Examsarbete

Semi-Supervised Learning for Object Detection

Examensarbete utfört i Reglerteknik
vid Tekniska högskolan vid Linköpings universitet
av

Mikael Rosell

LiTH-ISY-EX--14/4817--SE

Linköping 2015
Semi-Supervised Learning for Object Detection

Examensarbete utfört i Reglerteknik
vid Tekniska högskolan vid Linköpings universitet
av

Mikael Rosell

LiTH-ISY-EX--14/4817--SE

Handledare: Johan Dahlin
ISY, Linköpings universitet
Per Cronvall
Autoliv Electronics

Examinator: Martin Enqvist
ISY, Linköpings universitet

Linköping, 1 januari 2015
Many automotive safety applications in modern cars make use of cameras and object detection to analyze the surrounding environment. Pedestrians, animals and other vehicles can be detected and safety actions can be taken before dangerous situations arise. To detect occurrences of the different objects, these systems are traditionally trained to learn a classification model using a set of images that carry labels corresponding to their content. To obtain high performance with a variety of object appearances, the required amount of data is very large. Acquiring unlabeled images is easy, while the manual work of labeling is both time-consuming and costly. Semi-supervised learning refers to methods that utilize both labeled and unlabeled data, a situation that is highly desirable if it can lead to improved accuracy and at the same time alleviate the demand of labeled data. This has been an active area of research in the last few decades, but few studies have investigated the performance of these algorithms in larger systems.

In this thesis, we investigate if and how semi-supervised learning can be used in a large-scale pedestrian detection system. With the area of application being automotive safety, where real-time performance is of high importance, the work is focused around boosting classifiers. Results are presented on a few publicly available UCI data sets and on a large data set for pedestrian detection captured in real-life traffic situations. By evaluating the algorithms on the pedestrian data set, we add the complexity of data set size, a large variety of object appearances and high input dimension.

It is possible to find situations in low dimensions where an additional set of unlabeled data can be used successfully to improve a classification model, but the results show that it is hard to efficiently utilize semi-supervised learning in large-scale object detection systems. The results are hard to scale to large data sets of higher dimensions as pair-wise computations are of high complexity and proper similarity measures are hard to find.
Sammanfattning

I många moderna bilar finns säkerhetssystem som använder sig av kameror och objektigenkänning för att analysera omgivningen. Genom att identifiera och lokalisera fotgängare, djur och andra bilar kan farliga situationer förhindras innan de uppstår.

För att kunna upptäcka olika typer av objekt, tas en klassificeringsmodell fram med hjälp av bilder som är märkta baserade på innehållet i dem. För att uppnå bra prestanda krävs många bilder. Att samla in omärkt data är lätt, medan att märka data är ett manuellt arbete som är både tidskrävande och kostsamt. Semi-supervised learning är en typ av metoder som använder sig av både märkt och omärkt data för att ta fram en klassificeringsmodell, något som är eftertraktat om det kan leda till förbättrad prestanda och sänka behovet av märkt data. Under de senaste årtiondena har det forskats mycket inom detta område, men det är inte många som har undersökt prestandan hos dessa algoritmer i större system.

I detta examensarbete undersöks om semi-supervised learning kan användas i ett storskaligt system för detektion av fotgängare. Då syftet med applikationen är säkerhetssystem för bilar, där realtidsprestanda är av högsta prioritet, fokuseras arbetet kring boosting-klassificerare. Resultat presenteras för offentlig data från UCI samt bilddata från Autoliv ämnat för detektion av fotgängare. Genom att utvärdera algoritmerna på fotgängar-data tillkommer komplexitet av mängden bilder, den stora variationen i utseende bland objekten samt den höga dimensionen på datat som används.

Resultaten visar att det är svårt att effektivt använda semi-supervised learning i storskaliga system för objektigenkänning. Det går att hitta småskaliga exempel där ett kompletterande dataset med omärkt data kan användas för att förbättra en klassificeringsmodell, men tyvärr skalar inte dessa resultat till större mängder data med högre dimension, bland annat då parvisa beräkningar är mycket beräkningskrävande och det är svårt att hitta bra likhetsmått.
Abstract

Many automotive safety applications in modern cars make use of cameras and object detection to analyze the surrounding environment. Pedestrians, animals and other vehicles can be detected and safety actions can be taken before dangerous situations arise.

To detect occurrences of the different objects, these systems are traditionally trained to learn a classification model using a set of images that carry labels corresponding to their content. To obtain high performance with a variety of object appearances, the required amount of data is very large. Acquiring unlabeled images is easy, while the manual work of labeling is both time-consuming and costly. Semi-supervised learning refers to methods that utilize both labeled and unlabeled data, a situation that is highly desirable if it can lead to improved accuracy and at the same time alleviate the demand of labeled data. This has been an active area of research in the last few decades, but few studies have investigated the performance of these algorithms in larger systems.

In this thesis, we investigate if and how semi-supervised learning can be used in a large-scale pedestrian detection system. With the area of application being automotive safety, where real-time performance is of high importance, the work is focused around boosting classifiers. Results are presented on a few publicly available UCI data sets and on a large data set for pedestrian detection captured in real-life traffic situations. By evaluating the algorithms on the pedestrian data set, we add the complexity of data set size, a large variety of object appearances and high input dimension.

It is possible to find situations in low dimensions where an additional set of unlabeled data can be used successfully to improve a classification model, but the results show that it is hard to efficiently utilize semi-supervised learning in large-scale object detection systems. The results are hard to scale to large data sets of higher dimensions as pair-wise computations are of high complexity and proper similarity measures are hard to find.
Acknowledgments

First of all, I would like to thank Autoliv Electronics for giving me the opportunity and resources needed to carry out this Master’s thesis. A special thanks goes to my supervisor at Autoliv Electronics, Per Cronvall, for providing me with invaluable advice and support in both theoretical and practical questions throughout the work.

I would like to direct another great thanks to my supervisor at Linköping University, Ph.D. student Johan Dahlin, for always supplying fast and extensive feedback on both the work and the report. I would also like to thank my examiner Martin Enqvist, Associate Professor in Automatic Control at ISY, Linköping University, for providing me with valuable comments and insights.

Finally, I would like to thank my family and friends for all the support they have given me throughout the years.

Linköping, 2015
Mikael Rosell
# Contents

Notation xi

1 Introduction 1
   1.1 Related work .............................................. 1
   1.2 Problem formulation .................................... 3
   1.3 Contribution and results ................................ 3
   1.4 Limitations .................................................. 4
   1.5 Autoliv ......................................................... 4
   1.6 Outline ......................................................... 5

2 Object detection with boosting classifiers 7
   2.1 Object localization ...................................... 8
   2.2 Object categorization .................................... 8
      2.2.1 Supervised boosting .................................. 9
   2.3 Implementing a supervised boosting classifier for pedestrian detection ........................................ 19
      2.3.1 Feature extraction ..................................... 19
      2.3.2 Choice of weak learners ............................... 20
      2.3.3 Cascade of boosting classifiers ....................... 21

3 Semi-supervised learning 23
   3.1 Extracting information from unlabeled samples .... 25
      3.1.1 Reducing complexity of pair-wise computations 25
   3.2 Semi-supervised boosting ................................ 26
      3.2.1 SemiBoost ............................................. 27
      3.2.2 RegBoost .............................................. 30
   3.3 Self-learning ............................................... 35
      3.3.1 Self-learning with a boosting classifier ........... 36

4 Experimental results and discussion 41
   4.1 UCI benchmark data sets ................................. 41
   4.2 Pedestrian detection ...................................... 43
      4.2.1 RegBoost case study ................................. 48
Notation

Sets

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathbb{R} )</td>
<td>The set of real numbers</td>
</tr>
<tr>
<td>( \mathbb{Z} )</td>
<td>The set of integers</td>
</tr>
<tr>
<td>( \mathbb{L} )</td>
<td>The set of labeled samples</td>
</tr>
<tr>
<td>( \mathbb{U} )</td>
<td>The set of unlabeled samples</td>
</tr>
<tr>
<td>( \mathbb{S} )</td>
<td>The union of the labeled and the unlabeled samples</td>
</tr>
</tbody>
</table>

Variables

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_i )</td>
<td>Sample ( i ), represented by its feature vector</td>
</tr>
<tr>
<td>( y_i \in {-1,+1} )</td>
<td>Class label of labeled sample ( i \in \mathbb{L} )</td>
</tr>
<tr>
<td>( \tilde{y}_i \in {-1,+1} )</td>
<td>Pseudo-label assigned to unlabeled sample ( i \in \mathbb{U} )</td>
</tr>
<tr>
<td>( \hat{y}_i \in {-1,+1} )</td>
<td>Collective notation of true labels and pseudo-labels,( \hat{y}_i = y_i ) for ( i \in \mathbb{L} ) and ( \hat{y}_i = \tilde{y}_i ) for ( i \in \mathbb{U} )</td>
</tr>
<tr>
<td>( T \in {1,2,3,\ldots} )</td>
<td>Number of weak learners in the final classifier</td>
</tr>
<tr>
<td>( f_t(x) )</td>
<td>The ( t )-th weak learner, ( t = 1, \ldots, T )</td>
</tr>
<tr>
<td>( \alpha_t )</td>
<td>Weight corresponding to the ( t )-th weak learner</td>
</tr>
<tr>
<td>( F(x) )</td>
<td>The final classifier</td>
</tr>
<tr>
<td>( S_{i,j} \in \mathbb{R}^+ )</td>
<td>Similarity measure for two points ( x_i ) and ( x_j )</td>
</tr>
</tbody>
</table>

Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSL</td>
<td>Semi-supervised learning</td>
</tr>
<tr>
<td>HOG</td>
<td>Histogram of oriented gradients</td>
</tr>
</tbody>
</table>
In recent years, the importance of active safety systems has increased in automotive safety applications. Active safety refers to systems that can act prior to the occurrence of an accident. The majority of these systems make use of object detection in various forms e.g. pedestrian, animal, and vehicle detection.

The most successful image based object detection systems for automotive applications all make use of machine learning. Typically, a classifier is used to distinguish image regions containing an object of interest from regions that do not. It is common to use supervised learning to train a classifier to do this, meaning that labeled samples of both the fore- and background classes are used to learn a classification model. In contrast to this method, unsupervised learning corresponds to classifiers that are trained on samples without given labels.

To cover the great variety of possible appearances of object classes, the resulting amount of data required can be very large. Labeling is typically a manual process that is both very time-consuming and costly. Methods utilizing both labeled and unlabeled data are referred to as semi-supervised learning, i.e., being situated somewhere between the traditional disciplines of supervised and unsupervised learning, see Figure 1.1 for a visualization of the idea.

### 1.1 Related work

Object detection using machine learning has been an active field of research since the late 1950s (Nilsson, 2010). With ever increasing computational resources and new machine learning techniques to efficiently utilize those, object detection reached a point in the late 1990s where real-time performance was feasible. One popular classification technique is boosting, where a strong classifier is formed as a combination of simple classifiers called weak learners. Freund and Schapire (1997) introduced AdaBoost, which even today is one of the most popular boosting
Introduction

Figure 1.1: The idea of semi-supervised learning. A small set \( L \) of labeled samples are complemented with a larger set \( U \) of unlabeled samples in order to obtain a sufficient training set \( S \) to utilize in the training process of a classifier.

classifiers. Viola and Jones (2001) has highly influenced modern object detection systems, enabling applications to run in real-time. Their integral image is an image representation that allows for very fast feature evaluation. They also propose to search for the most significant features in a data set and to use a cascade of boosting classifiers, which greatly decreases the computational demand and the time needed for classification.

AdaBoost and many other successful classifiers are based on supervised learning, i.e., require a labeled set of samples for training. If labeling the data is problematic or expensive, an alternative approach can be semi-supervised learning (SSL), where an additional set of unlabeled samples is utilized together with the available set of labeled samples in the training process. The initial work in SSL is contributed to Scudder (1965) for his work on self-learning, where a supervised classifier is used to repeatedly label and add unlabeled samples to the training set.

In a literature survey of semi-supervised learning, Zhu (2008) concludes that it is possible to improve a classifier by utilizing unlabeled samples in the training process. Important aspects are the design of models, features, kernels and similarity functions. Incorrect assumptions or badly structured models can lead to degradation in classifier performance. Chapelle et al. (2006) formulate three formal assumptions for SSL: The semi-supervised smoothness assumption says that if two points in a high-density region are close, then so should their class labels, according to some measure. The cluster assumption says that points in the same cluster are likely to be of the same class. The manifold assumption says that high-dimensional data lie roughly on a low-dimensional manifold.

In this thesis, we focus on boosting classifiers since they are suitable for applications executing in real-time (Viola and Jones, 2001). Bennett et al. (2002) received first price at the NIPS’2001 workshop competition Unlabeled Data for Supervised Learning with their algorithm ASSEMBLE. AdaBoost, an AdaBoost specialization of their general SSL framework. In ASSEMBLE, the information in the unlabeled samples is extracted based on the confidence of the classifier alone. In order to improve the performance of SSL algorithms, Rosenberg et al. (2005)
propose to use an additional similarity measure that is not correlated with the confidence of the classifier. They state that the distribution of the labeled data at any particular iteration may not match the actual distribution of the data and therefore the confidence metric may be misleading.

Mallapragada et al. (2009) introduce SemiBoost, a framework for SSL to use with any boosting algorithm. SemiBoost works on both the cluster and the manifold assumptions. Leistner et al. (2008) propose an algorithm based on the SemiBoost framework (Mallapragada et al., 2007), where similarity between samples is learned using an a-priori classifier. Chen and Wang (2011) introduce an algorithm called RegBoost that makes use of all three SSL assumptions described above. Both SemiBoost and RegBoost utilizes a combination of the current classifier confidence and an independent similarity measure to handle the unlabeled samples.

Blum and Mitchell (1998) propose a framework for SSL called co-training where two classifiers are trained side-by-side. In an iterative process, unlabeled samples that are classified confidently by one classifier are added to the set of labeled samples for the other classifier. To obtain formal results they assume that the two feature sets used by respective classifier are statistically independent, which unfortunately is not the case in many real-world situations. Levin et al. (2003) further investigate co-training based on the relationship between prediction confidence and prediction margins. They show that even two closely related classifiers can be successfully co-trained, and hence conclude that co-training can be used in many real-world situations.

Belkin et al. (2006) introduce LapSVM, an SSL extension of Support Vector Machines (Cortes and Vapnik, 1995). Their work is inspired by spectral graph theory and they use the graph Laplacian as similarity measure. Habrard et al. (2013) present interesting results in the area of domain adaptation, which under some assumptions can be generalized to SSL.

1.2 Problem formulation

The main aim of this thesis is to investigate and evaluate SSL algorithms used to train a cascade of boosted classifiers for object detection. The objective of the SSL algorithms is to improve the accuracy of the classifier while alleviating the demand of labeled data, which requires tedious and time-consuming work.

The system should be able to handle large amounts of training data (some hundred thousand samples) in reasonable time (a few hours). One of the issues considered in the thesis is how to calculate similarity between samples in large data sets.

1.3 Contribution and results

Semi-supervised learning has been an active research area during the last few decades, but most of the results have been presented on relatively small data sets
with low input dimensions. In this thesis, we investigate two approaches to semi-supervised learning, *semi-supervised boosting* and *self-learning*. We implement and evaluate three SSL algorithms in a large-scale pedestrian detection system. The final results are obtained using 100,000 samples of 90×45 pixels images captured in real-life traffic situations. The results are presented as a comparative study of the implemented SSL algorithms and current state-of-the-art supervised algorithms trained under equivalent conditions. Performance results are also reported on a few UCI benchmark data sets (Bache and Lichman, 2013).

It is easy to illustrate the benefits and performance improvements of semi-supervised learning algorithms in low-dimensional examples. Unfortunately, these promising results are hard to generalize to larger data sets of higher dimension. In problems of higher dimension, it is harder to measure similarity among samples, which is important in order to obtain good results. It is also difficult to find parameters that work well in large data sets.

