TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

Tapio Manninen

Predictive Modeling Using Sparse Logistic Regression with Applications

Julkaisu 1190 • Publication 1190

Tampere 2014

Tampereen teknillinen yliopisto. Julkaisu 1190
Tampere University of Technology. Publication 1190

Tapio Manninen

Predictive Modeling Using Sparse Logistic Regression with Applications

Thesis for the degree of Doctor of Science in Technology to be presented with due permission for public examination and criticism in Tietotalo Building, Auditorium TB109, at Tampere University of Technology, on the 31st of January 2014, at 12 noon.

Tampereen teknillinen yliopisto - Tampere University of Technology
Tampere 2014

ISBN 978-952-15-3226-9 (printed)
ISBN 978-952-15-3233-7 (PDF)
ISSN 1459-2045

Abstract

In this thesis, sparse logistic regression models are applied in a set of real world machine learning applications. The studied cases include supervised image segmentation, cancer diagnosis, and MEG data classification. Image segmentation is applied both in component detection in inkjet printed electronics manufacturing and in cell detection from microscope images. The results indicate that a simple linear classification method such as logistic regression often outperforms more sophisticated methods. Further, it is shown that the interpretability of the linear model offers great advantage in many applications. Model validation and automatic feature selection by means of ℓ_1 regularized parameter estimation have a significant role in this thesis. It is shown that a combination of a careful model assessment scheme and automatic feature selection by means of logistic regression model and coefficient regularization create a powerful, yet simple and practical, tool chain for applications of supervised learning and classification.

Preface

This work has been carried out at the Department of Signal Processing, Tampere University of Technology (TUT), during 2011–2013. The funding was provided by Tampere Graduate School in Information Science and Engineering (TISE). I thank TISE and feel sorry that they had to shut the graduate school down. I surely miss those annual cruise seminars with the good people on board.

I wish to express my sincerest gratitude to my thesis supervisor Prof. Ari Visa. Without a doubt, he is the wisest man I have ever met. Whenever was I unsure about the impact of my dissertation for the scientific community, a short discussion with Prof. Visa helped me to continue with a perfect reasoning, structure, and logic in my mind.

In addition to Prof. Visa, I'm deeply obliged to thank my closest supervisor Heikki Huttunen. There is no question that Heikki is the single most influential person to my work as well as in gaining any of my academic achievements. Anyone else being my supervisor and I dare to claim that I would not be writing this preface right now. This claim is based on the simple fact that, to the best of my knowledge, Heikki is the most committed scientist there is. He always knows what to do and when to do, even in desperate situations. This has also reflected into my work via his supervision.

In addition to my supervisors, I'd like to express gratitude to all my coauthors and all the great people that I have been allowed to work with during the last couple of years. These people include Risto Rönkkä, the guy who had enough faith to leave his well established position in a big mobile phone company and start a printed electronics business with a bunch of tech heads like me. There is also Kalle Rutanen, with whom I have always had the most interesting and fruitful discussions (mainly about technical stuff such as geometric transformations) and Pekka Ruusuvuori, the tireless scientist with whom I'm grateful to have been able to do science as well as to make a couple of experience gaining conference trips. There are also others than the

people mentioned above and I thank you as well. It must be the positive and encouraging atmosphere at TUT, especially at the Department of Signal Processing, that collects all these good people here.

Last but not least, I'd like to let my family and friends know that your contribution to whatever I've been doing in my chamber for the last couple of years has been more substantial than you think. I will happily continue to clear my mom's laptop from malware and keep lying to my friends about curing the cancer as long as I can keep you in my life as a counter weight for my work. Special thanks `goto` the people in the greatest office band ever, to you Pasi, Frank, and Jenni, for offering me the joy of playing music and having fun with such a good group of people. I'd also like to mention all the great friends that I have, especially those made during the year 2013. I'm grateful to Maria who has introduced me to several magnificent people as well as to the London Gang with whom we have had multiple marvelous adventures. The last year has undoubtedly been *the best of my life* since childhood. That's because of you.

So, I started as a research assistant in the Department of Signal Processing in 2007 and was extremely lucky to be able to work through my Bachelor's, Master's, and PhD, all in a single continuous and logical sequence while simultaneously working in a series of interesting machine learning projects. Now that I'm finishing my PhD, it seems that the time has come for my academic career to come to end and for me to continue my journey in the scary world of *industry*. If one thing is sure, however, you never know what the future brings you.

Tapio Manninen
Tampere, 2014

Contents

Abstract i

Preface ii

List of Abbreviations vi

Mathematical Notation viii

1 Introduction 1
- 1.1 Machine Learning and Pattern Recognition 3
- 1.2 Introduction to Linear Classification 5
- 1.3 Objectives 8
- 1.4 Outline 9
- 1.5 Publications and Author's Contribution 10

2 Logistic Regression Classification and Sparse Parameter Estimation 12
- 2.1 Background 12
- 2.2 Logistic Regression Model 13
- 2.3 Learning the Model Coefficients 15
  - 2.3.1 Maximum Likelihood Method 15
  - 2.3.2 Regularization 16
  - 2.3.3 Bayesian Methods 19
- 2.4 Markov Random Field Priors 23

3 Model Selection and Error Estimation 28
- 3.1 Measuring Model Performance 28
  - 3.1.1 Counting Based Performance Measures 29
  - 3.1.2 Order Based Performance Measures 31
- 3.2 Model Validation and Automatic Parameter Selection . . 33
  - 3.2.1 Cross-validation 34

3.2.2 Bootstrapping 35
3.2.3 Parametric Methods 35
3.2.4 Pitfalls in Cross-Validation 38
3.2.5 The Effect of Sample Size in Error Estimation . . 41

4 Case Studies **44**
4.1 Supervised Image Segmentation 44
4.1.1 Overview of the Supervised Image Segmentation Framework 45
4.1.2 Segmentation of Cell Images 47
4.2 Object Detection in Inkjet Printed Electronics Manufacturing 50
4.2.1 Inkjet Printed Electronics and the Problem of Misaligned Components 50
4.2.2 Computer Vision Controlled Printing 52
4.2.3 Detection of Connection Pads from Camera Images 54
4.2.4 Experiments 57
4.3 Automated Diagnosis of Acute Myeloid Leukemia 59
4.3.1 Flow Cytometry in AML Diagnosis 59
4.3.2 Supervised Learning Methods for AML Diagnosis and Marker Analysis 61
4.3.3 Experiments 67
4.4 Mind Reading from MEG Data 71
4.4.1 Background 72
4.4.2 Data from ICANN 2011 Mind Reading Challenge 73
4.4.3 Using Logistic Regression for Recognition of Viewed Movie Type 75
4.4.4 Experiments 77
4.5 Discussion 80

5 Conclusions **82**

Bibliography **86**

List of Abbreviations

ACC	Accuracy
AIC	Akaike information criterion
AML	Acute myeloid leukemia
AUC	Area under curve
BCI	Brain computer interface
BEE	Bayesian error estimator
BIC	Bayesian information criterion
BFGS	Broyden-Fletcher-Goldfarb-Shanno update
CV	Cross-validation
DRM	Discriminative random field
EBIC	Extended BIC
EDF	Empirical cumulative distribution function
EEG	Electroencephalography
fMRI	Functional magnetic resonance imaging
GLM	Generalized linear model
HMM	Hidden Markov model
IC	Integrated circuit
IRLS	Iteratively reweighted least squares
L-BFGS	Limited memory BFGS update
LASSO	Least absolute shrinkage and selection operator
LDA	Linear discriminant analysis
LOO	Leave-one-out
LR	Logistic regression
MAP	Maximum a posteriori
MCMC	Markov chain monte carlo
MEG	Magnetoencephalography
ML	Maximum likelihood
MMSE	Minimum mean-square error
MRF	Markov random field
MSE	Mean squared error
NN	Nearest neighbors classifier

NPV	Negative predictive value
PCB	Printed circuit board
PML	Penalized maximum likelihood
PPM	Point pattern matching
PPV	Positive predictive value or precision
PR	Precision-recall
RFID	Radio Frequency Identification
ROC	Receiver operating characteristic
SMLR	Sparse multinomial logistic regression (algorithm)
SVM	Support vector machine
TNR	True negative rate or specificity
TPR	True positive rate or recall or sensitivity
VUS	Volume under the surface

Mathematical Notation

x, A	Scalars
$\mathbf{x}$	Vector
$\mathbf{A}$	Matrix
A_{ij}	Matrix element
$\mathcal{D}$	Set
$\hat{x}, \hat{\boldsymbol{\beta}}, \hat{\mathbf{A}}$	Estimates of scalars, vectors, and matrices
$\|\| \cdot \|\|_p$	Vector ℓ_p norm
$\|\| \cdot \|\|$	Vector ℓ_2 norm
$\mathbb{Z}^+$	The set of positive integers
$\mathbb{R}$	The set of real numbers
$\mathbb{R}^{\mathrm{n}}$	The set of n-dimensional real numbers
$F(\cdot), g(\cdot)$	Functions
$\nabla f(\cdot)$	Gradient function
$\exp(\cdot)$	The exponential function
$\log(\cdot)$	The natural logarithm function
$\mathrm{sgn}(\cdot)$	The sign function
$\mathrm{tr}(\cdot)$	Matrix trace
$\delta(\cdot)$	The unit impulse function
$\max_x\{\cdot\}$	Maximum value w.r.t x
$\arg\max_x\{\cdot\}$	Maximizing argument x
$p(\cdot)$	(Prior) probability
$p(\,\cdot\,\vert\,\cdot\,)$	Conditional probability
$\mathcal{O}(\cdot)$	The big-O notation for algorithm complexity

Chapter 1

Introduction

Humans possess a remarkable talent in using their senses to recognize patterns from the signals emitted from their surrounding world. The way we understand spoken language or written text, recognize people by their faces or the sound of their voice, or distinguish between chopped peach and squash merely by their taste is a result of our highly developed neural system and cognitive skills.

In today's modern world, there is a demand for building machines that can make similar decisions as humans can. In many fields, we have succeeded quite well. There exists an automated face recognition system in our pocket camera, there is a license plate recognition system in the car park automatically, without human supervision, reading the characters in the license plate shown in the surveillance camera image, our email client knows how to separate between spam and other mail and learns from its mistakes, our cell phones can automatically detect which song is playing on the background during a noisy evening in a night club, and so on.

For a human, a specific pattern recognition task may seem trivial. However, getting a machine to repeat the same thing can be extremely difficult. Several decades of scientific research and effort have been put in fields such as artificial intelligence, machine learning, and statistical pattern recognition in order to come up with computational models that can make decisions similar to what humans can. In this thesis, this same line of research is continued on a specific area of *probabilistic classification*, i.e., automatic determination of the probability of a particular event occurring (such as *does the patient have cancer or not*) given some set of input data (such as the patient's blood sample). Depending whether there are two or more possible outcomes or *classes*, the classification problem can be either *binary* or *multiclass*. In

some application areas such as document classification, the classified instances may belong to several classes at the same time (multi-label classification [1]). In this thesis, we focus on single-label problems only.

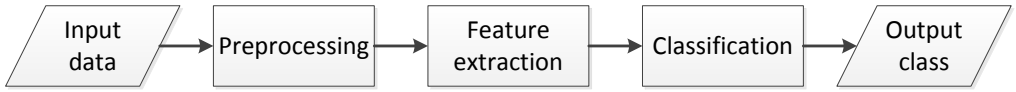


Figure 1.1: General processing pipeline in a classification problem.

Figure 1.1 shows a general processing pipeline of a computational model, i.e., a *classifier* that tries to figure out the type of the output class given some input data. Following the example in the book by Duda et al. [2], the input data can for example be an image of a fish on a conveyor belt while the output would tell whether the fish in the image is a salmon or a sea bass. The three main stages in the general classification pipeline are given below.

- **Preprocessing.** Process the given input data such that it is suitable for further usage. In the fish example, this could mean filtering the image in order to reduce noise and adjusting the brightness and contrast of the image in order to take account changes in the imaging environment.
- **Feature extraction.** Further process the input data in order to derive a set of *features* suitable for recognition of the target classes. For recognizing a salmon from a sea bass, the features are extracted from the camera image by means of automatic image analysis. They can include, e.g., the length and brightness of the fish and the number and shape of the fins it has.
- **Classification.** Use a computational model to map the set of features into a decision about the class. There are a vast number of different methods for making the classification rule. One of these is logistic regression, which is the topic in this thesis.

In this thesis, the focus is on the last part of the above pipeline, i.e., in classification. Specifically, we are studying a probabilistic classification model called *logistic regression*. The structure of the rest of this chapter is such that, in Section 1.1, a brief introduction to machine learning and pattern recognition, especially to the concepts of *supervised learning* and *overfitting*, is given. The basics of linear classification including logistic regression are reviewed in Section 1.2. A proper

mathematical foundation and methods for parameter estimation, i.e., *training the model*, are given later in the thesis. Finally, in Sections 1.3 and 1.4, the scientific objectives of the thesis and the outline into the structure of the rest of the thesis are given, respectively.

1.1 Machine Learning and Pattern Recognition

Machine learning can be categorized as a subfield of *artificial intelligence* that focuses on developing intelligent machines and software. The core of machine learning is in developing and studying software algorithms and models that *enable computers to learn through experience without explicit programming* as defined in 1959 by Arthur Samuel, the developer of a checkers playing game declared as the world's first self-learning computer program [3].

Pattern recognition methods, especially the *supervised learning* algorithms, play an essential part in machine learning. The aim in supervised learning is to find a mapping F between the input features organized as a vector $\mathrm{x} \in \mathbb{R}^d$ and the output, which can either be continuous $y \in \mathbb{R}$ (prediction or regression problems) or categorical $c \in \mathbb{Z}^+$ (classification problems) [4]. The application cases in Chapter 4 of this thesis are all classification problems, i.e, the output of the classification model is a positive integer denoting the *class label* of the classified input feature vector.

In a supervised classification problem, the parameters of the classification model are learned from a set of N training samples consisting of feature vectors x_i and the corresponding class labels c_i, $i = 1, \ldots, N$. The most important issue in the training process is that the resulting classifier should be able to perform well in classifying the samples but also *generalize* on unseen data. It is easy to achieve a good classification performance on the training data without the classifier being able to generalize on new data. This phenomenon is called *overfitting* and it is a common problem in supervised learning. Generally speaking, several sources of uncertainty make it extremely hard to design a good classifier and make critical judgements about different classification methods compared to others. Even the state-of-the-art scientific results can become misleading if care is not taken in conducting new research on the field [5].

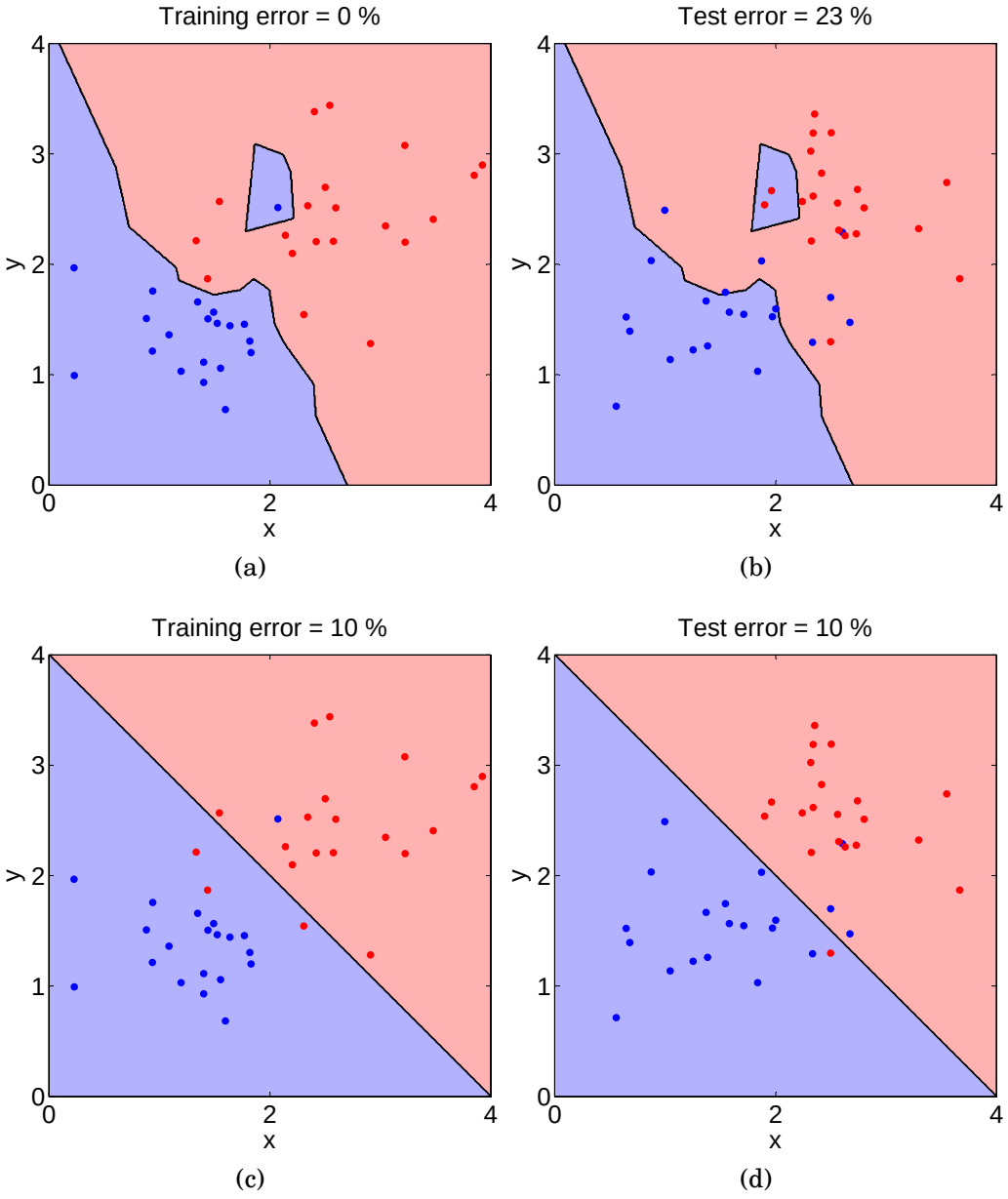


Figure 1.2: An example of overfitting a set of training data that has binary class labels. A one-nearest-neighbor classifier perfectly classifies the training data (a) while giving significantly higher error rates with independent test data (b) when compared to the Bayes optimal classifier (c,d).

Overfitting has been demonstrated in Figure 1.2 by using a toy example. Blue and red dots represent some 2-d training data that has been randomly generated from two Gaussian distributions with different means. The background color shows the corresponding decision areas as learned by a one-nearest-neighbor classifier (Figure 1.2a) or as given by an optimal Bayes classifier knowing the underlying data distributions (Figure 1.2c). The nearest neighbor classifier always results in a perfect training performance. However, bad generalizability can be expected, as noticed when comparing against the optimal classifier. Indeed, by drawing new samples independent of the training samples, the misclassification error of the nearest neighbor classifier gets significantly higher compared to the optimal classifier (Figures 1.2b and 1.2d).

1.2 Introduction to Linear Classification

One of the simplest classification models is the linear model, which assigns a binary class label $\hat{c} \in \{1, 2\}$ to the classified sample according to a score value given by the linear combination of the set of d features $\mathbf{x} \in \mathbb{R}^d$ and a bias term such that

$$\hat{c} = \begin{cases} 1, & \text{if} \quad \beta_0 + \boldsymbol{\beta}^T \mathbf{x} < 0 \\ 2, & \text{otherwise} \end{cases}, \tag{1.1}$$

where $\boldsymbol{\theta} = \left(\beta_0, \boldsymbol{\beta}^T\right)^T$ are the model parameters [2, Chap. 5]. The linear model is easily extended to the multiclass case as shown later in Section 2.2.

Linear models are convenient because they are easy to interpret and their behaviour is widely studied. Linear models have planar decision boundaries like the line shown in the 2-d binary classification case in Figure 1.2c. In multiclass cases, the decision boundaries are piecewise planar honeycomb-like structures.

Popular linear classification methods include *linear discriminant analysis* (LDA) [6], *support vector machines* (SVM) [7] with linear kernels, and, the topic of this thesis, *logistic regression*. While all these methods share the same linear classification model in Equation 1.1, they are different in how the model parameters $\boldsymbol{\theta}$ are learned from the training data. In addition, there can be a difference in how the continuous score value $\beta_0 + \boldsymbol{\beta}^T \mathbf{x}$ is interpreted. In LDA, for example, the model parameters are learned by assuming the data to be normally

distributed and then maximizing the class separation, while, in SVM, decision boundaries are formed by maximizing the margin between the classes. In logistic regression, the parameters are estimated such that the posterior probability of the model given the training data is maximized. When comparing logistic regression and LDA, logistic regression makes fewer assumptions about the features and is considered more robust. The topic has been considered, e.g., by Press and Wilson [8] in the 70's. There is also a section about choosing between logistic regression and LDA in Hastie's book [9, Sec. 4.4.5].

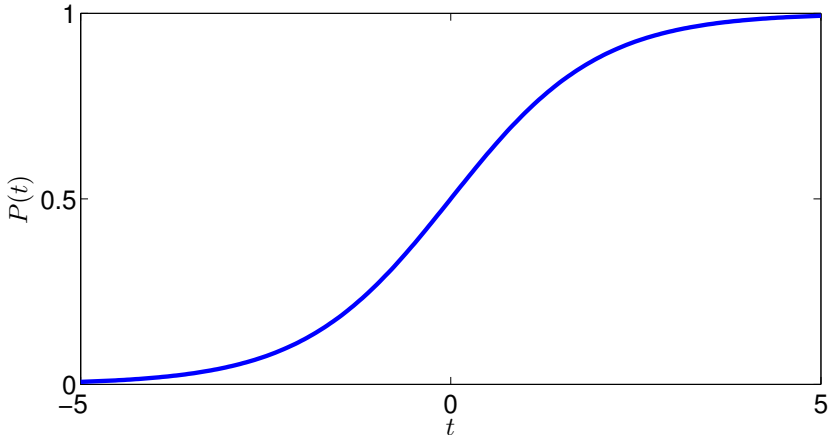


Figure 1.3: The logistic function.

As the first step towards understanding the basics of the logistic regression classifier, let's start by reviewing the *logistic function*

$$P(t) = \frac{1}{1 + \exp(-t)} \tag{1.2}$$

and its graph in Figure 1.3. The idea of the logistic regression model is to use the logistic function to map the linear combination of the set of d features $\mathbf{x} \in \mathbb{R}^d$ into an estimate of the probability that $\mathbf{x}$ belongs to the c^{th} class. Thus, logistic regression is a form of probabilistic classification. This has been illustrated by the diagram in Figure 1.4.

The logistic regression model is a special case of a *feedforward neural network* [10] with a single neuron. The lack of hidden neuron layers in the network makes logistic regression a linear classifier unlike neural networks in general. If seen as an extension of linear regression with a logistic link function, logistic regression belongs to the family of *generalized linear models* (GLM) [11]. The logistic function allows the

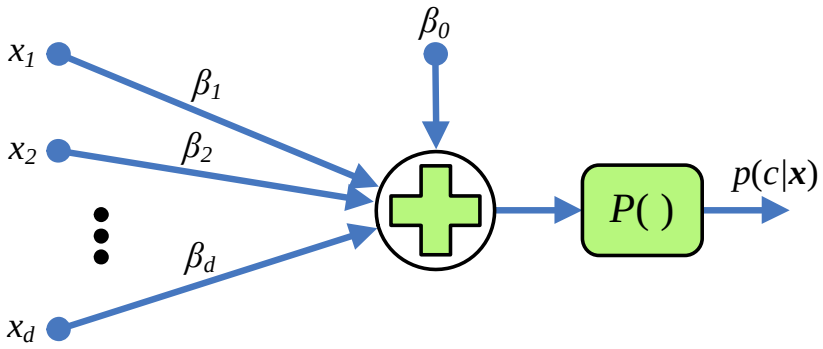


Figure 1.4: The two-class logistic regression model is a generalized linear model with a logistic link function P. It is also equivalent with a feedforward neural network with a single neuron.

modeling of binomially distributed response variables instead of Gaussians like in ordinary linear regression. Other link functions in place of the logistic function can be used for modeling different response distributions, e.g., exponential or Poisson distributions.

From the viewpoint of probabilistic classification, logistic regression is a form of *probabilistic discriminative modeling* [12], where the model directly estimates the probability $p(c|\mathbf{x})$[1] of a certain class c given the data $\mathbf{x}$. In an alternative method of *generative modeling* [12], both class-conditional probability distributions $p(\mathbf{x}|c)$ and the priors $p(c)$ are modeled separately. After this, Bayes' theorem is applied in order to find the posterior class probabilities for making the decision:

$$p(c|\mathbf{x}) \propto p(c)p(\mathbf{x}|c). \tag{1.3}$$

A typical example of a generative model is the *naïve Bayes classifier*, which assumes the features to be *conditionally independent* given the class labels. In this case, the Bayes' rule can be written as

$$p(c|\mathbf{x}) \propto p(c) \prod_{i=1}^{d} p(x_i|c). \tag{1.4}$$

The benefit compared to Equation 1.3 is that the class conditional probability distributions are now one-dimensional and can easily be modeled, e.g., by fitting Gaussian distributions. Common ways to model the

[1] In this thesis, notation $p(c|\mathbf{x})$ is used interchangeably, depending on the context, with notation $P(C = c \,|\, \mathbf{X} = \mathbf{x})$ for denoting the conditional probability of the realization c of the random variable C given that the random variable $\mathbf{X}$ has value $\mathbf{x}$.

prior distribution $p(c)$ is to assume uniform class probabilities or to use the training data to estimate the ratio between different class labels.

It is a widely discussed topic whether discriminative or generative models should be preferred in general [13, 14, 15, 16, 17, 12]. Generative models are versatile and can be used, e.g., for sampling of new $(\mathbf{x}, c)$ pairs [12]. Also, because generative models model the complete joint distribution, handling partially missing data is more intuitive compared to discriminative models. In a typical classification problem, however, discriminative models are generally considered more practical over the generative models. Typically, there are fewer parameters to estimate and model errors become less significant [4, Sec. 4.3]. In most practical applications of supervised learning such as those in Chapter 4, features like sampling are not needed and only the classification performance matters. In these applications, discriminative models such as logistic regression are justified and likely to give better results compared to generative models.

