DGK Deutsche Geodätische Kommission
der Bayerischen Akademie der Wissenschaften

Reihe C Dissertationen Heft Nr. 716

Ribana Roscher

Sequential Learning
Using Incremental Import Vector Machines
for Semantic Segmentation

München 2013

Verlag der Bayerischen Akademie der Wissenschaften
in Kommission beim Verlag C. H. Beck

ISSN 0065-5325 ISBN 978-3-7696-5128-7

Diese Arbeit ist gleichzeitig veröffentlicht in:
Schriftenreihe des Instituts für Geodäsie und Geoinformation
der Rheinischen Friedrich-Wilhelms Universität Bonn
ISSN 1864-1113, Nr. 40, Bonn 2013

DGK Deutsche Geodätische Kommission
der Bayerischen Akademie der Wissenschaften

Reihe C Dissertationen Heft Nr. 716

Sequential Learning
Using Incremental Import Vector Machines
for Semantic Segmentation

Inaugural-Dissertation zur
Erlangung des akademischen Grades
Doktor-Ingenieur (Dr.-Ing.)
der Hohen Landwirtschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms Universität
zu Bonn

vorgelegt

am 25.07.2012 von

Ribana Roscher

aus Erlabrunn

München 2013

Verlag der Bayerischen Akademie der Wissenschaften
in Kommission beim Verlag C. H. Beck

ISSN 0065-5325 ISBN 978-3-7696-5128-7

Diese Arbeit ist gleichzeitig veröffentlicht in:
Schriftenreihe des Instituts für Geodäsie und Geoinformation
der Rheinischen Friedrich-Wilhelms Universität Bonn
ISSN 1864-1113, Nr. 40, Bonn 2013

Adresse der Deutschen Geodätischen Kommission:

Deutsche Geodätische Kommission
Alfons-Goppel-Straße 11 ● D – 80 539 München
Telefon +49 – 89 – 23 031 1113 ● Telefax +49 – 89 – 23 031 - 1283 / - 1100
e-mail hornik@dgfi.badw.de ● http://www.dgk.badw.de

Diese Publikation ist als pdf-Dokument veröffentlicht im Internet unter der Adresse /
This volume is published in the internet

<http://dgk.badw.de> ● <http://hss.ulb.uni-bonn.de/2012/3009/3009.pdf>

Prüfungskommission

Referent:	Prof. Dr.-Ing. Dr. h.c. mult. Wolfgang Förstner
1. Korreferent:	Prof. Dr. Hans-Peter Helfrich

Tag der mündlichen Prüfung: 24.09..2012

ISSN 0065-5325 ISBN 978-3-7696-5128-7

Zusammenfassung

Sequentielles Lernen mit inkrementellen Import Vector Machines für die semantische Segmentierung

Wir entwickeln einen neuen maschinellen Lernalgorithmus für die Klassifikation namens inkrementelle Import Vector Machines. Der Klassifikator ist speziell für die Aufgabe des sequentiellen Lernes entworfen, bei dem die Daten nacheinander dem Klassifikator präsentiert werden.

Die Motivation für diese Arbeit entstand aus der Überzeugung, einen Klassifikator formulieren zu können, der die Herausforderungen des sequentiellen Lernens bewältigen kann und dabei leistungsstark bezüglich der Klassifikationsgenauigkeit, Effizienz und aussagekräftiger Ergebnisse ist. Eine Herausforderung des sequentiellen Lernens ist, dass die Daten dem Lerner zu einem bestimmten Zeitpunkt nicht vollständig zur Verfügung stehen, und in der Regel das Warten auf eine repräsentative Anzahl von Daten nicht erwünscht und unpraktisch ist. Um eine Klassifikation von gegebenen Daten zu jeder Zeit zu ermöglichen, muss somit die Lernphase des Klassifikatormodells umgehend begonnen werden, auch wenn nicht alle Trainingsbeispiele verfügbar sind. Eine weitere Herausforderung ist, dass die Anzahl von nacheinander ankommenden Daten sehr groß oder sogar unendlich sein kann und somit nicht alle Daten gespeichert werden können. Darüber hinaus kann die Verteilung der Daten über die Zeit variieren, und das Klassifikationsmodell muss stabil bleiben bezüglich nicht-relevanter Daten, aber auch flexibel bezüglich neuer, relevanter Daten.

Unser Hauptbeitrag besteht daher in der Entwicklung, Analyse und Evaluierung eines leistungsfähigen inkrementellen Lerners für das sequentielle Lernen, den wir inkrementelle Import Vector Machines (I^{2}VMs) nennen. Der Klassifikator basiert auf den nicht-inkrementellen Import Vector Machines, die von Zhu and Hastie (2005) entwickelt wurden. I^{2}VM ist ein kernel-basierter, diskriminativer Klassifikator und ist deswegen in der Lage mit komplexen Datenverteilungen umgehen zu können. Desweiteren ist der Klassifikator dünnbesetzt, was ein effizientes Training und Testen ermöglicht, und er ist probabilistisch. Ein zentraler Beitrag dieser Arbeit ist der Nachweis und die Analyse der diskriminativen und rekonstruktiven Modellkomponente von IVM und I^{2}VM. Während diskriminative Klassifikatoren versuchen die Klassen so gut wie möglich zu trennen, streben Klassifikatoren mit rekonstruktiver Komponente danach, möglichst viel Informationen über die Daten zu erhalten um die Verteilung approximieren zu können. Beide Eigenschaften sind notwendig um einen leistungsfähigen inkrementellen Klassifikator zu erhalten. Ein weiterer zentraler Beitrag ist die Formulierung der inkrementellen Lernstrategie für I^{2}VM. Die Strategie enthält das Hinzufügen und Entfernen von Datenpunkten und das Update der aktuellen Modellparameter. Desweiteren können sowohl neue Klassen als auch Merkmale hinzugefügt werden. Die Lernstrategie adaptiert kontinuierlich das Modell, hält es aber gleichzeitig stabil und effizient.

In unseren Experimenten untersuchen wir die Eignung von I^{2}VM für die semantische Segmentierung von Bildern aus einer Bilddatenbank, für die Klassifikation von Landbedeckungen großer Gebiete in überlappten Fernerkundungsdaten und für die Objektverfolgung in Bildfolgen. Wir zeigen, dass I^{2}VM höhere oder kompetitive Klassifikationsgenaugkeiten wie vergleichbare Klassifikatoren erzielen kann. Ein bedeutender Beitrag dieser Arbeit ist, dass die Güte von I^{2}VM unabhängig von der Reihenfolge der Daten ist und dass bereits bearbeitete Daten nicht nochmal für das Lernen in Betracht gezogen werden müssen. Ein weiterer Beitrag besteht darin, dass I^{2}VM in der Lage ist, ohne Verlust an Effizienz, mit langen Bildsequenzen umgehen zu können . Desweiteren liefert I^{2}VM verlässliche a posteriori Wahrscheinlichkeiten, sodass Datenpunkte mit hohen Wahrscheinlichkeiten akkurat klassifiziert werden und Datenpunkte mit niedriger Wahrscheinlichkeit eher falsch klassifiziert sein können.

Summary

Sequential learning using incremental import vector machines for semantic segmentation

We propose an innovative machine learning algorithm called incremental import vector machines that is used for classification purposes. The classifier is specifically designed for the task of sequential learning, in which the data samples are successively presented to the classifier.

The motivation for our work comes from the effort to formulate a classifier that can manage the major challenges of sequential learning problems, while being a powerful classifier in terms of classification accuracy, efficiency and meaningful output. One challenge of sequential learning is that data samples are not completely available to the learner at a given point of time and generally, waiting for a representative number of data is undesirable and impractical. Thus, in order to allow for a classification of given data samples at any time, the learning phase of the classifier model needs to start immediately, even if not all training samples are available. Another challenge is that the number of sequential arriving data samples can be very large or even infinite and thus, not all samples can be stored. Furthermore, the distribution of the sample can vary over time and the classifier model needs to remain stable and unchanged to irrelevant samples while being plastic to new, important samples.

Therefore our key contribution is to develop, analyze and evaluate a powerful incremental learner for sequential learning which we call incremental import vector machines (I^{2}VMs). The classifier is based on the batch machine learning algorithm import vector machines, which was developed by Zhu and Hastie (2005). I^{2}VM is a kernel-based, discriminative classifier and thus, is able to deal with complex data distributions. Additionally, the learner is sparse for an efficient training and testing and has a probabilistic output. A key achievement of this thesis is the verification and analysis of the discriminative and reconstructive model components of IVM and I^{2}VM. While discriminative classifiers try to separate the classes as well as possible, classifiers with a reconstructive component aspire to have a high information content in order to approximate the distribution of the data samples. Both properties are necessary for a powerful incremental classifier. A further key achievement is the formulation of the incremental learning strategy of I^{2}VM. The strategy deals with adding and removing data samples and the update of the current set of model parameters. Furthermore, also new classes and features can be incorporated. The learning strategy adapts the model continuously, while keeping it stable and efficient.

In our experiments we use I^{2}VM for the semantic segmentation of images from an image database, for large area land cover classification of overlapping remote sensing images and for object tracking in image sequences. We show that I^{2}VM results in superior or competitive classification accuracies to comparable classifiers. A substantial achievement of the thesis is that I^{2}VM's performance is independent of the ordering of the data samples and a reconsidering of already encountered samples for learning is not necessary. A further achievement is that I^{2}VM is able to deal with very long data streams without a loss in the efficiency. Furthermore, as another achievement, we show that I^{2}VM provide reliable posterior probabilities since samples with high class probabilities are accurately classified, whereas relatively low class probabilities are more likely referred to misclassified samples.

Contents

1 Introduction **9**

1.1 Overview . . . 9

1.2 Motivation . . . 10

1.3 Goal and Achievements of the Thesis . . . 10

1.4 Applications of the Proposed Classifier . . . 12

1.5 Organization of the Thesis . . . 14

2 Related Work **17**

2.1 Sequential Learning . . . 17

2.2 Incremental Learning Methods . . . 19

2.3 Semantic Segmentation . . . 22

2.3.1 Semantic Segmentation in Single Images . . . 23

2.3.2 Semantic Segmentation in Image Databases . . . 23

2.3.3 Semantic Segmentation in Image Sequences . . . 24

3 Theoretical Background **27**

3.1 Notation . . . 27

3.2 Supervised Sequential Learning Problem . . . 27

3.2.1 Learning Task . . . 28

3.2.2 Sequential Learning Task . . . 28

3.2.3 Learning with Sequential Data . . . 29

3.2.4 Reconstructive and Discriminative Model Components of Classifiers . 30

3.2.5 Discriminative Models for Classification . . . 33

3.2.5.1 Generalized Linear Models . . . 33

3.2.5.2 Classes of Kernels for Machine Learning . . . 34

3.2.5.3 Loss-functions . . . 34

3.3 Import Vector Machines . . . 37

3.3.1 Basis Model: Logistic Regression . . . 37

3.3.1.1 Parameter Estimation with Iteratively Reweighted Least Squares 39

3.3.1.2 Kernel Logistic Regression . . . 40

3.3.1.3 Sparse Kernel Logistic Regression . . . 41

3.3.2 Basic Import Vector Machines Algorithm . . . 42

3.3.3 Revised Import Vector Machines Algorithm . . . 43

3.4 Semantic Image Segmentation . . . 45

3.4.1 Definition of Semantic Image Segmentation . . . 45

3.4.2 Semantic Image Segmentation Problem . . . 45

3.4.3 Sequential Semantic Image Segmentation Problem . . . 48

4 Discriminative and Reconstructive Properties of IVM 49
4.1 Statement of the Problem . . . 49
4.2 Hypothesis . . . 49
4.3 Empirical Study on the Distribution of Import Vectors . . . 54
4.3.1 Data . . . 55
4.3.2 Experimental Setup . . . 55
4.3.3 Results and Discussion . . . 56
4.3.3.1 On the Usage of Various Kernels . . . 56
4.3.3.2 Analysis of the Distribution of Import Vectors . . . 56
4.3.3.3 Comparison to SVM Regarding the Distribution of Support Vectors . . . 59
4.4 Summary . . . 60

5 Incremental Learning with Import Vector Machines 63
5.1 Conceptual Procedure for an Incremental Update Step . . . 63
5.2 Representing the Current State of the Learner . . . 63
5.3 Learning Procedure for Incremental Import Vector Machines . . . 64
5.4 Incremental and Decremental Update . . . 65
5.4.1 Adding and Removing Training Vectors . . . 65
5.4.1.1 Adding Training Vectors . . . 65
5.4.1.2 Removing Training Vectors . . . 66
5.4.2 Adding and Removing Import Vectors . . . 67
5.4.2.1 Adding Import Vectors . . . 67
5.4.2.2 Removing Import Vectors . . . 68
5.5 Incorporation of New Classes and Features . . . 68
5.5.1 Incorporating New Classes . . . 68
5.5.2 Incorporating New Features . . . 69
5.6 Criteria for Removing Training Vectors . . . 69
5.6.1 Overview . . . 69
5.6.2 Cross-entropy . . . 70
5.6.3 Linear Independence . . . 70
5.6.4 Cook's Distance . . . 71
5.6.5 Weighted Sampling . . . 72
5.6.6 Increase of the Negative Log-likelihood Function . . . 73

6 Applications 75
6.1 Classification of Benchmark Data Sets . . . 76
6.1.1 Comparison of I^{2}VM to IVM and Recent Incremental Classifiers . . . 76
6.1.2 Data . . . 76
6.1.3 Methods . . . 77
6.1.4 Experimental Setup . . . 77
6.1.5 Results and Discussion . . . 77
6.1.5.1 Static Data . . . 77
6.1.5.2 Data with Concept-Drift . . . 78
6.1.6 Summary . . . 78
6.2 Semantic segmentation of Image Databases . . . 79
6.2.1 Comparison of Criteria for the Removal of Training Samples . . . 79
6.2.2 Data . . . 79
6.2.3 Experimental Setup . . . 80

6.2.4 Results and Discussion . . . 82

6.2.4.1 Synthetic Data Set . . . 82

6.2.4.2 MSRC Object Recognition Image Database . . . 84

6.2.5 Summary . . . 85

6.3 Large Area Land Cover Classification . . . 86

6.3.1 Evaluation of I^{2}VM for Large Area Land Cover Classification with Self-training . . . 86

6.3.2 Recent Approaches for Large Land Cover Classification . . . 86

6.3.3 Self-Training for Sequential Mosaic Classification . . . 87

6.3.4 Data . . . 89

6.3.5 Experimental Setup . . . 90

6.3.6 Results and Discussion . . . 91

6.3.7 Summary . . . 92

6.4 Tracking-by-Segmentation . . . 93

6.4.1 Evaluation of the Applicability of I^{2}VM for Data Streams . . . 93

6.4.2 Data Sets . . . 93

6.4.3 Tracking Framework . . . 95

6.4.3.1 Tracking Scheme . . . 95

6.4.3.2 Superpixel Tracking . . . 96

6.4.3.3 Features for Object Representation . . . 97

6.4.3.4 Segmentation . . . 97

6.4.4 Experimental Setup . . . 98

6.4.5 Results and Discussion . . . 98

6.4.5.1 FLOWER Image Sequence . . . 99

6.4.5.2 SegTrack Database . . . 99

6.4.5.3 Adafrag Database . . . 100

6.4.5.4 Long Image Sequences . . . 102

6.4.6 Summary . . . 103

7 Conclusion and Outlook 105

A Derivatives of the Negative Log-likelihood Function 107

A.1 Derivatives of the Posterior Probabilities P_{nc} w.r.t the Linear Function $g_{nc} = \boldsymbol{\alpha}_c^{\mathsf{T}}\mathbf{x}_n$ . . . 107

A.2 Derivative of the Negative Log-likelihood Function w.r.t. the Parameters . . . 108

B Relation Between Newton-Raphson and IRLS 111

C Matrix Inversion with Sherman-Morrisson Woodbury formula 113

Chapter 1

Introduction

1.1 Overview

This thesis formulates and evaluates an innovative *machine learning* algorithm that is used for classification purposes. Machine learning is a field of computer science, concerned with the development of computer programs that are able to learn from data samples collected through sensors. A more precise definition was formulated by Mitchell (1997) to specify the term "machine learning": "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E." In this thesis we address a program that learns from samples with given class-membership in order to solve a semantic image segmentation task, i.e. a partitioning of an image in semantically meaningful regions. In the first stage, we learn from experience decision boundaries between samples that are assigned to pre-defined classes. In the second stage, we use the learned decision boundaries in order to classify each sample from which we want to know the class-membership.

Particularly, we focus on the development of a machine learning algorithm which is specifically designed for the task of sequential learning. *Sequential learning* treats the data successively, either because the task was designed in this way or the data does not allow any other access. Instead of all data samples being available at once to the learning algorithm, the samples arrive over time (i.e. in streams), one by one or in batches. A *batch* is defined as group of samples which arrive simultaneously. The most common data stream is a video sequence, in which each image comprises a batch of samples. Sequential learning is also applied if the data is completely available (in one batch), but has to be processed sequentially. This occurs when the data set is too large to fit into the memory or to be processed in a practically tractable way. Although sequential data has a time reference, this is not necessarily important for the learning. The reference to time is due to the fact that a sample is made available to the learner at a certain point of time. This is not necessarily the acquisition time. In the case of real-time applications with continuous data streams the time of acquisition and usage can coincide. Thus, we define sequentially processed data also as sequential data.

We define learning with sequential data as *incremental learning*, or more precisely: We define sequential learning as the task to incrementally train a classifier with samples becoming successively available, one at time or in blocks. A learner is incremental if the classifier model is constantly updated with respect to the samples arriving sequentially. The model is built upon a limited number of samples and its former model. Throughout this thesis we use sequential learning and incremental learning interchangeably.

1.2 Motivation

Machine learning methods have to manage three major challenges when dealing with sequential data:

(a) The samples are not completely available to the learner at a given point of time and generally, waiting for a representative number of data is undesirable and impractical. Thus, in order to allow for a classification of given data samples at any time, the learning phase of the classifier model needs to start immediately.

(b) The number of sequentially arriving data samples can be very large or even infinite and thus not all samples can be stored.

(c) In general sequentially arriving data does not follow the same static underlying distribution. The variation in the distribution is called concept-drift. Different types of drifts are discussed in Hoens et al. (2012) and Jaber et al. (2011).

Batch learning algorithms, i.e. algorithms trained on one batch of data, are not able to deal with sequential data. According to (a) and (b) waiting for a representative number of data is undesirable and impractical for large data sets and even impossible for infinite data sets. Since the number of sequential data increases over time, a storage of all samples for a simultaneous processing is intractable. According to (c) batch learning algorithms ignore all new samples and therefore cannot be adapted to important changes in the data distribution. Once they have been learned, they are unable to describe newly arrived data samples.

In contrary to these algorithms an incremental learner can start to learn without any delay and can be updated constantly regarding the sequentially arriving samples. Therefore, these methods can adapt immediately to new samples. Furthermore, the learner does not need to store all samples and can remove a part or all samples with respect to a suitable criterion. Nevertheless, according to (c) an incremental learner is confronted with the so-called stability-plasticity dilemma (Grossberg, 1988) which deals with the question how the learner remains stable and unchanged to irrelevant samples while being plastic (i.e. flexible) to new, important samples.

Due to the positive properties of incremental learners, several classifiers have been proposed, extending classical and state-of-the-art batch classifiers. One classifier that has not been extended yet are the *import vector machines* (IVMs) developed by Zhu and Hastie (2005). They turn out to have properties which are useful for a powerful incremental learning and thus, IVM appears to be a suitable starting point for an incremental classifier. This thesis focuses on the development of an incremental version of IVM and shows that the classifier can manage the mentioned challenges when dealing with sequential data.

1.3 Goal and Achievements of the Thesis

The goal of this thesis is to develop and evaluate a powerful incremental learner for sequential learning which we call *incremental import vector machines* (I^{2}VMs). The learner is based on the batch machine learning algorithm IVM, which was developed by Zhu and Hastie (2005). A key achievement is the analysis of the yet unexplored properties of the learning scheme of IVM which turns out to be useful for incremental learning. We show that under certain conditions the IVM algorithm, though being a *discriminative* classifier, also has a *reconstructive* model component. While discriminative classifiers try to separate the classes as well as possible, classifiers with a reconstructive component aspire to have a high information content in order to approximate the distribution of the data samples. Both properties are necessary for a powerful incremental classifier. We analyze the IVM classification model in detail and show that IVM has both a high discriminative power as well as reconstructive abilities. Contrary to discriminative classifiers, IVM implicitly samples important parts of the underlying distribution. These representative samples can be used as an approximation to the distribution.

A further key achievement is the formulation of the incremental learning strategy of I^{2}VM. The strategy deals with adding and removing data samples and the update of the current set of model parameters. Furthermore, I^{2}VM is able to incorporate new classes and features, which has been paid only little attention in the literature so far. The I^{2}VM algorithm has the following properties:

(a) **Competitive performance.** It performs comparably to its batch learning counterpart. When applied to the same data samples, it is independent from the ordering of the data. Thus, the learner is able to handle concept-drifts in data distribution without suffering a loss in performance.

(b) **Discriminative power.** It separates the classes well, regardless of the complexity of data distribution.

(c) **Long sequences.** It is able to deal with arbitrarily long data streams.

(d) **Probabilities.** It provides reliable posterior probabilities in order to allow for a meaningful evaluation, to serve as input for further processing steps, e.g. graphical models, or to provide criteria to decide on irrelevant data to keep the model sparse.

To meet requirements (a) and (b), incremental methods in general need a reconstructive model component and should have a discriminative model component. A reconstructive model component represents the significant sub-domain of the data distribution. It offers robustness against the sequence of data samples as well as being effective in the case of many competing classes. Additionally, it is also capable of adapting to actual or apparent changes in the data distribution which appear as concept-drifts to the learner caused by either changes within the class (intra-class variability) or by changes between the classes (inter-class variability). A discriminative model component, though not necessary, is recommendable in order to be efficient for distinguishing similar classes. Moreover, kernel-based learning methods have shown good results when dealing with complex data distributions.

In order to meet requirement (c), I^{2}VM is able to add and remove data samples while at the same time adapting the classifier model to the current conditions. In contrary to kernel-based batch methods, incremental kernel-based approaches such as I^{2}VM demand strategies for adding and removing data samples. However, the criteria for the addition and removal of samples may differ depending on the application. When dealing with long data streams, a long-term memory may be necessary; this requires a memory-efficient classifier model. Among the diverse classifiers, sparse kernel-based batch learning methods have been shown considerable success for achieving good performance with highly complex data distribution while at the same time being sparse and therefore efficient.

In order to meet requirement (d), I^{2}VM is be based on the probabilistic IVM classifier model. While probabilistic models tend to rely on more computational resources than non-probabilistic models, probabilities are often of interest and give the opportunity for further processing steps.

In our experiments we prove that I^{2}VM can meet all requirements. We show that I^{2}VM results in competitive classification accuracies to comparable classifiers. A key achievement of the thesis is that I^{2}VM's performance is independent of the ordering of the data samples and a reconsideration of already encountered samples for learning is not necessary. A further achievement is that I^{2}VM is able to deal with very long data streams without a loss in the efficiency. Furthermore, as another achievement, we show that I^{2}VM provide reliable posterior probabilities since samples with high class probabilities are accurately classified, whereas relatively low class probabilities are more likely assigned to misclassified samples.

1.4 Applications of the Proposed Classifier

The applications of incremental learners are wide-ranging. They extend the scope of batch learners to areas in which a sequential data treatment is necessary. We subdivide these applications into the following groups:

(a) The generation of samples or blocks of samples is time-dependent.

(b) New samples arrive one by one or in blocks in arbitrary intervals in which the time-dependency is not of interest.

(c) The data set is completely available but processed sequentially, because it is too large to either fit into the memory or to be processed simultaneously in a practically tractable way.

Case (a) comprises all applications that deal with streams, i.e. samples which arrive time-dependently. The most common streams are time series data and video sequences, which we will discuss in Section 2.3.3. As an example, Figure 1.1 shows representative images of a video sequence, in which a rotating flower is meant to be tracked. The green contour indicates the detected boundary of the tracked object. The image sequence includes for each image to varying degrees, the challenge of changing color appearance, interframe motion and change in object shape. A sequential learning algorithm can learn the appearance of both the object and the background and is able to adapt to the changes automatically (Roscher et al., 2012; Santner et al., 2010; Tang et al., 2007; Avidan, 2007). Additonally, the time-dependency of the data stream can be exploited by introducing temporal relations between pixels or regions of successive frames and enforcing a temporal consistency of their predicted labels.

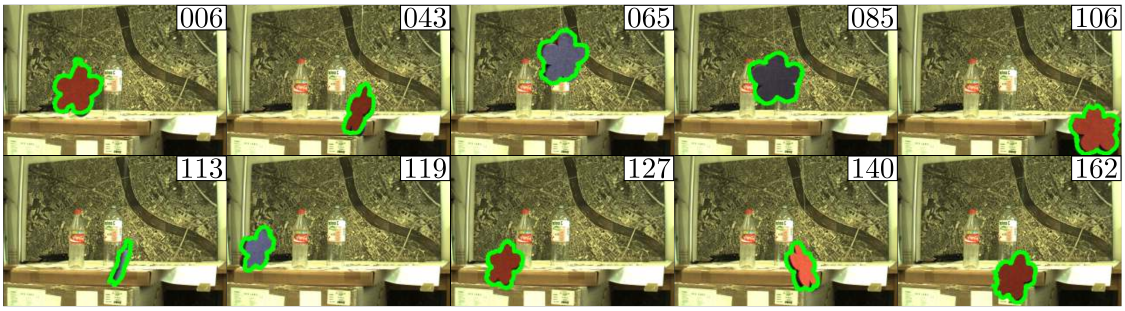


FIGURE 1.1: Representative frames (numbers t in the top right corners) of the tracking results for the image sequence.

Remote Sensing, as another example for case (a), deals with data becoming available in blocks or intervals. Earth sensing satellites acquire a large amount of data on worldwide environmental changes over time such as e.g. tropical rain forest deforestation, climate information such as meteorological, oceanographic or atmospheric data or the amount of snow and ice in the polar regions. The intervals can range from a few hours, depending on the temporal frequency of the satellite, to monitor short term changes, over few months to monitor glacier surface velocities (Raup et al., 2007). The interval can also be several years to monitor changes in vegetation (e.g. Stow et al. (2004); Turner (1990)). Of course, if the interval is large enough, the task can also be solved by a non-incremental learner. Nevertheless, due to the large amount of accumulated data, an incremental learner will be necessary because not all data can be processed simultaneously.

Typical applications for (b) are those that use the so-called active learning. It is an umbrella term for methods that are suitable for a sequential acquisition of samples using an oracle that defines the label of the samples. Since the labeling of data is time-consuming, expensive or potentially difficult, active learning enables a continuous training of the classifier. For a comprehensive literature survey, we refer to Tuia et al. (2009) and Settles (2009). *Self-training* can be considered as the easiest type of active learning. Starting with a limited number of labeled samples, the classification result is used to acquire new labeled samples in order to improve the classification result (Roscher et al., 2012; Li and Fei-Fei, 2010). Also co-training (Blum and Mitchell, 1998) is used to update a classifier model by using different, ideally complementary views of the data. These methods are also common in recent tracking approaches (see Sec. 2.3.3). Another example is a land cover classification of large areas, which consists of several composite remote sensing images. Generally, the images are not or only partly labeled and they are characterized by both spatial and temporal differences, see Figure 1.2. Thus, the labeled samples cannot represent the distribution of the features in all images. In order to obtain a classification of the whole area, active learning is applied for the acquisition of labeled samples in all images (Roscher et al., 2012; Knorn et al., 2009). The example given in Figure 1.2 shows an area of about 285000 km^2 around Rondonia in South America. The data set contains 9 Landsat 5 TM images from 2009, in which only the center image comes with labeled samples. Applications like large area land cover classification benefit from sequential learning strategies, because the classifier can be updated each time new acquired data is available and need not to be retrained from scratch.

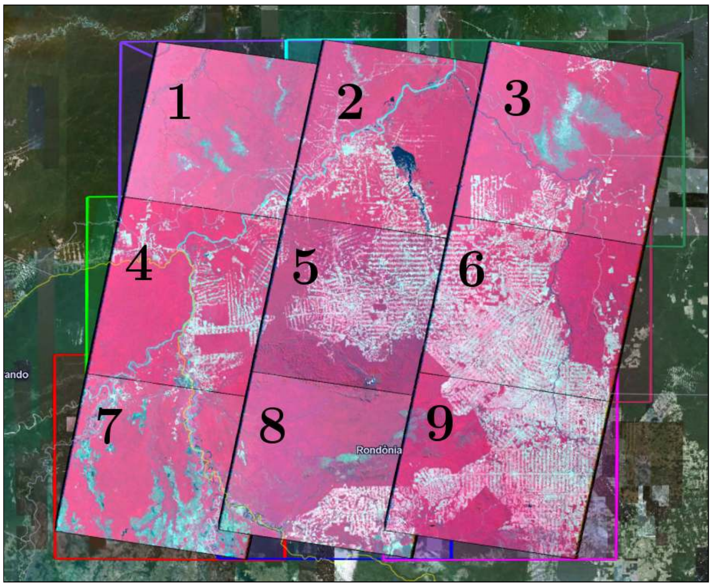


FIGURE 1.2: The area of Rondonia with 9 overlayed Landsat images displayed with bands 4-3-2. The three image strips are acquired in September (images 1, 4 and 7), July (images 2, 5 and 8), and August 2009 (images 3, 6 and 9). The underlaid image is taken from Google Earth. The classification aims at 4 land cover classes, focused on FOREST, AGRICULTURE, WATER and URBAN.

According to case (c), besides infinite data streams, also finite data sets can be too large to either fit into the memory or to be processed in a practically tractable way. Csató et al. (2001) predict wind speed and direction over the ocean surface over a huge area of several km^2 which is important for planning and construction activities in the oceanic area. They present an approach to sequentially update an approximation to the posterior distribution of the wind vectors which cannot be computed at once. In order to cluster huge amount of data samples, such as astronomic data or document collections of web sites, Littau and Boley (2009) sequentially scan each data sample in these data sets in order to find an approximation to the whole data set. The subset can be clustered more efficiently than using the original data set.

Also applications like the classification of large image databases can utilize sequential learning algorithms in order to learn the appearance of many classes in a practically tractable way by processing the images sequentially rather than simultaneously. One example for such an application is illustrated in Figure 1.3, in which images of the Microsoft Research Cambridge object recognition image database (Winn et al., 2005) are meant to be classified. The appearance of the classes may change with each newly processed image and therefore the classifier needs to be adapted during the learning process.

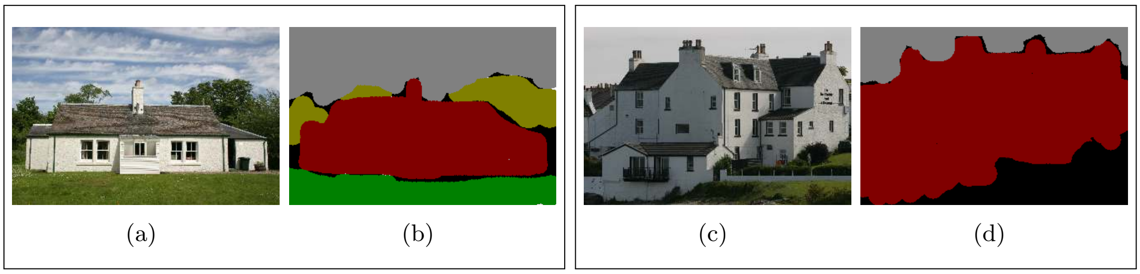


FIGURE 1.3: (a)(c) Images of Microsoft Research Cambridge object recognition image database. (b)(d) Manual classification aiming on the classes BUILDING (red), GRASS (green), TREE (light green), SKY (gray). Black pixels have the class VOID.

The application of incremental learners are numerous. In this thesis we focus on the field of semantic image segmentation in which each pixel is assigned to a pre-defined class. By grouping adjacent pixels with the same class, we obtain segments with a semantic meaning. In our experiments we exploit I^{2}VM for the semantic segmentation of images from an image database, for large area land cover classification of overlapping remote sensing images and for object tracking in image sequences.

1.5 Organization of the Thesis

The organization of the thesis is as follows. Chapter 2 gives an overview about sequential learning strategies. We define the term “sequential learning” in order to define the considered task in this thesis. We further review currently used algorithms for incremental learning and their application using single images, image databases and image sequences. The review of existing algorithms focuses on the properties mentioned in Section 1.3. Chapter 3 introduces the theoretical background of the IVM classifier. Additionally, we discuss the relation of IVM to similar classifiers. In Chapter 4 we show that IVM though being a discriminative model inherently have a reconstructive component. We verify this by means of the analysis of the objective function. Further, we underline our findings by an empirical study on the distribution of the import vectors and discuss the requirements which must be met in order to develop the reconstructive component. The core of the thesis is the formulation of the

I^{2}VM learner, an incremental realization of IVM. Chapter 5 develops the incremental learning scheme and describes the addition and removal of data samples and the incremental update of the classifier model. Furthermore, the incorporation of new classes and features is shown. Chapter 6 presents experimental results showing the performance of I^{2}VM. Our conducted experiments comprise the classification of well-known benchmark data sets with and without ordering effects, the semantic segmentation of large image databases, large area land cover classification and an object tracking application. We conclude in Chapter 7 and discuss possible further developments of IVM and I^{2}VM.

Chapter 2

Related Work

The overall goal in this thesis is to formulate, analyze and evaluate the learner I^{2}VM for sequential learning with focus on semantic segmentation. This chapter provides on overview of the most recent and significant work in the fields of the sequential learning problems, sequential learning algorithms and the application of sequential semantic segmentation. The first section reviews different views on the sequential learning task in order to distinguish this work from other conceptions. The second section reviews recently used algorithms which are developed to solve sequential learning problems. In the last section we introduce some applications in which the samples are treated successively with focus on semantic segmentation.

2.1 Sequential Learning

In the field of machine learning, the term "sequential learning" is treated in different ways. As stated in Chapter 1 we use sequential learning and incremental learning interchangeably throughout this thesis, in which we define sequential learning as the task to incrementally train a classifier with samples becoming successively available, one at time or in blocks. The main distinction in the literature is what the word "sequential" is referred to; either to the arrangement of the data or the learning task itself. One the hand the learning task can be defined as the identification of a particular sequence of labels, in which the sequence has a meaning (Dietterich, 2002). Such sequences are for example words as concatenation of letters (Krallinger et al., 2008; Brefeld et al., 2005)) or DNA sequences (Leslie et al., 2002). These samples are not drawn independently and identically from an joint distribution over the class memberships and the samples themselves. On the other hand the term "sequential" does not refer to the arrangement of the data but rather to the learning task, in which the learner has a sequential access to the data. We define sequential learning as the task to incrementally train a classifier with samples becoming available over time, one by one or in blocks. It is important to note that although the data become available over time, this does not necessarily mean that the time at which the sample is made available to the learner is of concern. We will restrict ourselves to sequentially treated data independently of their nature.

Our point of view of sequential learning is also known as incremental learning. The learning process deals with sequentially arriving data samples. The data can be of any nature, which can make dealing with them challenging. The overall data stream can be massive (Witten et al., 2011; Li and Fei-Fei, 2010; Hulten and Domingos, 2002), i.e. of high density with many samples arriving in short intervals over a long period, or even infinite (Monteleoni et al., 2011; Monteleoni and Kaariainen, 2007). To solve this learning task, incremental learning algorithms are exploited that are specially designed for this purpose. Although this type of understanding is not new, Jain et al. (2006) and Giraud-Carrier (2000) have helped to analyze and define the term "incremental learning" to make it better understood.

In the literature there exist several approaches considering different aspects of incremental learning. They can be subdivided into three groups, namely

(a) the way how old samples are used,

(b) the kind of information that can be incorporated,

(c) the choice of the update interval.

The way how old samples are used. Reinke and Michalski (1988) introduced the term full-memory learning. During the learning process the classifier model is updated by using the newly arrived samples. However, the learner does also make use of all saved previous training samples. In many applications, this would be impossible since the storage of the data is not feasible.

A good compromise is to store only a bounded number of chosen examples, which are selected by age (as in tracking scenarios, see Sec. 2.3.3) or by subset selection procedures. Lange and Zilles (2003) show that learning with a limited number of carefully chosen samples can be enough to ensure a good performance of incremental learners. Engelbrecht and Cloete (1999) identify and retain the most informative samples, i.e. those samples that have the highest influence on the learning objective. For example, the learning objective can be the reduction of the misclassification error. The approach has been extended by Engelbrecht and Brits (2001) by using only the most informative sample, which is identified by clustering a group of newly incoming data samples. Maloof and Michalski (2000) introduced the term "partial memory learning" in order to describe the usage of a bounded number of training samples. They restrict the number of newly acquired samples by choosing only the extreme ones, which are positioned near to the decision boundary. In a similar way, Zhang (1994) use only critical examples to train the incremental learner. The criterion for selecting critical samples depends on the information gain yielded by adding an sample. The information gain is defined by the reduction in the objective function after adding the sample. The sample, which leads to the largest reduction is chosen to be added.

Another possibility to retain knowledge seen so far is to built up a long-term memory, which can be seen as prior knowledge. Ferrari and Jensenius (2008) use an optimization problem to learn with newly arrived samples, subject to the prior knowledge formulated as constraints. Holmes et al. (2004) take a different approach by remembering models learned from groups of data rather than the data samples themselves. They divided the data set into single batches and learn each one non-incremental. After all batches have been proposed they combine the models learned in each batch to one global model.

Most work, however, strictly define incremental learning as an iterative learning of the classifier, which has only access to the newly arrived data and the last model. In this case, the classifier has no sample-memory. We will review such approaches regarding the used algorithm in Section 2.2.

The kind of information that can be incorporated. Zhou and Chen (2002) introduced the terms "example-incremental learning", "class-incremental learning" and "attribute incremental learning". Example-incremental learning updates a classifier model using only a few or none of previous seen samples. The new acquired knowledge of the samples is incorporated into the model. When using class-incremental learning, the classifier model is updated regarding the used classes. A new class is added without sacrificing the learned model too much.

In attribute-incremental learning the model is updated by adding new attributes (i.e. features) without sacrificing the previous learned model. For the last two methods there is only little work in literature, since most algorithms have been proposed dealing with example-incremental learning (see Section 2.2). Wenzel and Hotz (2010) subsequently add classes, one class after the other to analyze the role of sequences for incremental learning. Also Polikar et al. (2001) add classes that may be introduced with newly acquired data samples. Guan and Li (2001) sequentially add relevant features and thus, the feature dimension of the samples increases. They retain the old classifier model and combine it with a newly trained model with increased feature dimension to form an overall updated model. Skocaj et al. (2006) proposed an incremental learning method that can both incorporate new features and classes.

The choice of the update interval. Syed et al. (1999) introduced another distinction of incremental learning by introducing the so-called "instance learning" and "block-by-block learning". Instance-learning updates the model after each newly arrived sample, whereas block-by-block learning updates the model after grouping newly arrived samples into blocks of a suitable size. Also Hoens et al. (2012) define incremental learning as learning with data becoming available over time in streams of samples or batches, in which one batch is a block of samples. Langley (1995) distinguish between three different temporal resolutions in which the samples are processed. At the finest temporal resolution, the learner has access to features of an sample, one at a time. At the intermediate resolution, the learner is updated regarding the samples, one at a time and at the highest temporal resolution, the learner has access to distinct blocks of samples, one at a time. Most of the proposed algorithms (see Section 2.2) incrementally train the classifier using the samples one by one even if the samples arrive in blocks. This can be time-consuming because the incremental update steps need to be repeatedly applied to each single sample. Karasuyama and Takeuchi (2010), for example, extended the approach of Cauwenberghs and Poggio (2001) by using an optimization technique called parametric programming to perform the update with blocks of samples in a single step rather than a series of optimization problems.

We have reviewed different views on sequential learning, which are concerned mostly in the literature. In the next section, we discuss several recently used incremental classifiers, which are designed particularly to solve sequential learning problems.

2.2 Incremental Learning Methods

Several incremental learning methods have been suggested, extending classical and state-of-the-art batch methods. In the literature batch learning algorithms are also refereed to as offline classifiers and incremental classifiers are often denoted as online algorithms. We use the terms "incremental" or "sequential" and "batch" throughout the thesis. In this section, we review common incremental learners with respect to the properties of I^{2}VM mentioned in Section 1.3. I^{2}VM is a probabilistic multi-class learner with a high discriminative power, showing a comparable performance to IVM . We will later show, that IVM inherently has a reconstructive component to represent important parts of the underlying distribution.

Competitive performance. Incremental learning methods in general show comparable performances to their batch version when there is no change in the data distribution over time. The most recent methods showing these property vary from online discriminative kernel density estimation (Kristan and Leonardis, 2010), on-line random forests (ORF) (Saffari et al., 2009), incremental support vector machines (Zheng et al., 2010; Karasuyama and Takeuchi, 2010; Bordes et al., 2007; Fung and Mangasarian, 2002; Cauwenberghs and Poggio, 2001), online boosting algorithms (Grabner and Bischof, 2006) to online nearest neighbor classifier (Gu et al., 2010). These methods are able to handle complex distributions, in contrast to incremental linear subspace learning with linear discriminant analysis and principal component analysis (Uray et al., 2007; Kim et al., 2007). To deal with more complex data, some incremental subspace methods have been extended to a kernel version (Chin and Suter, 2007).

If the distribution of the samples changes over time, we speak of *concept-drifts*. They may cause severe problems for incremental learners. Like in tracking applications, old examples are often misleading, which is why newly gathered training vectors should be more beneficial than older ones. The incremental learner should be able to adapt to changes in the underlying distribution of data samples. For example, Gu et al. (2010), Tang et al. (2007) and Grabner et al. (2008) show that this necessary for a powerful object tracking. Explicitly identifying and handling concept-drift has been considered by Scholz and Klinkenberg (2007) and Klinkenberg and Joachims (2000), which show that incremental learner as online boosting and incremental support vectors machines can deal well with detected drifts.

A weaker but nevertheless relevant drifting-effect may result from changing the sequence of data samples, which results in so-called pseudo-concept-drifts, as shown by Wenzel and Hotz (2010), Langley (1995) and Cornuéjols (1993). Such drifts can occur for example in medical experiments if the data is collected for one patient at a time or in controlled experiments with a test plan for which process parameters are tested successively, as already stated by Rüping (2001). Several approaches have been suggested to overcome this problem. Talaveral and Roure (1998) introduced a buffer to store difficult samples for a later evaluation or McKusick and Langley (1991) rearranged the sequence . Bengio et al. (2009) introduced a schedule to present the samples in a special order to a neural network classifier to increase the performance. These methods are not useful for an incremental learner since in practice one usually cannot wait until a sufficient part or even all data samples are ready for processing.

To deal with concept-drifts Uray et al. (2007) and Skocaj et al. (2006) stated that incremental methods generally need a reconstructive model component. This is an inherent property of generative approaches. They model the joint probability of the labels and the features, and in a Bayesian way use priors to derive posterior probabilities. Incremental generative classifier, such as online discriminative kernel density estimation (Kristan and Leonardis, 2010), online nearest neighbors (Gu et al., 2010), an incremental Bayesian approach (Fei-Fei et al., 2007) or incremental linear discriminant analysis (Uray et al., 2007; Pang et al., 2005), can therefore represent current and even previously removed training samples. The approaches are very efficient, because the models can be defined by a few parameters describing the underlying distribution. This however requires the structure of the underlying distribution to be known and to be approximable with sufficient accuracy. This is not the case for many real-world data sets that have a complicated distribution without known underlying structure. To overcome this problem, non-parametric density estimation techniques such as k-nearest neighbor or kernel density estimation are used (Kristan and Leonardis, 2010; Gu et al., 2010). Although they have several advantages, these models can become very complex when the data distribution is approximated and the models have to be reduced for incremental learning settings. Generative models are able to directly incorporate unlabeled samples (Lasserre et al., 2006) and the model can be used as approximation of discarded training samples, which is needed for robust incremental learning (Skocaj et al., 2006; Uray et al., 2007).

Discriminative power. In contrary to generative classifiers, discriminative classifiers such as logistic regression directly model the posterior probability but do not contain a reconstructive component. Also support vector machines (SVMs) (Vapnik, 2000) that directly map the inputs into decisions by determining a discriminant function do not contain this component. Likewise decision trees (Breiman et al., 1984) are also based on the estimation of discriminant functions. During the induction of a decision tree the training data is sequentially partitioned into smaller increasingly homogeneous subsets by using a set of decision boundaries defined at each node. Usually only the most relevant feature is used at each node, resulting in many rather simple decision boundaries that are parallel to the axis in the feature space. The accuracy of decision tree classifiers is often improved by using ensemble strategies. Classifiers ensembles are based on the combination of independent variants of the same classifier, i.e. the base classifier. Various strategies for generating variants of the same classifier have been introduced, like boosting (Viola and Jones, 2004; Freund and Schapire, 1996), bagging (Breiman, 1996) and random subspace methods (Ho, 1998). The decision tree-based classifier ensemble random forests (Breiman, 2001) is a powerful alternative that combines the two latter strategies.

It has been shown for various applications that discriminative classifiers can achieve higher classification accuracies than generative classifiers if the posterior is a lot simpler than the underlying distribution (Kumar and Hebert, 2006; Vapnik, 2000). This transfers to incremental learners such as incremental SVM or ORF which present good performances, e.g. in tracking tasks (Tang et al., 2007; Saffari et al., 2009).

