Boosting Methods for Automatic Segmentation of Focal Liver Lesions

Boosting-Verfahren zur automatischen Segmentierung fokaler Leberläsionen

Der Technischen Fakultät
der Friedrich-Alexander-Universität
Erlangen-Nürnberg

zur

Erlangung des Doktorgrades Dr.-Ing.

vorgelegt von

Arne Militzer
aus
Heilbronn

Als Dissertation genehmigt
von der Technischen Fakultät
der Friedrich-Alexander-Universität Erlangen-Nürnberg

Tag der mündlichen Prüfung:	16.06.2014
Vorsitzende des Promotionsorgans:	Prof. Dr.-Ing. habil. Marion Merklein
Gutachter:	Prof. Dr.-Ing. Joachim Hornegger
	Prof. Dr. Philippe C. Cattin

Abstract

Over the past decades, huge progress has been made in treatment of cancer, decreasing fatality rates despite a growing number of cases. Technical achievements had a big share in this development.

With modern image acquisition techniques, most types of tumors can be made visible. Automatic processing of these images to support diagnosis and therapy, on the other hand, is still very basic. Marking lesions for volume measurements, intervention planning or tracking over time requires a lot of manual interaction, which is both tedious and error prone.

The work at hand therefore aims at providing tools for the automatic segmentation of liver lesions. A system is presented that receives a contrast enhanced CT image of the liver as input and, after several preprocessing steps, decides for each image voxel inside the liver whether it belongs to a tumor or not. That way, tumors are not only detected in the image but also precisely delineated in three dimensions. For the decision step, which is the main target of this thesis, we adopted the recently proposed Probabilistic Boosting Tree. In an offline learning phase, this classifier is trained using a number of example images. After training, it can process new and previously unseen images.

Such automatic segmentation systems are particularly valuable when it comes to monitoring tumors of a patient over a longer period of time. Therefore, we propose a method for learning a prior model to improve segmentation accuracy for such follow-up examinations. It is learned from a number of series of CT images, where each series contains images of one patient. Two different ways of incorporating the model into the segmentation system are investigated. When acquiring an image of a patient, the system can use the model to calculate a patient specific lesion prior from images of the same patient acquired earlier and thus guide the segmentation in the current image.

The validity of this approach is shown in a set of experiments on clinical images. When comparing the points of 90% sensitivity in these experiments, incorporating the prior improved the precision of the segmentation from 82.7% to 91.9%. This corresponds to a reduction of the number of false positive voxels per true positive voxel by 57.8%.

Finally, we address the issue of long processing times of classification based segmentation systems. During training, the Probabilistic Boosting Tree builds up a hierarchy of AdaBoost classifiers. In order to speed up classification during application phase, we modify this hierarchy so that simpler and thus faster AdaBoost classifiers are used in higher levels. To this end, we introduce a cost term into AdaBoost training that trades off discriminative power and computational complexity during feature selection. That way the optimization process can be guided to build less complex classifiers for higher levels of the tree and more complex and thus stronger ones for deeper levels. Results of an experimental evaluation on clinical images are presented, which show that this mechanism can reduce the overall cost during application phase by up to 76% without degrading classification accuracy. It is also shown that this mechanism could be used to optimize arbitrary secondary conditions during AdaBoost training.

Kurzfassung

In der Krebstherapie gab es in den vergangenen Jahrzehnten große Fortschritte zu verzeichnen. Während die Zahl der Krebsfälle weiter ansteigt, konnte die Sterblichkeit verringert werden. Einen großen Anteil an dieser Entwicklung hatte der technische Fortschritt.

Bildgebende Verfahren können heute die meisten Tumoren sichtbar machen. Die automatische Verarbeitung dieser Bilder für Diagnose und Therapie dagegen beschränkt sich weiterhin zumeist auf sehr einfache Verfahren. Läsionen können nur mit viel Handarbeit für Volumenbestimmung, Operationsplanung, oder zeitliche Überwachung eingezeichnet werden. Dieses Vorgehen ist nicht nur sehr mühsam sondern auch anfällig für Fehler.

Die vorliegende Arbeit soll deshalb neue Methoden für die automatische Segmentierung von Leberläsionen aufzeigen. Ein System wird vorgestellt, das in einem kontrastverstärkten CT-Bild der Leber nach einigen Vorverarbeitungsschritten für jeden Bildpunkt innerhalb der Leber entscheidet, ob er Teil eines Tumors ist oder nicht. So werden die Tumoren nicht nur detektiert sondern gleichzeitig auch in allen drei Raumrichtungen vom umliegenden Gewebe abgegrenzt. Der Entscheidungsschritt macht dabei den Kern dieser Arbeit aus. Als Klassifikator wurde hierfür der Probabilistic Boosting Tree gewählt. Er wird in einer separaten Lernphase vor seinem eigentlichen Einsatz anhand einer Reihe von Beispielbildern trainiert. Nach erfolgreichem Training kann er in der Anwendungsphase auch zuvor nicht gesehene Bilddaten verarbeiten.

Derartige automatische Segmentierungsverfahren sind besonders dann hilfreich, wenn Tumoren über einen längeren Zeitraum überwacht werden sollen. Wir präsentieren daher an dieser Stelle ein Verfahren, mit dem ein a priori Modell für Tumorwahrscheinlichkeiten erstellt werden kann. Das Modell wird aus einer Reihe von zeitlichen Serien von CT-Bildern gewonnen, wobei eine Serie jeweils Bilder eines Patienten enthält. Ziel ist es, die Qualität der Segmentierungsergebnisse für Folgeuntersuchungen zu verbessern. Es werden zwei verschiedene Methoden untersucht, wie das Modell in das bestehende Segmentierungssystem zu integrieren ist. Anschließend kann das System die Segmentierung bei einer Folgeuntersuchung steuern, indem es mit Hilfe des Modells aus früheren Aufnahmen desselben Patienten berechnet, wo Läsionen zu erwarten sind.

Die Wirksamkeit dieses Vorgehens wird anhand einer Reihe von Experimenten mit klinischen Aufnahmen belegt. Vergleicht man in diesen Experimenten jeweils den Punkt, an dem 90% der Tumorpunkte erkannt wurden, stellt man fest, dass sich der positive Vorhersagewert durch das a priori Modell von 82,7% auf 91,9% verbessert. Für die Zahl der falsch positiv klassifizierten Voxel je korrekt positiv klassifiziertem Voxel entspricht das einer Verbesserung um 57,8%.

Schließlich widmen wir uns der Problematik der Rechenzeit von klassifikationsbasierten Segmentierungsverfahren. In der Lernphase baut der Probabilistic Boosting Tree eine Hierarchie von AdaBoost-Klassifikatoren auf. Um die Klassifikation im Produktiveinsatz zu beschleunigen verändern wir diese Hierarchie dahingehend, dass auf den oberen Stufen einfachere und damit schnellere AdaBoost-Klassifikatoren verwendet werden. Zu diesem Zweck wird ein Kostenfaktor in das Lernverfahren für AdaBoost eingeführt, der während der Merkmalsauswahl die Komplexität eines Merkmals gegen seinen Nutzen für die Entscheidung abwägt. Auf diese Weise kann das Lernverfahren gezwungen werden, für die oberen Ebenen der Hierarchie einfachere und für die tieferen Ebenen komplexere Klassifikatoren zu erzeugen. Die Ergebnisse einer experimentellen Auswertung mit klinischen

Bildern belegen, dass diese Methode die Gesamtkosten der Entscheidung in der Anwendungsphase um bis zu 76% verringern kann ohne die Genauigkeit der Segmentierung zu beeinträchtigen. Des Weiteren wird in den Experimenten gezeigt, dass AdaBoost auf diese Art nicht nur die Laufzeit sondern beliebige Nebenbedingungen optimieren kann.

Acknowledgment

Writing a thesis is the often painful final step of a PhD project. But even getting to this point requires already so much more than a few good ideas. It takes several years of time, dedication and endurance, a good portion of luck, and lots of support by your environment. This thesis is therefore not the work of me alone, but of a large number of people who contributed to it directly or indirectly. In this section I want to take the opportunity and express my deepest gratitude to all of them.

Without him, all this would not have happened, so I want to first thank my supervisor, Prof. Dr.-Ing. Joachim Hornegger, head of the Pattern Recognition Lab at Friedrich-Alexander-University Erlangen-Nürnberg. He not only accepted but welcomed me as a PhD student. He believed in me, and supported and encouraged me over the past few years whenever possible.

The entire project was initiated and financially and otherwise supported by Siemens AG, Healthcare Sector, in Forchheim. It was started by Daniel Rinck, team manager at that time, and, together with the team, later handed over to Dr. Michael Scheuering. Initially, their colleague Dr. Michael Sühling supervised me and the project at Siemens. When he left the team, Dr. Christian Tietjen came into the topic. Though from an entirely different field, he took the challenge and really made an effort. I am deeply grateful for the lengths he went to provide me feedback, challenge my findings and ideas and help me in any way possible.

Of the other PhD students at Siemens at the time I want to specifically mention Dr. Jens Kaftan, Dr. Thomas Beck, and Andreas Wimmer. We shared not only our office space but also our knowledge – technical and otherwise –, many hours of discussions and experiences on all levels. Jens was always particularly open for discussions about image processing tools, new publications, or about the intricacies of boosting and ROC analysis. I owe him for deepening my understanding of these topics — often by posing seemingly simple questions I could not answer. Thomas provided as much input and moral support. He stayed with me until the end and became my closest friend over those years. Andreas not only provided very useful image processing modules but also connected my two offices, mentoring me when I started at the Pattern Recognition Lab.

At the Lab, most of the time I shared my office with Dr. Martin Spiegel, Dr. Florian Jäger and Eva Kollorz. We had a great atmosphere and lots of fruitful discussions. They also formed a good part of the segmentation colloquium, where I was challenged to present and defend my latest results on a regular basis. Dr. Dirk Kolb was also part of the colloquium. With him I probably had the most intense discussions about the nature and behavior of boosting methods.

Generally the time at the Pattern Recognition Lab and Siemens was special for me in more than one way. Some people strictly separate between friends and colleagues. During these years I found a number of very good friends among my colleagues in both locations.

Last, but not least, I want to thank my family for being so supportive, not only during my PhD project but during my entire school and university careers. My parents always believed in me and had an open ear whenever I needed it. My wife Verena probably had her doubts sometimes, but nonetheless she never questioned my decision and summoned

up a patience I never considered possible. She was there whenever I needed her and I will never forget the sacrifices she made.

Arne Militzer

Contents

Chapter 1 Introduction 1

1.1 X-Ray Computed Tomography Image Acquisition . . . 2
1.2 Anatomy and Physiology of the Liver . . . 3
1.3 Liver Tumors . . . 5
1.3.1 CT Imaging . . . 6
1.3.2 Therapy Options . . . 7
1.4 Focus and Contribution . . . 9
1.5 Outline . . . 10

Chapter 2 Relevant Machine Learning Methods 13

2.1 Statistical Learning Theory . . . 14
2.2 Boosting Methods . . . 15
2.2.1 AdaBoost . . . 16
2.2.2 Probabilistic Boosting Tree . . . 20

Chapter 3 Automatic Segmentation of Liver Lesions 25

3.1 Related Work . . . 27
3.2 Preprocessing . . . 27
3.2.1 Automatic Liver Segmentation . . . 27
3.2.2 Intensity Standardization . . . 28
3.3 Pointwise Classification . . . 29
3.3.1 Iterative Classifier Setup . . . 30
3.3.2 Feature Calculation . . . 31
3.4 Postprocessing . . . 33
3.5 Results and Discussion . . . 35
3.5.1 Subsampling of Training Data . . . 35
3.5.2 Evaluation Measures . . . 36
3.5.3 Effect of Standardization on Segmentation . . . 38
3.5.4 Effect of Iterative Classification on Segmentation . . . 38
3.5.5 Lesion Detection Performance . . . 42
3.6 Conclusion . . . 43

Chapter 4 Learning a Prior Model for Lesion Follow-up Segmentation 47

4.1 Related Work . . . 48
4.1.1 Image Change Detection . . . 49
4.1.2 Follow-up Image Registration and Segmentation . . . 49
4.1.3 Prior Models for Detection and Segmentation . . . 50

4.2 Processing Pipeline for Liver Lesion Segmentation . . . 51
4.2.1 Input Images . . . 51
4.2.2 Follow-up Liver Lesion Segmentation . . . 51
4.2.3 Follow-up Liver Registration . . . 52
4.3 Spatial Prior Models for Follow-up Segmentation . . . 52
4.4 Learning a Prior Model for Follow-up Segmentation . . . 55
4.4.1 Multiplicative Learned Prior . . . 56
4.4.2 Integrated Learned Prior . . . 58
4.5 Results and Discussion . . . 58
4.5.1 Image Database . . . 60
4.5.2 Experimental Setup . . . 61
4.5.3 Registration Evaluation . . . 61
4.5.4 Segmentation Evaluation . . . 62
4.5.5 Detection Evaluation . . . 68
4.5.6 Features . . . 69
4.6 Conclusion . . . 69

Chapter 5 A Cost Constrained Boosting Algorithm for Fast Object Detection 73

5.1 Related Work . . . 73
5.2 Constrained Boosting . . . 75
5.2.1 Speeding up Classification in the Probabilistic Boosting Tree . . . 76
5.2.2 Cost Constrained Hypothesis Training . . . 77
5.2.3 Adaptive Cost Constrained Hypothesis Training . . . 79
5.3 Results and Discussion . . . 79
5.3.1 Experimental Setup . . . 80
5.3.2 Cost Constrained Hypothesis Training . . . 81
5.3.3 Adaptive Cost Constrained Hypothesis Training . . . 81
5.3.4 Free Lunch? . . . 83
5.4 Conclusion . . . 84

Chapter 6 Outlook 87

Chapter 7 Summary 89

Chapter A Appendix 93

A.1 Evaluation Measures . . . 93

List of Symbols 95

List of Figures 97

List of Tables 99

Bibliography 101

CHAPTER 1

Introduction

1.1 X-Ray Computed Tomography Image Acquisition 2
1.2 Anatomy and Physiology of the Liver 3
1.3 Liver Tumors 5
1.4 Focus and Contribution 9
1.5 Outline 10

Medical imaging has come a long way since Wilhelm Conrad Röntgen discovered his "X-rays" in 1895 [Ront 95]. Exciting imaging modalities have been developed that provide insight into every part of the human body, from whole body images down to cell level. As these modalities measure different physical phenomena, the information they visualize may range from the morphology of arbitrary body regions or organs to body functions as complex as blood flow or localized brain activity.

X-ray imaging itself has evolved from lengthy procedures for generating blurry 2D transmission images to acquiring crystal clear 3D volumetric datasets at sub-millimeter accuracy in the blink of an eye. With state-of-the-art computed tomography devices these acquisitions are possible exposing the patient to only very low doses of the harmful ionizing X-ray radiation.

Having focused on improving image quality for decades, with the advent of stronger computers engineers started to establish new disciplines in medical imaging: computer aided diagnosis and therapy. Algorithms are now not only used to improve image quality visually by filtering, but try to interpret their content in order to assist medical staff. In some domains, such as cardiac imaging, algorithms can already provide valuable support to physicians, e. g. automatically analyzing coronaries and assessing infarction risk, suggesting optimal treatment or planning stent implantations. Other domains are still waiting for the big breakthrough in the automatic processing of their images.

One of the latter is oncology. The constantly growing number of cancer cases worldwide [Glob 11] and their high mortality rate create a particularly large need for computer support. Integrated into various steps of the clinical workflow, these algorithms would allow earlier diagnosis as well as more precise treatment. Typical tasks that could be solved algorithmically involve the detection and localization of tumors, their exact delineation, measurements and categorization, as well as monitoring and change assessment.

Unfortunately, the challenges for algorithmic solutions in this field are highly demanding, even when the method is limited to a single imaging modality, a single type of cancer, or a single target location. Of the typical clinical scenarios the fewest are well understood

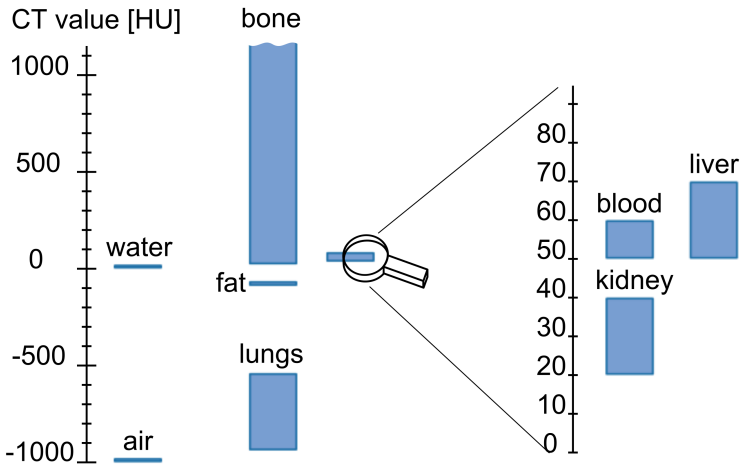


Figure 1.1: CT values of different tissue types. Adapted from [Kale 06].

from an image processing point of view. Consequently, not for many of them clinically acceptable algorithmic solutions exist, let alone a universal solution.

In the work at hand, we work towards a solution for one such scenario, the detection and segmentation of liver tumors in X-ray computed tomography images. A general segmentation system is developed and then used to broaden the understanding of and investigate further techniques for segmentation in the follow-up setting. Finally, a method for speeding up the proposed system or related ones is presented.

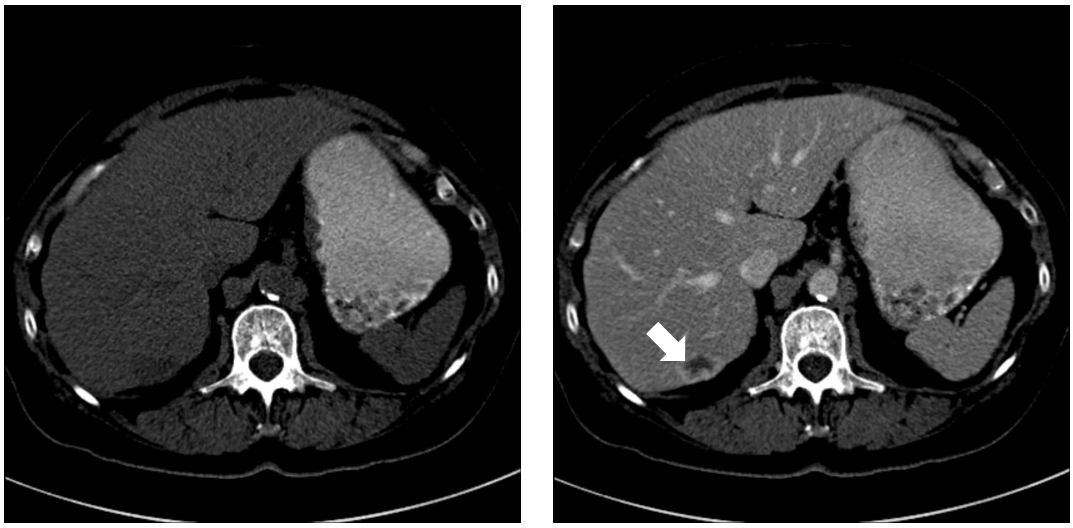
1.1 X-Ray Computed Tomography Image Acquisition

X-ray computed tomography (CT) nowadays is a standard imaging method for patients with liver tumors. It can be used in all stages of the therapy, from first diagnosis to treatment planning and follow-up examinations.

Images are acquired by rotating a fixed setup of an X-ray source and a detector array around a patient, where the detector elements measure the remaining intensity of the radiation generated by the source after transmission through the patient. From this data, a 3D volume is reconstructed showing morphological information of the examined body. The volume can be interpreted as a 3D image, where each image element ("voxel") has a certain spatial extent. Each voxel is assigned a value based on the attenuation of the tissue at the corresponding location in the patient body. The attenuation measurement refers not to a mathematical point but averages over the volume covered by the voxel, which can contain a mixture of different materials. The values, also called CT values, are given in Hounsfield units (HU). CT values are calculated by measuring the voxel's tissue specific attenuation coefficient ν_{tissue} and normalizing it to the attenuation coefficient of water ν_{water} according to Eq. (1.1):

$$[\text{CT value}_{tissue}] = \frac{\nu_{tissue} - \nu_{water}}{\nu_{water}} \cdot 1000\,\text{HU} \tag{1.1}$$

By convention, values range from -1024 to 3071, where air has a value of -1000 and water has the value 0. Other typical CT values for human tissue can be found in Fig. 1.1.



(a) (b)

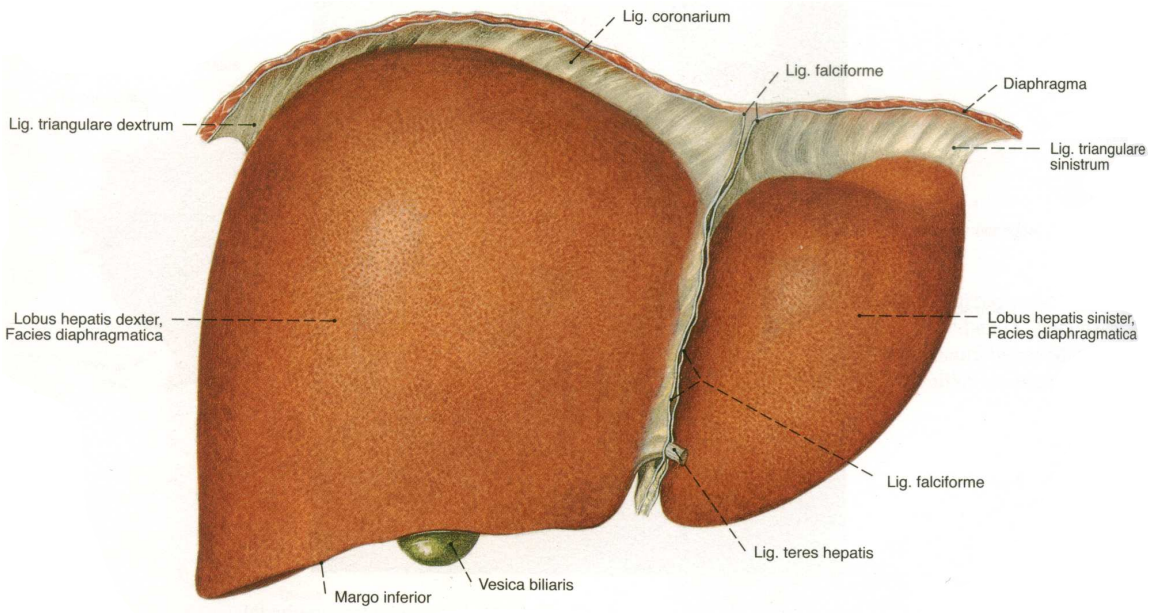
Figure 1.2: CT image slices of a patient demonstrating the effect of a positive contrast agent. Both images show the same range of gray levels. In the native image (a), the liver is much darker than in the contrast enhanced one (b). Bright structures in the liver in (b) designate liver veins, the dark spot marked by the arrow is a small tumor.

As can be seen from this graph, soft tissues fall into a rather narrow range. Paired with the comparatively high noise level in soft tissue images, this makes it difficult to distinguish between tissue types in the data. In particular, liver tissue and most liver tumors are nearly indistinguishable. When visualized as images, this manifests itself as low contrast between the different tissue types. To improve this contrast in the images and thus allow better distinction between healthy and tumorous tissue, it is often enhanced during image acquisition by bringing a so-called contrast agent into the region to be imaged (cf. Fig. 1.2). Mostly, this is done by injecting the contrast agent into the blood flow, either directly at the point of interest, which requires an interventional procedure, or peripherally so it is distributed all over the body by the blood circulation.

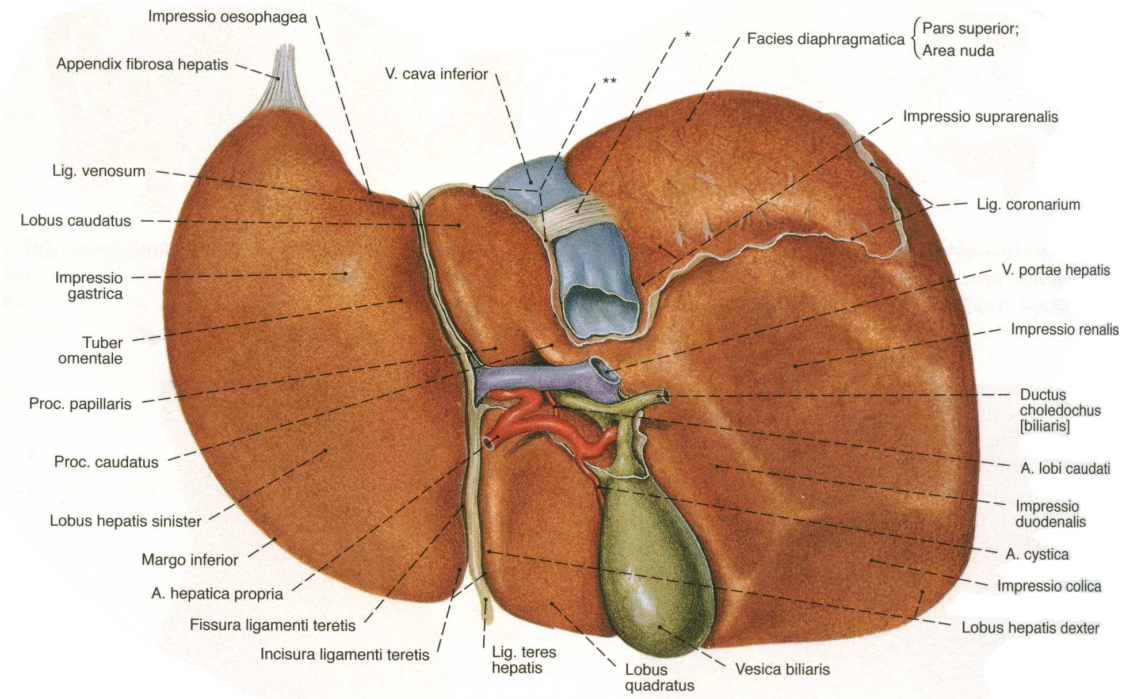
There are different types of contrast agent. X-ray positive agents exhibit a high attenuation coefficient and thus lead to a local shadow where they are present in the tissue, meaning that the image becomes brighter at these regions. This means that, in Eq. (1.1), ν_{tissue} is increased, leading to a higher CT value. X-ray negative agents show the opposite behavior, leading to darker image regions. The former ones are used far more often, especially for imaging the blood vessel system.

1.2 Anatomy and Physiology of the Liver

The liver is an integral part of the metabolism, functioning as a filter for blood arriving from the gastrointestinal tract. While the kidneys are a mere mechanical blood filter, the liver is more selective, processing the found substances in its specialized cells. That way it can extract and break down toxic substances like pharmaceuticals, but also fat, glucose and other nutrients. At the same time it functions as a gland, producing bile, which contains enzymes important for digestion inside the bowel, and synthesizes important proteins.



(a)



(b)

Figure 1.3: Surface view of the human liver, seen from the front (a) and back (b). Images taken from [Putz 04], courtesy of Urban & Fischer Verlag, Elsevier GmbH.

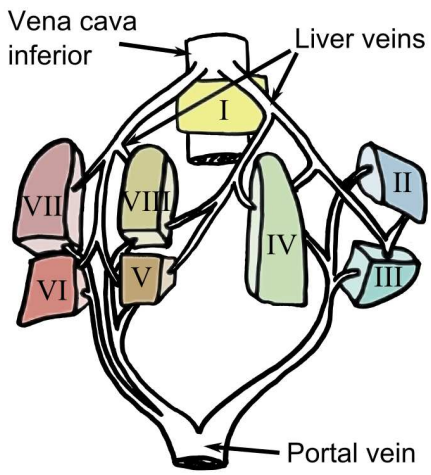


Figure 1.4: Functional segments as identified by Couinaud. Each segment (I — VIII) is defined as the region of influence of certain branches of the portal and the liver venous trees. Adapted from [Lagh 05].

With 1500 – 2000 g for an adult, the liver is the largest organ in the abdomen. It is located towards the right hand side, directly below the diaphragm. Anatomically, it can be divided into two parts, the liver lobes, separated by the ligamentum falciforme. Figure 1.3 provides an anatomical view of the liver. For liver surgery on the other hand, a subdivision into eight segments is more common [Coui 57]. These segments have the property of being functionally independent of each other, meaning that an entire segment can be resected without affecting the remaining liver tissue. This independence is possible due to the special tree-like structure of the vessels inside the liver (cf. Fig. 1.4).

Four vessel systems pervade the liver. The hepatic artery provides oxygenated blood to the liver cells. The portal vein brings blood from the gastrointestinal tract to be filtered. Approximately 75% of the blood flow towards the liver arrives via the portal vein, the remaining 25% via the hepatic artery. The liver's venous system collects all the blood and feeds it directly into the vena cava inferior leading to the heart. The fourth vessel system is the biliar system, collecting the bile produced at the liver cells and draining it towards gall bladder and duodenum. The aforementioned liver segments are independent not only in terms of their blood supply, but also in terms of venous and biliary drainage.

1.3 Liver Tumors

Primary liver cancer is the sixth most common cancer, with an estimated 748,000 cases worldwide in 2008. At the same time, it has an alarmingly high fatality rate. It is the third most fatal cancer, having caused about 695,000 deaths worldwide in the same year 2008 [Ferl 10]. While almost 85% of the cases occur in developing countries, even in Europe the 5-year survival rate is as low as 9.1% [Glob 11]. The most frequent primary malignant tumor in the liver is the hepatocellular carcinoma (HCC), which in most cases develops on top of a hepatitis infection or a liver cirrhosis. Far more often than primary liver cancer, however, are metastases from different other cancers. In fact, in Europe about 90% of all malignant liver tumors are metastases [Laye 08]. Patients with colorectal cancer, for instance, have a 70% chance of developing liver metastases at some time [Bipa 05]. In total, there are about 30 different clinically relevant tumors of the liver [Laye 08], though not all of them are malignant.

The location of liver tumors varies greatly from patient to patient. For some types of cancer rough information about their spatial distribution is available — hepatoblastoma, for instance, are known to be located in the right lobe in 60% of the cases [OGra 00]. In general, however, lesions can be found anywhere inside the liver. Especially metastases, since they can invade the liver via the portal vein as well as the liver artery and even

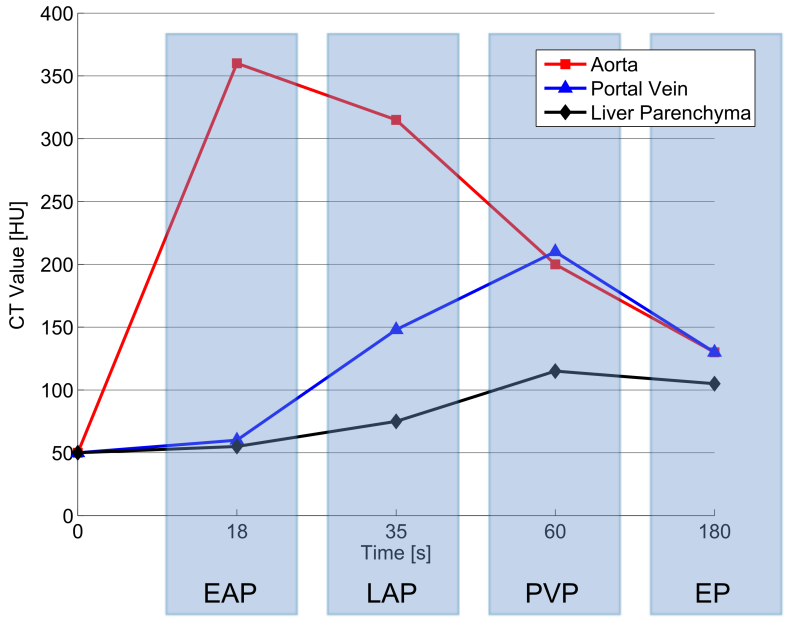


Figure 1.5: Enhancement patterns of aorta, liver parenchyma, and portal vein over time. The time scale starts at the moment of injection of the contrast agent into the peripheral vein, the points on the curves mark the time of image acquisition. Four phases of contrast enhancement are distinguished: Early arterial phase, when the contrast agent first reaches the liver via the aorta (EAP), late arterial phase, when it is washed into the liver arteries (LAP), portal venous phase, when the contrast agent is distributed all over the parenchyma via the portal vein (PVP), and equilibrium phase, sometimes also called washout phase, when only traces remain in the liver and the rest is washed out via the liver veins (EP). Image adapted from [Lagh 05].

originate from other liver tumors, are widely spread. Also, patients often have more than one tumor.

1.3.1 CT Imaging

In native CT images, most liver tumors are not distinguishable from surrounding liver tissue. Therefore usually contrast agent enhanced images are acquired for their diagnosis. After injecting the contrast agent into a peripheral vein, it is first washed into the liver via the hepatic artery. Later, a second wave of contrast agent enters the liver via the portal vein. Depending on acquisition timing, up to 4 phases of contrast enhancement are captured in the images (see Fig. 1.5).

Tumors grow differently depending on the type of cells they develop from. This influences their entire structure, including blood supply. They may therefore exhibit different enhancement patterns during contrast agent inflow, sometimes allowing identification from multiphase images alone, without biopsy. Tumors often have a very active metabolism and are therefore hypervascularized, with a large arterial blood supply. In arterial phase images, this shows as strong enhancement along the rim of the tumor, in the late arterial phase often as a strong enhancement of the entire lesion. In later phases, this hyperdense (brighter than surrounding) appearance often turns into a hypodense (darker than surrounding) one, as

the contrast agent is already washed out from the tumor, while the surrounding tissue is saturated with contrast agent entering the liver via the portal vein. In contrast, lesions may also be hypovascularized, leading to an isodense or hypodense appearance in arterial and a hypodense appearance in venous phase images. Some lesions even exhibit a portal venous hypervascularization. Some examples of lesions with different enhancement patterns can be found in Fig. 1.6.

Acquisition of all contrast phases is rare in clinical routine, since the number of used phases has a small influence on tumor detection and biopsy is still considered necessary for final identification of the lesion type. At the same time additional scans mean additional dose for the patient. Often one arterial and one venous image are acquired, although the benefit of the arterial phase is also subject to heavy discussions in the community. The portal venous phase (often only called venous phase) is the standard phase for tumor assessment and therefore always acquired. Depending on the suspected tumor presence, an additional arterial image might be obtained as well.

In this work it is therefore only assumed that an image with some kind of venous enhancement is available in order to keep the requirements as low as possible, although an extension to more phases is possible with our approach. Visual inspection of the data and some preliminary experiments indicate that adding an arterial phase image could improve performance considerably in some cases, if sufficient example images are provided for training of the system.

