Dissertation
submitted to the
Combined Faculties for the Natural Sciences and for Mathematics
of the Ruperto-Carola University of Heidelberg, Germany
for the degree of
Doctor of Natural Sciences

put forward by
Dipl.-Phys. Matthias Wieler
born in Mannheim, Germany

Oral examination: July 23, 2014

# Multiple Instance Learning with Random Forests and Applications in Industrial Optical Inspection

Referees: Prof. Dr. Fred A. Hamprecht
Prof. Dr. Luca Amendola

Hiermit erkläre ich, da ich die vorliegende Dissertation selbst verfasst und mich dabei keiner anderen als der von mir ausdrücklich bezeichneten Quellen and Hilfen bedient habe. Des Weiteren erkläre ich, dass ich an keiner anderen Stelle ein Prüfungsverfahren beantragt oder die Dissertation in dieser oder einer anderen Form bereits anderweitig als Prüfungsarbeit verwendet oder einer anderen Fakultät als Dissertation vorgelegt habe.

Heidelberg, _____ _____

Matthias Wieler

**Zusammenfassung**

Für die automatische Defekterkennung in der industriellen optischen Inspektion werden Algorithmen benötigt, die aus Daten lernen. Eine besondere Herausforderung sind Daten mit unvollständigen Labels. Eine der Methoden, die das Feld des maschinellen Lernens hervorgebrachte hat um mit unvollständigen Labels umzugehen, ist das sog. Multiple Instance Learning. Ein Merkmal dieses Ansatzes ist, dass dabei die Datenpunkte (Instanzen) zu sog. Bags zusammenfasst werden.

Wir schlagen eine neue Methode zur Berechnung der Bag-Wahrscheinlichkeiten aus den Instanz-Wahrscheinlichkeiten vor, die den Vorteil hat, dass die Ergebnisse nicht von der Größe der Bags abhängig sind. Weiterhin schlagen wir eine Erweiterung des Multiple Instance Modells vor, das es dem Benutzer erlaubt, die Anzahl der als positiv klassifizierten Instanzen zu steuern.

Wir implementieren diese Methoden mit einem Algorithmus, der auf dem wohlbekannten Random Forest-Klassifikator aufbaut. Der Algorithmus zeigt auf einem bekannten Benchmark-Datensatz eine konkurrenzfähige Klassifikationsleistung. Wir wenden diesen Algorithmus auf Bilddaten an, die die Besonderheiten der industriellen optischen Inspektion widerspiegeln, und zeigen, dass der Algorithmus in diesem Szenario den normalen Random Forest übertrifft.

**Abstract**

Automatic defect detection in industrial optical inspection requires algorithms that can learn from data. A special challenge is data with incomplete labels. One of the methods that the field of machine learning has brought forth to deal with incomplete labels is multiple instance learning. One trait of this setting is that it groups datapoints (instances) into bags.

We propose a novel method to predict bag probabilities from given instance probabilities that has the advantage that its results do not depend on bag size. Also, we propose an extension of the multiple instance model that allows the user to steer the number of instances that are classified as positive.

We implement these methods with an algorithm based on the well-known random forest classifier. Results on a standard benchmark dataset show competitive performance. Furthermore, we apply this algorithm to image data that reflects the challenges of industrial optical inspection, and we show that in this setting it improves over the standard random forest.

# Acknowledgments

I am indebted to my supervisor Prof. Fred A. Hamprecht. Without him I would probably not have discovered the exciting field of machine learning. He did not spare any effort to give me scientific advice. I am grateful for his unlimited support both in good times and in bad times.

I would also like to thank all my colleagues in the multidimensional image processing group, who gave me valuable scientific input, in particular Ullrich Köthe, Nikos Gianniotis, Björn Andres, Michael Hanselmann, Melih Kandemir, Frederik Kaster, Christoph Sommer, Bernhard Kausler, Marc Kirchner, and Xinghua Lou.

This work has been supported by Robert Bosch GmbH. I would like to cordially thank the persons in charge for making this possible. Special thanks go to Walter Happold who gave me all support I could wish for.

I would like to thank my advisers at Bosch, Christian Perwass and Ralf Zink, for many valuable discussions. I have profited from their advice.

Working on this thesis would have been less joyful without my colleagues at Bosch. In particular, I wish to thank Jens Röder for the numerous scientific discussions and personal conversations we have had. Cordial thanks go to my fellow doctorate students, namely Andreas Walstra, Patrick Sauer, Joachim Börger, Andreas Grützmann, Thomas Geiler, Marc Jäger, Linus Görlitz, and Stefan Trittler.

Also I would like to thank Prof. Luca Amendola, Prof. Rüdiger Klingeler, and Prof. Bernd Jähne for serving on my committee.

Deep thanks to my parents and my sister for their love and support.

# Contents

# Chapter 1

# Introduction

The starting point of this work has been the application of industrial optical inspection. Optical inspection is a widespread method to ensure the quality of components or devices directly after production. As in many areas, there is a need for automation to reduce cost and increase reliability. However, one drawback of automated optical inspection is the loss of flexibility compared to human visual inspection.

Recent advances in the field of machine learning make algorithms available that can learn from examples. These methods promise improved flexibility because it is easier to retrain a machine learning algorithm than to adapt an image processing algorithm to new requirements or boundary conditions.

The subject of this thesis is multiple instance learning, which is a machine learning setting that corresponds to the special requirements of industrial optical inspection. But before going into the details of multiple instance learning, we give an introduction of industrial optical inspection and image processing and describe the role of machine learning in this setting. Also, we present a datasets that we have used to assess our multiple instance learning methods for the target application of industrial optical inspection.

## 1.1 Industrial optical inspection

Industrial optical inspection is a method of quality control. The production process of technical devices usually involves many steps and can be quite complex. Even when great care is taken, it is often not possible to completely rule out that an error occurs. To make sure that the final product is free of defects, it is usually necessary to perform several inspections and/or tests.

Optical inspection is a suitable method in many cases because it is fast and very versatile. Often, it can be used to detect several different kinds of defects with a single inspection. Optical inspection can either be performed by humans, or it can be an automated system.

**Human visual inspection** Even in highly automated production lines human visual inspection is still common. The reason is that the flexibility and versatility of human visual inspection is very hard to achieve with an automated system (Kleeven & Hyvärinen 1999). Humans have an extremely good image understanding and can detect a wide variety of possible defects with little or even no training or instructions.

The main disadvantage of human visual inspection (besides cost) is the subjectivity and varying quality of human assessment (Schoonard, Gould & Miller 1973). Different inspectors often have different opinions as to whether a given component is to be classified as intact or as defective. Even a single inspector's assessment can be subject to fluctuation, caused by tiredness or other factors.

**Automated optical inspection** Automated optical inspection usually includes a handling device to position the components, special lighting, a camera with suitable optics and filters, and an image processing system. Handling and image acquisition can typically be performed within a few seconds or less, which allows for inspection of a component within the cycle time of serial production. Suitable lighting, optics, and filters are often essential to make the defects of interest clearly visible. A general rule is that the more effort is spent on image acquisition the less effort has to be spent later on image processing. For a more detailed introduction see (Demant, Streicher-Abel & Springhoff 2011) or (Beyerer, León & Frese 2012).

## 1.2 Image processing

Image processing deals with the problem of finding meaningful descriptions of an image from the raw data (matrix of gray values). This task can be divided into three steps: preprocessing, feature extraction, and image analysis (Jähne 2012). Preprocessing includes operations on single pixels like color adjustment or interpolation. The goal of feature extraction is to calculate local features from a small neighborhood of pixels. These features describe basic elements of the image like edges, ridges, corners, etc. Image analysis, finally, tries to provide a good high-level description of the image based on local features.

In industrial optical inspection, the final goal is to classify an image as either "intact" or "defective". Sometimes it is also required to specify which type of defect has occurred.

The third step of image analysis and classification can either by performed with classical image processing or with machine learning methods.

**Classical image analysis** Classical image analysis is state of the art in industrial optical inspection. It includes operations like image segmentation, morphological operations, quantitative characterization of the segments, inverse filtering, and others. While most problems can be solved with this approach, the solutions are very application-specific. It is necessary to develop and optimize the algorithm specifically for each application and type of defect, which often involves careful tuning of many parameters, setting decision thresholds, etc.

In industrial optical inspection, the requirements and boundary conditions for image classification change regularly because of variations in the production process, changing customer demands, or because new types of defects arise. This requires regular adjustment of the image analysis algorithm which causes considerable effort. To reduce the cost of adaptation, a different approach based on machine learning is necessary.

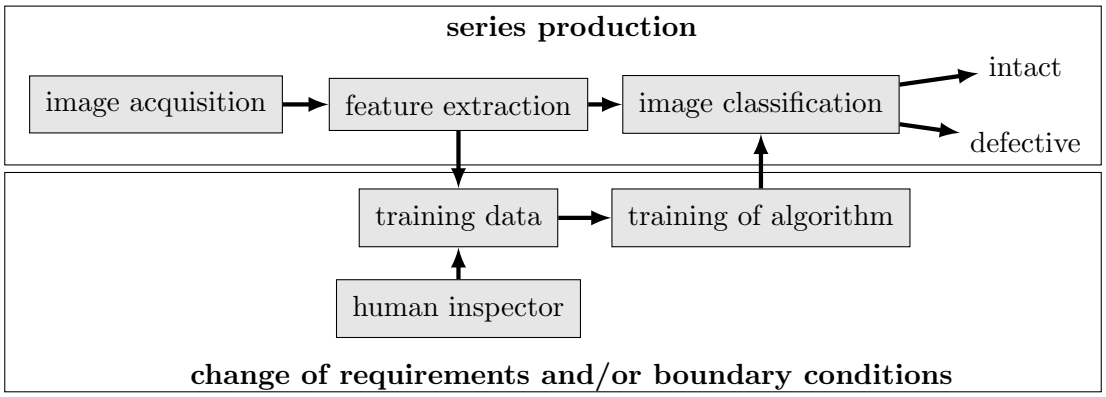


Figure 1.1: Overview of applying machine learning in industrial optical inspection.

## 1.3 Machine learning

Machine learning concerns the development of algorithms that can learn from examples. It has the potential to reduce the effort of customization and adaption of image processing systems significantly.

The setting of machine learning in industrial optical inspection is illustrated in Figure 1.1. It includes both the continually running series production (upper box), and the occasional adjustment to new requirements and boundary conditions (lower box). In series production, an important requirement on the inspection system is speed. Feature extraction and image classification must be executed within the fixed cycle time. The main concern of system adaptation (lower box) is to ensure the accuracy of image classification (i. e. the reliability of inspection) and to minimize the amount of human work.

The core subject of machine learning (and of this thesis) is algorithm training and image classification (rightmost text boxes). For the application in industrial optical inspection and to test the performance of the algorithms, we also have to consider feature extraction and training data.

**Training data** Training a machine learning algorithm requires a training dataset that consists of (i) example images of intact and defective components and (ii) a label for each image describing whether it is "intact" or "defective". The example images can usually be acquired easily with the available image acquisition device, but the labels must be provided by a human inspector.

Note that labeled example images are needed both for machine learning and for classical image analysis. However, machine learning is more dependent on the amount and quality of the training data than classical image analysis. A human image processing expert can work with unstructured and incomplete data, and few example images are often sufficient. Machine learning, on the contrary, requires a complete dataset in a fixed format, and more

example images are needed to achieve good accuracy.

The increased effort of acquiring training data mitigates the benefit of automated training. On the other hand, it increases objectivity and quantifiability. In machine learning, the training data and accuracy estimates are documented as a matter of course, and the result can be reproduced and checked at any time. This is not necessarily the case in classical image analysis, where there is less emphasis on training data and statistical estimates of prediction accuracy.

## 1.4 Weak labels and multiple instance learning

To keep labeling effort within acceptable limits, it is not possible to label each defect exactly on the pixel-level. Instead, we have to use labels on either the image-level (image is labeled as a whole) or on the region-level (the image region containing the defect is specified). In both cases the regions labeled as "defective" contain (besides the defect) a considerable percentage of the "intact" image. In this sense the provided labels are "weak".

The central topic of this thesis is to develop machine learning algorithms that can deal with this kind of weak labels. Although it is possible to use standard supervised machine learning methods with weak labels, the mislabeled datapoints (labeled "defective" although they are in fact "intact") impair classification accuracy. We have found that the so-called "multiple instance" setting is a suitable model for this task. While this model has originally been proposed for a different application (see Chapter 3), it can also be used for weakly labeled images. However, one can improve over the multiple instance model by using additional information about the size of the defect (i. e. the number of truly "defective" pixels), which will be the topic of Chapter 5.

## 1.5 DAGM datasets

To assess our proposed machine learning methods for the target application of industrial optical inspection, we have created an artificially generated benchmark dataset. It has been designed to imitate real world problems of industrial optical inspection. The data has been published[1] at the DAGM[2] symposium.

The DAGM data consists of 10 different datasets, each consisting of 1000 images showing the background texture without defects, and 150 images showing the background texture with one defect. The images in a single dataset are very similar, but each dataset is generated by a different texture model and defect model. The images are 8-bit grayscale with size 512-by-512. Example images are shown in Figure 1.2.

The defects are labeled by ellipses covering the defects. These labels are not exact on a pixel-level, but are weak in the sense described above. Thus the DAGM datasets are well suited to test the multiple instance learning algorithms proposed in this thesis.

[1] http://hci.iwr.uni-heidelberg.de/Staff/dagm2007/prizes.php3#industry

[2] Deutsche Arbeitsgemeinschaft für Mustererkennung e.V. (German Association for Pattern Recognition)

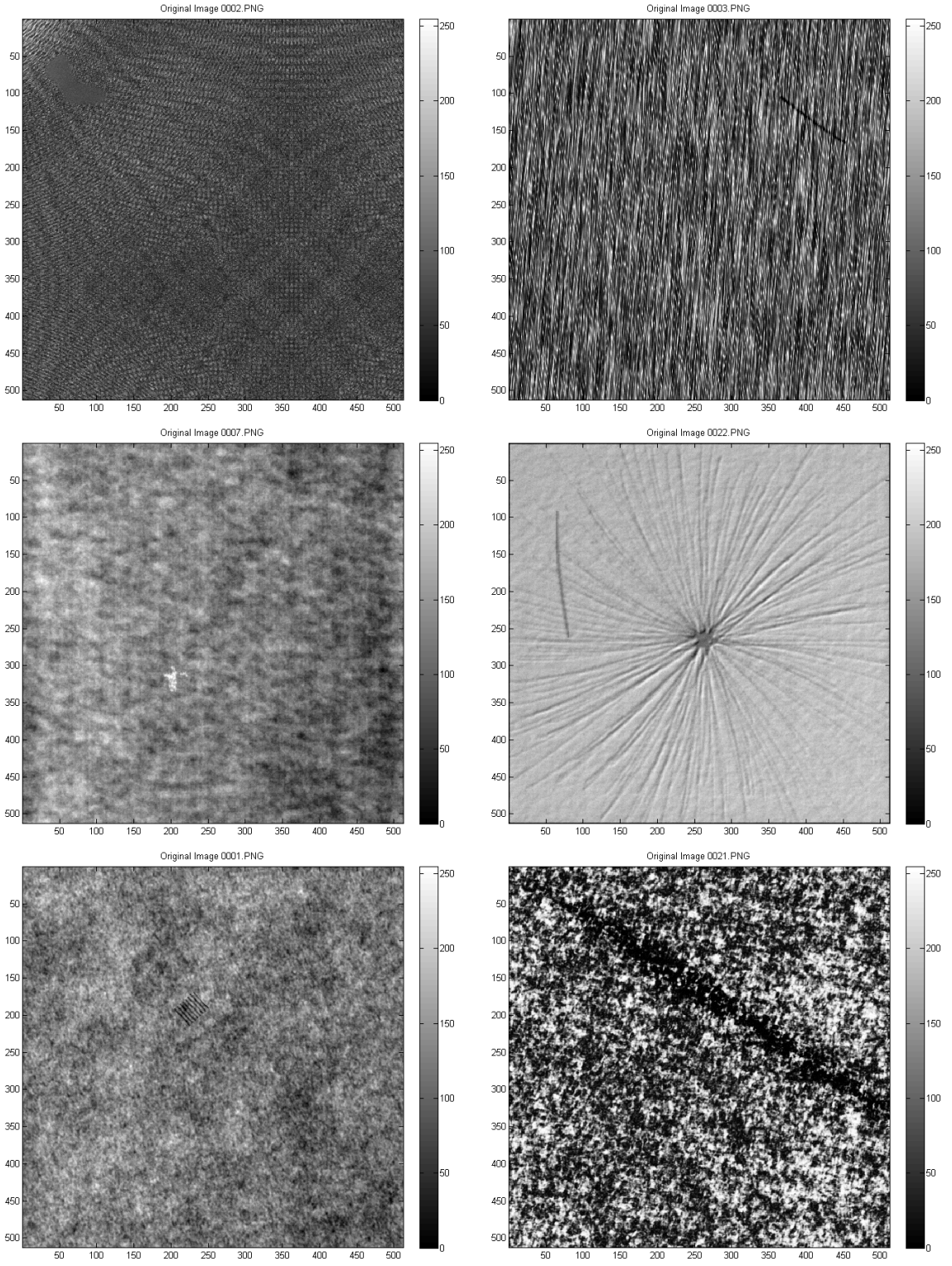


Figure 1.2: Example images of DAGM datasets. One defective image each of the first six datasets are shown.

**Image features used for DAGM datasets** Before image classification we need to extract local image features. Based on the work of (Sauer 2008), we chose a two-step approach to feature calculation. The first step consists of a wavelet filter bank, the second step consists of statistics calculated over small image patches.

For the first step we chose the so-called a "steerable pyramid" proposed by (Simoncelli & Freeman 1995, Karasaridis & Simoncelli 1996). This is a wavelet filter bank that comprises a set of oriented bandpass filters, yielding a radial-axial decomposition of frequency domain. We chose a resolution of 4 different scales and 6 different orientations, which gives a total of 24 different feature images.

For the second step we used patches of size 32-by-32 pixels with an overlap of 16 pixels (giving a total of 32-by-32 patches per image). Over each patch we calculated the 5 following statistics: minimum, maximum, mean, variance, and kurtosis. Note that the steerable pyramid outputs images of different resolution depending on the scale used, so the statistics are calculated over a different number of pixels of the pyramid result.

In addition to the above 120 features we used the position of the patch in the image ($x$- and $y$-coordinates) as two additional features. These features are useful for classification if the defects are located mainly in one region of the image (Class 7) or if the appearance of the defects depend on the location (Class 4).

# Chapter 2

# Bayesian Learning Theory

This chapter gives an overview of statistical theory for machine learning from a Bayesian viewpoint. The goal is to provide the theoretical foundation needed in later chapters.

We focus on the topics of semi-supervised learning and bag models (of which the multiple instance model is a special case), generative vs. discriminative learning (which is a prerequisite for the bag-size-independent MI model of Section 4.1), self-training, and bagging (which are needed for our proposed learning algorithm in Section 6). To obtain a coherent text, we found it necessary to also include some general topics.

The emphasis of this chapter is slightly more theoretical than standard expositions of machine learning. Of the well-known text books, our treatment is closest to (Bishop 2006).

## 2.1 The general setting

Machine learning is about the relation between an input $\boldsymbol{x}$ and an output $y$. For given inputs $\boldsymbol{x}$ one would like to predict the corresponding outputs $y$. The problem is that the relation itself is unknown. The only available information are examples of corresponding input-output pairs $(\boldsymbol{x}_n, y_n)$, the *training data*.

The input $\boldsymbol{x}$ is usually continuous and high-dimensional, the output $y$ is most often binary (positive or negative), but sometimes also categorical (multiclass learning) or continuous (regression).

Formally, the relation between $\boldsymbol{x}$ and $y$ can most generally be described by a probability distribution

$$P_{\text{true}}(\boldsymbol{x}, y). \tag{2.1}$$

Since this true distribution is unknown, one tries to model it with a set of candidate distributions, that are described by parameters $\boldsymbol{\theta}$

$$P(\boldsymbol{x}, y \,|\, \boldsymbol{\theta}). \tag{2.2}$$

A specific distribution (represented by its parameter vector $\boldsymbol{\theta}$) is called a *concept* or an *hypothesis*, and the parameter space $\Theta$ defines a set of possible concepts or an *hypothesis space*. One usually assumes that there is one true parameter vector $\boldsymbol{\theta}_{\text{true}}$ that correctly

describes the underlying distribution

$$P_{\text{true}}(\boldsymbol{x}, y) \quad = \quad P(\boldsymbol{x}, y \,|\, \boldsymbol{\theta}_{\text{true}}). \tag{2.3}$$

Since the true parameter vector is unknown, we have to consider a probability distribution over the parameters $P(\boldsymbol{\theta})$. This parameter prior is often taken as uniform, but one can also define a non-uniform parameter prior that makes certain concepts more likely than others. So most generally, a probabilistic learning model is described by

$$P(\boldsymbol{x}, y, \boldsymbol{\theta}) \quad = \quad P(\boldsymbol{x}, y \,|\, \boldsymbol{\theta}) \cdot P(\boldsymbol{\theta}). \tag{2.4}$$

This model can be represented by the Markov network shown in Figure 2.1. All datapoints are linked by the common concept $\boldsymbol{\theta}$. The training data points are grouped on the left side, the test points on the right side. Index $n$ runs from 1 to $N$, index $m$ runs from $N+1$ to $N+M$. The task is to infer the test outputs $y_m$ from the test inputs $\boldsymbol{x}_m$ and the training data $\mathcal{D} = \{\boldsymbol{x}_n, y_n\}$ via the latent parameters $\boldsymbol{\theta}$.

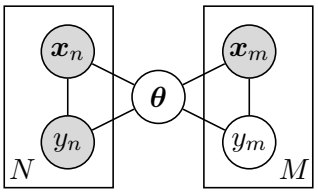


Figure 2.1: Markov network of supervised learning. Left side: $N$ training data points. Right side: $M$ test points. Shaded nodes represent observed variables, unshaded nodes must be inferred.

Besides this data-centered viewpoint, it is useful to consider the complete distribution of $\boldsymbol{x}$ and $y$, which is a superposition of the candidate distributions, weighted by the parameter distribution

$$P(\boldsymbol{x}, y) \quad = \quad \int P(\boldsymbol{x}, y \,|\, \boldsymbol{\theta}) P(\boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{\theta}. \tag{2.5}$$

This distribution can be seen as the intermediate result Note that we should require the prior $(\boldsymbol{x}, y)$-distribution to be uniform

$$P(\boldsymbol{x}, y) \quad = \quad U(\boldsymbol{x}, y), \tag{2.6}$$

otherwise the learning model (2.4) would be prejudiced. This requirement is met by standard learning techniques, but it is usually not stated explicitly.

After having observed the data $\mathcal{D}$, the prior (2.5) is replaced by the posterior

$$P(\boldsymbol{x}, y \,|\, \mathcal{D}) \quad = \quad \int P(\boldsymbol{x}, y \,|\, \boldsymbol{\theta}) \; P(\boldsymbol{\theta} \,|\, \mathcal{D}) \; \mathrm{d}\boldsymbol{\theta}. \tag{2.7}$$

Note that the posterior distribution (2.7) is not necessarily contained in the set of candidate distributions (2.2), so one could say that Bayesian inference "increases the flexibility" of prediction (Minka 2000).

**Train-test split and transduction** Note that the symbol $\mathcal{D}$ in (2.7) stands for *all* data, i. e. both training data and test data

$$\mathcal{D} \;=\; \mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{test}} \;=\; \{\boldsymbol{x}_n, y_n\} \cup \{\boldsymbol{x}_m\}. \tag{2.8}$$

In fact, the posterior parameter distribution depends in general not only on the training data but also on the test data.

$$P(\boldsymbol{\theta} \,|\, \mathcal{D}) \;=\; \frac{1}{Z} \underbrace{P(\boldsymbol{\theta})}_{\text{prior}} \underbrace{\prod_n P(\boldsymbol{x}_n, y_n \,|\, \boldsymbol{\theta})}_{\text{training likelihood}} \underbrace{\prod_m P(\boldsymbol{x}_m \,|\, \boldsymbol{\theta})}_{\text{test likelihood}}. \tag{2.9}$$

$$Z \;=\; P(\mathcal{D}) \;=\; \int \prod_n P(\boldsymbol{x}_n, y_n \,|\, \boldsymbol{\theta}) \prod_m P(\boldsymbol{x}_m \,|\, \boldsymbol{\theta}) \,\mathrm{d}\boldsymbol{\theta} \tag{2.10}$$

The test likelihood is usually neglected. In most cases this is does not cause a significant difference, because the information contained in the training data usually outweighs the information contained in the test data (i. e. the training likelihood has a sharper peak than the test likelihood).

The benefit of neglecting the test likelihood is that it simplifies inference by effectively splitting it in two subsequent steps: training and testing. During training one infers the posterior parameter distribution $P(\boldsymbol{\theta} \,|\, \mathcal{D}_{\text{train}})$ by taking into account the training data, but neglecting the test data. During testing one holds the parameter distribution constant and evaluates (2.11). This setting is shown in Figure 2.2.

The fact that the test data contain potentially useful information does not seem to be generally appreciated. In the context of semi-supervised learning it is more obvious, because unlabeled training points are actually the same as test points ($\boldsymbol{x}$ is observed, $y$ is not). In this context, the idea of using the test data in addition to training data to improve prediction is known as *transduction*. We will discuss this in more detail in Section 2.4.

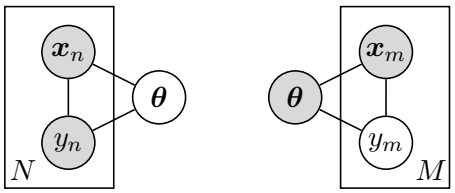


Figure 2.2: Markov networks of training stage (left side) and testing stage (right side). Note that during testing, $\boldsymbol{\theta}$ does not actually have a fixed value (which would be the usually meaning of a shaded node), but a fixed distribution. Nevertheless we find that the shaded node expresses the idea clearly enough.

## 2.2 Inference

In this section we discuss the main inference step of evaluating (2.7). As is common, we incorporate the train-test split and consider only the training data. Then, (2.7) and (2.9)

are replaced by

$$P(\boldsymbol{x}, y \,|\, \mathcal{D}_{\text{train}}) = \int P(\boldsymbol{x}, y \,|\, \boldsymbol{\theta})\, P(\boldsymbol{\theta} \,|\, \mathcal{D}_{\text{train}})\, \mathrm{d}\boldsymbol{\theta} \tag{2.11}$$

$$P(\boldsymbol{\theta} \,|\, \mathcal{D}_{\text{train}}) \propto P(\boldsymbol{\theta})\, \mathcal{L}(\boldsymbol{\theta}) \tag{2.12}$$

$$\mathcal{L}(\boldsymbol{\theta}) = \prod_n P(\boldsymbol{x}_n, y_n \,|\, \boldsymbol{\theta}) \tag{2.13}$$

The integration over $\boldsymbol{\theta}$ is usually infeasible because the parameter space is high-dimensional (it must be to allow for enough flexibility of the candidate distributions).

There are several approaches to find reasonable approximations for (2.11). They all have in common that they approximate (2.11) by a weighted sum.

$$P(\boldsymbol{x}, y \,|\, \mathcal{D}_{\text{train}}) \approx \sum_t w_t P(\boldsymbol{x}, y \,|\, \boldsymbol{\theta}_t) \tag{2.14}$$

The differences between these methods is in how they choose the support points $\{\boldsymbol{\theta}_t\}$, and how they estimate the weights $w_t$. Note that in general $w_t \neq P(\boldsymbol{\theta}_t \,|\, \mathcal{D}_{\text{train}})$.

Methods that learn multiple concepts $\boldsymbol{\theta}_t$ are often called *ensemble methods* or *committees*. They include Bayesian averaging, bagging, boosting, and others. But also point estimation methods like maximum posterior and maximum likelihood can be thought of as special cases of (2.14), with only one support point.

### 2.2.1 Maximum posterior and maximum likelihood

Both the maximum posterior and the maximum likelihood approach are point estimation methods, which means that they learn only one concept and have only one support point $\hat{\boldsymbol{\theta}}$ with weight 1. In this case the approximation (2.14) simplifies to

$$P(\boldsymbol{x}, y \,|\, \mathcal{D}_{\text{train}}) \approx P(\boldsymbol{x}, y \,|\, \hat{\boldsymbol{\theta}}). \tag{2.15}$$

It is obvious that the best choice of $\hat{\boldsymbol{\theta}}$ is the maximum of the parameter posterior

$$\hat{\boldsymbol{\theta}}_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} P(\boldsymbol{\theta} \,|\, \mathcal{D}). \tag{2.16}$$

To put it into words, the procedure is that one first tries to find the concept $\hat{\boldsymbol{\theta}}$ within the model that best explains the training data. Then this concept is used to predict the test class.

Another common choice of $\hat{\boldsymbol{\theta}}$ is the maximum likelihood estimate

$$\hat{\boldsymbol{\theta}}_{\text{ML}} = \arg\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \tag{2.17}$$

Although maximum likelihood is not originally a Bayesian method, it is quite natural to view it as an approximation of Bayesian inference as described above. Note that if the

parameter prior is uniform $P(\boldsymbol{\theta}) = U(\boldsymbol{\theta})$, the maximum posterior estimate (2.16) equals the maximum likelihood estimate (2.17). Often one does not have cogent prior information about the model parameters, so one chooses a more or less flat prior. In this case the maximum of the parameter posterior is dominated by the likelihood.

Note that both from a conceptual and a practical point of view there is big difference between the Bayesian approach (2.11) and the maximum posterior or maximum likelihood approach. The Bayesian approach puts the emphasis on integration, while the maximum posterior/likelihood puts the emphasis on optimization. This entails very different tools and techniques.

### 2.2.2 Bayesian averaging

The goal of Bayesian averaging is to approximate the full Bayesian integral (2.11). This can be done with the averaging approach (2.14) by setting

$$\boldsymbol{\theta}_t \sim P(\boldsymbol{\theta} \,|\, \mathcal{D}_{\text{train}}) \tag{2.18}$$

$$w_t = P(\boldsymbol{\theta}_t \,|\, \mathcal{D}_{\text{train}}) \tag{2.19}$$

The difficulty of this approach is the large computational cost. Especially, sampling from the posterior parameter distribution $P(\boldsymbol{\theta} \,|\, \mathcal{D}_{\text{train}})$ is hard, because $\boldsymbol{\theta}$ is high-dimensional, and $P(\boldsymbol{\theta} \,|\, \mathcal{D}_{\text{train}})$ is usually complex and multimodal.

### 2.2.3 Bagging

"Bagging" is an abbreviation of "bootstrap aggregating". It has been proposed by (Breiman 1996). The idea is to repeatedly draw bootstrap samples from the training data (sampling with replacement), and then train a classifier on each set of bootstrapped training data. The final classifier is the average (or majority vote) of all single classifiers.

Formally, a bagged classifier is described by Eq. (2.14) with concepts

$$\boldsymbol{\theta}_t = \arg\max_{\boldsymbol{\theta}} \; P(\boldsymbol{\theta} \,|\, \mathcal{D}_t) \tag{2.20}$$

where $\mathcal{D}_t$ is the $t$-th set of bootstrapped samples, and uniform weights

$$w_t = \frac{1}{T}. \tag{2.21}$$

The idea behind bagging is that taking the average over different datasets reduces the mean squared error of the prediction. It can easily be shown (Breiman 1996) that

$$E_{\mathcal{D}}(y - \hat{y}_{\mathcal{D}})^2 \geq (y - E_{\mathcal{D}}(\hat{y}_{\mathcal{D}}))^2, \tag{2.22}$$

where $E_{\mathcal{D}}$ denotes expectation over all possible datasets, $y$ is the true class, and $\hat{y}_{\mathcal{D}}$ is the estimate from dataset $\mathcal{D}$. Thus the average $E_{\mathcal{D}}(\hat{y}_{\mathcal{D}})$ improves over the point estimate $\hat{y}_{\mathcal{D}}$. Bagging is a direct attempt to obtain an estimate of $E_{\mathcal{D}}(\hat{y}_{\mathcal{D}})$.

The problem is that bootstrapped samples $\mathcal{D}_t$ are not drawn from the true data distribution $P(\boldsymbol{x}, y)$, but are resampled from the observed data. Technically speaking, they do not follow the same distribution as the training data, but are distributed according to a mix of delta distributions at the observed datapoints

$$\mathcal{D}_t \not\sim \mathcal{D} \tag{2.23}$$

$$\mathcal{D}_t \sim \frac{1}{T}\sum_n \delta(\boldsymbol{x}_n)\delta(y_n). \tag{2.24}$$

For some special models one can modify the bootstrapping procedure so that the inferred quantities $\boldsymbol{\theta}_t$ and $y_t$ are in fact distributed according to their posteriors $P(\boldsymbol{\theta} \mid \mathcal{D})$, or $P(y \mid \mathcal{D})$, even though the original samples are not (2.23). This is called the "Bayesian bootstrap" (Rubin 1981). For a complex machine learning model, however, this is not possible.

As a consequence of (2.23) we expect that the mean squared error of the bagged estimate is larger than the mean squared error of the hypothetical average of estimators

$$(y - E_{\mathcal{D}_t}(\hat{y}_{\mathcal{D}_t}))^2 \geq (y - E_{\mathcal{D}}(\hat{y}_{\mathcal{D}}))^2. \tag{2.25}$$

It is very difficult to exactly assess the size of the difference between the two sides of this inequality. Bagging works if the bagged estimate's error (left hand side of 2.25) is at least smaller than the point estimate's error (left hand side of 2.22).

**Stability of classifier** A measure to asses the size of inequality in (2.22) is the "stability" or "variability" of the classifier. This is a rather vague concept of how much the parameter estimate $\boldsymbol{\theta}_t$ changes with changing data $\mathcal{D}_t$. A perfectly stable classifier would have the same optimum for all data samples,

$$\boldsymbol{\theta}_s = \boldsymbol{\theta}_t \qquad \forall s, t, \tag{2.26}$$

and equality would hold in (2.22).

While some instability is needed to improve over the base classifier (i. e. inequality of 2.25), it seems plausible that too much instability will probably hurt performance, because the ensemble members trained on the bootstrapped samples deviate too much from the optimum point estimate

$$\boldsymbol{\theta}_t \neq \hat{\boldsymbol{\theta}}_{\mathrm{MAP}}, \tag{2.27}$$

and we expect very large inequality in 2.25.

**Bootstrap sample size** For a given classifier, bagging provides one parameter to control the stability or variability of $\boldsymbol{\theta}_t$: the bootstrap sample size $N_{\mathrm{boot}}$. For very large $N_{\mathrm{boot}}$, the bootstrapped data is just a multiplied replicate of the given data and the bagged classifier

converges to its base (non-bagged) classifier:

$$N_{\text{boot}} \to \infty \qquad \Longrightarrow \qquad \mathcal{D}_t \to \mathcal{D} \qquad \Longrightarrow \qquad \boldsymbol{\theta}_t \to \hat{\boldsymbol{\theta}}_{\text{MAP}} \tag{2.28}$$

This situation corresponds to a perfectly stable classifier. Likewise, a small bootstrap sample size corresponds to an unstable classifier.

Bagging has been shown to give good performance in many empirical settings. The most well-known example of the success of bagging is the so-called "random forest" (Breiman 2001), which is a bagged version of fully grown decision trees. For the random forest a bootstrap sample size equal to the number of available training points is the standard choice that usually gives good results (Hastie, Tibshirani & Friedman 2009). Because of the good reputation of the random forest, it will be the basis for our proposed algorithms for multiple instance learning (see Chapter 6).

## 2.3 Generative and discriminative models

**Basic idea** Up to now we have considered the general case of modeling the joint distribution $P_{\text{true}}(\boldsymbol{x}, y)$ with a single set of parameters $\boldsymbol{\theta}$. It is suggestive to split this joint distribution into a marginal and a conditional, and model the marginal and the conditional separately. There are two obvious ways to perform this split, one is called the *generative* approach, the other the *discriminative* approach

$$\text{generative:} \qquad P_{\text{true}}(\boldsymbol{x}, y) = P_{\text{true}}(\boldsymbol{x} \,|\, y)\; P_{\text{true}}(y) \tag{2.29}$$

$$\text{discriminative:} \qquad P_{\text{true}}(\boldsymbol{x}, y) = P_{\text{true}}(y \,|\, \boldsymbol{x})\; P_{\text{true}}(\boldsymbol{x}). \tag{2.30}$$

As we will see below, the density term $P_{\text{true}}(\boldsymbol{x})$ need not be modeled in the discriminative approach, because $\boldsymbol{x}$ is always observed. In this case the discriminative approach does not provide an expression for the joint distribution $P_{\text{true}}(\boldsymbol{x}, y)$, but only for the conditional $P_{\text{true}}(y \,|\, \boldsymbol{x})$. This is the origin of the names "generative" and "discriminative": A generative model can generate new inputs and outputs from the joint $P_{\text{true}}(\boldsymbol{x}, y)$, while the discriminative model can only discriminate between outputs for a given input from the conditional $P_{\text{true}}(y \,|\, \boldsymbol{x})$.

