

CZECH TECHNICAL UNIVERSITY

FACULTY OF ELECTRICAL ENGINEERING

---

On Parametric Model Creation with Neural  
Modeling Fields

---



MASTER THESIS

*Author:*  
Bc. Antonín ŠULC

*Supervisor:*  
Mgr. Michal VAVREČKA, Ph.D.

January 2014



České vysoké učení technické v Praze  
Fakulta elektrotechnická

Katedra kybernetiky

## ZADÁNÍ DIPLOMOVÉ PRÁCE

**Student:** Bc. Antonín Š u l c  
**Studijní program:** Otevřená informatika (magisterský)  
**Obor:** Umělá inteligence  
**Název tématu:** Tvorba parametrických modelů pomocí Neural Modeling Fields

### Pokyny pro vypracování:


Cílem práce je podrobně nastudovat rovnice Neural Modeling Fields popsané L. Perlovským a implementovat tyto rovnice pro vybrané jednorozměrné i mnoharozměrné distribuční funkce. Dále bude práce obsahovat popis a implementaci hierarchického modelu směsí s experty jako ukázkou aplikační domény principu odhadu maximální věrohodnosti. Výsledné modely budou podrobeny testům včetně porovnání s ostatními přístupy strojového učení.

### Seznam odborné literatury:


- [1] Perlovsky, L. I.; Kozma, R.; Eds.: Neurodynamics of Higher-Level Cognition and Consciousness. Springer-Verlag, Heidelberg, Germany, 2007.
- [2] Perlovsky, L. I.: Neural Networks and Intellect: using model-based concepts. Oxford University Press, New York, NY (3rd printing), 2001.

**Vedoucí diplomové práce:** Mgr. Michal Vavrečka, Ph.D.

**Platnost zadání:** do konce zimního semestru 2014/2015

  
doc. Dr. Ing. Jan Kybic  
vedoucí katedry



  
prof. Ing. Pavel Ripka, CSc.  
děkan

V Praze dne 14. 11. 2013



Czech Technical University in Prague  
Faculty of Electrical Engineering

Department of Cybernetics

## DIPLOMA THESIS ASSIGNMENT

**Student:** Bc. Antonín Šulc  
**Study programme:** Open Informatics  
**Specialisation:** Artificial Intelligence  
**Title of Diploma Thesis:** On Parametric Model Creation with Neural Modeling Fields

### Guidelines:

The goal of the thesis is to study Neural Modeling Fields equations described by L. Perlovsky and implement the equations for univariate and multivariate distribution functions. The work will contain description and implementation hierarchical mixtures of experts as example of an application domain of maximum likelihood estimation. Resulting models will be tested on real data including comparison with other machine learning techniques.


### Bibliography/Sources:

- [1] Perlovsky, L. I.; Kozma, R.; Eds.: Neurodynamics of Higher-Level Cognition and Consciousness. Springer-Verlag, Heidelberg, Germany, 2007.
- [2] Perlovsky, L. I.: Neural Networks and Intellect: using model-based concepts. Oxford University Press, New York, NY (3rd printing), 2001.

**Diploma Thesis Supervisor:** Mgr. Michal Vavrečka, Ph.D.

**Valid until:** the end of the winter semester of academic year 2014/2015



  
doc. Dr. Ing. Jan Kybic  
Head of Department

  
prof. Ing. Pavel Ripka, CSc.  
Dean

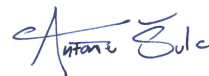
Prague, November 14, 2013

---

# Prohlášení autora práce

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 2. ledna 2014



.....  
Podpis autora práce

*“Představte si to ticho, kdyby lidé říkali jen to, co vědí.”*

Karel Čapek

## Abstrakt

Práce shrnuje současné poznatky z tvorby parametrických modelů pomocí přístupu založeném na Neural Modeling Fields (NMF). Součástí práce je podrobný popis NMF jako metody pro tvorbu parametrických modelů včetně vztahu k abstraktnímu pojetí NMF jako modelu, jehož formalismy umožňují algoritmus obecně vztáhnout k mysli.

Součástí práce je rovněž vlastní důkaz ekvivalence NMF s algoritmem známým jako Expectation Maximization (EM). Dokázaný vztah umožnil převzít snáze některé vztahy pro Gamma, Exponenciální, Normální, Lognormální, von Mises-Fisherovu, Wishartovu a Dirichletovu distribuční funkci, pro něž je algoritmus implementován. Uvedené funkce jsou podrobně analyzovány zejména ve vztahu k parametrickému modelování. Pro Exponenciální, vícerozměrné Lognormální a Dirichletovo rozdělení nebyla nalezena žádná relevantní informace o parametrických metodách ve směsích distribučních funkcí tudíž, práce obsahuje potřebné rovnice včetně odvození.

Experimentální část práce obsahuje vyhodnocení modelu učení s učitelem tzv. Hierarchical Mixture of Experts pro aproximační problémy jehož učení je založeno na principu maximalizace věrohodnosti. Dále pak experiment s NMF pro Normální rozdělení v hierarchii s Kohonenovou Samoorganizační mapou, jakožto klasifikátor vstupních obrazců. Vstupní data pro druhý problém jsou založena na tzv. Feature Integration Theory, na něž je MLANS adaptován. Poslední experiment byl realizován obrazový klasifikátor založený na matici vlastností klasifikovanou směsí Wishartova rozdělení.

## Abstrakt

The thesis summarizes state-of-the art of the parametric model creation with the Neural Modeling Fields (NMF) approach. The thesis contains a detailed description of the NMF as a method for parametric model creation including relation to NMF interpretation as model whose formalisms allows to use the NMF for a modeling mind processes.

The thesis contains proof of equivalence of the NMF and an algorithm known as Expectation-Maximization (EM). The proved equivalence allowed to take over more easily some relations for the Gamma, Exponential, Normal, Log-normal, von Mises-Fisher, Wishart and Dirichlet probability distribution functions for which the algorithm is implemented. Enumerated distributions are analysed in detail with in relation to parametric model creation. For Exponential, multivariate Log-normal and Dirichlet distributions no relevant resource about parametric model creation in a mixture of densities was found thus the thesis contains these equations including derivation of the equations.

Experimental part contains evaluation of the supervised parametric model based on Hierarchical Mixture of Experts for approximation problems whose learning is based on maximum likelihood principle. Further the thesis contains experiments of the NMF for Normal distribution in a hierarchy with Kohonen's Self Organizing Map, known as MLANS, as a classifier of input images where the data are based on Feature Integration Theory. The last experiment is region classifier of images based on feature matrices classified by a mixture of Wishart distribution.



## Acknowledgements

I would like to thank to MSc. Michal Vavrečka Ph.D. for the opportunity to be a part of his research group at Incognite, and for his continued encouragement, support and patience with my creativity explosions. I would like to thank to MSc. Karla Štěpánová, who advised me with the mathematical problems related to the thesis and provided detailed feedback.

I would like to thank my parents, grandparents and sister who let me do what they think I am good at and always supported me. I would like to thank my friends Pavel Hubáč and Babrobra Kafková for empathy and patience with my chaotic thought processes.

I would like to thank to the Faculty of Electrical Engineering, for providing me an environment that has enabled me to develop my skills in growing scientific area of cybernetics and artificial intelligence.

Special thanks belong to my former English teacher Ing. Dana Slámová CSc. who helped my with language corrections refinements of the text. Thanks also belongs to Professor Michal Pěchouček who gave me an opportunity to study abroad in Austria, where I found out what I really want to do in my carrier. Jana Zíchová as well as Leona Svobodová for support with administrative stuff and Jan Kubový for worth advices in my hard times.

# Contents

<b>Declaration of Authorship</b>	<b>iv</b>
<b>Abstract</b>	<b>vii</b>
<b>Acknowledgements</b>	<b>viii</b>
<b>List of Figures</b>	<b>xii</b>
<b>List of Tables</b>	<b>xiv</b>
<b>Abbreviations</b>	<b>xv</b>
<b>Symbols</b>	<b>xvi</b>
<b>1 Introduction to intelligent artificial systems</b>	<b>1</b>
1.1 Intelligence	1
1.1.1 Concept of intelligence	1
1.2 Artificial intelligence	2
1.2.1 Acting humanly	3
1.2.2 Thinking humanly	4
1.2.3 Think rationally	4
1.2.4 Acting rationally	5
1.3 Learning process	5
1.3.1 Learning as a philosophical problem	6
1.3.2 Learning with teacher	6
1.3.3 Learning without a teacher	7
1.3.3.1 Reinforcement learning	7
1.3.3.2 Unsupervised learning	7
1.3.4 Semi-supervised learning	7
1.4 History of the AI	8
1.4.1 Beginnings of logical reasoning	8
1.4.2 Connectionist era	9
1.4.3 History of density estimation	10
1.5 Adaptive systems	10
1.5.1 Rule-based systems	10
1.5.2 Non-parametric methods	10
1.5.2.1 Kernel method	11
1.5.2.2 Nearest neighbour methods	12
1.5.3 Parametric methods	12
1.5.3.1 Neural networks	13

<b>2</b>	<b>On Maximum likelihood estimation of some probability density functions</b>	<b>16</b>
2.1	Preliminaries	17
2.1.1	Likelihood and Log-Likelihood	17
2.1.2	Maximum likelihood estimation	17
2.1.3	Maximum likelihood estimation for a single function	18
2.1.4	Number of observations and accuracy of MLE	19
2.1.4.1	Analytic method	19
2.1.4.2	Numeric type method - Gradient ascent method	20
2.1.4.3	Numeric type method - Newton-Raphson method	21
2.1.4.4	Summary of MLE	22
2.2	Uni-variate distribution functions	23
2.2.1	Uni-variate Exponential distribution	23
2.2.1.1	Probability density function	23
2.2.1.2	$\mathcal{L}(\mathbf{x} \Theta)$ , $\ln \mathcal{L}(\mathbf{x} \Theta)$ and MLE	23
2.2.2	Uni-variate Gamma distribution	24
2.2.2.1	Probability density function	24
2.2.2.2	$\mathcal{L}(\Theta \mathbf{x})$ , $\ln \mathcal{L}(\Theta \mathbf{x})$ and MLE	24
2.3	Multivariate distributions	26
2.3.1	Von Mises-Fisher distribution	26
2.3.1.1	Probability density function	27
2.3.1.2	$\mathcal{L}(\Theta \mathbf{X})$ , $\ln \mathcal{L}(\Theta \mathbf{X})$ and MLE	27
2.3.2	Multivariate Normal distribution	28
2.3.2.1	Probability density function	28
2.3.2.2	$\mathcal{L}(\Theta \mathbf{X})$ , $\ln \mathcal{L}(\Theta \mathbf{X})$ and MLE	29
2.3.3	Multivariate Log-normal distribution	30
2.3.3.1	Probability density function	30
2.3.3.2	$\mathcal{L}(\Theta \mathbf{x})$ , $\ln \mathcal{L}(\Theta \mathbf{x})$ and MLE	30
2.3.4	Wishart distribution	31
2.3.4.1	Probability density functions	32
2.3.4.2	$\mathcal{L}(\Theta \mathbf{S})$ , $\ln \mathcal{L}(\Theta \mathbf{S})$ and MLE	33
2.3.5	Dirichlet distribution	33
2.3.5.1	Probability density function	33
2.3.5.2	$\mathcal{L}(\Theta \mathbf{X})$ , $\ln \mathcal{L}(\Theta \mathbf{X})$ and MLE	34
2.4	Exponential family distributions	35
<b>3</b>	<b>Neural Modeling Fields and Expectation maximization algorithm</b>	<b>37</b>
3.1	EM algorithm	38
3.1.1	EM algorithm as classifier	42
3.1.2	EM algorithm for various distribution functions	42
3.1.2.1	Univariate Exponential distribution	43
3.1.2.2	Univariate Gamma distribution	44
3.1.2.3	Multivariate Normal distribution	45
3.1.2.4	Multivariate Log-normal distribution	46
3.1.2.5	Multivariate von Mises Fisher distribution	46
3.1.2.6	Dirichlet distribution	47
3.1.2.7	Multivariate Wishart distribution	48
3.2	Neural Modeling Fields	48
3.2.1	Similarity measures	49
3.2.1.1	Aristotelian similarity	50
3.2.1.2	Fuzzy similarity	51
3.2.1.3	Adaptive Fuzzy similarity	51
3.2.2	Learning parametric models with NMF	51
3.2.3	NMF and EM algorithm equivalence	52

3.2.4	Maximum Likelihood Adaptive Neural System . . . . .	53
3.2.5	Perlovsky's theory of mind and NMF . . . . .	54
3.2.5.1	Understanding and meaning . . . . .	55
3.2.5.2	Imagination . . . . .	56
<b>4</b>	<b>Hierarchical mixture of experts</b>	<b>57</b>
4.1	Introduction . . . . .	57
4.1.1	Computation . . . . .	58
4.1.2	Interpretation of the architecture . . . . .	60
4.2	Learning the HME with EM algorithm . . . . .	61
4.3	Experiments . . . . .	64
4.3.1	Discussion . . . . .	67
<b>5</b>	<b>Feature Integration Theory - Experimental evaluation of MLANS</b>	<b>68</b>
5.1	Introduction . . . . .	68
5.1.1	Feature integration theory . . . . .	68
5.2	Description of experiment . . . . .	69
5.2.1	Scene features . . . . .	69
5.2.2	Clustering in the first layer . . . . .	71
5.2.2.1	Self Organising Maps . . . . .	71
5.3	Experiment . . . . .	72
5.3.1	First layer - training MLANS . . . . .	72
5.3.2	Second phase - Training the SOM . . . . .	74
5.3.2.1	SOM results . . . . .	74
5.3.3	SOM for feature clustering . . . . .	74
5.3.3.1	Comparison of SOM and MLANS . . . . .	77
<b>6</b>	<b>Region clustering of satellite images with Wishart distribution</b>	<b>81</b>
6.1	Introduction . . . . .	81
6.2	Image representation . . . . .	82
6.3	Classification of image regions . . . . .	84
6.4	Estimation of the parameters . . . . .	84
6.5	Experiments . . . . .	85
<b>7</b>	<b>Conclusion</b>	<b>89</b>
<b>A</b>	<b>Some special functions</b>	<b>91</b>
A.1	Approximation error of the Digamma function $\psi(x)$ . . . . .	91
A.2	Hessian matrix . . . . .	91
<b>B</b>	<b>Contents of attached CD</b>	<b>92</b>
B.1	Instructions to run the framework . . . . .	92
<b>Bibliography</b>		<b>93</b>

# List of Figures

1.1	Semi-supervised learning examples . . . . .	8
1.2	Classification of adaptive systems. . . . .	11
1.3	Kernel method illustration . . . . .	12
1.4	A feed-forward neural network trained by the back-propagation . . . . .	13
1.5	Illustration of one step of the SOM . . . . .	14
1.6	Two dimensional SOM example. . . . .	15
2.1	Example of log-likelihood function for mixture of densities . . . . .	18
2.2	MLE illustration . . . . .	19
2.3	Illustration of MLE estimation. . . . .	20
2.4	Newton-Raphson method illustration. . . . .	22
2.5	Values of $\text{Exp}(x \lambda)$ for various $\lambda$ parameter. . . . .	23
2.6	Values of $\text{Gam}(x k, \theta)$ for various parameters $k$ and $\theta$ . . . . .	24
2.7	MLE of Gamma PDF estimated by the NR method with 1000 observations $\mathbf{x}$ . . . . .	26
2.8	von Mises-Fisher distribution example . . . . .	27
2.9	Multivariate Normal PDF . . . . .	29
2.10	Log-normal and Normal distribution . . . . .	30
2.11	Probability density function of multivariate Log-normal distribution . . . . .	31
2.12	Dirichlet distribution function . . . . .	34
3.1	Bayesian Decision Boundary . . . . .	43
3.2	Example of the MLANS/EM algorithm density . . . . .	49
3.3	Example of the set partition . . . . .	50
3.4	Division of afferent and efferent signals. . . . .	55
4.1	Hierarchical Mixture of Experts architecture . . . . .	58
4.2	Divide and conquer principle in HME . . . . .	58
4.3	Experts group in sub-tree $i$ . . . . .	59
4.4	Separation with the soft-max function. . . . .	60
4.5	Benchmark functions for the HME . . . . .	66
4.6	The HME and the Neural Networks results comparison for a benchmark function. . . . .	66
4.7	The HME and the Neural Networks results comparison for a benchmark function. . . . .	66
5.1	Visualisation inputs for each possible concept of feature <i>Colour</i> . . . . .	70
5.2	Visualisation inputs for each possible concept of feature <i>Direction</i> . . . . .	70
5.3	Visualisation inputs for each possible concept of feature <i>Shape</i> . . . . .	70
5.4	Visualisation inputs for each possible concept of feature <i>Size</i> . . . . .	70
5.5	Visualisation inputs for each possible concept of feature <i>Texture</i> . . . . .	70
5.6	The structure of FIT system . . . . .	71
5.7	SOM, Component planes and the U-matrix . . . . .	72
5.8	MLANS mean for colour feature . . . . .	73
5.9	MLANS mean for direction feature . . . . .	73
5.10	MLANS mean for shape feature . . . . .	73

---

5.11	MLANS mean for size feature . . . . .	73
5.12	MLANS mean for texture feature . . . . .	73
5.13	Component planes for 10-by-10 SOM . . . . .	75
5.14	The U-matrix for 10-by-10 SOM . . . . .	75
5.15	Component planes for 50-by-50 SOM . . . . .	76
5.16	The U-matrix for 50-by-50 SOM . . . . .	76
5.17	Component planes for 90-by-45 SOM . . . . .	77
5.18	The U-matrix for 90-by-45 SOM . . . . .	77
5.19	The U-matrices of SOM for features which are performed with MLANS . . . . .	78
5.20	SOM and MLANS comparison of error . . . . .	78
5.21	The results for the colours and directions features of the smallest SOM . . . . .	79
5.22	The results for the sizes and shapes features of the smallest SOM . . . . .	80
5.23	The results for the textures feature of the smallest SOM . . . . .	80
6.1	Visualisation of particular attributes taken as a features. . . . .	83
6.2	Region clustering of Vilandi, Estonia . . . . .	86
6.3	Region clustering of Amsterdam, Netherlands . . . . .	87
6.4	Region clustering of Agricultural land of the Tadco company in Saudi Arabia . . . . .	88
A.1	Diagamma function approximation Eq. 3.21 error. . . . .	91

# List of Tables

1.1	Division of artificial intelligence into four categories. . . . .	3
3.1	Comparison between notion used in this thesis and Perlovsky's . . . . .	54
5.1	SOM and MLANS comparison of error . . . . .	79

# Abbreviations

<b>AI</b>	<b>Artificial Intelligence</b>
<b>RL</b>	<b>Reinforcement Learning</b>
<b>LT</b>	<b>Logic Theorist</b>
<b>AT</b>	<b>Advice Talker</b>
<b>GPU</b>	<b>General Problem Solver</b>
<b>NN</b>	<b>Neural Network</b>
<b>SOM</b>	<b>Self Organizing Map</b>
<b>BMU</b>	<b>Best Matching Unit</b>
<b>MLE</b>	<b>Maximum Likelihood Estimation</b>
<b>NR</b>	<b>Newton Raphson</b>
<b>FIT</b>	<b>Feature Integration Theory</b>
<b>MLANS</b>	<b>Maximum Likelihood Adaptive Neural System</b>
<b>EM</b>	<b>Expectation Maximisation</b>
<b>BDB</b>	<b>Bayesian Decision Boundary</b>
<b>SP</b>	<b>Set Partition</b>
<b>HME</b>	<b>Hierarchical Mixture of Experts</b>
<b>LSQ</b>	<b>Least Square Problem</b>
<b>WLSQ</b>	<b>Weighted Least Square Problem</b>
<b>R</b>	<b>Red</b>
<b>G</b>	<b>Green</b>
<b>B</b>	<b>Blue</b>
<b>w.r.t.</b>	<b>With RegardTo</b>
<b>i.i.d.</b>	<b>Identically Independently Distributed</b>



# Symbols

$x$	datum - observation scalar; $x \in \mathbb{R}$ .
$\mathbf{x}$	datum - observation vector with $D$ elements; $\mathbf{x} = [x_1, \dots, x_D]^T$ .
$ \mathbf{x} $	set cardinality, vector dimension
$\ \mathbf{x}\ _2$	Euclidean distance, $L_2$ metric
$n - by - m$	matrix, table, lattice with $n$ rows and $m$ columns
$\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$	data - set of $N$ observation vectors $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N = [\mathbf{x}_1 \dots \mathbf{x}_n]$
$d$	datum - desired value scalar for an observation $x$ or $\mathbf{x}$
$\mathbf{d}$	datum or data - $D$ dimensional desired vector or vector of desired values
$\mathbf{D}$	data - set of $N$ desired vectors $\mathbf{d}$ for observations $\mathbf{X}$
$a \equiv b$	$a$ and $b$ are equivalent concepts
$\mathbf{I}$	identity matrix
$p(x, y)$	joint probability of $x$ and $y$
$p(x y)$	conditional probability of $x$ given $y$
$p(\mathbf{x} \Theta)$	probability density function with parameters $\Theta$
$\mathbf{x} \sim p(\mathbf{x} \Theta)$	sample drawn from $p(\mathbf{x} \Theta)$
$f(\mathbf{x} \Theta)$	adaptive fuzzy similarity with parameters $\Theta$
$f(\mathbf{x} k)$	fixed fuzzy membership of a datum $\mathbf{x}$ to class $k$
$f_k(\mathbf{x} \Theta_k)$	$k$ -th probability density function value in mixture with parameters $\Theta_k$
$\Gamma(x)$	Gamma function $\Gamma(x) = (x-1)!$
$\psi(x), \psi^{(0)}(x)$	Digamma function $\psi(x) = \frac{\partial \ln \Gamma(x)}{\partial x}$
$\psi^{(1)}(x)$	Polygamma function $\psi^{(1)}(x) = \frac{\partial^2 \ln \Gamma(x)}{\partial x^2}$
$I_D(x)$	modified Bessel function of degree $D$
$\nabla_{\mathbf{y}} f(\mathbf{x})$	Gradient vector $\left[ \frac{\partial f(\mathbf{x})}{\partial x_1}(y_1), \dots, \frac{\partial f(\mathbf{x})}{\partial x_D}(y_D) \right]^T$ in $\mathbf{y}$
$\tilde{f}$	approximation of a function $f$
$\hat{\Theta}^{(i)}$	estimate of a set of parameters in the $i$ -th iteration of a learning algorithm
$\Theta$	a real set of parameters
$\Omega_{\Theta}$	a set of all admissible values of parameter/set of parameters $\Theta \in \Omega_{\Theta}$
$\text{diag}(\mathbf{X})$	diagonal elements of a matrix $\mathbf{X}$ in a column vector
$\text{diag}(\mathbf{x})$	square matrix whose diagonal elements are elements of the vector $\mathbf{x}$

---

$\text{tr}(\mathbf{X})$	trace of a square matrix $\mathbf{X}$ , sum of diagonal elements
$\det \mathbf{X}$	determinant of a square matrix $\mathbf{X}$
$\mathbf{S}$	scattering matrix of a set of observations with zero mean $\mathbf{S} = \sum_{n=1}^N \mathbf{x}_n \mathbf{x}_n^T$
$\mathbb{S}$	set of scattering matrices
$\mathbf{A} \cdot \mathbf{B}$	element wise multiplication of two matrices (Hadamard product)
$\mathcal{L}(\boldsymbol{\Theta} \mathbf{x})$	likelihood function of a univariate PDF $p(x \Theta)$ .
$\mathcal{L}(\boldsymbol{\Theta} \mathbf{X})$	likelihood function of a multivariate PDF $p(\mathbf{x} \Theta)$ .
$\ln \mathcal{L}(\boldsymbol{\Theta} \mathbf{x})$	log-likelihood function of a univariate PDF $p(x \Theta)$ .
$\ln \mathcal{L}(\boldsymbol{\Theta} \mathbf{X})$	log-likelihood function of a multivariate PDF $p(\mathbf{x} \Theta)$ .
$\text{Exp}(x \lambda)$	Exponential probability density function with parameter $\lambda$
$\text{Gam}(x k, \Theta)$	Gamma probability density function with parameters $k$ and $\Theta$
$\mathcal{N}(\mathbf{x} \mu, \Sigma)$	Normal probability density function with parameters $\mu$ and $\Sigma$
$\mathcal{LN}(\mathbf{x} \mu, \Sigma)$	Log-normal probability density function with parameters $\mu$ and $\Sigma$
$\mathcal{MF}(\mathbf{x} \mu, \Sigma)$	von Mises-Fisher probability density function with parameters $\mu$ and $\kappa$
$\mathcal{W}(\mathbf{S} N, \Sigma)$	Wishart probability density function with parameters $N$ and $\Sigma$
$\mathcal{D}(\mathbf{x} \mathbf{a})$	Dirichlet probability density function with vector of parameters $\mathbf{a}$

# Chapter 1

## Introduction to intelligent artificial systems

In this chapter, problems related to the concept of intelligence are discussed. Further the introductory part summarizes methods of model estimation applied in artificial intelligence.

### 1.1 Intelligence

For purposes of this thesis, proper intelligence and AI definition would be useful to determine capabilities of the artificial system build further.

#### 1.1.1 Concept of intelligence

Intelligence is one of the aspects of mind that forms character. Despite huge interest in modern artificial intelligence (AI), there is no generally accepted definition of intelligence. One problem arises because there is no technique to decide how to judge intelligence because the concept has no exact measurements. Human intelligence is naively referred as so called IQ. For example, one can contest IQ by the fact that there is nothing more than statistical correlation of an ability to solve problems that are focused on particular mind processes and IQ score, so the first emerging problem is exact specification of the way how to measure the intelligence. The other problem is that intelligence is used in diverse fields and so the basis of a definition uses concepts related to the particular field, while one can say that intelligence as an ability to solve some problems, the another can refer the intelligence as proper behaviour in hard situations.

Let us look at some definitions that try to determine intelligence. The first is related to AI [1]:

**Definition 1.1.** Intelligence measures an agent's ability to achieve goals in a wide range of environments.

The definition determines an intelligent agent as an entity that is focused on problem solving. If it is necessary to apply this definition in the psychological field it does not cover the area of self-criticism. Of course, mentioned the ability to achieve goals supposes that the agent already has knowledge of the problem, but how the agent would act in unknown environments? More generally is it necessary to require that an agent must be able to act in various (also unknown) environments?

On the other hand following definition [2] encapsulates that missed in Def. 1.1:

**Definition 1.2.** The ability to acquire and apply knowledge and skills .

The Def. 1.2 very aptly covers two crucial goals of intelligence, information acquisition (in terms of computer science - machine learning) and using acquired knowledge. The Def.1.2 covers what missed in Def. 1.1 but it says nothing about acting because the Def. 1.2 determines intelligence as an ability of interaction with the environment, but not intelligent behaviour. The definition bounds intelligence as a system without interaction with outside.

Let us investigate intelligence more from the human perspective. According to [3] the definition is formulated as follows:

**Definition 1.3.** The ability of an animal to form associative links between events or objects of which it has had no previous experience.

The Def. 1.3 formulates the same as the Def. 1.2 but in more biologically plausible way. The ability to form associative rules is the basis of artificial organisms. Despite problems related to the definition in the previous paragraph the Def. 1.3 approaches to a general connectionist principle of the intelligence perception that is applied widely in machine learning, e.g. in neural networks, which forms association rules based on available information by strengthening connections. The Def. 1.3 is used as the basis for further relation between system build in this thesis and concept of intelligence.

## 1.2 Artificial intelligence

As it is mentioned above, definition of intelligence is not that easy to determine just in one single definition. The intelligence is in the general domain of biological systems like humans or generally animals. A concept of intelligent agent is used instead of human or animal as an entity that embodies intelligence in AI. An intelligent agent can act *rationally* or has *human-like performance*. Rationality and human-like-performance can easily coincident. The human-like-performance is not necessary rational, because humans are not perfect <sup>1</sup> so the relation between rationality and human behaviour exist but it shall be distinguished.

The first pioneering work about how to judge intelligence as something artificial was proposed by Alan Turing with his *Turing Test*. Turing's test says that the computer is intelligent if human interrogator after posting some written questions cannot distinguish whether the responses come

<sup>1</sup>Not all humans are the best chess players [4].

from a computer or a human. The Turing's test gives rise to question whether the machines can behave intelligently or can have their minds. These two categories of simulated intelligence and machine mind is also referred as *weak* and *strong* AI. The weak AI supposes that the machine only behaves intelligently. The strong AI claims that Turing's test is not omnipotent judgement, it consider consciousness - being aware of its own states and actions. Strong AI does not simulate intelligence, it is intelligent itself. One thing that shall be taken into consideration is a limited amount of memory. For the both weak and strong AI it shall be considered that real machines are equipped with a finite number of memory cells, so it is possible to program  $2^k$  intelligent agents in  $k$  cells so the best agent can be constructed combinatorially, but the number of possible intelligent agents is not infinite.

Let us start with a definition of the AI taken from [3], which is defined as follows:

**Definition 1.4.** A field of computing concerned with the production of programs that perform tasks that requiring intelligence when they are done by people.

The Def. 1.4 does not say whether the AI is better or worse than an intelligent human, but it is still pretending intelligence thus it is weak AI. The human intelligence factor in AI can be perceived from different perspectives (see Tab.1.1); nevertheless, connection in between program, machine and intelligence is strong enough to consider the definition as consistent specification of requirements of the system that embodies AI. If the Def. 1.3 and Def. 1.4 are unified into one definition, the following defines what artificial intelligence is:

**Definition 1.5.** A field of computing concerned with the production of programs that perform tasks that requiring the ability to form associative links between events or objects of which it has had no previous experience.

Since this moment whenever AI is referred the basis relies on the Def.1.5.

Thinking Humanly	Thinking Rationally
Acting Humanly	Acting Rationally

TABLE 1.1: Division of artificial intelligence into four categories.

There are different views of how to analyse the AI. The AI is always related to human intelligence. Only difference is how the human intelligence and behaviour is reflected in AI. The [4] describes four quadrants (see. Tab. 1.1) of intelligence and rationality in relation to AI.

The four concepts in Tab. 1.1 are discussed in following sections in a more detail.

### 1.2.1 Acting humanly

The system that acts humanly is a system that imitate human acting. This approach is closest to robotics which tries to construct machines that can do what human can [4].

The art of creating machines that perform functions that require intelligence when performed by people [4].

The goal of a system is not problem solving but human acting (e.g. rescue robots, search agents and so on). The above mentioned Turing's test is just strongly focused on human acting. A machine is considered as intelligent if a human judge cannot decide whether written answers to asked questions were answered by a computer or a human.

### 1.2.2 Thinking humanly

Thinking humanly is focused on intelligent behaviour rather than acting.

The exciting new effort to make computers think ... machines with minds, in the full and literal sense [4].

If one wants to say that a computer works in the same way as a human thinks we must judge how humans really think. In [4] there are three perspectives of how to investigate systems that think humanly, namely:

1. through introspection - trying to understand ourselves.
2. through psychological experiments - mostly experimental, cognitive.
3. through brain imaging - observing how the brain works.

All above mentioned perspectives are not merely related to machines but also to cognitive sciences. While the systems that act humanly are more focused on the technical issues of the system; the systems that think humanly are more focused on how system performs actions internally.

### 1.2.3 Think rationally

The study of mental faculties through the use of computational models [5].

Antic philosopher Aristotle was the first who tried to codify rational (right) thinking [4] with theory of syllogisms. The aim was to investigate whether humans can make a conclusion based on a set of claims (premises) so that a new claim is correct to its premises. The following example shows that based on two premises: *All students that study hard will graduate* and *X study hard* can be concluded *X will graduate*:

$$\frac{\begin{array}{l} \text{All students that study hard will graduate} \\ \text{X study hard} \end{array}}{\text{X will graduate}} \quad (1.1)$$

This logical reasoning has become the basis for first-order logic, which is the basis of rationality in AI. Systems cannot diverse from the goals defined in a set of logical statements. In 19th century, formal apparatus has been developed to transform logical statement into so called predicate logic and then Eq. 1.1 can be formulated in more convenient form for computers as follows:

$$\frac{\forall x : \text{StudyHard}(x) \implies \text{Graduate}(x)}{\text{StudyHard}(X)} \quad (1.2)$$


---


$$\text{Graduate}(X)$$

Since the Aristotle theory of syllogisms much time has passed, but, nowadays, for AI the theory has brought one of the most important concepts where systems/agents think rationally using their knowledge (set of logical statements), regardless what a human would do. Bad news is that reasoning shown above require many cases exponential amount of time<sup>2</sup>.

### 1.2.4 Acting rationally

*Computational Intelligence is the study of the design of intelligent agents [4].*

Agents are very similar to objects in terms of object oriented programming, but the major difference is that the agent has its own behaviour. Often rationality is related to a performance measure (e.g. number of wins in a chess game, average price of processed tickets, etc.). Rational acting is acting where agent's behaviour is the good with respect to some performance measures.

As it is mentioned earlier, not all humans are rational. Great example is Allais paradox. People are asked to make a decision in a lottery with probability and choose between 1 and 2, and 3 and 4:

1. 80% chance to win \$4000,
2. 100% chance to win \$ 3000,
3. 20% chance to win \$ 4000,
4. 25% chance to win \$ 3000,

Most people prefer 2 over 1 and 3 over 4. If the prize is weighted by the probability of win the outcome has different preference ordering, 1 is preferred to 2. Why? Most people are risk aware. The risk awareness has its basis in certainty effect. People are strongly attached to certain gains but then do not act rationally. Agent that acts rationally is the agent that optimize certain utility regardless what human would do, moreover, the rational agent shall act better than human. In case of the Allais paradox, the agent shall always pick the best option regardless the human risk-awareness<sup>3</sup>.

## 1.3 Learning process

Learning is referred to as a process where new knowledge or skill is acquired based on given data. According to data provided there are two types of learning. First type supposes the existence

---

<sup>2</sup>Example of tractable logical statements for the machine is a knowledge base based on Horn's clauses which are solvable in polynomial time.

<sup>3</sup>As an agent that acts rationally is often referred the booking agent, that tries to find the best tickets.

of annotations/labelling of input signals while the second uses only inputs (eventually with a feedback as reinforcement learning). The learning process adapts some parameters  $\hat{\Theta}$  with regard to (w.r.t.) the data.

### 1.3.1 Learning as a philosophical problem

The idea of investigating learning and intelligence problem dates back to Antics. First attempts to formulate learning were made by Plato's theory of *Ideas*. Plato claimed that all that a human can know is in a world of Ideas, and human intelligence is formed exclusively by the connection with the world of Ideas. Initially, human has no associations with the world of Ideas. The learning is based on recalling all that is given *a priori*. In computer science, the notion of the world of the Ideas resembles abstract machine called *oracle*, which is a machine that always knows the correct answer for any question. The Plato's principle a-priority in practice means that any real object (as a concept) has relation with the world of Ideas that are shared by all humans, and the world of Ideas is static.

The Plato's world of Ideas lacks the learning process that is a natural part of the human intelligence (the learning is an integral part of the Def. 1.3). Aristotle, a Plato's pupil [6] criticised Plato's Ideas just because of absence of learning. The Aristotle claimed that intelligence does not come from the world of Ideas but rather evolves dynamically without any prior knowledge. In comparison to Plato's theory the Aristotle's supposed that human is born with no prior knowledge<sup>4</sup> and intelligence that evolve gradually as *educative* process<sup>5</sup>.

### 1.3.2 Learning with teacher

Learning with teacher, also referred to *supervised learning* type of learning where desired outputs for input data are given. A teacher has got knowledge about the problem domain, and data are associations of  $\mathbf{x}$  - output  $\mathbf{d}$  pairs, where  $\mathbf{d}$  are the desired outputs of the system annotated by the teacher. A learning algorithm adapts system's parameters  $\hat{\Theta}$  to minimize the difference between system's output for a particular datum  $\mathbf{x}$  and desired outputs  $\mathbf{d}$ . It is not always necessary to find exact mapping from input space  $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$  to (desired) outputs space  $\mathbf{D} = \{\mathbf{d}_n\}_{n=1}^N$  because, for example, only subset of the data is provided and some generalizations are needed. There are two major methods how to present the training data. The first method gives all inputs  $\mathbf{X}$  to the system, then new parameters are calculated for each datum and parameters are adapted. This learning method is often referred to as *offline* or *batch* learning because the training is performed in batches. The second method is called *online learning*, where training data are presented one-by-one, and for each input-output training datum the parameters  $\hat{\Theta}$  are adapted.

A practical example is raw learning of a foreign language vocabulary which can be performed by simple making bi-directional associations between foreign words and native language from dictionary. The learning method is online because each word association is made separately.

<sup>4</sup>Individual initially has no knowledge, which is called *Tabula rasa*

<sup>5</sup>Concept of *educative* process is used instead of *adaptivity* because it better corresponds to the learning process. The adaptivity is an ability to adapt to environment needs, whereas *educative* is wilful process of learning [6].



### 1.3.3 Learning without a teacher

Learning without teacher is a process where no annotations are given. Further learning without teacher can be divided into two categories: reinforcement learning that interacts systematically with an environment to improve the system's performance and unsupervised learning where no expected outputs are given and the system tries to find the underlying structure on input data.

#### 1.3.3.1 Reinforcement learning

The supervised learning supposes the existence a teacher who is able to respond to a set of training stimuli. The systems restricted to learning under these conditions are not adequate when it is costly, or event impossible, to obtain the required desired outputs. The reinforcement learning allows systems to learn from experiments instead of exclusively from teachers [7].

In reinforcement learning (RL), the learning of an input  $\mathbf{X}$  samples is performed through interaction with the environment in order to maximize utility [8]. There are no desired outputs. The RL is in class of learning methods based on *trial-and error* where the system repeats an experiment in given order until some criterion is met.

#### 1.3.3.2 Unsupervised learning

Unsupervised learning is a process where no information about outputs is given. The only information that is known is input samples  $\mathbf{X}$ . The most common task of unsupervised learning is clustering : detecting subsets that have some shared property/properties.

### 1.3.4 Semi-supervised learning

Semi-supervised learning is a combination of supervised and unsupervised learning. The semi-supervised learning is often referred when only a subset of the data is annotated or when some additional information is given.

The example of partially annotated data in Fig. 1.1 (left) where the task is to separate points into two classes. The labelled part of the data is used to find borderline between red and blue classes and the other unlabelled data are classified by the found borderline. The data with no extra information is not useless in the learning process. They can be, for example, used for finding a prior distribution of all data (regardless they are annotated or not annotated) in input space. Another case of semi-supervised learning is when the observations contain some uncertainty or noise. The example of presence of noisy data for regression problem can be seen in Fig. 1.1 (right), where two outliers are deviated from linear relation. If the regression problem was solved as an ordinary least square linear regression, then outlier would negatively affect results, and attract the solution to the wrong line. On the other hand, robust regression ignores the outliers and estimates regression as it was desired. In general, the semi-supervised learning means works partially annotated or with uncertain data.

