

Solvers and their implementations for machine learning problems and applications

Vývoj řešičů a jejich implementace pro úlohy strojového učení a aplikace

Ing. Marek Pecha

PhD Thesis

Supervisor: doc. Ing. David Horák, Ph.D.

Supervisor specialist: Dr. Richard Tran Mills

Ostrava, 2024

Abstrakt a přínos práce

Strojové učení (z anglického Machine Learning) je souhrn statistických technik a optimalizačních algoritmů, které se používají k vytvoření modelů na základě trénovacích (vzorových) dat bez explicitního programování instrukcí v rozhodovacím algoritmu. Cílem je dosáhnout generalizace modelu z vlastností (rysů) trénovacích dat pro co nejpřesnější predikci na vzorcích, které nebyly použity v rámci trénovacího procesu. Strojové učení se jako takové považuje za podoblast umělé inteligence (Artificial Intelligence).

Hlavní náplní této práce je studium použitelnosti klasických modelů strojového učení pro řešení komplexních a datově náročných aplikací, volba vhodných optimalizačních algoritmů a jejich adaptace pro paralelní trénování modelů na superpočítačových systémech, v neposlední řadě také navržení workflow zahrnující efektivní analýzu, fúzování a transformace velkých „big“ dat. Je věnována také pozornost vhodnému uložení dat např. do formátu HDF5 pro jejich efektivní paralelní načítání na superpočítačových systémech.

Významná část práce je věnována strojovému učení s učitelem (supervised learning), konkrétně klasifikačním modelům typu Support Vector Machines (SVM), jejich přizpůsobení pro sémantickou segmentaci multispektrálních satelitních snímků v čase a následné použití k lokalizaci výskytu lesních požárů na Aljašce v horizontu jednoho roku. Problematika lesních požárů byla řešena ve spolupráci se dvěma prestižními výzkumnými pracovišti v USA, konkrétně s národními laboratořemi Argonne a Oak Ridge. Pro vlastní natrénování segmentačních modelů byl použit open-source nástroj PermonSVM, jehož implementace je nedílnou součástí této disertační práce. Tento nástroj také podporuje trénování pravděpodobnostních modelů s použitím techniky Plattova škálování v kombinaci s modely typu SVM. Pro řešení optimalizační úlohy v rámci trénovacího procesu SVM modelů byly použity a adaptovány řešiče MPRGP, SMALXE a jejich varianty, které jsou implementované v softwarovém balíku PermonQP. Tyto řešiče byly vyvinuty a optimalizovány pro úlohy kvadratického programování a jsou dále rozvíjené skupinou profesora Dostála na Katedře aplikované matematiky (VŠB – Technická univerzita Ostrava) a na Ústavu geoniky AV ČR.

Druhá část této práce je zaměřena na strojové učení bez učitele (unsupervised learning). Konkrétně je zde provedena rešerše metod vektorové kvantifikace založených na algoritmech Lloydova typu a metod spektrálního shlukování. Dále je představena paralelní implementace těchto metod v programovacím jazyce C++ a statistický přístup, který je založený na Bartlettově testu homogenity variancí pro odhad násobnosti nulových vlastních čísel Laplaceovy matice. Počet nulových čísel této matice odpovídá počtu komponent souvislosti podobnostního grafu, které mohou představovat např. objekty na obrazové scéně. V praktické části jsou představeny dvě aplikace. První z nich se zaměřuje na detekci křehkých a houževnatých lomů na vzorku ocele API 5L X-70 pomocí technik vektorové kvantifikace. Ve druhé aplikaci je ukázáno použití spektrálního shlukování pro segmentaci obrazu bez anotovaných dat.

Hlavní přínos práce z pohledu autora spočívá v propojení několika disciplín zahrnujících paralelní programování pro distribuované trénování modelů, zpracování velkých dat a obrazu, strojové učení, statistiku, optimalizace a geoinformatiku, dále pak naprogramování softwarového balíku PermonSVM v programovacím jazyce C a vylepšení workflow pro lokalizaci lesních požárů na Aljašce pro použití klasických modelů strojového učení typu SVM. Přínosy práce zahrnují také rešerši teoretického základu SVM klasifikátorů a adaptace algoritmů SMALXE a MRPGP v rámci trénovacího procesu pro klasifikátory výše uvedeného typu. V části práce, která je zaměřená na strojové učení bez učitele, pak mezi přínosy práce lze zahrnout paralelní implementaci algoritmů Lloydova typu, drobnou modifikaci statistické metody pro odhad počtu komponent podobnostního grafu (odpovídá počtu nulových čísel Laplaceovy matice), a dále pak dvě aplikace z reálného světa, konkrétně detekce křehkých a houževnatých lomů a segmentaci obrazu.

Klíčová slova

Bartlettův test homogenity variancí, dualita, Google Earth Engine, kvadratické programování, křehké a houževnaté lomy, lokalizace lesních požárů na Aljašce, MTBS, MRPGP, paralelní trénování modelů, sémantická segmentace obrazu, SMALXE, strojové učení bez učitele, strojové učení s učitelem, Support Vector Machines, spektrální shlukování, umělá inteligence, vektorová kvantifikace, vzdálený průzkum Země, zpracování velkých dat

Abstract and Contributions

Machine learning is a set of statistical techniques and optimization algorithms used to create models based on training data without explicitly programming instructions in the decision algorithm. The goal is to achieve the generalization ability of a model from training data properties (features) for the most accurate prediction on unseen samples. Note that machine learning is considered as a subfield of artificial intelligence.

This work focuses mainly on studying the applicability of the classical machine learning models for solving complex and data-intensive applications, choosing appropriate optimization algorithms and their adaptation for parallel training of models on supercomputer systems, and last but not least on designing and programming a workflow involving efficient analysis, fusion and transformation for big data. Attention is also paid to storing the data in HDF5 file format for efficient parallel I/O operations on supercomputer systems.

A significant part of the work is devoted to supervised machine learning, specifically Support Vector Machines (SVM) classification models, their adaptation for semantic segmentation of multispectral-temporal satellite images, and subsequent use for wildfire localization in Alaska in a time horizon of one year. This application was addressed in collaboration with two world-leading research institutes in the USA, namely Argonne and Oak Ridge National Laboratories. The open-source tool called PermonSVM was used to train such segmentation models; implementation of this software is another integral part of this doctoral thesis. PermonSVM also supports training probabilistic models using Platt's scaling combined with models of the SVM type. The solvers MPRGP, SMALXE and their variants implemented in the PermonQP software package were used and adapted to solve an underlying optimization problem associated with training the models. These solvers have been developed and optimized for quadratic programming problems. They are further developed by Professor Dostal's group at the Department of Applied Mathematics (VSB – Technical University of Ostrava) and the Institute of Geonics of the Czech Academy of Sciences.

The second part of this thesis focuses on unsupervised machine learning. Specifically, a review of methods related to vector quantification based on Lloyd-type algorithms and spectral clustering is conducted. Furthermore, a parallel implementation of the vector quantification methods in the C++ programming language and a statistical approach based on the Bartlett's test of homogeneity of variances for estimating the multiplicity of zero eigenvalues related to the Laplace matrix are presented. The multiplicity of zero eigenvalues of this matrix corresponds to the number of components of the similarity graph (equals number of zero eigenvalues associated with graph Laplacian matrix); these components could represent objects in an image scene for example. In the practical part, two applications are introduced. The first focuses on detecting brittle and ductile fractures on steel sample (API 5L X-70) using vector quantization techniques. The second application shows employing spectral clustering

for image segmentation without annotated data.

The main contribution of this thesis (from the perspective of the author) consists in connecting several fields, including parallel programming for distributed model training, big data and image processing, machine learning, statistics, optimization, and geoinformatics, as well as the programming of the PermonSVM software package in the C programming language, and the improvement of the workflow for the wildfires localization in Alaska used for preparing data for classical machine learning models, e.g. SVM. Contributions of the work also include a compilation of the theoretical background related to SVM classifiers and the adaptation of the SMALXE and MRPGP algorithms in the training these classifiers. Further, contributions in part related to unsupervised learning include a parallel implementation of the Lloyd-type algorithms, a slight modification of a statistical method for estimating the number of components of a similarity graph, and two real-world applications, specifically brittle and ductile fracture detection and image segmentation.

