (Ph.D. thesis)

A-2003-1 J. Lindström: Optimistic concurrency control methods for real-time database systems. 111 pp. (Ph.D. thesis)

A-2003-2 H. Helin: Supporting nomadic agent-based applications in the FIPA agent architecture. 200+17 pp. (Ph.D. thesis)

A-2003-3 S. Campadello: Middleware infrastructure for distributed mobile applications. 164 pp. (Ph.D. thesis)

A-2003-4 J. Taina: Design and analysis of a distributed database architecture for IN/GSM data. 130 pp. (Ph.D. thesis)

A-2003-5 J. Kurhila: Considering individual differences in computer-supported special and elementary education. 135 pp. (Ph.D. thesis)

A-2003-6 V. Mäkinen: Parameterized approximate string matching and local-similarity-based point-pattern matching. 144 pp. (Ph.D. thesis)

A-2003-7 M. Luukkainen: A process algebraic reduction strategy for automata theoretic verification of untimed and timed concurrent systems. 141 pp. (Ph.D. thesis)

A-2003-8 J. Manner: Provision of quality of service in IP-based mobile access networks. 191 pp. (Ph.D. thesis)

A-2004-1 M. Koivisto: Sum-product algorithms for the analysis of genetic risks. 155 pp. (Ph.D. thesis)

A-2004-2 A. Gurtov: Efficient data transport in wireless overlay networks. [B 141 pp. (Ph.D. thesis)

A-2004-3 K. Vasko: Computational methods and models for paleoecology. 176 pp. (Ph.D. thesis)

A-2004-4 P. Sevon: Algorithms for Association-Based Gene Mapping. 101 pp. (Ph.D. thesis)

A-2004-5 J. Viljamaa: Applying Formal Concept Analysis to Extract Framework Reuse Interface Specifications from Source Code. 206 pp. (Ph.D. thesis)

A-2004-6 J. Ravantti: Computational Methods for Reconstructing Macromolecular Complexes from Cryo-Electron Microscopy Images. 100 pp. (Ph.D. thesis)

A-2004-7 M. Kääriäinen: Learning Small Trees and Graphs that Generalize. 45+49 pp. (Ph.D. thesis)

A-2004-8 T. Kivioja: Computational Tools for a Novel Transcriptional Profiling Method. 98 pp. (Ph.D. thesis)

A-2004-9 H. Tamm: On Minimality and Size Reduction of One-Tape and Multitape Finite Automata. 80 pp. (Ph.D. thesis)

A-2005-1 T. Mielikäinen: Summarization Techniques for Pattern Collections in Data Mining. 201 pp. (Ph.D. thesis)

A-2005-2 A. Doucet: Advanced Document Description, a Sequential Approach. 161 pp. (Ph.D. thesis)

A-2006-1 A. Viljamaa: Specifying Reuse Interfaces for Task-Oriented Framework Specialization. 285 pp. (Ph.D. thesis)

A-2006-2 S. Tarkoma: Efficient Content-based Routing, Mobility-aware Topologies, and Temporal Subspace Matching. 198 pp. (Ph.D. thesis)

A-2006-3 M. Lehtonen: Indexing Heterogeneous XML for Full-Text Search. 185+3 pp.(Ph.D. thesis).

A-2006-4 A. Rantanen: Algorithms for $^{13}C$ Metabolic Flux Analysis. 92+73 pp.(Ph.D. thesis).

A-2006-5 E. Terzi: Problems and Algorithms for Sequence Segmentations. 141 pp. (Ph.D. Thesis).

A-2007-1 P. Sarolahti: TCP Performance in Heterogeneous Wireless Networks.(Ph.D. Thesis).

A-2007-2 M. Raento: TCP Exploring privacy for ubiquitous computing: Tools, methods and experiments. (Ph.D. thesis).

A-2007-3 L. Aunimo: Methods for Answer Extraction in Textual Question Answering 127+18 pp. (Ph.D. Thesis).

A-2007-4 T. Roos: Statistical and Information-Theoretic Methods for Data Analysis. 82+75pp. (Ph.D. Thesis).

A-2007-5 S. Leggio: A Decentralized Session Management Framework for Heterogeneous Ad-Hoc and Fixed Networks. 230 pp. (Ph.D. Thesis).

A-2007-6 O. Riva: Middleware for Mobile Sensing Applications in Urban Environments. 195 pp. (Ph.D. thesis).

A-2007-7 K. Palin: Computational Methods for Locating and Analyzing Conserved Gene Regulatory DNA Elements. 130 pp. (Ph.D. Thesis).

A-2008-1 I. Autio: Modeling Efficient Classification as a Process of Confidence Assessment and Delegation. 212 pp. (Ph.D. Thesis).