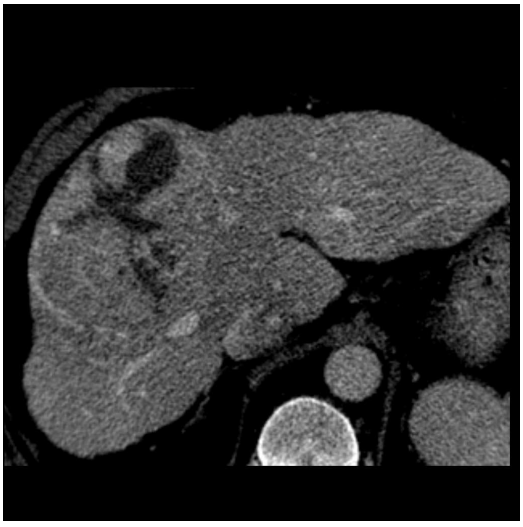
1.3.2 Therapy Options

Nowadays a whole bunch of different treatment options for liver tumors is available. Still, the standard therapy is a resection of the affected area. This resection can include one or more segments or even an entire liver lobe. This radical therapy becomes feasible due to the liver's amazing regeneration capabilities. However, in case of preexisting liver conditions like cirrhosis as much tissue as possible has to be preserved, leading to a different resection strategy. In these so-called atypical resections either a wedge-shaped portion is resected or only the tumor itself plus a small safety margin. The latter interventions require particularly thorough planning to ensure fully functional blood support and drainage for the remaining liver tissue. If the liver is damaged too severely, a transplantation is the only solution. In recent years, living donor transplantations have attracted a lot of attention, again extensively using the regenerative property of the liver.

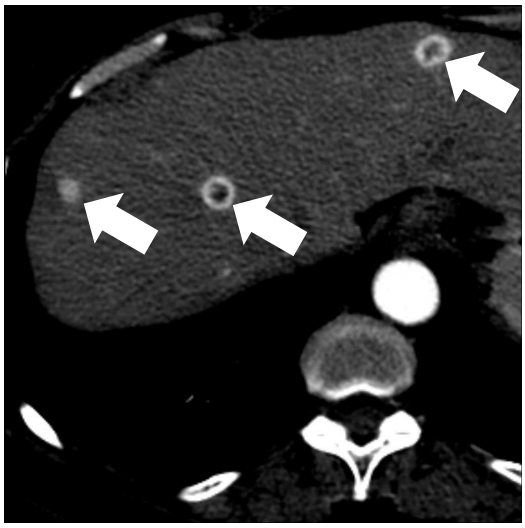
Besides surgery, there are numerous alternatives. These are mainly used in cases where a resection is not advised because the number of target lesions is too large, because the target lesions are not accessible or dangerously close to important vessels, because the loss of liver tissue would be too big, or because the patient's overall condition is too bad. These treatment options include the injection of ethanol, radiofrequency ablation, cryoablation, or radiation therapy. In all these cases the tumorous tissue is destroyed in situ. A chemotherapy may be applied either as preparation for any of the aforementioned treatment options or as primary therapy if the tumor(s) cannot be treated otherwise.

In any case, treatment success has to be assessed. Especially tumors that were not removed surgically need to be closely monitored. To this end, follow-up images are acquired on a regular basis at intervals of several months and lesions are measured. The standard guidelines for these measurements, the response evaluation criteria in solid tumors (RE-

(a)

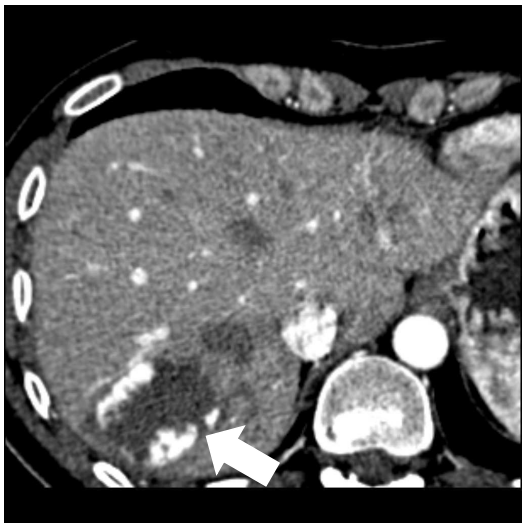


(b)

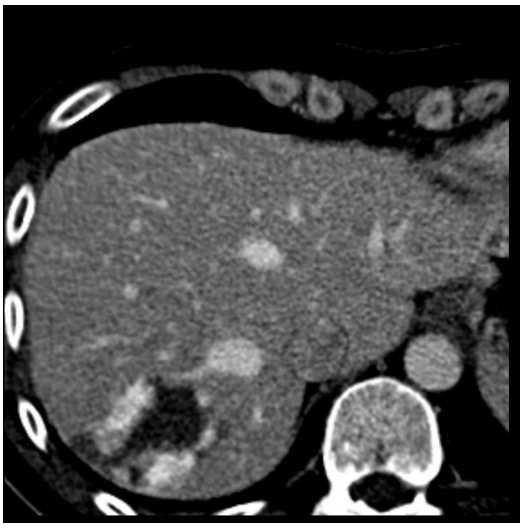


(c)

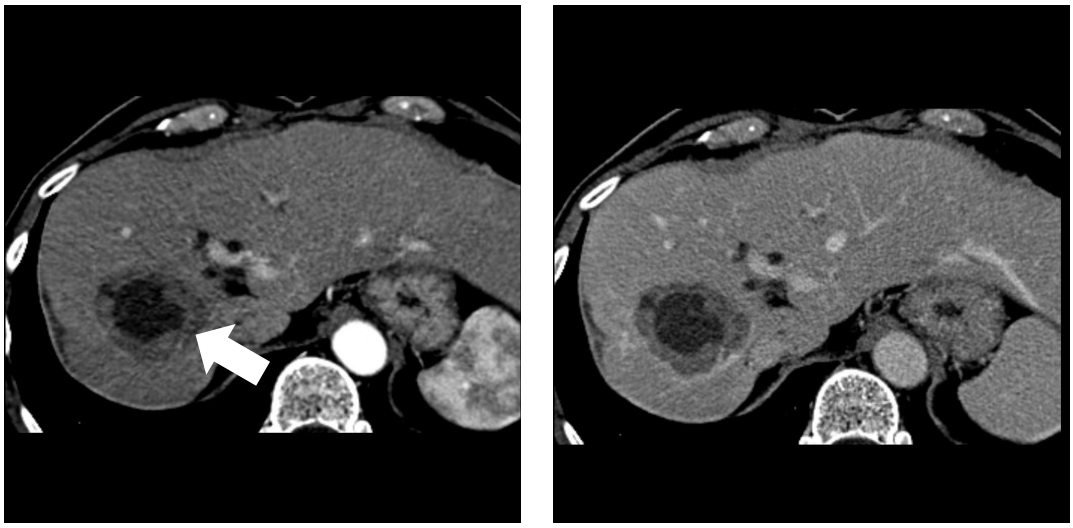
(d)



(e)



(f)



(g) (h)

Figure 1.6: Various types of liver tumors with different enhancement patterns. Images in the left column were acquired during the late arterial phase, those in the right column during the portal venous phase.

CIST [Eise 09]), define their size criteria based on the largest axial diameters of a subset of up to five representative target lesions. Based on these diameters and the total number of lesions, the total tumor burden is estimated and treatment response or disease progression is rated.

1.4 Focus and Contribution

In clinical routine working with CT images of liver tumor patients involves a lot of manual interaction for initial identification of lesions, intervention planning, or measurements during follow-up examinations. The work at hand presents machine learning based methods for automatic detection and segmentation of focal liver lesions. Detection in this context means the localization of lesions, whereas segmentation denotes their precise delineation.

The main focus of the proposed methods lies on the follow-up case. To keep the complexity at bay, the set of target lesions was limited. Since from an image processing point of view the actual tumor type is of secondary importance, the selection criterion was not biological but based on the lesions' appearance in the images. We limited our efforts to primarily hypodense lesions, which make the largest portion of all cases. Non-focal lesions like cirrhosis are not targeted here. However, the methods that will be presented in the following chapters are sufficiently generic to be extended to arbitrary kinds of focal lesions, provided that there is enough training data. This distinction of different cases only by appearance implies that the image databases used for our experimental evaluations may contain not only tumors but also other focal lesions like cysts. While a categorization of liver lesions from a small set of multiphasic CT images is generally not considered possible, identifying non-tumors like cysts might be. Still, any distinction beyond the categories hyperdense vs. hypodense is not in the scope of this thesis and therefore not considered further.

Starting point for the proposed methods is a learning based standard approach to object segmentation. It uses a previously trained classifier to divide the image into target objects and background regions by classifying every single image point and assigning it to one of the two classes. We start with a state-of-the-art system from literature [Shim 08] and refine it by

- replacing its AdaBoost classifier by a Probabilistic Boosting Tree [Tu 05],
- improving its preprocessing by incorporating an intensity standardization step originally developed for use with magnetic resonance images [Jage 09], and
- making classification results more smooth and robust by introducing an iterative classification approach [Morr 08].

A description and experimental evaluation of this system was published in 2010 [Mili 10].

With a system like this, lesions can be monitored over time by segmenting them in each follow-up image separately and matching and comparing them afterwards. In this setup, the segmentation step treats each image of a series as if they were independent. To improve accuracy, we modify this approach so that patient specific information gained from previous images is incorporated into the current segmentation process. This is achieved by

- learning a lesion prior from example datasets and either
- incorporating the prior directly into the classification step of the segmentation system or
- combining the prior with the output of the original classification step according to Bayes theory.

This approach is also described in a recent publication [Mili 13b].

Besides accuracy, response time is one of the main criteria for use of a system like this in clinical routine. Methods based on voxel classification have by nature a very high computational complexity, even with today's optimized algorithms and hardware. The last part of this thesis thus deals with techniques for speeding up the machine learning methods forming the core of the presented system. More specifically, we developed a modification of the AdaBoost algorithm, which

- makes it prefer simpler features, speeding up classification in hierarchical boosting classifiers without losing classification accuracy, and
- allows the optimization of arbitrary side conditions.

This cost constrained boosting algorithm was published in 2013 [Mili 13a].

1.5 Outline

After the above introduction to the medical background and the motivation for the work presented in the thesis at hand, the remainder of the document is organized as follows:

- Chapter 2 provides the reader with information about the used machine learning methods. General ideas that form the basis of all machine learning are outlined as well as general definitions from this field. Starting with some insights from the theory of machine learning, the boosting methods used throughout this thesis are explained in detail.
- Chapter 3 shows, how the aforementioned learning methods can be used for segmenting liver tumors in CT images. A complete segmentation system is described and evaluated on a non-public set of clinical images. This system forms the basis for the following two chapters.
- In Chapter 4, it is extended and adapted for the case of follow-up lesion segmentation. A novel method for incorporating knowledge gained from previous image acquisitions is presented. It includes learning a prior from the given image series and combining it with the existing segmentation system.
- Chapter 5 finally deals with an issue that concerns all learning based detection methods, i. e. their computational complexity and thus runtime. A method is presented that allows speeding up hierarchical boosting-based classification systems without affecting their classification performance negatively.
- Chapter 6 outlines still open or newly raised questions from this project, which are left for future work.
- Chapter 7 sums up the entire thesis.

CHAPTER 2

Relevant Machine Learning Methods

2.1 Statistical Learning Theory 14
2.2 Boosting Methods 15

The idea of "intelligent" and especially learning machines has always intrigued people and stimulated their imagination. Nowadays, machine learning (ML) methods are used extensively in tasks like object detection or categorization [Enzw 09, Zhan 13], market analysis [Bose 01, Rudi 13], and regulating control systems [Golu 13, Bris 06]. They have made it to the standard repertoire of pattern recognition algorithms. However, taking a closer look at these methods shows they are only vaguely related to the science fiction ideal of machine intelligence and our current understanding of mind and consciousness raises some doubts as to whether these fantasies will ever become reality.

Technically speaking, machine learning refers to the concept of algorithmically deriving knowledge from data. The learning process can be as simple as the calculation of a mean value over some data points or as complex as a multi-criteria optimization [Bish 07]. The knowledge may be represented explicitly in the form of human readable rules or implicitly as parameters of a model. All, that is essential, is that system has gained some knowledge and that this knowledge, in contrast to the original raw data, allows inference and prediction of new information and thus leads to a change in the system's behavior.

This coarse description already reveals one fundamental property of systems based on machine learning methods: Their life cycle is inherently divided into a learning or training phase and an application phase. The latter is the productive phase in which the system is used for the actual purpose it was designed for. The training phase in a way replaces the normal software development process, in which the desired program behavior is implemented. As Valiant puts it [Vali 84], learning is "the phenomenon of knowledge acquisition in the absence of explicit programming". So, the key concept and the difference to conventional methods is to not set parameters manually as a developer but only design the learning algorithm and let the system learn its desired functionality from "experience" in the form of examples. Depending on the kind of data available for training and the kind of predictions made by the system, machine learning algorithms are divided into different types.

Generally, the goal of the learning algorithm is a mapping $f : \mathbb{X} \mapsto \mathbb{Y}$ from the so called feature space $\mathbb{X}$, to a target variable $y \in \mathbb{Y}$. Often the feature space is $\mathbb{X} \subset \mathbb{R}^m$, but there are

learning methods that can deal with integral or categorical types just as well. The training data is represented as a number of feature vectors $\mathbf{x} \in \mathbb{X}$.

If the target variable is continuous, the prediction f is called regression, if $\mathbb{Y}$ is a discrete set of labels, the prediction is called classification and the (finite) target values are called categories or classes. In the latter case, the decision boundaries separating the classes form hyperplanes in the input space.

Since the methods presented in this thesis heavily rely on classification algorithms, the following descriptions will focus on this aspect of machine learning.

Supervised Learning

In supervised learning methods the feature vectors $\mathbf{x}$ in the set of training examples $\mathbf{X} \subseteq \mathbb{X}$ are complemented by the corresponding class labels, forming the training set $\mathbf{V} = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)) \subseteq \mathbb{X} \times \mathbb{Y}$. That means, for each feature vector $\mathbf{x}_i \in \mathbf{X}$, the correct target value or class label is known and provided to the learning algorithm as a teacher signal. That way the classifier can adapt its parameters to match the desired output and thus minimize its error rate on the training data $\sum_{i=1}^{n} |y_i - f(\mathbf{x}_i)|$ or some other kind of loss function based on this discrepancy. The goal is a low classification error during application phase, i. e. when applied to previously unseen examples. This step of transferring knowledge gained from examples to new input is called generalization.

Unsupervised Learning

In an unsupervised learning setting the training data does not contain class labels or any other teacher signal. The goal here is to find patterns in the examples so that data points or regions of the input space are grouped in a way that within a group the points are very similar to each other whereas objects of different groups are dissimilar. Examples for unsupervised learning algorithms are the family of clustering methods as well as feature reduction techniques like principal component analysis.

Reinforcement Learning

Reinforcement learning has a special position between the other two categories. There is no explicit teacher signal and thus no explicit loss. Learning success is, instead, measured by means of a reward. This kind of learning is often applied where agents interact with a dynamic environments, so that each action has consequences and thus may lead to some reward. However, since the action may influence all future time steps, the reward may also not be available immediately. This is very similar to human learning processes.

2.1 Statistical Learning Theory

Statistical learning theory deals with topics around the learnability of classes of functions, the learning capabilities of algorithms, and the data used for learning. In this context, a key concept that was introduced by Valiant in 1984 [Vali 84] is what was later called probably approximately correct (PAC) framework. In this framework, a class of concepts $\mathscr{C}$ is considered strongly learnable, if there is an algorithm such that for any concept $\xi \in \mathscr{C}$ it will produce a good hypothesis h for ξ with high probability.

A (binary) concept over the feature space $\mathbb{X}$, for example, could be a subset of the space and defined by

$$\{\mathbf{x} \in \mathbb{X} : \xi(\mathbf{x}) = 1\}. \tag{2.1}$$

If a learning algorithm generates a hypothesis h for this concept based on a set of training examples drawn from $\mathbb{X}$ according to a probability distribution D, then its error probability is defined as

$$\theta = \mathrm{E}_D[h(\mathbf{x}) \neq \xi(\mathbf{x})][\text{Bish 07}]. \tag{2.2}$$

The full definition of PAC learnability is then as follows [Vali 84, Hayk 99, Bish 07]:

Definition 1. A class $\mathscr{C}$ of concepts is called PAC learnable if there exists an algorithm so that

- for every $\xi \in \mathscr{C}$
- for every probability distribution D over the input space and
- for every $\delta \in (0, 0.5)$ and $\rho \in (0, 0.5)$

with probability at least $1 - \delta$ the learning algorithm produces a hypothesis h with an error probability $\theta < \rho$ for any training sample $\mathbf{V} = \{(\mathbf{x}_1, \xi(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, \xi(\mathbf{x}_n))\}$ with $n \geq n_0$ drawn according to D.

The $n_0 \in \mathbb{N}$ in this definition may vary for different distributions or for different values for δ and ρ. An algorithm is called PAC or a strong learner if it fulfills this definition for the concept class $\mathscr{C}$ being the hypothesis space $\mathscr{H}$, which is the set of all hypotheses that can be generated by the algorithm.

Analogously, a weak learner is an algorithm, for which in this definition the expected error θ can only be guaranteed to stay below 0.5, i. e. it is at least better than random guessing.

2.2 Boosting Methods

Boosting is a meta-learning algorithm, i. e. it does not define a learning algorithm by itself, but a method for improving the performance of arbitrary learning algorithms. It was presented by Schapire in 1990 [Scha 90] as the affirmative response to the question raised by Kearns and Valiant [Kear 89] whether the concepts of weak and strong learnability are equivalent. With the boosting method Schapire presented not only a proof for his affirmative answer to this question but also a means for transforming any weak learning algorithm into a strong one.

The core idea is to run the weak learner several times, where the later iterations focus on the hard examples, and combine the responses of the single weak learners to form an overall response. In the original publication, the focusing is implemented by assuming a practically infinite amount of training data and using earlier stages to filter the training samples for later ones. That way later stages can focus on those examples that were previously misclassified. In later versions, boosting by resampling or boosting by reweighting were preferred.

Boosting by resampling uses a finite set of training examples, from which a new subset is sampled according to a probability distribution for each iteration. By calculating the error w. r. t. the entire training set and adapting the distribution, the sampling focuses on the hard examples in later iterations. Sampling by reweighting uses the same set of training examples for each iteration, but assigns a weight to each individual example. During each iteration, the error of the weak learner is calculated w. r. t. the sample weights and in turn used to adapt the weights, so that misclassified examples receive more attention.

2.2.1 AdaBoost

Many realizations of the boosting principle have been proposed since its origin in 1990. One of the most successful and most frequently used to this day is the AdaBoost algorithm by Freund and Schapire [Freu 95]. Since it is used extensively in this thesis, it will be described in some detail on the next few pages. It is based on three key concepts:

- A weighted training set,
- iterative calls to a weak learning procedure and
- focusing on hard examples by reweighting the training set.

Hard examples in this context are those that are close to the decision boundary.

Let $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^m$ be a set of feature vectors with class labels $y_i \in \mathbb{Y}$, $i \in \{1, \ldots, n\}$. AdaBoost was originally designed for binary classification problems, so we assume $\mathbb{Y} = \{-1, +1\}$. These vectors $\mathbf{X}$ together with their class labels form the set of training instances $\mathbf{V} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$. The goal of the learning algorithm is a mapping $\mathbb{R}^m \mapsto \mathbb{Y}$ of the form

$$H(\mathbf{x}) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right),\ \alpha_t > 0,\ h_t(\mathbf{x}) \in \{-1, +1\}, \tag{2.3}$$

where often we will use $g(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$ for the sake of simplicity. During training (Algorithm 1), AdaBoost iteratively calls a weak learning algorithm that provides the hypotheses h_t for this linear combination. Additionally, a distribution D assigning a weight to each of the training instances is maintained. D is updated after each iteration, putting more emphasis on the previously misclassified samples: Weights of previously misclassified samples are increased by a factor $\exp(-\alpha_t)$, those of correctly classified samples are decreased by a factor of $\exp(\alpha_t)$. Afterwards, the samples are normalized again to ensure that $\sum_i D_{t+1}(i) = 1$. The weighting factor α_t that is assigned to each h_t is a confidence weighting determined based on its training error with respect to D_t: Hypotheses with a high training error receive low weight and vice versa.

AdaBoost is a meta-learning algorithm, i. e. it does not regulate which kind of weak learner should be used. Consequently, it has been combined with all sorts of classifiers, from simple decision trees to fully grown neural networks or support vector machines. While the latter ones are strong learners by themselves, the boosting theory states that their performance will improve when they are plugged into the AdaBoost algorithm.

Throughout this thesis, one-level decision trees, also known as decision stumps, act as weak learners. These apply a threshold to a single feature of the input vector $\mathbf{x}$ (cf. right

part of Fig. 2.1). Training a weak learner then means selecting the feature and threshold producing the lowest error on the training set. That way, during each iteration AdaBoost greedily picks the "best" feature. Compared to other learning algorithms this has the advantage that the user does not have to put much effort into selecting a small number of optimal features. Instead, one can run AdaBoost with a large set of features and let the algorithm perform the feature selection along the way. As an additional benefit, the result provides insight into the nature of the underlying classification problem and the usefulness of certain features.

Algorithm 1 The AdaBoost algorithm [Freu 96].

Input: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ where $\mathbf{x}_i \in \mathbb{R}^m$, $y_i \in \{-1, +1\}$.

1: Initialize $D_1(i) = 1/n$.
2: **for** $t = 1 \ldots T$ **do**
3: Train weak hypothesis $h_t : \mathbb{R}^m \mapsto \mathbb{Y}$ using distribution D_t.
4: Get weighted training error of hypothesis $\theta_t = \sum_{i:h_t(\mathbf{x}_i) \neq y_i} D_t(i)$.
5: Set $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\theta_t}{\theta_t} \right)$
6: Update

$$D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)}}{Z_t}$$

with

$$Z_t = \sum_i D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)} = 2\sqrt{\theta_t(1-\theta_t)}.$$

7: **end for**

Output: Final hypothesis $H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}) \right) = \text{sign}(g(\mathbf{x}))$.

Error Bounds

One reason why AdaBoost was quickly adopted by the community is certainly its simplicity, another is the guarantee for success stated by its inventors. Freund and Schapire provided a thorough analysis of expected error rates [Freu 95]. Not only could they show that the training error for AdaBoost is bounded by

$$\Theta \leq \prod_{t=1}^{T} 2\sqrt{\theta_t(1-\theta_t)} = \prod_{t=1}^{T} \sqrt{1-4\gamma_t{}^2} \leq \exp\left(-2 \sum_{t=1}^{T} \gamma_t{}^2 \right), \; \gamma_t = 0.5 - \theta_t, \tag{2.4}$$

they also analyzed the expected generalization error and proved it is with high probability at most

$$\frac{\Theta}{n} + O(\sqrt{\frac{Tv}{n}}). \tag{2.5}$$

v in this equation denotes the VC-dimension of the hypotheses, a measure for the complexity of the hypothesis space that is named after Vapnik and Chervonenkis, who developed an

extensive theory around this concept. While it was later refined several times, the bound in Eq. (2.5) already gave some hints on what to expect from the AdaBoost learning algorithm.

Choice of α

The error bound in Eq. (2.4) can be confirmed by unraveling the calculation of the weight updates for the training examples postulated in Alg. 1. For the sake of notational simplicity, we use $\sum_t$ for $\sum_{t=1}^{T}$ and $\sum_i$ for $\sum_{i=1}^{n}$, as well as $\prod_t$ for $\prod_{t=1}^{T}$.

$$\begin{aligned} D_{T+1}(i) &= \frac{1}{n}\frac{e^{-\alpha_1 y_i h_1(\mathbf{x}_i)}}{Z_1} \cdot \ldots \cdot \frac{e^{-\alpha_T y_i h_T(\mathbf{x}_i)}}{Z_T} \\ &= \frac{e^{\sum_t -\alpha_t y_i h_t(\mathbf{x}_i)}}{n\prod_t Z_t} \\ &= \frac{e^{-y_i \sum_t \alpha_t h_t(\mathbf{x}_i)}}{n\prod_t Z_t} \\ &= \frac{e^{-y g(\mathbf{x}_i)}}{n\prod_t Z_t} \end{aligned} \tag{2.6}$$

In the case of a misclassified example we have $H(\mathbf{x}_i) \neq y_i$, such that $y_i g(\mathbf{x}_i) \leq 0$ and thus $\exp(-y_i g(\mathbf{x}_i)) \geq 1$. For the following conversions, we use the notation

$$[\![\langle condition \rangle]\!] = \begin{cases} 1 & \text{if } \langle condition \rangle \text{ is true} \\ 0 & \text{otherwise.} \end{cases} \tag{2.7}$$

With this, it is easily agreed that

$$[\![H(\mathbf{x}_i) \neq y_i]\!] \leq e^{-y_i g(\mathbf{x}_i)} \tag{2.8}$$

and, summing over all feature vectors in the training set,

$$\frac{1}{n}\sum_i [\![H(\mathbf{x}_i) \neq y_i]\!] \leq \frac{1}{n}\sum_i e^{-y_i g(\mathbf{x}_i)}. \tag{2.9}$$

Bringing together the findings from Eq. (2.9) and Eq. (2.6), one comes to the conclusion that

$$\begin{aligned} \Theta = \frac{1}{n}\sum_i [\![H(\mathbf{x}_i) \neq y_i]\!] &\leq \frac{1}{n}\sum_i e^{-y_i g(\mathbf{x}_i)} \\ &= \sum_i D_{T+1}(i)\prod_t Z_t \\ &= \prod_t Z_t \sum_i D_{T+1}(i) \\ &= \prod_t Z_t. \end{aligned} \tag{2.10}$$

The last conversion step is possible due to the fact that the distribution D is normalized in each iteration, so that $\sum_i D_{T+1}(i) = 1$.

One interesting conclusion that can be drawn from this inequality is, that AdaBoost essentially performs a steepest gradient minimization of $\prod_t Z_t$ by minimizing in each iteration the exponential loss function

$$Z_t = 2\sqrt{\theta_t(1-\theta_t)} = \sum_{i=1}^{n} D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)}. \tag{2.11}$$

In order to minimize Z_t, two parameters have to be chosen: h_t and α_t. The former is the result of the weak learner training. As mentioned before, AdaBoost is a meta-learning algorithm and does not make a statement as to what kind of weak learner has to be used, as long as its error on the weighted training set remains below 0.5. Thus, we take h_t as given here. With that set fixed, what remains is to minimize Eq. (2.11) w. r. t. α_t.

The optimal value for α_t can thus be determined by solving $\frac{dZ_t(\alpha)}{d\alpha_t} = 0$. For better readability, we will do this for a single iteration and as a consequence leave t out of the following formulation.

$$\begin{aligned}
\frac{dZ(\alpha)}{d\alpha} &= \frac{d}{d\alpha} && \sum_i D(i) e^{-\alpha y_i h(\mathbf{x}_i)} \\
&= - && \sum_i D(i) y_i h(\mathbf{x}_i) e^{-\alpha y_i h(\mathbf{x}_i)} \\
&= - && \sum_{i:h(\mathbf{x}_i)\neq y_i} D(i) y_i h(\mathbf{x}_i) e^{-\alpha y_i h(\mathbf{x}_i)} && - \sum_{i:h(\mathbf{x}_i)= y_i} D(i) y_i h(\mathbf{x}_i) e^{-\alpha y_i h(\mathbf{x}_i)} \\
&\overset{(i)}{=} && \sum_{i:h(\mathbf{x}_i)\neq y_i} D(i) e^{\alpha} && - \sum_{i:h(\mathbf{x}_i)= y_i} D(i) e^{-\alpha} \\
&= && e^{\alpha} \sum_{i:h(\mathbf{x}_i)\neq y_i} D(i) && - e^{-\alpha} \sum_{i:h(\mathbf{x}_i)= y_i} D(i) \\
&\overset{(ii)}{=} && e^{\alpha}\theta && - e^{-\alpha}(1-\theta) \\
&= && \frac{e^{2\alpha}\theta - 1 + \theta}{e^{\alpha}} \\
&\overset{!}{=} && 0
\end{aligned} \tag{2.12}$$

In step (i) of this calculation, we make use of the fact that $y_i \in \{-1,1\}$ and $h(\mathbf{x}_i) \in \{-1,1\}$. Step (ii) applies the definition of the training error of a hypothesis from Alg. 1 and the fact that D is a distribution.

Reformulating Eq. (2.12) further yields

$$\begin{aligned}
e^{2\alpha}\theta - 1 + \theta &= 0 \\
&\Leftrightarrow \\
e^{2\alpha} &= \frac{1-\theta}{\theta} \\
&\Leftrightarrow \\
\alpha &= \frac{1}{2}\ln\left(\frac{1-\theta}{\theta}\right).
\end{aligned} \tag{2.13}$$

Plugging this into $\frac{d^2Z(\alpha)}{d^2\alpha}$ confirms, that this value of α in fact represents the minimum of $Z(\alpha)$ and thus the optimal choice for the hypothesis weight in this version of AdaBoost.

Proof for this claim as well as the error bound above has been presented in several publications. The one that is formulated here largely follows the argumentation by Freund, Schapire and Singer [Freu 95, Scha 99].

Interpretation of AdaBoost

Since its publication, AdaBoost and its capabilities of transforming weak classifiers into strong ones have been the subject of extensive research. Many different views of the learning algorithm have been proposed, though none of them can explain all of its behavior. While algorithmically clear and simple, mathematically AdaBoost is very hard to grasp. One of the more widely accepted perspectives is the margin theory. It basically states that AdaBoost training maximizes the margin of the hypotheses, which is defined as the distance of a sample to the decision boundary [Scha 98]

$$\phi_i(H) = y_i \frac{\sum_t \alpha_t h(\mathbf{x}_i)}{\sum_t \alpha_t}. \tag{2.14}$$

With this concept, Schapire et al. explained an effect observed in many experiments: Even after many iterations, when the training error has reached zero, the generalization error keeps decreasing. In most learning algorithms, the generalization error would increase in this situation, as the classifier adapts too closely to the training data; a phenomenon that is known as overfitting. The margin idea was picked up and developed further in several other publications, leading to approaches directly optimizing the margin via gradient descent methods [Maso 99, Maso 00, Rats 01].

Whether AdaBoost overfits or not has long been subject to discussions. The current consensus is summarized, e. g., by Dietterich [Diet 00] and Mease and Wyner [Meas 08]: AdaBoost in general shows little overfitting, but it can severely overfit for datasets with a high noise level. The reason for the latter is founded in the weight modification procedure for misclassified samples.

A second view of AdaBoost originated from an influential paper by Friedman et al. [Frie 00]. There, the authors develop a statistical argument, describing AdaBoost by means of additive models. This leads them to the conclusion that AdaBoost approaches logistic regression with

$$g(\mathbf{x}) \approx \frac{1}{2} \ln \frac{p(y=+1|\mathbf{x})}{p(y=-1|\mathbf{x})} \tag{2.15}$$

for posterior probabilities $p(y|\mathbf{x}), y \in \{-1,+1\}$. The most important implication from this for the work at hand is that the output of the AdaBoost classifier can directly be translated into an approximate probability value.

2.2.2 Probabilistic Boosting Tree

The PBT is a hierarchical classifier proposed by Tu in 2005 [Tu 05]. Designed as a two-class classifier, it builds a binary tree of strong learners during training (cf. Fig. 2.1). Much like a decision tree, the PBT classifies examples in a divide & conquer manner where at

each node of the tree the result is refined. However, bearing strong learners, the nodes of a PBT have much more discriminative power than those of a decision tree, which usually consist of only a single feature. Also, the decisions made at each node of the PBT while traversing the tree are not binary but represent probabilities, by default incorporating both subtrees. It is due to this latter fact that its mode of operation is therefore often referred to as that of a soft decision tree. Its way of decision making also strongly resembles the hierarchical mixture of experts approach [Jord 94, Hayk 99] with a dynamic gating network.

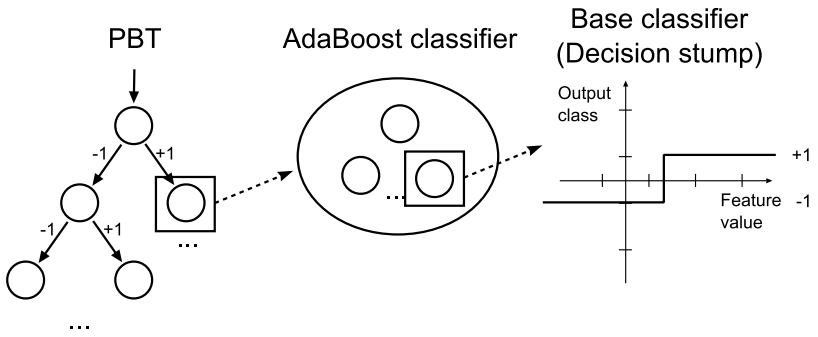


Figure 2.1: Structure of the PBT. Each tree node contains an AdaBoost classifier, which in turn consists of a number of weak learners, in this case decision stumps.

Training Phase

The training procedure of the PBT is recursive: Starting with a set of weighted training examples **S**, at the root node of the tree a strong learner is trained, stopping early. Using AdaBoost to train these as Tu recommends has the advantage that their output H allows the computation of approximate posterior probabilities $q(y|\mathbf{x})$ for a sample $\mathbf{x}$ as (cf. Eq. (2.15))

$$q(\pm 1|\mathbf{x}) = \frac{\exp(\pm 2g(\mathbf{x}))}{1+\exp(\pm 2g(\mathbf{x}))}. \tag{2.16}$$

Using these probabilities, the training samples are separated by the newly trained strong learner (cf. Fig. 2.2):

- Samples, for which $q(+1|\mathbf{x}) > \frac{1}{2}+\varepsilon$ are put into a positive subset;
- samples, for which $q(-1|\mathbf{x}) = 1-q(+1|\mathbf{x}) > \frac{1}{2}+\varepsilon$ are put into a negative subset;
- samples, for which $q(\pm 1|\mathbf{x}) \in [\frac{1}{2}-\varepsilon, \frac{1}{2}+\varepsilon]$ for a user defined ε are considered ambiguous and put into both subsets.

Ambiguous samples are those located close to the decision boundary of the strong learner, so that $g(\mathbf{x})$ id close to 0. The two subsets are then reweighted, placing less emphasis on the ambiguous samples, and used to train the right (positive) and left (negative) subtrees. For the full training procedure see Alg. 2.

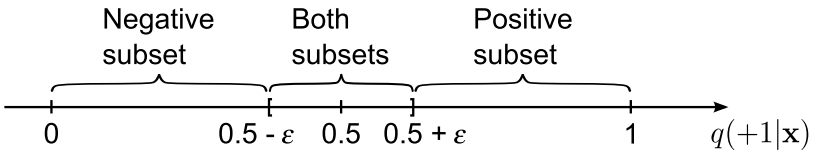


Figure 2.2: During PBT training, at each node the samples are separated into positive and negative subsets according to the approximate posterior probabilities $q(y|\mathbf{x})$ calculated by the node's strong learner. The softness of the split ε is a user defined training parameter.

Algorithm 2 The training procedure of the PBT.

Input: $\mathbf{S} = \{(\mathbf{x}_1, y_1, w_1), \ldots, (\mathbf{x}_n, y_n, w_n)\}$. where $\mathbf{x}_i \in \mathbb{R}^m$, $y_i \in \{-1, +1\}$, and $\sum_i w_i = 1$. Maximum tree depth L. Split softness $\varepsilon \in [0, 0.5]$.