Keywords

artificial intelligence, Bartlett's test of homogeneity variances, big data analysis, clustering, duality, Google Earth Engine, quadratic programming, brittle and ductile fractures, MRPGP, MTBS, parallel model training, semantic segmentation, SMALXE, unsupervised learning, supervised learning, Support Vector Machines, spectral clustering, vector quantification, volumetric images, remote sensing, wildfires localization in Alaska

Acknowledgement

I would like to express a heartfelt gratitude to my supervisor, doc. Ing. David Horák, Ph.D., for a time devoted to consultations mainly in the field of quadratic programming, intellectual freedom in other fields introduced in this thesis, his friendly approach, and unwavering support. Specially, I am deeply thankful to my co-supervisor, Dr. Richard Trans Mills, for motivating discussions in the field of machine learning, helping me to train models on GPUs, providing a challenging application related to wildfire localization in Alaska, and introducing me to remote sensing team from Oak Ridge National Laboratory and PETSc team at Argonne National Laboratory. I am deeply grateful to Mgr. Stanislav Systala, Ph.D., for his time and lots of valuable advices and comments. Big thank goes to my family, friends, and colleagues from PERMON team, mainly Jakub and Václav, and colleagues Jana, Ivo, Bohdan and Vít from Ostravian-Opavian seismic team, who supported me during my doctoral studies.

Last but not least, I would like to thank Prof. Zdeněk Dostál, whose advice helped to improve the formal aspects of this text, and other valuable comments on how to speed up training the machine learning models. Other big thanks belong to Dr. Zachary Langford and Dr. Jitedra Kumar for their excellent cooperation on the application related to wildfire localization, Prof. Svetozar Margenov and Dr. Stanislav Harizanov, who helped me with 2-phase volumetric image segmentation. The last big thank belongs to Prof. Radim Blaheta, who introduced me to the Institute of Geonics in Ostrava.

The research presented in this thesis was supported by HPC-EUROPA3 the H2020 Programme (INFRAIA-2016-1-730897), the grant for Support for Science and Research in the Moravia-Silesia Region 2017 (RRC/10/2017), and the Strategy AV21 research programme VP30 Dynamic Planet Earth.

Contents

Acronyms	xi
Symbols	xv
List of Figures	xvii
List of Tables	xxi
1 Introduction	1
1.1 Thesis description and objectives	4
1.2 Outline	6
I Supervised learning	9
2 Maximal-margin classifiers	11
2.1 Motivation	12
2.2 Support vector machines for classification	15
2.3 SVM dual formulation	21
3 Soft-margin support vector machines	29
3.1 Soft-margin ℓ_1 -loss linear classifiers	30
3.2 Soft-margin ℓ_2 -loss linear classifiers	38
3.3 Relaxed-bias classification	42
4 Performance of classification models	45
4.1 Model performance metrics	45
4.2 Parameter optimization	51
5 Deterministic solvers	53
5.1 Introduction	54

5.2	The MPRGP algorithm	57
5.3	Expansion strategies for the MPRGP algorithm	59
5.4	The SMALXE algorithm	61
5.5	Benchmarks	62
6	PermonSVM: SVM implementation on top of PETSc	71
6.1	Introduction	71
6.2	Parallel data loading	75
6.3	Application programming interface	76
7	Wildfires localization in Alaska	79
7.1	Motivation	81
7.2	Data processing	85
7.3	Benchmarks	87
II	Unsupervised learning	97
8	Vector quantification	99
8.1	Introduction	99
8.2	The k-means algorithm	100
8.3	The k-means++ algorithm	104
8.4	Parallel implementation	106
8.5	Benchmarks	109
9	Spectral clustering	115
9.1	Unnormalized spectral clustering	116
9.2	Normalized spectral clustering overview	120
9.3	Benchmarks	124
10	Conclusions	135
10.1	Overview of author's contribution	138
10.2	Possible directions of further research	139
A	List of author's publications	153
	Appendices	153
B	Projects and grants	157

C	Other activities during PhD studies	159
C.1	Supervising students	159
C.2	Internships and short research stays	160
C.3	Software development	161
C.4	Media	161

Acronyms

AI Artificial Intelligence

API Application Programming Interface

CHI Calinski-Harabasz Index

CPU Central Processing Unit

CT Computed Tomography

DBI Davies-Bouldin Index

ESA European Space Agency

ESM Earth System Model

EVI Enhanced Vegetation Index

FN False Negative

FP False Positive

FPR False Positive Rate

GPU Graphics Processing Unit

HPC High Performance Computing

IID (i.i.d) Independent and Identically Distributed

IoU Intersection over Union

KKT Karush–Kuhn–Tucker conditions

ML Machine Learning

MODIS Moderate Resolution Imaging Spectroradiometer

MPI Message Passing Interface

MPRGP Modified Proportioning with Reduced Gradient Projection

NDVI Normalized Difference Vegetation Index

NP Non-deterministic Polynomial-time

NP-hard Non-deterministic Polynomial-time hardness

NRSS Normalized Residual Sum of Squares

OvO One-vs-One binarization strategy

OvO One-vs-Rest binarization strategy

PAM Partitioning Around Medoids

PB Peta Byte

PCA Principal Component Analysis

PCM Pulse Code Modulation

PETSc Portable, Extensible Toolkit for Scientific Computation

QP Quadratic Program or Programming

RBF Radial Basis Function

ReLU Rectified Linear Unit

ROC Receiver Operating Characteristic

RREF Reduced Row Echelon Form

RSS Residual Sum of Squares

SMALXE Semimonotonic augmented Lagrangian

SSQ Sum of Squared Distance

SPS Symmetric Positive Semidefinite

SV Support Vector

SVM Support Vector Machine

TB Terra Byte

TN True Negative

TP True Positive

VC dimension Vapnik-Chervonenkis dimension

WLOG (w.l.o.g) Without Loss of Generality

Symbols

\mathbb{R}	real numbers
\mathbb{R}^+	positive real numbers
\mathbb{R}_0^+	non-negative real numbers
\mathbb{R}^n	linear space of all real-valued column vectors with n entries
$\mathbb{R}^{m \times n}$	linear space of all real-valued matrices with m rows and n columns
$\text{card}(\mathbb{S})$	size (cardinality) of set \mathbb{S}
$\stackrel{\text{def}}{=}$	defined
Ω	domain
$\langle \cdot, \cdot \rangle$	dot product
∇	gradient
\mathbf{g}^P	projected gradient
\mathbf{g}^f	free gradient
\mathbf{g}^c	chopped gradient
$\stackrel{\text{iid}}{\sim}$	independent and identically drawn from distribution
\max	maximum
\min	minimum
\sup	supremum
\inf	infimum
$\arg \max$	argument of maxima
$\arg \min$	argument of minima
$\lceil \cdot \rceil$	smallest integer greater than or equal to (ceil)
$\mathbb{X}_{\text{train}}$	training data set
\mathbb{X}_{test}	test data set
$\mathbb{X}_{\text{calib}}$	calibration data set
H	hyperplane H
\mathbf{w}	normal vector

ξ	misclassification error
b	bias
\mathbf{l}_b	lower bound
\mathbf{u}_b	upper bound
\mathbf{x}_{sv}	support vector
\mathbb{I}	index set
\mathbb{I}_{sv}	support vector index set
Θ	vector of parameters
γ	geometric margin
h_{Θ}	model
\dim_{vc}	Vapnik-Chervonenkis dimension
sign	signum
\mathbf{Q}	Hessian matrix
\mathbf{B}_E	equality constrained matrix
λ	Lagrange multiplier (supervised learning) or eigen value (unsupervised learning)
\mathcal{P}	primal functional
\mathcal{D}	dual functional
\mathcal{L}	Lagrangian
\mathbf{W}	weighted adjacency matrix
\mathbf{L}	graph Laplacian
cut	graph cut
ncut	normalized graph cut
\mathbf{O}	zero matrix
\mathbf{I}	identity matrix
Σ	covariance matrix
\mathbf{A}^T	transposition
\mathbf{A}^{-1}	inverse
\mathbf{A}^{\dagger}	generalized inverse
\mathbf{o}	zero vector
\mathbf{e}	vector of ones
$\ \cdot\ $	L2 norm
Ker	kernel
Im	image
Tr	trace