Logistic regression is one of the oldest discriminative models, which historically goes back early up to 19th century (Cramer, 2003). The basic concept has been extended in many respects, such as by Friedman et al. (2000), which integrated the concept of boosting to an additive logistic regression model or by Deselaers et al. (2011), which have incorporated latent variables. Logistic regression is the basis for advanced models such as incremental logistic regression (Bishop, 2006), kernel logistic regression (Karsmakers et al., 2007; Keerthi et al., 2005; Cawley and Talbot, 2004; Roth, 2001) and sparse kernel logistic regression (Cawley et al., 2007; Tipping, 2001). IVM constitutes a realization of sparse kernel logistic regression. Among various developments of kernel discriminant classifiers, SVMs are presently one of the most popular approach in recent applications. However, (Roscher et al., 2012; Braun et al., 2011; Zhang et al., 2011) have shown that IVMs are comparable to SVMs regarding the performance. A comprehensive empirical comparison of common batch classifiers was conducted by Caruana and Niculescu-Mizil (2006).

The positive properties of generative and discriminative models have been integrated in hybrid generative/discriminative classifiers. The motivation is to formulate a classifier with a reconstructive component while exploiting the discriminative power. Most of the approaches combine a generative and a discriminative classifier (Xue and Titterington, 2010; Uray et al., 2007; Skocaj et al., 2006; Fritz et al., 2005). Others learn a generative model using a sequence of discriminative models (Tu, 2007), interpolate between generative and discriminative parameter estimation (McCallum et al., 2006; Lasserre et al., 2006; Raina et al., 2003) or learn a sparse generative model while retaining as much discriminative power as possible (Kristan and Leonardis, 2010). In considerations about IVM, we will show that the model develops a reconstructive component while at the same time possessing a high discriminative power. In contrary to this, discriminative classifiers such as SVM and random forests inherently do not have a reconstructive component.

Long sequences. Incremental linear subspace learning methods adapt the model parameters by processing newly available data samples, while directly discarding the samples after they have been processed. In contrast, kernel-based algorithms allow a simple adaption of the model by adding new training samples. In order to avoid a steadily increasing complexity of the model when learning over a long time, all kernel-based learners need a component for removing data. Approaches such that of Karasuyama and Takeuchi (2010), Kivinen et al. (2004) or Fung and Mangasarian (2002) show considerable success in incremental learning settings and in general provide update steps to remove data samples in a similar way as data samples are added. The challenge is to perform efficient incremental update steps without suffering a loss in performance.

Probabilities. Besides the predicted class labels of test data, a probabilistic output of the classifier is often of interest. The probabilities can serve as confidence and can therefore be used for either a more accurate evaluation of the classifier or for a further analysis of the classified data. Giacco et al. (2010) for example use the probabilities for uncertainty analysis of landcover classes. The probabilities can also serve as input into a graphical model, as carried out by Roscher et al. (2012) and Kumar and Hebert (2006). In the context of active learning, the probabilities can be used to decide on what data samples are useful in order to extend or replace the current training set (Li et al., 2011; Iglesias et al., 2011). Roscher et al. (2011) have shown that active learning can be performed using an incremental learner without the need to repeatedly retrain the classifier. The probabilities are used to identify relevant samples for updating the incremental learner, and non-informative samples are removed by exploiting diagnostics tools. In contrast to kernel-based probabilistic models such as IVM and sparse multinomial kernel logistic regression (Cawley et al., 2007; Krishnapuram et al., 2005; Tipping, 2001), SVM is non-probabilistic. However, the output of SVM can be transformed to be probabilistic. The most common technique to transform the output to a probability is to fit a sigmoid to the output as proposed by Platt et al. (2000). Other methods are proposed by Grandvalet et al. (2005) and Sollich (2002). Though resulting in an output lying in the range $[0, 1]$, this does not imply that the transformed output is a good approximation of the posterior, i.e. the reliability of these values could be inadequate (Rüping, 2004; Tipping, 2001). The most recent discussion about this aspect can be found in (Franc et al., 2011), in which a generative and semi-parametric probabilistic model is presented that is equivalent to linear SVM. Decision trees provide probability estimates, which arise from frequency-based calculations at the leaf nodes. Simply using the counts of classes at the leaf nodes does not give good probability estimates, as already stated by Chawla and Cieslak (2006). Therefore smoothing methods (Niculescu-Mizil and Caruana, 2005; Provost and Domingos, 2003; Zadrozny and Elkan, 2001) or bagging (Chawla and Cieslak, 2006) have been used to improve the reliability of the estimates, which is why random forests generate better estimates than decision trees.

We have discussed several classifiers regarding their performance, discriminative power, their ability to deal with long-sequences and the their possibly probabilistic output. In the next section, we review the application of incremental classifiers for semantic segmentation.

2.3 Semantic Segmentation

In this thesis, we focus on the application of I^{2}VM for semantic image segmentation, in which each pixel is assigned to a pre-defined class. By grouping adjacent pixels with the same class, we get segments with a semantic meaning. Like all incremental learners discussed before, I^{2}VM is not restricted to this application. In this section, we review the application of incremental learners for the semantic segmentation in single images, images from large databases and image sequences.

Since semantic segmentation is a crucial step for automatic image understanding, the field of applications and algorithms is wide-ranging. The variety of further applications include e.g. object detection/recognition, content-analysis, image understanding or image retrieval and indexing. A review of batch/off-line methods for image segmentation can be found in (Shankar, 2007), (Rogowska, 2000) and (Pal and Pal, 1993).

2.3.1 Semantic Segmentation in Single Images

In the field of semantic segmentation of single images incremental learning can be used to adapt an pre-learned coarse model to the current image. Since the pre-learned model already provides a good approximate solution to the final segmentation in the current image, usually no further human interaction is necessary. Li et al. (2007) incrementally segment skin regions or Wismüller et al. (2004) segments magnetic resonance data sets of the human brain by an incremental self-organized model adaptation.

Remotely sensed single images can be very large and labeling is costly. Therefore, active learning is widely used to incrementally update the set of labeled samples to improve the classification result (Tuia et al., 2009). Furthermore, a large land cover area may be divided into multiple scenes, in which each of them can have both spatial and temporal differences. For the sequential classification of the individual scenes, the classification model must be adapted to the current spectral features. In contrary to Knorn et al. (2009), which re-train the classifier after each data acquisition step, Roscher et al. (2012), Roscher et al. (2012) and Bruzzone and Fernàndez Prieto (1999) also update the classifier incrementally.

Incremental supervised methods are used in the field of user-interactive image segmentation. The user specifies strokes or regions (Duchenne et al., 2008; Rother et al., 2004; Boykov and Jolly, 2001), boxes (Lempitsky et al., 2009; Rother et al., 2004) or contours (Mortensen and Barrett, 1995) within the image to define the training data. The training data is used to learn the classifier model, which is directly applied to the whole image. Once the segmentation has been obtained it can be refined by user-interaction. While most of the approaches re-train the classifier after each refining step, Zhang and Ji (2011) update the classifier model based on the former model. Besides semantic segmentation approaches that classify the image into pre-defined classes such as object and background, contour-based methods search for an optimal contour in the image between these classes. So-called live-wire approaches such as intelligent scissors (Mortensen and Barrett, 1998) or live wire on the fly (Falco et al., 2000), or the more efficient live lane approaches (Kang and Shin, 2002) sequentially obtain a segmentation by defining the contour of the object step-by-step with the help of human user interaction with the mouse. The user roughly traces the boundary by setting some seed points, in which the algorithm chooses the minimum cost contour. The cost function is adapted depending on the previous defined contour. These applications show that incremental learners are useful to efficiently segment in single images.

2.3.2 Semantic Segmentation in Image Databases

Similar to incremental learning in single images, also large image databases can be treated sequentially. The images in the database are presented sequentially to the classifier, because the database can either be too large to be processed simultaneously or the number of images increases continuously.

Vachkov (2010) classifies sequentially arriving images from an image database by comparing them to a core set of images, which define categories using some similarity measure. The core set is extended by a further category if the current image is dissimilar to all given images in the core set. This approach would be equally well suited for semantic image segmentation, where the first few classes with associated features are defined and further classes are added sequentially. Ozawa et al. (2005) and Artac et al. (2002) use incremental PCA to sequentially learn and recognize images in image databases, while keeping only the subspace representation of the input images.

Another approach is online dictionary learning for sparse coding (Mairal et al., 2009), which can treat millions of training samples. The goal is the ability to approximate each part in an image as a linear combination of a few elements from the dictionary, which is also known as sparse coding. Each dictionary element can additionally have a class label, as proposed by Jiang et al. (2011), Ramirez et al. (2010) and Mairal et al. (2008) and dictionary elements with the same class labels can be comprised to a class-specific dictionary. This approach can be used for semantic segmentation by searching for this class-specific dictionary that can approximate a part of the image best. Mairal et al. (2009) sequentially learn from image databases by adapting the content of each dictionary to current images presented to the learning algorithm.

These applications show that incremental learning algorithms are necessary to adapt to current appearances of the classes. In the mentioned application the time-reference is unimportant. Thus, the sequence of the images presented to the learner can be arbitrary and should not influence the overall segmentation result of all images to be tested. The situation is different when using image sequences in which the acquisition time and the time the image is presented to the learner coincide. Instead of treating each image independently, the time can be uses to formulate temporal relation between the image, which can improve the performance of a classifier.

2.3.3 Semantic Segmentation in Image Sequences

One of the largest application for incremental learning is the tracking of objects in image sequences. The main approaches are tracking-by-detection and tracking-by-segmentation. In both cases the tracking task is defined as a classification of successively arriving images into object and background. In contrast to tracking-by-detection where often only a bounding box or an ellipse around the object is obtained, tracking-by-segmentation enables a tight object boundary.

In the last years several tracking-by-detection approaches have emerged which use incremental learning methods in order to handle appearance variability of a tracked object and/or the background. The variability among others comprises pose variations, shape deformations, illumination changes and camera motion. Lim et al. (2004) and Ross et al. (2004) use incremental subspace learning to update the tracking model and Avidan (2007) use an adaptive ensemble of classifiers. Grabner and Bischof (2006) use an incremental version of AdaBoost in order to handle appearance changes and drifting of the detected object boundary. To overcome the drifting problem that occurs when the update of the learner is performed with incorrectly labeled data, the approach was extended to semi-supervised boosting (Grabner et al., 2008). Tang et al. (2007) use semi-supervised online SVM in a co-training framework utilizing color and histogram-of-gradients features (Dalal and Triggs, 2005). Santner et al. (2010) use ORF as adaptive learner in combination with a complementary non-adaptive template-based tracking approach to be robust against the drifting problem while at the same time being adaptive to appearance changes. Babenko et al. (2011) introduce a robust tracker that uses so-called multiple instance learning. This variation of supervised learning deals with groups of acquired samples, which are labeled as object or background instead of labeling each single data sample, which makes the tracking approach more robust against label errors.

Several methods have been proposed treating tracking as a pixelwise binary classification of object and background, which is also known as tracking-by-segmentation. Donoser et al. (2011) deal with an adaptive semantic segmentation in images in an unsupervised manner, whereby the current segmentation is constrained to be consistent to that one in the latter frame. Similarly to this, Charron and Hicks (2010) update a learned mixture of Gaussian model in subsequent frames enforcing coherency across successive frames. Ren and Malik (2007) use a conditional random field (Lafferty et al., 2001) combining an adaptively learned appearance model with the prior knowledge of a temporal-spatial model. Wang et al. (2011) extract simple linear iterative clustering (SLIC) superpixels (Achanta et al., 2010) in each image frame and update a learned generative appearance model to distinguish between the object and the background. Chockalingam et al. (2009) also divide an image into multiple regions and represent the object as a Gaussian mixture in feature-spatial space, i.e. each region is represented by one sub-distribution of the mixture model. The regions are adapted to the current image by a region growing procedure and the Gaussian mixture model is updated by using the average mean of the last and current image statistics. Level-sets are exploited to obtain accurate object contours.

The review indicate that incremental methods are necessary to adapt to the current appearance of object and background. A static model only would be able to track an object as long as its appearance do not change. The review also show that recent approaches use temporal information in order to increase the performance of the algorithm.

The literature review about applications in the field of semantic segmentation shows that the field of applications using incremental learning methods is wide-ranging and most recent approaches have shown that incremental learners are more flexible than batch methods in the mentioned applications. Incremental methods can adapt to current appearance changes, what batch learner are not able to. Given the large area of applications, the biggest challenge is to overcome the stability-plasticity dilemma, which deals with the question how the learner can be stable to irrelevant samples and adaptive to new samples. After the classification of our conception of sequential learning, the review of current incremental learners and applications, in the next chapter we will introduce the theoretical background for the formulation of the I^{2}VM classifier.

Chapter 3

Theoretical Background

This chapter aims at specifying the theoretical background to formulate the incremental classifier I^{2}VM and the task which is meant to be solved by I^{2}VM. In Section 3.1 we define our notation that it used throughout the thesis. In Section 3.2 we describe the supervised sequential learning task comprising the task formulation and the definition of the class of classification model that is used in this thesis. The theoretical background of the I^{2}VM classifier is presented in Section 3.3. In the last section we specify the sequential semantic segmentation problem by means of the formulation of the learning task, the evaluation criteria and the learning experience.

3.1 Notation

In this section we summarize the notation that is used throughout the thesis. We denote vectors $\boldsymbol{g} = [g_i] = [g_1, \ldots, g_I]^\mathsf{T}$ with small bold symbols and matrices $\mathsf{G} = [g_{ij}] = [\boldsymbol{g}_1, \ldots, \boldsymbol{g}_J]$ with elements g_{ij} and column vectors $\boldsymbol{g}_j$ with capital symbols. We use calligraphy symbols for sets. The elements (scalars or vectors) of a set $\mathcal{G}$ can be collected in a vector $\boldsymbol{g}$ or a matrix G by concatenation, using the same letter of the alphabet. The matrix or vector of the concatenated elements of a subset $\mathcal{F} \in \mathcal{G}$ are denoted with $\mathsf{G}_\mathcal{F}$ or $\boldsymbol{g}_\mathcal{F}$, respectively.

In order to simplify the representation of certain matrix-operations, we use a selection operator $\Psi_\mathcal{I}(\mathsf{B})$, selecting the rows $\mathcal{I}$ from a $N \times M$ matrix B, defined as

$$\mathsf{B}_- = \Psi_\mathcal{I}(\mathsf{B}) = \left[\boldsymbol{e}_i^N\right]_{i\in\mathcal{I}}^\mathsf{T} \mathsf{B} \,. \tag{3.1}$$

with $\boldsymbol{e}_i^N$ the i-th unit vector of length N. In a similar manner, the removal of columns can be formulated with the transposition of the considered matrix.

3.2 Supervised Sequential Learning Problem

In this section we specify the general *learning task* for supervised classification problems and expand it to the *sequential learning task*. Furthermore, we define groups of classification models, define the reconstructive and discriminative model components and discuss their meanings for incremental learning. In the last part we consider linear discriminative models for classification which are meant to solve classification problems.

3.2.1 Learning Task

In the following we denote the involved data and define the supervised learning task. We assume to have a training set

$$\{\boldsymbol{x}_n, y_n\} \in \mathcal{T}, \quad n = 1, \ldots, N \tag{3.2}$$

of N labeled samples with M-dimensional feature vectors $\boldsymbol{x}_n \in \mathrm{I\!R}^M$ and class labels $y_n \in \mathcal{C} = \{1, \ldots, c, \ldots, C\}$. The observations are collected in the $(M \times N)$-matrix $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]^{\mathsf{T}}$, while the corresponding labels are summarized in the vector $\boldsymbol{y} = [y_1, \ldots, y_N]^{\mathsf{T}}$. Furthermore, we have given a test set

$$\{\boldsymbol{x}_u, y_u\} \in \mathcal{U}, \quad u = 1, \ldots, U \tag{3.3}$$

of U samples with M-dimensional feature vectors $\boldsymbol{x}_u \in \mathrm{I\!R}^M$ and class labels $y_u \in \mathcal{C} = \{1, \ldots, c, \ldots, C\}$.

We define the task of supervised learning as follows: Our goal is to infer a function f

$$\widetilde{\boldsymbol{y}}_{\mathcal{U}} = f(X_{\mathcal{U}}, \mathcal{M}), \tag{3.4}$$

which models the dependency of the labels $\boldsymbol{y}_{\mathcal{U}}$ to given feature vectors $X_{\mathcal{U}}$, so that the function f behaves similar as learned on the training set $\mathcal{T}$. In the ideal case the predicted labels $\widetilde{\boldsymbol{y}}_{\mathcal{U}}$ are equal or at least close to the true labels $\boldsymbol{y}_{\mathcal{U}}$. The learned model is represented with $\mathcal{M}$ and the function f is fixed and parametrized by $\mathcal{M}$.

The function f can be of a different nature: In this thesis we specify f as a nonlinear function, as described in Section 3.2.5. The function can also be a set of simple decision rules as for decision trees or a set of sub-functions as this is the case for ensemble classifiers. Depending on the specified function, the model $\mathcal{M}$ comprises the following entities: model parameters $\boldsymbol{\alpha}$ and optionally the training set $\mathcal{T}$ and meta-parameters such as a graph structure or tuning parameters. We refer to the function f, the components of $\mathcal{M}$ and the values of the components as the *internal representation of a classifier.*

Until now we have assumed that the training and test data are available a priori. In the next section we define the learning task for successively arriving training and test samples.

3.2.2 Sequential Learning Task

When learning from successively arriving training samples, we assume that at time step t there exist a learned model $\mathcal{M}_t$ comprising a training set

$$\{X, \boldsymbol{y}\}_t \in \mathcal{T}_t \tag{3.5}$$

and the current parameters. The training set $\mathcal{T}_t$ consists of a subset of $\mathcal{T}_{t-1}$ and the new encountered samples. At time step $t+1$ a training set $\mathcal{T}_{t+1}$ including labeled training samples is made available to the classifier. Given a test set

$$\{X_{\mathcal{U}}, \boldsymbol{y}_{\mathcal{U}}\}_t \in \mathcal{U}_t \tag{3.6}$$

the learned model $\mathcal{M}_t$ provides a (probabilistic) class label $\widetilde{\boldsymbol{y}}_{\mathcal{U},t}$ for $X_{\mathcal{U},t}$. The test set may remain the same over time, but can also change steadily. The time step is referred to as the time when the data is presented to the learner rather than the acquisition time. The task of sequential supervised learning is defined as follows: At time step t our goal is to infer a function f_t

$$\widetilde{\boldsymbol{y}}_{\mathcal{U}_t} = f_t(X_{\mathcal{U}_t}, \mathcal{M}_t, \mathcal{M}_{t-1}), \tag{3.7}$$

which models the dependency of the labels $\boldsymbol{y}_{\mathcal{U}_t}$ to given feature vectors $X_{\mathcal{U}_t}$. This function also depends on the last model $\mathcal{M}_{t-1}$. The predicted labels $\widetilde{\boldsymbol{y}}_{\mathcal{U}_t}$ should be as close as possible to the true labels $\boldsymbol{y}_{\mathcal{U}_t}$.

We have described the learning task if the samples successively become available to the learner. In the next section we describe different schemes to learn with sequential data in order to solve a sequential learning task.

3.2.3 Learning with Sequential Data

In this section we mention various examples for learning with sequential data by means of the change of the internal representation of a classifier. The most common changes are listed below:

Adaption of the parameters. The model parameters are adapted with respect to the new encountered samples. The adaptation is performed by adjusting the values of the parameters. Examples using this approach are incremental logistic regression (Bishop (2006), Chapter 3) and incremental subspace methods (Skocaj and Leonardis, 2003). The new samples are discarded immediately after they have been processed and the number of parameters remains the same.

Adaption of the number of parameters. The number of parameters are adjusted. For example, online kernel density estimation (Kristan and Leonardis (2010)) determines the class-conditional distribution of all data samples seen so far using a mixture model. If the distribution changes, the number of used mixture components is adapted resulting in a changed number of parameters. The number of parameters also changes if the number of considered features or the number of classes changes, as this is the case for models such as incremental logistic regression (Wenzel and Hotz (2010)).

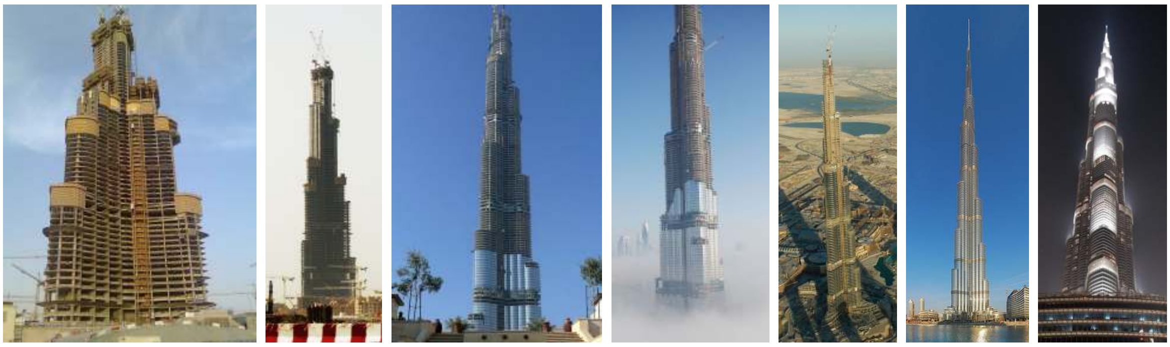
Adaption of the set of training samples. Various classification methods need to store training samples besides the model parameters. Thus, the set of stored samples changes when new encountered samples are added or non-representative ones are removed. Examples using this approach are incremental SVM (Bordes et al. (2007)) and I^{2}VM. In this case, because the decisions made for the classification process arise from a linear combination of the stored training samples and the model parameter, an adaption of the set of training samples means at the same time an adaption of the parameters.

Adaption of the internal complexity. Sparse classification models only use a part of the current internal representation to make decisions for classification purposes. For example, incremental SVM only use a subset of the stored training samples and just the parameters assigned to these samples are non-zero. The complexity changes if the set of used samples changes. Another example are online random forests (Saffari et al. (2009)). The structure of the combination of decision trees are adapted when new samples are presented to the learner.

In most sequential learning task problems more than one adaption need to be used. In this thesis we employ all four adaptations for I^{2}VM, which is discussed in more detail in Chapter 5.

The adaptation of the internal representation is particularly needed if concept-drifts occur.

We speak of concept-drift if the function f_t significantly changes over time. Figure 3.1 [1] [2] [3] shows an example for a concept-drift in an image database comprising images of the same object, namely the newly constructed building Burj Khalifa. During the last years more and more images were added successively. The database first included images of the construction phase and later of the finished object. Furthermore, images by day and night or in different seasons were added. Concept-drifts can be detected either by evaluating the performance measures, e.g. the accuracy, or by evaluation the properties of the classification model. This can be done by analyzing the complexity, or by evaluating the properties of the samples or by analyzing the distribution of the samples.



(a) August 2006 (b) March 2007 (c) December 2007 (d) January 2008, in mist (e) March 2008 (f) December 2009 (g) 2010, by night

FIGURE 3.1: Timeline of the Burj Khalifa building in Dubai under different weather conditions. All images can be found for example on the Google Image Search. Using such a database, a classifier that sequentially learn the appearance of the building over time has to deal with concept-drift, i.e. a significantly change in the distribution of the features of the images.

We have specified various changes of the internal representation of the classifier model when dealing with sequential data. We further explained concept-drifts which occur if the distribution of the samples changes over time. In the next section we specify the reconstructive and discriminative model component of classifier, which are both necessary to enable a robust incremental learning.

3.2.4 Reconstructive and Discriminative Model Components of Classifiers

In this section we specify the reconstructive and discriminative classifier model component by means of the model definition and various examples. Following Skocaj et al. (2006), we define a reconstructive and discriminative model component as follows.

Reconstructive component. Classifiers with a reconstructive component aspire to have a high information content in order to approximate the distribution of the data samples as well as possible. Their main goal is to cover the variability of the training samples rather than to be task-dependent (i.e. to solve the classification task). For our further consideration we relax the explanation to the following definition: Classifiers with a reconstructive component are able to represent the important parts of the distribution. In this case important means that the samples are on the one hand necessary for discrimination and on the other hand necessary to cover the variability of all training samples.

[1] http://www.allaboutskyscrapers.com

[2] http://en.wikipedia.org/wiki/Burj_Khalifa

[3] http://tnwp.blogspot.de/2010/06/at-top-of-burj-khalifa.html

Discriminative component. Contrary to reconstructive models, classifiers with a discriminative component are task-dependent and generally do not provide a good reconstruction of the training samples. They try to discriminate the classes as well as possible. It has been shown for various applications that classifiers with such a component can achieve higher classification accuracies than models with a reconstructive component, if the posterior is a lot simpler than the distribution of the samples (Kumar and Hebert, 2006; Vapnik, 2000).

In respect of a powerful incremental classifier, we are seeking for a model with a high reconstructive power, as discussed by Skocaj et al. (2006) and Uray et al. (2007). This is due to the following fact: Incremental learning methods strive to be efficient, with the result that only a few samples can be kept in memory and the learner have only access to a bounded number of already seen samples. If the model becomes to large, data samples have to be removed with respect to a suitable criterion. Consequently, only the representation of the previously encountered samples are available to the learner. In case of reconstructive methods, the model component can be used as approximation for all discarded samples. In contrary to this, discriminative models are not able to issue a good approximation due to the lack of information about the data distribution.

Figure 3.2 shows exemplary the necessity of a reconstructive model component for incremental learning. The figure illustrates the classification result of a reconstructive and a discriminative model. At time step t the decision boundaries of both models are similar, in contrast to the result at time step $t + 1$. After time step t new samples are added and non-representative ones are discarded. The discriminative model has no information about the distribution of the samples and only these ones positioned near to the decision boundary are kept in the model. The kept samples identified by the reconstructive model still represent the distribution of the samples. Thus, reconstructive models proves to be stable with respect to old learned information and flexible to new encountered ones.

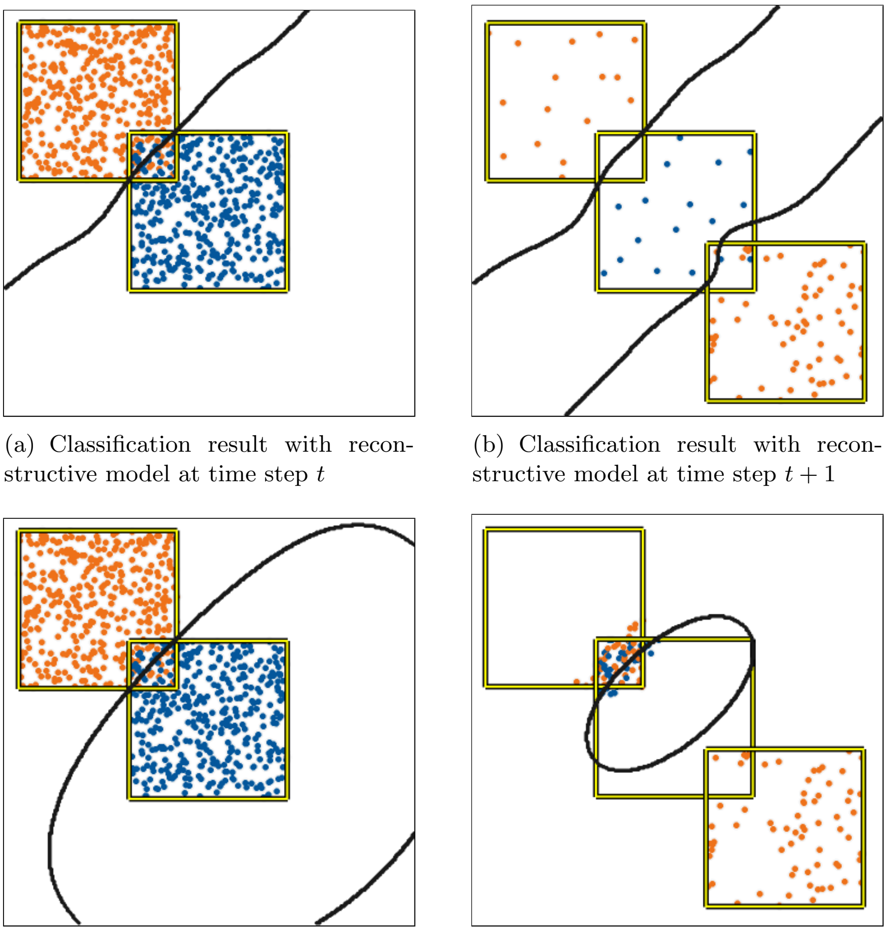
In order to assign the components to different methods, we define groups of classification models. Following Bishop (2006), we distinguish between three approaches to solve a classification problem.

Discriminant functions. The simplest model determines linear discriminant functions $g(\boldsymbol{x})$, which divide the feature space into disjoint decision regions. A feature vector $\boldsymbol{x}$ is directly assigned to a class. In this case, probabilities play no role. An example classifier using this approach is SVM. Classifiers that estimates discriminant functions only have a discriminative model component and no reconstructive one.

Generative models. This approach determines the posterior probabilities $P(\mathcal{C}|X)$ with a generative approach by modeling the class conditional probabilities $P(X|\mathcal{C})$ of the samples and the prior probabilities $P(\mathcal{C})$ separately for each class $c \in \mathcal{C}$. Using Bayes' theorem results in the posterior probabilities $P(\mathcal{C}|X)$,

$$P(\mathcal{C}|X) = \frac{P(X|\mathcal{C})P(\mathcal{C})}{P(X)}\,, \tag{3.8}$$

which allow for decisions. Generative models explicitly or implicitly model the distribution of training as well as testing samples. They are called generative, because they are able to generate new data points by sampling. The most common used classifier which is constructed from a generative model is the maximum likelihood classifier. The classifier can be combined with non-parametric methods such as kernel density estimation to define the class-conditional distribution. Generative models are reconstructive models.



(a) Classification result with reconstructive model at time step t

(b) Classification result with reconstructive model at time step $t+1$

(c) Classification result with discriminative model at time step t

(d) Classification result with discriminative model at time step $t+1$

FIGURE 3.2: Classification result with a reconstructive (top row) and a discriminative model (bottom row) at time step t and time step $t+1$. At time step t the same samples are presented to both models yielding a similar decision boundary (shown in black). After time step t new samples are presented to the classifier and non-representative samples were discarded. The reconstructive model keeps samples that represent the underlying distribution of all samples, whereas the discriminative models keeps the samples that are necessary for discrimination. Both the reconstructive and discriminative model results in similar decision boundaries at time step t but significantly different decision boundaries at time step $t+1$. The contours of the true distribution from which the points are sampled from are given in yellow.

Discriminative models. These models directly determine the posterior probabilities $P(\mathcal{C}|X)$ without any assumption about the underlying distribution of the samples. Thus, they model the dependency of on sample $\boldsymbol{x}$ and an outcome y in a probabilistic way. Using the probabilities, decisions are made to assign a sample $\boldsymbol{x}$ to a class $c \in \mathcal{C}$. The best known classifier which is constructed using a discriminative model is the logistic regression classifier. Discriminative models are not able to represent the underlying distribution of each class and are strictly speaking no reconstructive models. Nevertheless, they model the posterior probability. With our relaxed definition of a reconstructive model component, we can show that under certain requirements the discriminative classifier IVM is also able to develop such a component. We discuss these aspects in detail in Section 4.

We have introduced our relaxed definition of a reconstructive component of a classifier model. Furthermore, we divided the classifiers into groups and label them, whether they have

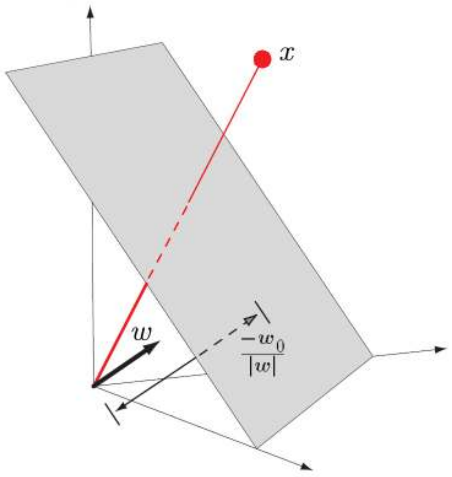


FIGURE 3.3: Illustration of a linear discriminant function in a three-dimensional feature space. The decision surface (shown in gray) is perpendicular to the parameter vector $\boldsymbol{w}$ and the distance to the origin is controlled by the bias w_0. In the two-class case the sign of $\boldsymbol{w}^\mathsf{T}\boldsymbol{x} + w_0$ states the class the sample $\boldsymbol{x}$ is assigned to. Figure modified from (Duda et al. (2001), Chapter 5)

a reconstructive or discriminative component or both of them. In the next section we specify the function f, which we use for classification.

3.2.5 Discriminative Models for Classification

Here we specify the function f for classification in (3.4). In this thesis we use a discriminative model with a non-linear function f, which components are considered in the next section.

3.2.5.1 Generalized Linear Models

For the classification problem we use a class of models which are called *generalized linear models.* They are described by

$$\boldsymbol{y} = f\left(\mathsf{G}\right) \tag{3.9}$$

with $\mathsf{G} = [g_{nc}]$ defined by

$$g_{nc} = \boldsymbol{w}_c^\mathsf{T}\boldsymbol{x}_n + w_{c0} \tag{3.10}$$

with $\{\boldsymbol{w}_c, w_{c0}\}$ as the parameters of a discriminant function defining a hyperplane in the M-dimensional feature space. We specify the function f as

$$f = \operatorname{argmax}_c \frac{\exp\left(\boldsymbol{w}_c^\mathsf{T}\boldsymbol{x}_n + w_{c0}\right)}{\sum\limits_{c'}\exp\left(\boldsymbol{w}_{c'}^\mathsf{T}\boldsymbol{x}_n + w_{c'0}\right)}, \tag{3.11}$$

which is known as the *logistic regression model.* The model can be simplified for the two-class case, in which only one discriminant has to be determined. We will introduce this model in a detailed way in 3.3.1. Figure 3.3 shows a discriminant $\boldsymbol{g}$, i.e. a separating hyperplane, and a feature vector $\boldsymbol{x}$ in the two-class problem in a three dimensional feature space. The so-called bias parameter w_0 specifies the location of the discriminant. The feature vector is assigned to the class $y = 1$, if $f(\boldsymbol{x}, \boldsymbol{w}) \geq 0$, if and only if the dot product $\boldsymbol{w}^\mathsf{T}\boldsymbol{x}$ is larger than the bias w_0. The feature vector $\boldsymbol{x}$ is assigned to the class $y = 2$, if $f(\boldsymbol{x}, \boldsymbol{w}) \leq 0$. The decision surface is $f(\boldsymbol{x}, \boldsymbol{w}) = 0$. The weight vector $\boldsymbol{w}$ specifies the orientation of the discriminant.

We defined the class of models called generalized linear models, which we use in this thesis. If the data samples are not linear separable, the estimation of linear discriminants in the feature space may not be powerful enough for classification. In the next section we will introduce kernels, which are used to estimate complex discriminant functions.

3.2.5.2 Classes of Kernels for Machine Learning

Using so-called *kernel functions*, from now on referred to as kernels, can increase the performance of linear classifiers, as already stated by Vapnik (2000) and Cristianini and Shawe-Taylor (2000). They enable the estimation of complex decision boundaries in the original feature space. The original features $\boldsymbol{x}_n = [x_{nm}], n = 1, \ldots, N, m = 1, \ldots, M$ are transformed into new ones $\boldsymbol{k}_n = [k_{nn'}], n, n' = 1, \ldots, N$ using a kernel function

$$k_{nn'} := k\left(\boldsymbol{x}_n, \boldsymbol{x}_{n'}\right) \qquad \boldsymbol{x}_n, \boldsymbol{x}_{n'} \in \mathcal{T}. \tag{3.12}$$

The vectors $\boldsymbol{k}_n$ are concatenated in a $N \times N$-kernel matrix $\mathcal{K} = [\boldsymbol{k}_n]$.

Besides the necessary condition of being symmetric and continuous, the kernel matrix should be preferably positive (semi-)definite to ensure that a convex optimization problem remains convex when using kernels. Following Genton (2002), we distinguish between different types of kernels:

Stationary kernels. Stationary kernels are translation invariant:

$$k\left(\boldsymbol{x}_n, \boldsymbol{x}_{n'}\right) = k\left(\boldsymbol{x}_n - \boldsymbol{x}_{n'}\right), \tag{3.13}$$

i.e. given an arbitrary translation vector $\boldsymbol{r}$, the kernel depends only on the directional vector between the features $\boldsymbol{x}_n$ and $\boldsymbol{x}_{n'}$. Thus, the function is independent of the position of the features. Examples of non-stationary kernels are the linear and the polynomial kernel.

Isotropic kernels. Isotropic stationary kernels, also known as radial basis functions, only depend on the norm of the directional vector between two features, and thus are only a function of the distance. In contrast, anisotropic kernels also depend on the direction and the length of the features. Isotropic stationary kernels are also used in non-parametric density estimation techniques. Examples for such kernels are the Cauchy kernel, the exponential kernel and the Gaussian kernel, see Table 3.1.

In Chapter 4 we will show that using an isotropic stationary kernel is necessary to obtain a reconstructive component for IVM. Using such a kernel, consequently, the kernel matrix $\mathcal{K}$ consists of affinities between the given features of the training set and contains the new features as rows or columns, see Fig. 3.4.

We have introduced kernel functions, which are used to estimate complex decision boundaries in the feature space. In the next section we consider different loss functions, which are one part in the objective function. We estimate the parameters by minimizing the objective function.

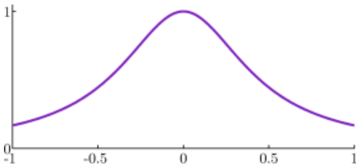
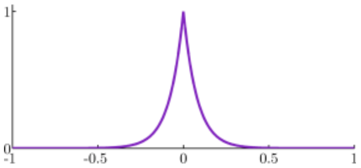
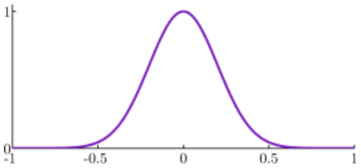
3.2.5.3 Loss-functions

In this section we define various commonly used *loss-functions* for generalized linear models. The optimization function $\mathcal{Q}$ consists of a loss function $\mathcal{Q}_0$ and a penalty/regularization term $\mathcal{Q}_R$,

$$\mathcal{Q} = \mathcal{Q}_0 + \mathcal{Q}_R. \tag{3.14}$$

In the following we introduce specific loss-functions $\mathcal{Q}_0$ used by common classifiers. Conveniently, we assume a two-class problem with targets $t = \{-1, 1\}$ and parameters $\boldsymbol{\alpha}$. The function h can be an arbitrary function as defined in Section 3.2.

Table 3.1: Commonly used isotropic stationary kernels.

Name of the kernel	kernel function	plot
Cauchy quadratic	$\frac{1}{1+\frac{\lvert\boldsymbol{x}_n-\boldsymbol{x}_{n'}\rvert^2}{\sigma}}$	
Exponential	$\exp\left(-\frac{1}{2}\frac{\lvert\boldsymbol{x}_n-\boldsymbol{x}_{n'}\rvert}{\sigma^2}\right)$	
Gaussian	$\exp\left(-\frac{1}{2}\frac{\lvert\boldsymbol{x}_n-\boldsymbol{x}_{n'}\rvert^2}{\sigma^2}\right)$	

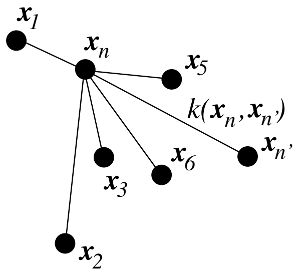


Figure 3.4: Kernel features. Instead of using the original n-th feature vector $\boldsymbol{x}_n$ one uses the affinities $\boldsymbol{k}_n = [k(\boldsymbol{x}_n, \boldsymbol{x}_{n'})]$ of the n-th data sample to all other N training samples n', which depends on the distance between $\boldsymbol{x}_n$ to all features $\boldsymbol{x}_{n'}$.

0-1 loss function. If we only want to count the misclassifications we made, a logical decision is to use the 0-1 loss function $\mathcal{Q}_{0,\text{0-1}}$,

$$\mathcal{Q}_{0,\text{0-1}} = \sum_n I\left(t_n h\left(\boldsymbol{x}_n; \boldsymbol{\alpha}\right) \leq 0\right) , \tag{3.15}$$

where I is the indicator function

$$I\left(t_n h\left(\boldsymbol{x}_n; \boldsymbol{\alpha}\right) \leq 0\right) = \begin{cases} 1 & \text{if} \quad t_n h\left(\boldsymbol{x}_n; \boldsymbol{\alpha}\right) \leq 0 \quad \text{is true} \\ 0 & \text{if} \quad t_n h\left(\boldsymbol{x}_n; \boldsymbol{\alpha}\right) \leq 0 \quad \text{is false} \end{cases} \tag{3.16}$$

For a correct classification, the function $h\left(\boldsymbol{x}_n; \boldsymbol{\alpha}\right)$ is positive and thus, a misclassification is penalized. This loss function is hard to use, leading to a non-convex optimization problem for which there is no efficient algorithm.

Hinge loss. The loss-function especially used by SVM is called hinge-loss defined by

$$\mathcal{Q}_{0,\text{hinge}} = \sum_n \left[1 - t_n h\left(\boldsymbol{x}_n; \boldsymbol{\alpha}\right)\right]_+ \tag{3.17}$$

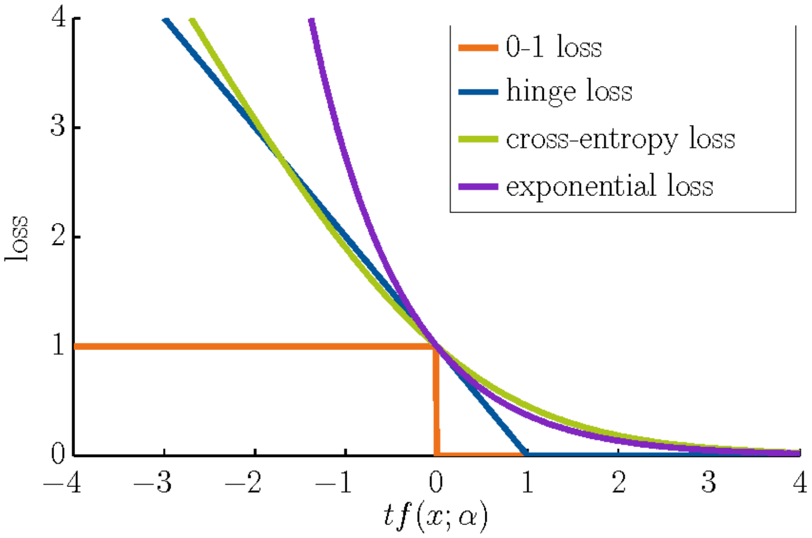


FIGURE 3.5: Loss functions for commonly used classifier: hinge loss (SVM), cross-entropy loss (logistic regression), exponential loss (AdaBoost). Also shown is the 0-1 loss, which counts the misclassifications. The cross-entropy loss is rescaled by a factor of $\frac{1}{\log 2}$, so that it passes through the point $(0, 1)$.

with $[\cdot]_+$ denoting the positive part. SVMs are a maximum-margin classifier, i.e. $\min_n y_n h(\boldsymbol{x}_n; \boldsymbol{\alpha})$. The algorithm chooses a decision boundary which separates the samples of the classes as cleanly as possible, i.e. it allows for some misclassified samples while maximizing the margin between the nearest correct classified samples. The loss function indicates that the more the margin is violated, the higher the penalty is.

Cross-entropy loss. The cross-entropy loss is used for logistic regression models and is defined with

$$\mathcal{Q}_{0,\text{ce}} = \sum_n \log\left(1 + \exp\left(-t_n h\left(\boldsymbol{x}_n; \boldsymbol{\alpha}\right)\right)\right). \tag{3.18}$$

Besides giving the class-membership of each sample, logistic regression also offers a probability estimate. In general when using the cross-entropy loss-function the optimization problem is no longer quadratic and all parameters are non-zero.

Exponential loss. The exponential loss is used in AdaBoost (Freund and Schapire, 1996). The loss function used there is

$$\mathcal{Q}_{0,\text{e}} = \sum_n \exp\left(-t_n h\left(\boldsymbol{x}_n; \boldsymbol{\alpha}\right)\right). \tag{3.19}$$

All mentioned loss functions are plotted in Figure 3.5. For further loss-function in their relation to commonly used classifiers, we refer to Li and Yang (2003). The hinge loss, cross-entropy loss and exponential loss can be seen as an approximation to the 0-1 loss. All are monotonically decreasing functions and differ in the strength of penalizing misclassified samples. The exponential loss penalizes most, whereby the hinge and the cross-entropy loss show a linearly behavior for samples far away from the decision boundary. The exponential loss and the cross-entropy loss are smooth functions. It turns out that even if the training error is zero, the optimization function need not to be converged and will further drive the estimates to an optimal solution in terms of probability estimates. Due to this, using these loss functions does not inherently lead to sparse solutions, in that sense that all samples influence the solution.

We have introduced various loss-functions that are used in well-known classifiers such as Adaboost, SVM and logistic regression. In the next section we rely on the introduced theoretical background to formulate the IVM classifier, which is a sparse kernel logistic regression approach.