Compute and store empirical distribution $\hat{q}_c(y) = \sum_{i:y_i=y} w_i$.
Using AdaBoost, train strong learner H_c for current node c from $\mathbf{S}_c$, stopping early, for example when hypothesis error $\theta > 0.45$.
if reached tree depth L **then**
    return
else
    Split $\mathbf{S}_c$ into $\mathbf{S}_{left}$ and $\mathbf{S}_{right}$:
    for all $\mathbf{x}_i \in \mathbf{S}_c$ **do**
        Using H_c, compute $q(\pm 1|\mathbf{x}_i)$.
        if $q(+1|\mathbf{x}_i) > 0.5 + \varepsilon$ **then**
            Add $(\mathbf{x}_i, y_i, 1)$ to $\mathbf{S}_{right}$.
        else if $q(-1|\mathbf{x}_i) > 0.5 + \varepsilon$ **then**
            Add $(\mathbf{x}_i, y_i, 1)$ to $\mathbf{S}_{left}$.
        else
            Add $(\mathbf{x}_i, y_i, q(-1|\mathbf{x}_i))$ to $\mathbf{S}_{left}$ and $(\mathbf{x}_i, y_i, q(+1|\mathbf{x}_i))$ to $\mathbf{S}_{right}$.
        end if
    end for
    Normalize sample weights w_i in $\mathbf{S}_{left}$, so that $\sum_i w_i = 1$.
    Train left subtree with $\mathbf{S}_{left}$.
    Normalize sample weights w_i in $\mathbf{S}_{right}$, so that $\sum_i w_i = 1$.
    Train right subtree with $\mathbf{S}_{right}$.
end if

Output: Trained PBT

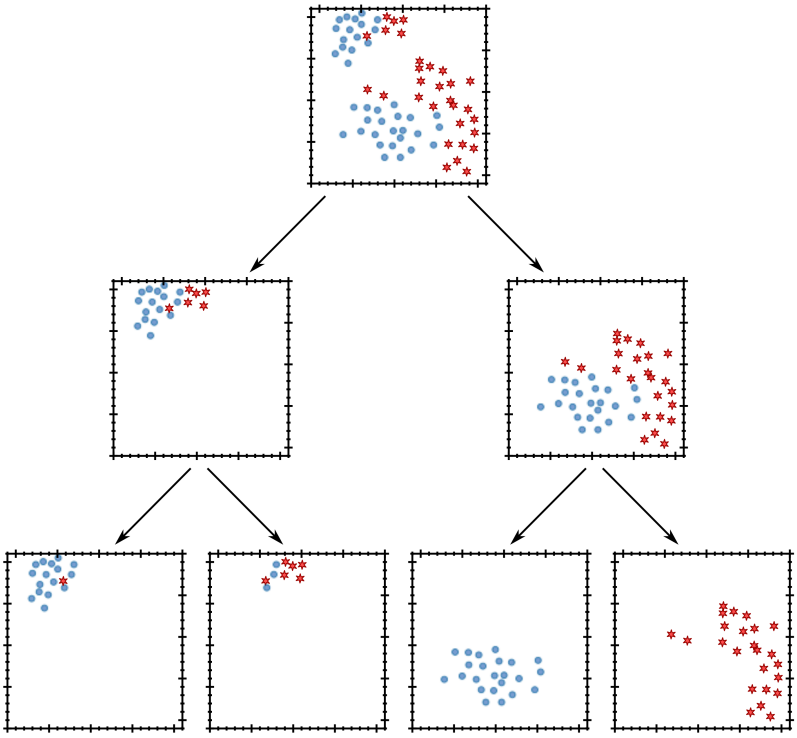


Figure 2.3: The PBT splits the feature space hierarchically similar to a decision tree. Samples can, however, end up in both subtrees.

Inference Phase

Applying a previously trained PBT to classify a new pattern works analogously: Starting from the root, the sample is passed down the tree. At each node, the strong learner's posteriors $q(y|\mathbf{x})$ are computed. Based on these, the subtrees' results are computed and combined, so that at the root node the overall approximate posterior $\tilde{p}(y|\mathbf{x})$ is returned. Whenever a strong learner is very certain about the sample's class ($q(y|\mathbf{x}) > \varepsilon + \frac{1}{2}$), the respective other subtree is omitted, or more precisely, its empirical class distribution determined during training is assumed as posterior. This on-the-fly subtree pruning can speed up classification considerably. At the same time, it does not influence accuracy negatively, since only those subtrees are omitted that have a small influence on the overall result. The final result $\tilde{p}$ can then be used to directly trade off the tree's sensitivity vs. its specificity via a single threshold. The procedure is described in detail in Alg. 3.

Since the output of the PBT is not a category but a probability value, it is strictly speaking not a classification. However, since the PBT is trained with a binary teacher signal, we will stick to this terminology, interpreting the output as confidence rated classification.

The PBT has been successfully applied to various challenging classification problems such as polyp detection in virtual colonoscopy CT images [Tu 06], detection of fetal anatomies in ultrasound images [Carn 08], and segmentation of pediatric brain tumors in MR images [Wels 08]. Its divide & conquer strategy makes it particularly well suited for problems with high dimensional feature spaces and high intra-class variability. Splitting the training samples at each node effectively subdivides the classification task by subdividing the feature space (Figure 2.3), restricting the classification in deeper tree nodes to

Algorithm 3 Calculation of the approximate posterior $\tilde{p}_c(y|\mathbf{x})$ at node c of a trained PBT for a sample $\mathbf{x}$. c_{right} here means the subtree rooted at the right child of node c, $\hat{q}_c$ denotes the weighted empirical class distribution at node c as computed during training.

Using the node's strong classifier H_c, compute $q(\pm 1|\mathbf{x})$
if $q(+1|\mathbf{x}) > 0.5 + \varepsilon$ **then**
    return $q(-1|\mathbf{x})\hat{q}_{c_{left}}(y) + q(+1|\mathbf{x})\tilde{p}_{c_{right}}(y|\mathbf{x})$
else if $q(-1|\mathbf{x}) > 0.5 + \varepsilon$ **then**
    return $q(-1|\mathbf{x})\tilde{p}_{c_{left}}(y|\mathbf{x}) + q(+1|\mathbf{x})\hat{q}_{c_{right}}(y)$
else
    return $q(-1|\mathbf{x})\tilde{p}_{c_{left}}(y|\mathbf{x}) + q(+1|\mathbf{x})\tilde{p}_{c_{right}}(y|\mathbf{x})$
end if

Output: Approximate posterior $\tilde{p}_c(y|\mathbf{x})$

smaller subspaces. Compared to a monolithic design, this leads to a more efficient problem representation, using less weak learners. Also, together with stopping the training early, it stabilizes AdaBoost: The AdaBoost classifiers are less prone to overfitting and the occurrence of conflicting hypotheses is reduced. For the PBT itself, overfitting can be controlled via the tree depth and, to some extent, via the split softness ε. With all other parameters fixed, a lower ε will lead to a stronger separation of the subtrees and thus a higher risk of overfitting.

As additional benefit compared to other classifiers, the hierarchical subdivision of the feature space together with the feature selection process inherent to AdaBoost provide insight into the nature of the classification problem at hand and the relative importance of different features. While features that are selected at early stages are highly discriminative and important for a rough estimate, those chosen in deeper tree nodes encode fine details about the decision boundary and may only be useful for a small number of samples.

CHAPTER 3

Automatic Segmentation of Liver Lesions

3.1 Related Work 27
3.2 Preprocessing 27
3.3 Pointwise Classification 29
3.4 Postprocessing 33
3.5 Results and Discussion 35
3.6 Conclusion 43

An automatic assessment of liver lesions requires automatic solutions for both their detection and segmentation. These two tasks could be carried out separately and consecutively by first finding relevant lesion locations as regions of interest (ROI) and then running a different algorithm to delineate the lesions. In contrast, the approach described on the following pages performs both detection and segmentation simultaneously. This is achieved by letting a previously trained classifier mark each single point in the input image as belonging to a liver lesion or to the background. The only required input for this system is a CT image of the patient's liver with venous contrast enhancement.

The principle of segmentation by pointwise classification is a fairly standard approach, so there are many parallels between our approach and, e. g., the one by Shimizu et al. [Shim 08]. Algorithmically important differences to their method will be pointed out along the description throughout this chapter.

Before we detail on our own approach, a brief overview of existing literature on the topic of liver lesion segmentation will be given. Next, the entire processing pipeline (Fig. 3.1) is described, from image preprocessing via the actual segmentation step to the postprocessing necessary to generate contiguous lesion masks out of the point classifications. Finally, the presented system is evaluated in a series of experiments highlighting different aspects of the method.

The work described in this chapter was published at the "20th International Conference on Pattern Recognition" (ICPR) in 2010 [Mili 10].

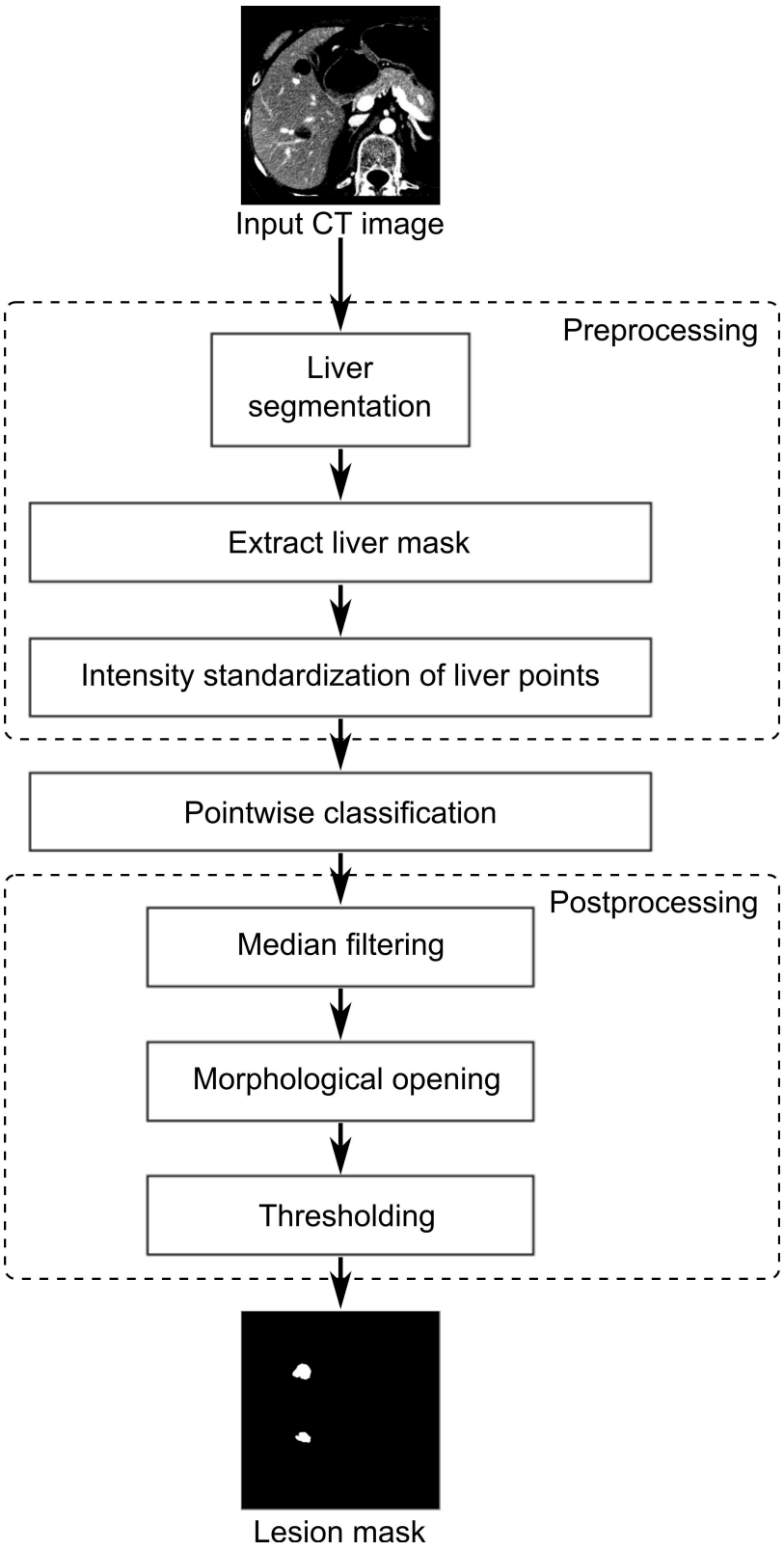


Figure 3.1: Liver lesions are segmented by classifying each voxel inside the liver as tumor or background. The probability image output by the classifier is smoothed and binarized.

3.1 Related Work

A number of automatic methods for liver lesion segmentation have been proposed over the past few years. These include histogram based methods like combining adaptive multi-thresholding with morphological operators [Bile 04] or k-means clustering on mean shift filtered images [Mass 08]. Such methods require, however, a good and fairly constant contrast between lesions and parenchyma.

Machine learning techniques like AdaBoost seem more promising in this respect due to their higher flexibility and the resulting ability to adapt to different tumor types, lesion shapes and sizes, image qualities, or contrast enhancements. They have been used to locate lesion boundaries by classifying 1D intensity profiles calculated around a manually set seed point [Li 06a] or to automatically identify tumorous tissue based on its image texture [Pesc 08]. If the lesion position is known, ML and level set methods can be used in combination to delineate its boundaries [Smee 10].

As part of the MICCAI conference in 2008, a workshop on liver lesion segmentation was held in order to compare state of the art methods. Several promising algorithms at different levels of automation were presented, the best fully automatic one yielding a Jaccard index of up to 0.71 [Shim 08] on the provided set of test images containing 10 tumors. This system, which was proposed by Shimizu et al., is the one most closely related to ours. They trained two AdaBoost classifiers with a set of gray value statistical and gradient features calculated on normalized images, as well as features based on a convergence index filter that enhances blob-like structures. One classifier was trained for segmenting large, the other for segmenting small lesions. After applying both classifiers to the points of an image separately, their results were merged to form a final output.

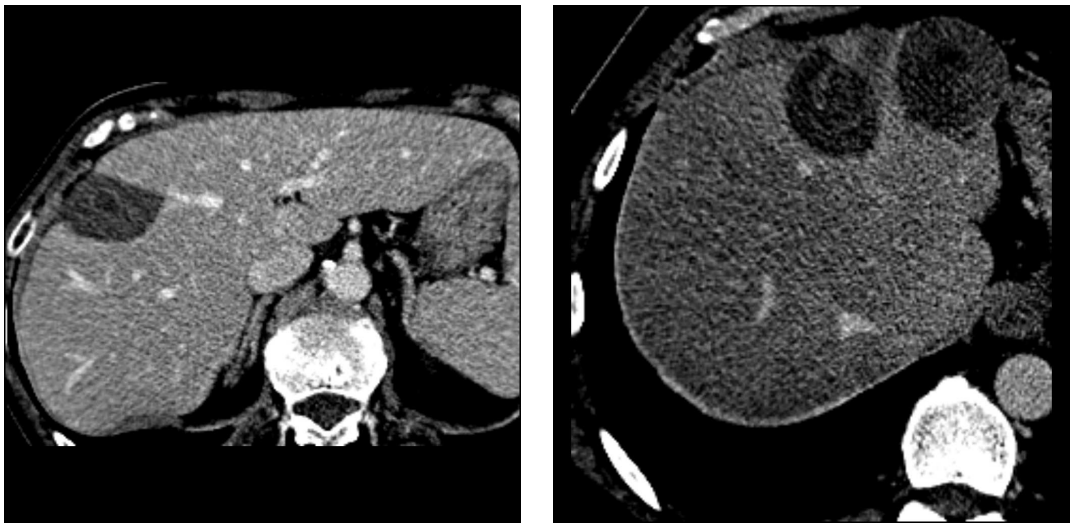
3.2 Preprocessing

Machine learning methods have proven to be highly flexible and capable of adapting to most complex environments or tasks if provided with sufficient training data. Nevertheless, in order to improve robustness of the classification and reduce the need for training data and feature dimensions, one wants to keep problem complexity as low as possible. To this end, the preprocessing step in the presented system removes several sources of variability by first automatically segmenting the liver and then standardizing image intensities in the input data.

3.2.1 Automatic Liver Segmentation

Providing a segmentation of the liver to the lesion segmentation system has two great benefits. Identifying the liver region in the image allows constraining the search space to relevant areas and thus saves computation time. From the point of view of the classifier used in the segmentation step, all image points outside the liver belong to the background. Not considering these at all therefore also reduces the complexity of the feature space and especially the intra-class variance of the background. This makes the classification task more feasible and reduces the risk of spurious detections.

The method adopted for liver segmentation here was first proposed by Ling et al. [Ling 08]. They model the liver by a hierarchical mesh-based shape representation. First,



(a) (b)

Figure 3.2: Images of two patients illustrating the need for intensity standardization. Both images were acquired with a protocol for venous contrast enhancement. Still, the tumor in (a) has approximately the same intensity as large parts of the parenchyma in (b). For both images the displayed intensity window is centered at 100 HU, with a width of 200 HU.

the liver is detected estimating its location, orientation and scale on the coarsest level using the marginal space learning scheme. Then, the model is refined applying a learning-based boundary localization which helps the system to become robust against heterogeneous intensity patterns. The liver surface is decomposed into patches depending on the surrounding anatomic structures, and patch dependent classifiers are employed to cope with the different texture patterns.

The result of this step is a binary mask of the liver, which is used to define the region of interest in the image for the intensity standardization step and all further processing.

3.2.2 Intensity Standardization

Several factors influence the distribution of contrast agent inside the liver at the time of acquisition and thus the image intensities. One factor is acquisition timing, another, even less controllable, the perfusion of the liver, which depends on the health status of the patient and his metabolism. Stenoses or other local disturbances in perfusion may also cause changes in intensity. The result is that even images that were acquired during the same phase of contrast enhancement can have a very different overall intensity level.

In order to make the input images more comparable, intensities inside the liver are standardized. This is a fairly common preprocessing step in segmentation. Its realization, however, in our case differs from the standard approach. In most cases, voxel intensities I are either modified using histogram equalization [Pesc 08], which has a mainly visual effect, or normalized according to [Shim 08]

$$I' = \frac{I - \mu}{\sigma} \tag{3.1}$$

with I' being the processed intensity and μ and σ being the mean and standard deviation of intensities inside the liver. For simple cases like small, strictly hypodense lesions in otherwise healthy livers, this will usually be sufficient. In more complex cases, however, like partly hyperdense or rim-enhancing lesions, cirrhosis or other basal diseases, the histogram of the liver voxels may be multi-modal with varying distances between the single modes, so that normalization will not match intensity ranges across images correctly.

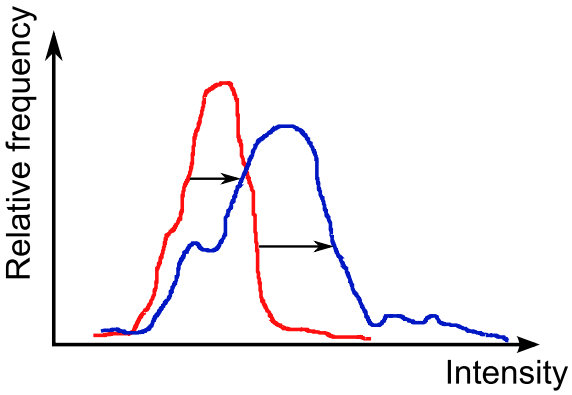


Figure 3.3: The histogram of the target image (red) is non-rigidly registered to the histogram of the reference image to determine the intensity mapping for the standardization.

To be able to handle these cases as well, we took a different approach, which was originally developed for standardizing intensities in nuclear magnetic resonance images (MRI) and was there applied to head and even full body images [Jage 09]. The idea is to select a representative image, in which the liver has an appearance that is considered typical for the database. This image's intensities' probability density function inside the liver defines the ideal histogram. Each target image is then standardized by matching its histogram non-rigidly to that of the reference image and applying the resulting intensity mapping to the intensities in the target image (Fig. 3.3).

3.3 Pointwise Classification

The core of the system, responsible for the actual segmentation, consists of a classifier that assigns a value in the range $[0,1]$ to each voxel within the liver. This value reflects the estimated posterior probability $p(y = 1|\mathbf{x}_a)$ of the voxel belonging to a liver lesion. It is calculated by a cascade of previously trained Probabilistic Boosting Trees (PBT), based on a vector of features $\mathbf{x}_a$ describing the appearance of the voxel in the image. Strictly speaking, the result is not a real classification, but since the PBT is trained with class labels as target values and the output values are a mere confidence rating, we will stick to this terminology.

There are two standard ways to use classification in object detection: window classification and point classification. The former receives a large number of rectangular regions of various sizes and positions in the image (windows) and decides which ones contain a target object. The latter decides for each single point in the image, whether it belongs to a target object, providing a segmentation at the same time. While window classification allows the simple incorporation of features describing an entire target object like in template matching approaches, point classification has the advantage of allowing the detection of objects of arbitrary size and shape. Especially in the case of liver tumors, which vary greatly in their size and appearance, this is an invaluable side effect. Also, the point classification approach requires a much smaller patient database for classifier training, since each single tumor point can function as a positive example, as opposed to each tumor in the window classification case.

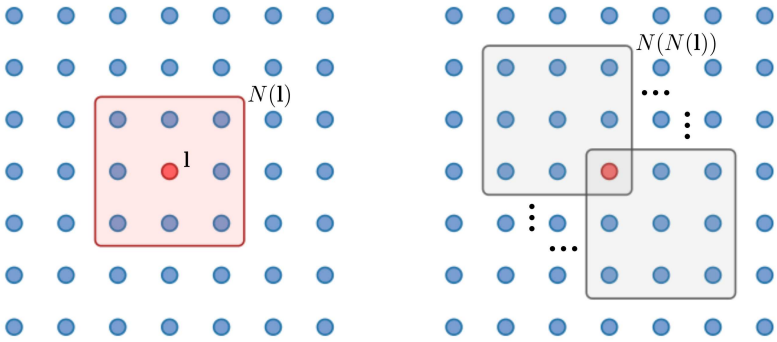


Figure 3.4: The red point in the center marks the current voxel, at image location **l**. The boxes denote the neighborhood $N(\mathbf{l})$ around this voxel (left) and the neighborhoods around the voxels in $N(\mathbf{l})$ (right). $N(\mathbf{l})$ marks the region from which the probability features for the voxel at location **l** are calculated.

3.3.1 Iterative Classifier Setup

The highly variable appearance of both parenchyma and lesions makes it difficult for a single classifier like AdaBoost or a support vector machine (SVM) to globally find an appropriate model for each target class and thus an optimal decision boundary in the input space. To be able to account for at least different lesion sizes and thus eliminate one source of variation, Shimizu et al. [Shim 08] used two AdaBoost classifiers in their segmentation step. The approach presented here is more general, scaling well for segmentation tasks of different complexities. The classifier we chose to adopt is the recently proposed Probabilistic Boosting Tree [Tu 05] in combination with AdaBoost as strong– and decision stumps as weak learners (compare Sec. 2.2). Due to its hierarchical nature, the PBT should be able to handle the high dimensional feature space at hand, as well as the high intra-class variability of the task.

One drawback of the voxel classification approach is the fact, that it treats the classification of each point in the image as an independent problem, which is obviously not true when segmenting contiguous objects. This wrong assumption is usually compensated for by incorporating context information into the classification, e. g. by designing special features or averaging features over some neighborhood. An entirely different approach is proposed by Morra et al. [Morr 08], who exploit the fact that neighboring points with similar properties tend to belong to the same class. The rationale is the following: When classifying a point it would be helpful to know the correct label of the surrounding points and incorporate this information into the decision making. In practice, during application phase class labels are not available, but the classifier does provide an estimate of the posterior probabilities. Consequently, each point's classification is formulated to depend on that of the surrounding points. Since these depend on the result for the current point, this leads to an iterative scheme, in which all points initially have the same lesion probability of 0.5 and are updated alternatingly until convergence. The final estimated posterior prob-

ability of the voxel at location $\mathbf{l}$ with corresponding feature vector $\mathbf{x}_a$ after k iterations is then calculated as:

$$\begin{aligned}
&k=0:\ p(y|\mathbf{l}) = p_0(y|\mathbf{l}) = p_0(y|\mathbf{x}_a)\\
&k=1:\ p(y|\mathbf{l}) = p_1(y|\mathbf{l}) = p_1(y|\mathbf{x}_r) = p_1\left(y|\mathbf{x}_a, \{p_0(y|\mathbf{l}^*) \mid \mathbf{l}^* \in N(\mathbf{l})\}\right)\\
&\dots\\
&k=K:\ p(y|\mathbf{l}) = p_K(y|\mathbf{l}) = p_K(y|\mathbf{x}_r) = p_K\left(y|\mathbf{x}_a, \{p_{K-1}(y|\mathbf{l}^*) \mid \mathbf{l}^* \in N(\mathbf{l})\}\right) \qquad (3.2)
\end{aligned}$$

$N(\mathbf{l})$ here denotes the voxel locations in a neighborhood of location $\mathbf{l}$ (Figure 3.4) and, accordingly, $\{p_i(y|\mathbf{l}^*) \mid \mathbf{l}^* \in N(\mathbf{l})\}$ denotes the probability values at these locations as provided by classifier i. We incorporated this scheme into our voxel classification step by training a sequence of PBTs. Each PBT in the sequence receives as input not only the feature vector $\mathbf{x}_a$ describing the appearance of the voxel under consideration, but also a vector $\mathbf{x}_p$ of features calculated from the output probability image of the preceding classifier (Figure 3.5). Both feature vectors are concatenated as $\mathbf{x}_r = (\mathbf{x}_a, \mathbf{x}_p)$ to form a combined feature space for the PBT. That way, when classifying a point, previously gained knowledge of the classes of surrounding points can function as a prior. This iterative training procedure is also described in Alg. 4. For the task at hand three to four iterations turned out to be sufficient; further steps rarely brought any benefit.

Algorithm 4 Training procedure for the iterative classification scheme

Input: $\mathbf{V} = \{(\mathbf{x}_{a,1}, y_1), \dots, (\mathbf{x}_{a,n}, y_n)\}$ where the $\mathbf{x}_{a,i} \in \mathbb{R}^m$ were calculated from image locations $\mathbf{l}_i$ in the training images and $y_i \in \{-1, +1\}$. Number of iterations $K+1$.

1: Train classifier PBT_0 using examples $\mathbf{V}$
2: **for** $k = 1 \dots K$ **do**
3: Use PBT_{k-1} to classify the training images, i. e. for every single voxel location $\mathbf{l}_j$ in the images, calculate the corresponding feature vector $\mathbf{x}_{a,j}$ and the posterior $p(y|\mathbf{x}_{a,j})$ (cf. Alg. 3).
4: For each training sample location $\mathbf{l}_1, \dots, \mathbf{l}_n$, calculate probability feature vector $\mathbf{x}_{p,i}$ from the output of PBT_{k-1}
5: Concatenate $\mathbf{x}_{a,i}$ and $\mathbf{x}_{p,i}$ to $\mathbf{x}_{r,i}$ to form new set of training examples $\mathbf{V}_k = \{(\mathbf{x}_{r,1}, y_1), \dots, (\mathbf{x}_{r,n}, y_n)\}$
6: Train classifier PBT_k using examples $\mathbf{V}_k$
7: **end for**

Output: Cascade of classifiers $\mathrm{PBT}_0, \dots, \mathrm{PBT}_K$

3.3.2 Feature Calculation

Selecting features describing the objects to be classified is a crucial step in designing a classification system. Only if they capture the full variability of the objects and the differences between the classes, the classifier can learn to distinguish them. At the same time adding new dimensions means additional complexity for the classifier and can decrease robustness. AdaBoost in the version that is used here does not suffer from this latter issue

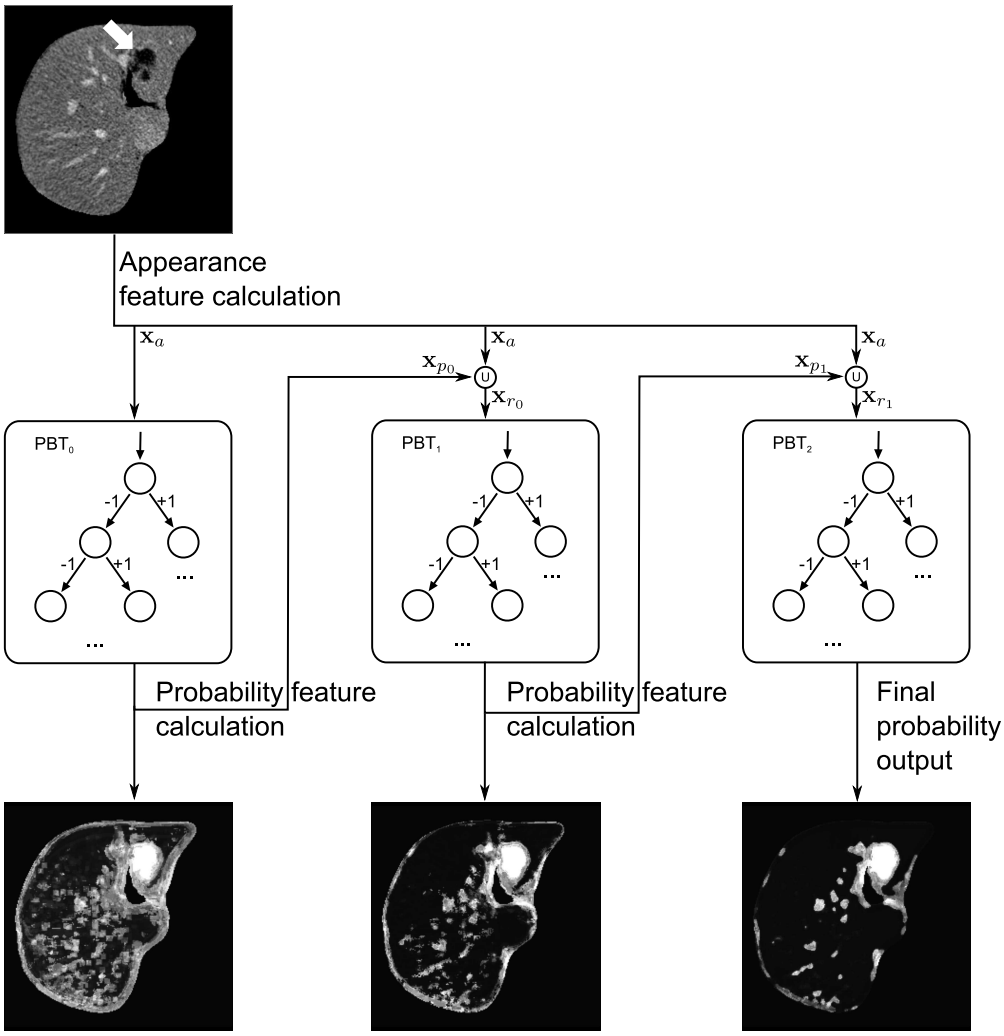


Figure 3.5: The iterative classification scheme used for separating tumor from background voxels, here with three iterations (in our experiments three to four iterations turned out to be sufficient for convergence). Each voxel of the input image is classified by each PBT sequentially. The feature vector $\mathbf{x}_r$ consists of appearance features $\mathbf{x}_a$ calculated from the input image and probability features $\mathbf{x}_p$ calculated from the output of the previous iteration. In the output images, bright parts denote voxels that were assigned a high lesion probability by the classifier. The arrow in the input image shows the location of the target lesion.

due to its built-in feature selection mechanism that restricts the learning to the most relevant dimensions. This is not only convenient but also opens up new worlds of features that might be too weak to be used in other classifiers. Instead of carefully selecting a small number of strong and uncorrelated features, an AdaBoost user can work with a large number of features and experiment with new types. Not only will AdaBoost still learn the task, it will also provide information about which features are the most useful ones.

A good example for this policy are the rectangular Haar-features often used in object detection. Originally, these were designed for a license plate detection task [Papa 98, Papa 00]. With the introduction of integral images by Viola and Jones [Viol 01], these became a standard set of features for object detection. Each feature is calculated as the difference between the integral over two or more rectangular image regions around the current location $\mathbf{l}$. A number of different combinations have been proposed in 2D [Papa 00, Viol 01, Lien 02], 3D [Tu 06] and arbitrary dimensions [Feul 11a]. Calculating these at various scales leads to a number of features that can easily reach several hundreds of thousands, often even exceeding the number of training examples.

For detecting and segmenting liver lesions, we designed a set of features capturing different aspects of the appearance of liver parenchyma, lesions, their borders, as well as other structures in the liver that might be confused with the target lesions. The feature set comprises

- intensity statistics of neighborhoods of various sizes,
- the skewness of an intensity profile sampled across the point under consideration,
- gradient based features, including a simplified version of the adaptive convergence index [Shim 05, Shim 08],
- a number of Haar-like features, as well as
- a vesselness measure [Sato 97].

The adaptive convergence index is a gradient based filter for enhancing spherical structures and was proposed by Shimizu et al. for detection of HCC. A detailed list of the features is given in Tab. 3.1.

In contrast to other approaches [Shim 08, Li 06a], our neighborhood is computed not on a voxel but on a millimeter scale, making the approach robust against the use of images acquired with different CT scanners and acquisition protocols.

The features that are calculated from the intermediate probability output images to form the feature vector $\mathbf{x}_p$ are the point's own probability value, as well as some simple statistics and weighted sums of the values in the point's neighborhood (cf. Tab. 3.2).

3.4 Postprocessing

The iterative classification in the previous step smooths larger regions in the output image as well as the course of lesion boundaries, rendering any elaborate postprocessing unnecessary. The final probability map is therefore only treated with a median filter and a morphological opening operation with a kernel size of $4 \times 4 \times 4$ in order to eliminate last small and isolated false positive detections. Finally, the image is converted into a lesion candidate mask by thresholding the probability values.

Feature category	Feature name	Scales/Neighborhoods	Total #
Intensity statistics	Intensity		1
	Max	1.2 mm – 16.0 mm	5
	Min		5
	Range		5
	Contrast		5
	Mean		5
	Variance		5
	Skewness		5
	2D Median		5
	3D Median		5
	Difference from global mean	1.2 mm – 4.0 mm	3
	Absolute difference from global mean		3
	Mean profile skewness		3
	Max profile skewness		3
Gradients	Central differences in x, y, z		3
	Sum of squared central differences		1
	Sobel operator		6
	Laplace operator 2D		1
	Adaptive convergence index	overlap 1.0 mm – 5.0 mm, radius 2.0 mm – 10.0 mm	6
Vesselness	Sato vesselness		1
Haar features	Edge in x, y, z	0.8 mm – 41.0 mm	735
	Line in x, y, z		735
	Diagonal in xy, xz, yz		735
	Center-surround		245
			2526

Table 3.1: Appearance features $\mathbf{x}_a$ used to separate tumor from background points.

Feature category	Feature name	Scales/Neighborhoods	Total #
Intensity statistics	Intensity		1
	Mean	0.8 mm – 16.0 mm	6
	Sum over 2D surrounding		6
	Sum over 3D surrounding		6
	Variance		6
	2D Median		6
	3D Median		6
	Gaussian-weighted sum in 2D		6
	Gaussian-weighted sum in 3D		6
	Difference from global mean		6
	Absolute difference from global mean		6
			61

Table 3.2: Features $\mathbf{x}_p$ calculated from the probability image generated by the previous classification stage.

3.5 Results and Discussion

To assess the system's performance, a series of experiments was conducted, each highlighting a different aspect of the algorithm. In total, 15 CT liver datasets with venous phase contrast enhancement from three different clinical sites were used in the experiments. All images had been acquired at 120 kVp. Their voxel resolution varied from 0.547 to 0.832 mm in x and y directions, and 1 to 3 mm in z direction. For training and testing, datasets were subsampled to a slice thickness of 3 mm where applicable. As mentioned in Sec. 1.4, the images contained only lesions that were mainly hypodense.