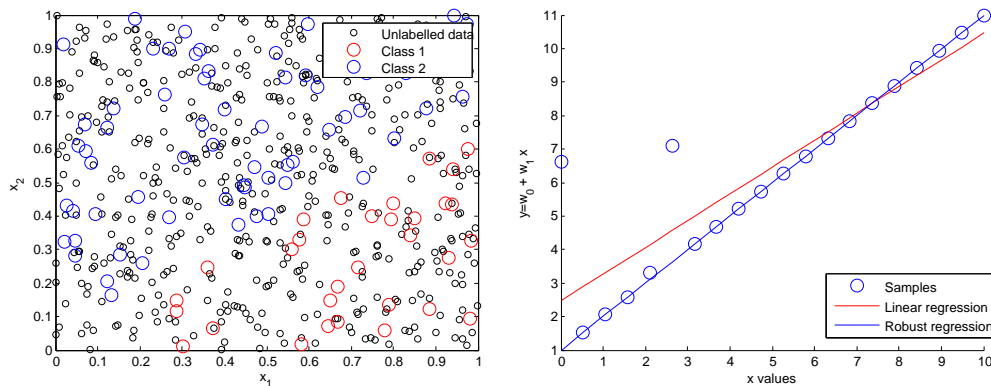


FIGURE 1.1: Semi-supervised learning. Left: Classification of partially annotated data. Part of the data is labelled (red and blue circles), and part is unlabelled. Right: Regression where some undesired disturbances are present. The red line is the result of linear regression where parameters are adapted with disturbance while robust regression ignored the outliers.

## 1.4 History of the AI

It is hard to decide when AI came into awareness as a branch of computer science, due to the fact there is no commonly accepted definition thus it is very hard to determine where is the boundary between the theoretical model on the paper and the system that embodies an AI as it is perceived by the definition Def. 1.5. There are some theories that existed only on paper before they were simulated on the first computer. A big question also is when the first computer was constructed and what is and what is not a computer. If the computer is considered to be something that can perform elementary arithmetic operations, then the first computer is Pascal's mechanic calculator constructed in the 17th century. A semiconductor can be identified as a real milestone of computing since this is the material which is used for microprocessor construction. It shall be emphasized that artificial neural networks were modelled with analogue circuits even before.

As the first attempts of the AI is referred a first artificial neuron designed by McCulloch and Pitts in 1943. Inputs of the neuron are weighted by synapses and output is binary - *on* or *off*. The theory of artificial neuron also supposed construction of neural network which can perform simulation of arbitrary logical function. Afterwards, Donald Hebb came with *Hebbian learning rule* performed on the artificial neurons. The Hebbian learning rule is a basic principle which strengthen weights of simultaneously active neurons. In 1950 Marvin Minsky and Dean Edmonds came with the first artificial neural network which is called *SNARC*. The SNARC used 3000 vacuum tubes and surplus automatic pilot mechanism to simulate a network of 40 neurons [4].

### 1.4.1 Beginnings of logical reasoning

Two researchers from Carnegie Mellon University Allen Newel and Herbert Simon showed reasoning program called *Logic Theorist (LT)* on a workshop in 1956 [4]. Simon claimed the following about the LT:

We invented a computer program capable of thinking non-numerically, and thereby solved the venerable mind-body problem, explaining how system composed of a matter can have the properties of mind. [4]

The program was able to prove most of the theorems from Russel's and Whitehead's book *Principia Mathematica*. The statement, in which the LT solved a the mind-body problem classifies the LT as program that embodies strong AI [4].

In 1958, McCarthy introduced a hypothetical program called *Advice Taker* (AT) that used knowledge same as the LT, but AT supposed existence of general knowledge which contains all possible rules of the World. Later McCarthy formed a new team at Stanford, afterwards, one member of the team discovered resolution method for the first order logic [4].

Later on, in 1959, Simon and Shaw came with a *General Problem Solver* (GPU). The idea of the GPU was to approach human way of problem solving by its division into subtasks. Rather than to solve the problem in general, the GPU tries to approach the way how a human being solves it. In the same year, Herbert Gelernter constructed a *Geometry Problem Solver* which was able to solve problems that some students had found out as tricky [4].

### 1.4.2 Connectionist era

The logic based systems had shown that their power decreases rapidly with a task complexity. The problem of reasoning with representation where all information is defined in some rules suffers from higher computational requirements. Generally, the logical reasoning does not take into consideration the optimization of the input problems; they are supposed to be theoretically solved in a finite time; thus new approaches were necessary to shift AI forward to be a regular scientific discipline. The ideas of McCulloch and Pitts in 1960 gave a rise to adaptive linear neural networks, also ADALINEs and perceptrons.

At the given time, all neural networks were one layered because simply no learning algorithm had existed for the multi-layered neural networks. This bound was the real limit of capabilities of the neural networks. Invention of back-propagation in the middle of 1980 for multi-layered feed-forward networks changed completely pessimistic perception of the future of connectionist principle and the entire AI. The invention of back-propagation opened many problems that seemed intractable for AI. The real break-even point was Kolomogorov's theorem that has claimed that any continuous function of  $n$  variables can be represented by a finite network of functions of a single argument, where addition is used as the only function of several arguments [9].

Some euphoria from the times when back-propagation was invented still alive these days, mainly because the illusion of the increasing number of neurons can simulate arbitrary mapping.

### 1.4.3 History of density estimation

The historical background of density estimation problems is a bit different branch of AI. It evolved in statistics rather than in AI because the density estimation was initially purely statistical discipline which later became a part of AI, as a machine learning branch.

The earliest reference to literature on the algorithm that reminds EM algorithm dates back to 1886 where the Newcomb considered an estimation of parameters of a mixture of two univariate normal distributions. Later in 1926 McKendrick gave a medical application of the problem in the spirit of the EM algorithm. In 1956, iterative method for estimating missing values which turned out to be an EM algorithm by Healey and Westmacott in 1956 was proposed. In 1958, Harley gave a treatment of the general case of count data and the first idea about formalization of an EM type algorithm as it is known nowadays, was published. Later on Buck in 1960 published a paper which considered estimation of a multivariate Normal distribution with a mean vector and a covariance matrix with the idea that only part of data is observed. The Buck's method is also known as semi-supervised learning. It uses given observations to regress missing values. Interesting is that Buck's method gives MLE under certain conditions, so the solution is near to EM. In 1970 Blight in 1970 tried to solve the problem of finding MLEs for exponential family distributions and his solution had turned out to be EM algorithm. Blight also proved some convergence results. Baum's series of papers in 1967 and 1970 can be perceived as a real beginning of the EM algorithm. They applied EM algorithm in Markov Model. Later on in 1972 Orchard and Woodbury introduced the principle of missing information. Orchard's contribution includes incomplete and complete likelihood function that are crucial for the algorithm. Finally Dempster, Laird and Rubin [10] published the algorithm that is called EM algorithm.

## 1.5 Adaptive systems

The classification of adaptive systems as a branch of the machine learning is shown in Fig. 1.2.

### 1.5.1 Rule-based systems

Rule based systems are systems that work with **if-then** statements. The knowledge base is set of all rules which the system consists of. The procedure of obtaining an answer from rule based system is based on asking whether a query fulfil one of the if-then rules. If there is any **then** conclusion, then it is taken as the answer of the system.

Rule-based systems are, for example, used for medical diagnostics where **if** statement are some set of known disease symptoms and **then** statements are the treatments.

### 1.5.2 Non-parametric methods

Non-parametric methods (NPM) play important role in the machine learning. The NPM have no functional form but allow the form of the density to be determined entirely by the data. The

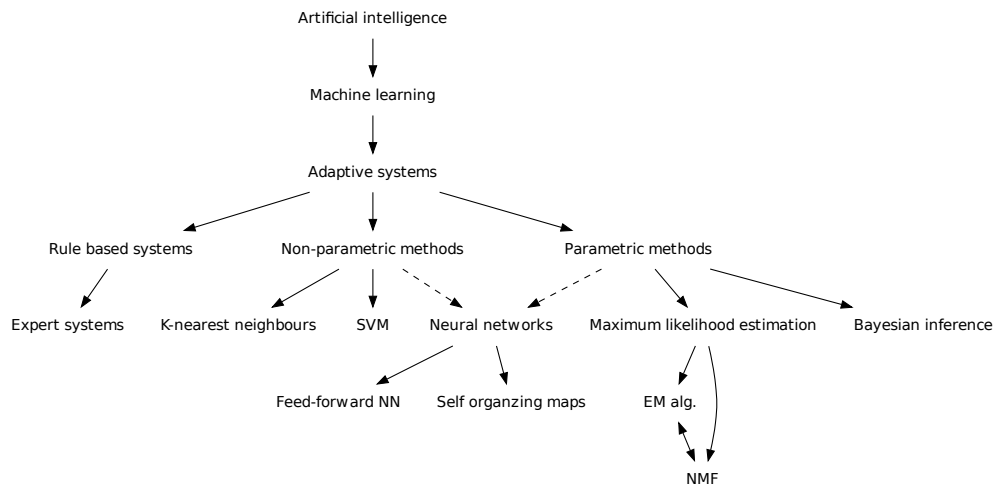


FIGURE 1.2: Classification of adaptive systems.

NPM can be used, for example, to assess the multi-modality, skewness, or any other structure in the distribution of the data. It can also be used for the classification and discriminant analysis. The NPM are alternatives to the parametric approaches, in which one specifies a model up to a few parameters and then estimates the parameters via e.g. the maximum likelihood principle. Currently, the most popular NPM methods for density estimation are Kernel methods and Nearest neighbours methods [11].

### 1.5.2.1 Kernel method

The goal of Kernel method is to approximate an unknown probability density function  $p(\mathbf{x}|\Theta)$  of a random variable  $\mathbf{x}$ . Assume we have  $N$  observations  $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$  drawn from the  $p(\mathbf{x}|\Theta)$ . The estimate of density based the Kernel method  $f(\mathbf{x}_i|h)$  at a point  $\mathbf{x}$  is defined as

$$f(\mathbf{x}_i|h) = \frac{1}{Nh} \sum_{n=1}^N K\left(\frac{\mathbf{x}_i - \mathbf{x}_n}{h}\right) \quad \mathbf{x}_i \notin \mathbf{X} \quad (1.3)$$

The  $K(\mathbf{x})$  is symmetric probability density function<sup>6</sup> which is commonly Gaussian-like distributions. The parameter  $h$  is called bandwidth (smoothing) and determines how wide is a kernel window. The task of the Kernel method is to choose an appropriate kernel function  $K(\mathbf{x})$  and bandwidth parameter  $h$  for a given set of observations  $\mathbf{X}$ . There is a trade-off between bias and variance. If the  $h$  is too large then variance of the model is small (see right bottom in Fig. 1.3), conversely if  $h$  is too small then variance is large (see left top in Fig. 1.3).

<sup>6</sup>Symmetry means  $K(\mathbf{x}) = K(-\mathbf{x})$  and distribution function property always gives 1 for the sum over all possible values of random variable.

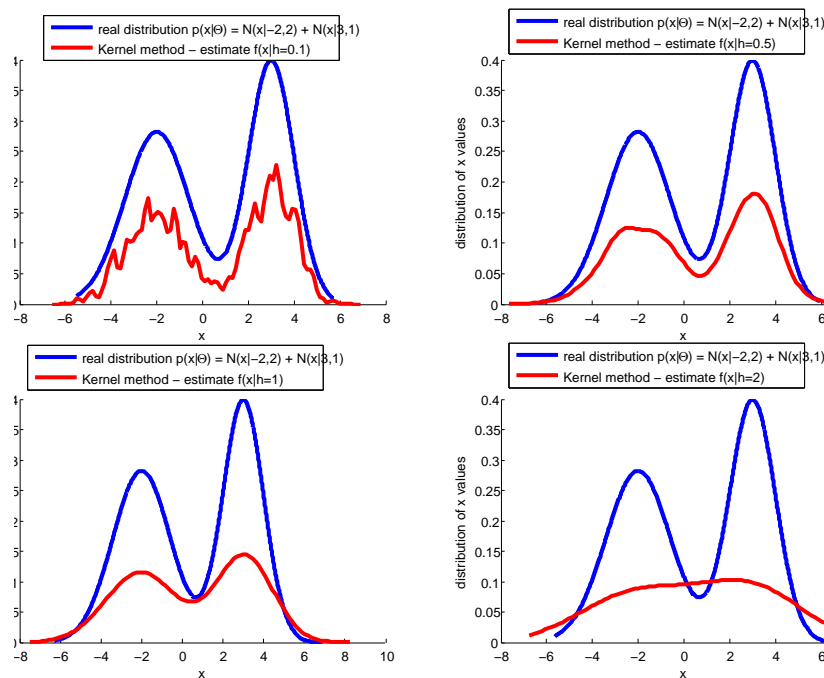


FIGURE 1.3: Example of Kernel method for density estimation with varying bandwidth and Gaussian kernel where the data are drawn from two univariate Gaussians.

### 1.5.2.2 Nearest neighbour methods

One of the potential problems of Kernel method for density estimation arises from fixed bandwidth parameter  $h$  for all data points  $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$ . If bandwidth is too large, then some regions are over-smoothed but reducing  $h$ , may cause noisiness on dense regions. The Nearest neighbour methods let bandwidth  $h$  to vary. Since the  $h$  may change, the neighbourhood with volume  $V$  of a value  $\mathbf{x}$ , where the density is estimated, absorbs more and more observations until certain  $k$  observations lie in the neighbourhood that grows as  $h$  is getting larger. Intuitively, the above mentioned facts can be formalized into following ratio neighbourhood [12]:

$$\hat{p}(\mathbf{x}) = \frac{k}{NV(h)} = \frac{\text{\#points in neighbourhood}}{\text{\# all points} * \text{increasing volume of neighbourhood depending on } h} \quad (1.4)$$

The Nearest neighbourhood has one parameter, which critically affects performance - number of  $k$  neighbours. If the  $k$  is too large, model variance is small and conversely. If  $k$  is too small then variance is very high and thus small number of other points regions, where the density of observations is larger the Nearest neighbour method exhibits very peaky estimations [12]

### 1.5.3 Parametric methods

Parametric methods (PM) are those which have a specific functional form for the estimated model. The functional form is defined by a set of parameters  $\Theta$  which are optimized by adapting

their values to fit the parametric model to the observations. The PM the one of the most straightforward approaches in adaptive systems. The most frequently used parametric model for the density estimation is the (multivariate) normal distribution which has convenient analytical and statistical properties. Two major techniques for PM are: *maximum likelihood* and *Bayesian inference*. In the first technique the parameters of functional form that describes an estimate are adjusted to find the most likely values of parameters (estimate)  $\hat{\Theta}$ . On the other hand, Bayesian inference does not use maximization principle for the likelihood function but computes expected values based on observations [12]. As parametric methods are also referred neural networks and mixture of densities. In some literature [12, 13] they are referred as semi-parametric because the parameters also specifies functional form, but the functional form is closed and then parameters of both can be estimated through PM [12], thus both methods are considered as PM here.

### 1.5.3.1 Neural networks

The term neural network (NN) has evolved to encompass a large class of models and learning methods. The NN is an attempt at modeling the information processing capabilities of nervous systems [9]. The NN can perform a wide range of learning tasks for supervised, unsupervised and semi-supervised learning. The NN is a graph that consists of elementary units called neurons. Each neuron performs defined computation where each unit has adaptive parameters called *weights*. The principle of the NN is based on distribution of knowledge among neurons to represent desired problem by the weights and interconnections between the neurons.

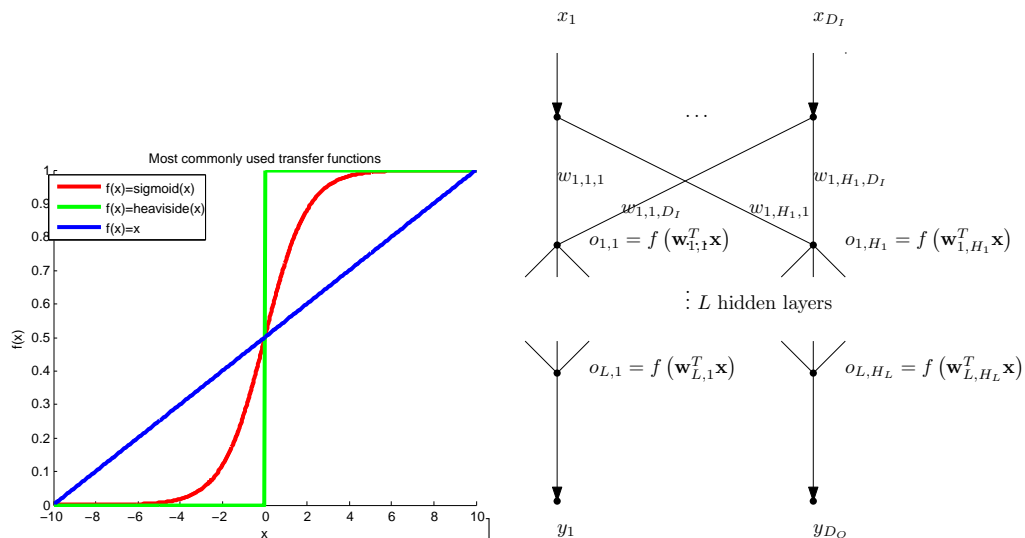


FIGURE 1.4: Left: The most commonly used transfer functions  $f(x)$ . Right: General structure of a feed-forward neural network trained by back-propagation. The first layer presents input vectors  $\mathbf{x}$  to hidden  $H_1$  units in the first layer. After all  $L$  layers processes outputs from previous layers, the outputs from the last hidden layer are presented to the outputs. The network parameters are determined by  $D_I H_1 + \sum_{l=2}^L H_l H_{l-1}$  weights where  $H_i$  is the number of units in  $i$ -th hidden layer.

As a supervised realization of the NN is usually referred feed-forward network whose architecture is defined in directed graph without cycles where the direction is from inputs to outputs (see

Fig. 1.4, left). The learning algorithm for multi-layered NNs is usually back-propagation algorithm. The back-propagation algorithm is a general weight adaptation rule based on gradient descent of the error function. Major disadvantage of the feed-forward approach is the non-transparent interpretation of the network structure (architecture, weights) for more complex architectures.

As an unsupervised realization of the NN is usually referred Kohonen's Self Organising Map (SOM). It provides a mapping from a high-dimensional input space to a lower-dimensional, often two-dimensional, output space. In the process of this mapping input patterns, that are located close to each other in the input space, will also be located closely in the output space, while dissimilar patterns will be mapped on opposite regions of the trained SOM [14].

The SOM provides a sort of clustering of the data. Basically, the SOM is a low-dimensional lattice, consisting of  $m$  neurons or units. The map lattice can have different topologies, in this thesis only rectangular lattices are used. For each neuron in the output space, a weight vector  $\mathbf{w}_i$  of the dimensionality of the input space is linked to a position on the two-dimensional map lattice. In the training phase, the best matching unit (neuron, weight vector) is identified for all input vectors by using a distance function, which is most usually Euclidean distance, After that, the best matching unit is identified, its weight vector. The weight vectors of neighbouring units are shifted towards the input vector as it is illustrated in Fig. 1.5. Example of the entire procedure of the SOM training on two dimensional data with two normally distributed clusters is shown in Fig. 1.6.

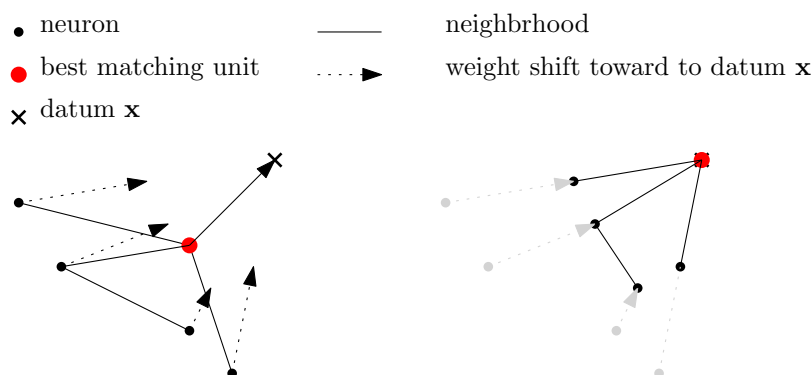


FIGURE 1.5: Weight adaptation of the SOM for the best matching unit (BMU) and its neighbours. After the BMU is found, the weight is shifted toward to the datum  $\mathbf{x}$  together with its neighbouring neurons where the the neighbours are shifted less w.r.t. as the topological distance from the BMU grows.



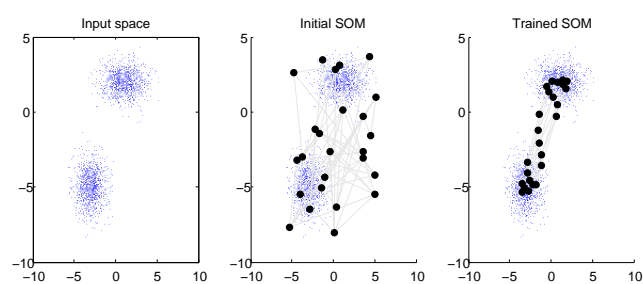


FIGURE 1.6: Two dimensional SOM example. *Left*: Input samples. *Middle*: Initial state of the SOM. *Right*: Trained SOM.

## Chapter 2

# On Maximum likelihood estimation of some probability density functions

The distribution functions are chosen to cover diverse types of data. As the first, the most commonly used Normal distribution is shown, which is naturally universal choice in the most cases. Further, there are skewed distributions Gamma and Log-normal that are useful in cases where some asymmetries, unlike the Normal distribution are needed. Another class are distributions which require special input format, namely von Mises-Fisher and Dirichlet distributions. The Von Mises-Fisher is distribution which requires unit norm can be useful in situations where only direction is relevant or input data lies on a unit sphere. The Dirichlet distribution requires unit sum which means the data lies in  $D - 1$  simplex. For example, the Dirichlet distribution is applied for probabilistic topics model creation [15, 16]. The Wishart distribution can be applied for some matrix estimation, which is shown, for example, in Chapter 6.

For each distribution function likelihood and log-likelihood functions are derived. The likelihood and log-likelihood functions are used for comparison between models as a measure of goodness between two models, and the concept plays an important role in further chapters to define some special functions based on these functions.

The likelihood and log-likelihood functions are also useful in finding maximum likelihood estimate based on observation to guess parameters of an original distribution function. In Chapter 3 it will be shown more complicated situation with a mixture of the distribution functions where the maximum likelihood estimation results are used. The results are necessary basis for the problem of the estimation of the original parameters with a mixture of densities.

The distribution functions are divided into two categories. The first is univariate where the modeled random variable is one-dimensional ( $D = 1$ ) and multivariate where the random variable can have more than one dimension ( $D \geq 1$ ).

## 2.1 Preliminaries

### 2.1.1 Likelihood and Log-Likelihood

Likelihood is a function of parameters  $\Theta$  for a given probability density function  $p(\mathbf{x}|\Theta)$  (PDF) and finite set of  $N$  observations  $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$  (each observation  $\mathbf{x}_n$  is supposed to be  $D$  dimensional). The likelihood function gives a measure for comparison how likely a set of observations  $\mathbf{X}$  is drawn with a set of parameters  $\Theta$ .

**Definition 2.1.** Given set of  $N$  observations  $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$  drawn from a known PDF  $x \sim p(\mathbf{x}|\Theta)$  then the likelihood function is calculated as follows:

$$\mathcal{L}(\Theta|\mathbf{X}) = \prod_{n=1}^N p(\mathbf{x}_n|\Theta) \quad (2.1)$$

For some applications it is more convenient to work with the log-likelihood denoted as  $\ln \mathcal{L}(\Theta|\mathbf{X})$  which is logarithm of likelihood function defined in the Eq. 2.1. The logarithm is a monotonically increasing function which ensures that when one parameter setting has greater likelihood than the other, the parameter has greater log-likelihood too<sup>1</sup>. The logarithm has useful property which changes the product over all observations in Eq. 2.1 to a sum of logarithms over of the PDFs which is easier to handle:

$$\ln \mathcal{L}(\Theta|\mathbf{X}) = \ln \prod_{n=1}^N p(\mathbf{x}_n|\Theta) = \sum_{n=1}^N \ln p(\mathbf{x}_n|\Theta) \quad (2.2)$$

Both likelihood and log-likelihood are equivalent for comparison, thus it is not necessary to strictly distinguish in between log-likelihood and likelihood functions.

### 2.1.2 Maximum likelihood estimation

Maximum likelihood estimation (MLE) is a technique used to estimate parameters of a given PDF based on a likelihood of a set of observations drawn from a PDF. Formally MLE finds such parameters that reach the maximum for a given set of observations of the likelihood function. The Def. 2.2 states MLE in more rigorous way.

**Definition 2.2.** The MLE maximizes likelihood function for a given set of  $N$  observations  $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$  by finding parameters  $\hat{\Theta}^{MLE}$  such that:

$$\hat{\Theta}^{MLE} = \arg \max_{\hat{\Theta} \in \Omega_{\Theta}} \mathcal{L}(\hat{\Theta}|\mathbf{X}) = \arg \max_{\hat{\Theta} \in \Omega_{\Theta}} \ln \mathcal{L}(\hat{\Theta}|\mathbf{X}) \quad (2.3)$$

<sup>1</sup>Mathematically speaking following relation holds :  $\mathcal{L}(\Theta_1|\mathbf{X}) \geq \mathcal{L}(\Theta_2|\mathbf{X}) \Leftrightarrow \ln \mathcal{L}(\Theta_1|\mathbf{X}) \geq \ln \mathcal{L}(\Theta_2|\mathbf{X})$ .

The Def. 2.2 does not specify how the function  $p(\mathbf{x}|\Theta)$  inside the likelihood or log-likelihood function is formulated thus the function  $f(\mathbf{x}|\Theta)$  can be formulated as a weighted sum of  $K$  PDFs  $f_k(\mathbf{x}|\Theta_k)$  weighted by coefficients  $\pi_k$  which is called *mixture of densities*:

$$p(\mathbf{x}_n|\Theta) = \sum_{k=1}^K \pi_k f_k(\mathbf{x}_n|\Theta_k) \quad (2.4)$$

Where the  $\Theta$  is a set of parameters  $\Theta = \{\pi_1, \dots, \pi_K, \Theta_1, \dots, \Theta_K\}$  for all PDFs and each PDF has one additional parameter called a *mixture coefficient*  $\pi_k$ . The problem of MLE for a mixture of densities is investigated in Chapter 3 where EM algorithm and NMF are shown. The likelihood function for mixture of densities is multi-modal (see Fig. 2.1) thus both methods (which are shown as equivalent) iteratively converge to a locally maximal stationary point based on initial guess of parameters  $\hat{\Theta}^{(0)}$ .

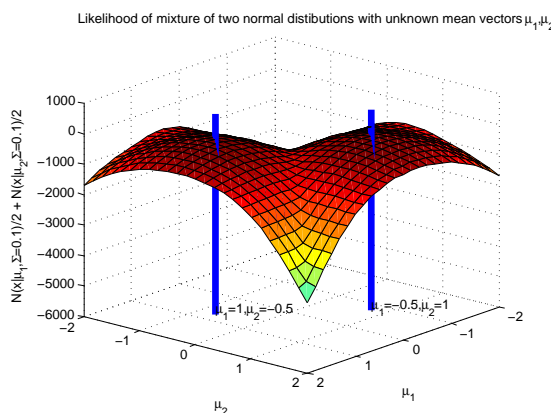


FIGURE 2.1: Example of log-likelihood function for mixture of two univariate Normal PDFs  $\mathcal{N}(x|\mu_1, \Sigma = 0.1)/2 + \mathcal{N}(x|\mu_2, \Sigma = 0.1)/2$ . The estimated parameters are mean vectors  $\mu_1, \mu_2$  and  $\Sigma$  is known and given as  $\Sigma$ , each PDF drew 100 observations. The blue lines depict points where the likelihood is maximal. There are at least two maxima, one for each combination of the means  $(\mu_1, \mu_2)$  and  $(\mu_2, \mu_1)$ .

### 2.1.3 Maximum likelihood estimation for a single function

The MLE has a particular exploitation in statistics where it is used to estimate parameters of PDF based on a set of observations  $\mathbf{X}$ . The MLE is also frequently used in machine learning, mostly for MLE in the mixture of densities in Eq. 2.4, but it is investigated in more detail in the Chapter 3.

There are several ways how to perform MLE. The first and the most straightforward method is an analytic one, which finds values of the  $\hat{\Theta}$  that lie on stationary point (see Fig. 2.2). The analytic method is the most accurate w.r.t. likelihood function, but it is not available for all PDFs. For example, the (multivariate) Normal distribution has analytic form for MLE but the Gamma PDF does not. In case, there is no analytic solution, there are another methods. The first method is numerically approximate the analytic solution with the gradual approaches to the stationary point with a gradient ascent or use some sophisticated second-order methods like

a Newton-Raphson. The second method and the worst is to enumerate parameters and pick the parameter setting with the highest likelihood (log-likelihood).

All PDFs used in this thesis are members of so called exponential family functions and share some useful properties. The most important result is a uni-modality of a likelihood function stated in Theorem 2.6, which guarantees that there is a single stationary point of the likelihood function w.r.t. the parameters  $\Theta$  for a single PDF; thus all PDFs shown here have unique MLE solution  $\hat{\Theta}^{MLE}$  whose likelihood is guaranteed to be maximal.

An illustration of MLE of PDF from the exponential family distribution is shown in Fig. 2.2.

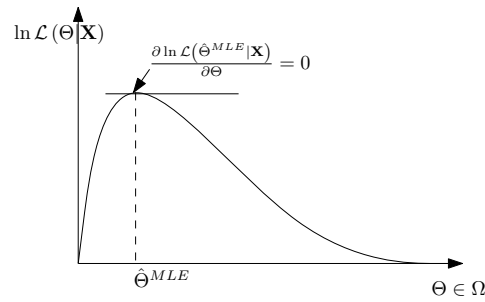


FIGURE 2.2: MLE illustration of uni-modal log-likelihood function.

#### 2.1.4 Number of observations and accuracy of MLE

The quality of MLE depends on the number of observations provided. In the ideal case a set of observations is infinite which is stated in Theorem 2.3, but it is not applicable for real cases, but the cardinality of observations should be large enough. The more data is given the more accurate estimates are.

**Theorem 2.3.** *Having sufficiently many observations  $\mathbf{X} = \{\mathbf{x}\}_{n=1}^N$  drawn from a PDF  $\mathbf{x} \sim p(\mathbf{x}|\Theta)$ . In limit case is possible to find original parameters  $\Theta$  from which observations were drawn, which is formulated as follows:*

$$\lim_{|\mathbf{X}| \rightarrow \infty} \arg \max_{\hat{\Theta} \in \Omega_{\Theta}} \mathcal{L}(\hat{\Theta}|\mathbf{X}) = \lim_{|\mathbf{X}| \rightarrow \infty} \arg \max_{\hat{\Theta} \in \Omega_{\Theta}} \ln \mathcal{L}(\hat{\Theta}|\mathbf{X}) = \Theta \quad (2.5)$$

##### 2.1.4.1 Analytic method

There are basic steps of how to find the MLE in analytic form. The MLE with analytic method is found with the help of known property of stationary points from mathematical analysis of differentiable functions. The points, where function derivative w.r.t. the optimized parameter is zero is the stationary point which lie in a maximum or minimum of a function (the example of stationary point in Fig. 2.2). If the function is uni-modal, then there is one unique solution (exponential family), otherwise, the problem shall be solved by some other methods like gradient ascent or Newton-Raphson method.

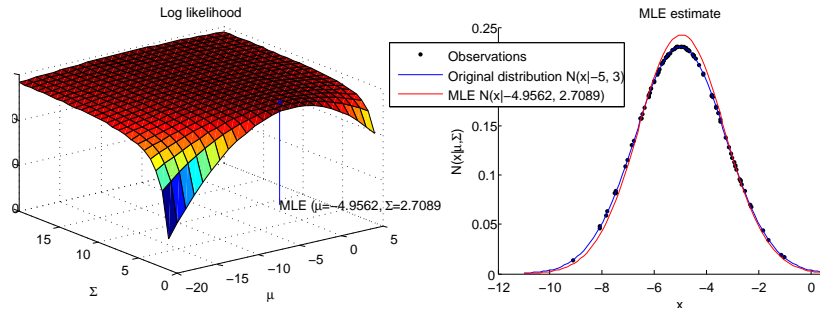


FIGURE 2.3: Illustration of MLE estimation. The graph on the left shows the log-likelihood function of univariate Normal distribution  $\mathcal{N}(x|\mu = -5, \Sigma = 3)$  with 100 observations. The blue line depicts where the log-likelihood is maximal which is MLE solution. In the right chart are shown values of original PDF and PDF with parameters calculated by MLE.

The basic steps of analytic method are as follows: finding the derivative of the likelihood or log-likelihood w.r.t. the parameter and setting it to zero. The resulting equation is solved w.r.t. the parameter  $\theta_k \in \Theta$ . If a solution exists it is the stationary point where the likelihood and log-likelihood are maximal. Generally the first step for likelihood and log-likelihood formulated as follows:

$$\begin{aligned} \frac{\partial \ln \mathcal{L}(\hat{\Theta}|\mathbf{X})}{\partial \theta_k} &= \sum_{n=1}^N \frac{\partial}{\partial \theta_k} \ln f(\mathbf{x}_n|\hat{\Theta}) \quad \hat{\theta}_k \in \hat{\Theta} \\ \text{or} & \\ \frac{\partial \mathcal{L}(\hat{\Theta}|\mathbf{X})}{\partial \theta_k} &= \prod_{n=1}^N \frac{\partial}{\partial \theta_k} f(\mathbf{x}_n|\hat{\Theta}) \quad \hat{\theta}_k \in \hat{\Theta} \end{aligned} \quad (2.6)$$

In the Eq. 2.6 each parameter is treated separately and other parameters are constants. The second step is to find the stationary point from the derivatives in Eq. 2.6, which means solving the equation w.r.t. the parameter  $\theta_k \in \Theta$ :

$$\begin{aligned} \frac{\partial \ln \mathcal{L}(\hat{\Theta}|\mathbf{X})}{\partial \theta_k} &= 0 \quad \hat{\theta}_k \in \hat{\Theta} \\ \text{or} & \\ \frac{\partial \mathcal{L}(\hat{\Theta}|\mathbf{X})}{\partial \theta_k} &= 0 \quad \hat{\theta}_k \in \hat{\Theta} \end{aligned} \quad (2.7)$$

For example, MLE for univariate Normal distribution in Fig. 2.3 is calculated with analytic equations from Eq. 2.38 and 2.39.

#### 2.1.4.2 Numeric type method - Gradient ascent method

The analytic method is not always available for all functions and so some approximate methods must be used instead. An alternative for the analytic method is a gradient ascent which is based on the fact that the gradient of a function  $\mathcal{L}(\Theta|\mathbf{X})$  with  $K$  parameters  $\Theta = [\theta_1, \dots, \theta_K]^2$  gives direction where the likelihood function  $\mathcal{L}(\Theta|\mathbf{X})$  grows the most in a particular position. The gradient of the  $\mathcal{L}(\Theta|\mathbf{X})$  at  $\hat{\Theta}^{(i)}$  is formulated as follows:

<sup>2</sup>The parameters must have some ordering.

$$\nabla_{\hat{\Theta}^{(i)}} \mathcal{L}(\hat{\Theta}|\mathbf{X}) = \left[ \frac{\partial \mathcal{L}(\hat{\Theta}|\mathbf{X})}{\partial \hat{\theta}_1}(\hat{\theta}_1^{(i)}), \dots, \frac{\partial \mathcal{L}(\hat{\Theta}|\mathbf{X})}{\partial \hat{\theta}_K}(\hat{\theta}_K^{(i)}) \right] \quad (2.8)$$

Suppose some initial guess of the parameters  $\hat{\Theta}^{(0)}$ . The gradient  $\nabla_{\hat{\Theta}^{(0)}} \mathcal{L}(\hat{\Theta}|\mathbf{X})$  points to position where the likelihood shall be higher<sup>3</sup> than for the parameters  $\hat{\Theta}^{(0)}$ . By iterative repeats of the previously stated formulates the gradient ascent method as follows:

$$\hat{\Theta}^{(i+1)} = \hat{\Theta}^{(i)} + \alpha \nabla_{\hat{\Theta}^{(i)}} \mathcal{L}(\hat{\Theta}|\mathbf{X}) \quad (2.9)$$

Where the  $\alpha$  is step length of the gradient.

### 2.1.4.3 Numeric type method - Newton-Raphson method

The Newton-Raphson (NR) method is based on an approximation with Taylor series in quadratic form [17]. The quadratic form has one unique stationary point which can be found analytically thus the NR method is fast for quadratic and near quadratic function optimization. The only requirement of the method is to have defined first and second derivatives of the optimized function. Likelihood functions are typically quadratic and, so the NR is relatively fast alternative if no solution for the analytic method exists. The NR is an iterative method; it works with some initial guess of parameters  $\hat{\Theta}^{(0)}$  which are iteratively replaced by newly estimated (usually) better parameters, where the newly estimated parameters adapts according to the stationary point of the current Taylor expansion. At the beginning, the likelihood function is approximated in a single point  $\hat{\Theta}^{(0)}$  (initial guess) as follows:

$$\begin{aligned} \mathcal{L}(\hat{\Theta}|\mathbf{X}) &\approx \tilde{\mathcal{L}}_{\hat{\Theta}^{(0)}}(\hat{\Theta}|\mathbf{X}) \\ &= \mathcal{L}(\hat{\Theta}^{(0)}|\mathbf{X}) (\hat{\Theta} - \hat{\Theta}^{(0)})^T \mathbf{g}_{\hat{\Theta}^{(0)}} \\ &\quad + \frac{1}{2} (\hat{\Theta} - \hat{\Theta}^{(0)})^T \mathbf{H}_{\hat{\Theta}^{(0)}} (\hat{\Theta} - \hat{\Theta}^{(0)}) \end{aligned} \quad (2.10)$$

Where  $\mathbf{g}_{\hat{\Theta}^{(0)}} \equiv \nabla_{\hat{\Theta}^{(0)}} \mathcal{L}(\hat{\Theta}|\mathbf{X})$  is a gradient vector in a point  $\hat{\Theta}^{(0)}$  and  $\mathbf{H}_{\hat{\Theta}^{(0)}} \equiv \nabla_{\hat{\Theta}^{(0)}}^2 \mathcal{L}(\hat{\Theta}|\mathbf{X})$  is a Hessian (matrix of second derivatives) in the same point. Both are coefficients of a quadratic function for which stationary point can be found analytically. The principle is to find stationary point of the quadratic approximation  $\tilde{\mathcal{L}}_{\hat{\Theta}^{(0)}}(\hat{\Theta}|\mathbf{X})$  instead of the likelihood function. Firstly the gradient of the  $\tilde{\mathcal{L}}_{\hat{\Theta}^{(0)}}(\hat{\Theta}|\mathbf{X})$  by the chosen parameters must be taken:

$$\nabla \tilde{\mathcal{L}}_{\hat{\Theta}^{(0)}}(\hat{\Theta}|\mathbf{X}) = \nabla_{\hat{\Theta}^{(0)}} \mathcal{L}(\hat{\Theta}|\mathbf{X}) + (\hat{\Theta} - \hat{\Theta}^{(0)})^T \nabla_{\hat{\Theta}^{(0)}}^2 \mathcal{L}(\hat{\Theta}|\mathbf{X}) \quad (2.11)$$

By setting the result in Eq. 2.11 as zero and solving it, the equation gives the stationary point of the approximation  $\tilde{\mathcal{L}}_{\hat{\Theta}^{(0)}}(\hat{\Theta}|\mathbf{X})$  which is a linear function formulated as follows [17]:

<sup>3</sup>This is in the ideal case, if the length of the gradient update is too large then it may jump over the better likelihood to worse values.

$$\hat{\Theta} = \hat{\Theta}^{(0)} - (\mathbf{g}_{\hat{\Theta}^{(0)}})^T (\mathbf{H}_{\hat{\Theta}^{(0)}})^{-1} \quad (2.12)$$

The equation can be reformulated as iterative procedure:

$$\hat{\Theta}^{(i+1)} = \hat{\Theta}^{(i)} - (\mathbf{g}_{\hat{\Theta}^{(i)}})^T (\mathbf{H}_{\hat{\Theta}^{(i)}})^{-1} \quad (2.13)$$

If the likelihood function is quadratic the approximation is exact and NR method converges to the stationary point in one iteration (see Fig. 2.4, left). If the function is concave then it is guaranteed that the NR method converges to the stationary point (see Fig. 2.4, right). If the function is convex for some  $\Theta$  then the NR method does not always converge to the local maximum. The Eq. 2.13 formulates the NR method as iterative procedure for the likelihood function maximization (the same can be applied for log-likelihood).