3.3 Import Vector Machines

The IVM classifier is based on the classical model of logistic regression, enriched by using kernel features and made efficient by introducing sparsity. This section provides the theoretical basis in order to lay open the reconstructive properties of IVM in the next chapter to motivate the extension to I^{2}VM. We introduce IVM following the derivation of Zhu and Hastie (2005).

3.3.1 Basis Model: Logistic Regression

The basic model of logistic regression starts from the two-class classification problem where the posterior probability $P_n\left(y_n = 1|\boldsymbol{x}_n;\boldsymbol{\alpha}\right)$ for class 1 of a feature vector $\boldsymbol{x}_n$ is assumed to follow the logistic regression model

$$P_{n1}\left(\boldsymbol{\alpha}\right) := P\left(y_n = 1|\boldsymbol{x}_n;\boldsymbol{\alpha}\right) = \frac{1}{1+\exp\left(-\boldsymbol{\alpha}^{\mathsf{T}}\mathbf{x}_n\right)}\,.$$

The posterior probability for class 2 are accordingly $P_{n2} = 1 - P_{n1}$. The extended feature vector is $\mathbf{x}_n^{\mathsf{T}} = [1, \boldsymbol{x}_n^{\mathsf{T}}] \in \mathbb{R}^{M+1}$ and the extended parameters are $\boldsymbol{\alpha}^{\mathsf{T}} = [w_0, \boldsymbol{w}^{\mathsf{T}}] \in \mathbb{R}^{M+1}$ containing the bias w_0 and the weight vector $\boldsymbol{w}$.

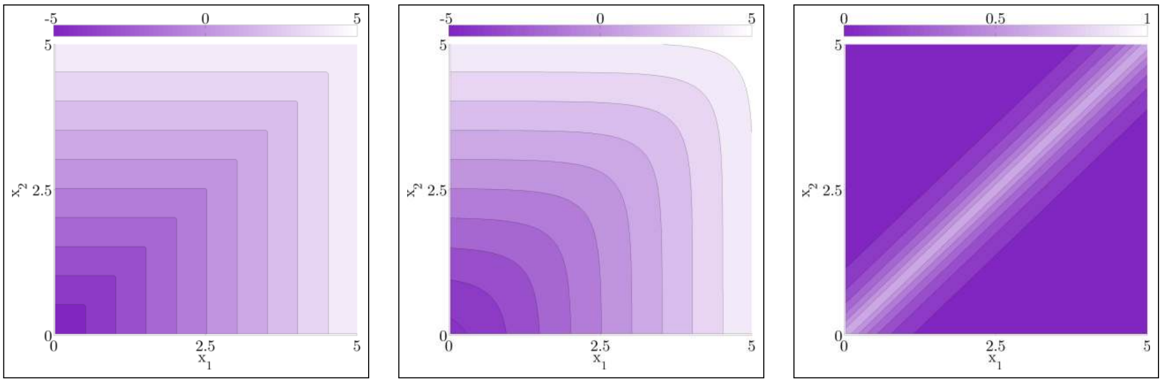
The model can be generalized to the multi-class case with the probabilities $\mathsf{P} = [\boldsymbol{p}_1, \ldots, \boldsymbol{p}_N]$ obtained by

$$P_{nc}\left(\boldsymbol{\alpha}\right) := P\left(y_n = c|\boldsymbol{x}_n;\boldsymbol{\alpha}\right) = \frac{\exp\left(\boldsymbol{\alpha}_c^{\mathsf{T}}\mathbf{x}_n\right)}{\sum_{c'}\exp\left(\boldsymbol{\alpha}_{c'}^{\mathsf{T}}\mathbf{x}_n\right)}\,.$$

The unknown vector $\boldsymbol{\alpha}$ of length $\left((M+1)\,C\right)$ is a concatenation of all C parameter vectors $\boldsymbol{\alpha}_c$. There exist one parameter vector $\boldsymbol{\alpha}_c$ for each class defining a hyperplane in the feature space and thus, the number of model parameters increases linearly with the number of classes. A sample is assigned to a class according to its posterior probability. The logistic regression model uses the *soft-maximum function* instead of the original maximum function, which is why (3.20) is often referred to as softmax activation function. The soft-maximum is contained in the denominator and defined as

$$\log\left(\exp\left(\boldsymbol{\alpha}_1^{\mathsf{T}}\mathbf{x}_n\right) + \ldots + \exp\left(\boldsymbol{\alpha}_C^{\mathsf{T}}\mathbf{x}_n\right)\right)\,. \tag{3.20}$$

The soft-maximum function approximates the original, hard-maximum by smoothing the sharp corners. This is because the difference of the exponents of two values is a lot bigger than the difference of the original values. As Figure 3.6 shows, both, the hard-maximum $\max(x_1, x_2)$ and soft-maximum $\log(\exp(x_1) + \exp(x_2))$ are convex functions. But the soft-maximum function is smooth and indefinitely often differentiable, which make it comfortable to use in convex optimization problems. Using this example, in a three-dimensional space, the hard-maximum function is defined as the intersection of two planes, whereby the soft-maximum function is a smoothed version of it. Transferred to logistic regression, the intersections are the decision boundaries defining the convex acceptance regions for each class. Three



(a) Hard-maximum

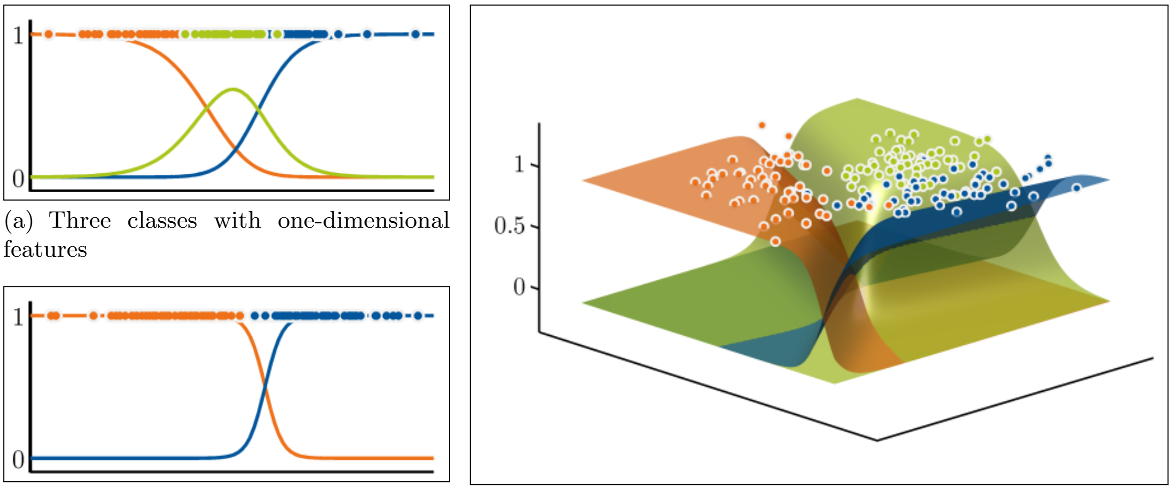
(b) Soft-maximum

(c) Difference of hard-maximum and soft-maximum

FIGURE 3.6: Contour plots of hard-maximum $\max(x_1, x_2)$, soft-maximum $\log(\exp(x_1) + \exp(x_2))$ and the difference between both. The soft-maximum function approximates the original, hard-maximum, but has smooth corners.

synthetic examples are illustrated in Figure 3.7, showing classification results with logistic regression. The posterior probabilities appear intuitive.

Because the probabilities of the classes sum to 1, the model only contains $(M+1)(C-1)$ independent parameters. This can be seen by reducing the ratio in (3.20), e.g. by $\exp\left(\boldsymbol{\alpha}_1^\mathsf{T}\mathbf{x}_n\right)$ without changing the posteriors, leading to $P_{n1} = 1/\left(1+\sum_{c'=2}^{C}\exp\left(\left(\boldsymbol{\alpha}_{c'}-\boldsymbol{\alpha}_1\right)^\mathsf{T}\mathbf{x}_n\right)\right)$. Because of its symmetry, we prefer (3.20) in general, but sometimes use (3.20) for derivations. The learning task is to estimate optimal parameters $\boldsymbol{\alpha}_c$ for each class from the training data, while taking into account that $M+1$ parameters are not identifiable.



(a) Three classes with one-dimensional features

(b) Two classes with one-dimensional features

(c) Three classes with two-dimensional features

FIGURE 3.7: Classwise posterior probabilities (vertical axis) arising from the multi-class logistic regression classifier.

3.3.1.1 Parameter Estimation with Iteratively Reweighted Least Squares

The parameters $\{\boldsymbol{\alpha}_c\}$ are directly determined by maximum likelihood estimation. The likelihood function is given by

$$P\left(\boldsymbol{y}|\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_C\right)=\frac{1}{N}\prod_n\prod_c P_{nc}^{t_{nc}}\,. \tag{3.21}$$

The estimation of the parameters is performed by minimizing the negative log-likelihood function $\mathcal{Q}_0\left(\boldsymbol{\alpha}\right)$,

$$\mathcal{Q}_0\left(\boldsymbol{\alpha}\right)=-\frac{1}{N}\sum_c\sum_n t_{nc}\log P_{nc}\left(\boldsymbol{\alpha}\right) \tag{3.22}$$

with respect to $\boldsymbol{\alpha}$ and $\nabla\mathcal{Q}_0\left(\boldsymbol{\alpha}\right)=0$ as condition for the minimum. The indicator vector $\boldsymbol{t}_n$ for a feature vector $\boldsymbol{x}_n$ belonging to class $y_n=c$ is the n-th unit vector with all elements zero except element c. The function $\mathcal{Q}_0$, up to the factor N, is the cross-entropy between the target probabilities t_{nc} and the estimated probabilities P_{nc}. The iteration process needs the $((M+1)C)$-dimensional gradient and the $((M+1)C\times(M+1)C)$-dimensional Hessian

$$\nabla\mathcal{Q}_0\left(\boldsymbol{\alpha}\right)=\frac{1}{N}[\mathsf{X}^\mathsf{T}\left(\boldsymbol{p}_c\left(\boldsymbol{\alpha}\right)-\boldsymbol{t}_c\right)]_{c=1,\ldots,C}\,, \tag{3.23}$$

$$\nabla^2\mathcal{Q}_0\left(\boldsymbol{\alpha}\right)=\frac{1}{N}[\mathsf{X}^\mathsf{T}\mathsf{R}_{cc'}\left(\boldsymbol{\alpha}\right)\mathsf{X}]_{c,c'=1,\ldots,C} \tag{3.24}$$

with the diagonal $(N\times N)$-matrices

$$\mathsf{R}_{cc'}=\mathrm{Diag}\left([\,P_{nc}\left(\boldsymbol{\alpha}\right)\left(\delta(c,c')-P_{nc'}\left(\boldsymbol{\alpha}\right)\right)]_{n=1,\ldots,N}\right) \tag{3.25}$$

depending on the parameters $\boldsymbol{\alpha}$ (see Appendix A). The $(N\times M)$-matrix X is the concatenation of all feature vectors $\mathbf{x}_n$. The vector of the total gradient $\nabla\mathcal{Q}_0\left(\boldsymbol{\alpha}\right)$ is the concatenation of the gradients $\nabla_{\boldsymbol{\alpha}_c}\mathcal{Q}_0\left(\boldsymbol{\alpha}\right)=\frac{1}{N}\mathsf{X}^\mathsf{T}\left(\boldsymbol{p}_c\left(\boldsymbol{\alpha}\right)-\boldsymbol{t}_c\right)$ w.r.t. each class. The total Hessian consists of $(C\times C)$ blocks $\nabla^2_{\boldsymbol{\alpha}_c,\boldsymbol{\alpha}_{c'}}\mathcal{Q}_0\left(\boldsymbol{\alpha}\right)=\frac{1}{N}\mathsf{X}^\mathsf{T}\mathsf{R}_{cc'}\mathsf{X}$. The Hessian is negative semi-definite. It has a rank deficiency of $(M+1)$, since it can be written as

$$\frac{1}{N}\sum_n\left(P_{nc}\left(\boldsymbol{\alpha}\right)-\left(\delta(c,c')-P_{nc'}\left(\boldsymbol{\alpha}\right)\right)\right)\otimes\mathbf{x}_n\mathbf{x}_n^\mathsf{T}$$

and the left factor of the Kronecker product has rank $(C-1)$ and eigenvector $\mathbf{1}_C$, reflecting that $M+1$ parameters are not identifiable. In the special case of the asymmetric two-class model of (3.20), which we will use for showing I^{2}VM in Section 5, the gradient and the Hessian consist of only one component, referring to the parameters of class one.

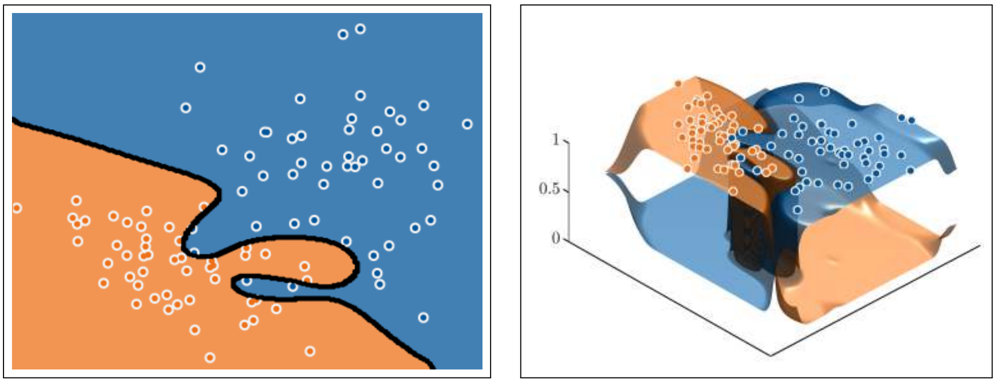
In order to prevent overfitting one may introduce a prior over the parameters and optimize

$$\mathcal{Q}\left(\boldsymbol{\alpha}\right)=\mathcal{Q}_0\left(\boldsymbol{\alpha}\right)+\frac{\lambda}{2}\boldsymbol{\alpha}^\mathsf{T}\mathsf{L}\boldsymbol{\alpha} \tag{3.26}$$

with a positive definite symmetric matrix L and a positive regularization parameter λ, usually determined by cross-validation on the training data set. The matrix L is assumed to be the unit matrix, thus $\mathsf{L}=\mathsf{I}_{(M+1)C}$, but in order to prevent a regularization of the bias parameters w_{0c} the diagonal entries for the bias are set to zero. Thus, only the steepness of the sigmoid function, defined by $\boldsymbol{w}_c$, is regularized.

The iteration scheme can simply be formulated with the Newton-Raphson iteration method

$$\boldsymbol{\alpha}_{(t)}=\boldsymbol{\alpha}_{(t-1)}-\left(\nabla^2\mathcal{Q}_0\left(\boldsymbol{\alpha}_{(t)}\right)+\lambda\mathsf{L}\right)^{-1}\left(\nabla\mathcal{Q}_0\left(\boldsymbol{\alpha}_{(t)}\right)+\lambda\mathsf{L}\boldsymbol{\alpha}_{(t-1)}\right)\,, \tag{3.27}$$



(a) Decision boundary

(b) Classwise posterior probabilities (shown as value on the vertical axis)

FIGURE 3.8: Result of a two-class problem with two-dimensional features arising from the kernel logistic regression classifier.

where the regularization at the same time prevent overfitting and enforces numerical stability. The current iteration is given by t. The Newton-Raphson iteration procedure can be interpreted as the iteratively reweighted least squares (IRLS) optimization method by rearranging (B.1):

$$\boldsymbol{\alpha}_{c,(t)} = \left(\frac{1}{N} \mathsf{X}^\mathsf{T} \mathsf{R}_c \mathsf{X} + \lambda \mathsf{L}\right)^{-1} \mathsf{X}^\mathsf{T} \mathsf{R}_c \boldsymbol{z}_c \tag{3.28}$$

with $\mathsf{R}_c = \mathsf{R}_{cc}$ and $\mathsf{L} = \mathsf{I}_{M+1}$, where $l_{11} = 0$, and

$$\boldsymbol{z}_c = \frac{1}{N} \left(\mathsf{X} \boldsymbol{\alpha}_{c,(t-1)} - \mathsf{R}_c^{-1} \left(\boldsymbol{p}_c - \boldsymbol{t}_c\right)\right) . \tag{3.29}$$

The derivation can be found in Appendix B.

3.3.1.2 Kernel Logistic Regression

The model of kernel logistic regression now presumes the a posteriori probabilities are given by

$$P_{nc}\left(\boldsymbol{\alpha}\right) = \frac{\exp\left(\boldsymbol{\alpha}_c^\mathsf{T} \boldsymbol{k}_n\right)}{\sum_{c'} \exp\left(\boldsymbol{\alpha}_{c'}^\mathsf{T} \boldsymbol{k}_n\right)} . \tag{3.30}$$

with $\boldsymbol{k}_n$ as the n-th column of the kernel matrix K, the unknown model parameters $\boldsymbol{\alpha} = [\ldots; \boldsymbol{\alpha}_c; \ldots]$ referring to the C classes.

The parameters are determined in an iterative way with

$$\boldsymbol{\alpha}_{c,(t)} = \left(\frac{1}{N} \mathsf{K}^\mathsf{T} \mathsf{R}_c \mathsf{K} + \lambda \mathsf{K}\right)^{-1} \mathsf{K}^\mathsf{T} \mathsf{R}_c \boldsymbol{z}_c \tag{3.31}$$

$$\boldsymbol{z}_c = \frac{1}{N} \left(\mathsf{K} \boldsymbol{\alpha}_{c,(t-1)} + \mathsf{R}_c^{-1} \left(\boldsymbol{p}_c - \boldsymbol{t}_c\right)\right) , \tag{3.32}$$

by optimizing the regularized objective function (3.26) with $\mathsf{L} = \mathsf{K}$ similar to (3.28) and (3.29).

Figure 3.8 shows a synthetic example classified with kernel logistic regression. The example illustrates that the classes are separated with a complex decision boundary.

3.3.1.3 Sparse Kernel Logistic Regression

With the rapid development of kernel-based methods, e.g. Keerthi et al. (2005) and Cawley and Talbot (2004) have extended the logistic regression to kernel logistic regression showing a better accuracy but a higher complexity. Using logistic regression or especially its kernel realization can be prohibitive regarding memory and time requirements if the dimension of the features or the number of training samples is large. This is because all training samples are used to train the classifier, which is computationally excessive and memory intensive for large data sets.

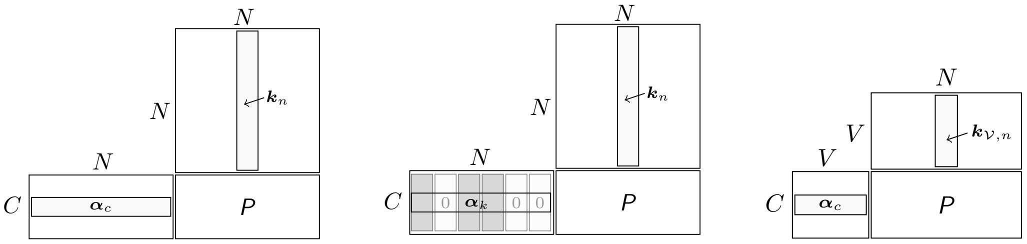
Several *sparse algorithms* have been developed in the last years which enforce sparsity in the kernel logistic regression model in order to control both the generalization capability of the learned classifier model and the complexity. These are, for example, the explicit usage of a sparsity enforcing prior over the parameters, an implicit prior, truncation methods or a greedy subset selection as for the concept of import vector machines.

The relevance vector machines (RVMs) (Tipping, 2001) use an implicit prior as regularization term, the so-called ARD (automatic relevance determination) (Neal, 1996) prior, to induce sparsity. The prior includes several regularization parameters, also called hyperparameters, which are determined during the optimization process. The algorithm have shown to be very sparse, but also tends to underfit leading to a non well-generalized model (Krishnapuram et al., 2005). Additionally, the RVM uses an expectation-maximization (EM)-like learning method and therefore, can suffer from local minima leading to non-optimal classification results.

Alternatively, Cawley et al. (2007) and Krishnapuram et al. (2005) use a Laplace prior enforcing sparsity which assigned regularization parameter is determined via cross-validation. From the regularization point of view, a Laplace prior is the same as L_1-regularization. It encourages the parameters to be significantly large or zero, which means at the same time, that irrelevant features are removed from the model. Liu et al. (2007) propose the L_p with $p < 1$ and show, though leading to a non-convex optimization problem, that this approach outperforms the L_1- regularization. Although these approaches induce sparsity, the optimization is still very expensive. For this reason, efficient optimization procedures such as block-based Gauss-Seidel iterative procedure (Borges et al., 2006) or LORSAL (Li et al., 2011) have been exploited.

Another way to obtain a sparse solution is proposed by Hérault and Grandvalet (2007) which truncate the posterior probabilities to an interval $[P_{min}, P_{max}]$. In preliminary results the approach yields improvements over standard logistic regression regarding classification accuracy.

There are two most common used approaches in order to enforce sparsity. The first approach is this one used by Tipping (2001), Cawley et al. (2007) and Krishnapuram et al. (2005) that introduces a prior over the parameters, which claims that most of the parameters are zero. The second approach is that used by Zhu and Hastie (2005) for IVM that selects a representative subset of samples in order to approximate the full model as well as possible. We refer to this approach as subset-search. In the first approach, the subset results automatically after parameter estimation. The second approach particularly search for the subset, whereas the parameters are estimated simultaneously or subsequently. Figure 3.9 illustrates the difference between both approaches and to kernel logistic regression. The figure schematically shows the matrix multiplication, which is conducted in the softmax function (3.20). For convenience, the lying parameter vectors $\{\boldsymbol{\alpha}_c\}$ are concatenated in a $C \times N$ matrix. The softmax function has to be evaluated in all iterations of both the optimization procedure and the classification step. For kernel logistic regression the whole kernel matrix is set up and



(a) Kernel logistic regression

(b) Kernel logistic regression with sparsity enforcing prior

(c) Kernel logistic regression with subset selection

FIGURE 3.9: Schematic figure of the matrix multiplication conducted in the softmax function for different realizations of kernel logistic regression. The boxes depict the kernel (upper right), parameter (left) and resulting posterior probability matrices (lower right), by which the sizes of each matrix is written directly to it.

all parameters have non-zero values. When using a sparsity enforcing prior some parameters are set to zero, while the number of features remains the same. The corresponding features contained in the kernel matrix therefore have no effect. In contrast to this, sparse kernel logistic regression with subset-search only uses a fraction of the features. As a rule, in all cases only a small percentage of the kernel matrix has to be computed, never the complete matrix.

In the following we will consider the sparse kernel logistic approach that finds a subset $\mathcal{V}$ of V samples out of the training set $\mathcal{T}$, comprising samples $X_{\mathcal{V}} = [\boldsymbol{x}_v]$, $v = 1, \ldots, V$ by subset-search. Thus, only affinities $\boldsymbol{k}_v$ between samples $\mathcal{T}$ and samples in the subset $\mathcal{V}$ are collected in a kernel matrix $K_{\mathcal{V}}$, whereas all other affinities are left out of consideration. Following (3.31) the parameters in iteration t are determined by

$$\boldsymbol{\alpha}_{c,(t)} = \left(\frac{1}{N} K_{\mathcal{V}}^{\mathsf{T}} R_c K_{\mathcal{V}} + \lambda K_R\right)^{-1} K_{\mathcal{V}}^{\mathsf{T}} R_c \boldsymbol{z}_c\,, \tag{3.33}$$

$$\boldsymbol{z}_c = \frac{1}{N}\left(K_{\mathcal{V}} \boldsymbol{\alpha}_{c,(t-1)} + R_c^{-1}\left(\boldsymbol{p}_c - \boldsymbol{t}_c\right)\right)\,. \tag{3.34}$$

The kernel matrix is given by $K_{\mathcal{V}} = [k(\boldsymbol{x}_n, \boldsymbol{x}_v)]$ with $\boldsymbol{x}_n \in \mathcal{T}$, and the regularization matrix by $K_R = [k(\boldsymbol{x}_v, \boldsymbol{x}_{v'})]$ with $\boldsymbol{x}_v, \boldsymbol{x}_{v'} \in \mathcal{V}$. In the following we omit the subscript $\mathcal{V}$ and use K instead of $K_{\mathcal{V}}$.

We have defined sparse kernel logistic regression, in which the subset of samples is found by subset-search. In the following we define the import vector machines algorithm, which is a realization of sparse kernel logistic regression.

3.3.2 Basic Import Vector Machines Algorithm

Following Zhu and Hastie (2005), we define import vector machines (IVM) as a classifier that finds a sub-model to approximate the full kernel logistic regression model using subset-search. In particular, the subset is found by using both the samples and the output, i.e. the posterior probabilities. This distinguishes this approach from those that only use the samples, such as random sampling or methods that identify cluster representatives. We refer to the samples in the subset as import vectors.

The search through all possible subsets to determine the best set is intractable. Therefore, Zhu and Hastie (2005) proposed the determination of the subset in a greedy forward manner.

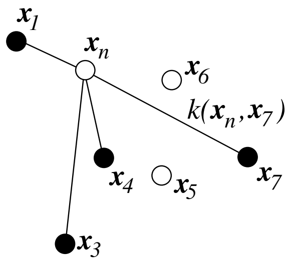


FIGURE 3.10: Import vectors. In order to achieve a small feature vector, here the length of four, only a sparse set of training data is marked as important for the classification which is the set of the so-called import vectors, here being the set $\{1, 3, 4, 7\}$. Thus, only a subset of the the affinities between all samples are used.

We refer to this procedure within the IVM classifier as IVM algorithm, being aware that the subset can also by determined by other techniques. The set of import vectors $\mathcal{V}$ is chosen by successively adding data samples, one at a time, to the initially empty set until a convergence criterion is reached. The data samples are selected according their contribution to the solution, i.e. how much their incorporation to $\mathcal{V}$ decreases the negative log-likelihood function. Figure 3.10 illustrates the effect when only a subset of data samples is used. We revised the IVM algorithm in some respects, which is discussed in the following.

3.3.3 Revised Import Vector Machines Algorithm

In contrast to Zhu and Hastie (2005), we use a hybrid forward/backward strategy, which successively adds import vectors to the set, but also tests if import vectors can be removed in each step. Since we start with an empty import vector set and only add import vectors sequentially in the first iterations, the decision boundary can be very different from its final position. A pure forward selection is unable to remove import vectors that become obsolete after the addition of other import vectors. Therefore a removal of import vectors can lead to a sparser and more accurate solution than only using forward selection steps. To prevent infinite loops between forward and backward steps leading to solutions with similar objective value, deselected import vectors are excluded from selection for the next few iterations. This strategy follows the idea of tabu-search introduced by Glover (1989), in which a memory is used that remembers already visited solutions and user-defined rules. If a potential solution was already selected in the last few iteration or it violates a rule, it is marked as tabu and not selected in the current iteration. The forward/backward strategy is defined in Algorithm 1.

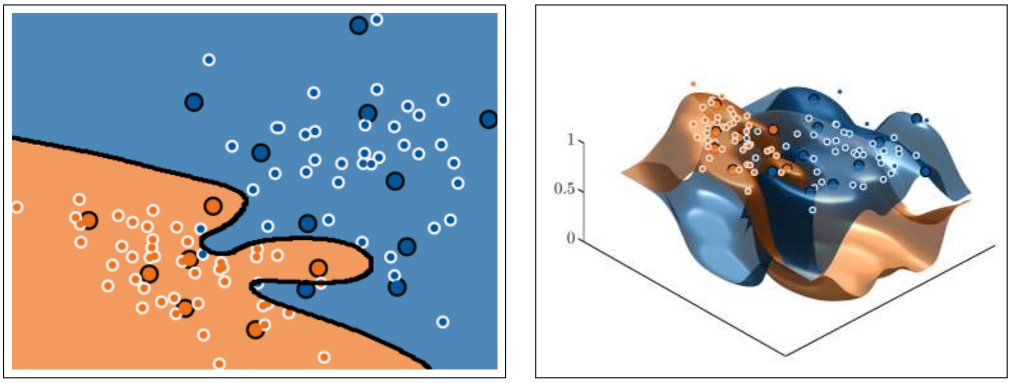
We have carried out another modification regarding the determination of kernel parameter σ and the kernel parameter λ for which we use gridsearch other to Zhu and Hastie (2005): For a given kernel parameter σ, they use a simultaneous selection of the import vector set $\mathcal{V}$ and the regularization parameter λ. First, they split all training samples into a training and a tuning set. Instead of searching through all combinations of λ and σ, as gridsearch works, they propose to start with an empty set $\mathcal{V}$ and a large regularization parameter. Each time the algorithm converges, λ is reduced until a lower bound is reached. The optimal λ corresponds to the minimum classification error on the tuning set. In contrary to Zhu and Hastie (2005), we use gridsearch with cross-validation to determine both the kernel and regularization parameter. We decided to use gridsearch, because in our experiments we observed higher accuracies, especially if the number of training samples is small. We use a five-fold cross validation and test through $\sigma \in \{2^{-3}, 2^{-2.5}, \ldots, 2^4\}$ and $\lambda \in \{\exp(-12), \exp(-11), \ldots, \exp(-4)\}$. To guarantee that these are suitable parameters, we normalize our samples to be in the range $[-1, 1]$.

```
Initialize $\mathcal{V}_0 := \{\}$, $\mathcal{T} := \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, $t := 0$;
repeat
    Compute $\boldsymbol{\alpha}_{c,(t)}$ for each class $c$ from the current set $\mathcal{V}_{(t)}$;
    foreach $(\boldsymbol{x}_n, y_n) \in \mathcal{T} \setminus \mathcal{V}_{(t)}$ do
        Let $\mathcal{V}^n_{(t)} := \mathcal{V}_{(t)} \cup (\boldsymbol{x}_n, y_n)$;
        For each class determine $\boldsymbol{\alpha}^n_{c,(t)}$ from $\mathcal{V}^n_{(t)}$ in a one-step iteration;
        Evaluate error function $\mathcal{Q}^n_{(t)}$;
    end
    Find best point $(\boldsymbol{x}^*, y^*) = (\boldsymbol{x}_n, y_n)$ with $n = \operatorname{argmin}_n \mathcal{Q}^n_{(t)}$, $\mathcal{Q}_{(t)} := \min \mathcal{Q}^n_{(t)}$;
    Update $\mathcal{V}_{(t)} := \mathcal{V}_{(t)} \cup (\boldsymbol{x}^*, y^*)$;
    repeat
        Compute $\boldsymbol{\alpha}_{c,(t)}$ for each class $c$ from the current set $\mathcal{V}_{(t)}$;
        foreach $(\boldsymbol{x}_v, y_v) \in \mathcal{V}_{(t)}$ do
            Let $\mathcal{V}^v_{(t)} := \mathcal{V}_{(t)} \setminus (\boldsymbol{x}_n, y_n)$;
            For each class compute $\boldsymbol{\alpha}^v_{c,(t)}$ from $\mathcal{V}^v_{(t)}$ in a one-step iteration;
            Evaluate error function $\mathcal{Q}^v_{(t)}$;
        end
        Find best point $(\boldsymbol{x}^*, y^*) = (\boldsymbol{x}_v, y_v)$ with $v = \operatorname{argmin}_v \mathcal{Q}^v_{(t)}$;
        if $\min \mathcal{Q}^v_{(t)} \leq \left(\mathcal{Q}_{(t)} + \mu\right)$ then
            Update $\mathcal{V}_{(t)} := \mathcal{V}_{(t)} \setminus (\boldsymbol{x}^*, y^*)$;
        end
    until $\mathcal{V}_{(t)}$ cannot be reduced any more ;
    Update $\mathcal{V}_{(t+1)} := \mathcal{V}_{(t)}$, $t := t + 1$;
until $\mathcal{Q}$ converged ;
```

Algorithm 1: IVM: In every iteration t, each sample $(\boldsymbol{x}_n, y_n) \in \mathcal{T}_{(t)}$ from the current training set $\mathcal{T}_{(t)}$ is tested to be in the set of import vectors $\mathcal{V}_{(t)}$. The point $(\boldsymbol{x}^*, y^*_n)$ yielding the lowest error $\mathcal{Q}^n_{(t)}$ is included. Import vectors are removed from $\mathcal{V}_{(t)}$ if their exclusion do not increase the optimization function $\mathcal{Q}$ plus a small value ϵ. The algorithm stops as soon as $\mathcal{Q}$ converged.

We use the ratio $\epsilon = |\mathcal{Q}_{(t)} - \mathcal{Q}_{(t-\Delta t)}|/|\mathcal{Q}_{(t)}|$ as convergence criterion with a small integer Δt, for example $\Delta t = 1$, as proposed in Zhu and Hastie (2005). Such as the regularization and kernel parameter, the threshold for excluding import vectors μ influences the sparsity of the model. Figure 3.11 shows a synthetic example classified with IVM. The algorithm approximates the result of kernel logistic regression, which is underlined by a visual comparison with Figure 3.8.

We have introduced the modified IVM algorithm, which is the basis for the incremental realization I^{2}VM. We compared it to other sparse realizations of kernel logistic regression. For our further considerations about the discriminative and reconstructive component, we restrict ourselves to the analysis of IVM. However, one can expect that the results look similar for other realizations. The next section specifies the task of semantic image segmentation the I^{2}VM is meant to solve and explains the criteria by means of which we evaluate the performance of I^{2}VM on the task.



(a) Decision boundary

(b) Classwise posterior probabilities (shown as value on the vertical axis)

Figure 3.11: Result of a two-class problem with two-dimensional features arising from IVM classifier. The import vectors are bold plotted with a black boundary.

3.4 Semantic Image Segmentation

In this section we introduce the task of *semantic image segmentation* that I^{2}VM is meant to solve. Furthermore, we mention the criteria by means of which our proposed classifier is evaluated.

3.4.1 Definition of Semantic Image Segmentation

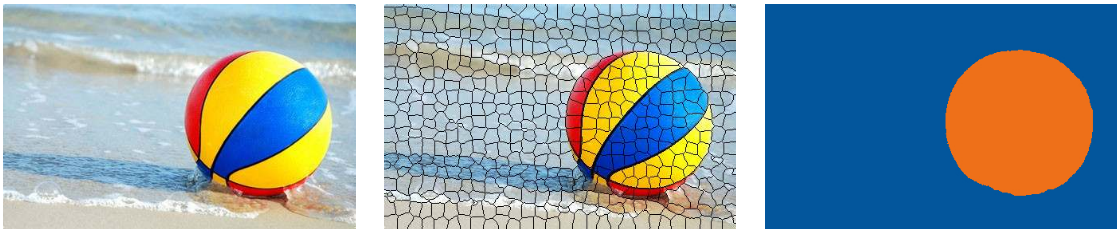
In this section we define the term "semantic segmentation". In general, a segmentation refers to partitioning of an image into segments of equal features such as brightness, color, or texture. However, they need not to be related to a meaningful object. These segments sometimes are referred to as superpixels (Achanta et al., 2010; Levinshtein et al., 2009; Shi and Malik, 2000). Unlike image segmentation, semantic image segmentation aims to a partitioning of the image into regions that are relevant and nameable. Figure 3.12[4] shows an example of the segmentation and the semantic segmentation of an image. In the middle figure, the image is partitioned into distinct regions of equal color features. In the right figure, the image is partitioned into the semantic meaningful segments with the classes `beach ball` and `background`.

3.4.2 Semantic Image Segmentation Problem

The semantic image segmentation problem is one kind of learning problems. Following the definition of a learning problem by Mitchell (1997), we will discuss the semantic image segmentation task, the performance measure and the learning experience. The learning experience is discussed by means of the type of training experience available to the learner, the controllability of the sequence of samples and the representativity of training samples in relation to test samples. We will discuss the (batch) semantic image segmentation problem and the semantic image segmentation with sequentially arriving training samples. The semantic image segmentation problem is given as follows.

Task. The task of semantic image segmentation is the partitioning of an image into semantic meaningful segments. To achieve this the image is divided into non-overlapping regions such

[4] `Photoshopessentials.com`



(a) Original image

(b) Segmented image with overlayed boundaries of the superpixels using the approach of Levinshtein et al. (2009)

(c) Semantic segmented image

FIGURE 3.12: Example for segmentation and semantic segmentation: Segmentation aims on the partitioning of the image into distinct segments of equal color features (middle), whereas the semantic segmentation aims on the partitioning of the image into non-overlapping segments with a semantic meaning (right image). In this case the image is partitioned into the classes `beach ball` and `background`.

TABLE 3.2: The structure of a confusion matrix. The major diagonal elements depict the true positives $N_{\mathrm{tp},c}$. The false negatives $N_{\mathrm{fn},c}$ and false positives $N_{\mathrm{fp},c}$ can be computed be summing over off-diagonal elements in the corresponding row or column, respectively. The overall accuracy a_{oa} is the sum of major diagonal elements divided by the total number of elements. The class-wise accuracies $a_{\mathrm{aa},c}$ are the sums of the rows divided by the total number of elements and the class-wise reliabilities a_{r_c} are the sums of the columns divided by the total number of elements. In general only the gray part is displayed for evaluation.

		Reference class			Accuracy
		1	2	c	
Predicted class	1	$N_{\mathrm{tp},1}$	$N_{\mathrm{fn},1}$		a_{aa_1}
	2	$N_{\mathrm{fp},1}$	$N_{\mathrm{tp},2}$		a_{aa_2}
	c			$N_{\mathrm{tp},c}$	a_{aa_c}
Reliability		a_{r_1}	a_{r_2}	a_{r_c}	a_{oa}

as pixels, rectangular blocks or superpixels. Each region is classified with a suitable method into a pre-defined class. Neighboring region with the same class are grouped to one semantic segment.

Performance Measure. We evaluate the performance of a classifier by comparing the predicted class labels $\widetilde{\boldsymbol{y}}_{\mathcal{U}}$ with the true labels $\boldsymbol{y}_{\mathcal{U}}$ using 3 criteria. The criteria can be determined by evaluating the so-called confusion matrix. The confusion matrix is a square array expressing the number of samples assigned to the true class relative to the number of samples assigned to the predicted class.

In order to specify our used performance measure, we define three class-specific quantities, which are the number of true positives $N_{tp,c}$,

$$N_{\mathrm{tp},c} = \sum_u \delta\left(\widetilde{y}_u, c\right) \delta\left(y_u, c\right) \tag{3.35}$$

the number of false negatives $N_{\mathrm{fn},c}$,

$$N_{\mathrm{fn},c} = \sum_u \left(1 - \delta\left(\widetilde{y}_u, c\right)\right) \delta\left(y_u, c\right) \tag{3.36}$$

and the number of false positives $N_{\mathrm{fp},c}$,

$$N_{\mathrm{fp},c} = \sum_u \delta\left(\widetilde{y}_u, c\right) \left(1 - \delta\left(y_u, c\right)\right). \tag{3.37}$$

True positive sample are correctly classified as class c, false negative samples belong to class c but are incorrectly classified and false positive samples are incorrectly classified as class c.

The first criterion is the overall accuracy a_{oa},

$$a_{\mathrm{oa}} = \frac{1}{U} \sum_{u} 1 - \delta\left(\widetilde{y}_u, y_u\right) = \sum_{c} N_{\mathrm{tp},c} \tag{3.38}$$

that is the percentage of misclassified pixels. The performance measure is meaningful if the classes are equivalent and comprise the same number of samples. However, a more likely used criterion is the test error a_e, which directly depends on a_{oa}, since $a_e = 1 - a_{oa}$.

The second criterion is the average accuracy a_{aa}. The class-specific accuracy $a_{\mathrm{aa},c}$ is given by

$$a_{\mathrm{aa},c} = \frac{N_{\mathrm{tp},c}}{N_{\mathrm{tp},c} + N_{\mathrm{fn},c}} \tag{3.39}$$

from which the average accuracy is derived, defined by the mean of the class-specific accuracies,

$$a_{\mathrm{aa}} = \frac{1}{C} \sum_{c} a_{\mathrm{aa},c}. \tag{3.40}$$

In contrary to the average accuracy, this performance measure considers different class sizes.

When dealing with remote sensing images, it is common to use the kappa coefficient, which was introduced by Cohen (1960). Similar to a correlation coefficient is determines if two segmentation results are significantly different. The kappa coefficient κ is given by

$$\kappa = \frac{a_{\mathrm{oa}} - a_{\mathrm{e}}}{1 - a_{\mathrm{e}}} \tag{3.41}$$

with a_{e} as the so-called chance agreement

$$a_{\mathrm{e}} = \frac{1}{U^2} \sum_{c} \left(N_{\mathrm{tp},c} + N_{\mathrm{fn},c}\right)\left(N_{\mathrm{tp},c} + N_{\mathrm{fp},c}\right) . \tag{3.42}$$

The performance measure ranges from $[-1, 1]$. Landis and Koch (1977) suggest the following interpretation of the values of κ: $\kappa < 0$ means a poor agreement, $\kappa = [0, 0.2]$ means a slightly agreement, $\kappa =]0.2, 0.4]$ is a fair agreement, $\kappa =]0.4, 0.6]$ is moderate agreement, $\kappa =]0.6, 0.8]$ is a substantial agreement and $\kappa =]0.8, 1.0]$ is and almost perfect agreement.

Besides the error rates, the probabilistic output can be used to analyze the reliability and the uncertainty of the classification result. We assess the reliability of the probabilities by rejecting uncertain test samples and deriving the classification accuracy on the non-rejected test points. Following Giacco et al. (2010), the accuracy provided by a classifier can be represented as a function of the rejection rate in discrete intervals. The rejection rate is given by a threshold on the posterior probability. A classifier yield reliable posterior probabilities if samples with high class probabilities are accurately classified, whereas relatively low class probabilities are more likely assigned to misclassified samples. Further the number of retained samples can be represented as a function of the rejection rate, which indicates the uncertainty of the classification result.

Training experience. The classifier has a direct access to labeled samples each consisting of a feature vector extracted from the pixel itself and its neighborhood and a class membership. Furthermore, the classifier has a simultaneous access to all samples and thus, the sequence generally has no influence onto the classification result. To ensure a good performance of the classifier, the distribution of the training samples should be similar to that of the test samples, i.e. the training samples need to be representative.

We have defined the semantic image segmentation problem if all training samples are given a priori. In the next section we apply this definition to the case of successively arriving samples.

3.4.3 Sequential Semantic Image Segmentation Problem

In this section we define the learning problem for sequential semantic image segmentation with successively arriving samples.

Task. As specified for semantic image segmentation, the task of sequential semantic image segmentation is the partitioning of an image into semantic meaningful segments.

Performance measure. Besides the performance measures we use for semantic image segmentation, we further use the stepwise overall accuracy $a_{\text{oa},(t)}$, which is defined as the overall accuracy for each time step t. Since a learned classifier model can be seen as best model learned so far, the learner can at any time produce a prediction for all test samples. If the set of test samples do not change, the performance of the incremental learner should improve over time.

Training experience. The training experience of an incremental classifier generally increases over time. Nevertheless, if new encountered samples do not contain any new information for the learner, the training experience can stagnate. Furthermore, if the classifier is allowed to forget knowledge, the experience may decrease. If the forgetting has a large negative effect onto the learning task, we speak of "*catastrophic forgetting*".

In order to encounter new experience the learner needs access to labeled samples becoming available over time. Labeled samples can be provided randomly outside of the learners control or in a selective way, e.g. samples positioned near to the decision boundary. In many applications the availability of labels for each sample cannot be guaranteed, which is why active learning approaches are applied. In this case, the learner interactively query a user, another information source or even the learner itself (also known as self-training) to label unlabeled samples. These approaches need to be treated with caution, since it cannot be guaranteed that the class assignments are correct.

In sequential learning problems the representativity of the training samples inherently cannot be guaranteed. Thus, suitable criteria are needed to add and remove training samples to keep the set representative. We will discuss criteria for removing training samples in Section 5.6.

We have discussed the sequential semantic image segmentation problem which is meant to be solved by I^{2}VM. The next chapter will consider the reconstructive and discriminative model component of IVM, which are both needed for a powerful incremental learning.

Chapter 4

Discriminative and Reconstructive Properties of Import Vector Machines

In Section 3.2.4 we defined classifiers with a reconstructive component as those which are able to represent the important parts of the distribution. Important samples are those that are necessary for discrimination and at the same time necessary to cover the variability of all training samples. Classifiers with a discriminative model component try to discriminate the the classes as well as possible.

In this section we show that the IVM though being a discriminative model inherently have a reconstructive component. We verify this fact by means of the analysis of the objective function. Further, we underline our findings by an empirical study on the distribution of the import vectors and the influence of various kernels onto the reconstructive component.

4.1 Statement of the Problem

In order to obtain a powerful incremental classifier, it need a high discriminative but also reconstructive power, as stated in Section 3.2.4. Only generative classification models use an estimate for the conditional distribution of the samples $P(X|Y)$, which is the base for an efficient long-term learning of many classes. Probabilistic discriminative models including IVM provide an estimate of $P(Y|X)$. In contrast, discriminative function estimation procedures such as SVM only provide estimates for the class membership of the data samples, which can be transformed to probabilistic outputs. However, as stated in Section 2 this does not imply that the transformed output is a good approximation of the posterior. Other probabilistic classifier such as decision trees and random forests model frequency-based probabilities rather than the conditional distribution. Thus, we are confronted with the problem that discriminative learning methods, including probabilistic ones, do not necessarily contain a reconstructive model component as generative models do. Nevertheless, in the next section we formulate a hypothesis that IVM have both a discriminative and reconstructive model component.