List of Figures

1.1	A Venn diagram illustrating relationship between artificial intelligence and machine learning	2
1.2	ML subgroups	3
1.3	Eminent achievement presented in the thesis: PermonSVM as a part of the PERMON toolbox is mentioned on the PETSc webpages	5
2.1	A model generalization ability demonstrated on binary classification problem .	14
2.2	Simplified binary classification problem solved by hard-margin SVM	19
3.1	A case of penalized test samples: the encircled (test) samples set are correctly classified. However, they are on the wrong side of their related margin	31
3.2	A case of misclassifying (test) samples: the encircled samples are not correctly classified	31
4.1	Visual representations of precision and sensitivity performance scores	48
4.2	Performance of a semantic segmentation model for wildfire localization	49
4.3	IoU score used for evaluating results of object detection	50
4.4	Fine-tuning hyperparameters using a grid search	51
4.5	Stratified cross-validation performing on 3 folds	52
5.1	Oxford IIIT Pet Dataset: dictionary learning using SIFT descriptors (relaxed-bias ℓ^1 -loss SVM). Tested on SALOMON cluster (IT4Innovations).	54
5.2	SMALXE-M and SMALXE- ρ variants.	62
5.3	Australian dataset: Number of Hessian multiplications for ℓ^1 -loss and ℓ^2 -loss depends on varying α_u	63
5.4	Australian dataset: Comparison of expansion strategies in the term of the number of the Hessian multiplications depending on α_u (Alpha user)	64
5.5	Alaska areas burned by wildfires. Red and green squares represent the training and test data sets, respectively. Wildfires localization are aggregated over 152 days from May to September 2004	65

5.6	Eigenvalues of the Hessian $\mathbf{A} + \rho \mathbf{y} \mathbf{y}^T$, $C = 1$	69
5.7	Eigenvalues for $\mathbf{Q} + \rho \mathbf{y}_{norm} \mathbf{y}_{norm}^T$ for Heart dataset, $C = 1$	69
5.8	Eigenvalues for $\mathbf{PAP} + \rho \mathbf{y}_{norm} \mathbf{y}_{norm}^T$ for Heart dataset, $C = 1$	70
6.1	PermonSVM: strong parallel scalability on URL dataset	73
6.2	The PERMON toolbox is mentioned as an external software library that use PETSc	74
7.1	Satellite image of wildfires raging across the width of Alaska using the MODIS instrument	80
7.2	Visualization of aggregated wildfire localization over 2005 in Alaska and the USA using MTBS product	81
7.3	Epoch scaling for CosmoFlow deep learning benchmark	82
7.4	A conceptual diagram illustrating the separation between supported GPU pro- gramming models for user code and the PETSc backend	83
7.5	Graphical user interface of software for streaming and downloading data	86
7.6	Alaska wildfire season in 2004 (aggregated data)	87
7.7	Highly unbalanced data set: 3,317,870 of background pixels and 70,631 pixels are associated with wildfire regions	88
7.8	Vegetation distribution in Alaska (August 8, 2004)	88
7.9	Example of eigenbands created from reflectance data (Alaska, 2004)	89
7.10	Side by side comparison of ground truth and predicted localization of wildfires obtained using the best SVM model	90
7.11	A side by side comparison of wildfire predicted by means of the best ℓ_1 -loss and ℓ_2 -loss models and the ground truth on the training and test data sets . .	93
7.12	Developing of mIoU score depending on retained variance for PCA, and PCA combined with Savitzky-Golay filter (regularized ℓ_1 -loss SVM)	94
7.13	Developing training time depending on retained variance for PCA, and PCA combined with Savitzky-Golay filter (regularized ℓ_1 -loss SVM)	95
7.14	Visualization of preliminary results attained by Platt scaling approach.	96
8.1	Visualisations of 2 and 3 dimensional Voronoi diagram	100
8.2	Photo depicting a fracture area of API 5L X-70 sheet steel (18.7 mm) after performing DWTT test	109
8.3	Reconstructed mesh and its features represented by normal vector in x , y , and z direction.	110
8.4	Visualization of 3 best results (k -means, k -means++, PAM++) and the worst one (k -medians)	112
8.5	A strong-scalability tests of PAM and PAM++ algorithms	113

9.1	Visual comparison results attained by the k -means and spectral clustering methods on the two spiral problem	116
9.2	Examples of radial basis functions	117
9.3	This showcase provides a simplified example of an undirected graph consisting of 6 vertices. A minimum degree of this graph is 1, and a maximal one equals 3. For this undirected graph, an adjacency matrix, a degree and a graph Laplacian matrices are outlined either.	118
9.4	Example of transverse CT image of fibre-reinforced concrete.	124
9.5	Binary masks of achieved material regions, namely from left concrete, air, fibres.	125
9.6	Visualization of volumetric images used in this section for benchmarking. . . .	126
9.7	This screeplot illustrates a profile of the first 10 eigenvalues associated with the 3D unnormalized Laplacian for a trabecular bone depicted in Figure 9.6a. Finite differences were computed using 26 nearest neighbours, and similarity among them was determined using an RBF function (standard deviation $\sigma = 0.01$). We solved associated eigenproblem employing the SLEPc framework (solver type ARPACK, $\text{rtol}=1\text{e}-1$, $\text{nev}=10$).	127
9.8	An example of spectrum related to the normalized Laplacian affected by rounding-off errors caused by limits of floating point arithmetic so that it consists of a negative eigenvalue.	128
9.9	A visualization of a solution associated with the normalized Laplacian of the 6 knn, that coordinations of new representations were normalized. Misclassified voxels are red colored.	130
9.10	From left: piecewise smooth image function $g(x, y) : \Omega_{roi} \rightarrow \mathbb{R}^3$ and its C_{loc}^1 fuction approximation $u(x, y) : \Omega_{roi} \rightarrow \mathbb{R}^3$	132
9.11	Cattedrale di Santa Maria del Fiore: the demonstration of image segmentation-based object categorization	133
9.12	Tatra T603: illustration of non-glueing image domain decomposition	133

List of Tables

4.1	A confusion matrix related to a binary classifier	46
5.1	Description related to training data set and test one. Proportions of samples in each category are pointed out as percents.	65
5.2	Solutions related to the complete SVM formulations using SMALXE + MPRGP. A default stopping condition is used. Results are attained using 64 MPI processes on the KAROLINA supercomputer. Setting of an inner solver: $\Gamma = 100$, a relative tolerance $\text{rtol} = 1e - 2$ and $\text{divtol} = 1e10$ for an outer loop (SMALXE). Penalty $C = 1$	66
5.3	Attained results using 64 MPI processes (KAROLINA). Solver: MPRGP, an expansion step is performed using the projected CG step, $\Gamma = 100$ in proportion criterion, a relative tolerance was set to 0.1; penalty $C = 1$	66
5.4	The best performance scores related to models trained employing the complete and relaxed-bias SVM formulations (on the test data set).	67
5.5	SMALXE and MPRGP convergence (l_2 -loss SVM with equality) for Heart dataset: the Hessian matrix is $\mathbf{A} + \rho \mathbf{y} \mathbf{y}^T$, $\rho = \ \mathbf{A}\ $, $\alpha_u = 1$, $\mathbf{x}_0 = \max(\mathbf{e}, \mathbf{l}) / \mathbf{x}_0 = \mathbf{l}$, and $C = 1$	68
5.6	SMALXE and MPRGP convergence (l_2 -loss SVM with equality) for Heart dataset: the Hessian matrix is $\mathbf{A} + \rho \mathbf{y}_{norm} \mathbf{y}_{norm}^T$, $\rho = \ \mathbf{A}\ $, $\alpha_u = 1$, $\mathbf{x}_0 = \max(\mathbf{e}, \mathbf{l}) / \mathbf{x}_0 = \mathbf{l}$, and $C = 1$	68
5.7	SMALXE and MPRGP convergence (l_2 -loss SVM with equality) for Heart dataset: the Hessian matrix is $\mathbf{PAP} + \rho \mathbf{y}_{norm} \mathbf{y}_{norm}^T$, $\rho = \ \mathbf{A}\ $, $\alpha_u = 1$, $\mathbf{x}_0 = \max(\mathbf{e}, \mathbf{l}) / \mathbf{x}_0 = \mathbf{l}$, and $C = 1$	68
6.1	Attained the performance scores of a classification model on a test data set trained employing a full dual ℓ_2 -loss SVM formulation on the URL data set . .	73
6.2	Training times of a classification models (a full dual ℓ_2 -loss SVM) on a different number of cores. Solver: MPRGP/SMALXE (default settings)	73