### 1.4 Limitations

The implemented algorithms are evaluated on the two-class problem of detecting pedestrians. They can be extended to object detection of an arbitrary number of classes by assuming an *one-vs-all* structure. This generalization is left as future work.

Object detection of a specific class is often a highly skewed problem with many more samples of the background class than of the foreground class. In images captured in real-life traffic situations, there are naturally a higher amount of sub-windows not containing objects of a specific class, e.g., pedestrians, than sub-windows containing objects of that class. In order to obtain a balance between samples from the fore- and background classes, we generate data sets for training, validation and testing using an existing cascade of classifiers. By drawing samples from a late stage in the cascade, many easy to classify sub-windows of the background class have been rejected, and the ratio between fore- and background objects is nearly balanced.

The main purpose of this thesis is to investigate and compare SSL algorithms suitable for boosting classifiers. Algorithms based on other types of classifiers are not investigated. Due to time limitations, the training of a complete classifier cascade is not evaluated and some reasoning and results are based on subsets of the entire data set provided by Autoliv.

### 1.5 Autoliv

Autoliv was founded in Vårgårda in 1953 by the two brothers Lennart and Stig Lindblad as a dealership and repair shop for motor vehicles called Lindblads Autoservice. In 1956, they started manufacturing their first safety product, a twopoint seatbelt, and in 1968 the company changed their name to Autoliv AB which stands for AUTOservice Lindblad In Vårgårda. Autoliv AB and Morton ASP merged in 1997 and formed the company Autoliv Inc. Today, Autoliv is one of the
the world leaders in automotive safety systems, developing and manufacturing safety systems for all major automotive manufacturers and saving over 30,000 lives every year (Autoliv, 2015).

Autoliv Electronics, a division of Autoliv, develops vision, night vision and radar systems as well as central electronic control units and satellite sensors. They have about 1,500 employees primarily in France, Sweden, US and China. At Autoliv Electronics in Linköping, there are about 200 people working with camera based active safety systems. These systems incorporate features such as detection of pedestrians, animals and cars as well as lane departure warning, speed sign recognition and vehicle collision mitigation.

The Autoliv Inc. group has approximately 58,000 employees in 29 countries, with about 5,000 people working within research and development.

1.6 Outline

In Chapter 2, we give a historical background to object detection. The reader is introduced to supervised learning and two popular boosting classifiers. A simple example is presented to illustrate the classification problem. This example is revisited throughout the thesis to give an intuition to the other implemented algorithms.

Chapter 3 covers both theory and algorithms for semi-supervised learning. We introduce the SSL settings, discuss general issues and outline two conceptually different approaches, semi-supervised boosting and self-learning. Three algorithms are described and implemented for evaluation in later chapters.

The results are presented and discussed in Chapter 4. The performance of the implemented SSL algorithms is presented and compared to supervised algorithms trained under equivalent settings. The thesis is summarized in Chapter 5 with concluding remarks and ideas for future work in the area.
In this chapter, we introduce the reader to object detection using boosting classifiers. This work is based on the definition from Lehmann et al. (2011):

Object detection is the problem of joint localization and categorization of objects in images. It involves two tasks: learning an accurate object model and the actual search for objects, i.e., applying the model to new images.

The two tasks of learning an object model and applying it to new data are referred to as training and evaluation respectively. In this thesis, we focus on the training procedure and use the evaluation process to measure the success of the implemented algorithms. The two central issues localization and categorization are discussed in Sections 2.1 and 2.2.

To categorize samples we use a boosting classifier. In the context of this thesis, the classifier is trained to model the appearance of pedestrians in images captured by a camera mounted in a vehicle. The purpose of the classifier is to accurately determine if any pedestrians are present in new images presented to the system.

Historically, most boosting algorithms are based on supervised learning, i.e., require a labeled set of samples for training. We define \( \mathcal{L} \) as the set of labeled samples

\[
\mathcal{L} = \{(x_1, y_1), \ldots, (x_{|\mathcal{L}|}, y_{|\mathcal{L}|})\},
\]

where \( y_i \) denotes the class label of sample \( x_i \), \( i = 1, \ldots, |\mathcal{L}| \). In this chapter, we base our work on this supervised setting and present two well-known algorithms, AdaBoost (Freund and Schapire, 1997) and LogitBoost (Friedman et al., 2000). These algorithms act as the starting points when we in Chapter 3 extend our work to include unlabeled samples in the training procedure.

First, we present object detection in general, and later, in Section 2.3, discuss how to adapt the system to pedestrian detection.
2.1 Object localization

Objects in an image may appear at different locations and in different scales. A common solution to this problem is the *sliding window* approach (Viola and Jones, 2001). All images are exhaustively scanned in both location and scale using a rectangular frame with proportions matching the object of interest; hence a large set of candidate images called windows are generated. The classifier is applied to all these windows in order to find objects of interest. If an object is detected, its location and scale can be extracted from the position and size of the window.

The search can be accelerated using a coarse-to-fine evaluation strategy. Starting with a coarse grid of windows, some areas can be excluded for further search if there is no indication of an object according to the classifier. Computations are reduced by evaluating only a subset of all possible windows of different locations and scales.

2.2 Object categorization

To represent objects of different types, we assign each class a numerical value, i.e., a label. A classifier is trained to output these labels when presented a sample from respective class. We separate between two different approaches to this prediction problem: *Classification* and *regression*.

The term *classification* is used when the classes are represented using a set of qualitative labels. *Binary classification* is a special case that arises when there are only two classes to separate. Object detection of one specific class can be considered binary classification, where the qualitative outputs are either *object detected* or *object not detected*. It is convenient to use the numerical labels \{-1, +1\} to represent these two classes. In this thesis, we mainly consider pedestrian detection, and use the notation:

\[
y_i = \begin{cases} 
  +1, & \text{pedestrian detected,} \\
  -1, & \text{no pedestrian detected,} 
\end{cases} \tag{2.2}
\]

where \(y_i\) denotes the class label of sample \(x_i\).

In contrast, when the goal of the classifier is to output a quantitative response, the task is called *regression*. Different methods have been developed for classification and regression, respectively. In this thesis, we approach pedestrian detection using both classification and regression algorithms. Using regression, the discrete labels (2.2) are obtained by thresholding the continuous response value, asserting that responses within specific ranges correspond to respective classes. With this brief motivation, we claim that binary classification can be solved using either classification algorithms or regression algorithms followed by appropriate thresholding. For a more detailed analysis of classification and regression, see, for example Hastie et al. (2009) and Murphy (2012).
2.2 Object categorization

2.2.1 Supervised boosting

In this thesis, we use boosting classifiers to categorize samples into either of the two classes (2.2). Here, we present the fundamental theory of boosting, explain the key behind its success and present two popular algorithms.

The boosting classifier is based on the assumption that a set of simple classifiers, called weak learners, together may form a stronger classifier whose accuracy is better than that of the best single weak learner. The strong learner is created by a linear combination of \( T \) weak learners \( f_t \) with coefficients \( \alpha_t \). Each weak learner is a classifier that outputs a predicted label for each sample \( x \). For binary classification, \( f_t(x) \in \{-1, +1\} \). The only requirement on the weak learners is that they correctly classify a sample with an accuracy slightly better than a random guess. The final classification can be seen as a majority vote of the weak learners given by

\[
F(x) = \sum_{t=1}^{T} \alpha_t f_t(x),
\]

(2.3)

where \( F(x) \) can be interpreted as the confidence in classifying sample \( x \) as belonging to a certain class. In the case of binary classification, a sample \( x \) is usually classified with the classification rule given by

\[
y_p = \text{sign} [F(x)] = \text{sign} \left[ \sum_{t=1}^{T} \alpha_t f_t(x) \right],
\]

(2.4)

where \( y_p \in \{-1, +1\} \) denotes the predicted class label of the sample \( x \), using the notation in (2.2). Here, \( x \) denotes a set of features representing the content of the image rather than the raw image data itself. We use this notation throughout the thesis, interchangeably discussing images and their respective feature representation. The choice of what features to use is problem dependent and not covered in this thesis. We recommend the interested reader to see, e.g., Viola and Jones (2001) or Dalal and Triggs (2005) for further details. In Section 2.3.1, we briefly discuss the latter method, which has proven very efficient in pedestrian detection tasks.

Most boosting algorithms are based on the supervised setting, i.e., a set of labeled samples is available for training. The boosting classifier is trained in an iterative process, where each weak learner is fitted to an adjusted version of the data set. Figure 2.1 presents an overview of the training procedure. A training set is formed by weighting or subsampling the available labeled data set in every iteration of the training loop. The training set reflects the influence of the samples, e.g., emphasizing samples that were misclassified by the previous weak learners. Each weak learner is trained separately and successively added to the strong classifier (2.3). The procedure is repeated until a predetermined number \( T \) of weak learners have been added to the strong classifier.

The strength of the boosting classifier lies in (2.3). Boosting fits an additive model with weak learners as basis functions. This basis expansion is a convenient way to handle non-linearity with linear theory. The weak learners are often non-linear transformations of the input. Since the strong classifier \( F(x) \) is linear in the
object detection with boosting classifiers

Supervised boosting

Labeled data ⇒ Training set ⇒ Supervised training of weak learner ⇒ Strong learner

Sampling/reweighting

Output final classifier

Figure 2.1: Overview of a supervised boosting training procedure. The classifier is trained in an iterative loop where weak learners are successively added to the strong learner.

weak learners, it can be handled with linear methods. The choice of weak learners is further discussed in Section 2.3.2.

Forward stagewise additive modeling

The success of boosting can be explained by linear models. Most real systems are non-linear in nature and approximations using linear models are often necessary to reduce complexity. Many non-linearities are hard to model from empirical data and models of lower order often generalize better, as stated by Occam’s razor (Encyclopædia Britannica Online, 2014):

Plurality should not be posited without necessity.

Models that are linear in the input, \( x = (x_1, x_2, \ldots, x_d) \), are commonly used and extensively discussed in the literature, see, for example Murphy (2012). Linear models can be extended to non-linear models by transforming the input \( x \) using the transform \( b(x; \gamma_t) : \mathbb{R}^d \mapsto \mathbb{R} \). The resulting linear basis expansion can be written as

\[
F(x) = \sum_{t=1}^{T} \beta_t b(x; \gamma_t),
\]

where \( \beta_t, t = 1, \ldots, T \) denote the expansion coefficients. \( b(x; \gamma_t) \) denotes non-linear functions of the input \( x \), which are characterized by a set of parameters \( \gamma_t \). Once the basis functions \( b(x; \gamma_t) \) are determined, the model \( F(x) \) is linear in these new variables. Some widely used basis functions are:

- \( b(x; \gamma_t) \equiv x \), which reproduces the original linear model.
- \( b(x; \gamma_t) = I(x, \gamma_t) \in \{0, 1\} \), an indicator function that divides the input space in regions parameterized by \( \gamma_t \), resulting in a piece-wise constant model.
- \( b(x; \gamma_t) = \psi(x, \gamma_t) \), a wavelet function, which is popular in signal processing, see Mallat (1989). The location and scale shifts of the mother wavelet are parameterized by \( \gamma_t \).
- \( b(x; \gamma_t) = f(x, \gamma_t) \), a decision tree, which is what we use as weak learners in this thesis, see Section 2.3.2. \( \gamma_t \) parameterizes the split variables and split points at the internal nodes, and the predictions at the terminal nodes.
### Algorithm 2.1 Forward stagewise additive modeling

**Inputs:** Set of labeled data $\mathcal{L} = \{(x_1, y_1), \ldots, (x_{|\mathcal{L}|}, y_{|\mathcal{L}|})\}$, loss function $\mathcal{L}(y, \mathcal{F}(x))$ and basis function $b(x; \gamma)$.

**Output:** Model $\mathcal{F}_T(x) = \sum_{t=1}^{T} \beta_t^* b(x; \gamma_t^*) \approx \mathcal{F}(x)$ from (2.5).

1: Initialize $\mathcal{F}_0(x) = 0$
2: for $t$ from 1 to $T$ do
   a: Compute
      \[
      (\beta_t^*, \gamma_t^*) = \arg \min_{\beta_t, \gamma_t} \sum_{i=1}^{N} \mathcal{L}(y_i, \mathcal{F}_{t-1}(x_i) + \beta_t b(x_i; \gamma_t)).
      \]
   b: Set $\mathcal{F}_t(x) = \mathcal{F}_{t-1}(x) + \beta_t^* b(x; \gamma_t^*)$.
3: end for

Fitting a model with a basis function expansion as in (2.5) is typically done by minimizing a loss function averaged over the training data and all possible basis functions,

\[
\min_{(\beta_t, \gamma_t)} \sum_{i=1}^{N} \mathcal{L}(y_i, \sum_{t=1}^{T} \beta_t b(x_i; \gamma_t)) \tag{2.6}
\]

where $N$ and $T$ denotes the number of data points and the number of basis functions respectively. $\mathcal{L}(y, \mathcal{F}(x))$ denotes a loss function penalizing the difference between the desired output $y$ and the predicted output $\mathcal{F}(x)$. Solving this optimization problem usually requires computationally expensive numerical optimization techniques. However, a simpler alternative is to use a greedy strategy where we solve the optimization for a single basis function

\[
\min_{\beta, \gamma} \sum_{i=1}^{N} \mathcal{L}(y_i, \beta b(x_i; \gamma)) \tag{2.7}
\]

and add it to the sum of previous basis functions. This technique, called *forward stagewise additive modeling* approximates the solution to (2.6) by sequentially adding new basis functions $b(x; \gamma_t)$ with weights $\beta_t$, without adjusting any of the previously added terms. The procedure is described in Algorithm 2.1.

The training of a boosting classifier can be performed using forward stagewise additive modeling. The weights $\beta_t$ in (2.5) correspond to the weights $\alpha_t$ in (2.3). The basis functions $b(x; \gamma_t)$ correspond to the weak learners $f_t(x)$, parameterized by $\gamma_t$. The theory of additive models and this interpretation explain much of the success of boosting. For a more extensive analysis of linear basis expansion and forward stagewise modeling, see Bishop (2006) or Hastie et al. (2009).

The choice of loss function $\mathcal{L}(y, \mathcal{F}(x))$ is important in order to find good models. In classification, the purpose of the loss function is to penalize incorrect classifications in order to find samples to emphasize in the training algorithm.
We define the classification margin of a sample as $yF(x)$. In the case of binary classification, the classification margin is small if the confidence is low, large positive if the classifier correctly classifies the sample with high confidence and large negative if the sample is misclassified with high confidence.

In regression, the purpose of the loss function is to minimize the error between the classifier output and the true labels of the samples in the training set. For this purpose, we define the regression margin as the residual $y - F(x)$.

The appropriateness of specific loss functions can be motivated by examining them in a plot. Different loss functions for classification and regression are presented in Figures 2.2 and 2.3, where they are plotted as functions of respective margin.

For classification, the misclassification is a zero/one loss function given by

$$
L_{\text{mis}}(y, F(x)) = \begin{cases} 
0, & y = \text{sign}[F(x)], \\
1, & y \neq \text{sign}[F(x)],
\end{cases}
$$

(2.8)

that penalizes misclassification with unit weight. This is a simple and interpretable loss function but it has obvious drawbacks. Misclassified samples are given the same penalty irrespectively of the classifier confidence and correctly classified samples are given no penalty independent of the confidence they are classified with. The exponential loss function, see Figure 2.2, is given by

$$
L_{\text{exp}}(y, F(x)) = \exp[-yF(x)],
$$

(2.9)

and is a monotonously decreasing function useful in classification tasks. An increased loss for decreasing classification margin puts higher emphasis on samples that are misclassified with high confidence and instead of discarding all samples that are classified correctly, emphasize is still put on those of low confidence. A major advantage with the exponential loss is the computationally cheap updates it leads to, see the weight update in Step (2d) of AdaBoost later in Algorithm 2.2.