1.3 Objectives

The objective of this thesis is to show the versatility and practicality as well as study the limitations of a simple linear logistic regression model combined with sparsity promoting coefficient regularization that works as an automated feature selector embedded in the parameter estimation procedure. This is done in an application oriented manner in real machine learning applications.

In this thesis, the simplicity of the linear model is emphasized over more complex classifiers. The usefulness of the interpretability of the linear model coefficients is stressed in several application cases. Further, a simple model structure helps to avoid overfitting, which is an extremely important aspect to be taken into account when well performing and generalizable classifiers are desired, which is usually the case. Another important objective of this thesis is to discuss overfitting avoidance by means of proper model validation and automatic parameter selection in Section 3.2.

Specific claims in this thesis include:

- A comprehensively engineered initial feature set combined with automatic feature selection and a linear classification model outperforms other methods in several selected application fields involving supervised classification.

- Logistic regression with sparsity promoting coefficient estimation can be used for combining feature selection and classifier training. The benefit is that the feature set gets optimized for the specific model resulting in improved performance compared to other state-of-the-art methods in several selected application fields.
- Combining feature selection and coefficient estimation also reduces the computational load because no explicit feature selection algorithms are required. This improvement can be crucial in applications requiring short response times.
- Due to the simplicity of the model, the model coefficients are easily interpretable bringing extra knowledge, which is useful in several selected application cases.
- Iteratively improving a classification model by means of subsequent cross-validation may lead to overly optimistic results and should be avoided when possible.

Support for the above claims are given by means of the results obtained in the application cases of Section 4. Majority of these results are based on those presented in peer-reviewed publications listed in Section 1.5. The last item in the above list is related to parameter selection but can also be consider in a more general context as discussed in Section 3.2.

1.4 Outline

The rest of this thesis is organized as follows: First, in Chapter 2, the logistic regression model and the basics of sparse coefficient estimation are reviewed. Next, Chapter 3 focuses on how to measure the performance of a pre-trained classification model (Section 3.1) and how to do model validation, parameter selection, and to avoid overfitting (Section 3.2). The most of the scientific results of this thesis are presented in Chapter 4 that introduces and gives the results on applying the sparse logistic regression classifier in several different application fields. Finally, in Chapter 5, some concluding remarks are given.

1.5 Publications and Author's Contribution

Most of the results in this thesis are based on the author's first-author publications [18, 19] and the co-authored publications [20, 21, 22, 23]. No other dissertation is based on these publications. Author's contribution to each of the publications is the following.

In [18], the author is the responsible author for the design, implementation, and validation of the prediction model as well as running the experiments. Prof. Matti Nykter gave insights into the application field and provided text for the introductory part. Assoc. Prof. Heikki Huttunen provided parts of the text, especially in the discussion part, and participated in the design of the experiments. Dr. Tech. Pekka Ruusuvuori contributed in the preprocessing of the data, conducted the gating experiments, and provided parts of the text in the material section. The results related to [18] are presented in Section 4.3.

The author is the main contributor of the design, implementation, experimentation, and writing of [19], which provides the background and the baseline method for object detection in inkjet printed electronics manufacturing covered in Section 4.2. The other contributors are Ville Pekkanen who was responsible for operating the printer in the practical experiments, Kalle Rutanen and Pekka Ruusuvuori who participated in the design of the algorithms, Risto Rönkkä who participated in the design of the experiements and provided text for the introductory part, and Heikki Huttunen who participated both in the design of algorithms and experiments, especially by providing the general idea of the connection pad detection pipeline.

A new approach for object detection involving logistic regression segmentation is given in [20], where the author has contributed in the design, implementation, and data collection of the experiments as well as in providing text for the logistic regression classification part. In addition to the printed electronics case, the image segmentation framework introduced in [20] is utilized in the cell segmentation case in Section 4.1.2.

In [21] and [22], the author has provided parts of the text and algorithm implementation, and contributed into the model validation and design of experiments. Majority of the scientific input has been provided by Heikki Huttunen, while Jukka-Pekka Kauppi and Jussi Tohka have mainly contributed into the issues and text concerning the application field. In [22], the author has designed and implemented

the experiments for illustration of the wrong and right ways to apply cross-validation as also discussed in Section 3.2.4. The actual application case of [22] is given in Section 4.4.

Finally, in [23], the author was responsible for producing part of the result figures for the experiments. In addition, he participated in the design of experiments. The topic of using Bayesian error estimation in selection of the logistic regression model and the related results are discussed in Section 3.2.3.

Chapter 2

Logistic Regression Classification and Sparse Parameter Estimation

In the following sections, first, a brief introduction into some historical aspects of logistic regression classification is given in Section 2.1. Next, a definition of the logistic regression model is given for both binary and multiclass cases in Section 2.2. Finally, in Section 2.3, methods for parameter estimation, i.e., ways for training the model by using training data, are given. The theory of logistic regression classification is more extensively presented, e.g., in the book by Hastie et al. [9, Sec. 4.4].

2.1 Background

Logistic regression is a well established classification method taught to students on a basic university statistics course. The logistic function given in Equation 1.2 in Section 1.2 dates back to the 19^{th} century when a Belgian mathematician Pierre François Verhulst first used it for modeling the growth of human population after realizing that the conventional exponential model would eventually lead into impossibly large values [24].

Since its first introduction, the logistic function has been used in several applications related to exponential growth limited, e.g., by the amount of available resources. Pioneering studies using the logistic function include modeling of the population growth, modeling product concentrations in autocatalytic chemical reactions, and modeling death

rate as a function of drug dosage. It has since been used for various tasks in economics, epidemiology, and social sciences, for instance. [24]

The employment of the logistic function in applications of logistic regression classification emerged during the advent of the computer era in the 70's. Similar to that with the logistic function, pioneering applications came from fields such as medical, economics, and social sciences. Daniel McFadden earned the Nobel Prize in Economical Sciences in 2000 from his work in developing the theory of discrete choice modeling. A great deal of the theory in choice modeling owes to the multinomial logistic regression model. [24]

More recent advances in logistic regression classification are related to *coefficient regularization* [25, 26, 27, 28]. Via regularization, one can handle issues often present in practical machine learning problems such as small sample size, high sample dimensionality, and redundancy between features. Especially *sparsity promoting regularization* has proven useful in practical machine learning applications, which makes it an interesting tool in the context of this thesis. With "sparse" we mean some of the parameter estimates being equal to zero. This results in a logistic regression model, where only a subset of the initial features are chosen into the model. Such an elegantly combined feature selection and classification frees us from using traditional explicit feature selection methods that are often time consuming and do not necessarily result in good predictive power.

2.2 Logistic Regression Model

Like mentioned in Section 1.2, logistic regression uses the logistic function in Equation 1.2 to model the probability of the occurrence of an event. In a binary classification problem, the probability of class one given the d-dimensional feature vector $\mathbf{x} \in \mathbb{R}^d$ is modeled as

$$p(C = 1 \,|\, X = \mathbf{x}) = \frac{1}{1 + \exp\left(-\left(\beta_0 + \boldsymbol{\beta}^T \mathbf{x}\right)\right)} = \frac{\exp\left(\beta_0 + \boldsymbol{\beta}^T \mathbf{x}\right)}{1 + \exp\left(\beta_0 + \boldsymbol{\beta}^T \mathbf{x}\right)}, \quad (2.1)$$

where $\beta_0 \in \mathbb{R}$ and $\boldsymbol{\beta} \in \mathbb{R}^d$ are the model parameters, collectively denoted by the parameter vector $\boldsymbol{\theta} = \left(\beta_0, \boldsymbol{\beta}^T\right)^T$.

The generalization of the logistic regression model into multinomial case for classes $c = 1, \ldots, K$ is achieved by modeling the probability of

each class separately as

$$p(c \,|\, \mathbf{x}) = \frac{\exp\left(\beta_{c0} + \boldsymbol{\beta}_c^T \mathbf{x}\right)}{1 + \sum_{k=1}^{K-1} \exp\left(\beta_{k0} + \boldsymbol{\beta}_k^T \mathbf{x}\right)}, \quad c = 1, \ldots, K-1,$$
$$p(K \,|\, \mathbf{x}) = \frac{1}{1 + \sum_{k=1}^{K-1} \exp\left(\beta_{k0} + \boldsymbol{\beta}_k^T \mathbf{x}\right)}, \tag{2.2}$$

where $p(c \,|\, \mathbf{x})$ is a short hand notation for $p(C = c \,|\, X = \mathbf{x})$[1]. There are now $(K-1)(d+1)$ model parameters, which we denote in a single vector as $\boldsymbol{\theta} = \left(\beta_{10}, \boldsymbol{\beta}_1^T, \ldots, \beta_{(K-1)0}, \boldsymbol{\beta}_{K-1}^T\right)^T$. Notice that the denominator in the model is chosen such that the probabilities $p(c \,|\, \mathbf{x})$, $c = 1, \ldots, K$, sum up to one.

Equation 2.2 gives the traditional way of defining the multinomial logistic regression model. An alternative and slightly more convenient definition is given by the *symmetric* model that models the class probabilities as

$$p(c \,|\, \mathbf{x}) = \frac{\exp\left(\beta_{c0} + \boldsymbol{\beta}_c^T \mathbf{x}\right)}{\sum_{k=1}^{K} \exp\left(\beta_{k0} + \boldsymbol{\beta}_k^T \mathbf{x}\right)}, \quad c = 1, \ldots, K. \tag{2.3}$$

The parameter vector is now of the form $\boldsymbol{\theta} = \left(\beta_{10}, \boldsymbol{\beta}_1^T, \ldots, \beta_{K0}, \boldsymbol{\beta}_K^T\right)^T$, i.e., there are $d + 1$ parameters more than in the traditional model. This makes the symmetric model ambiguous. Indeed, adding a constant value to the bias terms β_{k0} does not change the model. It turns out, however, that the redundancy will not be a problem when using constrained optimization, i.e., *regularization*, in estimating the model parameters. Thus, the symmetric model is used in the experiments of this thesis unless otherwise stated.

Regardless of whether the traditional or symmetric model is used, the predicted class $\hat{c} = 1, \ldots, K$ for sample $\mathbf{x}$ is given by the maximum of the class probabilities, i.e.,

$$\hat{c} = \underset{c}{\operatorname{argmax}}\, p(c \,|\, \mathbf{x}). \tag{2.4}$$

The above maximization problem is equivalently written as

$$\hat{c} = \underset{c}{\operatorname{argmax}} \log\left(p(c \,|\, \mathbf{x})\right) = \underset{c}{\operatorname{argmax}}\, g_c(\mathbf{x}), \tag{2.5}$$

i.e., as a maximization problem of the discriminant functions $g_c(\mathbf{x})$, which, in the case of the traditional model, are defined as

$$g_c(\mathbf{x}) = \begin{cases} \beta_{c0} + \boldsymbol{\beta}_c^T \mathbf{x} & , c \neq K \\ 0 & , c = K \end{cases} \tag{2.6}$$

[1] We use similar notation where useful in the future

and in the case of the symmetric model as

$$g_c(\mathbf{x}) = \beta_{c0} + \boldsymbol{\beta}_c^T \mathbf{x}. \tag{2.7}$$

In other words, logistic regression has "linear" discriminant functions $g_c(\mathbf{x})$ and is, thus, a linear classifier. [29, p. 161]

2.3 Learning the Model Coefficients

As discussed earlier in Section 1.2, the difference between different linear classifiers is in how the model coefficients $\boldsymbol{\theta}$ are estimated. The central paradigm in supervised learning is to use a set of training data $\mathcal{D} = \{\mathbf{x}_i, c_i\}_{i=1}^N$ having N example feature vectors and the corresponding class labels, which are known beforehand.

There are several different ways how to estimate the parameters of the model from the training data. With the logistic regression model, the conventional way is to use the *maximum likelihood* method, which maximizes the likelihood of the model producing the labels in the training data given the feature vectors. The theoretical core of this thesis relies on *sparsity promoting parameter estimation via regularization*, where the learning procedure is combined with a regularization that limits the search space of the coefficient optimization problem simultaneously preferring solutions where many of the estimated coefficient values become exactly zero. Mathematically convenient approach for sparse coefficient estimation can be established with a Bayesian approach where the model coefficients are assumed to be random variables. Different types of prior distributions then produce different types of sparse solutions.

The structure of the forthcoming sections is the following. In Section 2.3.1, the *maximum likelihood* method is introduced. Applying regularization via *penalized maximum likelihood* is discussed in Section 2.3.2 and, further, from the aspect of Bayesian parameter estimation, in Section 2.3.3. Basics of the Bayesian approach have also been covered, e.g., in the book by Bishop [4, Sec. 4.5].

2.3.1 Maximum Likelihood Method

In logistic regression, the parameter estimation is usually done with *maximum likelihood* (ML) that maximizes the likelihood of the training

data $\mathcal{D} = \{\mathbf{x}_i, c_i\}_{i=1}^N$, i.e., by maximizing the likelihood function

$$L(\boldsymbol{\theta}; \mathcal{D}) = \prod_{i=1}^{N} p(c_i|\mathbf{x}_i). \tag{2.8}$$

Equivalently, one can maximize the log-likelihood function

$$l(\boldsymbol{\theta}) = \log L(\boldsymbol{\theta}; \mathcal{D}) = \sum_{i=1}^{N} \log p(c_i|\mathbf{x}_i). \tag{2.9}$$

The log-likelihood function in Equation 2.9 behaves nicely from optimization point of view because it is concave and has analytical first and second derivatives. Thus, a standard Newton-Raphson method can be used for iteratively finding $\boldsymbol{\theta}$ that maximizes $l(\boldsymbol{\theta})$. In the Newton-Raphson method, a new $\boldsymbol{\theta}^{(k+1)}$ is found by updating the previous solution $\boldsymbol{\theta}^{(k)}$ as

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + H^{-1}\left(\boldsymbol{\theta}^{(k)}\right) \nabla l\left(\boldsymbol{\theta}^{(k)}\right). \tag{2.10}$$

Above, $\nabla l = (\partial l/\partial\theta_1, \partial l/\partial\theta_2, \ldots)^T$ is the gradient function and H is the Hessian, i.e., a matrix with elements $H_{ij} = \partial^2 l/\partial\theta_i\partial\theta_j$.

Inverting the Hessian matrix H in Equation 2.10 can be computationally expensive. *Quasi-Newton* methods can be used for directly approximating the inverse of the Hessian in an iterative manner. One of the most used and efficient quasi-Newton algorithms uses the *BFGS* (Broyden-Fletcher-Goldfarb-Shanno) update or its limited memory version L-BFGS. [30]

Another approach is to approximate the Hessian matrix by its expectation. This allows the reformulation of the maximum likelihood problem into a weighted least squares problem, where the weights depend on $\boldsymbol{\theta}$ (see details from [31]). This type of problem is then efficiently solved by using the *iteratively reweighted least squares (IRLS)* algorithm [32], which in each iteration solves a weighted least squares problem and then updates the weights for the next iteration accordingly. A downside in IRLS is that it requires plenty of training data in order for the approximation of the Hessian to be accurate. As a rule of thumb, the training data should contain at least ten times more samples per class compared to the number features [33].

2.3.2 Regularization

In many real life classification problems, we have an arbitrary set of initial features. Some of the features can be highly redundant and

some may not include any relevant information from the viewpoint of the classification task. In such a case, the ML problem may become ill-posed. Another problem occurs when the training data is linearly separable. In this case, the log-likelihood in Equation 2.9 saturates and any separating hyperplane will do for the ML [4, p. 206]. This inevitably decreases the classifier's ability to generalize.

A traditional solution for the feature redundancy problem is to introduce a feature selection or dimension reduction step that tries to only take account the relevant features or make the features uncorrelated or independent. However, a more sophisticated way to go is to use *sparsity promoting parameter regularization*. This kind of regularization works as an embedded feature selector simultaneously handling the saturation problem in the case of linearly separable data.

For simplicity, let's consider the binary classification problem where the parameter vector of the logistic regression classifier is of the form $\boldsymbol{\theta} = (\beta_0, \boldsymbol{\beta}^T)^T$. The simplest sparsity promoting regularization technique defines a maximum value $t \geq 0$ allowed for $||\boldsymbol{\beta}||_1 = \sum_{k=1}^{d} |\beta_k|$, i.e., the ℓ_1 norm of the coefficient vector $\boldsymbol{\beta}$:

$$\max_{\boldsymbol{\theta}} \{l(\boldsymbol{\theta}; \mathcal{D})\}, \quad \text{s.t. } ||\boldsymbol{\beta}||_1 \leq t. \tag{2.11}$$

Equivalently, this can be formulated in the *Lagrangian form* as a *penalized maximum likelihood* (PML) problem [9, sec. 3.4.2]:

$$\max_{\boldsymbol{\theta}} \{l(\boldsymbol{\theta}) - \lambda ||\boldsymbol{\beta}||_1\}, \tag{2.12}$$

where $\lambda \geq 0$ is the *regularization parameter* having a one-to-one relationship with the previously used t. The additional penalty term proportional to λ is also known as the *LASSO (Least Absolute Shrinkage and Selection Operator) penalty*, originally developed for use in linear regression [34] and also known as *basis pursuit* [35] in signal processing literature. Note that in multi-class case, the more convenient symmetric logistic regression model in Equation 2.3 can be used instead of the traditional one as regularization solves the model ambiguity in a natural way [26].

The sparsity property of the LASSO penalty is visualized in Figure 2.1 by comparing against the more traditional *ridge* penalty [36] that uses squared ℓ_2 norm $\left(||\boldsymbol{\beta}||^2 = \sum_{k=1}^{d} \beta_k^2\right)$ in place of the ℓ_1 norm in Equation 2.12:

$$\max_{\boldsymbol{\theta}} \left\{l(\boldsymbol{\theta}) - \lambda ||\boldsymbol{\beta}||^2\right\}. \tag{2.13}$$

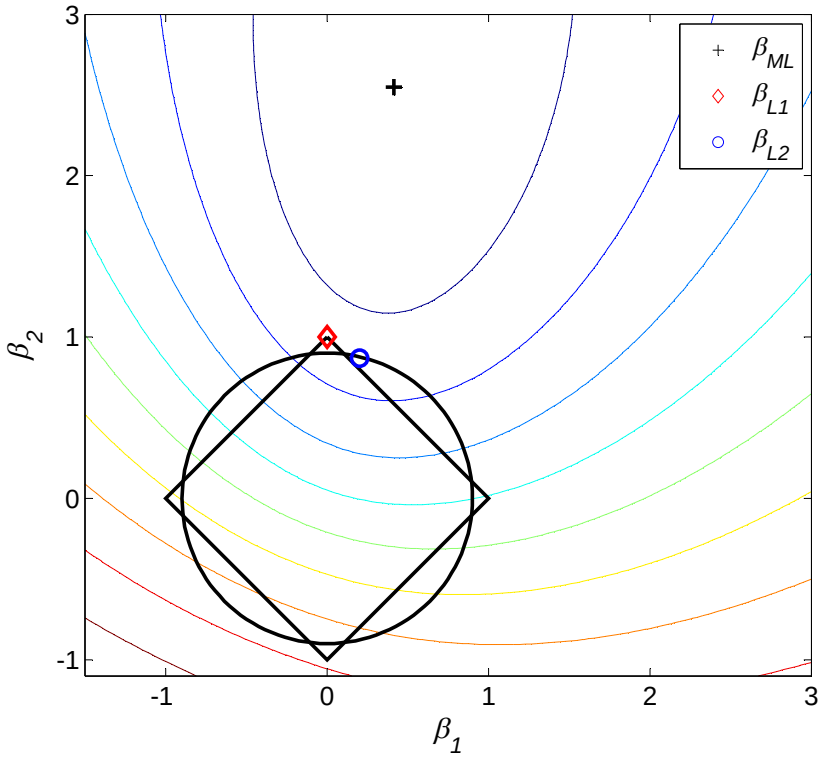


Figure 2.1: The contour lines show an example of the log-likelihood function to be maximized. Regularization sets constraints to the search space. In ℓ_2 regularization the searched area is circular, in ℓ_1 regularization it is diamond shaped.

The contour lines in the image show an example of the log-likelihood function with respect to model parameters β_1 and β_2 in a 2-d case. The ML solution β_{ML} has been marked with the black plus sign. Regularization sets constraints into the parameter space, which are shown as a circular area in the case of ℓ_2 regularization and as a diamond shaped area in the case of ℓ_1 regularization. The constrained maximization problem is to find the optimum inside these areas. The diamond shaped area of the ℓ_1 penalty favors solutions that reside on one or more of the coordinate axes. This makes some of the coefficient values equal to zero. Increasing the value of the regularization parameter λ makes the diamond smaller and increases the number of zero coefficients. In many real life problems, the feature selection property of the LASSO penalty results in better classifier generalizability over the ridge penalty, especially when irrelevant features are present [37].

Sometimes it is desirable to adjust between the ℓ_2 regularization that averages the features by jointly bringing the model coefficients towards zero and the ℓ_1 regularization that has the feature selection property. An efficient and practical way for achieving this is to use *elastic net* regularization [28], which optimizes the log-likelihood as

$$\max_{\boldsymbol{\theta}} \left\{ l(\boldsymbol{\theta}) - \lambda \left(\alpha ||\boldsymbol{\beta}||_1 + (1-\alpha)||\boldsymbol{\beta}||_2^2 \right) \right\}. \tag{2.14}$$

The penalization term is now a convex combination of the ℓ_1 norm and the squared ℓ_2 norm of the coefficient vector $\boldsymbol{\beta}$. The mixing parameter $\alpha \in (0, 1)$ is used for determining the proportions between the different types of regularizations. Setting $\alpha = 1$ produces the LASSO penalty (Equation 2.12) and setting $\alpha = 0$ produces the ridge penalty (Equation 2.13).

Similar to that in the unregularized case, the IRLS algorithm can be used for solving the regularized problems as well. However, simple and resource friendly cyclical coordinate descent methods such as `glmnet` [26] and the *sparse multinomial logistic regression* (SMLR) algorithm [38] have been proven to work fast and practically, especially in large scale problems like text [39] and microarray data classification [40]. Especially the `glmnet` algorithm proposed by Friedman *et al.* [26] is versatile and fast. It can be used in the case of linear regression, binary logistic regression, as well as multiclass logistic regression and works together with ℓ_1, ℓ_2, and elastic net penalties. The algorithm estimates the model coefficients efficiently for complete regularization paths with different λ values, which are needed for automatic selection of λ, e.g., by using cross-validation or BEE (see Section 3.2 and [23]). The computational efficiency of the `glmnet` algorithm is shown to outperform competing methods. Its time complexity in one cycle of updating each coefficient estimate is $\mathcal{O}(dN)$ compared to $\mathcal{O}(d^3 + dN)$ of the IRLS algorithm and the SMLR algorithm if using ℓ_1 penalty. Unless otherwise stated, `glmnet` is used for coefficient estimation in the applications of this thesis.

2.3.3 Bayesian Methods

In the previous section, parameters $\boldsymbol{\theta}$ of the logistic regression model were considered to be fixed but unknown. In *Bayesian framework*, however, the parameters are considered random variables with some prior distribution. In the case of logistic regression, complete Bayesian inference is intractable [4, Sec. 4.5]. However, it is possible to find

the *maximum a posteriori* (MAP) point estimate for $\boldsymbol{\theta}$ and apply, e.g., *Markov Chain Monte Carlo* (MCMC) sampling if a variance estimate is needed [41].

According to the Bayes rule, the posterior probability $p(\boldsymbol{\theta}|\mathcal{D})$ is proportional to the product of the likelihood and the prior, i.e.,

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})} \propto p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta}), \tag{2.15}$$

where $p(\mathcal{D}|\boldsymbol{\theta}) = L(\boldsymbol{\theta}; \mathcal{D})$ is the likelihood of the data (as given in Equation 2.8), $p(\boldsymbol{\theta})$ is the prior distribution of the coefficients $\boldsymbol{\theta}$, and $p(\mathcal{D})$ is the prior distribution of the training data $\mathcal{D}$.

The MAP estimator is achieved by maximizing $p(\boldsymbol{\theta}|\mathcal{D})$. Notice, that $p(\mathcal{D})$ is independent of $\boldsymbol{\theta}$, which makes the MAP equivalent to maximizing $p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})$. Further, by taking a logarithm, one can see that the MAP estimator is equivalent of the PML estimator where the penalization term equals to the negative of the logarithm of the prior $p(\boldsymbol{\theta})$:

$$\arg\max_{\boldsymbol{\theta}} \{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})\} = \arg\max_{\boldsymbol{\theta}} \{l(\boldsymbol{\theta}) + \log p(\boldsymbol{\theta})\}. \tag{2.16}$$

In the following, the key idea is to assume different shapes for the prior distribution $p(\boldsymbol{\theta})$ in order to attain different types of regularization.

First, consider the case where the coefficients β_k ($k = 1, \ldots, d$) are assumed independent and normally distributed with zero mean and equal variance σ^2, i.e.,

$$p(\beta_k) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\beta_k^2}{2\sigma^2}\right) \tag{2.17}$$

$$\Rightarrow p(\boldsymbol{\theta}) = \prod_{k=1}^{d} p(\beta_k) = \frac{1}{\left(2\pi\sigma^2\right)^{d/2}} \exp\left(-\frac{||\boldsymbol{\beta}||^2}{2\sigma^2}\right). \tag{2.18}$$

In many real life problems all the coefficients β_k rarely have equal prior variance because the corresponding features can have rather different scales. In practice, however, this is taken care of by standardizing the features to have zero mean and unit variance.

Now, assuming that σ^2 is known, the logarithm of the prior in Equation 2.18 can be written as

$$\log p(\boldsymbol{\theta}) = -\frac{||\boldsymbol{\beta}||^2}{2\sigma^2} - \frac{d}{2}\log 2\pi\sigma^2. \tag{2.19}$$

Thus, the MAP estimator becomes equal to

$$\arg\max_{\boldsymbol{\theta}} \{l(\boldsymbol{\theta}) + \log p(\boldsymbol{\theta})\} = \arg\max_{\boldsymbol{\theta}} \left\{l(\boldsymbol{\theta}) - \frac{||\boldsymbol{\beta}||^2}{2\sigma^2}\right\}, \tag{2.20}$$

i.e., equivalent to the PML estimator with ridge penalty and the regularization parameter equal to $\lambda = 1/(2\sigma^2)$. In other words, the more we allow the coefficients β_k to vary around zero the less we apply regularization.