4.2 Hypothesis

We analyze the following hypothesis: Using a suitable kernel, the discriminative IVM classifier develop a reconstructive model component with these properties:

(a) The IVs cover the distribution of the data samples.

(b) Non-overlapping areas with a high density of data samples achieve a high posterior probability.

(c) Areas with no training samples obtain a posterior probability of approximately $\frac{1}{C}$.

Properties (a) and (b) are necessary to enable a robust incremental learning even if the distribution of the data samples changes. Property (c) is necessary to enable an immediate response to the emergence of other classes in these areas and to make reliable claims about the posterior probability of test samples. The latter claim is quite intuitive, because we cannot make any assumption about the class membership of test samples lying in areas with no training samples.

In order to obtain both a reconstructive and discriminative component, we exploit the following property of IVM: Although IVM are based on the discriminative logistic regression model, which estimates decision boundaries between the classes $\mathcal{C}$, the IVs are selected in order to decrease the negative log-likelihood function (see (3.22)), also called cross-entropy objective function

$$\mathcal{Q}_0\left(\boldsymbol{\alpha}\right) = -\frac{1}{N}\sum_n\sum_c t_{nc}\log P_{nc}\left(y_n = c|\boldsymbol{x}_n\right)\ . \tag{4.1}$$

All samples $\boldsymbol{x}_n$ with label $y_n = c$ pursue to achieve a high posterior probability P_{nc} for class c. Therefore, all samples chosen as IVs contribute to the posterior probability P_{nc}, though they not necessarily influence the position of the decision boundary. In the following we rewrite (4.1) in order to show the parts of the equation which lead the reconstructive and discriminative component.

The non-regularized negative log-likelihood function of IVM which is meant to be minimized is given by

$$\mathcal{Q}_0\left(\boldsymbol{\alpha}\right) = -\frac{1}{N}\sum_n\sum_c t_{nc}\log\left(\frac{1}{\sum_{c'}\exp g_{nc'}}\exp g_{nc}\right) \tag{4.2}$$

with $g_{nc} = \boldsymbol{\alpha}_c^{\mathsf{T}}\boldsymbol{k}_n$. We rewrite the equation yielding

$$\mathcal{Q}_0\left(\boldsymbol{\alpha}\right) = -\frac{1}{N}\sum_n\sum_c t_{nc}\left(\log\left(\frac{1}{\sum_{c'}\exp g_{nc'}}\right) + g_{nc}\right)\ , \tag{4.3}$$

$$= \underbrace{-\frac{1}{N}\sum_n\log\left(\frac{1}{\sum_{c'}\exp g_{nc'}}\right)}_{\text{log-partition function}} \underbrace{-\frac{1}{N}\sum_n\sum_c t_{nc}g_{nc}}_{\text{weighted KDE}}\ . \tag{4.4}$$

The first part sometimes is called log-partition function and the second part can be seen as an unnormalized kernel density estimation. The log-partition function is not to be confused with the logarithm of the partition function, as $\mathcal{Q}_0\left(\boldsymbol{\alpha}\right)$ is not normalized. Conveniently, we first discuss the second part.

The second part forms an unnormalized weighted kernel density estimation. Kernel density estimation (KDE) is a non-parametric method, often used in combination with generative classifiers, to estimate the density $p(X)$ of given data samples. Generally, the weighted kernel density estimator is

$$p(X) = \frac{1}{N}\sum_n \gamma_n k(x_n, X)\,, \tag{4.5}$$

where γ_n are the weights for each point and k is the kernel function. If some weights are zero, the estimator turns out to be sparse. For standard KDE the weights have to fulfill the condition $\sum_n \gamma_n = 1$, whereas for kernel logistic regression this condition is not necessary. This means that the optimization of the second part in (4.4), considered on its own, leads to a kernel density estimation of the given training data, additionally taking account of the targets of the samples. Contrary to standard KDE, kernel logistic regression does not model the density separately for each class and thus, is not a generative classifier. Assigned to kernel logistic regression the parameters $\boldsymbol{\alpha}$ are the weights $\boldsymbol{\gamma}$ and the IVs are the centers of the used kernels.

The partition function $\frac{1}{\sum_{c'} \exp g_{nc'}}$ in the first part of the equation is used as normalization in order to determine the posterior probabilities P in (3.20), which are per definition in the range $[0, 1]$. The function includes the samples of all classes and thus, changes in g_{nc} also influences all neighbored samples, independently of their class-membership. The magnitude of the influence is defined by the kernel function $k(\boldsymbol{x}_n, \cdot)$ and its hyperparameters. For example, using a Gaussian radial basis function kernel, the IV centered on it highly influences all samples in the 1-σ confidence region, whereas only slightly influences the samples beyond the 3-σ confidence region.

The value of the partition function depends on the position of the selected IVs, due to the usage of the exponent:

$$\sum_n \exp(g_{nc}) = \sum_n \exp(\boldsymbol{\alpha}_c^\mathsf{T} \boldsymbol{k}_n)\,, \tag{4.6}$$

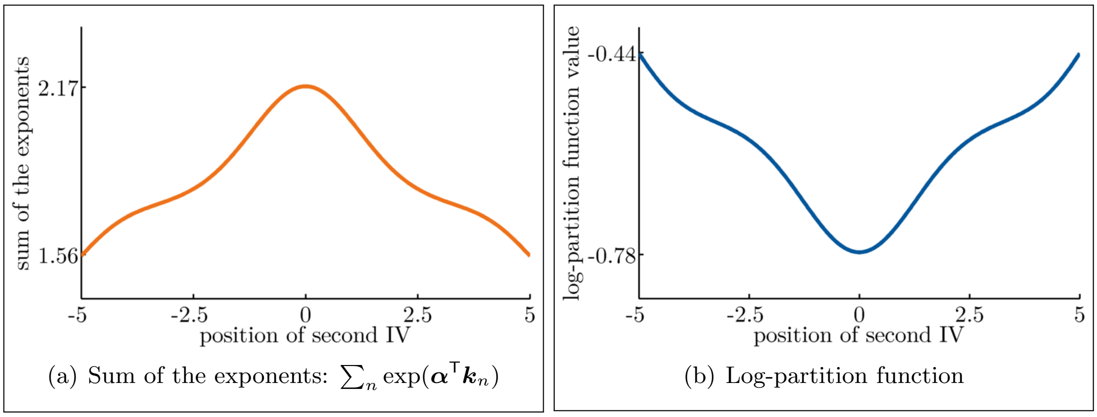
$$= \exp(\boldsymbol{\alpha}_c^\mathsf{T} \boldsymbol{k}_1) + \ldots \exp(\boldsymbol{\alpha}_c^\mathsf{T} \boldsymbol{k}_n) + \ldots + \exp(\boldsymbol{\alpha}_c^\mathsf{T} \boldsymbol{k}_N)\,. \tag{4.7}$$

If we consider 2 samples and g_{1c} is a little bigger than g_{2c}, $\exp(g_{1c})$ will be a lot bigger than $\exp(g_{2c})$. That means, exponentiation exaggerates the difference between the values of g_{1c} and g_{2c}. Due to this, the IVs tend to lie far from each other resulting in an uniformly coverage of the IVs.

Working example 1. In order to show the dependency between the value of the log-partition function and the position of the IVs, we design a working example with the following boundary conditions: We choose 1000 one-dimensional samples within the range $[-5, 5]$ and neglect the class labels, because they have no influence on the example. The samples are arranged in a grid resulting in equal distances between the samples. We use a Gaussian radial basis function kernel with kernel width $\sigma = 1$. We select the first IV to be positioned at $x_n = 0$ and vary the position of the second IV. We plot the value of $\sum_n \exp(g_n)$ and the log-partition function as a function of the position of the second IV.

Figure 4.1 shows that the higher the distance between the IVs, the lower is the function value of the log-partition function. Because the negative log-likelihood function includes the negative log-partition function, the objective function is decreased most if the IVs are position far from each other. The reason is that nearby IVs occasionally cause very high values, which are enhanced by the exponent, and far apart IVs only cause relatively small values.

Working example 2. Figure 4.2 shows a working example that demonstrates the sample-wise contribution of both discussed parts to the objective function. We choose 2 classes, whereas each comprises 500 one-dimensional samples within the range $[-5, 5]$. The samples are arranged in a grid resulting in equal distances, whereas the samples of class 1 lie in the range $[-5, -2[$ and $[2, 5]$ and the samples of class 2 lie in the range $[-2, 2[$. We select the first IV to be positioned at $x_n = 0$ with label $c_n = 2$. We plot the function value that each

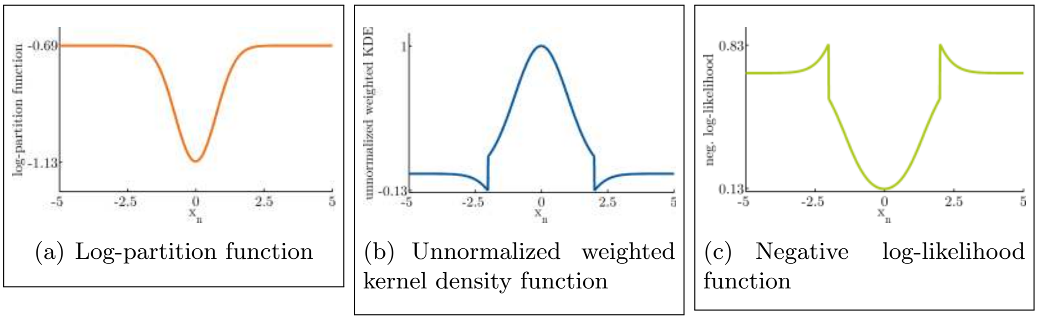


(a) Sum of the exponents: $\sum_n \exp(\boldsymbol{\alpha}^\mathsf{T}\boldsymbol{k}_n)$

(b) Log-partition function

FIGURE 4.1: The dependency between the value of $\sum_n \exp(\boldsymbol{\alpha}^\mathsf{T}\boldsymbol{k}_n)$ and the log-partition function and the position of the IVs. The first IV is positioned at $x_n = 0$ and the position of the second IV is varied. The higher the distance between the IVs, the lower is the function value of the log-partition function.

sample contribute to the log-partition function, the unnormalized KDE and the negative log-likelihood function. We use a Gaussian radial basis function kernel with kernel width $\sigma = 0.1$. Conveniently, we set $\alpha = [-1, 1]$.



(a) Log-partition function

(b) Unnormalized weighted kernel density function

(c) Negative log-likelihood function

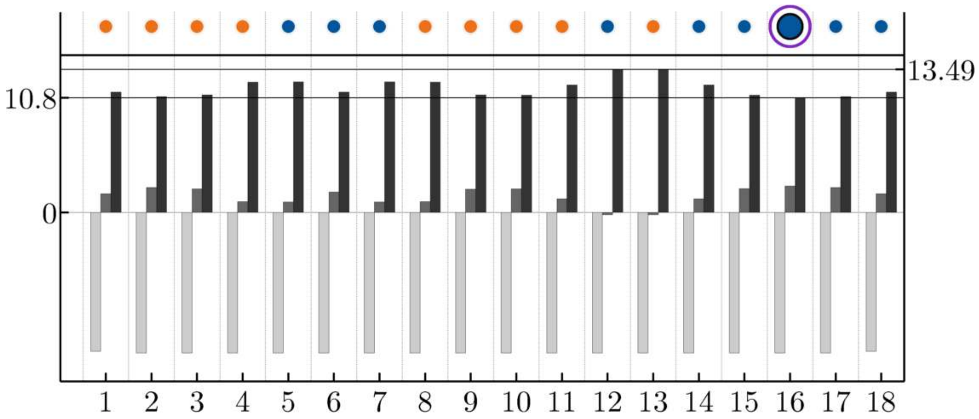
FIGURE 4.2: Sample-wise contribution to the log-partition function, the unnormalized kernel density function and the negative log-likelihood function after the selection of the first IV positioned at $x_n = 0$. The the negative log-likelihood function is given by the sum of the negative log-partition function values and the negative unnormalized weighted kernel density function values.

The log-partition function, illustrated in Figure 4.2 (a), results from the influence of all samples. Figure 4.2 (b) shows the unnormalized weighted kernel density estimation. Due to the multiplication with the targets the plot is characterized by two edges, which indicate a change in the labels. Figure 4.2 (c) illustrates the negative log-likelihood. In plot 4.2 (b) it can be seen that samples within the range $[-2, 2[$ have small values and samples beyond the range have large function values. The largest values are obtained from samples that are near to the selected IV with competitive class labels. Thus, due to the usage of both parts in the objective function, the IVs are not only selected for reconstructive purposes but also retain the discriminative power by penalizing the loss of discrimination.

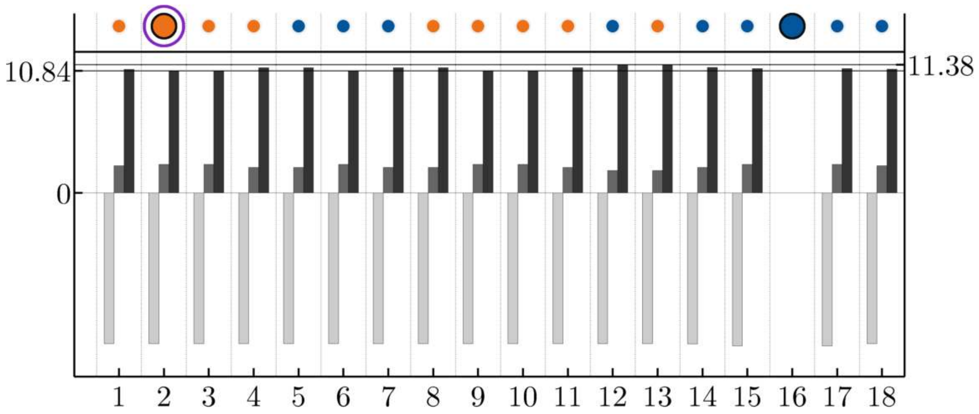
Working example 3. In the following we consider a working example demonstrating the successive selection of the IVs by means of the value of the log-partition function, the unnormalized weighted kernel density function and the negative log-likelihood function after the selection. We choose 2 classes, whereas each comprises 9 one-dimensional samples within the range $[0, 1]$. We use a Gaussian radial basis function kernel with kernel width $\sigma = 0.1$ and conveniently, we fix the parameters for each IV and set it to $\boldsymbol{\alpha}_v = [-1, 1]$. Furthermore, we set the regularization parameter to $\lambda = 0$.

The following plots show the one-dimensional data samples of the working example in the upper part and the corresponding values of the log-partition function (light gray), the unnormalized kernel density function value (medium gray) and the negative log-likelihood function (dark gray) after the selection of the corresponding IV directly below as a bar plot. The values are obtained by evaluating the objective function after the selection of the corresponding IV. On the abscissa the values denote both the coordinates of the data samples and the numbering of the points. One vertical line specifies the minimum value of negative log-likelihood function (marked left on the ordinate) and one vertical line indicates the maximum value (marked right on the ordinate). The data sample with the lowest negative log-likelihood function value is selected as an IV. The IVs are bold plotted and the currently selected IV is bordered violet.

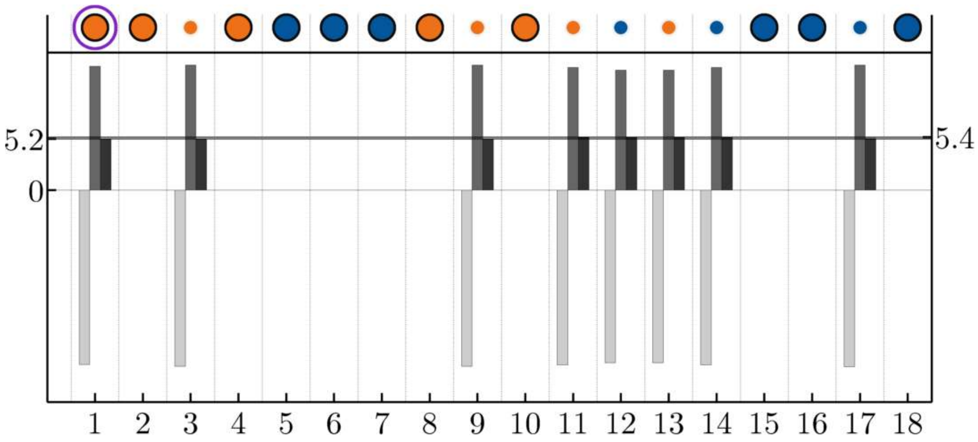
The first IV is that one with the most neighbors of the same class membership. The value of the log-partition function is the same for all candidate IVs, except for those on the margin. The class-wise kernel density function value differs significantly depending on the class labels of the neighbored samples.



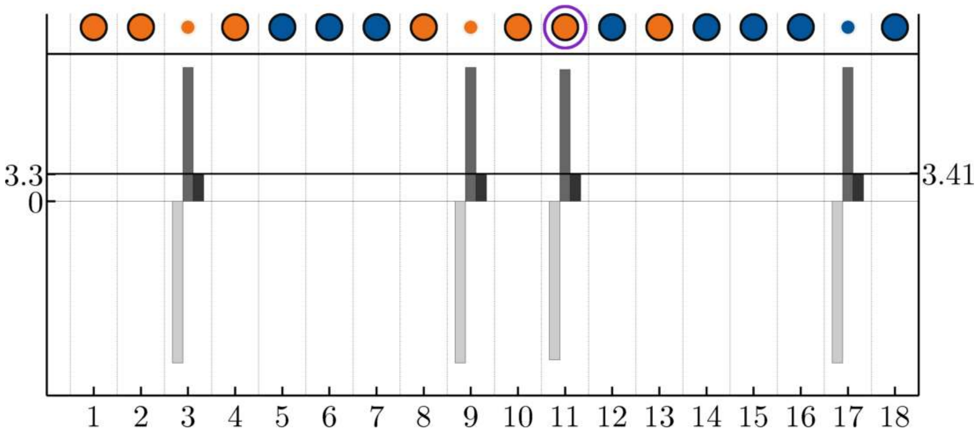
The second selected IV lies in the second densest region of samples with equal class label. Again, the values for the log-partition function differ only slighty, whereas the unnormalized weighted kernel density function value highly depends on the class labels of the neighbored samples.



When selecting the 11. IV we can see that the remaining samples are these ones, which neighbors are already selected IVs and belonging exclusively to the same class, and these ones, which neighbors are no IVs and belonging to the competitive class. The values for the log-partition function is lower in areas with low density of IVs.



When selecting the 15. IV, the plot indicates that the remaining samples are these ones, which neighbors are already selected IVs and belonging exclusively to the same class. Thus, samples that are already covered by neighbored IVs of the same class result in the least decrease of the objective function and are selected in the end.



Our considerations show that the objective function (4.4) which is used by IVM, causes a coverage of the samples by IVs. Using the greedy selection procedure, the first selected IVs are positioned in areas with a high density. Candidate IVs near to already selected ones are only chosen if they can significantly decrease the objective function, i.e. if they can contribute to a higher discriminative power. Usually, this is only the case when they have a competitive class label. Due to this and the fact that the hyperparameters are selected equally for all IVs, the IVs uniformly cover the samples independent of their density. Beside this reconstructive properties, the selection of the IVs seeks a high level of discriminative power, resulting in the effect that IVs beyond the decision boundary are only selected if they do not decrease the discriminative power. In the next section we empirically analyze our hypothesis.

4.3 Empirical Study on the Distribution of Import Vectors

In this section we empirically analyze the distribution of IVs. First, we consider the usage of different kernels to show that a kernel needs to have certain properties so that IVM develop a reconstructive component. Second, we analyze the distribution of the IVs using synthetic data sets with separable and overlapping classes and real data sets characterized by high dimensional features. Third, we show the influence of the kernel - and regularization parameter onto the classification result. Our results are compared to the distribution of support vectors of SVM.

4.3.1 Data

For our considerations we use the following data sets.

Ripley data set. For the analysis of the usage of various kernels we use the Ripley data set (Ripley (2008))[1], which is a well-known machine learning data set consisting of two overlapping classes with 250 samples. The two-dimensional samples for each class have been generated by a mixture of two Gaussian distributions.

Synthetic data set. For the analysis of the distribution of the IVs we use two synthetic non-overlapping data sets. For each data set we generate 500 two-dimensional samples for each class, which are simulated from a Gaussian mixture with two components and from an uniform distribution. Furthermore, we use three data sets with overlapping classes, in which each class contains 500 two-dimensional features. We use two data sets with 2 classes and one with 4 classes.

Digits data set. We also analyze the distribution of IVs in a high-dimensional feature space. We use the DIGITS data set of Seewald (2005) consisting of 1900 training and 1800 test images of digits from 0 to 9. We compute HoG features of each image and use them for classification purposes. Class 1 is comprised of the even images of digits 0, 2, 4, 6 and 8, which we refer to as sub-classes and class 2 is comprised of the odd images of digits 1, 3, 5, 7 and 9.

4.3.2 Experimental Setup

On the usage of various kernels. As described in Section 3.2.5.2, we can use different kernels for kernel logistic regression. In this experiment we visually analyze the classification result of IVM by means of property (a)-(c), which are necessary to obtain a suitable reconstructive component. We use non-stationary kernels, isotropic stationary kernels with compact support and isotropic stationary kernels with infinite support (see Section 3.2.5.2). The kernel and regularization parameter were determined via 5-fold cross-validation.

Analysis of the distribution of import vectors. To analyze the distribution of IVs we perform IVM using some representative examples with two classes, see Figure 4.4. We start with two well separable classes, illustrate the distribution of the IVs and later show the distribution for overlapping classes. In case of separable classes we run the total experiment twice, once with the first IV selected according to Algorithm 1 and once selected randomly. In all cases we use a common Gaussian radial basis function kernel and choose both suitable kernel parameter $\sigma = 0.25$ and regularization parameter $\lambda = \exp(-5)$ to achieve a small misclassification error. We repeat the procedure and accumulate the IVs, until at least 2000 IVs are accumulated. To visualize the distribution of the IVs we plot the accumulated IVs shown as orange and blue dots. Furthermore, we attach to each IV its Gaussian kernel with assigned kernel width (see Section 3.3.1.2), so that the accumulation of the kernels results in

[1] available at `http://www.stats.ox.ac.uk/pub/PRNN/`

the background shading in the images. In each run we compute the distances between all IVs of each class and report the accumulated histograms. For comparison we show the theoretical distribution of the distances between randomly selected samples derived from the underlying distribution. In each run we choose as many random samples as IVs were selected.

4.3.3 Results and Discussion

In the following we present and discuss our results regarding the usage of different kernels and the distribution of the IVs.

4.3.3.1 On the Usage of Various Kernels

Figure 4.3 shows the obtained classification results using common kernels, namely non-stationary kernels such as linear and quadratic kernels, isotropic stationary kernels with compact support such as uniform and Epanechnikov kernel and isotropic stationary kernels with infinite support such as exponential, Cauchy and the most commonly used Gaussian kernel.

Non-stationary kernels. Figure 4.3 (a) and (b) show the linear and the quadratic kernel. The kernel function value depends in the position of each sample. This is shown in that the greater the distance to the decision boundary the higher the posterior probability, independently of the distance to IVs. Using a higher order polynomial kernel can lead to a more complex and therefore better discriminating decision boundary, but property (c) cannot be met.

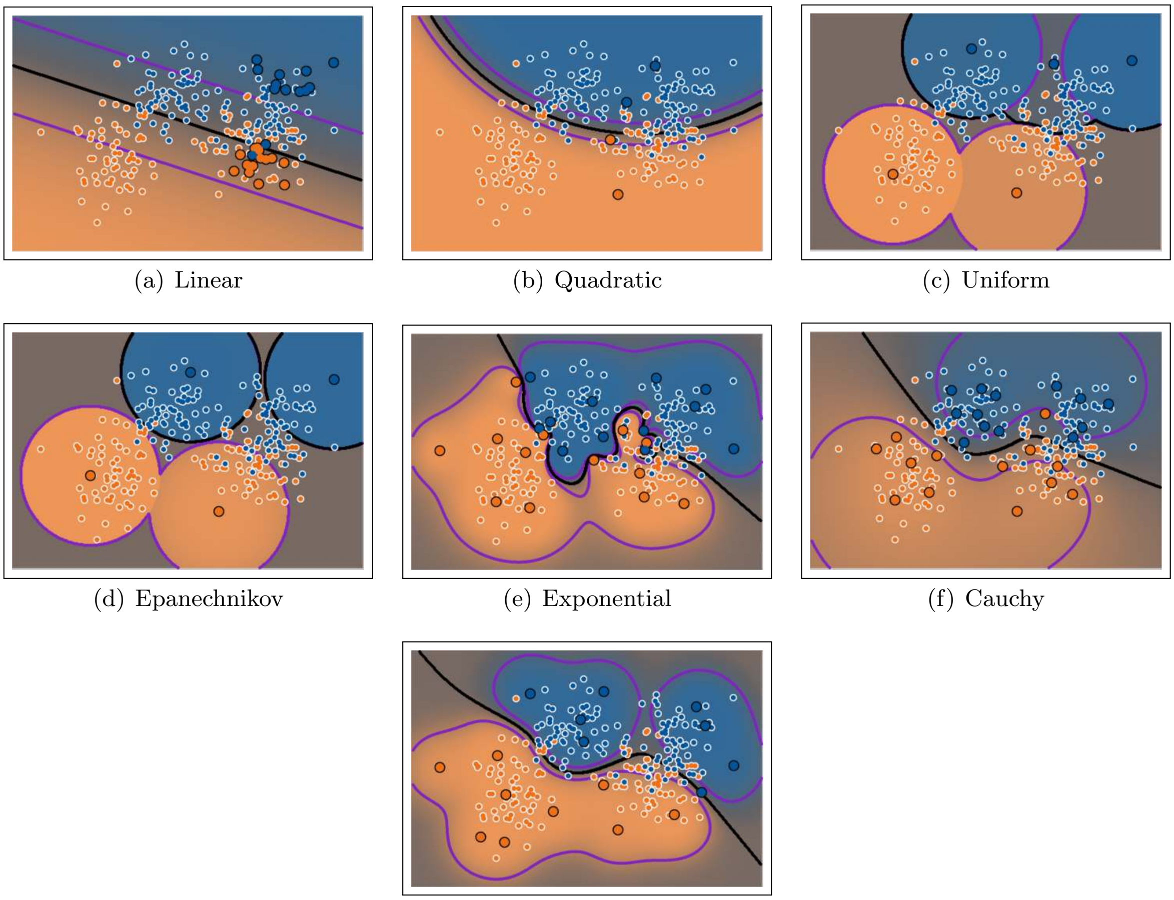
Stationary kernels. Figure 4.3 (c) and (d) show isotropic stationary kernels with a compact support. Although all properties are met, these kernels cannot model a suitable posterior probability. The posterior probability distribution is not smooth, so that the decision boundary and the 0.75 isolines coincides with the parts of the border of the compact support of the kernel centered on the IVs.

Figure 4.3 (e) and (h) shows stationary kernels with infinite support. These kernels meet all properties and obtain also a smooth posterior probability distribution.

4.3.3.2 Analysis of the Distribution of Import Vectors

In the following we discuss our results regarding the distribution of IVs using synthetic data shown in Figure 4.4. Once we select the first IV according to Algorithm 1 (left column) and once with select it randomly (right column). Furthermore, we discuss the case of overlapping classes by means of the results given in Figure 4.6. We consider samples with high dimensional features in Figure 4.8.

Separable classes. Figure 4.4 indicate that the IVs obviously are spread over the entire data set. The left column in Fig. 4.4 shows that there is a hole around the center of each class distribution. The reason is, that IVs which are selected first lie in a region with a high density measured by the kernel, i.e. the center of each class, see Figure 4.5. Furthermore, because of the sparsity property of the IVM algorithm and the property, that IVs tend to gain distance towards one another, the neighboring samples are "covered" by IVs in the center of the distribution. Therefore, the distribution of the IVs is not identical to the density of the data samples. IVs are rather distributed so that areas with an adequate occurrence of data samples are uniformly covered by IVs. These empirical findings underline our hypothesis



(a) Linear

(b) Quadratic

(c) Uniform

(d) Epanechnikov

(e) Exponential

(f) Cauchy

(g) Gaussian

FIGURE 4.3: Commonly used kernels: Non-stationary kernels such as linear and quadratic kernels, isotropic stationary kernels with compact support such as uniform and Epanechnikov kernel and isotropic stationary kernels with infinite support such as exponential, Cauchy and the most commonly used Gaussian kernel. The kernel parameter, i.e. the width of the kernel, and regularization parameter were determined via 5-fold cross-validation. The IVs are bold plotted with a black boundary.

formulated in Section 4.2, in which we explain these effects by means of the analysis of the objective function.

The results are underlined by the histogram $h(d)$ of the distances between IVs. The histogram $h(d)$ of the Gaussian mixture distribution as well as the uniform distribution tend to have two peaks. Distances around the right peak belong to distances between samples in the center and non-centered IVs, and distances between the sub-distributions in case of the Gaussian mixture distribution, and the left peak are distances between IVs in the center or distances outside the center.

We also report the analysis, provided that the first IV is chosen randomly in order to identify its purpose, see right column in Figure 4.4. Now both peaks are less distinctive. Nevertheless, small distances between IVs appear to be less frequent than sampled from the original density.

Overlapping classes. The situation is slightly different in case of overlapping classes, see Figure 4.6. Mostly, the IVs belonging to one class do not lie in the acceptance region of the other class, but they uniformly cover the posterior probability distribution. Again, we

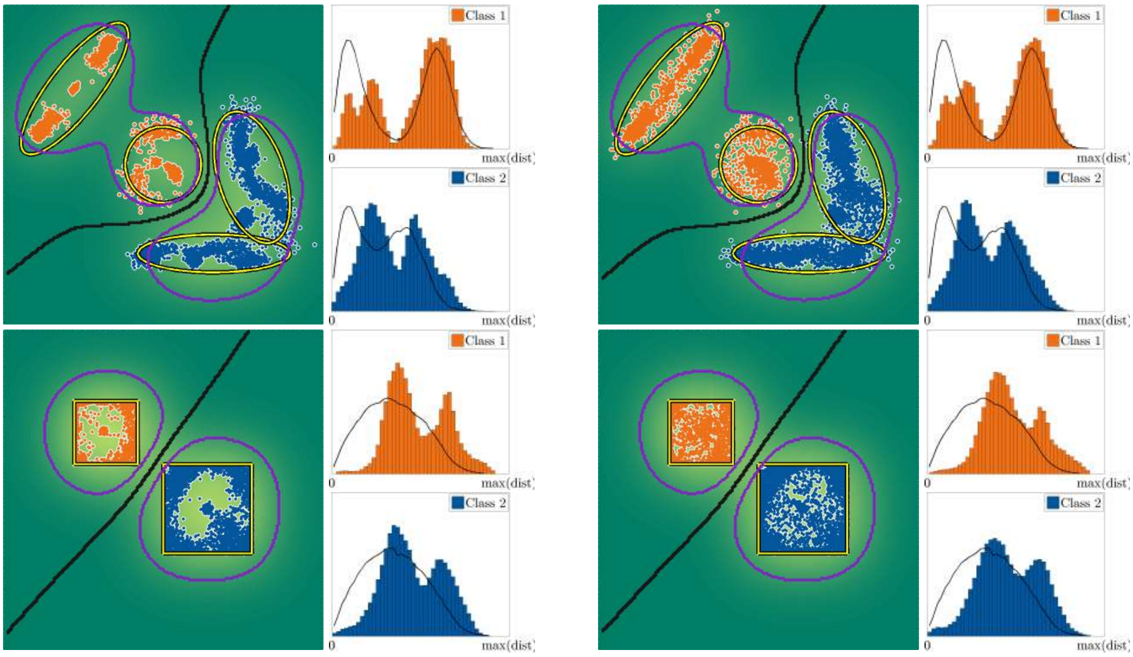


FIGURE 4.4: Empirical distribution (dots) of import vectors (IVs) for features in a two class scenario with well separated classes: Gaussian (upper row) and uniform (lower row) distribution of the features. The size of the shown region is 2.5×2.5. The decision boundaries (black) estimated by IVM and the isolines of the 25 % and the 75 % posterior probabilities (violet) are given. Joint distribution from 500 runs of an IVM, each yielding appr. 15 IVs. The incremental sampling of the IVs starts either with the best vector (left, see Algorithm 1) or with a random vector (right). The contours of the true distributions are given in yellow (3-σ contour for the upper row). The agglomerated histograms give the distribution of the distances between the IVs per run, indicating that IVs tend to lie separately, as the occurrences of small distances between IVs appear to be less frequent, compared to the theoretical distribution shown as lines. For the explanation of the holes see the text below.

observe the existence of IVs far off the decision boundary. The sparsity property in all cases is underlined by the fact that very small distances are the exception. The IVs try to resist each other, i.e. they do not represent a density function.

Figure 4.7 shows an example with 4 overlapping classes using different kernel and regularization parameter. Like in the two-class scenarios we can observe that the IVs are covering the whole distribution. The center of the plots, which are characterized by an equal density of all classes, are in most cases not covered. An IV in the overlapping area is only selected if the discriminative power is not reduced. This is only the case if the range of influence of the IV, defined by σ, comprises significantly more samples of the same class than of competitive classes. This can be seen in Figure 4.7 (a) and (b), where the kernel parameter is small and thus, the range of influence is also small.

It can be seen that the results shown in Figure 4.7 (a), (c) and (e) have no 75% contours of the posterior. That means, the maximum probability for all classes is smaller than 75%. This is due to the regularization parameter, which influences the sharpness of the probabilities. The results illustrated in Figure 4.7 (b), (d) and (f) have these contours, because the regularization parameter is smaller than for the results shown in (a), (c) and (e). By means of these contours we can also see the influence of the kernel parameter. Using a small kernel parameter yield contours that enclose the areas with high density, whereas the usage of large kernel parameters yield contours that also enclose areas with a low density. Thus, the determination of a suitable kernel and regularization is crucial and demand for strategies as grid-search with cross-validation.

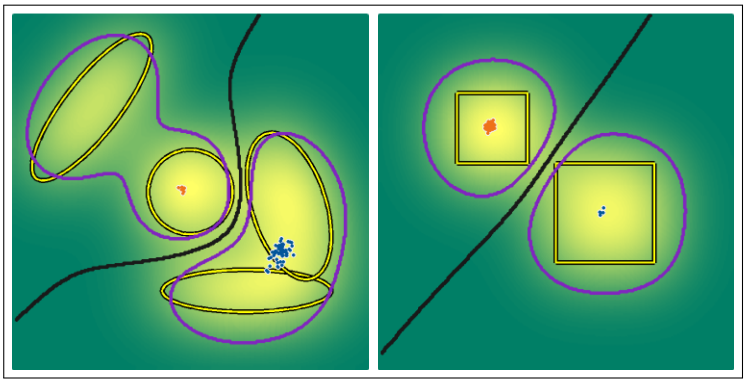


FIGURE 4.5: Empirical distribution (dots) of the first selected import vectors for the Gaussian (left) and uniform (right) distribution of the features. The decision boundaries (black) estimated by IVM and the isolines of the 25 % and the 75 % posterior probabilities (violet) are given.

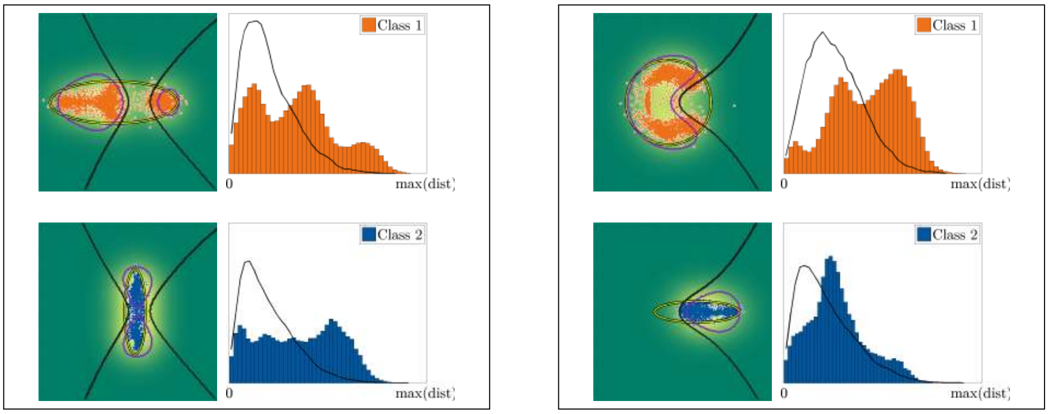


FIGURE 4.6: Empirical distribution (dots) of import vectors (IVs) for Gaussian-distributed features in a two class scenario with overlapping classes. The 3-σ contour of $P(\boldsymbol{x} \mid \mathcal{C})$ is given in yellow. The decision boundaries (black) estimated by IVM and the isolines of the 25 % and the 75 % posterior probabilities (violet) are given. Most of the IVs of a class lie in the acceptance region of the corresponding class.

Digits data set. The left column of Figure 4.8 shows the images of the IVs and the right column shows the histogram of the distances between them. As stated before, the results indicate that short distances between the IVs are the exception. All subclasses are covered by the IVs as the left images show. Thus, also in high-dimensional spaces the selected IVs represent the important parts of the distribution.

4.3.3.3 Comparison to SVM Regarding the Distribution of Support Vectors

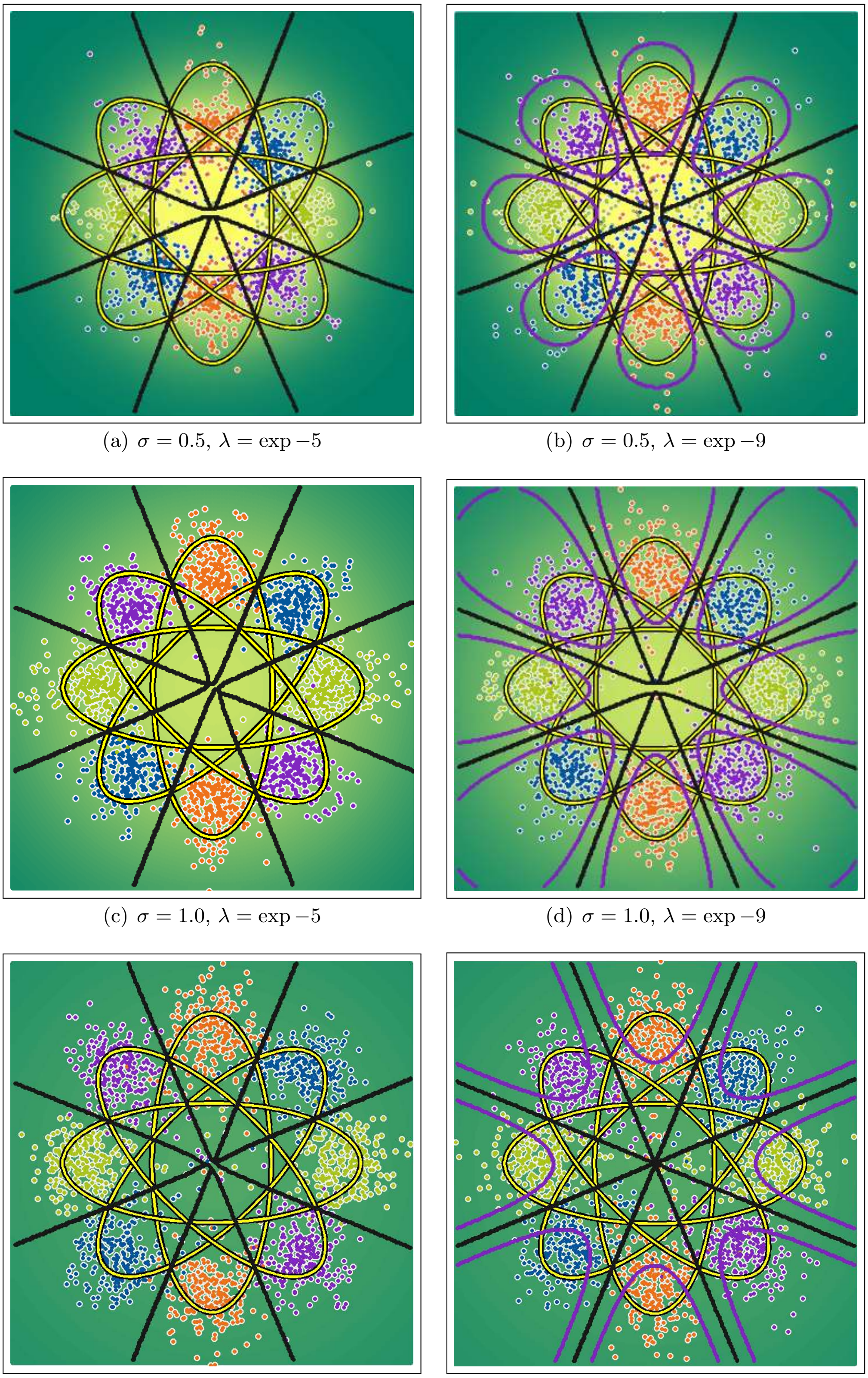
We compare our results to that of standard SVM. Fig. 4.9 shows the results for SVM with kernel parameter $\sigma = 0.25$, which we also use for the experiments with IVM. We use $\lambda = \exp(2)$ for the respective left plot and $\lambda = \exp(0)$ for the respective right plot for both the Gaussian mixture and the uniform distribution. In contrast to the uniformly distributed IVs, support vectors (SVs) of SVM are only selected if they are necessary for discrimination. Thus, most of the SVs are positioned near the decision boundary within the obtained margin in the feature space of the kernels. This not necessarily results in the statement, that SVs are positioned near to the decision boundary in the original feature space, as can also be seen in the examples given by Schölkopf and Smola (2002), p. 207, and Hastie et al. (2009), p. 425. We also plot the 75 % isolines of the posterior probabilities (violet) which we obtain using

Platt's method Platt et al. (2000). The figures show that the probabilities strongly depends on the decision boundaries. Areas, which lie far from the decision boundary achieve a high posterior probability, no matter if there are data samples. We observe the same effect, when using non-stationary kernels with IVM.

Fig. 4.10 shows the results for SVM for overlapping classes. The kernel and regularization parameter are determined via cross-validation. In these examples the SVs tend to be positioned near the decision boundary. This is underline by the histograms over the distances between the SVs, which shows that SVs are positioned close together covering only a small area. It can also be observed that overlapping areas also achieve a relatively high posterior probability, if they lie far enough away from the decision boundary.

4.4 Summary

We analyzed the objective function of IVM and showed that the classifier have both a reconstructive and discriminative model component. The reconstructive part of the objective function causes a uniformly coverage of the samples by IVs. Beside this reconstructive properties, the selection of the IVs seeks a high level of discriminative power, resulting in the effect that IVs beyond the decision boundary are only selected if they do not decrease the discriminative power. The analysis of the usage of various kernels has shown that only stationary kernels with an infinite support are suitable for IVM in order to develop a reconstructive component. The empirical analysis on the distribution of the IVs clearly indicates that the IV selection leads to a sampling of the acceptance domain of the given training data. We demonstrated that non-overlapping areas with a high density of data samples achieve a high posterior probability and areas with no training samples obtain a posterior probability of approximately $\frac{1}{C}$. Thus, we confirmed our hypothesis that to our definition in Section 3.2.4, IVM using a stationary kernel inherently have a reconstructive model component. In the next chapter we formulate the I^{2}VM classifier.



(a) $\sigma = 0.5,\ \lambda = \exp -5$

(b) $\sigma = 0.5,\ \lambda = \exp -9$

(c) $\sigma = 1.0,\ \lambda = \exp -5$

(d) $\sigma = 1.0,\ \lambda = \exp -9$

(e) $\sigma = 2.0,\ \lambda = \exp -5$

(f) $\sigma = 2.0,\ \lambda = \exp -9$

Figure 4.7: Empirical distribution (dots) of import vectors (IVs) for Gaussian-distributed features in a four class scenario with overlapping classes. The 3-σ contour of $P(\boldsymbol{x} \mid \mathcal{C})$ is given in yellow. The decision boundaries (black) estimated by IVM and the isolines of the 25 % and the 75 % posterior probabilities (violet) are given. Plot (a)-(f) show various combinations of different kernel and regularization parameters. A Gaussian radial basis function kernel is used.

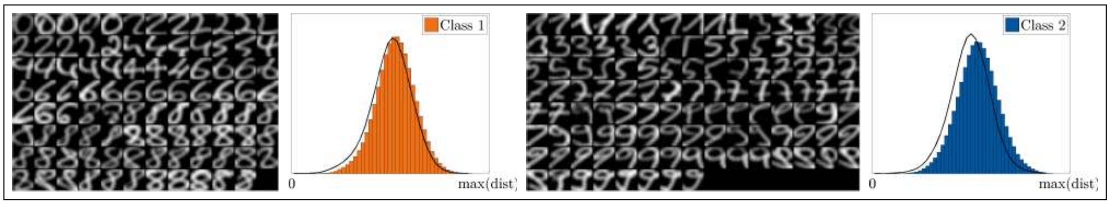


FIGURE 4.8: The import vectors illustrated as images and the histogram of the distances between them. Class 1 (left 2 plots) is comprised of the even images of digits 0, 2, 4, 6 and 8, and class 2 (right 2 plots) is comprised of the odd images of digits 1, 3, 5, 7 and 9. The selection of IVs from all subclasses verifies that the IVs cover the entire feature space.