In the regression setting, a symmetric loss function is useful since it penalizes under- and overshoots equally, see Figure 2.3. The absolute error,

$$
L_{\text{abs}}(y, F(x)) = |y - F(x)|,
$$

(2.10)

fulfills this criterion but it often results in complicated algorithms due to its non-linearity. To resolve this problem, the squared error loss can be used,

$$
L_{\text{sq}}(y, F(x)) = (y - F(x))^2,
$$

(2.11)

and it is by far the most used loss function in statistical learning (Hastie et al., 2009, page 18). The behavior of the squared error loss function is desired in the regression setting and the reason for this can be seen in Figure 2.3. The squared error loss function fulfills the criterion of symmetry and resembles the absolute error for small values of the regression margin.

By examining the squared error loss function in Figure 2.2, we conclude that it is inappropriate to use in classification tasks since the penalty increases when the classification margin is greater than one, $yF(x) > 1$. This leads to increased importance of samples that are already classified correctly and hence it contradicts the purpose of the loss function for classification.
2.2 Object categorization

Figure 2.2: Three common loss functions for classification plotted as a function of the classification margin $y_F(x)$. The exponential loss function is the most appropriate for classification.

Figure 2.3: Two common loss functions for regression plotted as a function of the regression margin $y - F(x)$. The squared error loss function better for regression and is by far the most common.
AdaBoost

Robert Schapire and Yoav Freund were two of the first researchers to propose successful boosting algorithms. Together, they developed AdaBoost (Freund and Schapire, 1997), which even today is one of the most popular algorithms for classification. For this work they received the prestigious Gödel Prize in 2003 (Bein, 2003). An overview of AdaBoost specialized to binary classification is presented in Algorithm 2.2.

Since the introduction of AdaBoost, much work has been devoted to explain its success in producing accurate classifiers. A few years after the introduction of the algorithm, it was shown that it is equivalent to forward stagewise additive modeling with the exponential loss function (2.9) (Friedman et al., 2000).

The expected value of (2.9) is minimized at

\[ F(x) = \frac{1}{2} \log \left( \frac{P(y = 1 | x)}{P(y = -1 | x)} \right), \]  

and hence

\[ P(y = 1 | x) = \frac{e^{F(x)}}{e^{-F(x)} + e^{F(x)}}, \]  

(2.13a)

\[ P(y = -1 | x) = \frac{e^{-F(x)}}{e^{-F(x)} + e^{F(x)}}. \]  

(2.13b)

By multiplying (2.13a) by \( e^{F(x)} \), we get the logistic model

\[ p(x) = \frac{e^{2F(x)}}{1 + e^{2F(x)}}. \]  

(2.14)

Hence, the function \( F(x) \) that minimizes (2.9) is the symmetric logistic transform of \( P(y = 1 | x) \). We let this motivate our interpretation of \( F(x) \) as a measure of confidence in classifying the sample \( x \) as belonging to one of the respective classes in the binary classification task. For more details, see Friedman et al. (2000).

In AdaBoost, emphasize is put on samples that are misclassified by the current weak learner. Misclassified samples receive higher weights and are thereby prioritized in the fitting of the next weak learner, see Step (2d) in Algorithm 2.2. The only requirement on the weak learners is that they all classify a sample correctly with accuracy slightly higher than 50%. According to Hastie et al. (2009), Breiman referred to AdaBoost with trees as the "best off-the-shelf classifier in the world" during a NIPS workshop in 1996 (see also Breiman (1998)).

Algorithm 2.2 shows the iterative training process of a boosting classifier. In Step (2a), a weak learner \( f_t(x) \) is fitted to the weighted training set, corresponding to a minimization of (2.7) with input \( x_i \) weighted by \( w_i \). The error of the weak learner \( f_t(x) \) for predicting the labels \( y \) is computed in Step (2b). Each linear model coefficient \( \alpha_t \) is computed in Step (2c). The exponential loss function results in the simple weight update in Step (2d), where \( I \) denotes the indicator function given by,

\[ I(u, v) = \begin{cases} 1, & \text{if } u \neq v, \\ 0, & \text{if } u = v. \end{cases} \]  

(2.15)
Algorithm 2.2 Binary AdaBoost

**Input:** Set of labeled samples $L = \{(x_1, y_1), \ldots, (x_{|L|}, y_{|L|})\}$, $y_i \in \{-1, +1\}$.

**Output:** Classifier $\text{sign}[F(x)] = \text{sign}\left[\sum_{t=1}^{T} \alpha_t f_t(x)\right]$.

1: Initialize the observation weights $w_i = 1/N$, for $i = 1, 2, \ldots, N$.
2: for $t = 1, \ldots, T$ do
   a: Fit a classifier $f_t(x) \in \{-1, +1\}$ using weights $w_i$ on the training data.
   b: Compute the estimation error $\epsilon_t = \frac{\sum_{i=1}^{N} w_i I(y_i, f_t(x_i))}{\sum_{i=1}^{N} w_i}$.
   c: Compute the coefficient $\alpha_t = \log\left(\frac{(1 - \epsilon_t)}{\epsilon_t}\right)$.
   d: Set $w_i \leftarrow w_i \exp[\alpha_t I(y_i, f_t(x_i))]$, $i = 1, 2, \ldots, N$, and renormalize to obtain $\sum_{i=1}^{N} w_i = 1$.
3: end for

The Binary AdaBoost algorithm is presented since it is one of the most popular algorithms and it is used as a benchmark later when we evaluate the performance of the semi-supervised algorithms. Many SSL boosting algorithms such as RegBoost, introduced in Section 3.2.2, simplifies to AdaBoost when there are only labeled samples in the training set.

A classification example

We now introduce a simple example to illustrate the task of binary classification. A synthetic data set is generated with two concentric circles of different radii representing two separate classes, see Figure 2.4a. Each circle consists of 100 points. The classification problem is to find a decision boundary that separates the two circles.

In Example 2.1, we present how AdaBoost finds a decision boundary using the iterative boosting training procedure. The decision boundary is successively updated and improved as more weak learners are added.

We return to this example in Chapter 3 to investigate the situation where only a small amount of labeled samples are available. There, we present three SSL algorithms to present how the decision boundary can be improved by utilizing unlabeled samples in the training procedure.
2.1 Example: Classification with AdaBoost

In this example, we seek to find a decision boundary to separate the two circles in Figure 2.4a using AdaBoost. In Figures 2.4b to 2.4f, we present the current strong learner at different stages in the iterative training procedure. The confidence of the current strong learner is visualized as a shaded background, where a bright color corresponds to high confidence of a point at that position belonging to the positive class (the inner circle). A darker shade corresponds to high confidence of a point belonging to the outer circle. The decision boundary, found by thresholding the confidence for maximum classification rate for the training set, is drawn as a line.

![Diagram of AdaBoost training procedure](image)

Figure 2.4: The training procedure of AdaBoost.

We see that the classification model is successively improved as more weak learners are added. In this example we use decision stumps as weak learners, for details see Section 2.3.2.

LogitBoost

Another well-known supervised boosting algorithm is LogitBoost (Friedman et al., 2000). In contrast to AdaBoost, LogitBoost is developed for regression problems, i.e., the output from the weak learners are continuous values rather than discrete class labels. The algorithm fits an additive logistic regression model by stage-wise optimization of the Bernoulli log-likelihood. An overview of LogitBoost is
2.2 Object categorization

Algorithm 2.3 Binary LogitBoost

**Input:** Set of labeled samples \( L = \{(x_1, y_1^*), \ldots, (x_{|L|}, y_{|L|}^*)\}, y_i^* \in \{0, 1\}. 

**Output:** Classifier

\[
\text{sign}[F(x)] = \text{sign}\left[\sum_{t=1}^T f_t(x)\right].
\]

1: Initialize the observation weights \( w_i = 1/N \), for \( i = 1, 2, \ldots, N \), \( F(x) = 0 \) and probability estimates \( p(x_i) = 1/2 \).

2: for \( t = 1, \ldots, T \) do
   
a: Compute the working response and weights
   
   \[
z_i = \frac{y_i^* - p(x_i)}{p(x_i)(1 - p(x_i))}, \quad w_i = p(x_i)(1 - p(x_i)).
   \]

   b: Fit the function \( f_t(x) \) by a weighted least-squares regression of \( z_i \) to \( x_i \) using weights \( w_i \).

   c: Update \( F(x) \leftarrow F(x) + \frac{1}{2} f_t(x) \) and \( p(x) \leftarrow e^{F(x)}/(e^{F(x)} + e^{-F(x)}) \).

3: end for

presented in Algorithm 2.3.

In order to keep the original formulation of LogitBoost, we use a 0/1 response:

\[
y_i^* = \begin{cases} 
1, & \text{pedestrian detected,} \\
0, & \text{no pedestrian detected,}
\end{cases}
\]

(2.16)

where \( y_i^* \) denotes the class label of sample \( x_i \) in the training set. \( y_i^* = 0 \) corresponds to \( y_i = -1 \) in (2.2). The final classifier \( \text{sign}[F(x)] \) predicts labels using the notation in (2.2). The probability of \( y_i^* = 1 \) is modeled by

\[
p(x) = \frac{e^{F(x)}}{e^{F(x)} + e^{-F(x)}}.
\]

(2.17)

We find the optimal weak learner \( f_t(x) \) to add to the strong learner \( F(x) \) by forming the expected log-likelihood

\[
E[\text{I}(F(x) + f_t(x))] = E\left[2y^*(F(x) + f_t(x)) - \log(1 + e^{2(F(x) + f_t(x))})\right].
\]

(2.18)

Conditioning on \( x \) and forming a *Newton update*, the weak learner solves the weighted least-squares approximation to the log-likelihood

\[
\min_{f_t(x)} E_{w(x)}[F(x) + \frac{1}{2} \frac{y_i^* - p(x_i)}{p(x_i)(1 - p(x_i))} - (F(x) + f_t(x))]^2,
\]

(2.19)

where

\[
E_{w(x)}[g(x, y)] = \frac{E[w(x, y)g(x, y)]}{E[w(x, y)]},
\]

(2.20)
refers to a \textit{weighted expectation}, see Friedman et al. (2000) for details.

In Example 2.2, we revisit the concentric circles introduced in Example 2.1 to visualize the training procedure of LogitBoost.

---

**2.2 Example: Classification with LogitBoost**

In this example, we present the LogitBoost training procedure using the concentric circles. As in Example 2.1, the classification model is improved as more weak learners are added, see Figures 2.5b to 2.5f. We let the shading represent the output from the current strong learner \( F(x) \), with a bright color representing a higher confidence of a point belonging to the positive class (the inner circle).

\[
\begin{align*}
(a) & \text{ Initial settings.} \\
(b) & \text{ } T = 1. \\
(c) & \text{ } T = 2. \\
(d) & \text{ } T = 5. \\
(e) & \text{ } T = 50. \\
(f) & \text{ } T = 100.
\end{align*}
\]

\textbf{Figure 2.5: Binary classification with LogitBoost.}

Similarly to AdaBoost, LogitBoost finds a proper decision boundary separating the two circles. In this example we use \textit{regression stumps} as weak learners, for details see Section 2.3.2.

The performance of boosting algorithms based on the exponential loss function (2.9), e.g. AdaBoost, has been empirically observed to dramatically degrade when there are outliers or wrongly labeled samples in the training set (Hastie et al., 2009, chapter 10.6). The squared error loss function (2.11) is also sensitive to outliers, which produces high penalties. By introducing limits on the working response, \(-z_{\text{max}} < z_i < z_{\text{max}}\), LogitBoost has proven to be robust against outliers. The authors found by empirical studies that values of \( z_{\text{max}} \in [2, 4] \) works well (Friedman et al., 2000). In Chapter 3 we use a \textit{self-learning} algorithm to train a semi-supervised classifier based on LogitBoost.
2.3 Implementing a supervised boosting classifier for pedestrian detection

The above described boosting algorithms can be used in many different tasks. In order to construct a good classifier, it is always good to have knowledge about the application area. The following sections describe and discuss some general practices for pedestrian detection using boosting classifiers.

2.3.1 Feature extraction

In all machine learning systems it is important to find a set of suitable features to describe the input. For a classification problem, we want features that are discriminative, compact and computationally efficient.

Histogram of oriented gradients, HOG, is a feature extraction method introduced by Dalal and Triggs (2005). HOG describes local object appearance and shape by counting occurrences of gradient orientations in different parts of images.

A gradient image describes the change in intensity in an image and is computed by combining the horizontal and vertical derivative images. The derivative images are obtained by filtering the original image with the respective filter kernels

\[ H_h = [-1, 0, 1], \]  
\[ H_v = [-1, 0, 1]^T. \]

In order to compute the HOG feature of an image, its gradient image is computed and divided into a grid of cells. In every cell, histograms are computed with each gradient quantized by its angle and weighted by its magnitude. Each histogram is normalized with the norm of the combined histograms of adjacent cells, resulting in an expansion of the number of elements in the feature vector. In order to reduce the effects of outliers, values in the feature vector bigger than a specified level are truncated.

The HOG features are proven to be very efficient in pedestrian detection tasks. For details, see Dalal and Triggs (2005). In this thesis, we use an implementation of HOG developed by Dollár et al. (2014). We use cells of size 8 × 8 pixels, normalize with the \( L_2 \)-norm of the four adjacent 2 × 2 cells and truncate values above 0.2. In Figure 2.6, we present an example image from Autoliv’s pedestrian data set and two visualizations of its HOG feature representation. The images are 90 × 45 pixels, resulting in a 11 × 5 grid of cells, as seen in Figures 2.6b and 2.6c, where the spatially placed circles and arrows represent the histograms for the respective cells. In Figure 2.6b, each angle in the histogram is represented by a line of that angle, centered at the cell center, with intensity proportional to its weight count. Contours in the example image are seen as distinct angles of higher intensity. Figure 2.6c presents the most significant angles in an arrow plot, where
the arrows lengths represent the mean magnitude of the respective cells. We see that the pedestrian’s head and the object in her hands are visible in the feature representation. With $90 \times 45$ pixels images and the settings described above, we obtain a HOG feature vector in $\mathbb{R}^{1980}$ to represent each image.

![Example image.](image1) ![Full HOG representation.](image2) ![Most significant angles.](image3)

**Figure 2.6:** Example image of a young pedestrian, her HOG feature vector and an arrow plot visualizing the most significant angles, weighted by the mean magnitude of respective cell.

### 2.3.2 Choice of weak learners

Boosting algorithms are not standalone classifiers, but rather frameworks to improve other classifiers. These classifiers, the weak learners, can be chosen arbitrarily and the boosting algorithm will create a stronger classifier by combining them. Much has been written about the ability of boosting to increase the classification accuracy of trees, and especially decision stumps, trees with only one node (Hastie et al., 2009). Throughout the thesis, we use decision/regression stumps as weak learners in the classification and regression algorithms, respectively.

A **decision stump** is a decision tree with only one split variable, which reduces to an indicator function that indicates the class of sample $x$ according to

$$f(x) = \begin{cases} +1, & \text{if } x \geq \gamma, \\ -1, & \text{if } x < \gamma, \end{cases}$$

for some threshold $\gamma$. A regression tree can be seen as a sum of weighted indicator functions

$$f(x) = \sum_{m=1}^{M+1} c_m I(x, \gamma),$$
where $M$ denotes the number of splits in the input space and $c_m$ are the respective weights of those regions. The regression stump is a regression tree with only one split variable and two weighted regions.

### 2.3.3 Cascade of boosting classifiers

Viola and Jones (2001) propose to combine successively more complex classifiers in a *cascade* structure, see Figure 2.7. The key motivation is that simpler, and therefore more efficient, classifiers can be constructed to reject many samples from the negative class while detecting almost all samples of the positive class. The samples that we process in the classifiers are sub-windows of images (recall Section 2.1). In an image, there is naturally a higher rate of sub-windows not containing pedestrians (samples of the negative class) than sub-windows containing pedestrians (samples of the positive class).