Further assume that the variance σ^2 is not fixed but for each variable it is independently distributed according to the exponential distribution, i.e.,

$$p(\sigma^2) = \begin{cases} r\exp\left(-r\sigma^2\right), & \sigma^2 > 0 \\ 0 & \sigma^2 \leq 0 \end{cases}, \tag{2.21}$$

where $r > 0$ is the rate parameter. Integrating out σ^2 reveals that the prior $p(\boldsymbol{\theta})$ is now a multivariate Laplace distribution or a *double exponential*

$$p(\beta_k) = \int_{-\infty}^{\infty} p(\beta_k|\sigma^2)p(\sigma^2)\,\mathrm{d}\sigma^2 = \frac{\lambda}{2}\exp(-\lambda|\beta_k|) \qquad \text{(Laplace t.f.}^2\text{)} \tag{2.22}$$

$$\Rightarrow p(\boldsymbol{\theta}) = \prod_{k=1}^{d} p(\beta_k) = \left(\frac{\lambda}{2}\right)^d \exp(-\lambda||\boldsymbol{\beta}||_1), \tag{2.23}$$

where $\lambda = \sqrt{2r}$ is now the rate parameter [43, 39, 27]. With this prior the corresponding MAP estimator becomes equivalent to the PML estimator with LASSO penalty having the regularization parameter equal to λ. Thus, the Laplace distribution works as a sparsity promoting prior equivalent to the ℓ_1 penalty in PML estimation.

As the next step, let's further assume that, instead of being fixed, also λ is a random variable. Interesting results are obtained if we assume λ to be the same for each β_k and to have a non-informative Jeffreys prior $p(\lambda) \propto 1/\lambda$ [25]. By marginalizing Equation 2.23 over λ, the prior probability $p(\boldsymbol{\theta})$ becomes

$$\begin{aligned} p(\boldsymbol{\theta}) &= \int_{-\infty}^{\infty} p(\boldsymbol{\theta}|\lambda)p(\lambda)\,\mathrm{d}\lambda = \frac{1}{2^d}\int_0^{\infty} \lambda^{d-1}\exp(-\lambda||\boldsymbol{\beta}||_1)\,\mathrm{d}\lambda \\ &= \frac{1}{2^d||\boldsymbol{\beta}||_1^d}\int_0^{\infty} t^{d-1}\exp(-t)\,\mathrm{d}t = \frac{\Gamma(d)}{2^d||\boldsymbol{\beta}||_1^d}, \end{aligned} \tag{2.24}$$

where $\Gamma(x) = \int_0^{\infty} t^{x-1}\exp(-t)\,\mathrm{d}t$ is the gamma function [25]. In the above integration, a change of variables ($\lambda = t/||\boldsymbol{\beta}||_1$, $\mathrm{d}\lambda = \mathrm{d}t/||\boldsymbol{\beta}||_1$) is performed in order to pull out the gamma integral.

[2]The integral in Equation 2.22 can be carried out by noticing that, when choosing $s = \beta_k^2/2$, prior $p(\beta_k)$ is equal to the Laplace transformation $F(s)$ of function $f(t) = -\frac{r}{\sqrt{2\pi}}t^{-\frac{3}{2}}\exp\left(-\frac{r}{t}\right)$. See [42] for details.

As a result, there are no unknown regularization parameters in the posterior probability $p(\boldsymbol{\theta}|\mathcal{D})$ and the MAP estimator becomes

$$\arg\max_{\boldsymbol{\theta}} \{l(\boldsymbol{\theta}) + \log p(\boldsymbol{\theta})\} = \arg\max_{\boldsymbol{\theta}} \{l(\boldsymbol{\theta}) - d \log ||\boldsymbol{\beta}||_1\}\,, \tag{2.25}$$

i.e., equivalent to the PML problem with penalty term $d \log ||\boldsymbol{\beta}||_1$.

From a practical point of view, using the prior in Equation 2.24 results in a computationally fast way for doing sparse coefficient estimation. This is because there is no need for a time consuming cross-validation step normally used for selecting the regularization parameters. In addition, a possible source of selection bias by parameter tuning can be avoided [44]. Cawley and Talbot [25] have shown that, in a large scale gene selection problem, replacing the MAP estimator's Laplace prior with the one in Equation 2.24 reduces their computation time from two days to two minutes while the model generalization performance is almost the same in both cases.

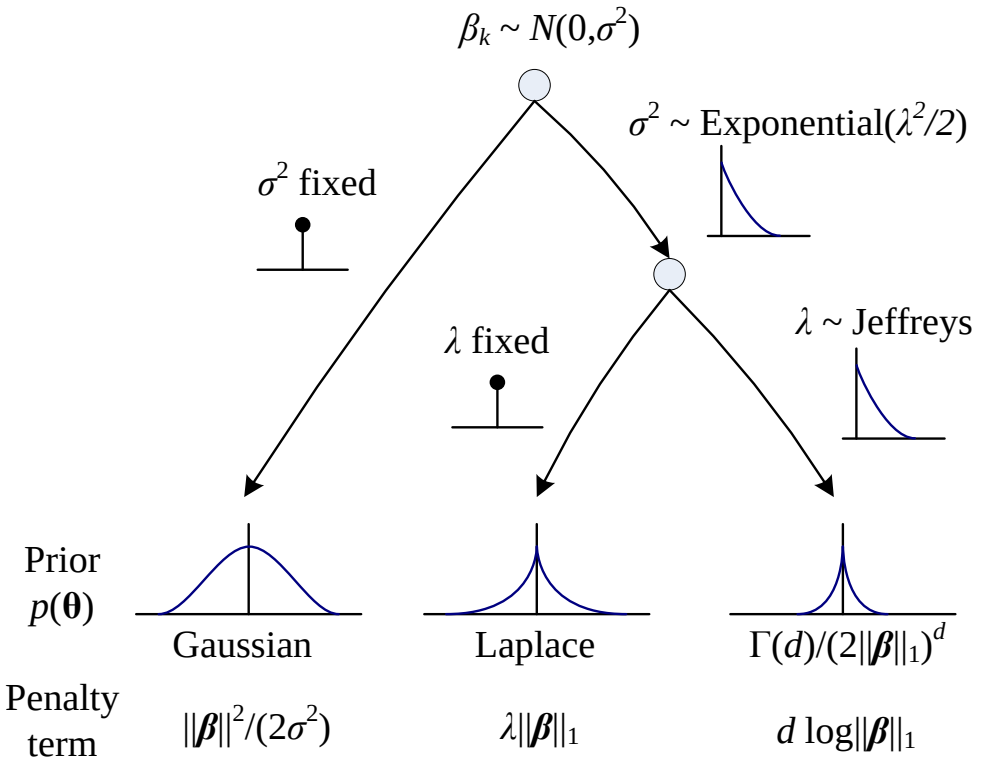


Figure 2.2: The diagram shows a hierarchy leading to three different shapes for the prior probability $p(\boldsymbol{\theta})$ attained by different choice of the priors of its hyper parameters.

As a conclusion for the above introduced Bayesian priors $p(\boldsymbol{\theta})$ and the penalty terms in the corresponding PML problems, see Figure 2.2. Further prior models have been proposed by Caron and Doucet [45] who introduced a general form gamma prior for σ^2 that was shown to give sparse estimates and reproduce both Laplace (Equation 2.23) and Jeffreys (Equation 2.24) priors as special cases. They also showed that, in linear regression, the general gamma prior gave better results than the other sparsity promoting priors. However, two hyper parameters needs to be chosen by cross-validation, which makes the method less practical compared to Laplacian prior (one parameter) and Jeffreys prior (no parameters).

2.4 Markov Random Field Priors

In many applications involving classification, the sample points often share some sort of dependencies, e.g., spatially or temporally. Exploiting these dependencies is tempting and can potentially result in significant improvements in the classification performance.

Graphical models [4, Chap. 8] offer a probabilistic tool for modeling dependencies between random variables and are commonly used in machine learning to model spatial and temporal relationships between the classified samples. Graphical models are characterized by a graph representation where the random variables correspond to nodes that are connected by vertices indicating direct conditional dependencies between the variables. An example of a graphical model is the *hidden Markov model* (HMM), which is often used in speech processing for linking together the subsequent phonemes or words such that their prior distribution follows that of in the spoken language [46].

HMM is an example of a typical type of a graphical model, namely, a *Bayesian network*. In Bayesian networks, the dependencies are directed and acyclic and, as such, are suitable for modeling temporal dependencies where future events cannot affect the past. In image analysis, neighboring pixel intensities have spatial dependencies that can work in any direction. A corresponding way to exploit the class prior knowledge in 2-d is to apply another type of a graphical models, namely *Markov random fields* (MRF). In MRF the dependencies are undirected and can be cyclic. [4, Chap. 8]

The application that we are interested in this thesis is *image segmentation*, i.e., classifying each individual image pixel into either foreground or background. The key idea in using the MRF prior is to as-

sume that the prior class distribution of each pixel follows the Markov principle, i.e., it depends on the classes of the neighboring pixels but only on the nearest ones.

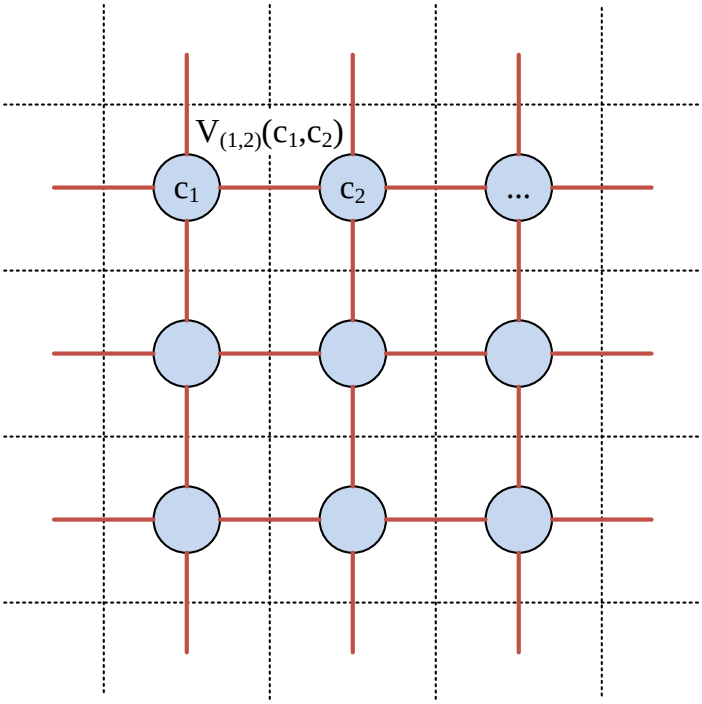


Figure 2.3: Image analysis often uses the Ising model in representing the image. Each pixel corresponds to a graph node $(c_1, \ldots, c_N)$ connected to the neighboring pixels only. The cliques correspond to the graph edges and have clique potentials $V_{(i,j)}(c_i, c_j)$.

Similar to that with graphical models in general, a graph theoretical approach is taken in explaining how MRFs work. In the image segmentation problem, we use a representation where each image pixel forms a node in an undirected cyclic graph such that each neighboring pixel is connected with an edge like shown in Figure 2.3. This is the *Ising model* [47], most common and simplest formulation conventionally used in image analysis [48] originally introduced by Ernst Ising in 1925 for modeling atomic spins in ferromagnetism [49].

Geman and Geman [50], who were the first to propose MRFs for computer vision applications, model the prior configuration of pixel

classes $\mathbf{c} = (c_1, \ldots, c_N)^T$ by using the Gibbs distribution

$$p(\mathbf{c}) \propto \exp\left(-\sum_i V_i(\mathbf{c})\right), \tag{2.26}$$

where i goes through the set of cliques[3] of the graph and $V_i(\mathbf{c})$ are so called clique potentials.

When using the Ising model, the cliques simply correspond to the edges connecting each pixel. In this thesis, a definition of clique potential equivalent to that used by Borges et al. [51] is used. This results in a simplified form $p(\mathbf{c})$ defined as

$$p(\mathbf{c}) \propto \exp\left(\gamma \sum_{(i,j)} \delta(c_i - c_j)\right), \tag{2.27}$$

where (i, j) go through the cliques (each pair of neighboring pixels) and

$$\delta(x) = \begin{cases} 1, & x = 0 \\ 0, & x \neq 0 \end{cases} \tag{2.28}$$

is the unit impulse function. In the Equation 2.27, equal labels c_i and c_j for neighboring pixels clearly increase the value of the prior, thus, creating a smoothing effect by favoring segmentations with a large number of neighboring pixels having the same class label. The amount of smoothing is controlled by using the constant $\gamma > 0$.

Combining the MRF prior in Equation 2.27 with a pixelwise logistic regression classifier has been proposed by Borges et al. [51] and, further, by Ruusuvuori et al. [20]. The problem is that we would like to find the pixel labeling $\hat{\mathbf{c}}$ that maximizes the posterior probability with the MRF prior, i.e.,

$$\hat{\mathbf{c}} = \arg\max_{\mathbf{c}} p(\mathbf{x}|\mathbf{c})p(\mathbf{c}), \tag{2.29}$$

where $p(\mathbf{x}|\mathbf{c})$ is the likelihood of the data with labels $\mathbf{c}$ and $p(\mathbf{c})$ as defined in Equation 2.27. However, instead of the likelihood $p(\mathbf{x}_i|c_i)$, logistic regression estimates posterior probabilities $p(c_i|\mathbf{x}_i)$ for each pixel i. This is resolved by using Bayes formula in the unusual direction [51]:

$$p(\mathbf{x}_i|c_i) = p(c_i|\mathbf{x}_i)p(\mathbf{x}_i)/p(c_i), \tag{2.30}$$

[3] A clique is a subset of the graph nodes where all nodes are directly connected with each other.

or, if further assuming conditional independence and discarding the constant term $p(\mathbf{x}_i)$:

$$p(\mathbf{x}|\mathbf{c}) \propto \prod_i \frac{p(c_i|\mathbf{x}_i)}{p(c_i)}. \tag{2.31}$$

If we further assume equal class probabilities, the denominator can also be omitted. This way we will end up with the definition of the maximum a posteriori (MAP) segmentation:

$$\begin{aligned} \hat{\mathbf{c}} &= \arg\max_{\mathbf{c}} \; p(\mathbf{x}|\mathbf{c})p(\mathbf{c}) \\ &= \arg\max_{\mathbf{c}} \left(\sum_i \log p(c_i|\mathbf{x}_i) + \gamma \sum_{(i,j)} \delta(c_i - c_j) \right). \end{aligned} \tag{2.32}$$

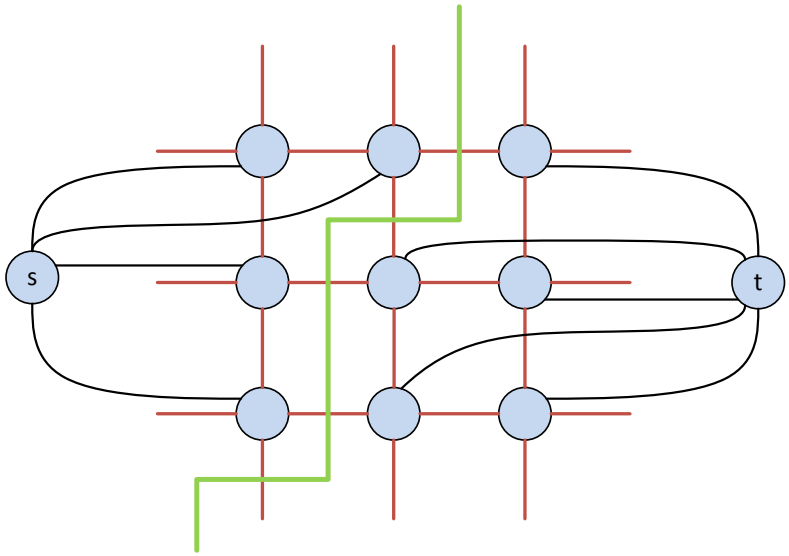


Figure 2.4: Graph cut methods create two extra nodes, source (s) and sink (t), and find the cut of the original graph that minimizes the sum of the weights of the cut edges (green line).

In the original paper by Geman and Geman [50], the maximization problem in Equation 2.32 was solved by using simulated annealing, which was slow and only resulted in approximate solutions. Later, Greig et al. [52] discovered that the MAP segmentation problem is equivalent of the commonly known problem in graph theory, namely the min-cut/max-flow problem for which polynomial time exact solutions exist. These so called *graph cut* methods solve the equivalent

problem of splitting a graph into two disconnected parts such that the foreground and background nodes are in different partitions as illustrated by Figure 2.4. Generally, graph cut algorithms applied for image segmentation using the Ising model have time complexity of $\mathcal{O}(N^3)$, which can be potentially high. However, a fast implementation for graph cuts that can operate near to real-time speed in normal computer vision applications has been proposed by Boykov and Kolmogorov [53]. This algorithm is also used in this thesis.

In this thesis, MRFs are used in a supervised image segmentation setting as a spatial prior in a pixelwise logistic regression classifier combined with automated feature selection by means of ℓ_1 regularization [20]. Graph cuts are used for fast computation of the image segmentation. This type of segmentation framework will be introduced later in Section 4.1 together with experiments in spot detection from cell images (Section 4.1.2) and object detection in inkjet printed electronics manufacturing (Section 4.2).

Chapter 3

Model Selection and Error Estimation

There are two basic cases where it is important to have an idea about the performance of a classification model. First, we would like to know about the classifier's ability to *generalize* on data not seen during the training phase. Second, we often want to *compare* the model with other models in order to improve it or to be able to choose the best one among different models. Comparing classifiers is a bit easier task compared to measuring the absolute generalization performance because only the relative performance needs to be known.

This chapter introduces methods for estimating the model error and selecting the best performing model among many models. First, in Section 3.1, error or performance measures used for measuring the goodness of trained classifiers are defined. Second, in Section 3.2, means for comparing and validating different models with each other are presented.

3.1 Measuring Model Performance

There exists a vast number of different metrics for measuring the performance of a classification model. Both in measuring absolute generalizability and relative performance, it is important to pay attention to the choice of the used error or performance measure such that it corresponds to what one really wants to measure. In addition, many performance measures only work with binary class labels and cannot be used in a multiclass setting. Thus, the choice of the used performance

measure depends mainly on the application and user needs. Next, a set of performance and error metrics are introduced.

3.1.1 Counting Based Performance Measures

The distribution of the different types of classifications made by the classifier can be characterized by using a *confusion matrix*, which is a type of a contingency table as shown by the examples in Figure 3.1. In a binary classification problem, where the classes are often called *positive* and *negative* class, different classification outcomes include *true positive*, *true negative*, *false positive*, and *false negative*, according to what was the predicted class label and what was the true underlying class. As shown by Figure 3.1a, a confusion matrix tabulates the numbers of these classification outcomes denoted by tp, tn, fp, and fn, respectively. Next, a set of performance measures based on tp, tn, fp, and fn are introduced. A broader overview into the subject can be found in [54].

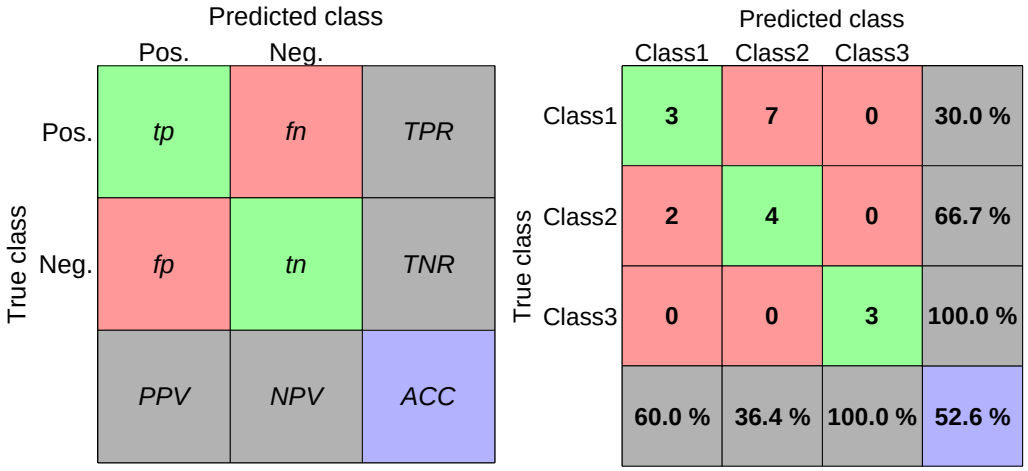


Figure 3.1: Figure (a) shows a prototype confusion matrix and related performance measures calculated from the counts of the different classification outcomes in a binary case. Figure (b) shows an example confusion matrix in a three-class classification case. In this case, *ACC* is shown in the bottom right blue box while the grey cells show the percentage of correct classifications per each row/column.

We start by introducing the simplest and often the most relevant performance measure in a classification task: the percentage of correctly made classifications. We call this the classification performance (or *accuracy* (*ACC*)) and define it simply as the ratio of correct classifications from all classifications:

$$ACC = \frac{tp + tn}{tp + tn + fp + fn}. \tag{3.1}$$

The classification performance is often shown in the bottom right cell of the confusion matrix as seen in Figure 3.1.

While classification performance measures the overall accuracy of the classifier, it loses information about the distribution of true/false positives/negatives. Different performance measures have been developed for measuring different types of errors. Four basic measures are derived from the ratio of correct classifications from each row or column of the confusion matrix. These measures are the *true positive rate* (*TPR*) (or *recall* or *sensitivity*)

$$TPR = \frac{tp}{tp + fn}, \tag{3.2}$$

true negative rate (*TNR*) (or *specificity*)

$$TNR = \frac{tn}{tn + fp}, \tag{3.3}$$

positive predictive value (*PPV*) (or *precision*)

$$PPV = \frac{tp}{tp + fp}, \tag{3.4}$$

and *negative predictive value* (*NPV*)

$$NPV = \frac{tn}{tn + fn}. \tag{3.5}$$

The naming of the above measures varies according to context. For instance, *TPR* and *TNR* are often used together to see the distribution of correctly made positive and negative classifications in which case they are usually referred as sensitivity and specificity. Similarly, when using together *PPV* and *TPR*, names precision and recall are often used.

A popular performance measure called the *F-score* can be derived by calculating the harmonic mean of precision and recall:

$$F = 2\frac{PPV \cdot TPR}{PPV + TPR} = \frac{2tp}{2tp + fn + fp}. \tag{3.6}$$

F-score measures the effectiveness of the classifier in making true positive classifications. It is also possible to use a weighted version of the F-score where the proportions of the contribution of precision and recall can be selected by a choice of weight parameter value. This makes it possible to tune the F-score according to application needs. Notice that F-score doesn't depend on the number of true negatives. This should be noted when using F-score in classification applications.

As shown by the example in Figure 3.1b, the confusion matrix is easily extended into multiclass case. Also the performance measures introduced above can be extended into multiclass by defining tp_c, tn_c, fp_c, and fn_c for each class $c = 1, \ldots, K$ separately in a *one-vs-all* manner. In other words, tp_c is the number of correct classifications into class c, tn_c is the number of correct classifications into other than class c, fp_c is the number of false classifications into class c, and fn_c is the number of false classifications into other than class c. Using these definitions, there are two possibilities to calculate the above performance measures [55, 54, 56, 57]:

1. **Macro-averaging.** Calculate the performance measure for each class separately and then average.

2. **Micro-averaging.** Accumulate tp_c, tn_c, fp_c, and fn_c over each class and calculate the performance measure by using the accumulated values.

The difference in the above two approaches is that macro-averaging discards the effect of the class sizes while micro-averaging takes also account the number of instance in each class by weighting the performance measure in question in a corresponding way.

3.1.2 Order Based Performance Measures

Many classification methods have a parameter that is used for setting a balance between the sensitivity of positive and negative classifications. Increasing the value of this parameter increases the number of positive classifications while decreasing it increases the number of negative classifications. With a classifier with continuous output such as the logistic regression classifier, this balance can be controlled by thresholding the classifier output such that values above the threshold are classified as positives and those under the threshold as negatives. With the logistic regression classifier the threshold should be set to 0.5 because the model is designed to produce class probabilities and

not some arbitrary score values, for example. However, changing the threshold is useful when calculating further performance measures as explained next.

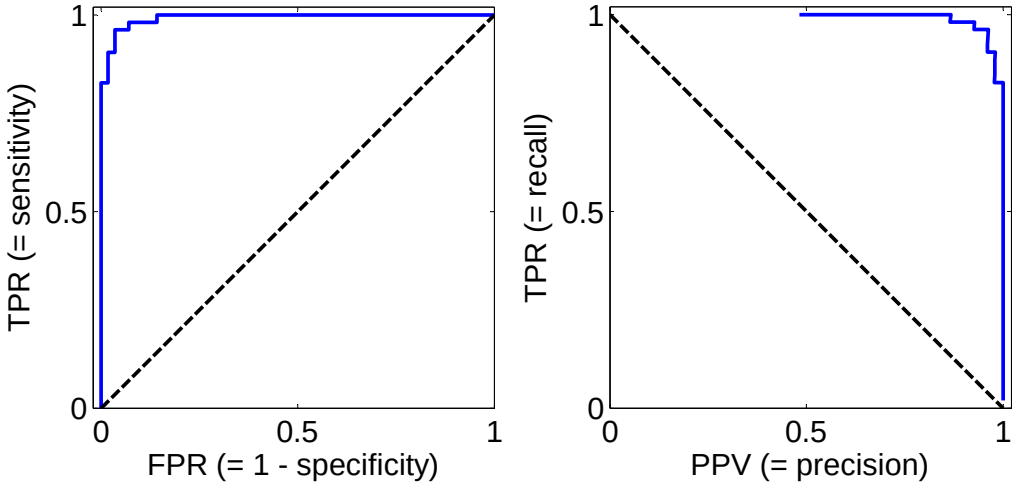


Figure 3.2: Examples of an ROC (a) and a PR (b) curves. Increasing the classifier threshold eventually leads into zero positive classifications making PPV undefined and causing the PR curve to break unlike ROC curve, which always goes from $(0, 0)$ to $(1, 1)$.

A commonly used method for assessing the model performance is to gradually increase the classifier threshold and see what happens to, e.g., sensitivity and specificity. This procedure can be visualized by using a *receiver operating characteristic* (ROC) curve [58, 59] as shown in Figure 3.2a. A popular alternative is to plot precision versus recall (PR) as shown by Figure 3.2b.

In the curves shown in Figure 3.2 the larger the *area under the curve* (AUC) [60] the better. The diagonal dashed line has AUC equal to 0.5 and can be thought to be a result of randomly assigning the class labels. AUC value provides a nice single performance measure similar to correct classification rate or F-score. The AUC value of the ROC curve is equivalent of the probability that a randomly chosen positive gets a higher ranking from the classifier compared to a randomly chosen negative sample [59]. Similarly, AUC of the ROC can be shown to be equivalent to the Mann-Whitney U statistic for the median of the difference between the prediction scores in the two classes [61]. Thus,

AUC values give information about the ranking capability of the classifier without considering what the classifier threshold should be and what is the absolute classification performance.