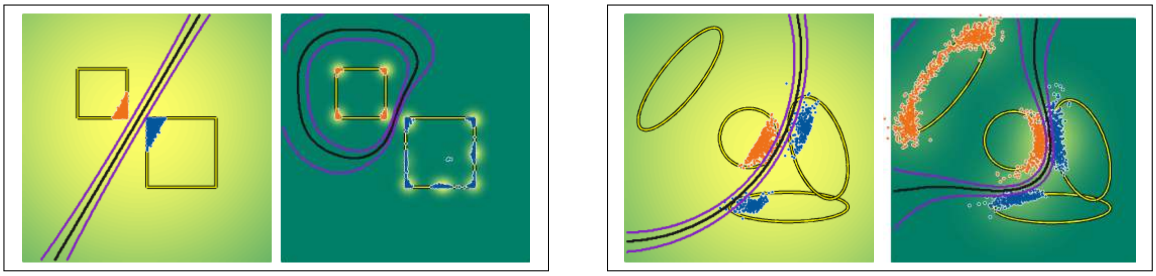


FIGURE 4.9: Empirical distribution (dots) of support vectors (SVs) for features in a two class scenario with well separated classes: Gaussian (left) and uniform (right) distribution of the features. The decision boundaries (black) estimated by SVM and the 25 % and the 75 % posterior probabilities (violet) obtained by Platt's method Platt et al. (2000) are given. We use $\sigma = 0.25$ for the kernel parameter and $\lambda = \exp(2)$ for the respective left plot and $\lambda = \exp(0)$ for the respective right plot.

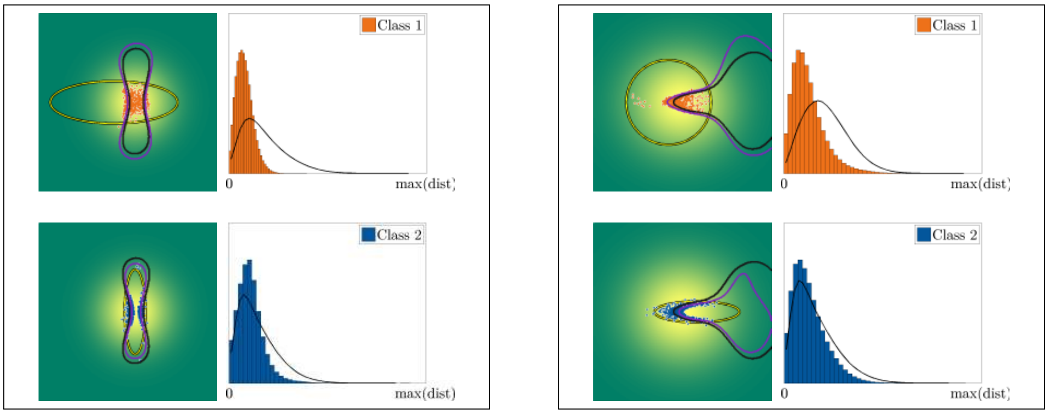


FIGURE 4.10: Empirical distribution (dots) of support vectors (SVs) for Gaussian-distributed features in a two class scenario with overlapping classes. The 3-σ contour of $P(\boldsymbol{x} \mid \mathcal{C})$ is given in yellow, the 75 % contours of the posterior $P(\mathcal{C} \mid \boldsymbol{x})$ obtained by Platt's method Platt et al. (2000) are given in violet for class c_k resp. Most of the SVs lie near to the decision boundary.

Chapter 5

Incremental Learning with Import Vector Machines

If the data samples become available sequentially, it is reasonable and more efficient to update the IVM incrementally, rather than recomputing them from scratch. In this chapter we introduce the I^{2}VM learner. We describe how to deal with new training samples becoming available at the current time step, how to remove training samples from older time steps and how to update the set of import vectors. We further show the steps to incorporate new classes and features. We will describe the learning procedure for the two class situation, dropping the index c for a simplified notation.

5.1 Conceptual Procedure for an Incremental Update Step

In this section we introduce the conceptual procedure for one incremental update step. We distinguish between three update steps that serve a different purpose.

(1) The addition of training vectors allows the learner to add new samples to already existing ones. As a result, the learner is able to adapt to new class-specific samples or enhance or support existing ones.

(2) The removal of training vectors is necessary to maintain the learner of bounded size. Depending on the application non-informative, counter-supportive or the oldest data samples are removed.

(3) The set of import vectors has to be adapted, like in classical IVM. This includes the adding of the new IVs from the new training samples to adapt to the current samples and the removal of some import vectors, which do not represent the best subset any more.

After introducing the internal representation of the learner in the next section, we explain the update steps specified in this section in detail in Section 5.3.

5.2 Representing the Current State of the Learner

In this section we describe the current *state of the learner* by means of the model components in order to define the internal representation of the classifier.

The state of the learner is represented by two sets and additional matrices and vectors, which will change over time:

- A training set $\mathcal{T} = \{(\boldsymbol{x}_n, t_n)\}, n = 1, \ldots, N$ being a subset of all training samples seen so far.
- An import vector set $\mathcal{V} = \{\boldsymbol{x}_v, t_v\}, v = 1, \ldots, V$ being a subset of $\mathcal{T}$. It is used for defining the classifier model and implicitly specifies the decision boundaries.

The following vectors and matrices support efficient learning, and are partially redundant.

- the $(N \times V)$-kernel matrix $\mathsf{K} = [\boldsymbol{k}_v]$,
- the V-vector $\boldsymbol{\alpha}$ of parameters,
- the N-vector $\boldsymbol{p}$ of probabilities, and for efficiency the $N \times N$-matrix R,
- the $(V \times V)$-matrix $\mathsf{H} = \nabla^2 Q\ (\boldsymbol{\alpha})$ being the Hessian of the optimization function and, for efficiency, its inverse H^{-1}, and
- the V-vector $\boldsymbol{z}$.

Thus the state is represented by

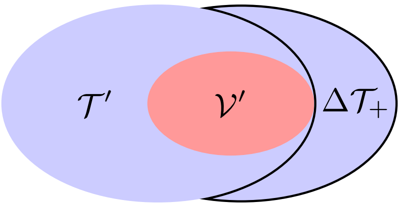
$$S = \{\mathcal{T}, \mathcal{V}; \mathsf{K}, \boldsymbol{\alpha}, \boldsymbol{p}, \mathsf{R}, \mathsf{H}, \mathsf{H}^{-1}, \boldsymbol{z}\}\,. \tag{5.1}$$

When working in the incremental setting, we indicate the current time step with one prime and next time step with a double prime. The learning procedure is a Markov chain in which a new state S'' only depends on the previous state S', and the set of training data $\Delta\mathcal{T}_+$ becomes available between state S' and S'' . We defined the state of the learner at a time step t in order to specify the internal representation of I^{2}VM. In the next section we introduce the learning procedure for I^{2}VM.

5.3 Learning Procedure for Incremental Import Vector Machines

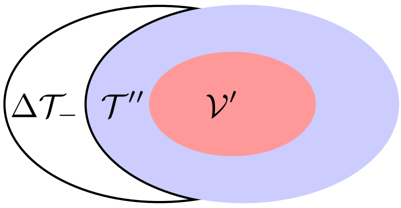
In this section we describe the update steps specified in Section 5.1 in detail. We have the following three update steps:

1. Adding training vectors: Extend the old set of training vectors $\mathcal{T}'$ to $\mathcal{T}_+$ by including new vectors $\Delta\mathcal{T}_+ = \Delta\mathcal{T}''$:



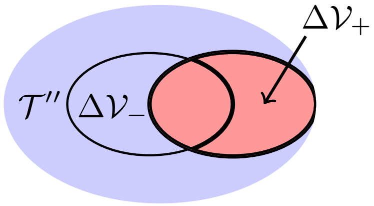
$$\mathcal{T}_+ = \mathcal{T}' \cup \Delta\mathcal{T}_+ \tag{5.2}$$

2. Remove training vectors $\Delta\mathcal{T}_-$, but not if they are currently included in the set of import vectors $\mathcal{V}'$:



$$\mathcal{T}'' = \mathcal{T}_+ \setminus \Delta\mathcal{T}_- \tag{5.3}$$

3. Incremental and decremental IV-learning: Add import vectors $\Delta\mathcal{V}_+ \subset (\mathcal{T}'' \setminus \mathcal{V}')$, and remove import vectors $\Delta\mathcal{V}_- \subset \mathcal{V}'$ with a hybrid greedy forward/backward selection as described in Algorithm 1 in Section 3.3 until the objective function $\mathcal{Q}$ converges:



$$\mathcal{V}'' = \left(\mathcal{V}' \setminus \Delta\mathcal{V}_-\right) \cup \Delta\mathcal{V}_+ \tag{5.4}$$

The criteria for deleting training vectors depend on the application. In tracking applications, training vectors can be removed according their age, or if older class-specific features are contradictory to newer ones. In contrary to this, for the incremental learning of large data sets the sequence of the incoming training vectors should not influence the result. In this case, a suitable criteria is to remove the training vectors depending on their contribution to the model. We discuss several criteria in Section 5.6.

There are also several ways to realize step 3. One possibility is to test each data sample in the current training set to be an import vector. Another possibility, which more likely corresponds to incremental learning, is only to test the newly acquired data samples. We choose the latter possibility, which is more efficient than the first one. In the next section the three update steps are now made explicit.

5.4 Incremental and Decremental Update

In the following we describe the incremental and decremental update of training and import vectors for the two-class case. For convenience we show the update of import vectors only for one single import vector. The update can be extended to simultaneously or successively add or remove groups of import vectors.

5.4.1 Adding and Removing Training Vectors

In the following we describe the addition and removal of one or more training samples.

5.4.1.1 Adding Training Vectors

We add ΔN_+ training vectors ΔX_+ with targets $\Delta\boldsymbol{t}_+$ so that $N_+ := N' + \Delta N_+$. The kernel submatrix of the new training vectors is given by $\Delta\mathsf{K}_+$. We extend the kernel matrix and the target vector to

$$\underset{N_+\times V}{\mathsf{K}_+} = \begin{bmatrix} \mathsf{K}' \\ \Delta\mathsf{K}_+ \end{bmatrix}, \tag{5.5}$$

$$\underset{N_+\times 1}{\boldsymbol{t}_+} = \begin{bmatrix} \boldsymbol{t}' \\ \Delta\boldsymbol{t}_+ \end{bmatrix}. \tag{5.6}$$

Using the current parameters $\boldsymbol{\alpha}'$ we first obtain the posteriors $\Delta\boldsymbol{p}_+$ from (3.30), $\Delta\mathsf{K}_+$ from (3.12), $\Delta\mathsf{R}_+$ from (3.25) using $\Delta\boldsymbol{p}_+$ and $\Delta\boldsymbol{z}_+$ from (3.32).

Starting from the Hessian

$$\mathsf{H}' = \frac{1}{N'}\mathsf{K}'^{\mathsf{T}}\mathsf{R}'\mathsf{K}' + \lambda\mathsf{K}'_R \tag{5.7}$$

and its inverse from the previous iteration we update these matrices and the parameters $\boldsymbol{\alpha}'$ using the IRLS procedure, yielding

$$\boldsymbol{\alpha}_+ = H_+^{-1}\left(K'^{\mathsf{T}} R' \boldsymbol{z}' + \Delta K_+^{\mathsf{T}} \Delta R_+ \Delta \boldsymbol{z}_+\right) \tag{5.8}$$

with

$$\begin{aligned}
H_+ &= \frac{1}{N_+} K_+^{\mathsf{T}} \begin{bmatrix} R' & \\ & \Delta R_+ \end{bmatrix} K_+ + \lambda K'_R \,, \\
&= H' + \Delta H_+ \,, \\
&= \frac{N'}{N_+}\left(H' + \frac{1}{N'} \Delta K_+^{\mathsf{T}} \Delta R_+ \Delta K_+ + \lambda \frac{\Delta N_+}{N'} K_R\right) \,, && (5.9) \\
\Delta \boldsymbol{z}_+ &= \Delta K_+ \boldsymbol{\alpha}' + \Delta R_+^{-1}\left(\Delta \boldsymbol{p}_+ - \Delta \boldsymbol{t}_+\right) \,. && (5.10)
\end{aligned}$$

With H' given from (5.7) we update and invert the Hessian with (5.9). Since the inverse H_+^{-1} is only of size $(V' \times V')$ and the number of import vectors is usually small, the inverse can be computed significantly faster than for non-sparse kernel logistic regression.

Applying $\boldsymbol{\alpha}_+$ we finally determine $\boldsymbol{p}_+$ from (3.30), R_+ from (3.25), and finally $\boldsymbol{z}_+$ from (3.32), yielding the state $S_+ = \{\mathcal{T}_+, \mathcal{V}'; K_+, \boldsymbol{\alpha}_+, \boldsymbol{p}_+, R_+, H_+, H_+^{-1}, \boldsymbol{z}_+\}$.

5.4.1.2 Removing Training Vectors

We delete I training vectors with indices $\mathcal{I}$ and obtain the partitioning of the matrix K_+

$$K_+ = \begin{bmatrix} K_- \\ \Delta K_- \end{bmatrix} \text{ with } \Delta K_- = \Psi_{\mathcal{I}}\left(K_+\right) \,, \tag{5.11}$$

the target vector $\boldsymbol{t}_+$

$$\boldsymbol{t}_+ = \begin{bmatrix} \boldsymbol{t}_- \\ \Delta \boldsymbol{t}_- \end{bmatrix} \text{ with } \Delta \boldsymbol{t}_- = \Psi_{\mathcal{I}}\left(\boldsymbol{t}_+\right) \,, \tag{5.12}$$

and the deleted parts of R_+ and $\boldsymbol{z}_+$

$$\begin{aligned}
\Delta R_- &= \Psi_{\mathcal{I}}\left(\Psi_{\mathcal{I}}\left(R_+\right)^{\mathsf{T}}\right) \,, && (5.13) \\
\Delta \boldsymbol{z}_- &= \Psi_{\mathcal{I}}\left(\boldsymbol{z}_+\right) \,. && (5.14)
\end{aligned}$$

The update (5.8) can be formulated for a decreasing number of training vectors $N'' := N_+ - \Delta N_-$ in a similar manner, using an efficient determination of H_- as in (5.9):

$$\boldsymbol{\alpha}_- = H_-^{-1}\left({K_+}^{\mathsf{T}} R_+ \boldsymbol{z}_+ - \Delta K_-^{\mathsf{T}} \Delta R_- \Delta \boldsymbol{z}_-\right) \tag{5.15}$$

with

$$\begin{aligned}
H_- &= \frac{1}{N''} K_-^{\mathsf{T}} \left(\Psi_{\mathcal{T}''\setminus\mathcal{I}}\left(\Psi_{\mathcal{T}''\setminus\mathcal{I}}\left(R_+\right)^{\mathsf{T}}\right)\right) K_- + \lambda K'_R \,, \\
&= H_+ - \Delta H_- \,, \\
&= \frac{N_+}{N''}\left(H_+ - \frac{1}{N_+} \Delta K_+^{\mathsf{T}} \Delta R_+ \Delta K_+ - \lambda \frac{\Delta N_-}{N_+} K_R\right) && (5.16)
\end{aligned}$$

and the Hessian H_+ from (5.9). Applying $\boldsymbol{\alpha}_-$ we again obtain $\boldsymbol{p}_-$, R_-, and $\boldsymbol{z}_-$ using (3.30), (3.25) and (3.32), yielding the state $S_- = \{\mathcal{T}'', \mathcal{V}'; K_-, \boldsymbol{\alpha}_-, \boldsymbol{p}_-, R_-, H_-, H_-^{-1}, \boldsymbol{z}_-\}$.

5.4.2 Adding and Removing Import Vectors

In the following we show how to add and remove import vectors. For convenience we only show the addition and the removal of one import vector, in each case given the state S_-. In general an update step consists of several insertions and removals of import vectors, depending on how much the model must be adapted.

5.4.2.1 Adding Import Vectors

To add an import vector $\{\boldsymbol{x}^{\text{add}}, y^{\text{add}}\} \in \mathcal{T}''$ out of the the current set of N'' training vectors, we define a kernel vector $\Delta\widetilde{\boldsymbol{k}}_+ = \left[k\left(\boldsymbol{x}_n, \boldsymbol{x}^{\text{add}}\right)\right]$, a kernel vector $\Delta\widetilde{\boldsymbol{k}}_{R,+} = \left[k\left(\boldsymbol{x}_v, \boldsymbol{x}^{\text{add}}\right)\right]$ and a kernel scalar $\Delta\widetilde{k}_{R,+} = k\left(\boldsymbol{x}^{\text{add}}, \boldsymbol{x}^{\text{add}}\right)$. We extend the kernel matrix and the regularization matrix to

$$\widetilde{\mathsf{K}}_+ = \left[\begin{array}{cc} \mathsf{K}_- & \Delta\widetilde{\boldsymbol{k}}_+ \end{array}\right], \tag{5.17}$$

$$\widetilde{\mathsf{K}}_{R,+} = \left[\begin{array}{cc} \mathsf{K}'_R & \Delta\widetilde{\boldsymbol{k}}_{R,+} \\ \Delta\widetilde{\boldsymbol{k}}^{\mathsf{T}}_{R,+} & \Delta\widetilde{k}_{R,+} \end{array}\right], \tag{5.18}$$

leading to $V_+ := V' + 1$ import vectors. The parameters $\widetilde{\boldsymbol{\alpha}}_+$ are obtained from

$$\widetilde{\boldsymbol{\alpha}}_+ = \widetilde{\mathsf{H}}_+^{-1} \widetilde{\mathsf{K}}_+^{\mathsf{T}} \mathsf{R}_- \boldsymbol{z}_- \,. \tag{5.19}$$

We use the Sherman-Morrison-Woodbury (SMW) formula Higham (2002) to efficiently compute the inverse in (5.19) with

$$\begin{aligned} \widetilde{\mathsf{H}}_+^{-1} &= \left(\frac{1}{N''}\widetilde{\mathsf{K}}_+^{\mathsf{T}}\mathsf{R}_-\widetilde{\mathsf{K}}_+ + \lambda\widetilde{\mathsf{K}}_{R,+}\right)^{-1}, \\ &= \left[\begin{array}{cc} \mathsf{H}_- & \boldsymbol{a} \\ \boldsymbol{a}^{\mathsf{T}} & b \end{array}\right]^{-1} = \left[\begin{array}{cc} \mathsf{H}_-^{-1} + \left(\mathsf{H}_-^{-1}\boldsymbol{a}\right)e^{-1}\left(\boldsymbol{a}^{\mathsf{T}}\mathsf{H}_-^{-1}\right) & -\mathsf{H}_-^{-1}\boldsymbol{a}e^{-1} \\ -e^{-1}\boldsymbol{a}^{\mathsf{T}}\mathsf{H}_-^{-1} & e^{-1} \end{array}\right], \end{aligned} \tag{5.20}$$

where

$$\boldsymbol{a} = \frac{1}{N''}\widetilde{\mathsf{K}}_+^{\mathsf{T}}\mathsf{R}_-\Delta\widetilde{\boldsymbol{k}}_+ + \lambda\Delta\widetilde{\boldsymbol{k}}_{R,+}\,, \tag{5.21}$$

$$b = \frac{1}{N''}\Delta\widetilde{\boldsymbol{k}}_+^{\mathsf{T}}\mathsf{R}_-\Delta\widetilde{\boldsymbol{k}}_+ + \lambda\Delta\widetilde{k}_{R,+}\,, \tag{5.22}$$

$$e = b - \boldsymbol{a}^{\mathsf{T}}\mathsf{H}_-^{-1}\boldsymbol{a}. \tag{5.23}$$

Note that the determination of the inverse $\widetilde{\mathsf{H}}_+^{-1}$ only incorporates an inverse e^{-1} of size (1×1), since the inverse H_-^{-1} is already given from the last step. Using $\widetilde{\boldsymbol{\alpha}}_+$ we obtain $\widetilde{\boldsymbol{p}}_+$, $\widetilde{\mathsf{R}}_+$, and $\widetilde{\boldsymbol{z}}_+$, yielding the state $\widetilde{S}_+ = \{\mathcal{T}'', \mathcal{V}_+; \widetilde{\mathsf{K}}_+, \widetilde{\boldsymbol{\alpha}}_+, \widetilde{\boldsymbol{p}}_+, \widetilde{\mathsf{R}}_+, \widetilde{\mathsf{H}}_+, \widetilde{\mathsf{H}}_+^{-1}, \widetilde{\boldsymbol{z}}_+\}$. Further import vectors can be added by extending the kernel and the regularization matrix again and using the updated entities of $\widetilde{S}_+$.

5.4.2.2 Removing Import Vectors

Using the set $\mathcal{V}' = \{1, \ldots, i, \ldots, V'\}$ of indices of the current import vectors a removal of an import vector $\{\boldsymbol{x}_i, y_i\}$ reduces the kernel and regularization matrix and yields

$$\widetilde{K}_- = \left(\Psi_{\mathcal{V}'\backslash i}\left(K_-^{\mathsf{T}}\right)\right)^{\mathsf{T}} , \tag{5.24}$$

$$\Delta\widetilde{K}_- = \left(\Psi_i\left(K_-^{\mathsf{T}}\right)\right)^{\mathsf{T}} , \tag{5.25}$$

$$\widetilde{K}_{R,-} = \Psi_{\mathcal{V}'\backslash i}\left(\Psi_{\mathcal{V}'\backslash i}\left(K'_R\right)^{\mathsf{T}}\right) , \tag{5.26}$$

$$\Delta\widetilde{k}^{\mathsf{T}}_{R,-} = \Psi_i\left(\Psi_i\left(K'_R\right)^{\mathsf{T}}\right) , \tag{5.27}$$

the consequence being that the Hessian H_- is reduced to

$$\widetilde{H}_- = \Psi_{\mathcal{V}'\backslash i}\left(\Psi_{\mathcal{V}'\backslash i}\left(H_-\right)^{\mathsf{T}}\right) .$$

We can efficiently compute the inverse Hessian $\widetilde{H}_-^{-1}$ by updating H_-^{-1}, given from the last step, with

$$\widetilde{H}_-^{-1} = H_-^{-1} - h_{ii}\boldsymbol{h}_i^{\mathsf{T}}\boldsymbol{h}_i, \tag{5.28}$$

whereby $\boldsymbol{h}_i$ is the ith row from the Hessian H_-. Both, the i-th row and the i-th column of $\widetilde{H}_-^{-1}$ become zero and must be removed to compute the updated parameters with

$$\widetilde{\boldsymbol{\alpha}}_- = \widetilde{H}^{-1}\widetilde{K}_-^{\mathsf{T}} R_- \boldsymbol{z}_-. \tag{5.29}$$

Applying $\widetilde{\boldsymbol{\alpha}}_-$ we obtain $\widetilde{\boldsymbol{p}}_-$, $\widetilde{R}_-^{\mathsf{T}}$, and $\widetilde{\boldsymbol{z}}_-$. Further import vectors can be removed in the same way by a further reduction of the kernel matrix and the Hessian.

Depending on the number of insertions and/or deletions of training and import vectors we finally have the new state S'', consisting of the new sets $\mathcal{T}''$ and $\mathcal{V}''$, the kernel matrix K'', the parameters $\boldsymbol{\alpha}''$, the probabilities $\boldsymbol{p}''$ and R'', the Hessian H'', its inverse H''^{-1}, and the corrections $\boldsymbol{z}''$, which will be updated in the next iteration.

We have described the incremental an decremental update steps for training samples and import vectors. In the next section we show explain the steps which are necessary to add further classes and features, which may be introduced with new samples.

5.5 Incorporation of New Classes and Features

In the last section we have formulated the incremental and decremental update of training and import vectors. We specify the update for the two-class case with an unchanged number of classes and features, not to be confused with feature vectors. Although the incorporation of further classes and features is relevant to incremental learning strategies, there is only little work in the literature which deals with this problem, see Section 2.1. Both, new classes and features may be introduced with new samples. In the following sections we show that the I^{2}VM classifier is able to add new classes as well as new features.

5.5.1 Incorporating New Classes

In some cases the number of classes is not know beforehand and a re-training is undesirable. In this section we show the incorporation of additional classes for the I^{2}VM classifier.

Before the learning procedure starts, we add ΔC_+ classes so that $C'' = C' + \Delta C_+$. Thus, the old set of classes $\mathcal{C}'$ is extended by $\Delta\mathcal{C}_+$, yielding the new set of classes $\mathcal{C}'' = \{1, \dots, c, \dots, C''\}$. The target vector $\boldsymbol{t}'$ is extended via the recursive update

$$\underset{N' \times C''}{\boldsymbol{t}'} := \left[\begin{array}{cc} \boldsymbol{t}' & \mathbf{0}_{N' \times \Delta C_+} \end{array} \right] . \tag{5.30}$$

We further extend the parameter vector so that $\boldsymbol{\alpha}'_{c'}$ for each $c' \in \Delta\mathcal{C}_+$, such that

$$\underset{V' \times 1}{\boldsymbol{\alpha}'_{c'}} = \boldsymbol{\epsilon}_{V' \times 1}, \tag{5.31}$$

where $\boldsymbol{\epsilon}$ is a vector with small values preventing a division by zero when the softmax function is evaluated. After this, the incremental and decremental update steps can be performed as introduced in Section 5.4.

5.5.2 Incorporating New Features

In this section we show the incorporation of additional features, sometimes also referred to as attributes, for the I^{2}VM algorithm. When dealing with kernel-based classifiers, an incorporation of features only changes the current stored training and import vector set. Because the kernel only depends on relations between the samples rather than the samples themselves, an incorporation of features only affects the original feature vectors but not the kernel vectors. Before the learning procedure starts, the original feature vectors $\boldsymbol{x}_n \in \mathcal{T}'$ and $\boldsymbol{x}_v \in \mathcal{V}'$ are extended by ΔM_+ features so that $M'' = M' + \Delta M_+$:

$$\underset{M'' \times 1}{\boldsymbol{x}'_n} := \left[\begin{array}{c} \boldsymbol{x}'_n \\ \mathbf{0}_{\Delta M_+ \times 1} \end{array} \right] , \tag{5.32}$$

$$\underset{M'' \times 1}{\boldsymbol{x}'_v} := \left[\begin{array}{c} \boldsymbol{x}'_v \\ \mathbf{0}_{\Delta M_+ \times 1} \end{array} \right] . \tag{5.33}$$

Note that for the considered kernel classes the extension does not affect the kernel matrix K', since the additional feature dimensions are all set to zero. After performing (5.32) and (5.33), new samples with M'' features can be added as described in Section 5.4.

So far, we described the incorporation of further classes and features, which may be introduced with new samples. In the next section we consider different criteria to identify training samples, which are meant to be removed.

5.6 Criteria for Removing Training Vectors

In this section we introduce various criteria for the removal of training vectors. The removal of data samples is necessary in order to keep the model efficient and to keep the set of samples representative. With a suitable criteria *non-representative* training samples are removed, leading to an adapted internal representation of the classifier which models the dependency of the labels $\boldsymbol{y}_{\mathcal{U}_t}$ to given feature vectors $\mathsf{X}_{\mathcal{U}_t}$ at time step t as well as possible. Non-representative samples are both not necessary for discrimination and reconstruction. In the following we give an overview about criteria for removing training samples.

5.6.1 Overview

The most naive way for sparse incremental learning methods is to remove all samples, whose assigned parameters in the current classifier model are zero. For example Syed et al. (1999)

and Rüping (2001) train the SVM classifier on the newly acquired training samples and on the previous learned support vectors. When using SVM this reason seems intuitive because the training on all samples yields the same result as training on selected support vectors only. Nevertheless, as Cauwenberghs and Poggio (2001) stated this gives only approximate results and also training samples could be kept in the model.

In various applications it is desirable to remove training vectors depending on their age. In this case the algorithm focus more on recently acquired training samples than on older ones. This is quite useful in tracking applications when the appearance of an object changes and the recent observations are more likely to represent the current characteristics of the object than would more distant ones.

One possibility is remove all samples which are older than a pre-defined number of time steps. In this case a hard weight of 1 or 0 is assigned to an sample. Another possibility is to additionally down-weight the samples depending on their age. This approach assumes that the sequence of the data has a meaning and therefore cannot be applied to applications where the arrangement of the data is irrelevant. The other way around, Rüping (2001) increase the weight for old data samples to deal with concept-drifts.

Another intuitive way is to randomly remove training samples. This approach works well if the distribution of the data does not change and new training samples are randomly acquired from their underlying distribution. However, concept-drifts in the distribution will lead to an imbalanced removal of samples, i.e. early seen parts of the distribution are more likely to be under-represented than later seen ones.

In the following we introduce five different criteria that compute the influence of training vectors, in which influential means that they are necessary to represent the model. Using these criteria non-influential training vectors can be removed to keep the classifier model efficient.

5.6.2 Cross-entropy

The first criterion referred to as cross-entropy computes a distance between the target vector and the estimated probability. The cross-entropy is given by

$$b_{n,\mathrm{ce}} = -\boldsymbol{t}_n^{\mathsf{T}} \log \boldsymbol{p}_n. \tag{5.34}$$

This measure is part of the *cross-entropy* loss function, which is discussed in Section 3.2.5.3. A small value means a small contribution to the objective function. Thus, all samples that have a small influence onto the estimation of the model parameters are removed. This method tries to remain as much discriminative power as possible by keeping samples near to the decision boundary, but also samples in areas with a low posterior probability, e.g. the boundary of dense regions.

5.6.3 Linear Independence

Nguyen-Tuong and Peters (2010) introduced a measure for *linear independence.* The measure identifies samples that can be approximated by a linear combination of existing samples in the training set. They define that a sample $\boldsymbol{x}_n$ that cannot be explained be other samples has a high linear independence and a sample $\boldsymbol{x}_n$ that can be approximated well using other samples has a low linear independence and can be removed. Following Nguyen-Tuong and

Peters (2010) we introduce the independence measure as

$$b_{m,\text{ind}} = \left\| \sum_{n \backslash m} a_n \boldsymbol{x}_n - \boldsymbol{x}_m \right\|^2 , \tag{5.35}$$

$$= \sum_{n',n' \backslash m} a_n a_{n'} \boldsymbol{x}_n^{\mathsf{T}} \boldsymbol{x}_n - 2 \sum_{n'} a_{n'} \boldsymbol{x}_n^{\mathsf{T}} \boldsymbol{x}_m + \boldsymbol{x}_m^{\mathsf{T}} \boldsymbol{x}_m . \tag{5.36}$$

The measure can be seen as the reconstruction error of the kernel vector, where the parameters $\boldsymbol{a}$ denote the coefficients of linear dependence. We replace the dot product of $\boldsymbol{x}_n^{\mathsf{T}} \boldsymbol{x}_{n'}$ by the kernel $k_{nn'}$ yielding

$$b_{m,\text{ind}} = \boldsymbol{a}^{\mathsf{T}} \widetilde{K}_m \boldsymbol{a} - 2 \boldsymbol{a}^{\mathsf{T}} \boldsymbol{k}_m + k_{mm} , \tag{5.37}$$

where $\widetilde{K}_m = \left(\Psi_m \left(K^{-1} \right)^{\mathsf{T}} \right)^{\mathsf{T}}$, $\boldsymbol{k}_m$ is the m-th row of the kernel matrix K and k_{mm} the m-th diagonal element of the kernel matrix. The optimal parameter vector $\boldsymbol{a}$ can be determined by minimizing $b_{m,\text{ind}}$ obtaining $\boldsymbol{a} = \widetilde{K}^{-1} \boldsymbol{k}_m$. Thus, the independence measure is

$$b_{m,\text{ind}} = k_{mm} - \boldsymbol{k}_m \boldsymbol{a}. \tag{5.38}$$

The measure only depends on the kernel and is therefore independent of the current classifier model. This method tries to remain as much reconstructive power as possible. The larger the value of $b_{n,\text{ind}}$, the more independent is $\boldsymbol{x}_n$ from the current training set.

5.6.4 Cook's Distance

The *Cook distance* criteria identifies influential training vectors by means of the discriminative power. In order to define the criteria, we make use of linear regression diagnostics, which can be transferred to logistic regression diagnostics (Hosmer and Lemeshow, 2000). The latter one can be directly applied to IVM. The hat matrix N is given by

$$N = R^{\frac{1}{2}} K \left(K^{\mathsf{T}} R K \right)^{-1} K^{\mathsf{T}} R^{\frac{1}{2}} . \tag{5.39}$$

The diagonal elements of the hat matrix are the leverage values

$$\boldsymbol{l} = \operatorname{diag}(N) . \tag{5.40}$$

To compute the influence, which has a data sample onto the result, we use an analogous measure to the Cook's distance (Cook and Weisberg, 1982) for linear regression. The distance $b_{n,\text{Cook}}$ is given by

$$b_{n,\text{Cook}} = \frac{r_n^2 l_n}{(1 - l_n)^2} \tag{5.41}$$

with

$$r_n = \frac{(t_n - p_n)}{\sqrt{p_n (1 - p_n)}} \tag{5.42}$$

defined as the Pearson residual. As in linear regression is the sum of all leverage values $\sum_n l_n = V$ is equal to the number of parameters. Contrary to the interpretation of the leverage values in linear regression, where the leverage values increases with the distance from the mean, in logistic regression extreme data samples in the feature space may not have a high leverage value. (Hosmer and Lemeshow, 2000) shows that data samples with small and large probabilities $0.1 < p_n < 0.9$ tend to have a small leverage value, whereby samples with probabilities in between have leverage values, which can be interpreted as some kind of distance from the mean. Because we interpret import vectors, which have a large probability, as data samples with high influence onto the result, we use the *modified Cook distance* resulting from the usage of a modified version of (5.40),

$$\boldsymbol{l}_{\text{mod}} = \text{diag}\left(\mathsf{K}\left(\mathsf{K}^{\mathsf{T}}\mathsf{R}\mathsf{K}\right)^{-1}\mathsf{K}^{\mathsf{T}}\right). \tag{5.43}$$

The modified distance $b_{n,\text{CookMod}}$ is than given by

$$b_{n,\text{CookMod}} = \frac{r_n^2 l_{n,\text{mod}}}{(1 - l_{n,\text{mod}})^2}. \tag{5.44}$$

The subset of remaining training samples tries to remain as much discriminative power as possible.

5.6.5 Weighted Sampling

This criterion uses a weighted random sampling strategy, in which the weights are derived from the density of the samples using the kernel matrix. As stated in Chapter 4, the distribution of the IVs represent the coverage of the training samples rather than their density. In order to approximate the density of the samples, we use the non-parametric density estimation method with histograms. The V bins of the histogram are the cells in a Voronoi diagram defined on the set of the IVs $\mathcal{V}$. The heights of the bins are given by the number of samples N_v included. The distance from a training sample to each IV can directly be obtained from the kernel matrix. For example, using the Gaussian radial basis function kernel an entry of 1 means the sample is identical to the IV and a very small value indicate that the sample is far away from the IV. In this case, we first determine the index j_n of the maximum value in each row of the kernel matrix $\boldsymbol{k}_n$,

$$j_n = \text{argmax}_v\, \boldsymbol{k}_n. \tag{5.45}$$

Further, we introduce an indicated $(N \times V)$-matrix K_I, in which each row is a unit vector with all elements zero except element j_n. The $(V \times 1)$-histogram $\boldsymbol{h}$ of $\boldsymbol{f}$ is derived by

$$h_v = \sum_n \delta(j_n, v). \tag{5.46}$$

Accordingly, the unnormalized weight for each kernel vector $\boldsymbol{k}_n$ is

$$\widetilde{w}_v = \mathsf{K}_I \boldsymbol{h} \tag{5.47}$$

and the normalized weights used for the sampling strategy are given by

$$w_v = \frac{1}{\sum_v \widetilde{w}_v}\widetilde{w}_v. \tag{5.48}$$

Using this method ensures that dense regions are thinned out and sparsely covered regions are prevented from disappearing. As well as the independence measure, this method tries to remain as much reconstructive power as possible.

5.6.6 Increase of the Negative Log-likelihood Function

The last criterion operates similar to the greedy backward selection of import vectors. Each sample is tested to be removed from the training set and the negative log-likelihood function with the new set is computed. The sample that least increases the objective function value is removed from the set. The value can be negative, what means that the removal of the sample currently improves the classifier model. This criterion differs from all other, because the model needs to be recomputed several times for each removal. All other criteria determine the samples to be removed by only using the current model.

In this chapter we have introduced the I^{2}VM classifier. We explained the learning scheme and the update steps in order to add and remove training samples as well as to adapt the set of IVs. Further we introduced various criteria that remove non-representative training samples. We will use these criteria in our experiments when we deal with large image databases in which not all samples can be stored. In this case, a suitable criteria will lead to a comparable performance as been reached with the usage of all samples. In the next chapter, we show the applicability of I^{2}VM for various applications.

Chapter 6

Applications

In the following section the potential of I^{2}VM is analyzed in several applications. The experiments are part of our work in (Roscher et al., 2012), (Roscher et al., 2012) and (Roscher et al., 2012). We use the experiments in order to experimentally verify that I^{2}VM have the properties mentioned in Section 1.3, which are necessary to be a powerful incremental classifier. Remember, the properties are a high performance independently of the ordering of the samples and comparably to its batch counterpart, a high discriminative power, the ability to deal with arbitrary long sequences and the applicability of the probabilistic output.

In our first experiment we compare the performance of I^{2}VM by means of the accuracy to other, well-known incremental learners using machine learning benchmark data sets. We will also compare I^{2}VM to its batch counterpart. We will further analyze the influence of ordering effects onto the performance of the learners. Due to the discriminative power of I^{2}VM we expect similar results as powerful classifiers such as SVM. Furthermore, due to the reconstructive power, we expect that the classifier can cope with concept-drifts.

In our second experiment we show that I^{2}VM can deal with long sequences of sequentially encountered samples. We evaluate our introduced criteria from Section 5.6 to remove samples in order to keep the model efficient. For this purpose, we use I^{2}VM for semantic segmentation of images from the Microsoft Research Cambridge Object Recognition Image database. Starting from a coarse classifier model, we will refine the model by sequentially learning from newly arriving images. We expect that I^{2}VM can deal with long sequences and a criterion can be found that can remove training samples without a loss in performance.

In our third experiment we show that I^{2}VM is able to adapt to changes in the distribution of the training samples in order to classify a current set of test samples as well as possible. New training samples are acquired using the probabilistic output of the I^{2}VM. We apply self-training to classify the landcover of a large area consisting of composite remote sensing images, which were acquired by a Landsat satellite. The scenes are characterized by both spatial and temporal differences. We expect that I^{2}VM can adapt to changes in the distribution of the training samples and the probabilities are useful to acquire new samples.

In our forth experiment we show that I^{2}VM is able to simultaneously cope with all challenges that appear when dealing with sequential data. We evaluate the algorithm for object tracking in image sequences. Image sequences are arbitrarily long data streams that a characterized by concept-drifts. The distinction of object and background requires a high discriminative power. We use the algorithm within a tracking framework using the tracking-by-segmentation approach. We use different benchmark image sequences, which are challenging due to moving objects, changing illumination, varying object appearance, size and shape, changing or moving background, and/or occlusions. We expect that I^{2}VM turns out to be an adaptive classifier that can deal with arbitrary long data streams. We further expect that

I^{2}VM is suitable to be integrated into a tracking framework showing at least similar results as batch tracking methods.

6.1 Classification of Benchmark Data Sets

6.1.1 Comparison of I^{2}VM to IVM and Recent Incremental Classifiers

In this experiment we evaluate the performance of I^{2}VM by means of the accuracy. In order to do this, we show that the incremental method is competitive to a) its batch counterpart IVM and b) to other incremental learning algorithms, by using benchmark data sets. Furthermore, we quantitatively and visually analyze the influence of the concept-drifts onto the performance of I^{2}VM. Due to the discriminative power of I^{2}VM we expect a similar accuracy as can be reached by SVM. We expect that classifiers that also have a reconstructive model component perform better than classifiers that only have a discriminative model component.

6.1.2 Data

Machine learning benchmark data sets with static distribution. We use well-known machine learning data sets, including the Digits data set (Seewald, 2005), consisting of images of digits from 0 to 9, as well as the data sets DNA, USPS, SatImage and Vowel from the LibSVM repository (Chang and Lin, 2001). For the Digits data set we compute HoG features of each image and use them for classification purposes. For the Digits (2) data set, class 1 is composed of the images of digits 0, 2, 4, 6 and 8 and class 2 is composed of the images of digits 1, 3, 5, 7 and 9. The characteristics of the data sets are collected in Table 6.1.

Table 6.1: Characteristics of the used benchmark data sets.

Data set	#Train	#Test	#Classes	#Features
Digits	1900	1800	10	612
Digits (2)	1900	1800	2	612
DNA	1400	1186	3	180
USPS	7291	2007	10	256
SatImage	3104	2000	6	36
Vowel	528	462	11	10

Machine learning benchmark data sets with concept-drift. In order to simulate a concept-drift we sort the machine learning benchmark data sets. We sorted the training data according to their class membership, i.e. the first part of the sequence contains all data samples with class label $y_n < \frac{C}{2}$ and the second part contains all data samples with class label $y_n \geq= \frac{C}{2}$.

Synthetic data set with concept-drift. We further use synthetically generated data with two classes, of which one class is fixed and the other has a concept-drift. The fixed class is a Gaussian distribution placed in the center of the data set, see Figure 6.1. The drifting class is generated by adding Gaussian distributed data samples with drifted mean and changed variance, each time according to the sequence shown in the figure. We start with 25 samples per class simulated from two Gaussian distributions to train the classifier. Then the distribution of the second class changes for each time step. The test data is generated from the complete Gaussian mixture distribution.

6.1.3 Methods

We compare I^{2}VM to three multi-class classifiers: the online random forests (ORF) (Saffari et al. (2009)), incremental PCA/LDA (Uray et al. (2007)) and LaRank (Bordes et al. (2007)), an incremental SVM. Furthermore, batch IVM is used for comparison. For the ORF and the LaRank algorithm we use the software available online and for the incremental PCA/LDA, IVM and I^{2}VM we use our own implementation in Matlab and C++. The kernel and regularization parameter for LaRank and I^{2}VM are determined via 5-fold crossvalidation. For the ORF we report the results for 1 and 10 epochs, where every individual data sample is read once in each epoch. For the LaRank algorithm we only report the results for 1 epoch, because the implementation with Gaussian radial basis function kernel does not provide the usage of more epochs.

6.1.4 Experimental Setup

In our first experiment we analyze the performance of I^{2}VM on machine learning data sets without a removal of data samples. We conduct the experiments by providing each classifier the same (unsorted) sequence of training data. In our second experiment we use the sorted machine learning data sets to simulate a concept-drift within the data. We report the error rate in percentage of all incremental classifiers and IVM and the difference between the results obtained by the sorted and unsorted data sets. The error rate is the difference of the overall accuracy to 100%, see Section 3.4.2. In the last experiment we visually compare the results of IVM and I^{2}VM using the two-dimensional synthetic data set with concept-drift.

6.1.5 Results and Discussion

We summarize the results of IVM, I^{2}VM and other incremental classifiers using benchmark machine learning data sets with static distribution in Table 6.2. The classification results for the benchmark machine learning data sets with concept-drift are given in Table 6.3.

6.1.5.1 Static Data

Table 6.2: Classification error on common machine learning data sets. Bold numbers indicate the best result of an incremental learner. For comparison we also report the result of IVM in the last column.

Data set	Incremental					Batch
	Inc. PCA/LDA [%]	Online RF [%]		LaRank [%]	I^{2}VM [%]	IVM [%]
		1 epoch	10 epochs			
Digits	25.7	26.7	18.4	11.8	**10.0**	9.6
Digits (2 classes)	39.3	13.9	9.6	**6.4**	7.0	7.1
DNA	23.4	22.4	7.9	**5.6**	5.8	5.5
USPS	11.8	20.7	10.9	**4.3**	5.4	5.6
SatImage	13.6	15.7	11.5	**9.4**	9.6	9.6
Vowel	84.4	58.0	44.8	46.1	**42.6**	42.2
Synthetic	34.5	19.9	6.3	7.4	**5.3**	5.8

It can be seen that I^{2}VM is competitive to its batch version IVM. The largest difference between both classifiers is 0.4% on the Digits data set. The results indicate that discriminative models, such as LaRank, I^{2}VM and ORF, perform better than the generative classifier PCA/LDA. Among the discriminative models the ORF classifier performs worse and needs several epochs to provide a reliable result.

6.1.5.2 Data with Concept-Drift

Table 6.3: Classification error on synthetic data set with pseudo-concept-drift in one class and sorted machine learning data sets. The best results of an incremental learner are bold printed. The numbers in brackets indicate the difference to the corresponding result with unsorted data using the same classifier.