By placing successively more complex classifiers after each other and allowing each classifier to abort further classification of any sample, the speed of an object detection system is significantly increased. Also, the computational demand is reduced since the more complex classifiers only need to process the samples that are hard to classify.

![Cascade structure](image)

**Figure 2.7:** Classifiers placed after each other in a cascade structure. The initial classifiers eliminate a large number of negative samples with very little processing. After several stages of processing, the number of samples to process has been reduced significantly.
Semi-supervised learning

When the available amount of labeled data is insufficient or when the labeling process is costly, it would be desirable to utilize unlabeled data to improve classification accuracy. This is the objective in semi-supervised learning (SSL), which we introduce in this chapter. We define the semi-supervised setting, discuss some general issues and present three SSL algorithms based on boosting classifiers.

By utilizing the information in an additional set of unlabeled samples, we seek to find a better classification model than what can be found using only a set of labeled samples. We define the set of unlabeled samples

\[ U = \{ x_{|L|+1}, \ldots, x_{|L|+|U|} \}. \]  

(3.1)

The available training set in a semi-supervised algorithm is the union of the labeled and unlabeled data, i.e. \( S = L \cup U \), using \( L \) as defined in (2.1).

In order to successfully learn from the unlabeled samples, we need to assume that they are distributed with some underlying structure. We again consider the SSL assumptions discussed in Chapter 1, as formulated by Chapelle et al. (2006):

**Semi-supervised smoothness assumption:** If two points in a high-density region are close, then so should their class labels, according to some measure.

**Cluster assumption:** Points in the same cluster are likely to be of the same class.

**Manifold assumption:** High-dimensional data lies roughly on a low-dimensional manifold.

Algorithms have been formulated around different assumptions and naturally there exist data that do not fulfill these assumptions. This insight implies that the choice of SSL algorithm should be made with consideration to the data and that knowledge about the data’s underlying distribution is vital, as in all machine learning problems (Murphy, 2012).
In this thesis, we investigate two different approaches to semi-supervised learning, *semi-supervised boosting* and *self-learning*. In Section 3.2, we extend the formulation of supervised boosting from Section 2.2.1 to include unlabeled samples. We call this semi-supervised boosting and describe two algorithms of this character: *SemiBoost* (Mallapragada et al., 2009) and *RegBoost* (Chen and Wang, 2011). In Section 3.3, we outline a self-learning scheme that utilizes a supervised classifier to label the unlabeled samples in successive runs.

After each algorithm is presented, its behavior and performance is illustrated using the example of concentric circles introduced in Example 2.1. In order to demonstrate the potential improvements of SSL, we focus on the situation where only a small amount of data points are labeled. We introduce this case using AdaBoost in Example 3.1.

### 3.1 Example: Training AdaBoost on a small amount of labeled data

Here, we revisit the concentric circles from Examples 2.1 and 2.2 under slightly changed conditions. We use the same data points, but assume that only four samples from each of the classes are labeled, marked as squares and circles in Figure 3.1a. The remaining samples are available for training but carry no labels.

![Image](image1.png)

*(a) Initial settings.*

![Image](image2.png)

*(b) T = 1.*

![Image](image3.png)

*(c) T = 100.*

**Figure 3.1:** The training procedure of AdaBoost when only a small amount of labeled samples is available. No satisfactory decision boundary is found due to an insufficient amount of labeled samples.

Supervised learning algorithms, such as AdaBoost and LogitBoost, can only utilize information in labeled samples. As seen in Figure 3.1c, the AdaBoost algorithm can not find a decision boundary that separates the two circles since there is an insufficient amount of labeled samples available.

We let this simple example illustrate the need and potential improvements of using semi-supervised learning. If the unlabeled points could be utilized in the training, it might be possible to obtain a better classification model. Before we present the SSL algorithms implemented in this thesis, we describe two of the most important issues of semi-supervised learning, assigning *pseudo-labels* to the unlabeled samples and measuring the *similarity* between samples.
3.1 Extracting information from unlabeled samples

In the ssl setting, there are no labels given to the unlabeled samples. In order to utilize these samples in the training of a classifier, they first have to be assigned pseudo-labels, labels based on knowledge about the labeled samples in the training set. The next step is to find which of the unlabeled samples to select for training. We refer to this as pseudo-labeling and selection of the unlabeled samples.

Intuitively, it seems reasonable to assign pseudo-labels based on the confidence level of the classifier, but since there is no exact way of telling whether a pseudo-label is right or wrong, the labeling has to be done with care. By introducing another measure, independent of the classifier confidence, we can improve the accuracy of both pseudo-labels and selection. An independent similarity measure proposed in several SSL algorithms is the pair-wise distances between samples, see Rosenberg et al. (2005).

In this thesis, we assign pseudo-labels based on a combination of the current classifier’s confidence and the similarity among samples, emphasizing the SSL assumptions. The similarity is based on a radial basis function,

\[ S_{i,j} = \exp \left( -\frac{d_{i,j}^2}{\sigma^2} \right), \quad (3.2) \]

where \( d_{i,j}^2 \) denotes the squared Euclidean distance between the samples \( x_i \) and \( x_j \) and \( \sigma \) is a kernel width parameter. For pedestrian detection we use the distances between the samples’ HOG feature vectors, see Section 2.3.1 and Dalal and Triggs (2005).

In order to successfully extract useful information from the unlabeled samples, a sufficient amount of data is required. Large data sets put high demands on time and memory consumption, and pair-wise computations are undesirable since they naturally result in algorithms of complexity \( \mathcal{O}(N^2) \), where \( N \) is the number of samples. By examining the similarity function (3.2) one realizes that samples far away affect each other with exponentially decreasing influence. Using this insight, we can formulate an algorithm that computes only the most important pair-wise similarity weights without having to keep all \( N^2 \) distances in the memory at the same time.

3.1.1 Reducing complexity of pair-wise computations

To decrease the computational demand of calculating (3.2), one approach is to consider only a share of the nearest points when computing the distances. A partial distance search can be utilized to speed up the calculation and reduce the memory consumption of pair-wise distances, see Chang-Da and Gray (1985).

The squared Euclidean distance \( d^2 \) between two vectors \( x = (x_1, x_2, ..., x_k) \) and \( y = (y_1, y_2, ..., y_k) \) can be written as

\[ d^2(x, y) = \sum_{n=1}^{k} (x_n - y_n)^2. \quad (3.3) \]
The distance is increasing with every term, hence the calculation can be aborted and discarded when a partial sum is greater than the current largest distance in the set of nearest samples. Since additional comparisons are required after each summation, it is not obvious that the complexity is reduced. The approach has yet proven to be useful in practice (Chang-Da and Gray, 1985).

### 3.2 Semi-supervised boosting

We refer to *semi-supervised boosting* as boosting algorithms specifically designed for the SSL setting, i.e., formulated to handle a training set $S = L \cup U$ of both labeled and unlabeled samples. In order to handle the SSL setting, we extend our formulation of supervised boosting from Section 2.2.1 by adding functionality for pseudo-labeling and selection of the unlabeled samples. An overview of the training procedure for a semi-supervised boosting algorithm is presented in Figure 3.2. We encourage the reader to compare this to the training procedure of a supervised boosting algorithm in Figure 2.1.

![Figure 3.2: Overview of a semi-supervised boosting training procedure. The supervised setting is extended by adding a set of unlabeled data. A similarity measure is combined with the confidence of the classifier to improve pseudo-labeling and selection.](image)

Semi-supervised boosting algorithms handle both pseudo-labeling and selection of the unlabeled samples in each iteration of the training loop. To increase accuracy, both the pseudo-labeling and the selection are based on a combination of the current classifier’s confidence - the strong learner - and the similarity measure. Each weak learner is fit to a selection of samples from both the labeled and the unlabeled set. If there are no unlabeled samples in the training set, the training procedure simplifies to that of supervised boosting.

In a boosting classifier based on exponential loss, incorrectly labeled samples are prone to be assigned high weights and hence get high priority in the
weak learners. This may reduce the accuracy of the final classifier significantly (Hastie et al., 2009, Chapter 10.6). Selecting incorrect pseudo-labeled samples for training also increases the chance of adding other samples with incorrect labels in successive steps. We now describe two semi-supervised boosting algorithms: SemiBoost (Mallapragada et al., 2009) and RegBoost (Chen and Wang, 2011).

### 3.2.1 SemiBoost

SemiBoost (Mallapragada et al., 2009) is a framework for semi-supervised boosting based on both the cluster and the manifold assumptions. The two assumptions are accounted for by forming a loss function based on both confidence of the current classifier and pair-wise similarity, such as

\[ L_{SB}(y, S) = L_{lu}(y, S) + CL_u(y_u, S), \]  

(3.4a)

where

\[ L_{lu}(y, S) = \sum_{i=1}^{n_l} \sum_{j=1}^{n_u} S_{i,j} \exp(-2y_i^l y_j^u), \]  

(3.4b)

\[ L_u(y_u, S) = \sum_{i,j=1}^{n_u} S_{i,j} \exp(y_i^u - y_j^u). \]  

(3.4c)

Here, (3.4b) and (3.4c) emphasize the similarity between the labeled and the unlabeled samples, and the similarity among the unlabeled samples, respectively. Similarity is emphasized by penalizing inconsistency between labels of similar samples. \( C \in \mathbb{R} \) denotes a constant weight reflecting the importance of the unlabeled samples and \( S_{i,j} \) is the similarity measure (3.2). In this formulation, \( y_i^l \) denotes the true labels of the labeled samples and \( y_i^u \) are imputed pseudo-labels of the unlabeled samples.

The optimization of (3.4a) is further simplified by formulating the upper bound \( \bar{L}_{SB} \), see Mallapragada et al. (2009) for details. The authors show that minimizing (3.4a) is equivalent to minimizing the function

\[ \bar{L}_{SB} \leq \sum_{i=1}^{n_u} (p_i - q_i)(e^{2\alpha} + e^{-2\alpha} - 1) - \sum_{i=1}^{n_u} 2\alpha f(x_i)(p_i - q_i), \]  

(3.5a)

where

\[ p_i = \sum_{j=1}^{n_l} S_{i,j} e^{-2F(x_i)} \delta(y_j, 1) + \frac{C}{2} \sum_{j=1}^{n_u} S_{i,j} e^{F(x_j) - F(x_i)}, \]  

(3.5b)

\[ q_i = \sum_{j=1}^{n_l} S_{i,j} e^{-2F(x_i)} \delta(y_j, -1) + \frac{C}{2} \sum_{j=1}^{n_u} S_{i,j} e^{F(x_i) - F(x_j)}, \]  

(3.5c)

and \( \delta(u, v) = 1 \) when \( u = v \) and 0 otherwise. Here, \( F(x) \) is the confidence of the classifier and \( f(x) \) is the output of the current weak learner. The values of \( p_i \)
Algorithm 3.1 SemiBoost

Input: Training set \( S = L \cup U \).
Output: Classifier \( \text{sign} [F(x)] = \text{sign} \left[ \sum_{t=1}^{T} \alpha_t f_t(x) \right] \).

1: Compute the pair-wise similarity \( S_{i,j} \) using (3.2) for all samples \( i, j \in S \).
2: Initialize \( F(x) = 0 \).
3: for \( t = 1, \ldots, T \) do
   a: Compute \( p_i \) and \( q_i \) for every sample using (3.5b) and (3.5c).
   b: Compute pseudo-label \( \tilde{y}_i = \text{sign}[p_i - q_i] \) for the unlabeled samples.
   c: Sample unlabeled samples \( U' \) from \( U \) according to (3.7). Form the training set \( L \cup U' \).
   d: Apply a supervised algorithm to find weak learner \( f_t(x) \) using the training set from Step 2c and their collective class labels \( \hat{y}_i \).
   e: Compute weight \( \alpha_t \) using (3.6).
   f: if \( \alpha_t \leq 0 \) then exit loop.
   g: else Update the strong learner \( F(x) \leftarrow F(x) + \alpha_t f_t(x) \).
4: end for

and \( q_i \) can be interpreted as the confidence in classifying the unlabeled sample \( x_i \in U \) into the positive and the negative class, respectively. By differentiating (3.5a) with respect to \( \alpha \) and setting it equal to 0, the optimal weight is given by

\[
\alpha = \frac{1}{4} \ln \frac{\sum_{i=1}^{n_u} p_i \delta(f(x_i), 1) + \sum_{i=1}^{n_u} q_i \delta(f(x_i), -1)}{\sum_{i=1}^{n_u} p_i \delta(f(x_i), -1) + \sum_{i=1}^{n_u} q_i \delta(f(x_i), 1)},
\]

(3.6)

A value of \( \alpha \leq 0 \) indicates that the addition of the weak learner \( f(x) \) would increase the loss function (3.5a). In this situation, we abort further execution and return the current strong learner \( F(x) \) as the final classifier.

The SemiBoost procedure is summarized in Algorithm 3.1. Each weak learner \( f_t(x) \) is added to the strong learner \( F(x) \) with the weight \( \alpha_t \). At each round, a new training set is formed of the labeled samples and a set of unlabeled samples sampled from the distribution

\[
P(x_i) = \frac{|p_i - q_i|}{\sum_{i=1}^{n_u} |p_i - q_i|}.
\]

(3.7)

The pseudo-labels of the unlabeled samples are determined by

\[
\tilde{y}_i = \text{sign}[p_i - q_i].
\]

(3.8)

The training procedure of SemiBoost is visualized in Example 3.2. In Appendix A, we present the parameter values that are used.
3.2 Example: Concentric circles revisited with SemiBoost

In this example, we again return to the concentric circles introduced in Example 3.1. Figure 3.3a presents the initial settings with 10 labeled samples in each of the classes, and the remaining 90 points of each class available for training, but without labels. The decision boundary obtained using AdaBoost is presented in Figure 3.3b. Similarly to the result in Example 3.1, AdaBoost is not able to find a satisfactory decision boundary that separates the two circles.

In Figures 3.3c to 3.3f, we present the SemiBoost training procedure. Samples marked with squares and circles are sampled with corresponding pseudo-labels and used in the training of respective weak learner.

The observant reader sees that some samples are sampled with incorrect pseudo-labels, see e.g. the circle in the lower left corner of Figure 3.3f. This is one of the risks of semi-supervised learning, but fortunately the few mislabeled samples do not have large enough influence to distort the classification model. Despite some mislabeled samples, the SemiBoost algorithm finds a better decision region than AdaBoost (c.f. Figures 3.3f and 3.3b).

In this illustrative example, we see that there exist situations where unlabeled samples can help improve a classifier.
3.2.2 RegBoost

Chen and Wang (2011) propose an SSL algorithm called RegBoost. The algorithm is a general boosting framework with regularization based on all three semi-supervised assumptions, developed within the generic margin cost functional framework for boosting (Mason et al., 2000). In this framework, boosting is treated as a greedy yet stagewise functional minimization procedure. The authors present an algorithm to use with an arbitrary loss function. In its general form, the complete RegBoost loss function is given by

\[
L_{RB}(y, F(x)) = \sum_{i \in S} \left\{ \frac{1}{|L|} I_{i, L} \gamma_i L(\hat{y}_i, F(x_i)) + \frac{1}{|U|} I_{i, U} \beta_i |N(i)|^{-1} \sum_{j \in N(i)} S_{i,j} L(\hat{y}_j, F(x_i)) \right\}.
\] (3.9)

Here, \(\gamma_i, \beta_i \in \mathbb{R}^+\) denote weights for the labeled and the unlabeled samples respectively. The purpose of \(\gamma_i\) is to reflect prior knowledge or importance of the labeled samples while \(\beta_i\) is computed from the sample density. A neighborhood \(N(i)\) is found for each unlabeled point \(i\) using a \(K\) nearest neighbor algorithm. The neighborhood size \(|N(i)| = K\), where \(K\) is the number of neighbors. \(S_{i,j}\) is a similarity measure and \(\hat{y}_i\) is a collective notation of the true labels for the labeled samples and pseudo-labels for the unlabeled samples; \(\hat{y}_i = y_i\) for \(i \in L\) and \(\hat{y}_i = \tilde{y}_i\) for \(i \in U\). The indicator function \(I\) is defined as

\[
I_{i,X} = \begin{cases} 1, & \text{if } x_i \in X, \\ 0, & \text{if } x_i \notin X, \end{cases}
\] (3.10)

where \(X\) denotes a set of samples.