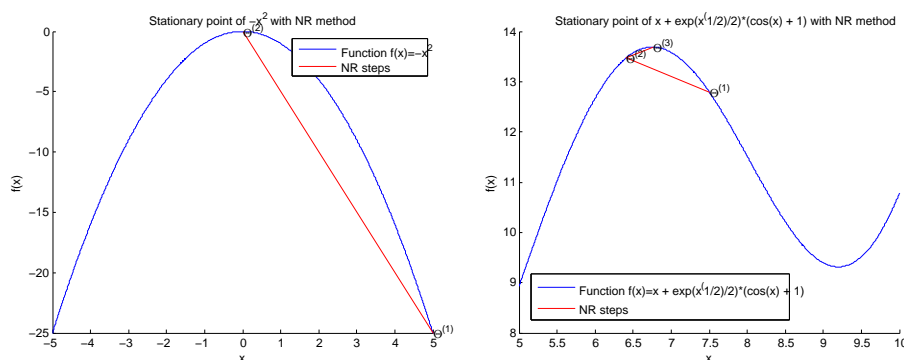


FIGURE 2.4: Newton-Raphson method illustration. Left: quadratic function maximization with the NR. The NR finds stationary point in one iteration. Right: If the function is not quadratic ( $x + \exp\left(\frac{\sqrt{x}}{2}\right) (\cos x + 1)$ ), the method still converges to a stationary point, but in more than one iteration.

All functions shown in this chapter are members of the exponential family for which is guaranteed concavity thus the NR method converges.

#### 2.1.4.4 Summary of MLE

The analytic method gives the most accurate estimation w.r.t. likelihood or log-likelihood, but it is not always necessary. Sometimes some regularization is needed. The regularization problem is a general issue and well known drawback of the MLE. The easiest method is to perform some sampling of the observations to have smaller subset of original observations.

Good alternatives are the gradient ascent and NR methods which tries to find MLE in iterative way. The NR method is always convergent for the functions from the exponential family. Only issue of both methods is that they do not consider inadmissible parameters.



## 2.2 Uni-variate distribution functions

### 2.2.1 Uni-variate Exponential distribution

The Exponential distribution plays an important role in life testing, reliability [18].

#### 2.2.1.1 Probability density function

The PDF of Exponential distribution denoted as  $\text{Exp}(x|\lambda)$  is defined as follows [19]:

$$\text{Exp}(x|\lambda) = \begin{cases} \lambda e^{-\lambda x}, & x \geq 0, \\ 0, & x < 0. \end{cases} \quad (2.14)$$

Where  $\lambda$  is referred as a *rate parameter*. The smaller is the  $\lambda$ , the steeper is function  $\text{Exp}(x|\lambda)$ . The Exponential PDF for various  $\lambda$  setting is shown in Fig. 2.5.

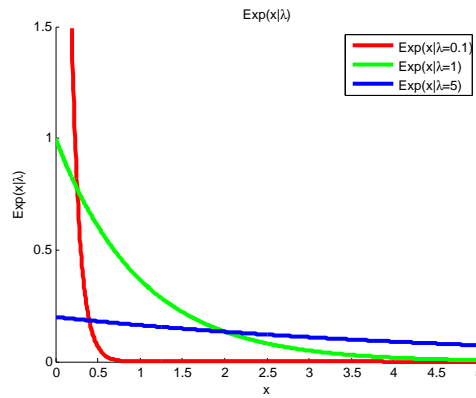


FIGURE 2.5: Values of  $\text{Exp}(x|\lambda)$  for various  $\lambda$  parameter.

#### 2.2.1.2 $\mathcal{L}(\mathbf{x}|\Theta)$ , $\ln \mathcal{L}(\mathbf{x}|\Theta)$ and MLE

Suppose that there is a set of  $N$  independently identically distributed (i.i.d.) observations  $\mathbf{x} = \{x_n\}_{n=1}^N$  drawn from the Exponential distribution  $x \sim \text{Exp}(x|\lambda)$ . The likelihood function for the set of observations  $\mathbf{x}$  is the product of the evaluations of the PDF  $\text{Exp}(x|\lambda)$  for each observation  $x_n \in \mathbf{x}$ :

$$\begin{aligned} \mathcal{L}(\lambda|\mathbf{x}) &= \prod_{n=1}^N \text{Exp}(x_n|\lambda) = \prod_{n=1}^N \lambda \exp(-\lambda x_n) \\ &= \lambda^N \exp\left(-\lambda \sum_{n=1}^N x_n\right) \end{aligned} \quad (2.15)$$

The log-likelihood  $\ln \mathcal{L}(\lambda|\mathbf{x})$  of the Exponential distribution is defined as follows:

$$\begin{aligned} \ln \mathcal{L}(\lambda|\mathbf{x}) &= \sum_{n=1}^N \ln \text{Exp}(x_n|\lambda) \\ &= N \ln \lambda - \lambda \sum_{n=1}^N x_n \end{aligned} \quad (2.16)$$

Taking partial derivative of Eq. 2.16 by the  $\lambda$  gives an analytic MLE which is defined as follows:

$$\hat{\lambda}^{MLE} = \frac{N}{\sum_{n=1}^N x_n} \quad (2.17)$$

## 2.2.2 Uni-variate Gamma distribution

The univariate Gamma distribution denoted as  $\text{Gam}(x|k, \theta)$  can be used in a wide range of disciplines where variances of data are estimated. It gives rise to areas like daily rainfall amounts [20] where the Gamma distribution is successfully applied. Various combinations of distribution parameters can rapidly change the shape of the PDF, so there are many degrees of freedom in combination of parameters to fit desired function. The Gamma distribution is conjugate prior to inverse of variance of the univariate Normal distribution thus it can be applied in Bayesian inference [13].

### 2.2.2.1 Probability density function

The PDF of the Gamma distribution is defined as follows [13]:

$$\text{Gam}(x|k, \theta) = \frac{1}{\theta^k \Gamma(k)} x^{k-1} \exp\left(-\frac{x}{\theta}\right) \quad (2.18)$$

Where  $k$  is referred to as a *shape parameter* and  $\theta$  as a *scale parameter*. The  $\theta$  and  $k$  completely determines the PDF. The Fig. 2.6 depicts how the  $\text{Gam}(x|k, \theta)$  differs for various parameter setting of the  $\theta$  and  $k$ .

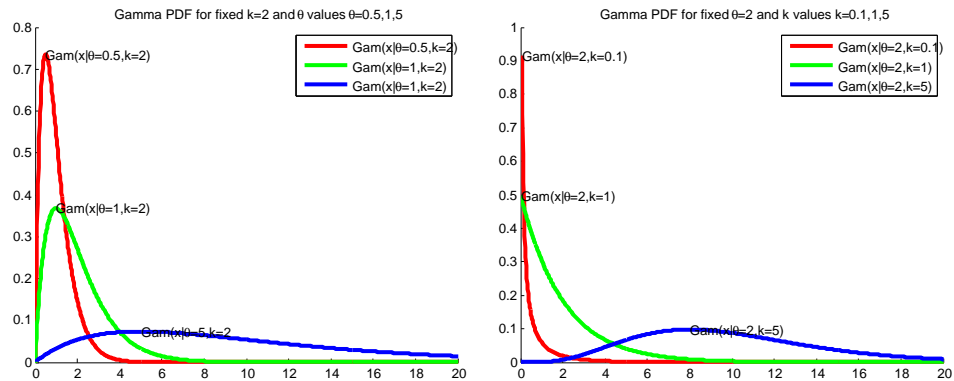


FIGURE 2.6: Left: Values of the  $\text{Gam}(x|k, \theta)$  for fixed  $k = 2$  and variable  $\theta$  in range  $x \in (0, 20)$ ; Right: Values of  $\text{Gam}(x|k, \theta)$  for fixed  $\theta = 2$  and variable  $k$  in range  $x \in (0, 20)$

### 2.2.2.2 $\mathcal{L}(\Theta|\mathbf{x})$ , $\ln \mathcal{L}(\Theta|\mathbf{x})$ and MLE

Suppose that there is a set of  $N$  i.i.d. observations  $\mathbf{x} = \{x_n\}_{n=1}^N$  drawn from Gamma distribution  $x \sim \text{Gam}(x|k, \theta)$ . The likelihood function for the set of observations  $\mathbf{x}$  is a product of the

evaluations of the PDF  $\text{Gam}(x|k, \theta)$  for each observation  $x_n \in \mathbf{x}$ :

$$\begin{aligned}\mathcal{L}(k, \theta|\mathbf{x}) &= \prod_{n=1}^N \text{Gam}(x_n|k, \theta) \\ &= \prod_{n=1}^N \frac{1}{\theta^k \Gamma(k)} x_n^{k-1} \exp\left(-\frac{x_n}{\theta}\right) \\ &= \frac{1}{\theta^{kN} \Gamma(k)^N} \prod_{n=1}^N x_n^{k-1} \exp\left(-\frac{x_n}{\theta}\right)\end{aligned}\quad (2.19)$$

Corresponding log-likelihood function  $\ln \mathcal{L}(k, \theta|\mathbf{x})$  is defined as:

$$\ln \mathcal{L}(k, \theta|\mathbf{x}) = (k-1) \sum_{n=1}^N \ln x_n - \frac{1}{\theta} \sum_{n=1}^N x_n - N \ln \Gamma(k) - Nk \ln \theta \quad (2.20)$$

To obtain MLE, derivatives of the log-likelihood of the Eq. 2.20 have to be taken by the parameters  $\theta$  and  $k$ :

$$\begin{aligned}\frac{\partial \ln \mathcal{L}(k, \theta|\mathbf{x})}{\partial \theta} &= -\frac{Nk}{\theta} + \frac{1}{\theta^2} \sum_{n=1}^N x_n \\ \frac{\partial \ln \mathcal{L}(k, \theta|\mathbf{x})}{\partial k} &= -N \ln \theta - N \underbrace{\frac{\partial \ln \Gamma(k)}{\partial k}}_{\psi(x)} + \sum_{n=1}^N \ln x_n\end{aligned}\quad (2.21)$$

The first equation for the  $\theta$  is solvable analytically thus the MLE for  $\hat{\theta}^{MLE}$  is determined by the following formula:

$$\hat{\theta}^{MLE} = \frac{1}{Nk} \sum_{n=1}^N x_n \quad (2.22)$$

A problem arises with the MLE for the  $k$ . There is no closed form for  $\frac{\partial \ln \Gamma(k)}{\partial k}$ , but this derivative is known as Digamma function  $\psi(k)$  which has no closed form. The  $\psi(x)$  is available in the most of the mathematical packages, alternatively numerical approximation can be used. By substituting  $\theta^{MLE}$  into  $\mathcal{L}(\theta^{MLE}, k|\mathbf{x})$  the following equation is obtained [21]:

$$\ln \mathcal{L}(k, \hat{\theta}^{MLE}|\mathbf{x}) = N(k-1) \sum_{n=1}^N \ln x_n - N \ln \Gamma(k) - Nk \ln \sum_{n=1}^N x_n + Nk \ln k - Nk \quad (2.23)$$

The Eq. 2.23 has no closed form but the MLE can be found iteratively by the gradient ascent or more effectively with the NR method. For the NR method the first and the second partial derivatives by the optimized parameter  $k$  must be calculated. The corresponding derivatives by the  $k$  in Eq. 2.23 are following:

$$\begin{aligned}\frac{\partial \ln \mathcal{L}(\theta^{MLE}, k|\mathbf{x})}{\partial k} &= -N \ln \theta^{MLE} - N\psi^{(0)}(k) + \sum_{n=1}^N \ln x_n \\ \frac{\partial \ln^2 \mathcal{L}(\theta^{MLE}, k|\mathbf{x})}{\partial k^2} &= -N\psi^{(1)}(k)\end{aligned}\quad (2.24)$$

The derivatives in Eq. 2.24 give all the necessary information to substitute unknowns in the NR iterative equation Eq. 2.13 to formulate MLE with NR method. In the most cases the NR method is faster than the gradient ascent for the MLE, so the gradient ascent is shown only for completeness (and also it is intermediate step of NR method). The gradient ascent of the

log-likelihood function Eq. 2.20:

$$\begin{aligned}\hat{k}^{(i+1)} &= \hat{k}^{(i)} + \alpha \frac{\partial \ln \mathcal{L}(\hat{\theta}^{MLE}, \hat{k}^{(i)} | \mathbf{x})}{\partial \hat{k}^{(i)}} \\ &= \hat{k}^{(i)} + \left( -N \ln \theta^{MLE} - N \psi(k) + \sum_{n=1}^N \ln x_n \right)\end{aligned}\quad (2.25)$$

The faster method for maximizing log-likelihood Eq. 2.20 is the NR method, which can be formulated for the parameter  $k$  by substituting the results from Eq. 2.24 into Eq. 2.13 as follows:

$$\begin{aligned}\hat{k}^{(i+1)} &= \hat{k}^{(i)} - \frac{\frac{\partial}{\partial k^{(i)}} \ln \mathcal{L}(\hat{\theta}^{MLE}, \hat{k}^{(i)} | \mathbf{x})}{\frac{\partial^2}{\partial k^{(i)2}} \ln \mathcal{L}(\hat{\theta}^{MLE}, \hat{k}^{(i)} | \mathbf{x})} \\ &= \hat{k}^{(i)} - \frac{-N \ln \theta^{MLE} - N \psi^{(0)}(k) + \sum_{n=1}^N \ln x_n}{-N \psi^{(1)}(k)}\end{aligned}\quad (2.26)$$

Example of MLE solution with NR method formulated in Eq. 2.26 for 1000 random samples drawn from  $x \sim \text{Gam}(x|k=2, \theta=3)$  is shown in Fig. 2.7.

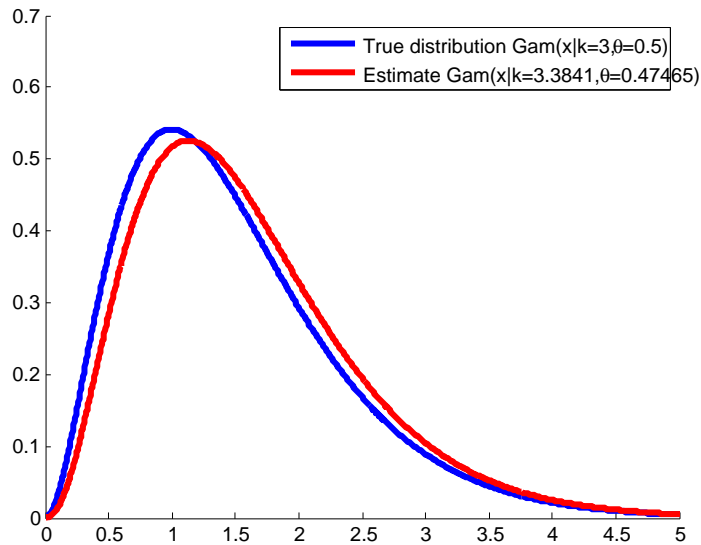


FIGURE 2.7: MLE of Gamma PDF estimated by the NR method with 1000 observations  $\mathbf{x}$ .

## 2.3 Multivariate distributions

### 2.3.1 Von Mises-Fisher distribution

The Von Mises-Fisher denoted as  $\mathcal{MF}(\mathbf{x}|\mu, \kappa)$  is PDF that describes the distribution of observations that lie on a sphere. The spherical constraint of observations means that all values  $\mathbf{x}$  are equidistant from origin with norm 1 ( $\forall \mathbf{x} : \|\mathbf{x}\|_2 = 1$ ).  $\mathcal{MF}$  is practically applicable in domains where the equidistant coordinates are needed (e.g. characteristics of the Earth), where observations are periodic or if only an angle is significant (distance from the origin can be ignored just by equidistant constraint).

### 2.3.1.1 Probability density function

The PDF of  $\mathcal{MF}$  distribution in  $D$  dimensions is defined as follows:

$$\mathcal{MF}(\mathbf{x}|\mu, \kappa) = c_D(\kappa) \exp(\kappa \mu^T \mathbf{x}) \quad \forall \mathbf{x} : \|\mathbf{x}\|_2 = 1 \quad (2.27)$$

Where  $\mu$  is the *mean direction parameter* for which the same property of unit norm as for observations must hold and  $\kappa$  is the *concentration parameter* which characterizes how strongly the unit vectors are drawn in a particular direction  $\mu$ . Larger  $\kappa$  implies higher density of the region in direction of the mean  $\mu$ . The function  $c_D(\kappa)$  is normalizing constant defined as follows: [22]:

$$c_D(\kappa) = \frac{\kappa^{\frac{D}{2}-1}}{(2\pi)^{\frac{D}{2}} I_{\frac{D}{2}-1}(\kappa)} \quad (2.28)$$

The  $I_{\frac{D}{2}-1}(\kappa)$  is modified Bessel function of the first kind with order  $\frac{D}{2}-1$ . The modified Bessel function has no analytic form. For  $D=3$  the  $c_3(\kappa)$  can be formulated [23] in more convenient form as follows:

$$c_3(\kappa) = \frac{\kappa}{\sinh \kappa} \quad (2.29)$$

The Fig. 2.8 shows an example of observations drawn from  $\mathcal{MF}$  in three dimensions.

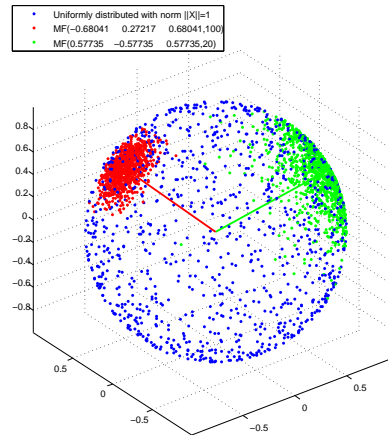


FIGURE 2.8: Blue points are observations uniformly distributed in unit sphere. The red ones are points drawn from  $\mathcal{MF}$  with  $\mu = [-0.6804, 0.2722, 0.6804]$  and  $\kappa = 100$  and green points are drawn from  $\mathcal{MF}$  with  $\mu = [0.5774, -0.5774, 0.5774]$  and  $\kappa = 20$ .

### 2.3.1.2 $\mathcal{L}(\Theta|\mathbf{X})$ , $\ln \mathcal{L}(\Theta|\mathbf{X})$ and MLE

Suppose that there is a set of  $N$  i.i.d. observations  $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$  drawn from the von Mises-Fisher distribution  $\mathbf{x} \sim \mathcal{MF}(x|\mu, \kappa)$ . The likelihood function for the set of observations  $\mathbf{X}$  is a product of the evaluations of the PDF  $\mathcal{MF}(\mathbf{x}|\mu, \kappa)$  for each observation  $\mathbf{x}_n \in \mathbf{X}$ :

$$\begin{aligned} \mathcal{L}(\mu, \kappa|\mathbf{X}) &= \prod_{n=1}^N \mathcal{MF}(\mathbf{x}_n|\mu, \kappa) \\ &= \prod_{n=1}^N c_D(\kappa) \exp(\kappa \mu^T \mathbf{x}_n) \\ &= c_D(\kappa)^N \exp(N\kappa \mu^T \mathbf{r}) \end{aligned} \quad (2.30)$$

Where  $\mathbf{r} = \sum_{n=1}^N \mathbf{x}_n$ . Corresponding log-likelihood function  $\ln \mathcal{L}(\mu, \kappa | \mathbf{x})$  is formulated as follows:

$$\begin{aligned} \ln \mathcal{L}(\mu, \kappa | \mathbf{X}) &= \sum_{n=1}^N \ln \mathcal{MF}(\mathbf{x}_n | \mu, \kappa) \\ &= \sum_{n=1}^N (\ln c_D(\kappa) + \kappa \mu^T \mathbf{x}_n) \\ &= N \ln c_D(\kappa) + N \kappa \mu^T \mathbf{r} \end{aligned} \quad (2.31)$$

By taking partial derivative of Eq. 2.31 by the  $\mu$  the analytic MLE for the  $\hat{\mu}^{MLE}$  is obtained [22]:

$$\hat{\mu}^{MLE} = \frac{\mathbf{r}}{\|\mathbf{r}\|} \quad (2.32)$$

For the parameter  $\kappa$  the MLE is more complicated. The  $\kappa$  occurs in the modified Bessel function as a parameter which requires some advanced analysis. The problem is resolved in [22] where the  $\hat{\kappa}^{MLE}$  is formulated as follows:

$$\hat{\kappa}^{MLE} = \frac{D\bar{r} - \bar{r}^3}{1 - \bar{r}^2} \quad (2.33)$$

### 2.3.2 Multivariate Normal distribution

Multivariate Normal distribution denoted by  $\mathcal{N}(\mathbf{x} | \mu, \Sigma)$  is the most commonly used PDF in density estimation. The  $\mathcal{N}(\mathbf{x} | \mu, \Sigma)$  occurs quite often in statistics thanks to its symmetric bell-shape. There is a strong relation with nature, in other words, the events that occur in nature are usually normally distributed. In mathematics  $\mathcal{N}(\mathbf{x} | \mu, \Sigma)$  is related to the central limit theorem that estimates behaviour of an arithmetic mean which is normally distributed for the i.i.d. for a large number of observations.

The parameter  $\mu$  can be interpreted as a position where the probability is highest thus it can be interpreted as a significant information about observations modelled by  $\mathcal{N}(\mathbf{x} | \mu, \Sigma)$ . The  $\Sigma$  says how the model varies from the mean. The  $\Sigma$  values can be interpreted as the measurement of how much the observations are distant from  $\mu$ , in other words how much are the observations  $\mathbf{X}$  consistent with  $\mu$  in a particular direction. The  $\Sigma$  also specifies the direction of the variances which can be obtained from  $\Sigma$  by eigendecomposition. The eigenvectors with the highest eigenvalues are directions with the highest variance (the Fig. 2.9 depicts that the blue eigenvector with highest eigenvalue points to the direction with the highest variance).

#### 2.3.2.1 Probability density function

The PDF of multivariate Normal distribution in  $D$  dimensions is defined as follows:

$$\mathcal{N}(\mathbf{x} | \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^D \det \Sigma}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1}(\mathbf{x} - \mu)\right) \quad (2.34)$$

Where the  $\mu = [\mu_1, \dots, \mu_D]^T$  is a parameter called *mean*. The mean vector determines position of mode of the distribution and the  $\Sigma \in \mathbb{R}^{D,D}$  is a symmetric positive definite *covariance matrix* that determines size and orientation of the Gaussian.

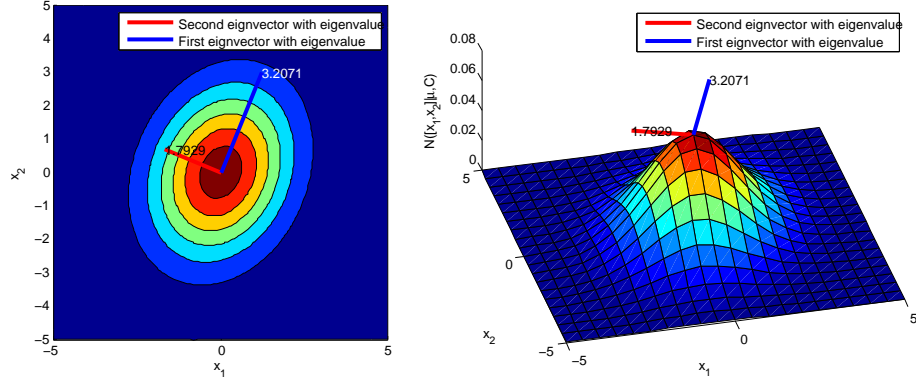


FIGURE 2.9: Multivariate Normal PDF with parameter setting  $\mu = [0, 0]^T$  and  $\Sigma = \begin{bmatrix} 2 & \frac{1}{2} \\ \frac{1}{2} & 3 \end{bmatrix}$ . The eigenvectors are  $\mathbf{v}_1 = [0.38270, 0.9239]^T$ ,  $\mathbf{v}_2 = [-0.9239, 0.3827]^T$ , and their corresponding eigenvalues are  $d_1 = 3.2071$ ,  $d_2 = 1.7929$ .

### 2.3.2.2 $\mathcal{L}(\Theta|\mathbf{X})$ , $\ln \mathcal{L}(\Theta|\mathbf{X})$ and MLE

Suppose that there is a set of  $N$  i.i.d. observations  $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$  drawn from the multivariate Normal distribution  $\mathbf{x} \sim \mathcal{N}(\mathbf{x}|\mu, \Sigma)$ . The likelihood function for the set of observations  $\mathbf{X}$  is a product of the evaluations of the PDF  $\mathcal{N}(\mathbf{x}|\mu, \Sigma)$  for each observation  $\mathbf{x}_n \in \mathbf{X}$ :

$$\begin{aligned} \mathcal{L}(\mu, \Sigma|\mathbf{X}) &= \prod_{n=1}^N \mathcal{N}(\mathbf{x}_n|\mu, \Sigma) \\ &= \frac{1}{(\sqrt{(2\pi)^D \det \Sigma})^N} \exp\left(-\frac{1}{2} \sum_{n=1}^N (\mathbf{x}_n - \mu)^T \Sigma^{-1} (\mathbf{x}_n - \mu)\right) \end{aligned} \quad (2.35)$$

Corresponding log-likelihood function  $\ln \mathcal{L}(\mu, \Sigma|\mathbf{X})$  based on the likelihood in Eq. 2.35 is defined as follows:

$$\begin{aligned} \ln \mathcal{L}(\mu, \Sigma|\mathbf{X}) &= \sum_{n=1}^N \ln \mathcal{N}(\mathbf{x}_n|\mu, \Sigma) \\ &= -\frac{ND}{2} \ln(2\pi) + \frac{N}{2} \ln \det \Sigma - \frac{1}{2} \sum_{n=1}^N (\mathbf{x}_n - \mu)^T \Sigma^{-1} (\mathbf{x}_n - \mu) \end{aligned} \quad (2.36)$$

The MLE solution for  $\mathcal{N}(\mathbf{x}|\mu, \Sigma)$  is obtained by taking partial derivative of  $\ln \mathcal{L}(\mu, \Sigma|\mathbf{X})$  by the parameters  $\mu$  and  $\Sigma$ . Following equations are the derivatives of the the  $\ln \mathcal{L}(\Theta|\mathbf{X})$  [24]:

$$\begin{aligned} \frac{\partial \ln \mathcal{L}(\mu, \Sigma|\mathbf{X})}{\partial \mu} &= \hat{\Sigma}^{-1} \sum_{n=1}^N (\mathbf{x}_n - \hat{\mu}) \\ \frac{\partial \ln \mathcal{L}(\mu, \Sigma|\mathbf{X})}{\partial \Sigma} &= \sum_{n=1}^N \left( \Sigma^{-1} + \underbrace{\Sigma^{-1} \mathbf{x}_n \mathbf{x}_n^T \Sigma^{-1} - \frac{1}{2} (\Sigma^{-1} \mathbf{x}_n \mathbf{x}_n^T \Sigma^{-1} - \Sigma^{-1}) \cdot \mathbf{I}}_{\text{diagonal}} \right) \end{aligned} \quad (2.37)$$

The symbol  $\cdot$  stands for element-wise multiplication (also referred as Hadamard product) and  $\mathbf{I}$  is a  $D - by - D$  identity matrix. The MLE for  $\hat{\mu}^{MLE}$  can be obtained by multiplying Eq. 2.37 with  $\hat{\Sigma}$  and rearranging terms. The analytic MLE of the parameter  $\hat{\mu}^{MLE}$  is defined as follows:

$$\hat{\mu}^{MLE} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n \quad (2.38)$$

Some advanced algebra operations are required to obtain  $\hat{\Sigma}^{MLE}$ . The analytic MLE for the parameter  $\hat{\Sigma}^{MLE}$  is defined as follows [13]:

$$\hat{\Sigma}^{MLE} = \frac{1}{N} \sum_{n=1}^N (\mathbf{x}_n - \mu) (\mathbf{x}_n - \mu)^T \quad (2.39)$$

### 2.3.3 Multivariate Log-normal distribution

The symmetry of the Normal distribution is not always desired property and the skewness of the distribution with similar properties as  $\mathcal{N}(\mathbf{x}|\mu, \Sigma)$  is useful. The Log-normal distribution denoted as  $\mathcal{LN}(\mathbf{x}|\mu, \Sigma)$  is the PDF whose logarithm is the Normal distribution. The logarithm property changes the skewness of the distribution from  $\mathcal{N}(\mathbf{x}|\mu, \Sigma)$  and the resulting distribution is asymmetric, which is so called *long tailed* (see difference Fig. 2.10).

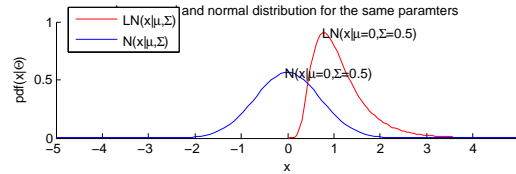


FIGURE 2.10: The Log-Normal and Normal distributions with the same parameters  $\mu = 0, \Sigma = 0.5$ .

#### 2.3.3.1 Probability density function

The PDF of multivariate Log-normal distribution  $\mathcal{LN}(\mathbf{x}|\mu, \Sigma)$  in  $D$  dimensions is defined as follows:

$$\mathcal{LN}(\mathbf{x}|\mu, \Sigma) = \frac{1}{(2\pi \det \Sigma)^{\frac{D}{2}} \prod_{n=1}^N x_i} \exp\left(\frac{1}{2} (\ln \mathbf{x} - \mu)^T \Sigma^{-1} (\ln \mathbf{x} - \mu)\right) \quad (2.40)$$

Due to the logarithm of random vector the admissible vectors are only positive values. The parameters in Eq. 2.40 have the same names as in multivariate Normal distribution, but different meaning which is caused by the logarithm property. The  $\mu$  is not in the mode of the PDF and the  $\Sigma$  has not the same meaning as in case of the Normal distribution. Skewness of the  $\mathcal{LN}(\mathbf{x}|\mu, \Sigma)$  causes that the eigenvectors does not reflect the directions of variances as in case of the  $\mathcal{N}(\mathbf{x}|\mu, \Sigma)$ .

#### 2.3.3.2 $\mathcal{L}(\Theta|\mathbf{x})$ , $\ln \mathcal{L}(\Theta|\mathbf{x})$ and MLE

Suppose that there is a set of  $N$  i.i.d. observations  $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$  drawn from the multivariate Log-normal distribution  $\mathbf{x} \sim \mathcal{LN}(\mathbf{x}|\mu, \Sigma)$ . The likelihood function for the set of observations  $\mathbf{X}$  is the product of the evaluations of the PDF  $\mathcal{LN}(\mathbf{x}|\mu, \Sigma)$  for each observation  $\mathbf{x}_n \in \mathbf{X}$ :

$$\begin{aligned} \mathcal{L}(\mu, \Sigma|\mathbf{X}) &= \prod_{n=1}^N \mathcal{LN}(\mathbf{x}_n|\mu, \Sigma) = \\ &= \prod_{n=1}^N \frac{1}{(2\pi \det \Sigma)^{\frac{D}{2}} \prod_{d=1}^D x_{d,n}} \exp\left(\frac{1}{2} (\ln \mathbf{x}_n - \mu)^T \Sigma^{-1} (\ln \mathbf{x}_n - \mu)\right) \\ &= \frac{1}{(2\pi \det \Sigma)^{\frac{ND}{2}} \prod_{n=1}^N \prod_{d=1}^D x_{d,n}} \exp\left(\frac{1}{2} (\ln \mathbf{x}_n - \mu)^T \Sigma^{-1} (\ln \mathbf{x}_n - \mu)\right) \end{aligned} \quad (2.41)$$



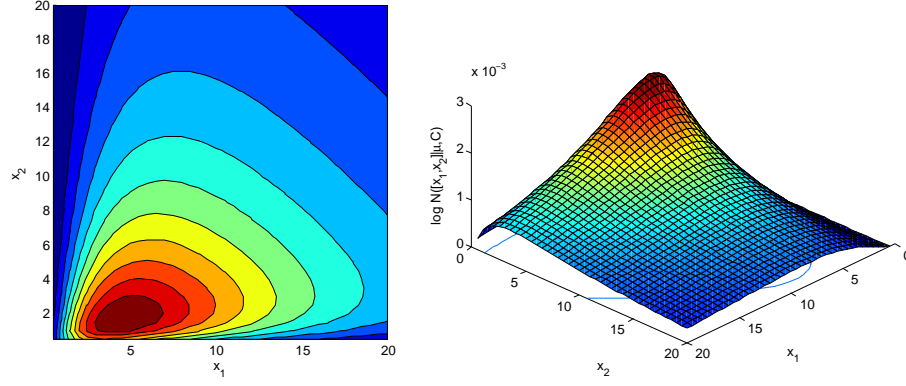


FIGURE 2.11: Probability density function of multivariate Log-normal distribution with  $\mu = [3, 3]^T$  and  $[[1, \frac{1}{2}]; [\frac{1}{2}, 2]]$ .

Corresponding log-likelihood function  $\ln \mathcal{L}(\mu, \Sigma | \mathbf{X})$  based on the likelihood Eq. 2.41 is defined as follows:

$$\begin{aligned}
 \ln \mathcal{L}(\Theta | \mathbf{X}) &= \sum_{n=1}^N \ln \mathcal{LN}(\mathbf{x} | \mu, \Sigma) = \\
 &= -\frac{ND}{2} \ln(2\pi \det \Sigma) - \underbrace{\sum_{n=1}^N \sum_{d=1}^D \ln x_{d,n}}_{1^{st} \text{ difference}} + \\
 &+ \sum_{n=1}^N \frac{1}{2} \left( \underbrace{\ln \mathbf{x}_n}_{2^{nd} \text{ difference}} - \mu \right)^T \Sigma^{-1} \left( \underbrace{\ln \mathbf{x}_n}_{2^{nd} \text{ difference}} - \mu \right)
 \end{aligned} \tag{2.42}$$

To obtain the MLE solution for the multivariate Log-normal distribution the results from MLE of the  $\mathcal{N}(\mathbf{x} | \mu, \Sigma)$  can be reused with a substitution. The log-likelihood functions Eq. 2.36 and 2.42 differ in the sum of random variable  $\ln \mathbf{x}_n$  (emphasized in Eq. 2.42 as the 1<sup>st</sup> difference) and logarithm of the random variable  $\mathbf{x}$  in the exponential function (emphasized in Eq. 2.42 as the 2<sup>nd</sup> difference). The first difference is ruled out by taking the derivative by the parameter, thus the difference can be ignored. The second difference can be resolved by substitution  $\mathbf{z} = \ln \mathbf{x}$ , which gives exactly the same  $\ln \mathcal{L}(\mu, \Sigma | \mathbf{X})$  as for  $\mathcal{N}(\mathbf{x} | \mu, \Sigma)$ . The MLE for the  $\mathcal{LN}(\mathbf{x} | \mu, \Sigma)$  after back-substitution of the  $\mathbf{z}$  is formulated as follows:

$$\begin{aligned}
 \hat{\mu}^{MLE} &= \frac{1}{N} \sum_{n=1}^N \ln \mathbf{x}_n \\
 \hat{\Sigma}^{MLE} &= \frac{1}{N} \sum_{n=1}^N (\ln \mathbf{x}_n - \mu)^T (\ln \mathbf{x}_n - \mu)
 \end{aligned} \tag{2.43}$$

### 2.3.4 Wishart distribution

The Wishart distribution denoted by  $\mathcal{W}(\mathbf{S} | N, \Sigma)$  is the multidimensional version of the chi-square distribution. The Wishart distribution characterizes the covariance matrix of the observations drawn from  $\mathcal{N}(\mathbf{x} | \mu, \Sigma)$  from a scattering matrix [25].

The scattering matrix is an important statistical characteristics which describes how a set of data varies. Computation of the scattering matrix of a set of data  $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$  is defined as

follows:

$$\mathbf{S} = \sum_{n=1}^N (\mathbf{x} - \mu) (\mathbf{x} - \mu)^T \quad (2.44)$$

Where the  $\mu$  stands for arithmetic mean of the data.

**Theorem 2.4.** *If  $\Sigma$  is non-negative definite  $D - by - D$  matrix where  $D > 2$  is dimension and  $N \geq D$  degrees of freedom of Wishart distribution  $\mathcal{W}(\mathbf{S}|N, \Sigma)$ , then for scattering matrix holds following relation:*

$$\mathbf{S} \sim \mathcal{W}(\mathbf{S}|N, \Sigma) \quad (2.45)$$

Where the values  $\{\mathbf{x}_1 \dots \mathbf{x}_N\}_{n=1}^N$  are observations drawn from:

$$\mathbf{x}_n \sim \mathcal{N}(\mathbf{x}_n|0, \Sigma) \quad (2.46)$$

The Theorem 2.4 can be used to measure scattering matrix similarity in a probabilistic way. With framework shown in the Chapter 3 the scattering matrices can be classified by a mixture of Wishart distributions and the Theorem 2.4 is used in the Chapter 6 where the mixture of the Wishart PDFs are used to classify regions of images characterized by the scattering matrices calculated from feature vectors. Generally speaking, the Wishart distribution is useful in cases where the set of observations is drawn from  $\mathcal{N}(\mathbf{x}|0, \Sigma)$  and shares some properties which are reflected in scattering matrices.