7.1	Attained models for wildfires localization trained using 16 GPUs NVIDIA Tesla V100. Solver: MPRGP, an expansion step is performed using the projected CG step, $\Gamma = 10$ in proportion criterion, $rtol = 0.1$, loss type ℓ^1 -loss. The best penalty C was selected using hyper-parameter optimization performed employing cross-validation combined with grid-search	91
7.2	Attained models for wildfires localization trained using 16 GPUs NVIDIA Tesla V100. Solver: MPRGP, an expansion step is performed using the projected CG step, $\Gamma = 10$ in proportion criterion, $rtol = 0.1$, loss type ℓ^2 -loss, single precision. The best penalty C was selected using hyper-parameter optimization performed employing cross-validation combined with grid-search	92
8.1	Evaluation of attained results using Calinski-Harabasz (CHI) and Davies-Bouldin (DBI) criterions. <i>These results were published in [101].</i>	112
9.1	Parameters of Hugo multiprocessor system (Huawei FusionServer 2488H V5). .	129
9.2	Comparison of the results attained by means of the unnormalized and normalized Laplacians associated with the foam64 (1 voxel) dataset depicted in Figure 9.6c. The results are summarized from 100 runs of each setting and observations reported as $mean \pm std$ if that makes a sense. Associated eigen-problems were solved employing the SLEPc framework such that solver type set to ARPACK, $rtol=1e-3$, $nev=50$	131

Chapter 1

Introduction

“The 19th century was the Age of steam, the epoch of vintage locomotives, marvellous industrial factories, and the first Benz’s automobile, that children know at an elementary school. Surprisingly, the first Turing-complete, the steam-powered computer called Analytical Engine was designed by an inventor Charles Babbage. He assembled small parts of this machine before his death in 1871, however, Analytical Engine was not completely built until 1991. Nowadays, this machine is exposed at the Science Museum of London.”

From the same point of view, the 20th century is referred to be the era of the Digital revolution – also known as the Third industrial revolution. We could say, it was literally started in 1947 when three researchers John Bardeen, Walter Brattain, and William Shockley invented and presented a transistor at Bell Labs (Murray Hill, New Jersey). It took almost six years after the presentation at Bell Labs then, at the University of Manchester, the first transistor computer constructed by Richard Grimsdale and D. C. Webb became operational. Years between 1947 and 1964 are known as the era of the second generation of computers, in which the ground-breaking semiconductor devices were exploited instead of vacuum tubes used in the first generation. The notable representative of this second generation is scientific computer IBM 7090 with the processing speed of around 100 Kflops/s that was designed for “large-scale scientific and technological applications.”

In 1958, Jack Kilby, American electrical engineer, along with Robert Noyce took part in the realisation of the first integrated circuit. Kilby’s invention, for which he was awarded the Nobel Prize in Physics in 2000, laid down the foundations of microprocessors and the next generations of computers. A hallmark of the third (1965 – 1971) and the fourth (1971 – 1980) generations was improving integrated circuits (IC) and very-large-scale integration of IC leading to the invention of the microprocessor, respectively. In 1982, Japan Ministry of International Trade and Industry initiated using massively parallel processing in computer

systems, called the fifth generation of computer systems, as a platform for future development in the artificial intelligence (AI). It was the inception of what that, today, the experts call the Fourth industrial revolution, in which AI plays a significant role.

Since 1871 through innovation of microprocessor in 70's, and Japan Ministry initiative in 1982, computers and their abilities have changed dramatically and, especially as they have been marked by incredible progress in the last two decades. For example, recent smartphone flagships are hugely more powerful than computers even just 10 years old. Internet users, not only on the social networks, and scientists in application research create a mind-boggling amount of data per day, therefore developing advanced technologies is mostly driven by collecting/storing and analysing such amount of data - by modern buzzwords called "big data."

Alongside with innovation of hardware, leading technology giants, namely Google, Microsoft, Apple, Netflix, Facebook, Amazon, Uber, and Tesla make trends in developing AI and machine learning (ML) algorithms. Essentially, AI is a capability of a computer performing tasks that are characteristic of human intelligence while ML is a category of algorithms that allow software to automatically learn and improve from experience without being explicitly programmed. From this point of view, AI is a kind of software written at a highly abstract level that solves general problems using, e.g. knowledge bases, and employing ML techniques. The relationship is depicted as a Venn diagram in Figure 1.1.

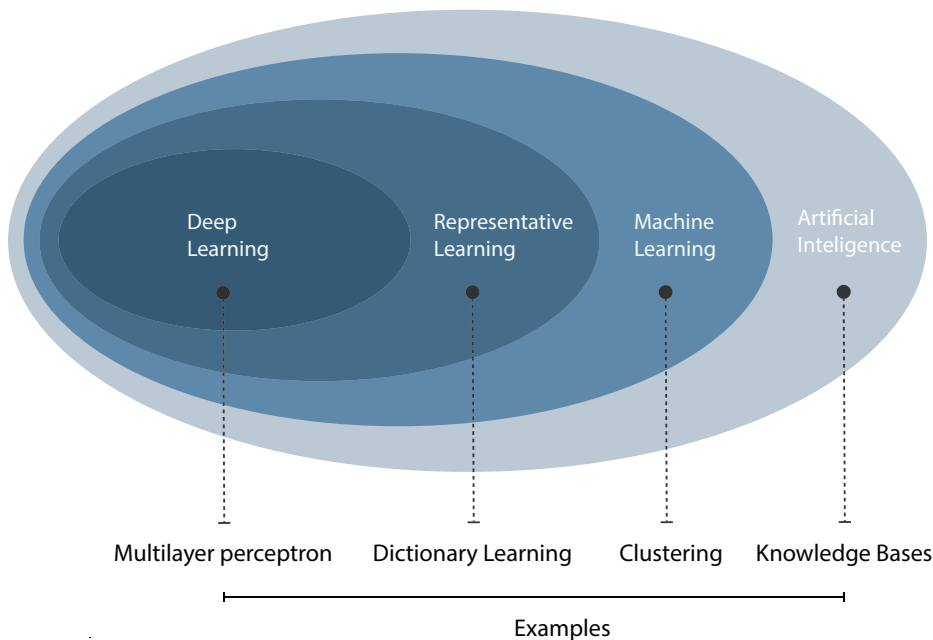


Figure 1.1: A Venn diagram illustrating that artificial intelligence employs machine learning, and, how deep learning is a kind of representative learning, which, in turn, is a kind of machine learning. Each section of the Venn diagram includes an example.

Acceptable performance of ML methods highly depends on right data representation, e.g. using features engineering methods. Representative learning is a set of techniques that allows a system (e.g. a new movie recommendation system implemented at the Netflix) to determine such an appropriate features or data representation automatically to perform a specific task. The prevailing (deep) neural networks [1] incorporate feature engineering inherently so that a series of hidden layers extract abstract features. It allows the computer system to build complex concepts out of simpler ones hierarchically, e.g. representing an image of a person face by combining corners and contours that are in turn defined in terms of edges [2, 3]. For this reason, the deep learning is considered to be a part of the representation learning [4].

In 1997, Tom M. Mitchell, American computer scientist and a professor at the Carnegie Mellon University, provided a widely quoted, more formal definition of the algorithms studied in the ML field:

“A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T , as measured by P , improves with experience E .” – Tom M. Mitchell, Machine Learning [5].

According to a concept and purpose, we commonly distinguish 3 groups of the ML algorithms, namely supervised [6], unsupervised learning [7], and the third ML paradigm is related to the reinforced learning [8, 9], see illustration in Figure 1.2.