Each single experiment was set up as a 5-fold cross-validation. During each fold, 12 images were used to train the classifiers and the remaining ones were held out as independent test set.

3.5.1 Subsampling of Training Data

While the tests were performed on all available points of the testing images, the training images were randomly subsampled to reduce training time. The subsampling was not uniform: Homogeneous image regions can easily be represented in the training data by only a few points. On the other hand, regions around tumor boundaries and blood vessels, as well as regions with generally high entropy, contain a high degree of variation which is difficult to capture in a few examples. The sampling routine therefore focused on these difficult regions to ensure that in the training set each image region was represented proportionally to its importance, not its size.

A sampling probability map for each training image was created by combining two filtered versions of the image:

- One was based on each point's absolute distance to the closest lesion boundary, with a value of 1 at the boundary and falling off exponentially from there.

- The other was based on the variability of the intensity image, measuring for each point the absolute difference in intensity between the local neighborhood and the global mean. Only the top 0.5% of the points were kept, with their values scaled to the range $[0, 1]$.

Both images were combined additively, plus a small constant to not completely suppress samples outside these regions. Training images were then subsampled without replacement according to these sampling maps.

The learning algorithm was started with 50,000 positive and 50,000 negative instances. During training, additional samples could be drawn whenever necessary. While this is a rather coarse subsampling, experiments indicate, that it does capture the entire variation contained in the training data.

3.5.2 Evaluation Measures

For segmentation quality assessment, receiver operator characteristics (ROC) curves were generated. ROC curves are one common way of presenting the performance of a classification or segmentation algorithm graphically, plotting the system's sensitivity against its false positive rate. The point of perfect classification in these diagrams is the upper left corner; the closer a curve is located towards this corner, the better is the classifier it was generated by [Fawc 06].

Since the output of the presented algorithm is a probability map containing a lesion probability for each individual image location, a threshold has to be applied to determine the final segmentation. Varying this threshold from 0 to 1 and calculating sensitivity and specificity values from the resulting confusion matrices (Sec. A.1, Tab. A.1) yields the points of the ROC curve. Here, each curve is the result of a full cross-validation experiment and is obtained by calculating the confusion matrices over all datasets.

In addition to the ROC curves, several overlap measures were calculated in the experiments (Appendix A, Tab. A.2) to allow a quantitative assessment of the results. The Jaccard and Dice similarity coefficients are widely used for evaluation of segmentation methods. Like sensitivity and specificity, they both try to capture the accuracy of a method by measuring the similarity of the reference segmentation and the algorithm result, however taking into account both over- and under-segmentation.

While these measures allow a simple comparison of different methods, their meaning is hard to grasp intuitively, especially in a domain with very few positive samples such as the problem at hand. Thus, we resorted to a measure with a simple intuitive meaning, namely the positive predictive value, also known as precision. Precision denotes the probability of a positive decision made by the classifier being correct. While for a cancer patient the sensitivity seems most important, precision is equally important for physicians in clinical routine, as it describes how reliable the decisions by the system are and thus how many false positives one has to expect on average. The simplest and most intuitive performance measure in this respect is the ratio of false positives and true positives $\frac{\#fp}{\#tp}$, since it describes directly what to expect visually.

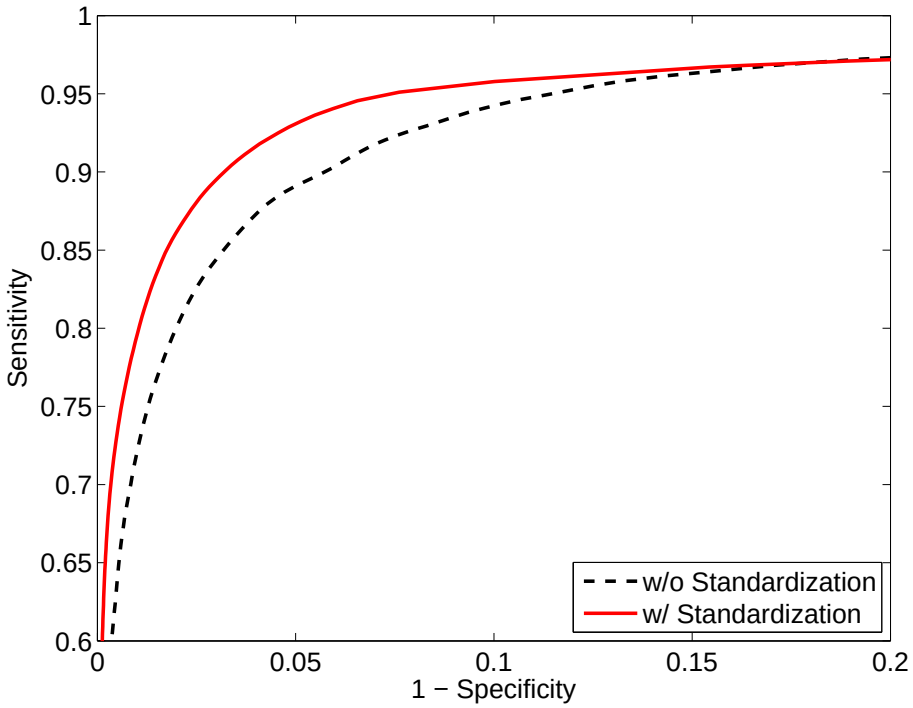


Figure 3.6: ROC curves comparing the classification performance with and without the standardization step during preprocessing. The shown curves represent a system designed with only one PBT, which is equivalent to using the output of the first iteration in our scheme as result. Standardizing image intensities clearly helps the PBT by simplifying the task.

3.5.3 Effect of Standardization on Segmentation

Figure 3.6 compares the classification performance of PBTs working with standardized images and those having to deal with the original intensities. This experiment was run without the filter based postprocessing in order to show only the accuracy of the classifier. Also, no iterative classification was used. The system using intensity standardization clearly outperforms the one using the unprocessed images, showing an increase in precision of 39.2% at 90% sensitivity (cf. Tab. 3.3). While the PBT in principle is capable of finding the decision boundary with the original images as well, simplifying the task will certainly result in more compact and robust classifiers.

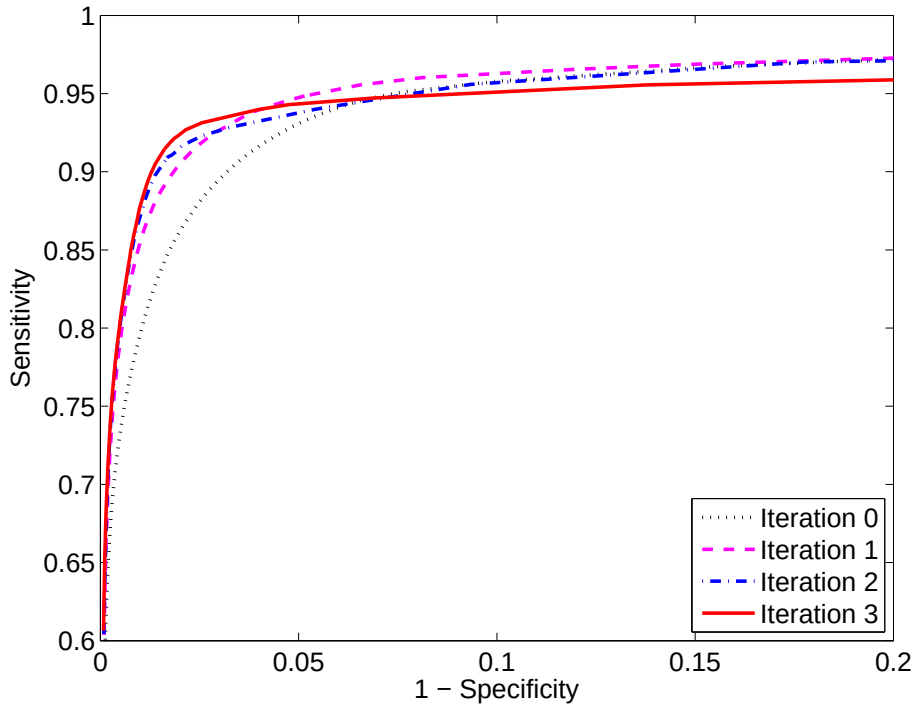
3.5.4 Effect of Iterative Classification on Segmentation

Iteratively classifying image voxels can stabilize the classification, resulting in smoother and more homogeneous probability images. Figure 3.7 indicates that the provided probability features were extensively used by the classifiers. All classifiers use the same set of appearance features, yet when adding the probability features, each classification stage outperforms the preceding one. In this cascade, the performance gain becomes smaller with each step. While adding the first classifier with probability features improves precision at the point of 90% sensitivity by 26.6%, the next two steps yield only improvements of 8.6% and 3.3%, respectively (cf. Tab. 3.3). The improvement achieved by training more than three iterations turned out to be marginal in most experiments. If the filter based postprocessing is activated, the gain becomes even smaller, as can be seen from Fig. 3.7 (b). Like the iterative classification, the filters smooth regions in the probability image as well as lesion borders, and remove small false positive detections. Figure 3.8 shows how similar the effects of both techniques are. While in the first iteration the postprocessing has a visible effect on the ROC, in later iterations an effect is only noticeable in those parts of the curve with low specificity. Still, deactivating the postprocessing completely is not recommended. Not all isolated misclassifications are removed by the iterative classification, so that the postprocessing filters are necessary to reduce the number of false positive detections.

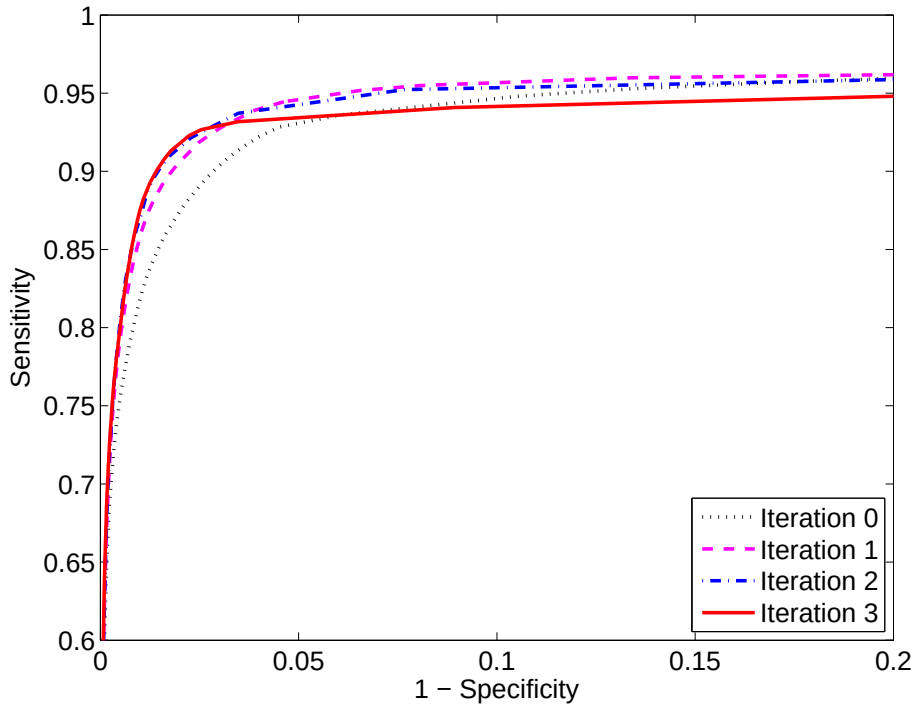
Figure 3.9 finally shows curves for a system trained and tested on images that were not standardized. The same effect as in Fig. 3.7 can be observed. As with the postprocessing, the improvement caused by the standardization step becomes smaller in later iterations, showing how heavily the classifier relies on the probability features in these cases.

Table 3.3 contains the performance figures for the segmentation evaluation without postprocessing. All numbers are calculated for the threshold value yielding 90% sensitivity. The effect of the iterative scheme is clearly visible: All specificity related measures improve considerably in later iterations, the improvement between the iterations being higher in the early steps.

Table 3.4 presents the maximum Jaccard and Dice similarity coefficients that could be yielded in the experiments for a single threshold, together with their corresponding sensitivity values. In contrast to the previous experiments, these values were determined not by setting the desired sensitivity. Instead, for each cross-validation experiment and each iteration the single threshold value was determined that yielded the maximum average Jaccard and Dice similarity over all images in the experiment.



(a)



(b)

Figure 3.7: ROC curves visualizing the performance gain introduced by the iterative classification scheme. In comparison to the original output of the classifiers (a), the curves generated after postprocessing (b) are only slightly shifted towards the upper left corner.

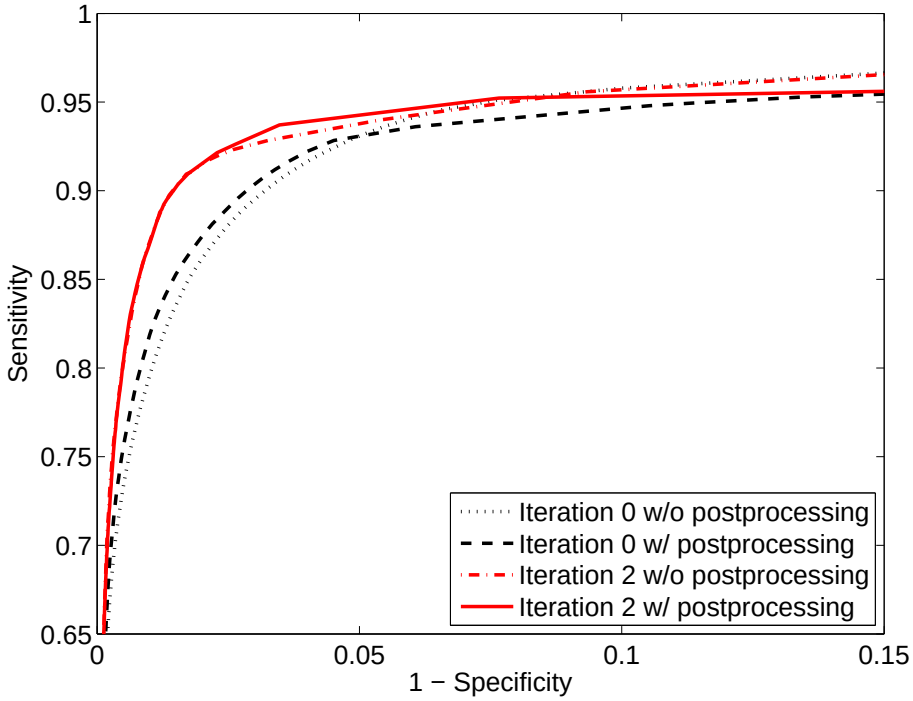


Figure 3.8: ROC curves of two exemplary iterations, directly comparing the results with and without postprocessing. In later iterations, the influence of the postprocessing diminishes: The classifier result is stabilized by the iterative scheme, reducing the number of small false positive detections, which would otherwise be removed by the postprocessing filters.

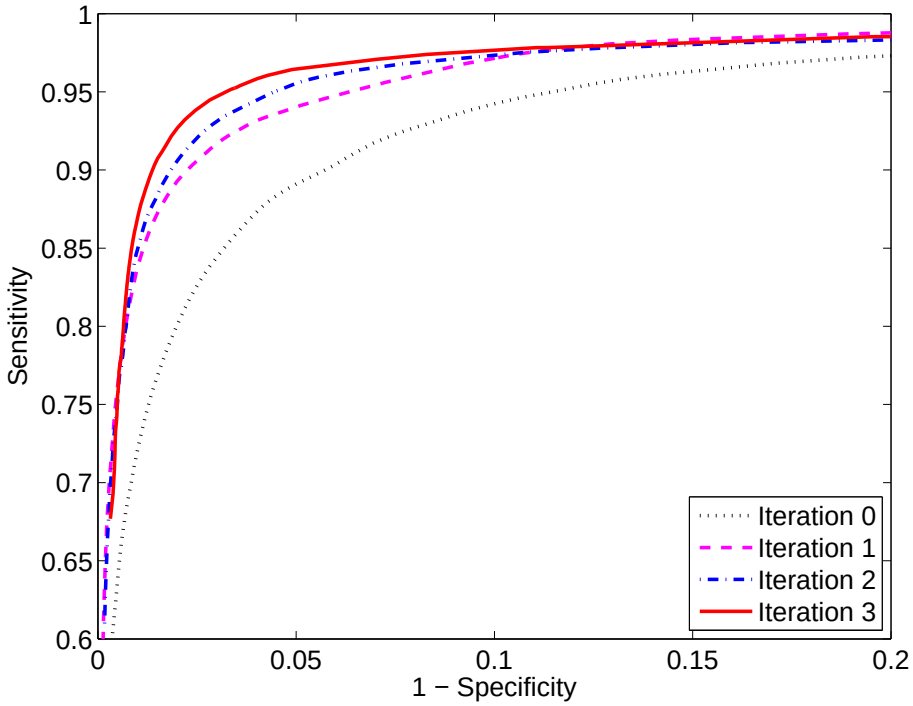


Figure 3.9: ROC curves visualizing the influence of iterative classification on a system trained without intensity standardization.

	w/o Standardization				w/ Standardization			
	Iteration							
	0	1	2	3	0	1	2	3
Sensitivity (%)	90	90	90	90	90	90	90	90
Specificity (%)	94.3	97.7	98.2	98.6	96.8	98.2	98.6	98.7
Jaccard index	0.35	0.55	0.60	0.65	0.48	0.60	0.64	0.66
Dice coefficient	0.51	0.71	0.74	0.79	0.65	0.75	0.78	0.80
Precision (%)	36.2	59.0	63.4	70.3	50.4	63.8	69.3	71.6
#fp/#tp	1.76	0.70	0.57	0.42	0.98	0.57	0.44	0.40

Table 3.3: Comparison of segmentation performance of different classifiers. All numbers were calculated from the unprocessed classifier output, i. e. with no filter based postprocessing involved.

	w/o Standardization				w/ Standardization			
	Iteration							
	0	1	2	3	0	1	2	3
Jaccard index	0.57	0.67	0.67	0.69	0.64	0.70	0.71	0.71
Dice coefficient	0.72	0.80	0.80	0.82	0.78	0.82	0.83	0.83
at sensitivity (%)	68	78	83	84	72	79	81	81

Table 3.4: Maximum values for Jaccard and Dice similarity coefficients, measured from classifier output without filter based postprocessing.

3.5.5 Lesion Detection Performance

The presented segmentation method requires no separate lesion detection step beforehand. Instead, detection and segmentation of lesions are performed simultaneously by classifying each single voxel within the liver. While it does not seem reasonable to discard the segmentation and use the method for detection purposes only, it is still interesting to see, how many lesions can be found by the algorithm and how many false alarms are raised, since this has a major influence on its acceptance by clinicians. Unfortunately, this is rarely done in literature. Massoptier and Casciaro [Mass 08] do report figures on lesion detection performance, however, they do not clearly state their criteria for considering a lesion correctly detected or not, which makes a comparison difficult.

Thus, detection quality is assessed in this section in a lesion based evaluation. The images in these experiments all underwent the filter based postprocessing. Lesion candidates were identified from the segmentation mask generated by the algorithm by running a connected components analysis. A reference lesion was considered detected if

1. its axis aligned bounding box overlapped with the bounding box of a lesion candidate so that the candidate covered at least 25% of the reference and

2. the candidate box's center lay within the reference box.

While the first criterion measures only the sensitivity of the method, the second one ensures a certain specificity as well, since it penalizes too large candidates. Any candidate that was not matched by these criteria for a reference lesion was considered a false positive or false alarm. The entire analysis was restricted to objects with a volume larger than 0.125 ml, which corresponds to a cube with 5 mm edge length.

Applying these criteria to the final output of the system, we achieved a maximum detection rate of 68% in the last iteration at 3.4 false positives per correctly detected lesion, which equals 10.5 false positives per patient on this database.

In Fig. 3.10 and Fig. 3.11, the detection sensitivity was plotted against the false alarms per volume and per true detection. Each data point in the scatterplot corresponds to one value of the threshold that is applied to the probability output of the segmentation algorithm. The points were not interpolated to form a curve, because the detection criteria do not produce a continuous output for increasing values of the probability threshold.

While the false positive rates appear very high, closer examination reveals that most of the false positives are located at the liver boundary or in fissures and exhibit characteristic shapes, so that they could probably be filtered using another classifier. Experimental support for this conjecture, however, has to be provided in follow-up projects.

A look at the segmentation result generated with the same threshold as for the maximum detection sensitivity value illustrates the differences in the requirements for the two tasks: The corresponding segmentation sensitivity is as high as 94.3%, however at a specificity of only 95.3%, meaning there are 1.39 false positive voxels per true positive. This impression is confirmed when Tab. 3.5 is considered. This table gives the detection results of each configuration and classifier stage at the settings yielding 90% segmentation sensitivity. At 61.7%, detection sensitivity is far below its maximum value of 68%, however, at the same time detection errors are much lower as well (1.7 vs. 3.4 false positives per correctly detected lesion). Of course, the mentioned classification based candidate filter would

improve specificity not only for detection but also for segmentation, potentially allowing lower thresholds and thus a higher sensitivity.

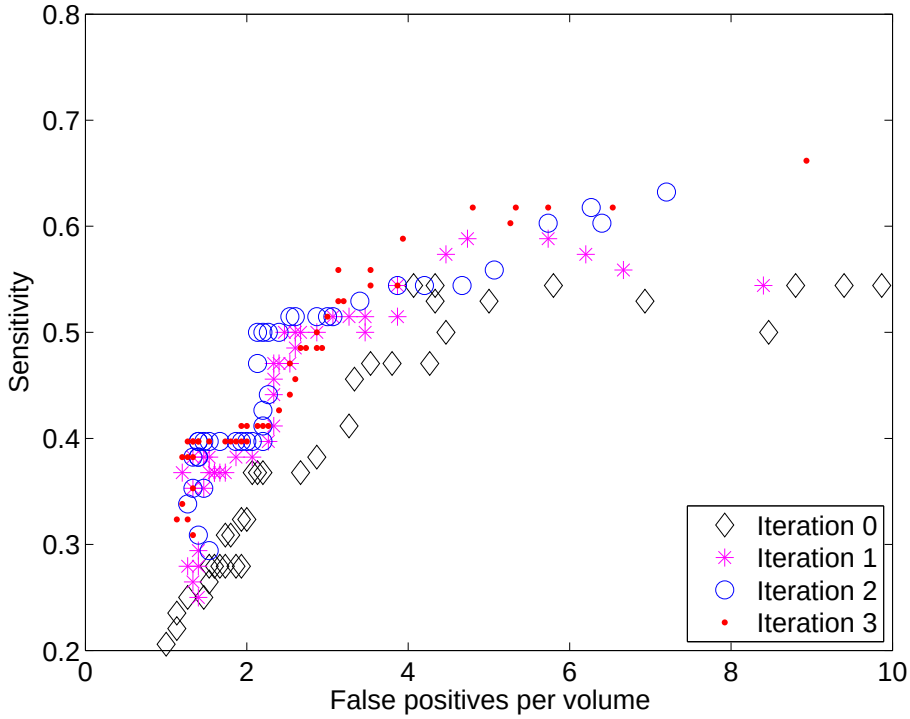
	w/o Standardization				w/ Standardization			
	Iteration							
Method	0	1	2	3	0	1	2	3
Segmentation sensitivity (%)	90.0	90.0	90.0	90.0	90.0	90.0	90.0	90.0
Detection sensitivity (%)	47.7	41.5	41.5	50.8	54.4	54.4	58.9	61.7
Detection precision (%)	18.1	26.2	28.7	40.1	18.9	25.2	32.0	36.8
Detection #fp/#tp	4.5	2.8	2.5	1.5	4.3	3.0	2.1	1.7
Detection #fp/volume	9.3	5.1	4.5	3.2	10.6	7.3	5.7	4.8

Table 3.5: Detection quality at the setting yielding 90% segmentation sensitivity.

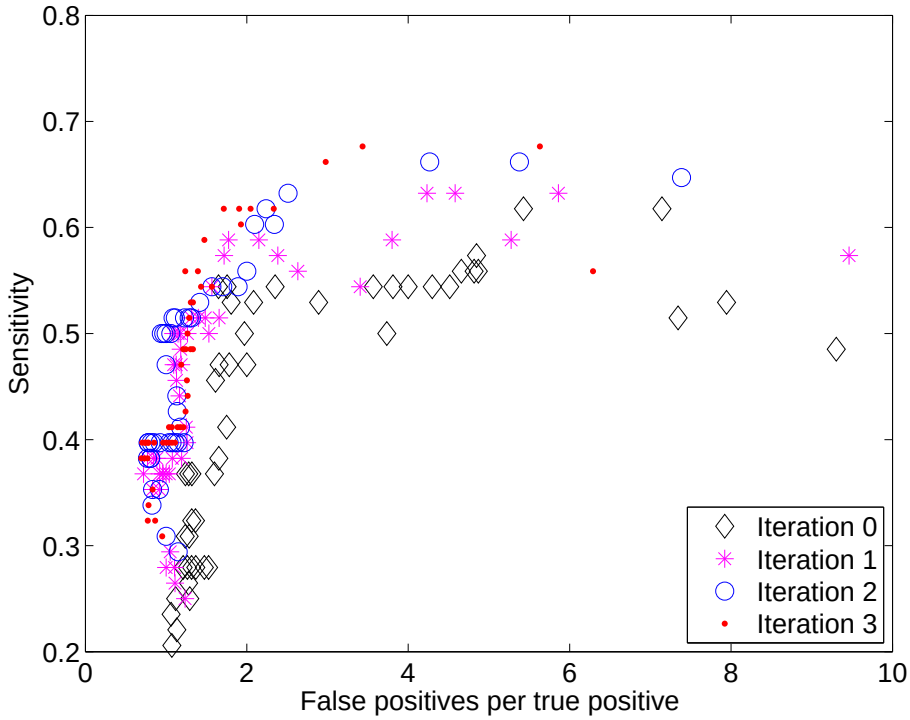
3.6 Conclusion

In this chapter, we presented a system for automatic detection and segmentation of focal liver lesions in CT images. It is based on pointwise classification of liver voxels. Compared to previous approaches, we incorporated a novel intensity standardization step adopted from MR imaging. For classification we used a cascade of Probabilistic Boosting Trees instead of a single classifier, increasing flexibility and improving the handling of complex input spaces. Both measures account for an increased robustness of the system, setting track for the segmentation of difficult cases like rim enhancing lesions.

The validity of the approach was shown in an experimental evaluation with 15 clinical datasets containing mostly hypodense liver lesions.

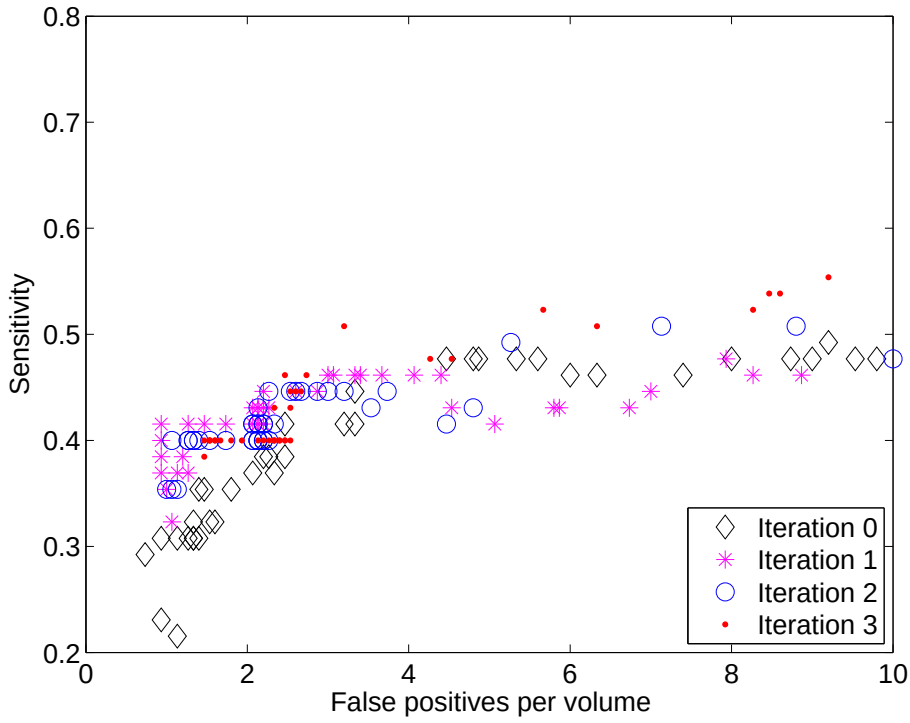


(a)

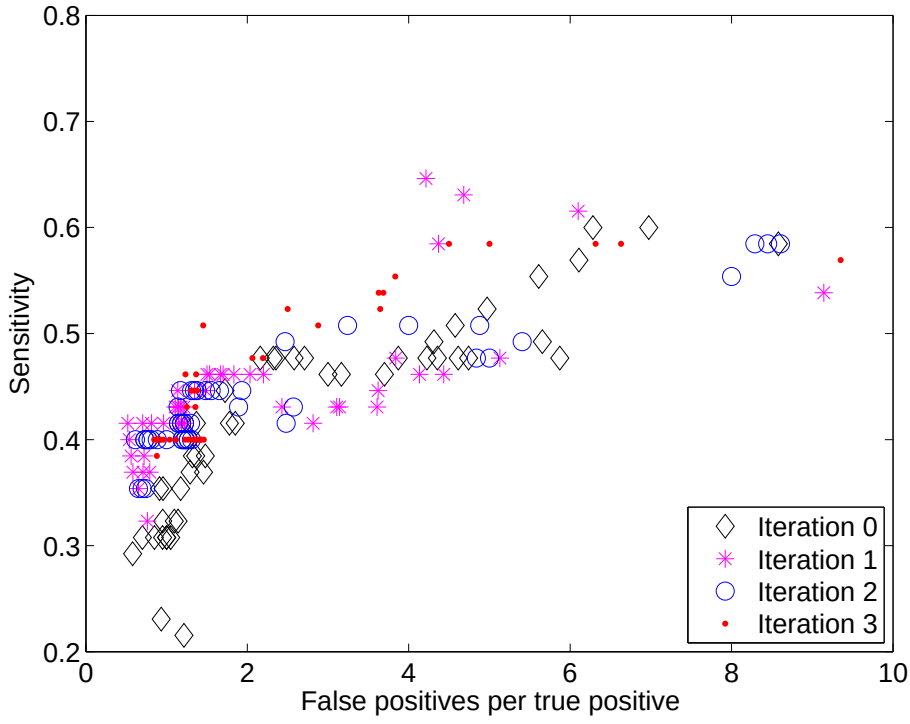


(b)

Figure 3.10: Detection performance of classifiers when using standardized data. Each marker represents the result for one value of the threshold applied to the output probability image.



(a)



(b)

Figure 3.11: Detection performance of classifiers when using original image intensities. Each marker represents the result for one value of the threshold applied to the output probability image.

CHAPTER 4

Learning a Prior Model for Lesion Follow-up Segmentation

4.1 Related Work 48
4.2 Processing Pipeline for Liver Lesion Segmentation 51
4.3 Spatial Prior Models for Follow-up Segmentation 52
4.4 Learning a Prior Model for Follow-up Segmentation 55
4.5 Results and Discussion 58
4.6 Conclusion 69

As described in Sec. 1.3.2, there are many scenarios in which liver tumors are not removed surgically. Such tumors have to be monitored using, e.g., CT images in order to detect growth or shrinkage and thus rate disease progression or treatment response. To this end, after the initial examination, follow-up exams are performed every few months. During these, images are acquired and the lesion sizes are measured and compared. As a patient may have well over a dozen lesions, performing these measurements manually is error prone and tedious. The RECIST guidelines for lesion assessment in follow-up examinations [Eise 09] therefore define their criteria for disease progression based on the largest axial diameters of a subset of representative lesions, selected by the physician. While this approach somewhat reduces the effort for the physician, it also introduces new sources of error: Changes might occur either in lesions that were not selected as target lesions and are thus not monitored, or in lesions that are monitored, but grow or shrink in a direction other than the measured one.

Hence, there is a great need for automatic segmentation algorithms to overcome the limitations of this manual approach, allowing volumetric measurements and taking into account all liver lesions of the patient. Follow-up examinations are therefore the primary use case for such automatic segmentation methods. While in principle any of the automatic lesion segmentation methods presented in Chap. 3 could be used in this setting, the fact that several images of a single patient have to be segmented raises the question whether the results of the earlier examinations might be helpful for the later ones.

Incorporating prior knowledge is a common way of preventing a segmentation procedure from choosing a wrong parameterization or evolving in the wrong direction. This can be knowledge of the expected appearance of the image or target objects, of their size, or of

their location. For level set methods, for instance, the use of shape priors is well established [Rous 02, Nain 04, Farz 10]. These are modeled manually [Nain 04], statistically from a set of training shapes [Rous 02], or even dynamically [Farz 10], and integrated into the level set function.

Statistical atlases, consisting of a mean shape and a notion of frequent deviations, have been used in organ segmentation tasks [Lotj 10, Zhan 06]. The idea of a statistical atlas has been transferred to lymph node detection by Feulner et al. [Feul 10]. In an offline training phase, they generate an atlas of common lymph node locations from a patient database. For an image of a new patient, a spatial prior is calculated by registering the image to the atlas and combining the atlas information with a multi-organ segmentation of the patient. The prior is then multiplied with the probability output of their classification based lymph node detector to steer its attention.

Adopting the idea of prior knowledge for the problem at hand, we do not treat a series of images of the same patient as several isolated segmentation tasks, but instead use information gained from earlier images to improve the results on later ones.

To this end, we propose a novel learning based approach to generating priors for follow-up lesion segmentation. This chapter describes, how such a prior can be trained and integrated into our lesion segmentation system, discussing benefits and drawbacks of different integration methods. This work was already published in [Mili 13b].

In a training step, we first use a Probabilistic Boosting Tree to build a combined model of changes in tumor volume over time and corresponding changes in image intensities from a set of patient images. In the application phase this model is then used to guide the lesion segmentation in follow-up images of new patients by calculating a patient specific prior lesion probability.

Two different methods for building the model and integrating it into the segmentation system introduced in Chap. 3 are described and compared. One trains a PBT only for calculating the prior. This prior is then combined with the classifier output of the existing lesion segmentation system. The other method fuses the prior model and the original segmentation system into a new lesion detector that is specialized on follow-up images by incorporating all information into a single classifier.

In both cases, the input for the system consists of sets of three images each: The CT image acquired at the initial examination, called baseline image in the remainder of this chapter; a lesion segmentation of this image; and a CT image acquired during a follow-up exam of the same patient, which is thus called follow-up image. For training of the classifier that forms the core of the system, in addition, a lesion segmentation mask of the follow-up image is provided as reference. Both baseline image and baseline segmentation mask are registered to the follow-up image in the process to provide point correspondences.

In the remainder of this chapter, we will first review relevant algorithms from fields related to the segmentation of liver lesions in follow-up images. Next, a basic prior model is described in Sec. 4.3, before in Sec. 4.4 the proposed lesion prior model is explained in detail. The used methods are evaluated in Sec. 4.5.

4.1 Related Work

While in Sec. 3.1 general methods for automatic lesion segmentation were outlined, here we shift the focus towards algorithms addressing issues more specific to the follow-up

setting, first listing some publications describing methods for detecting changes between images, and then methods dealing with the problem of deformation in temporal image series. Finally, some existing work from the field of priors for detection and segmentation is detailed on.