**Model constraints** Note that the two equations (2.29) and (2.30) are kind of meaningless. They are valid for any distribution of $\boldsymbol{x}$ and $y$ and do not restrict the model in any way. To show this explicitly, we write down the general model (2.4) both in the generative form and in the discriminative form:

$$P(\boldsymbol{x}, y, \boldsymbol{\theta}) = P(\boldsymbol{x} \,|\, y, \boldsymbol{\theta})\; P(y \,|\, \boldsymbol{\theta})\; P(\boldsymbol{\theta}) \tag{2.31}$$

$$= P(y \,|\, \boldsymbol{x}, \boldsymbol{\theta})\; P(\boldsymbol{x} \,|\, \boldsymbol{\theta})\; P(\boldsymbol{\theta}) \tag{2.32}$$

The difference between generative and discriminative models arises only when the marginal and the conditional are "modeled separately", which means that there are two independent

sets of parameters for the marginal and the conditional:

$$P_{gen}(\boldsymbol{x}, y, \boldsymbol{\theta}_{cd}, \boldsymbol{\theta}_{cp}) = P(\boldsymbol{x} \,|\, y, \boldsymbol{\theta}_{cd})\; P(y \,|\, \boldsymbol{\theta}_{cp})\; P(\boldsymbol{\theta}_{cd}) P(\boldsymbol{\theta}_{cp}) \tag{2.33}$$

$$P_{disc}(\boldsymbol{x}, y, \boldsymbol{\theta}_{cc}, \boldsymbol{\theta}_{td}) = P(y \,|\, \boldsymbol{x}, \boldsymbol{\theta}_{cc})\; P(\boldsymbol{x} \,|\, \boldsymbol{\theta}_{td})\; P(\boldsymbol{\theta}_{cc}) P(\boldsymbol{\theta}_{td}), \tag{2.34}$$

where the subscripts stand for

| | |
|---|---|
| $cd$ : class density | $cc$ : class conditional |
| $cp$ : class prior | $td$ : total density. |

This structure can be represented by the graphical models in Figures 2.3 and 2.4. For simplicity, we have drawn only one node each for $\boldsymbol{x}$ and $y$, that can represent either training points or test points (cf. Figure 2.1).

For a full understanding it is necessary to identify the concrete differences between (2.31, 2.32) and (2.33, 2.34) which cannot be overcome by reparameterization or rearranging the formulas. By inspection we find that (2.33, 2.34) satisfy the following statistical independence relations which are not satisfied by the general model (2.31, 2.32):

| generative | discriminative | |
|---|---|---|
| $\boldsymbol{x} \perp\!\!\!\perp \boldsymbol{\theta}_{cp} \,\mid y$ | $y \perp\!\!\!\perp \boldsymbol{\theta}_{td} \,\mid \boldsymbol{x}$ | (2.35) |
| $y \perp\!\!\!\perp \boldsymbol{\theta}_{cd}$ | $\boldsymbol{x} \perp\!\!\!\perp \boldsymbol{\theta}_{cc}$ | (2.36) |

where $\perp\!\!\!\perp$ denotes statistical independence. For the following, we consider these independence relations as the defining properties of generative and discriminative models.

Note that the factorization of the parameter prior in (2.33, 2.34)

| generative | discriminative | |
|---|---|---|
| $P(\boldsymbol{\theta}) = P(\boldsymbol{\theta}_{cp})\; P(\boldsymbol{\theta}_{cd})$ | $P(\boldsymbol{\theta}) = P(\boldsymbol{\theta}_{td})\; P(\boldsymbol{\theta}_{cc})$ | (2.37) |

is essential, although it does not explicitly appear in (2.35, 2.36). If this factorization does not hold, both independence relations (2.35, 2.36) are not valid anymore, and the resulting model is equivalent to the general form (2.31, 2.32). This is illustrated in Figures 2.5 and 2.6. We will refer to (2.37) in the following as *parameter independence*.

**Effects of model constraints** Let us briefly consider the effects of parameter (in)dependence for generative and discriminative models.

For generative models parameter independence means that the class densities do not depend on the class ratios. This seems to be a reasonable assumption. Though one might image a situation where the class densities do depend on the class ratios, this seems to be rather artificial. For example one might assume that a class which occurs rarely should be confined to a small region in feature space which would lead to a more peaked estimate of the rare class's density. We do not know of a classification model that has this property, but we believe it is important to understand that such models are possible and are neither

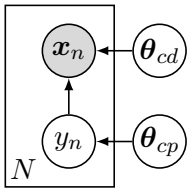


Figure 2.3: Bayesian network of generative models. For training points, $y$ is observed, for test points it is unobserved.

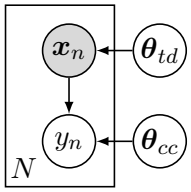


Figure 2.4: Bayesian network of discriminative models. For training points, $y$ is observed, for test points it is unobserved.

covered by the generative nor the discriminative framework. In Section 4.1 we will propose a bag model that is neither generative nor discriminative.

For discriminative models parameter independence has a much larger effect. The reason is that $\boldsymbol{x}$ is always observed which renders the two model parts completely independent. The outputs $y$ simply do not depend on the total density $\boldsymbol{\theta}_{td}$, see (2.35). From a practical viewpoint one might be happy about this, because one can just leave the total density out of the model (and out of consideration). In fact, it is often argued that discriminative models perform well in practice *because* they do not model the density but "focus on the more important class conditional $P(y \,|\, \boldsymbol{x})$". We believe that this argument is not really to the point. While the density model might of course be inappropriate and therefore lead to bad results, there is no reason why this must be so. So if generative models perform worse in experiments, we should rather try to find a better density model than to "blame it on the generative property". We should keep in mind that the total density carries useful information (otherwise semi-supervised learning would not work), and it would be unwise to discard this without need.

**Literature** To our knowledge, a discussion of generative and discriminative models similar to the above cannot be found in the literature. Usually, only the modeling approaches (2.29) and (2.30) are mentioned, but the concrete model constraints (2.33, 2.34) or (2.35–2.36) are

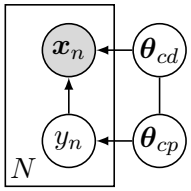


Figure 2.5: Bayesian network of generative models with prior parameter dependence.

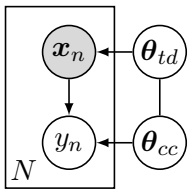


Figure 2.6: Bayesian network of discriminative models with prior parameter dependence.

not stated, and the central point of prior parameter independence between the submodels is not pointed out. As a consequence, there seems to be the believe that generative models were the most general approach, and that the defining trait of discriminative models was that they do not model the density.

However, the importance of prior parameter independence has been pointed out before by some authors in special contexts: (Seeger 2002) points out that discriminative models can be used for semi-supervised learning if there is a prior dependence between the parameters $\boldsymbol{\theta}_{td}$ and $\boldsymbol{\theta}_{cc}$. (Minka 2005) points out that the method of "discriminative training" is actually not a training method, but a change of model. One can impose the discriminative constraints on a generative model by doubling its parameters and make the two duplicates independent. The idea of (Minka 2005) has been worked out and a hybrid model where applied to object recognition by (Lasserre, Bishop & Minka 2006) and (Bishop & Lasserre 2007).

## 2.4 Semi-supervised learning

In many applications (e. g. in industrial optical inspection) there are plenty of datapoints $\boldsymbol{x}_n$ that could readily be used as training data, but acquiring labels $y_n$ for these data points is expensive or even prohibitive. The abundance of unlabeled data and scarcity of labeled data is often called the *labeling bottleneck*. In this situation the question arises whether unlabeled datapoints carry useful information, and if they do, how we can make use of unlabeled datapoints in practice.

The intuitive answer to the first question is: Yes, unlabeled datapoints seem to be helpful. The reason is illustrated in Figure 2.7[1]. Without unlabeled points, the best decision boundary seems to be a vertical line between the two labeled datapoints (top image). The unlabeled datapoints suggest, however, that the true decision boundary is completely different (bottom image).

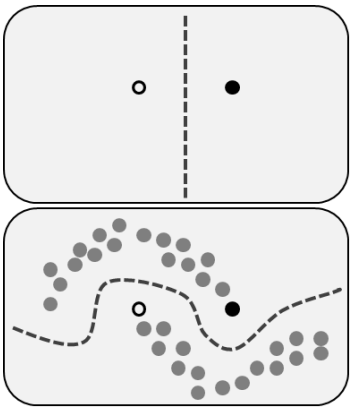

Figure 2.7: Illustration of semi-supervised learning.

In practice, it has also been shown by many authors that unlabeled data carry useful information (e. g. (Mitchell 1999), (Goldman & Zhou 2000), (Singh, Nowak & Zhu 2008), (Zhu 2010)).

The information of unlabeled points lies in their density distribution. This is somewhat contradictory to the discriminative approach to classification that does not consider the density at all. Indeed, truly discriminative models cannot learn from unlabeled datapoints, because the likelihood of any unlabeled point is equal, regardless of its location and regardless of the parameters of discriminative model.

$$P(\boldsymbol{x} \,|\, \boldsymbol{\theta}) \;=\; \sum_y P(\boldsymbol{x}, y \,|\, \boldsymbol{\theta}) \;=\; U(\boldsymbol{x}) \qquad \forall \boldsymbol{\theta} \tag{2.38}$$

[1] Source: Wikipedia

Since many well-known and successful classifiers are discriminative, this fact has lead to much discussion about the circumstances under which unlabeled points are useful (Chapelle, Schölkopf & Zien 2006).

Models for semi-supervised learning must exhibit a statistical dependence between the total density and class conditional (cf. Figure 2.6). (Chapelle et al. 2006) states four fundamental model assumptions that imply such a statistical dependence. They are more or less equivalent, but provide different perspectives of the issue, and motivate different kinds of models:

- Low-density separation: The decision boundary should preferably lie in low-density regions and should not cross high density regions.
- High-density smoothness assumption: Regions of high density should belong to the same class.
- Cluster assumption: If data points are in the same cluster, they are likely to be of the same class.
- Manifold assumption: The data points are assumed to lie roughly on a low-dimensional manifold in the high-dimensional feature space.

We would like to point out that the usefulness of these assumptions is not restricted to semi-supervised learning or to unlabeled data points. They are correct and should be considered for supervised learning as well (cf. Section 2.3).

### 2.4.1 Self-training

There is a variety of methods that have been developed for semi-supervised learning, among them are self-training, co-training, transductive inference, graph-based methods, and others. Here we focus on self-training, since this is the method we will employ later for multiple-instance learning.

The general idea of self-training is to estimate the unknown labels by the prediction of the classifier trained on the given labels. It is advantageous to do this in several steps, i. e. assigning only those labels, where the classifier has high confidence, and retrain the classifier with these labels to increase the confidence for the remaining datapoints.

More formally, self-learning alternates between updating the parameters $\boldsymbol{\theta}$, given the current estimate of instance classes $\boldsymbol{y}_i$, and updating the instance classes $\boldsymbol{y}$, given the current estimate of parameters $\boldsymbol{\theta}_i$. In general, the current estimates of instance classes and parameters are not given by single values $\boldsymbol{y}_i$, $\boldsymbol{\theta}_i$, but by distributions $P_i(\boldsymbol{y})$, or $P_i(\boldsymbol{\theta})$. Hence, the method of self-training is defined most generally by

$$P_i(\boldsymbol{\theta} \,|\, \boldsymbol{X}) \;\hat{=}\; \frac{1}{Z} \int P_M(\boldsymbol{y} \,|\, \boldsymbol{X}, \boldsymbol{\theta}) P_{i-1}(\boldsymbol{y} \,|\, \boldsymbol{X}) \, \mathrm{d}\boldsymbol{y} \tag{2.39}$$

$$P_i(\boldsymbol{y} \,|\, \boldsymbol{X}) \;\hat{=}\; \int P_M(\boldsymbol{y} \,|\, \boldsymbol{X}, \boldsymbol{\theta}) P_i(\boldsymbol{\theta} \,|\, \boldsymbol{X}) \, \mathrm{d}\boldsymbol{\theta}, \tag{2.40}$$

where we have distinguished between current distribution estimates $P_i$ at step $i$, and model likelihood $P_M$. $Z$ is a normalization constant.

Self-training allows for semi-supervised learning based on a discriminative model $P_M$, although discriminative models are in fact insensitive to unlabeled datapoints, as shown above. This is possible because self-training is not just an algorithmic procedure, but actually entails a change of model. This is apparent from Equation (2.39). If we dropped the indices $i$ and $M$ and interpreted it as a general probabilistic statement, it would actually be wrong! By defining the posterior $P_i(\boldsymbol{\theta} \,|\, \boldsymbol{X})$ in this"wrong" way, we effectively change the model to favor low-density separation.

To explain this effect, let us assume we have found a set of parameters $\hat{\boldsymbol{\theta}}$, that is a fixpoint of the sequence (2.39,2.40)

$$P(\hat{\boldsymbol{\theta}} \,|\, \boldsymbol{X}) = \frac{1}{Z} \int P_M(\boldsymbol{y} \,|\, \boldsymbol{X}, \hat{\boldsymbol{\theta}}) P(\boldsymbol{y} \,|\, \boldsymbol{X}) \, \mathrm{d}\boldsymbol{y} \tag{2.41}$$

$$P(\boldsymbol{y} \,|\, \boldsymbol{X}) = P_M(\boldsymbol{y} \,|\, \boldsymbol{X}, \hat{\boldsymbol{\theta}}), \tag{2.42}$$

Inserting (2.41) into (2.42) yields

$$P(\hat{\boldsymbol{\theta}} \,|\, \boldsymbol{X}) \hat{=} \frac{1}{Z} \int \left[P_M(\boldsymbol{y} \,|\, \boldsymbol{X}, \hat{\boldsymbol{\theta}})\right]^2 \mathrm{d}\boldsymbol{y} \tag{2.43}$$

$$= \frac{1}{Z} \prod_n \int \left[P_M(y_n \,|\, \boldsymbol{x}_n, \boldsymbol{\theta})\right]^2 \mathrm{d}y_n \tag{2.44}$$

$$= \frac{1}{Z} \prod_n \left(p_n^2 + (1-p_n)^2\right), \tag{2.45}$$

where we have used the short hand $p_n \hat{=} p_M(y_n = 1 \,|\, \boldsymbol{x}_n, \boldsymbol{\theta})$. Eq. (2.45) describes the effective parameter distribution of self-training, whose local optima correspond to the fixpoints of the sequence (2.39,2.40).

The contribution of a single datapoint to the effective parameter distribution (2.45) is plotted in Figure 2.8. As an illustrative example, let us consider two parameter vectors $\hat{\boldsymbol{\theta}}_{db}$ and $\hat{\boldsymbol{\theta}}_u$, that give the same $p_n$ for all datapoints except one. Let us assume further that for $\hat{\boldsymbol{\theta}}_{db}$, the remaining datapoint has $p_n = 1/2$ (it is located at the decision boundary), while for $\hat{\boldsymbol{\theta}}_u$, the remaining datapoint has $p_n = 0$ or $p_n = 1$ (it is unambiguously assigned to one class). In this case, the effective parameter probability (2.45) would be twice as large for the unique class assignment $\hat{\boldsymbol{\theta}}_u$ than for the undecided class assignment $\hat{\boldsymbol{\theta}}_{db}$. The same argument holds for all datapoints, so each datapoint that is close to the decision boundary reduces the relative parameter posterior by a factor of up to two, which of course penalizes decision boundaries in high density regions.

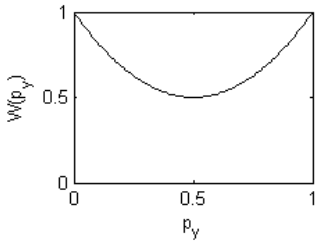


Figure 2.8: Plot of effective class purity-dependent weight of self-training.

## 2.5 Learning from weak labels

Supervised and semi-supervised learning consider the extreme cases of either complete class information about a datapoint (labeled) or zero class information about a datapoint (unlabeled). In practice we sometimes have a situation in between these two extremes, i. e. we have some *weak* information about the class of a datapoint.

The weakness of information can have different reasons. For example, the given label might be subject to statistical error, so that it only provides a prior class *probability* of the corresponding datapoint. Another possible reason is that we only have aggregate information about a group (or *bag*) of many datapoints. In this case, though the bag label is exact, it only gives a vague information about the class of a single datapoint.

In the following we give a short overview of weak label models, focusing on bag models, since the multiple instance model is a special case of a bag model.

### 2.5.1 Noisy labels

There is no single widely accepted term to refer to the situation where the available labels are subject to errors. It has been called "noisy labels" (Natarajan, Dhillon, Ravikumar & Tewari 2013), "errors in labels" (Buehler, Zisserman & Everingham 2009), "uncertain labels" (Bouveyron & Girard 2009), "imperfect labels" (Tabassiana, Ghaderia & Ebrahimpourb 2012), and "ambiguous labels" (Hüllermeier & Beringer 2005). An experimental survey of the effect of noisy labels can be found in (Nettleton, Orriols-Puig & Fornells 2010).

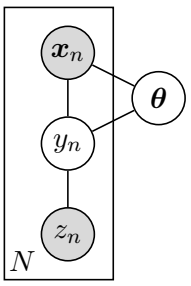


Figure 2.9: Markov network of learning from labels with errors.

The common trait of these models is that the classes $y_n$ are not observed directly, but only a third quantity $z_n$, which is related to the class $y_n$, is observed. The corresponding graphical model is shown in Figure 2.9.

**Binary classification and Bernoulli model** In the case of binary classification, the relation between $y_n$ and $z_n$ is most generally described by four parameters with one normalization constraint

| $P(y_n, z_n)$ | $y_n = 0$ | $y_n = 1$ |
|---|---|---|
| $z_n = 0$ | $\beta_1$ | $\beta_2$ |
| $z_n = 1$ | $\beta_3$ | $\beta_4$ |

(2.46)

$$\beta_1 + \beta_2 + \beta_3 + \beta_4 = 1. \tag{2.47}$$

Because $z_n$ is always observed, it is sufficient to model the conditional $P(y_n \mid z_n)$

| $P(y_n \mid z_n)$ | $y_n = 0$ | $y_n = 1$ |
|---|---|---|
| $z_n = 0$ | $1 - \beta_{FN}$ | $\beta_{FN}$ |
| $z_n = 1$ | $\beta_{FP}$ | $1 - \beta_{FP}$ |

(2.48)

where $\beta_{FN}$ is the rate of false negative labels, and $\beta_{FP}$ is the rate of false positive labels. Often it is appropriate to set $\beta = \beta_{FN} = \beta_{FP}$. Since the conditional $P(y_n \mid z_n)$ is a Bernoulli distribution, we call the model (2.48) the "Bernoulli model". In chapter 5, we propose such a Bernoulli model as an addition to the multiple instance model.

The approach to model only the conditional distribution instead of the complete joint distribution is reminiscent of the discriminative model split that we discussed in Section 2.3. But note that in this case the parameters $\boldsymbol{\beta}$ are assumed fixed, so parameter independence is not an additional constraint, and (2.48) is equivalent to (2.46).

**Multiple labels** When exact labels are not available, it is a natural idea to improve over a single noisy label by taking multiple noisy labels (see Figure 2.10). Even when it is possible to obtain exact labels, it might be cheaper to acquire multiple noisy labels. For this reason the setting of multiple noisy labels has received considerable treatment in the literature. The model of multiple noisy labels was introduced by (Jin & Ghahramani 2002). Up-to-date reviews about this topic can be found in (Zhang & Zhou 2013) and (Sorower 2010).

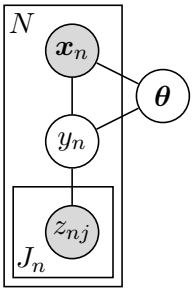


Figure 2.10: Markov network of learning from multiple labels.

### 2.5.2 Bag labels

Another form of weak labels arises if we are not given the class information for each single datapoint, but only aggregate information about many datapoints. Each set of datapoints gives rise to a multiset – or *bag* – of classes, and for each bag we are given one *bag label* $c$. In this context a single datapoint is usually referred to as an *instance* in order to distinguish it from its respective bag. The corresponding graphical model is shown in Figure 2.11.

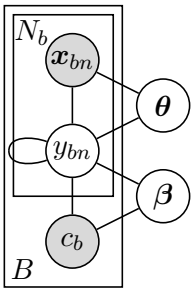


Figure 2.11: Markov network of learning from bag labels with bag parameters $\boldsymbol{\beta}$. The loop at node $y_{bn}$ indicates that all $\{y_{bn}\}_n$ are pairwise connected (see text for explanation).

The most well-known bag model seems to be the so-called multiple instance model, where the bag label is negative if *all* instance labels are negative, and positive if *at least one* instance label is positive. This model is a main topic of this work, and we will discuss it in detail in the following section. Another (more informative) bag label would be the counts of instance classes within each bag. Such labels have been used for learning by (Kück & de Freitas 2005).

Note that the essential difference between bag models and noisy labels is that bag models allow for correlations between the instances of the same bag. Formally, the bag model factor $Q_B$ does not factorize into instances, but only into bags:

$$Q_B(\boldsymbol{Y}, \boldsymbol{c}, \boldsymbol{\beta}) = \prod_b Q_b((\boldsymbol{y}_b, c_b, \boldsymbol{\beta}) \tag{2.49}$$

$$\neq \prod_b \prod_n Q_b((y_{bn}, c_b, \boldsymbol{\beta}), \tag{2.50}$$

where the bold symbols stand for sets of variables: $\boldsymbol{Y} = \{y_{bn}\}_{bn}$, $\boldsymbol{y}_b = \{y_{bn}\}_n$, $\boldsymbol{c} = \{c_b\}_b$. To represent the factorization (2.49) by a the Markov network, all $y$-nodes on each $N$-plate must be pairwise connected. We have indicated this in Figure 2.11 by a loop.

The fact that bag models do not factorize into instances makes learning difficult. We will encounter this problem in Chapter 6, when devising an algorithm for multiple instance learning.

### 2.5.3 Inference in bag models

When dealing with bag models, we must take into account the interaction between the classifier and the bag model. The latent instance classes $\boldsymbol{y}$ depend both on the classification model $P(\boldsymbol{x}, y, \boldsymbol{\theta})$ and on the bag model $P(\boldsymbol{y}, c, \boldsymbol{\beta})$. This has the consequence that the two submodels cannot be regarded as separate anymore, but they become factors $Q_{\mathrm{Cl}}$, $Q_B$ of the joint model $P(\boldsymbol{X}, \boldsymbol{y}, c, \boldsymbol{\theta}, \boldsymbol{\beta})$.

In this section, we derive the expressions that are needed to do inference in general bag models. To our knowledge, these expressions have not been published before. For clarity, we state the expressions for only one bag and leave out the products over bag indices $b$.

Let us denote the factor describing the classification model by $q_{\mathrm{Cl}}$ and the factor describing the bag model by $Q_B$. The joint probability of the complete model is

$$P(\boldsymbol{X}, \boldsymbol{y}, c, \boldsymbol{\theta}, \boldsymbol{\beta}) = \frac{1}{Z} Q_{\mathrm{Cl}}(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{\theta}) Q_B(\boldsymbol{y}, c, \boldsymbol{\beta}) \tag{2.51}$$

$$Q_{\mathrm{Cl}}(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{\theta}) = \prod_n q_{\mathrm{Cl}}(\boldsymbol{x}_n, y_n, \boldsymbol{\theta}) \tag{2.52}$$

$$Z = \int Q_{\mathrm{Cl}}\, Q_B \;\, \mathrm{d}\boldsymbol{X}\, \mathrm{d}\boldsymbol{y}\, \mathrm{d}c\, \mathrm{d}\boldsymbol{\theta}\, \mathrm{d}\boldsymbol{\beta}, \tag{2.53}$$

where $\boldsymbol{X} = \{\boldsymbol{x}_n\}$ denotes the set of all feature vectors of one bag, and the partition function $Z$ ensures normalization.

Note that although the factors $q_{\mathrm{Cl}}$ and $Q_B$ represent the probabilities $P(\boldsymbol{x}_n, y_n, \boldsymbol{\theta})$ and $P(\boldsymbol{y}, c, \boldsymbol{\beta})$, resp., they are not equal to the corresponding marginal probabilities of the joint model

$$q_{\mathrm{Cl}} \neq P(\boldsymbol{x}_n, y_n, \boldsymbol{\theta}) = \int P(\boldsymbol{X}, \boldsymbol{y}, c, \boldsymbol{\theta}, \boldsymbol{\beta})\, \mathrm{d}\boldsymbol{x}_{m\neq n}\, \mathrm{d}y_{m\neq n}\, \mathrm{d}c\, \mathrm{d}\boldsymbol{\beta} \tag{2.54}$$

$$Q_B \neq P(\boldsymbol{y}, c, \boldsymbol{\beta}) = \int P(\boldsymbol{X}, \boldsymbol{y}, c, \boldsymbol{\theta}, \boldsymbol{\beta})\, \mathrm{d}\boldsymbol{X}\, \mathrm{d}\boldsymbol{\theta}. \tag{2.55}$$

This can be quite counter-intuitive, as we will discuss in Section 5.1.4 for the case of the Bernoulli bag model.

It is convenient to use a distinct symbol $M_y$ for the marginal over the latent variables $\boldsymbol{y}$ and focus on its dependence on the parameters $(\boldsymbol{\theta}, \boldsymbol{\beta})$ and on the bag class $c$.

$$M_y(\boldsymbol{\theta}, \boldsymbol{\beta}, c) = P(c, \boldsymbol{X}, \boldsymbol{\theta}, \beta) = \int Q_B\, Q_{\mathrm{Cl}}\, \mathrm{d}\boldsymbol{y} \tag{2.56}$$

The parameter posterior $P(\boldsymbol{\theta}, \boldsymbol{\beta} \,|\, \boldsymbol{x}, c)$ (needed for training) and the bag class probability $P(c \,|\, \boldsymbol{X}, \boldsymbol{\theta}, \beta)$ (needed for testing) now take on the following simple forms

$$P(\boldsymbol{\theta}, \beta \,|\, \boldsymbol{X}, c) = \frac{M_y}{\int M_y\, \mathrm{d}\boldsymbol{\theta}\, \mathrm{d}\beta} \propto M_y(\boldsymbol{\theta}, \boldsymbol{\beta}) \tag{2.57}$$

$$P(c \,|\, \boldsymbol{X}, \boldsymbol{\theta}, \beta) = \frac{M_y}{\int M_y\, \mathrm{d}c} = \frac{M_y(c)}{M_y(c{=}0) + M_y(c{=}1)} \tag{2.58}$$

Note that for maximization of or sampling from the parameter posterior (2.57) it is not necessary to know the denominator (normalization constant), but for bag classification (2.58) the denominator is essential.

For discriminative bag models, the above expressions can be simplified (see Section 4.1). Since the multiple instance model is a discriminative bag model, it does not necessarily require the general expressions above. However, for the improvements and generalizations we propose in Chapters 4 and 5, the general expressions stated above are indeed necessary, and we will refer to Equations (2.56–2.58) when discussing and analyzing these models.

# Chapter 3

# Multiple Instance Learning

Multiple instance learning is the central topic of this work. It allows to learn from bag labels, where a negative bag label implies that all of its instances are negative, and a positive bag label implies that at least one of its instances are positive. This bag model corresponds to the setting of industrial optical inspection, where an image is labeled as negative only if it is entirely free of defects, and labeled positive if it contains at least one defect.

Multiple instance learning has originally been proposed by (Dietterich, Lathrop & Lozano-Pérez 1997) for an application of drug activity prediction. Since then it has found several other applications in image classification, text categorization, data mining, and others. It has received continued attention from the machine learning community and several algorithms for multiple instance learning have been proposed.

In this chapter we will give an overview of the multiple instance model, its applications, and available algorithms.

## 3.1 The multiple instance model

### 3.1.1 Model definition

The multiple instance model is actually a deterministic bag model. However, it is only useful in combination with a probabilistic classifier. Therefore, we state below both the deterministic and the probabilistic expressions.

**Multiple instance bag model (deterministic)** The multiple instance model relates the bag class $c$ to the instance classes $\boldsymbol{y} = \{y_n\}$. The informal definition is

$$\boxed{\begin{array}{lcl} \text{bag negative} & \Leftrightarrow & \text{all instances negative} \\ \text{bag positive} & \Leftrightarrow & \text{at least one instance positive} \end{array}} \tag{3.1}$$

Formally, this is expressed most concisely as

$$c = \max_n y_n = \mathrm{OR}_n(y_n) \qquad \Longleftrightarrow \tag{3.2}$$

$$1 - c = \min_n 1 - y_n = \mathrm{AND}_n(1 - y_n), \tag{3.3}$$

where $c$ and $y_n$ are binary variables with $c = 0$ (or $y_n = 0$) denoting a negative class membership and $c = 1$ (or $y_n = 1$) denoting a positive class membership.

An example of a bag containing two instances is

| $c$ | $y_2{=}0$ | $y_2{=}1$ |
|---|---|---|
| $y_1{=}0$ | 0 | 1 |
| $y_1{=}1$ | 1 | 1 |

(3.4)

For bags containing more instance, one can imagine an $N$-dimensional cube where one orthant is 0 and all others are 1.

Note that the multiple instance bag model is in fact a deterministic model. If the instance classes $\boldsymbol{y}$ are known, the bag class $c$ is determined exactly. The probabilistic nature of multiple instance learning originates solely from the probabilistic classifier.

**Multiple instance classification model (probabilistic)** The multiple instance bag model (3.2) is always used together with a probabilistic classifier. The combined model (classifier plus MI) is characterized by the $y$-marginal $M_y$. To evaluate it, we first we state (3.2) as a degenerate probability distribution

$$Q_{\mathrm{MI}}(\boldsymbol{y}, c) \;=\; \delta\left(c, \max_n y_n\right), \tag{3.5}$$

and we introduce the short-hand

$$p_n \;=\; q_{\mathrm{Cl}}(\boldsymbol{x}_n, y_n, \boldsymbol{\theta}) \tag{3.6}$$

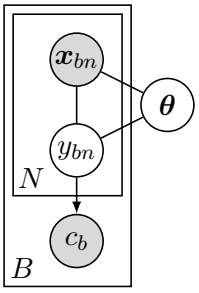


Figure 3.1: Graphical model of multiple instance learning.

In the following, we refer to $p_n$ as the "soft outputs" of the instance classifier. Now, Eq. (2.56) evaluates as

$$M_{\mathrm{MI}}(c{=}0) \;=\; \prod_n (1 - p_n) \;=\; \mathrm{AND}_n(1 - p_n) \tag{3.7}$$

$$M_{\mathrm{MI}}(c{=}1) \;=\; 1 - \prod_n (1 - p_n) \;=\; \mathrm{OR}_n(p_n). \tag{3.8}$$

Since $M_{\mathrm{MI}}(c{=}0) + M_{\mathrm{MI}}(c{=}1) = 1$, the denominator of (2.58) is trivial and we have

$$P(c{=}1) \;=\; 1 - \prod_n (1 - p_n) \;=\; \mathrm{OR}_n(p_n). \tag{3.9}$$

We have used the notation "AND" and "OR" for probabilistic conjunction and disjunction, resp. Probabilistic OR is often called "noisy-OR" in the literature. The expression (3.9) was first stated by (Maron & Lozano-Pérez 1998) and is well-known in the literature. However, the conceptual distinction between the (deterministic) label model itself and its

(probabilistic) combination with the classifier is usually not made.

Note that the term "multiple instance learning" is a bit misleading because actually *all* bag models involve multiple instances, not just the specific model defined above. A more descriptive term would be "bag-OR model". We point this out because the improper name has led to some confusion; for instance, (Amores 2013) uses the term "multiple instance" as referring to the class of all bag models. But since the majority of papers uses the term "multiple instance learning" according to the above definition, we adopt this term although it is somewhat misleading.

### 3.1.2 Model training

To get an idea of the effect of the multiple instance (MI) model on classifier training, we examine the negative log-likelihood of the model as a function of the soft outputs of the instance classifier. Comparing Equations (2.57), (3.7), and (3.7), we find

$$\mathrm{NLL}^{-}_{\mathrm{MI}}(\boldsymbol{p}) \;=\; -\log(M_{\mathrm{MI}}(c{=}0)) \;=\; -\sum_n \log(1-p_n) \tag{3.10}$$

$$\mathrm{NLL}^{+}_{\mathrm{MI}}(\boldsymbol{p}) \;=\; -\log\left(M_{\mathrm{MI}}(c{=}1)\right) \;=\; -\log\left(1-\prod_n (1-p_n)\right), \tag{3.11}$$

where (3.10) is valid for bags labeled "negative" and (3.10) is valid for bags labeled "positive". Note that the model likelihood for positive bags does not factorize, which makes MI learning hard.

To understand the behavior of the model, it is useful to separate a single instance $p_j$ in the formulas (3.10,3.11), so that we see the effect of changing this instance's class probability on the bag model likelihood. For negative bags this is trivial because it factorizes into instances, but for positive bags we have to take some care.

It is convenient to summarize all instances other than $j$ in a single term $\boldsymbol{p}_{\setminus j} = \{p_n\}_{n\neq j}$. Then we can express the complete bag's negative log-likelihood as a function of the "reduced" bag's negative log-likelihood $\mathrm{NLL}(\boldsymbol{p}_{\setminus j})$ and $p_j$:

$$\mathrm{NLL}^{-}_{\mathrm{MI}}(p_j, \boldsymbol{p}_{\setminus j}) \;=\; -\log(1-p_j) + \mathrm{NLL}^{-}_{\mathrm{MI}}(\boldsymbol{p}_{\setminus j}) \tag{3.12}$$

$$\mathrm{NLL}^{+}_{\mathrm{MI}}(p_j, \boldsymbol{p}_{\setminus j}) \;=\; -\log\left[1-\left(1-p_j\right)\left(1-\exp\left[\mathrm{NLL}^{+}_{\mathrm{MI}}(\boldsymbol{p}_{\setminus j})\right]\right)\right] \tag{3.13}$$

These functions of two variables can be plotted as a set of curves, as is done in Figure 3.2. The slope of the curves characterizes the impact of a single instance on the total bag likelihood. If the slope is large, the multiple instance model will "push" the instance classifier hard to change the class assignment of that instance.

As expected, an instance's impact is the larger the more its class probability contradicts the given bag label. For negative bags, completely positive instances ($p_j = 1$) are forbidden (dashed curves). For positive bags, an instance's impact on the bag-NLL depends on the remaining instances of its bag. If one (or several) of them is positive, they have already

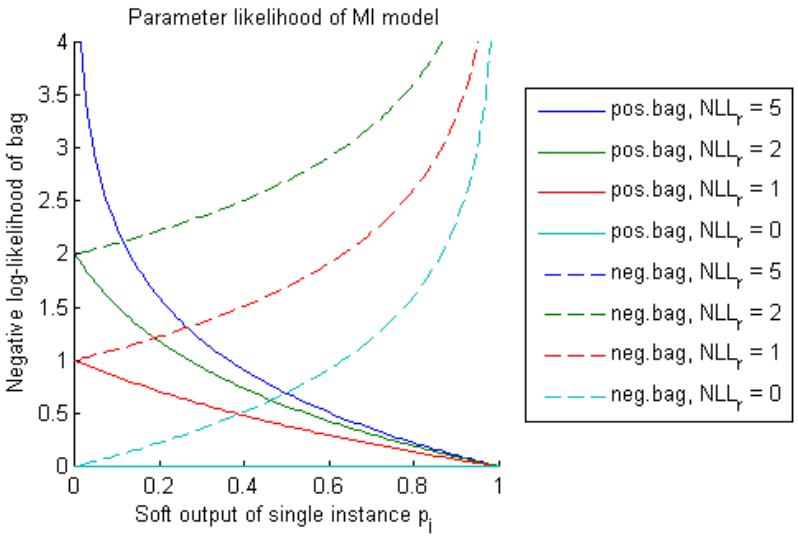


Figure 3.2: Contribution of a single instance to the negative log-likelihood of the multiple instance model. The negative log-likelihood of the "reduced" bag is denoted by $\mathrm{NLL}_r = \mathrm{NLL}(\boldsymbol{p}_{\setminus j})$.

"justified" the positive bag label ($\mathrm{NLL}(\boldsymbol{p}_{\setminus j}) = 0$), and the class assignment for instance $j$ becomes irrelevant (constant cyan curve). If none of the remaining instances are positive, the positive bag label is not yet "justified" ($\mathrm{NLL}(\boldsymbol{p}_{\setminus j}) \gg 1$), and the remaining instance is strongly pushed to a positive class assignment (solid blue curve).

### 3.1.3 Relation to semi-supervised learning

Multiple instance (MI) learning has some similarity with semi-supervised learning (SSL). To make this apparent, consider an asymmetric semi-supervised setting where all labeled datapoints are negative (i. e. there are no positive labels at all). Then the labeled datapoints correspond to the instances from negative MI-bags, while the unlabeled datapoints correspond to instances from positive MI-bags.