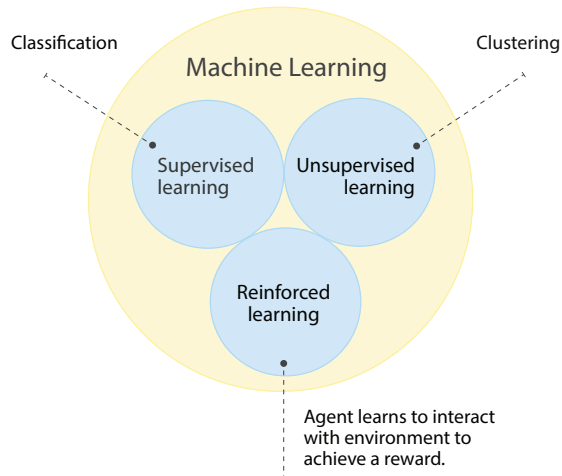


Figure 1.2: ML subgroups.

As the name suggests, the supervised learning is the group of algorithms that learning process is governed. An essential idea of this ML type is to predict the outcome given a set of input-output, i.e. samples-labels, pairs that is called training dataset. On the other hand, unsupervised learning algorithms are formulated in a such way that they identify commonalities, patterns, or hidden structures of unlabelled data.

The ML researchers observe, the learning process could be considerably improved in the sense of accuracy and time-consumption, when unlabelled and labelled data are reasonably combined. In application domains that unlabelled data are plentiful with a small proportion of labelled samples, semi-supervised learning [10] algorithms could provide a powerful framework, which is considered as a halfway between the supervised and unsupervised learning. Typically, the techniques make use of unlabelled data for training that is improved by means of exploiting labelled samples. The acquisition of

algorithms could provide a powerful framework, which is considered as a halfway between the supervised and unsupervised learning. Typically, the techniques make use of unlabelled data for training that is improved by means of exploiting labelled samples. The acquisition of

labelled data, which commonly present some *a priori* knowledge associated with the problem, requires the human experts and/or additional physical experiments.

The last learning paradigm mentioned above follows the *carrot and stick* policy. Specifically, an agent (robot or software component) learns how to interact with the surrounding environment, and to behave in it. An agent decision is quantifying so that it gets rewards (the carrot) when it makes a correct response or is punished (the stick) otherwise. It allows teaching an agent or series of agents to solve complex problems such as a resource management in computer cluster [11], traffic light control [12], or achieving super-human performance to beat the best players of video games – in the Dota 2 (the OpenAI platform) [13]. Nowadays, OpenAI is mainly known for services based on generative AI such as ChatGPT [14], Dall-E [15], and SORA [16]. The last mentioned product (SORA) has been launched 2 weeks before this thesis submission, it is a diffusion model for generating realistic videos from text prompts.

1.1 Thesis description and objectives

This doctoral thesis is closely connected with the ongoing ML research at the Department of Applied Mathematics (VŠB – Technical University of Ostrava) and Institute of Geonics (Czech Academy of Sciences) in cooperation with two world-leading research institutes, namely the Argonne National Laboratory and the Oak Ridge National Laboratory (USA). The thesis was completed under the guidance of two advisors. The principal supervisor of this thesis is doc. Ing. David Horák, Ph.D. from VSB – Technical University of Ostrava and the Institute of Geonics, while the co-supervisor Dr. Richard Mills represents the Argonne National Laboratory.

The general aim of the thesis is to study the applicability of classical ML models for solving complex and data-intensive applications, adapting optimization solvers for efficient parallel training of the ML models on supercomputer systems, and demonstrating the ability of the classical approaches on real-world applications.

More specifically, it consists of two parts related to supervised and unsupervised learning. The integral part of the first one is associated with the background of classification approaches based on Support Vector Machines (SVMs), which belong to maximal-margin models, and adapting algorithms for quadratic programming (QP) problems developed by Prof. Dostál's group. These algorithms are then subsequently used to solve underlying optimization problems arising from the SVM formulations so that they allow the training of the models in parallel using a (multi-node) multi-GPU approach. Note that traditional ML libraries are limited to up to tens of thousands of samples to solve the complete SVM dual formulations and, moreover, do not support multi-node multi-GPU training of these classification models. A software package called PermonSVM [17] outperforms these issues; The largest full dual problem successfully solved using PermonSVM was the benchmark of suspicious URL

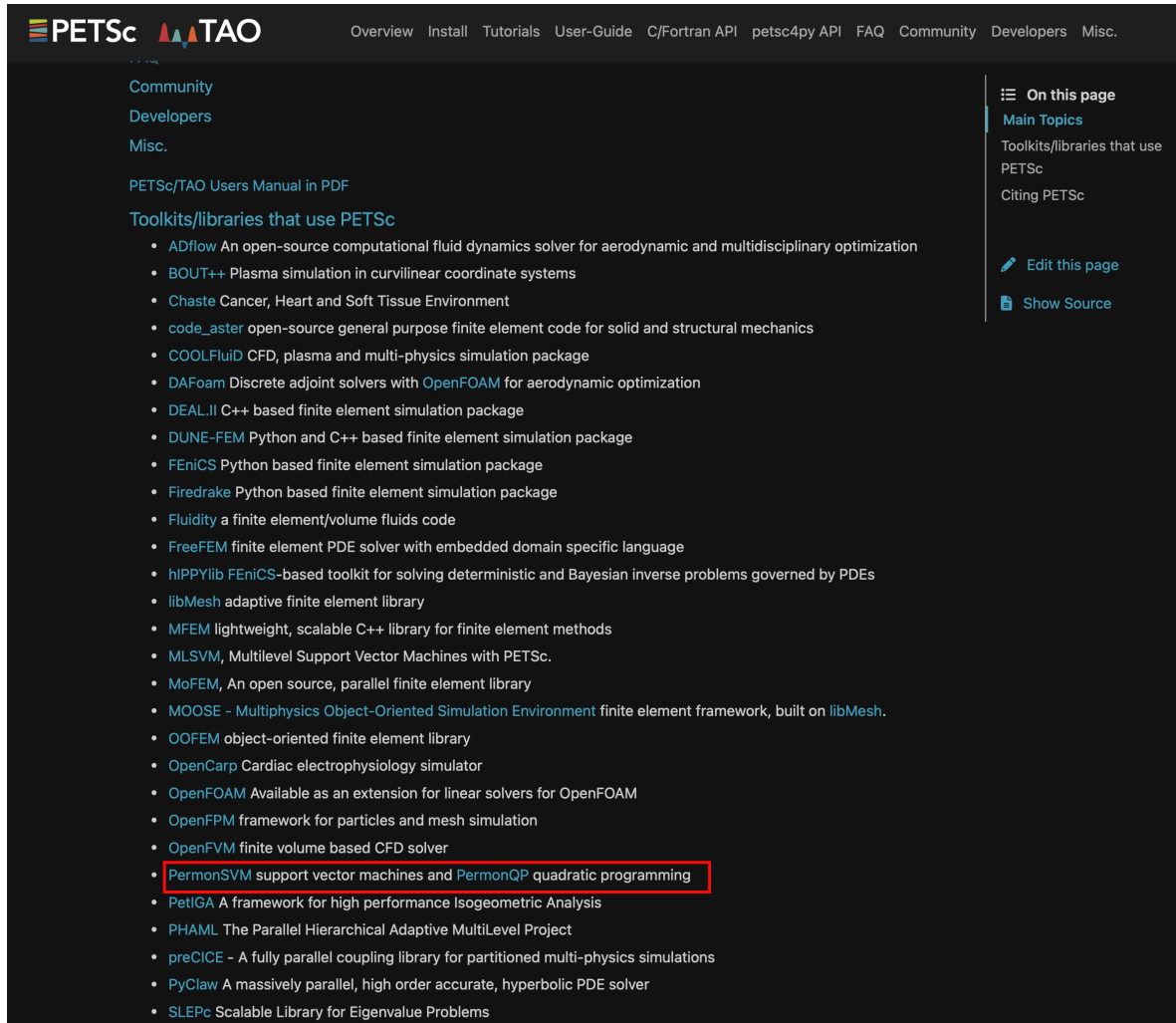


Figure 1.3: PermonSVM and PermonQP are mentioned as external software libraries, which use PETSc. On-line available on the following PETSc webpages <https://petsc.org/release/#toolkits-libraries-that-use-petsc>.

prediction [18] with more than 1.6 million training samples and over 3 million features. Furthermore, benchmarking on the fastest supercomputers in the world (Frontier and Summit) showed that PermonSVM could effectively scale up to hundreds of GPUs, solving relaxed-bias formulations. The implementation of PermonSVM is another core part of this thesis. The package is programmatically an extension of the PermonQP module, which is written on the top of the PETSc framework [19]. We are grateful for the success that PermonSVM, as well as the whole of the PERMON toolbox, are incorporated as external software that uses the PETSc framework, see Figure 1.3 on the previous page. This is one of the great achievements presented in this work.