4.1.1 Image Change Detection

The methods in Sec. 3.1 segment lesions in an image using only information from this image. We on the other hand are dealing with follow-up images, meaning that previous acquisitions of the same patient are available. These provide additional information, which can be used to improve segmentation performance in basically two ways: Information gained from the baseline image and its segmentation can be projected to the follow-up image, or changes between the images can be measured. In literature, there is a wide range of methods for detecting changes between images [Radk 05, Paja 06, Paja 09]. The fact that changes in medical images indicate growth or other pathologic processes has been used in automatic detection or segmentation systems before.

Saha et al. [Saha 11] take advantage of the symmetry properties of the brain to identify brain tumors and edemas by comparing local intensity statistics of the hemispheres. Similar to the scenario presented here, Rey et al. [Rey 02] work with temporal image series of the brain to detect and segment multiple sclerosis lesions. In contrast to the task at hand, however, they have no baseline segmentation available and thus only build upon tissue deformations and intensity changes that occur between acquisitions.

4.1.2 Follow-up Image Registration and Segmentation

In order to segment lesions in follow-up images, one would ideally map the baseline lesion mask to the follow-up image and apply a model of tumor growth, in order to predict the lesion boundaries in the follow-up image and thus provide a segmentation. For brain MR images, Gooya et al. [Gooy 11] achieve this by using a biophysical model of lesion growth and tissue deformation. Starting from a lesion seed point in an atlas, they grow their virtual tumor based on the model and register it to the patient image using an expectation-maximization framework. However, they focus their work on a single type of brain tumor (glioma). In the liver, on the other hand, about 30 types of clinically relevant tumors can occur, each of which would require a different model. This is further complicated by the fact that their actual identification is generally considered impossible without biopsy.

A prerequisite for fusing information from different images is a coordinate mapping between them, which usually implies their registration. The main challenges in intra-patient registration of follow-up liver images lie in the nature of liver tissue, i.e. its low contrast in CT images and its elasticity. It is very soft, allowing large deformations even within a single breathing cycle. Tumor growth or other structural changes inside the liver of a patient may change its shape and appearance even further.

Charnoz et al. [Char 05] try to overcome these issues by using anatomical landmarks for their registration algorithm. They calculate the deformation field by matching hepatic vessel trees of baseline and follow-up images and interpolating the deformation for the remaining image points. The baseline lesion mask is mapped to the follow-up lesions, allowing their comparison without segmenting follow-up lesions. The interpolation scheme,

however, is limited to space-occupying tumor growth since infiltrating growth of a tumor is not necessarily reflected in a deformation of the surrounding vessels. Also, while the vessel tree provides very robust landmarks for registration, its segmentation is a challenging problem by itself.

Moltz et al. [Molt 09] propose a lesion detection framework for follow-up images that requires only the segmentation of the lesions in the baseline scan. They apply their algorithm to the detection of liver metastases, of lung nodules and of lymph nodes. After a rigid registration step, baseline lesion masks are mapped to the follow-up image. Around each mask, lesion candidates are generated by gray value thresholding and detecting circular structures. The correct candidate is then selected by a template matching algorithm based on normalized cross-correlation and forms the input to a segmentation algorithm. Large deformations of the liver or the target lesions, or changes in the lesion texture as they may be induced by chemotherapy, naturally pose a problem for this method.

4.1.3 Prior Models for Detection and Segmentation

Feulner et al. [Feul 10] show that detection or segmentation methods based on classification can be further stabilized by combining the output of the classifier with a spatial prior calculated from a statistical atlas. Assuming, the location $\mathbf{l} \in \mathbb{R}^3$ of a point in the image and its feature vector $\mathbf{x} \in \mathbb{R}^m$ (with the number of features m depending on the actual segmentation task) are conditionally independent, this corresponds to calculating the posterior object probability as [Bish 07]

$$\begin{aligned} p(y|\mathbf{x},\mathbf{l}) &\propto p(\mathbf{x},\mathbf{l}|y)p(y) = p(\mathbf{x}|y)p(\mathbf{l}|y)p(y) \\ &= \frac{p(y|\mathbf{x})p(y|\mathbf{l})}{p(y)}, \end{aligned} \tag{4.1}$$

where $y \in \{-1,1\}$ denotes the point's class label, $p(y|\mathbf{x})$ the posterior generated by the classifier and $p(y|\mathbf{l})$ the spatial prior. Although the independence assumption is not true for a given image, this simplification seems not to affect the system's performance negatively while at the same time reducing the complexity of the learning problem for the classifier.

Unfortunately, since the location of lesions inside the liver is fairly random, an atlas-based spatial prior as proposed for lymph node detection in [Feul 10] is not suitable for the task of liver lesion segmentation. A prior generated from such a general atlas of lesion locations would be almost uniform and certainly not discriminative enough to restrict a segmentation algorithm.

In a similar fashion, Feulner et al. [Feul 11b] use a prior built from a smoothed segmentation of esophageal air to detect the esophagus position within a CT image slice. In their case, however, the air segmentation is based on the same image the esophagus is to be detected in. In the tumor follow-up setting at hand, a lesion segmentation is mapped from the baseline to a follow-up image. Thus, there are growth and shrinkage processes involved, which are not known to the algorithm and cause additional variability and uncertainty. These make two separately filtered versions of the mapped lesion mask necessary, one for the growth and one for the shrinkage cases. Also, since the volume change is not known beforehand, manual interaction becomes necessary to set the filter parameters.

4.2 Processing Pipeline for Liver Lesion Segmentation

This section describes the framework in which we embedded the proposed method for follow-up lesion segmentation. It is based on the system presented in Chap. 3, but adapted to the special follow-up setting. The subsequent sections explain how the lesion detector was extended for this scenario. This includes a description of the image data occurring in the follow-up scenario as well as information about the registration component that is incorporated to allow handling of image series. Details on how each image is used are provided in the respective algorithmic sections, because the different methods handle the images in slightly different ways.

4.2.1 Input Images

What makes the follow-up setting special is the fact that there is not a single input image but a series of images of the same patient. The initial CT scan is called baseline image. All later ones, acquired in order to monitor the treatment response or progression of the same disease, are called follow-up images.

The goal in the scenario at hand is to segment liver lesions in a follow-up image of a patient. The baseline image is available, as well as a lesion segmentation for this baseline image. In practice, the baseline segmentation may be the result of any general lesion segmentation method, either manual or (semi-)automatic. For our experiments, however, these lesion masks were the result of a manual segmentation in order to purely evaluate the follow-up segmentation method. The segmentations were partly performed by experienced radiologists and partly by the authors and reviewed by radiologists. In the same way, reference lesion segmentations of the follow-up images were generated for training and evaluating the system.

While in general any number of images may be acquired for a patient, the presented system currently works with pairs of one baseline and one follow-up image. Thus, for those patients in the test database that had more than two images acquired, the images were split into several pairs, so that the input into the system always consisted of a set of three images.

4.2.2 Follow-up Liver Lesion Segmentation

Figure 4.1 shows how the input images are processed in the follow-up setting. The depicted segmentation pipeline is an extension of the one shown in Fig. 3.1. The first processing step in this case consists of non-rigidly registering the baseline image to the follow-up image and transforming it accordingly. The same transformation is applied to the baseline segmentation, so that for any further processing, all images lie in the coordinate frame of the follow-up image, providing point correspondences.

The follow-up image is the target of the entire processing, so the result of the liver segmentation step applied to this image defines the region of interest for the detection step. Both intensity images, however, are standardized as in the original method (cf. Sec. 3.2.2) before points in the follow-up image are classified.

This classification is where the proposed prior comes into play. Originally, classifying an image position $\mathbf{l}$ in a follow-up image would involve assigning a lesion probability

based on the appearance model $p(y|\mathbf{x}_a)$, with features in $\mathbf{x}_a$ calculated from the follow-up image. To train this classifier, baseline and follow-up images could both be used to form the training set. Since this classifier uses information from only one image for its decision, the time point of acquisition would be irrelevant. The input for the PBT training phase would consist of a set of n training samples $\mathbf{V} = \{(\mathbf{x}_{a,1}, y_1), .., (\mathbf{x}_{a,n}, y_n)\}$, calculated at randomly drawn locations in the selected training images, where $y_i \in \{-1, 1\}$ denotes the target class determined from the corresponding lesion mask.

In addition, in the system described here, baseline and follow-up images are aligned by the registration step. Thus, whenever a voxel in the follow-up image is to be classified, the corresponding voxel position in the baseline image and the baseline lesion mask is known. This allows the calculation of features encoding information not only from the follow-up image, but also from the other two images. This information is used to form the prior. Depending on which of the methods described in Sec. 4.4 is considered, the prior either replaces or complements the appearance model in this setup.

4.2.3 Follow-up Liver Registration

While registration is not the focus of our work, it is a key component of the presented algorithm, since a good mapping from baseline to follow-up image is required for fusing their information.

The algorithm we chose to use for this task was originally designed for non-rigid multi-modal image registration [Chef 02]. It is a variational approach, optimizing a similarity criterion based on local cross-correlation. A template propagation method is introduced to be able to deal with large deformations as they may occur in multi-modal scenarios. These deformations are recovered by combining small displacements along the gradient of the similarity measure. For regularization purposes, Gaussian filtering is applied to these small steps that are calculated in each iteration.

For the setting at hand this algorithm yielded satisfying results in matching the liver surface and vessel structures. Tumor growth, however, results in a strong local deformation that can be very irregular and that will rarely be consistent with the overall liver deformation caused by motion. After registering the images, baseline lesions will therefore be correctly localized in the follow-up image, but because of the rather strong regularization, their boundaries will not necessarily match those of the follow-up lesions (Figure 4.2). This is the desired behavior, since the volume change is the target of our learning algorithm.

For the remainder of this chapter, all mentioned coordinates refer to the coordinate system of the follow-up image, i.e., it is assumed that the baseline image has been registered to the follow-up image and transformed accordingly, together with its lesion mask.

4.3 Spatial Prior Models for Follow-up Segmentation

Given a mapping of the baseline segmentation to the follow-up image, patient specific information about previous lesion locations provides a strong cue on where to expect lesions in the follow-up image and forms the basis for the prior models we use for guiding our lesion detector.

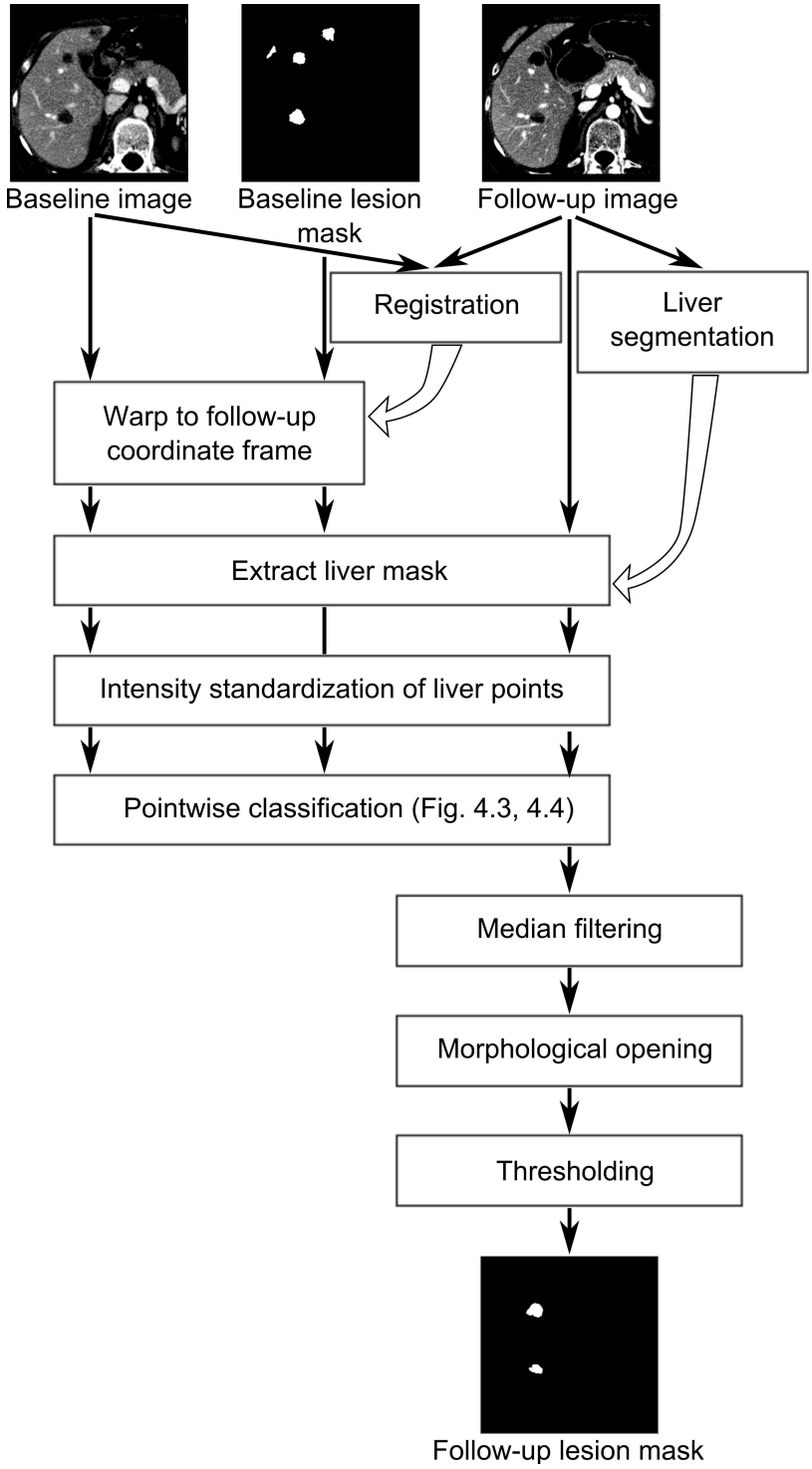
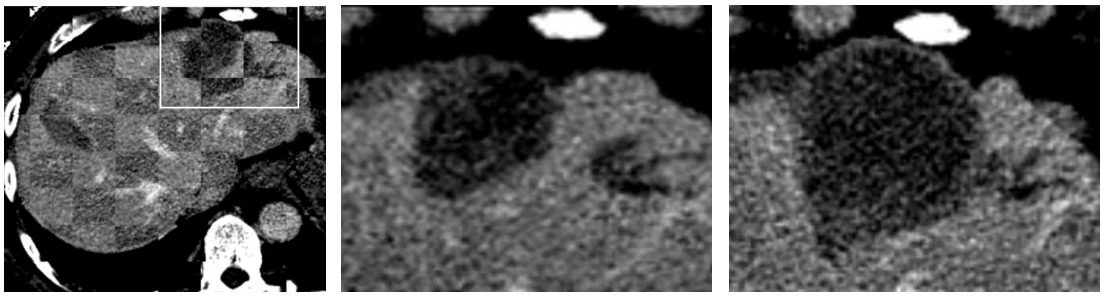


Figure 4.1: Processing pipeline for segmenting follow-up lesions. The baseline image is non-rigidly registered to the follow-up image and transformed accordingly, together with its lesion mask. After segmenting the liver, the intensity images are standardized and the points in the follow-up image are classified as tumor or background, using features calculated from all three images (depending on the type of prior). The classification step differs depending on how the prior is incorporated into the segmentation procedure (compare Fig. 4.3, 4.4). Its output is a probability image, which is postprocessed and transformed into a lesion mask by a thresholding operation.



(a) (b) (c)

Figure 4.2: Checkerboard representation of a registered pair of baseline and follow-up images (a) with a tumor that has grown between acquisitions. For better visibility of the checkerboard pattern, the images are shown using different windowing functions. The cutouts (white box) show a baseline (b) and follow-up (c) lesion after registration.

The simplest prior in this scenario, given only for reference, would be the registered baseline lesion segmentation at location $\mathbf{l}$ itself:

$$p(y|\mathbf{l}) = M_0(\mathbf{l}) = \begin{cases} 1, & \mathbf{l} \text{ inside baseline lesion} \\ 0, & \mathbf{l} \text{ outside baseline lesions} \end{cases} \tag{4.2}$$

M_0 here denotes the baseline lesion mask image. Let $p(y|\mathbf{x}_a)$ be the posterior calculated from the follow-up image by a previously trained appearance model as described in Chap. 3 and Sec. 4.2.2, which forms the pointwise classification step in Fig. 4.1. Following the argument in [Feul 10, Bish 07], this probability can then during application stage be combined with the prior (compare Eq. (4.1)):

$$p(y|\mathbf{x}_a, \mathbf{l}) \propto p(y|\mathbf{x}_a) p(y|\mathbf{l}) = p^*(y|\mathbf{x}_a, \mathbf{l}) \tag{4.3}$$

The prior probability $p(y)$ is omitted here as a further simplification, since it is constant for a given scenario and the final goal of the method is not the posterior probability itself but rather a binary decision. The resulting probability map $p^*(y|\mathbf{x}_a, \mathbf{l})$ is postprocessed and converted into a lesion mask as before, where the scaling with $p(y)$ is accounted for in the final thresholding operation.

The underlying assumption that the target lesions did not change between acquisitions is obviously too restrictive, but this naïve version does provide a reference result as well as some insight into how the prior should be constructed. Since this prior is binary, multiplying it with the output of the lesion detector completely suppresses all regions not covered by baseline lesions, so that no lesion growth nor new tumors can be detected. Within the boundaries of the baseline lesions, the detector response remains unchanged, without suppressing shrunk lesions.

A seemingly straightforward extension that is similar to the prior used by Feulner et al. for esophagus detection [Feul 11b] involves filtering the mask image M_0 to soften the prior: Assuming that the volume change of tumors follows a zero-mean normal distribution, the mask image is filtered using Gaussian smoothing. The prior is then a combination of a

growth component, filtered with standard deviation σ_g, and a shrinkage component, filtered with standard deviation σ_s:

$$p(y|\mathbf{l}) = \begin{cases} 1 - \lambda_s \cdot e^{-\frac{d(\mathbf{l})^2}{2{\sigma_s}^2}} & \text{if } M_0(\mathbf{l}) = 1 \\ \lambda_g \cdot e^{-\frac{d(\mathbf{l})^2}{2{\sigma_g}^2}} & \text{else} \end{cases} \tag{4.4}$$

with $d(\mathbf{l})$ the distance of the location $\mathbf{l}$ to the closest lesion boundary. This prior is multiplicatively combined with the appearance model just like the mask prior described above.

The weighting factors $\lambda_g, \lambda_s \in [0,1]$ control the influence of the growth component vs. the shrinkage component. Both the optimal values for these weighting factors and the optimal width of the Gaussian filters vary between images, even between lesions within the same image. For best segmentation results, the parameters would thus have to be defined interactively for each lesion by the user. Even though this is not the fully automatic solution that is sought for here, quantitative results for this prior are presented in Sec. 4.5 for comparison. These were generated, however, using a fixed set of parameters for all images to allow a direct comparison with the other methods.

4.4 Learning a Prior Model for Follow-up Segmentation

To overcome the limitations of above's mask based priors such as inflexibility and need for user interaction, a novel learning based approach was developed. As mentioned before, a "generative" growth model that predicts lesion boundaries based on a biophysical model as Gooya et al. developed for gliomas [Gooy 11] is not feasible for liver tumors. Therefore, we decided to use a data driven approach instead.

As for the original lesion detector, for the prior a discriminative model for lesion probability at an image position $\mathbf{l}$ is learned by the PBT algorithm. But in contrast to the previous setting, now all three input images are taken into account. In Sec. 4.4.1 a prior is learned from these images and combined with the result of the appearance based classification $p(y|\mathbf{x}_a)$ (Figure 4.3). Section 4.4.2 goes all the way and combines the information from the prior and the appearance model into one single classifier (Figure 4.4).

Feature category	Feature name	Scales/Neighborhoods	Total #
Spatial features	Time between scans		1
	Mask value		1
	Distance to closest lesion		1
	Gaussian of distance	0.1 – 10.0	13
	Gaussian of distance, time scaled	0.1 – 10.0	13
			29

Table 4.1: Spatial features in $\mathbf{x}_s$ calculated from baseline lesion mask to learn the prior.

Feature category	Feature name	Scales/Neighborhoods	Total #
Intensity statistics	Intensity difference	0.0 mm – 32.0 mm	10
	Weighted intensity sums	0.0 mm – 32.0 mm	30
	Difference between local variances	0.8 mm – 32.0 mm	9
Vesselness	Sato vesselness		1
			50

Table 4.2: Intensity change features in $\mathbf{x}_s$ used to learn the prior.

4.4.1 Multiplicative Learned Prior

Rather than explicitly modeling tumor growth [Gooy 11], the learned prior model should implicitly encode volume changes in the tumors. To achieve this, a feature vector $\mathbf{x}_s \in \mathbb{R}^l$ is used for classification that is a combination of two sets of features.

One set consists mainly of spatial features (Table 4.1) calculated from the baseline mask and encodes the actual tumor growth and shrinking. Features include various filtered versions of the signed distance from the current image location $\mathbf{l}$ to the closest lesion boundary, as well as the time between acquisitions. These features basically contain the same information as the filtered mask prior in Sec. 4.3 and are thus not much more flexible or patient specific. In order to facilitate patient specific adaptations of this model, these purely geometric features are complemented by a second set of features encoding changes between the baseline and the follow-up intensity images (Table 4.2). Instead of complex change detection methods, simple difference features are calculated from the registered baseline and follow-up images, leaving their interpretation to the learning algorithm of the PBT. This feature set contains weighted intensity differences and sums over various neighborhoods of the current location, as well as a vesselness measure.

With a number of patient datasets for training, the PBT modeling $p(y|\mathbf{x}_s)$ is built. Using this prior to guide the segmentation in the follow-up image yields (compare Eq. (4.1))

$$p(y|\mathbf{x}_a,\mathbf{x}_s) \propto p(y|\mathbf{x}_a)p(y|\mathbf{x}_s) = p^*(y|\mathbf{x}_a,\mathbf{x}_s). \tag{4.5}$$

Thus, for this version of the prior, the classification step in Fig. 4.1 becomes the product of the output of the original lesion detector and the prior $p(y|\mathbf{x}_s)$ (cf. Fig. 4.3).

Due to its hierarchical nature, the PBT can be seen as learning not a single but a whole family of models, very similar to the hierarchical mixture of experts scheme [Hayk 99]. When the trained PBT is then used to classify a new instance, the intensity change features guide it in its decision which model should be applied, and adapt the geometric model to characteristic deviations from the normal lesion volume change found in the current patient. Moreover, they also support the detection of newly emerged lesions. Large changes in tumor structure or texture induced by chemotherapy may also be reflected in these features. However, since on a larger scale these changes only lead to a slightly different intensity, they do not affect the prior learning negatively.

Overall, in our experiments the vector $\mathbf{x}_s$ held $l = 79$ features. The PBT for the prior is then trained with a set of n training samples $\mathbf{V}_{lp} = \{(\mathbf{x}_{s,1}, y_1), .., (\mathbf{x}_{s,n}, y_n)\}$ calculated at randomly drawn locations in the training follow-up images, where the target class labels y_i are determined from the follow-up lesion mask.

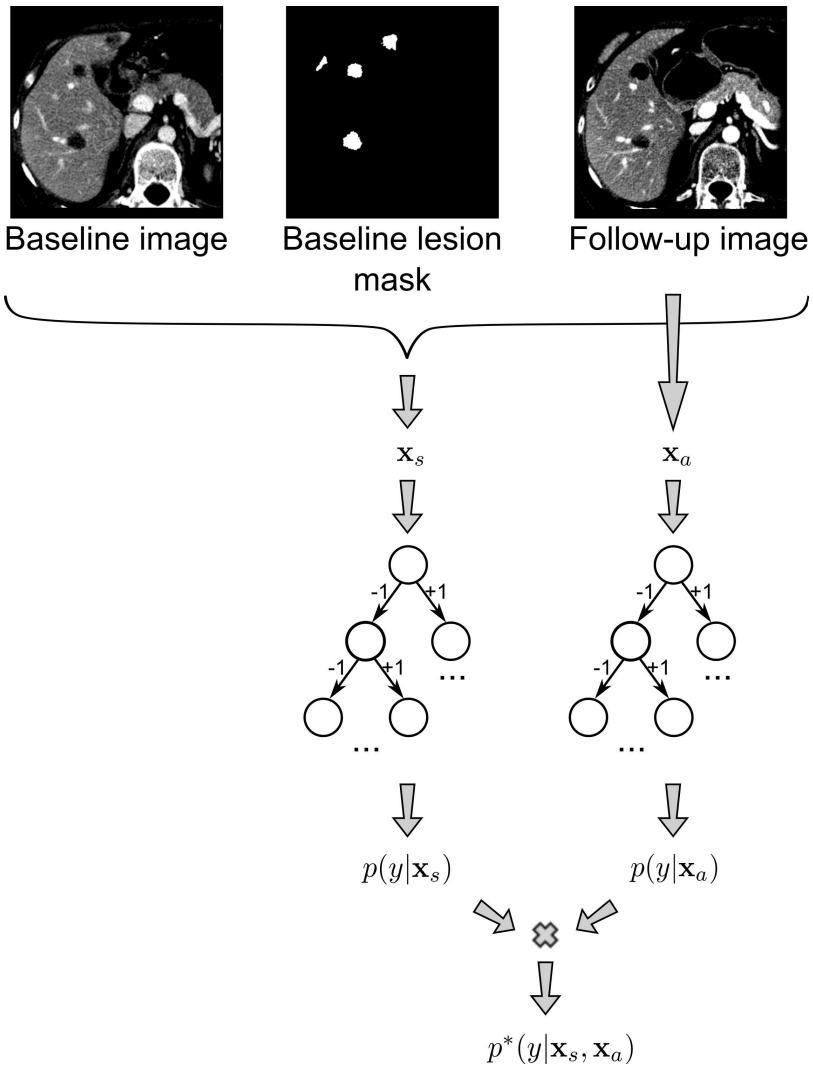


Figure 4.3: Setup for classification step using a learned multiplicative lesion prior. In addition to the original lesion detector that works with appearance features $\mathbf{x}_a$ calculated from the follow-up image, a second PBT is trained. It receives a vector of prior features $\mathbf{x}_s$ calculated from baseline image, baseline lesion mask and follow-up image and computes the lesion prior. Both classifiers' outputs are multiplied to form the final lesion probability $p^*(y|\mathbf{x}_s, \mathbf{x}_a)$.

The formulation in Eq. 4.5 again considers the feature vectors $\mathbf{x}_s$ and $\mathbf{x}_a$ being conditionally independent. While it is almost certainly not true, this model assumption does help dealing with the complex, high-dimensional feature space at hand. It allows splitting the space and training two separate models on the two subspaces. This has the advantage of making the classification task more manageable. Furthermore, it is also convenient and intuitive in that the same appearance based classifier from the original segmentation system is used to segment all images, only refined in the follow-up case by the prior generated by the second classifier.

4.4.2 Integrated Learned Prior

One of the major benefits of the PBT is its hierarchical decision making. This characteristic allows the PBT to learn complex feature spaces and should thus enable it to correlate image appearance, changes in intensities, and tumor presence effectively. In fact, as could be shown in a different context, training a PBT on a more complex feature space might even be superior to manually splitting the problem and training several PBTs on smaller subproblems [Mili 09]. So, to see whether the PBT can exploit existing correlations and dependencies in the data to improve classification accuracy, another PBT was trained to calculate $p(y|\mathbf{x}_a,\mathbf{x}_s)$ as $p(y|\mathbf{x}_c)$, $\mathbf{x}_c = (\mathbf{x}_a,\mathbf{x}_s) \in \mathbb{R}^{m+l}$. The feature vectors were simply concatenated, so that all the input images and features were used in a single classifier (Figure 4.4), dropping the independence assumption introduced before. The training data then consists of samples $\mathbf{V}_{ip} = \{(\mathbf{x}_{c,1},y_1),..,(\mathbf{x}_{c,n},y_n)\}$ again calculated at randomly chosen locations in the follow-up images selected for training, where the target class labels are determined from the follow-up lesion masks as before. This classifier can then be used for lesion segmentation in follow-up images directly, replacing the combination of the original lesion detector and the prior described in Sec. 4.4.1.

While it seems desirable to leave the decision on the optimal combination of features to the training procedure, the higher dimensional and substantially more complex feature space makes greater demands on both the training data and the learning algorithm. This effect could be observed in some of the experiments conducted for this work, so the topic is addressed further in Sec. 4.5.5. Also, from an application point of view, this scenario has the potential drawback that different features may be chosen by the learning algorithm for segmentation in baseline and follow-up images, leading to different behavior of the two systems and different types of errors. Still, provided that sufficient training data is available, this is the method of choice, because it combines the advantages of the multiplicative learned prior with the detection capabilities of the original segmentation system.

4.5 Results and Discussion

In order to evaluate and compare the priors described above, a set of experiments was performed on clinical images. First, the accuracy of the non-rigid registration was measured. Next, various aspects of the methods' segmentation performance were evaluated. The system's lesion detection capabilities were investigated separately, before finally the features selected by the learning algorithm were analyzed.

The segmentation methods compared on the following pages are:

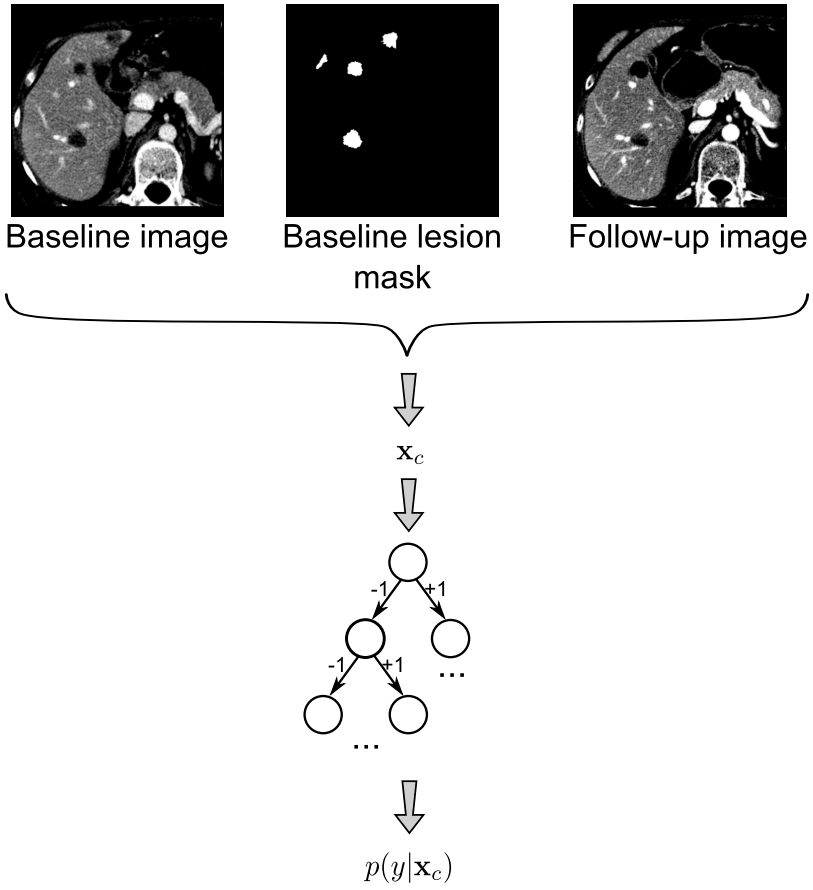


Figure 4.4: Classifier setup used for integrating the prior information into the lesion detector directly: The feature sets of the lesion detector ($\mathbf{x}_a$) and the prior ($\mathbf{x}_s$) are combined into one feature vector $\mathbf{x}_c$. A single PBT is then used to assign the posterior lesion probability $p(y|\mathbf{x}_c)$.

- "No Prior" - The lesion detector of the original segmentation system as described in Chap. 3, using only follow-up images as input.
- "Mask Prior" - Uses the registered baseline lesion mask as prior for follow-up segmentation by multiplying it with the lesion detector's posterior probability output $p(y|\mathbf{x}_a)$ (cf. Sec. 4.3).
- "Filtered Mask Prior" - Applies two Gaussian filters to the registered baseline lesion mask before multiplying it with the lesion detector's posterior probability output $p(y|\mathbf{x}_a)$ (cf. Sec. 4.3).
- "Learned Prior" - Trains a PBT to generate the prior $p(y|\mathbf{x}_s)$ and multiplies this prior with the lesion detector's posterior probability output $p(y|\mathbf{x}_a)$ (cf. Sec. 4.4.1).
- "Integrated Prior" - Trains a PBT that combines the features $\mathbf{x}_s$ of the "Learned Prior" with the appearance features of the lesion detector $\mathbf{x}_a$, directly calculating $p(y|\mathbf{x}_c)$ (cf. Sec. 4.4.2).

The feature vectors used by the different classifiers are again summarized in Tab. 4.3.

Feature vector	Description	Calculated from
$\mathbf{x}_a$	Appearance of voxel, used for general lesion segmentation	Single (follow-up) image
$\mathbf{x}_s$	Encodes lesion growth, used for calculating lesion prior	Baseline lesion mask (geometric features), baseline and follow-up images (change features)
$\mathbf{x}_c$	Combination of above feature vectors (concatenation), used for follow-up lesion segmentation	Single (follow-up) image (appearance features), baseline lesion mask (geometric features), baseline and follow-up images (change features)

Table 4.3: Feature vectors used in different discriminative models.

4.5.1 Image Database

To the best of our knowledge there is no publicly available database containing follow-up series of liver CT images. Therefore, we used our own test database, which comprised liver CT images of 14 patients, acquired at 4 different clinical sites. The images had a voxel resolution between 0.531 and 0.830 mm in x and y directions, and 2 to 4 mm in z direction. All had been acquired at 120 kVp. For each patient at least one baseline and one follow-up image were available, with a free interval between the scans of 1 – 13 months. For some patients several follow-up images had been taken at different times, so that a total of 17 image sets of baseline image, baseline lesion mask and follow-up image could be built. All images were acquired after injection of a contrast agent and showed a venous enhancement of the liver. The baseline images contained 80 mostly hypodense lesions. In 6 of the 17 image sets a growth of the contained lesions was observed, with a median growth of the

total lesion volume per image of 41%, a minimum of 3% and a maximum of 564%. In the remaining cases, the lesions diminished, with a median total volume loss of 35%, a minimum of 1% and a maximum of 95%. 13 new lesions emerged during the studies. This broad range of variations in the images poses a great challenge for the algorithm, making this a very realistic test scenario.

4.5.2 Experimental Setup

As before, each single experiment was set up as a 5-fold cross-validation. During each fold, 13–14 image sets were used to train the classifiers and the remaining ones were held out as independent test set, making sure no patient occurred in both training and test. This was also ensured for the experiments with the multiplicative learned prior, where for each fold two classifiers had to be trained and tested. That way, training and testing data were kept separated in all experiments.

The training images were again subsampled using the procedure described in Sec. 3.5.1; for testing all liver voxels were used.