Similar to that with the performance measures based on the confusion matrix, also AUC can be extended into the multiclass case by either macro- or micro-averaging. Further, methods for computing multidimensional ROCs and the *volume under the surface* (VUS) [62, 63] exist. These are out of the scope of this thesis, however.

3.2 Model Validation and Automatic Parameter Selection

The performance measures introduced in the previous section require the evaluation of a pre-trained classifier by using a set of test data. The simplest method for error estimation is *resubstitution*, where the same data is used both for training and testing [64]. However, such a procedure often gives overly optimistic results that depend on the classification model. Generally speaking, the resubstitution error estimator becomes more optimistic as the classification model gets more complex and the number of training samples decreases [65].

In order to avoid optimistic error estimation results, the test data should be independent of the data used for training the classifier. In many cases, however, the amount of available data is small, e.g., due to expensive or time consuming data collection process. In this case, we would like to use some procedure to reliably assess the model performance without losing data from training. In addition to determining the generalizability of the model, similar methods can be used for automatic selection of model parameters by estimating the relative performance between subsequent models with different parameter values.

Next, some procedures for model validation and automatic parameter selection are given. First, in Section 3.2.1, the popular cross-validation method is introduced. After this, rather a similar approach of bootstrapping is given in Section 3.2.2. While both cross-validation and bootstrapping are classification algorithm independent ways to do error estimation, Section 3.2.3 gives methods targeted for the linear model only. Finally, in Section 3.2.4 we discus the potential optimism that might occur due to misuse of cross-validation and, in Section 3.2.5, we tackle the issue of the effect of the sample size in error estimation.

3.2.1 Cross-validation

The simplest way to assess model performance after resubstitution is to split the available training data set into two parts and use the first one for training and the second one for assessing the model performance. If the amount of data is low, the performance estimates will end up being pessimistic because only part of the data is used for training. On the other hand, we need to have large enough test set in order to get reliable results.

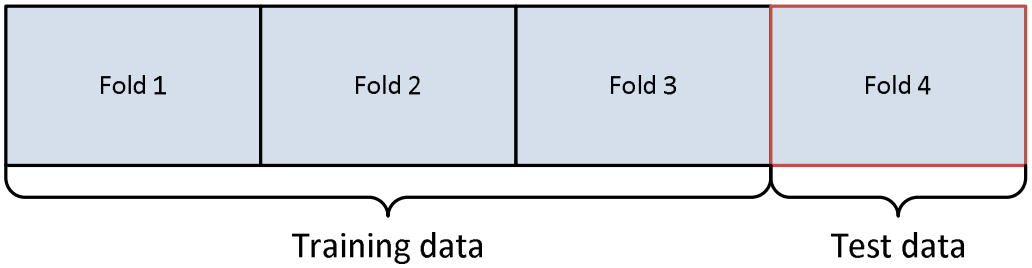


Figure 3.3: K-fold CV splits the whole data into K folds and uses one fold at a time for testing and the rest for training.

The next step from simply splitting the data in two parts is to use *cross-validation* (CV) methods [2, sec. 9.6.2]. In K-fold CV (see Figure 3.3) the data is split into K approximately equal parts, or folds, which are all used as the test set one by one while the remaining $K-1$ folds are used for training. Thus, training and testing is done K times after which the results are combined, e.g., by averaging. A special case occurs when K equals to the number of training samples. This is called the *leave-one-out* (LOO) CV because each sample gets tested one by one while the other ones are used for training.

The split of the data in CV is usually done randomly. Sometimes stratification is used to make the distribution of class labels approximately equal in each fold. This is especially useful in cases where there is a low number of samples available per class. Fixing the seed in the random number generator ensures repeatable results but also restricts us to a particular split of data, which is not necessarily very representative of the underlying data distribution. An alternative is to apply *Monte Carlo* repetitions, i.e., repeat the CV with different splits to reduce the variance of the results.

CV is a simple and popular way for the assessment of model performance. It is also non-parametric in the sense that it doesn't make any assumptions about the tested model and, thus, can easily be used

for comparing models of different types. The downside in CV is that it can easily become computationally expensive because of the repeated training and testing steps. There are also other things one needs to be careful at when doing CV. These are discussed in more detail in Section 3.2.4.

3.2.2 Bootstrapping

Similar to CV, bootstrapping [66, 67] divides the data set of N samples into training and test sets. In bootstrapping, however, the training data is chosen among all the samples by randomly sampling N samples *with replacement* (the bootstrap sample) while the left out samples are used for testing. The procedure is repeated by using independent replicates of the bootstrap sample in order to get a Monte-Carlo estimate of the desired performance or error measure.

Randomly choosing N samples with replacement among N samples results in a training set of size equal to $(1 - \exp(-1))\, N \approx 0.632N$ on the average. Compared to N, this is rather small and easily leads to biased error estimates [64]. This bias can be alleviated by using the 0.632 *bootstrap estimator* [67], which is calculated as the weighted average of the resubstitution estimate $\varepsilon_{\text{resub}}$ and the normal bootstrap estimate ε_0:

$$\varepsilon_{b632} = (1 - 0.632)\varepsilon_{\text{resub}} + 0.632\varepsilon_0. \tag{3.7}$$

The problem in the 0.632 bootstrap estimator is that it uses the resubstitution error, which, like mentioned earlier, is optimistic depending on the classification model [68].

3.2.3 Parametric Methods

Unlike CV, the parametric methods introduced in this section don't require splitting of the data. Instead, given the entire training data set and some properties of the classification model, the parametric methods return a single value inferring something about the model performance and generalizability. Usually, however, some assumptions need to be made, e.g., about the distribution of the data.

AIC

The *Akaike information criterion* (AIC) [69] is an information theoretical tool to measure the relative information loss due to using a trained

classifier model to predict the class labels instead of applying the true underlying labeling process. AIC is defined as

$$\boldsymbol{AIC} = -2l(\boldsymbol{\theta}; \mathcal{D}) + 2d^*, \tag{3.8}$$

where $l(\boldsymbol{\theta}; \mathcal{D})$ is the log-likelihood for the trained model (the definition was given in Section 2.3.1) and d^* is the number of free parameters of the model (the number of non-zero classifier coefficients). As shown by the above equation, the information loss gets higher as likelihood decreases or the number of parameters increases. Because AIC only measures the *relative* information loss, it is suitable for model selection, not for estimating absolute performance or error.

BIC

One of the most used statistical error estimators in addition to AIC is the *Bayesian information criterion* (BIC) [70] developed from the Bayesian perspective to estimate the posterior probability of the model. Recently, Chen and Chen [71] have developed the *extended BIC* (EBIC) that is better suited for high dimensional ill-posed problems. For a sparse logistic regression classifier, EBIC is defined as

$$\boldsymbol{EBIC} = -2l(\boldsymbol{\theta}; \mathcal{D}) + d^* \log(N) + 2d^* \delta \log(d), \tag{3.9}$$

where $l(\boldsymbol{\theta}; \mathcal{D})$ is the log-likelihood (see Section 2.3.1), d^* is the number of free parameters of the model (the number of non-zero classifier coefficients), N is the number of training samples, and d is the dimensionality of the data. Parameter $\delta \geq 0$ is used for tuning the sensitivity of EBIC for number of selected features. The higher is the value of δ the stronger is the effect of the number of selected features compared to the data likelihood term. In the experiments, we use $\delta = 0.5$, which has been reported as being a good compromise in many applications [71].

Although derived from different principles, AIC and BIC share similarities as noticed by looking at their equations (3.8 and 3.9). The practical difference is that BIC tends to penalize the number of parameters more strongly compared to AIC.

BEE

Another Bayesian method for classifier error estimation, namely the *Bayesian minimum mean-square error estimator* (BEE), was recently

introduced for a binary classification problem using discrete classifiers [72] and linear classifiers such as the logistic regression classifier [73]. BEE is a *minimum mean-square error* (MMSE) estimator estimating the classification error

$$\boldsymbol{BEE} = p(C=1)\varepsilon_1 + (1-p(C=1))\varepsilon_2, \tag{3.10}$$

where $p(C=1)$ is the prior probability of class 1 (negative class) and ε_1 and ε_2 are the probabilities of false positive and false negative classifications, respectively.

Assuming that the class conditional data distributions are Gaussian and that the class covariance matrices follow the inverse-Wishart-distribution, [73] gives a closed form solution for BEE in the case of linear classifier with parameters β_0 and $\boldsymbol{\beta}$ (see Section 2.2 for the linear classification model). In this thesis, a simple scaled identity covariance model is assumed for the class conditional densities and non-informative class prior is used. In our experiments, the covariances and the means of the class conditional densities are further limited in a minimal way by setting $\nu = 0$, $\kappa = 0$, and $S = 0$. See more details in [73]. Now, consider the usual sample estimates for the class means

$$\widehat{\mu_c} = \frac{1}{N_c} \sum_{k:\ c_k=c} \mathbf{x}_k \tag{3.11}$$

and covariance matrices

$$\widehat{\Sigma_c} = \frac{1}{N_c - 1} \sum_{k:\ c_k=c} (\mathbf{x}_k - \widehat{\mu_c})(\mathbf{x}_k - \widehat{\mu_c})^T, \tag{3.12}$$

where N_c is the number of samples in class $c = 1, 2$. The above assumptions lead to ε_c being defined as

$$\varepsilon_c = \frac{1}{2}\left(1 + \mathbf{sgn}\,(A)\, B\left(\frac{A^2}{A^2 + (N_c - 1)\,\mathbf{tr}\left(\widehat{\Sigma_c}\right)}; \frac{1}{2}, \alpha\right)\right), \tag{3.13}$$

where

$$\mathbf{sgn}(z) = \begin{cases} -1, & z < 0 \\ 1, & z \geq 0 \end{cases} \tag{3.14}$$

is the sign function,

$$B(x; a, b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \int_0^x t^{(a-1)}(1-t)^{(b-1)}\, \mathrm{d}t \tag{3.15}$$

is the regularized incomplete beta function, tr$(\cdot)$ is matrix trace, and α and A are real-valued quantities summarizing the data and the classifier such that

$$\alpha = \frac{d(d+1)}{2} - 1 \tag{3.16}$$

and

$$A = \frac{\beta_0 + \boldsymbol{\beta}^T \widehat{\mu_c}}{||\boldsymbol{\beta}||} \sqrt{\frac{N_c}{N_c + 1}}. \tag{3.17}$$

The mathematical derivation for Equation 3.13 given the above assumptions is found in the Appendix D of [73].

In a recent study, we have shown that BEE is especially efficient in problems, where the amount of training data is small and CV methods suffer from splitting the data [23]. This topic, along with experiments, is further discussed in Section 3.2.5. We have made our MATLAB implementation of the above BEE formulation available at `https://sites.google.com/site/bayesianerrorestimate/`.

3.2.4 Pitfalls in Cross-Validation

Hastie et al. [9] describe a wrong and right way to do CV. The key idea is that the different folds in CV need to be independent in order to prevent the results from being optimistic. In scientific literature, too often all the data is used for making decisions about the model after which CV is run. In particular, Hastie et al. concentrate on feature selection: The incorrect strategy consists of three steps [9, Sec. 7.10.2]:

1. Find a subset of good predictors that exhibit strong correlation with the class labels.
2. Using only this subset, design a classifier.
3. Use CV to tune the model and estimate the prediction error.

The problem above is that all the data is used in the first step, thus, producing optimistic error estimates and worse performance for truly independent test data. However, it turns out that the misuse can be a lot more subtle and difficult to recognize as we will see.

Following the example of [9], we did the following experiment [22].

1. Generate $N = 100$ samples of random data of dimension $d = 30$.
2. Generate binary class labels for the samples, also at random.

3. **Honest way:** Design a classifier and calculate the classification error using 10-fold cross validation. In this example we used a 3-nearest-neighbor (3NN) classifier.

4. **Cheating (version 1):** Violate the CV principles as in [9]:

   (a) Find the single feature that gives the smallest classifier error with 3NN.

   (b) Estimate the error as in step 3, but with the best feature only.

5. **Cheating (version 2):** Use 10-fold CV twice:

   (a) Estimate the classification error for each single feature using 10-fold CV.

   (b) Estimate the error using 10-fold CV as in step 3, but with only the best feature found in step 5a.

6. **Cheating (version 3):** Proceed as in step 5, but change the random seed between the two CVs, thus, using different division of samples.

The above procedure was repeated 1000 times and the histograms of the estimated classifier performances (ACC) for each case (honest and cheating 1–3) are shown in Figure 3.4. It can be seen that the honest way without any feature selection estimates the accuracy in a realistic manner, i.e., the performance estimate is close to random chance 0.5. However, all three ways of cheating give an optimistic estimate of the performance clearly above 0.5.

One might think that using cross-validation in the feature selection step (cheating versions 2 and 3) would be less harmful than using all data (cheating version 1) because the best performing feature in each CV fold is selected by testing on an independent set of data. However, it turns out that the double-CV approaches are clearly the most dangerous ways of incorrectly assessing the classification performance. There are at least two reasons for this to happen: 1) The illusion about the independence of the CV folds is broken right after the first feature selection CV when the CV results of the best performing features are combined by averaging. 2) Both the first and the second CV use 10 folds, which allows the first feature selection CV to exploit the information about the exact sample size that is going to be used in the second CV that estimates the final classification performance.

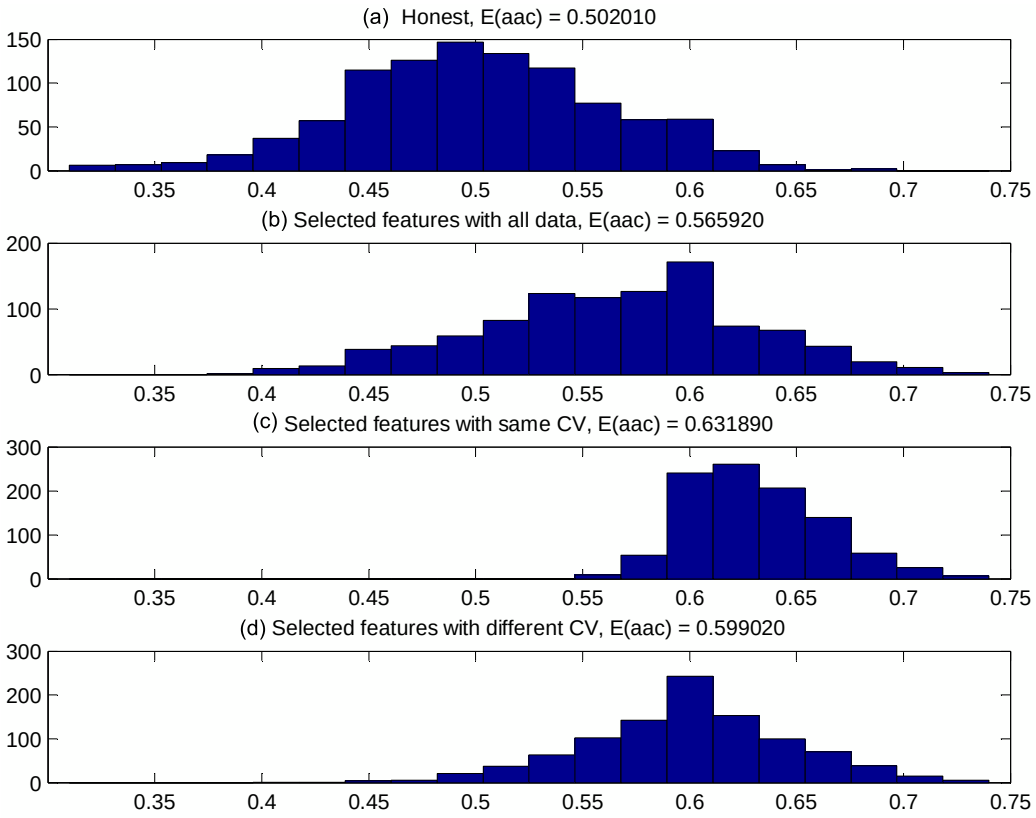


Figure 3.4: Wrong and right ways to do CV: (a) accuracy over 1000 tests without feature selection, (b) feature selection with all data, (c) feature selection using 10-fold CV twice, (d) feature selection using 10-fold CV twice with different seeds.

The lesson of the experiment in Figure 3.4 is that doing several CV runs in a sequence and using information from the previous CV in the new one can be more harmful than making the classical mistake of using all data, e.g., for feature selection. In practice, however, these kind of sequential CV runs are done extremely often as demonstrated by the next example.

Consider a situation where you are validating your classification model and get a performance estimate by using CV. Next, you think of ways to improve your model. Let say you decide to put some new features into the model and see what happens. You run the CV again, using the same random number generator seed than previously in order to make the new performance estimate comparable with the old one. You find out that the new model gives better CV performance than the

previous one, which will naturally make you choose the new set of features over the previous one. At this point, however, by comparing the new CV performance with the old one, you are doing exactly what was shown to be the most wrong way of doing CV. In an extreme case, you could even iterate the same procedure until you had found the perfect set of features for your particular CV and division of data into folds. In fact, this is how classical feature selection methods such as *floating selection* [74] work; by sequentially adding or removing features and assessing the classification performance at each step by using CV.

The unfortunate thing in the above example is that it is hard, or even impossible, to prevent this kind of optimism from happening. How do you improve your model if you are not allowed to make any decisions based on previous CV runs? There is no solution but to try to make decisions that are likely to create as little optimism in the results as possible. For instance, tweaking a couple of model parameters and seeing how the CV estimate changes is clearly less harmful than systematically going through all possible feature combinations and ultimately choosing the one that gives the best CV result.

3.2.5 The Effect of Sample Size in Error Estimation

An important factor affecting the choice of the used error estimator is the sample size, which is what we have studied in a recent article [23]. Figure 3.5 shows an example of the effect of the training sample size on the performance of different error estimators in a model selection problem where the task is to find a suitable value for the regularization parameter λ in the ℓ_1 regularized logistic regression model (see Equation 2.12). The experiment has been conducted by using a high-resolution ovarian cancer data set from the FDA-NCI Clinical Proteomics Program Databank[1]. The data set is used in a binary classification task of diagnosing whether the patient has ovarian cancer or not. The data comes from laser desorption and ionization (SELDI) protein mass spectrometry and has 15000 features corresponding to ion intensity levels at different mass/charge values. There are 216 patients in the data set, 95 controls and 121 ovarian cancers.

Five different estimators for estimating the model classification performance have been tested in Figure 3.5: BEE, EBIC, and 2-fold, 5-fold, and 10-fold CVs. The vertical axis shows the difference between the classification performance of the optimal logistic regression model that

[1] `http://home.ccr.cancer.gov/ncifdaproteomics/ppatterns.asp`

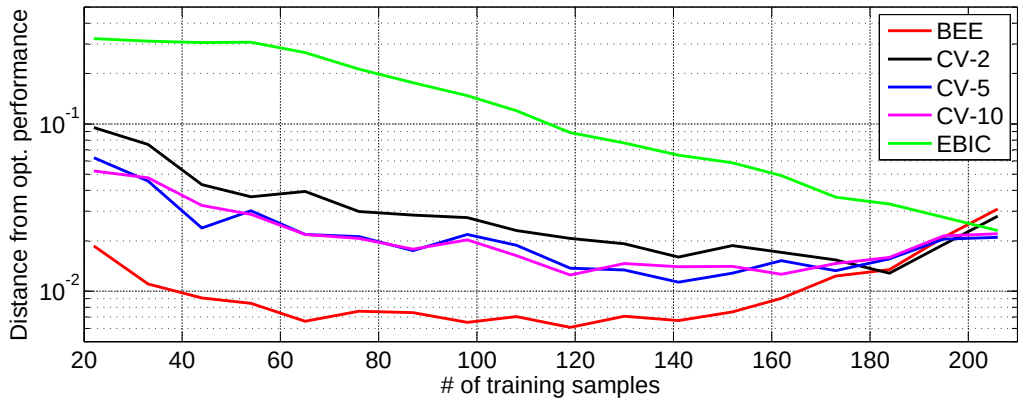


Figure 3.5: Comparison of different error estimation methods in selecting a suitable value for the regularization parameter λ.

has a value of λ that minimizes the classification error with the test data and the model where λ has been chosen such that it minimizes the value of the particular error estimator evaluated by using the training data only. The number of training samples is varying on the horizontal axis. In each case, the test set is comprised of the samples left over from the training set. Splitting into training and test set has been done randomly and the results are averaged over 100 such splits.

As shown by Figure 3.5, the amount of training data has a significant impact on the performance of different error estimators. Especially with the small training sample size, BEE is clearly the best estimator. All CV schemes give approximately same results while EBIC is clearly the worst. With large training sample size, the differences between the performances of error estimators become close to zero. The fact that errors start to increase with BEE and CVs at the end is because of the lack of testing samples when most data is used for training. For instance, the rightmost data point of Figure 3.5 has only 11 test samples (5 % of all data), which makes the test error highly variable. Thus, the results with large training sets are not reliable in this experiment.

In addition to the performance of a particular error estimator, the computational evaluation time often plays a significant role in a practical use. When used as a method for automatic parameter selection, such as the regularization parameter λ in a logistic regression model, the estimators are often evaluated inside a higher level validation procedure (CV loop) evaluating the actual performance of the model. In

this case, computational efficiency becomes important as the estimator is possibly evaluated several times for different splits of the data.

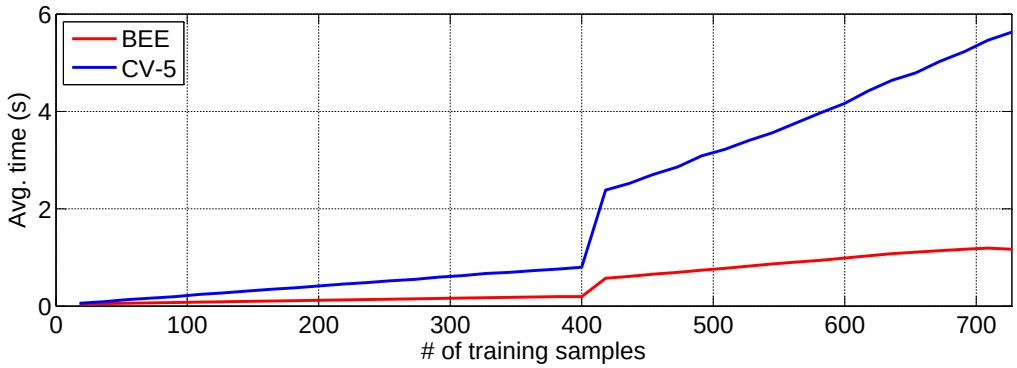


Figure 3.6: Comparing the execution times of BEE and 5-fold CV in estimating the performance of ℓ_1 regularized multiclass logistic regression models with different values for the regularization parameter λ.

Figure 3.6 shows a comparison between the execution times of BEE and 5-fold CV as a function of the size of the training data. The execution time on the vertical axis is measured as the average of 250 times to evaluate the performance of an ℓ_1 regularized logistic regression model with 100 different values for the regularization parameter λ. The horizontal axis shows the amount of training samples. In this test we used the MEG data set, which will be described in the application case in Section 4.4.

As seen in Figure 3.6, BEE clearly outperforms CV in the execution time. This is because there is no need to repeat the time consuming training step unlike in CV. Coordinate descent is used here for training the logistic regression models [26]. The particular implementation switches between two different coordinate descent algorithms depending on the ratio of sample size and dimensionality of the data. The sudden step in the execution times is due to this switch.

Chapter 4

Case Studies

In this chapter, results on real life classification problems where the sparse logistic regression model has been successfully used are introduced. The following work is based on publications and data challenges that the author has been involved in.

Section 4.1 first introduces a general purpose sparse logistic regression based image segmentation framework and a related application in the field of cell segmentation from microscope images. Next, in Section 4.2, the same supervised image segmentation approach is applied in object detection in inkjet printed electronics manufacturing. Other applications of sparse logistic regression are related to diagnosis of acute myeloid leukemia (Section 4.3) and brain signal analysis (Section 4.4). All the introduced application cases use sparse logistic regression, which has proven to be a universal classification method usable in many application areas.

4.1 Supervised Image Segmentation

In this section, we introduce a general purpose supervised image segmentation framework that uses sparse logistic regression for classifying each pixel of an image into foreground or background [20] (Section 4.1.1). Segmentation is used in several applications in the field of image analysis. One of these is the segmentation of cells from microscope images which is presented in Section 4.1.2.

4.1.1 Overview of the Supervised Image Segmentation Framework

The traditional approach for solving segmentation and detection problems in image analysis is very application oriented. Often the designer is an expert in image analysis techniques. However, the end user of the image analysis tool, especially in the field of medical image analysis, is usually not too experienced in the field and may lack the skills for manual tuning, which is often necessary in order to get satisfactory segmentation results.

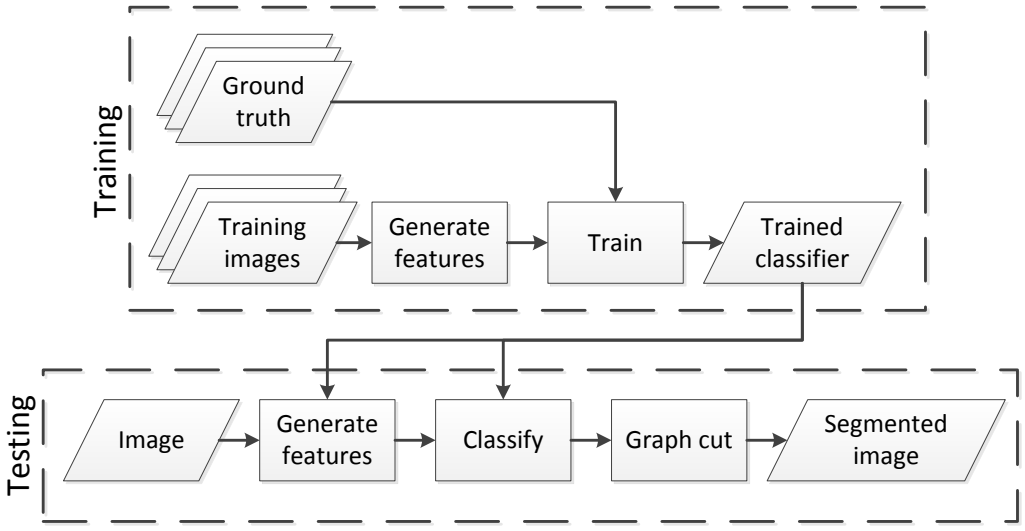


Figure 4.1: Block diagram of the supervised image segmentation framework.