The correspondence is not exact, because the instances from positive MI bags are not really unlabeled. Instead, at least one of the instances from each positive MI-bag must be positive. This can be viewed as a constraint on the instance classes, and we will refer to this in the following as the *MI-constraint*.

Note that the hypothetical asymmetric semi-supervised setting above is actually ill-defined, because the best fit to the observed data (without any positive labels) is the trivial one of classifying all datapoints as negative. The only thing that prevents this degenerate solution in the multiple instance setting are the MI-constraints.

Because of the close correspondence between multiple instance learning and semi-supervised

learning many methods that have been proposed for one of these settings has also been applied to the other. Notable semi-supervised methods that are related to multiple instance learning are (Kück, Carbonetto & de Freitas 2004), (Goldman & Rahmani 2006), (Zhou & Xu 2007), (Leistner, Saffari, Santner & Bischof 2009), and (Zeisl, Leistner, Saffari & & Bischof 2010).

**Low-density separation** During training the multiple instance model, we can distinguish between two different situations. For positive bags with large negative likelihood, we say that the MI-constraint is active, for positive bags with small negative likelihood, the MI-constraint is inactive.

The situation where the MI-constraint is inactive corresponds to semi-supervised learning. In this case, we have to adopt one of the four model assumptions of semi-supervised learning (see Section 2.4). As mentioned above, these assumptions are more or less equivalent. We find the assumption of low-density separation most intuitive.

Our algorithm described in Section 6 is based on self-training and therefore exhibits the kind of low-density separation described in Section 2.4.1.

## 3.2 Interpretations and variants of the multiple instance model

When the multiple instance model was first proposed (Dietterich et al. 1997), it was not stated as formally as above, but as a verbal description of the application's requirements. Besides the above described model, there are some slightly different variants that are also plausible and correspond to the verbal description (3.1).

In this section we describe these different variants and analyze in which applications they are appropriate.

### 3.2.1 Standard MI (sMI): Estimate latent instance classes

To distinguish the MI model as defined in Section 3.1 from the following MI variants, we call it the "standard MI" or "sMI" model.

The main point that distinguishes the sMI model from the following MI models is the role of the instance classes $\boldsymbol{y}$. In sMI, they are treated as latent variables that are unknown, but are explicitly estimated during training and passed as input to the instance classifier.

### 3.2.2 Discard non-positive instances (dMI)

Let us denote the datapoints from positive bags that are in fact negative as the "non-positive" instances. While in the standard MI model the non-positives are regarded as negative, it is also possible to discard them, so they have no effect on the instance classifier. We call this approach "dMI".

This approach seems reasonable if there are already many negative instances from negative bags. One might consider this as the "safe" option, because we do not know for sure which

instances are the non-positives are never known for sure to be negative, because the MI learner might have erred.

However, discarding the non-positives has another effect: The MI learner now prefers to discard as many instances as possible and to estimate as few instances as possible as truly positive. This leaves only the most extreme instance of each positive bag, and the MI learner will place the decision boundary in the middle between the most extreme instances of the negative bags and the most extreme instances of the positive bags. In the following we will call the "most extreme" (i. e. most positive) instance of a bag as the bag's *witness*.

The idea to discard non-positives was first proposed by (Andrews, Tsochantaridis & Hofmann 2002) for the adaption of support vector machines to multiple instance-learning. They named their implementation of dMI "MI-SVM" and their implementation of sMI "mi-SVM". Improved versions of these algorithms have been proposed by (Gehler & Chapelle 2007), called "AW-SVM" and "AL-SVM", resp., where the W stands for "witness" and L stands for "all labels". We believe that a clear and ambiguous terminology is needed to distinguish between the two models, irrespective of algorithmic details, so we propose the terminology "sMI" and "dMI" as stated in the headlines.

**Formal statement of dMI-model** Proceeding to a formal statement of the dMI-model, we label the negative class by $y=-1$ and introduce a third instance "class" $y=0$ that labels discarded instances. The bag model of dMI is represented by (cf. Eq. (3.2))

$$Q_{\mathrm{dMI}}(c=-1,\boldsymbol{y}) = \begin{cases} 1 & \text{if } \max_n y_n = -1 \\ 0 & \text{otherwise} \end{cases} \tag{3.14}$$

$$Q_{\mathrm{dMI}}(c=1,\boldsymbol{y}) = \begin{cases} 1 & \text{if } (\max_n y_n = 1) \text{ AND } (\min_n y_n > -1). \\ 0 & \text{otherwise} \end{cases} \tag{3.15}$$

An example of a bag containing two instances is the following (This might be compared to the example in Section 4.1.2):

| $Q_{\mathrm{dMI}}(c=-1,\boldsymbol{y})$ | $y_2=-1$ | $y_2=0$ | $y_2=1$ |
|---|---|---|---|
| $y_1=-1$ | 1 | 0 | 0 |
| $y_1=0$ | 0 | 0 | 0 |
| $y_1=1$ | 0 | 0 | 0 |

(3.16)

| $Q_{\mathrm{dMI}}(c=1,\boldsymbol{y})$ | $y_2=-1$ | $y_2=0$ | $y_2=1$ |
|---|---|---|---|
| $y_1=-1$ | 0 | 0 | 0 |
| $y_1=0$ | 0 | 0 | 1 |
| $y_1=1$ | 0 | 1 | 1 |

(3.17)

Next we need to define how "discarded" instances ($y_n = 0$) are treated by the instance classifier. From a practical point of view, one can just leave the discarded points out of the corresponding factor (cf. Eq. (2.52))

$$Q_{\mathrm{Cl}}(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{\theta}) = \prod_{n:y_n \neq 0} q_{\mathrm{Cl}}(\boldsymbol{x}_n, y_n, \boldsymbol{\theta}) \tag{3.18}$$

The problem with this approach is that the factor (2.52) is in fact determined by the laws of Bayesian inference, and we are not allowed to change it at will. Actually, our newly defined factor (3.20) corresponds to a different graphical model, shown in Figure 3.3. Note that the discarded points $\boldsymbol{x}_{n''}$ have no connection to any other nodes. So during training, one actually does not only infer the values of parameters and latent variables, but the model structure itself is flexible. There are methods that can learn the model structure (Daly, Shen & Aitken 2011), but this is actually a bit over the top for the model above.

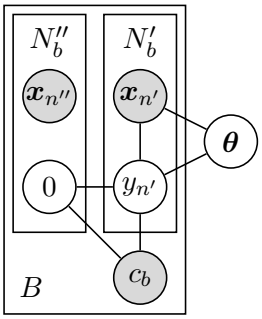


Figure 3.3: Markov network of dMI-model (multiple instance with discarding of non-positives).

Instead, it is much easier to introduce another factor $q'_{\mathrm{Cl}}(\boldsymbol{x}, y, \boldsymbol{\theta})$ than to change the model structure.

$$q'_{\mathrm{Cl}}(\boldsymbol{x}_n, y_n, \boldsymbol{\theta}) = \begin{cases} \frac{1}{q_{\mathrm{Cl}}(\boldsymbol{x}_n, y_n, \boldsymbol{\theta})} & \text{if } y_n = 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.19}$$

$$Q_{\mathrm{Cl}}(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{\theta}) = \prod_n q_{\mathrm{Cl}}(\boldsymbol{x}_n, y_n, \boldsymbol{\theta})\, q'_{\mathrm{Cl}}(\boldsymbol{x}_n, y_n, \boldsymbol{\theta}) \tag{3.20}$$

Note that $Q_{\mathrm{Cl}}$ as defined above is not normalized, so care must be taken when interpreting it as a classification probability.

**Testing the dMI model** A change of model affects both training and testing. Since dMI discards instances during training, the same is allowed during testing. This corresponds to the "threshold" method (cf. Section 4.2.2).

**One-class learning of witnesses (wMI)** In the original multiple instance application of drug activity prediction, it is known that the positive region in feature space is very small. In this case it is plausible to try and find the smallest positive region that explains all positive bags. Indeed, this is the underlying idea of the first proposed MI algorithms "APR" and "diverse density" (see Section 3.4).

This approach is actually a form of one-class learning of the positive bags' witnesses (most positive points of each bag). In this approach the negative bags are used merely to identify the witnesses. Once the witnesses are known, the decision boundary is fitted around them as tight as possible without taking into account the negative points anymore.

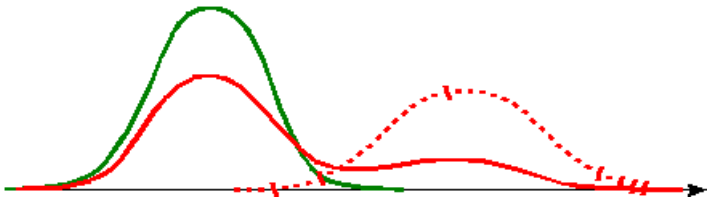

Figure 3.4: Illustration of sesqui-class learning. Solid lines: Observed densities of positive and negative bags. Dashed: True positive density.

### 3.2.3 Sesqui-class learning (SCL)

Sesqui-class learning is based on the generative approach, i. e. the classification model is divided into class densities $P(\boldsymbol{x} \,|\, y, \boldsymbol{\theta})$ and a class prior $P(y \,|\, \pi)$ (see Section 2.3)

$$P(\boldsymbol{x}, y, \boldsymbol{\theta}, \pi) \;=\; P(\boldsymbol{x} \,|\, y, \boldsymbol{\theta}) P(y \,|\, \pi) P(\boldsymbol{\theta}) P(\pi) \tag{3.21}$$

In the MI setting we cannot observe these densities directly because the instance classes $y_n$ are latent. We only know the bag class $c$, and according to the MI model the corresponding densities can be written as

$$P(\boldsymbol{x} \,|\, c{=}0) \;=\; P(\boldsymbol{x} \,|\, y{=}0) \tag{3.22}$$

$$P(\boldsymbol{x} \,|\, c{=}1) \;=\; \alpha P(\boldsymbol{x} \,|\, y{=}1) + (1-\alpha) P(\boldsymbol{x} \,|\, y{=}0) \qquad 0 < \alpha < 1. \tag{3.23}$$

Negative bags contain only negative instances, but positive bags contain both positive and negative instances, so their combined density is a mixture of the true positive and true negative densities. The observed bag densities and the true instance class densities are illustrated in Figure 3.4.

The mixture parameter $\alpha$ is given by the bag sizes and the class prior. Let $(N^+, N^-)$ be the numbers of instances from positive and negative bags, resp. Then

$$P(y{=}1 \,|\, \pi) \;=\; \pi \;=\; \frac{\alpha N^+}{N^+ + N^-} \tag{3.24}$$

$$\alpha \;=\; \pi \left(1 + \frac{N^-}{N^+}\right) \tag{3.25}$$

Since the bag sizes are known, estimation of the mixture parameter $\alpha$ and the class prior $\pi$ is equivalent.

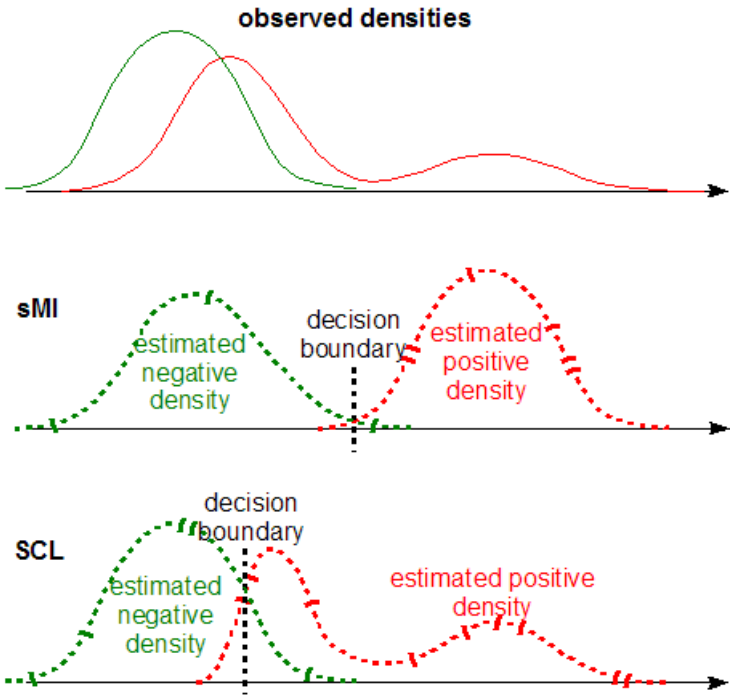


Figure 3.6: Comparison of sesqui-class learning (SCL) with standard MI (sMI).

**Relation to standard MI (sMI)** While standard MI learning estimates both densities and the class prior in one step, sesqui-class learning divides the procedure into two steps. In the first step, one learns the negative class density $P(\boldsymbol{x} \mid y=-1, \boldsymbol{\theta})$ from the negative bags only; in the second step, one learns the positive density $P(\boldsymbol{x} \mid y=1, \boldsymbol{\theta})$ and the class prior $P(y \mid \pi)$ from the positive bags, while keeping the negative density constant.

We would like to stress that this two-step approach is not just an algorithmic choice, but it actually corresponds to a different model that has a different optimum. The difference lies in the role of the non-positives (true negative instances from positive bags). While in sMI they can influence the learned negative density, this influence is canceled in sesqui-class learning. Figure 3.5 shows the graphical model for both steps of sesqui-class learning.

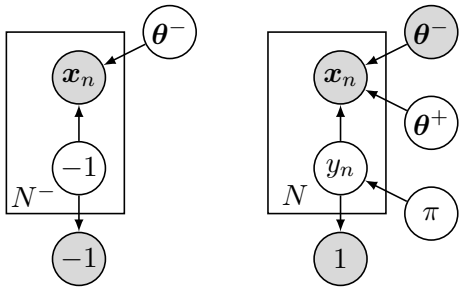


Figure 3.5: Bayesian network of sesqui-class learning (multiple instance learning with fixed negative density). Left: Learning the negative density from negative bags (the node labels $y_n$ and $c$ have been replaced by their fixed values $-1$). Right: Learning positive density and class prior from all bags.

To compare sesqui-class learning with standard MI learning, consider the example shown in Figure 3.6. The left mode of the observed positive density is similar to the negative density, but slightly shifted. Standard MI learning puts the decision boundary in the low density region, although in this case the resulting estimated negative density differs somewhat from the negative density as observed from negative bags. Sesqui-class learning, on the contrary, abides by the negative density as observed from negative bags and instead explains the observed positive density with a bimodal true positive density.

We think that standard MI is more plausible in this example, but this might be different in other examples. To judge, one should consider which of the following is the more reliable indicator of the true decision boundary: (i) the low density region or (ii) the border of the density from negative bags.

**Training the sMI and SCL models** In the generative approach the y-marginal (2.56) reads

$$M_y(c{=}0) = \prod_n (1-\pi) f_0(\boldsymbol{x}_n) \tag{3.26}$$

$$M_y(c{=}1) = \prod_n \Big[\pi f_1(\boldsymbol{x}_n) + (1-\pi) f_0(\boldsymbol{x}_n)\Big] - \prod_n (1-\pi) f_0(\boldsymbol{x}_n), \tag{3.27}$$

where we have used the short-hands

$$f_0(\boldsymbol{x}) = P(\boldsymbol{x}, y{=}0, \boldsymbol{\theta}) \tag{3.28}$$

$$f_1(\boldsymbol{x}) = P(\boldsymbol{x}, y{=}1, \boldsymbol{\theta}). \tag{3.29}$$

During training, $M_y$ this is optimized over $\boldsymbol{\theta}$ and $\pi$. The second summand of (3.27) is the MI constraint; it should be very small.

As already mentioned before, standard MI has a tendency to assign all points as negative $\pi = 0$. Only the MI constraint prevents this degenerate solution. However, we can be sure of a good solution only if the MI constraint is not "active", i. e. the second summand of (3.27) is very small.

Sesqui: Since $f_0$ is not allowed to change, all deviations of the positive bag density must be explained by $f_1$. However, $\pi$ is held as small as possible, because otherwise the negative bags would not be explained anymore.

We would like to stress that the negative bags are not disregarded in the second SCL step. The negative bags play a vital role in the estimation of the prior $\pi$ (i. e. the mixture parameter $\alpha$). If the negative bags were disregarded, one could simply assign all instances from positive bags as positive:

$$\pi = 1 \tag{3.30}$$

$$P(\boldsymbol{x} \,|\, y{=}1) = P(\boldsymbol{x} \,|\, c{=}1) \tag{3.31}$$

This degenerate solution is prevented by the negative bags which require a $\pi$ close to zero. So the classifier is encouraged to assign as negative all points from positive bags that comply

with the negative density.

**Relation to generative supervised learning** Let $r_y$ and $r_c$ be the ratios of instance class densities and bag class densities, resp.

$$r_y(\boldsymbol{x}) = \frac{P(\boldsymbol{x} \mid y{=}1, \boldsymbol{\theta})}{P(\boldsymbol{x} \mid y{=}0, \boldsymbol{\theta})} \tag{3.32}$$

$$r_c(\boldsymbol{x}) = \frac{P(\boldsymbol{x} \mid c{=}1, \boldsymbol{\theta})}{P(\boldsymbol{x} \mid c{=}0, \boldsymbol{\theta})} \tag{3.33}$$

Then we can derive from (3.22) and (3.23):

$$r_y(\boldsymbol{x}) = 1 + \frac{1}{\alpha}\Big[r_c(\boldsymbol{x}) - 1\Big] \tag{3.34}$$

$$r_c(\boldsymbol{x}) = 1 + \alpha\Big[r_y(\boldsymbol{x}) - 1\Big] \tag{3.35}$$

We are most interested in the decision boundary $r_y = 1$.

$$r_y(\boldsymbol{x}) = 1 \quad \Longleftrightarrow \quad r_c(\boldsymbol{x}) = 1 \tag{3.36}$$

This means that the true decision boundary $r_y(\boldsymbol{x}) = 1$ is equal to the decision boundary of the bag densities $r_c(\boldsymbol{x}) = 1$. When knowing the bag densities (taking the bag labels as true instance labels), we can learn the correct decision boundary $r_y(\boldsymbol{x})$ *without estimating* $\alpha$. This means that generative supervised learning directly yields the correct decision boundary of sesqui-class learning. (However, if we want to put the decision boundary at a density ratio other than $r = 1$, we have to estimate $\alpha$. Without knowing $\alpha$, we can still optimize the critical density ratio as a hyperparameter. In this case we do not know $\alpha$ or the corresponding critical instance density ratio $r_y$.)

**Relation to discriminative supervised learning** While the above applies for the generative setting where we have access to the densities, most common classifiers are discriminative, i. e. they yield the class probabilities instead of densities. In this case the densities must be determined from the class probabilities and the class priors.

Let us define the following short-hands:

$$p_y = P(y{=}1 \mid \boldsymbol{x}, \boldsymbol{\theta}) \qquad \pi_y = P(y{=}1 \mid \boldsymbol{\theta}) \tag{3.37}$$

$$p_c = P(c{=}1 \mid \boldsymbol{x}, \boldsymbol{\theta}) \qquad \pi_c = P(c{=}1 \mid \boldsymbol{\theta}) \tag{3.38}$$

The Bayes theorem gives following relationship between density ratios $r$, class probabilities $p$, and class priors $\pi$ (valid both on the bag level (index $c$) and on the instance level (index $y$))

$$r = \frac{p}{1-p}\,\frac{1-\pi}{\pi}. \tag{3.39}$$

Furthermore, the instance class prior $\pi_y$ and the bag class prior $\pi_c$ are related via the mixture parameter $\alpha$

$$\pi_y = \alpha\pi_c. \tag{3.40}$$

Inserting (3.39) and (3.40) into (3.34) and rearranging, we obtain

$$\frac{p_y}{1-p_y}\frac{1-\alpha\pi_c}{\alpha\pi_c} = 1+\frac{1}{\alpha}\left(\frac{1-\pi_c}{\pi_c}\frac{p_c}{1-p_c}-1\right) \tag{3.41}$$

$$\frac{p_y}{1-p_y} = \frac{1}{1/\pi_c-\alpha}\left(\alpha-1+\frac{1-\pi_c}{\pi_c}\frac{p_c}{1-p_c}\right). \tag{3.42}$$

This gives an expression for the sought instance class probability $p_y$ in terms of the bag class probability $p_c$, the bag class prior $\pi_c$ and the mixture parameter $\alpha$. The bag class probability $p_c$ can be learned by discriminative supervised learning using the bag labels. The bag class prior can be estimated from the number of instances in positive bags $\pi_c = N^+/N$. $\alpha$ must be estimated. Interesting special cases are

$$\pi_c{=}1/2: \qquad \frac{p_y}{1-p_y} = \frac{\alpha-1+lo_c}{2-\alpha} \tag{3.43}$$

$$lo_c{=}1,\ \pi_c{=}1/2: \qquad \frac{p_y}{1-p_y} = \frac{\alpha}{2-\alpha}. \tag{3.44}$$

The condition $\pi_c = \frac{1}{2}$ can be met by balancing the data before classification. The standard decision boundary is $lo_c = 1$.

This means that the sesqui-class decision boundary is a contour of bag class probability. Without $\alpha$, we do not know which contour is the correct one, but this could be found by optimization on the training data after having learned the discriminative supervised classifier.

It has been experimentally observed by (Ray & Craven 2005) that supervised learning often performs surprisingly well on multiple instance data. To our knowledge, no explanation for this has been given in the literature. The above result explains this observation for the case when sesqui-class learning is the correct model, and the number of instances from positive and negative bags is approximately equal.

**One-class learning of negative bags** We would like to mention one last possibility to approach MI problems: one-class learning of the negative bags. This approach can be seen as an aborted sesqui-class learning. Instead of learning the positive density and a class prior, we just have to estimate a threshold value for the negative density. Regions with lower negative density are classified as positive.

### 3.2.4 Generalizations of the multiple instance model

For completeness we mention two generalizations of the multiple instance model that have been proposed in the literature. A more extensive literature survey can be found in (Foulds & Frank 2010).

**Multiple concepts** (Scott, Zhang & Brown 2005) proposed a generalization of the multiple instance setting where each positive bag must contain at least one instance from each of *multiple* concepts.

This model would be appropriate for example in natural scene classification. Suppose we tried to find images showing a beach. Beach images contain both sand and sea, so it is a good approach to learn two concepts for sand and sea and classify an image as "beach" only if it contains at least one patch classified as "sand" and one patch classifier as "sea".

Naturally, this model requires a multiclass classification on the instance-level. Computationally it is very expensive, which has prevented more widespread use.

**Class ratio labels** Instead of binary bag labels that only indicate if the bag contains at least one instance, it would be more informative to know the number of positive instances. However, in most applications the number of positive datapoints per positive bag is not known but can only be roughly estimated.

Several authors have considered possibilities to control the number of positive instances that will be found by their multiple instance algorithms, see (Bunescu & Mooney 2007), (Gehler & Chapelle 2007), (Li, Tax, Duin & Loog 2013). Our proposed models in Chapter 5 have the same goal, but we take a more general approach based on probabilistic models.

## 3.3 Applications

Multiple instance learning can be applied in several branches of machine learning. In this section we give a brief overview of possible multiple instance applications.

Multiple instance learning is applicable in many cases where we have bag labels relating to many datapoints. Such aggregate labels can be the result of natural variety of the object to be categorized (drug activity prediction), missing labels (image classification, text categorization), or intentional anonymization (data mining). Possible benefits of MI learning over standard supervised learning are reduced labeling effort, reduced computational complexity, and improved classification performance.

An early review of multiple instance applications can be found in (Yang 2005).

### 3.3.1 Drug activity prediction

Drug activity prediction was the first application of MI learning (Dietterich et al. 1997). The task is to predict whether a given drug molecule will bind to a specific binding site of a larger protein. Binding depends not only on the atomic composition and chemical bonds of the drug molecule (primary and secondary structure), but also on its exact geometrical

shape (tertiary structure). The geometrical shape changes dynamically, as some bonds are free to rotate. If the given drug molecule can adopt the correct shape, it will bond and is thus classified as positive or "active".

In the language of multiple instance learning, a drug molecule is a bag, and each of its possible shapes is an instance. Because the "correct" shape is precisely defined by the binding site, one can assume that there is only one small positive region in feature space (shape space). When designing an algorithm specifically for this application, one can make use of this additional information that greatly reduces the complexity of considered classification functions (as done by (Dietterich et al. 1997) and (Maron & Lozano-Pérez 1998)).

### 3.3.2 Image classification

Image data fit quite naturally into the bag-of-instances setting, because an image is usually composed of several different objects or segments (e. g. a car, a house, the street, and the sky). Accordingly, an image is described by a set of feature vectors, one for each image patch or interest point. When classifying images according to their content, it usually suffices if a single image patch shows the object of interest for the whole image to be classified accordingly. This situation corresponds exactly to the MI setting.

A very specific image classification problem is face detection. Multiple instance learning has been applied to face recognition by (Viola, Platt & Zhang 2006), (Zhang & Viola 2007), and (Babenko, Dollár, Tu & Belongie 2008). The survey by (Zhang & Zhang 2010) points out the usefulness MI learning for the reduction of labeling effort in face detection.

Another image classification problem is content-based image retrieval (CBIR). Work on multiple instance learning in this field has been done by (Maron & Ratan 1998), (Zhang, Goldman, Yu & Fritts 2002), (Vijayanarasimhan & Grauman 2008), and others.

Furthermore, the problem of visual tracking has been approached with multiple-instance methods by (Babenko, Yang & Belongie 2009), (Babenko, Yang & Belongie 2011), (Li, Kwok & Lu 2010), (Zeisl et al. 2010), and (Zhang & Song 2013). Finally, multiple instance learning has been applied to image segmentation by (Vezhnevets & Buhmann 2010).

### 3.3.3 Others

Besides the main applications of image classification, multiple instance learning has also been applied in many other fields. Among them are text categorization (Andrews et al. 2002), data mining (Kück & de Freitas 2005), computer security (Ruffo 2000), aerosol prediction (Wang, Radosavljevic, Han, Obradovic & Vucetic 2008), and activity recognition (Stikic & Schiele 2009).

Also, the multiple instance setting has been applied to regression problems by (Ray 2001) and (Zhang & Goldman 2001).

## 3.4 Algorithms

This section gives an overview of multiple instance algorithms. It is ordered roughly chronologically. We start with special purpose techniques that have been designed for the specific application of drug activity prediction (APR, DD, EMDD). They are very restrictive in that they allow for only one convex positive region in feature space, which is inappropriate for most applications. Therefore, recent multiple instance algorithms use modifications of standard classification algorithms like SVMs and random forests.

The difficulty of learning from multiple instance data is the latent variables (unknown instance classes). The full Bayesian solution would require integration over all possible combinations of instance classes, which is infeasible, because there are about $2^{N^+}$ combinations, where $N_+$ is the total number of datapoints in all positive bags (see box below).

Literature reviews of algorithms for multiple instance learning can be found in (Zhou 2004), (Foulds & Frank 2010), and (Amores 2013).

**Instance-level vs. bag-level algorithms** The algorithms discussed below can be divided into two classes, instance-level algorithms and bag-level algorithms.

The natural approach to the MI setting is an instance-level algorithm, because (i) the provided features describe single instances and not whole bags, and (ii) the class-membership of a bag is defined via the class-membership of its instances. Therefore, a direct implementation of the MI setting must be based on an instance-level classifier. In fact, most proposed algorithms follow this approach, and typically, this results in a two-step iterative algorithm that alternates between estimation of the latent instance labels and standard supervised classification with the estimated labels. (In diverse density, the latent instance classes are not estimated but integrated over.)

However, some authors have also proposed bag-level classifiers. The idea behind this is to first construct bag-wise features from the given instance features in a preprocessing step. Then, the bags are classified directly without decomposition into instances.

### 3.4.1 Axes-parallel rectangles (APR)

The APR-algorithm has been the first published MI-algorithm. It is a special-purpose classifier that has been devised for the specific application of drug activity prediction. It assumes that the positive region in feature space is a single axes-parallel rectangle, all instances outside this rectangle are negative.

Actually, three different algorithms have been proposed by (Dietterich et al. 1997): A noise-tolerant "standard" algorithm, an "outside-in" algorithm, and an "inside-out" algorithm. Tests showed that the latter algorithm clearly outperforms all others, so we will consider only this one and refer to it as the APR-algorithm.

This algorithm comprises two steps. The first step searches the smallest possible APR that explains all bags. This is achieved by a combined iterative search for (i) the true positive instance of each positive bag and (ii) the most discriminative features. The true positive instances are found in each iteration by an exhaustive backfitting search. In the

**APR-algorithm (pseudo-code)**

| | |
|---|---|
| start with arbitrary datapoint | |
| **until convergence** (usually 3-4 iterations) | |
| **for each positive bags** | search best witnesses |
| find point from all unexplained bags that can be explained by least enlarging the APR | |
| enlarge APR | |
| **for each explained bag** | backfitting |
| check if any other point of explained bag allows for smaller APR, if yes, replace points | |
| calculate discriminative matrix (number of features)-by-(number of negative points) | search best features |
| **until all negative points are discriminated against** | |
| find feature that discriminates most points | |
| drop corresponding points from list | |
| **for each feature** | enlarge APR |
| choose kernel width, using user-supplied $\tau$ | |
| perform 1D kernel density estimate | |
| enlarge APR along current feature, so that left and right kde-tails are $\epsilon/2$ | |

second step, the minimal APR is enlarged to improve generalization performance, which is done according to a data-dependent heuristic, based on a kernel density estimate.

APR discards the non-positives, so it is an implementation of the dMI-model.

### 3.4.2 Diverse density (DD)

Like APR, diverse density by (Maron & Lozano-Pérez 1998) has been devised for the specific application of drug activity prediction, and it is a realization of dMI. It models the instance-wise class probability as a single non-normalized Gaussian in feature space

$$P(y_n{=}1 \,|\, \boldsymbol{x}_n, \theta) = \exp\Big((\boldsymbol{x}_n - \boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x}_n - \boldsymbol{\mu})\Big) \tag{3.45}$$

$$P(y_n{=}0 \,|\, \boldsymbol{x}_n, \theta) = 1 - \exp\Big((\boldsymbol{x}_n - \boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x}_n - \boldsymbol{\mu})\Big) \tag{3.46}$$

It is possible to predefine the covariance matrix $\Sigma$ (usually as isotropic) and to optimize only the position $\boldsymbol{\mu}$. More commonly, however, $\Sigma$ is modeled as diagonal and its components are

optimized together with $\boldsymbol{\mu}$.

The main contribution of diverse density is that it is based on a well-founded probabilistic model of the MI setting. Given the instance-wise class probabilities $p_n = P(y_n = 1 \,|\, \boldsymbol{x}_n, \theta)$ of all instances in one bag, the bag-wise class probability reads:

$$P(c=1 \,|\, \boldsymbol{p}) \;=\; 1 - \prod_n (1 - p_n) \tag{3.47}$$

$$P(c=0 \,|\, \boldsymbol{p}) \;=\; \prod_n (1 - p_n). \tag{3.48}$$

Inserting (3.45) and (3.46) into (3.47) and (3.48) and multiplying all bags yields the complete likelihood of the diverse density model

$$\mathcal{L}(\boldsymbol{\mu}, \Sigma) \;=\; \prod_b P(c_b \,|\, \boldsymbol{p}_b). \tag{3.49}$$

The diverse density algorithm searches for the maximum likelihood by a two-step gradient ascent (Maron 1998). In the first step, $\boldsymbol{\mu}$ is optimized at constant $\Sigma$, in the second step, $\Sigma$ is optimized at constant $\boldsymbol{\mu}$. $\boldsymbol{\mu}$ is initialized at an instance from a positive bag. The algorithm is run multiple times, until each instance from any positive bag has been used as starting point. This heuristic is costly, but makes it very likely that the global optimum is found.

It should be noted that the name "diverse density" is a misnomer. By "density" one usually understands a probability distribution over feature space $P(\boldsymbol{x} \,|\, \dots)$. What is called "diverse density", however, is in fact the likelihood $P(\mathcal{D} \,|\, \mu, \Sigma)$, taken as a function of $\mu$. Like a density, this function is defined on feature space and large values indicate a positive region, but conceptually, the two things have nothing in common. The diverse density model is actually a discriminative model, so no density is ever modeled or calculated.

The intention behind the naming might have been to distinguish between the densities $P(x \,|\, c=1)$ and $P(x \,|\, y=1)$. The former is the "standard" positive density that arises if all instances from positive bags are taken as truly positive. The latter is the "true" positive density that would be modeled in a generative multiple instance model (as we will do in Chapter ??). But again, this has nothing to do with the above "diverse density" model.

### 3.4.3 Diverse density with expectation maximization (EM-DD)

The speed of the diverse density algorithm described above is moderate, because the objective function (3.49) is quite costly to evaluate. (While the product over all negative bags (3.48) can be reduced to a sum of logarithms, this is not possible for the positive bags (3.47)). It would be much simpler if it were known which instances from the positive bags are truly positive.

EM-DD attempts to find the true positive instances via expectation maximization (Zhang & Goldman 2001). The E-step finds the most positive instance (for each positive bag), while the M-step optimizes the parameters considering only the most positive instance from each

positive bag (which is standard supervised learning). Explicitly:

$$i_j^{(t)} = \arg\max_{i_j} P(y_{ji_j} \mid \boldsymbol{x}_{ji_j}, \boldsymbol{\mu}^{(t)}, \Sigma^{(t)}) \qquad \text{E-step} \qquad (3.50)$$

$$\left(\boldsymbol{\mu}^{(t+1)}, \Sigma^{(t+1)}\right) = \arg\max_{\mu,\Sigma} \prod_j P(y_{ji^{(t)}} \mid \boldsymbol{x}_{ji^{(t)}}, \boldsymbol{\mu}^{(t)}, \Sigma^{(t)}) \qquad \text{M-step} \qquad (3.51)$$

From a theoretical viewpoint, DD would be the preferred choice over EM-DD, because it performs exact Bayesian inference (integration over latent variables), whereas EM-DD uses an approximate estimate of the latent variables. In practice however, EM-DD does not have worse performance than DD, and its reduced computational cost is significant.

### 3.4.4 Multiple instance SVMs

Support vector machines were one of the first general-purpose classifiers that have been adapted to multiple instance learning. (Gärtner, Flach, Kowalczyk & Smola. 2002) suggested to construct kernels on the bag-level and then classify the bags directly with an SVM. They proposed two different kernels, one based on the sum of all feature vectors within a bag, and one based on the component-wise minimum and maximum of all features vectors. They showed that these bag-level kernel can separate the same concepts as their corresponding instance-level kernels. The drawback of this approach is that the bag-level kernel is quite arbitrary and that the classifier does not output the truly positive instances (i. e. which of the instances of a positive bag is responsible for the positive label).

A more direct implementation of the MI-setting with SVMs was proposed by (Andrews et al. 2002). They explicitly estimate the hidden instance labels and apply an SVM on the instance-level. The difficulty is to find the correct instance labels which is a mixed integer programming problem. An efficient approach is a two step iterative algorithm that alternatingly (i) estimates the hidden instance labels given the discriminant function and (ii) finds the optimum discriminant function given the hidden instance labels.

(Gehler & Chapelle 2007) proposed deterministic annealing as a method to find the correct instance labels. DA is a special case of a homotopy method to find global optima. Homotopy methods construct a sequence of functions starting with a simple (convex) one and ending with the true objective function. Each function is optimized using the optimum of the previous function as initial value. For a good sequence of functions, the path of optima can lead to the global optimum (although this is not guaranteed). Deterministic annealing constructs the function sequence by adding a convex entropy term. The scaling factor of the entropy term can be interpreted as a temperature. (In contrast to simulated annealing, deterministic annealing carries out a minimization during each step, not a probabilistic sampling.

(Gehler & Chapelle 2007) observed that DA yields better optima than minimizing the objective function directly. However, this does not directly yield better classification performance, because the success of the MI model depends on the true structure of the data. They proposed to introduce the instance class ratio as an additional user-defined parameter. By

manual optimization of this parameter the performance could be improved over (Andrews et al. 2002).

The work of (Gärtner et al. 2002) was extended by (Bunescu & Mooney 2007). They proposed two variants of SVM-constraints for Gärtner's kernel based on sums of feature vectors. These constraints are designed to more exactly represent the multiple instance model. Moreover, they proposed "sparse balanced MIL" which features an additional user-defined parameter that describes the expected ratio of positive and negative instances in the positive bags.

According to (Han, Tao & JueWang 2010), many MI methods tend to produce false positives rather than false negatives in natural scene classification. They attributed this to the fact that positive images often contain image patches that are related to the positive concept, but are in fact negative (e. g. a waterfall is often accompanied by mist, but mist does not define the concept waterfall). There is the danger of falsely classifying the mist as "waterfall" which leads to false positives if the test set contains images of mist or clouds without waterfall. To avoid this, (Han et al. 2010) propose a projection constraint for each positive bag that encourages large variance of the bags instances perpendicular to the decision boundary, or vice versa, that encourages the decision boundary be perpendicular to the direction of greatest variance of a bags instances. They report good performance, but conceptually it is somewhat unclear under which circumstances the projection constraint is appropriate and whether there are detrimental side-effects.