Data set (sorted)	Inc. PCA/LDA [%]	Online RF [%]		LaRank [%]	I^{2}VM [%]
		1 epoch	10 epochs		
Digits	25.7(±0.0)	23.9(-2.8)	17.7(-0.7)	14.6(+2.8)	**9.8(-0.2)**
Digits (2)	42.9(+3.6)	18.3(+4.4)	12.4(+2.8)	15.8(+9.4)	**7.2(+0.2)**
Digits reverse (2)	43.1(+3.8)	15.5(+2.8)	11.7(+2.1)	10.7(+4.3)	**7.1(+0.1)**
DNA	23.2(-0.2)	42.1(+27.2)	11.4(+0.9)	28.5(+22.9)	**5.8(±0.0)**
USPS	11.6(-0.2)	18.6(-2.1)	14.3(+3.4)	13.7(+9.4)	**5.8(+0.4)**
SatImage	14.3(+0.7)	18.5(+2.8)	16.3(+4.8)	10.5(+1.1)	**10.1(+0.5)**
Vowel	90.9(+6.5)	53.0(-5.0)	44.4(-0.4)	47.5(+1.4)	**43.1(+0.5)**
Synthetic	34.5(±0.0)	24.0(+4.1)	6.5(+0.2)	24.0(+16.6)	**5.5(+0.2)**

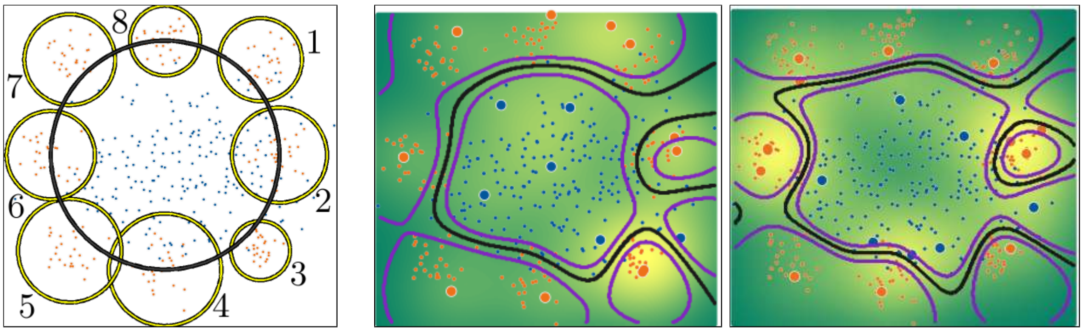
All incremental classifiers except I^{2}VM result in significantly larger error rates for the sorted data set in comparison to the unsorted data set. The average loss in accuracy is 1.9% for incremental PCA/LDA, 1.9% for ORF, 8.5% for LaRank and 0.3% for I^{2}VM. This indicates that I^{2}VM is on the one hand stable with respect to already encountered samples and on the other hand flexible with respect to new encountered samples. Although, incremental PCA/LDA and ORF outperform LaRank concerning the average loss in accuracy, LaRank still results in higher accuracies. Thus, I^{2}VM combines the power of a discriminative classifier with a reconstructive component in order to be robust against concept-drifts.

Figure 6.1 shows the generated synthetic data set with concept-drift and the resulting classification results of IVM and I^{2}VM. The middle and the right plot show that the IVs are spread over the entire data set, i.e. the IVs covers old and new samples equally. Both, the batch and incremental version result in similar posterior probabilities, indicated in violet, and similar decision boundaries, given in black.

Figure 6.2 shows the classification error of I^{2}VM as function of the used training samples. The number of training samples is identical to the number of added training samples, because there is no need for an initial trained model. The plot shows the result for the synthetic data set with concept-drift and random ordering. The stepwise shape of the curve of the sorted data set indicates when a mixture of the distribution of class 2 was learned and included in the model. Both orderings yield the same result verifying that I^{2}VM is independent of the ordering of the samples.

6.1.6 Summary

In this experiment we have shown that I^{2}VM results in similar accuracies as IVM. We underlined that I^{2}VM is independent of the ordering of the samples and thus, can deal with concept-drifts. The results verify that I^{2}VM has a high discriminative power like SVM and



(a) Synthetic data set (b) Classification result of IVM (c) Classification result of I^2VM

FIGURE 6.1: Synthetic data set with generated concept-drift in the distribution of one class. Left: The yellow circles are the 3-σ intervals of the covariance matrices of the Gaussian mixture distribution of class 1 and the dark gray circle is the Gaussian distribution of class 2 without concept-drift. The numbers next to the yellow circles indicate the order, in which the samples are presented to the learner. We start with 2 Gaussian distributions, one for each class, and further ones with concept-drift to the training samples. Middle: Result with IVM. Right: Result with I^2VM. The underlying color in the middle and right image represents the frequency of the import vectors, the black line is the estimated decision boundary and the violet lines are the $P(\mathcal{C}|X) = 0.75$ isolines. The fat blue and orange dots are the IVs after finishing the training. The small blue and orange dots are the training samples.

the reconstructive component causes the model to be stable and at the same time flexible as this is a property of generative classifiers.

6.2 Semantic segmentation of Image Databases

6.2.1 Comparison of Criteria for the Removal of Training Samples

In this experiment we analyze the performance of I^2VM when dealing with long sequences in which not all seen training samples are meant to be stored in the internal representation of the classifier. Thus, a part of the training samples have to be removed in order to keep the model efficient. We evaluate I^2VMs by applying them for the semantic segmentation of image databases. Furthermore, we compare the applicability of various criteria mentioned in Section 5.6 to remove training samples. The semantic segmentation of image databases is challenging because the database can be very large and the ordering of the images in which they presented to the classifier can be arbitrary. We expect that I^2VM can adapt to new encountered samples and a criterion can be found that can remove non-representative training samples.

Besides the performance evaluation we use the probabilistic output to analyze the reliability and the uncertainty of the classification result. We expect that I^2VM is able to provide reliable posterior probabilities. That means, it can be assumed that samples with high class probabilities are accurately classified, whereas relatively low class probabilities are more likely assigned to misclassified samples.

6.2.2 Data

Synthetic data set. We generate a synthetic data set with concept-drift as introduced in Section 6.1.2 with 4800 training samples and 4800 test samples. The data set and the reference classification result is shown in Figure 6.3. The data set simulates the variability of image databases.

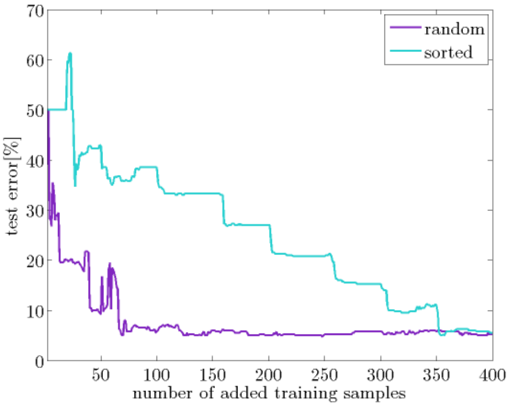


FIGURE 6.2: Classification error of I^{2}VM as a function of used training samples in the synthetic data set with concept-drift and random ordering. The number of used training samples is identical to the number of added training samples, because there is no need for an initial trained model.

FIGURE 6.3: Synthetic data set with 4800 training samples and generated concept-drift in the distribution of one class. Shown is the reference result with no removed training samples. The underlying color in images represents the frequency of the import vectors, the black line is the estimated decision boundary and the violet lines are the $P(\mathcal{C}|X) = 0.75$ isolines. The fat blue and orange dots are the IVs after finishing the training. The small blue and orange dots are the training samples.

The Microsoft Research Cambridge Object Recognition Image database. The Microsoft Research Cambridge Object Recognition Image (MSRC) database (Winn et al., 2005) consists of 240 manually segmented and labeled 8-bit RGB photographs of size 213×320 pixels. The classification task is aiming on 9 classes, namely BUILDING, GRASS, TREE, COW, SKY, AEROPLANE, FACE, CAR and BICYCLE. There is also an class VOID for samples, which are not meant to be classified. Because in this database there are not enough training regions to learn reasonable models of HORSES, WATER, MOUNTAIN and SHEEP we do not consider these classes for training and testing. We divide the images into non-overlapping blocks, each of size 10×10 pixels. From each block we extract Lab color features and HoG features and concatenate the vectorized features. The class of the patch is obtained by majority voting. The characteristics are summarized in Table 6.4.

6.2.3 Experimental Setup

In our experiments we incrementally train the I^{2}VM and report the overall error rate computed on the whole test set. For the synthetic data set we determine the error rate after

Table 6.4: Characteristics of 9-class MSRC database

Class	Color	# Train samples	# Test samples
VOID	⬛	24087	26426
BUILDING	🟫	8923	8389
GRASS	🟩	18095	17430
TREE	🟨	7681	7227
COW	🟦	4343	4203
SKY	⬜	7848	7578
AEROPLANE	🟥	2084	2107
FACE	🟧	2483	2445
CAR	🟪	4852	4646
BICYCLE	🟪	4084	4029
		60393	58054

each new encountered training samples and for the MSRC database we report the error after every 50th new encountered training sample.

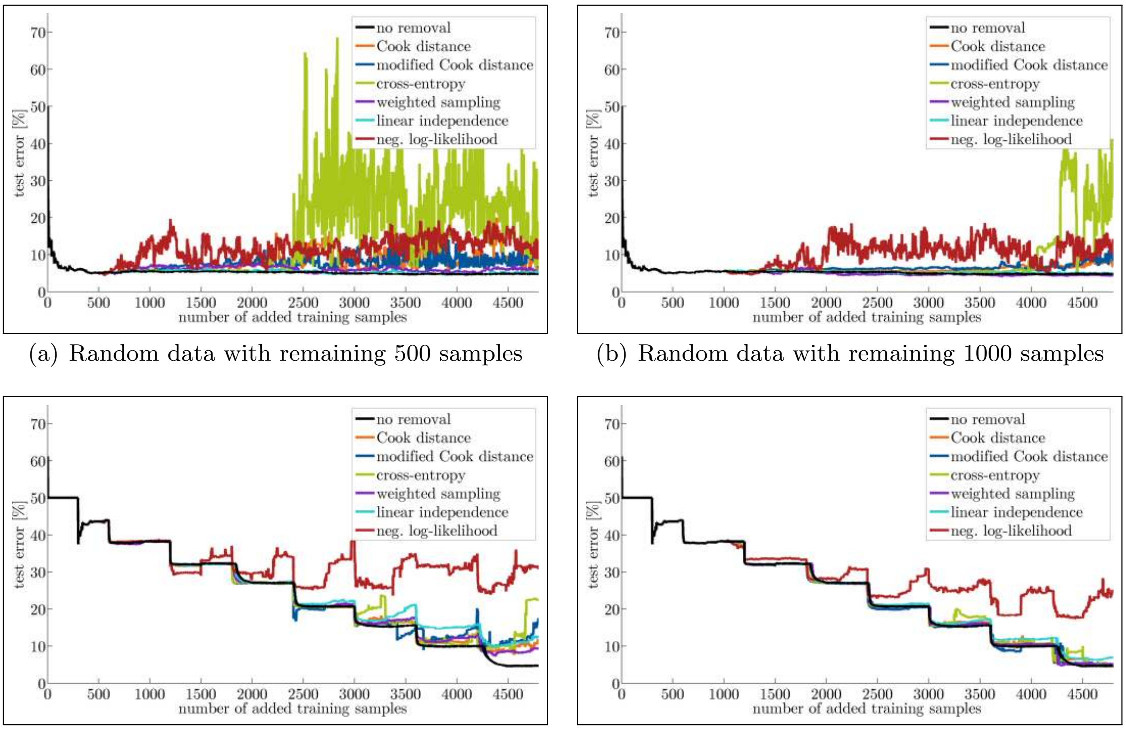
Furthermore, we compare the criteria for removing non-representative training samples by means of the obtained error rates. The criteria that are evaluated are based on different influencing entities. The Cook distance and the modified Cook distance compute the influence of the removal of samples onto the current classifier model. They are based on the posterior probabilities and therefore on the model parameters. The cross-entropy measures the difference between the targets and the posterior probabilities and is therefore also based on the current model. All three criteria try to retain as much discriminative power as possible. The approach of Nguyen-Tuong and Peters (2010) uses an independence measure to remove training samples. Samples that can be well approximated by a linear combination of existing samples are removed. The criteria only depends on the current set of training samples rather than the classifier model. The weighted sampling computes a density of the training samples using the current kernel and the set of import vectors. These criteria try to retain as much reconstructive power as possible. The last criterion uses a greedy backward selection procedure as for the removal of import vectors. These samples that least increase the objective function are removed. This criterion tries to retain as much reconstructive, but also discriminative power. Because the classifier model needs to be recomputed for each tested training sample, the usage for the MSRC database is intractable. Thus, we only evaluated this criterion for the synthetic data set.

Besides the error rates we use the probabilistic output to analyze the reliability and the uncertainty of the classification result. We assess the reliability of the probabilities by rejecting uncertain test samples and deriving the classification accuracy on the non-rejected test points. Following Giacco et al. (2010) we show the accuracy provided by I^{2}VM as a function of the rejection rate in discrete intervals. The rejection rate is given by a threshold on the posterior probability. We choose the threshold for rejection from 1 to 0 in steps of 0.01. We further report the number of retained samples, which indicates the uncertainty of the classification result.

6.2.4 Results and Discussion

In the following sections we discuss our obtained results on the synthetic data sets and the MSRC database.

6.2.4.1 Synthetic Data Set



(a) Random data with remaining 500 samples

(b) Random data with remaining 1000 samples

(c) Sorted data with remaining 500 samples

(d) Sorted data with remaining 1000 samples

FIGURE 6.4: Test errors on a synthetic data set with 4800 training samples. We report the results for 500 and 1000 remaining samples and for the sorted and randomized data set.

Figure 6.4 (a)-(d) show the test errors on the data set using the the mentioned criteria and the reference result with no removal of samples as solid black line. Table 6.5 and 6.6 summarize the accuracies and reports the number of import vectors. The plots and tables indicate that remaining 1000 training samples yields significantly better results than remaining 500 samples. Figure 6.4 (a) shows that only the linear independence criteria and the weighted sampling are able to achieve a comparable accuracy to the reference result. The most unstable results give the cross-entropy and the negative log-likelihood criteria. The reason for the worse result of the negative log-likelihood is that the criterion tries to achieve a low objective function value by removing sample. This is first achieved in that all samples are removed that are incorrect classified and those which are near to the decision boundary. Due to this, the decision boundary deteriorates and more and more samples near to the current decision boundary are incorrect classified. As a result, only samples of one class and the IVs remain, yielding a low objective function value. A possibility to improve this criteria is to introduce an independent validation set on which the objective function value is evaluated. However, the choice of a representative validation set remains challenging. Thus, the criteria that retain as much reconstructive power as possible result in the highest accuracies.

Although plots (a) and (b) show that 500 samples are enough to achieve a low test error

TABLE 6.5: Test errors with remaining 500 training samples using various criteria to remove samples. Reported is the final test error, the mean error and the number of import vectors (IVs).

	random			sorted		
	final [%]	mean [%]	# IVs	final [%]	mean [%]	# IVs
No removal	4.7	5.3	33	4.7	24.6	30
Cook distance	6.5	7.7	30	11.6	**25.3**	26
Mod. Cook distance	7.6	7.7	55	17.5	25.4	36
Cross-entropy	19.0	15.0	6	22.5	25.8	23
Weighted sampling	5.4	6.2	48	**9.5**	**25.3**	50
Independence	**5.1**	**5.5**	39	12.6	26.3	42
Negative log-likelihood	13.5	11.2	47	33.2	31.5	31

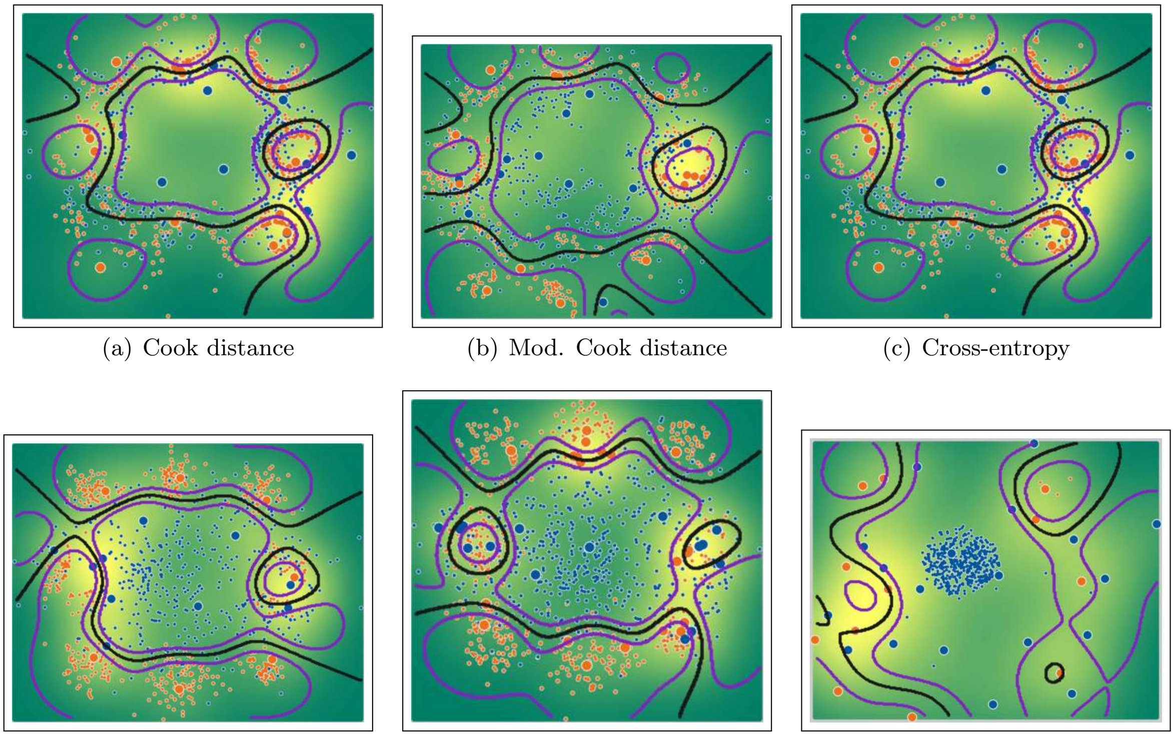
TABLE 6.6: Test errors with remaining 1000 training samples using various criteria to remove samples. Reported is the final test error, the mean error and the number of import vectors (IVs).

	random			sorted		
	final [%]	mean [%]	# IVs	final [%]	mean [%]	# IVs
No removal	4.7	5.3	33	4.7	24.6	30
Cook distance	8.5	6.0	26	**4.6**	24.7	26
Mod. Cook distance	8.4	6.4	30	5.2	**24.4**	28
Cross-entropy	34.6	8.1	6	5.0	24.9	31
Weighted sampling	**4.4**	**5.1**	29	5.2	24.7	23
Independence	5.0	5.4	31	7.0	25.2	35
Negative log-likelihood	14.1	9.4	34	24.7	30.1	25

none of the criteria is able to achieve a comparable accuracy in the sorted data set. However, using 1000 samples for nearly all criteria result in accuracies, which do not differ significantly to the reference result.

Figure 6.5 shows the classification results for all criteria with 1000 remaining training samples on the synthetic data set with concept drift. Figure 6.3 is used as reference since no samples were removed. Both the Cook distance and the modified Cook distance keep samples that are positioned near to the decision boundary. That is reasonable, because these samples have a high leverage value and influence the model most. Also the cross-entropy criteria keeps only samples at the boundary and in areas with a low probability, e.g. overlapping areas. The visually best result gives the linear independence criteria. Although using this criteria gives the worst result (see Table 6.6 on the sorted data set) it looks the most similar to the reference result. The worse result can be explained by the fact that less samples are needed to accurately define the decision boundary than to represent the distribution of the samples. Nevertheless, as shown in Section 3.2.4, only defining the decision boundary can lead to worse results.

These results indicate that criteria that retain as much reconstructive power as possible show better results if the number of kept samples is small. Criteria which retain as much discriminative power as possible need more samples to show comparable accuracies.



(a) Cook distance

(b) Mod. Cook distance

(c) Cross-entropy

(d) Weighted Sampling

(e) Independence

(f) Negative log-likelihood

Figure 6.5: Synthetic data set with 4800 training samples and generated concept-drift in the distribution of one class. Shown is the result with remaining 1000 training samples using various criteria to remove training samples. The underlying color in images represents the frequency of the import vectors, the black line is the estimated decision boundary and the violet lines are the $P(\mathcal{C}|X) = 0.75$ isolines. The fat blue and orange dots are the IVs after finishing the training. The small blue and orange dots are the training samples.

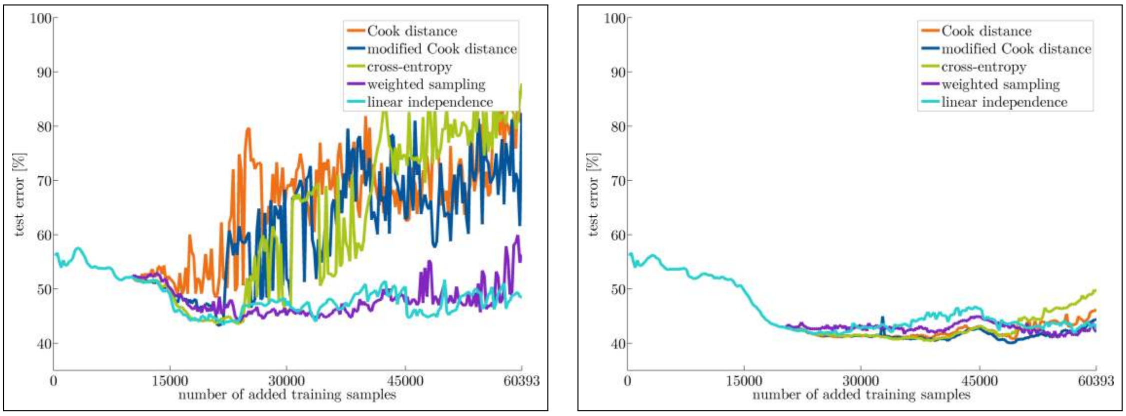
6.2.4.2 Microsoft Research Cambridge Object Recognition Image Database

Figure 6.6 shows the classification results for all criteria with 10000 and 15000 remaining training samples on the MSRC data set. The results show that keeping 15000 training samples in the model shows significantly better results than keeping 10000 samples.

The criteria that retain as much reconstructive power as possible show better results than these one that retain as much discriminative power as possible if the sample size is small. In case of 15000 remaining samples the weighted sampling criterion and the independence criterion show the best final error, but the discriminative models show a better mean error.

The confusion matrix of the classification result with weighted sampling and the modified Cook distance is illustrated in Figure 6.7 (a) and (b). The matrices show that in both cases the classes `grass` and `sky` are classified best. The classes `aeroplane` and `cow` give the worst classification result. Both criteria show high accuracies in the best predicted classes, however, the results differ significantly from each other.

Figure 6.8 presents 5 classification results of I^{2}VM with modified Cook distance (second row) and weighted sampling strategie (third row). The qualitative evaluation of the results shows that I^{2}VM yields reasonable results. As already stated, there exists some misclassification for each class. For example, the results underline that the classes `grass` and `sky` result in high accuracies, whereas the class `cow` is falsely classified as `bicycle` when using the modified Cook distance and as `face` when using the weighted sampling strategy. The class `aeroplane` cannot be classified correctly.



(a) 10000 remaining samples

(b) 15000 remaining samples

FIGURE 6.6: Test errors on the MSRC data set with 10000 and 15000 remaining samples.

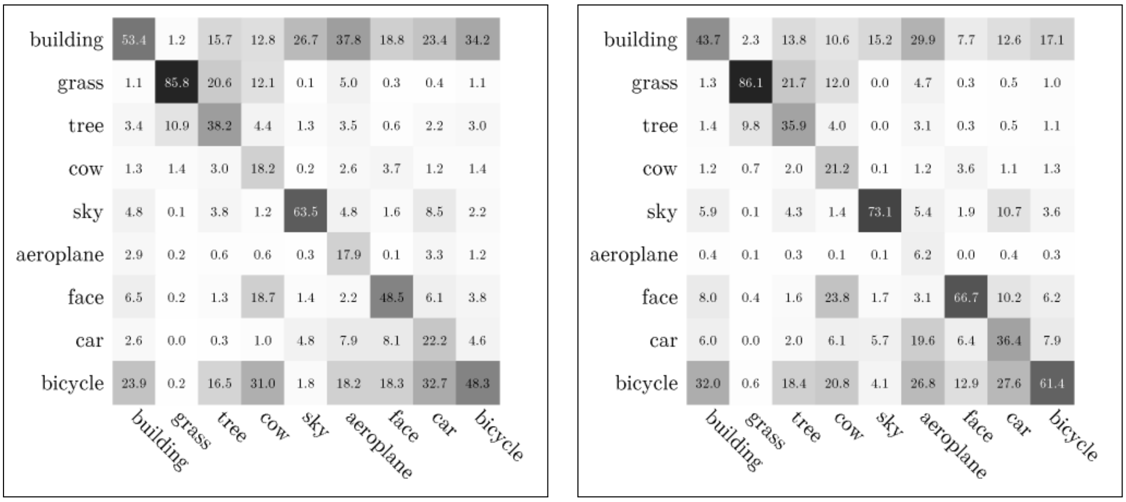
TABLE 6.7: Test errors with remaining 10000 training and 15000 samples using various criteria to remove samples. Reported is the final test error, the mean error and the number of import vectors (IVs).

	10000 remaining samples			15000 remaining samples		
	final [%]	mean [%]	# IVs	final [%]	mean [%]	# IVs
Cook distance	80.7	64.1	99	46.1	45.2	77
Mod. Cook distance	82.4	60.3	79	44.3	**44.8**	77
Cross-entropy	87.8	61.5	99	49.5	45.6	72
Weighted sampling	56.3	49.0	32	**42.1**	45.6	89
Independence	**48.5**	**48.6**	125	42.6	46.0	100

In order to evaluate the probabilistic output, an analysis is shown in Figure 6.9. With an increasing rejection threshold I^{2}VM provides in both cases a decreasing test error. Consequently, it can be assumed that samples with high class probabilities are accurately classified by I^{2}VM, whereas relatively low class probabilities are more likely referred to misclassified samples. Figure 6.9 (a) and (c) indicate that the classes `grass` and `sky` are most certain, whereas `aeroplane` and `cow` primarily contains uncertain classified pixel. Thus, the plots underline our finding from Figure 6.7. Figure 6.9 (b) and (d) show that I^{2}VM is able to provide reliable posterior probabilities. If samples with a posterior smaller than 0.5 are rejected, a test error of 0% can be obtained. A comparison of the two criteria shows that the curves have the same trends, but the curves of the weighted sampling are more pronounced.

6.2.5 Summary

We analyzed the performance of I^{2}VM when dealing with long sequences in which not all seen training samples are meant to be stored in the internal representation of the classifier. We showed on a synthetic data set that a criterion for the removal of training samples can be found with that I^{2}VM can achieve the same accuracy as if all samples are used. Both the synthetic data set and the MRSC data set verify that I^{2}VM is able to remove training samples. The criteria that retain reconstructive power show better results than the criteria that retain discriminative power if the number of remained samples is small. If the sample size is large enough also the criteria that retain discriminative power show good results. Furthermore, we have shown that I^{2}VM is able to provide reliable posterior probabilities,



(a) Modified Cook distance

	building	grass	tree	cow	sky	aeroplane	face	car	bicycle
building	53.4	1.2	15.7	12.8	26.7	37.8	18.8	23.4	34.2
grass	1.1	85.8	20.6	12.1	0.1	5.0	0.3	0.4	1.1
tree	3.4	10.9	38.2	4.4	1.3	3.5	0.6	2.2	3.0
cow	1.3	1.4	3.0	18.2	0.2	2.6	3.7	1.2	1.4
sky	4.8	0.1	3.8	1.2	63.5	4.8	1.6	8.5	2.2
aeroplane	2.9	0.2	0.6	0.6	0.3	17.9	0.1	3.3	1.2
face	6.5	0.2	1.3	18.7	1.4	2.2	48.5	6.1	3.8
car	2.6	0.0	0.3	1.0	4.8	7.9	8.1	22.2	4.6
bicycle	23.9	0.2	16.5	31.0	1.8	18.2	18.3	32.7	48.3

(b) Weighted Sampling

	building	grass	tree	cow	sky	aeroplane	face	car	bicycle
building	43.7	2.3	13.8	10.6	15.2	29.9	7.7	12.6	17.1
grass	1.3	86.1	21.7	12.0	0.0	4.7	0.3	0.5	1.0
tree	1.4	9.8	35.9	4.0	0.0	3.1	0.3	0.5	1.1
cow	1.2	0.7	2.0	21.2	0.1	1.2	3.6	1.1	1.3
sky	5.9	0.1	4.3	1.4	73.1	5.4	1.9	10.7	3.6
aeroplane	0.4	0.1	0.3	0.1	0.1	6.2	0.0	0.4	0.3
face	8.0	0.4	1.6	23.8	1.7	3.1	66.7	10.2	6.2
car	6.0	0.0	2.0	6.1	5.7	19.6	6.4	36.4	7.9
bicycle	32.0	0.6	18.4	20.8	4.1	26.8	12.9	27.6	61.4

Figure 6.7: Confusion matrix of classification results obtained with modified Cook distance and weighted sampling criterion.

which can be used for uncertainty analysis. Despite the deletion of samples, this ability is preserved. Summarizing, I^{2}VM can deal with long sequences in which not all samples can be stored. Furthermore, I^{2}VM is able to predict both the class-memberships and the posterior probabilities of test samples.

6.3 Large Area Land Cover Classification

6.3.1 Evaluation of I^{2}VM for Large Area Land Cover Classification with Self-training

In this experiment we evaluate I^{2}VM concerning the ability for self-training in order to adapt to changes in the data distribution. For this purpose we use a large area consisting of 9 composite remote sensing images, which were acquired by a Landsat satellite. Because we have only access to labeled training samples in one image, we acquire new labeled training samples using the probabilistic output of the I^{2}VM. We apply self-training to update the classifier in order to classify the landcover of all images. The scenes are characterized by both spatial and temporal differences. We expect that I^{2}VM can adapt to changes in the distribution of the training samples and the probabilities are useful to acquire new samples.

6.3.2 Recent Approaches for Large Land Cover Classification

Land cover classification is one of the main applications in the field of remote sensing image analysis. Currently, the development in land cover classification of remote sensing images is mainly driven by methods based on the field of machine learning and pattern recognition and the increased availability of remote sensing data (Richards, 2005). Furthermore, increasing processing power and more sophisticated algorithms enables faster processing of huge data sets. However, most methods are focusing on spatial limited areas, while many applications require large area land cover maps (Cihlar, 2000; Franklin and Wulder, 2002). Hence, an appropriate classification of large areas is an important and ongoing research topic in the field of remote sensing (Knorn et al., 2009; Pekkarinen et al., 2009; Walker et al., 2010). The classifi-

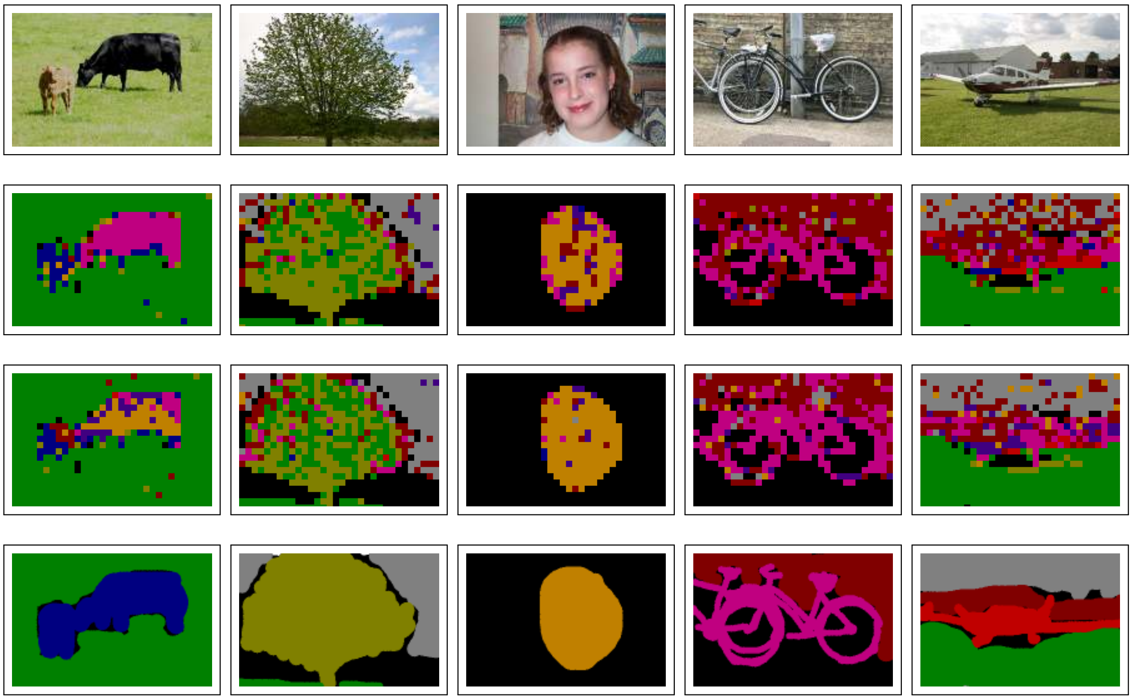
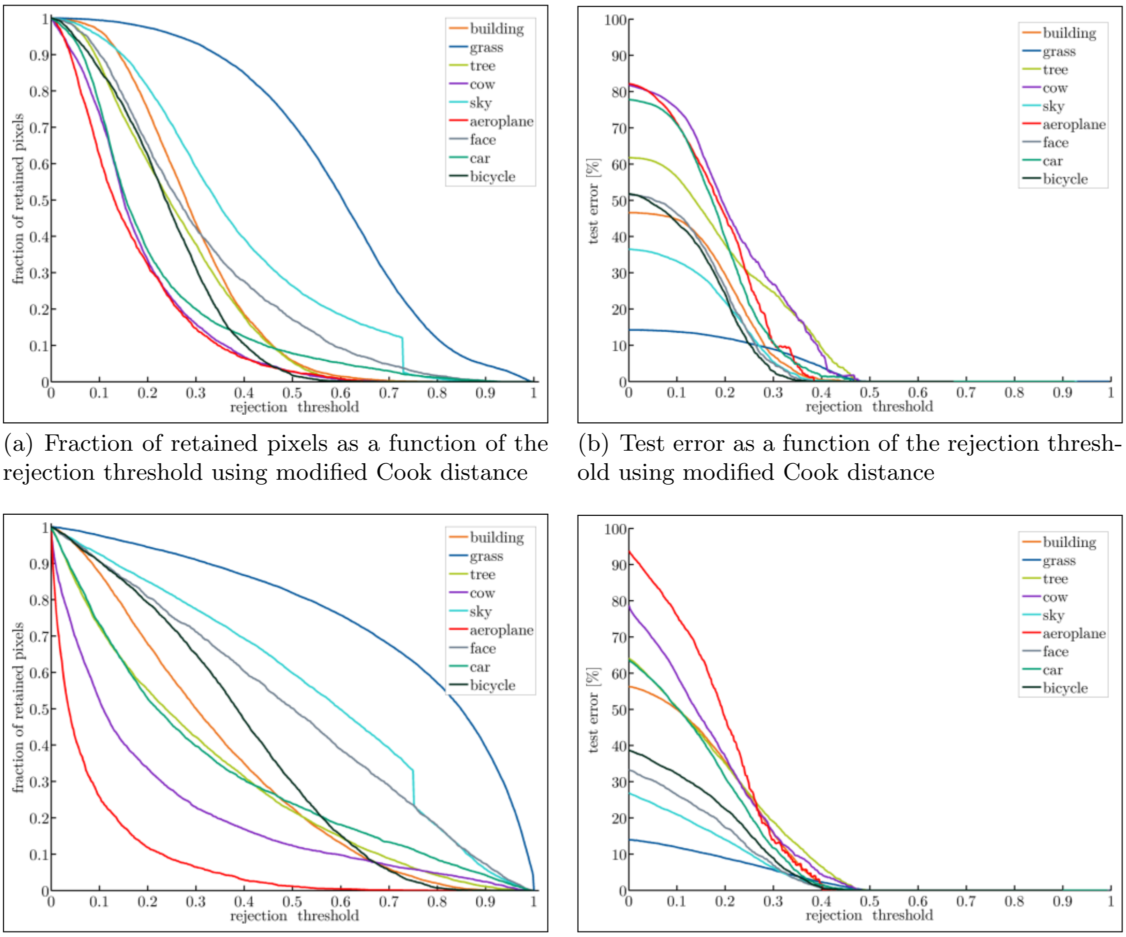


Figure 6.8: Classification results of 5 images of the MSRC database (top row) with modified Cook distance (second row) and weighted sampling strategy (third row). The ground truth is given in the bottom row.

cation of data with high spatial resolution and often relative narrow swath, e.g. Landsat-line data, is usually performed on individual images. In addition, most classifications are based on a supervised method, which requires the – often costly and time consuming – selection of adequate training data in the area covered by each individual image. As a matter of fact, the mapping of large areas is challenging and requires an efficient classification concept. Nevertheless, the use of Landsat data seems interesting, due to (i) the long-term data archive, (ii) the availability of free Landsat data by the USGS and (iii) the relatively large swath width and adequate spatial resolution. Knorn et al. (2009), for example overcome the challenges mentioned above by using overlapping areas of neighboring Landsat scenes. In the beginning an initial image, with corresponding training data, is individually classified by a standard SVM. The classification result is used to identify new training samples in the overlapping areas. These training samples are used to train a new SVM model for the neighboring scene. While this strategy was successfully used for generating a forest / non-forest map (i.e. , a binary classification), we extended the concept to multi-class problems (Roscher et al., 2012). Moreover, we use I^{2}VM, which enables the efficient classification of neighboring Landsat scenes with a larger number of training data.

6.3.3 Self-Training for Sequential Mosaic Classification

Self-training as a special case of active learning can be used to classify each scene in a mosaic, when only a limited number of labeled data samples is available. For further discussion of active learning we refer to Tuia et al. (2009). We pursue the following strategy: Starting with an initial classification in a single image, the classification result in the overlapping area is used to retrain or update the learned classifier for a neighboring scene. Given two neighboring scenes I_c, the current scene, and I_t, the target scene, a classifier is trained on I_c and labels the unlabeled data in the overlapping area. The predictions which are selected with respect



(a) Fraction of retained pixels as a function of the rejection threshold using modified Cook distance

(b) Test error as a function of the rejection threshold using modified Cook distance

(c) Fraction of retained pixels as a function of the rejection threshold using weighted sampling

(d) Test error as a function of the rejection threshold using weighted sampling

FIGURE 6.9: Evaluation of the probabilistic output using modified Cook distance and weighted sampling. The reliability of the probabilities is obtained by rejecting uncertain test samples and deriving the classification accuracy on the non-rejected test points. The uncertainty of the classification result is indicated by analyzing the number of retained samples.

to a criteria are used as new training labels. These samples and the features of I_t are used as new training samples to retrain or update the classifier, and finally to classify the image. Subsequently the remaining images can be classified by following the same procedure.

The general concept of this approach seems advantageous due to the fact that training samples are only required for the initial image. However, this approach requires that all classes of interest occur in each overlapping area. This cannot be guaranteed particularly for small classes. To overcome this problem we keep old labeled training data and use I^{2}VM to incrementally adapt the model to the current target scene.

Although this approach is useful to acquire new labeled samples, it should be noted that using self-training by itself does not necessarily lead to better results. The main problem of self-training is that acquired samples with incorrect labels reinforce themselves. It is caused by the assumption that the classifier's own predictions with high confidence are correct. As one can imagine this may not always be the case. Also correctly acquired samples may deteriorate the model if they cause an imbalance in the data distribution or their distribution is different to that of the labeled samples. If the number of labeled samples in the initial scene is to small the model may not be accurate enough to acquire new samples with correct labels. On the other hand, if the number of labeled samples is too big, the model may not be improved anymore. Therefore, the identification of confident samples, which also ensure an improvement of the classifier model usually demand further strategies.

In the following we want to exploit the possibility of transferring a classifier over multiple neighboring scenes using I^{2}VM. Though we use the original data we are aware that adequate preprocessing, see e.g. Bodart et al. (2011) can reduce the spatial and temporal variability and thus further increase the performance.

6.3.4 Data

Figure 6.10 shows our chosen study site around Rondonia in South America. The data set contains 9 Landsat 5 TM images from 2009, which are freely available by USGS Landsat archive. The area covers about 285000 km^2.

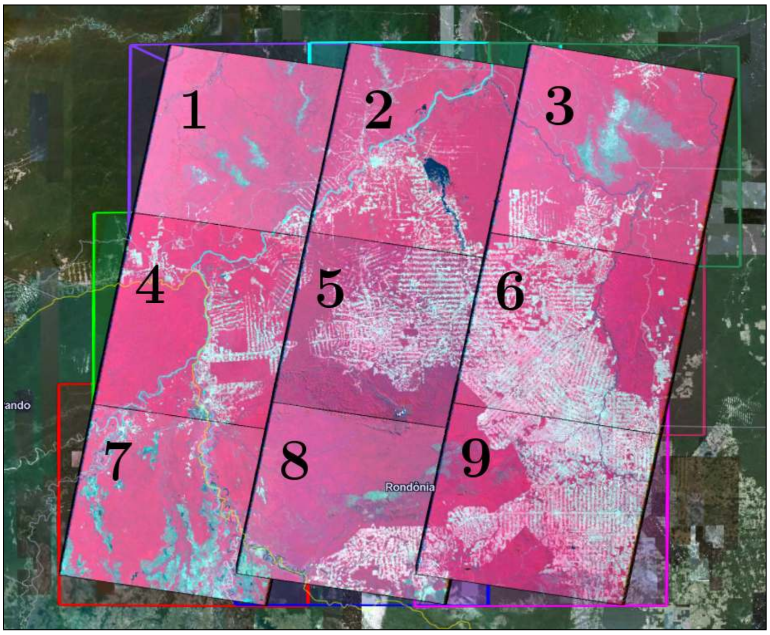


Figure 6.10: The area of Rondonia with overlayed Landsat images displayed with bands 4-3-2. Numbers in the upper left corner of the Landsat images are referred to the numbers in the text. The three image strips are acquired in September (images 1, 4 and 7), July (images 2, 5 and 8), and August 2009 (images 3, 6 and 9). The underlaid image is taken from Google Earth.

The study site is dominated by forests and agriculture/deforestation, and characterized by typical spatial patterns and temporal variation caused by different acquisition dates, i.e. , three image strips are acquired in September (images 1, 4 and 7), July (images 2, 5 and 8), and August 2009 (images 3, 6 and 9).

The classification aims at 4 land cover classes, focused on Forest, Agriculture, Water and Urban. A limited number of training and test data were collected by photointerpretation; using the Landsat images themselves and high-resolution images in Google Earth. The land cover classes occur in every scene, but not in every overlap area. The availability is given in Table 6.8 showing the number of pixels per class in each of the 9 scenes. We choose 500 pixels from each class for training and the rest for testing.

Table 6.8: Number of labeled pixels in the 9 scenes.

Scene	Forest	agriculture	Water	Urban
1	12693	21187	29067	2135
2	27668	49901	38307	40830
3	21168	46299	2131	3667
4	44730	31988	8685	19344
5	63697	21459	5849	12860
6	15913	32735	5113	12367
7	23913	112416	56190	15804
8	99127	106057	16930	4140
9	71931	90364	5494	52603

6.3.5 Experimental Setup

In our experiments we use the center image I_5 as initial scene to train the initial classifier. We randomly choose 500 samples per class and train the IVM as described in Section 3.3.

We compare different paths for transferring the classification model from the center image to the other images:

1. We transfer the classification model to the direct neighbors, i.e. , 2, 4, 6 and 8, using the corresponding overlapping regions 5-2, 5-4, 5-6, and 5-8.

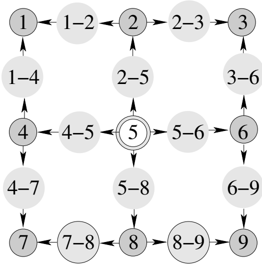


Figure 6.11: Paths between the 9 images across-track and along-track. Each pair of overlapping images i and j has a common area $i - j$ which is used to transfer the classification model from i to j or vice versa.

2. We transfer the classification model to the indirect neighbors, i.e. , to 1, 3, 7 and 9. This can again be performed in two ways:

   (a) We use single paths via one of the two the direct neighbors, e. g. $5 \rightarrow 2 \rightarrow 1$ or $5 \rightarrow 4 \rightarrow 1$.

   (b) We merge two single paths. E.g. we simultaneously use the overlapping regions 5-4 and 5-2 for reaching the image pair (2,4), and then use the two overlapping regions 1-2 and 1-4 for reaching the image 1, see Figure 6.12, left.

Within each overlapping area we choose the most confident predictions measured by the probability and randomly select not more than 1000 training samples per class. We fix the confidence threshold to 0.8, i.e. samples with a posterior probability in $[0, 0.8]$ are discarded and only those in $]0.8, 1]$ are considered as new training samples. All training samples are at least 10 pixel apart from training samples assigned to other classes to ensure a more reliable acquisition.

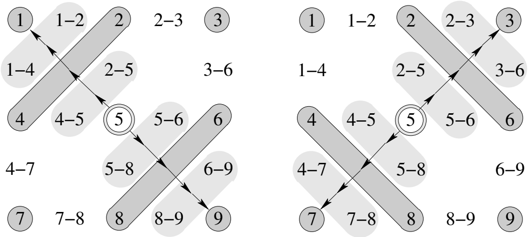


FIGURE 6.12: Paths between the 9 images across-track, i. e. diagonal to the grid. The transfer from one image to a diagonal one uses the two neighbors and the four overlapping regions.