### 2.3.4.1 Probability density functions

The PDF of Wishart distribution is defined as follows [13]:

$$\mathcal{W}(\mathbf{S}|N, \Sigma) = \frac{(\det \mathbf{S})^{\frac{N-D-1}{2}} \exp\left(\frac{1}{2} \text{Tr}(\Sigma^{-1} \mathbf{S})\right)}{2^{\frac{ND}{2}} \pi^{\frac{D(D-1)}{4}} (\det \Sigma)^{\frac{N}{2}} \prod_{j=1}^D \Gamma\left(\frac{1}{2}(N-j+1)\right)} \quad (2.47)$$

Where the parameter  $\Sigma$  is called *scale matrix* that specifies the covariance matrix  $\Sigma$  of the original observations  $\mathbf{x} \sim \mathcal{N}(\mathbf{x}|0, \Sigma)$  from which is the scattering matrix  $\mathbf{S}$  calculated. The  $N$  is called *degrees of freedom* which specify how many observations  $\mathbf{x}$  are used to calculate scattering matrix  $\mathbf{S}$ .

The visualisation of the  $\mathcal{W}(\mathbf{S}|N, \Sigma)$  for different parameter setting is not that straightforward as in other distributions, because the random variable is a matrix with at least  $2 - by - 2$  elements. Some visualisation methods, especially for matrix distributions are described in paper [26].

The Wishart PDF as it is given in Eq. 2.47 has a problem with MLE because the  $N$  occurs in the  $\Gamma(x)$  function. Fortunately, there is a Wishart distribution which does not require the  $N$ . The distribution is called simplified Wishart distribution. The simplified Wishart distribution is formulated as follows [6]:

$$\mathcal{W}(\mathbf{S}|\Sigma) = (\pi)^{-3} (\det \Sigma)^{-1} \exp(-\text{tr}(\mathbf{S}\Sigma^{-1})) \quad (2.48)$$

Since this moment the Wishart distribution is supposed to be formulated in simplified form in Eq. 2.48.

### 2.3.4.2 $\mathcal{L}(\Theta|\mathbb{S})$ , $\ln \mathcal{L}(\Theta|\mathbb{S})$ and MLE

Suppose that there are  $N$  i.i.d. scattering matrices  $\mathbb{S} = \{\mathbf{S}_1, \dots, \mathbf{S}_N\}$  calculated from the set of arbitrary number of observations drawn from the multivariate Normal distribution  $\mathbf{x} \sim \mathcal{N}(\mathbf{x}|0, \Sigma)$ . The likelihood function for the set of observations  $\mathbb{S}$  is the product of the evaluations of the PDFs  $\mathcal{W}(\mathbf{S}_n|\Sigma)$  for each scattering matrix  $\mathbf{S}_n$  from a set of scattering matrices  $\mathbb{S}$ :

$$\begin{aligned} \mathcal{L}(\Sigma|\mathbb{S}) &= \prod_{n=1}^N \mathcal{W}(\mathbf{S}_n|\Sigma) \\ &= \prod_{n=1}^N (\pi)^{-3} (\det \Sigma)^{-1} \exp(-\text{tr}(\mathbf{S}_n \Sigma^{-1})) \\ &= (\pi)^{-3N} (\det \Sigma)^{-N} \exp\left(-\sum_{n=1}^N \text{tr}(\mathbf{S}_n \Sigma^{-1})\right) \end{aligned} \quad (2.49)$$

Corresponding log-likelihood function of the Eq. 2.49 is

$$\begin{aligned} \ln \mathcal{L}(\Sigma|\mathbb{S}) &= \sum_{n=1}^N \ln \mathcal{W}(\mathbf{S}_n|\Sigma) \\ &= -3N \ln \pi - N \ln \det \Sigma - \sum_{n=1}^N \text{tr}(\mathbf{S}_n \Sigma^{-1}) \end{aligned} \quad (2.50)$$

The MLE for  $\hat{\Sigma}^{MLE}$  has an analytic solution that is arithmetic average of scattering matrices [6]:

$$\hat{\Sigma}^{MLE} = \frac{\sum_{n=1}^N \mathbf{S}_n}{N} \quad (2.51)$$

## 2.3.5 Dirichlet distribution

The Dirichlet distribution denoted as  $\mathcal{D}(\mathbf{x}|\mathbf{a})$  is a multivariate extension of beta distribution. One application area where the Dirichlet has shown as useful is in modelling of the distribution of words in the text documents [15] modelled by mixture of Dirichlet densities. The sum of random vector  $\mathbf{x}$  must be 1 thus the PDF forms  $D - 1$  dimensional simplex (see Fig. 2.12). The Dirichlet distribution is conjugate prior to multinomial distribution which is useful in Bayesian inference.

### 2.3.5.1 Probability density function

The PDF of Dirichlet distribution is defined as follows [15]:

$$\mathcal{D}(\mathbf{x}|\mathbf{a}) = \frac{\Gamma\left(\sum_{d=1}^D a_d\right)}{\prod_{d=1}^D \Gamma(a_d)} \prod_{d=1}^D x_d^{a_d-1} \quad (2.52)$$

Where the parameter  $\mathbf{a}$  is  $D$  dimensional column vector where all elements must be positive and for all random vectors  $\sum_{d=1}^D x_d = 1$  must hold. The Dirichlet PDF with various parameter settings is shown in Fig. 2.12.

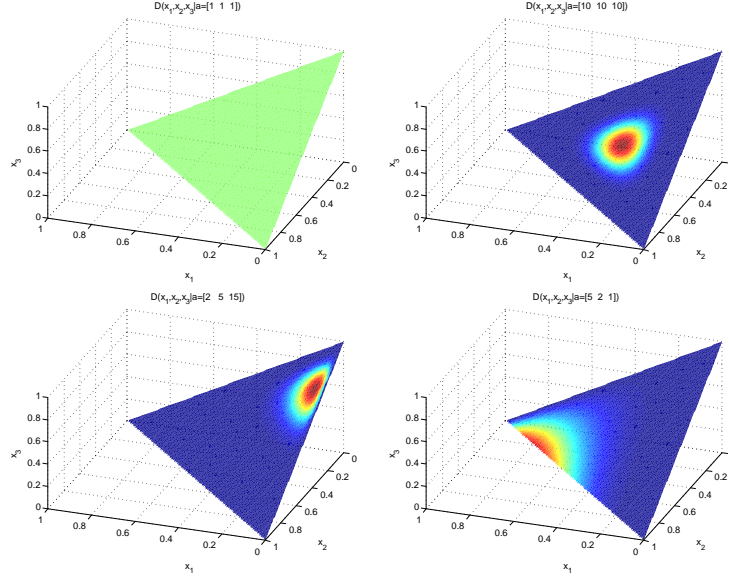


FIGURE 2.12: The  $\mathcal{D}(\mathbf{x}|\mathbf{a})$  for different parameter  $\mathbf{a}$  were:  $\mathbf{a} = [1, 1, 1]^T$  (left upper),  $\mathbf{a} = [10, 10, 10]^T$  (right upper),  $\mathbf{a} = [2, 5, 15]^T$  (left bottom) and  $\mathbf{a} = [5, 2, 1]^T$  (right bottom).

### 2.3.5.2 $\mathcal{L}(\Theta|\mathbf{X})$ , $\ln \mathcal{L}(\Theta|\mathbf{X})$ and MLE

Suppose there is a set of  $N$  i.i.d. observations  $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$  drawn from the Dirichlet distribution  $\mathbf{x} \sim \mathcal{D}(\mathbf{x}|\mathbf{a})$ . The likelihood function for the set of observations  $\mathbf{X}$  is the product of the evaluations of the PDF  $\mathcal{D}(\mathbf{x}|\mathbf{a})$  for each observation  $\mathbf{x}_n \in \mathbf{X}$

$$\begin{aligned}
 \mathcal{L}(\mathbf{a}|\mathbf{X}) &= \prod_{n=1}^N \mathcal{D}(\mathbf{x}_n|\mathbf{a}) \\
 &= \prod_{n=1}^N \frac{\Gamma(\sum_{d=1}^D a_d)}{\prod_{d=1}^D \Gamma(a_d)} \prod_{d=1}^D x_d^{a_d-1} \\
 &= \left( \frac{\Gamma(\sum_{d=1}^D a_d)}{\prod_{d=1}^D \Gamma(a_d)} \right)^N \prod_{n=1}^N \prod_{d=1}^D x_{d,n}^{a_d-1}
 \end{aligned} \tag{2.53}$$

Corresponding log-likelihood function  $\ln \mathcal{L}(\mathbf{a}|\mathbf{X})$  based on the likelihood Eq. 2.54

$$\begin{aligned}
 \ln \mathcal{L}(\mathbf{a}|\mathbf{X}) &= \sum_{n=1}^N \ln \mathcal{D}(\mathbf{x}_n|\mathbf{a}) \\
 &= N \ln \Gamma \left( \sum_{d=1}^D a_d \right) - N \sum_{d=1}^D \ln \Gamma(a_d) + \sum_{n=1}^N \sum_{d=1}^D (a_d - 1) \ln x_{d,n}
 \end{aligned} \tag{2.54}$$

The  $\Gamma(x)$  again prevents from obtaining the analytic solution for the MLE. The  $\mathcal{D}(\mathbf{x}|\mathbf{a})$  is a member of the exponential family thus there is one unique MLE solution that can be found iteratively by gradient or NR method. Taking partial derivative of the log-likelihood function Eq. 2.54 by one of the parameters  $a_d \in \mathbf{a}$  gives relation for gradient ascent method:

$$\frac{\partial}{\partial a_d} \ln \mathcal{L}(\mathbf{a}|\mathbf{X}) = N\psi \left( \sum_{d=1}^D a_d \right) - N\psi(a_d) + \sum_{n=1}^N \ln x_{d,n} \tag{2.55}$$

The gradient from Eq. 2.55 can be used to find MLE solution iteratively and for each  $a_d \in \mathbf{a}$  separately. The iterative equation for each  $a_d$  is defined as follows:

$$\begin{aligned}\hat{a}_d^{(i+1)} &= \hat{a}_d^{(i)} + \alpha \frac{\partial}{\partial a_d} \ln \mathcal{L}(\Theta_k | \mathbf{X}) \\ &= \hat{a}_d^{(i)} + \alpha \left( N\psi \left( \sum_{d=1}^D \hat{a}_d^{(i)} \right) - N\psi \left( \hat{a}_d^{(i)} \right) + \sum_{n=1}^N \ln x_{d,n} \right)\end{aligned}\quad (2.56)$$

Where the  $\alpha$  is a constant that regulates gradient step length.

## 2.4 Exponential family distributions

All distribution functions shown so far are members of family of distribution functions called exponential. The family of exponential PDFs has some important properties that can be derived from general formulation of the PDF. The most important results that come from the exponential family is stated in Theorem 2.6 which claims that all likelihood and log-likelihood functions are uni-modal.

**Definition 2.5.** A distribution function is member of exponential family if the PDF can be formulated as following exponential function [27]:

$$p_{\mathcal{E}\mathcal{X}\mathcal{P}}(\mathbf{x}|\Theta) = h(\mathbf{x}) \exp(\Theta^T \mathbf{T}(\mathbf{x}) - A(\Theta)) \quad (2.57)$$

Where  $\mathbf{T}(\mathbf{x})$  is a vector function of *sufficient statistics*,  $A(\Theta)$  is the coefficient that ensures that sum/integral over PDF is one and  $h(\mathbf{x})$  is arbitrary function of variable  $\mathbf{x}$ .

There are three the most important theorems that come from the exponential family. The first and the second derivative (moments) and the MLE for Eq. 2.57. First and second results are not important in context of our problems but third is crucial because it guarantees the uni-modality of likelihood function.

First of all, the log-likelihood for a set of i.i.d observations  $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$  drawn from the  $\mathbf{x} \sim p_{\mathcal{E}\mathcal{X}\mathcal{P}}(\mathbf{x}|\Theta)$  that is a member of exponential family. The log-likelihood of the set of observations is defined as follows:

$$\ln \mathcal{L}(\Theta | \mathbf{X}) = \sum_{n=1}^N \ln h(\mathbf{x}_n) + \Theta^T \mathbf{T}(\mathbf{x}_n) - A(\Theta) \quad (2.58)$$

To obtain maxima of the log-likelihood function, gradient  $\nabla_{\Theta}$  of the Eq. 2.58 is taken:

$$\nabla_{\Theta} \ln \mathcal{L}(\Theta | \mathbf{X}) = -N \nabla_{\Theta} A(\Theta) + \sum_{n=1}^N \mathbf{T}(\mathbf{x}_n) \quad (2.59)$$

Rearrangement of the terms gives:

$$\nabla_{\Theta} A(\Theta) = \frac{1}{N} \sum_{n=1}^N \mathbf{T}(\mathbf{x}_n) \quad (2.60)$$

The Eq. 2.60 says that MLE solution depends only on terms  $\mathbf{T}(\mathbf{x})$  which are called *sufficient statistics*. The Eq. 2.60 gives general form for MLE of an arbitrary PDF from the exponential family [27].

**Theorem 2.6.** *In exponential families the log-likelihood function has at most one local maximum in  $\hat{\Theta}^{MLE}$ . This is then equal to the global maximum and determined by the unique solution to the Eq. 2.60 w.r.t.  $\Theta$ . [28]*

The basis of proof of the Theorem 2.6 is hidden in the second gradient of the Eq. 2.60 which guarantees that the function is strictly concave [28].

## Chapter 3

# Neural Modeling Fields and Expectation maximization algorithm

In this chapter, the Neural Modeling Fields (NMF) and Expectation Maximisation algorithm (EM) are shown. The both find the MLE solution of a mixture of density functions mentioned briefly in Chapter 2 in Eq. 2.4. The literature about the NMF [6, 29] does not strictly specify whether the learning equations of the NMF are based on the EM or not; nevertheless the, both approaches are equivalent which is later proved in this chapter. Moreover, the EM algorithm is better developed from the theoretical perspective so the notion used is mostly based on the EM algorithm.

In the first section, the EM algorithm is introduced, including derivation of the equations that find the MLE for the densities introduced in Chapter 2. In the second section, the NMF are shown with discussion about similarity measures [6] and proof of equivalence of the NMF and the EM. The derived relation between the EM and MLE stated in Theorem 3.5 opened entire problem of the density estimation as widely applicable approach. Further a Maximum Likelihood Artificial Neural System (MLANS) is shown, which is particular realization of the NMF for multivariate Normal distribution.

To avoid confusion, in Chapter 2, all PDFs are members of the exponential family thus their likelihood function is uni-modal which is guaranteed by Theorem 2.6, but the theorem does not guarantee the uni-modality of the likelihood functions for the mixtures of densities Eq. 3.1. Simple multi-modality is illustrated in Fig. 2.1 where mixture of two univariate Normal distributions has two optima.

### 3.1 EM algorithm

The origin of the EM algorithm dates back to 1977 when the algorithm that finds MLE solution for mixtures of densities function was published [10]. The proposed algorithm is an iterative procedure of finding MLE of the parameters  $\Theta$  of an underlying distribution  $p(\mathbf{x}|\Theta)$  that converges local the maximum of the marginal *a posteriori* probability  $p(\Theta|\mathbf{x}) = p(\mathbf{x}|\Theta)p(\Theta)$  [30]. The algorithm proceeds in two steps: **E** step that stands for *expectation* phase where a conditional expectation in Eq. 3.4 is calculated and **M** that stands for *maximization* step where the parameters  $\hat{\Theta}^{(i)}$  are updated such that the conditional expectation Eq. 3.4 is maximal.

The mixture of densities is formulated as a sum of  $K$  PDFs weighted by coefficients  $\pi$ :

$$p(\mathbf{x}|\Theta) = \sum_{k=1}^K \pi_k f_k(\mathbf{x}|\Theta_k) \quad (3.1)$$

The purpose of the algorithm is to estimate a set of parameters  $\Theta = \{\pi_1 \dots \pi_K, \Theta_1, \dots, \Theta_K\}$  of an underlining distribution of a mixture of densities formulated in Eq. 3.1.

The weighting parameters  $\pi_1, \dots, \pi_K$  are general for all types of density functions and they are called *mixture coefficients*. The  $\pi_k$  can be interpreted as a prior  $\pi_k \equiv (\Theta_k)$  of a PDF in a mixture of densities.

**Definition 3.1.** A conditional expectation w.r.t. a conditional distribution  $p(x|y)$  of a  $X$  for given  $y$  is defined as follows [13]:

$$\mathbb{E}[f(x)|y] = \sum_{x \in \Omega_x} p(x|y) f(x) \quad (3.2)$$

The EM algorithm in general formulation supposes the existence of unobserved data  $\mathbf{y} = \{y_n\}_{n=1}^N$  for each observed datum  $\mathbf{x}_n \in \mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$ . The observed and unobserved data are called *complete data*  $\{\mathbf{X}, \mathbf{y}\} = \{\mathbf{x}_n, y_n\}_{n=1}^N$ . For the complete data the joint density  $p(\mathbf{x}, \mathbf{y}|\Theta)$  function is defined as follows [31]:

$$p(\mathbf{x}, \mathbf{y}|\Theta) = p(\mathbf{y}|\mathbf{x}, \Theta) p(\mathbf{x}|\Theta) \quad (3.3)$$

The joint density function in Eq. 3.3 is used to construct *complete log-likelihood*  $\ln \mathcal{L}(\Theta|\mathbf{X}, \mathbf{y})$  functions. The unobserved data  $\mathbf{y}$  are not given, but their probabilities based on current parameter estimates are used instead thus the unobserved are treated as a random variable.

The principle of EM algorithm is to maximize expected value of the complete data log-likelihood  $\mathcal{L}(\Theta|\mathbf{X}, \mathbf{y})$  based on the complete data  $\{\mathbf{X}, \mathbf{y}\}$ . The computation is formulated as so called *Q-function* (the conditional expectation of the complete log-likelihood) [31]:

$$Q(\hat{\Theta}^{(i+1)}, \hat{\Theta}^{(i)}) = \mathbb{E} \left[ \ln \mathcal{L}(\hat{\Theta}^{(i+1)}|\mathbf{X}, \mathbf{y}) \mid \mathbf{X}, \hat{\Theta}^{(i)} \right] \quad (3.4)$$



The conditional expectation in Eq. 3.4 with help of Def. 3.1 can be expanded in following form [31]:

$$\begin{aligned} \mathbb{E} \left[ \ln \mathcal{L} \left( \hat{\Theta}^{(i+1)} | \mathbf{X}, \mathbf{y} \right) | \mathbf{X}, \hat{\Theta}^{(i)} \right] &= Q \left( \hat{\Theta}^{(i+1)}, \hat{\Theta}^{(i)} \right) \\ &= \underbrace{\sum_{\bar{\mathbf{y}} \in \Omega_{\mathbf{y}}} p \left( \bar{\mathbf{y}} | \mathbf{X}, \hat{\Theta}^{(i)} \right)}_{\text{marginalization}} \underbrace{\ln \mathcal{L} \left( \hat{\Theta}^{(i+1)} | \mathbf{X}, \bar{\mathbf{y}} \right)}_{\text{complete log-likelihood}} \end{aligned} \quad (3.5)$$

The reason why the estimation of  $\hat{\Theta}$  does not come out only from observed data is due to the sum of densities inside the logarithm of incomplete log-likelihood  $\ln \mathcal{L}(\Theta | \mathbf{X})$ .

$$\ln \mathcal{L} \left( \hat{\Theta} | \mathbf{X} \right) = \sum_{n=1}^N \ln p \left( \mathbf{x}_n | \hat{\Theta} \right) = \sum_{n=1}^N \ln \sum_{k=1}^K \hat{\pi}_k f_k \left( \mathbf{x}_n | \hat{\Theta}_k \right) \quad (3.6)$$

The  $\ln \mathcal{L} \left( \hat{\Theta} | \mathbf{X} \right)$  is useful as measure of quality of current estimate for the mixture, but useless for estimation. Maximization of the  $\ln \mathcal{L}(\Theta | \mathbf{X})$  does not give the general formula for varying  $K$  as the  $Q$ -function does.

Suppose for a moment that it is known which of the distributions from mixture Eq. 3.1 drew the datum  $\mathbf{x}_n$ . The distribution that drew is determined by unobserved data  $\mathbf{y} = \{y_n\}_{n=1}^N$  where the value  $y_n$  is equal to  $k$  if and only if the datum  $\mathbf{x}_n$  was drawn from  $k$ -th PDF in the mixture of  $K$  densities. If  $y_n$ -s was known the calculation would be straightforward because each PDF  $k$  in the mixture Eq. 3.1 would compute separate MLE as it is described in Chapter 2.

The algorithm works iteratively with some initial guess of parameters  $\hat{\Theta}^{(0)}$ . The guess  $\hat{\Theta}^{(0)}$  together with observed data  $\mathbf{X}$  via Bayes formula can give current probability of the assignment  $y_n = k$ . The formula Eq. 3.3 gives answer to  $p(y | \mathbf{x}, \Theta)$ . The probability of unobserved variable  $y_n$  can be interpreted as the degree of membership of the  $\mathbf{x}_n$  in a density function  $k$  in mixture:

$$p \left( y_n = k | \mathbf{x}_n, \hat{\Theta}^{(i)} \right) = \frac{\hat{\pi}_k^{(i)} f_k \left( \mathbf{x}_n | \hat{\Theta}_k^{(i)} \right)}{p \left( \mathbf{x}_n | \hat{\Theta}^{(i)} \right)} = \frac{\hat{\pi}_k^{(i)} f_k \left( \mathbf{x}_n | \hat{\Theta}_k^{(i)} \right)}{\sum_{j=1}^K \hat{\pi}_j^{(i)} f_j \left( \mathbf{x}_n | \hat{\Theta}_j^{(i)} \right)} \quad (3.7)$$

Because observed data are supposed to be drawn independently so the probability of complete data  $p(\mathbf{y} | \mathbf{X}, \hat{\Theta})$  is calculated as a set of independent events as follows:

$$p \left( \mathbf{y} | \mathbf{X}, \hat{\Theta} \right) = \prod_{n=1}^N p \left( y_n | \mathbf{x}_n, \hat{\Theta} \right) \quad (3.8)$$

The log-likelihood of the complete data  $\{\mathbf{X}, \mathbf{y}\}$  is defined as follows:

$$\ln \mathcal{L}(\Theta | \mathbf{X}, \mathbf{y}) = \sum_{n=1}^N \ln p(\mathbf{x}_n, y_n | \Theta) = \sum_{n=1}^N \ln \pi_{y_n} f_{y_n}(\mathbf{x}_n | \Theta_{y_n}) \quad (3.9)$$

Plugging Eq. 3.8 and 3.9 into Eq. 3.5 formulates  $Q$ -function that is later used for obtaining general equations for the **E** and **M** steps of the EM algorithm [31]:

$$\begin{aligned}
Q\left(\hat{\Theta}^{(i+1)}, \hat{\Theta}^{(i)}\right) &= \sum_{\mathbf{y} \in \Omega_{\mathbf{y}}} \ln \mathcal{L}\left(\hat{\Theta}^{(i+1)} | \mathbf{X}, \mathbf{y}\right) p\left(\mathbf{y} | \mathbf{X}, \hat{\Theta}^{(i)}\right) \\
&= \sum_{\mathbf{y} \in \Omega_{\mathbf{y}}} \sum_{n=1}^N \ln \hat{\pi}_{y_n}^{(i+1)} f_{y_n}\left(\mathbf{x}_n | \hat{\Theta}_{y_n}^{(i+1)}\right) \prod_{n=1}^N p\left(y_n | \mathbf{x}_n, \hat{\Theta}^{(i)}\right) \\
&= \sum_{y_1=1}^K \cdots \sum_{y_N=1}^K \sum_{n=1}^N \ln \hat{\pi}_{y_n}^{(i+1)} \cdots \\
&\cdots f_{y_n}\left(\mathbf{x}_n | \hat{\Theta}_{y_n}^{(i+1)}\right) \prod_{n=1}^N p\left(y_n | \mathbf{x}_n, \hat{\Theta}^{(i)}\right) \\
&= \sum_{y_1=1}^K \cdots \sum_{y_N=1}^K \sum_{n=1}^N \sum_{j=1}^K \cdots \\
&\cdots \delta_{j, y_n} \ln \hat{\pi}_j^{(i+1)} f_j\left(\mathbf{x}_n | \hat{\Theta}_j^{(i+1)}\right) \prod_{n=1}^N p\left(y_n | \mathbf{x}_n, \hat{\Theta}^{(i)}\right) \\
&= \sum_{j=1}^K \sum_{n=1}^N \ln \hat{\pi}_j^{(i+1)} f_j\left(\mathbf{x}_n | \hat{\Theta}_j^{(i+1)}\right) \cdots \\
&\cdots \underbrace{\sum_{y_1=1}^K \cdots \sum_{y_N=1}^K \delta_{j, y_n} \prod_{m=1}^N p\left(y_m | \mathbf{x}_m, \hat{\Theta}^{(i)}\right)}_{p\left(y_n=j | \mathbf{x}_n, \hat{\Theta}^{(i)}\right)} \\
&= \sum_{j=1}^K \sum_{n=1}^N \ln \left(\hat{\pi}_j^{(i+1)} f_j\left(\mathbf{x}_n | \hat{\Theta}_j^{(i+1)}\right) p\left(y_n=j | \mathbf{x}_n, \hat{\Theta}^{(i)}\right)\right) \\
&= \sum_{j=1}^K \sum_{n=1}^N \ln \hat{\pi}_j^{(i+1)} p\left(y_n=j | \mathbf{x}_n, \hat{\Theta}^{(i)}\right) \\
&+ \sum_{j=1}^K \sum_{n=1}^N \ln f_j\left(\mathbf{x}_n | \hat{\Theta}_j^{(i+1)}\right) p\left(y_n=j | \mathbf{x}_n, \hat{\Theta}^{(i)}\right)
\end{aligned} \tag{3.10}$$

The probabilities  $p\left(y_n=j | \mathbf{x}_n, \hat{\Theta}^{(i)}\right)$  and  $p\left(y_n | \mathbf{x}_n, \hat{\Theta}^{(i)}\right)$  may seem a bit confusing. Notion of  $p\left(y_n=k | \mathbf{x}, \hat{\Theta}^{(i)}\right)$  is used to emphasize particular assignment of the value  $k$ .

In the fourth step of Eq. 3.10 marginalization is substituted by  $p\left(y_n=k | \mathbf{x}_n, \Theta^{(i)}\right)$  since the  $\sum_{k=1}^K p\left(y_n=k | \mathbf{x}_n, \Theta^{(i)}\right) = 1$  holds the underscored part is simplified into:

$$\begin{aligned}
&\sum_{y_1=1}^K \cdots \sum_{y_N=1}^K \delta_{j, y_n} \prod_{m=1}^N p\left(y_m | \mathbf{x}_m, \Theta^{(i)}\right) \\
&= \left( \sum_{y_1=1}^K \cdots \sum_{y_{n-1}=1}^K \sum_{y_{n+1}=1}^K \cdots \sum_{y_N=1, m=1, m \neq j}^K \prod p\left(y_m | \mathbf{x}_m, \Theta^{(i)}\right) \right) p\left(y_n=j | \mathbf{x}_n, \Theta^{(i)}\right) \\
&= \prod_{m=1, m \neq n}^N \left( \sum_{y_m=1}^K p\left(y_m | \mathbf{x}_m, \Theta^{(i)}\right) \right) p\left(y_n=j | \mathbf{x}_n, \Theta^{(i)}\right) \\
&= p\left(y_n=j | \mathbf{x}_n, \Theta^{(i)}\right)
\end{aligned} \tag{3.11}$$

From algorithmic perspective the steps of the EM are divided into two major parts: **E** where the  $Q$ -function Eq. 3.10 is calculated based on the current Eq. 3.7 and observed data  $\mathbf{X}$ , and maximization **M** where the new estimate  $\hat{\Theta}^{(i+1)}$  is calculated from maximization of the  $Q$ -function in Eq. 3.10. The pseudo-code in Alg. 1 summarizes entire procedure.

The stop criterion in Alg. 1 is widely discussed problem that has no universally the best option. Usually only criterion is that incomplete log-likelihood  $\ln \mathcal{L}\left(\hat{\Theta}^{(i)} | \mathbf{X}\right)$  shall not decrease. If the EM procedure converged to some maxima the incomplete log-likelihood would not change which seems as good stop criterion. The mostly used stop criteria are following:

1. Bounded number of iterations ( $i > MAX\_ITER$ ).
2. Incomplete log-likelihood reached desired level ( $\ln \mathcal{L}(\Theta^{(i)}|\mathbf{X}) > desired$ ).
3. Incomplete log-likelihood does not change in last few iterations:  

$$\sum_{j=1}^{\Delta} |\ln \mathcal{L}(\Theta^{(i-\Delta+j)}|\mathbf{X}) - \ln \mathcal{L}(\Theta^{(i-\Delta+j-1)}|\mathbf{X})| \leq \tau$$

---

**Algorithm 1:** EM algorithm in generalized form.

---

**input** : Set of observed values  $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$ , list of  $K$  PDFs to be estimated.

**output**: Set of parameters  $\Theta = \{\pi_1, \dots, \pi_K, \Theta_1 \dots \Theta_K\}$

$\Theta^{(0)} \leftarrow$  guess initial parameters;

$i \leftarrow 1$ ;

**while** stop criterion **do**

**for**  $k \leftarrow 1$  **to**  $K$  **do**

// **E** step, calculate conditional expectation

$Q(\hat{\Theta}^{(i+1)}, \hat{\Theta}^{(i)})$ ; // (Eq. 3.10)

// **M** step.

$\hat{\Theta}_k^{(i+1)} \leftarrow \arg \max_{\hat{\Theta}_k^{(i+1)}} Q(\hat{\Theta}^{(i+1)}, \hat{\Theta}^{(i)})$ ; // maximization of the (Eq. 3.10)

$\hat{\pi}_k^{(i+1)} \leftarrow \frac{1}{N} \sum_{n=1}^N p(y_n = k | \mathbf{x}_n, \Theta^{(i)})$ ; // (Eq. 3.12)

The  $Q$ -function expression in last term of the Eq. 3.10 gives more convenient form of how to maximize the parameters  $\Theta_k$  of each PDF. The  $Q$ -function is general framework of solving the MLE for mixture of densities.

The EM increases log-likelihood in each iteration by approaching to local maximum of the log-likelihood function which is performed by finding maximum of the  $Q$ -function for the current parameters  $\hat{\Theta}^{(i)}$ . The maximization of  $Q$ -function will never decrease the likelihood function and so the EM always converges to a certain stationary point [30].

The  $Q$ -function is formulated in way which allows to treat separately  $\pi_k$ -s and parameters of the PDFs  $f_k$ -s and thus the **M** step for the mixture coefficients  $\pi$  can be found independently on functions  $f_k$ . Taking partial derivative of the Eq. 3.10 by arbitrary mixture coefficient  $\hat{\pi}_k^{(i)}$ , setting the resulting derivative as zero with constraint  $\sum_{k=1}^K \pi_k = 1$  gives the equation that determines **M** step for the  $\pi_k$ -s for independently of PDF. The **M** step for the  $\hat{\pi}_k^{(i+1)}$  is formulated as follows [31]:

$$\begin{aligned}
 \frac{\partial}{\partial \hat{\pi}_j^{(i)}} Q(\hat{\Theta}^{(i+1)}, \hat{\Theta}^{(i)}) &= 0 \\
 \frac{\partial}{\partial \hat{\pi}_j^{(i)}} \left( \sum_{k=1}^K \sum_{n=1}^N \ln \hat{\pi}_k^{(i)} p(y_n = j | \mathbf{x}_n, \hat{\Theta}^{(i)}) + \lambda \left( \sum_{k=1}^K \hat{\pi}_k^{(i)} - 1 \right) \right) &= 0 \\
 \sum_{n=1}^N \frac{p(y_n = j | \mathbf{x}_n, \hat{\Theta}^{(i)})}{\hat{\pi}_j^{(i)}} + \lambda &= 0 \\
 \frac{1}{N} \sum_{n=1}^N p(y_n = j | \mathbf{x}_n, \hat{\Theta}^{(i)}) &= \hat{\pi}_j^{(i+1)}
 \end{aligned} \tag{3.12}$$

The procedure of finding stationary point of the  $Q$ -function for the  $\hat{\Theta}_k^{(i)}$  is different. The equations for **M** step must be derived for each PDF separately, nevertheless, later it will be useful for defining general steps that lead to parameter estimation of the  $\hat{\Theta}_k^{(i+1)}$  for an arbitrary function

$f_k(\mathbf{x}|\Theta_k)$ <sup>1</sup>:

$$\begin{aligned} \frac{\partial}{\partial \Theta_j} Q(\Theta^{(i+1)}, \Theta^{(i)}) &= \frac{\partial}{\partial \Theta_j} \sum_{k=1}^K \sum_{n=1}^N p(y_n = k | \mathbf{x}_n, \Theta^{(i)}) \ln f_k(\mathbf{x}_n | \Theta_k^{(i)}) \\ &= \sum_{n=1}^N p(y_n = j | \mathbf{x}_n, \Theta^{(i)}) \frac{\partial}{\partial \Theta_j} \ln f_j(\mathbf{x}_n | \Theta_j^{(i)}) \end{aligned} \quad (3.13)$$

The Eq. 3.12 and 3.13 provide all that is fundamental framework to formulate **E** and **M** steps of the EM algorithm for arbitrary PDF. The Eq. 3.13 must be derived individually for each type of PDF.

### 3.1.1 EM algorithm as classifier

So far the EM was supposed to perform the density estimation of mixture of densities. But the EM is known as generalization of so called K-means algorithm [13] which is used purely for classification. The probabilities of unobserved variable provide the information which the PDF most probably drew a given observation thus the probability can be used for classification. Practically the higher  $p(y_n = j | \mathbf{x}_n, \Theta_j)$  value the better is to choose  $j$ -th class as a label for a datum  $\mathbf{x}_n$ .

The Bayesian Decision Boundary (BDB) gives general and intuitive clue how to perform classification. The BDB lies exactly at the intersection of two PDFs  $f_1(\mathbf{x}_n | \Theta_1)$  and  $f_2(\mathbf{x}_n | \Theta_2)$ . If a datum  $\mathbf{x}_n$  is moved slightly from the BDB the probability of the one PDF increases and the other decreases. If the datum is shifted from the BDB to any side the  $p(y_n = 1 | \mathbf{x}_n, \Theta_1) > p(y_n = 2 | \mathbf{x}_n, \Theta_1)$  or  $p(y_n = 1 | \mathbf{x}_n, \Theta_1) < p(y_n = 2 | \mathbf{x}_n, \Theta_1)$  which means that it is better to choose the one with higher  $p(y_n | \mathbf{x}_n, \Theta)$ . The BDB is illustrated in Fig. 3.1. The entire classification procedure can be formalized as following procedure:

$$\text{Classify}(\mathbf{x}_n, \hat{\Theta}) = \arg \max_{k \in \{1, \dots, K\}} p(y = k | \mathbf{x}_n, \hat{\Theta}) = \arg \max_{k \in \{1, \dots, K\}} \frac{\hat{\pi}_k f_k(\mathbf{x}_n | \hat{\Theta}_k)}{\sum_{i=1}^K \hat{\pi}_i f_i(\mathbf{x}_n | \hat{\Theta}_i)} \quad (3.14)$$

### 3.1.2 EM algorithm for various distribution functions

This subsection deals with solutions of the **M** step of the Alg. 1 for PDFs that are described in Chapter 2. All PDFs except of the Gamma and Dirichlet distributions have an analytic solution for stationary point of  $Q$ -function formulated in Eq. 3.13.

The form of  $Q$ -function allows the division of computation into two parts where mixture coefficients  $\hat{\pi}_k^{(i+1)}$  and  $\hat{\Theta}_k^{(i+1)}$  can be treated separately. The part of the  $Q$ -function where the mixture coefficients  $\pi$  are defined is supposed to be a constant  $C$ . The mixture coefficients are already solved in Eq. 3.12

The log-likelihoods from Chapter 2 are reused here because the log-likelihoods only slightly differ from maximizing problem in  $Q$ -functions.

<sup>1</sup>At this moment, the notion slightly diverges here because mixture coefficients are a part of  $\Theta_k$  so suppose for a moment that  $\pi \notin \Theta_k$

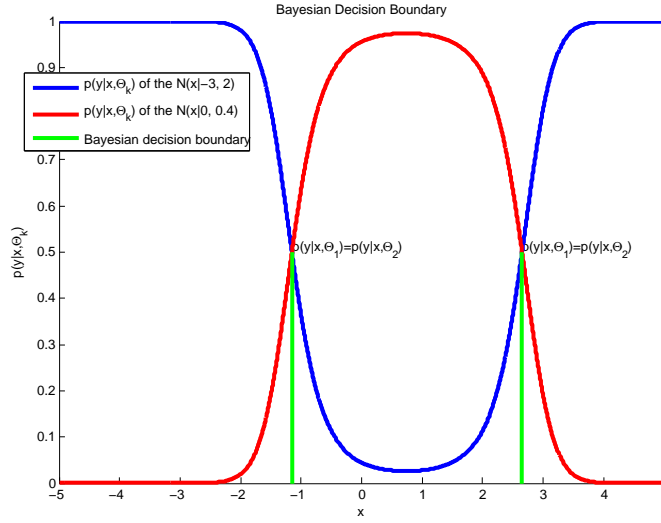


FIGURE 3.1: Intersection of two univariate Normal distributions is called Bayes decision boundary.