(Li et al. 2013) proposed a single-mixture-parameter approach based on an SVM. The idea is to assign positive labels to all instances from positive bags, train a standard SVM, and adjust the obtained class probabilities with the help of an estimate of the percentage of true positives. Furthermore, (Li et al. 2013) propose several heuristics based on ideas of classifier combination in order to translate the obtained instance-wise class probability to a bag class. The reason for this remains a bit unclear, however, since there is actually a well-defined and feasible solution to MI-bag classification given instance-wise class probabilities, as shown by (Andrews et al. 2002).

### 3.4.5 Multiple instance learning based on decision trees

Using decision trees for multiple instance learning was first proposed by (Ruffo 2000). Later works include (Chevaleyre & Zucker 2001) and (Blockeel, Page & Srinivasan 2005).

In recent times, ensembles of trees (e. g. random forests) have become much more popular in machine learning than single decision trees. Still, we are aware of only one work that proposes a random forest-based multiple instance algorithm (Leistner, Saffari & Bischof 2010). They proposed an iterative algorithm similar to our proposal in Chapter 6. In addition, they borrowed the idea of (Gehler & Chapelle 2007) to use deterministic annealing to overcome local minima. According to our experience, however, the random nature of random forests is already sufficient to prevent trapping in local minima (see results in Chapter 6). Therefore, we believe that deterministic annealing is not needed, but rather slows down convergence.

### 3.4.6 Others

There have been proposals for multiple instance algorithms based on many different supervised learning methods.

One of the first general multiple instance algorithms, which is based on $k$-nearest neighbor, has been proposed by (Wang & Zucker 2000). Other early work on multiple instance learning has been done on neural nets by (Ramon & De Raedt 2000) and by (Zhou & Zhang 2002).

More closely related to our present work are ensemble methods like the one proposed by (Zhou & Zhang 2003), which is based on classifier combination. Other multiple instance ensemble methods based on boosting have been proposed by (Andrews & Hofmann 2004) and by (Viola et al. 2006).

Finally, a new idea for multiple instance learning is conditional random fields, see (Deselaers & Ferrari 2010).

# Chapter 4

# Improving Multiple Instance Classification

The challenging part of classification is usually the training stage (i. e. parameter optimization or evaluation of parameter posterior), while testing new data with a learned classifier is mostly straightforward. Accordingly, little attention has been paid in the literature to the task of classifying unknown bags in the multiple instance setting. However, as will be shown in this chapter, there is some room to improve the calculation of bag class probability from given instance class probabilities.

In Section 4.1 we show that bag classification of the standard multiple instance model depends on the bag size: the larger the bag, the larger the probability that it is classified as positive. In many applications this behavior is not appropriate. Therefore we propose a bag size independent version of the multiple instance model, which is equivalent to the standard multiple instance model during training, but does not exhibit a bag size dependent bias during testing.

In Section 4.2 we consider multiple instance learning with ensemble classifiers. Since ensemble classifiers provide multiple versions of trained classifiers, each version can be used to predict the complete bag. As we will show, this method differs from the standard approach of averaging over ensemble members at the instance level and applying the bag model only afterwards. If the predicted instance class probabilities are not i. i. d. but correlated, the standard "noisy-OR" approach yields a wrong result, but our proposed method of "treewise"[1] bag classification is still correct.

## 4.1 Bag size independent multiple instance classification

In this section we will propose and examine a variation of the multiple instance model. It differs from the standard multiple instance model by a factor that depends on the bag size and on the mean of the predicted instance class probabilities over *all* bags.

To motivate the approach we first introduce the notions of generative and discriminative bag models, and we find that the standard multiple instance model is discriminative. Then we consider a generative version of the multiple instance model, which has the property that it is "a-priori" bag size independent. For bag classification, however, the posterior distribution is relevant. To render the model bag size independent after training, we have to adjust the normalization factor depending on the mean outcome of the instance classifier.

[1] in reference to the random forest classifier that we use in Chapter 6 and whose ensemble members are trees

Finally we test the model on synthetic data. In Section 6.3.2 we will show that it also works on real datasets.

### 4.1.1 Bag size dependent bias

To inspect the bag size dependent bias of the discriminative multiple instance model, we plot the predicted bag class probability for different bag sizes. As derived in Section 3.1, the bag class probability is given by the "noisy-OR" formula (3.8), which we reprint here for readability:

$$P(c{=}1 \,|\, \boldsymbol{x}_n, \boldsymbol{\theta}) \;=\; 1 - \prod_n (1 - p_n) \;=\; \mathrm{OR}_n(p_n) \tag{4.1}$$

$$p_n \;=\; P(y_n{=}1 \,|\, \boldsymbol{x}_n, \boldsymbol{\theta}) \tag{4.2}$$

We inspect two special cases: (i) all instance class probabilities are equal $p_n = p_{\mathrm{neg}}$ and (ii) all instance class probabilities can take on one of two values $p_n = p_{\mathrm{neg}}$ or $p_n = p_{\mathrm{pos}}$.

$$P(c{=}1 \,|\, \boldsymbol{x}_n, \boldsymbol{\theta}) \;=\; 1 - (1 - p_{\mathrm{neg}})^{N_b} \tag{4.3}$$

$$P(c{=}1 \,|\, \boldsymbol{x}_n, \boldsymbol{\theta}) \;=\; 1 - (1 - p_{\mathrm{neg}})^{N_b^-} \cdot (1 - p_{\mathrm{pos}})^{N_b^+} \tag{4.4}$$

Formula (4.3) represents a negative bag with $N_b$ instances and false positive instance rate $p_{\mathrm{pos}}$. Formula (4.4) represents a positive bag with $N_b^+$ positive instances, $N_b^-$ negative instances, and a true positive instance rate of $p_{\mathrm{pos}}$.

The corresponding plots are shown in Figures 4.1 and 4.2. In all cases the bag class probability tends to $P(c{=}1 \,|\, \boldsymbol{x}, \boldsymbol{\theta}) = 1$ for increasing bag size. Even negative bags with a

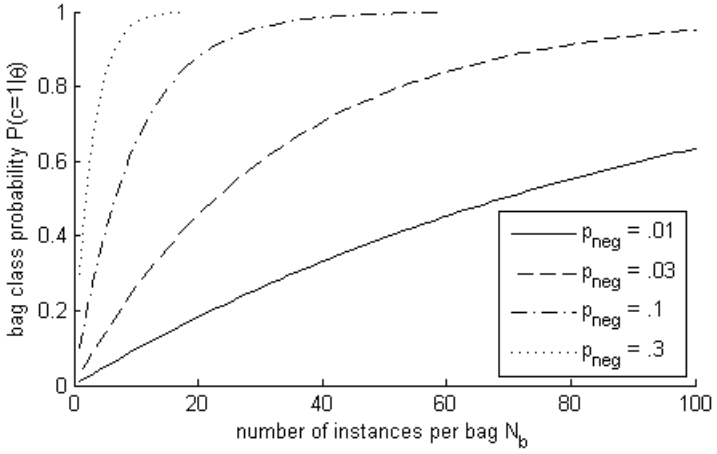


Figure 4.1: False positive rate of negative bags as a function of bag size $N_b$ and instance class probability $p_{\mathrm{neg}}$ (false positive rate of negative instances).

false positive instance rate of just $p_{\text{neg}} = 0.01$ will be predicted as positive if the bag size exceeds 80 instances. For positive bags which have at least one instance with large positive probability $p_{\text{pos}}$, the bag class probability will always be larger than $p_{\text{pos}}$.

**Bag class prior** Of specially interest is the prior situation, i. e. the situation before any data has been observed or before the classifier has been trained. We can assume an unprejudiced instance classifier with a uniform instance class prior

$$P(y) \;=\; \int P(\boldsymbol{x}, y, \boldsymbol{\theta})\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{\theta} \;=\; \begin{cases} 1/2 & \text{for } y{=}0 \\ 1/2 & \text{for } y{=}1. \end{cases} \tag{4.5}$$

The corresponding bag class prior is given by (4.3) with $p_{\text{neg}} = 1/2$

$$P(c) \;=\; \begin{cases} \left(\frac{1}{2}\right)^N & \text{for } c{=}0 \\ 1 - \left(\frac{1}{2}\right)^N & \text{for } c{=}1, \end{cases} \tag{4.6}$$

which shows a heavy preference towards positive for large bags.

### 4.1.2 Generative, discriminative, and general bag models

**Discriminative MI model** The multiple instance model as described above regards the instance labels $\boldsymbol{y}$ as given. Hence it models the conditional probability $P(c \,|\, \boldsymbol{y})$, and not the joint probability $P(\boldsymbol{y}, c)$. The instance labels $\boldsymbol{y}$ are defined solely by the instance classification model $P(\boldsymbol{x}_n, y_n, \boldsymbol{\theta})$. This model structure is shown in Figure 4.3. In analogy to discriminative classifiers, we call this model the *discriminative* multiple instance model,

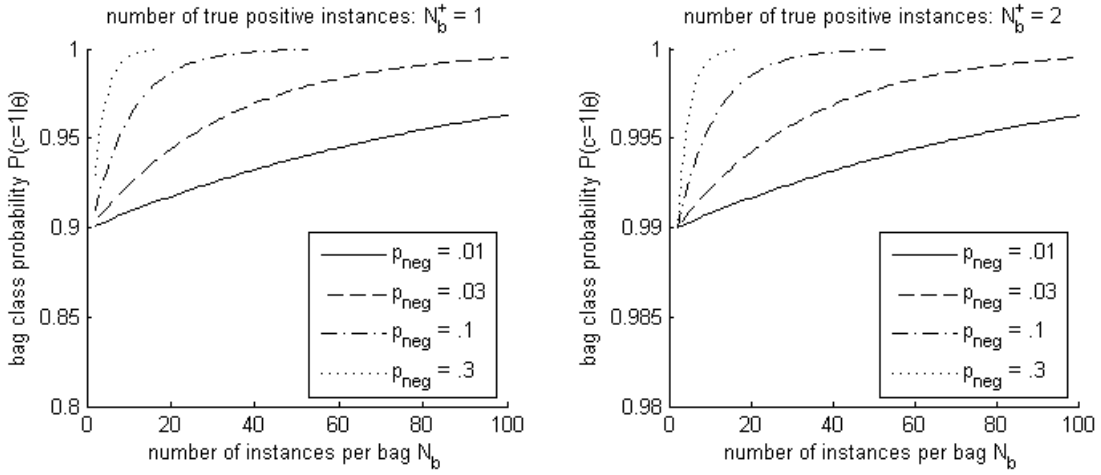


Figure 4.2: False negative rate of positive bags. Left: $N_b^+ = 1$, right: $N_b^+ = 2$. Note the different scales of $y$-axes. Class probability of true positive instances $p_{\text{pos}} = 0.9$.

since it allows to discriminate between bag classes $c$ for given instance classes $\boldsymbol{y}$, but cannot generate new instance classes.

We would like to construct a model that satisfies the multiple instance definition *and* yields a uniform bag class prior. With a discriminative bag model this is not possible, because the constraint $\int P(c \,|\, \boldsymbol{y}) \, \mathrm{d}c = 1$ together with the MI property (3.2) uniquely defines the standard discriminative MI model described above. So we need to relax this constraint and model the joint probability $P(\boldsymbol{y}, c)$, which is a more general bag model.

**General bag models** We need to be careful, however, when combining such a general bag model with a classifier, because – loosely spoken – both the bag model and the classifier "try to define $\boldsymbol{y}$". More formally, the problem is that the two marginal distributions $P(\boldsymbol{y}) = \int P(\boldsymbol{y}, c) \, \mathrm{d}c$ and $P(\boldsymbol{y}) = \int P(\boldsymbol{X}, \, \mathrm{d}\boldsymbol{y}, \boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{X} \, \mathrm{d}\boldsymbol{y}$ are in general not equal. To set up a valid model, it is reasonable define the complete joint as the normalized product

$$P(\boldsymbol{X}, \boldsymbol{y}, c, \boldsymbol{\theta}) \;\; = \;\; \frac{1}{Z} Q_{\mathrm{MI}}(\boldsymbol{y}, c) \prod_n q_{\mathrm{Cl}}(\boldsymbol{x}_n, y_n, \boldsymbol{\theta}), \tag{4.7}$$

but doing this one should keep in mind that the single model factors $Q_{\mathrm{MI}}$ and $q_{\mathrm{Cl}}$ are not equal anymore to the probabilities which they are intended to represent

$$Q_{\mathrm{MI}}(\boldsymbol{y}, c) \;\; \neq \;\; P(\boldsymbol{y}, c) \;=\; \int P(\boldsymbol{X}, \boldsymbol{y}, c, \boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{X} \, \mathrm{d}\boldsymbol{\theta} \tag{4.8}$$

$$q_{\mathrm{Cl}}(\boldsymbol{x}_n, y_n, \boldsymbol{\theta}) \;\; \neq \;\; P(\boldsymbol{x}_n, y_n, \boldsymbol{\theta}) \;=\; \int P(\boldsymbol{X}, \boldsymbol{y}, c, \boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{X}_{\bar{n}} \, \mathrm{d}\boldsymbol{y}_{\bar{n}} \, \mathrm{d}c, \tag{4.9}$$

where the subscript $\bar{n}$ denotes the vector where component $n$ has been omitted. We have discussed these general bag models already in Section 2.5.2, and here we will make use of the results.

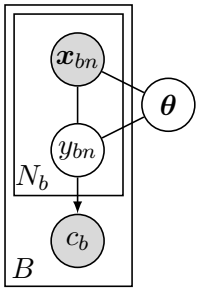


Figure 4.3: Graphical model of the standard "discriminative" multiple instance model.

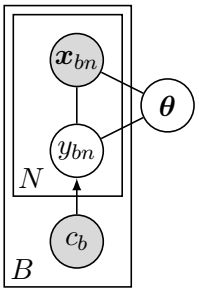


Figure 4.4: Graphical model of the proposed "generative" multiple instance model.

**Generative bag models** Modeling the joint $P(\boldsymbol{y}, c)$ under the constraint (4.10) is equivalent to modeling the conditional $P(\boldsymbol{y} \,|\, c)$. In analogy to classifiers, we call such a bag model a *generative* bag model, because it allows to generate new instance labels $\boldsymbol{y}$ for a given bag label $c$. The corresponding graphical model is shown in Figure 4.4.

### 4.1.3 The generative multiple instance model

**Model definition** To set up a MI model with uniform bag class prior, we start with the discriminative MI model (3.5) and renormalize it to satisfy

$$\int Q_{\mathrm{MIgen}}(\boldsymbol{y}, c)\, \mathrm{d}\boldsymbol{y} \;=\; 1, \tag{4.10}$$

which ensures a uniform bag class prior

$$\begin{aligned} P(c) &= \int P(\boldsymbol{X}, \boldsymbol{y}, c, \boldsymbol{\theta})\, \mathrm{d}\boldsymbol{X}\, \mathrm{d}\boldsymbol{y}\, \mathrm{d}\boldsymbol{\theta} && (4.11)\\ &= \frac{1}{Z} \int Q_{\mathrm{MI}}(\boldsymbol{y}, c) \prod_n \left( \int q_{\mathrm{Cl}}(\boldsymbol{x}_n, y_n, \boldsymbol{\theta})\, \mathrm{d}\boldsymbol{x}_n\, \mathrm{d}\boldsymbol{\theta} \right) \mathrm{d}\boldsymbol{y} && (4.12)\\ &= \frac{1}{Z} \int Q_{\mathrm{MI}}(\boldsymbol{y}, c)(1/2)^N\, \mathrm{d}\boldsymbol{y} && (4.13)\\ &= \frac{1}{2} \qquad \text{for } Z = (1/2)^{N-1}. && (4.14) \end{aligned}$$

The result of the renormalization is

| $Q_{\mathrm{MIgen}}$ | $\boldsymbol{y}=\boldsymbol{0}$ | $\boldsymbol{y}\neq\boldsymbol{0}$ |
|---|---|---|
| $c=0$ | 1 | 0 |
| $c=1$ | 0 | $\frac{1}{2^N-1}$ |

(4.15)

and as an example we write out the result for a bag containing two instances $N = 2$:

| $Q_{\mathrm{MIgen}}(c=0, \boldsymbol{y})$ | $y_2=0$ | $y_2=1$ |
|---|---|---|
| $y_1=0$ | 1 | 0 |
| $y_1=1$ | 0 | 0 |

| $Q_{\mathrm{MIgen}}(c=1, \boldsymbol{y})$ | $y_2=0$ | $y_2=1$ |
|---|---|---|
| $y_1=0$ | 0 | 1/3 |
| $y_1=1$ | 1/3 | 1/3 |

(4.16)

**Training the generative MI model** The difference between the discriminative MI model (3.2) and the generative MI model (4.15) is just the factor of $1/(2^N - 1)$ on positive bags. During training the bag labels are given, and this factor is canceled in the posterior. So for the training stage, the generative and the discriminative MI model are equivalent.

**Classification** To evaluate the bag class probability of the generative multiple instance model, we insert the definition (4.15) into the equations for inference in general bag models

(2.56, 2.58), using the notation (4.2). The result is

$$M_{\text{MIgen}}(c) = \begin{cases} \prod_n (1-p_n) & \text{for } c=0 \\ \frac{1-\prod_n (1-p_n)}{2^N-1} & \text{for } c=1 \end{cases} \tag{4.17}$$

$$P_{\text{MIgen}}(c{=}0 \,|\, \boldsymbol{X}, \boldsymbol{\theta}) = \frac{M_{\text{MIgen}}(c{=}0)}{M_{\text{MIgen}}(c{=}0) + \frac{1-M_{\text{MIgen}}(c=0)}{2^N-1}} \tag{4.18}$$

$$= \frac{1}{1 + \frac{1/M_{\text{MIgen}}(c=0)-1}{2^N-1}}. \tag{4.19}$$

To compare this result to the discriminative MI model, we show plots in Figures 4.5 and 4.6, which are to be compared to Figures 4.1 and 4.2. As required, a bag with undecided instances $p_{\text{neg}} = 0.5$ has undecided bag class $P(c{=}1) = 0.5$. If the instances are (slightly) positive $p_{\text{neg}} > 0.5$ the bag class probability increases with bag size, but for (slightly) negative instances $p_{\text{neg}} < 0.5$ the bag class probability decreases with bag size. Note that the curves in Figure 4.5 are not symmetric: the curve with $p_{\text{neg}} = 0.75$ approaches 1 faster than the curve with $p_{\text{neg}} = 0.25$ approaches 0. Additional instances with large instance class probability shift the "initial position" upwards, but they can be overruled by many negative instances Figure 4.6. For example, a bag containing two instances with $p_n = 0.75$ and ten instances with $p_n = 0.4$ would be classified as negative (dotted line in right plot of Fig. 4.6).

Figure 4.7 shows the bag class probability as a function of a single instance class probability. For the discriminative MI model (left plot) the dependence is always linear and the bag is classified as positive ($P(c \,|\, \boldsymbol{p}) > 0.5$) for all shown settings. For the generative MI

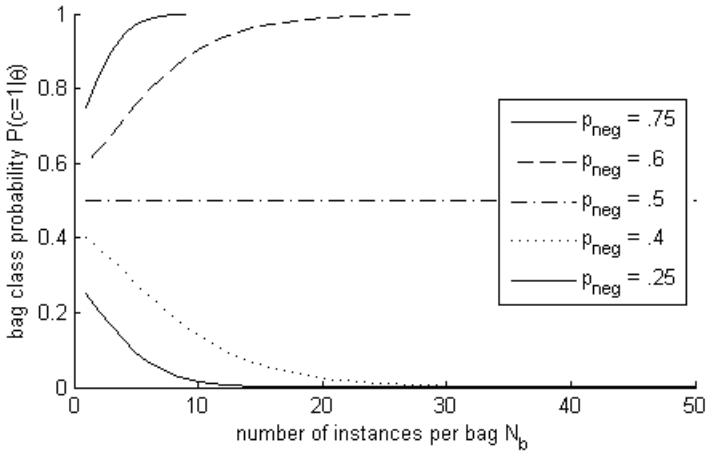


Figure 4.5: Bag class probability of the generative MI model as a function of bag size $N_b$ and instance class probability $p_{\text{neg}}$.

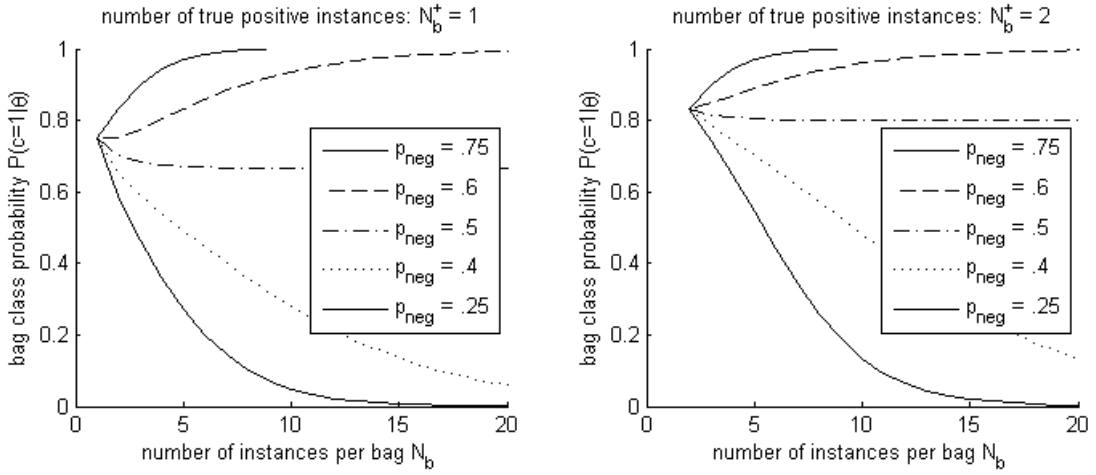


Figure 4.6: Bag class probability of generative MI model with additional instances with class probability $p_{\text{pos}} = 0.75$. Left: one additional instance, right: two additional instances.

model (right plot) the dependence is nonlinear. As long as the instance class probability is small (negative instances), small changes do not affect the bag class probability much. But when the instance class probability tends to 1 (positive instance), the bag class probability tends to 1 as well, regardless of the other instances in the bag, as is required by the MI

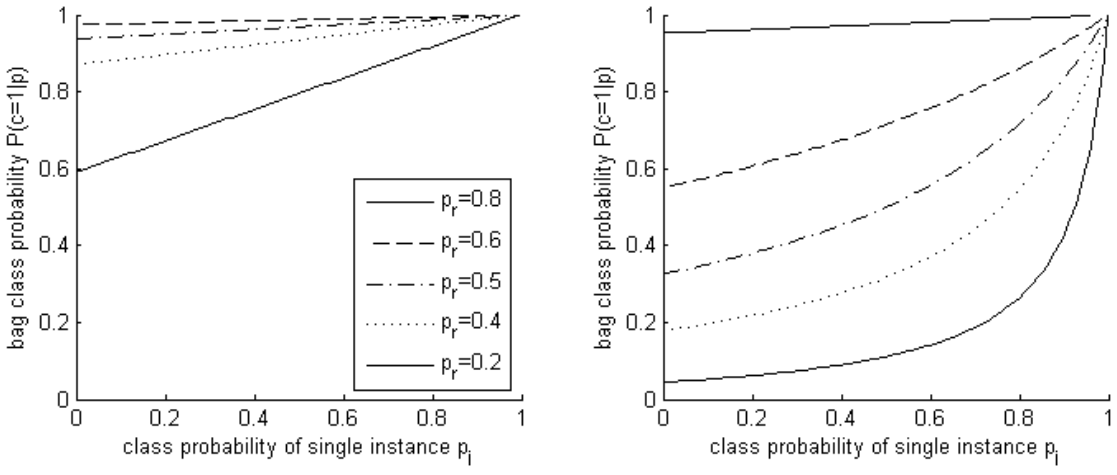


Figure 4.7: Bag class probability as a function of a single instance class probability. Left: discriminative MI model, right: generative MI model. Bag size $N = 5$, $p_r$: instance class probability of 4 remaining instances.

property.

Let us consider the lower solid line. It corresponds to a bag with five instances, four of which have been assigned a positive class probability of 0.2 by the instance classifier. If the fifth instance is assigned a positive class probability of 0.9, the bag is still classified as negative, because of the four other instances. It needs a positive class probability larger than ca. 0.95 to overrule the other instances and classify the complete bag as positive. Note, however, that if an instance is *known* to be positive ($p_j = 1$), its bag is always classified positive, no matter how many negative instances there are, as is required by the MI property.

### 4.1.4 Bag size independent MI model

The generative MI model satisfies the requirement that the bag class probability $P(c)$ is independent of bag size for the prior instance class probabilities $P(y_n) = 1/2$. For classification, however, the relevant quantity is not the prior but the posterior instance class probabilities $P(y_n \,|\, \mathcal{D}) = p_n$.

For optimum classification, we seek a model that is independent of bag size *after* training. We can achieve this by choosing a bag size dependent normalization factor that is different from the "generative" normalization factor $F(N) \neq 2^N - 1$ (cf. Eq. 4.15).

**Derivation of normalization factor F(N)** As a first step, let us determine the general normalization factor $F(\boldsymbol{p})$ so that $P(c \,|\, \boldsymbol{p}) = 1/2$. From equations (4.17) and (4.19) it follows easily that

$$P(c \,|\, \boldsymbol{p}) = \frac{M(c)}{M(c{=}0) + M(c{=}1)} \stackrel{!}{=} 1/2 \qquad \Longleftrightarrow \tag{4.20}$$

$$M(c{=}0) = M(c{=}1) \tag{4.21}$$

$$\prod_n (1 - p_n) = \frac{1 - \prod (1 - p_n)}{F(\boldsymbol{p})} \tag{4.22}$$

$$F(\boldsymbol{p}) = \frac{1}{\prod_n (1 - p_n)} - 1 \tag{4.23}$$

Of course, this is not reasonable, because in this case *any* bag, positive or negative, will have $P(c \,|\, \boldsymbol{p}) = 1/2$, which does not allow for classification.

The normalization factor $F$ should not depend on the instance classification results $\boldsymbol{p}$, but only on bag size $F(N)$. Let us assume that the $p_n$ are i. i. d. with distribution $P(p)$. Then we can write the product in (4.23) as a sum of logarithms, and replace the summand with the weighted integral. We obtain

$$F(N) = \frac{1}{\exp\left(\sum_n \ln(1 - p)\right)} - 1. \tag{4.24}$$

$$= \exp\left(-N \int P(p) \cdot \ln(1 - p) \, \mathrm{d}p\right) - 1. \tag{4.25}$$

The distribution $P(p)$ can be estimated from the training data. There are two plausible ways: take all instances from negative and positive bags $P(p)$, take instances from negative bags only $P(p \mid c=0)$. We prefer to use $P(p \mid c=0)$, because this is more in line with the MI model. In this case, negative bags will have $P(c \mid \boldsymbol{p}, p_0) = 1/2$, independent of bag size, while positive bags will have $P(c \mid \boldsymbol{p}, p_0) > 1/2$. Using $P(p)$ a positive bag would have to have a certain *ratio* of positive points.

**Parametrization and special cases** In effect, the normalization is an additional degree of freedom of the model. It is useful to introduce a single parameter that specifies the normalization. A convenient parameter is the instance class probability $p_{1/2}$ for which the bag class probability $P(c \mid \boldsymbol{p})$ equals $1/2$, independent of bag size.

$$P(c \mid \boldsymbol{p}=[p_{1/2}, p_{1/2}, \ldots, p_{1/2}]) = \frac{1}{2} \tag{4.26}$$

$$\tag{4.27}$$

Comparison of (4.23) with (4.24), and (4.25) yields

$$F(N, p_{1/2}) = \left(\frac{1}{1-p_{1/2}}\right)^N - 1 \tag{4.28}$$

$$p_{1/2} = 1 - \exp\left(-\int P(p) \cdot \ln(1-p)\, \mathrm{d}p\right) \tag{4.29}$$

$$p_{1/2} = 1 - \exp\left(\frac{1}{N}\sum_n \ln(1-p_n)\right), \tag{4.30}$$

The graphical model of the bsiMI model with the additional parameter $p_{1/2}$ is shown in Figure 4.8.

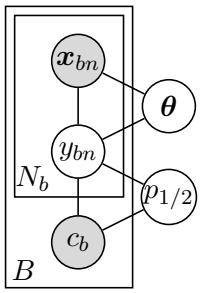


Figure 4.8: Graphical model of the extended "bsiMI" model (bag size independent MI).

In principle, we would have to optimize the new parameter $p_{1/2}$ during training together with all other parameters $\boldsymbol{\theta}$. For practical purposes, however, it is completely satisfactory to specify $p_{1/2}$ after training by using (4.29). In our implementation we chose to use only negative instances from negative bags for calculation of the mean value in (4.29). The reason is that we expect positive instances to have a large scatter of predicted class probability which could impair the method. Choosing $p_{1/2}$ so that the prediction of all negative bags is bag size independent should work as well and is probably more stable.

Note that for $p_{1/2} = 1/2$, the bsiMI model is equal to the generative MI model. For $p_{1/2} = 0$ and $p_{1/2} = 1$, the equations degenerate because of division by zero. In cases where the trained classifier assigns exactly zero class probability to negative instances ($p_n = 0\ \forall n$), we should use the discriminative MI model.

### 4.1.5 Assessment on synthetic data

To assess the potential improvements of the bag size independent MI model compared to the standard discriminative MI model, we test it on synthetic instance probabilities $p_n$. We prepared 1000 positive and 1000 negative bags. The bag sizes are distributed according to the maximum entropy distribution $P(N_b) \propto \exp(((N_b - A)/B)^2)$ with $A$ and $B$ chosen so that the mean value is $\mu = 10$ and the variance is $\sigma = 3$. Each positive bag contains 3 positive instances (corresponding to a ratio of$\alpha = 0.3$). For the instance probability $p_n$ we chose a Beta distribution with a mean of $\mu = 0.3$ (negative instances) and $\mu = 0.6$ (positive instances) and variance $\sigma = 0.1$ for both positive and negative instances. Histograms of the dataset are shown in Figure 4.9.

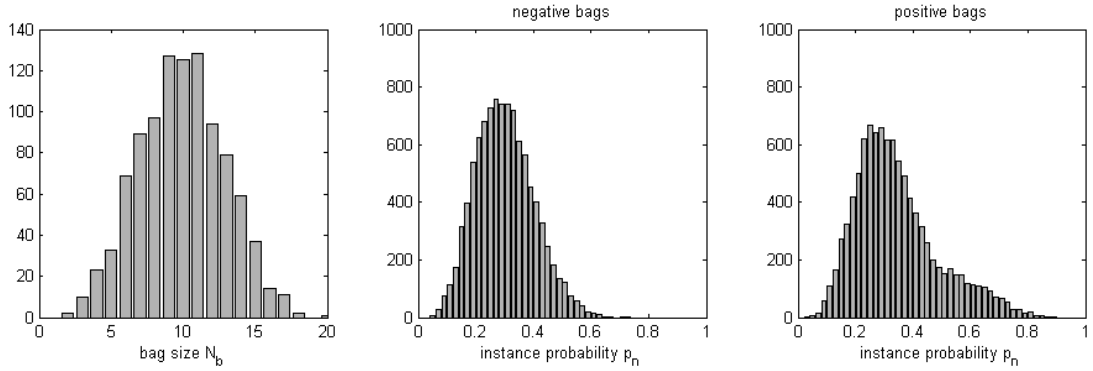


Figure 4.9: Synthetic data for the assessment of the generative and bag size independent MI models. Left: histogram of bag sizes $\mu = 10$, $\sigma = 3$. Middle: class probabilities of instances in negative bags $\mu = 0.3$, $\sigma = 0.1$. Right: class probabilities of instances in positive bags $\alpha = 0.3$ $\mu = 0.6$, $\sigma = 0.1$.

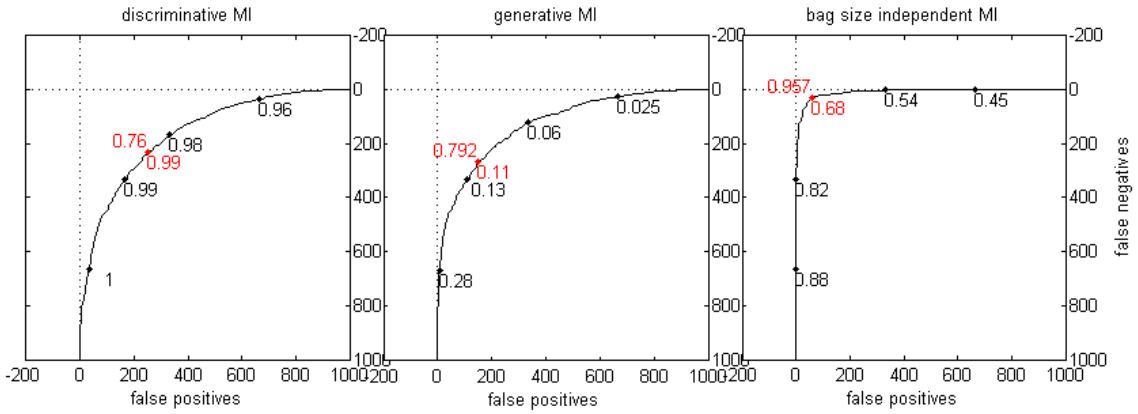


Figure 4.10: Receiver operator characteristic (ROC) of discriminative, generative, and bag size independent MI models on synthetic data.

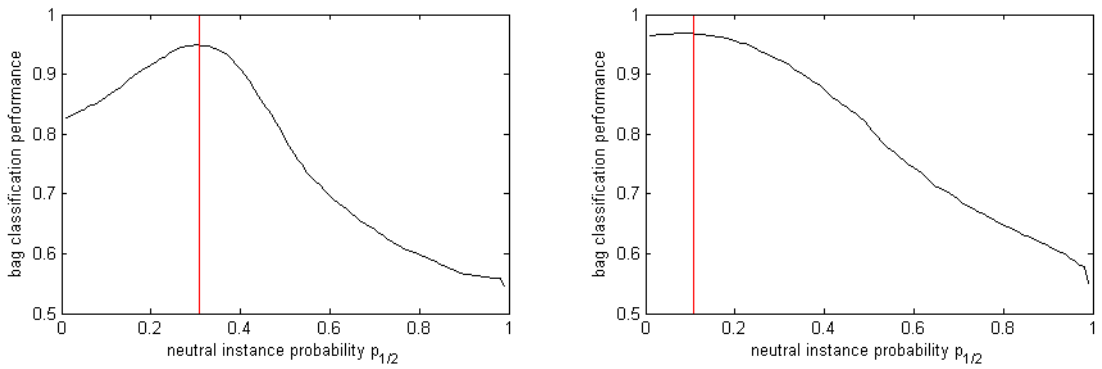


Figure 4.11: Classification performance as a function of neutral instance probability $p_{1/2}$. Left: synthetic data shown in Figure 4.9; right: synthetic data with $\mu_{\neg} = 0.1$, $\mu_{\text{pos}} = 0.1$ and 1 positive instance per positive bag. The red line indicates the estimate (4.29) using the instances from negative bags.

The bag classification performance of the discriminative, the generative, and the bag size independent MI model on the synthetic dataset is shown in Figure 4.10. The bag size independent MI model clearly gives the best result. The corresponding neutral instance probability is $p_{1/2} = 0.308$.

To check the estimation of neutral instance probability $p_{1/2} = 0.30761$, we plot the classification performance as a function of $p_{1/2}$ in Figure 4.11. The optimum is indeed at (or very close to) our estimate.

## 4.2 Multiple instance classification with ensemble classifiers

When using an ensemble classifier with a bag model, there are two plausible procedures for classifying an unknown bag: The ensemble average can be taken either at the instance level or at the bag level.

The standard procedure is to take the ensemble average at the instance level. In this case, the ensemble classifier represents a probabilistic instance classifier, i. e. the ensemble vote is taken as the instance class probability. Thus the ensemble classifier can easily be replaced by a different probabilistic instance classifier.

From a theoretical viewpoint, however, the correct procedure is to take the ensemble average at the bag level. This means that we evaluate the bag model separately for the prediction of each ensemble member, and only afterwards take the ensemble average.

We discuss both procedures from a general viewpoint and examine the specific example of using the random forest (ensemble classifier) with the multiple instance model (bag model).