In [20], we have proposed to use ℓ_1 regularized logistic regression together with an MRF spatial prior for a general segmentation task. The approach is supervised, i.e., there are separate training and test phases. These have been illustrated by the block diagram in Figure 4.1. The different stages of processing are explained in the following starting from the training stage that has three steps:

- **Collecting training data.** First, a set of training data needs to be collected for training the classifier. In practical applications this is basically a set of *training images* together with their *ground truth* segmentation, e.g., manually created by the user.
- **Feature generation.** A set of features is generated for each training image pixel by applying a set of manually defined image

filters with a range of parameters. The idea is to use a large pool of features from which a good subset is likely to be found for different types of applications. The features can include, e.g., low pass filters, edge filters, texture features, morphological filters, wavelet decomposition features [75], and local binary patterns [76], all of which are applied with varying parameter values, kernel sizes, etc.

- **Training.** A binary logistic regression classifier is designed to determine if a pixel belongs to the foreground or to the background. The key idea is to use ℓ_1 regularization in the coefficient estimation in order to automatically select the features that are relevant from the viewpoint of the application at hand. The `glmnet` package [26] is used for coefficient estimation and a proper value for the regularization parameter λ is chosen by using CV.

The actual classification stage is the following:

- **Feature generation.** For each image pixel, features are calculated. Only the features selected by the regularization in the training phase need to be computed. Thus, computationally efficient segmentation can be achieved even if the feature set used in the training phase was large.

- **Classification.** The trained logistic regression classifier is used for predicting the probability of each pixel belonging to the foreground.

- **Graph cut.** The graph cut algorithm is applied to produce the MAP segmentation of the image that combines the MRF spatial prior with the pixel-wise posterior probabilities produced by the logistic regression model as explained in Section 2.4.

The main idea in the above segmentation framework is that with a comprehensive choice of the initial set of image features, the same training and actual segmentation steps can be used in many different applications with little effort or special knowledge from the user. Similar ideas for supervised segmentation with feature selection have been used in pixel classification with hyperspectral images [51, 77] and in backscatter image segmentation [78]. Of these, the first inputs multichannel images acquired at different wavelengths directly with no filtering, while the second uses a fixed collection of filters, but selects

the most relevant ones with simple forward selection. Moreover, the latter paper does not estimate the class probabilities, which have a significant role in our framework. Further, a highly similar approach of *discriminative random fields* (DRM) has been proposed by Kumar and Hebert [79] who use discriminative logistic regression modeling together with a spatial prior. The key difference in this approach is that the prior probabilities are modeled by using a combination of the Ising model and a data dependent logistic regression model for which the coefficients are learned jointly with the pixelwise class probability model. In our case, however, automatic feature selection is in a central role. Estimating the coefficients of the pixelwise class probability model separately from the prior allows us to use conventional learning algorithms [26] and, thus, efficient ℓ_1 regularization for achieving the feature selection property.

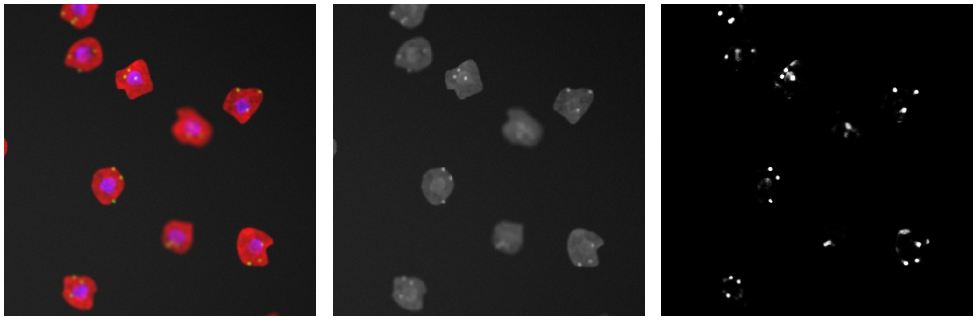
Two real life application cases using the segmentation framework are introduced in this thesis: subcellular spot detection (Section 4.1.2) and object detection in inkjet printed electronics manufacturing (Section 4.2). In addition to these applications, similar approach has been used in outline-based cell segmentation [80].

4.1.2 Segmentation of Cell Images

The first application case utilizing the supervised segmentation framework involves detection of subcellular objects from simulated microscope images [81]. Reliable automated subcellular object detection is needed in several applications including interpretation of complex cellular phenotypes, analysis of cellular structures from fluorescence and confocal microscope images, and live-cell tracking [82].

We have applied the proposed supervised segmentation framework for cell segmentation in [20]. In this study, we use simulated cell images instead of real microscope images. The main benefit in using simulated images is in the availability of the ground truth for the number and locations of the subcellular objects. The simulated image set we use is designed for benchmarking purposes [82] and publicly available at `http://www.cs.tut.fi/sgn/csb/simcep/benchmark`.

The simulated image set contains 20 images with a fixed number of cells and subcellular objects per cell. An example of a simulated cell image is shown in Figure 4.2a. The images contain cells with nuclei (blue channel, B), cytoplasmic areas (red channel, R), and subcellular objects (green channel, G). The segmentation task is to detect the sub-



(a) (b) (c)

Figure 4.2: An example of a simulated cell image (a), its gray scale version (b), and the estimated probabilities (c) of individual pixels belonging to a subcellular spot. The probabilities have been attained by applying a logistic regression model together with a Markov Random Field prior.

cellular objects from a grayscale image obtained by using the standard conversion $0.2989R + 0.5870G + 0.1140B$ (Figure 4.2b).

Table 4.1: Features and parameter ranges used in the experiments. In total, 106 features were used in this study.

Feature	Parameter	Values
Gaussian lpf	kernel width σ	$\{3, 5, 7, \ldots, 83\}$
Unsharp masking	kernel width σ	$\{3, 5, 7, \ldots, 83\}$
Morphological top-hat	kernel size	$\{3, 5, 7, \ldots, 83\}$
Morphological bottom-hat	kernel size	$\{3, 5, 7, \ldots, 83\}$
Masking with h-maxima transform	h	$\{3, 5, 7, \ldots, 83\}$
Local binary patterns	-	-
Edge enhancement	kernel size	$\{3, 5, 7, \ldots, 83\}$
Wavelet decomposition	depth layer	$\{1, 2, 3\}$

As an experiment, we estimate the segmentation performance by computing the pixel-wise F-score, precision, recall, and the numbers of true positives, false positives, and false negatives between the segmentation result and the ground truth segmentation provided by the image simulator. To avoid optimism in the segmentation performance, the training and testing of the logistic regression classifier is performed in a LOO CV manner such that each of the 20 images in the above image set are used for training at a turn and the remaining 19 are used

for testing. From each image, 70 foreground and 700 background pixels are used for training purposes. The initial set of features computed for each pixel consists of 106 features with different parameters as listed in Table 4.1.

Table 4.2: Subcellular object detection results for simulated images. The results are averaged over the set of test images.

tp	*fp*	*fn*	prec.	rec.	F-score
121.11	17.26	20.00	0.876	0.858	0.867

Averaged results over the 20 training and testing iterations are shown in Table 4.2. The table shows the average number of true positives, false positives, and false negatives, and the average precision, recall, and F-score. In [81], a similar analysis was conducted over the same set of 20 simulated cell images by using 9 conventional segmentation methods. In this study, the majority of the F-scores measuring the overall segmentation performance were located between 0.4 and 0.6, while the best result was obtained by using *Multiscale wavelets* [75] that had F-score of 0.63. Clearly, the supervised segmentation framework outperforms all these methods with F-score of 0.867.

An example of the output of the logistic regression model (the probability of the foreground) is shown in Figure 4.2c. In this case, the trained model is extremely easy, namely, regularization only selects 5 of the available 106 features into the model. These features are *h-maxima masking* with $h = 63$ and $h = 68$ (H_{63}, H_{68}), edge enhancement and top-hat filtering with kernel size 8 (E_8, T_8), and wavelet decomposition with depth layer 3 (W_3). Given the feature vector $\mathbf{x} = (H_{63}, H_{68}, E_8, T_8, W_3)^T$, the model can be written as

$$p\,(\text{"foreground"} \mid \mathbf{x}) = \frac{\exp\left(g(\mathbf{x})\right)}{1 + \exp\left(g(\mathbf{x})\right)}, \tag{4.1}$$

where the discriminant function is given as

$$g(\mathbf{x}) = -5.23 + 0.04H_{63} + 0.003H_{68} + 0.0007E_8 + 0.011T_8 + 0.3W_3. \tag{4.2}$$

From the coefficient magnitudes one can see that most of the contribution into the classification rule comes from the wavelet decomposition W_3. All the features have been normalized to unit variance in order for the comparison of the coefficient magnitudes to make sense.

As a conclusion, when it comes to the segmentation of subcellular spots, the proposed supervised segmentation framework seems to

clearly outperform the 9 traditional image processing approaches as given in [81]. The difference between the performances is further signified by the fact that, in [81], a grid search was conducted in the parameter space of the different segmentation methods in order to tune them specifically for the 20 tested images, while CV was used in our experiments in training the logistic regression model. In addition to improved segmentation performance, the sparsity property of the regularized logistic regression classifier is able to point out only a handful of simple image processing operations that are adequate in conducting the segmentation.

4.2 Object Detection in Inkjet Printed Electronics Manufacturing

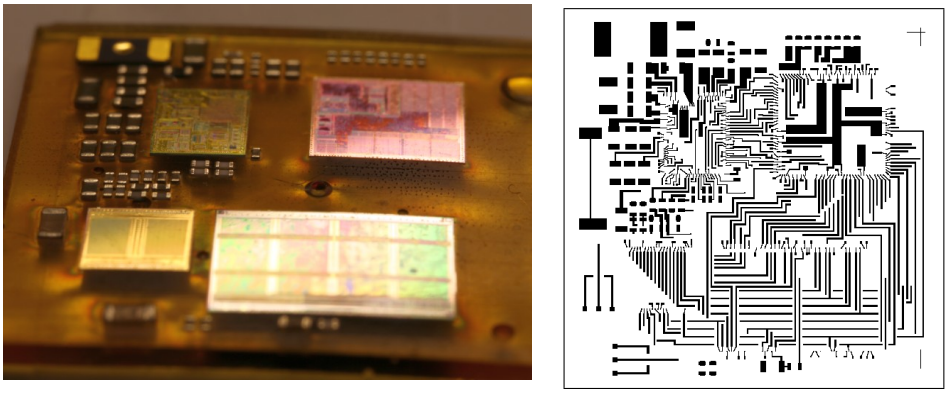
The next application case comes from the field of electronics, namely, *inkjet printed electronics manufacturing*. In this application, the image segmentation framework described in Section 4.1.1 is used as a part of a computer vision and image processing pipeline. The aim is in automatic detection of electronics components from camera images in order to do online error compensation for the manufacturing process.

The next sections are organized such that, first, in Section 4.2.1, a brief introduction is given to a specific manufacturing method of inkjet printed electronics and a particular problem that arises due to misaligned components. Second, in Section 4.2.2, the concept of computer vision controlled printing for fixing this problem is introduced as proposed in our earlier work [19]. Finally, methods and results for connection pad detection are given in Section 4.2.3.

4.2.1 Inkjet Printed Electronics and the Problem of Misaligned Components

Printed electronics is a manufacturing process of electronics where traditional printing devices are used for interconnecting or manufacturing of electronic components. The area of research is relatively new. Unlike conventionally used etching processes which are based on material removal, printed electronics is an additive manufacturing process where material is only added where needed. Advances in new materials allow novel devices, such as flexible displays or *Radio Frequency Identification* (RFID) labels to be generated using this approach.

One of the fundamental printing technologies is *inkjet* printing that relies on the principles of traditional inkjets with fluid materials targeted for electronics. An example application of this technology uses conductive nano particle and dielectric inks to create interconnections between integrated circuits (ICs) and discrete components that have been molded onto the background surface [83]. An example module of this kind is shown in Figure 4.3.



(a) (b)

Figure 4.3: A module with four integrated circuits (ICs) (a) ready for printing of the interconnection layer (b).

One of the most significant challenges in the printed module concept shown in Figure 4.3 is the accuracy of the manufacturing process. Inaccuracies in the component placement process, molding process related movement, and molding material shrinkage and bending all create errors in component locations. The latter category is typically the most significant one, and also the most unpredictable [84].

The errors in the component locations create a *misalignment problem* between the printed layer and the target surface. Traditional inspection approaches [85] for printed circuit board (PCB) manufacturing are unsuitable solutions for the problem because, in PCB manufacturing, the background materials are rigid, which allows only translation and rotation. In this case, the substrate may shrink or expand causing the components to move with respect to each other. In addition to the complexity of the transformation, most existing approaches concentrate on quality control instead of actually fixing the problem.

4.2.2 Computer Vision Controlled Printing

In our earlier work [19], we have proposed a computer vision system to compensate the errors created due to misaligned components. Computer vision enables a fast, easy, and efficient way for analyzing printouts and printed surfaces. Without the necessity for a physical contact, measuring from an image is especially suited for applications of inkjet printed electronics. Figure 4.4a shows an example of integrating a computer vision system into the printing device in order to detect the exact position and orientation of an electronics component (Figure 4.4b) on the printing stage.

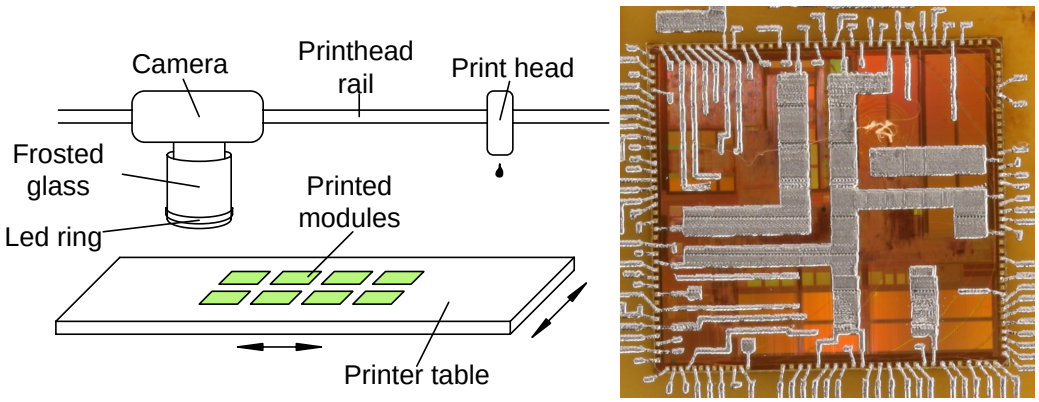


Figure 4.4: Figure (a) illustrates a camera solution integrated into a printing device. A mobile printing stage enables a fixed camera mounting. In Figure (b), the orientation of the background component has been detected by using computer vision before printing the wiring layer. Thus, it has been possible to align and modify the print pattern to perfectly fit the underlying component.

A software level framework diagram for the computer vision controlled correction system is shown in Figure 4.5. Briefly, the idea is that any distinguishable features in the camera image serve as *control points*, which are first automatically detected from the image and then associated with the corresponding control points in the design. The resulting displacement information is then used to modify the printing data before printing. There are three main stages in the processing:

- **Object detection.** The processing starts with an image processing step where *candidate control points* are detected from the

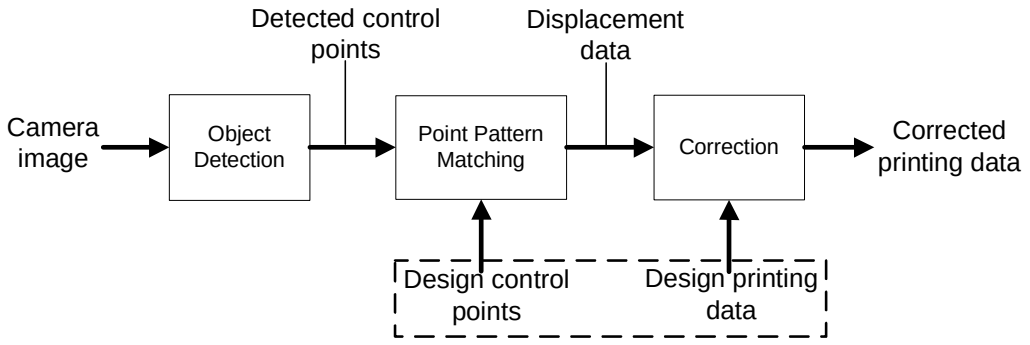


Figure 4.5: A diagram of a general computer vision based adaptation system for inkjet printed electronics that operates on the design printing data based on the feedback from the camera image and the image analysis pipeline.

camera image. Detected objects usually have relatively simple appearance (e.g., connection pads), which creates plenty of missing and outlier points amongst the candidates. This is the processing block where supervised learning and classification is applied as described in Section 4.1.1.

- **Point pattern matching.** After detecting the candidate points, they are associated with their counterparts in the design. The target is to determine the errors in the locations of the control points relative to the design. This is done with a *point pattern matching* (PPM) algorithm, also known as *procrustes analysis* [86] in statistics. Generally, the PPM problem is rather complex and the state-of-the-art algorithms assume various properties of the point sets to ease the computation. Particularly in the printed electronics case, however, the control point sets are usually man-made (like the connection pads on the ICs) and, thus, contain regular and symmetrical patterns. These kinds of point sets complicate the matching as several well matching subset pairs are present. We have proposed a PPM algorithm suitable for these applications in [87].

- **Correction.** The final step is to compensate the detected errors. In the example case in Figure 4.3 this means redrawing the wiring prior to the printing. We have proposed a smooth geometrical transformation that remaps the wiring image without affecting its electrical properties in [88]. In other applications the

correction could be done, e.g., by tuning the printer parameters or by merely discarding the module in an inspection style if it's impossible to apply any correction.

An automated correction system such as the one explained above is essential for improving the manufacturing process of printed electronics and making inkjet printing a manufacturing method suitable for mass production. The relevant part from the viewpoint of this thesis in such a system is the one utilizing machine learning in the image processing task. This is the topic of the next section.

4.2.3 Detection of Connection Pads from Camera Images

From the viewpoint of this thesis, we are interested in applying machine learning for automatic detection of connection pads on electronics components. This is done in order to solve the component misalignment problem of the inkjet printing application as described in the previous Sections 4.2.1 and 4.2.2.

A previous method for connection pad detection utilizes a traditional image processing pipeline with the following three steps [19, 89]:

1. **IC extraction.** The ICs are first located in order to limit the further processing inside the IC areas only. This is done by using template matching such that a template image of the corresponding module type with known IC locations is matched against the camera image. A successful match results in the bounding boxes of the ICs. The detected IC boundaries are only approximate since the IC misalignment characteristics and the module orientation can vary between the template and the image under processing.

2. **Pad candidate detection.** Second, a set of feasible candidates for IC connection pad locations is detected. This is done inside the previously detected IC areas by, first, highlighting the pad-like objects by using template matching and, then, using a modification of a local thresholding operation called *logical level thresholding* [90]. Logical level thresholding is based on the assumption that, after highlighting with template matching, the connection pads appear as small bright spots. The result of the logical level thresholding is further filtered by using morphological shape filtering and the knowledge of the minimum distance between two neighboring connection pads.

3. **Classification.** Finally, the set of pad candidates is classified into actual pads and false detections by using supervised classification. Artificial feedforward neural networks are used here after resizing each color image patch of a connection pad candidate into the size of 6×6 pixels. The adjustment of the classification threshold affecting the true positive and true negative rates of the classifier is done by using the knowledge of how many connection pads should be detected from each IC.

With the above method, high connection pad detection rates can be achieved [19]. However, the main problem is that the algorithm has several processing steps that make it computationally inefficient and, thus, unsuitable for online use. In addition, a training step is needed not only for the neural network classifier but also for template matching in both IC and pad candidate detection stages. This makes the practical use of the algorithm rather cumbersome.

As the new connection pad segmentation method, the supervised segmentation framework introduced in Section 4.1.1 is proposed. The idea is to apply logistic regression segmentation for full size camera images, take the detected object centroids as detected connection pad candidates, and then use PPM to find individual ICs and exact connection pad locations among these. The main benefits compared to the previously described method are that

- IC extraction and a separate classification step can be omitted because the segmentation framework is efficient enough to work on full size images and the segmentation is reliable enough for the PPM algorithm to detect individual ICs among the connection pads detected from the entire image.

- The template matching step used in the previous method for IC extraction cannot handle excess changes in module orientation. Due to the PPM algorithm, the new method is completely rotation, translation, and scale invariant (provided that the connection pad formations on the ICs are not rotation symmetric).

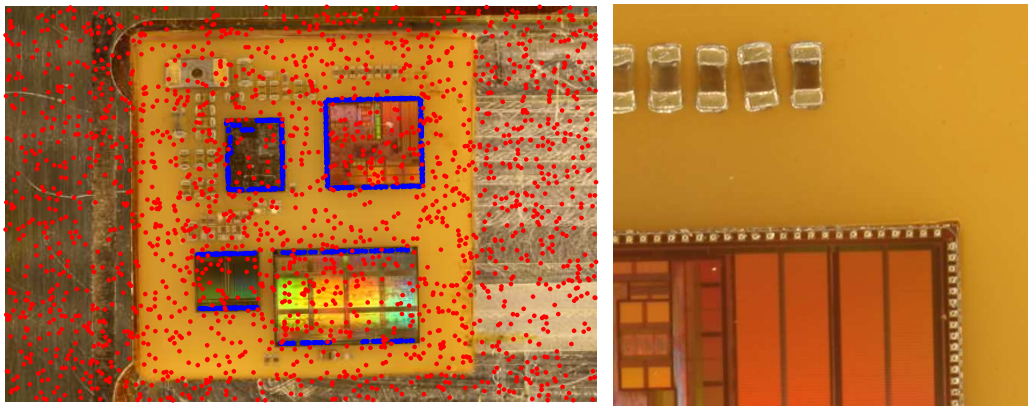
- No parameters need to be tuned unlike in the previous method, where each processing step had several parameters such as assumed pad size, thresholds and neighborhood size for logical level thresholding, and shape parameters for morphological filtering.

- Training is easy and can be done with a single module image. Only the ground truth segmentation is needed for training unlike

in the previous method, where example templates of the module and the connection pads were needed in addition to a comprehensive training set of positive and negative connection pad images for training the neural network classifier.

- There's a significant reduction in algorithm complexity and, thus, in execution time. This enables the use of the proposed algorithm in an online process.

The main benefit of the new approach is in its simplicity and computational effectiveness. However, because the entire camera image is considered at once and a separate classification step is omitted, it results in more false positives compared to the old method. Fortunately, our experiments have shown that this decrease in detection performance is tolerable by the PPM algorithm, which couples the individual ICs with the connection pad candidate points detected from the image. The important point is that the number of missed connection pads should be as low as possible as this can have a serious effect on the performance of the PPM algorithm.



(a) (b)

Figure 4.6: An electronics module with 4 ICs used in training a classifier for IC connection pad segmentation. The positive and negative training pixels are shown with blue and red dots, respectively. A close-up of a corner of one of the ICs shows connection pads located on the sides of the IC.

An example of the training data and the connection pads to be detected is shown in Figure 4.6. In Figure 4.6a, a set of positive (blue)

and negative (red) training pixels have been randomly chosen from a single module image by using the ground truth segmentation, which has been obtained by knowing the connection pad radius and locations. In this case the pad locations were obtained by using the old connection pad detection method and by verifying the result visually. This type of training procedure is used in the next experiment where we compare the segmentation framework with some other choices for connection pad segmentation.

4.2.4 Experiments

The experiments in the cell segmentation case in Section 4.1.2 indicated that supervised segmentation can easily outperform many traditional unsupervised algorithms. In this section, the performance of different supervised methods comparable to the proposed segmentation framework is studied in the segmentation of the connection pads. In addition to the proposed regularized logistic regression combined with the graph cut post processing (LR-MRF), also a regularized logistic regression classifier without graph cut (LR) as well as support vector machine (SVM) and a 10 nearest neighbors (10-NN) classifiers are tested.

We use 12 Mpix color images taken from modules with four ICs as shown in Figure 4.6a. There are a total of 409 connection pads located at the sides of each IC. Training data is collected by randomly picking 1000 positive and 2000 negative training samples per image as indicated by the blue and red dots, respectively. The actual connection pads can be seen in a close-up of one of the ICs in Figure 4.6b.

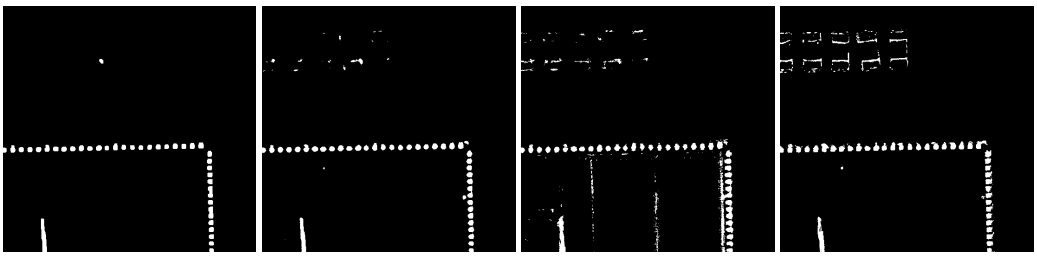
The features used in the classifier are the same as used in the cell segmentation experiments in Section 4.1.2 (see Table 4.1). In the case of SVM and 10-NN, an additional sequential forward selection step is used to reduce the amount of intentionally redundant features. In sequential forward selection, single features are selected into the model according to how well they improve the classification performance as estimated by CV in this case [74]. Features are added into the model with this fashion until the CV performance stops improving.

In SVM, we use linear kernel, which makes the model linear as in logistic regression. In 10-NN, ℓ_1 norm is used as the distance metric, i.e., the classifier decision is determined by the majority vote of the ten nearest neighbors in the training data in terms of the sum of the absolute differences between the corresponding feature values. In all the methods, the features are first normalized to unit variance.

The performance of the different classifiers is assessed by using a simple two-fold CV such that the classifiers are trained with a single module image similar to that shown in Figure 4.6a and tested with another one. As the topic is about assessing the segmentation performance, similarly to that in the cell segmentation case in Section 4.1.2, we settle for measuring the pixel-wise F-score, precision, recall, and true positive, false positive, and false negative rates rather than actual connection pad detection rates.