### 3.1.2.1 Univariate Exponential distribution

The  $Q$ -function where the  $k$ -th PDF is the Exponential distribution, with one parameter  $\lambda$ , is defined as follows:

$$\begin{aligned}
 Q\left(\hat{\theta}^{(i+1)}, \hat{\theta}^{(i)}\right) &= C + \sum_{k=1}^K \sum_{n=1}^N p\left(y_n = k | x_n, \hat{\theta}^{(i)}\right) \ln\left(\text{Exp}\left(x_n | \hat{\lambda}_k^{(i+1)}\right)\right) \\
 &= C + \sum_{k=1}^K \sum_{n=1}^N p\left(y_n = k | x_n, \hat{\theta}^{(i)}\right) \left(\ln \hat{\lambda}_k^{(i+1)} - \hat{\lambda}_k^{(i+1)} x_n\right)
 \end{aligned} \tag{3.15}$$

By taking partial derivative of Eq. 3.15 by the parameter  $\hat{\lambda}_k^{(i+1)}$  the following equation is obtained:

$$\frac{\partial}{\partial \hat{\lambda}_k^{(i+1)}} Q\left(\hat{\theta}^{(i+1)}, \hat{\theta}^{(i)}\right) = \sum_{n=1}^N p\left(y_n = k | x_n, \hat{\theta}^{(i)}\right) \left(\frac{1}{\hat{\lambda}_k^{(i+1)}} - x_n\right) \tag{3.16}$$

The equation above can be directly solved analytically for the  $\hat{\lambda}_k^{(i+1)}$ , so the solution for the  $\mathbf{M}$  step for the Exponential distribution is defined as follows

$$\arg \max_{\hat{\theta}^{(i+1)}} Q\left(\hat{\theta}^{(i+1)}, \hat{\theta}^{(i)}\right) = \hat{\lambda}_k^{(i+1)} = \frac{N}{\sum_{n=1}^N p\left(y_n = k | x_n, \hat{\theta}^{(i)}\right) x_n} \tag{3.17}$$

### 3.1.2.2 Univariate Gamma distribution

The  $Q$ -function where the  $m$ -th PDF is the Gamma distribution with two parameters  $k$  and  $\theta$  is defined as follows:

$$\begin{aligned}
Q\left(\hat{\Theta}^{(i+1)}, \hat{\Theta}^{(i)}\right) &= C + \sum_{m=1}^K \sum_{n=1}^N p\left(y_n = m | x_n, \hat{\Theta}^{(i)}\right) \ln\left(\text{Gam}\left(x_n | \hat{k}_m^{(i+1)}, \hat{\theta}_m^{(i+1)}\right)\right) \\
&= C + \sum_{m=1}^K \sum_{n=1}^N p\left(y_n = m | x_n, \hat{\Theta}^{(i)}\right) \dots \\
&\dots \left( \ln \frac{1}{\left(\hat{\theta}_m^{(i+1)}\right)^{\hat{k}_m^{(i+1)}} \Gamma\left(\hat{k}_m^{(i+1)}\right)} x_n^{\hat{k}_m^{(i+1)}-1} \exp\left(-\frac{x_n}{\hat{\theta}_m^{(i+1)}}\right) \right) \\
&= C + \sum_{m=1}^K \sum_{n=1}^N p\left(y_n = m | x_n, \hat{\Theta}^{(i)}\right) \left( \left(\hat{k}_m^{(i+1)} - 1\right) \ln x_n \dots \right. \\
&\dots \left. - \frac{1}{\hat{\theta}_m^{(i+1)}} x_n - \ln \Gamma\left(\hat{k}_m^{(i+1)}\right) - \hat{k}_m^{(i+1)} \ln \hat{\theta}_m^{(i+1)} \right)
\end{aligned} \tag{3.18}$$

By taking partial derivative of Eq. 3.18 by the parameters  $\hat{\theta}_m^{(i+1)}$  and  $\hat{k}_m^{(i+1)}$  following equations are obtained:

$$\begin{aligned}
\frac{\partial}{\partial \hat{\theta}_m^{(i+1)}} Q\left(\hat{\Theta}^{(i+1)}, \hat{\Theta}^{(i)}\right) &= \sum_{n=1}^N p\left(y_n = m | x_n, \hat{\Theta}^{(i)}\right) \left( -\frac{N \hat{k}_m^{(i+1)}}{\hat{\theta}_m^{(i+1)}} + \frac{x_n}{\hat{\theta}_m^{(i+1)}} \right) \\
\frac{\partial}{\partial \hat{k}_m^{(i+1)}} Q\left(\hat{\Theta}^{(i+1)}, \hat{\Theta}^{(i)}\right) &= \sum_{n=1}^N p\left(y_n = m | x_n, \hat{\Theta}^{(i)}\right) \left( -\ln \hat{\theta}_m^{(i+1)} - \frac{\partial \ln \Gamma\left(\hat{k}_m^{(i+1)}\right)}{\partial \hat{k}_m^{(i+1)}} + \ln x_n \right)
\end{aligned} \tag{3.19}$$

For the  $\hat{\theta}_m^{(i+1)}$  the solution is straightforward which is defined as follows:

$$\arg \max_{\hat{\Theta}^{(i+1)}} Q\left(\hat{\Theta}^{(i+1)}, \hat{\Theta}^{(i)}\right) = \hat{\theta}_m^{(i+1)} = \frac{\hat{k}_m^{(i)} \sum_{n=1}^N p\left(y_n = m | x_n, \hat{\Theta}_k^{(i)}\right)}{\sum_{n=1}^N x_n p\left(y_n = m | x_n, \hat{\Theta}_k^{(i)}\right)} \tag{3.20}$$

The problem of the  $k_m$  is more complicated due to  $\Gamma(x)$  function. The first derivative (see Chapter 2, where analogous problem for the MLE is solved) of  $\Gamma(x)$  is called Digamma function  $\psi(x)$ . As very useful is shown the approximation of the Digamma function, because the  $\psi(x)$  is not available in some packages. In [32] they approximated the value  $\psi(x)$  as follows:

$$\psi(x) \approx \tilde{\psi}(x) = \ln\left(x - \frac{1}{2}\right) + \frac{1}{24\left(x - \frac{1}{2}\right)^2} \tag{3.21}$$

The differences between the real value and approximation is insignificant as long as the values are relatively small. The approximation is mentioned for completeness, in MATLAB the  $\psi(x)$  is implemented as function named `psi`.

In fact, the EM algorithm is actually a gradient-based technique, i.e., in each iteration the parameter change has positive projection on the gradient of the likelihood function with respect

to the  $\Theta$  parameters so gradient is used to find  $\hat{k}_m^{(i+1)}$ , which is formulated as follows [32]:

$$\begin{aligned} \arg \max_{\hat{\Theta}^{(i+1)}} Q \left( \hat{\Theta}^{(i+1)}, \hat{\Theta}^{(i)} \right) &= \hat{k}_m^{(i+1)} \\ &= \hat{k}_m^{(i)} + \frac{c}{N} \sum_{n=1}^N p \left( y_n = m | x_n, \hat{\Theta}^{(i)} \right) \dots \\ &\dots \left( -\ln \hat{\theta}_m^{(i)} + \ln x_n - \psi(x_n) \right) \end{aligned} \quad (3.22)$$

Where the  $c > 0$  is constant that regulates the size of steps.

### 3.1.2.3 Multivariate Normal distribution

The  $Q$ -function where the  $k$ -th PDF is the multivariate Normal distribution, with two parameters  $\mu$  and  $\Sigma$  is defined as follows:

$$\begin{aligned} Q \left( \hat{\Theta}^{(i+1)}, \hat{\Theta}^{(i)} \right) &= C + \sum_{k=1}^K \sum_{n=1}^N p \left( y_n = k | \mathbf{x}_n, \Theta^{(i)} \right) \ln \left( \mathcal{N} \left( \mathbf{x}_n | \hat{\mu}^{(i+1)}, \hat{\Sigma}^{(i+1)} \right) \right) \\ &= C + \sum_{k=1}^K \sum_{n=1}^N p \left( y_n = k | \mathbf{x}_n, \hat{\Theta}^{(i)} \right) \left( -\frac{D}{2} \ln(2\pi) \dots \right. \\ &\dots \left. - \frac{\ln \det \hat{\Sigma}_k^{(i+1)}}{2} - \frac{1}{2} \sum_{n=1}^N \left( \mathbf{x}_n - \hat{\mu}_k^{(i+1)} \right) \left( \hat{\Sigma}_k^{(i+1)} \right)^{-1} \left( \mathbf{x}_n - \hat{\mu}_k^{(i+1)} \right)^T \right) \end{aligned} \quad (3.23)$$

By taking partial derivative of Eq. 3.23 by the parameters  $\hat{\mu}_k^{(i+1)}$ ,  $\hat{\Sigma}_k^{(i+1)}$  the following equations are obtained: [24]:

$$\begin{aligned} \frac{\partial}{\partial \hat{\mu}_k^{(i+1)}} Q \left( \hat{\Theta}^{(i+1)}, \hat{\Theta}^{(i)} \right) &= \left( \hat{\Sigma}_k^{(i)} \right)^{-1} \sum_{n=1}^N p \left( y_n = k | \mathbf{x}_n, \hat{\Theta}_k^{(i)} \right) \left( \mathbf{x}_n - \hat{\mu}_k^{(i+1)} \right) \\ \frac{\partial}{\partial \hat{\Sigma}_k^{(i+1)}} Q \left( \hat{\Theta}^{(i+1)}, \hat{\Theta}^{(i)} \right) &= \sum_{n=1}^N p \left( y_n = k | \mathbf{x}_n, \hat{\Theta}^{(i)} \right) \left( \left( \hat{\Sigma}_k^{(i+1)} \right)^{-1} \dots \right. \\ &\dots \left. + \left( \hat{\Sigma}_k^{(i+1)} \right)^{-1} \mathbf{x}_n \mathbf{x}_n^T \left( \hat{\Sigma}_k^{(i+1)} \right)^{-1} \dots \right. \\ &\dots \left. - \frac{1}{2} \left( \left( \hat{\Sigma}_k^{(i+1)} \right)^{-1} \mathbf{x}_n \mathbf{x}_n^T \left( \hat{\Sigma}_k^{(i+1)} \right)^{-1} - \left( \hat{\Sigma}_k^{(i+1)} \right)^{-1} \right) \cdot \mathbf{I} \right) \end{aligned} \quad (3.24)$$

The equations Eq. 3.24 can be solved analytically for the parameters  $\hat{\mu}_k^{(i+1)}$  and  $\hat{\Sigma}_k^{(i+1)}$  thus the solutions for  $\mathbf{M}$  step with multivariate Normal distribution are formulated as follows [13]:

$$\begin{aligned} \arg \max_{\hat{\Theta}^{(i+1)}} Q \left( \hat{\Theta}^{(i+1)}, \hat{\Theta}^{(i)} \right) &= \hat{\mu}_k^{(i+1)} = \frac{\sum_{n=1}^N \mathbf{x}_n p \left( y_n = k | \mathbf{x}_n, \hat{\Theta}_k^{(i)} \right)}{\sum_{n=1}^N p \left( y_n = k | \mathbf{x}_n, \hat{\Theta}_k^{(i)} \right)} \\ \arg \max_{\hat{\Theta}^{(i+1)}} Q \left( \hat{\Theta}^{(i+1)}, \hat{\Theta}^{(i)} \right) &= \hat{\Sigma}_k^{(i+1)} = \frac{\sum_{n=1}^N p \left( y_n = k | \mathbf{x}_n, \hat{\Theta}_k^{(i)} \right) \left( \mathbf{x}_n - \hat{\mu}_k^{(i)} \right) \left( \mathbf{x}_n - \hat{\mu}_k^{(i)} \right)^T}{\sum_{n=1}^N p \left( y_n = k | \mathbf{x}_n, \hat{\Theta}_k^{(i)} \right)} \end{aligned} \quad (3.25)$$

### 3.1.2.4 Multivariate Log-normal distribution

The  $Q$ -function where the  $k$ -th PDF is the multivariate Log-normal distribution, with two parameters  $\mu$  and  $\Sigma$  is defined as follows:

$$\begin{aligned}
Q\left(\hat{\Theta}^{(i+1)}, \hat{\Theta}^{(i)}\right) &= C + \sum_{k=1}^K \sum_{n=1}^N p\left(y_n = k | \mathbf{x}_n, \hat{\Theta}^{(i)}\right) \ln\left(\mathcal{LN}\left(\mathbf{x}_n | \hat{\mu}_k^{(i+1)}, \hat{\Sigma}_k^{(i+1)}\right)\right) \\
&= C + \sum_{k=1}^K \sum_{n=1}^N p\left(y_n = k | \mathbf{x}_n, \hat{\Theta}^{(i)}\right) \left(-\frac{D}{2} \ln(2\pi) \dots \right. \\
&\dots - \sum_{d=1}^D \ln x_{d,n} - \frac{\ln \det \hat{\Sigma}_k^{(i+1)}}{2} \\
&\dots \left. - \frac{1}{2} \sum_{n=1}^N \left(\ln \mathbf{x}_n - \hat{\mu}_k^{(i+1)}\right) \left(\hat{\Sigma}_k^{(i+1)}\right)^{-1} \left(\ln \mathbf{x}_n - \hat{\mu}_k^{(i+1)}\right)^T\right)
\end{aligned} \tag{3.26}$$

The procedure of obtaining stationary point of  $Q$  function in Eq. 3.26 can be reused from Eq. 3.23 (the same procedure as in Chapter 2) with substitution  $\mathbf{z}_n = \ln \mathbf{x}_n$ . The  $Q$ -function with the substitution is identical with  $Q$ -function for the Normal distribution except of  $\sum_{d=1}^D \ln x_{d,n}$ . The term  $\sum_{d=1}^D \ln x_{d,n}$ , that makes difference is ruled out by derivatives by parameters  $\hat{\mu}_k^{(i+1)}$  and  $\hat{\Sigma}_k^{(i+1)}$  thus the analytic solution from Eq. 3.25 can be reused. The  $\mathbf{M}$  step after back-substitution is defined as follows:

$$\begin{aligned}
\arg \max_{\hat{\Theta}^{(i+1)}} Q\left(\hat{\Theta}^{(i+1)}, \hat{\Theta}^{(i)}\right) &= \hat{\mu}_k^{(i+1)} = \frac{\sum_{n=1}^N \ln(\mathbf{x}_n) p(y_n = k | \mathbf{x}_n, \hat{\Theta}_k^{(i)})}{\sum_{n=1}^N p(y_n = k | \mathbf{x}_n, \hat{\Theta}_k^{(i)})} \\
\arg \max_{\hat{\Theta}^{(i+1)}} Q\left(\hat{\Theta}^{(i+1)}, \hat{\Theta}^{(i)}\right) &= \hat{\Sigma}_k^{(i+1)} = \frac{\sum_{n=1}^N p(y_n = k | \mathbf{x}_n, \hat{\Theta}_k^{(i)}) \left(\ln(\mathbf{x}_n) - \hat{\mu}_k^{(i+1)}\right) \left(\ln(\mathbf{x}_n) - \hat{\mu}_k^{(i+1)}\right)^T}{\sum_{n=1}^N p(y_n = k | \mathbf{x}_n, \hat{\Theta}_k^{(i)})}
\end{aligned} \tag{3.27}$$

### 3.1.2.5 Multivariate von Mises Fisher distribution

The  $Q$ -function where the  $k$ -th PDF is the von Mises-Fisher distribution, with two parameters  $\mu$  and  $\kappa$  is defined as follows [22]:

$$\begin{aligned}
Q\left(\hat{\Theta}^{(i+1)}, \hat{\Theta}^{(i)}\right) &= C + \sum_{k=1}^K \sum_{n=1}^N p\left(y_n = k | \mathbf{x}_n, \hat{\Theta}^{(i)}\right) \ln\left(\mathcal{MF}\left(\mathbf{x}_n | \hat{\mu}_k^{(i+1)}, \hat{\kappa}_k^{(i+1)}\right)\right) \\
&= C + \sum_{k=1}^K \sum_{n=1}^N p\left(y_n = k | \mathbf{x}_n, \hat{\Theta}^{(i)}\right) \left(\ln c_D\left(\hat{\kappa}_k^{(i+1)}\right) + \dots \right. \\
&\dots \left. \hat{\kappa}_k^{(i+1)} \left(\hat{\mu}_k^{(i+1)}\right)^T \mathbf{x}_n + \lambda_k \left(1 - \left(\hat{\mu}_k^{(i+1)}\right)^T \hat{\mu}_k^{(i+1)}\right)\right)
\end{aligned} \tag{3.28}$$

Where  $\lambda_k$  is Lagrange multiplier which guarantees that constraint  $\left(\hat{\mu}_k^{(i+1)}\right)^T \hat{\mu}_k^{(i+1)} = 1$  will be kept. Taking partial derivatives of Eq. 3.28 by the parameter  $\hat{\mu}_k^{(i+1)}$  and Lagrange multiplier  $\lambda_k$ ,



setting resulting the equations as zero, the following relations are obtained [22]:

$$\begin{aligned}\hat{\mu}_k^{(i+1)} &= \frac{\hat{\kappa}_k^{(i)}}{2\lambda_k} \sum_{n=1}^N p(y_n = k | \mathbf{x}_n, \hat{\Theta}^{(i)}) \mathbf{x}_n \\ (\hat{\mu}_k^{(i+1)})^T \hat{\mu}_k^{(i+1)} &= 1\end{aligned}\quad (3.29)$$

By solving system of equations Eq. 3.29 the **M** step for  $\mu_k$  [22]:

$$\arg \max_{\hat{\Theta}^{(i+1)}} Q(\hat{\Theta}^{(i+1)}, \hat{\Theta}^{(i)}) = \hat{\mu}_k^{(i+1)} = \frac{\lambda_k}{\left\| \sum_{n=1}^N p(y_n = k | \mathbf{x}_n, \hat{\Theta}^{(i)}) \mathbf{x}_n \right\|} \quad (3.30)$$

The parameter  $\kappa_k$  requires some advanced mathematical results so only solution for **M** step is shown here, which is defined as follows [22]:

$$\arg \max_{\hat{\Theta}^{(i+1)}} Q(\hat{\Theta}^{(i+1)}, \hat{\Theta}^{(i)}) = \hat{\kappa}_k^{(i+1)} = \frac{D\bar{r} - \bar{r}^3}{1 - \bar{r}^2} \quad (3.31)$$

Where  $\bar{r}$  is defined as  $\bar{r} = \frac{\left\| \sum_{n=1}^N p(y_n = k | \mathbf{x}_n, \hat{\Theta}^{(i)}) \mathbf{x}_n \right\|}{N\pi_k^{(i)}}$ .

### 3.1.2.6 Dirichlet distribution

The  $Q$ -function where  $k$ -th PDF is the Dirichlet distribution  $\mathcal{D}(\mathbf{x}_n | \mathbf{a})$  with  $D$  parameters  $\mathbf{a} = [a_1, \dots, a_D]^T$  is defined as follows:

$$\begin{aligned}Q(\hat{\Theta}^{(i+1)}, \hat{\Theta}^{(i)}) &= C + \sum_{k=1}^K \sum_{n=1}^N p(y_n = k | x_n, \hat{\Theta}^{(i)}) \ln(\mathcal{D}(\mathbf{x} | \hat{\mathbf{a}}_k^{(i+1)})) \\ &= C + \sum_{k=1}^K \sum_{n=1}^N p(y_n = k | x_n, \hat{\Theta}^{(i)}) \dots \\ &\dots \left( \ln \frac{\Gamma(\sum_{d=1}^D \hat{a}_{d,k}^{(i+1)})}{\prod_{d=1}^D \Gamma(\hat{a}_{d,k}^{(i+1)})} \prod_{d=1}^D x_{d,n}^{\hat{a}_{d,k}^{(i+1)} - 1} \right) \\ &= C + \sum_{k=1}^K \sum_{n=1}^N p(y_n = k | x_n, \hat{\Theta}^{(i)}) \left( \ln \Gamma \left( \sum_{d=1}^D \hat{a}_{d,k}^{(i+1)} \right) \dots \right. \\ &\dots \left. - \sum_{d=1}^D \ln \Gamma(\hat{a}_{d,k}^{(i+1)}) + \sum_{d=1}^D (\hat{a}_{d,k}^{(i+1)} - 1) \ln x_{d,n} \right)\end{aligned}\quad (3.32)$$

Taking partial derivative of Eq. 3.32 by any parameter  $a_{d,k}$ , the following equation is obtained:

$$\begin{aligned}\frac{\partial}{\partial \hat{a}_{d,k}^{(i)}} Q(\hat{\Theta}^{(i+1)}, \hat{\Theta}^{(i)}) &= \sum_{n=1}^N p(y_n = k | x_n, \Theta^{(i)}) \left( \psi \left( \sum_{d=1}^D \hat{a}_{d,k}^{(i+1)} \right) \right. \\ &\quad \left. + \psi \left( \hat{a}_{d,k}^{(i+1)} \right) + \ln x_{d,n} \right)\end{aligned}\quad (3.33)$$

The parameters  $\hat{a}_{d,k}^{(i+1)}$  occur in  $\Gamma(x)$  function thus they have to be calculated by Digamma  $\psi(x)$  or approximation  $\tilde{\psi}(x)$  (Eq. 3.21):

$$\begin{aligned} \arg \max_{\hat{\Theta}^{(i)}} Q\left(\hat{\Theta}^{(i+1)}, \hat{\Theta}^{(i)}\right) &= \hat{a}_{d,k}^{(i+1)} \\ &= \hat{a}_{d,k}^{(i)} \\ &+ \sum_{n=1}^N \frac{p(y_n=k|\mathbf{x}_n, \Theta_k^{(i)})}{N} \left( \psi\left(\sum_{j=1}^D \hat{a}_{j,k}^{(i)}\right) - \psi\left(\hat{a}_{d,k}^{(i)}\right) + \ln x_{d,n} \right) \end{aligned} \quad (3.34)$$

The gradient equation formulated in Eq. 3.34 suffers from the usual problems of the gradient ascent - the procedure does not consider inadmissible values and the step size is constant. The step size does not reflect the current steepness of the  $Q$ -function thus convergence of the Eq. 3.34 requires more iterations.

### 3.1.2.7 Multivariate Wishart distribution

The  $Q$ -function where the  $k$ -th PDF is the Wishart distribution, with one parameter  $\Sigma$  is defined as follows:

$$\begin{aligned} Q\left(\hat{\Theta}^{(i+1)}, \hat{\Theta}^{(i)}\right) &= C + \sum_{k=1}^K \sum_{n=1}^N p\left(y_n = k | \mathbf{S}_n, \hat{\Theta}^{(i)}\right) \ln(\mathcal{W}(\mathbf{S}_n | \Sigma_k)) \\ &= C + \sum_{k=1}^K \sum_{n=1}^N p\left(y_n = k | \mathbf{S}_n, \hat{\Theta}^{(i)}\right) (-3 \ln \pi \dots \\ &\dots \ln \det \hat{\Sigma}_k - \text{tr}(\mathbf{S} \Sigma^{-1})) \end{aligned} \quad (3.35)$$

The  $\Sigma_k$  requires some advanced techniques so only solution for  $\mathbf{M}$  step will be shown which is defined as follows [6]:

$$\arg \max_{\hat{\Theta}^{(i)}} Q\left(\hat{\Theta}^{(i+1)}, \hat{\Theta}^{(i)}\right) = \Sigma_k^{(i+1)} = \frac{\sum_{n=1}^N p\left(y_n = k | \mathbf{S}_n, \hat{\Theta}^{(i)}\right) \mathbf{S}_n}{\sum_{n=1}^N p\left(y_n = k | \mathbf{S}_n, \hat{\Theta}^{(i)}\right)} \quad (3.36)$$

## 3.2 Neural Modeling Fields

The NMF associates lower-level signals with higher-level concept-models where each concept (class) is represented as one parametric model. The NMF tries to generalize low level signals by adapting model parameters. The parametric model creation is achieved by using measures of similarity between the concept models and the input signals which is performed by an adaptive fuzzy similarity ( $AZ - LL$ ). Fuzzy approach extends the crisp (Aristotelian) logic with the concept of fuzziness where membership of signals is not restricted on two values but where the signal is either present or not. Input signals  $\mathbf{X}$  (data, observations) are associated (recognized, grouped into) with the concepts according to the representation models and similarity measures. In the process of association-recognition, the models are adapted for better representation of the input signals. The initial uncertainty of the models is high and so is the fuzziness of the similarity measure; in the process of learning models, the fuzziness becomes more accurate and

as the similarity getting higher the fuzziness is more crispy between the input signals and the associated parametric model [33].

From the computational perspective NMF are set of equations that find MLE of a mixture of densities for parametric model. Example of NMF result is shown in Fig. 3.2.

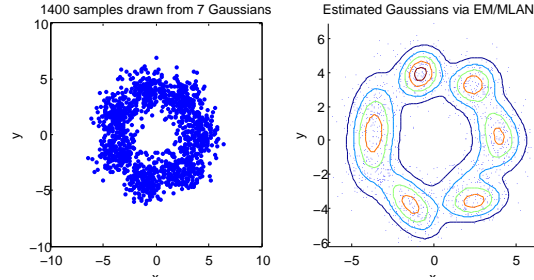


FIGURE 3.2: Example of the MLANS/EM algorithm density estimation on 7 Gaussians with means placed in the circle.

### 3.2.1 Similarity measures

In [6] three kinds of similarity measures between trained NMF and the input signals data are shown where each of the functions differs slightly from the others depending on, how strictly the membership of a datum is judged. Set partition is the crucial concept in this context. The basis of the problem is how perform clustering of a set of data into  $K$  classes. This problem is also known as a set partition. The SP plays a critical role in this context. The SP divides input space into  $K$  subsets such that a rank of the partition is maximal<sup>2</sup>. It can be perceived as a division of space into  $K$  regions (subset of the current set partition) where each region is assigned to one agent that forms a model on that region regardless other data that belong to the other agents. The number of set partitions of  $N$  data into  $K$  subsets can be computed by Stirling numbers of the second kind  $S_2(N, K)$  as it is formulated in Def. 3.2.

**Definition 3.2.** The Stirling number of the second kind,  $S_2(N, K)$ , is the number of partitions of an  $N$ -element set into  $K$  non-empty subsets. The  $S_2(N, K)$  is defined recursively as:

$$S_2(N, K) = S_2(N - 1, K - 1) + K S_2(N - 1, K) \quad (3.37)$$

Or alternatively:

$$S_2(N, K) = \frac{1}{K!} \sum_{k=1}^K (-1)^{K-k} \binom{K}{k} k^N \quad (3.38)$$

[34]

**Theorem 3.3.** The Stirling number of the second kind is bounded by exponential function for the parameter  $N$  [35].

From Def. 3.2 and Theorem 3.2 implies that the set partition is the problem that is solvable for worst cases in exponential time thus finding optimal set partition difficult task from  $\mathcal{NP}$ -complete class of problems. The exponential complexity is also referred as *combinatorial complexity* [6].

<sup>2</sup>The one of the sub-problems solved by the SP

Another problem how to guess how many concept classes  $K$  shall be used if the  $K$  is not given arises. There is no confident principle how to choose  $K$ -s but there are some useful methods how to guess the  $K$  based on data, the most popular is the *Dirichlet process* gives a hint how to guess the  $K$  [15].

In the next sections following notion is used: Given data (observations)  $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$ , a datum  $\mathbf{x}_n$  is the member of a subset  $k$  with parameters  $\Theta_k$  with  $f(\mathbf{x}_n|\Theta_k) \in \langle 0, 1 \rangle$ <sup>3</sup> membership where  $\sum_{k=1}^K f(\mathbf{x}_n|\Theta_k) = 1$  must hold (otherwise a datum may be the member of a concept class more than once). The  $f(\mathbf{x}|\Theta_k)$  is equivalent with  $p(y_k|\mathbf{x}, \Theta_k)$  used in the EM algorithm. The  $f(\mathbf{x}|k)$  stands for non-adaptive membership that is independent of adaptive parameters  $\Theta_k$ . Furthermore suppose any PDF  $f_k(\mathbf{x}_n|\Theta_k)$ . The  $f_k(\mathbf{x}_n|\Theta_k)$  measures how a datum  $\mathbf{x}_n$  is similar to the model determined by the function  $f_k$  with parameters  $\Theta_k$ .

A crisp set partition example is shown in Fig. 3.3.

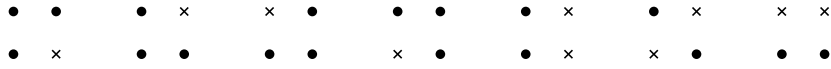


FIGURE 3.3: Example of set partition of four elements ( $N = 4$ ) into two classes ( $K = 2$ ).  $S_2(4, 2) = 7$ . The data membership is distinguished by crosses and circles.

### 3.2.1.1 Aristotelian similarity

*Aristotelian similarity* ( $A - LL$ ) is based on the principle of *excluded third* where each datum either belongs to a subset, or not (no third option) [6]. The crisp membership of presence/absence of datum to a certain class is determined by function  $f_A(\mathbf{x}_n|\Theta_k)$  whose definition is different from above stated.

$$f_A(\mathbf{x}_n|\Xi) = \begin{cases} 1 & \text{if } \mathbf{x}_n \text{ is a member of } k\text{-th class} \\ 0 & \text{else} \end{cases} \quad (3.39)$$

Where the  $\Xi$  is the current set partition. The  $A - LL$  is defined as follows [6]:

$$A - LL = \underbrace{\max_{\Xi} \sum_{k=1}^K}_{\text{combinatorial}} \max_{\Theta_k} \sum_{n=1}^N f_A(\mathbf{x}_n|\Xi) \ln f_k(\mathbf{x}_n|\Theta_k) \quad (3.40)$$

Because membership function is crisp, the Eq. 3.40 rules out all points whose membership function evaluates a datum  $\mathbf{x}$  as non-member of a subset thus the  $A - LL$  is calculated only from points that are members of the subset. The set partition of the data has two steps, at the first set is partitioned, and at the second the function parameters are optimized for the current partition. Unfortunately, the first step is exactly combinatorial problem of the set partition whose problem class is  $\mathcal{NP}$ -complete.

<sup>3</sup>For each subset, the membership function may be different

### 3.2.1.2 Fuzzy similarity

Another similarity measure extends  $A - LL$  with the possibility to have membership function in a range from 0 to 1 independently on the parameters  $\Theta$ . Assigning membership as the real value from 0 to 1 is the well-known principle studied in Fuzzy sets. The Fuzzy sets are an extension of the crisp sets where membership determines *how much a datum is involved as the member of a particular class*. This similarity is referred as Fuzzy ( $Z - LL$ ) in honour to Zadeh's authorship of fuzzy logic [6]. The  $Z - LL$  is calculated as follows:

$$Z - LL = \max_{\Xi} \sum_{k=1}^K \max_{\Theta_k} \sum_{n=1}^N \underbrace{f(\mathbf{x}_n|k)}_{\text{const.}} \ln f_k(\mathbf{x}_n|\Theta_k^{(i)}) \quad (3.41)$$

Due to fuzzy character, the maximization of the set of parameters is possible to be obtained by taking derivative of  $\Theta_k$ , setting equal the equation as zero and solving for the  $\Theta_k$ . The membership function is fixed value so  $f(\mathbf{x}|k)$  is treated as a constant and changes in parameters  $\Theta_k$  are not reflected in the  $f(\mathbf{x}|k)$ . Complexity of finding solution with  $Z - LL$  is  $\mathcal{O}(NK)$  because each function is adapted for each datum. The problem of  $Z - LL$  is a non-adaptive membership function  $f(\mathbf{x}|\Theta_k)$  which does not reflect changes in parameters  $\Theta$ .

### 3.2.1.3 Adaptive Fuzzy similarity

The last one is made combine advantages of Aristotelian and Fuzzy similarity, the adaptive segmentation from the  $A - LL$  and low computational complexity from the  $Z - LL$ . The  $AZ - LL$  which is the abbreviation of Adaptive Fuzzy similarity performs maximization parameters over all functions w.r.t. to log-likelihood. The  $AZ - LL$  reflect parameter changes in fuzzy membership functions  $f(\mathbf{x}|\Theta_k)$ . The measure is defined as follows:

$$AZ - LL = \max_{\Xi} \sum_{k=1}^K \max_{\Theta_k} \sum_{n=1}^N \underbrace{f(\mathbf{x}_n|\Theta_k^{(i)})}_{\text{adaptive}} \ln f_k(\mathbf{x}_n|\Theta_k^{(i)}) \quad (3.42)$$

The difference in between  $AZ - LL$  and  $Z - LL$  is the way how  $f(\mathbf{x}|\Theta_k^{(i)})$  is defined. While in  $Z - LL$  the membership  $f(\mathbf{x}|k)$  is supposed to be given a priori and is non-adaptive, for the  $AZ - LL$  the value of  $f(\mathbf{x}|\Theta_k^{(i)})$  changes adaptively as parameters  $\Theta_k^{(i)}$  are adapted to maximize the log-likelihood. It is important to note that the  $AZ - LL$  is equivalent of the  $Q$ -function defined in the Eq. 3.4.

## 3.2.2 Learning parametric models with NMF

Before the the NMF are defined some notion shall be refined. In the literature that deals with the NMF [6, 29] is quite confusing notion. The original one is replaced by the probabilistic which is introduced for the EM above (see Tab. 3.1 for comparison). On the other hand, the basis of this thesis are NMF so the symbol  $f(\mathbf{x}_n|\Theta_k)$  is used here instead of  $p(y_n = k|\mathbf{x}_n, \Theta_k)$

to emphasize the relation with original symbol for the adaptive fuzzy similarity  $f(n|k)$  which is defined as follows:

$$f(\mathbf{x}_n|\Theta_k) = \frac{\pi_k f_k(\mathbf{x}_n|\Theta_k)}{\sum_{i=1}^N \pi_i f_i(\mathbf{x}_n|\Theta_i)} \quad (3.43)$$

The symbol  $f(\mathbf{x}_n|\Theta_k)$  is called *adaptive fuzzy similarity* and measures membership of the datum  $\mathbf{x}$  in class  $k$ . The normalization guarantees the  $\sum_{k=1}^K f(\mathbf{x}|\Theta_k) = 1$  holds.

**Definition 3.4.** Suppose task of MLE of parameters for a mixture of  $K$  functions  $f_1, \dots, f_K$  with a set of initial parameters  $\hat{\Theta}^{(0)} = \left\{ \hat{\pi}_k^{(0)}, \hat{\Theta}_k^{(0)} \right\}_{k=1}^K$  from Eq. 3.1. The NMF learning equations are defined as follows:

$$\begin{aligned} \frac{\partial AZ-LL}{\partial \hat{\Theta}^{(i+1)}} &= \alpha \sum_{n=1}^N f(\mathbf{x}_n|\hat{\Theta}_k^{(i)}) \frac{\partial \ln f_k(\mathbf{x}_n|\hat{\Theta}_k^{(i+1)})}{\partial \hat{\Theta}^{(i+1)}} \\ \hat{\pi}_k^{(i+1)} &= \frac{\sum_{k=1}^K f(\mathbf{x}_n|\hat{\Theta}_k)}{N} \end{aligned} \quad (3.44)$$

Where the  $\alpha$  is coefficient that regulates learning step length.

### 3.2.3 NMF and EM algorithm equivalence

In the [6] the NMF are referred as a framework for the fuzzy membership function adaptation which turned out as the same as the mixture density estimation (where the parametric model creation is performed by the EM algorithm). In the literature that deal with NMF [6] are some unambiguous references of relationship of the NMF to EM algorithm but the both have shown as equivalent. The proof below proves that the NMF and EM algorithm perform the same computation. The reason why the NMF are referred as EM but not in vice versa is because the first reference of EM (1977 in [10]) is many years before any reference of the NMF (1991 in [36]) occurred.

**Theorem 3.5.** *The EM algorithm and NMF are identical.*

*Proof.* The EM at first computes conditional expectation (**E** phase) of unobserved value, which is formulated in the Eq. 3.7. The same computation is performed for the NMF by the Eq. 3.43. The  $f(\mathbf{x}|\Theta_k)$  is membership function that measures how much the datum belongs to a  $k$ -th class thus  $f(\mathbf{x}|\Theta_k)$  and  $p(y_k|\mathbf{x}, \Theta_k)$  can be supposed as equivalent because both stand for the same. By substitution of  $p(y_k|\mathbf{x}, \Theta_k)$  by  $f(\mathbf{x}|\Theta_k)$  into Eq. 3.13 can be shown that  $\frac{\partial AZ-LL}{\partial \Theta^{(i)}}$  does the same as  $Q$ -function and vice versa:

$$\begin{aligned}
\frac{\partial}{\partial \hat{\Theta}_k^{(i+1)}} Q\left(\hat{\Theta}^{(i+1)}, \hat{\Theta}^{(i)}\right) &= \frac{\partial}{\partial \hat{\Theta}_k^{(i+1)}} \sum_{k=1}^K \sum_{n=1}^N \underbrace{p\left(y_n = j | \mathbf{x}_n, \hat{\Theta}^{(i)}\right)}_{\text{EM}} \underbrace{\ln f_k\left(\mathbf{x}_n | \hat{\Theta}_k^{(i+1)}\right)}_{\text{EM} \equiv \text{NMF}} \\
&= \sum_{n=1}^N \underbrace{p\left(y_n = j | \mathbf{x}_n, \hat{\Theta}^{(i)}\right)}_{\text{EM}} \underbrace{\frac{\partial}{\partial \hat{\Theta}_k^{(i+1)}} \ln f_k\left(\mathbf{x}_n | \hat{\Theta}_k^{(i+1)}\right)}_{\text{EM} \equiv \text{NMF}} \\
&= \sum_{n=1}^N \underbrace{f\left(\mathbf{x}_n | \hat{\Theta}_k^{(i)}\right)}_{\text{NMF}} \underbrace{\frac{\partial}{\partial \hat{\Theta}_k^{(i+1)}} \ln f_k\left(\mathbf{x}_n | \hat{\Theta}_k^{(i+1)}\right)}_{\text{EM} \equiv \text{NMF}} \\
&\equiv \underbrace{\frac{1}{\alpha}}_{\text{const.}} \frac{\partial AZ-LL}{\partial \Theta^{(i+1)}}
\end{aligned} \tag{3.45}$$

The learning constant  $\alpha$  can be ignored because for  $\alpha = 1$  the parameter vanishes.

The mixture coefficients defined in Eq. 3.12 are equivalent with the prior in 3.44:

$$\begin{aligned}
\hat{\pi}_j^{(i+1)} &= \frac{1}{N} \sum_{n=1}^N \underbrace{p\left(y_n = j | \mathbf{x}_n, \hat{\Theta}^{(i)}\right)}_{\text{EM}} \\
&= \frac{1}{N} \sum_{n=1}^N \underbrace{f\left(\mathbf{x}_n | \hat{\Theta}_k^{(i)}\right)}_{\text{EM} \equiv \text{NMF}}
\end{aligned} \tag{3.46}$$

Since the  $AZ - LL$  and  $Q$ -function stand for the same, the both approaches can be supposed as equivalent. The EM performs MLE just by maximizing the  $Q$ -function and the NMF by  $AZ - LL$ .  $\square$

For completeness, there are written down in Tab. 3.1 the symbols that are used in this thesis and in [6, 29].

The equations described in Def. 3.4 describes adaptive process as unsupervised where parameters are changed in order to maximize similarity between input signals  $\mathbf{X}$  and mixture of densities. The NMF can be switched to supervised or semi-supervised learning if a teacher explicitly determines the memberships of input signals  $\mathbf{X}$  in corresponding classes  $f\left(\mathbf{x}_n | \hat{\Theta}_k^{(i)}\right)$ . Once the memberships are set up they should be kept as constants regardless to current parameters  $\hat{\Theta}_k^{(i)}$ .

### 3.2.4 Maximum Likelihood Adaptive Neural System

Maximum Likelihood Adaptive Neural System (MLANS) [36] plays a special role for NMF. The MLANS is specific realization of the NMF equations for multivariate Normal distribution. MLANS is intended for problems which require an adaptive estimation of metrics in clustering input spaces. The adaptivity of the metrics is achieved by adapting covariance matrix  $\Sigma$  that determines how a current estimate varies. The model itself is represented by mean  $\mu$ . The MLANS can be used as unsupervised, supervised and semi-supervised system. The supervised learning supposes given  $f\left(\mathbf{x}_n | \Theta_k\right) \forall \mathbf{x}_n \in \mathbf{X}, \forall k \in \{1, \dots, K\}$ .