### 4.2.1 Ensemble average at bag level

According to the model structure of discriminative multiple instance learning shown in Figure 4.12, the bag class probability during testing is

$$P(c \,|\, \boldsymbol{X}, \mathcal{D}) = \int P(c, \boldsymbol{y}, \boldsymbol{\theta} \,|\, \boldsymbol{X}, \mathcal{D}) \, \mathrm{d}\boldsymbol{y} \, \mathrm{d}\boldsymbol{\theta} \tag{4.31}$$

$$= \int P(c \,|\, \boldsymbol{y}) \int \prod_n P(y_n \,|\, \boldsymbol{x}_n, \boldsymbol{\theta}) P(\boldsymbol{\theta} \,|\, \mathcal{D}) \, \mathrm{d}\boldsymbol{\theta} \, \mathrm{d}\boldsymbol{y}, \tag{4.32}$$

where $\boldsymbol{X}$ stands for the set of all input vectors $\boldsymbol{X} = \{\boldsymbol{x}_n\}_n$. This is a special case of the general class probability in bag models (2.58) where there is no interaction between the model factors.

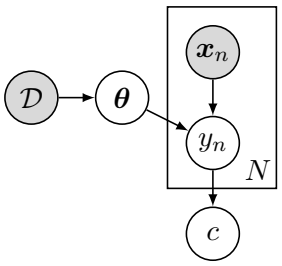


Figure 4.12: Bayesian network for multiple instance classification. The training stage is simplified to describe the posterior parameter distribution $P(\boldsymbol{\theta} \,|\, \mathcal{D})$.

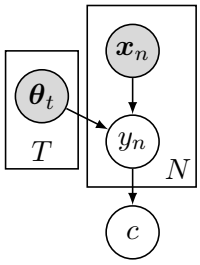


Figure 4.13: Bayesian network for multiple instance classification with an ensemble classifier.

The single terms of (4.32) are given as follows:

$$P(\boldsymbol{\theta} \,|\, \mathcal{D}) = \frac{1}{T} \sum_t \delta(\boldsymbol{\theta}_t) \tag{4.33}$$

$$P(y_n \,|\, \boldsymbol{x}_n, \boldsymbol{\theta}_t) = \delta(y_{nt}) \qquad y_{nt} \in \{0, 1\} \tag{4.34}$$

$$P(c \,|\, \boldsymbol{y}) = \delta(c(\boldsymbol{y})) \qquad c(\boldsymbol{y}) = \max_n y_n \tag{4.35}$$

Eq. (4.33) defines an ensemble classifier (compare to Figure 4.13), Eq. (4.34) describes the fact that each ensemble member $t$ provides a hard output for each instance $n$ (deterministic class prediction, no class probability), and Eq. (4.35) is the (deterministic) multiple instance model. Inserting these equations one after the other into (4.32) yields

$$P(c \,|\, \boldsymbol{X}, \mathcal{D}) = \int P(c \,|\, \boldsymbol{y}) \, \frac{1}{T} \sum_t \prod_n P(y_n \,|\, \boldsymbol{x}_n, \boldsymbol{\theta}_t) \, \mathrm{d}\boldsymbol{y} \tag{4.36}$$

$$= \frac{1}{T} \sum_t P(c \,|\, \boldsymbol{y}_t) \tag{4.37}$$

$$= \frac{1}{T} \sum_t \max_n y_{nt} \tag{4.38}$$

It is worth to state the result in words: For each ensemble member $t$ we evaluate the multiple instance model to obtain "hard" predictions of the bag class $\hat{c}_t = \max_n y_{nt}$; afterwards, we take the ensemble average a "soft" bag class probability $P(c \,|\, \boldsymbol{X}, \mathcal{D})$. In the following we will call this procedure the *treewise MI* method (in reference to the random forest, where the ensemble members are trees).

### 4.2.2 Ensemble average at instance level

Instead of the above procedure, another plausible idea is to apply the ensemble average (i. e. integration over $\boldsymbol{\theta}$) to each instance separately, instead of applying it to the complete bag at once. This procedure is illustrated in Figure 4.14. Inference is divided into two steps: First we infer the instance class probabilities $p_n = P(y_n = 1)$, afterwards we apply the bag model to the $p_n$.

The effect of this division is that it cancels the correlations between instance classes $y_n$ (or their class probabilities $p_n$). While each single ensemble member predicts a product distribution $P(\boldsymbol{y} \,|\, \boldsymbol{\theta}) = \prod_n P(y_n \,|\, \boldsymbol{\theta})$, the sum (or weighted integral) of many product distri-

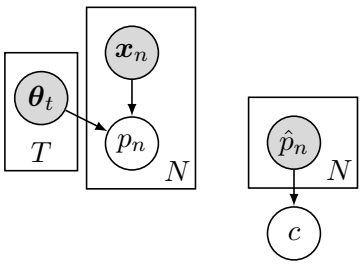


Figure 4.14: Bayesian network for multiple instance classification by applying the ensemble average at the instance level.

butions is in general *not* a product distribution anymore

$$P(\boldsymbol{y}) \;=\; \int \prod_n P(y_n \,|\, \boldsymbol{\theta}) P(\boldsymbol{\theta}) \,\mathrm{d}\boldsymbol{\theta} \;\;\neq\;\; \prod_n q(y_n) \tag{4.39}$$

for any $q$. However, a plausible approximation for $P(\boldsymbol{y})$ is the product of its marginals $P(\boldsymbol{y}) \approx \prod_n P(y_n)$.

Formally, this procedure is described by interchanging the inner integral with the product in (4.32), which leads to

$$P'(c \,|\, \boldsymbol{X}, \mathcal{D}) \;=\; \int P(c \,|\, \boldsymbol{y}) \prod_n \int P(y_n \,|\, \boldsymbol{x}_n, \boldsymbol{\theta}) P(\boldsymbol{\theta} \,|\, \mathcal{D}) \,\mathrm{d}\boldsymbol{\theta} \,\mathrm{d}\boldsymbol{y}. \tag{4.40}$$

Inserting the ensemble predictions (4.33, 4.34) yields

$$P'(c \,|\, \boldsymbol{X}, \mathcal{D}) \;=\; \int P(c \,|\, \boldsymbol{y}) \prod_n \frac{1}{T} \sum_t y_{nt} \,\mathrm{d}\boldsymbol{y}, \tag{4.41}$$

and inserting the multiple instance model (4.35) finally yields the well-known "noisy-OR" result which we have already encountered in Section 3.1

$$P'(c{=}1 \,|\, \boldsymbol{X}, \mathcal{D}) \;=\; 1 - \prod_n (1 - p_n) \tag{4.42}$$

$$p_n \;=\; \frac{1}{T} \sum_t y_{nt}. \tag{4.43}$$

Again, we state the result in words: We use the ensemble predictions to obtain a class probability $p_n = P(y_n = 1)$ for each instance $n$; afterwards, we apply the "probabilistic" multiple instance model (noisy-OR) to obtain the bag class probability $P(c \,|\, \boldsymbol{X}, \mathcal{D})$. In the following we will refer to this procedure as the *noisy-OR* method.

**Threshold method** For completeness we would like to state a third plausible procedure. Instead of using the instance-wise class probabilities $\hat{p}_n$ to calculate the bag class prediction, one might make a class decision $\hat{y}_n$ on the instance level and only afterwards apply the bag model. In the graphical model this difference can be expressed by replacing the node $\hat{p}_n$ with $\hat{y}_n$ (see Figure 4.15). The corresponding general formula is

$$P''(c \,|\, \boldsymbol{X}, \mathcal{D}) = \tag{4.44}$$
$$\int P(c \,|\, \boldsymbol{y}) \prod_n \left[ \int P(y_n \,|\, \boldsymbol{x}_n, \boldsymbol{\theta}) P(\boldsymbol{\theta} \,|\, \mathcal{D}) \,\mathrm{d}\boldsymbol{\theta} \right] \,\mathrm{d}\boldsymbol{y},$$

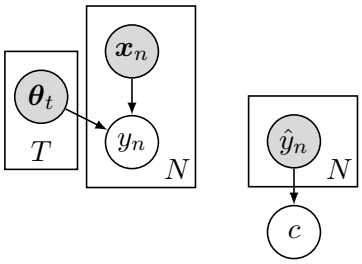


Figure 4.15: Bayesian network for multiple instance classification with the "threshold method".

where square brackets denote rounding (cf. (4.32) and (4.40)). Since $y_n$ can take on only two values $y_n \in \{0, 1\}$, the probability for one of the values is always $> 1/2$ while the probability for the other value is $< 1/2$; if $P(y_n = 0 \,|\, \boldsymbol{x}_n, \boldsymbol{\theta}) = P(y_n = 1 \,|\, \boldsymbol{x}_n, \boldsymbol{\theta}) = 1/2$, we define $P(y_n = 0 \,|\, \boldsymbol{x}_n, \boldsymbol{\theta}) = 1$. Inserting (4.33–4.35) yields

$$P''(c \,|\, \boldsymbol{X}, \mathcal{D}) = \delta(\hat{c}) \tag{4.45}$$

$$\hat{c} = \max_n \hat{y}_n \tag{4.46}$$

$$\hat{y}_n = \begin{cases} 0 & \text{if } 1/T \sum_t y_{nt} \leq 1/2 \\ 1 & \text{if } 1/T \sum_t y_{nt} > 1/2. \end{cases} \tag{4.47}$$

This representation is overly complex. We can put it in one line as follows

$$c = 1 \qquad \text{iff} \qquad \max_n \left( \sum_t y_{nt} \right) > \frac{T}{2}. \tag{4.48}$$

In words: A bag is classified as positive iff the tree count of any instance is larger than $T/2$. Hence we will refer to this procedure as the *threshold* method.

### 4.2.3 Overview of bag classification methods

In the section above we have focused on the theoretical derivation of the different procedures for bag classification. In this subsection we would like to give a more practical overview.

It is convenient to consider the following three steps that have to be carried out to classify a bag when given the instance-wise predictions of an ensemble classifier:

1. evaluation of bag model, i. e. integration over the latent instance classes $\boldsymbol{y}$
2. taking the ensemble average
3. taking the class decision, i. e. rounding the class probability to 0 or 1

The different bag classification methods correspond to different orders in which these steps are carried out. Not all orders make sense, for example class decision cannot be the first step. But there are three reasonable orders that are listed in Table 4.1.

| | **treewise MI** | **noisy-OR** | **threshold** |
|---|---|---|---|
| **step 1** | bag model | ensemble average | ensemble average |
| **step 2** | ensemble average | bag model | class decision |
| **step 3** | class decision | class decision | bag model |
| $P(c \,\|\, \{y_{nt}\})$ | $\frac{1}{T} \sum_t \left( \max_n y_{nt} \right)$ | $1 - \prod_n \left( 1 - \frac{1}{T} \sum_t y_{nt} \right)$ | $\max_n \left( \frac{1}{T} \sum_t y_{nt} \right)$ |

Table 4.1: Overview of bag classification methods.

The name "treewiseMI" has been chosen with reference to the random forest, since the random forest's ensemble members $\boldsymbol{\theta}_t$ are trees, and because we will the random forest use in Chapter 6. A more general (but also more awkward) name would be "ensemble-member-wise", which does not seem suitable.

**Example** To illustrate that the three methods can lead to very different results, we give one example. Consider the following instance class predictions of an ensemble classifier

| $y_{nt}$ | $t=1$ | $t=2$ | $t=3$ |
|---|---|---|---|
| $n=1$ | 1 | 0 | 0 |
| $n=2$ | 0 | 1 | 0 |
| $n=3$ | 0 | 0 | 1 |

(4.49)

Using the treewise MI method (4.38), each tree finds 1 positive instance (among 2 negative ones) and hence classifies the bag as positive. With three trees having classified the bag as positive, the final classification is positive, too, and seems to have a large confidence.

The noisy-OR method (4.42) yields the soft outputs $p_n = 1/3 \quad \forall n$, which leads to a bag probability of $P(c=1) = 1 - (2/3)^3 = 19/27 > 1/2$. The bag is classified as positive, but the confidence is low.

Using the threshold method (4.48), all instances $n$ are classified as negative (2 to 1 majority vote), so the whole bag is classified as negative, too.

### 4.2.4 Experimental results

We tested the three different bag classification methods on synthetic data, i. e. we simulated the raw output of an ensemble classifier $\{y_n t\}$ with specified properties (bag size, instance class ratios, etc.), then we compared the results of bag classification on this input.

As discussed in Section 4.2.2, we expect the largest difference between the methods for raw data with correlations between instance classes $y_n$. That's why we first examine the influence of correlation on the predicted bag class probabilities $P(c \,|\, y_{nt})$. The result for a relevant setting (bag class probability between 0 and 1) is shown in Figure 4.16. Without correlation (top row), both "treewise MI" and "noisy-OR" predict the correct value, while "threshold" strongly underestimates the bag class probability. With correlation (bottom row), the true bag class probability increases. This is due to the fact that with correlation, the positive instances are evenly distributed among bags, so there are more bags that have at least one positive instance. As expected, the only method that captures this effect correctly is "treewise MI", while both "noisy-OR" and "threshold" now underestimates the bag class probability.

Next we take a look at the scatter of the predictions. The scatter of "noisy-OR" is somewhat smaller than that of "treewise MI" and "threshold", which might be related to the fact that "noisy-OR" is a smooth method that does not require to take the maximum (cf. table in Section 4.2.3). Figure 4.17 shows the results for a reduced ensemble size of $T = 11$,

which confirms the above statements. Note that the predictions of "treewise MI" and "threshold" are restricted to fractions of ensemble votes $P(c\,|\,y_{nt}) = t/T$; the fluctuations in the histogram of "noisy-OR" are also an artifact of applying noisy-OR on such fractions.

For final performance, only the correct decision is important regardless of the exact predicted value of bag class probability. Figure 4.18 shows the dependence of the ratio of false bag predictions on the ratio of false instance predictions by each ensemble member. The settings of the left plot represent a small positive bag with three instances (they are the same as for Figure 4.17), the settings of the right plot represent a medium-sized negative bag. Positive bags are assumed to have correlated instance, negative bags are assumed to be uncorrelated. Note the different scales on the $x$-axis: The multiple instance model is very robust to false negative instances (unless there is only a single positive instance per

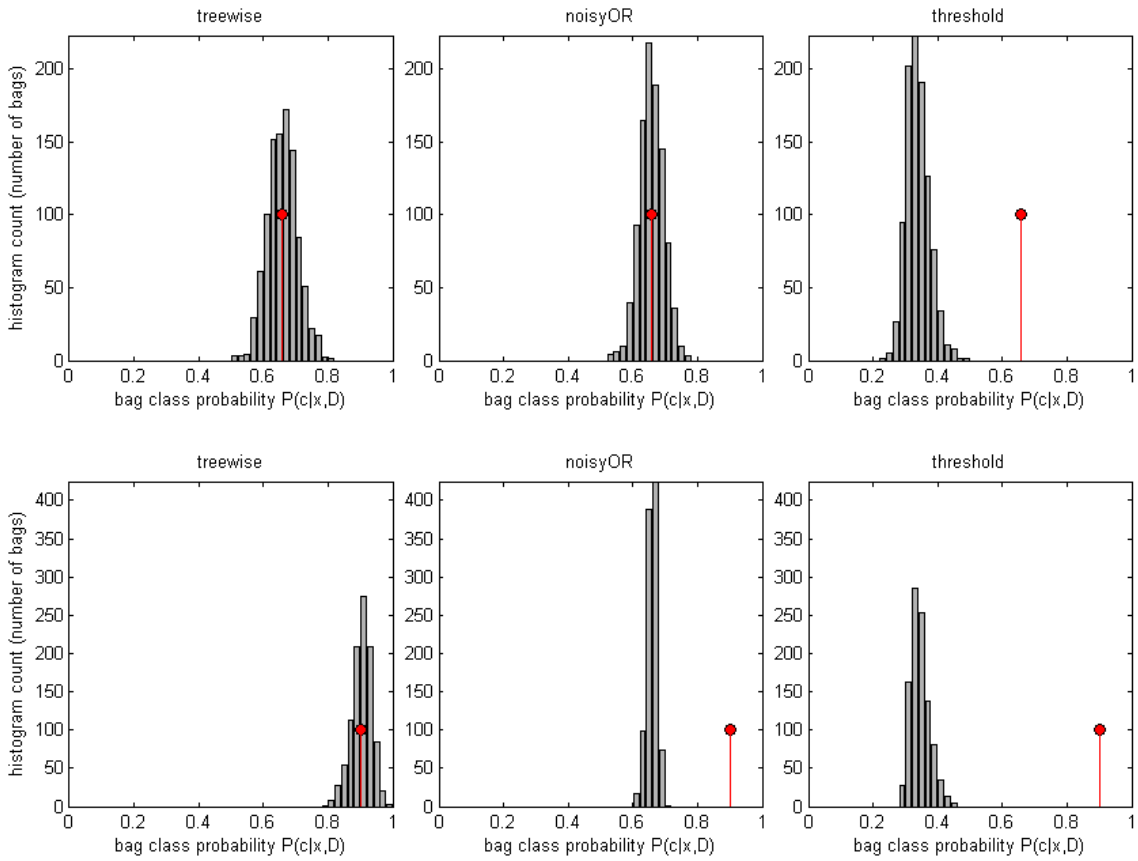


Figure 4.16: Predicted bag class probabilities of different classification methods. Each plot shows the histogram counts of 1000 simulated bags and indicates the true value by a red stem. Top row: no correlation between instance classes $y_n$, bottom row: strongest possible correlation. Settings: bag size $n = 3$, instance class probability $P(y_n = 1) = 0.3$, number of ensemble members $T = 101$ (odd number to avoid problems of undecided vote).

bag), but very sensitive to false positive instances.

"Treewise MI" and "noisy-OR" have the same false positive rate, but "treewise MI" has somewhat lower false negative rate and is therefore to be preferred. The "threshold" method always underestimates the bag class probability, and hence it has much lower false positive rate but much larger false negative rate than the other two methods.

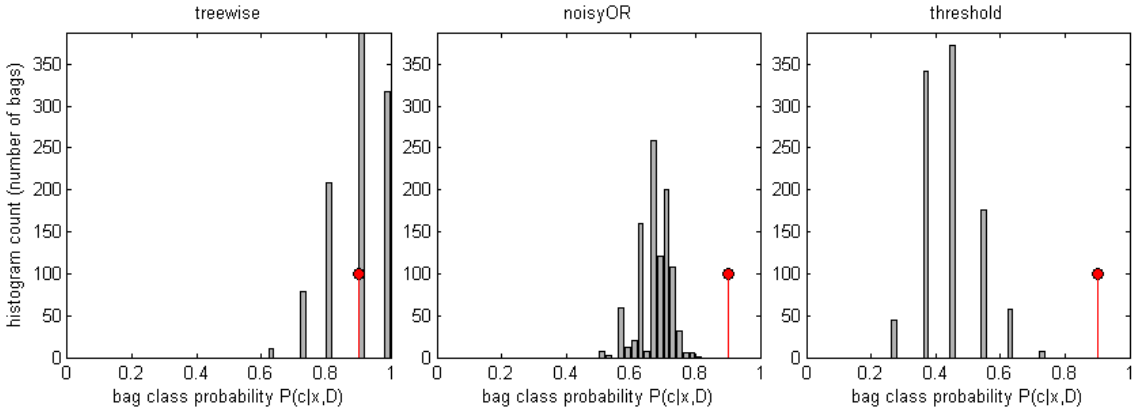


Figure 4.17: Predicted bag class probabilities of different classification methods with small ensemble ($T = 11$). Each plot shows the histogram counts of 1000 simulated bags and indicates the true value by a red stem. Settings: bag size $n = 3$, instance class probability $P(y_n = 1) = 0.3$, maximum correlation.

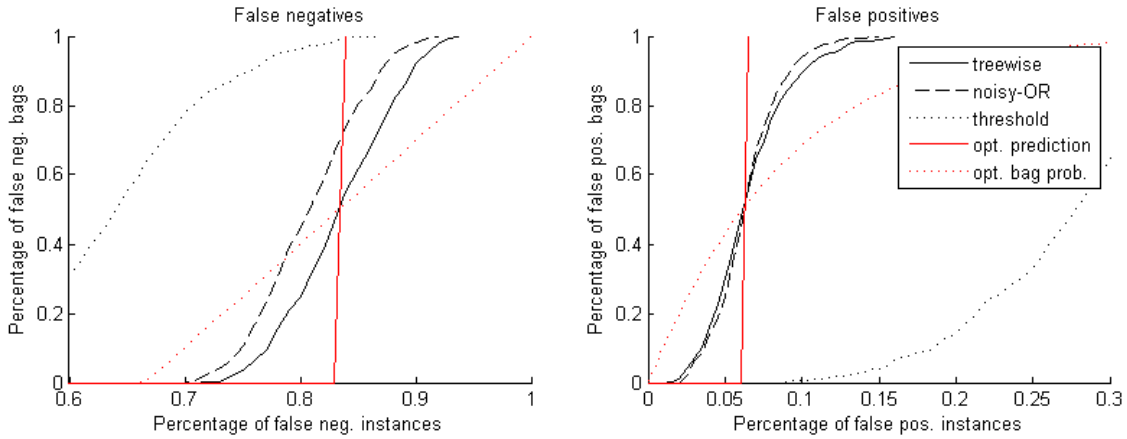


Figure 4.18: Percentage of misclassified bags for different classification methods. The red curves show the true bag probability and the corresponding optimal decision. Settings: number of ensemble members $T = 11$, number of positive instances per positive bag $n_{pos} = 3$ (relevant for left plot), number of negative instances per negative bag $n_{neg} = 11$ (relevant for right plot), decision threshold: 0.5. For each datapoint 500 bags have been simulated.

The misclassification rates shown in Figure 4.18 have been calculated for the standard (and theoretically correct) decision threshold of $P(c) = 0.5$. In practice, however, instances from different classes might have different misclassification rates, bags from different classes might have different misclassification cost, or the model might not exactly represent the situation at hand. All these reasons lead to an unwanted classification bias that one can correct by optimizing the decision threshold after training. The dependence of classification performance on the decision threshold is commonly depicted by the *receiver operator characteristic* (ROC).

We show the ROC-curves of all three methods for a relevant setting in Figure 4.19). Most notably, the "threshold" method performs surprisingly well; optimizing the decision boundary compensates its bias that we observed in the previous figures. Without correlation between instance classes (top row), "noisy-OR" performs best due to its low scatter (cf. Fig. 4.16). With correlation, the performance of all methods increases because of reduced scatter, but as expected, "treewise MI" profits most from the correlation because it models it correctly, an has best performance with correlation.

Simulations have also been conducted with different settings (more instances, smaller positive instance probability), but the results have always been similar to the above.

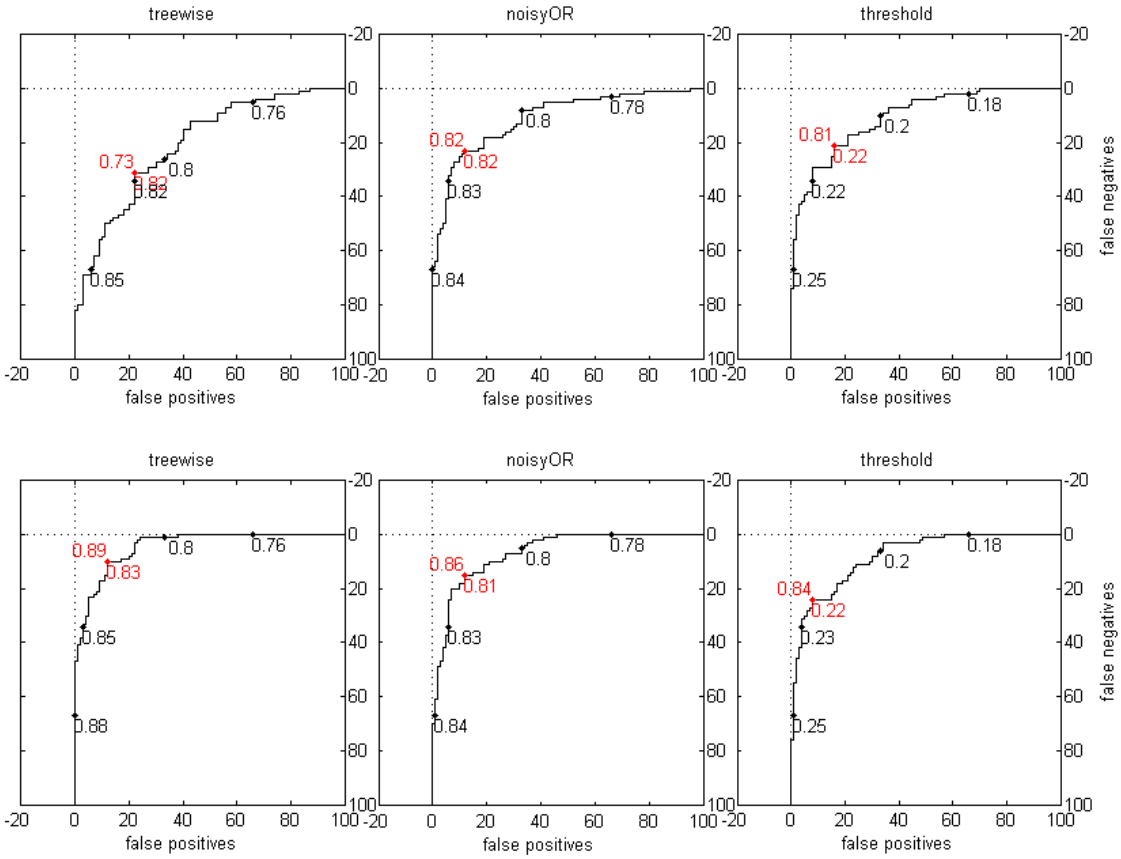


Figure 4.19: Receiver operator characteristics of different classification methods. Top row: no correlation between positive instances, bottom row: strongest possible correlation. The optimum accuracy and corresponding classification threshold of each curve is shown in red, further thresholds corresponding to different points of each curve are shown in black. Settings: number of ensemble members $T = 101$, numbers of instances per bag $[n_{pos}, n_{nonpos}, n_{neg}] = [3, 8, 11]$, corresponding instance class probabilities $[p_{pos}, p_{nonpos}, p_{neg}] = [0.2, 0.13, 0.13]$, number of bags: $[M_{pos}, M_{neg}] = [100, 100]$.

# Chapter 5

# Alternative Bag Models for Multiple Instance Applications

As we have seen in Chapter 3, the multiple instance model can be applied in many different settings. For many of them, we can assume that there is not only a single positive instance in each positive bag, but typically there is more than one positive instance.[1] We have seen above that estimating the instance classes is difficult and that there often is some ambiguity regarding the location of the decision boundary. The SCL model estimates a maximum number of instances as positive, while the tight dMI (single witness) model estimates a minimum number of positive instances.

Besides these fixed models, it is also possible to set up a parametrized model that can interpolate between the extreme cases. In this chapter we present two such bag models, and we show that they can improve performance on some datasets that are typically considered as "multiple instance datasets". We would like to point out that the additional parameter(s) that control the number of positive instances per positive bag is not intended to be optimized during training as a "hyperparameter", but is intended to be set by the user before training based on prior information about the application or the data.

## 5.1 Bernoulli model

### 5.1.1 Model definition

[1] A bag with a single positive instance is still be classified as positive, but it can be assumed as an extreme case, while for most bags we expect a larger number of positive instances.

The idea of the Bernoulli model is to add a factor for each instance that increases its probability to be positive. The corresponding graphical model is shown to the right. The Bernoulli factor is defined by

| $q_{\text{Bern}}(y, c, \beta)$ | $y=0$ | $y=1$ |
|---|---|---|
| $c=0$ | 1 | 0 |
| $c=1$ | $1-\beta$ | $\beta$ |

(5.1)

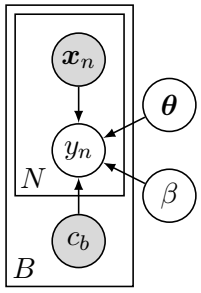


Figure 5.1: Baysian network of the generative Bernoulli model.

A negative bag ($c=0$) allows negative instances only, a positive bag ($c=1$) allows both positive and negative instances. To increase the instance's probability to be positive, $\beta$ must be larger than $1/2$. For $\beta = 1$, the Bernoulli model is equivalent to standard supervised learning. For $\beta = 1/2$ the Bernoulli factor would have no effect, and for $\beta < 1/2$ it would decrease the instance's probability to be positive. Sometimes it is convenient to write $\beta = 0$ for negative bags.

The Bernoulli model suggests itself because of its simplicity. The point is that it factorizes over instances, i. e. the factor (5.1) exists once for each instance, not once per bag. It is instructive to write down the product of (5.1) for a complete bag. For a bag containing two instances we have

| $Q_{\text{Bern}}(c=0, \boldsymbol{y})$ | $y_2=0$ | $y_2=1$ |
|---|---|---|
| $y_1=0$ | 1 | 0 |
| $y_1=1$ | 0 | 0 |

| $Q_{\text{Bern}}(c=1, \boldsymbol{y})$ | $y_2=0$ | $y_2=1$ |
|---|---|---|
| $y_1=0$ | $(1-\beta)^2$ | $\beta(1-\beta)$ |
| $y_1=1$ | $\beta(1-\beta)$ | $\beta^2$ |

(5.2)

For a bag containing an arbitrary number of instances we have

$$Q_{\text{Bern}}(c=1, \boldsymbol{y}) \quad = \quad \beta^{N_b^+} (1-\beta)^{N_b^-}, \tag{5.3}$$

with $\beta=0$ for $c=0$, where $N_b^+$ is the number of (estimated) positive instances in this bag, and $N_b^-$ is the number of (estimated) negative instances.

### 5.1.2 Model properties

To study the properties of the Bernoulli model, we need to state the complete bag classification model (i. e. instance classifier plus bag model). Let $p_n = P(y_n = 1 \,|\, \boldsymbol{x}_n, \boldsymbol{\theta})$ be the predictions of the instance classifier. Combining these predictions with the Bernoulli bag model yields

| $q_{\text{Cl}} \cdot q_{\text{Bern}}$ | $y=0$ | $y=1$ |
|---|---|---|
| $c=0$ | $1-p_n$ | 0 |
| $c=1$ | $(1-\beta)(1-p_n)$ | $\beta p$ |

(5.4)

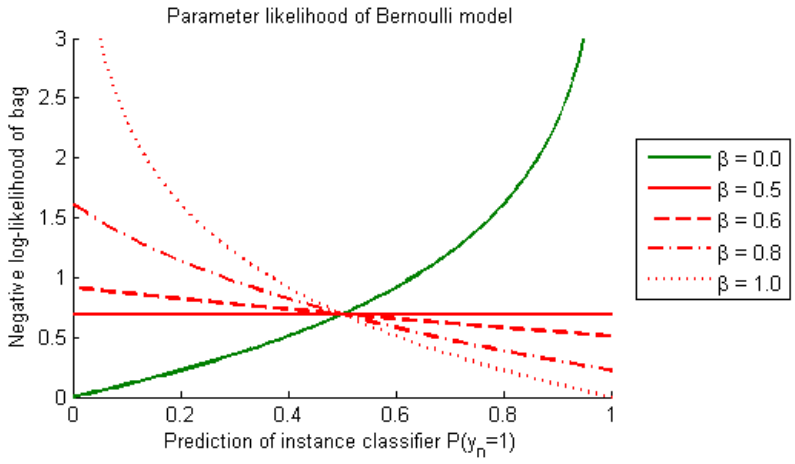


Figure 5.2: Contribution of a single instance from a negative bag (green) or a positive bag (red) to the parameter likelihood $P_{\text{Bern}}(\boldsymbol{X}, c \,|\, \boldsymbol{\theta})$, given the prediction of the instance classifier $p_n = P(y_n{=}1)$.

As described in Section 2.5.2, it is convenient to write down the marginal over the latent variables $M_{\boldsymbol{y}}$, because both the parameter posterior (needed for training) and the bag class probability (needed for testing) are simple expressions of $M_{\boldsymbol{y}}$ (cf. Eqs. 2.56–2.58) Inserting (5.4) into the general formula for factorizing models (2.13), we obtain

$$M_{\text{Bern}}(c{=}0) \;=\; \prod_n (1 - p_n) \tag{5.5}$$

$$M_{\text{Bern}}(c{=}1) \;=\; \prod_n \big[\beta p_n + (1-\beta)(1-p_n)\big]. \tag{5.6}$$

**Training** According to (2.57), the likelihood of a bag model is proportional to the y-marginal: $P(\boldsymbol{\theta} \,|\, \boldsymbol{X}, c) \propto M_{\text{Bern}}(c)$. To show the effect of the Bernoulli model on the learned parameters, we plot (5.6) as a function of instance class probability in Figure 5.2. The logarithmic scale is convenient, so that the contributions of the single instances are additive.

The solid lines correspond to the multiple instance model (which is equivalent to $\beta = 0$ for negative bags and $\beta = 1/2$ for positive bags). For positive bags, the MI model is completely indifferent regarding the class-membership of instances from positive bags. In order to induce the instance classifier to estimate instances from positive bags as positive, one must set $\beta > 1/2$ (dashed and dotted lines).

**Testing** The central motivation for the Bernoulli model is to change the training behavior of the algorithm, i. e. to induce the instance classifier to estimate more instances as positive and assign a larger region of feature space to the positive class. However, the model change

from the MI model to the Bernoulli model has consequences for testing as well, as we will show in the following. [2]

We obtain the bag class probability of the Bernoulli model by inserting (5.5) and (5.6) into (2.58). The result is

$$P_{\text{Bern}}(c{=}0\,|\,\boldsymbol{p}) = \frac{\prod_n (1-p_n)}{\prod_n (1-p_n) + \prod_n \left[\beta p_n + (1-\beta)(1-p_n)\right]} \tag{5.7}$$

$$= \frac{1}{1 + \prod_n \left[\beta \frac{p_n}{1-p_n} + (1-\beta)\right]} \tag{5.8}$$

$$= \frac{1}{1 + \prod_n \left(1 - \beta \frac{1-2p_n}{1-p_n}\right)} \tag{5.9}$$

and likewise

$$P_{\text{Bern}}(c{=}1\,|\,\boldsymbol{p}) = \frac{1}{1 + \prod_n \frac{1}{1-\beta\frac{1-2p_n}{1-p_n}}}. \tag{5.10}$$

The above expressions have the disadvantage that they are not additive over instances. Analyzing the situation becomes much easier when considering the log-odds (or logit) of the class probabilities

$$\text{logit}(P_{\text{Bern}}(c\,|\,\boldsymbol{p})) = c \cdot \sum_n \log\left(1 - \beta \cdot \frac{1-2p_n}{1-p_n}\right), \tag{5.11}$$

where the bag class $c$ is supposed to be denoted by $\pm 1$, so that it determines the sign of the right-hand side. The advantage of this expression over (5.9) is that it is additive over instances.

A plot of (5.11) is shown in Figure 5.3. For $\beta = 1$ (standard supervised learning) the curve is symmetric, i. e. negative and positive instances have the same influence on the bag class. The larger $\beta$ the steeper the curve, i. e. the more sensitive the bag class probability depends on the predicted instance class probability. Also, positive bags $p_n > 1/2$ have a larger impact on bag class probability $P(c = 1)$ than negative bags $p_n < 1/2$. A single instance with $p_n = 1$ can cause a bag to be classified as positive, overruling the negative instances in that bag.

### 5.1.3 Combination with MI model

The Bernoulli model does *not* imply the MI constraint (which ensures that each positive bag has at least one positive instance). The probability of a positive bag having only negative instances $Q_{\text{Bern}}(c{=}1, \boldsymbol{y}{=}\mathbf{0}) = (1-\beta)_b^N$ is small but not zero. If we want to ensure that

[2] From a practical point of view it is of course possible to classify unknown bags with any bag model regardless of which model has been used for training. But this is not methodologically sound, so we will refrain from doing this.

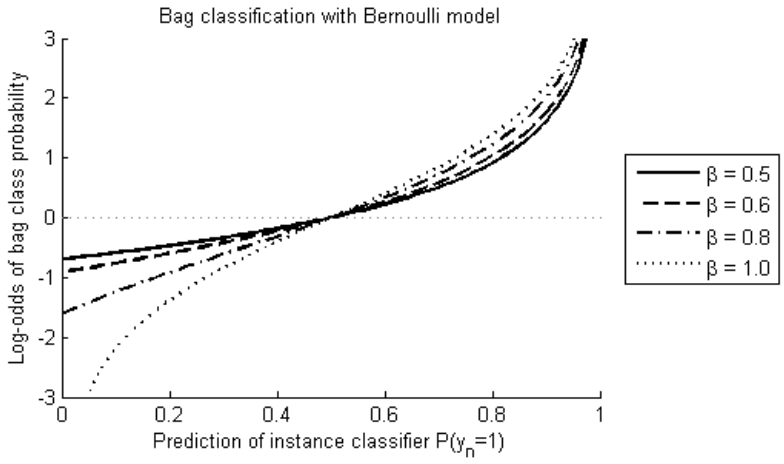


Figure 5.3: Contribution of a single instance to the bag class probability $P_{\text{Bern}}(c \,|\, p_n)$, given the prediction of the instance classifier $p_n = P(y_n{=}1)$.

the MI constraint is met, we need to set the above probability to zero. This is equivalent to multiplying the Bernoulli model with the MI model.

$$Q_{\text{BMI}} \;=\; Q_{\text{Bern}} \cdot Q_{\text{MI}} \tag{5.12}$$

In this section we repeat the above discussion of the Bernoulli model for the combination of Bernoulli with MI model 5.12. To avoid redundancy we do not repeat the complete argumentation but only state the differences between the pure Bernoulli and the BMI model.