Table 4.3: Connection pad detection results. The results are averaged over the 2 CV folds.

Method	tp	fp	fn	p	r	F-score
LR-MRF	24484	87133	460	0.219	0.982	0.359
LR	24683	131573	261	0.158	0.990	0.272
SVM	24498	154422	447	0.137	0.982	0.240
10-NN	24566	156066	378	0.136	0.985	0.239



(a) (b) (c) (d)

Figure 4.7: The segmentation result as given by different classification methods. (a) LR-MRF, (b) LR, (c) SVM, (d) 10-NN.

The results are given in Table 4.3. As expected, there are plenty of false positives in each case, which is due to the high amount of small bright spots other than connection pads in the images. Logistic regression gives slightly less false positives than SVM and 10-NN resulting in better precision and F-score. Using MRF together with the class probabilities estimated by the logistic regression remarkably reduces the number of false positives giving it by far the most accurate result. Figure 4.7 shows the part of the segmentation that corresponds to Figure 4.6b. The next step would be to compute the centroids of each detected object and apply PPM between these and the connection pad

locations of each individual IC in the design data in order to detect the ICs and the exact positions of the connection pads in the image.

As a conclusion, the previous experiment shows that the proposed segmentation framework works well in the connection pad segmentation application. Especially, it is shown that the ℓ_1 regularization combined with logistic regression classification outperforms SVM and 10-NN classifiers coupled with sequential forward selection. Combining the MRF prior with the classifier further improves the segmentation result.

4.3 Automated Diagnosis of Acute Myeloid Leukemia

Next, logistic regression is applied for classification of *flow cytometry* data in order to diagnose whether a patient has *acute myeloid leukemia* (AML) or not. The work is based on *DREAM6/FlowCAP2 Molecular Classification of Acute Myeloid Leukaemia Challenge* organized as a part of the DREAM6 conference in 2011. A 100 % classification accuracy was achieved on the test set of 20 AML-positive and 160 healthy patients by using a logistic regression classifier with ℓ_1 regularization [91, 92]. Further, the sparsity promoting property of the regularization can be used as a valuable tool in evaluating, which of the markers used in standard flow cytometry are actually needed in the diagnosis. These aspects have been studied in a recent article where we have further improved the original challenge submission and conducted comparisons with competing models [18].

The forthcoming sections are organized such that Section 4.3.1 gives and introduction to using flow cytometry for AML diagnosis and describes the data set from the DREAM6 AML challenge. Next, Section 4.3.2 explains methods for AML diagnosis using logistic regression and some other supervised learning methods. Finally, in Section 4.3.3, some experiments are run.

4.3.1 Flow Cytometry in AML Diagnosis

Leukemias are a common malignancy of blood cells emerging from different cell types [93]. AML, which is the focus of this work, emerges during myeloid differentiation. However, the traditional classification of leukemias relies predominantly on morphologic and cytochemical

features of the tumor cells rather than the developmental origin of the malignancy [94].

Blood cancers such as AML are diagnosed with various techniques including features from morphologic, cytochemic, cytogenetic, and flow cytometry. While morphologic, cytochemic, and cytogenetic analysis include standard pathological stainings that lead to low dimensional data that can typically be interpreted directly under microscope, analysis of flow cytometric data includes the interpretation of more complex high dimensional data distributions. Thus, computer assisted decision systems are needed to support diagnosis decision making. [95]

Flow Cytometry

Flow cytometry can be used to analyze a large number of individual cells and, thus, is well suited to detect, e.g., cells that express particular cancer related surface markers from blood samples. In the measurement process, cells are labeled with fluorescent dye. Fluorescent labeled cells are guided through a laser beam and the resulting fluorescence and scatter parameters, typically forward and side scatter, are detected by photo detector. Forward scatter is informative of the cell size and side scatter corresponds to cell granularity. [95, 96]

Table 4.4: Seven tubes (and one unspecified control tube) with different fluorescence markers (FL1–FL5) are provided in the DREAM6 AML prediction challenge data set. (Table from challenge web site[1].)

Tube	FL1	FL2	FL3	FL4	FL5
1	IgG1-FITC	IgG1-PE	CD45-ECD	IgG1-PC5	IgG1-PC7
2	Kappa-FIT	Lambda-PE	CD45-ECD	CD19-PC5	CD20-PC7
3	CD7-FITC	CD4-PE	CD45-ECD	CD8-PC5	CD2-PC7
4	CD15-FITC	CD13-PE	CD45-ECD	CD16-PC5	CD56-PC7
5	CD14-FITC	CD11c-PE	CD45-ECD	CD64-PC5	CD33-PC7
6	HLA-DR-FITC	CD117-PE	CD45-ECD	CD34-PC5	CD38-PC7
7	CD5-FITC	CD19-PE	CD45-ECD	CD3-PC5	CD10-PC7
8	N/A	N/A	N/A	N/A	N/A

Data from the DREAM6 AML Challenge

Here, we use the DREAM6 AML prediction challenge data that is available at the challenge website[1]. The data set consists of flow cytometry

[1] `http://www.the-dream-project.org/challenges/dream6flowcap2-molecular-classification-acute-myeloid-leukaemia-challenge`

measurements taken from 43 AML-positive and 316 healthy patients. The challenge data has seven aliquots or tubes containing 6764 to 49370 blood cells that each give a single measurement, i.e., event, per used marker. A different combination of five different markers is measured from each tube. These marker combinations (FL1–FL5) have been listed in Table 4.4. Each marker is only present in one of the tubes except for marker CD45, which is included in all of the tubes. In addition to fluorescence intensities, forward and side scatter readings are provided for each tube. Tube number eight is a control tube with non-specific-binding antibodies. We don't use this tube for prediction purposes.

Both raw and preprocessed flow cytometry data are provided in the challenge data set. For prediction, we use the preprocessed data, which is compensated/translated using the method described in [97]. The preprocessed data has the forward scatter in linear scale and the side scatter and the fluorescence intensities described in Table 4.4 in logarithmic scale. In addition to the provided preprocessing, some experiments were conducted with a regression based preprocessing method [98] directly applied for the raw data. However, no improvement was gained in the results. Thus, all results presented in this thesis are for the preprocessed contest data, which is available in the contest web site.

4.3.2 Supervised Learning Methods for AML Diagnosis and Marker Analysis

Similar to that in other classification applications, the basic idea in classifying flow cytometry data into AML-positive and healthy patients is to, first, extract a set of features from the data and, then, apply the trained classifier. A speciality when using flow cytometry data is that there are an arbitrary number of individual cells in each tube. Further, a different combination of markers is measured from each tube. Thus, each tube needs to have a separate processing step that extracts a set of features, which are then combined as a single feature vector for the classifier to use. The processing chain using logistic regression classification has been illustrated in Figure 4.8.

Because the individual cells and their number varies from measurement to another, the features extracted from each tube cannot be cell specific features but statistics summarizing the joint distribution of the marker values. However, density estimation with multidimensional data is known to be a difficult problem. The situation is further com-

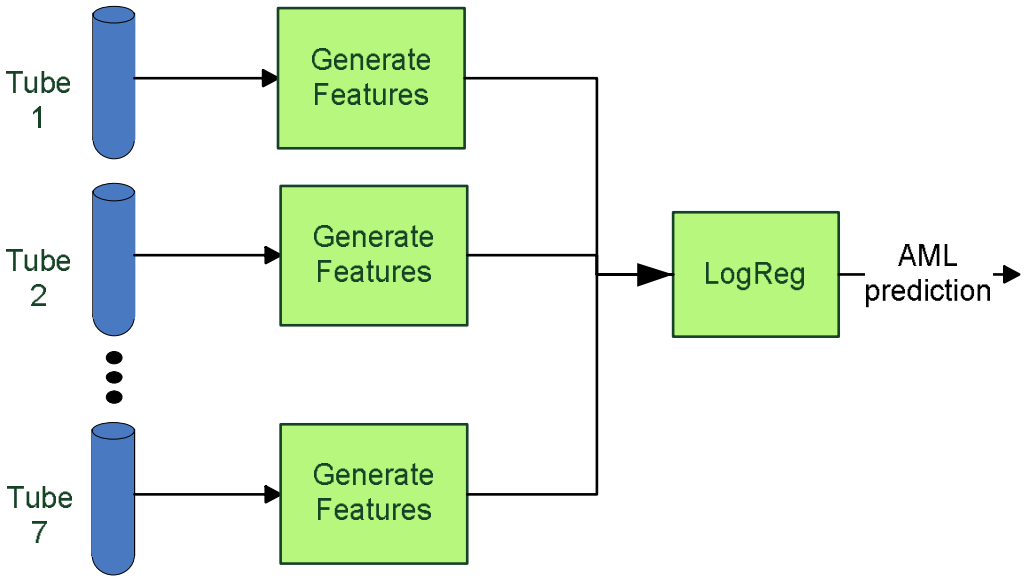


Figure 4.8: High level diagram on aggregating the flow cytometry data from individual test tubes for classification with a logistic regression model.

plicated by the large amount of data points. Thus, good feature engineering plays a particularly essential role in this application.

Sparse Logistic Regression with EDF MSE Features

Previously, we have proposed LDA as a supervised dimension reduction step for working out the problem of multidimensional density estimation [91]. In this approach, the distance between the *empirical cumulative distribution functions* (EDF) of the tested sample and those trained from AML and healthy patients are used as features for a sparse logistic regression classifier making the diagnosis. The proposed algorithm has 4 steps:

1. **Feature generation.** In a later processing stage, the prediction model uses rather simple linear operations and distribution comparison in one dimension. Thus, we cannot expect the model to discover very complicated relations between the input features. For this reason, the set of features from the initial 5 fluorescence intensities and 2 scatter features is artificially expanded. This is done by taking all possible inverses (7 pcs) and second powers

(7 pcs) of the features as well as multiplications (21 pcs) and divisions (42 pcs) of any combination of two different features. This way, the dimensionality of the data is increased from the initial 7 to 84.

2. **Dimension reduction.** To avoid the problem of multidimensional (in this case, 84-dimensional) density estimation, LDA is used to map each sample $\mathbf{x} \in \mathbb{R}^{84}$ generated in the previous step into 1-d:

$$x = \left(\widehat{\Sigma}_1 + \widehat{\Sigma}_0\right)^{-1} (\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0)\,\mathbf{x}. \tag{4.3}$$

Above, $\widehat{\Sigma}_1$ and $\widehat{\Sigma}_0$ are the sample covariance matrices and $\hat{\boldsymbol{\mu}}_1$ and $\hat{\boldsymbol{\mu}}_0$ are the sample means of both AML-positive and healthy training samples, respectively. LDA is a supervised method that uses training data for finding the linear mapping that best separates the AML-positive and healthy populations in 1-d. Applying the LDA significantly reduces the amount of data and allows us to use traditional 1-d methods for distribution comparison.

3. **EDF comparison.** The actual features from each flow cytometry tube are calculated as the distance of the sample distribution to trained distributions of AML-positive and healthy populations. As the distance measure, we use the mean squared error (MSE) between the EDFs:

$$\varepsilon_t^{(c)} = \frac{1}{K} \sum_{k=0}^{K-1} \left[F_t(x_0 + k\Delta x) - F_t^{(c)}(x_0 + k\Delta x) \right]^2, \tag{4.4}$$

where $t = 1, 2, \ldots, 7$ denote the seven different tubes. Above, $F_t(\cdot)$ is the EDF of the tested tube data after LDA mapping, and $F_t^{(c)}$ are the corresponding EDFs of the trained healthy tube ($c = 0$) and the trained AML-positive tube ($c = 1$). In Equation 4.4 the error is only evaluated in K discrete points uniformly chosen between points x_0 and $x_0 + (K-1)\Delta x$. Our validation tests show that we can use value as low as $K = 128$ without degrading the classification performance. We choose x_0 equal to the minimum of the training samples and Δx such that $x_0 + (K-1)\Delta x$ equals to the maximum of the training samples.

In addition to the EDF MSE, also more established distribution comparison methods were tested including *two-sample Cramér-von Mises test* [99], *Kolmogorov-Smirnov test* [100], and *Kullback-Leibler divergence* [101]. Further, error measures like correlation

and MSE between the probability density functions estimated with kernel smoothing density estimation were tested. However, based on validation results, EDF MSE gave the best classification performance.

4. **Classification.** Given the 14-dimensional feature vector $\boldsymbol{\varepsilon} = (\varepsilon_1^{(0)}, \varepsilon_1^{(1)}, \ldots, \varepsilon_7^{(0)}, \varepsilon_7^{(1)})^T$ corresponding to the EDF error measures of each tube as defined in Equation 4.4, the final classification into AML-positive or healthy patient is done by using ℓ_1 regularized logistic regression. In this application, the implicit feature selection via regularization automatically detects tubes irrelevant from the diagnosis point of view and excludes them from the model. The `glmnet` package [26] is used for coefficient estimation and a proper value for the regularization parameter λ is chosen by using CV.

When using all the available 359 patients in the DREAM6 challenge data set for training the logistic regression classifier with ℓ_1 regularization, the trained classifier can be written as

$$p\,(\text{"AML"} \mid \boldsymbol{\varepsilon}) = \frac{\exp\left(g(\boldsymbol{\varepsilon})\right)}{1 + \exp\left(g(\boldsymbol{\varepsilon})\right)}, \tag{4.5}$$

where the discriminant function is given as

$$g(\boldsymbol{\varepsilon}) = -4.5037 + 1.7980\varepsilon_4^{(0)} - 1.0845\varepsilon_4^{(1)} - 0.8694\varepsilon_5^{(1)} + 0.0753\varepsilon_6^{(0)}. \tag{4.6}$$

The remarkable notion above is that the ℓ_1 regularization only selects 4 of the available 14 features. Further, only tubes 4, 5, and 6 are included in the final model making tubes 1, 2, 3, and 7 non informative from the diagnosis point of view.

Based on the absolute values of the logistic regression coefficients in Equation 4.6, tube number 4 (see corresponding markers from table 4.4) seems to be the most important one. Also tube 5 has a substantial effect on the predictor output but only when comparing against AML-positive ($c = 1$) patients. Tube number 6 has a similar effect on the healthy patients. However, relatively small absolute coefficient value makes tube 6's contribution rather insignificant. Notice that comparing absolute coefficient values makes sense because the similarity scores $\varepsilon_i^{(c)}$ are normalized.

The full regularization path of the above case is shown in Figure 4.9. The final model indicated by the black dashed line is chosen by minimizing 10-fold CV error. The EDF distance features from tube 4 are the

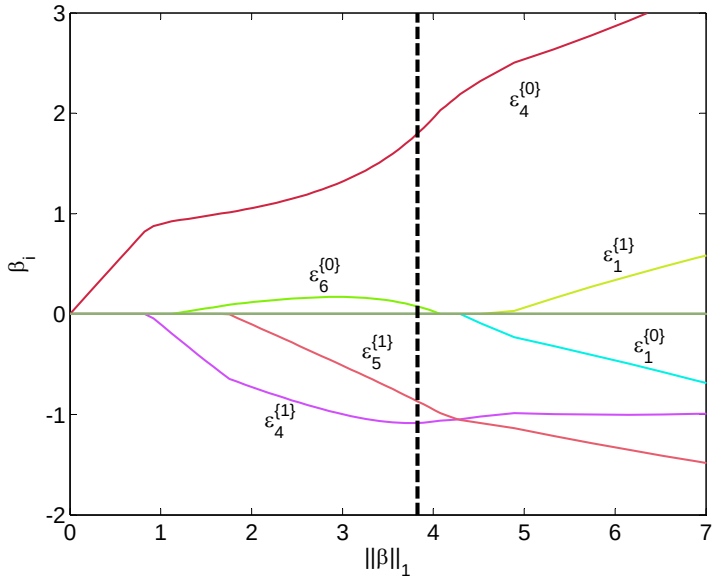


Figure 4.9: Regularization path of the ℓ_1 regularized logistic regression model with EDF MSE features. Coefficient values have been plotted with respect to the ℓ_1 norm of the coefficient vector. The black dashed line shows the final model chosen by CV.

first ones taken into the model when the regularization is gradually decreased (i.e., $||\boldsymbol{\beta}||_1$ increases). This further emphasizes the importance of tube 4. Interestingly, $\varepsilon_6^{\{0\}}$ is removed from the model right after the point of minimum CV error suggesting that similar performance could be attained by using only tubes 4 and 5. Notice, how the features from tube 1 approximately differ only by their sign. Further, the signs seem to be the wrong way around: smaller distance to the healthy population contributes towards classifying the patient as AML positive and the vice versa. This is probably due to overlearning.

Reference Methods

In the DREAM6 challenge, other good performing prediction models were provided by Vilar[2] and Biehl *et al.*[3] [102]. The basic idea in Vilar's predictor is similar to the previously introduced method: First, the distributions of data from AML-positive and healthy training pa-

[2] `http://www.ehu.es/biologiacomputacional/team21_vilar`
[3] `http://www.the-dream-project.org/story/code`

tients are compared against that of the tested patient for each tube. Second, resulting distribution similarity scores are aggregated in a logistic regression model to derive an AML confidence score between 0 and 1.

Vilar's method doesn't apply explicit dimension reduction for the data to alleviate the problem of multidimensional density estimation. Instead, the probability density of the 7-dimensional tube data is approximated by using multiple lower dimensional densities. These densities are constructed such that only one of the fluorescence markers FL1, FL2, FL4, or FL5 is taken into account in each density estimate while fluorescence marker FL3 (which is always CD45 regardless of the tube) and forward and side scatters are present in all of the low dimensional densities. This results in 4 different 4-dimensional distributions instead of one 7-dimensional. In practice, these densities are approximated by histograms with 9 bins per dimension uniformly spaced in logarithmic scale between $10^{0.01}$ and $10^{0.91}$ for all but the side scatter for which the corresponding range is between $10^{2.01}$ and $10^{2.91}$. An important feature, found out by our validation tests, is that any sample outside the range of the histogram is considered an outlier and discarded.

The comparison of the distributions in Vilar's method is done by using Kullback-Leibler divergence [101], i.e., *relative entropy*. The entropies from each tube are combined by simply summing together the relative entropies with the AML-positive population and subtracting the entropies with healthy population. This total entropy is mapped with the logistic function to get the final AML confidence score. In terms of logistic regression classification as viewed in this thesis, this means that the logistic regression model coefficients are not estimated from training data but fixed such that the relative entropies with AML-positive populations and healthy populations have coefficients equal to 1 and -1, respectively. Further, the bias term β_0 equals to zero.

Biehl's method differs from the proposed one and Vilar's by not trying to estimate the full densities of the data. Instead, six statistics; mean, standard deviation, skewness, kurtosis, median, and interquantile range, are calculated for each marker and used as features. For classification, Biehl uses *Generalized Matrix Relevance Learning Vector Quantization*. See [102] for details.

Sparse Logistic Regression with Simplified Features

In our recent article [18], we have shown that a similar classification performance to the introduced logistic regression model with EDF MSE features, and to the methods by Vilar and Biehl *et al.* can be achieved by simply using mean values of each marker directly as features in the logistic regression model. That is, the marker intensities in Table 4.4 along with the forward and side scatter values are averaged over the events in each tube creating 49 features. These features are then used for training an ℓ_1 regularized logistic regression model.

From model analysis point of view, the difference in using marker means as features compared to the previously proposed EDF MSE features is that the sparsity promoting regularization now selects individual markers instead of complete tubes when deciding the relevance of each feature. In the best case scenario, majority of the normally conducted flow cytometry measurements can be completely omitted when running the diagnosis. This is studied in the forthcoming experiment section along with comparisons of classification performance with other prediction models.

4.3.3 Experiments

As the first experiment, the coefficients of the logistic regression model with simple marker mean features are studied. The training is done with the entire DREAM6 AML challenge data set and ℓ_1 regularization is used in coefficient estimation. The resulting coefficient values corresponding to each feature are shown in Figure 4.10. It turns out that only 17 of the 49 variables are selected into the model. About half of the contribution in terms of ℓ_1 norm of the coefficient vector comes from the four most significant markers, which are CD34-PC5 from tube 6, side scatter from tube 5, and CD16-PC5 and CD13-PE from tube 4. Notice that comparing absolute coefficient values makes sense because the features have been normalized to zero mean and unit variance.

Figure 4.11 shows the regularization path (a) and the corresponding CV error (b) of the above experiment with respect to the ℓ_1 norm of the coefficient vector $\boldsymbol{\beta}$. The vertical dashed lines show the point where the final model has been selected. This point has been chosen by finding the minimum CV error and then using the one-standard-error rule, i.e., by picking the simplest model whose CV error is at most one standard error away from the minimum CV error.

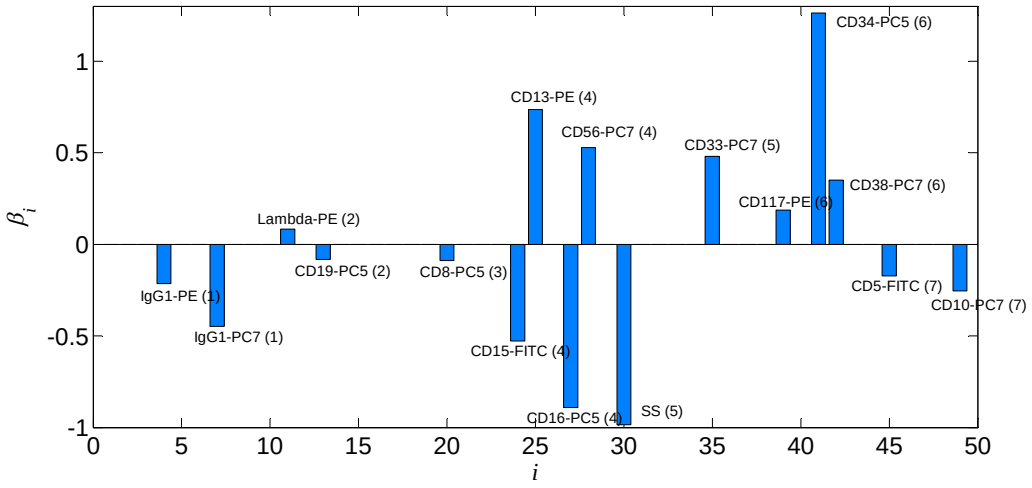
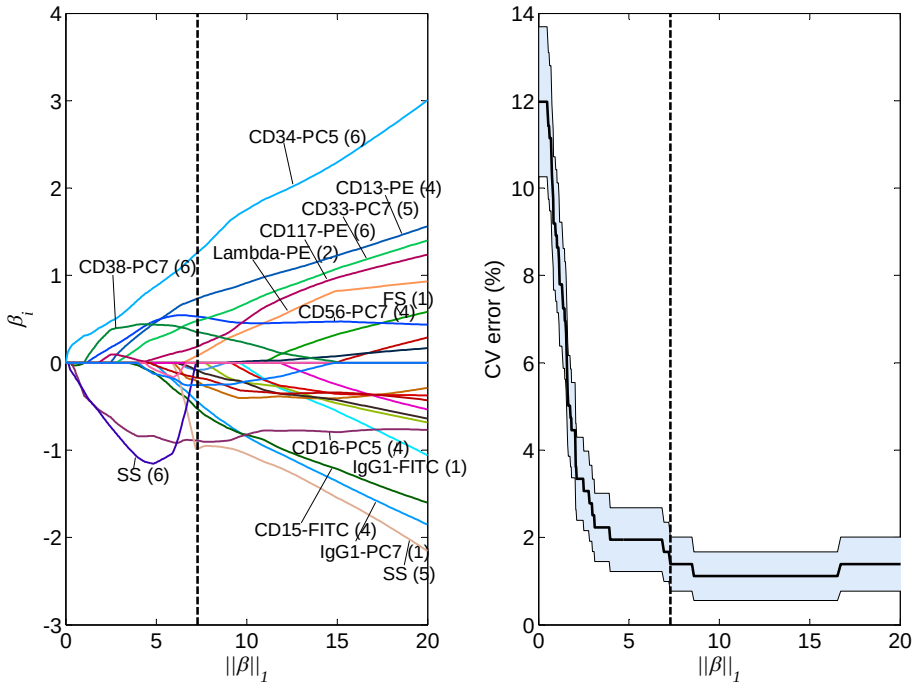


Figure 4.10: Coefficient values of the logistic regression classifier trained by using ℓ_1 regularization and sample means of the marker intensities as features.

The regularization path in Figure 4.11(a) shows that, e.g., the coefficient corresponding to the fluorescent marker CD34-PC5 gets rather high values throughout the path. Also side scatter (SS) and CD16-PC5 behave similarly, which emphasizes their importance in the classification task. Marker CD38-PC7 is picked into the model in an early stage but its importance decreases as more features enter the model. Interestingly, the side scatter of tube 6 is replaced by the side scatter from tube 5 just before the point of the selected model.

The above analysis of the most important features supports the discussion of Biehl *et al.* [102], where seven markers were recognized as the key features: forward scatter, side scatter, CD15-FITC, CD117-PE, CD16-PC5, CD34-PC5, and CD10-PC7. All these appear in our final model (Figure 4.10) or on the regularization path (Figure 4.11) before the selected model. An exception is made by the forward scatter measure. Missing forward scatter makes sense because in [102] they show that the predictive power of the forward scatter on linear scale relies on its higher moments, especially standard deviation and skew, rather than on the distribution average, which is used in our model.

In the model with marker means as features, scatters and CD45 intensities were not combined over the different tubes but the mean values from each tube were taken as individual features. Similar to that in Biehl's method [102], also pooling of these variables was tested. However, this decreased the validation performance. To our knowledge,



(a) (b)

Figure 4.11: (a) Regularization path showing the evolution of the model coefficients with respect to the ℓ_1 norm of the coefficient vector. The labels show the names of the relevant features (tube number in parentheses). (b) 10-fold CV error with standard error bounds. The dashed line shows the model selected by using the one-standard-error rule.

the intensity values of each marker should be independent of the tube they are measured from. One explanation for the above phenomenon is that there is some kind of overlearning occurring when not combining the corresponding markers from separate tubes.

As the second experiment, in order to evaluate the classification performance of our model, we have run a 10-fold CV test. Five different approaches have been compared:

- **Mean/LR(-LASSO).** This is our ℓ_1 regularized logistic regression model with average marker intensities as features as described in the previous section.
- **EDF-MSE/LR(-LASSO).** This is our original method submitted to the DREAM6 AML challenge [91] as described in the previous section.