The first term in (3.9) penalizes errors in the classification of the labeled samples, weighted by \(\gamma_i\). The second term penalizes inconsistency among the classification of sample \(i\) and its nearest neighbors \(j \in N(i)\), weighted by the similarity \(S_{i,j}\), the neighborhood size \(|N(i)|\) and the weights \(\beta_i\).

In our implementation of RegBoost, we use the exponential loss function (2.9), i.e., the same loss function as in AdaBoost in Section 2.2.1. Together with the similarity measure (3.2), the general loss function (3.9) reduces to

\[
L_{RB}(y, F(x)) = \sum_{i \in S} \left\{ \frac{1}{|L|} I_{i, L} \gamma_i e^{-\hat{y}_i F(x_i)} + \frac{1}{|U|} I_{i, U} \beta_i |N(i)|^{-1} \sum_{j \in N(i)} S_{i,j} e^{-\hat{y}_j F(x_i)} \right\}.
\] (3.11)

Values of \(\gamma_i\) and \(K\) and how to tune them are further discussed in Chapter 4. The weights \(\beta_i\) for the unlabeled samples are based on the sample density in the input space. A kernel density estimator can be used to approximate this density for the complete training set \(S\) using a Gaussian kernel function with bandwidth
3.2 Semi-supervised boosting

\[ p(x) = \frac{1}{|S|} \sum_{i=1}^{|S|} \exp \left( -\frac{||x - x_i||^2}{2h^2} \right). \quad (3.12) \]

The resulting weights for the unlabeled samples are computed by

\[ \beta_i = \sin \left( \frac{\pi}{2} \left[ \bar{p}(x_i) \right] \right), \quad (3.13) \]

where \( \eta \in \mathbb{Z}^+ \) determines the relative influence among unlabeled samples in areas of different density. Here, \( \bar{p}(x_i) \) denoted the normalized density estimate computed by

\[ \bar{p}(x_i) = \frac{p(x_i) - p_{\min}}{p_{\max} - p_{\min}}, \quad (3.14) \]

where \( p_{\max} \) and \( p_{\min} \) are the maximum and minimum density values over the whole training set \( S \), respectively. The formulation of \( \beta_i \) is based on the semi-supervised cluster assumption, which states that samples of the same class are likely to appear together in clusters of high density, separated by low density regions. The weights \( \beta_i \) allow regularization to be exerted severely on unlabeled points in high density regions while it lessens the regularization effects on unlabeled points in low density regions.

The SSL assumptions are emphasized as follows: The cluster assumption by considering the confidence of the classifier \( F(x) \) in both pseudo-labeling and selection, the manifold assumption by using the similarity \( S_{i,j} \) and the semi-supervised smoothness assumption by using the density based weights \( \beta_i \) to regularize the unlabeled points. In order to decrease the computational demand and the problem of tuning the kernel bandwidth \( h \) in (3.12), the semi-supervised smoothness assumption can be excluded by setting all \( \beta_i = 1/|U| \). We adopt this strategy in our evaluation on the large-scale pedestrian detection data set.

Chen and Wang (2011) shows that finding the optimal weak learner \( f(x) \) of (3.11) is equivalent to minimizing

\[ \sum_{i: f(x_i) \neq y_i} \hat{D}(i) + \sum_{i: f(x_i) = y_i} \left\{ \frac{-\bar{I}_i U R(i)}{Z |U|} \right\}, \quad (3.15a) \]

where

\[ R(i) = \beta_i |N(i)|^{-1} \left| \sum_{j \in N(i)} S_{i,j} y_j e^{-\hat{y}_j F(x_i)} \right|, \quad (3.15b) \]

\[ R_U(i) = \beta_i |N(i)|^{-1} \sum_{j \in N(i)} S_{i,j} e^{-\hat{y}_j F(x_i)}, \quad (3.15c) \]

\[ \hat{D}(i) = -\frac{1}{|L|} \bar{I}_i L Y_i e^{-\hat{y}_i F(x_i)} + \frac{1}{|U|} \bar{I}_i U R_U(i), \quad (3.15d) \]

\[ Z = -\sum_{k \in S} \left\{ \frac{1}{|L|} \bar{I}_k L Y_k e^{-\hat{y}_k F(x_k)} + \frac{1}{|U|} \bar{I}_k U R_U(k) \right\}. \quad (3.15e) \]
By minimizing (3.15a), both the misclassification error in the weak learner $f(x)$ and the class label inconsistency in the neighborhoods $N(i)$ are emphasized. $\tilde{D}(i)$ in (3.15e) is the empirical data distribution, from which to sample a training set in each iteration of the boosting loop. $R_U(i)$ describes the influence of the unlabeled samples in $\tilde{D}(i)$. $Z$ in (3.15e) is a normalization factor such that $\sum_{i \in S} \tilde{D}(i) = 1$. $R(i)$ is a density-dependent class label inconsistency measure, which we also use to find pseudo-labels of the unlabeled samples,

$$\bar{y}_i = \text{sign} \left[ \sum_{j \in N(i)} S_{i,j} \hat{y}_j e^{-\hat{y}_j F(x_i)} \right].$$  \hfill (3.16)

The first weak learner is fitted to an initial training set constructed by all the labeled samples and the pseudo-labeled unlabeled samples of top confidence. These pseudo-labels are assigned using (3.16) with a similarity measure $S_{i,j}$ computed for unlabeled samples $i \in U$ and labeled samples $j \in L$. The confidences of these pseudo-labels are computed as

$$B(i) = \left| \sum_{j \in N(i)} S_{i,j} \hat{y}_j \right|.$$ \hfill (3.17)

Once a proper weak learner $f_t$ is found, a step size $\alpha_t$ is computed to minimize $\mathcal{L}(F_{t-1} + \alpha_t f_t)$, recall Algorithm 2.1. In general, $\alpha_t$ is a step size for linear search and needs to be chosen based on a specific cost function (Mason et al., 2000). Using the loss function (2.9), Chen and Wang (2011) show that the optimal step size is given by

$$\alpha_t = \frac{1}{2} \ln \left( \frac{1}{|U|} \sum_{i \in S} \left\{ \tilde{I}_i, U \beta_i |N(i)|^{-1} \delta[f_t(x_i), -1] \sum_{j \in N(i)} \delta(\hat{y}_j, -1)S_{i,j} e^{F_{t-1}(x_i)} + \tilde{I}_i, U \beta_i |N(i)|^{-1} \delta[f_t(x_i), +1] \sum_{j \in N(i)} \delta(\hat{y}_j, +1)S_{i,j} e^{-F_{t-1}(x_i)} \right\} \right).$$ \hfill (3.18)

where

$$P(i) = \frac{1}{|U|} \sum_{i \in S} \left\{ \tilde{I}_i, U \beta_i |N(i)|^{-1} \delta[f_t(x_i), -1] \sum_{j \in N(i)} \delta(\hat{y}_j, -1)S_{i,j} e^{F_{t-1}(x_i)} + \tilde{I}_i, U \beta_i |N(i)|^{-1} \delta[f_t(x_i), +1] \sum_{j \in N(i)} \delta(\hat{y}_j, +1)S_{i,j} e^{-F_{t-1}(x_i)} \right\},$$

$$Q(i) = \frac{1}{|U|} \sum_{i \in S} \left\{ \tilde{I}_i, U \beta_i |N(i)|^{-1} \delta[f_t(x_i), -1] \sum_{j \in N(i)} \delta(\hat{y}_j, -1)S_{i,j} e^{F_{t-1}(x_i)} + \tilde{I}_i, U \beta_i |N(i)|^{-1} \delta[f_t(x_i), +1] \sum_{j \in N(i)} \delta(\hat{y}_j, +1)S_{i,j} e^{-F_{t-1}(x_i)} \right\}.$$ 

The boosting training procedure should be terminated when no weak learner can be found to further minimize the loss function (3.11). The RegBoost algorithm
is aborted, and the current strong learner \( F(x) \) is returned, when

\[
\sum_{i : f(x_i) \neq \hat{y}_i} \hat{D}(i) + \sum_{i : f(x_i) = \hat{y}_i} \left\{ \frac{\bar{I}_i, U R(i)}{Z|U|} \right\} > \frac{1}{2}.
\] (3.19)

An overview of RegBoost is presented in Algorithm 3.2. In Example 3.3, we return to the concentric circles to visualize how RegBoost works. See Chapter 4 for further analysis of its performance on other data sets and a case study of RegBoost for pedestrian detection. In Section 4.3.3, we present our simple intuition of the parameters and their influence. In Appendix A, we present the parameter values that are used.

**Algorithm 3.2 RegBoost**

**Input:** Training set \( S = L \cup U \).

**Output:** Classifier \( \text{sign} \left[ F(x) \right] = \text{sign} \left[ \sum_{t=1}^{T} \alpha_t f_t(x) \right] \).

1: Set weights \( \gamma_i \) for \( i \in L \). Set \( F(x) = 0 \).
2: Estimate density function on \( S \). Set weights \( \beta_i \) with (3.13) and compute neighborhood \( N(i) \) for \( i \in U \) using a K nearest neighbor algorithm.
3: Compute the similarity \( S_{i,j} \) using (3.2) for all samples \( i, j \in S \) and assign pseudo-labels \( \hat{y}_i \) to the unlabeled points \( i \in U \).
4: Compute confidence \( B(i) \) for the pseudo-labels and construct the initial training set \( L \cup U' \) where \( U' = \{ i | i \in U, B(i) \) in the \( \rho \)th percentile for \( B(i) > 0 \} \).
5: for \( t = 1, \ldots, T \) do
   a: Apply supervised algorithm to find weak learner \( f_t(x) \) using the training set and their class labels \( \hat{y}_i \).
   b: if Termination condition (3.19) is met
      then Return classifier \( \text{sign} \left[ \sum_{k=1}^{t} \alpha_k f_k(x) \right] \) and exit loop.
      else Compute weight \( \alpha_t \) using (3.18) and update the strong learner \( F_t(x) \leftarrow F_{t-1}(x) + \alpha_t f_t(x) \).
   c: Update pseudo-labels \( \hat{y}_i \) using (3.16) for unlabeled samples \( i \in U \).
   d: Update the empirical data distribution \( \hat{D}_t(i) \) using (3.15e) and sample a new training set from \( S \).
6: end for
3.3 Example: Concentric circles revisited with RegBoost

Here, we return to the concentric circles to investigate the training procedure of RegBoost. The initial setting, equivalent to that of Example 3.1, is presented in Figure 3.4a. The resulting and unsatisfactory decision region of AdaBoost is presented in Figure 3.4b. The RegBoost training procedure is illustrated in Figures 3.4c to 3.4f.

![Figure 3.4](image)

Figure 3.4: The SSL training procedure of RegBoost, utilizing unlabeled samples to find a better decision region than what is found using only labeled samples.

In RegBoost, the neighborhood $N(i)$ has a large influence on both the pseudo-labeling and the selection of unlabeled samples. The selected samples are often samples that are classified differently from the similar samples in its neighborhood. We have found that RegBoost has fewer incorrectly pseudo-labeled samples than SemiBoost, and motivate this with the influence of the neighborhood. The RegBoost algorithm is able to find a better decision region than AdaBoost (c.f. Figures 3.4f and 3.4b).
3.3 Self-learning

Self-learning was one of the first ideas of semi-supervised learning. In the literature it is also known as self-training, self-labeling or decision-directed learning (Chapelle et al., 2006). The idea of self-learning is to train a supervised classifier using the labeled samples and use its predictions to pseudo-label a set of unlabeled samples that are added to the set of labeled samples. The procedure can be repeated for an arbitrary number of rounds or until all unlabeled samples have been labeled. The self-learning algorithm can be seen as a framework to use with classifiers of arbitrary types.

Figure 3.5 presents an overview of the self-learning training procedure using a supervised boosting classifier. Motivated by the reasoning in Section 3.1, we complement the classifier confidence with an independent similarity measure to obtain a more accurate basis for pseudo-labeling and selection of the unlabeled samples.

Figure 3.5: Overview of a self-learning procedure. A fully supervised algorithm is trained in its usual manner. Unlabeled samples are pseudo-labeled and successively added to the training set based on the similarity measure and the confidence of the current classifier.

Once a metric for pseudo-labeling and selection is found, implementing a self-learning algorithm is rather straightforward, but performance improvements are not guaranteed. With a poor choice of supervised algorithm, such as empirical risk minimization and 0/1-loss, the unlabeled data will not add any new information and hence self-learning has no effect (Chapelle et al., 2006, Chapter 1.1.3). Note also that the execution time of the training procedure is proportional to the number of rounds, resulting in a significant increase in time needed for training compared to whatever supervised algorithm it is based on.

To obtain a classifier that is robust against incorrectly pseudo-labeled samples, we utilize the supervised boosting classifier *LogitBoost* that has proven to be resistant to noisy data (Friedman et al., 2000).
3.3.1 Self-learning with a boosting classifier

As mentioned in Section 3.3, the self-learning training procedure is repeated for an arbitrary number of rounds. A strong classifier is trained in every round and used to pseudo-label and select samples from the set of unlabeled samples, which are added to the training set for next round. In order to improve the accuracy of the pseudo-labels and the selection, we combine the classifier confidence with an independent similarity measure.

The situation is a bit different from the case of semi-supervised boosting. In semi-supervised boosting in Section 3.2, we sought after a weighted relation among samples. Here, we seek an indication of whether a sample belongs to the positive or the negative class, respectively. For this, we propose two different measures based on distances to nearby labeled samples. The first measure is obtained by summing the similarity (3.2) in the neighborhood $N(i)$ of a sample, weighted by the neighbors’ labels, i.e.,

$$D_i = \sum_{j \in N(i)} \hat{y}_j s_{i,j}, \quad (3.20)$$

which can be interpreted as the confidence of sample $i$ belonging to the class $\text{sign}[D_i]$ with the confidence $|D_i|$. We form another similarity measure as

$$D'_i = \frac{\overline{d^2}_{i,\text{pos}}}{\overline{d^2}_{i,\text{neg}} + \overline{d^2}_{i,\text{pos}}}, \quad (3.21)$$

where $\overline{d^2}_{i,\text{pos}}$ and $\overline{d^2}_{i,\text{neg}}$ are the average squared Euclidean distances from sample $i$ to its $K$ nearest positive and negative samples, respectively. We interpret $D'_i$ as a confidence measure of sample $i$ belonging to respective class. The value of $D'_i$ is in the interval $[0, 1]$, where a small value indicates that the sample is more similar to its nearest positive samples than its nearest negative samples. Hence, it strengthens our belief that the sample belongs to the positive class. In contrast, a large value of $D'_i$ indicates that it is more likely that the sample $i$ belongs to the negative class.

In Figures 3.6 and 3.7, we present the distributions of the radial basis function similarity $D_i$ and the mean distance measure $D'_i$, respectively, on a validation data set from the pedestrian data provided by Autoliv. This data set is further described in Chapter 4. The positive and negative classes are indicated with different colors and plotted with a horizontal shift in order to visualize overlapping areas. As we see, there is a better separation of the classes using the mean distance measure $D'_i$ from (3.21).

Based on this results, we conclude that it is better to use the mean distance based similarity measure $D'_i$ when we train a classifier for pedestrian detection. By empirical studies, we found that the radial basis function similarity gave better results on the example of concentric circles, see Example 3.4. This indicates that the choice of similarity measure is problem dependent. For evaluation, it is good to analyze the distributions on a validation data set, as we present in Figures 3.6 and 3.7.
3.3 Self-learning

Figure 3.6: Distribution of the radial basis function similarity $D_{t_i}$. The positive and negative classes are marked respectively. There is a big overlap of the distributions and hence, it is hard to separate them.