The effect of the MI-constraint can be seen most clearly in the example of a positive bag containing two instances. Instead of the right table of (5.2), we obtain for the BMI-model:

| $Q_{\text{BMI}}(c{=}1, \boldsymbol{y})$ | $y_2{=}0$ | $y_2{=}1$ |
|---|---|---|
| $y_1{=}0$ | $0$ | $\frac{\beta(1-\beta)}{1-(1-\beta)^2}$ |
| $y_1{=}1$ | $\frac{\beta(1-\beta)}{1-(1-\beta)^2}$ | $\frac{\beta^2}{1-(1-\beta)^2}$ |

(5.13)

The MI constraint forces the upper left entry of the table to zero. Since we want to keep the generative property of the Bernoulli model $\sum_{\boldsymbol{y}} Q_{\text{BMI}} = 1$ (uniform bag class prior, see Section 4.1), this entails the normalization factor $1 - (1 - \beta)^2$. For a bag with arbitrary number of instances, the MI constraint changes Eq. (5.3) to

$$Q_{\text{BMI}}(c{=}1, \boldsymbol{y}) \;=\; \begin{cases} 0 & \text{for } \boldsymbol{y} = \boldsymbol{0} \\ \frac{\beta^{N_b^+}(1-\beta)^{N_b^-}}{1-(1-\beta)^N} & \text{for } \boldsymbol{y} \neq \boldsymbol{0} \end{cases} \tag{5.14}$$

When combining the BMI model with the instance classifier, we obtain (instead of 5.6):

$$M_{\mathrm{BMI}}(c{=}1) = \frac{1}{1-(1-\beta)^N}\left(\prod_n \left[\beta p_n + (1{-}\beta)(1{-}p_n)\right] - \prod_n \left[(1{-}\beta)(1{-}p_n)\right]\right) \quad (5.15)$$

**Training** The behavior of the BMI model is more complex than the Bernoulli model because it does not factorize into instances. To still get an idea of the training behavior, we assume that the instance likelihoods $p_n = P_{\mathrm{BMI}}(y_n \,|\, \boldsymbol{\theta})$ of all but one instances are fixed and only the remaining instance likelihood $p_m$ can change. It is convenient to define the constants $A$, $B$, and $Z$, and rewrite (5.15) as

$$A = \prod_{n\neq m} \left[\beta p_n + (1-\beta)(1-p_n)\right] \quad (5.16)$$

$$B = \prod_{n\neq m} \left[(1-\beta)(1-p_n)\right] \qquad Z = 1-(1-\beta)^N \quad (5.17)$$

$$M_{\mathrm{BMI}}(c{=}1) = \frac{1}{Z}\left(A\Big[\beta p_m + (1{-}\beta)(1{-}p_m)\Big] - B\Big[(1{-}\beta)(1{-}p_m)\Big]\right) \quad (5.18)$$

The contribution of instance $m$ to the bag likelihood can then be written as

$$M_{\mathrm{BMI}}(c{=}1) \propto \Big[\beta p_m + (1-\beta)(1-p_m)\Big] - \frac{B}{A}\Big[(1-\beta)(1-p_m)\Big] \quad (5.19)$$

$$= \beta p_m + \left(1-\frac{B}{A}\right)(1-\beta)(1-p_m) \quad (5.20)$$

$$= \beta p_m + (1{-}\beta_{\mathrm{eff}})(1{-}p_m) \qquad \beta_{\mathrm{eff}} = 1-\left(1{-}\frac{B}{A}\right)(1{-}\beta) \quad (5.21)$$

Plots of (5.21) and (5.21) are shown in Figure 5.4. The larger $B/A$ the larger effective Bernoulli parameter $\beta_{\mathrm{eff}}$. This means that if the instances ($n \neq m$) are all negative (large $B$), the model strongly prefers the remaining instance $m$ to be positive (large $\beta$) to explain the positive bag label.

**Testing** To obtain the bag class probability of the Bernoulli model with MI constraint, we have to repeat the calculation (5.7) replacing the denominator with (5.15). We obtain

$$P_{\mathrm{Bern}}(c{=}0 \,|\, \boldsymbol{p}) = \frac{\prod_n (1-p_n)}{\prod_n (1-p_n) + \frac{1}{Z}\prod_n \left[\beta p_n + (1{-}\beta)(1{-}p_n)\right] - \frac{1}{Z}\prod_n \left[(1{-}\beta)(1{-}p_n)\right]} \quad (5.22)$$

$$= \frac{1}{1+\frac{1}{Z}\prod_n \left[\beta\frac{p_n}{1-p_n} + (1-\beta)\right] - \frac{1}{Z}\prod_n (1-\beta)} \quad (5.23)$$

$$= \frac{1}{1+\prod_n \left(1-\beta\frac{1-2p_n}{1-p_n}\right) - \frac{1}{Z}(1-\beta)^N} \quad (5.24)$$

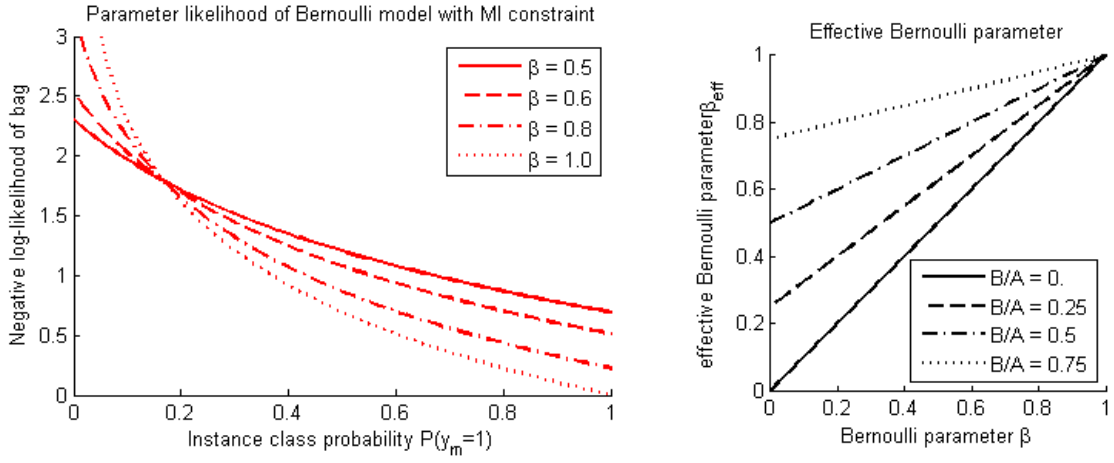


Figure 5.4: Left: Effective Bernoulli parameter of the BMI model. Right: Contribution of instance $m$ to the parameter likelihood $P_{\text{BMI}}(\boldsymbol{\theta} \,|\, \boldsymbol{X}, c)$ for $B/A = 0.8$ (cf. Figure 5.2).

As above we would like to extract the effect of a single instance $m$. To do that, it is convenient to define a constant $C$ and rearrange (5.24) as

$$C = \frac{\prod_{n \neq m} \left(1 - \beta \cdot \frac{1-2p_n}{1-p_n}\right)}{(1-\beta)^{N-1}} \tag{5.25}$$

$$P_{\text{Bern}}(c{=}0 \,|\, \boldsymbol{p}) = \frac{1}{1 + \frac{(1-\beta)^{N-1}}{Z}\left(-(1-\beta) + C\left(1 - \beta\frac{1-2p_m}{1-p_m}\right)\right)} \tag{5.26}$$

Note that $C \geq 1$ because of $0 \leq p_n \leq 1$. For comparison with the Bernoulli model without MI constraints (5.11) we again consider the log-odds

$$\text{logit}(P_{\text{Bern}}(c \,|\, \boldsymbol{p})) = \log\left(\frac{(1-\beta)^{N-1}}{Z}\right) + \log\left(C\left[1 - \beta\frac{1-2p_m}{1-p_m}\right] - 1 + \beta\right) \tag{5.27}$$

$$\text{logit}\left(\frac{P_{\text{Bern}}(c \,|\, p_m)}{P_{\text{Bern}}(c \,|\, p_m{=}1/2)}\right) = \log\left(C\left[1 - \beta\frac{1-2p_m}{1-p_m}\right] - 1 + \beta\right) - \log\left(C - 1 + \beta\right). \tag{5.28}$$

For $C \gg 1$ the summands not containing $C$ are outweighed and (5.28) converges to (5.11). This is the case if the instances $(n \neq m)$ already contain a positive instance, so that the remaining instance $m$ can hardly render the bag negative anymore. If, on the other hand, the instances $(n \neq m)$ are all negative, then $C = 1$, and (5.28) converges to

$$\text{logit}\left(\frac{P_{\text{Bern}}(c \,|\, p_m)}{P_{\text{Bern}}(c \,|\, p_m{=}1/2)}\right) = \log\left(1 - \frac{1-2p_m}{1-p_m}\right) \tag{5.29}$$

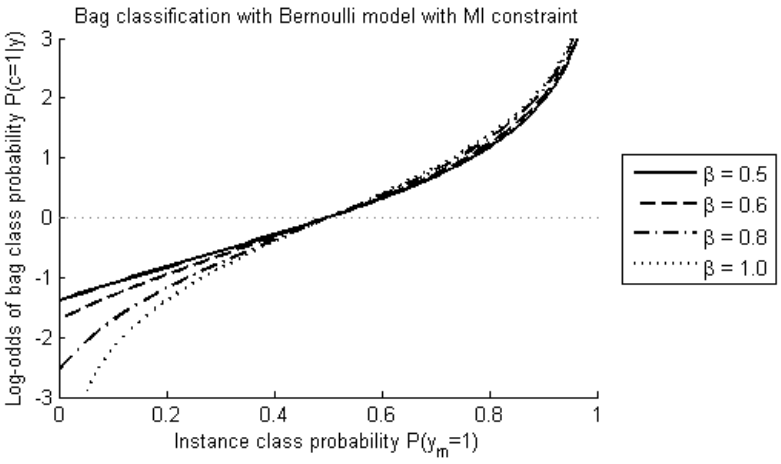


Figure 5.5: Contribution of a single instance to the bag class probability $P_{\text{Bern}}(c \,|\, p_n)$, given the prediction of the instance classifier $p_n = P(y_n = 1)$. This figure is to be compared to Figure 5.3.

which is equivalent to (5.28) with $\beta = 1$. For an intermediate value of $C = 1.5$, equation (5.28) is plotted in Figure 5.5.

### 5.1.4 Notes on the Bernoulli model

**Discriminative Bernoulli model** The Bernoulli model as stated above is a generative bag model, i. e. it provides a probability of instance classes given the bag class $P(\boldsymbol{y} \,|\, c)$. It is also possible to formulate a discriminative Bernoulli model that provides a probability of the bag class for given instance classes $P(c \,|\, \boldsymbol{y})$ as follows

| $q_{\text{Bern,disc}}(y, c, \gamma)$ | $y=0$ | $y=1$ |
|---|---|---|
| $c=0$ | $1-\gamma$ | 0 |
| $c=1$ | $\gamma$ | 1 |

(5.30)

In this case, the Bernoulli parameter $\gamma$ describes the probability of a negative instance to belong to a positive bag.

The difference between the generative and the discriminative Bernoulli model is just the normalization. As we have seen in Section 4.1.2, however, the normalization has a large impact on the bag prior. For the discriminative Bernoulli model the bag prior reads

$$P_{\text{Bern,disc}}(c=0) \;=\; \left(\frac{1-\gamma}{2}\right)^N . \tag{5.31}$$

The bag prior depends both on the Bernoulli parameter $\gamma$ and on the bag size $N$, which makes implementing the discriminative Bernoulli model more complicated than the genera-

tive Bernoulli model. Since there is no clear advantage of the discriminative Bernoulli model over the generative one, we will not pursue the discriminative Bernoulli model further.[3]

**Interpretation of the Bernoulli parameter** It is suggestive to interpret the Bernoulli parameter $\beta$ as the (observed or expected) ratio of positive instances in positive bags. This would indeed be the case if the instance classes were determined by the Bernoulli model alone, since $\beta$ is defined as $\beta = q_{\text{Bern}}(y{=}1 \,|\, c{=}1)$. However, the instance classes are in fact determined by both the Bernoulli model *and* the instance classifier:

$$P_{\text{Bern}}(y_n{=}1 \,|\, c{=}1) = \frac{1}{Z} \cdot q_{\text{Cl}}(y_n \,|\, \boldsymbol{x}_n, \boldsymbol{\theta}) \cdot q_{\text{Bern}}(y_n \,|\, c) \tag{5.32}$$

$$= \frac{p_n \beta}{p_n \beta + (1 - p_n)(1 - \beta)}. \tag{5.33}$$

Moreover, the instance classifier often has a "strong opinion" about the instance class, meaning that $p_n$ is close to zero or close to one, while $\beta$ is usually in a "moderate" range. In other words, the instance classifier dominates the instance class prediction, while the Bernoulli model just "pushes the decision boundary a little", so that more instances get classified as positive; the larger $\beta$ the stronger the push.

As described above, the MI model is equivalent to $\beta = 1/2$, but of course this does not imply that exactly half of the instances from positive bags must be positive, while the other half must be negative.

To pinpoint the issue, let us consider a bag that contains less positive than negative instances. According to the above mentioned (faulty) interpretation, the Bernoulli parameter should be $\beta < 1/2$. But then, the Bernoulli model would penalize positive instances in positive bags, which is obviously wrong.

**Estimation of the Bernoulli parameter** It is not clear a-priori which value of the Bernoulli parameter $\beta$ yields the best result. Therefore, it is suggestive to optimize $\beta$ during training to automatically find the best value.

However, as we will discuss in this paragraph, this is not possible.

The reason can be seen by inspecting the parameter likelihood in Figure 5.2. Let us first assume that the result of the instance classifier $p_n$ is fixed. For any instance with $p_n < 1/2$ the optimum is $\beta = 0$, for any instance with $p_n > 1/2$, the optimum is $\beta = 1$. So $\beta$ always diverges to the extreme values.

If we have a bag with more than one instance, the situation is a bit more complex. There are well-behaved situations, e. g.

$$\frac{\mathrm{d}}{\mathrm{d}\beta} \log\left[\beta p + (1 - \beta)(1 - p)\right] = \frac{1}{\beta + \frac{1-p}{2p-1}} \tag{5.34}$$

[3] In effect, correcting for the unwanted bag prior of the discriminative Bernoulli model would lead back to the generative Bernoulli model.

At the optimum, the sum of derivatives must be zero. For two instance we obtain

$$\beta_{\text{opt}} + \frac{1-p_1}{2p_1-1} = -\beta_{\text{opt}} - \frac{1-p_2}{2p_2-1} \tag{5.35}$$

$$\beta_{\text{opt}} = \frac{1}{2}\left(\frac{p_1-1}{2p_1-1} + \frac{p_2-1}{2p_2-1}\right) \tag{5.36}$$

Equation (5.36) is plotted in Figure 5.6. For more than two instances, we expect a similar behavior: For many values of $p_n$, $\beta_{\text{opt}}$ is degenerate in many cases (if all $p_n > 1/2$, then $\beta_{\text{opt}} = 1$, if all $p_n < 1/2$, then $\beta_{\text{opt}} = 0$). For the small band of non-degenerateness, $beta_{\text{opt}}$ is very sensitive to small changes of $p_n$.

The second problem with the estimation of $\beta$ is the interaction with the instance classifier. Even if $\beta_{\text{opt}}$ has a reasonable value (between 0.5 and 0.9) for fixed $p_n$, we must also take into account that for $\beta > 0.5$, the likelihood increases with increasing $p_n$. Again, we analyze the situation for a bag containing a single instance: There is only one stationary point (at $\beta = p = 1/2$), and this is not a stable minimum, but a saddle point. The two degenerate

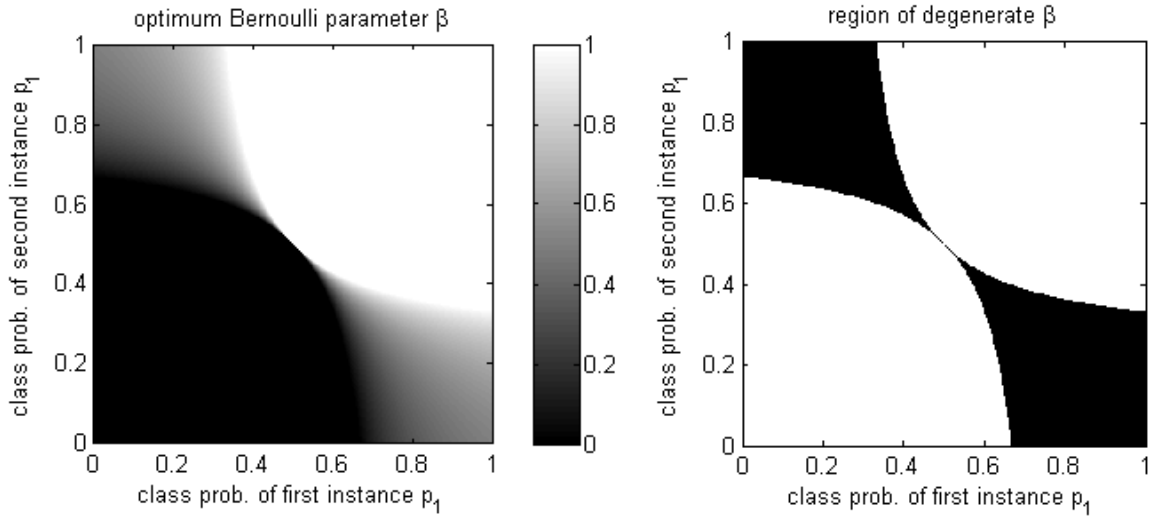


Figure 5.6: Left: Value of Bernoulli parameter $\beta_{\text{opt}}$ that optimizes the likelihood for a bag with two instances with fixed instance class probabilities $p_1$, $p_2$. Right: In the white region, $\beta_{\text{opt}}$ is degenerate $\beta_{\text{opt}} = 0$ or $\beta_{\text{opt}} = 1$.

minima are located at $\beta = p = 1$ and at $\beta = p = 0$.

$$\frac{\mathrm{d}^2 NLL}{\mathrm{d}\beta^2} = \frac{-1}{\left(\beta + \frac{1-p}{2p-1}\right)^2} \tag{5.37}$$

$$\frac{\mathrm{d}^2 NLL}{\mathrm{d}\beta\ \mathrm{d}p} = \frac{-1}{(\beta(2p-1)+1-p)^2} \tag{5.38}$$

$$H(\beta{=}p{=}1/2) = \begin{pmatrix} 0 & -4 \\ -4 & 0 \end{pmatrix} \tag{5.39}$$

To sum up: The Bernoulli parameter does not describe the ratio of positive instances in positive bags, but it is a rather heuristic parameter that induces the algorithm to find more positive instance. How many more positive instances will be found depends on the details of the dataset and the instance classifier. $\beta$ does not have a clear meaning.

## 5.2 Power model

The original idea of the Bernoulli model was that the user can define a preferred ratio of positive instances in positive bags. As we have seen above, however, the Bernoulli parameter $\beta$ does not fulfill this role. In this section we propose a bag model that is better suited to control the ratio of positive instances.

### 5.2.1 Model definition

To examine the properties of a bag model, it is expedient to plot its factor for the complete bag as a function of the ratio of positive instances $r$ in a positive bag. For the Bernoulli model we obtain

$$Q(\boldsymbol{y}, c) = \prod_n P(y_n \,|\, c) = \beta^{N^+}(1-\beta)^{N^-} = \left(\beta^r(1-\beta)^{1-r}\right)^N \tag{5.40}$$

$$r = \frac{N^+}{N} = \frac{1}{N}\sum_n y_n. \tag{5.41}$$

This function is plotted for $N = 1$ in Figure 5.7. For $r = 1$ the function degenerates. Also, for large $N$ the function will degenerate, too. As noted above, the Bernoulli model does not encompass the MI constraint $Q(\boldsymbol{y}{=}\mathbf{0}, c) > 0$, unless it is degenerate.

Figure 5.7 suggests a different approach to set up a bag model, namely to model $Q(r)$ directly, without dependence on the bag size $N$. We should demand that $Q(r{=}0) = 0$ (MI constraint). We propose the following power model

$$Q_{\mathrm{Pow}}(r, N, \gamma) = r^{(\gamma \cdot N)}, \tag{5.42}$$

which is plotted in Figure 5.8.

Besides plotting the bag model factors $Q(r^+)$ it is also useful to plot their logarithm, because the logarithm is additive over instances. It is apparent from Figure 5.9 that in the Bernoulli model each instance contributes the same to the bag model factor, independent of the other instances, while in the power model there is an effect of "diminishing" returns, meaning that a estimating an instance as positive reduces the negative log-likelihood very much if there are only few positive instances in the bag, but it does not change the log-likelihood much, if there are already many positive instances explaining the positive bag.

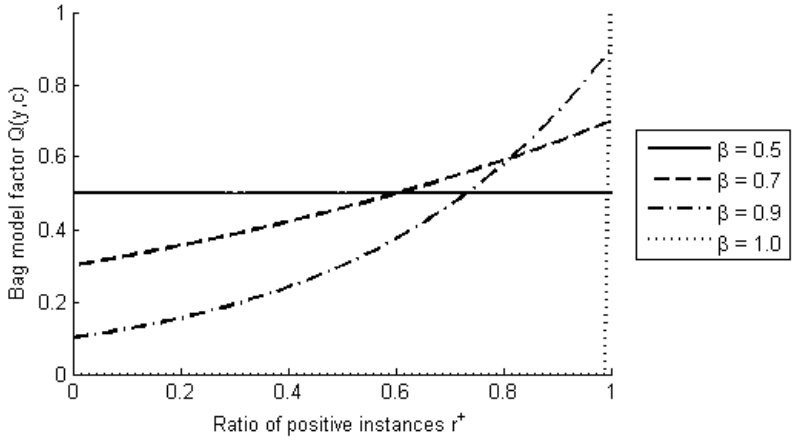


Figure 5.7: Bag model factor as a function of the ratio of positive instances for the Bernoulli model ($N = 1$).

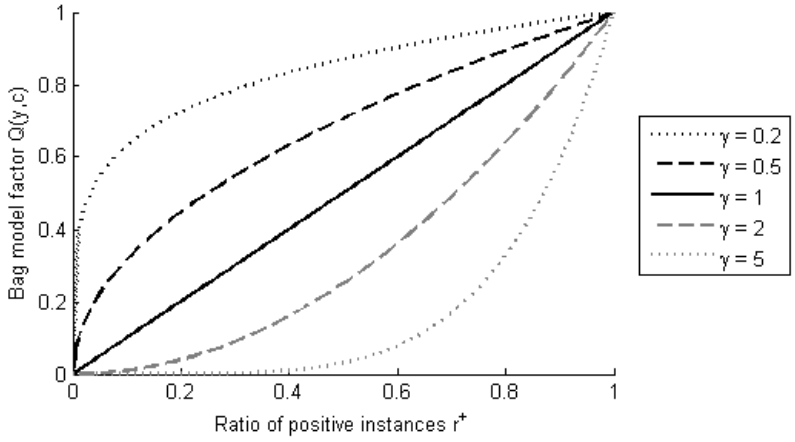


Figure 5.8: Bag model factor as a function of the ratio of positive instances for the power model ($N = 1$).

### 5.2.2 Implementation

To implement the power model in our sampling-based self-training framework (see the following Chapter 6), we need to draw samples from the instance class distribution of each positive bag. This is somewhat more difficult for the power model than for the Bernoulli model, because the power model does not factorize over instances. Our approach is to use rejection sampling with a factorizing proposal distribution that is optimized for each bag to give a good sampling efficiency.

The instance class distribution from which we need to draw samples is $P(\boldsymbol{y} \,|\, c{=}1, \boldsymbol{X}, \boldsymbol{\theta}, \gamma)$ (cf. Eq. 6.21). The bag class $c$ will be left out in the following for brevity, and the information of the data $\boldsymbol{X}$ and the classification parameters $\boldsymbol{\theta}$ can be summarized by the classifier's predictions $P$. So we have

$$P(\boldsymbol{y} \,|\, \boldsymbol{p}, \gamma) \;\; \propto \;\; Q_{\mathrm{Pow}} \cdot Q_{\mathrm{Cl}} \;=\; r^{(\gamma \cdot N)} \cdot \prod_n q_{\mathrm{Cl}}(y_n) \tag{5.43}$$

For rejection sampling we use the following factorizing proposal distribution.

$$P_{\mathrm{prop}}(\boldsymbol{y} \,|\, \boldsymbol{p}, \beta) \;\; \propto \;\; Q_{\mathrm{prop}} \cdot Q_{\mathrm{Cl}} \;=\; \prod_n q_{\mathrm{prop}}(y_n, \beta) \cdot q_{\mathrm{Cl}}(y_n, p_n) \tag{5.44}$$

$$q_{\mathrm{prop}}(y_n, \beta) \;\; = \;\; \begin{cases} \beta & \text{for } y_n = 1 \\ 1-\beta & \text{for } y_n = 0 \end{cases} \tag{5.45}$$

The parameter $\beta$ can be chosen freely, and in the following we will describe how to optimize $\beta$ for maximum sampling efficiency.

Sampling efficiency is determined by the acceptance rate $R$ which is the ratio between target and proposal distribution. Since the contribution of the instance classifier $Q_{\mathrm{Cl}}$ cancels,

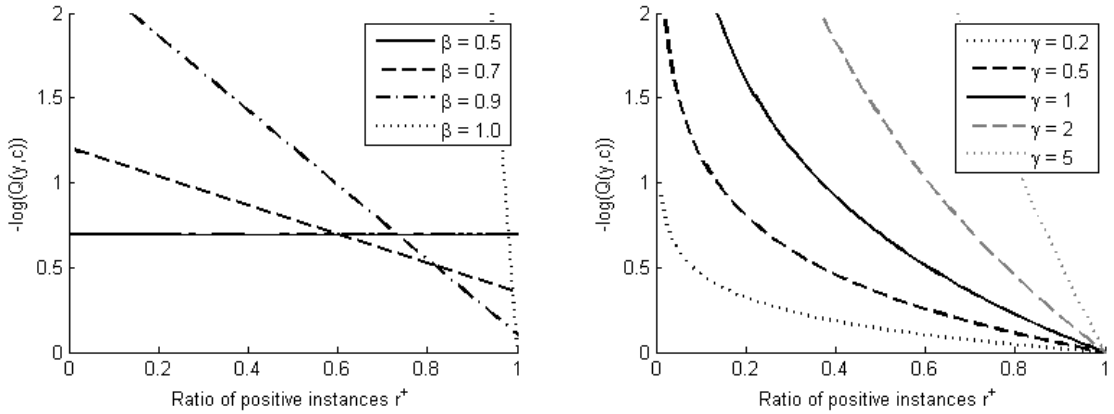


Figure 5.9: Negative log-likelihood of bag model factor $Q(r^+)$ for the Bernoulli and the power model.

we have

$$R(r,\beta) = \frac{Q_{\text{Pow}}(r,N,\gamma)}{Q_{\text{prop}}(r,N,\beta)} \frac{Q_{\text{prop}}(r_{\max},N,\beta)}{Q_{\text{Pow}}(r_{\max},N,\gamma)} \tag{5.46}$$

$$r_{\max}(\beta) = \arg\max_r \frac{Q_{\text{Pow}}(r,N,\gamma)}{Q_{\text{prop}}(r,N,\beta)}. \tag{5.47}$$

The term containing $r_{\max}$ ensures that the acceptance rate does not exceed 1: $R(r) \leq 1$ with equality at $r = r_{\max}$. Note that both factors $Q_{\text{Pow}}$ and $Q_{\text{prop}}$ depend only on the instance counts $(r, N)$, and not on the exact combination of instance classes $\boldsymbol{y}$. This makes handling the acceptance rates significantly easier.

The acceptance rate $R$ is maximum at $r = r_{\max}$ and decreases with increasing deviation $|r = r_{\max}|$. So to achieve good sampling efficiency, $r_{\max}$ should be a typical instance class ratio. We choose $r_{\max}$ as the expected value of $r$ of the target distribution (5.43). This expectation can be approximated by

$$r_{\max} = E_r\left[\frac{1}{Z} \cdot Q_{\text{Pow}}(r,N,\gamma) \cdot Q_{\text{Cl}}\right] \tag{5.48}$$

$$\approx \frac{\sum_r r \cdot \binom{N}{Nr} r^{(\gamma N)} \left(\bar{p}^r \cdot (1-\bar{p})^{1-r}\right)^N}{\sum_r \binom{N}{Nr} r^{(\gamma N)} \left(\bar{p}^r \cdot (1-\bar{p})^{1-r}\right)^N} \tag{5.49}$$

The binomial coefficient accounts for the different number of instance class combinations that yield the same instance class ratio $r$. The terms containing $\bar{p}$ approximate the Poisson binomial distribution with a binomial distribution with mean instance class probability $\bar{p} = 1/N \sum_n p_n$.

Actually, we cannot select the chosen $r_{\max}$ directly, because it is determined by (5.47). Instead, we have to choose $\beta$ so that it yields the desired $r_{\max}$. To find the corresponding value of $\beta$, we have to solve (5.47) for $\beta$. We achieve this by setting equal the derivatives of the logarithms of numerator and denominator

$$\frac{\mathrm{d}}{\mathrm{d}r} \log Q_{\text{Pow}}(r,N,\gamma) = \frac{\mathrm{d}}{\mathrm{d}r} N\gamma \log r = \frac{N\gamma}{r} \tag{5.50}$$

$$\frac{\mathrm{d}}{\mathrm{d}r} \log Q_{\text{prop}}(r,N,\beta) = \frac{\mathrm{d}}{\mathrm{d}r} \left(Nr \log\beta + N(1-r)\log(1-\beta)\right) = N \cdot \log\left(\frac{\beta}{1-\beta}\right) \tag{5.51}$$

$$\log\left(\frac{\beta}{1-\beta}\right) = \frac{\gamma}{r_{\max}} \tag{5.52}$$

$$\beta = \frac{1}{1+e^{-\gamma/r_{\max}}} \tag{5.53}$$

# Chapter 6

# Self-Training Multiple Instance Random Forest (SMIRF)

In this chapter we present a multiple instance (MI) classifier that is based on the random forest (RF). The latent variables (unknown instance classes) are estimated via the semi-supervised method of self-training. We propose and explore three variants of this algorithm: one for the pure multiple instance model, one for the Bernoulli model proposed in Section 5.1, and one for the power model proposed in Section 5.2. Also we test the different bag classification methods discussed in Chapter 4.

## 6.1 Random forests

The random forest is an ensemble classifier that is based on decision trees. In this section we give a short description of decision trees and the random forest for later reference.

**Decision trees for classification** A decision tree is a discriminative classifier that defines the class $y(\boldsymbol{x})$ by a sequence of binary decisions rules $\eta(\boldsymbol{x})$. The result of each binary decision either specifies the next decision rule to apply or it assigns the class $y(\boldsymbol{x})$. This gives rise to a binary tree structure, as illustrated in Figure 6.1 Each internal node represents a decision rule, and each leaf node represents a class assignment, either $y = 0$ or $y = 1$. For a given decision tree, the input $\boldsymbol{x}$ specifies the path through the tree while the leaf node where the path ends specifies the corresponding class assignment $y(\boldsymbol{x})$.

The binary decision rules (also called splits) should be simple and fast to evaluate. The most common choice is to consider only univariate step functions

$$\eta(\boldsymbol{x}; d, s) \quad = \quad \begin{cases} 1 & \text{if } x_d \leq s \\ 0 & \text{if } x_d > s. \end{cases} \tag{6.1}$$

In this case each split is defined by the split dimension $d$ and the split location $s$. Another possibility is to consider all linear step functions

$$\eta(\boldsymbol{x}; \boldsymbol{v}, s) \quad = \quad \begin{cases} 1 & \text{if } \boldsymbol{v}^T\boldsymbol{x} \leq s \\ 0 & \text{if } \boldsymbol{v}^T\boldsymbol{x} > s, \end{cases} \tag{6.2}$$

which is defined by the split direction $\boldsymbol{v}$ (normalized vector) and split location $s$. Random forests based on (6.2) are called *oblique* random forest (Menze, Kelm, Splitthoff, Koethe & Hamprecht 2011). We will use the standard random forest with splits (6.1). In effect, a decision tree subdivides the feature space into (cuboid) regions and assigns a class to each region.

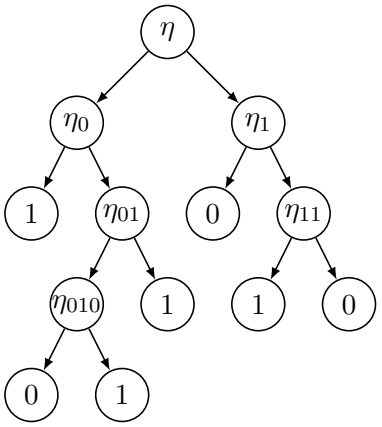


Figure 6.1: Decision tree for classification. At internal nodes, a simple binary criterion $\eta$ is evaluated, to determining the route. Terminal nodes (leaves) assign a class (0 or 1).

**Training a decision tree** A decision tree is trained (or grown) by recursively choosing the split that gives optimum class discrimination on the training data. A single split usually cannot perfectly separate the two classes, so that the resulting two sets of datapoints are still *impure*. The splitting procedure is then repeated on both sets of datapoints until the resulting sets are pure.

Formally, a split is chosen by maximizing the sum of the resulting class purities

$$(s, d) \quad = \quad \arg\max_{s,d} \; R(\boldsymbol{y}_{\boldsymbol{n}_0}) + R(\boldsymbol{y}_{\boldsymbol{n}_1}) \tag{6.3}$$

$$\boldsymbol{n}_0 \quad = \quad \{n : \eta(\boldsymbol{x}_n) = 0\} \tag{6.4}$$

$$\boldsymbol{n}_1 \quad = \quad \{n : \eta(\boldsymbol{x}_n) = 1\}, \tag{6.5}$$

where $\boldsymbol{n}_0$ and $\boldsymbol{n}_1$ denote the sets of datapoints of the two child nodes. The most common

measures of class purity are the Gini impurity and the entropy

$$R_{\text{Gini}}(\boldsymbol{y}) = r_0 \cdot r_1 \tag{6.6}$$
$$R_{\text{Entropy}}(\boldsymbol{y}) = -r_0 \ln r_0 - r_1 \ln r_1 \tag{6.7}$$
$$r_0 = \frac{1}{N} \sum_n 1 - y_n \tag{6.8}$$
$$r_1 = \frac{1}{N} \sum_n y_n, \tag{6.9}$$

where $r_0$ ($r_1$) is the ratio of negative (positive) data points in the given set $\boldsymbol{y}$.

This procedure of tree growing is a "greedy" method, since each single split tries to achieve the best purity. This does not guarantee that the resulting tree has the minimum number of nodes or the minimum depth, but it is a good practical method to find good decision trees very fast.

Altogether, a trained decision tree $t$ is specified by all split dimensions $\boldsymbol{d}$, all split locations $\boldsymbol{s}$, and the tree structure $T_S$ that defines the order in which the decision rules are to be applied. Thus we can make the connection to the generic symbol $\boldsymbol{\theta}$ for the classifier's parameters, which has been used in the previous chapters, by writing

$$\boldsymbol{\theta} = \{\boldsymbol{d}, \boldsymbol{s}, T_S\}. \tag{6.10}$$

**Random forest** The random forest is a bagged version of the decision trees described above. A general description of bagging in the context of ensemble classifiers and Bayesian inference has been given in Section 2.2. A crucial feature of bagging is the randomization that ensures the required instability (or variability) of the ensemble members. The random forest combines two methods of randomization: (i) subsampling of datapoints and (ii) subsampling of features (or dimensions).

Subsampling of datapoints means that for each tree only a subset of all datapoints $\mathcal{D}_s$ is used for training. The subset is sampled with replacement (bootstrap sample). The typical sample size is equal to the total number of datapoints $N$. Note that bootstrap sampling will lead to some datapoints being present twice or more in the subsampled dataset while other datapoints are left out. On average, each tree will use a fraction of $1 - 1/e \approx 63\%$ of the available datapoints.

Subsampling of features means that for each split, only a subsample of features is taken into account, so (6.3) is replaced by

$$(s, d) = \arg\max_{s,d} \left[R(\boldsymbol{y}_{\boldsymbol{n}_0}) + R(\boldsymbol{y}_{\boldsymbol{n}_1})\right] \cdot I(d) \tag{6.11}$$

where $I(d)$ is the indicator function of the available features. The available features are drawn randomly without replacement. The typical sample size is the square root of the total number of features $\sqrt{D}$.