- **Mean/LDA.** This is the same as the first method but uses LDA classifier instead of logistic regression.
- **Vilar.** This is the method by Vilar as described in the previous section.
- **Biehl *et al.*** This is the method by Biehl *et al.* [102] as described in the previous section.

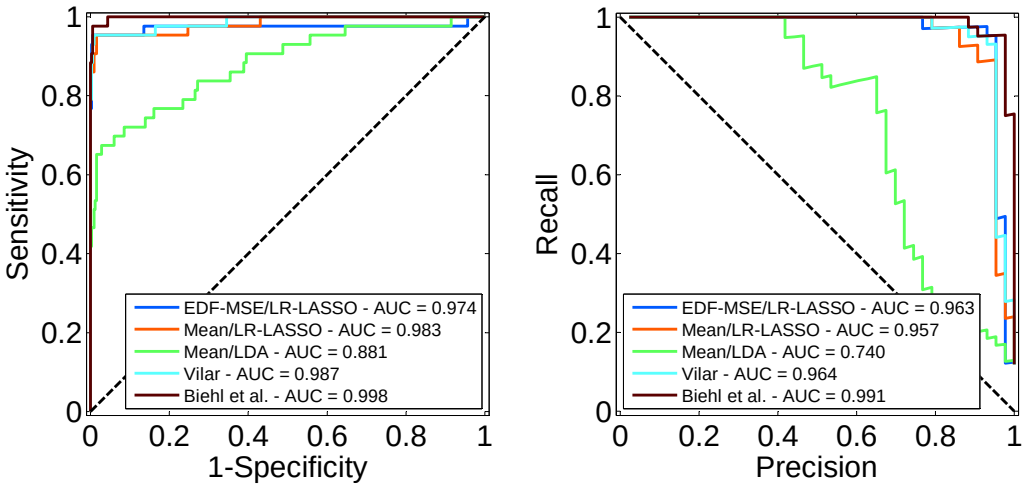


Figure 4.12: ROC (left) and PR (right) curves of a 10-fold CV test. The legend shows the AUC values for different predictors. Except for LDA (green line), the differences in AUC values are statistically insignificant.

The ROC and PR curves of the 10-fold CV test are shown in Figure 4.12. AUC values for each predictor given in the legend start to saturate close to one. The highest scoring method is the one by Biehl *et al.* However, according to the AUC analysis tool StAR [103], only LDA with mean features (AUC = 0.88) differs from the other methods (AUC ≈ 0.98) with a statistical significance. The StAR tool is based on a two-sided Mann-Whitney test with a null hypothesis that two AUC values are the same. A confidence level of $\alpha = 0.05$ was used in the significance test. Exact p-values for comparison of any two methods are given in Table 4.5.

As a conclusion of the comparison between different classification methods for AML diagnosis, the key benefit in the proposed predictor model is that, in the training phase, the ℓ_1 regularized logistic re-

Table 4.5: The p-values of the significance test testing the difference of the AUC ROC values between each pair of prediction methods.

Method	Biehl *et al.*	Vilar	Mean/ LR	EDF-MSE/ LR	Mean/LDA
Biehl *et al.*	-	0.1598	0.1537	0.2509	0.0003
Vilar	0.1598	-	0.6809	0.3486	0.0004
Mean/LR	0.1537	0.6809	-	0.6509	0.0019
EDF-MSE/LR	0.2509	0.3486	0.6509	-	0.0040
Mean/LDA	0.0003	0.0004	0.0019	0.0040	-

gression model automatically estimates the relevance of each flow cytometry tube or marker. Only a subset of the available features (3 of the 7 tubes with the EDF MSE features or 17 of the 49 markers with the mean intensity features) are selected still reaching a performance equivalent to the alternative methods. Thus, in a testing phase, only a fraction of the usual flow cytometry measurements are needed. Another beneficial feature in the proposed model is that it has virtually no parameters that would need manual tuning. Instead, the regularization parameter λ is automatically chosen by CV or BEE and the number of EDF bins K is, by construction, robust against changes in the data. This is in contrast, e.g., with Vilar's algorithm where even a small change in the dimensions of the density histogram was noticed to result in deterioration of the CV performance by several percentage units.

Experiments show that all the studied methods are almost perfect in terms of accuracy. In this case, the simplest solution should be favored as generally suggested by several studies (see, e.g., [104, 5]). Hence, due to its simplicity, it is reasonable to believe that the proposed use of regularized logistic regression has the best generalization for future samples also in this application case.

4.4 Mind Reading from MEG Data

In the next application case, elastic net penalized logistic regression is applied in the field of computational neuroscience, namely, in the classification and analysis of *magnetoencephalography* (MEG) measurements. The specific task is to determine *the type of the movie that a test person is watching*. There are five different predetermined movie types available. Thus, the task can be formulated as a multiclass classification problem.

The classification model and the results presented in this section are based on the submission to *Mind reading from MEG* challenge organized in conjunction with the International Conference on Artificial Neural Networks (ICANN 2011)[4]. With the originally submitted solution, a 68 % classification accuracy was achieved on the challenge test data, which was enough to win the challenge with a clear margin to the second best submission [105]. Recently, we have further improved the classifier and run tests to show the superiority of the logistic regression over other common classifiers in this application [21, 22].

The forthcoming sections are organized such that Section 4.4.1 gives some background on using machine learning in neuroimaging, in Section 4.4.2, a brief introduction is given to the data set used in the experiments, Section 4.4.3 explains how logistic regression can be applied with MEG data, and, finally, in Section 4.4.4, we run some experiments.

4.4.1 Background

During the recent years, supervised classification has become increasingly important in analyzing functional neuroimaging data [106] in *functional magnetic resonance imaging* (fMRI) [107] as well as in *electroencephalography* (EEG) and MEG (for reviews, see, e.g., [108, 109]). In brain research, classification can be used to answer the questions *is there* information about a variable of interest (pattern discrimination), *where* is the information (pattern localization) and *how* is the information encoded (pattern characterization) as explained in more detail in [109] and [110].

The use of machine learning on EEG and MEG data has concentrated on applications of *brain computer interfaces* (BCI) for which there a large amount of literature exists [111, 112]. The majority of studies in BCIs have focused on EEG with relatively few channels. For example, Van Gerven *et al.* [113] used regularized logistic regression to classify imagined movements of right or left hand based on EEG data from 16 channels. Perhaps more relevant to the present work is [114], where regularized logistic regression was applied for two different problems with 64-channel EEG data: 1) two-category self-paced finger tapping task from the BCI Competition 2003 and 2) a P300 speller system task from the BCI competition III, which is a 36-category classification problem. In the BCI-IV competition[5], one task was the classi-

[4] `http://www.cis.hut.fi/icann2011/mindreading.php`

[5] `http://www.bbci.de/competition/iv/index.html`

fication of the direction of wrist movements based on 10-channel MEG data [115].

Other studies focusing on the decoding of MEG data include Zhdanov *et al.* [116], who applied regularized linear discriminant analysis to MEG signals recorded while subjects were presented with images from two different categories (faces and houses). Chan *et al.* [117] applied an SVM classifier to decode the data that was recorded using simultaneous scalp EEG and MEG while the subjects were performing auditory and visual versions of a language task. Rieger *et al.* [118] applied an SVM to test whether it is possible to predict the recognition of briefly presented natural scenes from single trial MEG-recordings of brain activity and to investigate the properties of the brain activity that is predictive of later recognition. Besserve *et al.* [119] applied an SVM to classify between MEG data recorded during a visuomotor task and resting condition.

There are some key differences between typical BCI and our "Mind reading from MEG" decoding applications as laid out by Zhdanov *et al.* [116]. Perhaps most importantly, the dimension of input data is much higher in MEG than that of EEG typically used in BCI applications and the number of samples is much smaller. Also the behavioral paradigm is much more complex here. Moreover, all the above cited uses of supervised classifiers in MEG [116, 117, 118, 119] differ from the prediction task in the ICANN competition in that they were based on strictly controlled behavioral paradigm and the knowledge about the paradigm was often applied in the feature extraction. Moreover, all except Chan *et al.* [117] considered only a binary classification problem. The elastic net has been used in neuroimaging with fMRI data sets in the context of classification [120, 107] and regression [121] problems, but not with MEG data and, in the classification setting, not in a naturalistic behavioral paradigm like movie watching studied in this thesis.

4.4.2 Data from ICANN 2011 Mind Reading Challenge

MEG is used for measuring brain activity via magnetic field created by electric currents occurring in the person's brain. In our experiments, we use MEG data released in the ICANN 2011 mind reading competition. The data set is publicly available at the ICANN 2011 website[6].

[6] `http://www.cis.hut.fi/icann2011/meg/measurements.html`

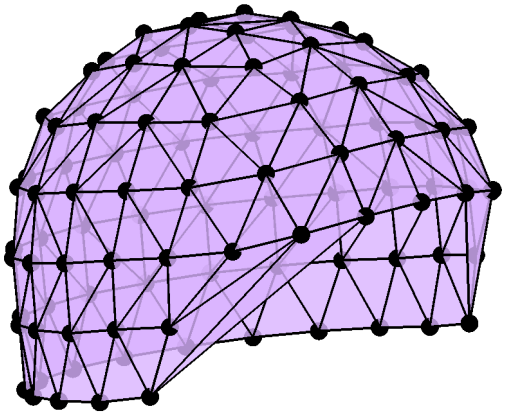


Figure 4.13: Sensor locations in the MEG data used in the ICANN mind reading challenge. There are 102 sensor locations denoted by the black dots around the test subject's skull. Each location has two gradiometer channels producing MEG measurements.

The provided measurements in the ICANN data set consist of two gradiometer channels for each of the 102 sensors that are organized around the test subject's head. Each channel produces MEG data with 200 Hz sampling rate. The sensor locations have been illustrated in Figure 4.13.

The signals have been recorded from a single test person watching five different video stimuli without audio. The five different movie types are as follows:

1. **Artificial.** Animated shapes or text
2. **Nature.** Clips of a nature documentary
3. **Football.** Clips of a soccer match
4. **Bean.** Part from the comedy series "Mr. Bean"
5. **Chaplin.** Part from a Chaplin movie

The MEG measurements are recorded on two separate days such that the same set of video stimuli was shown to a test person on both days. Stimuli labeled as either Artificial, Nature, or Football (short clips) were presented as randomly ordered sequences of length 6 – 26 s with a 5 s rest period between the clips, while Bean and Chaplin (movies) were presented in two consecutive clips of approximately 10 minutes.

The measurements are cut into one-second epochs that are in a randomized order such that the complete signal cannot be reconstructed based on the individual signal epochs. There are a total of 1380 epochs of data, 677 epochs from the first day and 703 epochs from the second day. In the challenge, 50 of the 703 second day epochs were left as a secret test set and the rest including the entire first day was provided for the competitors for training. The data is approximately class-balanced, i.e., there are about the same number of measurements from each movie type. A more detailed description of the data can be found in the ICANN 2011 challenge report by Klami *et al.* [105].

4.4.3 Using Logistic Regression for Recognition of Viewed Movie Type

The data set has 1380 one second samples recorded from 204 channels with 200 Hz sampling rate. Thus, the dimensionality of the data ($204 \cdot 200\,\text{Hz} \cdot 1\,\text{s} = 40800$) is significantly higher than the number of samples (1380). In addition, the features are highly redundant because of time correlation and the overlap between the measurements from neighboring MEG sensors.

As a solution for the ill-posedness of the problem, we first reduce the dimensionality of the data by extracting only a couple of statistical features from each time frame. After this, elastic net regularized logistic regression (Equation 2.14) is applied for automatic selection between the relevant MEG sensors for each movie type. The `glmnet` package [26] is used for coefficient estimation. The motivation in using elastic net regularization is that the ℓ_1 penalization is assumed to be able to point out the relevant features and the relevant areas of the brain while ℓ_2 regularization is used for shrinking and averaging of data from highly correlating neighboring sensors.

A logistic regression classifier with 5 classes and 40800-dimensional data has $5 \cdot 40800 + 5 = 204005$ parameters (one per feature plus the bias term for each class) to estimate. With 1380 samples, this seems a lot. Indeed, in [22] we have shown that even though the elastic net framework is superior to other feature selection strategies such as *forward and backward floating selection* [74] and *simulated annealing* [122], manual expert design of feature sets should still complement the automatic selection in high dimensional cases in order to avoid overfitting.

In [21] we have shown that reducing the 200 sample MEG sequence into two statistical quantities, *mean* and *detrended standard deviation*,

gives good results. In detrending, a least squares fitted line is subtracted from the MEG signal $s_k(n)$ with channels $k = 1, \ldots, 204$ and samples $n = 1, \ldots, 200$ in order to get the detrended signals

$$\tilde{s}_k(n) = s_k(n) - a_k(n - 100.5) - b_k. \tag{4.7}$$

Here, the slopes a_k and intercepts b_k are obtained by minimizing the residual sum $\sum_{n=1}^{200} \tilde{s}_k(n)$. The value 100.5 subtracted from n is selected as the mid-point of time indices $1, \ldots, 200$. With this particular value the intercept b_k becomes equal to the sample mean.

Detrending is done in order to remove the effect of slowly changing nonzero bias in the MEG recordings due to irrelevant fluctuations in brain activity. Similar effect is attained with the usual preprocessing of several MEG studies, which use a bandpass filter to remove low frequency components, e.g., below 5 Hz. Due to the boundary problems that filtering operations encounter with short signals such as in this case, detrending is a better option compared to frequency domain filtering.

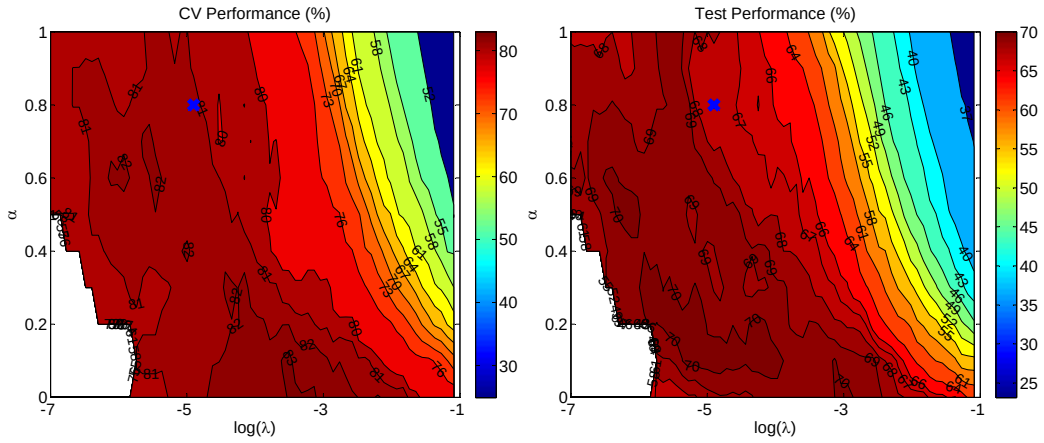
After feature extraction, the data is reduced into $2 \cdot 204 = 408$ dimensions and the number of estimated logistic regression model parameters becomes $5 \cdot 408 + 5 = 2045$. The problem is still overdetermined but this is taken care by the elastic net regularization.

As discussed in Section 2.3.2, elastic net regularization needs two parameters to operate: the conventional regularization parameter λ and the mixing parameter α that control the amount of total regularization and the proportion of ℓ_1 and ℓ_2 regularizations as defined in Equation 2.14, respectively. Figure 4.14 shows a performance plot for different values tested for α and λ.

Figure 4.14a shows a 5-fold CV performance for the original ICANN training data while 4.14b shows the corresponding performance for the secret test data. In the 5-fold CV, the second day data is given a larger weight compared to the first day data in order to better estimate the performance with the secret test data known to originate from the second day alone (see [21] for more details about the weighting).

The plots in Figure 4.14a are very similar in shape and the top is flat for a wide range of α and λ. This indicates that the model is robust to the choice of parameters. In particular, the algorithm is insensitive to the selection of α: For any α, there exists a value of λ within 1.7 %-points from the absolute optimum for the secret test data. Thus, the performance is insensitive to slightly erroneous parameter settings.

Using CV for assessing the performance of an elastic net regularized logistic regression model easily becomes computationally heavy



(a) (b)

Figure 4.14: (a) The cross-validated performance for training data with different values for the parameters λ and α. (b) The true performance for the secret test data with different λ and α. The blue cross denotes the parameters of our ICANN submission.

if, inside each CV loop, α and λ are selected by using 2-d CV like the one shown in Figure 4.14a. Instead, the preferred method is to fix α and only select λ automatically with 1-d CV similar to that when using plain ℓ_1 regularization. In fact, instead of 2-d CV, a better CV performance is achieved in this case by setting $\alpha = 0.8$ and selecting λ with CV. When training with the entire training data, this results in the model denoted by the blue crosses in Figure 4.14.

4.4.4 Experiments

When the features used in the logistic regression classifier are normalized to zero mean and unit variance, the absolute values of the corresponding coefficients can be used for indicating the relative importance of the features. With MEG data, we are especially interested in the importance of each feature from the viewpoint of the location of the corresponding sensor because this enables us to create connections between different brain areas and different movies types.

The topographic plots in Figure 4.15 illustrate the absolute values of the model coefficients when using all the ICANN data (see Section 4.4.2) for training and mean and detrended standard deviation as features as described in Section 4.4.3. The locations of the 102 gra-

diometer sensors are marked by black dots and the color indicates the sum of the absolute values of all the four coefficients per each sensor location.

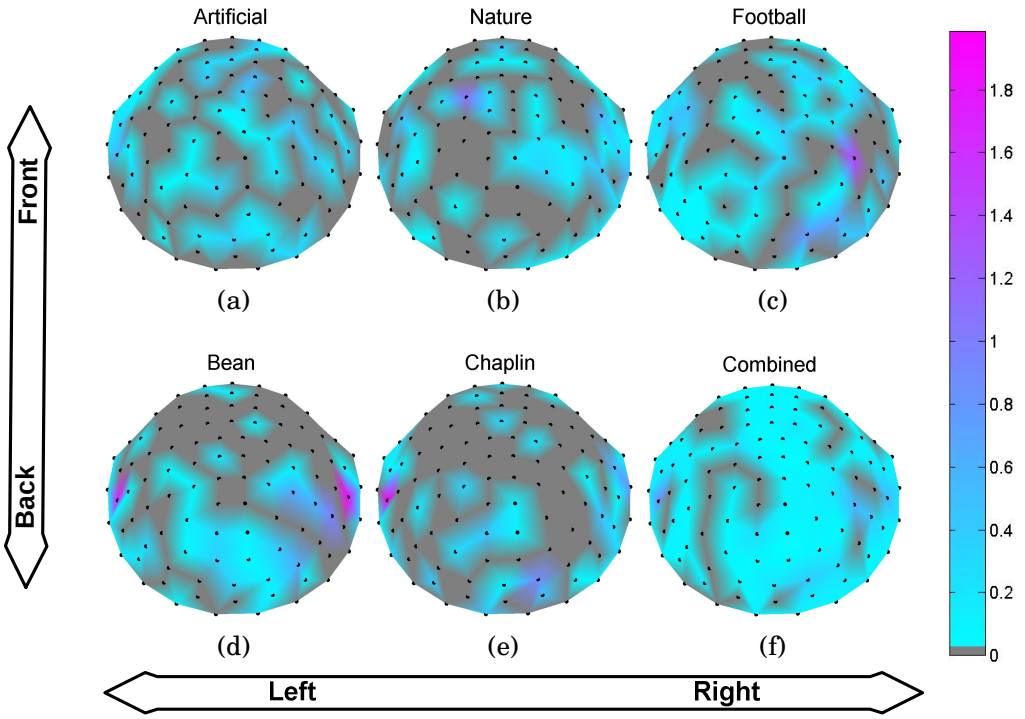


Figure 4.15: Illustration of the sum of the model coefficient magnitudes for each class separately (a – e) and for all classes combined (f). The features have been normalized in order for the coefficient values to be comparable.

Figures 4.15a – 4.15e correspond to each class and the plotted coefficients are those corresponding to each component of the symmetric multinomial logistic regression model as defined in Equation 2.3. Figure 4.15f shows the average over coefficients for all classes. Gray areas in the topographic plots correspond to coefficient values exactly equal to zero, and are thus not used by the classifier at all. MATLAB toolbox FieldTrip [123] was used in drawing the topographic plots.

Because our approach is data-driven, care must be taken for not to over-interpret the results shown in the topographic plots. Moreover, it has recently been shown that the visualization and interpretation of a model depends on the regularization although the predictive performance may seem stable over a range of regularization parameter

values [107]. In [21], however, we provide some possible interpretations of the plots and show that a neurologically sensible explanation can be given to many of the features of the distribution of the coefficient magnitudes on the topographical plots.

In addition to the multi-class classification case, it is interesting to study the problem of classifying between movies with a storyline (Bean, Chaplin) and short video clips with no storyline (Artificial, Nature, and Football). This can be done by combining classifier outputs corresponding to storyline classes as one class and those corresponding to short clips as another class as was done in the ICANN challenge proceedings [105]. In this case, the classification performance of our classifier on the secret test data was 89.7 %. However, a 5-class linear classifier applied to a 2-class problem in the above manner is not a linear classifier, but a classifier with a piece-wise linear decision surface. Hence, there is a risk that the classifier won't be regularized enough to achieve the best possible generalization performance for the 2-class case.

As a result of training a binary classifier particularly for the 2-class task, the classification performance on the secret test data increases from 89.7 % to 96.5 %. This indicates that a simple (linear) classifier is preferred over a more complex classifier (piecewise linear). For simplicity, the same initial feature set of mean and detrended standard deviation and the elastic net mixing parameter $\alpha = 0.8$ as in the multinomial case is used here without further optimization.

Figure 4.16 illustrates the absolute values of the model coefficients after feature standardization. As in Figure 4.15, the black nodes mark the locations of the sensors, which are a source of four model coefficients each. Due to the traditional logistic regression model used in this case (and not the symmetric one) we only have one set of coefficients $\{\beta_0, \boldsymbol{\beta}\}$ instead of one set per class. Using the symmetric model in the 2-class case would result in two identical plots because the coefficients between the classes only differ by their signs. The figure shows the areas that the classifier uses to discriminate between the two classes. As expected, visual comparison indicates some similarities between Figure 4.16 and and corresponding areas in the multi-class case in Figure 4.15f. Further analysis from a neuroscientific point of view is excluded from this thesis but has been given in [21] for the interested.

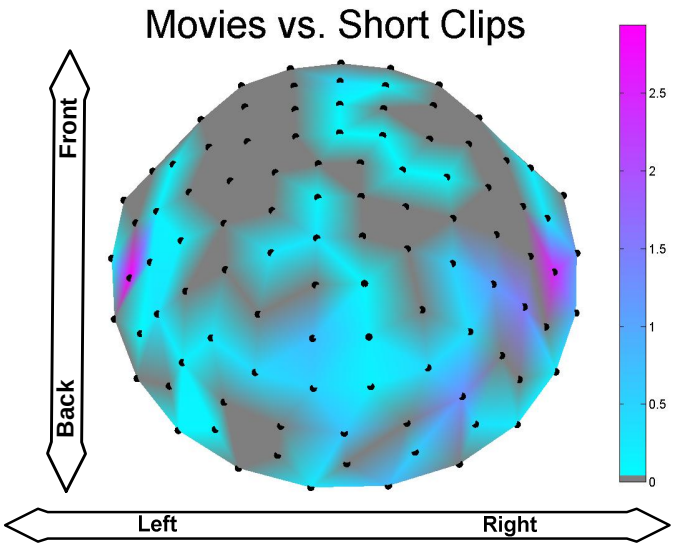


Figure 4.16: The coefficient magnitudes of a binary classifier trained to distinct movies with a storyline from short clips.

4.5 Discussion

The previous sections have introduced several real life applications where logistic regression with sparsity promoting coefficient estimation has been used. Despite its simple linear structure, the logistic regression classifier has proven to outperform many of the more sophisticated methods in classification accuracy. This indicates that many of the real life problems are linearly separable, at least with proper feature engineering.

In practical applications, not only is a good classification performance an essential feature of the method used for classification. What also accounts is computational efficiency and the interpretability of the model. With the ℓ_1 regularized logistic regression model, these are guaranteed by the linearity of the model and recent improvements in the training algorithms.

Sparse coefficient estimation has proven extremely useful in many applications. It enables the embedding of automatic feature selection into model training removing the need for explicit feature selection algorithms that are often time consuming and don't necessarily take into account any classifier specific issues that might have an effect on a good choice of features. The image segmentation framework introduced in Section 4.1 shows that the feature selection property of the sparse lo-

gistic regression classifier is extremely versatile working in different application areas without special knowledge of the application or the need for major changes into the model, initial set of features, or the learning algorithms when changing from an application to another.

Chapter 5

Conclusions

This thesis has introduced several applications for sparse logistic regression classification. The results indicate that this simple linear classification model together with proper feature engineering and automated feature selection can perform extremely well with various types of data and applications. From the practical point of view, sparse logistic regression has shown to have several benefits over many other classification methods.

One of the main benefits shown for sparse logistic regression is that, with proper choice of the initial set of features, ℓ_1 regularized logistic regression classifiers can outperform other more complex models in many different applications. Further, regularization of the ℓ_1 norm of the coefficient vector results in a sophisticated way to do automated and computationally efficient feature selection optimized for the linear model. Using a simple linear structure of the classifier also helps to avoid overfitting and the coefficients of the linear model are easy to interpret, which is useful in several applications. Despite this simple linear structure, top performing results have been attained in real life classification problems, which indicates that these problems are linearly separable by nature.

From the practical point of view, only one (LASSO) or two (elastic net) parameters need to be set requiring little effort or expert skills in training and using the sparse logistic regression model. In addition, recent development in training algorithms allows efficient optimization of the ℓ_1 penalized likelihood function enabling feasible training times. Finally, the logistic regression model is versatile in the sense that it can be used both with real valued or categorical input data and with binary or multinomial class labels extending its applicability to many different applications.

The application cases where sparse logistic regression classification has successfully been applied in this thesis consist of diagnosis of leukemia from flow cytometry data, analysis and classification of brain activity from MEG data, and supervised image segmentation in two different application fields: object detection in electronics manufacturing and spot detection from cell images.