Figure 3.7: Distribution of the mean distance similarity $D_{t_i}^*$. The classes are better separated than in the radial basis function similarity.
Algorithm 3.3 Self-learning procedure

**Input:** Training set $S = L \cup U$.

**Output:** Classifier $\text{sign} \left[ F_R(x) \right] = \text{sign} \left[ \sum_{t=1}^{T} \alpha_t f_t(x) \right]$.

1. Start with initial training set $T = L$.
2. for $r = 1, \ldots, R$ do
   a: Train a strong classifier $F_r(x)$ on the training set $T$.
   b: Sample and pseudo-label a set of confident unlabeled samples using (3.22d) and (3.23).
   c: Add the unlabeled samples of highest confidence to the training set $T$.
3. end for

Once a similarity measure $D$ is chosen, we combine it with the classifier confidence $F(x)$ using Bayes’ formula to derive the probability of a sample belonging to the positive class, $y = 1$,

$$p(y = 1 \mid F, D) = \frac{p(F, D \mid y = 1)p(y = 1)}{p(F, D)} \quad (3.22a)$$

$$= \frac{p(F, D \mid y = 1)p(y = 1)}{p(F, D \mid y = 1)p(y = 1) + p(F, D \mid y = -1)p(y = -1)} \quad (3.22b)$$

$$= \frac{1}{1 + \frac{p(F, D \mid y = -1)p(y = -1)}{p(F, D \mid y = 1)p(y = 1)}} \quad (3.22c)$$

Bayesian inference and Bayes’ rule are extensively discussed and exemplified in Barber (2012). By assuming that $F$ and $D$ are independent, an assumption found reasonable by empirical results, we obtain

$$p(y = 1 \mid F, D) = \frac{1}{1 + \frac{p(F \mid y = -1)p(D \mid y = -1)p(y = -1)}{p(F \mid y = 1)p(D \mid y = 1)p(y = 1)}} \quad (3.22d)$$

The corresponding probability distribution of a sample belonging to the negative class, $y = -1$, is in the case of binary classification given by

$$p(y = -1 \mid F, D) = 1 - p(y = 1 \mid F, D). \quad (3.23)$$

We use (3.22d) and (3.23) to find unlabeled samples that we are confident in classifying. In each round, we sample a set of unlabeled samples that are added to the current training set using pseudo-labels based on the probability. A detailed description of the self-learning training procedure is presented in Algorithm 3.3.

In Example 3.4, we return to the classification problem of concentric circles using this self-learning approach. The self-learning algorithm is further evaluated and discussed in Chapter 4. Our simple intuition of the parameters and their influence on the algorithm is presented in Section 4.3.3. Parameter values that are used in the experiments are presented in Appendix A.
3.3 Self-learning

3.4 Example: Concentric circles revisited with self-learning

In this example, we revisit the concentric circles with a self-learning algorithm that utilizes LogitBoost. The initial setting is equivalent to that of Example 3.2, with 10 labeled samples of each class. The initial setting and the result of LogitBoost are presented in Figures 3.8a and 3.8b.

The self-learning algorithm is repeated for 10 rounds and a set of unlabeled samples are pseudo-labeled and added to the training set after each round. In our formulation, we use (3.22d) and (3.23) to pseudo-label and select samples.

Figure 3.8: Improved classification using a self-learning algorithm. The unlabeled samples are pseudo-labeled and permanently added to the training set.

During successive rounds, a larger training set is obtained by adding further unlabeled samples. By correctly pseudo-labeling and adding samples in the leftmost part of the outer circle, the self-learning algorithm finds a proper decision region after a couple of rounds.

In contrast to the semi-supervised boosting algorithms in Examples 3.2 and 3.3, the selected unlabeled samples are permanently added to the training set and handled with equal confidence as the labeled samples.
In this chapter, we present and discuss the performance of the implemented algorithms. In order to adequately evaluate the performance of the SSL algorithms, we compare them to the performance of equivalent supervised classifiers trained under the same conditions, i.e., using only the labeled samples $L$. We also present an upper bound of the performance, i.e., the performance of a supervised algorithm trained with all samples labeled. In the experiments, we utilize AdaBoost and LogitBoost as reference supervised classifiers.

Although synthetic and of simple character, Examples 3.2 to 3.4 motivate that semi-supervised learning algorithms can improve the classification model significantly compared to a supervised classifier. Here, we present results on a few common and publicly available UCI data sets (Bache and Lichman, 2013). We also present the performance on a pedestrian data set provided by Autoliv, containing images captured in real traffic situations that have been labeled manually.

### 4.1 UCI benchmark data sets

Here, we present performance results on a few UCI data sets (Bache and Lichman, 2013). The data sets are Statlog - Australian Credit Approval (AUS), Bank Marketing (BANK), Ionosphere (ION) and Breast Cancer Wisconsin - Diagnostic (WDBC). Some of these data sets are slightly skewed, with an imbalance between the amount of samples of the positive and negative classes, respectively. Motivated by the small amount of available samples in all the sets, except BANK, we use the sets as they are provided, without balancing the classes. In the BANK data set, which is highly skewed with a majority of negative samples, we simplify the analysis by balancing the ratio between samples of the different classes. We consider only numerical features in order to simplify implementation. The sizes and input dimensions of respective data sets are presented in Table 4.1.
For proper evaluation, we first randomly split the data in half to obtain independent training and test sets. To construct an SSL setting, we utilize only 10% of the labels in the training sets. The remaining 90% of the samples in the training sets are used as the unlabeled training set $U$ in the SSL algorithms.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Size</th>
<th>Number of pos/neg</th>
<th>Input dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUS</td>
<td>690</td>
<td>307/383</td>
<td>14</td>
</tr>
<tr>
<td>BANK</td>
<td>9280</td>
<td>4640/4640</td>
<td>10</td>
</tr>
<tr>
<td>ION</td>
<td>351</td>
<td>225/126</td>
<td>34</td>
</tr>
<tr>
<td>WDBC</td>
<td>569</td>
<td>357/212</td>
<td>30</td>
</tr>
</tbody>
</table>

Table 4.1: The investigated UCI data sets and their respective size, ratio between positive and negative samples and input dimensions.

In the experiments on the UCI data sets, we evaluate SemiBoost, RegBoost and a self-learning scheme utilizing AdaBoost. We make use of AdaBoost in the self-learning algorithm since it makes a good comparison when the two other SSL algorithms are based on the exponential loss function (2.9). For comparison, we also present results of a supervised AdaBoost classifier trained under the SSL settings and another AdaBoost classifier that utilizes all the labeled samples in the training set. In all experiments, we train the classifiers with 100 weak learners and use decision stumps as weak learners. We focus our analysis on the results obtained using decision stumps, but also present results obtained using 3NN classifiers as weak learners for reference.

In order to find parameters for the SSL algorithms, we perform an exhaustive search and evaluate the performance using 10-fold cross validation on the training set. For 10 consecutive iterations, a 10th of the training set is secluded and used as a validation set on which we evaluate the current classifier. Hence, 10 classifiers are trained for each parameter set and the parameters of the classifiers with best average performance over the 10-fold cross-validation is chosen for evaluation on the test set. By evaluating the parameters using cross-validation on the training set, we avoid overfitting the classification model to the test set. See Hastie et al. (2009, Section 7.10) for a more detailed description of cross validation.

In Table 4.2, we present the obtained error rates in percent on the test sets (mean ± std), when classified using the 10 classifiers obtained in the cross validation. Results obtained using AdaBoost under equivalent settings and using all samples with labels, respectively, are presented in Table 4.3. As we see in Tables 4.2 and 4.3, there are some situations where the SSL algorithms perform better on average than the supervised AdaBoost trained under the SSL settings, but we cannot say that they are better with statistical significance. We see that RegBoost is slightly better than SemiBoost in many cases. In most cases, the self-learning algorithm performs best and it is better than the equivalent supervised classifier on three of the four data sets. The parameter values used in the experiments are presented in Appendix A.

The UCI data sets are quite small, see Table 4.1. This makes it difficult to build statistical models and to generalize the results. The BANK data set is larger than the others and gives more stable results, which is seen as lower standard
4.2 Pedestrian detection

In this section, we present performance results on a data set for pedestrian detection provided by Autoliv. On this data set, we evaluate SemiBoost, RegBoost and a self-learning algorithm utilizing LogitBoost. We use LogitBoost since it has proven resistant to outliers and is robust even if data are noisy. The classifiers are trained using a training set of 100,000 images with 90×45 pixels containing pedestrians and non-pedestrians at a 1:1 ratio. We use a balanced data set with equally many positive and negative samples in order to simplify the analysis. The results presented are obtained on an independent test set containing 50,000 samples with a 1:1 ratio between positive and negative samples. The images are generated from a late stage in an existing cascade of boosting classifiers. At this stage in the cascade, a large amount of negative samples have been rejected and the classes are nearly balanced. With the easy negative samples already rejected, we have a data set consisting of pedestrians and other objects resembling pedestrians in some sense. We present a few examples from the positive and the negative classes in the pedestrian data set in Figures 4.1 and 4.2, respectively.

We evaluate the three SSL algorithms on the pedestrian data set using 10,000 labeled samples from each class. The remaining 80,000 samples are available as unlabeled samples in the training process. We train classifiers using 300 weak

Table 4.2: Test errors in percent (mean ± std) on UCI binary classification data sets. 10% of the data set is labeled in the training.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>3NN</th>
<th>Decision stump</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data set</td>
<td>SemiBoost</td>
<td>RegBoost</td>
</tr>
<tr>
<td>AUS</td>
<td>36.6 ± 5.8</td>
<td>37.7 ± 6.2</td>
</tr>
<tr>
<td>BANK</td>
<td>16.2 ± 0.5</td>
<td>18.4 ± 0.6</td>
</tr>
<tr>
<td>ION</td>
<td>28.7 ± 6.6</td>
<td>24.2 ± 5.2</td>
</tr>
<tr>
<td>WDBC</td>
<td>9.7 ± 2.1</td>
<td>10.6 ± 2.9</td>
</tr>
</tbody>
</table>

Table 4.3: Test errors obtained using AdaBoost. The numbers in parentheses represent the percent of labeled samples that are utilized in the training.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>3NN</th>
<th>Decision stump</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data set</td>
<td>AdaBoost (10)</td>
<td>AdaBoost (100)</td>
</tr>
<tr>
<td>AUS</td>
<td>38.0 ± 5.0</td>
<td>38.6 ± 1.2</td>
</tr>
<tr>
<td>BANK</td>
<td>20.1 ± 1.1</td>
<td>20.5 ± 0.4</td>
</tr>
<tr>
<td>ION</td>
<td>22.7 ± 5.7</td>
<td>16.1 ± 1.2</td>
</tr>
<tr>
<td>WDBC</td>
<td>10.6 ± 3.3</td>
<td>11.1 ± 1.2</td>
</tr>
</tbody>
</table>
learners and utilize decision/regression stumps as weak learners. In the self-
learning scheme, we add unlabeled data and re-train classifiers in 10 rounds. We
use an independent validation set of 20 000 samples to choose which of the self-
learning classifiers to evaluate on the test set. The validation set is separate from
both the training set and the test set. For comparison, we train two supervised
LogitBoost classifiers. One using only the 10 000 labeled samples as in the SSL
setting and another using all 100 000 labeled samples.

![Example images from the positive class in the pedestrian data set provided by Autoliv.](image1)

![Example images from the negative class in the pedestrian data set provided by Autoliv.](image2)

In Section 4.1, we presented a procedure for a parameter search using 10-fold
cross validation. This process is time consuming and becomes impractical for
large data sets. Due to the time limitation of this project, we could not perform an
exhaustive parameter search for the SSL algorithms on the pedestrian data set. In
our reported results, we have chosen parameter values based on recommendations
from the authors and empirical results on subsets of the data.

We present the results of the evaluations on the pedestrian data set in ROC
(Receiver Operating Characteristic) curves, see Figure 4.3. An ROC curve is ob-
tained by plotting the hit rate against the false alarm rate. The hit rate is the rate of
true positives, samples of the positive class that are correctly classified as positive
samples. The alarm rate is the rate of false positives, negative samples that are
incorrectly classified as positive samples. The ROC curve is an implicit function
of the confidence threshold that determines the class prediction of a sample, see Murphy (2012, page 181) for more details. In the ROC curve, a higher value of the hit rate corresponds to better classification.

In Table 4.4, we present the $AUC$ (Area Under the Curve) value corresponding to each of the ROC curves in Figure 4.3. $AUC$ summarizes the ROC curve with a single number between 0 and 1, where a higher score is better. We also present the maximum classification rate defined as

$$\gamma = \frac{\sum_{i=1}^{N} I(F(x_i) \geq \tau, y_i = 1) + \sum_{i=1}^{N} I(F(x_i) < \tau, y_i = -1)}{N},$$

where $\tau$ is the confidence threshold that maximizes $\gamma$. $I(u, v)$ denotes an indicator function defined as 1 if both of its arguments $u$ and $v$ are true and 0 if not. In practical situations, $\tau$ is set to a fixed value and the objective is to achieve high performance for that chosen threshold. Hence, it is more important to find a high maximum classification rate rather than to obtain a high $AUC$ value.

In Figure 4.3 and Table 4.4, we see that the supervised LogitBoost classifier trained on all 100 000 samples as expected performs the best. We interpret this result as an upper bound on the performance. Higher performance than that is not expected. Instead, we aim to improve the results of LogitBoost under the SSL settings.

In the presented results, we see that none of the SSL algorithms is able to find a classification model that is better than the equivalent LogitBoost classifier. The ROC curve of the self-learning algorithm is very similar to the ROC curve of the SSL LogitBoost classifier. The self-learning classifier obtains a marginally improved maximum classification accuracy, but both SemiBoost and RegBoost performs worse than the equivalent supervised classifier. We again remind the reader that there exists no method to prevent incorrectly pseudo-labeled samples from being used in the training process. Using incorrect labels will distort the classification model and decrease the accuracy. This is probably what happens in the SemiBoost and RegBoost algorithms when applied to the pedestrian data set. Recall Example 3.2, where we saw that SemiBoost was prone to use incorrectly pseudo-labeled samples in the training. In that small example, the few incorrectly pseudo-labeled samples did not influence the final decision boundary, but as we see in Figure 4.3 and Table 4.4, there is a risk of heavily decreasing the performance of a classifier by unsuccessfully extracting information from unlabeled samples. SemiBoost and RegBoost are both similar to AdaBoost, formulated around an exponential loss function based on the classification margin $yF(x)$. The performance of AdaBoost has been observed empirically to dramatically degrade in situations where samples are incorrectly labeled (Hastie et al., 2009, page 348). In contrast, LogitBoost has proven resistant against outliers (Friedman et al., 2000) and is hence less sensitive to incorrect pseudo-labels.

In Figures 4.4 and 4.5, we present the classification error of the strong learner after each component on the training and test sets, respectively. We see that the performance of SemiBoost decreases after 50 weak learners on both the training and the test set. Based on this and previously presented results, we conclude that
Experimental results and discussion

Figure 4.3: ROC curves for the evaluated algorithms.

Table 4.4: AUC and maximal classification rate $\Upsilon$, reported for the evaluated algorithms.
4.2 Pedestrian detection

![Graph showing misclassification rate on the training set.](image1)

**Figure 4.4:** Misclassification rate on the training set.

![Graph showing misclassification rate on the test set.](image2)

**Figure 4.5:** Misclassification rate on the test set.
SemiBoost is not suitable for pedestrian detection using our choice of implementation.

There is only a small difference between the AUC values of the two LogitBoost classifiers, even though an additional 80 000 labeled samples are used in the better classifier. This indicates that classification of pedestrians is a hard problem to solve. Recall that the pedestrian data set is obtained in a late stage of an existing classifier cascade, hence the samples are naturally hard to separate.