Finally, the predicted class probability of a random forest is the arithmetic mean of the

class predictions of all trees

$$P(y{=}y_t \,|\, \boldsymbol{x}, \boldsymbol{\theta}) \;\; = \;\; \frac{1}{T} \sum_t y_t(\boldsymbol{x}, \boldsymbol{\theta}). \tag{6.12}$$

The random forest has been proposed by (Breiman 2001). It has become one of the most-used classifiers. Advantages are that it is easy to understand and implement, fast, and robust. The most important disadvantage is probably that it is not based on a probabilistic model. The rules for randomization (subsampling) are rather heuristic. However, random forests have consistently shown good classification performance in practice (Hastie et al. 2009).

## 6.2 Self-training random forest for standard multiple instance learning

To learn the multiple instance model one has to estimate the unknown instance classes $\boldsymbol{y}$ explicitly. This situation is equivalent to constrained semi-supervised learning, as described in Section 3.1.3. Negative bags correspond to labeled instance and positive bags correspond to unlabeled instances. The constraints are given by the requirement of the multiple instance model that each positive bag must contain at least one positive instance.

In this section we propose a self-training approach based on the random forest for multiple instance learning. We first describe the baseline algorithm for the standard MI model, then we describe how the Bernoulli model and the power model proposed in Chapter 5 can be incorporated.

### 6.2.1 Self-training the multiple instance model

A special difficulty of the MI model is that its bag likelihood does not factorize over instances, but instead is a sum (or integral) over all combinations of instance classes

$$P(c \,|\, \boldsymbol{X}, \boldsymbol{\theta}) \;\; = \;\; \int P_{\mathrm{MI}}(c \,|\, \boldsymbol{y}) \prod_n P(y_n \,|\, \boldsymbol{x}_n, \boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{y} \tag{6.13}$$

This integral is infeasible because there are $2^{N_b} - 1$ combinations of instance classes $\boldsymbol{y}$ for each bag $b$.

The difficulty can be overcome with a self-training approach. We start by reprinting the basic equations of self-training from Section 2.4.1:

$$P_i(\boldsymbol{\theta} \,|\, \boldsymbol{X}) \;\; \hat{=} \;\; \frac{1}{Z} \int P_M(\boldsymbol{y} \,|\, \boldsymbol{X}, \boldsymbol{\theta}) P_{i-1}(\boldsymbol{y} \,|\, \boldsymbol{X}) \, \mathrm{d}\boldsymbol{y} \tag{6.14}$$

$$P_i(\boldsymbol{y} \,|\, \boldsymbol{X}) \;\; \hat{=} \;\; \int P_M(\boldsymbol{y} \,|\, \boldsymbol{X}, \boldsymbol{\theta}) P_i(\boldsymbol{\theta} \,|\, \boldsymbol{X}) \, \mathrm{d}\boldsymbol{\theta}, \tag{6.15}$$

Both steps require the evaluation of the instance class probabilities $P_M(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta})$. For reference, we first specify this term for semi-supervised learning:

$$P_{\mathrm{SSL}}(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}) \quad = \quad \prod_n P(y_n \mid \boldsymbol{x}_n, \boldsymbol{\theta}) \tag{6.16}$$

For the multiple instance model, we additionally have to take into account the "multiple instance constraints" which are specified by the bag labels $c$. The required probability $P_M(\boldsymbol{y} \mid c, \boldsymbol{X}, \boldsymbol{\theta})$ can be evaluated via

$$P_{\mathrm{MI}}(\boldsymbol{y} \mid c, \boldsymbol{X}, \boldsymbol{\theta}) \quad = \quad \frac{P_{\mathrm{MI}}(c \mid \boldsymbol{y}) \prod_n P(y_n \mid \boldsymbol{x}_n, \boldsymbol{\theta})}{\int P_{\mathrm{MI}}(c \mid \boldsymbol{y}) \prod_n P(y_n \mid \boldsymbol{x}_n, \boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{y}}, \tag{6.17}$$

which yields

| $P_{\mathrm{MI}}(\boldsymbol{y} \mid c, \boldsymbol{X}, \boldsymbol{\theta})$ | $\boldsymbol{y}=\boldsymbol{0}$ | $\boldsymbol{y}\neq\boldsymbol{0}$ |
|---|---|---|
| $c=0$ | 1 | 0 |
| $c=1$ | 0 | $\frac{\prod_n P(y_n \mid \boldsymbol{x}_n, \boldsymbol{\theta})}{1-\prod_n P(y_n=0 \mid \boldsymbol{x}_n, \boldsymbol{\theta})}$, |

(6.18)

The denominator of the lower right entry is a normalization factor. If it is zero, i. e. $P(y_n = 0 \mid \boldsymbol{x}_n, \boldsymbol{\theta}) = 1 \; \forall n$, (6.18) cannot be normalized and is not well-defined.

**Factorizing the parameter update** Just like the bag likelihood (6.13), the distribution of latent instance classes (6.18) does not factorize over instances. However, the two-step procedure of self-training allows for the following simplification:

We approximate the parameter update (6.14) by using the factorizing equation (6.16) instead of the complete MI model (6.18). The MI constraints are enforced only by the update of instance classes (6.15), where we have to use the correct distribution (6.18).

Following the same line of argument as in Section 2.4.1, we find that this procedure leads to the following effective parameter distribution at a fixpoint (or local optimum) $\hat{\boldsymbol{\theta}}$:

$$P(\hat{\boldsymbol{\theta}} \mid c, \boldsymbol{X}) \quad = \quad \frac{1}{Z} \int P_{\mathrm{MI}}(\boldsymbol{y} \mid c, \boldsymbol{X}, \hat{\boldsymbol{\theta}}) P_{\mathrm{SSL}}(\boldsymbol{y} \mid \boldsymbol{X}, \hat{\boldsymbol{\theta}}) \, \mathrm{d}\boldsymbol{y}, \tag{6.19}$$

This ensures that despite the approximation of the parameter update the MI constraints will be satisfied when the algorithm has converged, since the MI factor $P_{\mathrm{MI}}(\boldsymbol{y} \mid c, \boldsymbol{X}, \boldsymbol{\theta})$ is zero if the MI constraints are not satisfies, which dominates over the approximated factor $P_{\mathrm{SSL}}(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta})$.

### 6.2.2 Sampling approach and out-of-bag estimate

In Section 2.4.1 we have formulated self-training in the most general probabilistic form. However, the random forest cannot process probabilistic class estimates $P(\boldsymbol{y})$, but requires concrete samples $\hat{\boldsymbol{y}}$ to grow a tree. Since the random forest can also be regarded as a tree

sampling method as described (cf. Sections 2.2.3 and 6.1), this leads to a full sampling approach of self-training.

**Sampling approach to self-training** Rewriting equations (6.14) and (6.15) in terms of concrete samples and factorizing the parameter update (as described in the previous section) leads to

$$\boldsymbol{\theta}_{ti} \sim \frac{1}{S} \sum_{s} \prod_{n} P(y_{ns,i-1} \,|\, \boldsymbol{x}_n, \boldsymbol{\theta}) \tag{6.20}$$

$$\boldsymbol{y}_{si} \sim \frac{1}{T} \sum_{t} P_{\mathrm{MI}}(\boldsymbol{y} \,|\, c, \boldsymbol{X}, \boldsymbol{\theta}_{ti}) \tag{6.21}$$

In each step $i$ we draw $T$ parameter samples (i. e. grow $T$ trees), and draw $S$ samples of the latent variables $\boldsymbol{y}$.

This procedure can be regarded as a kind of blocked Gibbs sampling. One block of variables contains the instance classes $\boldsymbol{y}$, the other block contains the parameters $\boldsymbol{\theta}$.

The sampling approach introduces variability into the otherwise deterministic self-training procedure, which makes the algorithm robust to local minima. We do not need additional methods like deterministic annealing to avoid getting trapped, as is required for multiple instance SVMs (Gehler & Chapelle 2007).

**Out-of-bag estimate** To draw samples $\boldsymbol{y}_{si}$ from (6.21), we need class predictions for the training data $P(y_n \,|\, \boldsymbol{x}_n, \boldsymbol{\theta}_i)$. However, the standard formula for random forest class prediction (6.12) is valid only for test data that have not been seen during training. (For the training data, the trees just return the training labels they had been given.)

Fortunately, the random forest as a bagging method makes possible the so-called "out-of-bag" (oob) estimation of $P(y_n \,|\, \boldsymbol{x}_n, \boldsymbol{\theta}_i)$ on the training data. For the out-of-bag estimate at datapoint $\boldsymbol{x}_n$, only those trees are allowed to vote that have *not* seen $\boldsymbol{x}_n$, i. e. where the point $\boldsymbol{x}_n$ has not been part of the bootstrap sample.

$$P_{\mathrm{oob}}(y_n \,|\, \boldsymbol{x}_n, \boldsymbol{\theta}) = \begin{cases} \frac{1}{\#U} \sum_{t \in U} y_{nt} & \text{for } y_n = 0 \\ 1 - \frac{1}{\#U} \sum_{t \in U} y_{nt} & \text{for } y_n = 1 \end{cases} = \left| y_n - \frac{1}{U} \sum_{t \in U} y_{nt} \right|, \tag{6.22}$$

where $y_{nt} = y(\boldsymbol{x}_n, \boldsymbol{\theta}_t)$. The out-of-bag estimate is almost identical to the estimate obtained from $N$-fold cross-validation (Hastie et al. 2009). Therefore, we can use the oob-estimate (6.22) as an estimate of $P_{\mathrm{MI}}(\boldsymbol{y} \,|\, c, \boldsymbol{X}, \boldsymbol{\theta}_{ti})$ in (6.21).

We performed bagging on the bag-level, meaning that a bag is either completely contained in the bootstrap sample or completely left out. This is beneficial, because it allows for online accuracy prediction from the oob-estimates. This significantly facilitates performance assessment because a single run of the algorithm on the complete dataset suffices for performance assessment. Cross-validation is not needed for a quick test or for parameter tuning.

**Pseudo-code of SMIRF (self-training multiple instance random forest)**

initialize instance classes as equal to their bag class $P_0(y_{bn}) = \delta(y_{bn} - c_b)$
**for each iteration** $i$ (usually 5 iterations until convergence)
  **for each tree** $t$
  - draw instance classes $\hat{\boldsymbol{y}}_{ti}$ from $P_{i-1}(\boldsymbol{y})$, ensuring that MI-constraint is met for each bag: $\hat{\boldsymbol{y}}_{b,ti} \neq \mathbf{0} \quad \forall b$
  - draw bootstrap sample on bag level $\boldsymbol{b}_{ti} \subset \{1, \ldots, B\}$ and create corresponding set of instances $\boldsymbol{n}_{ti}$
  - grow tree $\hat{\boldsymbol{\theta}}_{ti} = \text{growTree}(\boldsymbol{X}_{\boldsymbol{n}_t,i}, \hat{\boldsymbol{y}}_{\boldsymbol{n}_t,ti})$

  calculate oob-estimate $P_i(\boldsymbol{y}) = \text{oobRF}(\boldsymbol{X}, \hat{\boldsymbol{\theta}}_{ti})$

### 6.2.3 Algorithm details and behavior

In this section we describe some details of the algorithm and test its behavior on a synthetic dataset.

First of all, we provide the pseudocode of the algorithm in the text box above. Note that the initialization corresponds to the supervised setting. Therefore, the algorithm automatically provides a supervised random forest for comparison to SCL-learning (see Section 3.2.3).

**Drawing instance classes** We draw samples of $\boldsymbol{y}$ according to (6.21) by rejection sampling. We first draw samples from the factorizing distribution (6.16), then we reject those samples that do not satisfy the MI constraint (6.18).

For each $n$ we draw an instance-wise sample $y_n$ from the Bernoulli distribution whose parameter is equal to the ratio of positive tree votes for the data point $\boldsymbol{x}_n$.

$$y_{n,si} \;\sim\; \frac{1}{T}\sum_t P(y_n \,|\, \boldsymbol{x}_n, \boldsymbol{\theta}_{ti}) \;=\; \text{Bern}\left(\frac{\sum_t \hat{y}_{nti}}{T}\right) \tag{6.23}$$

Afterwards we check if $\boldsymbol{y} \neq \mathbf{0}$. If $\boldsymbol{y} = \mathbf{0}$, we reject the complete sample $\boldsymbol{y}$ and redraw samples from (6.23). This is repeated until the MI-constraint is satisfied.

This procedure is simple and efficient unless the given instance probabilities are strongly negative for all instances in the bag: $P(y_n = 1 \,|\, \boldsymbol{x}_n, \boldsymbol{\theta}_{t,i-1}) \ll 1 \quad \forall n$.

**Behavior of baseline algorithm** We tested the algorithm described above on a synthetic multiple instance dataset, where we know the ground truth,i. e. which instances of the positive bags are truly positive. The data was generated by a mixture of 13 Gaussians in 20 dimensions. The centers of the Gaussians were drawn from a uniform distribution on the unit hypercube, the covariances were diagonal with entries drawn uniformly from

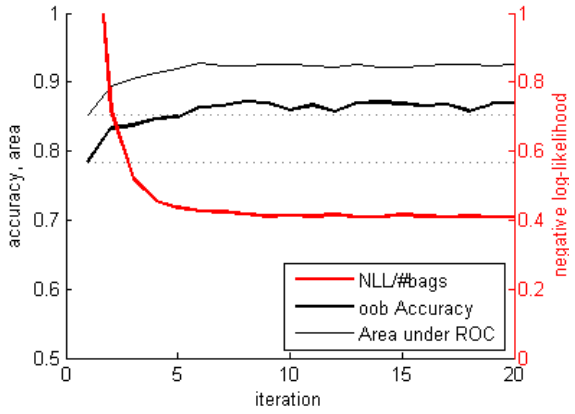


Figure 6.2: Behavior of baseline algorithm on synthetic dataset. "NLL/#bags" stands for the negative log-likelihood per bag. Settings: Number of trees: $T = 100$, bag size $n = 5$, 2 positive instances per positive bag.

[0,0.5]. From the five negative Gaussians we draw 4000 datapoints, and from the 8 positive Gaussians we draw 1000 datapoints. These 5000 datapoints were grouped in 500 negative bags with an average of 5 negative instances, and 500 positive bags with an average of 2 positive and 3 negative instances.

The progress of the SMIRF algorithm on the synthetic data is plotted in Figure 6.2. The

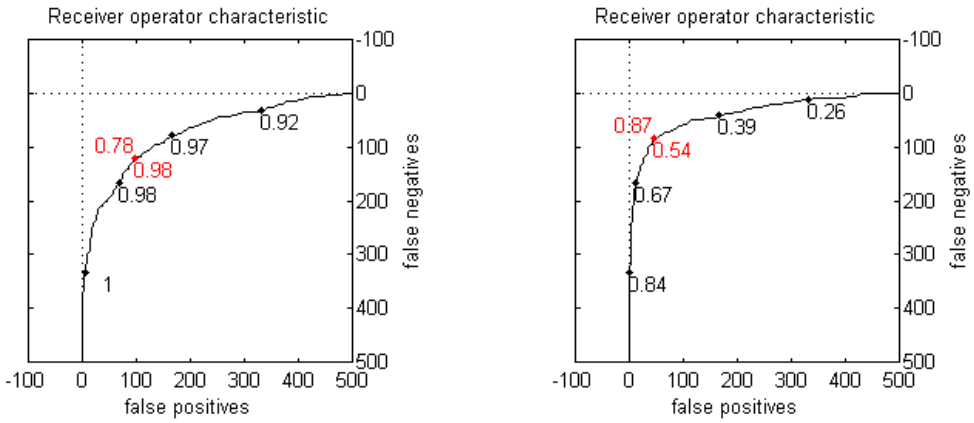


Figure 6.3: Receiver-operator curves of first and last iteration of baseline algorithm applied on synthetic dataset. Left: first iteration. Right: last (20th) iteration. The number above the curve is the optimum classification accuracy, the numbers below the curve are the corresponding decision thresholds.

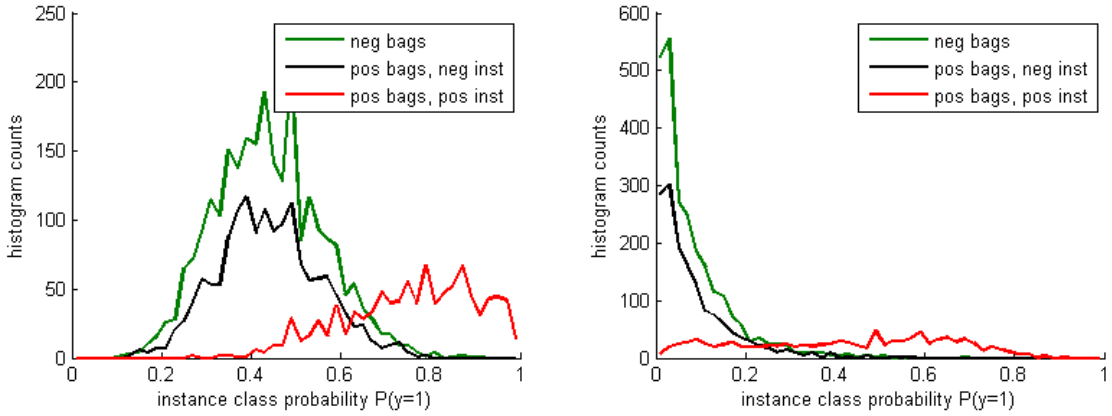


Figure 6.4: Histograms of estimated instance class probabilities. Left: first iteration. Right: last (20th) iteration.

negative log-likelihood of the bags

$$-\log P(c\,|\,\boldsymbol{\theta}) \quad = \quad -\log\left[P_{\mathrm{MI}}(\boldsymbol{y})\prod_n P_{\mathrm{oob}}(y_n\,|\,\boldsymbol{\theta})\right] \tag{6.24}$$

decreases with increasing number of iterations (see ), while the classification performance (oob-estimate of accuracy with optimum decision threshold and oob-estimate of area-under-ROC) increases. Convergence is reached after about 5 iterations.

Receiver-operator curves of the first and last iteration are shown in Figure 6.3. Besides the increase in classification performance, one can see a strong decrease of optimum decision threshold (0.98 to 0.54). At the first iteration there are many false positive labels, which causes positive predictions of all bags (with a decision threshold of 0.5 one would have 100% false positives). After the algorithm has estimated the latent instance classes, the observed optimum decision threshold is close to the theoretical value of 0.5.

The estimated instance class probabilities are shown in Figure 6.4. At the first iteration, the algorithm assigns a pretty large positive probability ($P(y = 1) \approx 0.4$) to the negative instances, because many of them have a positive bag labels. After the algorithm has estimated the instance classes, most truly negative instances are assigned a very small positive probability.

Figure 6.5 shows one tree of the first iteration's random forest and one from the last iteration. The last iteration's tree has a significantly lower number of leaves than the first iteration's tree. This shows that the tree has relaxed to a simpler structure as some instances from positive bags have adopted a negative class label. The average progress of the structure of all trees is shown in Figure 6.6. In agreement with the previous figure we observe that the number of leaves needed to fit the data drops significantly with the number of iterations, while the mean and maximum depth of the tree remain approximately constant.

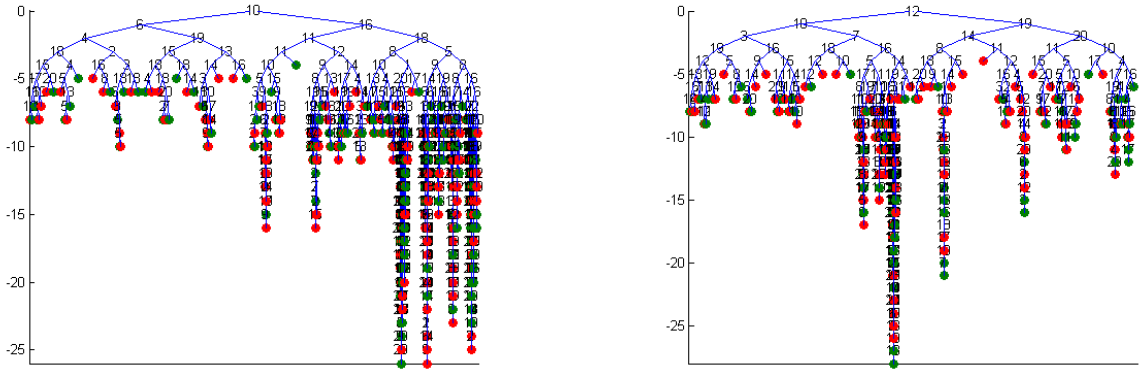

Figure 6.5: Structure of grown trees. Left: first iteration. Right: last (20th) iteration. Red bullets indicate positive leaves, green bullets indicate negative leaves. At each split the dimension used for that split is printed.

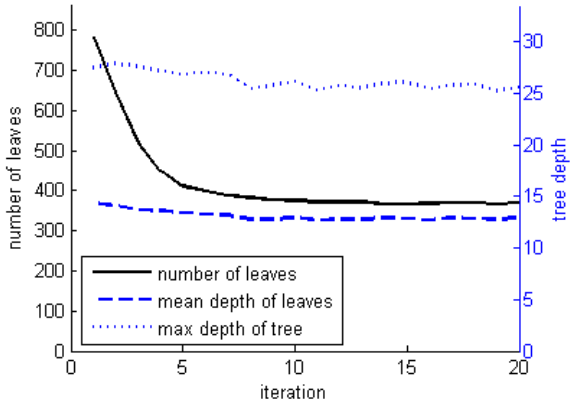


Figure 6.6: Progress of tree structure over iterations.

| Dataset | #Features (dimensions) | #Instances | #Bags (pos / neg) | #Instances per bag |
|---|---|---|---|---|
| MUSK1 | 166 | 476 | 47 / 45 | 2–40 |
| MUSK2 | 166 | 6598 | 39 / 63 | 1–1044 |

Table 6.1: Characteristics of MUSK datasets

## 6.3 Results on MUSK datasets

The MUSK datasets are a standard benchmark for multiple instance learning. They have been the first published datasets for the multiple-instance setting (Dietterich et al. 1997). The numbers of dimensions, bags, and instances is given in Table 6.1. Special traits of the MUSK datasets are their large variation of bag size (especially for MUSK2), and the low number of true positive instances in positive bags. (Although the true positive instance are not known, the origin of the dataset from drug activity prediction strongly suggests that there is rarely more than one positive instance per bag.)

The result of SMIRF on the MUSK1 dataset is shown in Figure 6.7. The negative log-likelihood decreases with the number of iterations, which shows that the algorithm works correctly. Surprisingly, however, the classification accuracy (oob-estimate) decreases as well. A possible explanation is that SMIRF estimates some of the bags with very high probability (leading to low NLL), but estimates other bags very badly (leading to low classification accuracy). This could happen if a few true positive instances are very close to a cluster of negative instances. In this case, SMIRF would be compelled to assign negative labels to these few true positives to increase the likelihood of the negative bags.

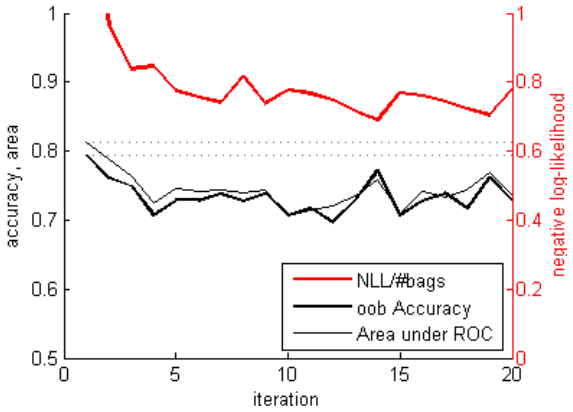


Figure 6.7: Results of SMIRF on MUSK1. Settings: Number of trees: $T = 100$, bootstrap sample size: $B = 92$, feature sample size: $[\sqrt{D}] = 13$.

Note that the standard random forest already yields a quite good classification accuracy (80% oob-estimate). A similar result has also been found by (Leistner et al. 2010), see Table 6.2.

One characteristic of the MUSK datasets is the low number of true positive instances. Since it is well-known that large class imbalance can severely impair classification performance, it is suggestive to try data balancing to improve the performance of SMIRF.

### 6.3.1 Data balancing

The term *imbalanced data* describes a dataset that has significantly more datapoints of one class than of the other. A comprehensive review of algorithms and tools for learning from imbalanced data is given by (He & Garcia 2009). There are two main classes of methods: Sampling methods and cost-sensitive methods. Sampling methods resample from the available data to obtain a balanced dataset that is used for classification. One advantage of these methods is that they are a preprocessing technique that can be used with any classifier without modifications. Cost-sensitive methods assign a very high cost to misclassification of the minority class. The objective function of the classifier has to be modified to incorporate the desired weights.

Methods to deal with imbalanced data have first been used with the random forest by (Chen, Liaw & Breiman 2004). They found that their proposed sampling method and cost-sensitive method both performed very well. The good result of Chen's sampling method has been confirmed by (Liu, Wu & Zhou 2009). Imbalanced classification is usually used in a supervised setting. We are aware of only one publication that uses balancing methods for semi-supervised learning (Li, Wang, Zhou & Lee 2011). For multiple instance learning, balancing is a new idea.

**SMIRF with data balancing** To deal with the problem of imbalanced data, we choose an oversampling method. At each step of SMIRF, we oversample the positive instances with replacement to obtain the same number of instances for both classes. This oversampling is performed after the instance classes have been drawn according to (6.21) and before the next iteration's trees are grown according to (6.20).

The result of SMIRF with data balancing on MUSK1 is shown in Figure 6.8. In contrast to previous section's results, the classification accuracy indeed increases over iterations as expected. Convergence is reach after about 20 iterations, and the predicted classification accuracy is on a competitive level (see Table 6.2).

To further explore why data balancing improves performance on MUSK1, we show example trees and the progress of tree shapes for both non-balanced and balanced SMIRF in Figure 6.9. Deep trees seem to be important for good classification performance. What happens during tree growing on balanced data is the following: The few positive instances are heavily oversampled, therefore many splits are made right next to the oversampled instances. To separate the oversampled instance from all surrounding negative instances, many consecutive splits in different dimensions are necessary. So oversampling positive

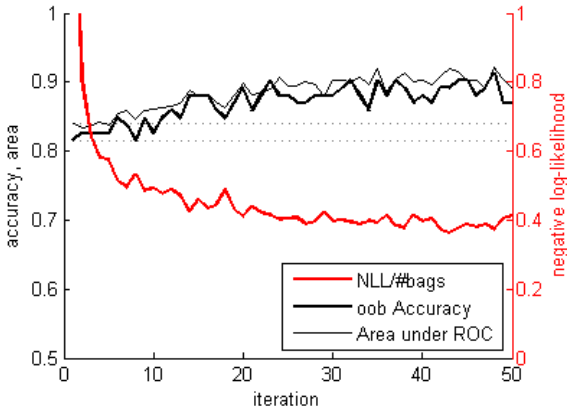


Figure 6.8: Results of MIRF with data balancing on MUSK1. Settings: Number of trees: $T = 100$, bootstrap sample size: $B = 92$, feature sample size: $[\sqrt{D}] = 13$.

instances does not lead to "overruling" of neighboring negative instances, but to a better confinement of the positive instances in many directions.

### 6.3.2 Bag-size-independent classification

In Chapter 4 we have proposed two methods to improve bag classification for given instance class probabilities, and we have shown that they work well on synthetic datasets. Here we apply these methods to the instance class probabilities as predicted by balanced SMIRF on the MUSK1 dataset. The results are shown in Figure 6.10. While the standard method (noisy-OR) gives an accuracy of 88%, both bag-size-independent and treewise bag classification improve the accuracy to 91%.

Figure 6.11 shows the receiver-operator-characteristic corresponding to bag-size-independent prediction. As expected, the optimum decision threshold is close to the theoretical value of 0.5.

To illustrate the difference between standard noisy-OR classification and bag-size-independent classification, we plot the predicted bag probability as a function of bag size for both classification methods (see Figure 6.12). One can clearly see that noisy-OR is biased towards predicting large bags as positive, which impedes class separation, while bag-size-independent prediction is bias-free.

**MUSK2** We applied the same method that has proven successful on MUSK1 (i. e. balanced SMIRF with bag-size-independent classification) on the MUSK2 dataset. The result is shown in Figure 6.13. With noisy-OR classification we achieve only 77% classification accuracy (mean oob-estimate), which is clearly below the reported performance other methods. With bag-size-independent classification accuracy improves to 85%, which is a competitive result (see Table 6.2.

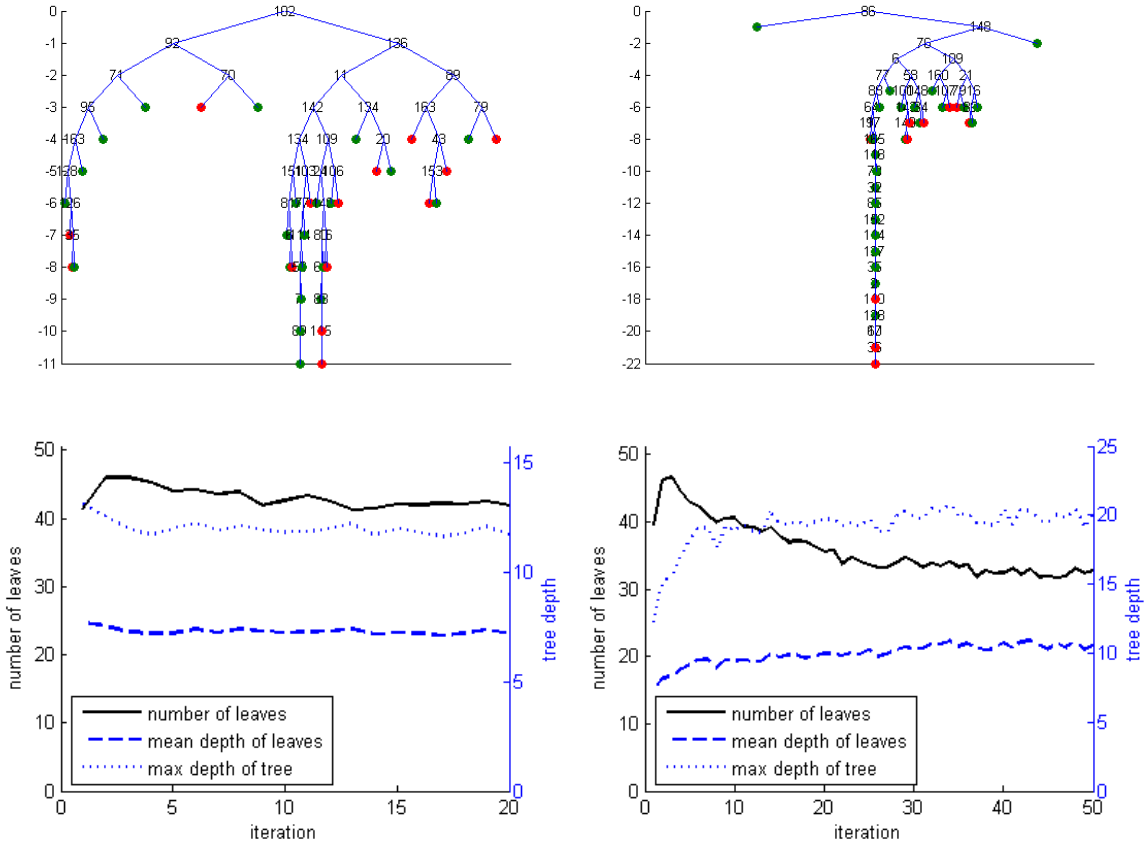


Figure 6.9: Example trees and progress of tree shapes over iterations for non-balanced SMIRF (left) and balanced SMIRF (right) on MUSK1. Settings: Number of trees: $T = 100$, bootstrap sample size: $B = 92$, feature sample size: $\lceil\sqrt{D}\rceil = 13$.

The trees grown by balanced SMIRF on MUSK2 have a similar shape to those grown on MUSK1. As can be seen in Figure 6.14, they have a very large depth in order to confine the positive region in feature space in many dimensions.

**Cross-validation** To check the above results, we performed 10-fold cross-validation on both MUSK1 and MUSK2. Surprisingly, the results were significantly worse than the out-of-bag predictions: 86% and 81% (CV estimate) vs. 91% and 85% (oob-prediction). The explanation for this effect is probably that each cross-validation run does not use all available data, but only 9/10 of the data. In large datasets this difference is not significant, but in the MUSK datasets there only 47 (39, resp.) positive bags, so that each single missing bag can impair classification.

While the oob-estimate is more meaningful for prediction accuracy of new unknown data, the cross-validation results are more significant for comparison with other methods, since

cross-validation is the standard methods that is available for all classifiers, not only those based on bagging.

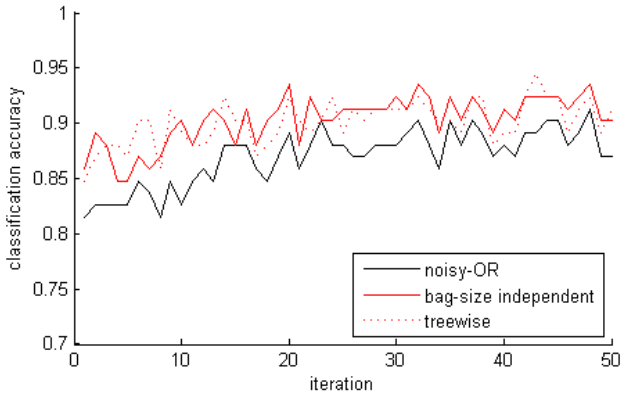


Figure 6.10: Comparison of bag classification methods on results of balanced SMIRF on MUSK1.

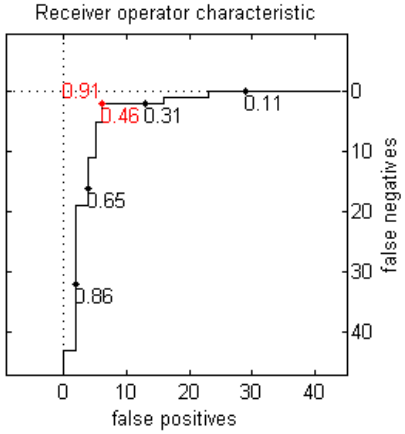


Figure 6.11: Receiver-operator-characteristic of balanced SMIRF on MUSK1 with bag-size-independent prediction. The number above the curve denotes optimum classification accuracy, the numbers below the curve denote corresponding decision thresholds.

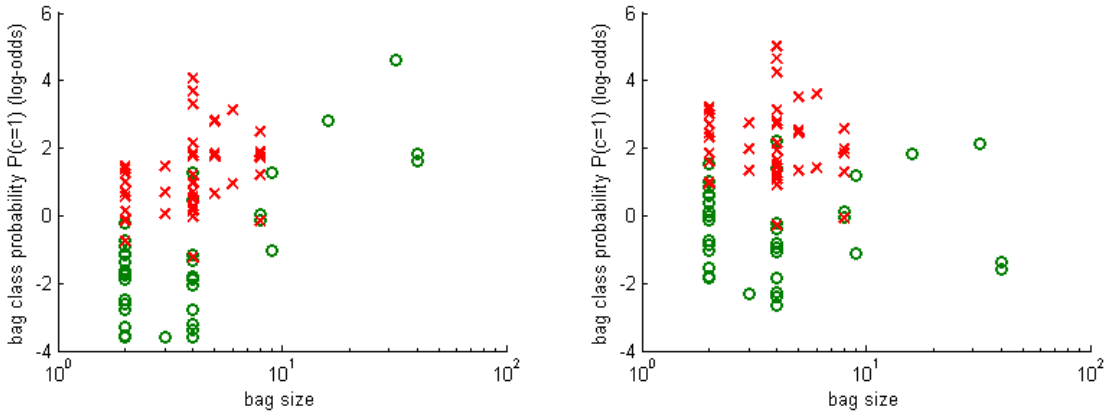


Figure 6.12: Predicted bag probabilities of standard noisy-OR (left) and bag-size-independent classification (right). Instance class probabilities are as predicted by balanced SMIRF on MUSK1.

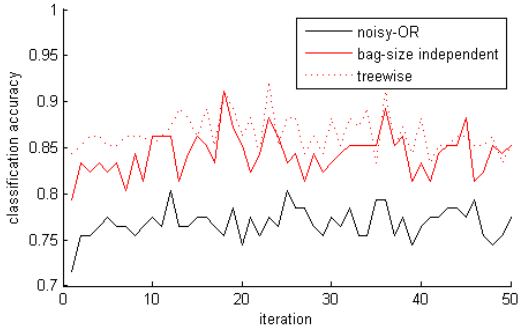


Figure 6.13: Comparison of bag classification methods on results of balanced SMIRF on MUSK2. Settings: Number of trees: $T = 100$, bootstrap sample size: $B = 92$, feature sample size: $[\sqrt{D}] = 13$.

