University of Magdeburg

School of Computer Science

Bachelor Thesis

[The Title of the Thesis]

Author:

[Forename] [Surname]

[Month 13, 2014]

Advisors:

Prof. *[Name]*

Department of *[...]*

***[Surname]*, *[Forename]*:**
[The Title of the Thesis]
Bachelor Thesis, University of Magdeburg, *[2014]*.

Abstract

[...]

Contents

List of Figures

List of Tables

List of Code Listings

1. Introduction

[...]

Goal of this Thesis

[...]

Structure of the Thesis

[...]

2. Background and Related Work

The main focus of our work is the estimation of classifier performance in the special case of an active learning scenario. To help better understand the problem setting, we will give a brief overview of the concepts of active learning as well as an in-depth look of existing methods for said estimation. We also establish parts of the notation that will be used for the remainder of this work.

2.1 Classification and Active Learning

Many problems to be solved encompass the differentiation between certain "classes" to which their inputs can be assigned. To explain the concept of **classification** we assume the example of a thermostat. Its purpose is to monitor the temperature in a certain area and fire up a heating unit to raise the temperature if necessary. But for that to happen, the thermostat has to *classify* the measured temperature in the categories "too low" and "warm enough". In this special case a simple threshold usually suffices. Generally, however, a *classifier* C is defined as a mapping of a *feature vector*, also called an *instance*, to its corresponding *class label* $y \in Y = \{y_1, ..., y_m\}$. It is convenient to describe said vector as an n-tuple of *feature values*: $\vec{x} = (f_{a_1,1}, ..., f_{a_n,n})$ with $f_{i,j} \in F_j$ $\forall i = \{1, ..., |F_j|\}, j = \{1, ..., n\}$ and F_j as a *feature*. A classifier can then be written as

$$C : F_1 \times ... \times F_n \to Y \tag{2.1}$$

[RPL13] This definition makes it clear that classification is not restricted to single-variable problems. To stick with our example of a thermostat, you may want to consider the humidity, temperature outdoors or whether any windows are open to decide if heating is necessary.

To obtain a classifier for a specific problem, one makes use of what we will call a *learning algorithm* A. In a process called *training* they take a set of instances, the *training data* $X_T = \{\vec{x}_1, ..., \vec{x}_n\}$, and create such a mapping C using an optimization

criterion. The training set is a subset of all expected data and ideally independently and identically distributed [RPL13]. While all learning algorithms are being fed feature vectors to create a classifier, they can be separated into three categories, depending on their requirement for labeling:

- **Supervised:** This type of algorithm requires all of their input data to be labeled as well as a list of all possible class labels. While it may seem like the best option, its drawbacks include the potential cost of the labeling. If the correct class labels are not inherently known, usually a human is required to assign the correct labels manually. With unlabeled data readily available for problems like speech recognition, this is a very expensive part of the process. Another issue is the propagation of errors; any mistake made during labeling is carried over into the modeling of the classifier.
- **Unsupervised:** Instead of requiring all data to be labeled, unsupervised algorithms ignore class labels completely. This saves the cost of labeling the data, but many algorithms require some sort of tuning parameters (e.g. the cardinality of Y and shape of underlying probability distribution). It is similar to the process of density estimation from statistics.
- **Semi-supervised:** As a compromise of the two extremes, semi-supervised techniques operate with a mix of labeled and unlabeled data. This way it seeks to combine the low effort for labeling with the advantages of supervised learning.

Regardless of their category, all learning methods make assumptions about the distribution of the data: feature vectors close to each other tend to belong to the same class and, consequently, data points are likely to form group-like structures.