With a large data set and high sample dimension, it is hard to measure similarity and to find suitable parameters for the SSL algorithms. In Sections 4.2.1 and 4.2.2, we perform two more comprehensive case studies of the training procedures of RegBoost and the self-learning algorithm on the pedestrian data set.

4.2.1 RegBoost case study

In order to appropriately analyze the semi-supervised boosting algorithms, we present a case study of the RegBoost training procedure on the pedestrian data set. In the following analysis, we refer to the steps of the RegBoost training procedure presented in Algorithm 3.2.

We use the same data as in the previously presented results, with a total amount of 100 000 samples and 10 000 labeled samples from each class. In Step (1), we set the weights $\gamma_i = 5$ for all the labeled samples, following the recommendations from the authors. In Step (2), we compute a neighborhood $N(i)$ using the 20 nearest neighbors of each sample. In order to save space in this case study, we only present the sample itself and its 5 nearest neighbors. To simplify implementation and the need for parameter tuning, we turn off the semi-supervised smoothness assumption by setting all $\beta_i = 1/|U|$. We compute the similarity $S_{i,j}$ for all samples and assign pseudo-labels in Step (3). The initial training set is constructed using all the labeled samples and the 15% of pseudo-labeled unlabeled samples of highest confidence $B(i)$ in Step (4).

Among the 35 000 samples in the initial training set, there is only 1 incorrectly pseudo-labeled sample, see Figure 4.6. The unlabeled sample is the left-most image and its five nearest neighbors are presented in the order of decreasing similarity. The images are visually similar, but only two of the five neighbors are samples of the positive class. The majority of the added unlabeled samples are drawn from the positive class. In the obtained initial training set, there are 22 454 samples of the positive class and 12 546 samples in the negative class. Naturally, there is less similarity among samples of the negative class since it contains all objects that are not pedestrians. Hence, the summed similarity measure $\sum S_{i,j}$ is in general larger among the samples in the positive class, which leads to this slightly imbalanced initial training set. In Figure 4.7, we present an unlabeled sample that is pseudo-labeled correctly.

In Step (5a), a weak learner is fitted to the initial training set. The decision stump we find as the first weak learner classifies 66% of the training set correctly. The obtained weak learner is then used to classify the entire data set. The termination condition in Step (5b) is checked and the pseudo-labels are updated in Step (5c). In the next step, the training set for next iteration is sampled from
Figure 4.6: An incorrectly pseudo-labeled sample in the initial training set (the left-most image) and its five nearest neighbors. Two of the five nearest neighbors are belonging to the positive class but there is a majority of negative samples, resulting in a negative pseudo-label. We suspect that the HOG feature is influenced by the distinct vertical lines that are apparent in all images. The images are placed in order of decreasing similarity weights.

Figure 4.7: A correctly pseudo-labeled sample and its five nearest neighbors. There is a strong visual similarity between the samples and they are all of the positive class, resulting in a confident and correct pseudo-label.

Figure 4.8: Rate of incorrect pseudo-labels $\tilde{y}_i$ among the unlabeled samples after each weak learner has been added.
both the labeled and the unlabeled data according to the empirical distribution $\hat{D}_t(i)$. In this distribution, there is a higher probability to draw samples that are similar to its nearest samples, but classified as belonging to the other class by the current strong learner $F(x)$. The distribution is also influenced by the parameters for labeled and unlabeled samples, $\gamma_i$ and $\beta_i$, respectively. Recall that we in this experiment use $\beta_i = 1/|U|$. With the set of unlabeled samples $U$ being large, $|U| = 80,000$, the unlabeled samples obtain very small weights and are sampled with low probability according to $\hat{D}_t(i)$. The weights $\beta_i$ highly influence the amount of unlabeled samples that are sampled and used at each iteration of the training procedure. Without turning off the semi-supervised smoothness assumption, these weights are dependent on a proper choice of bandwidth $h$ in the kernel estimator (3.12).

A training set to use in the fitting of the next weak learner is sampled according to $\hat{D}_t(i)$ in Step (5d). The labeled samples are added to the training set with their known labels and the unlabeled samples are added with the pseudo-labels computed in Step (5c). In order to successfully extract information from the unlabeled samples, it is important that the pseudo-labels of the unlabeled samples are accurate. In Figure 4.8, we present a graph showing the rate of incorrect pseudo-labels among the unlabeled samples after each weak learner has been added. We see that more of the pseudo-labels are incorrect after successive steps.

To find a possible reason of the behavior presented in Figure 4.8, we examine the expression (3.16). Inside the training loop of RegBoost, the pseudo-labels $\tilde{y}_i$ are based on the confidence of the current strong learner $F_t(x_i)$ and the pseudo-labels $\hat{y}_j$ of the samples in the neighborhood $j \in N(i)$. These values are combined in an exponential factor and then weighted by the pseudo-label $\hat{y}_j$ and the similarity $S_{i,j}$ of samples $i$ and $j$.

A weakness of the exponential term $e^{-\hat{y}_j F(x_i)}$ is that its size is highly dependent on the signs of $\hat{y}_j$ and $F(x_i)$. The exponential term is small when $\text{sign}[\hat{y}_j] = \text{sign}[F(x_i)]$, i.e., the pseudo-labels of its nearest neighbors are equal to the prediction of the sample $i$ by the current strong learner. When $\text{sign}[\hat{y}_j] \neq \text{sign}[F(x_i)]$, the exponential term increases significantly with $F(x_i)$ (recall that $\hat{y}_j \in \{-1, +1\}$). Hence, the pseudo-labels are sensitive to label switches in the samples in the neighborhood $N(i)$. For example, assume $F(x_i) = 3$. If a label switch occurs in the pseudo-label of a sample in the neighborhood, its influence increases from $e^{-3} \approx 0.05$ to $e^3 \approx 20.1$, and this pseudo-label may spread to other samples in the neighborhood.

We have observed this behavior in the RegBoost training procedure. An accurate initial training set is constructed and utilized in the fitting of the first weak learner, but the pseudo-labels computed in the training loop are sensitive to label switches. When a label switch occurs in the pseudo-label of one or a few samples in the neighborhood, these samples get so much influence that they are likely to corrupt the pseudo-labels of other samples as well. As we see in Figure 4.8, the accuracy of the pseudo-labels decreases throughout the training procedure. With so many incorrect pseudo-labels it is clear that there is a risk of utilizing the unlabeled samples and that they may decrease the over-all performance of the classifier. We find it interesting to further investigate if the use of soft pseudo-labels
can reduce the distortion induced by label switches. A possible approach could be to use pseudo-labels that can reflect the confidence of the pseudo-label and decrease the large change that occurs when a pseudo-label switches from $-1$ to $+1$. We suggest this to be an area for future research, see Section 5.1.

In RegBoost, new pseudo-labels are computed after each weak learner and it is therefore important that the current strong learner has good prediction accuracy from the first weak learner, an assumption that does not hold in many practical situations. This is a weakness of the semi-supervised boosting algorithms. See further discussion in Section 4.3.2 where we compare semi-supervised boosting and self-learning.

### 4.2.2 Self-learning case study

To get a better understanding of the self-learning procedure, we here perform a case study of the algorithm, as outlined in Algorithm 3.3. In this experiment we train a LogitBoost classifier for 10 consecutive rounds and after each round we sample 2.5% of the unlabeled samples that are pseudo-labeled and added to the training set for the next classifier. We use the 100 000 samples from the pedestrian data set provided by Autoliv and utilize 10 000 labeled samples from each class.

The first classifier, obtained in Step (2a), is a fully supervised LogitBoost classifier trained using only the labeled samples, i.e., equivalent to the reference classifier trained under the SSL settings. In Step (2a), we use this classifier and the mean distance similarity measure $D_i^*$ from (3.21) to compute the confidences (3.22d) and (3.23) of the unlabeled samples belonging to the positive and the negative classes, respectively. When the confidences are computed, we sample 1.75% of the total amount of samples, from each class. This results in 1 750 samples of the positive and 1 750 samples of the negative class. According to Step (2c), we only add the unlabeled samples of highest confidence to the training set, discarding samples with confidence below 95% of the maximum confidence. We use this threshold to reduce the addition of unlabeled samples with incorrect pseudo-labels. In the first round, we add 1 746 of the unlabeled samples to the initial training set. Out of the new samples, there are 120 samples that are added with incorrect pseudo-labels, resulting in a new training set with 21 746 samples and a total ratio of incorrectly labeled samples being $120/21746 = 0.0055$.

In Figures 4.9 and 4.10, we present an unlabeled sample (the left-most image) and its 5 nearest positive and negative samples in the labeled training set. This unlabeled sample is sampled and added to the training set with correct pseudo-label in the first round. The samples of the positive class share some visual similarities while the negative samples are quite different. For this sample, we compute the mean distance similarity measure

$$D_i^* = \frac{\overline{d^2_{i,pos}}}{\overline{d^2_{i,neg}} + \overline{d^2_{i,pos}}} = \frac{0.70}{0.84 + 0.70} = 0.45,$$

which indicates that the sample is more similar to its nearest positive samples than
to its nearest negative samples \( \langle d^2_{i,\text{pos}} < d^2_{i,\text{neg}} \rangle \). Together with the confidence from the classifier \( F_1(x_i) = 6.19 \), we obtain a confidence of the sample belonging to the positive class

\[
p(y = 1 \mid F, D) = 0.999\ldots
\]

(4.3)

This sample is pseudo-labeled as belonging to the positive class with high confidence and is added to the new training set.

In Figures 4.11 and 4.12, we present an unlabeled sample that is drawn and added to the training set with incorrect pseudo-label in the first round. For this sample, the mean distance similarity measure

\[
D^*_i = \frac{d^2_{i,\text{pos}}}{d^2_{i,\text{neg}} + d^2_{i,\text{pos}}} = \frac{0.70}{0.68 + 0.70} = 0.51,
\]

(4.4)

indicates that the sample is more similar to its nearest negative samples. The confidence from the classifier, \( F_1(x_i) = -1.04 \), further strengthens our belief that this sample belongs to the negative class. By combining the similarity and the confidence of the classifier, we obtain a confidence of the sample belonging to the positive class

\[
p(y = 1 \mid F, D) = 0.04,
\]

(4.5)

which leads to an incorrect pseudo-label. This fault occurs since both the similarity measure and the current classifier misclassifies the sample. We recommend the reader to compare the values of \( D^*_i \) obtained in (4.2) and (4.4) with the distribution presented in Figure 3.7. The value \( D^*_i = 0.45 \) corresponds to a high probability of the sample belonging to the positive class, while \( D^*_i = 0.51 \) is in the overlapping region and corresponds to a higher probability of the sample belonging to the negative class.

Steps (2a) to (2c) are repeated for 10 rounds and as a result we obtain 10 classifiers. The first classifier is equivalent to the supervised classifier trained under the SSL settings. The last classifier is trained on the 20 000 labeled samples in \( L \) and an additional 14 818 unlabeled samples, with a total ratio of incorrectly labeled samples being \( 1279/34818 = 0.0367 \). In Figure 4.13, we present the ratio of incorrectly labeled samples in the training set used in the respective round of the self-learning scheme. As we see, the increase is approximately linear, indicating that the ratio of incorrect samples that are added in each round is approximately constant slightly below 0.005.

In Figure 4.14, we present the misclassification rates of the classifiers from consecutive rounds in the self-learning scheme on a separate validation set. The validation data set contains 20 000 samples that have not been presented to the classifier during the training procedure and it is also separate from the test set that is used in the previous performance comparison. The misclassification rates indicate that the performance is increased during consecutive rounds in the algorithm. Based on the result presented in Figure 4.14, we choose to evaluate the classifier from round 10 on the test set. In Figure 4.15, we present the misclassification rates of the classifiers from consecutive rounds on the test set from the
previous performance evaluation. Here, we see that the results from the validation set do not generalize to the test set. On the test set, it is not the classifier obtained in round 10 that gives the best performance. Note that the classifier from round 1 is the LogitBoost classifier trained using only the 10,000 labeled samples in the pedestrian data set. By comparing this classifier with the others, we see that performance is increased by the self-learning algorithm in some stages, but it is hard to know when to stop and how well the performance of the chosen classifier will generalize to new data.

![Figure 4.9: A correctly pseudo-labeled sample and its five nearest samples among the positive samples in the training set. We see that the similarity measure finds samples that are visually similar, in two of the five nearest neighbors, the image contains two pedestrians, as in the left-most sample image.](image)

![Figure 4.10: A correctly pseudo-labeled sample and its five nearest samples among the negative samples in the training set. It is hard to see any visual similarities. This results in a higher average distance to the negative samples, resulting in a similarity measure that indicates that the sample is not of the negative class.](image)
Experimental results and discussion

Figure 4.11: An incorrectly pseudo-labeled sample and its five nearest samples among the positive samples in the training set. By examining the images, we see that the HOG features identify the position of the pedestrian and the vertical post to its left. This structure is apparent in all images.

Figure 4.12: An incorrectly pseudo-labeled sample and its five nearest samples among the negative samples in the training set.

Figure 4.13: Ratio of the total amount of incorrectly labeled samples in the training set after each round in the self-learning scheme. The increase is approximately linear, indicating that the ratio of samples that are added with incorrect pseudo-labels in each round is relatively constant.
Figure 4.14: Misclassification rates on a validation set for the classifiers trained in consecutive rounds in the self-learning scheme. This graph indicates that the performance varies as more data is added. By examining this plot we choose to evaluate the classifier obtained in round 10 on the test set, since it has the best performance on this validation data set.
Figure 4.15: Misclassification rates on the test set for the classifiers obtained in the self-learning algorithm. We see that the results from the validation set do not generalize to the test set. Unfortunately, the classifier obtained in round 10 is not the one with best performance on the test set, as is indicated on the validation set in Figure 4.14.
4.3 Discussion

In this section, we further discuss three issues we consider important in semi-supervised learning. The first problem is how to measure similarity in large data sets with samples of high dimension. Next, we discuss the difference between semi-supervised boosting and self-learning. In this analysis we point out some of the differences we can see in the case studies in Sections 4.2.1 and 4.2.2. Finally, we give a simple intuition to individual parameters used in the different algorithms.

4.3.1 Measuring similarity

In the investigated algorithms, we utilize a similarity measure based on pair-wise Euclidean distances. As the input dimension of the samples is increased, these distances become harder to interpret. It is not obvious that two points that are close in a high dimension are similar in the sense that we seek. In our implementation for pedestrian detection, we use HOG features, resulting in vectors in \( \mathbb{R}^{1980} \). Distances in \( \mathbb{R}^{1980} \) are not as easy to interpret as the distances in \( \mathbb{R}^2 \) that we handled in Examples 3.2 to 3.4, but with the use of the HOG features we saw that the similarity measure performed well on some example images containing pedestrians, see for example Figures 4.7 and 4.9.

In order to cover the great variety of object appearances we need a large data set. Pair-wise distances are undesirable due to the quadratic computational complexity, requiring both time and memory. We utilize similarity measures that are based on only the nearest samples and use a partial distance search to decrease the computational demand. The partial distance search has proven to be faster than calculating all the distances in many practical situations (Chang-Da and Gray, 1985). It also decreases the memory demand significantly by only keeping the most important distances in the memory. In Section 4.2.1, we saw that the radial basis function based similarity measure was able to obtain a good initial training set with only 1 incorrect pseudo-label among 15 000 unlabeled samples.

In the self-learning algorithm, we evaluate two different similarity measures, (3.20) and (3.21). The two measures performed well on different data sets. This indicates that the problem of finding a proper similarity measure is problem dependent and should be evaluated before being used extensively. The similarity measures can be evaluated on a validation set using histograms to represent the distributions of the similarities, see Section 3.3.1.