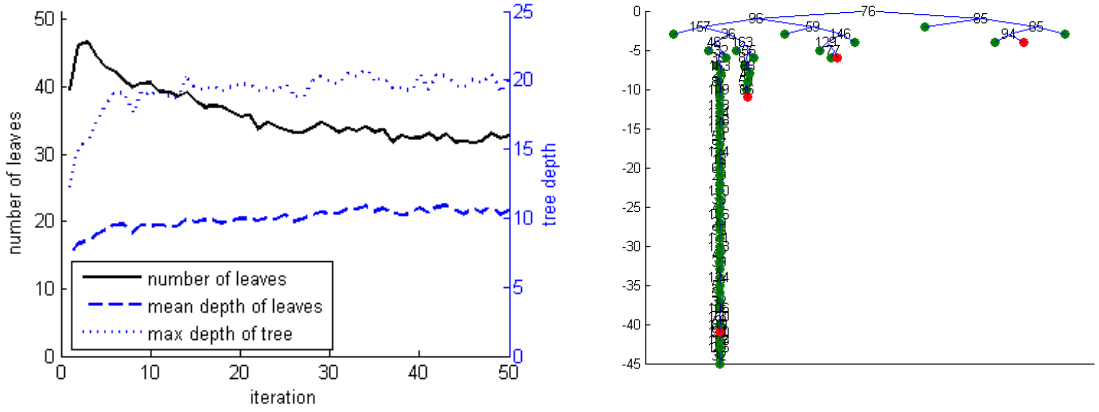


Figure 6.14: Shape of learned trees of balanced SMIRF on MUSK2.

| Algorithm | Reference | MUSK1 | MUSK2 |
|---|---|---|---|
| SMIRF oob | | 73 | |
| SMIRF bal. oob | | 88 | |
| SMIRF bal. bsi oob | | 91 | 85 |
| SMIRF bal. bsi CV | | 86 | 81 |
| Random forest | | 80 | 72 |
| Random forest | (Leistner et al. 2010) | 85 | 78 |
| MIForest | (Leistner et al. 2010) | 85 | 82 |
| PC-SVM | (Han et al. 2010) | **91** | **91** |
| miGraph | (Zhou, Sun & Li 2009) | 89 | 90 |
| AL-SVM | (Gehler & Chapelle 2007) | 86 | 83 |
| AW-SVM | (Gehler & Chapelle 2007) | 86 | 84 |
| mi-SVM | (Andrews et al. 2002) | 87 | 84 |
| MI-SVM | (Andrews et al. 2002) | 78 | 84 |
| EM-DD | (Zhang & Goldman 2001) | 85 | 85 |
| diverse density | (Maron & Lozano-Pérez 1998) | 89 | 82 |
| IAPR | (Dietterich et al. 1997) | **92** | 89 |

Table 6.2: Comparison of classification performance on MUSK datasets. If not specified otherwise, values have been obtained by 10-fold cross validation.

## 6.4 Results on DAGM data

Finally, we applied our proposed algorithm on the DAGM data presented in Section 1.5. In order to evaluate the classification performance, we divided each dataset into a training set and a test set of 575 images each.

The training datasets are very large (122 feature images for each of the 575 images), so that we cannot use the complete dataset for training. Note that there is a large class imbalance: only few image patches contain a defect (positive class), while most image patches show only the background (negative class). Therefore, we subsampled from the negative image patches, so that the resulting dataset is roughly balanced (approx. 1000–5000 image patches for each class).

The behavior of SMIRF on the DAGM datasets is shown in Figure 6.15. Shown is the result on a test image of "Class1". As already seen on the MUSK datasets, the first iteration (which corresponds to the standard random forest) yields an acceptable result. Although there is significant noise in the image (false positive instance class probabilities), the defect is clearly discernable.

With increasing number of iterations, SMIRF estimates more and more instances as negative. Consequently, the result image of iteration 5 is almost free of noise. However, on further iteration SMIRF also estimates more and more of the truly positive instances as negative. This is because SMIRF is satisfied if it finds a single positive instance per training image. If the positive patches in the test image do not correspond exactly to that single patch in the training image, they will not be recognized as positive. This inappropriate behavior of the multiple instance model can be corrected by replacing it with the power model (see Section 6.4.2), but before, we consider bag classification of a complete image.

### 6.4.1 Bag classification via threshold method

To arrive at a prediction for the complete image, we have to combine the class probabilities of the image patches to one image class probability. In terms of multiple instance learning, this corresponds to the task of bag classification.

However, we find that the standard bag classification method (noisy-OR) does not perform well on images. The reason is that noisy-OR effectively "adds up image noise", so that an image with some noise over a large area is assigned a larger defect probability than an image with a small defect but less large area noise.

A better choice is the threshold method, described in Section 4.2.2. From the viewpoint of the multiple instance model, the threshold method is a very crude approximation, but for image data it is a good practical choice.

Figure 6.16 shows ROC curves of the bag classification methods "noisy-OR" and "threshold" on the "Class1" dataset. While noisy-OR leads to a significant misclassification rate of approx. 7%, the threshold method achieves perfect classification.

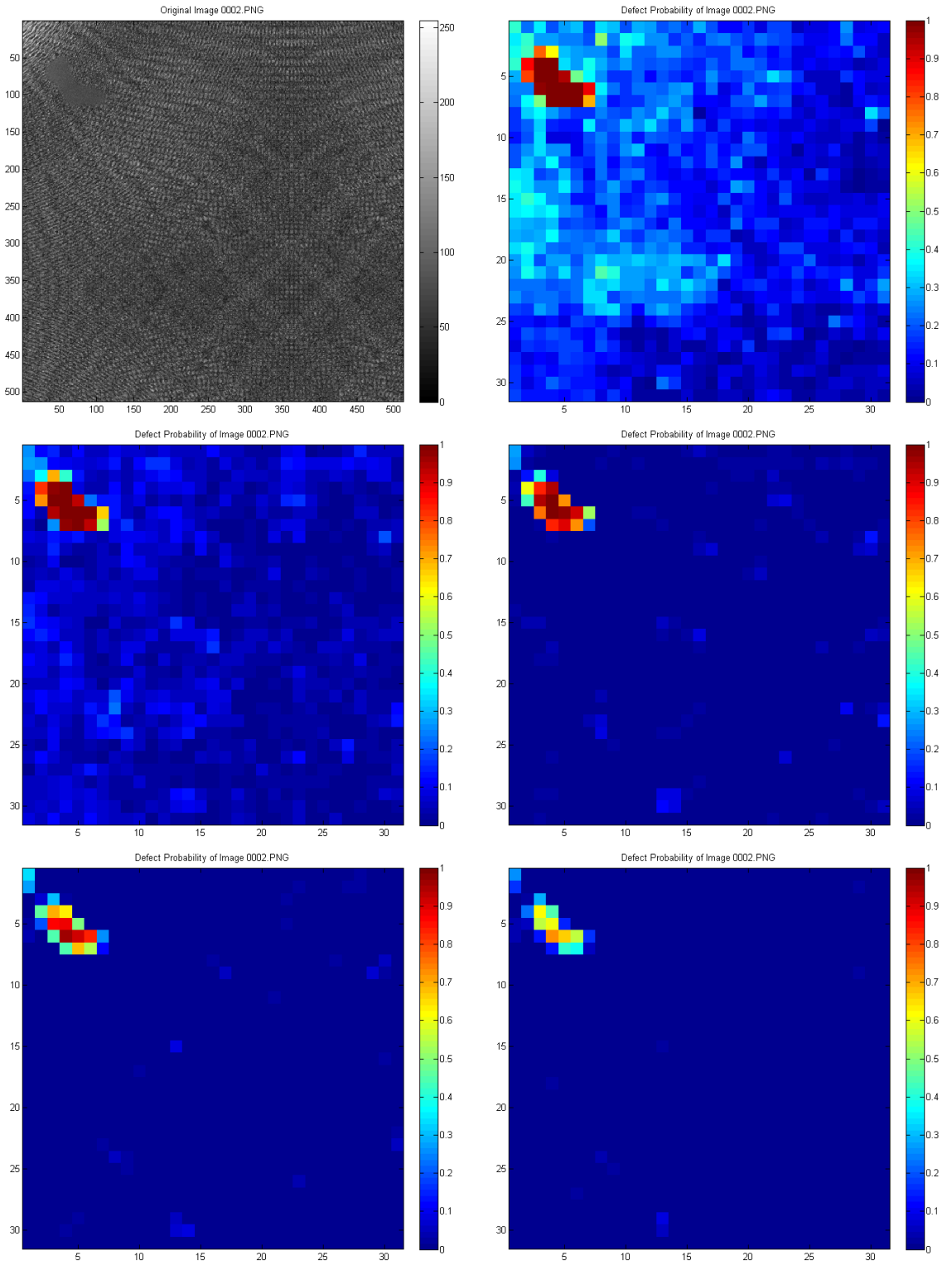


Figure 6.15: Behavior of SMIRF on DAGM dataset "Class1". Top left: original image. Remaining panels: instance class probabilities as estimated by SMIRF at iteration 1, 2, 5, 10, 20 (top to bottom).

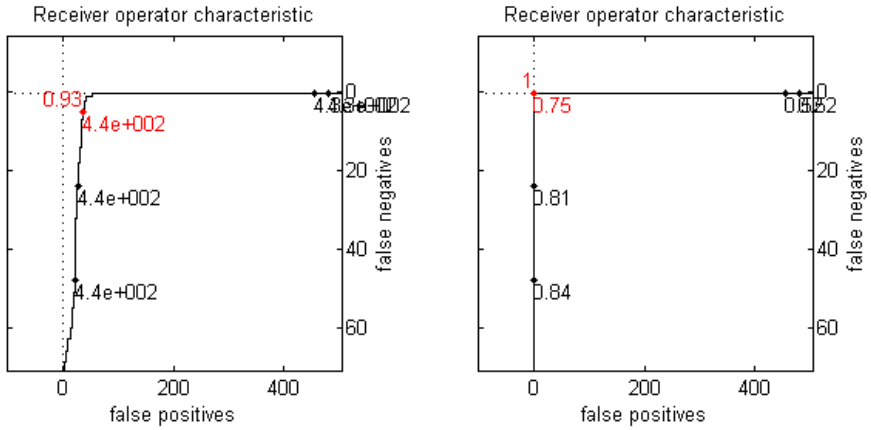


Figure 6.16: Comparison of the bag classification methods "noisy-OR" (left) and "threshold" (right).

### 6.4.2 SMIRF with power model

As observed above, the multiple instance model leads to inappropriate behavior of SMIRF of the DAGM data, because it searches for only one positive instance per bag, while there are in fact many. We have discussed this shortcoming of the multiple instance model in Chapter 5 and proposed two models for improvement.

The power model can be incorporated into SMIRF by adapting the sampling of instance classes. We replace the instance class probability $P_{\mathrm{MI}}$ in (6.21) with the instance class probability of the power model (5.43), and use the sampling technique described in 5.2.2.

The results of SMIRF with the multiple instance model and with the power model are shown in Table 6.3. The multiple instance model works well only on Class1 and Class3, on the other four datasets there is a significant number of misclassified images. The power model also does not achieve perfect classification on these four datasets, but the number of misclassfcations is much smaller.

One example image of each dataset is shown in Figures 6.17 and 6.17, together with the patch-wise class probabilities as estimated by SMIRF with the power model.

| | multiple instance | | power model | |
|---|---|---|---|---|
| | false pos | false neg | false pos | false neg |
| Class1 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 |
| Class2 | 1 | 7 | 1 | 0 |
| | 75 | 0 | 1 | 0 |
| Class3 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 |
| Class4 | 13 | 45 | 1 | 0 |
| | 457 | 0 | 1 | 0 |
| Class5 | 1 | 2 | 2 | 0 |
| | 16 | 0 | 2 | 0 |
| Class6 | 5 | 6 | 7 | 1 |
| | 42 | 0 | 9 | 0 |

Table 6.3: Comparison of SMIRF with standard multiple instance model and with power model ($\gamma = 1$) on DAGM dataset. In the upper line of each entry the decision threshold was set to achieve a minimum number of misclassifications; in the lower line it was set to achieve minimum false positives at zero false negatives.

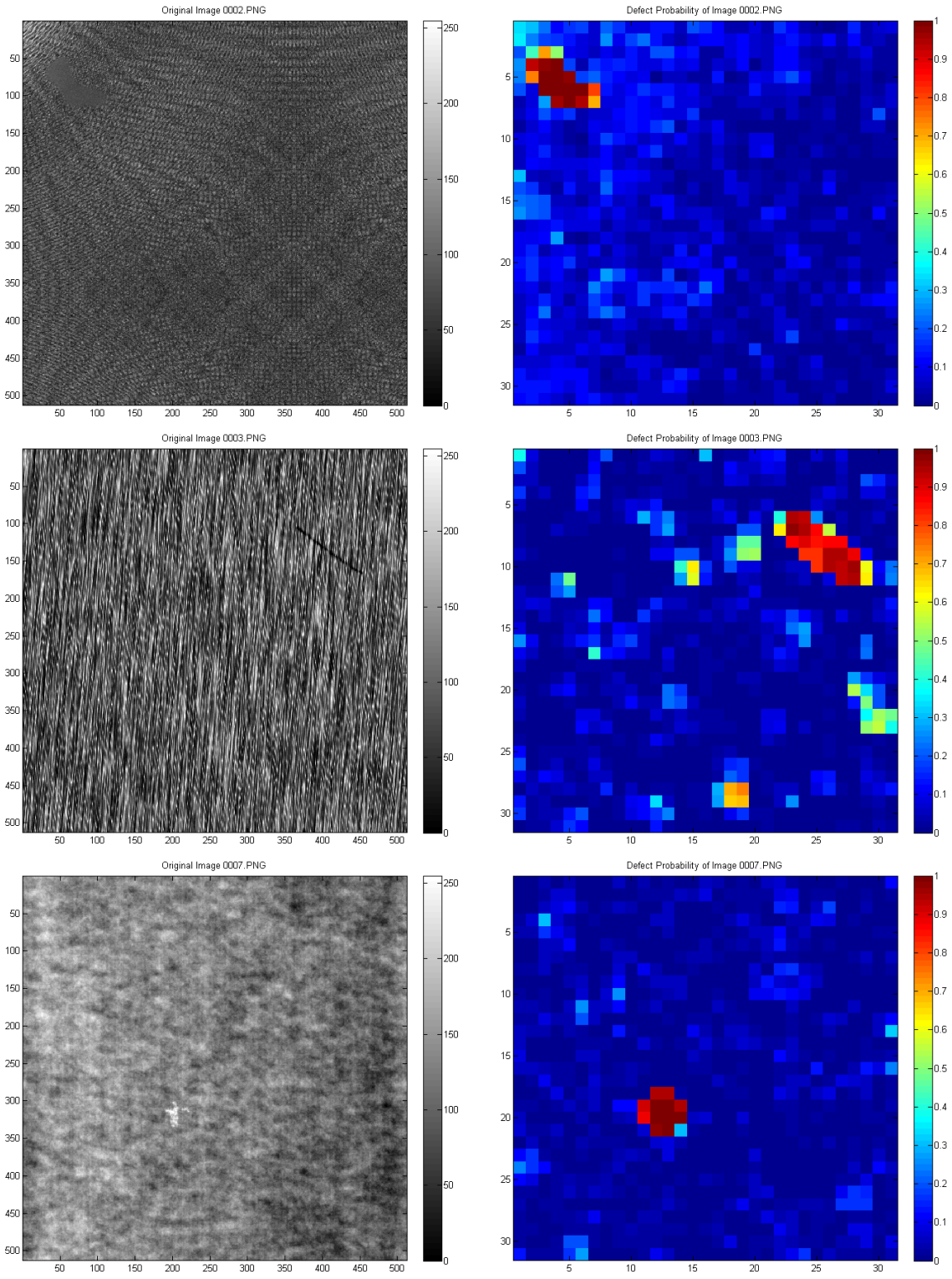


Figure 6.17: Results of SMIRF (power model with $\gamma = 1$) on DAGM datasets "Class1" to "Class3" (top to bottom).

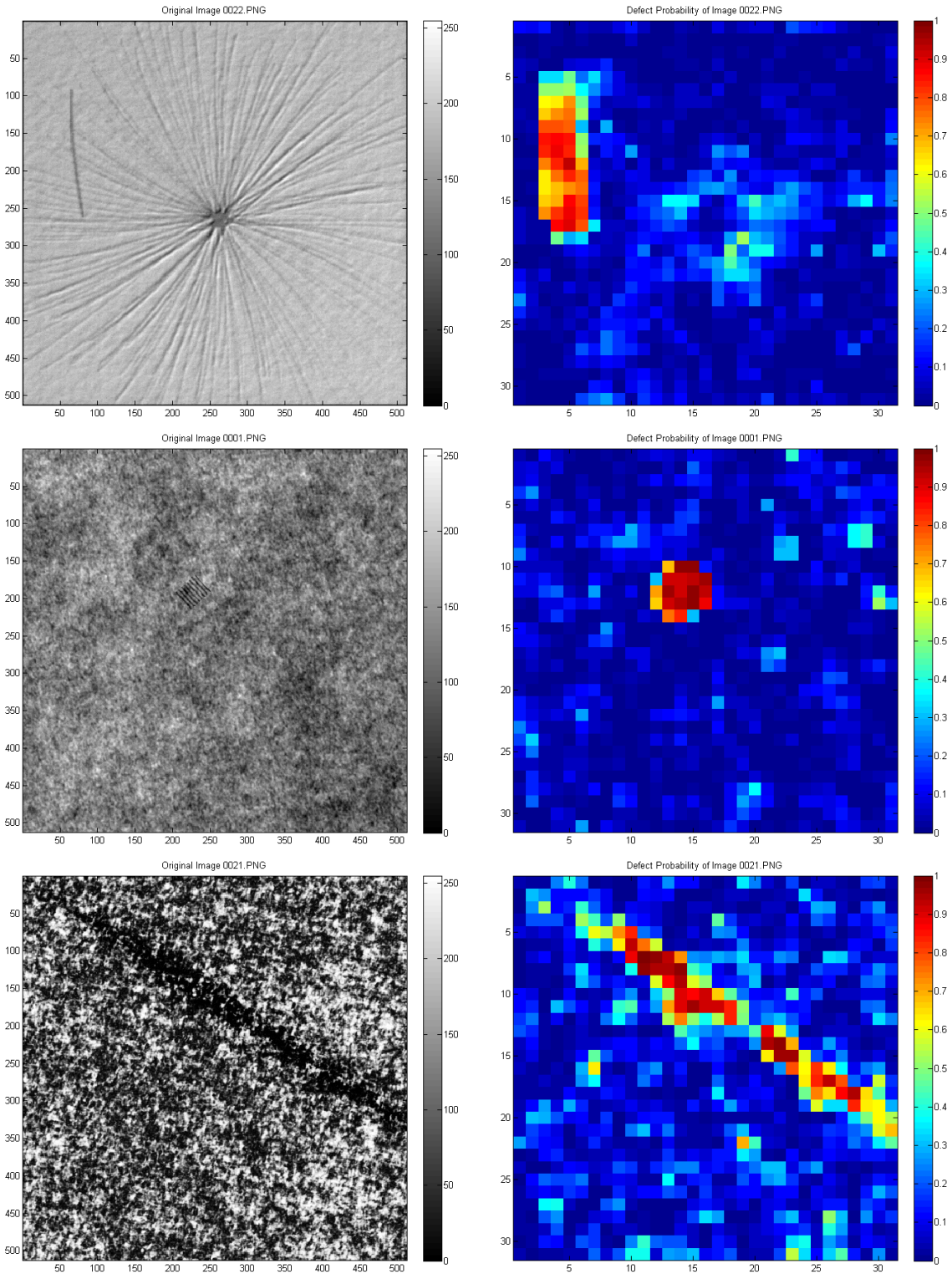


Figure 6.18: Results of SMIRF (power model with $\gamma = 1$) on DAGM datasets "Class4" to "Class6" (top to bottom).

# Chapter 7

# Conclusion

In this work we were concerned with learning from the type of labels that are available in industrial optical inspection and many other image processing applications: Negative images are completely negative, but positive images contain both negative and positive regions. In this situation multiple instance learning is an appropriate approach.

Most of the reported work on multiple instance learning is centered on specific algorithms and applications. We have treated the multiple instance model in greater depth and generality than can be found elsewhere. On this route we have discovered two new methods for bag classification (see below).

While the multiple instance model is an appropriate approach for our setting, there is room for improvement by using the information we have about the number of positive instances per bag. For this aim we have examined and implemented two alternatives to the multiple instance model.

The above suggestions have been implemented and tested on a standard multiple instance benchmark dataset and synthetic images that reflect the situation of industrial optical inspection

To summarize, the contributions of this thesis are:

1. **Bag size independent classification:** We have shown that the standard method of bag classification (noisy-OR) is biased towards classifying large bags as positive. This is true especially if the instance classifier provides noisy output. We have proposed a model that compensates for this behavior and yields a bag size independent estimate of the bag class probability. We have shown that this method can improve classification performance both on synthetic data and on a real-world dataset.

2. **Treewise classification:** We have shown that the noisy-OR method yields a wrong bag class probability if the instance class probabilities are correlated. Such a correlation is to be expected especially if there are only few positive instances per positive bag. We have suggested treewise bag classification, which is not impaired by correlation, and we have shown that it indeed improves classification performance on a standard multiple instance benchmark data.

3. **Bernoulli model:** In most applications where multiple instance learning is used, each positive bag contains not only one but multiple positive instances. The simplest way to adapt the multiple instance model to this setting is the Bernoulli model. We

have examined the properties of this model and found that it has a different effect on model training than it suggests at first sight, which makes it much less convincing.

4. **Power model:** As an alternative to the Bernoulli model we have suggested a power model. It has the advantage that its influence smoothly varies with the number of estimated positive instances per bag. In the case of few estimated positives per bag it strongly increases the positive instance probabilities to explain the bag, while in the case of many estimated positives it has only a minor effect.

5. **SMIRF algorithm:** We have proposed and implemented a self-training multiple instance random forest (SMIRF). Also, we have incorporated the Bernoulli model and the power model by adapting the method of instance class sampling. Using this algorithm, we have compared the behavior of the multiple instance model with the power model on an image dataset, and we have found that the power model indeed improves performance in this case.

# Bibliography


Amores, J. (2013), 'Multiple instance classification: Review, taxonomy and comparative study', *Artificial Intelligence* **201**(0), 81–105.

Andrews, S. & Hofmann, T. (2004), 'Multiple instance learning via disjunctive programming boosting', *Advances in Neural Information Processing Systems (NIPS)* **16**.

Andrews, S., Tsochantaridis, I. & Hofmann, T. (2002), 'Support vector machines for multiple-instance learning', *Advances in Neural Information Processing Systems (NIPS)* **15**, 561–568.

Babenko, B., Dollár, P., Tu, Z. & Belongie, S. (2008), Simultaneous learning and alignment: Multi-instance and multi-pose learning, *in* 'Workshop on faces in real-life images: Detection, alignment, and recognition'.

Babenko, B., Yang, M.-H. & Belongie, S. (2009), Visual tracking with online multiple instance learning, *in* 'Computer Vision and Pattern Recognition (CVPR)'.

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

Beyerer, J., León, F. P. & Frese, C. (2012), *Automatische Sichtprüfung: Grundlagen, Methoden und Praxis der Bildgewinnung und Bildauswertung*, Springer.

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

Bishop, C. M. & Lasserre, J. A. (2007), Generative or discriminative? getting the best of both worlds, *in* J. M. Bernardo, M. J. Bayarri, J. O. Berger, A. P. Dawid, D. Heckerman, A. F. M. Smith & M. West, eds, 'Bayesian Statistics 8', Oxford University Press, chapter 8, pp. 3–24.

Blockeel, H., Page, D. & Srinivasan, A. (2005), Multi-instance tree learning, *in* 'Proceedings of the 22nd International Conference on Machine Learning (ICML)', pp. 57–64.

Bouveyron, C. & Girard, S. (2009), 'Robust supervised classification with mixture models: Learning from data with uncertain labels', *Pattern Recognition* **42**(11), 2649–2658.

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

Breiman, L. (2001), 'Random forests', *Machine Learning* **45**, 5–32. Publisher: Kluwer Academic Publishers.

Buehler, P., Zisserman, A. & Everingham, M. (2009), Learning sign language by watching tv (using weakly aligned subtitles), *in* 'Computer Vision and Pattern Recognition (CVPR)'.

Bunescu, R. & Mooney, R. (2007), Multiple instance learning for sparse positive bags, *in* 'International Conference on Machine Learning (ICML)'.

Chapelle, O., Schölkopf, B. & Zien, A., eds (2006), *Semi-Supervised Learning*, The MIT Press.

Chen, C., Liaw, A. & Breiman, L. (2004), Using random forest to learn imbalanced data, Technical Report 666, Department of Statistics, Univ. of California, Berkeley.

Chevaleyre, Y. & Zucker, J.-D. (2001), Solving multiple instance and multiple part learning problems with decision trees and rule sets, application to the mutagenesis problem., *in* '14th Canadian Conference on Artificial Intelligence', pp. 204–214.

Daly, R., Shen, Q. & Aitken, S. (2011), 'Learning bayesian networks: approaches and issues', *The Knowledge Engineering Review* **26**(2), 99–157.

Demant, C., Streicher-Abel, B. & Springhoff, A. (2011), *Industrielle Bildverarbeitung: Wie optische Qualittskontrolle wirklich funktioniert*, Springer.

Deselaers, T. & Ferrari, V. (2010), A conditional random field for multiple-instance learning, *in* 'Proceedings of the 27th International Conference on Machine Learning (ICML)'.

Dietterich, T. G., Lathrop, R. H. & Lozano-Pérez, T. (1997), 'Solving the multiple-instance problem with axis-parallel rectangles', *Artificial Intelligence* **89**, 31–71.

Foulds, J. & Frank, E. (2010), 'A review of multi-instance learning assumptions', *The Knowledge Engineering Review* **25**(1), 1–25.

Gärtner, T., Flach, P., Kowalczyk, A. & Smola., A. (2002), Multi-instance kernels, *in* 'Proceedings of the 19th International Conference on Machine Learning (ICML)', pp. 179–186.

Gehler, P. V. & Chapelle, O. (2007), Deterministic annealing for multiple-instance learning, *in* 'International Conference on Artificial Intelligence and Statistics (AISTATS)', pp. 123–130.

Goldman, S. A. & Rahmani, R. (2006), Missl: Multiple-instance semi-supervised learning, *in* 'Proceedings of the 23rd International Conference on Machine Learning (ICML)'.

Goldman, S. & Zhou, Y. (2000), Enhancing supervised learning with unlabeled data, *in* 'International conference on machine learning (2000)'.

Han, Y., Tao, Q. & JueWang (2010), Avoiding false positive in multi-instance learning, *in* 'Advances in Neural Information Processing Systems (NIPS)', pp. 1–8.

Hastie, T., Tibshirani, R. & Friedman, J. (2009), *The Elements of Statistical Learning*, 2 edn, Springer.

He, H. & Garcia, E. A. (2009), 'Learning from imbalanced data', *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING* **21**(9), 1263–1284.

Hüllermeier, E. & Beringer, J. (2005), *Advances in Intelligent Data Analysis VI*, Springer Berlin Heidelberg, chapter Learning from ambiguously labeled examples, pp. 168–179.

Jähne, B. (2012), *Digitale Bildverarbeitung*, 7., revised edn, Springer.

Jin, R. & Ghahramani, Z. (2002), 'Learning with multiple labels', *Advances in neural information processing systems (NIPS)* pp. 897–904.

Karasaridis, A. & Simoncelli, E. (1996), A filter design technique for steerable pyramid image transforms, Technical report, GRASP Laboratory, University of Pennsylvania.

Kleeven, S. & Hyvärinen, L. (1999), 'Vision testing requirements for industry', *Materials Evaluation* **57**(8), 797–803. Introduction, http://www.asnt.org/publications/materialseval/basics/aug99basics/aug99basics.htm.

Kück, H., Carbonetto, P. & de Freitas, N. (2004), A constrained semi-supervised learning approach to data association, *in* 'Computer Vision-ECCV 2004', Springer Berlin Heidelberg, pp. 1–12.

Kück, H. & de Freitas, N. (2005), Learning about individuals from group statistics, *in* 'Proceedings of the Conference on Uncertainty in Artificial Intelligence'.

Lasserre, J. A., Bishop, C. M. & Minka, T. P. (2006), Principled hybrids of generative and discriminative models, *in* '2006 Conference on Computer Vision and Pattern Recognition (CVPR)', pp. 87–94. Kuer.

Leistner, C., Saffari, A. & Bischof, H. (2010), Miforest: Multiple-instance learning with randomized trees, *in* 'Proc. of the 11th European Conference on Computer Vision (ECCV)'.

Leistner, C., Saffari, A., Santner, J. & Bischof, H. (2009), Semi-supervised random forests, *in* 'Computer Vision, 2009 IEEE 12th International Conference on', pp. 506–513.

Li, M., Kwok, J. T. & Lu, B.-L. (2010), Online multiple instance learning with no regret, *in* 'Computer Vision and Pattern Recognition (CVPR)'.

Li, S., Wang, Z., Zhou, G. & Lee, S. Y. M. (2011), Semi-supervised learning for imbalanced sentiment classification, *in* 'Proc. 22nd international joint conference on Artificial Intelligence', Vol. 3, AAAI Press.

Li, Y., Tax, D. M. J., Duin, R. P. W. & Loog, M. (2013), 'Multiple-instance learning as a classifier combining problem', *Pattern Recognition* **46(3)**, 865–874.

Liu, X.-Y., Wu, J. & Zhou, Z.-H. (2009), 'Exploratory undersampling for class-imbalance learning', *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETIC-SPART B: CYBERNETICS* **39**(2), 539–550.

Maron, O. (1998), Learning from Ambiguity, PhD thesis, Massachusetts Institute of Technology.

Maron, O. M. & Lozano-Pérez, T. (1998), 'A framework for multiple-instance learning', *Advances in Neural Information Processing Systems (NIPS)* **10**, 570–576.

Maron, O. & Ratan, A. L. (1998), Multiple-instance learning for natural scene classification, *in* 'Fifteenth International Conference on Machine Learning (ICML)'.

Menze, B. H., Kelm, B. M., Splitthoff, D. N., Koethe, U. & Hamprecht, F. A. (2011), *Machine Learning and Knowledge Discovery in Databases*, Springer Berlin Heidelberg, chapter On oblique random forests, pp. 453–469.

Minka, T. P. (2000), Bayesian model averaging is not model combination, Technical report, MIT Media Lab Note.

Minka, T. P. (2005), Discriminative models, not discriminative training., Technical Report TR-2005-144, Microsoft Research, Cambridge, UK. Kuer.

Mitchell, T. M. (1999), The role of unlabeled data in supervised learning, *in* 'Proceedings of the sixth international colloquium on cognitive science'.

Natarajan, N., Dhillon, I., Ravikumar, P. & Tewari, A. (2013), Learning with noisy labels, *in* C. Burges, L. Bottou, M. Welling, Z. Ghahramani & K. Weinberger, eds, 'Advances in Neural Information Processing Systems 26', pp. 1196–1204.

Nettleton, D. F., Orriols-Puig, A. & Fornells, A. (2010), 'A study of the effect of different types of noise on the precision of supervised learning techniques', *Artificial Intelligence Review* **33**(4), 275–306.

Ramon, J. & De Raedt, L. (2000), Multi-instance neural networks, *in* 'Proc. of ICML-2000 Workshop on "Attribute-Value and Relational Learning"'.

Ray, S. (2001), Multiple instance regression, *in* 'Proc. of the 18th International Conference on Machine Learning (ICML)', Morgan Kaufmann, pp. 425–432.

Ray, S. & Craven, M. (2005), Supervised versus multiple instance learning: An empirical comparison, *in* 'Proc.s of 22nd International Conference on Machine Learning (ICML)'.

Rubin, D. B. (1981), 'The bayesian bootstrap', *The Annals of Statistics* **9**(1), 130–134.

Ruffo, G. (2000), Learning Single and Multiple Instance Decision Trees for Computer Security Applications, Doctoral dissertation, Department of Computer Science, University of Turin, Torino, Italy.

Sauer, P. (2008), Pattern recognition on statistically textured surfaces, Master's thesis, Ruperto-Carola University of Heidelberg, Germany.

Schoonard, J. W., Gould, J. D. & Miller, L. A. (1973), 'Studies of visual inspection', *Ergonomics* **16/4**, 365–379. Introduction.

Scott, S. D., Zhang, J. & Brown, J. (2005), 'On generalized multiple-instance learning', *International Journal of Computational Intelligence and Appliocations* **5**, 21–35.

Seeger, M. (2002), Learning with labeled and unlabeled data, Technical report, Institute for Adaptive and Neural Computation University of Edinburgh, 5 Forrest Hill, Edinburgh EH1 2QL.

Simoncelli, E. P. & Freeman, W. T. (1995), The steerable pyramid: A flexible architecture for multi-scale derivative computation, *in* 'Second International Conference on Image Processing'.

Singh, A., Nowak, R. & Zhu, X. (2008), Unlabeled data: Now it helps, now it doesn't, *in* 'Advances in Neural Information Processing Systems (NIPS)'.

Sorower, M. S. (2010), A literature survey on algorithms for multi-label learning, Technical report, Oregon State University, Corvallis, OR, USA.

Stikic, M. & Schiele, B. (2009), *Activity Recognition from Sparsely Labeled Data Using Multi-Instance Learning*, Vol. 5561/2009, Springer, chapter Lecture Notes in Computer Science, pp. 156–173.

Tabassiana, M., Ghaderia, R. & Ebrahimpourb, R. (2012), 'Combination of multiple diverse classifiers using belief functions for handling data with imperfect labels', *Expert Systems with Applications* **39**(2), 1698–1707.

Vezhnevets, A. & Buhmann, J. (2010), Towards weakly supervised semantic segmentation by means of multiple instance and multitask learning, *in* 'Computer Vision and Pattern Recognition (CVPR)'.

Vijayanarasimhan, S. & Grauman, K. (2008), Keywords to visual categories: Multiple-instance learning for weakly supervised object categorization, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)'.

Viola, P., Platt, J. C. & Zhang, C. (2006), 'Multiple instance boosting for object detection', *Advances in Neural Information Processing Systems (NIPS)* **18**, 1417–1426.

Wang, J. & Zucker, J.-D. (2000), Solving the multiple-instance problem: A lazy learning approach, *in* 'Proc. of the 17th International Conference on Machine Learning (ICML)', pp. 1119–1125.

Wang, Z., Radosavljevic, V., Han, B., Obradovic, Z. & Vucetic, S. (2008), Aerosol optical depth prediction from satellite observations by multiple instance regression, *in* 'Proc. 8th SIAM International Conference on Data Mining'.

Yang, J. (2005), Review of multi-instance learning and its applications, Technical report, School of Computer Science.

Zeisl, B., Leistner, C., Saffari, A. & & Bischof, H. (2010), On-line semi-supervised multiple-instance boosting, *in* 'Computer Vision and Pattern Recognition (CVPR)', pp. 1879–1879.

Zhang, C. & Viola, P. (2007), 'Multiple-instance pruning for learning efficient cascade detectors', *Advances in Neural Information Processing Systems (NIPS)* pp. 1681–1688.

Zhang, C. & Zhang, Z. (2010), A survey of recent advances in face detection, Technical report, Microsoft Research.

Zhang, K. & Song, H. (2013), 'Real-time visual tracking via online weighted multiple instance learning', *Pattern Recognition* **46**(1), 397–411.

Zhang, M. & Zhou, Z. (2013), 'A review on multi-label learning algorithms', *Knowledge and Data Engineering, IEEE Transactions on (Volume:PP , Issue: 99 )* **in press**(99).

Zhang, Q. & Goldman, S. A. (2001), 'EM-DD: An improved multiple-instance learning technique', *Advances in Neural Information Processing Systems (NIPS)* **14**, 1073–1080.

Zhang, Q., Goldman, S. A., Yu, W. & Fritts, J. E. (2002), Content-based image retrieval using multiple-instance learning, *in* 'Proc. of the 19th International Conference on Machine Learning (ICML)', pp. 682–689.

Zhou, Z.-H. (2004), Mulit-instance learning: A survey, Technical report, Nanjing University, National Laboratory for Novel Software Technology.

Zhou, Z.-H., Sun, Y.-Y. & Li, Y.-F. (2009), Multi-instance learning by treating instances as non-i.i.d. samples, *in* 'Proceedings of the 26th International Conference on Machine Learning (ICML'09)'.

Zhou, Z.-H. & Xu, J.-M. (2007), On the relation between multi-instance learning and semi-supervised learning, *in* 'Proc. of the 24th International Conference on Machine Learning (ICML)'.

Zhou, Z.-H. & Zhang, M.-L. (2002), Neural networks for multi-instance learning, Technical report, Nanjing University, National Laboratory for Novel Software Technology.

Zhou, Z.-H. & Zhang, M.-L. (2003), Ensembles of multi-instance learners, *in* 'Proc. of the 15th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)'.

Zhu, X. (2010), *Encyclopedia of Machine Learning*, Springer, chapter Semi-Supervised Learning.