The scalability of the QP algorithms implemented in PERMON used for training of mod-

els and PETSc as a building block became the main reasons for fruitful collaboration with world-leading research institutes Argonne and Oak Ridge National Laboratories (USA), aimed at wildfires localization in Alaska. From the author’s perspective, improvement of a model used in this application and its optimization are the most significant success achieved during his doctoral studies. The attained results were presented at the premier international conferences such as: *the AGU Annual Meeting* (USA), *the IALE North America Annual Meeting* (USA), *the IEEE International Conference on Data Mining* (USA), *the SIAM Conference on Computational Science and Engineering* (NL). The author of this thesis gave an invited talk at Argonne National Laboratory at the LANS Seminar series; please visit the following link for further info <https://www.anl.gov/event/wildfires-identification-in-alaska-using-satellite-images-and-machine-learning>.

The second part of this thesis focuses on the topics of unsupervised learning, mostly clustering approaches. This part consists of the very first results on the ML topics attained during the author’s doctoral studies. The theoretical background presented there is outlined in the form of a brief review and is related to vector quantification based on the Lloyd-type algorithms and spectral clustering. Furthermore, a parallel implementation of the vector quantification methods and a statistical approach based on the Bartlett’s test of homogeneity of variances for estimating the multiplicity of zero eigenvalue of the Laplace matrix are presented as well. In the practical part, two applications are introduced. The first one shows the detection of brittle and ductile fracture on a steel sample (API 5L X-70) using vector quantification techniques, and the second one employs spectral clustering for image segmentation without annotated data.

The results presented in this thesis are supported by the author’s publications in journal papers and conference papers. The list of these publications can be found in Appendix A. This thesis is also supported by research projects in which the author participates, involved, or is a principal investigator. The complete list related to these projects can be found in Appendix B.

1.2 Outline

The thesis is organized as follows:

Chapter 2 and Chapter 3 provide theoretical background concerning the hard-margin classifier of the SVM type and its soft-margin variants for non-linearly separable training data set, respectively. Chapter 3 outlines two soft-margin variants commonly known as the primal ℓ_1 -loss and ℓ_2 -loss SVM formulations, dualization of these formulations using the Lagrange duality approach, and introducing their primal and dual relaxed-bias variants. Comparison of the standard (complete) and relaxed-bias ℓ_1 -loss and ℓ_2 -loss soft-margin SVM formulations on the (initial) benchmark related to wildfire localization concludes Chapter 3.

Chapter 4 introduces and visually demonstrates evaluating classification models using various types of metrics, such as precision, sensitivity, F1, Intersection over Union (IoU), and others. In this chapter, an approach based on grid search combined with cross-validation for an optimal selection of model parameters is also presented.

Chapter 5 outlines an adaptation and optimizing algorithms MPRGP and SMALXE implemented in PermonQP for training SVM models. This includes ingredients associated with an optimal initial guess, adaptive expansion step length for the MPRGP algorithm, and testing of the SMALXE algorithm such as SMALXE-M and SMALXE- ρ .

Chapter 6 describes the implementation details of the package PermonSVM used for training classification models of the SVM type in parallel supporting multi-node multi-GPU training on NVidia and AMD graphic cards using the PETSc backends for CUDA and HIP, respectively. Effectively loading data employing the HDF5 file format and demonstrating parallel scalability of the package are presented and discussed as well.

Chapter 7 summarizes results attained by collaborating with the Argonne and Oak Ridge National Laboratories on an application aimed at wildfire localization in Alaska. This chapter also describes an improved workflow involving efficient analysis, fusion, and transformation for multispectral-temporal satellite images, streaming and visualization data from Google Earth Engine. This workflow was written entirely in Python, based on libraries such as branca [20], GDAL [21], Google EE Python API [22], folium [23], OpenCV [24], pandas [25], RAPIDS [26], scikit-learn [27], scipy [28], and others.

The following two sections are related to unsupervised learning topics focused on clustering approaches. Chapter 8 is related to vector quantification techniques based on the Lloyd-type algorithms such as k-means, k-means++, and their variants like PAM and others. This provides a brief description of the theoretical framework of this algorithm, and a parallel implementation in the C++ programming language is also discussed. The application for detecting brittle and ductile fracture areas concludes this chapter. While Chapter 9 introduces ability techniques based on spectral clustering for image segmentation without annotated data. This chapter briefly provides theoretical background related to unnormalized and normalized spectral clustering and outlines a method for estimating the multiplicity of zero eigenvalues of the Laplace matrix based on the Bartlett's test of homogeneity of variances.

Chapter 10 offers conclusions and an overview of the author's contributions to the topics introduced in thesis.

Part I

Supervised learning

Chapter 2

Maximal-margin classifiers

This chapter introduces the theoretical background concerning maximal-hard-margin classifiers, which are represented by hard-margin Support Vector Machines (SVM) in this thesis. In Section 2.1, we begin with motivation, where we demonstrate a classification approach based on the least squares model. Unfortunately, this approach does not provide any constraints on model qualities implicitly. Thus, we introduce the SVM model, which can overcome this issue. First, we outline a measure for quantifying a classifier capacity (and complexity) in Section 2.2, which was developed by Vapnik and Chervonenkis. This measure is commonly known as the Vapnik-Chervonenkis dimension in the ML community. A binary classifier having this dimension reasonably small attains a good generalization ability in the sense of its capacity and complexity. This assumption is essential for formulating an SVM classifier, which is introduced in its primal formulation later in Section 2.2. Further, the dualization of this primal formulation based on the Lagrange duality is discussed and outlined in Section 2.3.

Unless otherwise stated, let us consider a labelled data set belonging to two classes (categories) such that **samples are linearly separable**. It means there exists a hyperplane separating samples from the labelled data set into 2 groups such that samples in each group belong to same category. We then define the hyperplane H as follows:

$$H \stackrel{\text{def}}{=} \{\mathbf{x} \in \mathbb{R}^n : \langle \mathbf{w}, \mathbf{x} \rangle + b = 0\}, \quad (2.1)$$

where $\mathbf{w} \in \mathbb{R}^n$, $\mathbf{w} \neq \mathbf{o}$ is a normal vector of the hyperplane H and $b \in \mathbb{R}$ is its bias. The vector \mathbf{w} (associated with weights in ML models) together with a bias b are learnable parameters of ML models, including a single-layer perceptron or a neural network. Considering a connection to a context of ML, the weights control strengths of an input components, e.g. a sample attributes, to an output such as a classification function that the output is expressed as a weighted sum and where the bias is a constant term adjusting the strengths.

2.1 Motivation

Let us firstly denote a training set in a standard notation as a set of sample-label pairs such that its ordering is not mandatory:

$$\mathbb{X}_{\text{train}} \stackrel{\text{def}}{=} \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}. \quad (2.2)$$

Here, we assume that samples $\mathbf{x}_i \in \mathbb{R}^n$, for $i = 1, 2, \dots, m$ belong to two categories. We can encode these two categories by means of target labels y_1, \dots, y_m ¹ so that the corresponding target values are taken from the following set $\mathbb{Y} = \{-1, 1\}$.