The main results and conclusions for each application case are as follows:

Image segmentation

- The proposed supervised segmentation framework was shown to outperform conventional methods both in spot detection from cell images and in object detection in inkjet printed electronics.
- Logistic regression classification with ℓ_1 regularization combined with MRF spatial prior was shown to give a better segmentation result compared to other commonly used classifiers and feature selection algorithms.
- The ℓ_1 regularization was shown to work as an efficient feature selector allowing the proposed image segmentation framework to work on multiple application fields without the need for excess parameter tuning or special expertise from the user.
- Additional cost, especially in cell segmentation, is that the new method uses training data while conventional methods are usually unsupervised. Our experience, however, is that the training step is rather painless and comparable to tuning of parameters in a corresponding unsupervised method.
- Especially in the inkjet printed electronics case, the new method was able to replace multiple processing steps in the image processing pipeline. This resulted in reduced computation time being valuable in a real time application such as computer vision aided inkjet printed electronics manufacturing.

Diagnosis of AML from flow cytometry data

- Using simple marker mean values as features in an ℓ_1 regularized logistic regression classifier was shown to produce statistically equal classification results with state-of-the-art methods.

- The automatic feature selection property revealed that only 17 of the usually measured 49 variables in flow cytometry are actually needed in making the diagnosis.
- Inspecting the coefficient values of the linear model enabled to sort the markers according to their relevance from the viewpoint of AML diagnosis. It was found out that the contribution of the three most significant fluorescence markers to the diagnosis decision is approximately 50 % in terms of absolute coefficient magnitudes after feature standardization.

Mind reading and analysis from MEG data

- Elastic net regularized logistic regression using detrended mean and standard deviation of the MEG signal as features was shown to outperform several other methods in a multi-class classification task.
- The feature selection property of the elastic net regularization was able to detect the relevant areas on the skull for each different movie type.
- Potentially interesting neuroscientific information can be inferred from the relevance of the sensor locations on the skull. However, one should notice that they do not necessarily correspond to the areas with neuroscientifically relevant brain activity.
- With MEG data, it was shown that the elastic net regularized logistic regression classifier is rather insensitive to the choice of the mixing parameter α.

In addition to experimenting with the logistic regression model, error estimation, especially from the viewpoint of automatic selection of the regularization parameter in the ℓ_1 penalized classification model, was studied. Results show that, with small sample size, the recently developed Bayesian error estimator [72, 73] outperforms traditionally used cross-validation methods both in accuracy and computation time. Generalization of the Bayesian error estimator for multinomial logistic regression is currently under investigation.

Related to model selection, it was shown that conventional methods for benchmarking and improving classification models by means of subsequent cross-validation runs can lead to unpredictably over-optimistic

results. The results are unpredictable in the sense that doing cross-validation by using information from a previous cross-validation run was, at least in a special case, shown to be more harmful than the classical mistake of inferring information from all the training data before the actual cross-validation loop. For instance, conventionally used sequential feature selection algorithms often use repeated cross-validation to assess the performance of subsequent feature combinations. However, the results indicate that even if the cross-validation performance improves in such a procedure, in reality, the generalizability of the feature set can actually start to decrease at some point.

While the brief experiment in Section 3.2.4 showed that subsequent cross-validation brings optimism into the validation performance, the amount of optimism was not quantified. Thus, we still do not know how harmful it exactly is, e.g., to do five subsequent test runs in a cross-validation test bench and parameter tuning in between in order to improve our classification model. This is a potential topic for future research.

Bibliography

[1] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *International Journal of Data Warehousing and Mining (IJDWM)*, vol. 3, no. 3, pp. 1–13, 2007.

[2] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. Wiley-Interscience, 2001.

[3] P. Simon, *Too Big to Ignore: The Business Case for Big Data*, ser. Wiley and SAS Business Series. Wiley, 2013.

[4] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006, vol. 4.

[5] D. J. Hand, "Classifier technology and the illusion of progress," *Statistical Science*, vol. 21, no. 1, pp. 1–14, 2006.

[6] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[7] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[8] S. Press and S. Wilson, "Choosing between logistic regression and discriminant analysis," *Journal of the American Statistical Association*, pp. 699–705, 1978.

[9] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. Springer, 2009.

[10] S. Haykin, *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[11] P. MacCullagh and J. A. Nelder, *Generalized linear models*. CRC press, 1989, vol. 37.

[12] J.-H. Xue and D. M. Titterington, "Comment on "On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes"," *Neural processing letters*, vol. 28, no. 3, pp. 169–187, 2008.

[13] B. Efron, "The efficiency of logistic regression compared to normal discriminant analysis," *Journal of the American Statistical Association*, vol. 70, no. 352, pp. 892–898, 1975.

[14] A. Y. Ng and M. I. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes," in *Advances in Neural Information Processing Systems 14: Proceedings of the 2001 Conference*, Z. G. Thomas Glen Dietterich, Suzanna Becker, Ed. MIT Press, 2002, pp. 841–848.

[15] T. J. O'Neill, "The general distribution of the error rate of a classification procedure with application to logistic regression discrimination," *Journal of the American Statistical Association*, vol. 75, no. 369, pp. 154–160, 1980.

[16] Y. D. Rubinstein, T. Hastie *et al.*, "Discriminative vs informative learning," in *KDD*, vol. 5, 1997, pp. 49–53.

[17] D. Titterington, G. Murray, L. Murray, D. Spiegelhalter, A. Skene, J. Habbema, and G. Gelpke, "Comparison of discrimination techniques applied to a complex data set of head injured patients," *Journal of the Royal Statistical Society. Series A (General)*, pp. 145–175, 1981.

[18] T. Manninen, H. Huttunen, P. Ruusuvuori, and M. Nykter, "Leukemia prediction using sparse logistic regression," *PLoS ONE*, vol. 8, no. 8, p. e72932, Aug. 2013.

[19] T. Manninen, V. Pekkanen, K. Rutanen, P. Ruusuvuori, R. Rönkkä, and H. Huttunen, "Alignment of individually adapted print patterns for ink jet printed electronics," *Journal of Imaging Science and Technology*, vol. 54, no. 5, p. 050306, 2010.

[20] P. Ruusuvuori, T. Manninen, and H. Huttunen, "Image segmentation using sparse logistic regression with spatial prior," in *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*, Aug. 2012, pp. 2253–2257.

[21] H. Huttunen, T. Manninen, J.-P. Kauppi, and J. Tohka, "Mind reading with regularized multinomial logistic regression," *Machine Vision and Applications*, vol. 24, no. 6, pp. 1311–1325, 2013.

[22] H. Huttunen, T. Manninen, and J. Tohka, "MEG mind reading: Strategies for feature selection," in *Proceedings of the Federated Computer Science Event 2012*, 2012, pp. 42–49.

[23] ——, "Bayesian error estimation and model selection in sparse logistic regression," in *2013 IEEE International Workshop on Machine Learning for Signal Processing, MLSP2013*, Southampton, UK, Sep. 2013.

[24] J. Cramer, "The origins of logistic regression," Tinbergen Institute, Tinbergen Institute Discussion Papers 02-119/4, Dec. 2002.

[25] G. Cawley and N. Talbot, "Gene selection in cancer classification using sparse logistic regression with Bayesian regularization," *Bioinformatics*, vol. 22, no. 19, pp. 2348–2355, 2006.

[26] J. H. Friedman, T. Hastie, and R. Tibshirani, "Regularization paths for generalized linear models via coordinate descent," *Journal of Statistical Software*, vol. 33, no. 1, pp. 1–22, 2010.

[27] J. E. Griffin and P. J. Brown, "Bayesian hyper-Lassos with non-convex penalization," *Australian & New Zealand Journal of Statistics*, vol. 53, no. 4, pp. 423–442, 2011.

[28] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005.

[29] A. Webb, *Statistical Pattern Recognition*, 2nd ed. John Wiley and Sons Ltd., 2002.

[30] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Mathematical Programming*, vol. 45, no. 1–3, pp. 503–528, 1989. [Online]. Available: http://dx.doi.org/10.1007/BF01589116

[31] P. J. Green, "Iteratively reweighted least squares for maximum likelihood estimation, and some robust and resistant alternatives," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 149–192, 1984.

[32] D. Rubin, "Iteratively reweighted least squares," *Encyclopedia of statistical sciences*, vol. 4, pp. 272–275, 1983.

[33] P. Peduzzi, J. Concato, E. Kemper, T. R. Holford, and A. R. Feinstein, "A simulation study of the number of events per variable in logistic regression analysis," *Journal of clinical epidemiology*, vol. 49, no. 12, pp. 1373–1379, 1996.

[34] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[35] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM journal on scientific computing*, vol. 20, no. 1, pp. 33–61, 1998.

[36] A. Hoerl and R. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, pp. 55–67, 1970.

[37] A. Y. Ng, "Feature selection, L_1 vs. L_2 regularization, and rotational invariance," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 78.

[38] B. Krishnapuram, L. Carin, M. Figueiredo, and A. Hartemink, "Sparse multinomial logistic regression: fast algorithms and generalization bounds," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 6, pp. 957–968, Jun. 2005.

[39] A. Genkin, D. Lewis, and D. Madigan, "Large-scale Bayesian logistic regression for text categorization," *Technometrics*, vol. 49, no. 3, pp. 291–304, 2007.

[40] S. Shevade and S. Keerthi, "A simple and efficient algorithm for gene selection using sparse logistic regression," *Bioinformatics*, vol. 19, no. 17, pp. 2246–2253, 2003.

[41] S. Raman and V. Roth, "Sparse Bayesian regression for grouped variables in generalized linear models," in *Pattern Recognition*, ser. Lecture Notes in Computer Science, J. Denzler, G. Notni, and H. Süße, Eds. Springer Berlin / Heidelberg, 2009, vol. 5748, pp. 242–251.

[42] D. F. Andrews and C. L. Mallows, "Scale mixtures of normal distributions," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 99–102, 1974.

[43] M. A. Figueiredo, "Adaptive sparseness for supervised learning," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, no. 9, pp. 1150–1159, 2003.

[44] C. Ambroise and G. McLachlan, "Selection bias in gene extraction on the basis of microarray gene-expression data," *Proceedings of the National Academy of Sciences*, vol. 99, no. 10, p. 6562, 2002.

[45] F. Caron and A. Doucet, "Sparse Bayesian nonparametric regression," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 88–95.

[46] T. Virtanen, R. Singh, and B. Raj, *Techniques for Noise Robustness in Automatic Speech Recognition*. Wiley, 2012.

[47] K. Binder, "Ising model," *Encyclopedia of Mathematics*, 2011, uRL: http://www.encyclopediaofmath.org/index.php?title=Ising_model&oldid=15668.

[48] S. Li, *Markov Random Field Modeling in Image Analysis*, 3rd ed., ser. Advances in Computer Vision and Pattern Recognition. Springer, 2009.

[49] E. Ising, "Beitrag zur theorie des ferromagnetismus," *Zeitschrift für Physik A Hadrons and Nuclei*, vol. 31, no. 1, pp. 253–258, 1925.

[50] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 6, no. 6, pp. 721–741, Nov. 1984.

[51] J. Borges, J. Bioucas-Dias, and A. Marcal, "Bayesian hyperspectral image segmentation with discriminative class learning," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 49, no. 6, pp. 2151–2164, Jun. 2011.

[52] D. M. Greig, B. T. Porteous, and A. H. Seheult, "Exact maximum a posteriori estimation for binary images," *Journal of the Royal Statistical Society.*, vol. 51, no. 2, pp. 271–279, 1989.

[53] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision,"

Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 26, no. 9, pp. 1124–1137, Sep. 2004.

[54] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing & Management*, vol. 45, no. 4, pp. 427–437, 2009.

[55] D. D. Lewis, "Evaluating text categorization i." in *HLT*, vol. 91, 1991, pp. 312–318.

[56] G. Tsoumakas and I. Vlahavas, "Random k-labelsets: An ensemble method for multilabel classification," in *Machine Learning: ECML 2007*. Springer, 2007, pp. 406–417.

[57] Y. Yang, "An evaluation of statistical approaches to text categorization," *Information Retrieval*, vol. 1, no. 1–2, pp. 69–90, 1999.

[58] C. E. Metz, "Basic principles of ROC analysis," *Seminars in Nuclear Medicine*, vol. 8, no. 4, pp. 283–298, 1978.

[59] T. Fawcett, "An introduction to ROC analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[60] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[61] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (ROC) curve," *Radiology*, vol. 143, no. 1, pp. 29–36, Apr. 1982.

[62] D. J. Hand and R. J. Till, "A simple generalisation of the area under the ROC curve for multiple class classification problems," *Machine Learning*, vol. 45, no. 2, pp. 171–186, 2001.

[63] T. C. Landgrebe and R. P. Duin, "Approximating the multiclass ROC by pairwise analysis," *Pattern recognition letters*, vol. 28, no. 13, pp. 1747–1758, 2007.

[64] C. Sima, U. Braga-Neto, and E. R. Dougherty, "Superior feature-set ranking for small samples using bolstered error estimation," *Bioinformatics*, vol. 21, no. 7, pp. 1046–1054, 2005.

[65] V. N. Vapnik, *Statistical learning theory*. Wiley, 1998.

[66] B. Efron, "Bootstrap methods: another look at the jackknife," *The annals of Statistics*, pp. 1–26, 1979.

[67] ——, "Estimating the error rate of a prediction rule: Improvement on cross-validation," *Journal of the American Statistical Association*, vol. 78, no. 382, pp. 316–331, 1983.

[68] U. M. Braga-Neto and E. R. Dougherty, "Is cross-validation valid for small-sample microarray classification?" *Bioinformatics*, vol. 20, no. 3, pp. 374–380, 2004.

[69] H. Akaike, "A new look at the statistical model identification," *Automatic Control, IEEE Transactions on*, vol. 19, no. 6, pp. 716–723, 1974.

[70] G. Schwarz, "Estimating the dimension of a model," *The annals of statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[71] J. Chen and Z. Chen, "Extended BIC for small-n-large-P sparse GLM," *Statistica Sinica*, vol. 22, no. 2, p. 555, 2012.

[72] L. Dalton and E. Dougherty, "Bayesian minimum mean-square error estimation for classification error — part I: Definition and the Bayesian MMSE error estimator for discrete classification," *Signal Processing, IEEE Transactions on*, vol. 59, no. 1, pp. 115–129, 2011.

[73] ——, "Bayesian minimum mean-square error estimation for classification error — part II: Linear classification of Gaussian models," *Signal Processing, IEEE Transactions on*, vol. 59, no. 1, pp. 130–144, 2011.

[74] P. Pudil, J. Novovičová, and J. Kittler, "Floating search methods in feature selection," *Pattern Recogn. Lett.*, vol. 15, no. 11, pp. 1119–1125, Nov. 1994.

[75] J.-C. Olivo-Marin, "Extraction of spots in biological images using multiscale products," *Pattern Recognition*, vol. 35, no. 9, pp. 1989–1996, 2002.

[76] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[77] J. Li, "Discriminative image segmentation: Applications to hyperspectral data," Ph.D. dissertation, Universidade Tecnica De Lisboa, 2011.

[78] P. Paclik, R. Duin, G. van Kempen, and R. Kohlus, "Supervised segmentation of textures in backscatter images," in *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, vol. 2, 2002, pp. 490–493.

[79] S. Kumar and M. Hebert, "Discriminative random fields," *International Journal of Computer Vision*, vol. 68, no. 2, pp. 179–201, 2006.

[80] M. Farhan, P. Ruusuvuori, M. Emmenlauer, P. Rämö, C. Dehio, and O. Yli-Harja, "Multi-scale Gaussian representation and outline-learning based cell image segmentation," *BMC Bioinformatics*, vol. 14, no. 10, pp. 1–14, 2013.

[81] P. Ruusuvuori, T. Äijö, S. Chowdhury, C. Garmendia-Torres, J. Selinummi, M. Birbaumer, A. Dudley, L. Pelkmans, and O. Yli-Harja, "Evaluation of methods for detection of fluorescence labeled subcellular objects in microscope images." *BMC Bioinformatics*, vol. 11, p. 248, 2010.

[82] P. Ruusuvuori, A. Lehmussola, J. Selinummi, T. Rajala, H. Huttunen, and O. Yli-Harja, "Benchmark set of synthetic images for validating cell image analysis algorithms," in *Proceedings of the 16th European Signal Processing Conference*. EUSIPCO, 2008.

[83] J. Miettinen, V. Pekkanen, K. Kaija, P. Mansikkamäki, J. Mäntysalo, M. Mäntysalo, J. Niittynen, J. Pekkanen, T. Saviauk, and R. Rönkkä, "Inkjet printed system-in-package design and manufacturing," *Elsevier Microelectr. J.*, 2008.

[84] K. Kaija, V. Pekkanen, M. Mäntysalo, and P. Mansikkamäki, "Controlling warpage of molded package for inkjet manufacturing," *Microelectronic Engineering*, vol. 85, no. 3, pp. 518–526, 2008.

[85] M. Moganti, F. Ercal, C. Dagli, and S. Tsunekawa, "Automatic PCB inspection algorithms: A survey," *Computer Vision and Image Processing*, vol. 63, pp. 287–313, Mar. 1996.

[86] J. C. Gower and G. B. Dijksterhuis, *Procrustes problems*. Oxford University Press Oxford, 2004, vol. 3.

[87] T. Manninen, R. Rönkkä, and H. Huttunen, "Point pattern matching for 2-D point sets with regular structure," in *Image Analysis*, ser. Lecture Notes in Computer Science, A. Heyden and F. Kahl, Eds. Springer Berlin / Heidelberg, 2011, vol. 6688, pp. 156–165.

[88] H. Huttunen, T. Manninen, K. Rutanen, P. Ruusuvuori, R. Mäkinen, and R. Rönkkä, "Dynamic correction of interconnections in printed electronics manufacturing," in *International conference on digital printing technologies, NIP25*, Louisville, Kentucky, USA, Sep. 2009, pp. 589–592.

[89] H. Huttunen, P. Ruusuvuori, T. Manninen, K. Rutanen, R. Rönkkä, and A. Visa, "Object detection for dynamic adaptation of interconnections in inkjet printed electronics," in *15th IEEE International Conference on Image Processing, ICIP 2008*, San Diego, California, USA, Oct. 2008, pp. 2364–2367.

[90] M. Kamel and A. Zhao, "Extraction of binary character/graphics images from grayscale document images," *CVGIP: Graphical Models and Image Processing*, vol. 55, no. 3, pp. 203–217, 1993.

[91] T. Manninen, H. Huttunen, P. Ruusuvuori, and M. Nykter, "Logistic regression for AML prediction," in *Dialogue for Reverse Engineering Assessments and Methods, DREAM6*, Barcelona, Spain, Oct. 2011.

[92] N. Aghaeepour, G. Finak, The Flowcap Consortium, The DREAM Consortium, H. Hoos, T. R. Mosmann, R. Gottardo, R. R. Brinkman, and R. H. Scheuermann, "Critical assessment of automated flow cytometry data analysis techniques," *Nature methods*, vol. 10, no. 3, pp. 228–238, 2013.

[93] T. Willis and M. Dyer, "The role of immunoglobulin translocations in the pathogenesis of B-cell malignancies," *Blood*, vol. 96, no. 3, pp. 808–822, 2000.

[94] J. Vardiman, N. Harris, and R. Brunning, "The world health organization (WHO) classification of the myeloid neoplasms," *Blood*, vol. 100, no. 7, pp. 2292–2302, 2002.

[95] G. Henel and J. L. Schmitz, "Basic theory and clinical applications of flow cytometry," *Lab Med*, vol. 38, no. 7, pp. 428–436, 2007.

[96] E. Lugli, M. Roederer, and A. Cossarizza, "Data analysis in flow cytometry: The future just started," *Cytometry A*, vol. 77, no. 7, pp. 705–713, 2010.

[97] D. R. Parks, M. Roederer, and W. A. Moore, "A new "logicle" display method avoids deceptive effects of logarithmic scaling for low signals and compensated data," *Cytometry A*, vol. 69, no. 6, pp. 541–551, Jun. 2006.

[98] T. A. Knijnenburg, O. Roda, Y. Wan, G. P. Nolan, J. D. Aitchison, and I. Shmulevich, "A regression model approach to enable cell morphology correction in high-throughput flow cytometry," *Mol Syst Biol*, vol. 7, p. 531, 2011.

[99] T. W. Anderson, "On the distribution of the two-sample Cramér-von Mises criterion," *Ann Math Statist*, vol. 33, no. 3, pp. 1148–1159, 1962.

[100] F. J. Massey, "The Kolmogorov-Smirnov test for goodness of fit," *J Am Statist Assoc*, vol. 46, no. 253, pp. 68–78, 1951.

[101] S. Kullback and R. A. Leibler, "On information and sufficiency," *Ann Math Statist*, vol. 22, no. 1, pp. 79–86, 1951.

[102] M. Biehl, K. Bunte, and P. Schneider, "Analysis of flow cytometry data by matrix relevance learning vector quantization," *PLoS One*, vol. 8, no. 3, p. e59401, Mar. 2013.

[103] I. Vergara, T. Norambuena, E. Ferrada, A. Slater, and F. Melo, "StAR: a simple tool for the statistical comparison of ROC curves," *BMC Bioinformatics*, vol. 9, no. 1, p. 265, 2008.

[104] R. C. Holte, "Very simple classification rules perform well on most commonly used datasets," *Machine learning*, vol. 11, no. 1, pp. 63–90, 1993.

[105] A. Klami, P. Ramkumar, S. Virtanen, L. Parkkonen, R. Hari, and S. Kaski, "ICANN/PASCAL2 Challenge: MEG Mind-Reading — Overview and Results," 2011. [Online]. Available: http://www.cis.hut.fi/icann2011/meg/megicann_proceedings.pdf

[106] D. van De Ville and S.-W. Lee, "Brain decoding: Opportunities and challenges for pattern recognition," *Pattern Recognition, Special Issue on Brain Decoding*, vol. 45, no. 6, pp. 2033 – 2034, 2012.

[107] P. M. Rasmussen, L. K. Hansen, K. H. Madsen, N. W. Churchill, and S. C. Strother, "Model sparsity and brain pattern interpretation of classification models in neuroimaging," *Pattern Recognition*, vol. 45, no. 6, pp. 2085–2100, 2012.

[108] T. Naselaris, K. N. Kay, S. Nishimoto, and J. L. Gallant, "Encoding and decoding in fMRI," *NeuroImage*, vol. 56, no. 2, pp. 400–410, 2011.

[109] F. Pereira, T. Mitchell, and M. Botvinick, "Machine learning classifiers and fMRI: a tutorial overview," *NeuroImage*, vol. 45, no. Suppl 1, pp. S199 – S209, 2009.

[110] A. J. O'Toole, F. Jiang, H. Abdi, N. Pénard, J. P. Dunlop, and M. A. Parent, "Theoretical, statistical, and practical perspectives on pattern-based classification approaches to the analysis of functional neuroimaging data," *Journal of Cognitive Neuroscience*, vol. 19, no. 11, pp. 1735–1752, 2007.

[111] B. Blankertz, M. Tangermann, C. Vidaurre, S. Fazli, C. Sannelli, S. Haufe, C. Maeder, L. Ramsey, I. Sturm, G. Curio, and K.-R. Müller, "The Berlin Brain-Computer Interface: Non-medical uses of BCI technology," *Front Neurosci*, vol. 4, p. 198, 2010.

[112] B. Blankertz, K.-R. Müller, D. J. Krusienski, G. Schalk, J. R. Wolpaw, A. Schlögl, G. Pfurtscheller, J. del R Millán, M. Schröder, and N. Birbaumer, "The BCI competition III: Validating alternative approaches to actual BCI problems," *IEEE Trans Neural Syst Rehabil Eng*, vol. 14, no. 2, pp. 153–159, Jun. 2006.

[113] M. van Gerven, C. Hesse, O. Jensen, and T. Heskes, "Interpreting single trial data using groupwise regularisation," *Neuroimage*, vol. 46, pp. 665 – 676, 2009.

[114] R. Tomioka and K.-R. Müller, "A regularized discriminative framework for EEG analysis with application to brain-computer interface," *NeuroImage*, vol. 49, no. 1, pp. 415–432, 2010.

[115] M. Tangermann, K.-R. Müller, A. Aertsen, N. Birbaumer, C. Braun, C. Brunner, R. Leeb, C. Mehring, K. J. Miller, G. Mueller-Putz, G. Nolte, G. Pfurtscheller, H. Preissl, G. Schalk, A. Schlögl, C. Vidaurre, S. Waldert, and B. Blankertz, "Review of the BCI competition IV," *Frontiers in Neuroscience*, vol. 6, no. 00055, 2012.

[116] A. Zhdanov, T. Hendler, L. Ungerleider, and N. Intrator, "Inferring functional brain states using temporal evolution of regularized classifiers," *Comput Intell Neurosci*, p. 52609, 2007.

[117] A. M. Chan, E. Halgren, K. Marinkovic, and S. S. Cash, "Decoding word and category-specific spatiotemporal representations from MEG and EEG," *Neuroimage*, vol. 54, no. 4, pp. 3028–3039, Feb. 2011.

[118] J. W. Rieger, C. Reichert, K. R. Gegenfurtner, T. Noesselt, C. Braun, H.-J. Heinze, R. Kruse, and H. Hinrichs, "Predicting the recognition of natural scenes from single trial MEG recordings of brain activity," *Neuroimage*, vol. 42, no. 3, pp. 1056–1068, Sep. 2008.

[119] M. Besserve, K. Jerbi, F. Laurent, S. Baillet, J. Martinerie, and L. Garnero, "Classification methods for ongoing EEG and MEG signals," *Biol Res*, vol. 40, no. 4, pp. 415–437, 2007.

[120] L. Grosenick, S. Greer, and B. Knutson, "Interpretable classifiers for FMRI improve prediction of purchases," *IEEE Trans Neural Syst Rehabil Eng*, vol. 16, no. 6, pp. 539–548, Dec. 2008.

[121] M. K. Carroll, G. A. Cecchi, I. Rish, R. Garg, and A. R. Rao, "Prediction and interpretation of distributed neural activity with sparse models," *Neuroimage*, vol. 44, no. 1, pp. 112–122, Jan. 2009.

[122] S.-W. Lin, Z.-J. Lee, S.-C. Chen, and T.-Y. Tseng, "Parameter determination of support vector machine and feature selection using simulated annealing approach," *Appl. Soft Comput.*, vol. 8, pp. 1505–1512, 2008.

[123] R. Oostenveld, P. Fries, E. Maris, and J.-M. Schoffelen, "Fieldtrip: open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data," *Intell. Neuroscience*, vol. 2011, pp. 1:1–1:9, Jan. 2011.

Tampereen teknillinen yliopisto
PL 527
33101 Tampere

Tampere University of Technology
P.O.B. 527
FI-33101 Tampere, Finland

ISBN 978-952-15-3226-9
ISSN 1459-2045