4.3.2 Semi-supervised boosting compared to self-learning

By concept, the self-learning scheme is simpler than the semi-supervised boosting procedure. Extending an existing supervised classifier to semi-supervised learning via a self-learning algorithm is rather intuitive and relatively easy to implement. In contrast, the two semi-supervised boosting algorithms we investigated are both standalone classifiers, less intuitive and require more time to understand and implement.
The results of the experiments that we present in this thesis show that self-
learning generally is superior to semi-supervised boosting. In most of the tests we
perform, we see that the self-learning algorithm performs similarly to the semi-
supervised boosting algorithms or slightly better. In Figure 4.3, we see that the
self-learning algorithm performs much better than the semi-supervised boosting
algorithms and hence, scales better with an increase in data set size. Regarding
the performance on pedestrian detection, it is interesting to note the difference
between the regression algorithm LogitBoost and the classification algorithms
SemiBoost and RegBoost, which are based on exponential loss. However this
analysis is beyond the scope of this thesis.

During the training procedure of a semi-supervised boosting algorithm, the
decision boundary is updated using different samples in each step of the boosting
loop. Label switches are possible during the loop and the algorithm relies on a
proper similarity measure and good classification from the first weak learner. In
the self-learning scheme, a strong classifier is trained on labeled samples alone
before pseudo-labeling and selecting the first unlabeled samples. A weakness is
that once an unlabeled sample is added to the training set, its label is determined
for good and we use it with same confidence as the original labeled samples.
In the above presented case studies, we see that some unlabeled samples are
utilized with incorrect pseudo-labels in both algorithms, but the semi-supervised
boosting algorithm RegBoost is less robust and the ratio of unlabeled samples with
incorrect pseudo-labels increases undesirably during the training procedure.

We suggest that an independent validation set is used to evaluate the per-
formance during the training procedure. The validation set can be used to find
proper parameters without overfitting the classification model to the training set
or a test set, from which results and conclusion can be drawn. In Figures 4.14
and 4.15, we present the misclassification rates after each weak learner for all the
classifiers that are obtained in the self-learning algorithm on a validation and a
test set respectively. We see that the performance varies over successive rounds
and that it is hard to find a classifier that generalizes well to new data sets. The
classifier that has the best performance on the validation set is unfortunately not
the classifier that gives the best performance on the separate test set.

4.3.3 Influence of the parameters

In this section, we present some intuitions we have obtained during the work
regarding the parameters in the investigated algorithms. We have seen that some
parameters can be chosen more independently than others and they can influence
both the computational complexity and the classification accuracy. On the UCI
data sets, we use 10-fold cross validation to find a set of suitable parameters.
However, we were not able to utilize this scheme on the larger pedestrian data set
due to the time limitation of the thesis work.

In Table 4.5, we summarize our simple intuitive interpretation for a few gen-
eral SSL parameters. In Tables 4.6 to 4.8, we describe the most important param-
eters in the investigated SSL algorithms.
The percent of samples to sample for training. In SemiBoost and Reg-Boost, we search for an integer $\Gamma \in [5, 25]$ with the step size 1. These values are recommended by the authors and should be chosen based on the size of the data set and the amount of underlying structure in the data. In the self-learning algorithm, we perform a search for $\Gamma \in [1, 5]$ with the step size 0.5. If it is easy to assign correct pseudo-labels, we suggest to use a high $\Gamma$ in order to extract as much information as possible from the unlabeled samples. If the pseudo-labels are noisy, choosing a low $\Gamma$ will reduce the risk of using unlabeled samples with incorrect pseudo-labels.

The percent of labeled data available in the training set $l = |L|/|S|$. Most results in the literature are reported for $l \in [5, 20]$. The unlabeled data set has to be sufficiently large in order for the algorithms to successfully extract useful information (Chapelle et al., 2006). We focus our experiments on $l = 10$ and $l = 20$.

The number of nearest neighbors to consider when computing the pairwise distances. For large data sets it is not feasible to compute all these distances. If a radial basis function based similarity measure is used, samples far away will not affect each other significantly and their influence can be discarded.

<table>
<thead>
<tr>
<th>$\Gamma$</th>
<th>The percent of samples to sample for training.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l$</td>
<td>The percent of labeled data available in the training set $l =</td>
</tr>
<tr>
<td>$K_N$</td>
<td>The number of nearest neighbors to consider when computing the pairwise distances.</td>
</tr>
</tbody>
</table>

| $C$ | A constant used to weight the influence of the unlabeled samples. We use the recommendation by Mallapragada et al. (2009): $C = |L|/|U|$. |
| $\sigma$ | The width of the radial basis function similarity measure. Affects the relative weights between nearby samples and is determined by $\rho$. A large $\sigma$ leads to faster decreasing influence from samples far away. |
| $\rho$ | The design parameter that determines the value of $\sigma$, the width of the similarity measure. We use a recommendation from Chen and Wang (2011) and perform a search for $\rho \in [10, 90]$ with the step size 10. A larger $\rho$ results in a larger $\sigma$. |

**Table 4.5: General SSL parameters.**

**Table 4.6: SemiBoost parameters.**
The number of neighbors in the neighborhood $N(i)$ of each sample. Based on the recommendation from Chen and Wang (2011) and empirical results on larger data sets, we search for $K \in [3, 20]$ with the step size 1. In larger data sets we found a larger $K$ to give better performance, see Table A.2. The number of computations in the boosting loop increases linearly with the number of samples in the neighborhood.

The width of the similarity measure. Affects the relative weights between nearby samples. Chen and Wang (2011) recommend setting $\sigma$ to the standard deviation of all pair-wise distances.

The width of the kernel density estimator. The design parameter that affects the weights $\beta_i$ for the unlabeled samples. This parameter highly influence the amount of unlabeled samples that are utilized in the algorithm. An inappropriate choice of $h$ may result in none of the unlabeled samples being used in the training, hence removing the possibility of extracting information from them.

Weights of the labeled samples to reflect importance or prior knowledge. We use the recommendation from Chen and Wang (2011): $\gamma_i = \min \left( \frac{|U|}{|L|}, 5 \right)$.

Weights of the unlabeled samples. Found using a kernel density estimator, normalized with (3.13) using $\eta \in [4, 6]$. The semi-supervised smoothness assumption can be turned off by setting all $\beta_i$ to a constant. This simplifies the implementation and computations but may not be suitable depending on the underlying structure of the data.

The percent of most confident unlabeled samples to add to the initial training set. Chen and Wang (2011) recommend $\rho \in [10, 15]$. In Section 4.2.1, we saw that an initial training set was successfully constructed with only 1 mislabeled unlabeled sample among 15 000.

Table 4.7: RegBoost parameters.
R | The number of rounds to repeat the algorithm. As R is increased, the algorithm adds more unlabeled samples, which leads to increased computation time and may result in decreased accuracy in the pseudo-labels, since only samples that are hard to classify are left in the end. We have investigated $R = 10$ and $R = 20$.

K | The number of nearest neighbors to use in the computation of the similarity measure. When the mean distance similarity measure (3.21) is used, this value can not be larger than the amount of labeled samples in any of the classes.

$n_B$ | The number of bins in the histograms used to estimate the distributions of (3.22d) and (3.23). As more bins are used, the accuracy of the estimated distributions are increased, but discontinuities may arise if the validation set is not sufficiently large. The problems of discontinuities can be overcome by smoothing the histograms using a kernel density estimator, but this is not further investigated in this thesis. We use $n_B = 1000$ in all our experiments.

| Table 4.8: Self-learning parameters. |
In this thesis, we have implemented three semi-supervised learning algorithms and applied them to a large-scale pedestrian detection problem. Along with the derivation of the algorithms, we illustrated how they work using a simple and intuitive example of binary classification. The small-scale examples showed that the semi-supervised learning algorithms could improve the classification model of a supervised classifier by utilizing information in an additional set of unlabeled samples.

In Chapter 4, we applied the SSL algorithms to larger data sets with higher input dimension. Here, we saw that the success of the SSL algorithms that was seen in the small-scale examples did not generalize to higher dimensions. We found that it is important to find a set of suitable parameters in order to obtain successful results. To find appropriate parameters, we searched the parameter space using 10-fold cross validation, a procedure that is time consuming for large data sets. Due to the time limitation of this project, we could not perform an exhaustive search for parameters on the pedestrian data set. The results presented on the pedestrian data set prove that the use of SSL algorithms can decrease the performance compared to equivalent supervised methods. We cannot say if the decrease is caused by inappropriate parameter values or the high dimension of the input data.

Among the investigated algorithms, we found the self-learning algorithm to be the most intuitive, the easiest to implement and also, as proven by the presented results, the algorithm with the best performance. In the case study of RegBoost on the pedestrian data set in Section 4.2.1, we saw that the radial basis function similarity measure based on Euclidean distances between HOG features was able to construct a good initial training set. Unfortunately, RegBoost is sensitive to label switches in the pseudo-labels of the neighboring samples. We saw that incorrect pseudo-labels spread fast among the unlabeled samples during the boosting loop.
We also performed a case study of the self-learning algorithm on the pedestrian data set in Section 4.2.2. Only a small amount of unlabeled samples were drawn with incorrect pseudo-labels, but we did not see any significant increase in performance. We also presented results indicating that it is hard to find classifiers that generalize well to new data.

5.1 Future work

The hardest problem of semi-supervised learning is to accurately assign pseudo-labels to the unlabeled samples. In order to improve the pseudo-labels, most algorithms utilize a similarity measure to complement the classifier confidence. The pair-wise Euclidean distance similarity measure that we used is hard to interpret in higher dimensions and both time and memory consuming to compute. Since finding a proper similarity measure is essential in order to assign correct pseudo-labels, we suggest this to be an area of future work.

In the case study of RegBoost, we saw that the pseudo-labels assigned during the boosting loop were sensitive to label switches in the pseudo-labels of the neighboring samples. We find it interesting to investigate if using soft pseudo-labels could reduce this sensitivity.

A potential drawback of the self-learning algorithm is that the pseudo-labeled samples are used with the same confidence as the originally labeled samples. If the pseudo-labels are incorrect, this will distort the classification model and can lead to decreased accuracy. If the added unlabeled samples were handled separately and monitored throughout the training procedure, incorrectly pseudo-labeled samples might be found and discarded as the classification model has been improved, and thereby preventing further distortion. This is an area we leave for future work.

Object detection of a specific class is in practical situations often a highly skewed problem. This is usually solved by using a cascade of classifiers, where many negative samples can be rejected in the early stages. It is not yet investigated how the probability distribution of the unlabeled samples affect the problem in a semi-supervised setting. Since there in practice is a high probability that there will be more negative samples than positive samples among the unlabeled samples, we suggest this to be an area for further research.
Appendix
In this appendix, we present the parameters used in examples and experiments throughout the thesis, see Tables A.1 to A.3. A simple intuition of the parameters is given in Section 4.3.3. The number of samples in the available training set $S = L \cup U$ is denoted $n_{\text{tot}}$.

<table>
<thead>
<tr>
<th>Data set</th>
<th>$n_{\text{tot}}$</th>
<th>$l$</th>
<th>$l'$</th>
<th>$K_N$</th>
<th>$C$</th>
<th>$\rho$</th>
<th>$T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example 3.2</td>
<td>200</td>
<td>10%</td>
<td>10%</td>
<td>All</td>
<td>0.11</td>
<td>10</td>
<td>400</td>
</tr>
<tr>
<td>AUS</td>
<td>345</td>
<td>10%</td>
<td>5%</td>
<td>All</td>
<td>0.11</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>BANK</td>
<td>4640</td>
<td>10%</td>
<td>10%</td>
<td>All</td>
<td>0.11</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>ION</td>
<td>176</td>
<td>10%</td>
<td>25%</td>
<td>All</td>
<td>0.11</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>WDBC</td>
<td>285</td>
<td>10%</td>
<td>25%</td>
<td>All</td>
<td>0.11</td>
<td>30</td>
<td>100</td>
</tr>
<tr>
<td>Pedestrian</td>
<td>100000</td>
<td>20%</td>
<td>20%</td>
<td>20</td>
<td>0.25</td>
<td>10</td>
<td>300</td>
</tr>
</tbody>
</table>

*Table A.1*: SemiBoost parameters used in the experiments in the thesis.
<table>
<thead>
<tr>
<th>Data set</th>
<th>(n_{\text{tot}})</th>
<th>(l)</th>
<th>(\Gamma)</th>
<th>(K_N)</th>
<th>(K)</th>
<th>(h)</th>
<th>(\gamma_i)</th>
<th>(\rho)</th>
<th>(T)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example 3.3</td>
<td>200</td>
<td>4%</td>
<td>25%</td>
<td>All</td>
<td>7</td>
<td>0.75</td>
<td>5</td>
<td>15</td>
<td>400</td>
</tr>
<tr>
<td>AUS</td>
<td>345</td>
<td>10%</td>
<td>15%</td>
<td>All</td>
<td>4</td>
<td>200</td>
<td>5</td>
<td>15</td>
<td>100</td>
</tr>
<tr>
<td>BANK</td>
<td>4640</td>
<td>10%</td>
<td>20%</td>
<td>All</td>
<td>20</td>
<td>1</td>
<td>5</td>
<td>15</td>
<td>100</td>
</tr>
<tr>
<td>ION</td>
<td>176</td>
<td>10%</td>
<td>10%</td>
<td>All</td>
<td>7</td>
<td>10</td>
<td>5</td>
<td>15</td>
<td>100</td>
</tr>
<tr>
<td>WDBC</td>
<td>285</td>
<td>10%</td>
<td>15%</td>
<td>All</td>
<td>3</td>
<td>10</td>
<td>5</td>
<td>15</td>
<td>100</td>
</tr>
<tr>
<td>Pedestrian</td>
<td>100000</td>
<td>20%</td>
<td>25%</td>
<td>20</td>
<td>20</td>
<td>-</td>
<td>4</td>
<td>15</td>
<td>300</td>
</tr>
</tbody>
</table>

Table A.2: RegBoost parameters used in the experiments. On the pedestrian data set, the parameter \(h\) is not applicable since we turn off the semi-supervised smoothness assumption and set all \(\beta_i = 1.25 \cdot 10^{-5}\).

<table>
<thead>
<tr>
<th>Data set</th>
<th>(n_{\text{tot}})</th>
<th>(l)</th>
<th>(\Gamma)</th>
<th>(K_N)</th>
<th>(K)</th>
<th>(R)</th>
<th>(n_B)</th>
<th>(T)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example 3.4</td>
<td>200</td>
<td>10%</td>
<td>2.5%</td>
<td>All</td>
<td>2</td>
<td>10</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>AUS</td>
<td>345</td>
<td>10%</td>
<td>5%</td>
<td>All</td>
<td>20</td>
<td>20</td>
<td>1000</td>
<td>100</td>
</tr>
<tr>
<td>BANK</td>
<td>4640</td>
<td>10%</td>
<td>1%</td>
<td>All</td>
<td>20</td>
<td>10</td>
<td>1000</td>
<td>100</td>
</tr>
<tr>
<td>ION</td>
<td>176</td>
<td>10%</td>
<td>5%</td>
<td>All</td>
<td>2</td>
<td>20</td>
<td>1000</td>
<td>100</td>
</tr>
<tr>
<td>WDBC</td>
<td>285</td>
<td>10%</td>
<td>2.5%</td>
<td>All</td>
<td>2</td>
<td>20</td>
<td>1000</td>
<td>100</td>
</tr>
<tr>
<td>Pedestrian</td>
<td>100000</td>
<td>20%</td>
<td>2.5%</td>
<td>20</td>
<td>20</td>
<td>10</td>
<td>1000</td>
<td>300</td>
</tr>
</tbody>
</table>

Table A.3: Self-learning parameters used in the experiments.
Bibliography


Upphovsrätt

Detta dokument hålls tillgängligt på Internet — eller dess framtida ersättare — under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för icke-kommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

Copyright

The publishers will keep this document online on the Internet — or its possible replacement — for a period of 25 years from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for his/her own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: http://www.ep.liu.se/

© Mikael Rosell