Based on the result from Theorem 3.5, the MLANS is equivalent with a mixture of the Normal distributions where parametric model adaptation is performed by the EM.

The MLANS learning equations in unsupervised variant are defined as follows:

**Definition 3.6.** Suppose task of MLE of parameters  $\Theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^K$  for a mixture  $K$  multivariate Normal distributions  $\mathcal{N}(\mathbf{x}|\mu, \Sigma)$  from Eq. 3.1 with update equations:

$$\begin{aligned}
1. \quad & f(\mathbf{x}_n | \hat{\Theta}_k^{(i)}) = \frac{\hat{\pi}_k^{(i)} \mathcal{N}(\mathbf{x}_n | \hat{\Theta}_k^{(i)})}{\sum_{j=1}^K \hat{\pi}_j^{(i)} \mathcal{N}(\mathbf{x}_n | \hat{\Theta}_j^{(i)})} \\
2. \quad & \hat{\Sigma}_k^{(i+1)} = \alpha \sum_{n=1}^N f(\mathbf{x}_n | \hat{\Theta}_k^{(i)}) (\mathbf{x}_n - \hat{\mu}_k^{(i)}) (\mathbf{x}_n - \hat{\mu}_k^{(i)})^T \\
3. \quad & \hat{\mu}_k^{(i+1)} = \frac{\alpha}{N} \sum_{n=1}^N f(\mathbf{x}_n | \hat{\Theta}_k^{(i)}) \mathbf{x}_n \\
4. \quad & \hat{\pi}_k^{(i)} = \frac{\sum_{k=1}^K f(\mathbf{x}_n | \hat{\Theta}_k^{(i)})}{N}
\end{aligned} \tag{3.47}$$

Where the  $\alpha$  is coefficient that regulates learning step length. The MLANS works iteratively with some initial guess of parameters  $\hat{\Theta}^{(0)} = \{\hat{\pi}_k^{(0)}, \hat{\mu}_k^{(0)}, \hat{\Sigma}_k^{(0)}\}_{k=1}^K$ . The initial guess of covariance matrices  $\hat{\Sigma}^{(0)}$  shall have larger eigenvalues to reflect initial uncertainty of the model which is measured by log-likelihood Eq. 3.6.

Above stated supposes unsupervised learning where parameters  $\Sigma_k^{(i+1)}$  and  $\mu_k^{(i+1)}$  are initially (randomly) guessed and  $f(\mathbf{x}|\Theta_k)$  computed according to parameter setting. If supervised or semi-supervised learning is needed, the  $f(\mathbf{x}|\Theta_k)$  are set up by the known memberships for the input data  $\{\mathbf{x}_n, \mathbf{d}_n\}$ <sup>4</sup>.

EM algorithm	Neural Modeling Fields
$\pi_k p(\mathbf{x}_n   \Theta_k)$	$l(k n)$
$\ln \pi_k p(\mathbf{x}_n   \Theta_k)$	$ll(k n)$
$p(y_n = k   \mathbf{x}_n, \Theta_k)$	$f(k n)$
$\mathcal{L}(\Theta   \mathbf{x}_n)$	$l(n)$
$\ln \mathcal{L}(\Theta   \mathbf{x}_n)$	$ll(n)$
$\Theta_k$	$\mathbf{S}_k$
$\mu_k$	$\mathbf{M}_k$
$\Sigma_k$	$\mathbf{C}_k$
$\pi_k$	$r(k)$

TABLE 3.1: Comparison between notion used in this thesis (mostly based on [12, 13, 31]) and by Perlovsky's in NMF in [6, 29].

### 3.2.5 Perlovsky's theory of mind and NMF

The significant contribution of NMF is in approach how an internal conceptualization of input signals is made in the mind. The conceptualization is a generalization a process where input signals are processed to give a concept of general object described by features (here a set of parameters  $\Theta$ ) instead of storing all objects. For example, a car is an object which shares

<sup>4</sup>The  $|\mathbf{d}_n| = K$  where  $K$  is number classes.



similar shape and sound with the other cars, so their common features can be used as object conceptualization instead of storing the entire objects.

The adaptation mechanism of obtaining features of some objects is just described by NMF adaptive equations. The system starts initially with high uncertainty of fuzziness, and its parameters are consecutively adapted to achieve concept-input signal similarity.

A thought process involves a number of sub-tasks including working with internal representation of thought and their manipulation, attention, concept formation, knowledge retrieval, generalization, recognition, understanding, imagination, intuition, emotion, decisions and reasoning. A minimal subset of these processes is called *an elementary thought process* (ETP) [29, 33]. The ETP involves mechanisms for afferent and efferent signals (see Fig. 3.4). The afferent signals are represented by input signals  $\mathbf{X}$ , and the efferent are represented by a current model based on parameters  $\Theta$ . Resonances between afferent and efferent signals are caused by high similarity between input signals  $\mathbf{X}$ , and a model  $\Theta$ , it means that some input signals are represented by the concept  $k$  with internal parameters  $\Theta_k$ .

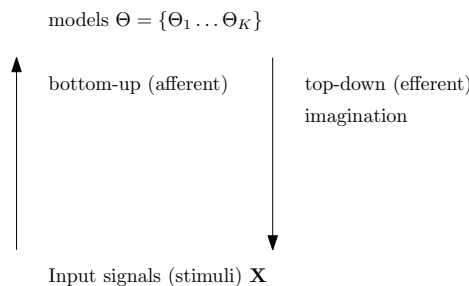


FIGURE 3.4: Division of afferent and efferent signals.

### 3.2.5.1 Understanding and meaning

The subsets of incoming signals are recognized in ETP by creating *phenomena* which are understood as objects. It means that subsets of signals are interpreted by its *meaning*. The objects are glued with models by emotional signals to instincts<sup>5</sup>. The NMF artificially embodies knowledge instinct (with models that respond to a particular signal) and behaviour of learning (adaptive equations in Def. 3.4) [33].

Another aspect of understanding and meaning is that stimuli (input signal) can be perceived as a more general concept in hierarchically higher layers (a car same as motorcycle can be perceived as more general concept vehicle although at the first sight, their visual features are different but they both have transportation purpose in common). As the stimuli goes toward the most general concept, the system can come up with such models [33]:

- scientific concept - a model of the universe.
- psychological concept - a model of self.

<sup>5</sup>An innate, typically fixed pattern of behaviour in animals in response to certain stimuli [37].

- philosophical concept - a model of meaning of existence.
- theological concept.

### 3.2.5.2 Imagination

Visual imagination involves excitation of a neural pattern in a visual cortex in the absence of an actual sensory stimulation. Imagination is often considered being a part of the thinking process. Kant likened process of thinking as [33]:

A play of cognitive function of imagination and understanding.

Kant's thought about thinking contains imagination as an integral part of the mind process. In terms of NMF, the imagination part of model excitation is a random stimulus (input signal) or taking parameters that significantly describe current model ( $\mu_k \subset \Theta_k$  which describes general concept of a certain class of inputs).

## Chapter 4

# Hierarchical mixture of experts

The EM algorithm is not restricted only on density estimation. The aim of this chapter is to show that the EM algorithm can carry out more sophisticated tasks than density estimation. Hierarchical Mixture of Experts (HME) is supervised network model<sup>1</sup> whose adaptation is based on NMF/EM maximum likelihood principle. The HME uses divide and conquer strategy to split the problem into smaller parts where specialized units called *gates* divide input space into multiple regions where units called *experts* are adapted to assigned region of the supervised dataset  $\{\mathbf{X}, \mathbf{d}\}$ .

### 4.1 Introduction

Hierarchical mixture of experts (HME) is tree structured supervised model, originally introduced in [38]. The HME resembles neural networks but the units that perform computation are divided into two types: *experts* and *gates*, in which each type has a different meaning. The experts are units which perform input-output  $\{\mathbf{X}, \mathbf{D}\}$  mapping and the gates determine how much the experts contribute to the output for a given input signal. Architecture the HME is parallel where both types of the units obtain an input vector and perform the computation. The experts made a decision and the same time gates perform weighting of outputs of the experts. The outputs from the previous computation are weighted again by other gates that lie hierarchically higher (nearer to output). Topology of HME with two layers is depicted in Fig. 4.1. The HME consists of  $n_1 + \dots + n_m$  experts where the experts are divided into  $m$  experts groups (see Fig. 4.3) and gates where each experts group has its gate and one gate for experts group outputs.

Informally the experts are the *output-makers* and gates are *divisors* that divide input space and assign regions of the input space to particular experts. In computer science the principle of dividing space is also known as *divide-and-conquer*. On the highest level (toward to output), the input space is divided into  $m$  subspaces where each experts group has the highest weight for

---

<sup>1</sup>The tree structure encourages to consider HME as a neural network but the reason why the term network is used is because the HME has a tree structure, which reminds neural networks.

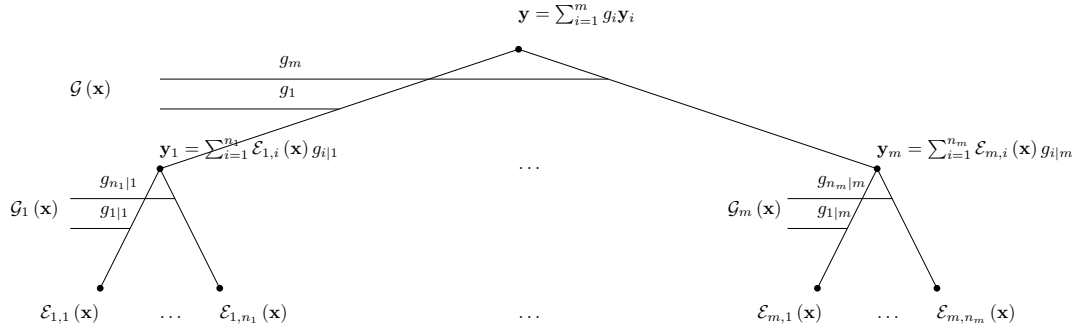


FIGURE 4.1: Hierarchical Mixture of Experts architecture

assigned region and as divisions go toward to experts the subspaces are divided into the other subspaces as Fig. 4.2 depicts.

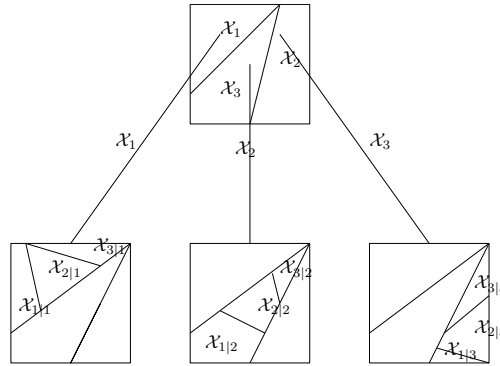
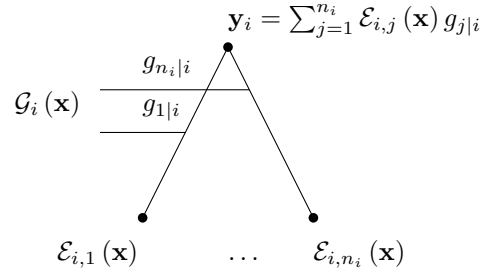


FIGURE 4.2: Divide and conquer principle applied on two dimensional input space. The space is consecutively divided into smaller pieces where each subspace has assigned one expert.

For purposes of this thesis, only two layered HME is supposed. Theoretically it is possible to construct HME with an arbitrary number of layers [39], but in most of the cases, this level of sophistication is not necessary. If the network is larger than necessary, some numerical instabilities may occur.

#### 4.1.1 Computation

The HME is supervised technique, so the goal of the learning phase is to find such parameters that the HME perform the mapping of input vectors  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N]$  on output space with the smallest error from the vector of desired outputs  $\mathbf{d} = [d_1, \dots, d_N]$ . The experts group is one sub-tree with gate and group of gates (see Fig. 4.3). Outputs of the experts are denoted by  $\mathcal{E}_{i,j}(\mathbf{x})$  where  $i$  is an experts group and  $j$  is the index of the expert in a particular experts group  $i$ . Outputs of the gates are denoted as  $g_{j|i}(\mathbf{x})$  for the gates connected to the experts outputs, where  $i$  stands for  $i$ -th experts group and  $j$  is the index of expert, which is weighted by the gate's output. The  $g_i(\mathbf{x})$  is the output of the gate at the top level where  $i$  is the weight for the  $i$ -th experts group.

FIGURE 4.3: Experts group in sub-tree  $i$ 

At the beginning, the inputs are presented to the experts. The inputs are presented sequentially datum-by-datum as it is done e.g. in neural networks. Each expert performs computation which is the evaluation of polynomial of  $DEG$ -th degree for the input vector  $\mathbf{x}$  (the polynomial regression is an extension of originally proposed linear [38]):

$$\mathcal{E}_{i,j}(\mathbf{x}) = \sum_{d=1}^D \sum_{deg=0}^{DEG} u_{d,deg}^{i,j} x_d^{deg} \quad (4.1)$$

The  $u_{i,j}$  are parameters of the expert. The  $u_{i,j}$  values can be represented as a matrix where rows are corresponding input dimensions and columns are coefficients of polynomial for each degree.

$$\mathbf{U}^{i,j} = \begin{pmatrix} u_{1,0}^{i,j} & u_{1,1}^{i,j} & \cdots & u_{1,DEG}^{i,j} \\ \vdots & \vdots & \ddots & \vdots \\ u_{D,0}^{i,j} & u_{D,1}^{i,j} & \cdots & u_{D,DEG}^{i,j} \end{pmatrix} \quad (4.2)$$

The first columns of the  $\mathbf{U}^{i,j}$  is bias parameter.

Every experts group has its gate  $\mathcal{G}_i(\mathbf{x}) = [g_{1|i}(\mathbf{x}), \dots, g_{n_i|i}(\mathbf{x})]$  with  $|\mathcal{G}_i(\mathbf{x})| = n_i$  outputs. The gate performs weighting of expert decisions in the frame of the experts group where  $g_{1|i}(\mathbf{x}) + \dots + g_{n_i|i}(\mathbf{x}) = 1$  must hold.

The regions are soft-partitioned by a straight line. The soft-partition is performed by *soft-max* function. Each gate has weight vector  $\mathbf{v}_{j|i} = [v_0^{j|i}, \dots, v_D^{j|i}]^T$  (the  $v_0^{j|i}$  is bias) for each expert where  $\mathbf{v}_{j|i}$  determines the position a line in an input space that divides the input  $D$ -dimensional Euclidean space into  $n_i$  regions as it is depicted in Fig. 4.4. The gate output  $g_{j|i}$  is calculated as follows:

$$\begin{aligned} \xi_{j|i} &= \mathbf{v}_{j|i}^T \mathbf{x} \\ g_{j|i}(\mathbf{x}) &= \frac{\exp \xi_{i,j}}{\sum_{k=1}^{n_i} \exp \xi_{i,k}} \end{aligned} \quad (4.3)$$

After gates outputs are calculated, the experts group  $i$  output is calculated as follows:

$$\mathbf{y}_i(\mathbf{x}) = \sum_{j=1}^{n_i} \mathcal{E}_{i,j}(\mathbf{x}) g_{j|i}(\mathbf{x}) \quad (4.4)$$

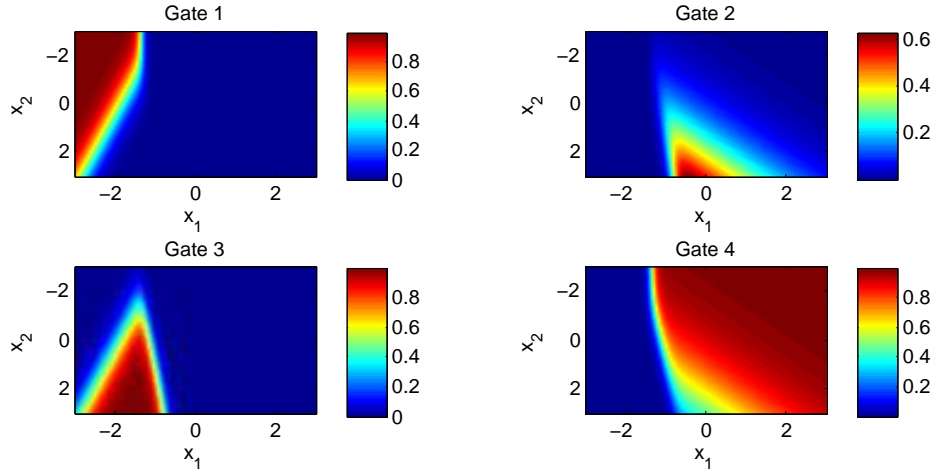


FIGURE 4.4: A particular space division into four parts performed by gates. Each figure represents the output for values  $[x_1, x_2]$  of one gate, which is weight for one expert outputs.

After each of  $m$  the experts group perform computation of  $\mathbf{y}_i$ , the value  $\mathbf{y}_i$  is again weighted by the output-level gate output vector  $\mathcal{G}(\mathbf{x}) = [g_1(\mathbf{x}), \dots, g_m(\mathbf{x})]^T$  as follows:

$$\mathbf{y}(\mathbf{x}) = \sum_{i=1}^m \mathbf{y}_i(\mathbf{x}) g_i(\mathbf{x}) \quad (4.5)$$

The output level gate has  $\mathbf{v}_i = [v_0^i, \dots, v_D^i]$  weights, where one  $\mathbf{v}_i$  is for one experts group. The calculation is performed similarly as for 4.3:

$$\begin{aligned} \xi_i &= \mathbf{v}_i^T \mathbf{x} \\ g_i(\mathbf{x}) &= \frac{\exp \xi_i}{\sum_{k=1}^m \exp \xi_k} \end{aligned} \quad (4.6)$$

The HME output for an input datum  $\mathbf{x}$  is formulated as follows:

$$\mathbf{y}(\mathbf{x}) = \sum_{i=1}^m g_i(\mathbf{x}) \underbrace{\sum_{j=1}^{n_i} \mathcal{E}_{i,j}(\mathbf{x}) g_{j|i}(\mathbf{x})}_{\text{experts group}} \quad (4.7)$$

From algorithmic perspective the HME performs calculation described in Alg. 2.

Each sum of the experts group in Eq. 4.7 reminds the mixture of densities. In the mixture of densities the coefficients are supposed to be constants (prior) while the coefficients in the HME are variables that depend on input  $\mathbf{x}$ .

### 4.1.2 Interpretation of the architecture

The outputs of the gates that perform division of the input space can be interpreted as a sequence of decisions that lead to a selection of proper expert whose output is the approximation input-output pairs [38]. In adaptation phase, the parameters are dependent on the level of expert



Probabilities of unobserved variable  $z_{i,j}^n$  (more formally  $p(z_{i,j}^n | \mathbf{y}(\mathbf{x}_n), \Theta)$ ) are present in HME to simplify maximization via  $Q$ -function as in EM algorithm. To stay consistent with the notion from [38] the probability of unobserved variable is denoted by symbol  $h_{i,j}^n$  instead of  $p(z_{i,j}^n | \mathbf{y}(\mathbf{x}_n), \Theta)$ . The probability of unobserved variable  $z_{i,j} = 1$  is defined as follows:

$$h_{i,j}^n = p(z_{i,j}^n = 1 | \mathbf{y}(\mathbf{x}_n), \Theta) = \frac{g_i(\mathbf{x}_n) g_{j|i}(\mathbf{x}_n) \mathcal{N}(\mathbf{y}(\mathbf{x}_n) | d_n, \Sigma)}{\sum_{k=1}^m \sum_{l=1}^{n_i} g_k(\mathbf{x}_n) g_{l|k}(\mathbf{x}_n) \mathcal{N}(\mathbf{y}(\mathbf{x}_n) | d_n, \Sigma)} \quad (4.9)$$

Additionally it is useful to define probabilities  $h_i^n$  and  $h_{j|i}^n$ . The  $h_i^n$  stands for the unobserved variable  $z_i^n$  which is activation of the gate at the top level and probability  $h_{j|i}^n$  of unobserved variable  $z_{j|i}$  that is activation of the  $j$ -th gate in  $i$ -th experts group  $i$ . The value  $h_i^n$  is obtained from Eq. 4.9 by marginalization over all values  $j$ -s<sup>3</sup>:

$$\begin{aligned} h_i^n &= \sum_{j=1}^{n_i} h_{i,j}^n \\ &= \sum_{j=1}^{n_i} p(z_{i,j}^n | \mathbf{y}(\mathbf{x}_n), \Theta) \\ &= \frac{g_i \sum_{l=1}^{n_i} g_{l|i}(\mathbf{x}_n) \mathcal{N}(\mathbf{y}(\mathbf{x}_n) | d_n, \Sigma)}{\sum_{k=1}^m \sum_{l=1}^{n_i} g_k(\mathbf{x}_n) g_{l|k}(\mathbf{x}_n) \mathcal{N}(\mathbf{y}(\mathbf{x}_n) | d_n, \Sigma)} \end{aligned} \quad (4.10)$$

The value of  $h_{j|i}$  is obtained from Eq. 4.9 by modifying with the formula for conditional probability<sup>4</sup>:

$$\begin{aligned} h_{j|i}^n &= \frac{h_{i,j}^n}{h_i^n} \\ &= \frac{g_{j|i} \mathcal{N}(\mathcal{E}_{i,j}(\mathbf{x}_n) | d_n, \Sigma)}{\sum_{l=1}^{n_i} g_{l|i} \mathcal{N}(\mathcal{E}_{i,l}(\mathbf{x}_n) | d_n, \Sigma)} \end{aligned} \quad (4.11)$$

The unobserved variable  $z_{i,j}^n$  eliminates paths to experts which do not contribute to output in complete likelihood function Eq. 4.12. If  $\mathcal{E}_{i,j}$  is the expert whose result is used as the output then  $z_{i,j} = 1$  and other paths vanish (because  $\forall c \in \mathbb{R} : c^0 = 0$ ) and only the  $\mathcal{E}_{i,j}$  expert's parameters are adapted to observed data. Complete log-likelihood based on the Eq. 4.8 is defined as follows:

$$\begin{aligned} \ln \mathcal{L}(\Theta | \mathbf{X}, \mathbf{D}, \mathbf{z}) &= \sum_{n=1}^N \sum_{i=1}^m \sum_{j=1}^{n_i} z_{i,j}^n \ln (g_i(\mathbf{x}_n) g_{j|i}(\mathbf{x}_n) \mathcal{N}(\mathbf{y}(\mathbf{x}_n) | d_n, \Sigma)) \\ &= \sum_{n=1}^N \sum_{i=1}^m \sum_{j=1}^{n_i} z_{i,j}^n (\ln g_i(\mathbf{x}_n) + \ln g_{j|i}(\mathbf{x}_n) + \ln \mathcal{N}(\mathbf{y}(\mathbf{x}_n) | d_n, \Sigma)) \end{aligned} \quad (4.12)$$

The same recipe as in Chapter 3 is applied here to eliminate the unobserved variables  $z_{i,j}^n$ . The unobserved variable  $z_{i,j}^n$  is modelled by probability  $h_{i,j}^n$  and the complete log-likelihood is defined in Eq. 4.12 thus the  $Q$ -function for the HME is defined as follows:

---


$$\begin{aligned} {}^3 p(\mathbf{x}) &= \int_{\mathbf{y} \in \Omega_{\mathbf{y}}} p(\mathbf{x}, \mathbf{y}) d\mathbf{y} \\ {}^4 p(\mathbf{x} | \mathbf{y}) &= \frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{y})} \end{aligned}$$



$$\begin{aligned}
Q(\Theta^{(t+1)}, \Theta^{(t)}) &= \sum_{n=1}^N \sum_{i=1}^m \sum_{j=1}^{n_i} \underbrace{h_{i,j}^n}_{p(z_{i,j}^n | \mathbf{y}(\mathbf{x}_n), \Theta)} \ln(g_i(\mathbf{x}_n) g_{j|i}(\mathbf{x}_n) \mathcal{N}(\mathbf{y}(\mathbf{x}_n) | d_n, \Sigma)) \\
&= \sum_{n=1}^N \sum_{i=1}^m \sum_{j=1}^{n_i} h_{i,j}^n (\ln g_i(\mathbf{x}_n) + \ln g_{j|i}(\mathbf{x}_n) + \ln \mathcal{N}(\mathbf{y}(\mathbf{x}_n) | d_n, \Sigma))
\end{aligned} \tag{4.13}$$

The EM algorithm performs two steps in each iteration. The first expectation step **E** where the unknowns of the  $Q$ -function are calculated (the  $h_{i,j}^n$ ) and the second maximization step **M** that maximize the conditional expectation of complete log-likelihood in Eq. 4.13 given observed variables, current parameters  $\Theta^{(i)}$  and probabilities of unobserved variables.

The set of estimated parameters is  $\Theta^{(k)} = \left\{ \left\{ \mathbf{v}_i^{(k)} \right\}_{j=1}^m, \left\{ \left\{ \mathbf{v}_{i|j}^{(k)} \right\}_{j=1}^{n_i} \right\}_{i=1}^m, \left\{ \left\{ \mathbf{U}_{i,j}^{(k)} \right\}_{j=1}^{n_i} \right\}_{i=1}^m \right\}$ .

The formulae for the **M** step that maximize the Eq. 4.13 can be obtained by taking partial derivative by optimized parameter. The **M** step can be divided into three independent maximization tasks [38]:

$$\begin{aligned}
\hat{\mathbf{U}}_{i,j}^{(k+1)} &= \arg \max_{\hat{\mathbf{U}}_{i,j}^{(k+1)}} \sum_{n=1}^N h_{i,j}^n \ln \mathcal{N}(\mathbf{y}(\mathbf{x}_n) | d_n, \Sigma) \\
\hat{\mathbf{v}}_{i|j}^{(k+1)} &= \arg \max_{\hat{\mathbf{v}}_{i|j}^{(k+1)}} \sum_{n=1}^N \sum_{i=1}^m h_i^n \sum_{j=1}^{n_i} h_{j|i}^n \ln g_{j|i}(\mathbf{x}_n) \\
\hat{\mathbf{v}}_i^{(k+1)} &= \arg \max_{\hat{\mathbf{v}}_i^{(k+1)}} \sum_{n=1}^N \sum_{i=1}^m h_i^n \ln g_i(\mathbf{x}_n)
\end{aligned} \tag{4.14}$$

All maximizations from Eq. 4.14 give rise to the least square problem (LSQ) and weighted least LSQ (WLSQ) [38]. The WLSQ for polynomial regression, which is the solution for Eq. 4.14, has closed form formalized by Def. 4.1.

**Definition 4.1.** Weighted least-squares solution  $\hat{\mathbf{u}}$  for polynomial regression of degree  $DEG$  for a set of observation values  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N]$  where each datum is a  $D$  dimensional column vector  $\mathbf{x}_n = [x_{n,1}, \dots, x_{n,D}]^T$ , desired values  $\mathbf{d} = [d_1 \dots d_N]^T$  and weights  $\text{diag}(\mathbf{W}) = [w_0, w_1, \dots, w_N]$  is defined as follows:

$$\hat{\mathbf{u}}^{MLE} = (\mathbf{X}_{DEG}^T \mathbf{W} \mathbf{X}_{DEG})^{-1} \mathbf{X}_{DEG}^T \mathbf{W} \mathbf{d} \tag{4.15}$$

where  $\mathbf{X}_{DEG}$  a Vandermonde matrix [40]:

$$\mathbf{X}_{DEG} = \begin{pmatrix} 1 & (\mathbf{x}_1^1)^T & \dots & (\mathbf{x}_1^{DEG})^T \\ \vdots & \vdots & \ddots & \vdots \\ 1 & (\mathbf{x}_N^1)^T & \dots & (\mathbf{x}_N^{DEG})^T \end{pmatrix} \tag{4.16}$$

Ordinary LSQ performed by Eq. 4.15 for  $\text{diag}(\mathbf{W}) = [1, \dots, 1]$ .

The **M** step in Eq. 4.14 is (solved by LSQ and WLSQ regression defined in Def. 4.1) formulated as follows:

$\hat{\mathbf{U}}_{i,j}^{t+1}$  For each expert  $i, j$  solve polynomial WLSQ of degree  $DEG$  with observations  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N]$ , desired values  $\mathbf{d} = [d_1, \dots, d_N]^T$  and weights  $\mathbf{h}_{i,j} = [h_{i,j}^1, \dots, h_{i,j}^N]^T$ .

$\hat{\mathbf{v}}_i^{t+1}$  For top-level gate  $i$ -th weight solve polynomial LSQ of degree 1 with observations  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N]$  and desired values  $\ln \mathbf{h}_i = [\ln h_i^1, \dots, \ln h_i^N]^T$ .

$\hat{\mathbf{v}}_{i,j}^{t+1}$  For gate in  $i$ -th experts group and  $j$ -th weight solve polynomial WLSQ of degree 1 with observations  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N]$ , desired values  $\ln \mathbf{h}_{j|i} = [\ln h_{j|i}^1, \dots, \ln h_{j|i}^N]^T$  and weights  $\mathbf{h}_i = [h_i^1, \dots, h_i^N]^T$ .

For covariance matrix,  $\Sigma$  is the situation quite tricky. Large  $\Sigma$  causes a high degree of freedom in learning because high error will not affect the update rapidly. On the other hand, small  $\Sigma$  causes strict elimination of candidate experts and then no experts adapts to this space. In [39] is recommended to calculate weighted covariance matrix with weights  $\mathbf{h}_{i,j}$ :

$$\hat{\Sigma}_{i,j}^{MLE} = \frac{1}{N} \sum_{n=1}^N h_{i,j}^n (\mathbf{y}_n - \mathbf{d}_n)^T (\mathbf{y}_n - \mathbf{d}_n) \quad (4.17)$$

The  $\hat{\Sigma}_{i,j}^{MLE}$  in Eq. 4.17 is actually the MLE solution for the  $\Sigma$  [39]. The  $\hat{\Sigma}_{i,j}^{MLE}$  does not remove the problem of small/large  $\Sigma$ . The  $\hat{\Sigma}_{i,j}^{MLE}$  is the source of many numerical instabilities, mainly if the expert has very small region then  $\hat{\Sigma}_{i,j}^{MLE}$  approaches zero and the whole procedure fails ( $\Sigma \approx 0$  is singular for  $\mathcal{N}(\mathbf{y}(\mathbf{x})|d, \Sigma)$  which is inadmissible). One method is to use uniform  $\Sigma$  and alternatively decrease the value with iterations, but it will not reflect currently assigned regions. The other method is to calculate ordinary covariance matrix as in Eq. 2.39 where  $\mu = \mathbf{d}$  or set up bounds for smallest eigenvalues (in this particular the  $|d_n| = 1$ , thus  $\Sigma$  is a single value) of the  $\Sigma$ . In implementation, the second mentioned method is used where minimal bound value is current ordinary covariance of the outputs and desired values which is defined as follows:

$$\hat{\Sigma}_{i,j}^{(k+1)} = \min \left\{ \hat{\Sigma}_{i,j}^{MLE}, \frac{1}{N} \sum_{n=1}^N (y_n - d_n)^2 \right\} \quad (4.18)$$

Entire learning procedure is summarized in Alg. 3.

### 4.3 Experiments

Two experiments are performed to compare performance of the HME with feed-forward neural network with one hidden layer (NN). As the benchmark two function approximation problems are chosen. The benchmark is a multi-modal one-dimensional function  $f(x) = \exp \sin(-x + 2)$  on interval  $(1, 2\pi)$  (see Fig. 4.5, left) and the second experiment is an approximation of multi-modal two-dimensional function  $f(x, y) = |\sin x|y$  in interval  $x, y \in (-3, 3)$  (see Fig. 4.5, right).

The HME has following configuration:

**The first approximation problem** Two experts groups where each group has two experts (22 weights).

**Algorithm 3:** The HME training procedure

---

**Inputs** : A set of input vectors  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N]$  with desired vectors  $\mathbf{d} = [d_1 \dots d_n]^T$ , number of experts in each experts group  $[n_1, \dots, n_m]$ , degree of expert's polynomial  $DEG$ .

**Outputs:** A parameter estimate  $\hat{\Theta} = \left\{ \left\{ \mathbf{v}_i \right\}_{j=1}^m, \left\{ \left\{ \mathbf{v}_{i|j} \right\}_{j=1}^{n_i} \right\}_{i=1}^m, \left\{ \left\{ \mathbf{U}_{i,j} \right\}_{j=1}^{n_i} \right\}_{i=1}^m \right\}$ .

```

// Guess the initial parameters  $\hat{\Theta}^{(0)}$ ;
 $\hat{\Theta}^{(0)} = \text{Initialize}()$ ;
// algorithm iterations
 $\mathbf{X}_{DEG} \leftarrow \text{Vandermore}(\mathbf{X})$  // calculate Vandermore matrix (Eq. 4.16)
for  $k \leftarrow 1$  to  $MAX\_ITER$  do
  // E step
  // Evaluate outputs with method Alg. 2
  // The outputs of the Alg. 2 are column vectors  $\mathbf{y} = [y_1, \dots, y_N]^T$ 
   $\mathbf{y} \leftarrow \text{Feedforward}(\mathbf{X}, \hat{\Theta}^{(k)})$ ;
  // Evaluate probabilities of unobserved variables  $h_{i,j}$  for each datum
  for  $n \leftarrow 1$  to  $N$  do
    for  $i \leftarrow 1$  to  $m$  do
      for  $j \leftarrow 1$  to  $n_i$  do
         $h_{i,j}^n = \frac{g_i(\mathbf{x}_n)g_{j|i}(\mathbf{x}_n)\mathcal{N}(\mathbf{y}(\mathbf{x}_n)|d_n, \Sigma)}{\sum_{k=1}^m \sum_{l=1}^{n_i} g_k(\mathbf{x}_n)g_{l|k}(\mathbf{x}_n)\mathcal{N}(\mathbf{y}(\mathbf{x}_n)|d_n, \Sigma)}$ ; // (Eq. 4.9)
  // Evaluate probabilities of unobserved variables  $h_i$  and  $h_{j|i}$  with  $h_{j,i}$ 
  for  $n \leftarrow 1$  to  $N$  do
     $h_i^n = \sum_{j=1}^m h_{i,j}^n$ ; // (Eq. 4.10)
     $h_{j|i}^n = \frac{h_{i,j}^n}{h_i^n}$ ; // (Eq. 4.11)
  // M step
  // Training experts
  // Experts:
  for  $i \leftarrow 1$  to  $m$  do
    for  $j \leftarrow 1$  to  $m_i$  do
       $\mathbf{W} \leftarrow \text{diag}(\mathbf{h}_{i,j})$ ; // form a diagonal matrix from  $\mathbf{h}_{i,j}$ -s
       $\hat{\mathbf{U}}_{i,j}^{(k)} \leftarrow (\mathbf{X}_p^T \mathbf{W} \mathbf{X}_p)^{-1} \mathbf{X}_p^T \mathbf{W} \mathbf{d}$ 
  // Top level gate
  for  $i \leftarrow 1$  to  $m$  do
     $\ln \mathbf{h}_i \leftarrow [\ln h_i^1 \dots \ln h_i^N]^T$ ; //  $h_i$ -s
     $\hat{\mathbf{v}}_i^{(k)} \leftarrow (\mathbf{X}_p^T \mathbf{X}_p)^{-1} \mathbf{X}_p^T \ln \mathbf{h}_i$ 
  // Experts gates
  for  $i \leftarrow 1$  to  $m$  do
    for  $j \leftarrow 1$  to  $n_i$  do
       $\mathbf{W} \leftarrow \text{diag}(\mathbf{h}_i)$ ; // form a diagonal matrix from  $\mathbf{h}_i$ -s
       $\hat{\mathbf{v}}_{j|i}^{(k)} \leftarrow (\mathbf{X}_p^T \mathbf{W} \mathbf{X}_p)^{-1} \mathbf{X}_p^T \mathbf{W} \ln \mathbf{h}_{j|i}$ 
 $\hat{\Theta} \leftarrow \hat{\Theta}^{(k)}$ ;

```

---

**The second approximation problem** Two experts groups where each group has four experts (37 weights).

The neural network has one hidden layer with 10 neurons with linear mapping units. The NN has 31 weights for the first problem and 41 weights for the second problem. Learning of the NN is performed by the scaled conjugate gradient method (NETLAB Toolbox for MATLAB [41]).

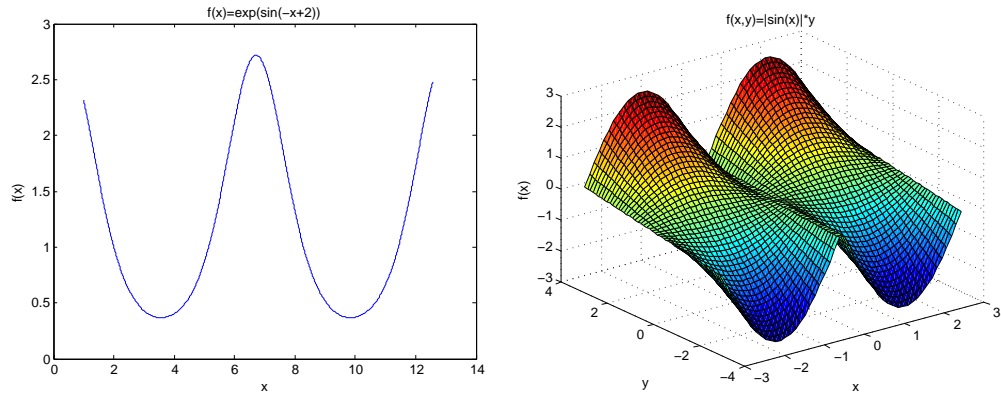


FIGURE 4.5: Benchmark functions for the HME. Left:  $f(x) = \exp \sin(-x+2)$  in interval  $x \in (1, 4\pi)$ ; Right:  $f(x, y) = |\sin x|y$  in interval  $x \in (-3, 3)$ .

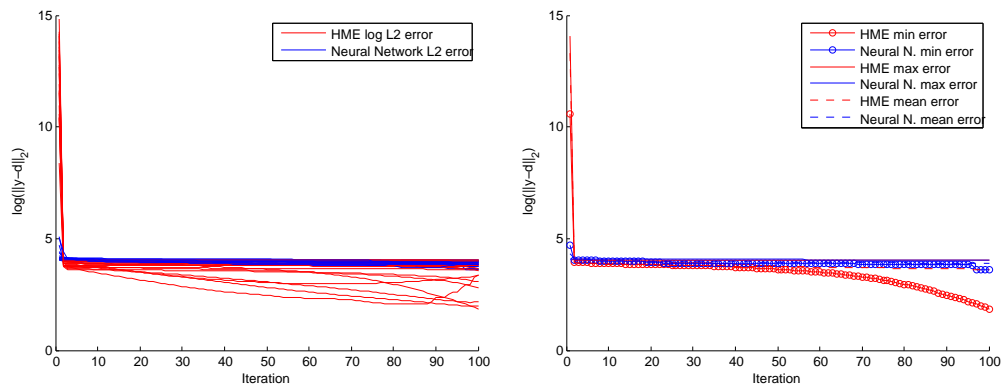


FIGURE 4.6: The HME and Neural Networks results comparison for the approximation of the first function. Left: Log-Euclidean distance between desired value  $d$  and output value for the HME and NN for 20 runs per each; Right: Special values of all experiments - minimal, mean and maximal Log-Euclidean error.