4.5.3 Registration Evaluation

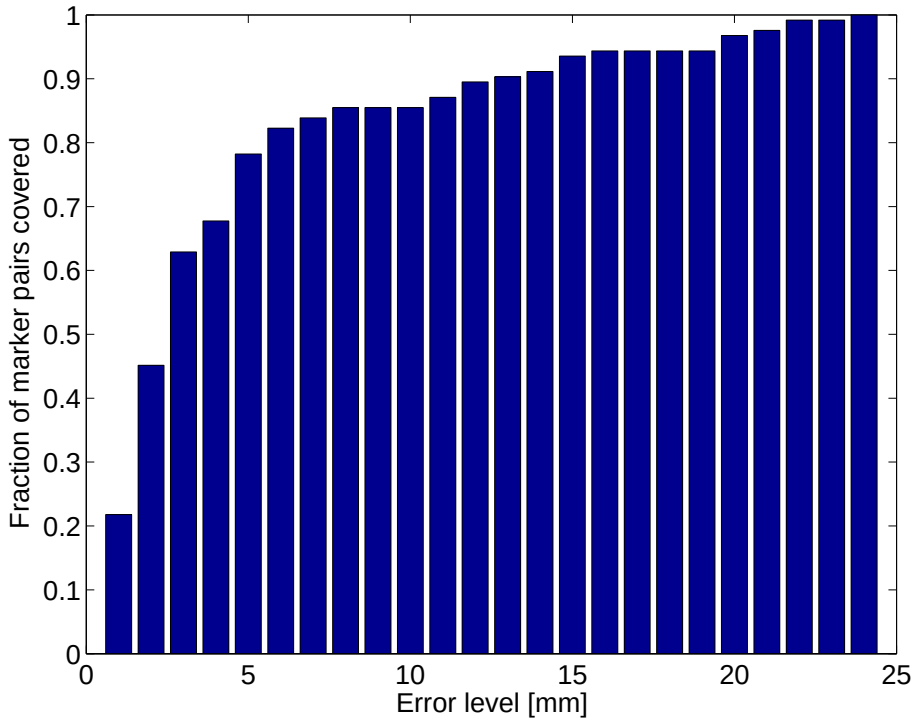


Figure 4.5: Cumulative histogram of registration errors on test database. For each error level on the *x*-axis, the bar shows how many of the 124 marker pairs fall within this error range.

The quality of the registration results was assessed with a marker based approach. For each image pair in the test database, 5-10 markers were placed manually at landmarks such as vessel bifurcations or inside tumors in both baseline and follow-up images (124 marker

pairs in total). After registering the images, the Euclidean distance between corresponding markers was measured. The average distance was 4.68 mm (minimum 0.58 mm, maximum 23.44 mm). Even though this evaluation was performed by a single observer and thus the placement error was not considered, it does give an impression of the algorithm's capabilities (compare Fig. 4.5).

The evaluation above shows, that the registration method is able to match landmarks in baseline and follow-up images. It does not evaluate, however, how the non-rigid warping influences the lesions contained in the liver. It turns out that, while it did not match lesion boundaries perfectly, the algorithm did slightly deform the lesions during registration. In most cases, the volume difference between baseline and follow-up lesions stayed in the same range as without the registration, changing e.g. from 20% to 33%. Only in one case, it changed from 33% growth to a 12% shrinkage. However, in practice the registered and thus deformed baseline lesion mask is only used to segment the undeformed follow-up lesions. After segmentation, the volumes of the unregistered and thus undeformed lesions are compared, so the differences introduced by the registration step do not affect the accuracy of the clinical growth assessment. The only constraint is that the same registration algorithm has to be used in all cases since the learning system used for segmentation adapts to its behavior.

4.5.4 Segmentation Evaluation

Segmentation performance of the system was assessed using the methods described in Sec. 3.5. A threshold was applied to the system's output probabilities to allow calculation of the volumetric overlap of the manual reference segmentation and each algorithm result. From this overlap, several performance figures were calculated and by varying the threshold, ROC curves were generated. As before, this analysis was limited to points inside the liver.

The first set of curves given in Fig. 4.6 reflects the system's segmentation performance without the filter based postprocessing, i. e. the classification accuracy. As a reference result, the dotted curve shows the performance of the original method as described in Chap. 3, without including any prior knowledge ("No Prior"). The "Mask Prior" curve was generated directly using the registered baseline lesion mask as prior (cf. Eq. (4.2)). The "Filtered Mask Prior" curve is the result of filtering this mask as described in Eq. (4.4). As expected, especially the unfiltered mask prior version features an excellent specificity, since its only false positive classifications stem from misregistered lesion boundaries. On the other hand, sensitivity drops considerably because lesion growth cannot be handled by this primitive prior. While the filtered mask prior generally supports lesion growth, its potential is limited by the fixed filter parameters.

If a PBT is trained to generate the prior using the features from tables 4.1 and 4.2, this limitation is overcome. The resulting prior effectively is like a filtered mask, where the filter parameters are adapted for each single voxel based on information from the current image, generating a patient specific prior. As the "Learned Prior" curve implies, this flexibility results in potentially higher false positive rates, but at the benefit of a better overall segmentation. Finally, providing all information to a single classifier and leaving the decision on how to combine it in the best possible way yields very similar results ("Integrated Prior" curve).

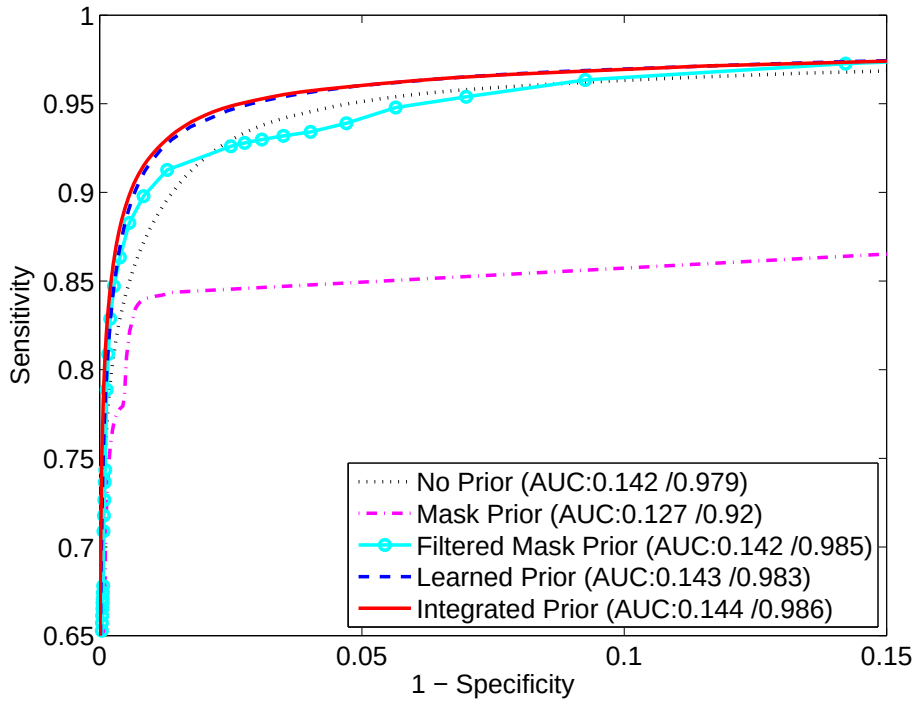


Figure 4.6: Segmentation performance without postprocessing. The reference curve ("No Prior") reflects the performance of the original lesion detector that uses no prior for follow-up segmentation. As expected, the versions using a mask based prior show excellent specificity at the cost of low sensitivity, missing any lesion growth as well as newly emerged lesions. Training a PBT to generate the prior can overcome these limitations and yields the best overall performance, independent of whether the prior model is trained separately ("Learned Prior") or integrated into the lesion detector ("Integrated Prior"). Area under the curve (AUC) values are provided for the partial ROC up to a false positive rate of 0.15 and for the entire ROC.

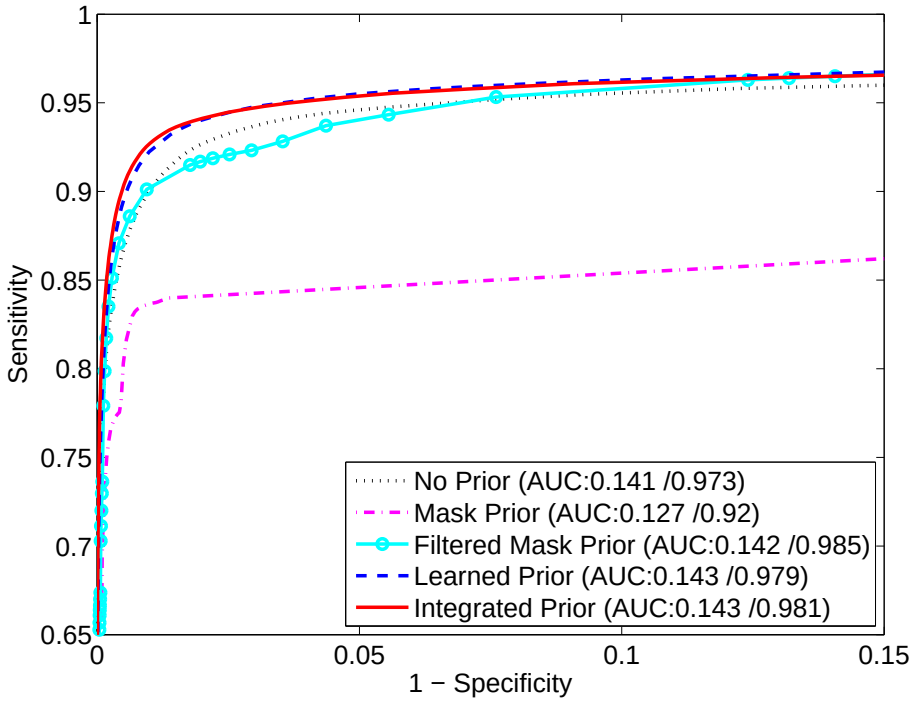


Figure 4.7: Segmentation performance with postprocessing. Compared to Fig. 4.6, the curves are all slightly shifted towards the upper left corner. Their relative positions, however, are nearly the same. AUC values are provided for the partial ROC up to a false positive rate of 0.15 and for the entire ROC.

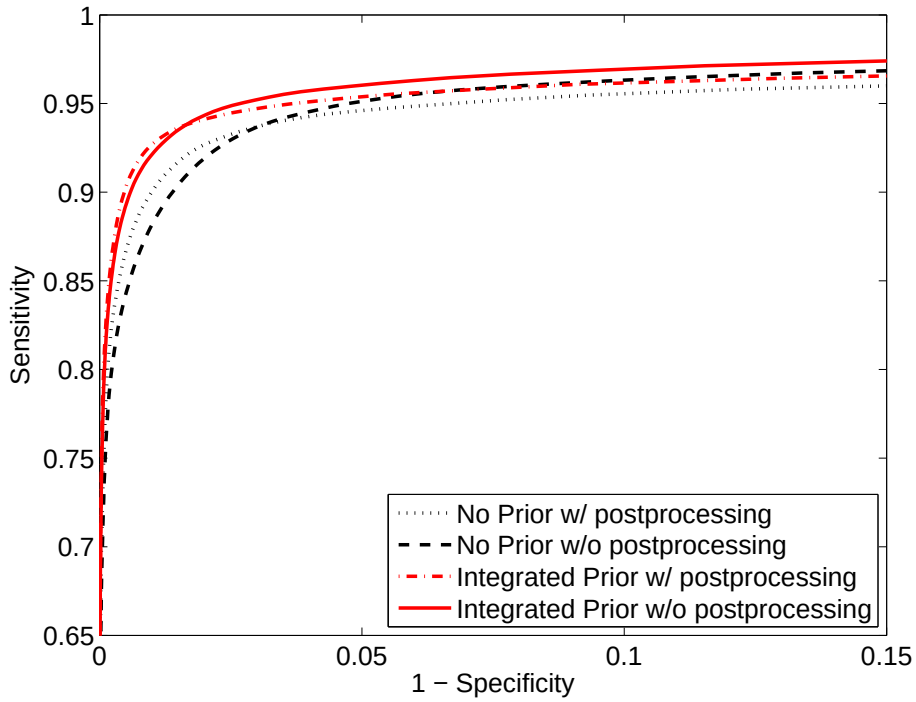


Figure 4.8: Comparison of segmentation performance with and without filter based post-processing for two exemplary configurations. The curves reveal a slight improvement in specificity at the cost of a slight decrease in sensitivity when postprocessing is applied. The latter explains, why for some configurations the AUC decreased between Fig. 4.6 and Fig. 4.7.

The curves in Fig. 4.7 reveal that the improvement due to introducing the prior seen in the experiments above is substantial: Here, the segmentation performance is measured after postprocessing. While all curves are slightly shifted to the left compared to the ones without postprocessing, their relative positions to each other remain nearly unchanged. Figure 4.8 demonstrates the influence of the postprocessing step even more clearly by comparing the unfiltered and the filtered results of the "No Prior" and the "Integrated Prior" curves directly. The effect shown in Fig. 4.6 is thus not leveled out by the postprocessing filters. Still, unless otherwise noted, the remainder of this section builds upon the raw classification results to eliminate all possible influence of the postprocessing.

Table 4.4 presents various performance figures for segmentation quality for the threshold value yielding 90% sensitivity. The change in specificity by adding prior information appears to be very small. However, since in most patients the largest part of the liver is made up by healthy tissue ($\approx 95\%$ on the test database), this may still have a large visual effect. The same observation holds for Jaccard and Dice coefficients. For these two, the maximum value achieved in the experiments is given in Tab. 4.5, together with the corresponding sensitivity value.

Method	No Prior	Learned Prior	Integrated Prior
Sensitivity (%)	90	90	90
Specificity (%)	98.6	99.3	99.4
Jaccard index	0.76	0.83	0.83
Dice coefficient	0.86	0.90	0.91
Precision (%)	82.7	90.8	91.9
#fp/#tp	0.21	0.10	0.09

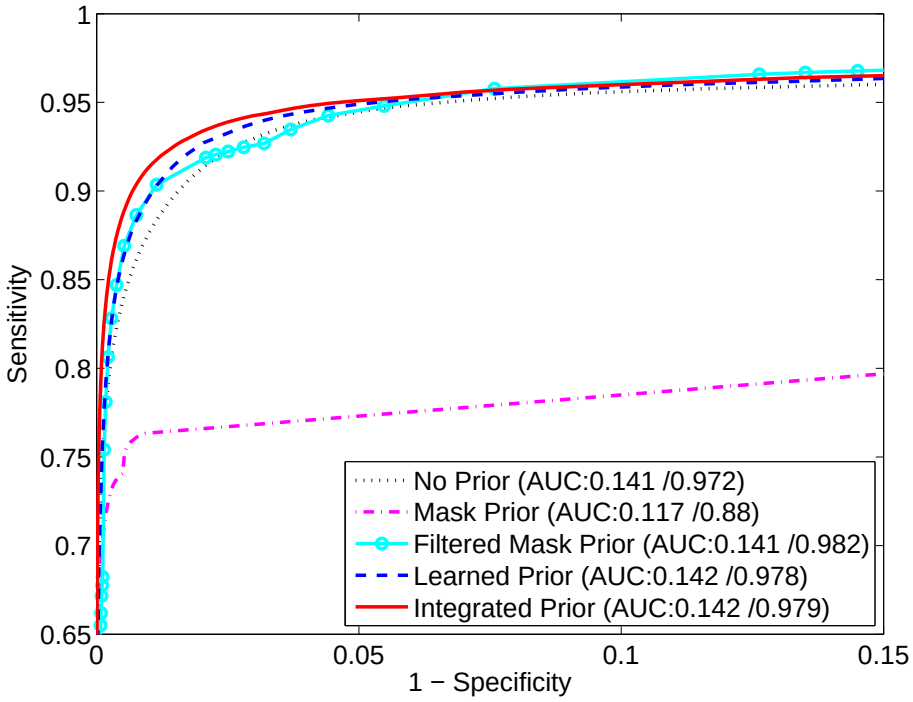
Table 4.4: Comparison of the segmentation quality of the learning based priors and the reference method at 90% sensitivity without postprocessing.

Method	No Prior	Learned Prior	Integrated Prior
Max Jaccard index	0.79	0.83	0.84
Max Dice coefficient	0.88	0.91	0.91
At sensitivity (%)	84.4	88.4	88.3

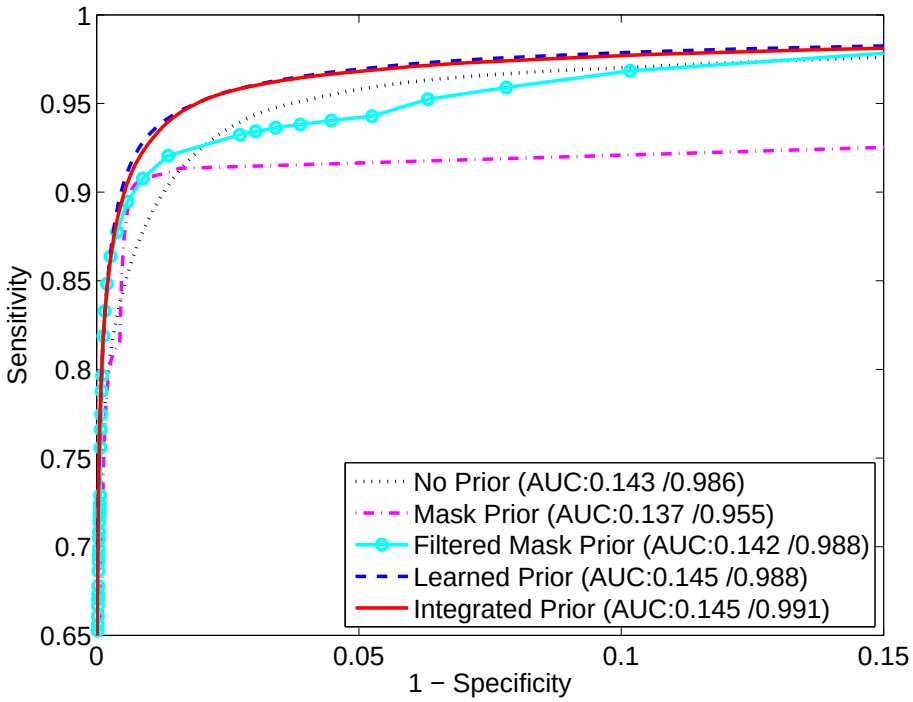
Table 4.5: Comparison of the optimal segmentation quality with respect to Jaccard and Dice similarity coefficients, together with corresponding sensitivity level.

The results for precision and the ratio $\frac{\#fp}{\#tp}$ show the full potential of the method in reducing the number of false positives: The positive predictive value is improved by 9.8% for the "Learned Prior" and 11.1% for the "Integrated Prior", the ratio $\frac{\#fp}{\#tp}$ is reduced by 51.8% ("Learned Prior") and 57.8% ("Integrated Prior") respectively, meaning a huge improvement for the physician.

Splitting up the testing data according to their treatment response reveals, that the shown improvement affects mainly those cases, where the target lesions responded to the chosen therapy (Figure 4.9). The fact that this is the case not only for the mask based but also for the learning based priors indicates, that the baseline lesion mask has a strong influence on these as well.



(a)



(b)

Figure 4.9: Segmentation performance without postprocessing on those datasets with lesion growth between acquisitions (a) and on those with shrinking lesions (b). AUC values in both cases are provided for the partial ROC up to a false positive rate of 0.15 and for the entire ROC.

4.5.5 Detection Evaluation

As for the original lesion segmentation system, we evaluated the detection performance of the different configurations described above separately. Figure 4.10 is a scatterplot of the detection results, which was generated just like the ROC curves above: Each point in the plot corresponds to one value of the threshold applied to the output probability of the system. The first aspect to notice in Fig. 4.10 is that, just as in the segmentation evaluation, including prior information considerably improves detection specificity, reducing the number of false alarms. A closer look at only the newly emerged lesions reveals, however, the limit of this claim. Both "Learned Prior" and "Integrated Prior" actually lead to a decrease in detection rate in these cases, yielding a maximum detection rate of 38% compared to 54% if no prior information is used.

The "Learned Prior" can hardly do better. It has only access to the prior features making up $\mathbf{x}_s$, i. e. the geometrical features in Tab. 4.1 and the intensity difference features in Tab. 4.2. Therefore, it acts very similar to the mask based priors and may suppress new detections in the follow-up image if the intensity change at this image location is not substantial.

The "Integrated Prior" on the other hand has access to the full feature set, i. e. the prior features $\mathbf{x}_s$ as well as the appearance features $\mathbf{x}_a$ the original system uses for detection and segmentation. In principle, it should thus be able to segment lesions that are present in both images as well as new ones. The limiting factor here is the training database: Only 13 lesions emerged between the acquisitions, meaning that samples from these lesions are highly underrepresented in the training data, which poses a challenge to the PBT with its greedy learning strategy. On the other hand, reducing the total number of cases in the database to achieve a more balanced dataset is not an option. The additional input images compared to the "No Prior" setup and the prior features calculated from them increase the complexity of the feature space considerably. Reducing the dataset would thus lead to instabilities in the training procedure and overfitting. Therefore, a larger training database with more new lesions is necessary to improve this somewhat disappointing result.

What may seem surprising is the performance of the "Mask Prior": Its maximum sensitivity is the same as that achieved by the other methods, except for the "Integrated Prior", at an even better false positive rate. This result is rooted in the registration algorithm. While this prior suppresses all tumor growth, all baseline lesions are mapped to their follow-up counterparts by the registration step. Thus, if the threshold is sufficiently low, these will be detected, even if not well segmented. At the same time, since it has a value of 0 outside the mapped baseline lesions (cf. Eq. (4.2)), this prior suppresses not only lesion growth but everything that is not within a baseline lesion, including any false positive the lesion detector might otherwise produce. The only way this system can make a false positive is in case registration fails or if a lesion disappeared, which is why the data points for this version are all located at very low false positive rates. The same effect, slightly weakened by the smoothing filters used during prior generation, can be observed for the "Filtered Mask Prior", which also exhibits an excellent specificity, with sensitivity quickly dropping off at about four false positives per volume.

The same drop-off can be observed for the "Learned" and "Integrated" priors. The reason for this behavior is not algorithmic but simply the detection criterion. As the threshold on the probability image is increased, the connected regions in the resulting lesion mask tend to fall apart, building lots of small connected components, each of which is considered

one lesion candidate. At the same time the number of correct detections decreases because of the failing overlap criterion. This effect can be seen for all methods, where a drop-off at lower false positive rates means a higher overall specificity. For the "No Prior" version the specificity is so low that the drop-off is outside the shown range.

The findings about the mask priors already indicate that the detection results given in Fig. 4.10 do not allow to draw conclusions about the quality of the segmentation generated with the same parameters, even though the detection criteria are based on an overlap measure. This is confirmed by looking at the setting yielding 90% sensitivity for segmentation again. The corresponding numbers for detection given in Tab. 4.6 show that there is a slight trade-off between optimal detection and segmentation.

Method	No Prior	Learned Prior	Integrated Prior
Segmentation sensitivity (%)	90.0	90.0	90.0
Detection sensitivity (%)	61.8	63.2	63.2
Detection precision (%)	37.5	61.4	66.2
Detection #fp/#tp	1.7	0.63	0.51

Table 4.6: Detection quality at the setting yielding 90% segmentation sensitivity.

4.5.6 Features

Finally, we examined the effectiveness of the features used for training the prior. Generally, two findings are worth noting. First, adding the spatial and intensity change features to the appearance features as in the "Integrated Prior" version does not change the relative distribution of the latter. Second, this version uses the intensity change features much more extensively than the "Learned Prior" version, putting (distributed over the entire PBT) 16 times as much weight on these as on the spatial features. For the "Learned Prior", this factor is only 1.2.

The spatial features used most extensively were the signed distance from the current location to the closest lesion, followed by the time scaled Gaussian of this distance. Of the intensity change features, the difference between baseline and follow-up image over some neighborhood was the most important feature, followed by the difference of local variances.

4.6 Conclusion

We presented a novel approach for integrating prior knowledge of expected lesion locations into a classification based algorithm for follow-up segmentation of liver lesions. A discriminative model is trained, which combines information about lesion appearance in CT images, lesion volume changes and intensity changes to form a decision. Two versions of this method were compared: One combines all information into a single PBT which is in turn used to segment follow-up lesions. The other method uses two PBTs, one for general liver lesion segmentation using the appearance features, the other for calculating a lesion prior for follow-up examinations. This prior is then multiplied with the output of the lesion detector.

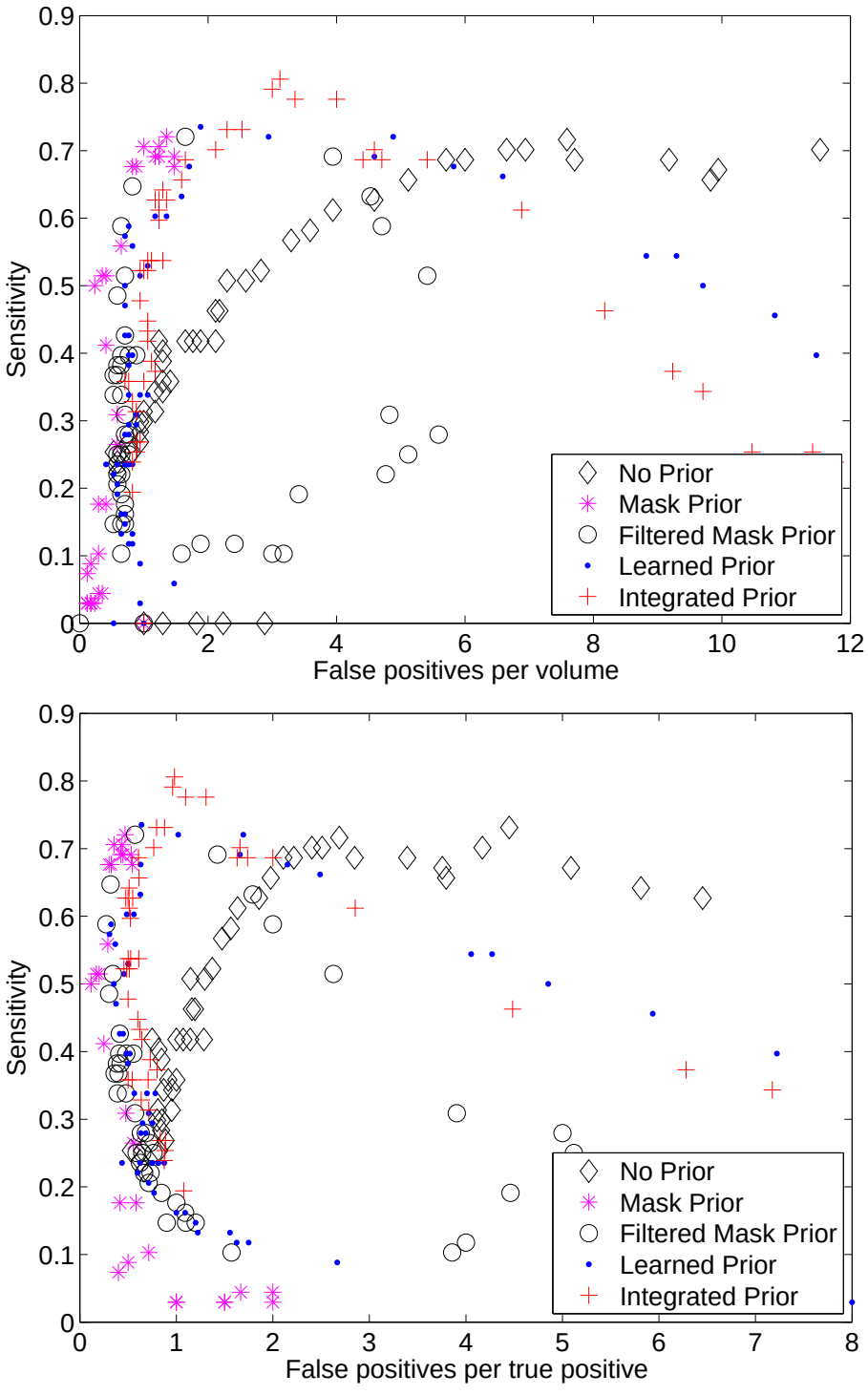


Figure 4.10: Detection performance with postprocessing.

For both methods, an improvement in classification precision compared to the original system with no prior knowledge could be shown in a set of experiments with clinical images. For the former version, the improvement was 11.1%, the latter achieved 9.8%. The two methods yield comparable results, where the improvement affects both segmentation and detection accuracy. Only for the detection of lesions that emerged between image acquisitions an improvement could not be shown, as these were underrepresented in the used database.

CHAPTER 5

A Cost Constrained Boosting Algorithm for Fast Object Detection

5.1 Related Work 73
5.2 Constrained Boosting 75
5.3 Results and Discussion 79
5.4 Conclusion 84

In the previous chapters, a system for automatic segmentation of liver lesions in CT images was introduced. It is based on categorizing image points into tumor and background by means of a Probabilistic Boosting Tree. The focus of the description so far was entirely on the accuracy of the system. While reliability is of course a prerequisite for any algorithm in this field, when it comes to clinical routine a second criterion gains importance: Response time. Since such learning based detectors have to classify large numbers of possible target object positions to filter out the true detections, this approach requires classification algorithms that are both accurate and efficient.

5.1 Related Work

In terms of accuracy, besides the boosting technique with its most popular descendant, AdaBoost [Freu 95], especially the support vector machine (SVM) [Vapn 95], has to be mentioned. Together with derived methods these are currently the most widely used techniques. However, for complex feature spaces like the one used for the task at hand, both the SVM and AdaBoost classifiers tend to become very large, slowing down classification. In domains with real-time requirements or with large amounts of data like CT images this is an issue. While recent speed increases in object detection largely stem from faster processing hardware or the incorporation of special domain knowledge, there have been a few influential algorithmic developments that made it to the standard repertoire in classification based detection. Regardless of whether sub-windows of the search space are classified to identify possible target positions or single points are categorized as belonging to either an object or the background, there are two starting points for possible speed-ups: Making the search space smaller to reduce the number of classifier evaluations, or making the classifier itself faster.

A widely used representative of the former category are coarse-to-fine strategies. The search space is first scanned on a very coarse grid and only those parts that induce a detector response are investigated with finer sampling as well. That way large portions of the search space can be omitted in the fine scans, greatly reducing the number of classifications.

Another very successful approach for speeding up detection systems is the implementation of rejection cascades. They are advantageous in detection settings, in which most of the input space consists of background and large parts of this background are well separable from the target objects. Probably the best known example of this architecture is the face detection cascade by Viola and Jones [Viol 01]. It meant a leap forward in object detection, as one of the first methods allowing real-time face detection, and by now is considered a standard approach. The cascade is built by training an AdaBoost classifier at each stage. The complexity of each classifier is controlled by stopping the training as soon as either the number of false rejections rises above a certain threshold or a desired rejection rate is reached. As the classification task becomes more difficult with each stage, more and more hypotheses are admitted to generate stronger AdaBoost classifiers. This mechanism, which Viola and Jones call "focus of attention", has the effect that classifiers at later stages are much slower than those at early stages. Or, put differently, the first stages can filter out most of the image background with very fast classifiers, leaving the few hard decisions to later, more complex ones.

The principle of the rejection cascade has been transferred to a hierarchy of SVMs by Heisele et al. [Heis 03] and picked up by Sahbi and Geman [Sahb 06]. Here, a cascade of linear SVMs is installed to filter out most of the background in a target image, before a final SVM with a second-degree polynomial kernel function separates the true detections from the most similar background regions. While lacking the automatic control over classifier complexity characterizing the AdaBoost cascade, this allows the combination of simple but fast linear SVMs with slow but accurate non-linear ones to achieve a similar behavior. An additional coarse-to-fine scheme yields a further speed-up. Finally, a feature reduction technique is applied for the non-linear SVM.

When optimizing the speed of classification systems, the core, meaning the classifier itself, is often left untouched. Still, a number of modifications to learning methods aiming at faster classification have been developed. For the SVM, several authors described ways of reducing the number of support vectors in the final decision function, thus potentially reducing the computational complexity.

These authors either introduce a pruning step to remove redundancies from the set of support vectors [Down 01, Nguy 06, Li 07], or modify SVM training in a way so that less support vectors are generated. Li et al. [Li 06b] achieve this by iteratively re-training the SVM on a dataset that is reduced in each step to more important points, generating an ever more compact solution. Parado-Hernández et al. [Parr 03] and Keerthi et al. [Keer 06] also devise iterative schemes, however starting with a simplistic SVM and expanding it in each iteration, thus directly controlling the size of the classifier. Osuna and Girosi take a more formal approach, reformulating the central optimization problem in a way that yields the same separating hyperplane in a possibly more compact representation [Osun 98].

For AdaBoost, there exist similar optimization approaches: Several pruning methods that try to reduce the number of hypotheses in the decision function after training have been presented and compared for effectiveness [Marg 97, Tamo 00]. These methods select the optimal set of hypotheses based on diversity in order to use the available features most

efficiently. While they were introduced to reduce memory consumption and overfitting effects, these techniques may also lead to a speed-up of the classifier.

Mahmoood and Khan [Mahm 09] and Utsumi et al. [Utsu 09] both use a trick to implement an exact on-the-fly pruning during application phase. They sort the hypotheses by their contributions to the final decision, in descending order. These contributions are measured by the hypothesis weights. When classifying a new sample, they can then limit the sequential evaluation of the hypotheses to those that are really necessary. Kawakita et al. [Kawa 11] extend this by learning an optimal order from the training data.

Taking on the gradient descent view of boosting, Bühlmann and Yu [Buhl 06] present a method for generating sparse classifiers with the L_2Boosting algorithm described by Friedman [Frie 01], which could be transferable to AdaBoost as well.

5.2 Constrained Boosting

In this chapter, a simple yet effective way of speeding up AdaBoost based classification is presented that is orthogonal to the ones mentioned above.

A first description of this approach was published in 2011 [Mili 11], a more elaborate version followed in 2013 [Mili 13a]. In the meantime, the same idea was independently picked up by Xu et al. [Xu 12, Xu 13], wrapped, however, into stagewise regression of limited depth regression trees [Xu 12] and a decision tree with strong learners at the nodes [Xu 13], respectively.

Like Margineantu and Dietterich [Marg 97], we aim at redefining what the “best” hypotheses for the final decision function should be. The approach, however, is a different one. (1) Our criterion for good hypotheses is not based on diversity but on computational complexity, leading not necessarily to less hypotheses, but to faster evaluation. (2) In contrast to the post-pruning approaches above, our solution is constructive. Instead of training the classifier with the original AdaBoost algorithm and adapting it afterwards, we modify the way the classifier is built in the first place.

In a nutshell, we yield a faster evaluation of the decision function by making the weak learner training prefer features that are fast to compute. The overall algorithm is defined so as to maintain classification accuracy. There are several ways how the desired effect can be achieved. A manual pre-selection of preferred features might be sufficient, although it requires a profound knowledge of the feature space and is a rather strict limitation of the learning algorithm. Restricting a learning algorithm artificially based on heuristics always bears the risk that it is prevented from learning certain patterns in the data. To keep the need for manual interaction at a minimum, we decided to instead equip the weak learning algorithm with some notion of feature cost, leading to a generic formulation of a cost constrained boosting algorithm. Two different options how the influence of the cost term can be controlled were investigated. They will be described in Sec. 5.2.2 and Sec. 5.2.3. A thorough analysis and discussion of their individual benefits is given in the evaluation section.

Though modifying the AdaBoost training procedure directly, this optimization aims at hierarchical systems based on AdaBoost. It was originally developed for use with the Probabilistic Boosting Tree that is the core of our segmentation system described in the previous chapters, but works just the same with the cascade [Viol 01] or similar architectures. Nevertheless, the PBT is used for the description on the following pages.