To evaluate our results we classified each image separately, using a common IVM classification and individual training data for each image (from now on referred to as reference classifications). We compare I^{2}VM to the following approaches:

Old model We use the initial trained model in scene I_c to classify the target scene I_t.

IVM (overlap) We only use the training data from the overlapping area of I_c and I_t as proposed in Knorn et al. (2009).

IVM (batch) We use the initial training data and the newly acquired training data from the overlapping area(s) and retrain the model from scratch.

I^{2}VM We update our initial classifier incrementally by adding the new training data without retraining from scratch.

We report the overall accuracy (OA), averaged class-wise accuracies (AA) and the kappa coefficient Cohen (1960) for the classification.

6.3.6 Results and Discussion

TABLE 6.9: Reference classifications using 500 training samples per class. Reported are the overall accuracy a_{oa}, the average accuracy a_{aa}, the kappa coefficient κ and the number of import vectors (IVs).

Scene	a_{oa}[%]	a_{aa}[%]	κ	# IV
1	93.0	91.3	0.89	19 IV
2	95.4	96.1	0.94	16 IV
3	95.8	96.4	0.92	14 IV
4	97.0	96.7	0.96	16 IV
5	97.3	96.1	0.95	20 IV
6	96.5	96.5	0.95	16 IV
7	92.6	94.4	0.88	19 IV
8	96.8	97.3	0.95	12 IV
9	93.5	94.3	0.90	16 IV

Table 6.9 shows the reference classification of each Landsat scene in the mosaic. The overall accuracy ranges from 93% for scene 1 to 97.3% in the initial image with ID 5. The results show that the number of IVs is not higher than 20, which is 1% of the used data samples. This indicates that the model is very sparse, ensuring a fast classification and less storage requirements.

Table 6.10 shows the accuracy assessment, following the proposed method.

Table 6.10: Accuracy assessment of classified target scenes. We report the overall accuracy a_{oa}, the average accuracy a_{aa} and the kappa coefficient κ. We distinguish the two separate paths via a single image (first 4 rows) and the unique path via the two direct neighbor images. The best results with respect to the overall accuracy is bold printed. Note that for IVM (overlap) in cases with no overlapping area no result can be obtained.

Transfer	Old model			IVM (overlap)			I^{2}VM			IVM (batch)		
	a_{oa}	a_{aa}	κ	a_{oa}	a_{aa}	κ	a_{oa}	a_{aa}	κ	a_{oa}	a_{aa}	κ
5→2	**93.9**	94.9	0.92	86.9	88.1	0.82	93.6	94.6	0.91	93.6	94.6	0.91
5→4	63.0	71.2	0.49	82.2	78.0	0.73	**92.0**	91.8	0.88	91.8	91.7	0.88
5→6	83.3	90.8	0.76	89.8	89.6	0.84	94.0	94.8	0.91	**94.2**	94.8	0.91
5→8	**97.2**	92.4	0.95	96.9	74.6	0.95	97.1	91.2	0.95	97.1	91.4	0.95
5→4→1	79.8	82.8	0.71	64.2	50.0	0.48	**80.5**	84.3	0.73	80.4	84.3	0.72
5→2→1	79.8	82.8	0.71	**93.8**	72.8	0.90	93.0	87.0	0.90	92.9	85.4	0.89
5→(4+2)→1	79.8	82.8	0.71	-	-	-	**92.6**	83.9	0.89	**92.6**	84.4	0.89
5→2→3	73.2	87.8	0.60	**92.2**	73.9	0.84	83.9	90.8	0.73	83.1	90.7	0.72
5→6→3	73.2	87.8	0.60	66.1	50.0	0.14	**90.3**	89.3	0.82	88.7	89.8	0.80
5→(2+6)→3	73.2	87.8	0.60	-	-	-	93.1	88.8	0.87	**93.2**	88.8	0.87
5→4→7	61.2	79.0	0.50	54.0	61.2	0.27	65.8	83.7	0.56	**66.0**	83.8	0.56
5→8→7	61.2	79.0	0.50	**92.2**	86.6	0.87	91.5	85.2	0.86	91.6	86.0	0.86
5→(4+8)→7	61.2	79.0	0.50	-	-	-	**91.4**	85.1	0.86	91.3	84.2	0.86
5→6→9	87.5	91.1	0.82	53.7	60.9	0.27	90.0	92.1	0.85	**90.8**	92.7	0.86
5→8→9	87.5	91.1	0.82	73.8	58.2	0.58	93.0	93.5	0.89	**93.4**	94.0	0.90
5→(6+8)→9	87.5	91.1	0.82	-	-	-	92.9	93.8	0.89	**93.0**	93.6	0.89

Classifying direct neighbors. The results show that in case of the initial transfer in the along track (5→2, 5→8) the old model achieve the highest accuracies. We assume that the reason for this fact is the identical acquisition date of all three images. However, the overlapping area does not necessarily contains training samples for each class. Thus, using solely the information in the overlapping area may results in a lower classification accuracy. In this case misclassified pixels are incorrectly used as new training labels.

Using the difference to the reference classification, the loss in the overall accuracy is 12.1% on average when using the old model, 7.5% when only training with the overlapping area and 2.3% when training with the batch or incremental version of the IVM using old and new data.

Classifying indirect neighbors. The results for the chain classification indicate that using the old model is not an adequate strategy for a reliable land cover classification. The average loss in accuracy using the old model is 18.3%, 5.7% when using only training data from the overlapping areas, 1.1% using I^{2}VM and 1.0% using the batch version.

The highest accuracy can be achieved when the chain contains scenes of the same acquisition date as long as possible. Also, going parallel and perpendicular to the strip direction appears to be better than to go diagonally. The results show that the difference in the accuracy between IVM and I^{2}VM is not significant.

6.3.7 Summary

As stated before, we demonstrated that I^{2}VM shows comparable results to IVM. We showed in our experiments that I^{2}VM is suitable for self-training, i.e. the model remains stable with respect to old samples and flexible with respect new encountered ones. The probabilistic output can be used to acquire new training samples in order to adapt the classifier model to changes in the data distribution.

6.4 Tracking-by-Segmentation

6.4.1 Evaluation of the Applicability of I^{2}VM for Data Streams

In this experiment we show that I^{2}VM is able to simultaneously cope with the mentioned challenges that appear when dealing with sequential data, see Chapter 1. For this purpose we integrate I^{2}VM into a framework for object tracking in image sequences following the concept of tracking-by-segmentation. The separation of object and background is achieved by a consecutive semantic superpixel segmentation of the images, yielding tight object boundaries. I.e. , in the first image a model of the object's characteristics is learned from an initial, incomplete annotation. This model is used to classify the superpixels of subsequent images to object and background employing graph-cut (Boykov et al., 2001). We assume the object boundaries to be tight-fitting and the object motion within the image to be affine. To adapt the model to radiometric and geometric changes we utilize I^{2}VM with self-training. We evaluate our tracking framework qualitatively and quantitatively on several image sequences. We expect that I^{2}VM can deal with arbitrary long data streams characterized by changes in the data distribution. We further expect that I^{2}VM is suitable to be integrated into a tracking framework showing at least similar results as batch tracking methods.

6.4.2 Data Sets

For tracking purposes we use the following data sets.

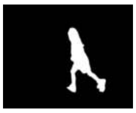
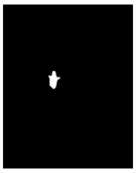
Flower image sequence. For the qualitative analysis we use the FLOWER image sequence comprising 163 images acquired with a stereo camera. In this sequence a flower is meant to be tracked. We will use this sequences to have a detailed look on the incremental update of the set of import vectors of I^{2}VM. The sequence is characterized by rapid changes in the shape and the color of the object.

TABLE 6.11: Characteristics of the FLOWER image sequence.

Data set	FLOWER
# Images	163
Length [sec]	7
Resolution [pixel]	1024×768
Used features	Lab+pos+disparity
Example	

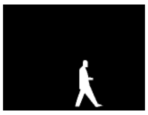
SegTrack database. For an accurate evaluation we use the SegTrack database of Tsai et al. (2011). The database comprises 6 image sequences with ground-truth segmentations for each image. The data set includes for each image to varying degrees, the challenge of color overlap between object and background appearance, interframe motion and change in object shape. The characteristics of the image sequences are summarized in Table 6.12.

Table 6.12: Characteristics of the SegTrack image sequences.

Data set	parachute	girl	monkeydog	penguin	birdfall	cheetah
# Images	51	21	71	42	30	29
Length [sec]	2	1	3	2	2	2
Resolution [pixel]	414×352	400×320	320×240	640×352	259×327	320×240
Used features	Lab+pos	Lab+pos	Lab+pos	Lab+pos	Lab+pos	Lab+pos
Example						
Annotation						

Adafrag database. For a further comparison we use the Adafrag database of Chockalingam et al. (2009). The database consists of 5 image sequences with provided ground truth in about every fifth frame of the intermediate frames. We skip one image sequence, because it is already included in the SegTrack database. We also use this database for a comparison to IVM, using only Lab color features. The characteristics of the image sequences are summarized in Table 6.13.

Table 6.13: Characteristics of the long image sequences.

Data set	elmo	walkTree	walkCar	girl
# Images	401	221	286	181
Length [sec]	16	9	12	7
Resolution [pixel]	320×240	320×240	320×240	320×240
Used features	Lab + pos	Lab + pos	Lab + pos	Lab + pos
Example				
Annotation				

Long image sequences. In order to evaluate the stability of I^{2}VM over a long term we use image sequences with more than 1000 frames. For a qualitatively analysis we use a famous sequence from the movie "American Beauty" called american beauty[1], a video from Youtube

[1] http://www.youtube.com/watch?v=gHxi-HSgNPc

called CHAMELEON[2] and a video acquired with a drone denoted with DRONE. Furthermore, we perform a quantitative evaluation on the PROST tracking-by detection database comprising long image sequences with given ground truth rectangles[3]. The characteristics of the image sequences are summarized in Table 6.14.

Table 6.14: Characteristics of the Adafrag image sequences.

Data set	LEMMING	LIQUOR	BEAUTY	CHAMELEON	DRONE
# Images	1336	1741	4446	1064	2354
Length [sec]	16	9	187	42	95
Resolution [pixel]	640×480	640×480	480×360	640×480	480×270
Used features	Lab + pos	Lab + pos	Lab + pos	HSV + pos	Lab + pos
Example					

6.4.3 Tracking Framework

In this section we explain our proposed tracking framework in detail. We also consider the extraction and tracking of superpixels and the formulation of the conditional random field (CRF) (Lafferty et al., 2001) model. In the next paragraph we will give an overview of the tracking framework and in the subsequent paragraphs we will explain the single steps in more detail. The further details and results we refer to the report of Roscher et al. (2012).

6.4.3.1 Tracking Scheme

Our tracking framework is schemed in Figure 6.13. The extraction of superpixels $\mathcal{R}_{(t)}$ in the image $\mathsf{I}_{(t)}$ at a given time step and the tracking of superpixels from one time step to the next one is explained in Section 6.4.3.2 and denoted with ST. If a stereo image pair $\mathcal{I}_{(t)}$ is given, we refer to the left image as reference image. Furthermore we have given a set of data samples $\mathcal{T}_{(t-1)}$, a set of import vectors $\mathcal{V}_{(t-1)}$ and a classifier model $\mathcal{M}_{(t-1)}$, which consists of the estimated model parameters and necessary entities, that must be stored.

The framework is based on a classification step C yielding the segmentation and a learning step L, in which the set of data samples and import vectors is updated and new model parameters are estimated. After the segmentation step, new labeled training samples are acquired, enabling the learned model $\mathcal{M}_{(t)}$ to adapt to radiometric and geometric changes in the learning step.

In the segmentation step we use a bounding box prior $\mathsf{P}_{B,(t)}$, which was obtained from the segmented image $\mathcal{C}_{(t-1)}$. We use optical flow to transform the position of the box into the current image $\mathsf{I}_{(t)}$. The posteriors $\mathsf{P}_{(t)}$ are obtained by using the model $\mathcal{M}_{(t-1)}$ as well as the bounding box, see Section 6.4.3.4. Given the posteriors we predict a new segmented image $\mathcal{C}_{(t)}$. From the segmentation $\mathcal{C}_{(t)}$ we sample the most uncertain predictions as new labeled data samples $\Delta\mathcal{T}_{+}$, i.e. superpixels, and use it for the learning step in order to update the model $\mathcal{M}_{(t-1)}$ yielding $\mathcal{M}_{(t)}$. In order to prevent in continuous growing of data samples, we delete the oldest ones $\Delta\mathcal{T}_{-}$.

[2] www.youtube.com/watch?v=KMT1FLzEn9I

[3] http://gpu4vision.icg.tugraz.at/index.php?content=subsites/prost/prost.php

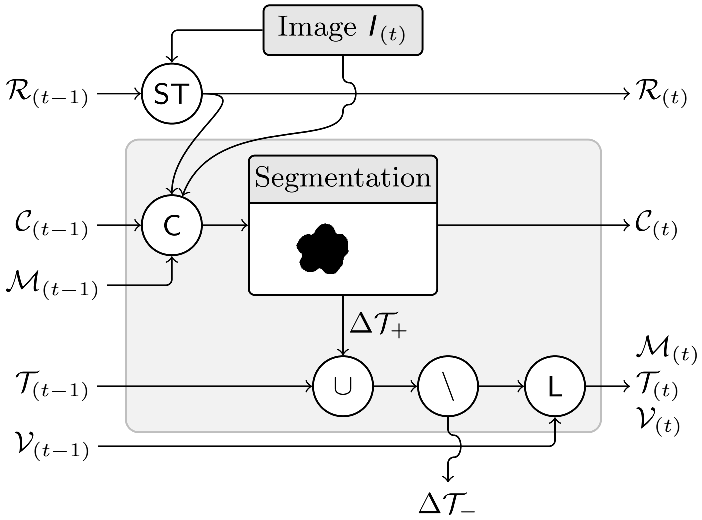


FIGURE 6.13: Using tracked superpixel regions $\mathcal{R}_{(t-1)}$, the classification $\mathcal{C}_{(t-1)}$ and the model $\mathcal{M}_{(t-1)}$ obtained from the incremental learning procedure, we predict a segmentation $\mathcal{C}_{(t)}$. We update the set of labeled training samples $\mathcal{T}_{(t-1)}$, import vectors $\mathcal{V}_{(t-1)}$ and the learned model $\mathcal{M}_{(t-1)}$ to $\mathcal{T}_{(t)}$, $\mathcal{V}_{(t)}$ and $\mathcal{M}_{(t)}$ to derive a segmentation in the next iteration.

In the learning step we incrementally learn the new model $\mathcal{M}_{(t)}$ in a self-training scheme, denoted with L. With the new model and the updated sets $\mathcal{T}_{(t)}$ and $\mathcal{V}_{(t)}$ we obtain the segmentation in the next iteration.

6.4.3.2 Superpixel Tracking

The usage of superpixels has become popular in several applications with the need of feature extraction. They catch redundancy in the image and reduce the complexity to train and to test classifiers.

We follow the idea of Achanta et al. (2010) to generate superpixels $\mathcal{R}$ that are compact, have a regular shape, but are also homogeneous in their spectral features. The complete set of pixels $\{\boldsymbol{x}_i\}$, $i \in \mathcal{I} = \{1, \ldots, I\}$ is clustered into a set of R superpixels $\mathcal{X}_r$ with $\bigcup_r \mathcal{X}_r = \mathcal{I}$ based on their spectral appearance $\boldsymbol{f}_n$ (Lab-color vector) and position in the image $\boldsymbol{b}_n$. The algorithm is initialized by sampling R cluster centers from the image at a regular grid interval of g pixels. Each pixel in the image is then assigned to one cluster center $\boldsymbol{q}_n$ using the K-means-algorithm, where the distance of pixels and cluster centers is computed by the similarity measure $d_{rn} = ||\boldsymbol{q}_r - \boldsymbol{q}_n||$ with $\boldsymbol{q}_{(\cdot)} = [\boldsymbol{f}_{(\cdot)}, \frac{w_S}{g}\boldsymbol{b}_{(\cdot)}]^\mathsf{T}$ (Achanta et al., 2010). The parameter w_S weights the influence of spatial proximity. The higher w_S is, the more compact are the superpixels. The grid interval g adapts this weight to the image resolution.

In order to keep the affiliation of image structures to superpixels over time, we extend this approach to a superpixel tracking scheme similar to the one presented in Levinshtein et al. (2010). The basic idea is to predict the position of the superpixel centers $\boldsymbol{b}_{r,(t-1)}$ in the subsequent frame $t-1$, using the optical flow information of its assigned pixels. The predicted centers $\widetilde{\boldsymbol{b}}_{r,(t)}$ are then used as initial cluster centers for the superpixel segmentation of frame t. For dense optical flow computation we employ the approach of Chambolle and Pock (2011). For a further description we refer to (Roscher et al., 2012).

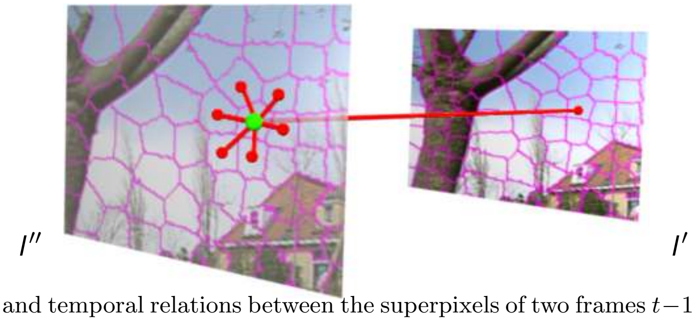


FIGURE 6.14: Spatial and temporal relations between the superpixels of two frames $t-1$ and t. Each superpixel is connected to its neighbors within the image and to its neighbor from the previous frame, assigned via superpixel tracking.

6.4.3.3 Features for Object Representation

For each superpixel segment in $\mathcal{R}_{(t)}$ we extract the following features: (1) radiometric appearance: mean RGB and Lab color, (2) motion: mean optical flow, and (3) geometry: mean position and mean disparity (if stereo images are available).

In order to obtain a disparity measurement for nearly every pixel within the image, we employ a dense stereo approach with an implementation from Gehrig et al. (2009), which is based in the Semi-Global Matching algorithm from Hirschmüller (2005). Pixels with no disparity information, e.g. , caused by stereo shadow, are flagged as invalid and are not considered for computing the mean disparity feature.

For optical flow the motion information is extracted using the approach of Chambolle and Pock (2011), yielding a displacement vector for every pixel.

We use radiometric features in combination with geometric and /or geometric and motion features within the I^{2}VM learning procedure. In order to identify the most discriminating features, we perform cross-validation using different concatenations of features in the first image. We choose these features with the highest cross-validation accuracy.

6.4.3.4 Segmentation

To incorporate prior knowledge about the spatial and temporal relations between tracked superpixels we model our task as a CRF, as shown in Figure 6.14. We prefer short object boundaries and temporal consistency of the superpixel labels.

Our CRF model is defined as

$$E(\boldsymbol{y}_{(t)}) = -\sum_{r\in\mathcal{R}_{(t)}} \log P\left(y_{r,(t)}|\mathcal{X}_{r,(t)}\right) + w_{\text{temp}} \sum_{r\in\mathcal{R}_{(t)}} \Psi\left(y_{r,(t)}, \widetilde{y}_{r,(t-1)}, \boldsymbol{q}_{r,(t)}, \widetilde{\boldsymbol{q}}_{r,(t)}\right) + w_{\text{spatial}} \sum_{(r,r')\in\mathcal{N}_{(t)}} \Phi\left(y_{r,(t)}, y_{r',(t)}, \mathcal{X}_{r,(t)}, \mathcal{X}_{r',(t)}\right) . \tag{6.1}$$

The labels of the superpixels are given by y_r. The first, unary term is defined as the negative logarithm of the posteriors $P_{(t)}$ obtained by I^{2}VM. The second, unary term given by the function Ψ is a temporal consistency term. Since we assume the final labeling of the superpixels to be smooth within the image, we introduce this prior knowledge by means of a data-depended Potts model in the third, binary term. The set of spatial neighbors is denoted by $\mathcal{N}_{(t)}$. The variables w_{temp} and w_{spatial} are the weights between the terms. We use graph-cut(Boykov et al., 2001) to solve for the best labeling $\widetilde{\boldsymbol{y}}_{(t)} = \operatorname{argmin}_{\boldsymbol{y}_{(t)}} E(\boldsymbol{y}_{(t)})$.

To obtain a reliable classification and sampling of new data we introduce a bounding box prior P_B. Depending on the object's position in the previous frame we define a bounding box around the slightly dilated object. I.e. , the bounding box prior P_B is expressed as a hard assignment of probability 0 to locations far away from the object's previous position. The final posterior probability is given by

$$P_{(t)} = \frac{1}{Z} P_{\mathcal{M},(t-1)} \odot P_{B,(t)} \tag{6.2}$$

with $\odot$ as the Hadamard-product and Z given as normalization factor. The unary temporal term is modeled as the similarity of the superpixel's current feature vector $\boldsymbol{q}_{r,(t)}$ and its predicted feature vector $\widetilde{\boldsymbol{q}}_{r,(t)}$ assigned via superpixel tracking. If the distance is small, it is likely that both superpixels belong to the same class, and vice versa. The similarity is defined as the cosine of the angle between the feature vectors $\psi_{r,t} = \cos\angle(\boldsymbol{q}_{r,t}, \widetilde{\boldsymbol{q}}_{r,t})$.

The binary term is modeled as the normalized length of the border $b_{rr'}$, only considered if two superpixels $\mathcal{X}_r$ and $\mathcal{X}_{r'}$ get different labels: $\phi_{rr'} = b_{rr'}/\widetilde{b}_{(t)}$. The normalization factor $\widetilde{b}_{(t)}$ is estimated in each frame t as mode of the beta-distribution of all border lengths $b_{rr'}$.

6.4.4 Experimental Setup

We start from an initial, user-defined, incomplete segmentation and train the first model $\mathcal{M}_1$ with the batch IVM. The output are posterior probabilities $P_{\mathcal{M}_1}$ of the features extracted in image I_1. The classification C yields a segmented image $\mathcal{C}_1$. In order to identify the most discriminating features, we perform cross-validation using different concatenations of features in the first image. The most discriminating features are identified using the obtained accuracy and considered for the rest of the tracking procedure.

The weighting parameters w_{temp} and w_{spatial} within the CRF model are set empirically via cross-validation using fully pixelwise annotation of the first three images of the image sequence.

For the Flower data set we analyze the added and removed import vectors in order to visually evaluate the I^{2}VM update procedure. For the SegTrack database we report the average number of false classified pixels over all frames. We compare our results to two batch-tracking methods of Tsai et al. (2011) and the online tracking method Chockalingam et al. (2009). For the Adafrag database we report the average number of false classified pixels computed over all frames of the sequence and the number of false classified pixels in each frame. We compare our results to that reported in Chockalingam et al. (2009), which use a generative approach, in particular a Gaussian mixture model within a level set framework. Furthermore, we compare our results to IVM.

For the two long image sequences of the PROST database, we compare our approach to five tracking methods (Santner et al., 2010; Babenko et al., 2011; Adam et al., 2006; Saffari et al., 2009; Klein and Cremers, 2011), that use a fixed-size bounding box. For evaluation we use the distance score, which is the average euclidean distance between the centers of the tracking rectangle and the ground truth rectangle. Our tracking rectangle is defined as the bounding box of our segmentation, generalizing the highly resolved object region. We also give the result with a fixed-size bounding box centered on the centroid of our segmentation. The pascal distance is given as the percentage of frames, where the overlapping area of the tracking rectangle and the ground truth rectangle exceeds 50 %.

6.4.5 Results and Discussion

In the following sections we discuss the results which are obtained by using I^{2}VM and the proposed tracking framework.

6.4.5.1 Flower Image Sequence

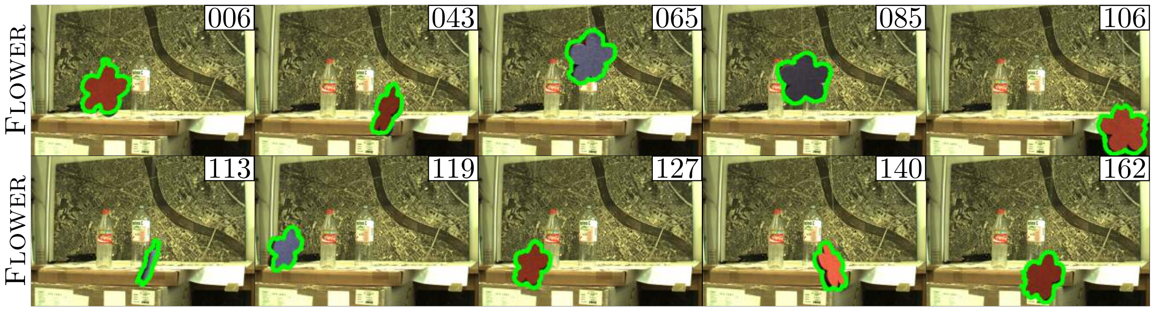


FIGURE 6.15: Representative frames (numbers t in the top right corners) of the tracking results for the FLOWER image sequence.

Figure 6.15 shows representative results of the FLOWER sequence. We use Lab, position and disparity feature within the framework. The results indicate that the flower can be tracked successfully despite changes in color, shape and position. In this sequence, a batch classifier as IVM would lost the object when the color of the object changes from red to violet, because only feature vectors for the color red are learned for the object.

TABLE 6.15: Added and removed import vectors in the FLOWER image sequence (part I).

Frame	4	21	27	32	55	59	60	65	74	75
Added IVs	🟥	🟫		🟥🟥		🟧		🟪		🟧
Removed IVs			🟫		🟥		🟥		🟪	

TABLE 6.16: Added and removed import vectors in the FLOWER image sequence (part II).

Frame	83	85	87	90	95	101	108	110	122	158
Added IVs		⬛	⬛	🟫	🟧				🟪	
Removed IVs	🟥🟧					🟩🟥	⬛	🟧		🟫

Table 6.15 and 6.16 show a part of the color features which are added and removed during the incremental learning scheme. The results show that color features which are necessary for discrimination are added into the set of import vectors. Due to the rather hardly changing disparity features and more slowly changing position feature the distribution drifts slightly and the color features can adapt to the current appearance. Also old import vectors, which are not supported by training samples any more, are removed.

6.4.5.2 SegTrack Database

Figure 6.16 shows representative segmentation results of the SegTrack database. In all image sequences we use Lab features in combination with position features. Due to the low image quality (e.g. jpeg compression artefacts) we could not use optical flow features. This was underlined by the fact that in nearly all image sequences the best parameter w_{temp} for the

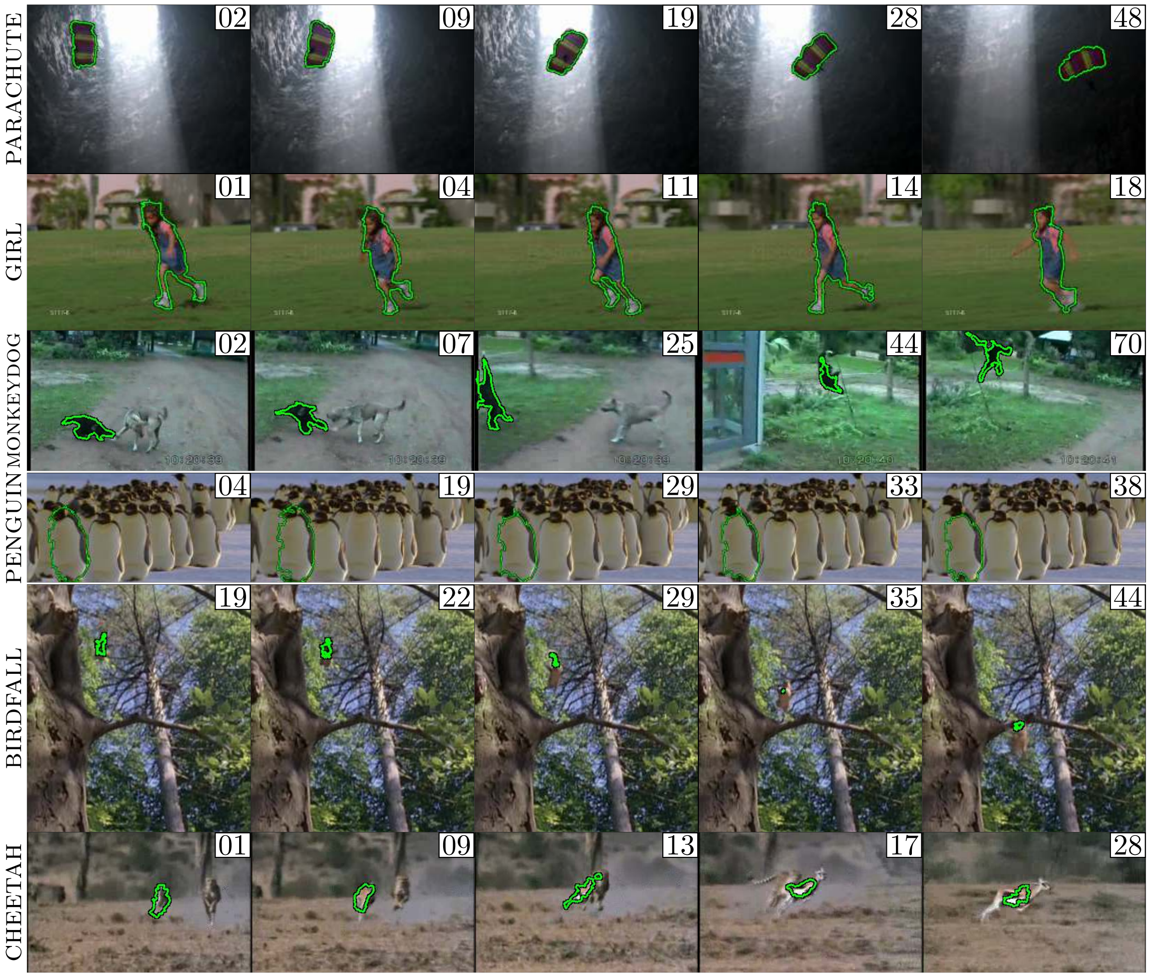


Figure 6.16: Representative frames (numbers t in the top right corners) of the tracking results for the SegTrack image sequences.

CRF is 0. The results show that in nearly all image sequences we could segment the object well. In the SegTrack-birdfall sequence the object is to small to track it correctly. It can be seen that the learner can be adapt to illumination changes (see parachute) and drastic changes in the position of the object (see monkeydog).

Table 6.17 shows the qualitative results. I^{2}VM with the proposed tracking framework yield better results than the incremental tracking approach of Chockalingam et al. (2009) and competitive results to the batch tracking framework of Tsai et al. (2011). All result were taken from Tsai et al. (2011).

6.4.5.3 Adafrag Database

Figure 6.17 shows representative segmentation results of the Adafrag database. The results indicate that all objects can be tracked successfully, even if the object disappears (see walkTree and girl). For the girl image sequence we could not obtain reliable optical flow features. Due to this and the heavy changes in the position of the object we did not use the bounding box prior in this sequence.

Figures 6.18 (a)-(d) show the normalized error per frame. In comparison to IVM, I^{2}VM results in similar or lower test errors in the Adafrag-elmo, Adafrag-walkTree and Adafrag-walkCar sequence, but in a higher error in the Adafrag-girl sequence. In the

TABLE 6.17: Scores on the SegTrack database. The score is the average number of false classified pixels per frame. The minimum error of an incremental algorithm is bold printed. Our tracking framework is compared to the batch-tracking method of Tsai et al. (2011), the batch volume graphcut as described by Tsai et al. (2011) and the incremental tracking method of Chockalingam et al. (2009). All result were taken from Tsai et al. (2011).

Data set	Batch algorithms		Incremental algorithms	
	Tsai et al. (2011)	Volume graphcut	Chockalingam et al. (2009)	I^{2}VM
PARACHUTE	235	213	502	**402**
GIRL	1304	1566	1755	**1381**
MONKEYDOG	563	726	683	**599**
PENGUIN	1705	7041	6627	**1825**
BIRDFALL	252	304	454	**444**
CHEETAH	1142	2183	1217	**1036**

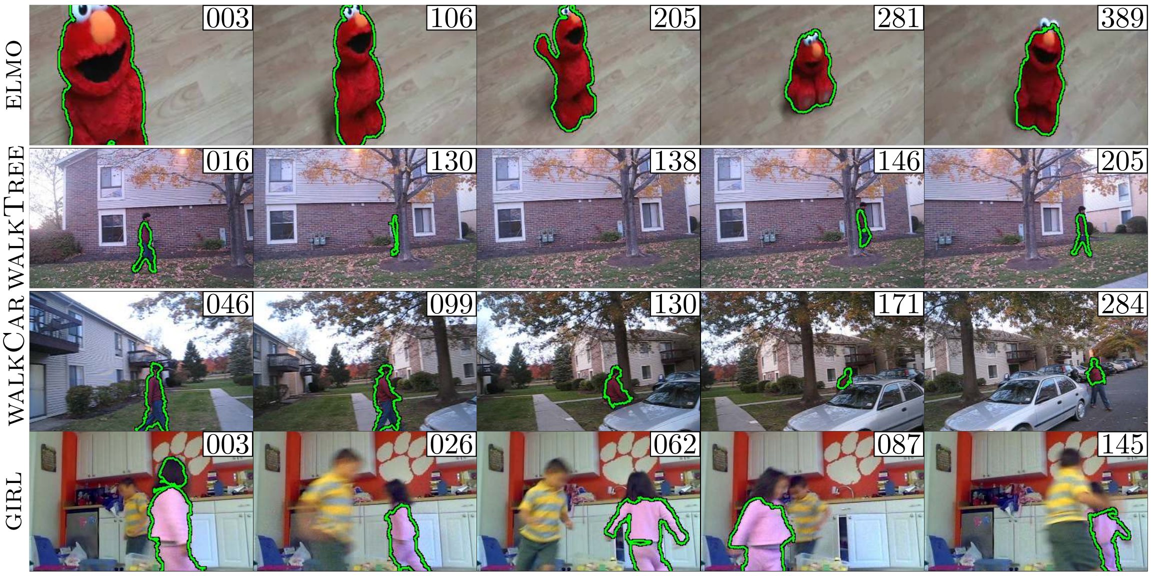
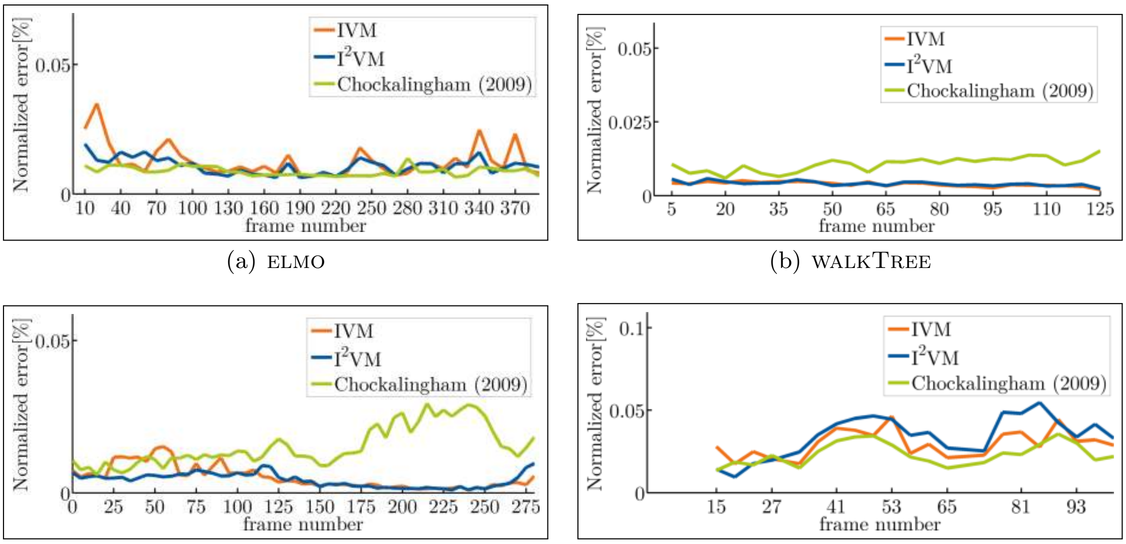


FIGURE 6.17: Representative frames (numbers t in the top right corners) of the tracking results for the evaluated sequences.

ADAFRAG-GIRL sequence the head of the girl cannot be tracked by I^{2}VM because it has a similar appearance as the background. Additionally, due to the rapid changes and consequently the erroneous optical flow we get worse results in the ADAFRAG-GIRL sequence. However, one must note that the image sequences are not characterized by sharp color changes as the FLOWER sequence and thus at least also a batch classifier is able to track the object. This comparison is therefore primarily used to show that I^{2}VM performs similarly as IVM in case of moderate changes between the frames and superior in case of severe appearance changes as within the FLOWER sequence. I^{2}VM performs comparable to the approach of Chockalingam et al. (2009) in the ADAFRAG-ELMO and ADAFRAG-WALKTREE image sequences and superior in the ADAFRAG-WALKCAR image sequence. Unfortunately, we have only annotated images until frame 125, that is, before the object disappears behind the tree. In the second part of the ADAFRAG-WALKCAR image sequence we perform very well even if the object is partly occluded.

A drawback of our proposed method can be seen in the last presented image of the



(a) elmo

(b) walkTree

(c) walkCar

(d) girl

Figure 6.18: Normalized error plots of the AdaFrag image sequences. The error is computed on each image of the sequence as the number of pixels in the image misclassified as foreground or background, normalized by the image size.

Adafrag-walkCar sequence. The man is occluded in nearly 150 frames. Consequently relevant features are deleted due to their age. Therefore, the trousers of the man could not segmented in the subsequent images. A meta-model, for example, storing relevant information is necessary to re-activate features which are necessary for discrimination.

6.4.5.4 Long Image Sequences

Table 6.18 shows the results of our proposed I^{2}VM tracking for the PROST-lemming and PROST-liquor image sequence of the PROST database in comparison to five other tracking methods,

Table 6.18 shows the results of our proposed tracking approach with I^{2}VM in comparison to five other methods that use tracking-by-detection with a fixed-size bounding box. I^{2}VM (BB) achieves a high accuracy in the PROST-lemming and PROST-liquor sequences dealing with various appearance variations, occlusions and different illumination. In most of the frames of the PROST-liquor sequence the bottle is segmented only partly, because the background surrounding the object has nearly the same appearance. Due to this, the pascal performance measure for I^{2}VM is low. The average percentage of the overlapping area of the tracking rectangle and the ground truth rectangle is 30%. Nevertheless our algorithm tracks the correct bottle during the whole time.

Figure 6.19 shows the number of import vectors over the whole image sequence. The plots show that for all sequences the number remains nearly the same and varies according the incremental and decremental learning of samples. Thus I^{2}VM is able to adapt to changes in the distribution while keeping the model efficient.

Figure 6.20 shows representative frames with the tracked objects in the long image sequences. The results show that the objects can be tracked successfully despite appearance changes of the object and the background.

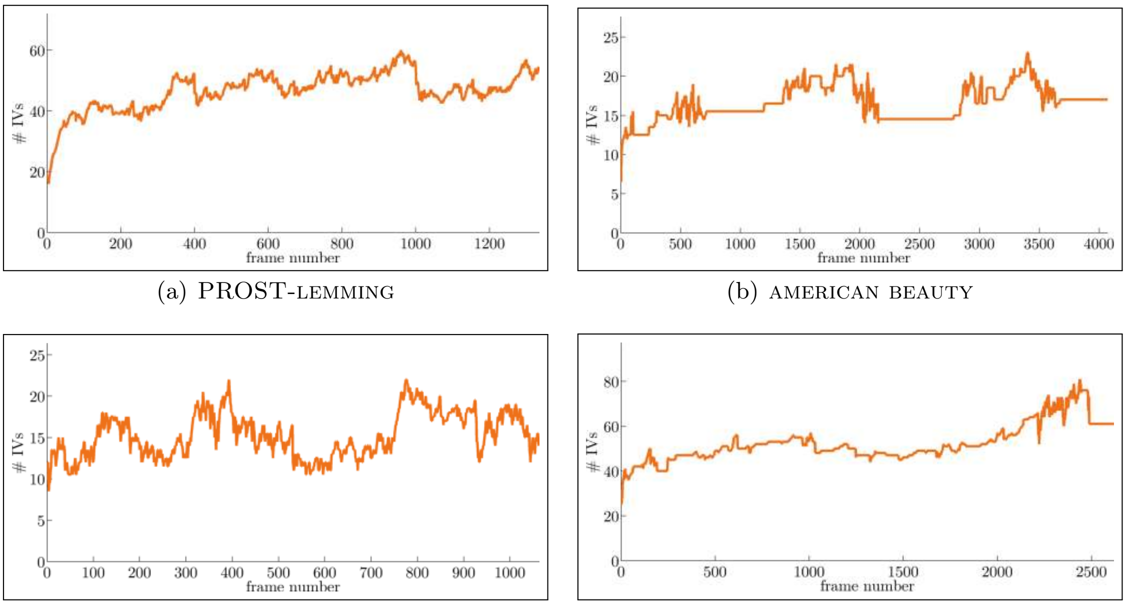
TABLE 6.18: Pascal distance and distance score in the PROST-LEMMING and PROST-LIQUOR sequences in comparison to five tracking methods. Distance score is the average euclidean distance between the centers of the tracking rectangle and the ground truth rectangle. Our tracking rectangle is the bounding box of our segmentation. In the last row (I^2VM (BB)) we evaluate a fixed-size bounding box centered on the centroid of our segmentation. The pascal distance is the percentage of frames, where the overlapping area of the tracking rectangle and the ground truth rectangle exceeds 50 %.

Algorithm	PROST-LEMMING		PROST-LIQUOR	
	pascal	distance	pascal	distance
PROST	70.5	25.1	85.4	21.5
MILTrack	83.6	14.9	20.6	165.1
FragTrack	54.9	82.8	79.9	30.7
ORF	17.2	166.3	53.6	67.3
GRAD	78.0	28.4	**91.4**	**11.9**
I^2VM	89.9	**10.6**	8.9	48.9
I^2VM (BB)	**94.8**	13.4	91.1	50.3

6.4.6 Summary

Our experiments show that our object tracking framework can handle occlusion and changes in appearance and geometry. Furthermore, it provides a detailed segmentation and not only a bounding box, making the algorithm versatile, for example for action recognition.

The computational bottlenecks of our tracking framework are the incremental learning and the superpixel tracking. Both of them can be parallelized. The current Matlab/C++ implementation takes about 1 second on an Intel(R) Dual Core with 3.0 GHz for the incremental learning. The extraction and tracking of 1000 superpixels takes about 1 second. Thus, the overall computational effort takes about 3 seconds.



(a) PROST-lemming

(b) american beauty

(c) chameleon

(d) drone

Figure 6.19: Number of IVs in the long image sequences. The number remains nearly the same over a long period and varies according the incremental and decremental learning of samples.

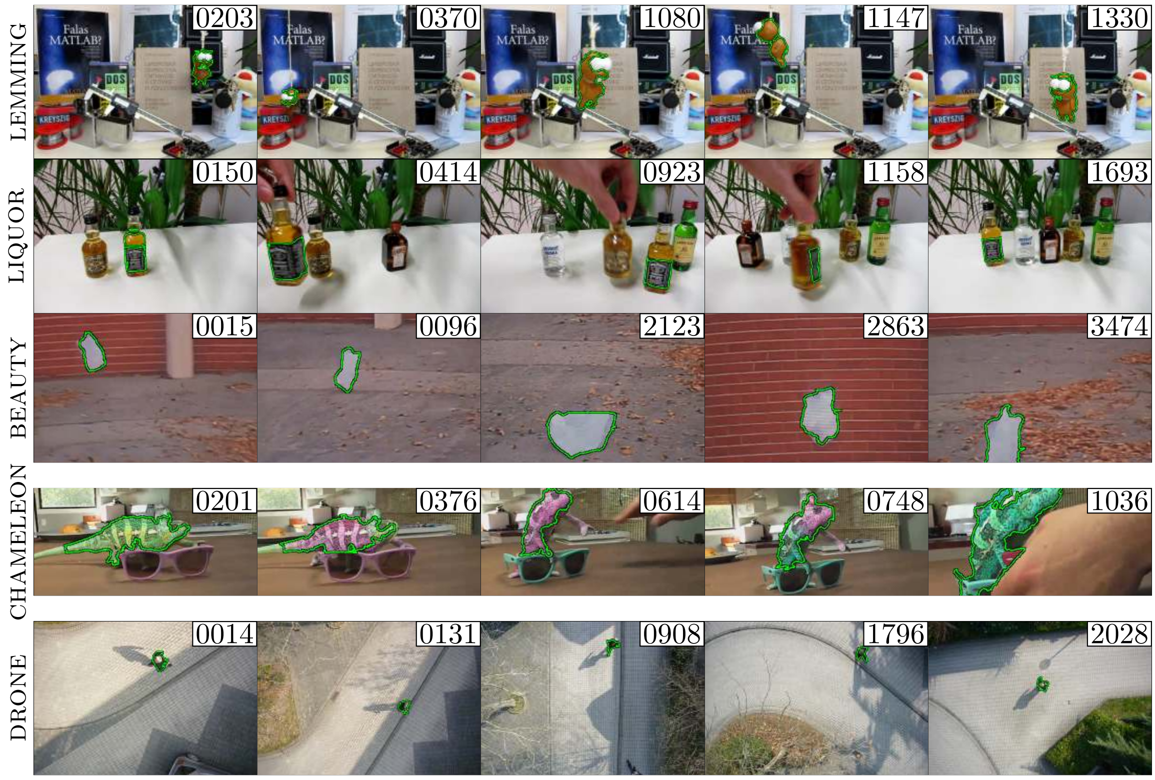


Figure 6.20: Representative frames (numbers t in the top right corners) of the tracking results in the long image sequences.