Active learning is the name of a subgroup of semi-supervised methods which will be a center point of this work. To reduce the amount of labeled data needed, they choose one or more data points to be labeled, commonly with the intent to incrementally improve the built classifier's performance by adding more labeled data every iteration. Various methods to select the next data point(s) to be labeled exist; we will briefly introduce two of them: *Uncertainty sampling* and *Probabilistic Active Learning (PAL)*. The *uncertainty* of a classifier with regard to a data point describes how unsure it is about its assigned class label. This can be either seen as the distance to a *decision boundary*, that is the entity separating data points of different classes [SDW01], or as the posterior probability estimation for the class assignment of your classifier [ZWYT08]. The probabilistic approach doesn't rely on uncertainty, instead it maximizes the expected performance gain in all data point neighbourhoods [KKS14]. A more in-depth description is given in Chapter 4.

2.2 Generalization Performance

When a learning algorithm is used to induce a classifier, an obvious question is how well said classifier performs. This can be used to select an algorithm when multiple are available or to simply get an estimate on how often the classifier will misclassify data.

To facilitate the calculations further down a closer look at the training set X_T is helpful. As earlier stated, the individual instances are supposed to be independently and identically distributed, which means the set can be seen as a random variable. Now the probability of a specific set depends on the probabilities to draw each of the instances and, in case of supervised and semi-supervised learning, their associated labels. Due to their independence, we can write it as

$$p(X_T) = p(\vec{x}_1, y_1) \cdot ... \cdot p(\vec{x}_n, y_n) = \prod_{i=1}^{n} p(\vec{x}_i, y_i) \tag{2.2}$$

[RPL13]. A similar formula applies for unsupervised learning; difference is only the probability of instances are relevant.

2.2.1 (Expected) prediction error, training error and accuracy

An intuitive metric for classifier performance is the so-called *training error*. After the training phase on a training set X_T is finished, one could classify every training element with the built classifier and use the ratio of correctly classified data points to the size of the training set, also known as *accuracy*. This way, classifiers of most algorithms will start with near-zero values when provided few instances and converge towards an accuracy of 100%. In reality however, judging a classifier solely based on the training error leads to *overfitting*; while the classifier operates well on the training set, at a certain point the performance on different data sets suffers [Die95].

2.2.2 Bias and variance

2.2.3 Classificator-based estimators

2.2.4 Cross-Validation

2.2.5 Bootstrapping

2.3 Learning Curves and Regression Models

2.3.1 Function families

2.3.2 Algorithms

3. Example Chapter

This chapter gives you some examples how to include graphics, create tables, or include code listings. But first, we start with a short description how you can efficiently cite in LaTeX. The following footnote shows you how to reference URLs and where this document is available online.[1]

3.1 Citation

There are several types of literature. The most citations are workshop and conference papers. Please use the inproceedings-tag for those citations (e.g., [**?**]). You should have short-hands for workshop and conference names to be sure the naming is consistent and uniform (see our BibTeX files how to do that).

Slightly different are articles published in journals (e.g., [**?**]). Make sure you that the volume and number-tags are present and that no inproceeding is tagged as article or vice versa.

You might want to take a look at the example BibTeX file to find out how to cite books [**?**], technical reports [**?**], websites [**?**], PhD theses, or master theses [**?** **?**].

3.2 Formulas

There are different types of mathematical environments to set formulas. The equation $E = m \cdot c^2$ is an inline formula. But you can also have formulas at a separate line (see Equation 3.1).

$$P = \big(\mathcal{A} \Rightarrow (\mathcal{B} \Leftrightarrow \mathcal{C}) \wedge (\mathcal{B} \Leftrightarrow \mathcal{D})\big) \wedge (\mathcal{B} \Rightarrow \mathcal{A}) \wedge (\mathcal{C} \Rightarrow \mathcal{A}) \wedge (\mathcal{D} \Rightarrow \mathcal{A}) \tag{3.1}$$

[1] http://www.ovgu.de/tthuem

If you need multiple lines that are aligned to each other, you might want to use the following code.