Let still consider that samples $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m$, are linearly separable. Then, we find parameters $\mathbf{w} \in \mathbb{R}^n$ and $b \in \mathbb{R}$ associated with a linear model so that:

$$\left. \begin{aligned} \langle \mathbf{w}, \mathbf{x}_i \rangle + b &\geq 0 \wedge y_i = +1 \\ \langle \mathbf{w}, \mathbf{x}_i \rangle + b &< 0 \wedge y_i = -1 \end{aligned} \right\} i = 1, 2, \dots, m. \quad (2.3)$$

A naive approach of determining model parameters would be based on fitting it on labelled (input) samples employing linear regression, which is briefly discussed in the following text. While a linear regression is widely used for predicting continuous target variables, i.e regression task, it can be slightly modified for classification problems. For training a classification model using a linear regression approach, the samples become explanatory variables and labels are then considered as dependent. After the regression model is determined, we can straightforwardly set a discrimination function² by means of the attained hyperplane H associated with the linear model so that:

$$\left. \begin{aligned} h_{\Theta}(\mathbf{x}) &\geq 0 \iff \hat{y} = +1 \\ h_{\Theta}(\mathbf{x}) &< 0 \iff \hat{y} = -1 \end{aligned} \right\} \text{ i.e. } \text{sign}(h_{\Theta}(\mathbf{x})) = \begin{cases} \hat{y} = +1 & \text{if } h_{\Theta}(\mathbf{x}) \geq 0 \\ \hat{y} = -1 & \text{if } h_{\Theta}(\mathbf{x}) < 0 \end{cases} \quad (2.4)$$

where:

$$h_{\Theta}(\mathbf{x}) \stackrel{\text{def}}{=} \langle \Theta_0, \mathbf{x} \rangle + \Theta_1 \quad (2.5)$$

is the model defined by the vector of parameters $\Theta = [\Theta_0^T, \Theta_1]^T$, specifically:

$$\Theta_0 = \mathbf{w}, \Theta_1 = b. \quad (2.6)$$

Fitting the model $h_{\Theta}(\mathbf{x})$ using the linear regression approach on the training set $\mathbb{X}_{\text{train}}$ is

¹We can convert categorical variables into integer format using an label encoding approach such as an ordinal or one-hot, when a clear ordering of categories is defined. We refer the following tutorial [29] for further details.

²A discrimination function is basically a decision rule that allows us to decide to which class a sample \mathbf{x} belongs to.

drawn on minimizing the Residuals Sum of Squares (RSS), which is defined as:

$$\text{RSS} \stackrel{\text{def}}{=} \sum_{i=1}^m (y_i - \hat{y}_i)^2 = \sum_{i=1}^m [y_i - \text{sign}(h_{\Theta}(\mathbf{x}_i))]^2. \quad (2.7)$$

The corresponding minimization problem reads as follows:

$$\Theta^* = \arg \min_{\Theta \in \mathbb{R}^{n+1}} \sum_{i=1}^m [y_i - \text{sign}(h_{\Theta}(\mathbf{x}_i))]^2. \quad (2.8)$$

We can find the solution Θ^* associated with this optimization problem iteratively employing a first-order optimization algorithm, gradient descent for example.

Now, let us take a closer look at the solution Θ^* related to model parameters and discuss its quality, limits and suitability of the model $h_{\Theta}(\mathbf{x})$ for classification problems. Recall that a model minimizes RSS, which measures a level of variance in a training data set $\mathbb{X}_{\text{train}}$, is not explained by the regression model itself. Since a standard deviation is sensitive to outliers, the model is consequently impacted by samples which are significantly different from the rest ones in a data set $\mathbb{X}_{\text{train}}$. Considering these remarks, we can subsequently conclude that the model $h_{\Theta}(\mathbf{x})$, which is determined by means of performing a linear regression, besides the discriminant function (2.4) is not convenient enough for classification problems. This arises from a hypothesis of linear dependency among attributes and target labels that along with the underlying loss function RSS, which the model $h_{\Theta}(\mathbf{x})$ minimizes as we mentioned above, causes a high sensitivity of the model to outliers.

Instead of considering linear dependency among attributes and labels, we can think about linear separability of samples associated with a binary classification problem and formulate the problem of finding a separating hyperplane in a more convenient way. The simplest concept for supervised learning of (binary) linear classifiers represents a perceptron such as McCulloch-Pitts' model, or its variants and adaptations. These approaches are more deeply investigated in [30]. Unlike RSS as the objective function used in linear regression, a perceptron of its initial architecture minimizes ReLU (Rectified Linear Unit) [31] during a model training as follows:

$$\arg \min_{\Theta \in \mathbb{R}^{n+1}} \sum_{i=1}^m [-y_i h_{\Theta}(\mathbf{x})]_+ \stackrel{\text{def}}{=} \arg \min_{\Theta \in \mathbb{R}^{n+1}} \sum_{i=1}^m \max\{0, -y_i h_{\Theta}(\mathbf{x})\}, \quad (2.9)$$

where $h_{\Theta}(\mathbf{x})$ is the linear model (2.5) and the ReLU function defined as follows:

$$f_{\text{ReLU}}(z) = \max\{0, z\}, \quad (2.10)$$

is piecewise linear, which yields the input value z directly when z is positive, otherwise, it results to zero.

Let us examine an objective function (2.9) associated with a training of a model related to

a perceptron in more details. Let consider that a sample \mathbf{x} is correctly classified by means of the linear model $h_{\Theta}(\mathbf{x})$ and the discrimination function (2.4). Then, the value related to the term $-y_i h_{\Theta}(\mathbf{x}_i)$ is negative, which outcomes that ReLU outputs zero. On the other hand, this value equals a perpendicular distance from a sample \mathbf{x} to the separating hyperplane H in the case of a sample misclassification. As a consequence of these observations, a training phase of a perceptron converges when a resulting model classifies all training samples correctly. Since there are no constraints on hyperplane qualities, the optimization problem (2.9) has infinitely many solutions so that resulting models have more or less good generalization ability, demonstrated in Figure 2.1. This actuality depends on initial weights and an arrangement of feeding samples into a training process.

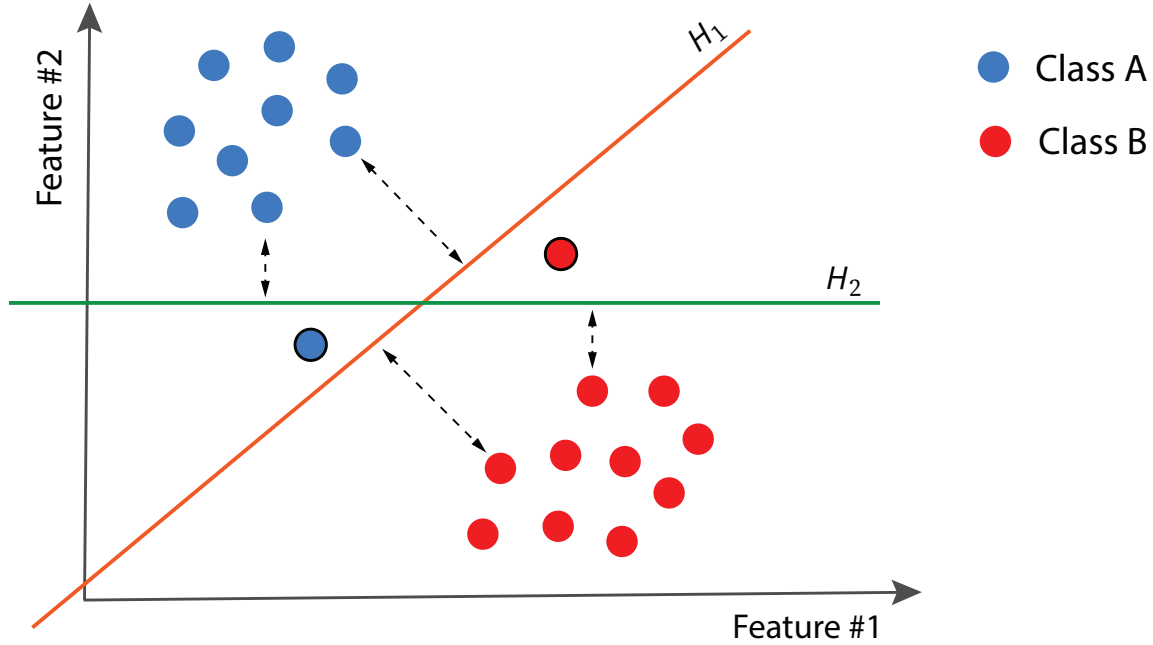


Figure 2.1: This example depicts two linear models and illustrates their generalization ability for a binary classification problem. Both models separate training samples $\mathbb{X}_{\text{train}}$ correctly. However, they differ in the distance of a related hyperplane (representing a model itself) to the closest point across classes. The primary goal is to maximize this distance to attain a good quality model. This example shows that the hyperplane H_1 (model 1) has a distance greater than the hyperplane H_2 (model 2). Thus, we can conclude that the model 1 performs better on unseen samples than the model 2. We demonstrate this fact on two test samples depicted as encircled points. The model 1 classifies these samples correctly. On the other hand, the model 2 has a 0% success rate on classifying the unseen samples.