The modification can be seen as a realization of the focus of attention mechanism of the cascade for feature sets with inhomogeneous complexities. The objective is that early stages of the hierarchy contain particularly simple classifiers, which in our case is realized by constraining the feature selection performed during AdaBoost training. To prevent a loss of classification accuracy compared to the unconstrained approach, the condition is weakened for later stages of the hierarchy.

As will become obvious from the description, the method is not limited to runtime optimization. It is fully generic, allowing for the optimization of arbitrary secondary objectives during AdaBoost training.

5.2.1 Speeding up Classification in the Probabilistic Boosting Tree

The PBT distributes a decision to a hierarchy of classifiers, each component of which can in turn be rather small. As described in Sec. 2.2.2, during application parts of the hierarchy may be omitted if their influence on the decision would be too small to matter. This on-the-fly subtree pruning mechanism partially compensates for the additional complexity of the hierarchical design. However, in contrast to the cascade the PBT's primary design goal was not classification speed but minimizing the average loss. The sequential evaluation of all the small classifiers in the hierarchy can therefore be slower than a single large classifier of equal power.

Looking at this classifier architecture and its mode of operation, two approaches for an additional speed-up come to mind almost immediately: Explicit pruning by restricting the depth of the hierarchy and implicit pruning by training stronger learners in the nodes.

The effect of the former is obvious. A reduced tree depth means less classifiers in the hierarchy and can thus save a great deal of processing time. Since the PBT is a binary tree, reducing the depth by one level can reduce the number of nodes by up to 50%. As Carneiro et al. [Carn 08] showed, a smaller tree does not necessarily imply lower classification accuracy. Since the PBT tends to overfit the training data, keeping its depth minimal is even beneficial for generalization performance.

In contrast, the implicit pruning approach does not add any restrictions to the training procedure. Instead, stronger AdaBoost classifiers are trained by relaxing its stopping criterion and using more or stronger hypotheses. While this may lead to an increased worst case classification time, on average less tree nodes will have to be evaluated. The stronger node classifiers generate more certain results, which in turn leads to more pruned subtrees.

While both approaches reduce processing time and in the case of explicit pruning also help to avoid overfitting, they also both contradict the idea of the PBT. This classifier's power stems from the fact that it is hierarchical. Stronger classifiers in the nodes would mean giving up this concept and ultimately approaching a single AdaBoost classifier.

We propose an entirely different method for speeding up classification. As for Heisele et al. [Heis 03] or the cascade, the starting point is the observation that for many samples not the entire tree is evaluated. Since evaluation can stop at any point of the hierarchy, the root node is the only one that is passed by all samples. Consequently, this node should have the fastest classifier. Going deeper in the tree, nodes have to classify less and less samples, meaning that for these fast classification is of lesser importance.

Since for simple classifiers like AdaBoost with decision stumps most time is consumed by feature calculation, this is the target of the presented approach. The idea is to use

simpler features in high tree nodes, where simple in this context means fast to compute. Using less complex features could obviously reduce the overall complexity of the classifier considerably. However, as simpler features often do not have the same discriminative power, we do not want to discard the more complex ones entirely. Therefore, in deeper nodes, where less samples have to be classified and thus speed is less crucial, all features should be available to the algorithm. A side effect of restricting feature selection is that the nodes using only simple features may be slightly weaker. It may thus happen that more weak learners are used in total and some samples go further down in the tree than originally, but experiments indicate that, if the differences in feature complexity are big enough, this will still lead to a substantial improvement.

In the following two sections, we describe two ways of incorporating this concept into the learning process of a PBT. As mentioned before, the principle is transferable to other hierarchical classifiers as well. Prerequisites are that each node of the hierarchy has its own, separately trained classifier and that during application deeper nodes classify less samples than high nodes. Also, features of different computational complexities have to be used.

5.2.2 Cost Constrained Hypothesis Training

In order to make AdaBoost prefer simpler features during training, the weak learning procedure has to be adapted. During each iteration t AdaBoost calls its weak learning algorithm that generates a new hypothesis h_t for the ensemble. h_t is the result of optimizing a function $\varphi : \mathscr{H} \mapsto \mathbb{R}$ with $\mathscr{H}$ being the hypothesis space containing all possible hypotheses that can be generated by the weak learning algorithm. In the simplest case, the optimization criterion is solely based on the misclassification error with respect to the current sample weight distribution D_t

$$\varphi_t(h) = \sum_{i:h(\mathbf{x}_i)\neq y_i} D_t(i).$$

This formulation is now extended by incorporating a cost term $\kappa : \mathscr{H} \mapsto [0,1]$ that is defined on the hypothesis space and assigns a cost to each hypothesis. The new criterion to be optimized by the weak learning algorithm then becomes

$$\psi_t(h) = \lambda \cdot \kappa(h) + (1-\lambda) \cdot \varphi_t(h) \tag{5.1}$$

with $\lambda \in [0,1]$ a weighting factor trading off the influence of the hypothesis cost vs. its training error.

In our segmentation system decision stumps are used for weak learners. Since a decision stump consists of the application of a threshold to a single feature, its training originally comes down to selecting the single most discriminative feature at each iteration based on its weighted error on the training data. Consequently, if classification speed is to be optimized by this mechanism, the cost of a hypothesis has to be defined as being the computational complexity of the used feature, may it be determined analytically, empirically (by runtime measurement) or heuristically (by manually setting arbitrary values).

Since the primary goal of the AdaBoost learning is still error reduction, the cost term is used exclusively for weak learner training while the rest of the procedure remains untouched (cf. Alg. 5): The hypothesis error $\theta_t = \sum_{i:h_t(\mathbf{x}_i)\neq y_i} D_t(i)$ is used for calculating α_t and D_{t+1} as before. Consequently, looking at Sec. 2.2 again, the derivation of the error

Algorithm 5 The cost constrained AdaBoost algorithm

Input: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ where $\mathbf{x}_i \in \mathbb{R}^m$, $y_i \in \{-1, +1\}$. A cost function $\kappa : \mathscr{H} \mapsto [0,1]$. A weighting factor $\lambda \in [0,1]$.

1: Initialize $D_1(i) = 1/n$.
2: **for** $t = 1 \ldots T$ **do**
3: Train weak hypothesis $h_t : \mathbb{R}^m \mapsto \mathbb{Y}$ that minimizes the objective function $\psi_t(h) = \lambda \cdot \kappa(h) + (1-\lambda) \cdot \varphi_t(h)$ using distribution D_t.
4: Get weighted training error of hypothesis $\theta_t = \sum_{i:h_t(\mathbf{x}_i) \neq y_i} D_t(i)$.
5: Set $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\theta_t}{\theta_t} \right)$
6: Update

$$D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)}}{Z_t}$$

with

$$Z_t = \sum_i D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)} = 2\sqrt{\theta_t(1-\theta_t)}.$$

7: **end for**

Output: Final hypothesis $H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}) \right) = \text{sign}(g(\mathbf{x}))$.

bounds for AdaBoost remains the same. All that is required for this derivation is that the hypothesis error θ_t stay below 0.5, which can always be guaranteed for a two-class classifier. Therefore, the initial upper bound on the training error Θ_f of the final ensemble identified by Freund and Schapire [Freu 95] remains intact as

$$\Theta_f \leq \prod_{t=1}^{T} \sqrt{1 - 4\gamma_t^2} \leq \exp\left(-2 \sum_{t=1}^{T} \gamma_t^2 \right), \; \gamma_t = 0.5 - \theta_t,$$

guaranteeing convergence as long as the minimum requirements for a weak learner are met. By iteratively rejecting the found hypothesis and using the next best with respect to ψ_t where necessary, even stricter requirements for the hypotheses may be met leading to tighter error bounds. Of course, using this combined optimization criterion can in general lead to the selection of suboptimal (with respect to their classification error) hypotheses. This has two implications: Since the individual γ_t may be larger than in the original version, convergence might be slower in the end as AdaBoost may have to generate more hypotheses than without the cost term. And, in the terms of Rudin et al. [Rudi 04], this is not "optimal" AdaBoost. For this case Rudin et al. could show that AdaBoost is not guaranteed to produce a maximum margin solution even if the "optimal" version always choosing the hypothesis with the lowest classification error would. However, experiments with clinical data indicate that in practice, this does not affect performance negatively in the hierarchical classifier setting.

To achieve the intended runtime optimization of the PBT, the selection of the weighting factor λ is crucial. The goal is to have fast AdaBoost classifiers at the top of the hierarchy and slower, more accurate ones at deeper tree levels. To this end, λ has to be set to a high

value for the root node to put a strong emphasis on the hypothesis cost and thus ensure the selection of simple features. At deeper tree levels, λ can be decreased to reduce the influence of the cost factor. That way the most accurate features are chosen regardless of their higher complexity.

One thing to note from Eq. (5.1) is its generic formulation. This approach is by no means limited to runtime optimization. The weak requirements on the cost function allow the introduction of arbitrary side conditions into the AdaBoost training procedure. This could be memory constraints, stability considerations, personal preferences like the transparency of a classifier decision, reliability or the actual cost for acquiring a feature value. Also, for other types of weak learners or objective functions, the cost term can be incorporated in a similar fashion.

5.2.3 Adaptive Cost Constrained Hypothesis Training

Besides the hypothesis cost function κ, the previous section introduced the weighting factor λ as additional parameter into the training procedure. This trades off complexity vs. discriminative power during weak learner training, allowing the generation of simple and fast AdaBoost classifiers as well as slow but accurate ones by changing only the value of λ. In this section, the method is automated even further, making it more elegant by removing the need for setting the weighting factor manually. Instead, it is set adaptively for each AdaBoost training based on the importance of the AdaBoost classifier currently being trained.

In our runtime optimization setting, node importance is measured by the amount of data that has to be processed, which is the fraction of samples that have to be classified by this node. During training, when starting a new AdaBoost training, the number of available training samples at the current node $|\mathbf{S}_c|$ is compared to the total number of samples $|\mathbf{S}|$ the root node was trained with. If the training set is representative for the data to be encountered during application stage, the weight factor can then be set according to the expected fraction of data to be processed by this node. In our experiments we set this parameter to be $\lambda = 0.5 \cdot \frac{|\mathbf{S}_c|}{|\mathbf{S}|}$. Depending on the classification problem and the structure of the hypothesis cost function κ, one might want to choose λ to approach 0 more rapidly or actually reach it.

5.3 Results and Discussion

The effect of the proposed constrained boosting method was evaluated with the original segmentation system described in Chap. 3 on the same set of 15 clinical images. However, since in this experiment we are only interested in changes in the computational complexity of the classifier, only the first PBT of the iterative classification step was used and postprocessing was deactivated.

As before, each experiment was set up as a five fold cross-validation. For the experiments, the PBT was trained with $\varepsilon = 0.2$ and a maximum depth of 5. AdaBoost training was stopped at a hypothesis error of $\theta_t \geq 0.45$ or after 30 iterations.

5.3.1 Experimental Setup

The optimization of the learning algorithm described above aims at reducing the computational complexity of the classification system during application stage. To measure this effect, experiments were run with and without the optimization and their outcome was compared in terms of application cost and classifier cost. Three experiments were run for each of the two described configurations:

- One training a classifier with the original learning method ("No Cost"),
- one with the cost constrained algorithm and a runtime based hypothesis cost function ("Empirical Cost") and
- one with randomly chosen cost values ("Random Cost"),

all other parameters set identically.

To determine the runtime based hypothesis cost function we chose a very hands-on approach. Each feature was calculated on the target machine and the elapsed time was measured, averaged over several thousand runs. For the random cost function, hypothesis cost values were uniformly randomly drawn from the range $[0,1]$. By using this cost that is completely unrelated to any feature semantics or power, the general optimization potential for arbitrary side conditions is demonstrated.

If the used cost function is based on the feature calculation time, a reduction in cost also means faster evaluation. A random cost function on the other hand cannot be expected to improve runtime, so the more general cost based approach had to be taken to show the effect of the cost constrained training for this setup.

The overall cost of classifying a new sample with the PBT cannot be determined analytically because of the subtree pruning. Instead, one has to actually apply the classifier to the sample and accumulate the costs of all encountered hypotheses along the way. This yields the classifier's application cost $\omega_{pbt}(\mathbf{x})$ for a sample $\mathbf{x}$. The overall cost for a dataset z, represented as the set of samples $\mathbf{X}_z = \{\mathbf{x}_1,..,\mathbf{x}_u\}$, would then be $\Omega_{pbt}(\mathbf{X}_z) = \sum_{i=1}^{u} \omega_{pbt}(\mathbf{x}_i)$. For each fold of the cross-validation, the total cost of all datasets in the test set is calculated. Thus, after a full cross-validation, the final result is the total evaluation cost of all datasets in the test database.

Besides comparing the actual cost when applying the different classifiers to a sample, we also analyzed how the modified training procedure changes the internal structure of the classifiers. To this end, their classifier cost was determined and compared. For this static analysis, the cost of the single hypotheses forming the PBT is accumulated. More precisely, the values for all hypotheses at a certain tree level are summed up and averaged over the number of nodes at this level. This procedure also unveils the influence of the weighting factor λ.

Obviously, measuring the improvement in runtime or cost is not sufficient to show the validity of the approach. We have to show, that the complexity reduction does not degrade classification accuracy over proportion. To this end, ROC curves of the classification accuracy of each configuration are compared.

5.3.2 Cost Constrained Hypothesis Training

For this set of experiments, the hypothesis cost values $\kappa(\cdot)$ were scaled to the range $[0, 0.5]$. This range was used instead of $[0, 1]$ in order to have the cost values in the same range as the hypothesis errors. Having the two values in the same range makes setting and interpreting the weighting factor λ during training straightforward. Here, λ was initially set to 0.5 and then linearly decreased to reach 0 at the deepest level of the hierarchy.

Compared to the standard PBT training, the total application cost could be reduced by 58% by the proposed feature selection method with an empirical cost function, without losing classification accuracy as can be seen from the ROC curve in Fig. 5.1(a). This cost takes into account only the feature calculation part of the classification, not the overhead produced by the hierarchical classifier. The actual runtime reduction depends on various implementation factors. For the system used in these experiments, the 58% lower cost translates to a runtime reduction of 52%.

Figure 5.2 shows the result of the static analysis, comparing the classifier cost for the different configurations. The constrained feature selection worked as expected: In high nodes, AdaBoost chose features with lower cost than with the standard PBT version, thus the much higher cost per tree level and per node of the unconstrained PBT ("No Cost") for depths $1-4$. In deeper nodes, which are rarely evaluated, the cost penalty in the constrained algorithm was decreased. Hence, the AdaBoost procedure focused more on the weak learners' classification error and the average cost increased.

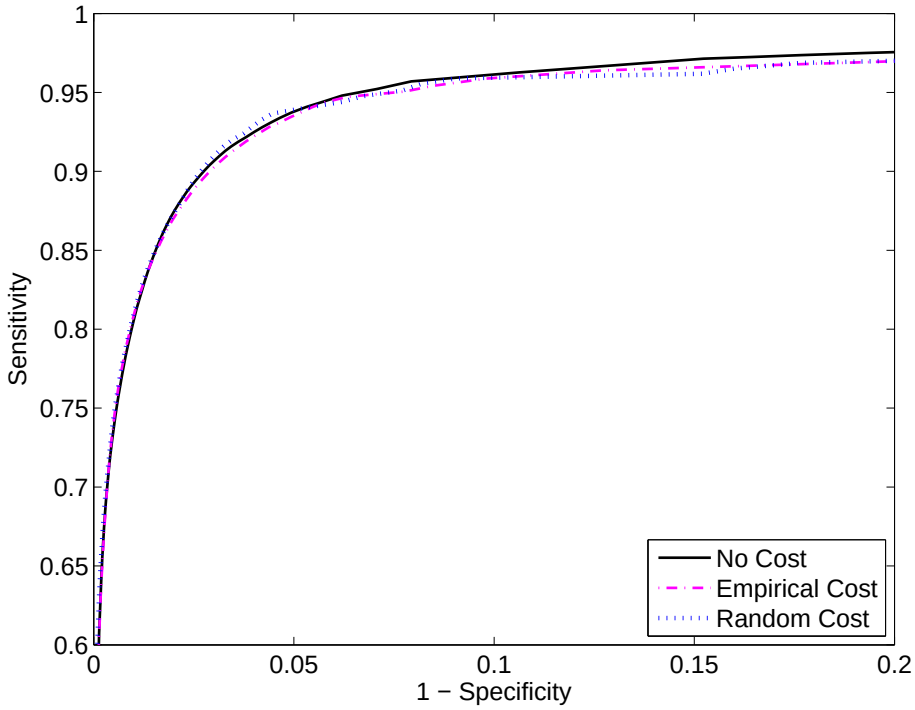
The fact that for the unconstrained training the average cost decreases with increasing tree depth supports the claim that there is a loose correlation between feature complexity and discriminative power. AdaBoost learning, by nature a greedy approach, chooses the strongest features first, which turn out to be the most expensive ones. For deeper tree levels, these do not provide any additional insight, so that simpler features are preferred.

The empirical cost constrained PBT exhibits the opposite behavior, using the cheapest features first and gradually switching to more expensive ones in deeper tree levels. That way its curve for the total cost per tree level is steeper, even exceeding the unconstrained one at leaf level, where λ is set to 0.

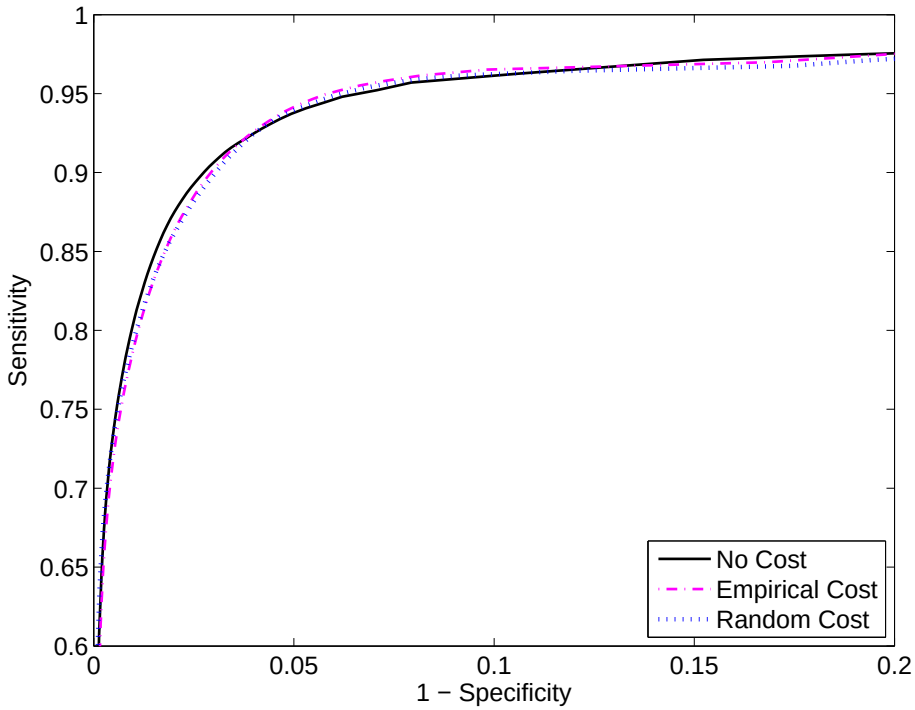
The experiments with random hypothesis cost show, however, that this correlation between feature cost and discriminative power is not required for successful optimization of a side condition. In Fig. 5.2(d) the average node cost remains fairly constant, showing the uniformity of the cost function. Still, even for these arbitrary assignments of cost values to features, the evaluation cost is reduced by 42%. This supports the claim made by Eq. (5.1): The proposed method is by no means limited to runtime optimization. Using this method, the PBT training can optimize any secondary condition without losing track of the classification error.

5.3.3 Adaptive Cost Constrained Hypothesis Training

Trading off the cost and the training error of the hypothesis adaptively as described in Sec. 5.2.3 lets the learning algorithm control the optimization process even further. The importance of a node and thus the value of λ now does not depend on the depth of the node any more, but directly on the fraction of samples that will have to be classified by this node. This method achieves an even higher application cost reduction than before, namely 76%. This equals a runtime reduction of 69%. The random cost function yields



(a)



(b)

Figure 5.1: Classification accuracy of the different training configurations. (a) shows the result with manually chosen λ, (b) with adaptively set λ.

63% improvement. Again, classification accuracy is not degraded as can be seen from Fig. 5.1(b).

A side effect of this setup is that the interpretation of the cost graphs (Figure 5.3) changes. For the total cost per tree level (Figure 5.3(a), (c)), the shape of the curve is still somewhat predictable: It starts with a low cost, since the situation at the root node remains the same as in the previous experiment. The slope may be steeper, because a good separation at one node leads to a low value of λ and thus a high cost at one of the child nodes. The total cost at leaf level is nevertheless lower than in the previous experiment, because there λ was set to 0 at leaf level while here the cost may still have some influence.

The average cost per node, on the other hand, does not show a clear behavior any more. Looking at the values of λ alone, one could argue that the average cost should stay nearly the same throughout the tree or decrease slightly. The samples at any node are split up and distributed to its children, so λ is distributed accordingly, with a small gain due to samples put into both subtrees. There are, however, many influence factors disturbing this fragile balance, such as the fact that any two nodes at the same level may have very different values of λ, the greediness of AdaBoost, or outliers from the correlation between feature cost and discriminative power. The plots in Fig. 5.3(b), (d) are thus only given for completeness.

5.3.4 Free Lunch?

At first glance the above results seem surprising. PBT training can be modified so that classification takes 69% less time without losing classification accuracy, just by restricting the use of certain features. The algorithm can even be used to improve not runtime but any (even random) constraint.

Thinking one step further it becomes clear what has to be happening during the training procedure. As stated in Sec. 2.2.1, one advantage of AdaBoost over other learning methods is its ability to deal with a large number of features. One can leave the algorithm with a huge set of features and during training it will select the most useful ones. In the end, the classifier will always use only a small subset of the provided features. Starting with a smaller number in the first place could result in the same classifier, but then the user would have to perform the feature selection. So, many features may simply be left out because there are cheaper ones that are only slightly worse w. r. t. the classification error. However, this may lead, as mentioned in Sec. 5.2, to a higher number of hypotheses.

The second effect yielded by the constrained feature selection compensates for this higher number of hypotheses: The order of evaluation of the features is changed. By introducing the weighting factor λ and adapting it at different tree levels/nodes, one can control the distribution of hypothesis cost over the tree. When looking at the case of a runtime based cost function, one can shift the use of the most expensive features away from the first few nodes that have to classify the highest number of samples. That way these nodes will likely lose some accuracy, so that some samples will move deeper in the tree than in the unconstrained case. But this will not be the case for all samples, which is why this constrained tree can, on average, be considerably faster. If a sample needs to be classified by all nodes of the tree, the effort needed to classify it is the same as in the unconstrained tree, only the order of evaluation changed.

So, to summarize the above, taking the metaphor further, this method does not claim any free lunch, but the original method paid too much. The redundancy in the original fea-

ture set gives room for optimization. Apart from this, only the order of feature evaluation is changed, with the goal to use the cheapest features as an early filter.

5.4 Conclusion

In this chapter, we introduced an extension to the AdaBoost algorithm that allows incorporating a user defined constraint into the hypothesis selection process during classifier training. In a study with clinical CT images we showed that, using this method, the computational complexity of the PBT that forms the basis of our recently proposed liver tumor segmentation system could be reduced by 76% by incorporating a measure of feature cost. The same method can be applied to the AdaBoost cascade [Viol 01] or other, similar hierarchical algorithms that contain a means of pruning parts of the classifier during application.

To fully optimize the runtime of an algorithm, a single modification will rarely be enough. The method presented here can be combined with others like the ones discussed in Sec. 5.1.

However, this extension to AdaBoost is by no means limited to complexity reduction. The algorithm works independent of the semantics of the cost function. This method thus gives the user a powerful means of control over the AdaBoost procedure, allowing the simultaneous optimization on any additional criterion that can be defined at hypothesis level.

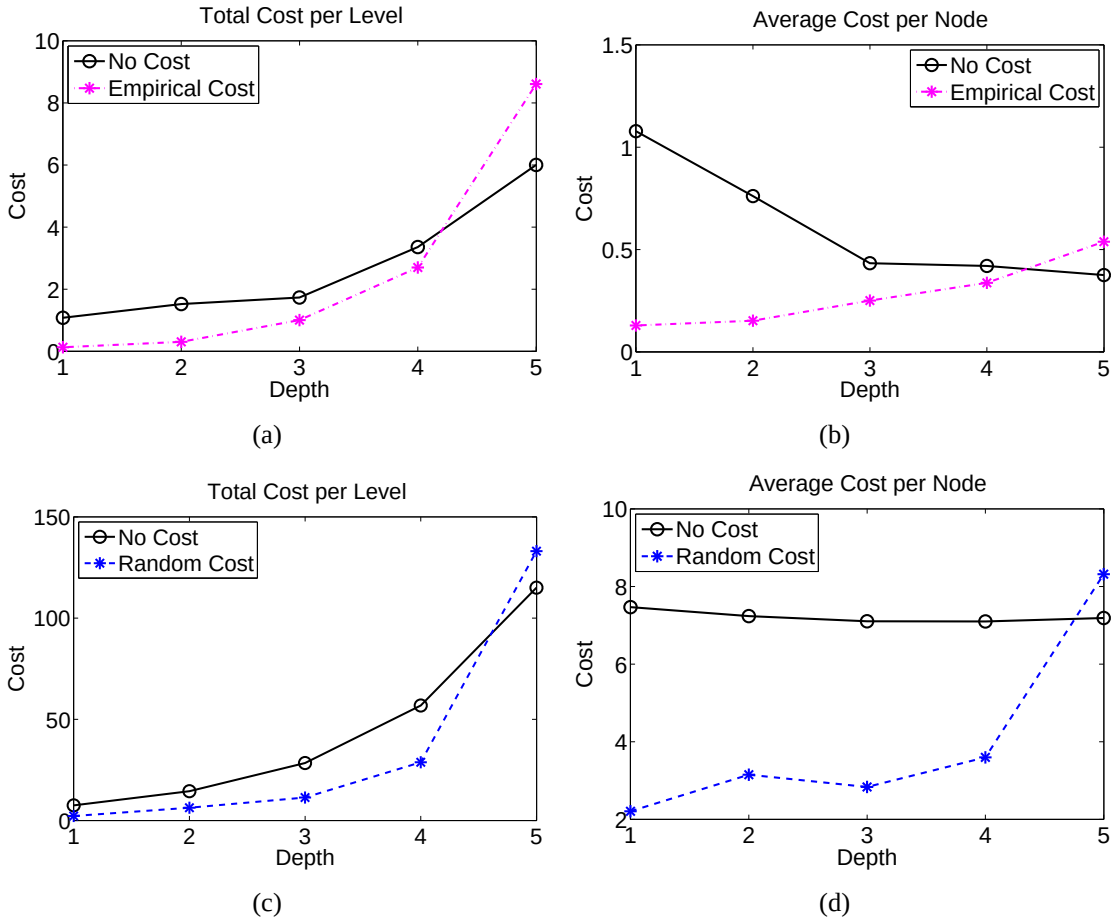


Figure 5.2: Comparison of classifier costs for the PBT if the weighting factor λ is set to decrease linearly from 0.5 at the root to 0 in depth 5. The PBTs are analyzed by accumulating the cost of all hypotheses at each tree level (=depth) ((a), (c)) and averaging the resulting values over the number of AdaBoost classifiers at this depth ((b), (d)). The top row shows the comparison of an unconstrained classifier, trained without any cost function (solid black curves), with a classifier trained with an empirical cost function. Both classifiers were evaluated using the same cost function to show the effect of said function on the training procedure. The bottom row shows the same comparison between the unconstrained classifier and one that was trained with a random cost function. Again, both classifiers were evaluated with the same random cost function.
In all configurations the total cost increases with depth, as the number of nodes (and thus the number of hypotheses) increases. The generally much higher costs in the random cost case (note the different scales at the *y*-axes) stem from the fact, that the feature set contains mostly Haar-like features, which receive very low cost values in the empirical cost function. Since the random cost function is uniformly distributed, it is likely to assign much higher cost values to these features.
Analysis of the average cost per node shows, that including the cost function in the training procedure has the expected effect: For the constrained PBT cheaper features are preferred in the higher nodes of the tree.

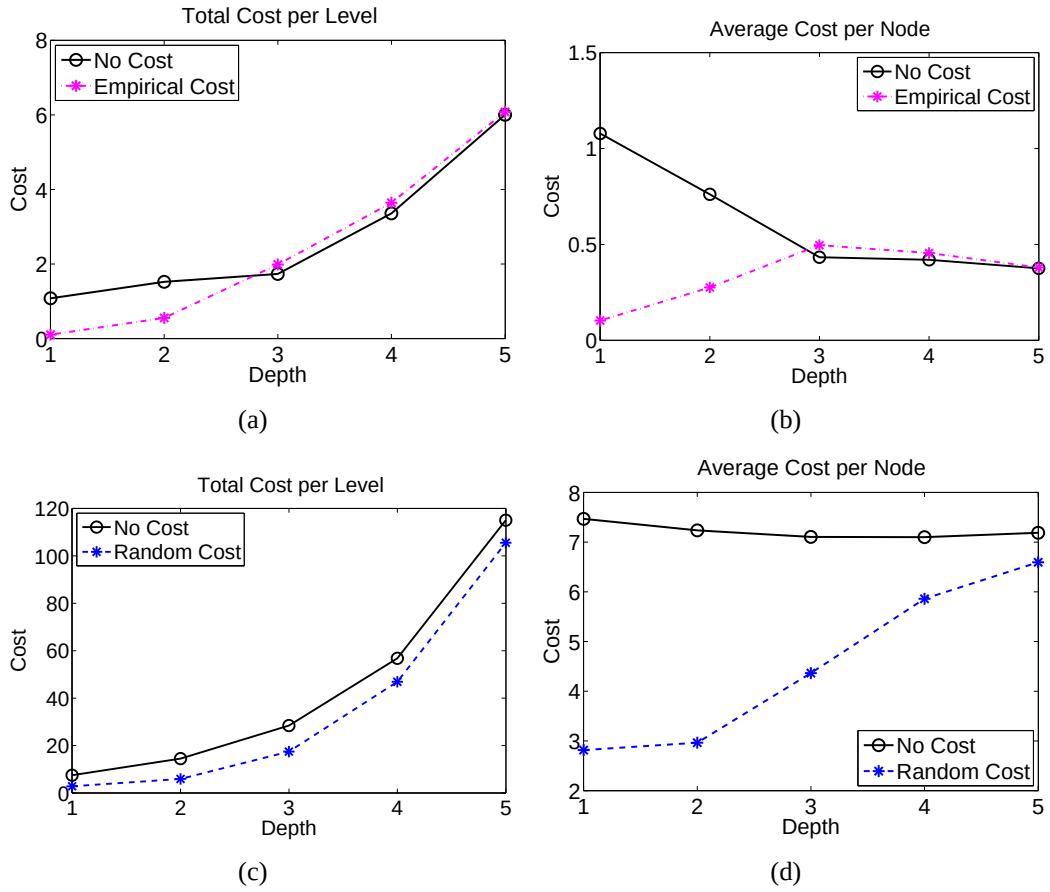


Figure 5.3: Comparison of classifier costs for the PBT with λ set proportionally to the fraction of samples to be classified at the respective node. As in Fig. 5.2, the cost of all hypotheses is accumulated at each tree level ((a), (c)) and averaged over the number of AdaBoost classifiers at this depth ((b), (d)).
The top row shows again the comparison of an unconstrained classifier (solid black curve) with one that was trained using the empirical cost function. The bottom row shows the same comparison between the unconstrained classifier and one that was trained with a random cost function.
The average node cost does not show a clear behavior, because λ is now independent of the depth and thus at any given depth nodes can have very different costs. The total cost per level is similar to the previous experiment, with a slightly steeper slope in early levels and a lower total cost at leaf level.

CHAPTER 6

Outlook

This work takes another step towards an automatic solution to the problem of segmenting liver tumors in CT images. It analyses new ways of learning priors from data and presents modifications of existing learning algorithms that reach well beyond the domain of medical image processing. However, for every project borders have to be drawn at some point, so that while many questions could be answered some new ones were raised as well.

One limitation of this work is its focus on a single kind of lesions. The focus was put on hypodense liver lesions in order to keep the complexity of the problem space at bay and that way reduce the necessary amount of training data. In principle, the approach is by no means limited to this type of tumors. It could just as well be applied to hyperdense, rim-enhancing or even completely inhomogeneous lesions, although besides the additional training data some more features capturing the intricacies of these targets may become necessary. One natural extension in this context would be to use not only images with venous contrast enhancement but combine them with arterial images or even other contrast phases. The combination could be either an early fusion approach, combining features of matched voxels into one feature vector and feeding this into a single classifier. Or one could have separate classifiers for each image type and combine their results afterwards. The implications of this design decision would be similar to those formulated for the integration of the lesion prior into the segmentation system described earlier (Section 4.4). Preliminary experiments with additional arterial phase images show promising results, especially for lesions like focal nodular hyperplasia, which often present themselves isodense in venous images but strongly hyperdense in arterial ones (cf. Fig. 1.6(a)). With sufficiently large databases one might even be able to settle the discussion among radiologists about the benefit of multiphasic image acquisitions for diagnosis of liver tumors.

Regarding our work on learning priors for follow-up segmentation there is one question that comes to mind almost immediately. One prerequisite for learning the prior from training image pairs and applying it to new ones is that the image pairs are registered non-rigidly. The state-of-the-art method used here performs reasonably well on this difficult task. It would, however, be interesting to see how different methods with different errors influence the performance of the learner and the segmentation system. Knowing the behavior of a registration algorithm one may even be able to generate realistic errors artificially, providing a higher degree of control over the evaluation. Given a sufficient number of follow-up images to allow reliable statements on influence factors this could provide valuable insight into the nature of the learning algorithm, its learning task and the used feature space.

In general, the methods and evaluations presented here are focused on automatic segmentation, i. e. a decision on voxel level what parts of the liver consist of healthy tissue and what parts belong to a lesion. The voxel classification approach does also provide a

detection and has the advantage that it is in principle independent of the size and shape of a target lesion. Pure detection methods like the window classification approach by Viola and Jones [Viol 01], on the other hand, provide access to new sets of features describing whole objects as opposed to single points inside an object. Also, they are in general more robust against noise in the input data. As a drawback, they require more training data because each lesion makes only one positive example (plus possibly affinely transformed versions of the lesion). For a voxel classification approach, each voxel of a lesion makes one positive example. Nevertheless, it would certainly be worth investigating how such a window based detection performs and how a separate module only for detection can simplify the subsequent segmentation task.

Regarding the detection capabilities of the system presented here, results can certainly be improved by introducing a filter step into postprocessing. This filter would consist of a classifier that receives a feature vector for each lesion candidate and decides whether this candidate is a true detection or a false positive. Of course, this filter must not remove any true detections, but still it should be able to identify some false alarms.

CHAPTER 7

Summary

Modern medical imaging devices are able to acquire images of an amazingly broad range of targets. With X-Ray Computed Tomography, anything from the blood vessels of a beating heart to the entire muscoskeletal system can be captured. Thanks to various types of contrast agents this modality even pushes into the original domain of Magnetic Resonance or Ultrasound imaging, the visualization of soft tissue. For diagnosis and monitoring of liver tumors it has become a standard tool in clinical routine. However, while image acquisition has made enormous progress over the past decades, computers and algorithms processing the ever increasing amount of images can hardly keep up with the pace of development. In fact, most processing to this day is focused on improving image quality in order to give a better or more clear presentation to a human observer while exposing the patient to ever less ionizing radiation. While this effort was without doubt very successful when looking at today's images in comparison to those of the early 90s, this is not the end of what computers can do. The tools for visualization or measurements provided to the physician are still very basic. A growing community of researchers and companies therefore bring methods of artificial intelligence, pattern recognition, statistics and image understanding into the market in order to provide tools that support clinical workflows by actually interpreting images and generating condensed information out of them.