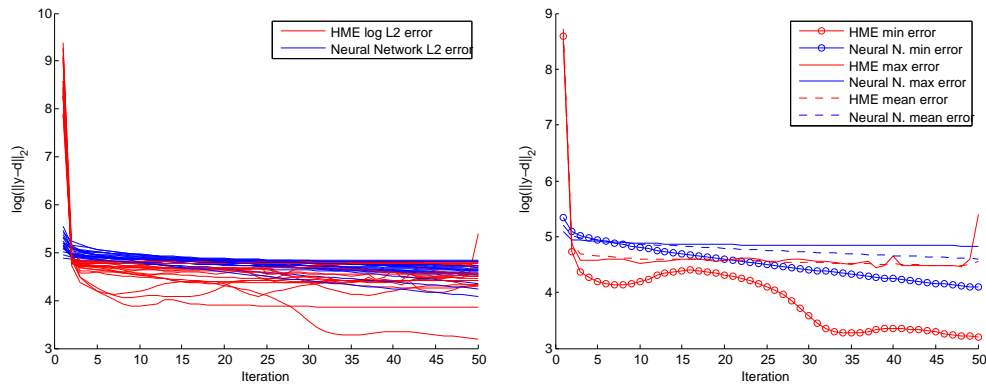


FIGURE 4.7: the HME and Neural Networks results comparison for the approximation of the second function. Left: Log-Euclidean distance between desired value  $d$  and output value for the HME and NN for 20 runs per each; Right: Special values of all experiments - minimal, mean and maximal Log-Euclidean error.

### 4.3.1 Discussion

The HME is a powerful approach for both approximation problems. The HME converges very quickly (10-20 iterations) to accurate approximations if it does not fail numerically. The large difference between the first and second iteration is because the **M** step in EM algorithm jumps to a maximum of the  $Q$ -function (Eq. 4.13). The initial guess of the parameters of the HME has much larger error in comparison to the initial guess of the NN. The  $Q$ -function has closed form (Def. 4.1) and converges very fast at the beginning, but the error difference rapidly slows down with iterations. It means that in the further iterations, the algorithm tries to refine parameter setting, but the first iteration of the Alg. 3 is essential. The only stop criterion of the HME is the number of iterations which, sometimes, caused an increasing trend of Euclidean error.

The NN converges slowly but, on the other hand, NN weight space is variable, and the results get better for more iterations. The 100 iterations for the first and 50 iterations for the second problem are too few. If the NN has 1000 iterations, then errors are rapidly smaller.

Despite worse results of the NN, the HME is not better only because HME minimize error faster. The main drawback of the HME is numerical instability for complex architectures. If an expert has too small  $h_{i,j}$ -s, it will never get over small  $h_{i,j}$  because it will never get a chance to update error due to too small weight in the WLSQ.

Significant advantage of the HME, in comparison to the NN, lies in interpretability of the architecture and parameters. The HME parameters are easy to interpret because the divide-and-conquer approach determines relevance of experts in regions of the input space (see Fig. 4.2 which is actually input space partition of one gate in experts group for the second approximation problem), while the NN is distributed approach where no universal recipe how to interpret weights and architecture is available.

## Chapter 5

# Feature Integration Theory - Experimental evaluation of MLANS

In this chapter, the hierarchical structure of the visual system based on Feature Integration Theory is shown. The proposed model consists of two layers where in the first layer features are classified into a corresponding class for each of the five features (colour, direction, size, texture, shape). The clustering is made by the NMF in the first layer. The NMF outputs are further classified in the second layer by the Kohonen's Self Organizing Map (SOM) into regions to reveal whether some of the low level signals exhibit some simultaneous activity. The Fig. 5.6 illustrates the structure of the entire system.

### 5.1 Introduction

#### 5.1.1 Feature integration theory

The Feature Integration Theory (FIT) of attention [42] suggests that object recognition in the human brain is performed by scene decomposition into a conjunction of features where each feature is processed separately if it is needed. Subsequently, the features are integrated and recognized as one particular object. The FIT supposes that the scene is analysed in early stages by specialized receptors that respond to properties as orientation, colour, spatial frequency or movement, and distribute those features into separate parts of brain [42]. The FIT registers early, automatically and in parallel across the visual field, while the objects are identified separately and only at later stage, which requires focused attention [42]. The problem of FIT lies in binding the decomposed scene. Once the scene is decomposed into separate features, the resulting information shall keep the information.

Suppose, for example, there are wooden doors to be recognized. The object's feature classes are rectangular shape, brown colour and wooded texture. Each feature class is recognized separately and, subsequently, in the integration phase the object is recognized from the present conjunction of feature classes as a door.

## 5.2 Description of experiment

The learning process is performed as unsupervised. One input consists of five 100 dimensional vectors:

$$\mathbf{x}_n = \{\mathbf{x}_n^{color}, \mathbf{x}_n^{direction}, \mathbf{x}_n^{shape}, \mathbf{x}_n^{size}, \mathbf{x}_n^{texture}\} \quad (5.1)$$

Each element of the  $\mathbf{x}_n$  stands for different feature, and each is processed by separate MLANS that classifies corresponding feature  $\mathbf{x}_n^{feature}$  of the input vector  $\mathbf{x}_n$ . Each input feature vector  $\mathbf{x}_n^{feature}$  is an image with 10 – by – 10 pixels thus learning is performed on 100 dimensional vectors per feature (100\*5 values for each input sample  $\mathbf{x}_n$ ). After each feature is processed by corresponding MLANS in the first layer, the output fuzzy memberships  $f(\mathbf{x}_n^{feature}|\Theta_k)$  (in notion of EM the probability  $p(y_n = concept|\mathbf{x}_n^{feature}|\Theta_k)$ ) are used as inputs at the second layer. The MLANSs outputs are again processed as vector of features  $\mathbf{y}_n$ :

$$\mathbf{y}_n = \{\mathbf{y}_n^{color}, \mathbf{y}_n^{direction}, \mathbf{y}_n^{shape}, \mathbf{y}_n^{size}, \mathbf{y}_n^{texture}\} \quad (5.2)$$

Each MLANS has  $K^{feature}$  classes where the  $K^{feature}$  is the number of different classes defined in labels of the input data. The outputs  $\mathbf{y}_n^{feature}$  of the MLANSs are the fuzzy memberships of presence of each feature the input vector  $\mathbf{x}_n^{feature}$ .

The second layer performs clustering of the memberships  $\mathbf{y}_n$  from the first layer. The purpose of the second layer is to find receptive fields of features without any prior knowledge for learning. As the classifier, the Kohonen's SOM is used. The SOM has some methods that are used to visualize high-dimensional models as matrices that are easy to visualise and interpret.

### 5.2.1 Scene features

The FIT describes how the scene processing is performed by the brain. The decomposition of the scene into several features is performed in earlier stages of the scene processing. Results of the scene decomposition are used for clustering of the objects of which is the scene made of. The features are restricted in following set of classes:

**Colour:** blue, cyan, green, magenta, olive, purple, red, teal, yellow.

**Directions:** diagonal, vertical, horizontal.

**Shapes:** crescent, cross, diamond, ellipse, heart, hexagon, pentagon, rectangle, star, triangle

**Sizes:** big, medium, small

**Textures:** dotted, grid, lined, plain, tweed

The visualisation of corresponding classes of features is shown in Fig. 5.1, 5.2, 5.3, 5.4 and 5.5

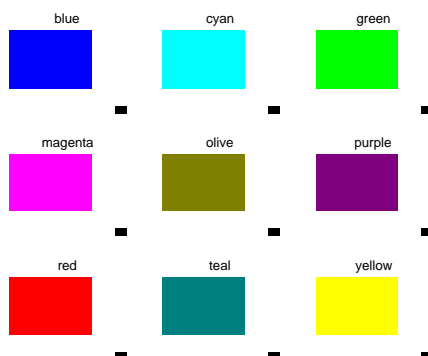


FIGURE 5.1: Visualisation inputs for each possible concept of feature *Colour*.

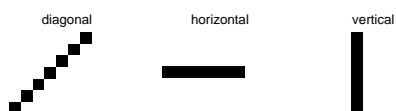


FIGURE 5.2: Visualisation inputs for each possible concept of feature *Direction*.

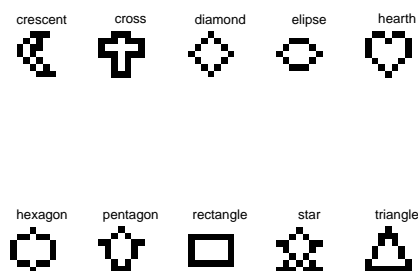


FIGURE 5.3: Visualisation inputs for each possible concept of feature *Shape*.



FIGURE 5.4: Visualisation inputs for each possible concept of feature *Size*.



FIGURE 5.5: Visualisation inputs for each possible concept of feature *Texture*.



### 5.2.2 Clustering in the first layer

The entire system has two layers. In the first layer, the features are clustered and classified into corresponding classes. The system has five separate MLANSs where each MLANS represents one of the features mentioned above. The input dimension is uniform for all features because the inputs are processed as images from the same scene. The number of separate classes of each feature is known thus MLANSs are initialized with their corresponding number of classes. In the second stage, the outputs of the MLANSs for each input vector  $\mathbf{x}_n$  are used as inputs  $\mathbf{y}_n$  for training the SOM (structure is shown in Fig. 5.6). The MLANSs outputs are encoded in  $1\text{-to-}K$  vectors, where the values are memberships of a particular class in the input vector. The SOM inputs  $\mathbf{y}_n$  has 30 dimensions (9 colours + 3 directions + 10 shapes + 3 sizes + 5 textures = 30 values), and there are 4050 possible combinations of features (9 colours  $\times$  3 directions  $\times$  10 shapes  $\times$  3 sizes  $\times$  5 textures = 4050 values).

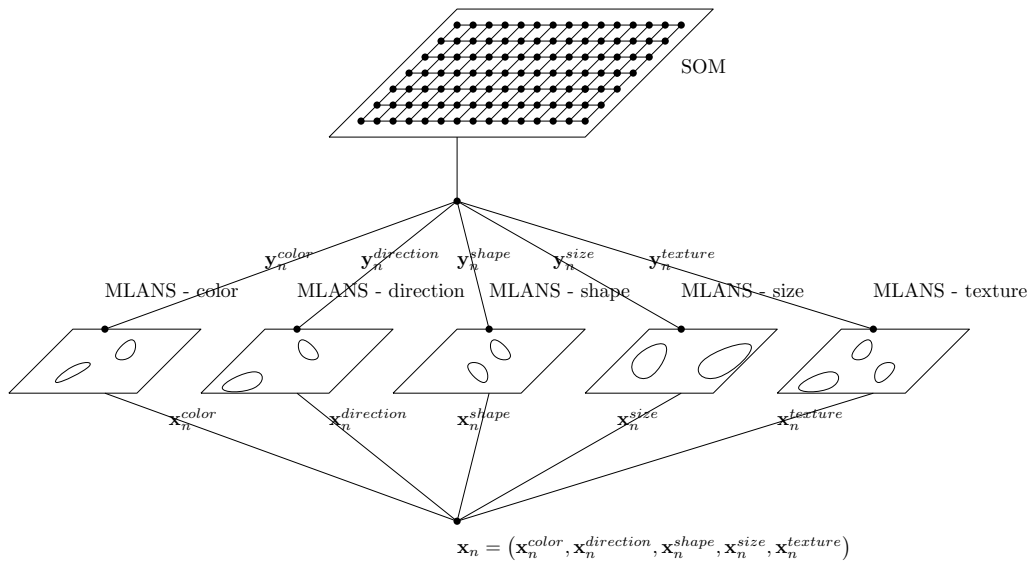


FIGURE 5.6: In the first layer input vectors,  $\mathbf{x}_n^{feature}$  for separate features are processed. After MLANS processes the input  $\mathbf{x}_n^{feature}$  the fuzzy memberships formed into  $\mathbf{y}_n$  are used as input for the SOM that finds receptive field of classified input vectors  $\mathbf{x}_n$ .

#### 5.2.2.1 Self Organising Maps

The results of the SOM can be evaluated from different perspectives. The first perspective is to use the best matching unit (the nearest neuron to an input datum w.r.t. some distance measure) as representative vector of a certain input class. The second perspective is to visualize weights of trained SOM. Visualization can give a clear insight into data separability and similarity of input vectors for relatively very high dimensionality. There are two important visualization tools, the first is called *U-matrix*, which is a matrix of distances of a neuron to topologically neighbouring neurons and the second tool is *component plane* which is a projection of a dimension of weight vectors into two dimensional space with respect to the topology.

Example of component planes and U-matrices for classification of the Fig. 1.6 are depicted in Fig. 5.7. The both tools provide relatively fast outlook into the trained SOM.

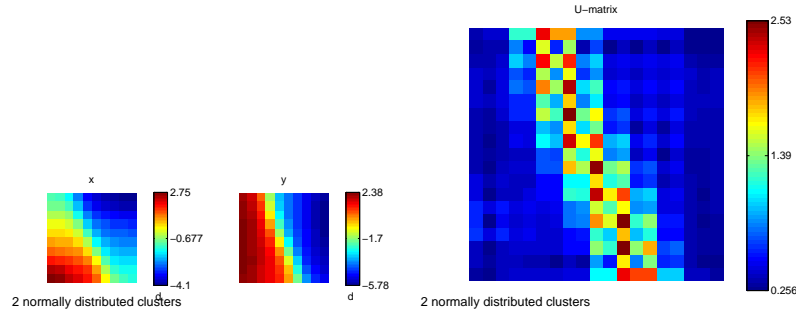


FIGURE 5.7: Left: Component planes, Right: The U-matrix. From the U-matrix, it is easy to see that there are two significant clusters because the distances of some neurons on the ridge are bigger than those in the clusters.

The plateaus in U-matrices represent a crowded group of neurons because their distance with neighbours is relatively small while ridges represent large distance with neighbours and represents the neurons that separate clusters.

## 5.3 Experiment

### 5.3.1 First layer - training MLANS

At the first phase the MLANSs are trained for clustering of each feature. Each feature class is represented as 100 dimensional Gaussian with mean vector  $\hat{\mu}^{(i)}$  and positive definite symmetric covariance matrix  $\hat{\Sigma}_k^{(i)}$ , thus each feature class is represented by  $100 + 100^2$  values (the  $\hat{\Sigma}^{(i)}$  is a symmetric matrix thus the values above or below diagonal may be ignore because they are redundant). The dataset has 81000 training vectors.

The MLANSs are initialized with respect to input vectors as follows:

$$\begin{aligned}\hat{\mu}_k^{(0)} &= \hat{\mu} + \frac{r}{10}\hat{\mu} \\ \hat{\Sigma}_k^{(0)} &= \frac{1}{N} \sum_{n=1}^N (\mathbf{x}_n - \hat{\mu})(\mathbf{x}_n - \hat{\mu})^T\end{aligned}\quad (5.3)$$

Where  $r \in (0, 1)$  is a random constant and  $\hat{\mu}$  is the arithmetic average of all inputs vectors  $\mathbf{x}_n^{feature}$  for the given *feature*. The initialization procedure of the  $\mu_k^{(0)}$  is practically only perturbed arithmetic mean. The  $\Sigma_k^{(0)}$  is initialised with covariance of all inputs for the given feature thus the eigenvalues are large enough to reflect high initial fuzziness.

The learning on MLANSs is performed 10 times and for each *feature* the MLANS with highest  $\ln \mathcal{L}(\Theta | \mathbf{X}^{feature})$  is taken. Since the mean vector  $\hat{\mu}^{(i)}$  describes significantly the a concept class the values  $\hat{\mu}^{(i)}$  are used as representatives. The mean vectors  $\hat{\mu}^{(end)}$  of the MLANSs with the highest likelihood are visualized in Fig. 5.8, 5.9, 5.10, 5.11 and 5.12 respectively.

From the Fig. 5.8, 5.9, 5.10, 5.11 and 5.12 is clear that some classes are not found or are clustered unambiguously. For example, the colours, the blue and yellow are clustered, because the algorithm felt into local optimum with general concepts where the colour is supposed to be

uniformly  $\approx 0.5$ . Another problem is that multiple classes are clustered as only one concept class, for example, the horizontal and the diagonal directions classes are clustered as one concept class while vertical is clustered into two same classes.

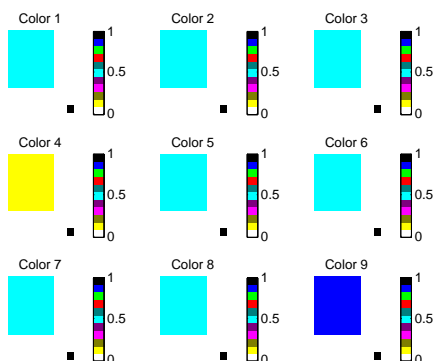


FIGURE 5.8: Mean values of trained MLANS for feature vectors with colour.

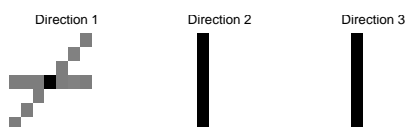


FIGURE 5.9: Mean values of trained MLANS for feature vectors with direction.

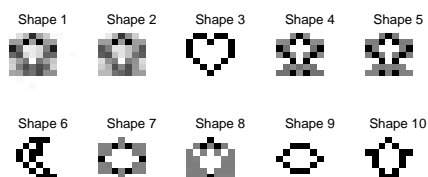


FIGURE 5.10: Mean values of trained MLANS for feature vectors with shape.

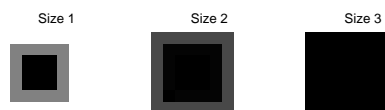


FIGURE 5.11: Mean values of trained MLANS for feature vectors with size.

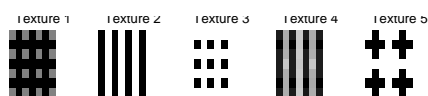


FIGURE 5.12: Mean values of trained MLANS for feature vectors with texture.

### 5.3.2 Second phase - Training the SOM

In the second phase, all input vectors are evaluated on learned MLANSs where the output vectors are probabilities of class memberships  $f(\mathbf{x}_n^{feature} | \Theta_{class}^{feature})$ . The MLANS outputs are used as 30 dimensional inputs for the SOM (9 colours, 3 directions, 10 shapes, 3 sizes and 5 textures).

The SOM Toolbox [43] is used for clustering the  $\mathbf{y}_n$ . Parameters of the SOM are: rectangular lattice, decreasing influence to neighbouring neurons that initially starts at 5 and decreases to 1 and 10 epochs.

#### 5.3.2.1 SOM results

The smallest lattice, whose results are significant enough, is the SOM with  $10 - by - 10$  neurons. The component planes for the SOM with  $10 - by - 10$  neurons are shown in Fig. 5.13 and the U-matrix in Fig. 5.14. The component planes for the SOM with  $50 - by - 50$  neurons are shown in Fig. 5.15 and the U-matrix in Fig. 5.14. The results of the experiment with the SOM with the same number of neurons as all possible combinations of features (4050) organized into  $90 - by - 45$  lattice is depicted in Fig. 5.17 and 5.18. Each component plane is one feature class because each input dimension of the SOM associated with one fuzzy membership of the MLANS. The visualization approximately reveals where there are bounds of the entire system. From Fig. 5.11 is apparent that MLANS which represents medium size is ambiguous which is reflected in a small difference between corresponding component planes. The worse results are for colour, where the MLANS clustered only three significant classes: one for blue, the another for yellow and reaming seven represent the third universal concept. The third and ninth component planes are clusters for yellow and blue, but the other classes are almost indistinguishable. For direction, only two is distinguished, one for the diagonal and horizontal and one for vertical direction. The second class is almost the same as the third. Texture clustering is well distinguished in comparison to other features. This is obvious from Fig. 5.12 where classes seem significantly different except of the second and fourth where there are black horizontal stripes, which propagate ambiguity in the corresponding component planes. Finally the third, sixth, ninth and tenth shapes are clustered perfectly as unique classes, which are reflected with very different component planes. The first and second classes are clustered as the same class as well as fourth and third.

### 5.3.3 SOM for feature clustering

The MLANS had shown relatively good results for some features (e.g. textures) but it is not generally robust technique. It would be useful to perform some experiments with the SOM on the same data as for MLANS and compare the results.

First of all, the smallest comparable SOM network is tested: as few neurons as possible and all features covered kept as the principal requirement. The following SOM configuration for the smallest SOM for feature clustering is used:

**Colour:**  $3 - by - 3$  SOM.

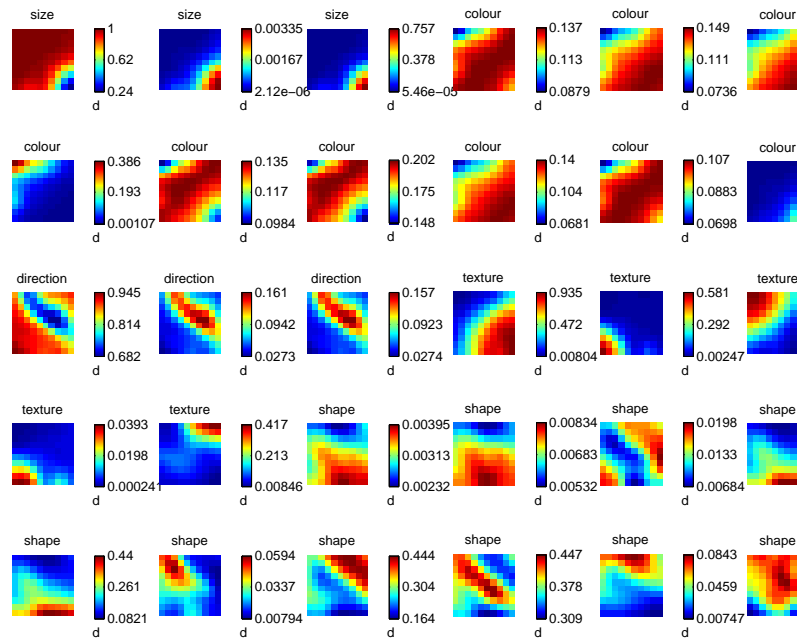


FIGURE 5.13: Component planes of SOM with 10-by-10 neurons (100 neurons) placed in rectangular lattice that perform clustering on 30 dimensional inputs taken from MLANS feature outputs.

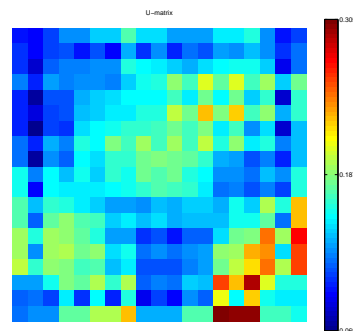


FIGURE 5.14: The U-matrix of SOM with 10-by-10 neurons (100 neurons) placed in rectangular lattice that perform clustering on 30 dimensional inputs taken from MLANS feature outputs.

**Directions:** 2 – by – 2 SOM.

**Shapes:** 4 – by – 4 SOM.

**Sizes:** 2 – by – 2 SOM.

**Textures:** 3 – by – 3 SOM.

The smallest SOM gives promising results for colours in comparison to MLANS because SOM distinguishes six of nine (see Fig. 5.21, left). The results for the textures and shapes are comparable with MLANS (see Fig. 5.23 and 5.22, right). On the other hand, the MLANS gives better results for directions where the two of three are recognized while SOM fails and considers all directions as almost identical (see Fig. 5.21, right)

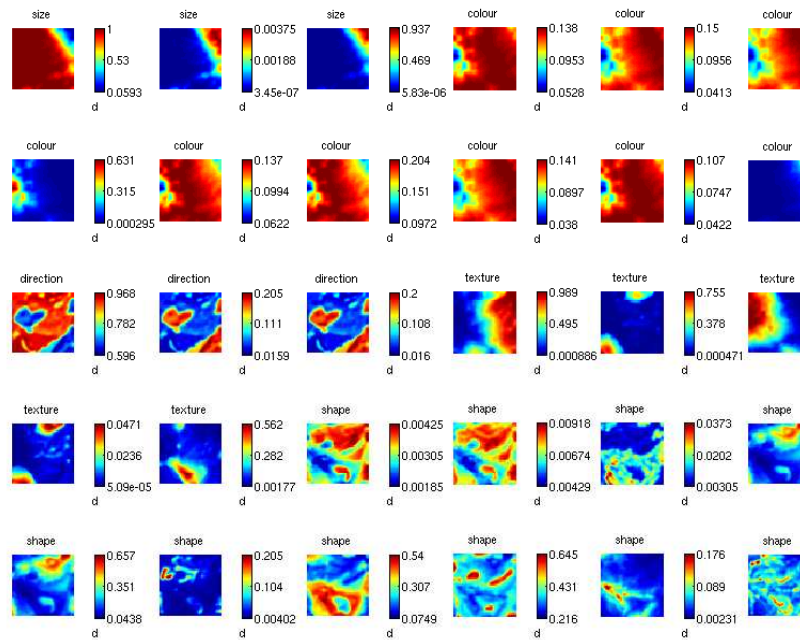


FIGURE 5.15: Component planes of SOM with 50-by-50 neurons (100 neurons) placed in rectangular lattice that perform clustering on 30 dimensional inputs taken from MLANS feature outputs.

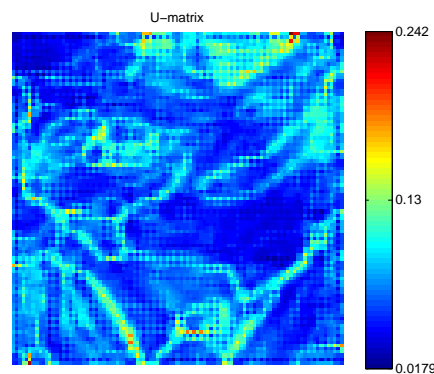


FIGURE 5.16: The U-matrix of SOM with 50-by-50 neurons (2500 neurons) placed in rectangular lattice that perform clustering on 30 dimensional inputs taken from MLANS feature outputs.

If the number of neurons is increased, the SOM gives much better results. For 10 – by – 10 SOM, learning phase gives at least one neuron whose weight is similar to one of the represented feature (see Fig. 5.19, upper row), for 50 – by – 50 SOM the results are significantly better than the others which can be easily seen from the U-matrices. The neurons form regions of weights, where each region is separated by ridge (separating neurons between patterns do not represent any feature), and plateaus are the separated concept classes.

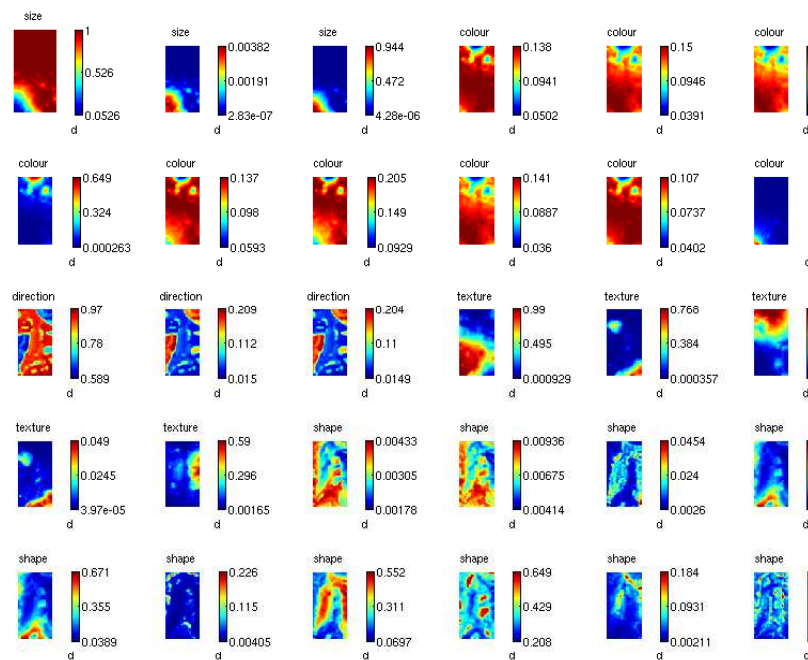


FIGURE 5.17: Component planes of SOM with 90-by-45 neurons (4050 neurons) placed in rectangular lattice that perform clustering on 30 dimensional inputs taken from MLANS feature outputs.

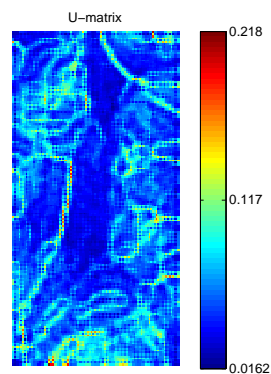


FIGURE 5.18: The U-matrix of SOM with 90-by-45 neurons (4050 neurons) placed in rectangular lattice that perform clustering on 30 dimensional inputs taken from MLANS feature outputs.

### 5.3.3.1 Comparison of SOM and MLANS

Main drawback of MLANS lies in the absence of any parameter that regulates complexity. Fixed complexity bounds the ability to tune and, alternatively, obtain better results. Once MLANS reaches a certain error, its results can not be improved by increasing  $K$ , which is only a complexity parameter. On the other hand, the SOM rapidly increases the quality of clustering with increasing complexity of the structure (number of neurons). Another problem of the MLANS is

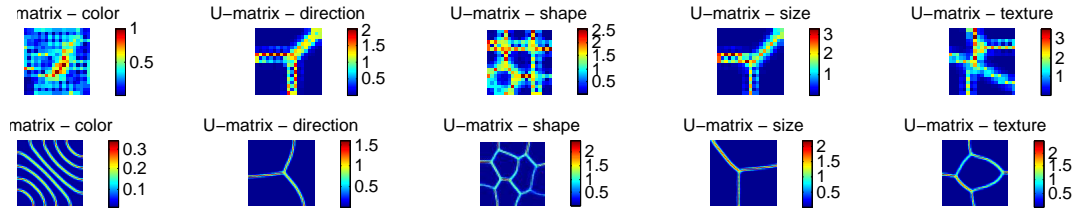


FIGURE 5.19: The U-matrices of 10 – by – 10 (upper row) and 50 – by – 50 SOM which are trained for the same task as the MLANS in the first phase. For each feature one SOM is trained.

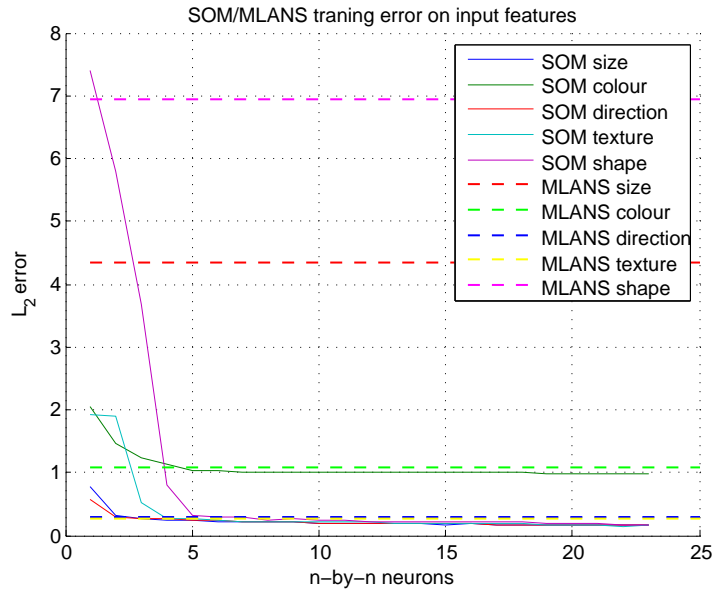


FIGURE 5.20: SOM and MLANS Euclidean distances from the most similar pattern (again measured by Euclidean metrics).

that results are strongly affected by initialisation. For example if  $\hat{\mu}^{(0)}$ -s are chosen inappropriately (for example near to each other) the algorithm is attracted by local optima, which represent some general concepts. The  $\hat{\Sigma}^{(0)}$  gives the alternative to complexity parameter, but there is no general procedure how to appropriately guess  $\hat{\Sigma}^{(0)}$  to get better results.

The SOM has better results as complexity increases (see Tab. 5.1 and Fig. 5.20).

The absence of complexity parameter, that can shift forward to the better results, restricts the MLANS in many domains where such flexibility is necessary. On the other hand, knowing the number of classes brings all necessary information, while SOM requires some tuning (neighbourhood, lattice, number of neurons)



	Size	Colour	Direction	Texture	Shape	$\Sigma$
MLANS	4.34	1.09	0.29	0.28	6.95	12.95
Smallest SOM	5.38	2.30	2.48	3.32	3.26	16.74
3 – <i>by</i> – 3 SOM	0.78	2.06	0.58	1.93	7.39	12.75
4 – <i>by</i> – 4 SOM	0.31	1.47	0.29	1.91	5.80	9.77
5 – <i>by</i> – 5 SOM	0.26	1.24	0.26	0.53	3.67	5.96
6 – <i>by</i> – 6 SOM	0.25	1.13	0.26	0.28	0.79	2.70
7 – <i>by</i> – 7 SOM	0.23	1.04	0.23	0.27	0.31	2.09
8 – <i>by</i> – 8 SOM	0.22	1.02	0.23	0.24	0.30	2.01
9 – <i>by</i> – 9 SOM	0.22	1.00	0.22	0.23	0.30	1.96
10 – <i>by</i> – 10 SOM	0.21	1.00	0.21	0.23	0.253	1.90
11 – <i>by</i> – 11 SOM	0.21	1.00	0.21	0.22	0.26	1.89
12 – <i>by</i> – 12 SOM	0.20	1.00	0.20	0.21	0.24	1.85
13 – <i>by</i> – 13 SOM	0.20	1.00	0.20	0.21	0.23	1.83
14 – <i>by</i> – 14 SOM	0.19	1.00	0.19	0.20	0.22	1.80
15 – <i>by</i> – 15 SOM	0.19	1.00	0.19	0.20	0.22	1.79
16 – <i>by</i> – 16 SOM	0.18	1.00	0.19	0.20	0.21	1.77
17 – <i>by</i> – 17 SOM	0.18	1.00	0.18	0.20	0.21	1.76
18 – <i>by</i> – 18 SOM	0.18	1.00	0.18	0.18	0.22	1.76
19 – <i>by</i> – 19 SOM	0.17	1.00	0.18	0.19	0.21	1.74
20 – <i>by</i> – 20 SOM	0.17	1.00	0.18	0.19	0.21	1.74
30 – <i>by</i> – 30 SOM	0.16	1.00	0.17	0.17	0.18	1.67
35 – <i>by</i> – 35 SOM	0.15	1.00	0.16	0.16	0.18	1.64
40 – <i>by</i> – 40 SOM	0.16	0.99	0.16	0.16	0.18	1.65
45 – <i>by</i> – 45 SOM	0.16	0.99	0.16	0.15	0.18	1.63
50 – <i>by</i> – 50 SOM	0.15	0.99	0.16	0.17	0.17	1.63
Mean	0.5936	1.1314	0.3093	0.4797	1.2851	3.7991
Variance	1.6867	0.1125	0.2113	0.5768	5.0414	18.7263

TABLE 5.1: SOM and MLANS Euclidean distances from the most similar pattern (again measured by Euclidean metrics).

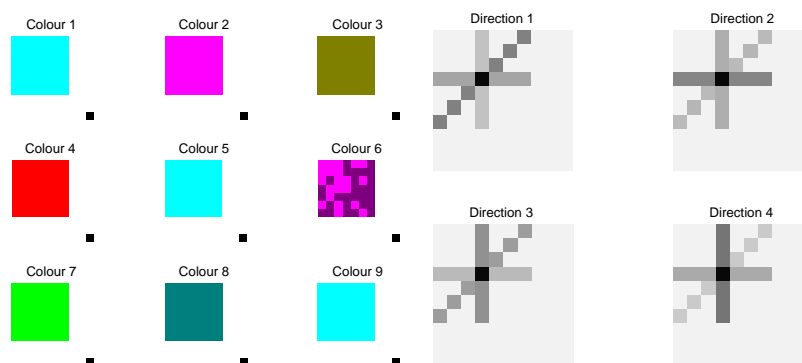


FIGURE 5.21: Left: All weight vectors of the SOM with 3 – *by* – 3 neurons in the rectangular lattice trained on the colour features; Right: All weight vectors of the SOM with 2 – *by* – 2 neurons in the rectangular lattice trained on the direction features.

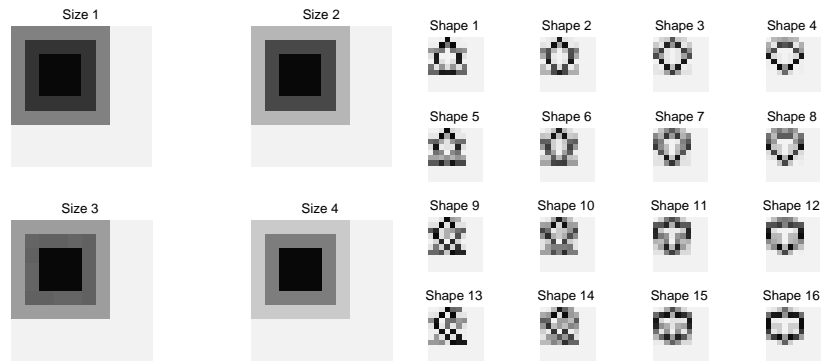


FIGURE 5.22: Left: All weight vectors of the SOM with  $2 - by - 2$  neurons in the rectangular lattice trained on the size features; Right: All weight vectors of the SOM with  $4 - by - 4$  neurons in the rectangular lattice trained on the shape features.

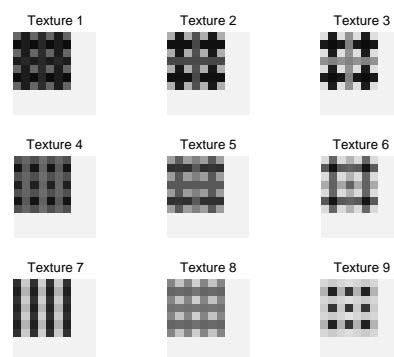


FIGURE 5.23: All weight vectors of the SOM with  $3 - by - 3$  neurons in the rectangular lattice trained on the texture features

## Chapter 6

# Region clustering of satellite images with Wishart distribution

In this chapter, a scheme for region clustering of images is shown. The images are divided into separate regions with constant width and height. Pixels of the images are transformed into feature vectors, and scattering matrix between the feature vectors is calculated [44]. The feature representation allows rapid reduction of storing requirements in comparison to the original image region because feature matrices are usually smaller. The feature matrices are used for further clustering by EM algorithm (MLANS) with a mixture of Wishart distributions. The Wishart distribution performs estimation of scattering matrices of normally distributed observations. The parameters of the Wishart distributions are estimated by the EM algorithm, and the estimated parameters are used for classification of regions.