Chapter 7

Conclusion and Outlook

We introduced a sparse kernel-based probabilistic discriminative incremental learner called I^{2}VM. We formulated an incremental learning strategy for I^{2}VM which contains update steps for the addition of new samples, the removal of old samples and the estimation of the model parameters without retraining from scratch. We further showed the incorporation of new classes and features, which has been paid only little attention in the literature so far. In order to keep the model efficient we evaluated various criteria to remove non-representative data samples.

We confirmed our hypothesis that IVM and thus, also I^{2}VM, inherently have a reconstructive model component. In Section 3.2.4 we defined that classifiers with a reconstructive component are able to represent the important parts of the distribution. Important means on the one hand that these parts are necessary for discrimination and on the other hand necessary to cover the variability of all training samples. We analyzed the reconstructive component of the discriminative classifier IVM, which primarily aims to optimize the quality of the posterior probability of the classification, rather than to optimize the decision boundary. In our empirical study which analyzes the distribution of the import vectors, we showed that stationary kernels are necessary to develop a reconstructive model component and the selected import vectors uniformly cover the acceptance region of their class.

According to the requirements defined in Section 1.3 we showed that I^{2}VM have the following properties:

(a) **Competitive performance.** In our experiments we showed that I^{2}VM performs comparably to its batch learning counterpart. Due to the reconstructive component the performance of I^{2}VM is independent from the sequential order of the data samples. Thus, it can handle concept-drifts in the data distribution. Furthermore, a reconsideration of already encountered samples is not necessary, which makes the algorithm efficient.

(b) **Discriminative power.** We showed that I^{2}VM is a discriminative classifier and thus is able to separate the classes well, even if the distribution is very complex. The classifier achieved superior or competitive results to other incremental learners.

(c) **Long sequences.** The incremental learning scheme includes a step to remove data samples. With a suitable criterion to identify non-relevant data samples the I^{2}VM model can be kept efficient in order to deal with long or even infinite data streams.

(d) **Probabilities.** In Chapter 4 and Section 6.2 we showed that I^{2}VM provides reliable posterior probabilities. The posterior probability is high in non-overlapping areas with a high density of data samples, decreases in the direction of the decision boundary and obtain a value of approximately $\frac{1}{C}$ in areas with no data samples. Further, we showed that samples with high class probabilities are accurately classified, whereas relatively low class probabilities are more likely assigned to misclassified samples.

Thus, I^{2}VM can meet all the mentioned requirements.

With our experiments we have indicated that the applications using I^{2}VM are wide-ranging. We used I^{2}VM for semantic segmentation of images from the Microsoft Research Cambridge Object Recognition Image database. With this experiment we showed that I^{2}VM can deal with long sequences and a criterion can be found that can remove training samples without a loss in performance. Furthermore, we used I^{2}VM and applied self-training to classify the landcover of a large area consisting of composite remote sensing images. The images are characterized by both spatial and temporal differences. We showed that I^{2}VM is able to adapt to changes in the distribution of the training samples in order to classify a current set of test samples as well as possible. We further evaluated the algorithm for object tracking in image sequences. I^{2}VM turned out to deal with very long data streams that are characterized by concept-drifts without a loss in efficiency. Moreover, I^{2}VM also has a high discriminative power in order to distinguish the object and background. Beside these application also other areas can use this classifier.

Currently, a limitation factor is the training time which depends on the number of training samples and import vectors that are involved in the model at a specific time step. Using the current implementation the computational effort is tractable for keeping several thousands of data samples in the model but intractable for a larger number. Thus, incremental learning tasks with many relevant samples that must be kept in the model may be practically not solvable yet. However, the greedy optimization procedure can be parallelized and the number of tested candidate import vectors can be reduced significantly using knowledge about the distribution of import vectors. Moreover, the proposed incremental learner I^{2}VM is not limited to be used with the iteratively reweighted least squares procedure. Thus, further work could evaluate alternative optimization methods, e.g. quasi-Newton methods, that could lead to a more efficient procedure.

Further research could also enhance focus on the development of a one-class classifier. These classification methods focus on the estimation of a decision boundary without explicitly defining any other class. I^{2}VM may be used to tackle this task exploiting the reconstructive properties and the estimated posterior probabilities in order to define such a decision boundary. An useful extension, especially for the tracking procedure, would be the development of a meta-model storing relevant information of removed samples, see Section 6.4.5.3. For this purpose, removed import vectors could be used as representatives of these samples,. Also further information, e.g. if they were removed when they lie in the acceptance area of another class, could be useful. Summarizing, I^{2}VM turned out to be a powerful probabilistic incremental classifier bearing a high potential for further research.

Appendix A

Derivatives of the Negative Log-likelihood Function

In the following we derive the derivatives for the negative log-likelihood function that are used within the Newton-Raphson optimization framework.

A.1 Derivatives of the Posterior Probabilities P_{nc} with Respect to the Linear Function $g_{nc} = \boldsymbol{\alpha}_c^{\mathsf{T}}\mathbf{x}_n$

We have given the posterior probabilities

$$P\left(y_n = c|X; \boldsymbol{\alpha}_c\right) \doteq P_{nc} = \frac{\exp\left(g_{nc}\right)}{\sum\limits_{c'} \exp\left(g_{nc'}\right)} \tag{A.1}$$

with $\boldsymbol{g}_{n(\cdot)} = \boldsymbol{\alpha}_{(\cdot)}^{\mathsf{T}}\mathbf{x}_n$. To obtain the first derivative, we rewrite the denominator

$$\frac{\exp\left(g_{nc}\right)}{\sum\limits_{c'} \exp\left(g_{nc'}\right)} = \frac{\exp\left(g_{nc}\right)}{\sum\limits_{c'\setminus c''} \exp\left(g_{nc'}\right) + \exp\left(g_{nc''}\right)} \tag{A.2}$$

yielding the derivative for $c'' = c$

$$\begin{aligned}
\frac{\partial P_{nc}}{g_{nc}} &= \frac{\exp\left(g_{nc}\right)\left(\sum\limits_{c'\setminus c} \exp\left(g_{nc'}\right) + \exp\left(g_{nc}\right)\right) - \exp\left(g_{nc}\right)\exp\left(g_{nc}\right)}{\left(\sum\limits_{c'} \exp\left(g_{nc'}\right)\right)^2}\,, \\
&= \frac{\exp\left(g_{nc}\right)}{\sum\limits_{c'} \exp\left(g_{nc'}\right)} - \frac{\exp\left(g_{nc}\right)^2}{\left(\sum\limits_{c'} \exp\left(g_{nc'}\right)\right)^2}\,, \\
&= P_{nc}\left(1 - P_{nc}\right)\,,
\end{aligned} \tag{A.3}$$

and for $c'' \neq c$

$$
\begin{aligned}
\frac{\partial P_{nc}}{g_{nc''}} &= -\frac{\exp\left(g_{nc}\right)\exp\left(g_{nc''}\right)}{\left(\sum\limits_{c'\setminus c''}\exp\left(g_{nc'}\right)+\exp\left(g_{nc''}\right)\right)^2}\,, \\
&= -\frac{\exp\left(g_{nc}\right)\exp\left(g_{nc''}\right)}{\left(\sum\limits_{c'}\exp\left(g_{nc'}\right)\right)^2}\,, \\
&= -P_{nc}P_{nc''}\,. \qquad \text{(A.4)}
\end{aligned}
$$

The derivatives of P_{nc} with respect to $g_{nc''}$ are thus in general

$$
\frac{\partial P_{nc}}{\partial g_{nc''}} = P_{nc}\left(\delta(c,c'') - P_{nc''}\right) \qquad \text{(A.5)}
$$

with $\delta(\cdot,\cdot)$ as the the Kronecker delta, which is 1 if both inputs are equal and 0 if they are unequal.

A.2 Derivative of the Negative Log-Likelihood Function with Respect to the Parameters

The first derivate can be obtained by exploiting the chain rule given by

$$
\nabla_{\boldsymbol{\alpha}_{c'}}\mathcal{Q}_0 \doteq \frac{\partial \mathcal{Q}_0}{\partial \boldsymbol{\alpha}_{c'}} = \sum_n\sum_c \frac{\partial \mathcal{Q}_0}{\partial P_{nc}}\sum_{c''}\frac{\partial P_{nc}}{\partial g_{nc''}}\frac{\partial g_{nc''}}{\partial \boldsymbol{\alpha}_{c'}}\,. \qquad \text{(A.6)}
$$

Using (A.5) we obtain

$$
\begin{aligned}
\frac{\partial \mathcal{Q}_0}{\partial P_{nc}} &= -\frac{t_{nc}}{P_{nc}}\,, && \text{(A.7)} \\
\frac{\partial P_{nc}}{\partial g_{nc''}} &= P_{nc}\left(\delta(c,c'') - P_{nc''}\right)\,, && \text{(A.8)} \\
\frac{\partial g_{nc''}}{\partial \boldsymbol{\alpha}_{c'}} &= \frac{\partial\left(\boldsymbol{\alpha}_{c''}^{\mathsf{T}}\mathbf{x}_n\right)}{\partial \boldsymbol{\alpha}_{c'}} = \delta(c'',c')\mathbf{x}_n\,. && \text{(A.9)}
\end{aligned}
$$

Thus, the gradient $\nabla_{\boldsymbol{\alpha}_{c'}}\mathcal{Q}_0$ is given by the multiplication of (A.7)-(A.9)

$$
\nabla_{\boldsymbol{\alpha}_{c'}}\mathcal{Q}_0 = -\sum_n\sum_c\frac{t_{nc}}{P_{nc}}\sum_{c''}P_{nc}\left(\delta(c,c'') - P_{nc''}\right)\delta(c'',c')\mathbf{x}_n\,. \qquad \text{(A.10)}
$$

The second part is 0 if $c'' \neq c'$, thus we set $c'' = c'$ yielding

$$
\begin{aligned}
\nabla_{\boldsymbol{\alpha}_{c'}}\mathcal{Q}_0 &= -\sum_n\sum_c\frac{t_{nc}}{\cancel{P_{nc}}}\cancel{P_{nc}}\left(\delta(c,c') - P_{nc'}\right)\mathbf{x}_n\,, && \text{(A.11)} \\
&= -\sum_n\left(t_{cn}\delta(c,c') + t_{nc}P_{nc'}\right)\mathbf{x}_n\,, && \text{(A.12)} \\
&= -\sum_n t_{nc'}\mathbf{x}_n + \sum_n\sum_c t_{nc}P_{nc'}\mathbf{x}_n\,. && \text{(A.13)}
\end{aligned}
$$

Using the fact that $\sum\limits_{c} t_{nc} = 1$, we obtain

$$\nabla_{\boldsymbol{\alpha}_{c'}} \mathcal{Q}_0 = \sum_n \left(P_{nc'} - t_{nc'}\right) \mathbf{x}_n. \tag{A.14}$$

The second derivative $\nabla_{\boldsymbol{\alpha}_c, \boldsymbol{\alpha}_{c'}} \mathcal{Q}_0$ can be obtained using (A.8) and (A.9) yielding

$$\nabla_{\boldsymbol{\alpha}_c, \boldsymbol{\alpha}_{c'}} \mathcal{Q}_0 = \sum_n P_{nc} \left(\delta(c, c') - P_{nc'}\right) \mathbf{x}_n \mathbf{x}_n^{\mathsf{T}} . \tag{A.15}$$

Appendix B

Relation Between Newton-Raphson and Iteratively Reweighted Least Squares

The Newton-Raphson iteration procedure can be interpreted as the iteratively reweighted least squares (IRLS) optimization method by rearranging

$$\boldsymbol{\alpha}_{(t)} = \boldsymbol{\alpha}_{(t-1)} - \left(\nabla^2 \mathcal{Q}_0\left(\boldsymbol{\alpha}_{(t)}\right) + \lambda L\right)^{-1} \left(\nabla \mathcal{Q}_0\left(\boldsymbol{\alpha}_{(t)}\right) + \lambda L \boldsymbol{\alpha}_{(t-1)}\right) , \tag{B.1}$$

The matrix L is assumed to be the unit matrix, thus $L = I_{(M+1)C}$, but in order to prevent a regularization of the bias parameters w_{0c} the diagonal entries for the bias are set to zero.

$$\begin{aligned}
\boldsymbol{\alpha}_{c,(t)} &= \boldsymbol{\alpha}_{c,(t-1)} - \left(\frac{1}{N} X R_c X + \lambda L\right)^{-1} \left(\frac{1}{N} X \left(\boldsymbol{p}_c(\boldsymbol{\alpha}) - \boldsymbol{t}_c\right) + \lambda L \boldsymbol{\alpha}_{c,(t-1)}\right) \tag{B.2}\\
&= \left(\frac{1}{N} X^{\mathsf{T}} R_c X + \lambda L\right)^{-1} \left(\frac{1}{N} X^{\mathsf{T}} R_c X + \lambda L\right) \boldsymbol{\alpha}_{c,(t-1)} - \\
&\quad \left(\frac{1}{N} X^{\mathsf{T}} R_c X + \lambda L\right)^{-1} \left(\frac{1}{N} X \left(\boldsymbol{p}_c(\boldsymbol{\alpha}) - \boldsymbol{t}_c\right) + \lambda L \boldsymbol{\alpha}_{c,(t-1)}\right) \tag{B.3}\\
&= \left(\frac{1}{N} X^{\mathsf{T}} R_c X + \lambda L\right)^{-1} \tag{B.4}\\
&\quad \left[\left(\frac{1}{N} X^{\mathsf{T}} R_c X + \lambda L\right) \boldsymbol{\alpha}_{c,(t-1)} - \left(\frac{1}{N} X \left(\boldsymbol{p}_c(\boldsymbol{\alpha}) - \boldsymbol{t}_c\right) + \lambda L \boldsymbol{\alpha}_{c,(t-1)}\right)\right] \tag{B.5}\\
&= \left(\frac{1}{N} X^{\mathsf{T}} R_c X + \lambda L\right)^{-1} \tag{B.6}\\
&\quad \left[\frac{1}{N} X^{\mathsf{T}} R_c X \boldsymbol{\alpha}_{c,(t-1)} + \cancel{\lambda L \boldsymbol{\alpha}_{c,(t-1)}} - \frac{1}{N} X^{\mathsf{T}} R_c R_c^{-1} \left(\boldsymbol{p}_c(\boldsymbol{\alpha}) - \boldsymbol{t}_c\right) + \cancel{\lambda L \boldsymbol{\alpha}_{c,(t-1)}}\right] \tag{B.7}\\
&= \left(\frac{1}{N} X^{\mathsf{T}} R_c X + \lambda L\right)^{-1} \frac{1}{N} X^{\mathsf{T}} R_c \left(X \boldsymbol{\alpha}_{c,(t-1)} - R_c^{-1} \left(\boldsymbol{p}_c(\boldsymbol{\alpha}) - \boldsymbol{t}_c\right)\right) \tag{B.8}
\end{aligned}$$

Throughout the thesis, we use the IRLS procedure for the formulation of I^{2}VM.

Appendix C

Matrix Inversion with Sherman-Morrisson Woodbury formula

Within our incremental learning scheme we use the Sherman-Morrison-Woodbury formula (SMW), see Higham (2002), to efficiently update the inverse of the Hessian if the number of import vectors grow.

Given an $M_1 \times M_1$-matrix A^{-1}, the updated inverse of size $(M_1 + M_2) \times (M_1 + M_2)$ can be computed with

$$\begin{aligned}\begin{bmatrix} A & U \\ V & C \end{bmatrix}^{-1} &= \begin{bmatrix} I & A^{-1}U \\ 0 & I \end{bmatrix}^{-1} \begin{bmatrix} A & 0 \\ 0 & C - VA^{-1}U \end{bmatrix}^{-1} \begin{bmatrix} I & 0 \\ VA^{-1} & I \end{bmatrix}^{-1} \\ &= \begin{bmatrix} I & -A^{-1}U \\ 0 & I \end{bmatrix} \begin{bmatrix} A^{-1} & 0 \\ 0 & (C - VA^{-1}U)^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -VA^{-1} & I \end{bmatrix} \\ &= \begin{bmatrix} A^{-1} + A^{-1}U(C - VA^{-1}U)^{-1}VA^{-1} & -A^{-1}U(C - VA^{-1}U)^{-1} \\ -(C - VA^{-1}U)^{-1}VA^{-1} & (C - VA^{-1}U)^{-1} \end{bmatrix} . \end{aligned} \tag{C.1}$$

Thus, instead of inverting the whole $(M_1 + M_2) \times (M_1 + M_2)$-matrix, we only have to compute an inverse of size $M_2 \times M_2$. We use this update formula for the incremental update of the Hessian when adding new training vectors.

Bibliography

Achanta, R., A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk (2010). SLIC superpixels. Technical report, EPFL.

Adam, A., E. Rivlin, and I. Shimshoni (2006). Robust fragments-based tracking using the integral histogram. In *IEEE Proc. Computer Vision and Pattern Recognition*.

Artac, M., M. Jogan, and A. Leonardis (2002). Incremental PCA for on-line visual learning and recognition. In *IEEE Proc. International Conference on Pattern Recognition*.

Avidan, S. (2007). Ensemble tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence 29*(2), 261–271.

Babenko, B., M. Yang, and S. Belongie (2011). Robust object tracking with online multiple instance learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence 33*(8), 1619–1632.

Bengio, Y., J. Louradour, R. Collobert, and J. Weston (2009). Curriculum learning. In *Proc. International Conference on Machine Learning*.

Bishop, C. (2006). *Pattern Recognition and Machine Learning*. Springer New York.

Blum, A. and T. Mitchell (1998). Combining labeled and unlabeled data with co-training. In *Proc. Conference on Computational Learning Theory*.

Bodart, C., H. Eva, R. Beuchle, R. Rasi, D. Simonetti, H. Stibig, A. Brink, E. Lindquist, and F. Achard (2011). Pre-processing of a sample of multi-scene and multi-date Landsatimagery used to monitor forest cover changes over the tropics. *ISPRS Journal of Photogrammetry and Remote Sensing 66*, 555–563.

Bordes, A., L. Bottou, P. Gallinari, and J. Weston (2007). Solving multiclass support vector machines with LaRank. In *Proc. International Conference on Machine Learning*.

Borges, J., J. Bioucas-Dias, and A. Marçal (2006). Fast sparse multinomial regression applied to hyperspectral data. In *Proc. International Conference on Image Analysis and Recognition*.

Boykov, Y. and M. Jolly (2001). Interactive graph cuts for optimal boundary & region segmentation of objects in ND images. In *Proc. International Conference on Computer Vision*.

Boykov, Y., O. Veksler, and R. Zabih (2001). Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence 23*(11), 2001.

Braun, A. C., U. Weidner, and S. Hinz (2011). Support vector machines, import vector machines and relevance vector machines for hyperspectral classification – a comparison. In *Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing.*

Brefeld, U., C. Büscher, and T. Scheffer (2005). Multi-view discriminative sequential learning. In *Proc. European Conference on Machine Learning.*

Breiman, L. (1996). Bagging predictors. *Machine Learning 24*(2), 123–140.

Breiman, L. (2001). Random forests. *Machine Learning 45*(1), 5–32.

Breiman, L., J. Friedman, R. Olshen, C. Stone, L. Breiman, W. Hoeffding, R. Serfling, J. Friedman, O. Hall, P. Buhlmann, et al. (1984). Classification and regression trees. *Machine Learning 19*, 293–325.

Bruzzone, L. and D. Fernàndez Prieto (1999). An incremental-learning neural network for the classification of remote-sensing images. *Pattern Recognition Letters 20*(11), 1241–1248.

Caruana, R. and A. Niculescu-Mizil (2006). An empirical comparison of supervised learning algorithms. In *Proc. International Conference on Machine Learning.*

Cauwenberghs, G. and T. Poggio (2001). Incremental and decremental support vector machine learning. In *Proc. Advances in Neural Information Processing Systems.*

Cawley, G. and N. Talbot (2004). Efficient model selection for kernel logistic regression. In *Proc. International Conference on Pattern Recognition.*

Cawley, G., N. Talbot, and M. Girolami (2007). Sparse Multinomial Logistic Regression via Bayesian L1 Regularisation. In *Proc. Advances in Neural Information Processing Systems.*

Chambolle, A. and T. Pock (2011). A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision 40*(1), 120–145.

Chang, C. and C. Lin (2001). LIBSVM: A library for support vector machines.

Charron, C. and Y. Hicks (2010). An evolving MoG for online image sequence segmentation. In *IEEE Proc. International Conference on Image Processing.*

Chawla, N. and D. Cieslak (2006). Evaluating probability estimates from decision trees. In *American Association for Artificial Intelligence.*

Chin, T. and D. Suter (2007). Incremental kernel principal component analysis. *IEEE Transactions on Image Processing 16*(6), 1662–1674.

Chockalingam, P., N. Pradeep, and S. Birchfield (2009). Adaptive fragments-based tracking of non-rigid objects using level sets. In *IEEE Proc. International Conference on Computer Vision.*

Cihlar, J. (2000). Land cover mapping of large areas from satellites: status and research priorities. *International Journal of Remote Sensing 21*(6–7), 1093–1114.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement 20*(1), 37–46.

Cook, R. and S. Weisberg (1982). *Residuals and Influence in Regression.* Chapman and Hall New York.

Cornuéjols, A. (1993). Getting order independence in incremental learning. In *Proc. European Conference on Machine Learning.*

Cramer, J. (2003). *Logit Models from Economics and Other Fields.* Cambridge University Press.

Cristianini, N. and J. Shawe-Taylor (2000). *An Introduction to Support Vector Machines: and Other Kernel-based Learning Methods.* Cambridge University Press.

Csató, L., D. Cornford, and M. Opper (2001). Online learning of wind-field models. In *Proc. International Conference on Artificial Neural Networks*, pp. 300–307.

Dalal, N. and B. Triggs (2005). Histograms of oriented gradients for human detection. In *IEEE Proc. Computer Vision and Pattern Recognition*, pp. 886–893.

Deselaers, T., T. Gass, G. Heigold, and H. Ney (2011). Latent log-linear models for handwritten digit classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (99), 1–1.

Dietterich, T. (2002). Machine learning for sequential data: A review. *Structural, Syntactic, and Statistical Pattern Recognition 2396*, 15–30.

Donoser, M., M. Urschler, H. Riemenschneider, and H. Bischof (2011). Highly consistent sequential segmentation. In *Proc. Scandinavian Conference on Image Analysis.*

Duchenne, O., J. Audibert, R. Keriven, J. Ponce, and F. Ségonne (2008). Segmentation by transduction. In *Proc. Computer Vision and Pattern Recognition.*

Duda, R., P. Hart, and D. Stork (2001). *Pattern classification.* Wiley-Interscience.

Engelbrecht, A. and R. Brits (2001). A clustering approach to incremental learning for feedforward neural networks. In *Proc. International Joint Conference on Neural Networks.*

Engelbrecht, A. and I. Cloete (1999). Incremental learning using sensitivity analysis. In *Proc. International Joint Conference on Neural Networks.*

Falco, A., J. Udupa, and F. Miyazawa (2000). An ultra-fast user-steered image segmentation paradigm: live wire on the fly. *IEEE Transactions on Medical Imaging 19*(1), 55–62.

Fei-Fei, L., R. Fergus, and P. Perona (2007). Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding 106*(1), 59–70.

Ferrari, S. and M. Jensenius (2008). A constrained optimization approach to preserving prior knowledge during incremental training. *IEEE Transactions on Neural Networks 19*(6), 996–1009.

Franc, V., A. Zien, and B. Schlkopf (2011). Support vector machines as probabilistic models. In *Proc. International Conference on Machine Learning.*

Franklin, S. and M. Wulder (2002). Remote sensing methods in medium spatial resolution satellite data land cover classification of large areas. *Progress in Physical Geography 26*(2), 173.

Freund, Y. and R. Schapire (1996). Experiments with a new boosting algorithm. In *Proc. International Conference on Machine Learning.*

Friedman, J., T. Hastie, and R. Tibshirani (2000). Additive logistic regression: A statistical view of boosting. *The Annals of Statistics 28*(2), 337–407.

Fritz, M., B. Leibe, B. Caputo, and B. Schiele (2005). Integrating representative and discriminant models for object category detection. In *IEEE Proc. International Conference on Computer Vision.*

Fung, G. and O. Mangasarian (2002). Incremental support vector machine classification. In *Proc. SIAM International Conference on Data Mining.*

Gehrig, S., F. Eberli, and T. Meyer (2009). A real-time low-power stereo vision engine using semi-global matching. *Computer Vision Systems 5815*, 134–143.

Genton, M. (2002). Classes of kernels for machine learning: a statistics perspective. *The Journal of Machine Learning Research 2*, 299–312.

Giacco, F., C. Thiel, L. Pugliese, S. Scarpetta, and M. Marinaro (2010). Uncertainty analysis for the classification of multispectral satellite images using SVMs and SOMs. *IEEE Transactions on Geoscience and Remote Sensing 48*(10), 3769–3779.

Giraud-Carrier, C. (2000). A note on the utility of incremental learning. *AI Communications 13*(4), 215–223.

Glover, F. (1989). Tabu search-part I. *ORSA Journal on Computing 1*(3), 190–206.

Grabner, H. and H. Bischof (2006). On-line boosting and vision. In *IEEE Proc. Computer Vision and Pattern Recognition.*

Grabner, H., C. Leistner, and H. Bischof (2008). Semi-supervised on-line boosting for robust tracking. In *Proc. European Conference on Computer Vision.*

Grandvalet, Y., J. Mariéthoz, and S. Bengio (2005). Interpretation of SVMs with an application to unbalanced classification. In *Proc. Advances in Neural Information Processing Systems.*

Grossberg, S. (1988). Nonlinear neural networks: principles, mechanisms, and architectures. *Neural Networks 1*(1), 17–61.

Gu, S., Y. Zheng, and C. Tomasi (2010). Efficient visual object tracking with online nearest neighbor classifier. In *Asian Conference on Computer Vision.*

Guan, S. and S. Li (2001). Incremental learning with respect to new incoming input attributes. *Neural Processing Letters 14*(3), 241–260.

Hastie, T., R. Tibshirani, and J. Friedman (2009). *The Elements of Statistical Learning*, Volume 1. Springer Series in Statistics.

Hérault, R. and Y. Grandvalet (2007). Sparse probabilistic classifiers. In *Proc. International Conference on Machine Learning.*

Higham, N. (2002). *Accuracy and Stability of Numerical Algorithms.* Society for Industrial Mathematics.

Hirschmüller, H. (2005). Accurate and efficient stereo processing by semi-global matching and mutual information. In *IEEE Proc. Computer Vision and Pattern Recognition.*

Ho, T. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence 20*(8), 832–844.

Hoens, T., R. Polikar, and N. Chawla (2012). Learning from streaming data with concept drift and imbalance: an overview. *Progress in Artificial Intelligence*. To appear.

Holmes, G., R. Kirkby, and D. Bainbridge (2004). Batch-incremental learning for mining data streams. Technical report, University of Waikato, Department of Computer science.

Hosmer, D. and S. Lemeshow (2000). *Applied Logistic Regression.* Wiley-Interscience.

Hulten, G. and P. Domingos (2002). Mining complex models from arbitrarily large databases in constant time. In *Proc. International Conference on Knowledge Discovery and Data Mining.*

Iglesias, J., E. Konukoglu, A. Montillo, Z. Tu, and A. Criminisi (2011). Combining generative and discriminative models for semantic segmentation of CT scans via active learning. In *Proc. International Conference on Information Processing in Medical Imaging*, pp. 25–36. Springer.

Jaber, G., A. Cornuéjols, and P. Tarroux (2011). Predicting concept changes using a committee of experts. In *Proc. Neural Information Processing Systems*, pp. 580–588. Springer.

Jain, S., S. Lange, and S. Zilles (2006). Towards a better understanding of incremental learning. In *Proc. International Conference on Algorithmic Learning Theory.*

Jiang, Z., Z. Lin, and L. Davis (2011). Learning a discriminative dictionary for sparse coding via label consistent K-SVD. In *IEEE Proc. Conference on Computer Vision and Pattern Recognition.*

Kang, H. and S. Shin (2002). Enhanced lane: interactive image segmentation by incremental path map construction. *Graphical Models 64*(5), 282–303.

Karasuyama, M. and I. Takeuchi (2010). Multiple incremental decremental learning of support vector machines. *IEEE Transactions on Neural Networks 21*(7), 1048–1059.

Karsmakers, P., K. Pelckmans, and J. Suykens (2007). Multi-class kernel logistic regression: a fixed-size implementation. In *Proc. of the International Joint Conference on Neural Networks.*

Keerthi, S., K. Duan, S. Shevade, and A. Poo (2005). A fast dual algorithm for kernel logistic regression. *Machine Learning 61*(1), 151–165.

Kim, T., S. Wong, B. Stenger, J. Kittler, and R. Cipolla (2007). Incremental linear discriminant analysis using sufficient spanning set approximations. In *IEEE Proc. Conference on Computer Vision and Pattern Recognition.*

Kivinen, J., A. Smola, and R. Williamson (2004). Online learning with kernels. *IEEE Transactions on Signal Processing 52*(8), 2165–2176.

Klein, D. and A. Cremers (2011). Boosting scalable gradient features for adaptive real-time tracking. In *IEEE Proc. International Conference on Robotics and Automation.*

Klinkenberg, R. and T. Joachims (2000). Detecting concept drift with support vector machines. In *Proc. International Conference on Machine Learning.*

Knorn, J., A. Rabe, V. Radeloff, T. Kuemmerle, J. Kozak, and P. Hostert (2009). Land cover mapping of large areas using chain classification of neighboring landsat satellite images. *Remote Sensing of Environment 113*(5), 957–964.

Krallinger, M., A. Morgan, L. Smith, F. Leitner, L. Tanabe, J. Wilbur, L. Hirschman, and A. Valencia (2008). Evaluation of text-mining systems for biology: overview of the second biocreative community challenge. *Genome Biology 9*(2), S1.

Krishnapuram, B., L. Carin, M. Figueiredo, and A. Hartemink (2005). Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence 27*(6), 957–968.

Kristan, M. and A. Leonardis (2010). Online discriminative kernel density estimation. In *Proc. International Conference on Pattern Recognition.*

Kumar, S. and M. Hebert (2006). Discriminative random fields. In *Proc. International Journal of Computer Vision.*

Lafferty, J. D., A. McCallum, and F. C. N. Pereira (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. International Conference on Machine Learning.*

Landis, J. and G. Koch (1977). The measurement of observer agreement for categorical data. *Biometrics 33*(1), 159–174.

Lange, S. and S. Zilles (2003). Formal models of incremental learning and their analysis. In *Proc. International Joint Conference on Neural Networks.*

Langley, P. (1995). Order effects in incremental learning. *Learning in Humans and Machines: Towards an Interdisciplinary Learning Science 136*, 137.

Lasserre, J., C. Bishop, and T. Minka (2006). Principled hybrids of generative and discriminative models. In *IEEE Proc. Conference on Computer Vision and Pattern Recognition.*

Lempitsky, V., P. Kohli, C. Rother, and T. Sharp (2009). Image segmentation with a bounding box prior. In *IEEE Proc. International Conference on Computer Vision.*

Leslie, C., E. Eskin, and W. Noble (2002). The spectrum kernel: A string kernel for SVM protein classification. In *Proc. of the Pacific Symposium on Biocomputing.*

Levinshtein, A., C. Sminchisescu, and S. Dickinson (2010). Spatiotemporal closure. In *Proc. Asian Conference on Computer Vision.*

Levinshtein, A., A. Stere, K. Kutulakos, D. Fleet, S. Dickinson, and K. Siddiqi (2009). Turbopixels: Fast superpixels using geometric flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence 31*(12), 2290–2297.

Li, B., X. Xue, and J. Fan (2007). A robust incremental learning framework for accurate skin region segmentation in color images. *Pattern Recognition 40*(12), 3621–3632.

Li, F. and Y. Yang (2003). A loss function analysis for classification methods in text categorization. In *Proc. International Conference on Machine Learning*, Volume 20, pp. 472.

Li, J., J. Bioucas-Dias, and A. Plaza (2011). Hyperspectral image segmentation using a new bayesian approach with active learning. *IEEE Transactions on Geoscience and Remote Sensing 49*(99), 1–14.

Li, L. and L. Fei-Fei (2010). Optimol: Automatic online picture collection via incremental model learning. *International Journal of Computer Vision 88*(2), 147–168.

Lim, J., D. Ross, R. Lin, and M. Yang (2004). Incremental learning for visual tracking. In *Proc. Advances in Neural Information Processing Systems.*

Littau, D. and D. Boley (2009). Clustering very large data sets using a low memory matrix factored representation. *Computational Intelligence 25*(2), 114–135.

Liu, Z., F. Jiang, G. Tian, S. Wang, F. Sato, S. Meltzer, and M. Tan (2007). Sparse logistic regression with lp penalty for biomarker identification. *Statistical Applications in Genetics and Molecular Biology 6*(1), 6.

Mairal, J., F. Bach, J. Ponce, and G. Sapiro (2009). Online dictionary learning for sparse coding. In *Proc. International Conference on Machine Learning.*

Mairal, J., F. Bach, J. Ponce, G. Sapiro, and A. Zisserman (2008). Supervised dictionary learning. In *Proc. Computer Vision and Pattern Recognition.*

Maloof, M. and R. Michalski (2000). Selecting examples for partial memory learning. *Machine Learning 41*(1), 27–52.

McCallum, A., C. Pal, G. Druck, and X. Wang (2006). Multi-conditional learning: Generative/discriminative training for clustering and classification. In *Proc. International Conference on Artificial Intelligence.*

McKusick, K. and P. Langley (1991). Constraints on tree structure in concept formation. In *Proc. International Joint Conference on Artificial Intelligence.*

Mitchell, T. M. (1997). *Machine Learning.* McGraw-Hill.

Monteleoni, C. and M. Kaariainen (2007). Practical online active learning for classification. In *IEEE Proc. Conference on Computer Vision and Pattern Recognition.*

Monteleoni, C., G. A. Schmidt, S. Saroha, and E. Asplund (2011). Tracking climate models. *Statistical Analysis and Data Mining 4*, 372–392.

Mortensen, E. and W. Barrett (1995). Intelligent scissors for image composition. In *Proc. Conference on Computer Graphics and Interactive Techniques.*

Mortensen, E. and W. Barrett (1998). Interactive segmentation with intelligent scissors. *Graphical Models and Image Processing 60*(5), 349–384.

Neal, R. (1996). *Bayesian Learning for Neural Networks.* Springer Verlag.

Nguyen-Tuong, D. and J. Peters (2010). Incremental sparsification for real-time online model learning. In *Proc. International Conference on Artifical Intelligence and Statistics.*

Niculescu-Mizil, A. and R. Caruana (2005). Predicting good probabilities with supervised learning. In *Proc. International Conference on Machine Learning.*

Ozawa, S., S. Toh, S. Abe, S. Pang, and N. Kasabov (2005). Incremental learning for online face recognition. In *IEEE Proc. International Joint Conference on Neural Networks.*

Pal, N. and S. Pal (1993). A review on image segmentation techniques. *Pattern Recognition 26*(9), 1277–1294.

Pang, S., S. Ozawa, and N. Kasabov (2005). Incremental linear discriminant analysis for classification of data streams. *Transactions on Systems, Man, and Cybernetics 35*(5), 905–914.

Pekkarinen, A., L. Reithmaier, and P. Strobl (2009). Pan-European forest/non-forest mapping with landsat ETM+ and CORINE land cover 2000 data. *ISPRS Journal of Photogrammetry and Remote Sensing 64*(2), 171–183.

Platt, J., N. Cristianini, and J. Shawe-Taylor (2000). Large margin DAGs for multiclass classification. *Advances in Neural Information Processing Systems 12*(3), 547–553.

Polikar, R., L. Upda, S. Upda, and V. Honavar (2001). Learn++: An incremental learning algorithm for supervised neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 31*(4), 497–508.

Provost, F. and P. Domingos (2003). Tree induction for probability-based ranking. *Machine Learning 52*(3), 199–215.

Raina, R., Y. Shen, A. Ng, and A. McCallum (2003). Classification with hybrid generative/discriminative models. In *Proc. Advances in Neural Information Processing Systems*, Volume 16.

Ramirez, I., P. Sprechmann, and G. Sapiro (2010). Classification and clustering via dictionary learning with structured incoherence and shared features. In *IEEE Proc. Conference on Computer Vision and Pattern Recognition.*

Raup, B., A. Kääb, J. Kargel, M. Bishop, G. Hamilton, E. Lee, F. Paul, F. Rau, D. Soltesz, S. Khalsa, et al. (2007). Remote sensing and GIS technology in the global land ice measurements from space (GLIMS) project. *Computers & Geosciences 33*(1), 104–125.

Reinke, R. and R. Michalski (1988). Incremental learning of concept descriptions: A method and experimental results. *Machine Intelligence 11*, 263–288.

Ren, X. and J. Malik (2007). Tracking as repeated figure/ground segmentation. In *IEEE Proc. Conference on Computer Vision and Pattern Recognition.*

Richards, J. (2005). Analysis of remotely sensed data: The formative decades and the future. *IEEE Transactions on Geoscience and Remote Sensing 43*(3), 422–432.

Ripley, B. (2008). *Pattern Recognition and Neural Networks.* Cambridge University Press.

Rogowska, J. (2000). *Handbook of Medical Imaging: Processing and Analysis Management*, Chapter 5: Overview and Fundamentals of Medical Image Segmentation, pp. 69–85. Academic Press.

Roscher, R., W. Förstner, and B. Waske (2012). I^{2}VM: Incremental import vector machines. *Image and Vision Computing 30*(4–5), 263–278.

Roscher, R., J. Siegemund, F. Schindler, and W. Förstner (2012). Object tracking by segmentation using incremental import vector machines. Technical report, Institute of Geodesy and Geoinformation, Department of Photogrammetry.

Roscher, R., B. Waske, and W. Förstner (2011). Incremental import vector machines for large area land cover classification. In *IEEE/ISPRS Proc. Workshop on Computer Vision for Remote Sensing of the Environment.*

Roscher, R., B. Waske, and W. Förstner (2012). Incremental import vector machines for classifying hyperspectral data. *Transactions on Geoscience and Remote Sensing 9*(99), 1–11.

Ross, D., J. Lim, and M. Yang (2004). Adaptive probabilistic visual tracking with incremental subspace update. In *Proc. European Conference on Computer Vision*.

Roth, V. (2001). Probabilistic discriminative kernel classifiers for multi-class problems. In *Proc. DAGM-Symposium on Pattern Recognition*.

Rother, C., V. Kolmogorov, and A. Blake (2004). Grabcut: Interactive foreground extraction using iterated graph cuts. *Transactions on Graphics 23*(3), 309–314.

Rüping, S. (2001). Incremental learning with support vector machines. In *IEEE Proc. International Conference on Data Mining*.

Rüping, S. (2004). A simple method for estimating conditional probabilities for svms. Technical report, Technische Universität Dortmund, Sonderforschungsbereich 475: Komplexitätsreduktion in multivariaten Datenstrukturen.

Saffari, A., C. Leistner, J. Santner, M. Godec, and H. Bischof (2009). On-line random forests. In *Proc. International Conference on Computer Vision, ICCV Workshops*.

Santner, J., C. Leistner, A. Saffari, T. Pock, and H. Bischof (2010). PROST: Parallel robust online simple tracking. In *IEEE Proc. Conference on Computer Vision and Pattern Recognition*.

Schölkopf, B. and A. Smola (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press.

Scholz, M. and R. Klinkenberg (2007). Boosting classifiers for drifting concepts. *Intelligent Data Analysis 11*(1), 3–28.

Seewald, A. K. (2005). Digits–a dataset for handwritten digit recognition. Technical report, Austrian Research Institut for Artificial Intelligence.

Settles, B. (2009). Active learning literature survey. Technical report, University of Wisconsin-Madison, Computer Sciences Technical Report 1648.

Shankar, B. (2007). Novel classification and segmentation techniques with application to remotely sensed images. *Transactions on Rough Sets VII 7*, 295–380.

Shi, J. and J. Malik (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*(8), 888–905.

Skocaj, D. and A. Leonardis (2003). Weighted and robust incremental method for subspace learning. In *IEEE Proc. International Conference on Computer Vision*.

Skocaj, D., M. Uray, A. Leonardis, and H. Bischof (2006). Why to combine reconstructive and discriminative information for incremental subspace learning. In *Proc. Computer Vision Winter Workshop*.

Sollich, P. (2002). Bayesian methods for support vector machines: Evidence and predictive class probabilities. *Machine Learning 46*(1), 21–52.

Stow, D., A. Hope, D. McGuire, D. Verbyla, J. Gamon, F. Huemmrich, S. Houston, C. Racine, M. Sturm, K. Tape, et al. (2004). Remote sensing of vegetation and land-cover change in arctic tundra ecosystems. *Remote Sensing of Environment 89*(3), 281–308.

Syed, N., H. Liu, and K. Sung (1999). Handling concept drifts in incremental learning with support vector machines. In *Proc. International Conference on Knowledge Discovery and Data Mining.*

Talaveral, L. and J. Roure (1998). A buffering strategy to avoid ordering effects in clustering. In *European Conference on Machine Learning.*

Tang, F., S. Brennan, Q. Zhao, and H. Tao (2007). Co-tracking using semi-supervised support vector machines. In *IEEE Proc. International Conference on Computer Vision.*

Tipping, M. (2001). Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research 1*(1), 211–244.

Tsai, D., M. Flagg, A. Nakazawa, and J. Rehg (2011). Motion coherent tracking using multi-label MRF optimization. *International Journal of Computer Vision*, 1–13.

Tu, Z. (2007). Learning generative models via discriminative approaches. In *IEEE Proc. Conference on Computer Vision and Pattern Recognition.*

Tuia, D., F. Ratle, F. Pacifici, M. Kanevski, and W. Emery (2009). Active learning methods for remote sensing image classification. *IEEE Transactions on Geoscience and Remote Sensing 47*(7), 2218–2232.

Turner, R. (1990). Long-term vegetation change at a fully protected sonoran desert site. *Ecology 71*(2), 464–477.

Uray, M., D. Skocaj, P. M. Roth, H. Bischof, and A. Leonardis (2007). Incremental LDA learning by combining reconstructive and discriminative approaches. In *Proc. British Machine Vision Conference.*

Vachkov, G. (2010). Online incremental image classification by use of human assisted fuzzy similarity. In *IEEE International Conference on Information and Automation.*

Vapnik, V. (2000). *The Nature of Statistical Learning Theory.* Springer Verlag.

Viola, P. and M. Jones (2004). Robust real-time face detection. *International Journal of Computer Vision 57*(2), 137–154.

Walker, W., C. Stickler, J. Kellndorfer, K. Kirsch, and D. Nepstad (2010). Large-area classification and mapping of forest and land cover in the brazilian amazon: A comparative analysis of alos/palsar and landsat data sources. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 3*(4), 594–604.

Wang, S., H. Lu, F. Yang, and M. Yang (2011). Superpixel tracking. In *IEEE Proc. International Conference on Computer Vision.*

Wenzel, S. and L. Hotz (2010). The role of sequences for incremental learning. In *Proc. of the International Conference on Agents and Artificial Intelligence.*

Winn, J., A. Criminisi, and T. Minka (2005). Object categorization by learned universal visual dictionary. In *IEEE Proc. International Conference on Computer Vision.*

Wismüller, A., F. Vietze, J. Behrends, A. Meyer-Baese, M. Reiser, and H. Ritter (2004). Fully automated biomedical image segmentation by self-organized model adaptation. *Neural Networks 17*(8-9), 1327–1344.

Witten, I., E. Frank, and M. Hall (2011). *Data Mining: Practical Machine Learning Tools and Techniques.* Morgan Kaufmann.

Xue, J. and D. Titterington (2010). On the generative-discriminative tradeoff approach: Interpretation, asymptotic efficiency and classification performance. *Computational Statistics & Data Analysis 54*(2), 438–451.

Zadrozny, B. and C. Elkan (2001). Learning and making decisions when costs and probabilities are both unknown. In *Proc. International Conference on Knowledge Discovery and Data Mining.*

Zhang, B. (1994). Accelerated learning by active example selection. *International Journal of Neural Systems 5*(1), 67–76.

Zhang, L. and Q. Ji (2011). A bayesian network model for automatic and interactive image segmentation. *IEEE Transactions on Image Processing* (99), 1–1.

Zhang, Z., G. Dai, and M. Jordan (2011). Bayesian generalized kernel mixed models. *The Journal of Machine Learning Research 12*, 111–139.

Zheng, J., H. Yu, F. Shen, and J. Zhao (2010). An online incremental learning support vector machine for large-scale data. In *Proc. International Conference on Artificial Neural Networks.*

Zhou, Z. and Z. Chen (2002). Hybrid decision tree. *Knowledge-based Systems 15*(8), 515–528.

Zhu, J. and T. Hastie (2005). Kernel logistic regression and the import vector machine. *Computational and Graphical Statistics 14*(1), 185–205.