The domain of medical images is a very complex one. Every patient has a different anatomy and metabolism. Many acquisition parameters influence the appearance of images. Differences between relevant structures and background or noise may be very small. That is the reason why to this day in most fields of application no algorithm can match a human when it comes to extracting relevant objects from the images or making decisions on disease status. Despite all progress made in specialized fields like skin cancer diagnosis or analysis of coronary arteries we are still barely scratching the surface of what could be done if only we had the right tools: algorithms that can adapt to this complex and ever changing environment as the human brain does seemingly without effort.

The thesis at hand takes one more step in this direction, investigating machine learning methods for the segmentation of liver tumors in CT images. Using contrast agent, CT images of the liver can be acquired showing lesions that would otherwise be isodense to the surrounding liver tissue. Especially useful for the assessment of liver tumors are images in which the contrast agent just entered the liver via the portal vein, because most tumors receive their blood supply via this path. In order to judge the disease status of a patient and to monitor it over time a physician will have to measure a representative subset of the lesions in each of the patient's images. The goal of an algorithmic solution here would be to provide a 3D delineation of all lesions of the patient in order to allow for precise volume measurements. For complex tasks like this, recently machine learning techniques have proven most useful. These allow to build systems that adjust their parameters and

thus learn their functionality based on examples. In the case at hand a classifier is built by providing it with example images of healthy and diseased livers. After this training phase the resulting classifier is ready to be used on new images. Its purpose is to decide for single points (=voxels) of a CT image whether they are part of a liver tumor or not. The classifier that is used here is a Probabilistic Boosting Tree (PBT), a hierarchical classifier combining a number of AdaBoost classifiers.

The PBT forms the core of the presented segmentation system. The input to the system consists of a liver CT image acquired during portal venous phase of contrast enhancement. The image is preprocessed by automatically segmenting the liver in the image and standardizing image intensities of the voxels inside it. The former reduces the search space for the further processing, the latter simplifies the classification task by eliminating differences in the images due to acquisition timing or other external factors. In the main segmentation step, for each voxel inside the liver a vector of about 2500 features describing the voxel's appearance is calculated. From this feature vector the PBT estimates a probability for the voxel to belong to a tumor. From the resulting probability image about 60 additional features are calculated and appended to the vector of appearance features. This new vector forms the input to another PBT. This process is repeated with up to four PBTs, resulting in a cascade of classifiers. Due to its hierarchical decision making, the PBT as a classifier is very well suited for complex problem spaces like the one at hand. Using a cascade instead of a single classifier leads to more robust results with less spurious misclassifications, improving precision by 38% at 90% sensitivity. Very similar to neural information processing the classifier cascade has the effect that neighboring voxels influence each other iteratively in such a way that each voxel's probability value stabilizes over time. Therefore, the final output of the cascade needs no elaborate postprocessing. Instead, it is only filtered using a morphological opening and a median filtering before thresholding it to yield the output lesion mask.

To monitor disease progression over time, one could segment and measure lesions in every image acquired from the patient during each examination separately. However, we chose to optimize the method for this task in order to further reduce error rates. To this end we propose a method for generating a patient specific lesion prior that can guide the system in its segmentation task. A prior model is learned from registered pairs of images of the same patient acquired at different times together with a reference segmentation for the first of the two images. This model encodes lesion growth and shrinkage over time. Given the first image of a new patient together with its lesion segmentation and an image acquired a few months later, the system can compute a patient specific prior from the model providing insight where to expect lesions in the second image. Two ways of incorporating the prior into the segmentation system are investigated. The prior can be combined multiplicatively with the output of the original lesion segmentation system for the second image. Or it can be incorporated into the system directly by training only one classifier that uses all features of the classifier used for segmentation and the one used in the prior model in one vector. While the latter is preferable because it can detect and exploit additional patterns and dependencies in the data, experiments indicate that it requires a substantial amount of additional training data. In general, our experiments on clinical data show a considerable advantage of the systems using a learned prior, no matter how it is used. Measuring the segmentation performance on voxel level, we found a reduction of false positive classi-

fications per true positive by 57.8% at 90% sensitivity if the prior is integrated into the system.

For systems like the one presented here, where a complex analysis is required for each image point, runtime can become an issue in clinical routine. With modern imaging devices, large 3D datasets can be acquired, whose processing can take several minutes up to hours depending on the type of analysis performed. With increasing capabilities of the methods their runtime demands have therefore to become target of research themselves. Starting from the observation that in detection tasks often most of the input images consists of background, large parts of which can easily be filtered out, we therefore develop a means to build a PBT that can save a great deal of processing time compared to the original version. We propose a redesign of the PBT, where the topmost nodes are constructed much simpler than the deeper ones. This makes use of a property of the PBT during application phase: A feature vector is handed down the tree and classified at each node along the way. If, however, such a node classifier is very certain about the label of the example, it is only handed down the corresponding subtree instead of both. This leads to the root node being the only one that has to process all examples, while deeper nodes only have to classify those that are most difficult to decide. Since most of the processing time in this classifier is spent calculating features, we incorporate a new term into the AdaBoost training procedure, so that it does not only optimize classification error but at the same time also the cost of the used features. During PBT training, the influence of this cost factor is reduced with increasing tree depth. That way more complex features, that may have larger discriminative power, can still be used towards lower levels of the tree. This modification of the AdaBoost and PBT learning algorithms reduces classification cost in the presented system by up to 76% without losing classification accuracy.

A topic of this complexity can never be studied exhaustively. Solving one question will raise two new ones that are worth investigating. In our case an extension to multi phase CT imaging and more types of liver lesions would be the most prominent directions of future research. Approaches that include a separate detection step based on window classification may improve performance further. And obviously a better understanding of the influence of the used registration algorithm on the behavior of the system would be desirable.

CHAPTER A

Appendix

A.1 Evaluation Measures

Comparing the output of a segmentation algorithm with a reference segmentation on a per-voxel basis yields the numbers of correctly and incorrectly classified points. With two possible target classes there are two types of correct classifications and two types of error, so four values in total. These can be represented as 2×2 matrix known as confusion matrix (Table A.1). From the values in this table, several overlap measures can be calculated to

Classification	Reference: lesion	Reference: background
Algorithm: lesion	True positive (tp)	False positive (fp)
Algorithm: background	False negative (fn)	True negative (tn)

Table A.1: Confusion matrix. Based on the numbers of true positives, false positives, true negatives and false negatives, the quality of the segmentation is assessed.

characterize the performance of the classification or segmentation algorithm (Table A.2). Each measure highlights different aspects of the segmentation by putting emphasis on different parts of the confusion matrix.

Sensitivity	$\frac{\#tp}{\#tp+\#fn}$
Specificity	$\frac{\#tn}{\#tn+\#fp}$
False positive rate = 1 - specificity	$\frac{\#fp}{\#fp+\#tn}$
Jaccard index	$\frac{\#tp}{\#tp+\#fp+\#fn}$
Dice coefficient	$\frac{2\cdot\#tp}{2\cdot\#tp+\#fp+\#fn}$
Precision	$\frac{\#tp}{\#tp+\#fp}$

Table A.2: Performance measures calculated from the entries of the confusion matrix.

List of Symbols

ν_{tissue}	Attenuation coefficient of material type *tissue*	2
f	Function mapping a feature space to a target space	13
$\mathbb{X}$	Feature space	13
m	Dimensionality of feature space $\mathbb{X}$	13
y	Target variable	13
$\mathbb{Y}$	Target space	13
$\mathbf{x}$	Feature vector	14
$\mathbf{x}_a$	Vector of appearance based features	29
$\mathbf{x}_p$	Vector of features based on probability images	31
$\mathbf{x}_s$	Vector of geometry features	56
$\mathbf{x}_c$	Concatenation of feature vectors $\mathbf{x}_a$ and $\mathbf{x}_s$	58
$\mathbf{X}$	Set of feature vectors	14
$\mathbf{V}$	Set of training examples (feature vectors) with class labels	14
n	Number of training examples in $\mathbf{V}$	14
E	Expectation value	15
ξ	Single concept	14
$\mathscr{C}$	Class of concepts	14
h	Single hypothesis, weak learner	14
D	Probability distribution over training examples	15
θ	Training error of a hypothesis	15
$\mathscr{H}$	Space of all possible hypotheses	15
H	Strong classifier	16
T	Maximum number of iterations t in AdaBoost	16
Θ	Bound for training error of AdaBoost	17
Z	Sample weight normalization factor used during AdaBoost training	17
α	Weight of single weak hypothesis in AdaBoost classifier	16
v	VC dimension	17
ϕ	Margin of a classifier for a sample	20
$\mathbf{S}$	Set of weighted training examples	21
w	Weight of training example	23
L	Maximum depth of PBT	23
q	Approximate posterior probability calculated by strong learner	21
ε	Splitting softness parameter of the PBT	21
σ	Standard deviation	29
I	Image intensity	29
μ	Mean value	29
$\mathbf{l}$	Voxel location in image space	31
k	Number of classification steps in iterative scheme	31
N	Spatial neighborhood around a location in image space	31
M_0	Lesion mask of baseline image	54

κ	Cost function	77
λ	Weight trading off different influence factors of a formula	77
φ	Original optimization criterion for AdaBoost	77
ψ	Optimization criterion for AdaBoost with cost factor	77
ω	Cost for classifying a sample $\mathbf{x}$	80
Ω	Accumulated cost for classifying all samples of a dataset z	80
u	Size of a set of feature vectors	80

List of Figures

1.1 CT values of different tissue types . . . 2
1.2 CT images demonstrating the effect of contrast agent . . . 3
1.3 Anatomical view of liver surface . . . 4
1.4 Functional liver segments . . . 5
1.5 Enhancement curves of liver parts after contrast agent injection . . . 6
1.6 Enhancement patterns of various liver tumors . . . 9

2.1 Structure of the PBT . . . 21
2.2 Margin for splitting the training samples . . . 22
2.3 Hierarchical subdivision of feature space performed by PBT . . . 23

3.1 Pipeline for liver lesion segmentation . . . 26
3.2 Intensity differences in liver images . . . 28
3.3 Histogram registration for intensity standardization . . . 29
3.4 Neighborhoods around voxel location **l** . . . 30
3.5 Iterative classification scheme . . . 32
3.6 Segmentation ROC: Influence of standardization . . . 37
3.7 Segmentation ROC: Iterative classification scheme with standardized data . 39
3.8 Segmentation ROC: Influence of postprocessing . . . 40
3.9 Segmentation ROC: Iterative classification scheme with original intensities 40
3.10 Detection performance with standardized data . . . 44
3.11 Detection performance with original intensities . . . 45

4.1 Pipeline for follow-up lesion segmentation . . . 53
4.2 Example registration result . . . 54
4.3 Classifier setup for multiplicative learned prior . . . 57
4.4 Classifier setup for integrated learned prior . . . 59
4.5 Cumulative histogram of registration errors . . . 61
4.6 Segmentation ROC: Different priors without postprocessing . . . 63
4.7 Segmentation ROC: Different priors with postprocessing . . . 64
4.8 Segmentation ROC: Comparison of priors with and without postprocessing 65
4.9 Segmentation ROC: Different priors, growing and shrinking lesions separately . . . 67
4.10 Detection result for priors with postprocessing . . . 70

5.1 Influence of speed optimization on classification accuracy . . . 82
5.2 Comparison of classifier costs for linearly decreasing λ . . . 85
5.3 Comparison of classifier costs with dynamically chosen λ . . . 86

List of Tables

3.1 Appearance features used to separate tumor from background points 34
3.2 Probability image features used to separate tumor from background points . 35
3.3 Influence of standardization and iterative classification scheme on segmentation performance 41
3.4 Influence of standardization and iterative classification scheme on maximum Jaccard and Dice coefficients 41
3.5 Influence of standardization and iterative scheme on lesion detection performance 43

4.1 Spatial features calculated from baseline lesion mask to learn the prior . . . 55
4.2 Intensity change features used to learn a prior 56
4.3 Feature vectors used in different discriminative models 60
4.4 Influence of prior information on follow-up segmentation performance . . . 66
4.5 Maximum Jaccard and Dice coefficients for follow-up segmentation with different prior configurations 66
4.6 Follow-up lesion detection performance with different priors 69

A.1 Confusion matrix 93
A.2 Performance measures 93

Bibliography

[Bile 04] M. Bilello, S. B. Göktürk, T. Desser, S. Napel, R. B. Jeffrey, and C. F. Beaulieu. "Automatic detection and classification of hypodense hepatic lesions on contrast–enhanced venous–phase CT". *Medical Physics*, Vol. 31, No. 9, pp. 2584–2593, Sept. 2004.

[Bipa 05] S. Bipat, M. van Leeuwen, E. Comans, M. Pijl, P. Bossuyt, A. Zwinderman, and J. Stoker. "Colorectal Liver Metastases: CT, MR Imaging, and PET for Diagnosis–Meta-Analysis". *Radiology*, Vol. 237, No. 1, pp. 123–131, Oct. 2005.

[Bish 07] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, NY (USA), 2007.

[Bose 01] I. Bose and R. K. Mahapatra. "Business data mining – a machine learning perspective". *Information & Management*, Vol. 39, No. 3, pp. 211–225, Dec. 2001.

[Bris 06] D. Bristow, M. Tharayil, and A. Alleyne. "A survey of iterative learning control". *Control Systems*, Vol. 26, No. 3, pp. 96–114, June 2006.

[Buhl 06] P. Bühlmann and B. Yu. "Sparse boosting". *Journal of Machine Learning Research*, Vol. 7, pp. 1001–1024, June 2006.

[Carn 08] G. Carneiro, B. Georgescu, S. Good, and D. Comaniciu. "Detection and measurement of fetal anatomies from ultrasound images using a constrained probabilistic boosting tree". *Transactions on Medical Imaging*, Vol. 27, No. 9, pp. 1342–1355, Sept. 2008.

[Char 05] A. Charnoz, V. Agnus, G. Malandain, C. Forest, M. Tajine, and L. Soler. "Liver registration for the follow-up of hepatic tumors". In: *Proc. MICCAI*, pp. 155–162, Springer, Palm Springs, CA (USA), Oct. 2005.

[Chef 02] C. Chefd'hotel, G. Hermosillo, and O. Faugeras. "Flows of Diffeomorphisms for Multimodal Image Registration". In: *Proc. ISBI*, pp. 753–756, IEEE, Washington, DC (USA), June 2002.

[Coui 57] C. Couinaud. *Le foie: Études Anatomiques et Chirurgicales*. Masson, Paris (France), 1957.

[Diet 00] T. G. Dietterich. "Ensemble Methods in Machine Learning". In: *Proc. MCS*, pp. 1–15, Springer, Cagliari (Italy), June 2000.

[Down 01] T. Downs, K. Gates, and A. Masters. "Exact simplification of support vector solutions". *Journal of Machine Learning Research*, Vol. 2, pp. 293–297, Dec. 2001.

[Eise 09] E. A. Eisenhauer, P. Therasse, J. Bogaerts, L. H. Schwartz, D. Sargent, R. Ford, J. Dancey, S. Arbuck, S. Gwyther, M. Mooney, *et al.* "New response evaluation criteria in solid tumours: revised RECIST guideline (version 1.1)". *European Journal of Cancer*, Vol. 45, No. 2, pp. 228–247, Oct. 2009.

[Enzw 09] M. Enzweiler and D. Gavrila. “Monocular Pedestrian Detection: Survey and Experiments”. *Transactions on Pattern Analysis and Machine Intelligence*, Vol. 31, No. 12, pp. 2179–2195, Dec. 2009.

[Farz 10] M. Farzinfar, Z. Xue, and E. K. Teoh. “A novel approach for curve evolution in segmentation of medical images”. *Computerized Medical Imaging and Graphics*, Vol. 34, No. 5, pp. 354–361, July 2010.

[Fawc 06] T. Fawcett. “An introduction to {ROC} analysis”. *Pattern Recognition Letters*, Vol. 27, No. 8, pp. 861 – 874, 2006. {ROC} Analysis in Pattern Recognition.

[Ferl 10] J. Ferlay, H. Shin, F. Bray, D. Forman, C. Mathers, and D. Parkin. “Estimates of worldwide burden of cancer in 2008: GLOBOCAN 2008”. *International Journal of Cancer*, Vol. 127, No. 12, pp. 2893–2917, June 2010.

[Feul 10] J. Feulner, S. K. Zhou, M. Huber, J. Hornegger, D. Comaniciu, and A. Cavallaro. “Lymph node detection in 3-D chest CT using a spatial prior probability”. In: *Proc. CVPR*, pp. 2926–2932, IEEE, San Francisco, CA (USA), June 2010.

[Feul 11a] J. Feulner, S. K. Zhou, E. Angelopoulou, S. Seifert, A. Cavallaro, J. Hornegger, and D. Comaniciu. “Comparing axial CT slices in quantized N-dimensional SURF descriptor space to estimate the visible body region”. *Computerized medical imaging and graphics*, Vol. 35, No. 3, pp. 227–236, Apr. 2011.

[Feul 11b] J. Feulner, S. Zhou, M. Hammon, S. Seifert, M. Huber, D. Comaniciu, J. Hornegger, and A. Cavallaro. “A Probabilistic Model for Automatic Segmentation of the Esophagus in 3-D CT Scans”. *Transactions on Medical Imaging*, Vol. 30, No. 6, pp. 1252–1264, June 2011.

[Freu 95] Y. Freund and R. E. Schapire. “A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting”. In: *Proc. EuroCOLT*, pp. 23–37, Springer, Barcelona (Spain), March 1995.

[Freu 96] Y. Freund and R. E. Schapire. “Experiments with a new Boosting Algorithm”. In: *Proc. ICML*, pp. 148–156, Bari (Italy), July 1996.

[Frie 00] J. Friedman, T. Hastie, and R. Tibshirani. “Additive logistic regression: A statistical view of boosting”. *Annals of Statistics*, Vol. 28, No. 2, pp. 337–407, April 2000.

[Frie 01] J. Friedman. “Greedy function approximation: A gradient boosting machine”. *Annals of Statistics*, Vol. 29, No. 5, pp. 1189–1232, Oct. 2001.

[Glob 11] *Global cancer facts & figures*. American Cancer Society, Atlanta, GA (USA), 2nd Ed., 2011.

[Golu 13] M. Golub, S. Chase, and B. M. Yu. “Learning an Internal Dynamics Model from Control Demonstration”. In: *Proc. ICML*, pp. 606–614, Atlanta, GA (USA), June 2013.

[Gooy 11] A. Gooya, K. Pohl, M. Bilello, G. Biros, and C. Davatzikos. “Joint segmentation and deformable registration of brain scans guided by a tumor growth model”. In: *Proc. MICCAI*, pp. 532–540, Springer, Toronto, ON (Canada), Sept. 2011.

[Hayk 99] S. Haykin. *Neural Networks: A Comprehensive Foundation*, Chap. 7, pp. 351–387. Prentice Hall PTR, 2. Ed., 1999.

[Heis 03] B. Heisele, T. Serre, S. Prentice, and T. Poggio. “Hierarchical classification and feature reduction for fast face detection with support vector machines”. *Pattern Recognition*, Vol. 36, No. 9, pp. 2007–2017, Sept. 2003.

[Jage 09] F. Jäger and J. Hornegger. “Nonrigid Registration of Joint Histograms for Intensity Standardization in Magnetic Resonance Imaging”. *Transactions on Medical Imaging*, Vol. 28, No. 1, pp. 137–150, Jan. 2009.

[Jord 94] M. Jordan and R. Jacobs. “Hierarchical mixtures of experts and the EM algorithm”. *Neural Computation*, Vol. 6, No. 2, pp. 181–214, March 1994.

[Kale 06] W. A. Kalender. *Computertomographie*. Publicis Corporate Publishing, Germany, 2006.

[Kawa 11] M. Kawakita, R. Izumi, J. Takeuchi, Y. Hu, T. Takamori, and H. Kameyama. “Acceleration technique for boosting classification and its application to face detection”. In: *Proc. ACML*, pp. 335–349, Taoyuan (Taiwan), Nov. 2011.

[Kear 89] M. Kearns and L. G. Valiant. “Cryptographic limitations on learning Boolean formulae and finite automata”. In: *Proc. ACM Symposium on Theory of Computing*, pp. 433–444, Seattle, WA (USA), May 1989.

[Keer 06] S. Keerthi, O. Chapelle, and D. DeCoste. “Building support vector machines with reduced classifier complexity”. *Journal of Machine Learning Research*, Vol. 7, pp. 1493–1515, July 2006.

[Lagh 05] A. Laghi, I. Sansoni, M. Celestre, P. Paolantonio, and R. Passariello. *Focal Liver Lesions*, Chap. Computed Tomography. Springer, Berlin (Germany), 2005.

[Laye 08] G. Layer and U. Gallkowski. “Lebertumoren”. In: G. Layer, G. Kaick, and S. Delorme, Eds., *Radiologische Diagnostik in der Onkologie*, pp. 87–119, Springer, Dec. 2008.

[Li 06a] Y. Li, S. Hara, and K. Shimura. “A Machine Learning Approach for Locating Boundaries of Liver Tumors in CT Images”. In: *Proc. ICPR*, pp. 400–403, IEEE, Hong Kong (China), Aug. 2006.

[Li 06b] Y. Li, W. Zhang, and C. Lin. “Simplify support vector machines by iterative learning”. *Neural Information Processing: Letters and Reviews*, Vol. 10, No. 1, pp. 11–17, Jan. 2006.

[Li 07] Q. Li, L. Jiao, and Y. Hao. “Adaptive simplification of solution for support vector machine”. *Pattern Recognition*, Vol. 40, No. 3, pp. 972–980, March 2007.

[Lien 02] R. Lienhart and J. Maydt. “An extended set of haar-like features for rapid object detection”. In: *Proc. ICIP*, IEEE, Rochester, NY (USA), Sept. 2002.

[Ling 08] H. Ling, S. K. Zhou, Y. Zheng, B. Georgescu, M. Sühling, and D. Comaniciu. “Hierarchical, learning-based automatic liver segmentation”. In: *Proc. CVPR*, pp. 1–8, IEEE, Anchorage, AK (USA), June 2008.

[Lotj 10] J. M. P. Lötjönen, R. Wolz, J. R. Koikkalainen, L. Thurfjell, G. Waldemar, H. Soininen, and D. Rueckert. “Fast and robust multi-atlas segmentation of brain magnetic resonance images”. *Neuroimage*, Vol. 49, No. 3, pp. 2352 – 2365, Feb. 2010.

[Mahm 09] A. Mahmood and S. Khan. “Early terminating algorithms for Adaboost based detectors”. In: *Proc. ICIP*, pp. 1209–1212, Cairo (Egypt), Nov. 2009.

[Marg 97] D. D. Margineantu and T. G. Dietterich. “Pruning Adaptive Boosting”. In: *Proc. ICML*, pp. 211–218, Nashville, TN (USA), July 1997.

[Maso 00] L. Mason, P. L. Bartlett, and J. Baxter. “Improved Generalization through Explicit Optimization of Margins”. *Machine Learning*, Vol. 38, No. 3, pp. 243–255, March 2000.

[Maso 99] L. Mason, J. Baxter, P. L. Bartlett, and M. R. Frean. “Boosting Algorithms as Gradient Descent”. In: S. A. Solla, T. K. Leen, and K.-R. Müller, Eds., *Proc. NIPS*, pp. 512–518, MIT Press, Denver, CO (USA), Nov. 1999.

[Mass 08] L. Massoptier and S. Casciaro. “A new fully automatic and robust algorithm for fast segmentation of liver tissue and tumors from CT scans”. *European Radiology*, Vol. 18, No. 8, pp. 1658–1665, Aug. 2008.

[Meas 08] D. Mease and A. Wyner. “Evidence contrary to the statistical view of boosting”. *Journal of Machine Learning Research*, Vol. 9, pp. 131–156, Feb. 2008.

[Mili 09] A. Militzer and F. Vega-Higuera. “Probabilistic boosting trees for automatic bone removal from CT angiography images”. In: *Proc. SPIE: Medical Imaging*, pp. 725946–1 – 725946–8, Orlando, FL (USA), Feb. 2009.

[Mili 10] A. Militzer, T. Hager, F. Jäger, C. Tietjen, and J. Hornegger. “Automatic detection and segmentation of focal liver lesions in contrast enhanced CT images”. In: *Proc. ICPR*, pp. 2524–2527, IEEE, Istanbul (Turkey), Aug. 2010.

[Mili 11] A. Militzer, C. Tietjen, and J. Hornegger. “A cost constrained boosting algorithm for fast lesion detection and segmentation”. In: *Proc. SPIE: Medical Imaging*, pp. 79631B–1 – 79631B–6, Orlando, FL (USA), Feb. 2011.

[Mili 13a] A. Militzer, C. Tietjen, and J. Hornegger. “A Cost Constrained Boosting Algorithm for Fast Object Detection”. Tech. Rep. CS-2013-04, Friedrich-Alexander-Universität Erlangen-Nürnberg, Dept. of Computer Science, Oct. 2013.

[Mili 13b] A. Militzer, C. Tietjen, and J. Hornegger. “Learning a Prior Model for Automatic Liver Lesion Segmentation in Follow-up CT Images”. Tech. Rep. CS-2013-03, Friedrich-Alexander-Universität Erlangen-Nürnberg, Dept. of Computer Science, Oct. 2013.

[Molt 09] J. H. Moltz, M. Schwier, and H. O. Peitgen. “A general framework for automatic detection of matching lesions in follow-up CT”. In: *Proc. ISBI*, pp. 843–846, IEEE, Boston, MA (USA), Apr. 2009.

[Morr 08] J. H. Morra, Z. Tu, L. G. Apostolova, A. E. Green, A. W. Toga, and P. M. Thompson. “Automatic Subcortical Segmentation Using a Contextual Model”. In: *Proc. MICCAI*, pp. 194–201, Springer, New York, NY (USA), Sept. 2008.

[Nain 04] D. Nain, A. Yezzi, and G. Turk. “Vessel segmentation using a shape driven flow”. In: *Proc. MICCAI*, pp. 51–59, Springer, Saint-Malo (France), Sept. 2004.

[Nguy 06] D. Nguyen and T. Ho. “A bottom-up method for simplifying support vector solutions”. *Transactions on Neural Networks*, Vol. 17, No. 3, pp. 792–796, May 2006.

[OGra 00] J. G. O'Grady. "Treatment Options for Other Hepatic Malignancies". *Liver Transplantation*, Vol. 6, No. 6, Suppl. 2, pp. S23–S29, Nov. 2000.

[Osun 98] E. E. Osuna and F. Girosi. "Reducing the run-time complexity in support vector machines". In: B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds., *Advances in Kernel Methods*, pp. 271–283, MIT Press, Cambridge, MA (USA), Dec. 1998.

[Paja 06] G. Pajares. "A Hopfield Neural Network for Image Change Detection". *Transactions on Neural Networks*, Vol. 17, No. 5, pp. 1250–1264, Sept. 2006.

[Paja 09] G. Pajares, J. Ruz, and J. de la Cruz. "Image change detection from difference image through deterministic simulated annealing". *Pattern Analysis and Applications*, Vol. 12, No. 2, pp. 137–150, June 2009.

[Papa 00] C. P. Papageorgiou and T. Poggio. "A Trainable System for Object Detection". *International Journal of Computer Vision*, Vol. 38, No. 1, pp. 15–33, June 2000.

[Papa 98] C. P. Papageorgiou, M. Oren, and T. Poggio. "A general framework for object detection". In: *Proc. ICCV*, pp. 555–562, IEEE, Bombay (India), Jan. 1998.

[Parr 03] E. Parrado-Hernández, I. Mora-Jiménez, J. Arenas-Garcıa, A. Figueiras-Vidal, and A. Navia-Vázquez. "Growing support vector classifiers with controlled complexity". *Pattern Recognition*, Vol. 36, No. 7, pp. 1479–1488, July 2003.

[Pesc 08] D. Pescia, N. Paragios, and S. Chemouny. "Automatic detection of liver tumors". In: *Proc. ISBI*, pp. 672–675, IEEE, Paris (France), May 2008.

[Putz 04] R. Putz and R. Pabst, Eds. *Sobotta: Atlas der Anatomie des Menschen*. Elsevier, 21 Ed., 2004.

[Radk 05] R. Radke, S. Andra, O. Al-Kofahi, and B. Roysam. "Image change detection algorithms: A systematic survey". *Transactions on Image Processing*, Vol. 14, No. 3, pp. 294–307, March 2005.

[Rats 01] G. Rätsch, T. Onoda, and K.-R. Müller. "Soft Margins for AdaBoost". *Machine Learning*, Vol. 42, No. 3, pp. 287–320, March 2001.

[Rey 02] D. Rey, G. Subsol, H. Delingette, and N. Ayache. "Automatic detection and segmentation of evolving processes in 3D medical images: Application to multiple sclerosis". *Medical Image Analysis*, Vol. 6, No. 2, pp. 163–179, June 2002.

[Ront 95] W. C. Röntgen. "Über eine neue Art von Strahlen. Vorläufige Mitteilung". In: *Sitzungsberichte der Physikalisch-medicinischen Gesellschaft Würzburg*, pp. 137–147, Würzburg (Germany), Dec. 1895.

[Rous 02] M. Rousson and N. Paragios. "Shape priors for level set representations". In: *Proc. ECCV*, pp. 416–418, Springer, Copenhagen (Denmark), May 2002.

[Rudi 04] C. Rudin, I. Daubechies, and R. E. Schapire. "The Dynamics of AdaBoost: Cyclic Behavior and Convergence of Margins". *Journal of Machine Learning Research*, Vol. 5, pp. 1557–1595, Dec. 2004.

[Rudi 13] C. Rudin, B. Letham, and D. Madigan. "Learning Theory Analysis for Association Rules and Sequential Event Prediction". *Journal of Machine Learning Research*, Vol. 14, pp. 3441 – 3492, Nov. 2013.

[Saha 11] B. N. Saha, N. Ray, R. Greiner, A. Murtha, and H. Zhang. “Quick detection of brain tumors and edemas: A bounding box method using symmetry.”. *Computerized Medical Imaging and Graphics*, Vol. 36, No. 2, pp. 1–13, March 2011.

[Sahb 06] H. Sahbi and D. Geman. “A hierarchy of support vector machines for pattern detection”. *Journal of Machine Learning Research*, Vol. 7, No. 2, pp. 2087–2123, Oct. 2006.

[Sato 97] Y. Sato, S. Nakajima, H. Atsumi, T. Koller, G. Gerig, S. Yoshida, and R. Kikinis. “3D multi-scale line filter for segmentation and visualization of curvilinear structures in medical images”. In: *Proc. CVRMed-MRCAS*, pp. 213–222, Springer, Grenoble (France), March 1997.

[Scha 90] R. E. Schapire. “The Strength of Weak Learnability”. *Machine Learning*, Vol. 5, No. 2, pp. 197–227, June 1990.

[Scha 98] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. “Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods”. *The Annals of Statistics*, Vol. 26, No. 5, pp. 1651–1686, Oct. 1998.

[Scha 99] R. Schapire and Y. Singer. “Improved Boosting Algorithms Using Confidence-rated Predictions”. *Machine Learning*, Vol. 37, No. 3, pp. 297–336, Dec. 1999.

[Shim 05] A. Shimizu, T. Kawamura, and H. Kobatake. “Proposal of computer-aided detection system for three dimensional CT images of liver cancer”. In: *Proc. CARS*, pp. 1157–1162, Elsevier, Berlin (Germany), May 2005.

[Shim 08] A. Shimizu, T. Narihira, D. Furukawa, H. Kobatake, S. Nawano, and K. Shinozaki. “Ensemble segmentation using AdaBoost with application to liver lesion extraction from a CT volume”. *MIDAS Journal: 3D Segmentation in the Clinic: A Grand Challenge. Liver Tumor Segmentation. 2008 MICCAI Workshop*, July 2008.

[Smee 10] D. Smeets, D. Loeckx, B. Stijnen, B. d. Dobbelaer, D. Vandermeulen, and P. Suetens. “Semi-automatic level set segmentation of liver tumors combining a spiral-scanning technique with supervised fuzzy pixel classification”. *Medical Image Analysis*, Vol. 14, No. 1, pp. 13–20, Feb. 2010.

[Tamo 00] C. Tamon and J. Xiang. “On the Boosting Pruning Problem”. In: *Proc. ECML*, pp. 404–412, Barcelona, (Spain), May 2000.

[Tu 05] Z. Tu. “Probabilistic Boosting-Tree: Learning Discriminative Models for Classification, Recognition, and Clustering”. In: *Proc. ICCV*, pp. 1589–1596, IEEE, Beijing (China), Oct. 2005.

[Tu 06] Z. Tu, X. S. Zhou, L. Bogoni, A. Barbu, and D. Comaniciu. “Probabilistic 3D Polyp Detection in CT Images: The Role of Sample Alignment”. In: *Proc. CVPR*, pp. 1544–1551, IEEE, New York, NY (USA), June 2006.

[Utsu 09] Y. Utsumi, Y. Matsumoto, and Y. Iwai. “An efficient branch and bound method for face recognition”. In: *Proc. ICSIPA*, pp. 156–161, IEEE, Kuala Lumpur (Malaysia), Nov 2009.

[Vali 84] L. Valiant. “A theory of the learnable”. *Communications of the ACM*, Vol. 27, No. 11, pp. 1134–1142, Nov. 1984.

[Vapn 95] V. Vapnik. *The nature of statistical learning theory*. Springer, New York, NY (USA), 1995.

[Viol 01] P. Viola and M. Jones. “Rapid Object Detection using a Boosted Cascade of Simple Features”. In: *Proc. CVPR*, pp. 511–518, IEEE, Kauai, HI (USA), Dec. 2001.

[Wels 08] M. Wels, G. Carneiro, A. Aplas, M. Huber, J. Hornegger, and D. Comaniciu. “A discriminative model-constrained graph cuts approach to fully automated pediatric brain tumor segmentation in 3-D MRI”. In: *Proc. MICCAI*, pp. 67–75, Springer, New York, NY (USA), Sept. 2008.

[Xu 12] Z. Xu, O. Chapelle, and K. Q. Weinberger. “The Greedy Miser: Learning under Test-time Budgets”. In: *Proc. ICML*, pp. 1175–1182, Edinburgh, Scotland (UK), June 2012.

[Xu 13] Z. Xu, M. Kusner, M. Chen, and K. Q. Weinberger. “Cost-Sensitive Tree of Classifiers”. In: *Proc. ICML*, pp. 133–141, Atlanta, GA (USA), June 2013.

[Zhan 06] L. Zhang, E. A. Hoffman, and J. M. Reinhardt. “Atlas-driven lung lobe segmentation in volumetric X-ray CT images”. *Transactions on Medical Imaging*, Vol. 25, No. 1, pp. 1–16, Jan. 2006.

[Zhan 13] H. Zhang, K. Zhao, Y.-Z. Song, and J. Guo. “Text extraction from natural scene image: A survey”. *Neurocomputing*, Vol. 122, No. 0, pp. 310 – 323, 2013. Advances in cognitive and ubiquitous computing Selected papers from the Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS-2012).