### 6.1 Introduction

Images can be perceived as a large dense matrix with many elements. The large images can be intractable for tasks that require larger computational efforts. Especially satellite images are taken with very high resolution and some representations that reduce computational efforts based on dense representation are needed.

One of the methods is to split image into regions and to reduce the number of elements in the regions by computing scattering matrix of image features between several features. The simplest region scattering matrix of an image is to take pixel intensities of the image region and compute scattering matrix between the pixel depths. The method is very non-robust for clustering because it does not reflect important properties of the neighbourhood pixels like region changes. In [44] they propose the feature representation that utilizes some extra features that describe image characteristics of each pixel. The utilized features are pixel depth intensities and absolute value of the sum of the first and second gradient over horizontal and vertical axes (see Fig. 6.1). The extra features are used to calculate the scattering matrix of the region that is the reduced feature matrix.

There are two main contributions within the method in [44]. Firstly, the scattering matrix of several image features is computed inside in a region of interest and used as a region descriptor. Instead of the joint distribution of the image statistics, the scattering matrix is used as the element that represents entire region, so the dimensionality is smaller. Secondly it provides a fast method of calculating feature representation using the  $7 - by - 7$  (or  $5 - by - 5$  if the colour is grey) matrix and thus the computational cost of the classified region is independent of the size of the region.

In original paper [44] the feature matrices are classified by eigenvalues of the feature matrix, where the clustering on eigenvalues is performed by  $k$ -means algorithm. The  $k$ -means gives promising results, but the algorithm ignores important information - the eigenvectors which can improve clustering accuracy.

Here the more sophisticated is used; problem of clustering of scattering matrices can be solved by Wishart distribution  $\mathcal{W}(\mathbf{S}|\Sigma)$  that estimates covariance matrix of normal distribution from where the observations of scattering matrices (feature matrices) were originally drawn. The Theorem 2.4 says more rigorously how the Wishart distribution can be useful in this context.

## 6.2 Image representation

Suppose an image  $\mathbf{I}$  where each pixel is defined by three colour depths: red (R), green (G) and blue (B). The pixel in  $i$ -th row,  $j$ -th column and  $k$ -th colour depth level<sup>1</sup> is accessed by  $\mathbf{I}(i, j, k)$ . For simplicity  $k = 1$  corresponds to red,  $k = 2$  to green and  $k = 3$  is for blue. Horizontal gradient of  $i$ -th row,  $j$ -th column and  $k$ -th colour depth level is denoted as  $\frac{\partial \mathbf{I}(i, j, k)}{\partial x}$  and is computed as follows:

$$\frac{\partial \mathbf{I}(i, j, k)}{\partial x} = \mathbf{I}(i + 1, j, k) - \mathbf{I}(i - 1, j, k) \quad (6.1)$$

A vertical gradient of a pixel in  $i$ -th row,  $j$ -th column and  $k$ -th colour depth is calculated as follows:

$$\frac{\partial \mathbf{I}(i, j, k)}{\partial y} = \mathbf{I}(i, j + 1, k) - \mathbf{I}(i, j - 1, k) \quad (6.2)$$

The second order gradient is computed by the same method as the first gradient but instead of  $\mathbf{I}$ , the result from corresponding first gradient is used. Each pixel has three colour depths so gradients of depths  $D$  are summed to have single value, which defined as follows:

$$\begin{aligned} \frac{\partial \mathbf{I}(i, j)}{\partial x} &= \sum_{d=1}^D \frac{\partial \mathbf{I}(i, j, d)}{\partial x} \\ \frac{\partial \mathbf{I}(i, j)}{\partial y} &= \sum_{d=1}^D \frac{\partial \mathbf{I}(i, j, d)}{\partial y} \end{aligned} \quad (6.3)$$

The same as for the first gradient is performed with the second order gradients:

$$\begin{aligned} \frac{\partial^2 \mathbf{I}(i, j)}{\partial x^2} &= \sum_{d=1}^D \frac{\partial^2 \mathbf{I}(i, j, d)}{\partial x^2} = \sum_{d=1}^D \frac{\partial}{\partial x} \left( \frac{\partial \mathbf{I}(i, j, d)}{\partial x} \right) \\ \frac{\partial^2 \mathbf{I}(i, j)}{\partial y^2} &= \sum_{d=1}^D \frac{\partial^2 \mathbf{I}(i, j, d)}{\partial y^2} = \sum_{d=1}^D \frac{\partial}{\partial y} \left( \frac{\partial \mathbf{I}(i, j, d)}{\partial y} \right) \end{aligned} \quad (6.4)$$

---

<sup>1</sup>Images restricted to depth 3 with R-G-B colour depths, but, for example infra-red images have more than three dimensions or different colour map can be used.

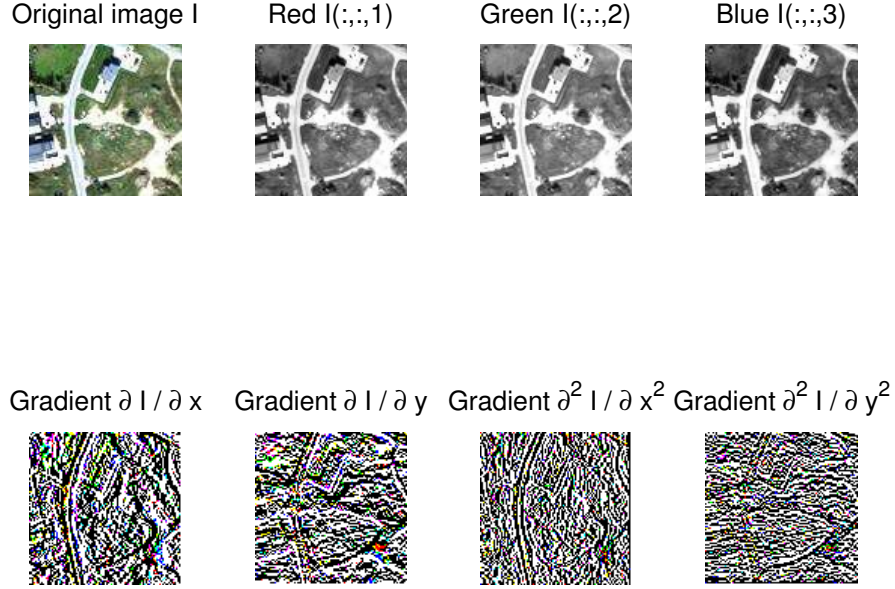


FIGURE 6.1: Visualisation of particular attributes taken as a features.

All above described features are used to form the feature vector  $\mathbf{f}(i, j)$  which is calculated for each pixel of the region. The feature vector is composed of following elements:

$$\mathbf{f}(i, j) = \left[ \mathbf{I}(i, j, 1), \mathbf{I}(i, j, 2), \mathbf{I}(i, j, 3), \left| \frac{\partial \mathbf{I}(i, j)}{\partial x} \right|, \left| \frac{\partial \mathbf{I}(i, j)}{\partial y} \right|, \left| \frac{\partial^2 \mathbf{I}(i, j)}{\partial x^2} \right|, \left| \frac{\partial^2 \mathbf{I}(i, j)}{\partial y^2} \right| \right]^T \quad (6.5)$$

The example of elements of a feature vector is shown in Fig. 6.1. The feature vectors are calculated for each pixel in processed region. The order of the feature vectors is irrelevant thus the pixels can be taken in arbitrary order. The scattering matrix is calculated from the well-known formula from statistics as follows:

$$\begin{aligned} \mathbf{S} &= \sum_{i=i_{start}}^{i_{end}} \sum_{j=j_{start}}^{j_{end}} (\mathbf{f}(i, j) - \bar{\mathbf{f}}) (\mathbf{f}(i, j) - \bar{\mathbf{f}})^T \\ \bar{\mathbf{f}} &= \frac{1}{N} \sum_{i=i_{start}}^{i_{end}} \sum_{j=j_{start}}^{j_{end}} \mathbf{f}(i, j) \end{aligned} \quad (6.6)$$

Where the  $i_{start}, i_{end}$  determine the first and last horizontal coordinate of the region respectively and  $j_{start}, j_{end}$  determine the first and last vertical coordinate of the region. The scattering matrix calculated by formula Eq. 6.6 where feature vectors are defined in Eq. 6.5. The feature vector has 7 elements thus scattering matrices are stored in  $7 \times 7$  arrays. The scattering matrices are used as inputs (observation) for the EM algorithm to adapt their parameter and automatically cluster the regions into  $K$  classes. The procedure of scattering matrix computation of an image region is formalized as the function  $\mathbf{S}_n = \text{Cov}(\mathbf{I}, \text{reg}_n)$  shown in Alg. 4.

---

**Algorithm 4:** The  $\mathbf{S}_n = \text{Cov}(\mathbf{I}, \mathbf{r}_n)$  method that performs transformation of single region into feature matrix  $\mathbf{S}_n$

---

**Data:** An image  $\mathbf{I}$  with rectangularly determined coordinates of a region

$$\text{reg}_r = \left[ \underbrace{i_{start}, i_{end}}_x, \underbrace{j_{start}, j_{end}}_y \right]$$

**Result:** Scattering matrix  $\mathbf{S}_r$  of the specified region.

// calculate average mean vector of features  $\bar{\mathbf{f}} = [0, \dots, 0]$ ;

```

for  $i \leftarrow i_{start}$  to  $i_{end}$  do
  for  $j \leftarrow j_{start}$  to  $j_{end}$  do
    //the feature vector  $\mathbf{f}(i, j)$  from Eq. 6.5
     $\bar{\mathbf{f}} = \bar{\mathbf{f}} + \mathbf{f}(i, j)$ ;

```

$\mathbf{S}_r = \mathbf{I}$ ;

```

for  $i \leftarrow i_{start}$  to  $i_{end}$  do
  for  $j \leftarrow j_{start}$  to  $j_{end}$  do
     $\mathbf{S}_r = \mathbf{S}_r + (\mathbf{f}(i, j) - \bar{\mathbf{f}})(\mathbf{f}(i, j) - \bar{\mathbf{f}})^T$ 

```

---

### 6.3 Classification of image regions

The Wishart distribution has the useful property for the scattering matrices as stated in Theorem 2.4. The Theorem 2.4 enables the Wishart distribution to be used as measurement of similarity between various classes of scattering matrices. The parameter adaptation is performed by the EM algorithm<sup>2</sup> (widely analysed in Chapter 3) that gives general framework for density estimation of mixtures. After the parameters of the densities in the mixture are adapted, the region is labelled with a density  $k$  which has the highest probability  $p(y_n = k | \mathbf{S}_n, \Theta)$ . The parameter that specifies degrees of freedom  $N$  in Eq. 2.47 is not needed (otherwise the EM will try to estimate the number of pixels from which the scattering matrix is calculated) thus simplified version of Wishart distribution defined in Eq. 2.48 is used instead.

The classification of a datum  $\mathbf{S}_n$  into one of the  $K$  regions for the parameters  $\Theta = \{\Sigma_1, \dots, \Sigma_K\}$  of the mixture of  $K$  Wishart PDFs is performed by function  $\text{Classify}(\mathbf{S})$  whose outcome is defined as follows:

$$\text{Classify}(\mathbf{S}_n, \hat{\Theta}) = \arg \max_{k \in \{1, \dots, K\}} p(y = k | \mathbf{S}_n, \hat{\Theta}) = \arg \max_{k \in \{1, \dots, K\}} \frac{\hat{\pi}_k \mathcal{W}(\mathbf{S}_n | \hat{\Sigma}_k)}{\sum_{i=1}^K \hat{\pi}_i \mathcal{W}(\mathbf{S}_n | \hat{\Sigma}_i)} \quad (6.7)$$

### 6.4 Estimation of the parameters

The estimation is performed with Eq. 3.36 shown in Chapter 3. The entire procedure of parameter estimation is described in Alg. 6.

---

<sup>2</sup>The reason why the EM is referred instead of NMF is because the notion of EM is used.

---

**Algorithm 5:** The  $\mathbf{S}_n = \text{FeatureExtraction}(\mathbf{I}, w_r, h_r)$  method that extracts feature (scattering) matrices from the input image

---

**Data:** The input image  $\mathbf{I}$  (with width  $w$  and height  $h$ ) and width  $w_r$  and height  $h_r$  of each region.

**Result:** Scattering matrices  $\{\mathbf{S}_1, \dots, \mathbf{S}_N\}$  all regions in image.

```

{ $\mathbf{S}_1 \leftarrow \mathbf{I}, \dots, \mathbf{S}_N \leftarrow \mathbf{I}$ };
 $k \leftarrow 1$ ;
 $i \leftarrow w_r$ ;
 $j \leftarrow h_r$ ;
while  $i \leq w$  do
    while  $j \leq h$  do
         $\mathbf{r}_k = [i - w_r, j - h_r, w_r, h_r]$ ;
         $\mathbf{S}_k = \text{Cov}(\mathbf{I}, \mathbf{r}_k)$ ;
         $i \leftarrow i + w_r$ ;
         $j \leftarrow j + h_r$ ;
         $k \leftarrow k + 1$ ;
     $i \leftarrow i + w_h$ ;

```

---



---

**Algorithm 6:** Parameter estimation of mixture of Wishart distribution from Eq. 3.1 an image  $\mathbf{I}$

---

**input** : An image  $\mathbf{I}$  whose regions are clustered with specified image width  $w$  and height  $h$  and dimensions of clustered regions  $w_r$  for width and  $h_r$  for height.

**output:** Set of parameters of the mixture in Eq. 3.1.

```

{ $\mathbf{S}_1, \dots, \mathbf{S}_N\} \leftarrow \text{FeatureExtraction}(\mathbf{I}, w_h, h_h)$ ;
 $i \leftarrow 0$ ; // iterations
 $\hat{\Theta}^{(0)} \leftarrow \{\hat{\Sigma}_1^{(0)}, \dots, \hat{\Sigma}_K^{(0)}\}$ ;
while  $i \leq \text{MAX\_ITER}$  do
    for  $k \leftarrow 1$  to  $K$  do
        // E step
         $p(y_n = k | \mathbf{S}_n, \hat{\Theta}^{(i)}) \leftarrow \text{Eq. 3.7}$ ;
        // M step.
         $\hat{\Sigma}_k^{(i+1)} \leftarrow \text{Eq. 3.36}$ ;
         $\hat{\pi}_k^{(i+1)} \leftarrow \text{Eq. 3.12}$ ;
     $i \leftarrow i + 1$ ;

```

---

## 6.5 Experiments

Three experiments with satellite images are performed. The number of classes is 5 ( $K = 5$ ).

The first image is taken from [45] which is the satellite image of Viljandi located in Estonia. Input image resolution is  $2500 - by - 2380$  pixels and region sizes are  $50 - by - 50$ , thus 2880 feature matrices are clustered. The result is shown in Fig. 6.2.

The second image is taken from [46] which is the satellite image of Amsterdam the capital of Netherlands. Input image resolution is  $2400 - by - 2400$  pixels and region sizes are  $50 - by - 50$ , thus 2209 feature matrices are clustered. The result is shown in Fig. 6.3.

The third image is taken from [47] which is the satellite image of Tadco Farms located in Saudi Arabia. Input image resolution is  $3873 \times 2112$  pixels and region sizes are  $100 \times 100$  thus 608 feature matrices are clustered. The result is shown in Fig. 6.4.

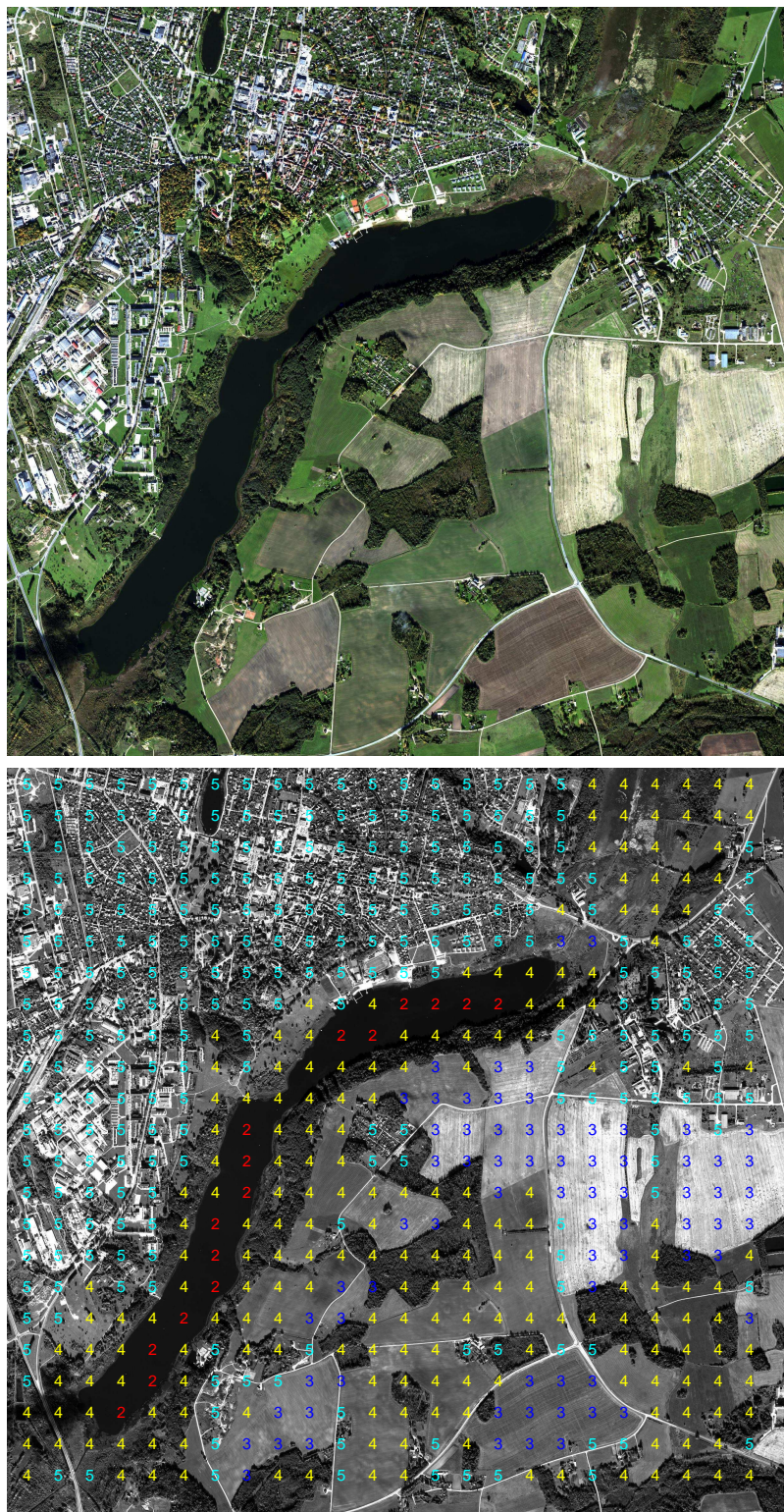


FIGURE 6.2: The clustering results (bottom) of the input image (upper) located in the Viljandi, Estonia [45].



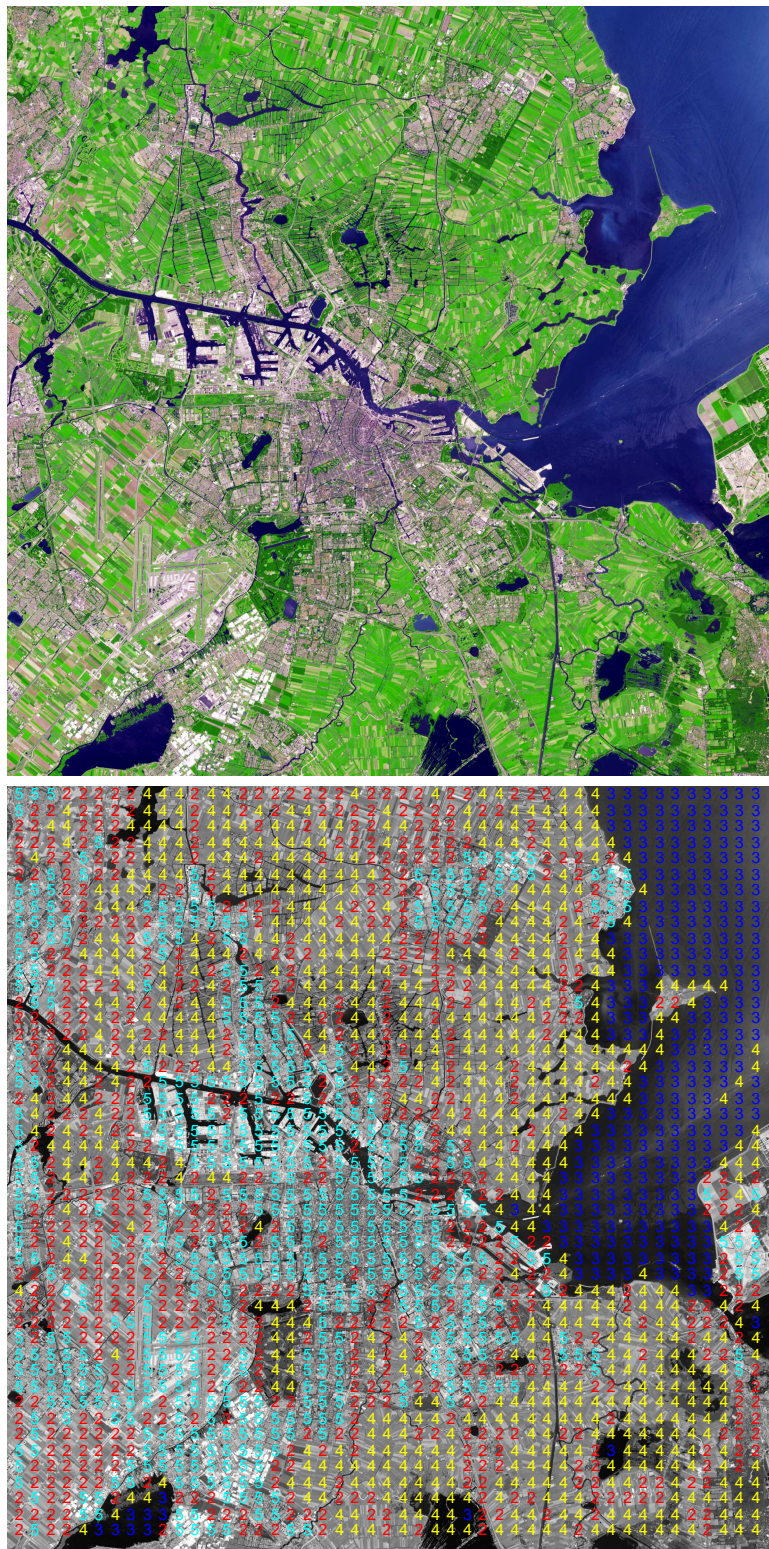


FIGURE 6.3: The clustering results (bottom) of the input image (upper) located in the Amsterdam, Netherlands [46].

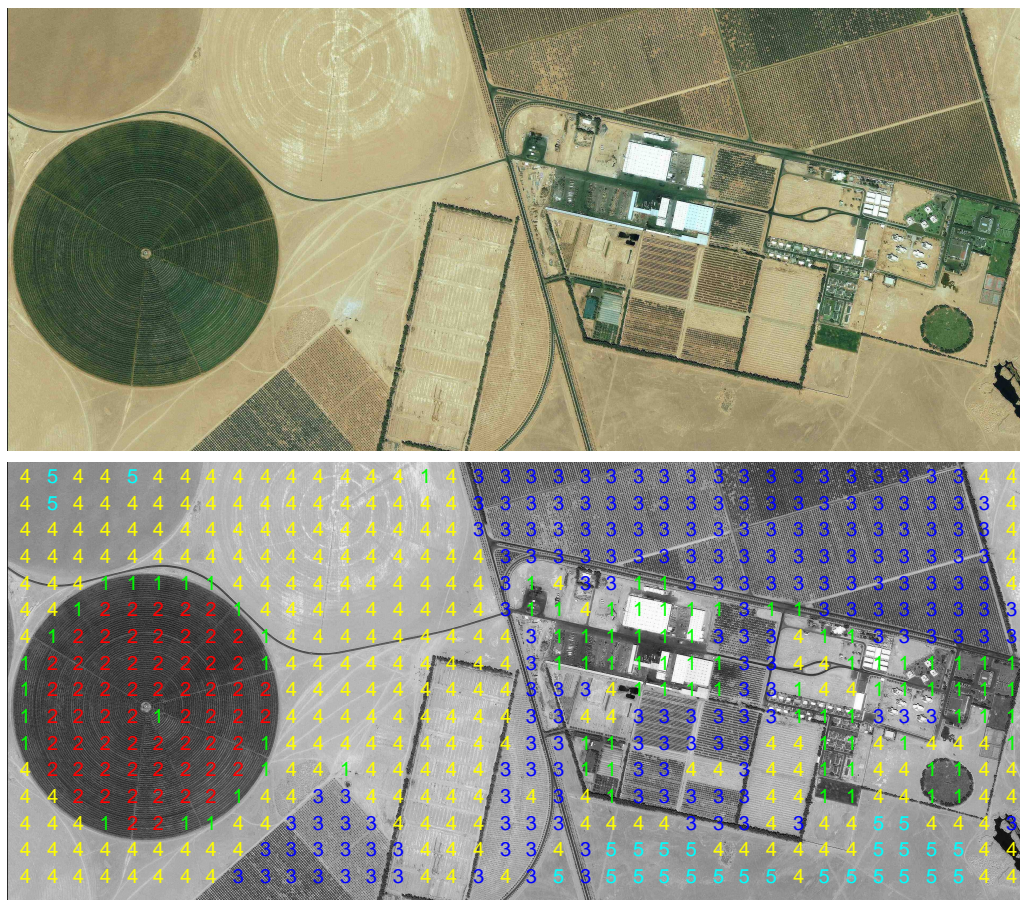


FIGURE 6.4: The clustering results (bottom) of the input image (upper) located in the Agricultural land of the Tadco company in Saudi Arabia [47].

## Chapter 7

# Conclusion

The thesis presents the Neural Modeling Fields (NMF) as a parametric model creator for diverse distribution functions and approaches. Since the equivalence of NMF with the EM (Expectation Maximization) algorithm was shown the NMF can be used for the same problems as the EM.

The first contribution of this thesis is a summary of frequently used distribution functions in statistics and machine learning. The summary contains formulations that can be used for mixture density estimation. For some of them, own equations for adaptive step of the NMF and EM algorithm are derived. The derived distributions are the Exponential, the Gamma, von Mises-Fisher, Log-normal, Normal, Wishart and Dirichlet distributions. For the Exponential, Log-normal and gradient ascent for the Dirichlet distributions formulations are derived.

The thesis contains detailed analysis of the EM algorithm including derivation of the general equations which is required for proof of the identity of the NMF and the EM algorithm. The proof itself is straightforward because it proves only identity between terms of both approaches.

The power of the NMF are simulated with three experiments. The first is Jordan's HME (Hierarchical Mixture of Experts) which is supervised network structure whose parameter adaptation is based on the maximum likelihood principle. The HME is shown including the derivation of the adaptive equations. Further the HME are slightly extended for polynomial regression (original paper supposes only linear), this extension brings more better results. The HME is compared on two approximation problems with neural networks. The HME has shown as very fast approach whose main drawback are numerical instabilities. The second experiment is data clustering and classification based on Feature Integration Theory where the NMF and SOM (Self Organizing Map) are compared. For the second problem, the SOM exhibits better results with increasing structure complexity. The third experiment is region clustering of satellite images calculated from feature matrices where the clustering classification is performed as a maximum likelihood task where the parametric model is the mixture of Wishart distributions. The experiment is based on a combination of the feature extraction method and evidence gained from distribution function analysis. The results from the third experiment were presented on Rektory's contest 2013.

---

Implementation of NMF for the analysed distribution functions, the HME and image classifier based on Wishart distribution in MATLAB are also contained in this thesis. The code is intended to be published as open-source toolbox because there is yet no such implementation.

The thesis covers wide range of problems. Some of problems in the thesis can be yet further developed in the future. The NMF (EM) is one of many methods in parametric model creation, which can be modified to heterogeneous tasks. The topic of this thesis covers only small fraction of what can be written about the parametric models creation with NMF. This thesis demonstrates relation between NMF and parametric models with chosen applications. The thesis contains many novelty results, mostly for the distribution functions, entire chapter about region clustering in the last chapter and some slight modifications for HME.

# Appendix A

## Some special functions

### A.1 Approximation error of the Digamma function $\psi(x)$

In Chapters 2 and 3 derivative of  $\ln \Gamma(x)$  function is needed. The value can be replaced by the Digamma function  $\psi(x)$ . The digamma function has no closed form, thus some approximations are needed. The approximation Eq. 3.21 whose error is shown in Fig. A.1.

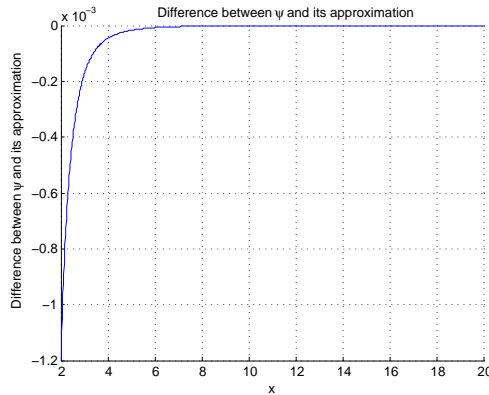


FIGURE A.1: Digamma function approximation Eq. 3.21 error.

### A.2 Hessian matrix

In Chapter 2 Hessian matrix is used. Elements of the Hessian matrix is formulated as follows:

$$\mathbf{H} = \nabla^2 f(\mathbf{x}) = \begin{pmatrix} \frac{\partial^2 f(\mathbf{x})}{\partial x_1^2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_D} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f(\mathbf{x})}{\partial x_D \partial x_1} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_D^2} \end{pmatrix} \quad (\text{A.1})$$

## Appendix B

# Contents of attached CD

The DVD contains the text of the thesis including source, MATLAB implementation of NMF for all analysed probability density functions, MATLAB implementation for two layered HME, the dataset that was used in Chapter 5 for Feature Integration Theory and implementation of region classification analysed in Chapter 6.

CD	
o thesis	
o source	
o pdf	
code	
o framework	(B.1)
o HME	
o FIT	
o region-classification	
data	
o FIT	

The **framework** contains some examples. The example files distinguished by the suffix **example.m**. The **region-classification** contains example code as well as some sample images.

### B.1 Instructions to run the framework

Some special paths must be added to run the **framework**. It can be added manually via File→Set path→Add folders or via MATLAB built-in commands:

```
addpath(' [path_to_framework]/functions/');  
addpath(' [path_to_framework]/functions/distributions/');  
addpath(' [path_to_framework]/functions/util/')
```

# Bibliography

- [1] Shane Legg and Marcus Hutter. A collection of definitions of intelligence. In *Proceedings of the 2007 conference on Advances in Artificial General Intelligence: Concepts, Architectures and Algorithms: Proceedings of the AGI Workshop 2006*, pages 17–24, Amsterdam, The Netherlands, The Netherlands, 2007. IOS Press. ISBN 978-1-58603-758-1. URL <http://dl.acm.org/citation.cfm?id=1565455.1565458>.
- [2] C. Soanes and S. Hawker. *Compact Oxford English Dictionary of Current English*. Oxford University Press, Incorporated, 2005. ISBN 9780198610229. URL <http://books.google.cz/books?id=d7B1QgAACAAJ>.
- [3] Alan Isaacs, John Daintith, and Elizabeth Martin. *A dictionary of science*. Oxford University Press New York, New York, USA, 1999.
- [4] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (2nd Edition)*. Prentice Hall, December 2002. ISBN 0137903952. URL <http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&path=ASIN/0137903952>.
- [5] Eugene Charniak. *Introduction to artificial intelligence*. Pearson Education India, 1985.
- [6] L.I. Perlovsky. *Neural Networks and Intellect: Using Model Based Concepts*. Neural Networks and Intellect: Using Model-based Concepts. OXFORD University Press, 2001. ISBN 9780195111620. URL <http://books.google.at/books?id=fMchKnr57rIC>.
- [7] Omid Omidvar and David L Elliott. *Neural systems for control*. Elsevier, 1997.
- [8] Simon S Haykin, Simon S Haykin, Simon S Haykin, and Simon S Haykin. *Neural networks and learning machines*, volume 3. Prentice Hall New York, 2009.
- [9] Raúl Rojas. *Neural Networks: A Systematic Introduction*. Springer, 1996.
- [10] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.
- [11] ZI Botev, JF Grotowski, and DP Kroese. Kernel density estimation via diffusion. *The Annals of Statistics*, 38(5):2916–2957, 2010.
- [12] Christopher M Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995.

- 
- [13] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 0387310738.
- [14] Robert Neumayer, Rudolf Mayer, G Polzlbauer, and Andreas Rauber. The metro visualisation of component planes for self-organising maps. In *Neural Networks, 2007. IJCNN 2007. International Joint Conference on*, pages 2788–2793. IEEE, 2007.
- [15] Bela A Frigyik, Amol Kapila, and Maya R Gupta. Introduction to the dirichlet distribution and related processes. *Department of Electrical Engineering, University of Washington, UWEETR-2010-0006*, 2010.
- [16] Mark Steyvers and Tom Griffiths. Probabilistic topic models. *Handbook of latent semantic analysis*, 427(7):424–440, 2007.
- [17] Kevin Quinn. The newton raphson algorithm for function optimization. *University of Washington, Seattle*, 2001.
- [18] Albert W Marshall and Ingram Olkin. A multivariate exponential distribution. *Journal of the American Statistical Association*, 62(317):30–44, 1967.
- [19] Narayanaswamy Balakrishnan and Asit P Basu. *The exponential distribution: theory, methods and applications*. CRC press, 1995.
- [20] Hafzullah Aksoy. Use of gamma distribution in hydrological analysis. *Turkish Journal of Engineering and Environmental Sciences*, 24(6):419–428, 2000.
- [21] Thomas P Minka. Estimating a gamma distribution. *unpublished paper (<http://research.microsoft.com/en-us/um/people/minka/papers/minka-gamma.pdf>)*, 2002.
- [22] Arindam Banerjee, Inderjit S Dhillon, Joydeep Ghosh, and Suvrit Sra. Clustering on the unit hypersphere using von mises-fisher distributions. In *Journal of Machine Learning Research*, pages 1345–1382, 2005.
- [23] Abdul Hasant, Oliver Alata, and Alain Tremeau. Hierarchical 3-d von mises-fisher mixture model, 2013.
- [24] Thomas P. Minka. Old and new matrix algebra useful for statistics. Technical report, 2001.
- [25] Christophe Saint-Jean and Frank Nielsen. A new implementation of k-mle for mixture modeling of wishart distributions. In *Geometric Science of Information*, pages 249–256. Springer, 2013.
- [26] T Tokuda, B Goodrich, I Van Mechelen, A Gelman, and F Tuerlinckx. Visualizing distributions of covariance matrices. Technical report, Technical report, University of Leuven, Belgium and Columbia University, USA. URL <http://www.stat.columbia.edu/~gelman/research/unpublished/Visualization.pdf>.(Cited on pages 114, 116, 117 and 119.), 2011.
- [27] Exponential family of distributions, 2004. URL <http://www.cs.columbia.edu/~jebara/4771/tutorials/lecture12.pdf>.



- [28] Maximum likelihood in exponential families, 2004. URL <http://www.stats.ox.ac.uk/~steffen/teaching/bs2siMT04/si6bw.pdf>.
- [29] L. Perlovsky, R. Deming, and R. Ilin. *Emotional Cognitive Neural Algorithms with Engineering Applications: Dynamic Logic: From Vague to Crisp*. Studies in Computational Intelligence. Springer, 2011. ISBN 9783642228292. URL <http://books.google.at/books?id=CZDeKMMakucC>.
- [30] Mário AT Figueiredo. Lecture notes on the em algorithm. URL: <http://www.stat.duke.edu/courses/Spring06/sta376/Support/EM/EM.Mixtures.Figueiredo>, 2004.
- [31] Jeff A Bilmes et al. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. *International Computer Science Institute*, 4(510):126, 1998.
- [32] Jalal Almhana, Zikuan Liu, Vartan Choulakian, and Robert McGorman. A recursive algorithm for gamma mixture models. In *Communications, 2006. ICC'06. IEEE International Conference on*, volume 1, pages 197–202. IEEE, 2006.
- [33] Leonid I Perlovsky. Neural networks, fuzzy models and dynamic logic. In *Aspects of Automatic Text Analysis*, pages 363–386. Springer, 2007.
- [34] Philippe Flajolet and Robert Sedgewick. *Analytic combinatorics*. cambridge University press, 2009.
- [35] E Rodney Canfield and Carl Pomerance. On the problem of uniqueness for the maximum stirling number (s) of the second kind. *INTEGERS: Electronic Journal of Combinatorial Number Theory*, 2(A01):2, 2002.
- [36] Leonid I Perlovsky and Margaret M McManus. Maximum likelihood neural networks for sensor fusion and adaptive classification. *Neural Networks*, 4(1):89–102, 1991.
- [37] Oxford dictionaries, 2013. URL <http://www.oxforddictionaries.com/definition/english/instinct?q=instinct>.
- [38] Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.
- [39] Michael I Jordan and Lei Xu. Convergence results for the em approach to mixtures of experts architectures. *Neural networks*, 8(9):1409–1431, 1995.
- [40] Walter Gander. Change of basis in polynomial interpolation. *Numerical linear algebra with applications*, 12(8):769–778, 2005.
- [41] Ian Nabney. Netlab toolbox, 2000. URL <http://homepages.cae.wisc.edu/~ece539/software/netlab/intro.htm>.
- [42] Anne M Treisman and Garry Gelade. A feature-integration theory of attention. *Cognitive psychology*, 12(1):97–136, 1980.
- [43] Esa Alhoniemi, Johan Himberg, Juha Parhankangas, and Juha Vesanto. Som toolbox, 2005. URL <http://www.cis.hut.fi/projects/somtoolbox/>.

- 
- [44] Oncel Tuzel, Fatih Porikli, and Peter Meer. Region covariance: A fast descriptor for detection and classification. In *Computer Vision–ECCV 2006*, pages 589–600. Springer, 2006.
- [45] Viljandi satellite image, 2013. URL <http://www.regio.ee/?op=body&id=239>.
- [46] Amsterdam satellite image, 2013. URL <http://visibleearth.nasa.gov/view.php?id=81689>.
- [47] Tadco farms, saudi arabia satellite image, 2013. URL <http://www.satimagingcorp.com/galleryimages/ikonos-80cm-image-saudi-tadco.jpg>.