$$
\begin{aligned}
& \text{GraphLibrary} \\
\wedge\ & (\text{GraphLibrary} \Rightarrow \text{Edges}) \wedge (\text{Edges} \vee \text{Algorithms} \Rightarrow \text{GraphLibrary}) \\
\wedge\ & (\text{Edges} \Leftrightarrow \text{Directed} \vee \text{Undirected}) \wedge (\neg\text{Directed} \vee \neg\text{Undirected}) \\
\wedge\ & (\text{Algorithms} \Leftrightarrow \text{Number} \vee \text{Cycle}) \\
\wedge\ & (\text{Cycle} \Rightarrow \text{Directed}).
\end{aligned}
$$

3.3 Graphics

In Figure 3.1, we give a small example how to insert and reference a figure.

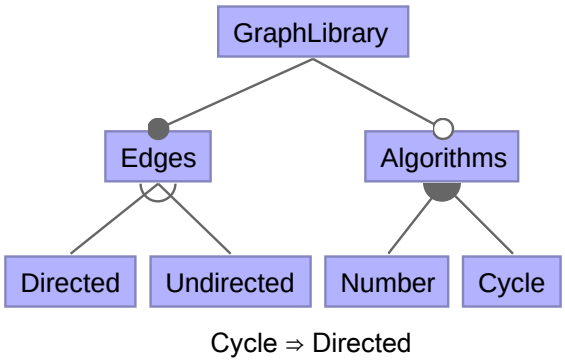

Figure 3.1: A feature model representing a graph product line

3.4 Tables

Table 3.1 shows the result of a simple tabular environment.

Group Type	Propositional Formula
And	$(P \Rightarrow C_{k_1} \wedge \ldots \wedge C_{k_m}) \wedge (C_1 \vee \ldots \vee C_n \Rightarrow P)$
Or	$P \Leftrightarrow C_1 \vee \ldots \vee C_n$
Alternative	$(P \Leftrightarrow C_1 \vee \ldots \vee C_n) \wedge \text{atmost1}(C_1, \ldots, C_n)$

Table 3.1: Mapping a feature model to a propositional formula

3.5 Code Listings

In Listing 3.1 on the next page, we give an example of a source code listing.

```
1  class A extends Object {
2      A() { super(); }
3  }
4  class B extends Object {
5      B() { super(); }
6  }
7  class Pair extends Object {
8      Object fst;
9      Object snd;
10     Pair(Object fst, Object snd) {
11         super(); this.fst=fst; this.snd=snd;
12     }
13     Pair setfst(Object newfst) {
14         return new Pair(newfst, this.snd);
15     }
16 }
```

Listing 3.1: Java source code

4. Evaluation

[...]

5. Related Work

Stuff

6. Conclusion

[...]

7. Future Work

[...]

A. Appendix

[...]

Bibliography

[Die95] Tom Dietterich. Overfitting and undercomputing in machine learning. *ACM Computing Surveys*, 27(3):326–327, September 1995. (cited on Page 5)

[KKS14] Georg Krempl, Daniel Kottke, and Myra Spiliopoulou. Probabilistic active learning: Toward combining versatility, optimality and efficiency. In *Proceedings of the 17th International Conference on Discovery Science*, October 2014. (cited on Page 4)

[RPL13] Juan D. Rodríguez, Aritz Pérez, and Jose A. Lozano. A general framework for the statistical analysis of the sources of variance for classification error estimators. *Pattern Recognition*, 46(3):855–864, March 2013. (cited on Page 3, 4, and 5)

[SDW01] Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active hidden markov models for information extraction. In *Proceedings of the Internation Symposium on Intelligent Data Analysis*, 2001. (cited on Page 4)

[ZWYT08] Jingbo Zhu, Huizhen Wang, Tianshun Yao, and Benjamin K. Tsou. Active learning with sampling by uncertainty and density for word sense disambiguation and text classification. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 1137–1144, August 2008. (cited on Page 4)

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Magdeburg, den *[. . .]*