2.2 Support vector machines for classification

A framework for quantifying a model performance was introduced by Vapnik and Chervonenkis in their computational learning theory [32], which attempts to explain a training process from a statistical point of view. They showed that a binary classifier having a reasonably small Vapnik-Chervonenkis dimension (VC dimension) attains a good generalization ability in the sense of its capacity and complexity; we denote this dimension as \dim_{vc} in the following text. For linear classifiers, an upper bound of the VC dimension is proportional to a geometric margin γ . To define this margin, still assume that the training samples belonging to $\mathbb{X}_{\text{train}} \subset \mathbb{R}^n$ are linearly separable.

Definition 1 (Geometric margin) *Let H be a hyperplane that separates a training data set $\mathbb{X}_{\text{train}}$ associated with a binary classification problem. A geometric margin of this hyperplane is defined as a distance from the hyperplane H to the closest point across both classes:*

$$\gamma(\mathbf{w}, b, \mathbb{X}_{\text{train}}) \stackrel{\text{def}}{=} \min_{(\mathbf{x}, y) \in \mathbb{X}_{\text{train}}} \frac{\hat{\gamma}(\mathbf{w}, b, (\mathbf{x}, y))}{\|\mathbf{w}\|}, \quad (2.11)$$

where $\hat{\gamma}(\mathbf{w}, b, \mathbf{x})$ is a functional margin such that:

$$\hat{\gamma}(\mathbf{w}, b, (\mathbf{x}, y)) \stackrel{\text{def}}{=} y (\langle \mathbf{w}, \mathbf{x} \rangle + b). \quad (2.12)$$

Unlike a functional margin $\hat{\gamma}(\mathbf{w}, b, (\mathbf{x}, y))$, the model parameters \mathbf{w} and b associated with a separating hyperplane H are normalized with respect to $\|\mathbf{w}\|$ in the case of its geometric margin $\gamma(\mathbf{w}, b, \mathbb{X}_{\text{train}})$. Thus, a scale of $\gamma(\mathbf{w}, b, \mathbb{X}_{\text{train}})$ is not affected by the magnitude of \mathbf{w} and b , which means that a geometric margin is a parameter scaling invariant [33].

Theorem 1 (Vapnik-Chervonenkis dimension) *Let $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\} \subseteq \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\| \leq R\}$, where \mathbb{X} be a set of samples belonging to $\mathbb{X}_{\text{train}}$ and R is a radius of the smallest hypersphere that covers \mathbb{X} :*

$$R = \max_{\mathbf{x} \in \mathbb{X}_{\text{train}}} \|\mathbf{x}\|. \quad (2.13)$$

Further, let $\Theta = [\mathbf{w}^T, b]^T$ be a vector of parameters and let $\mathbb{S} \subseteq \mathbb{X}$ then:

$$\mathcal{H}_{\mathbb{S}, \Lambda} \stackrel{\text{def}}{=} \{\mathbf{x} = \text{sign}(h_{\Theta}(\mathbf{x})) : \min_{\mathbf{x} \in \mathbb{S}} |h_{\Theta}(\mathbf{x})| = 1 \wedge 0 < \|\mathbf{w}\| \leq \Lambda\} \quad (2.14)$$

be a hypothesis class related to linear classifiers. Then, an upper bound of $\dim_{vc}(\mathcal{H}_{\mathbb{S}, \Lambda})$ satisfies:

$$\dim_{vc}(\mathcal{H}_{\mathbb{S}, \Lambda}) \leq \min \left\{ \left\lceil R^2 \Lambda^2 \right\rceil, m \right\} + 1, \quad (2.15)$$

where $\Lambda \stackrel{\text{def}}{=} \gamma(\mathbf{w}, b, \mathbb{X}_{\text{train}})^{-1}$, and $m = \text{card}(\mathbb{X}_{\text{train}})$ – a number of training samples.

An existing theory to Theorem 1 discussed by Burges, Vapnik in [34, 35] and Jayadeva [36] provides a tighter upper bound on a classifier complexity. It states that a classifier minimizing $R^2\Lambda^2$ in the sense a bias-variance tradeoff can be expected to give a better generalization performance.

This approach is considered as a general framework based on the VC dimension for quantifying a performance of a binary classifier in the sense of its complexity and capacity so far. The upper bound of the VC dimension also gives us a direction on how we can make to design a classifier of a good performance, i.e. a classifier with a small VC dimension. Since R is a radius of the smallest hypersphere that covers all samples from the training data set, the value of R is fixed for a given $\mathbb{X}_{\text{train}}$. Thus, we need to minimize Λ to obtain a reasonable small value related to the bound $R^2\Lambda^2$. Consequently, we find a separating hyperplane H with a maximal geometric margin, which is an essential ingredient for a definition of a maximal-margin classifier. Let us introduce and formulate a classifier that meets the qualities associated with a maximal geometric margin mentioned above using a scalar projection at first.

Definition 2 (Scalar projection) *Let \mathbf{f} and \mathbf{g} be vectors belonging to \mathbb{R}^n such that $\mathbf{f} \neq \mathbf{g}$, and let φ be an angle between \mathbf{f} and \mathbf{g} . A scalar projection $p_{\mathbf{g}}$ related to a projection \mathbf{f} onto \mathbf{g} is given by:*

$$p_{\mathbf{g}}(\mathbf{f}) = \|\mathbf{f}\| \cos \varphi = \langle \mathbf{f}, \hat{\mathbf{g}} \rangle, \quad (2.16)$$

where $\hat{\mathbf{g}} \stackrel{\text{def}}{=} \frac{\mathbf{g}}{\|\mathbf{g}\|}$.

Let \mathbf{w} be a normal vector related to a hyperplane H (2.1) and \mathbf{x} be a vector in a sample space \mathbb{R}^n . We assume this sample vector \mathbf{x} is the nearest one to the hyperplane H . Using these assumptions and additionally considering $\hat{\mathbf{w}} \stackrel{\text{def}}{=} \frac{\mathbf{w}}{\|\mathbf{w}\|}$, we can determine a projection of the sample vector \mathbf{x} onto \mathbf{w} by means of a scalar projection introduced above so that:

$$p_{\mathbf{w}}(\mathbf{x}) = \langle \mathbf{x}, \hat{\mathbf{w}} \rangle = \sum_{i=1}^n \hat{w}_i x_i = \|\mathbf{x}\| \cos \varphi. \quad (2.17)$$

Taking into account symmetry related to a dot product operator and consider $c \in \mathbb{R}$ be an arbitrary constant such that:

$$\|\mathbf{x}\| \cos \varphi = \frac{\langle \mathbf{w}, \mathbf{x} \rangle}{\|\mathbf{w}\|} \geq c \Rightarrow \frac{\langle \mathbf{w}, \mathbf{x} \rangle}{\|\mathbf{w}\|} - c \geq 0. \quad (2.18)$$

The last inequality in (2.18) is associated with a decision rule, which we previously defined in (2.4) for a linear model that is trained using a linear regression approach with a slight modification for classification tasks.

Definition 3 (Decision rule for linear classifier) Let $h_{\Theta}(\mathbf{x})$ be a model related to a linear classifier as follows:

$$h_{\Theta}(\mathbf{x}) = \langle \Theta_1, \mathbf{x} \rangle + \Theta_0 = \langle \mathbf{w}, \mathbf{x} \rangle + b, \quad (2.19)$$

where \mathbf{w} represents a normal vector of a hyperplane H and b denotes a bias, i.e. a displacement of a hyperplane from an origin. Consider that target values belong to the set $\mathbb{Y} = \{-1, 1\}$. Then, we can predict a category of a random sample \mathbf{x} by means of the following discrimination (classification) function:

$$\left. \begin{aligned} \langle \mathbf{w}, \mathbf{x} \rangle + b \geq 0 &\iff y = +1 \dots \text{positive class} \\ \langle \mathbf{w}, \mathbf{x} \rangle + b < 0 &\iff y = -1 \dots \text{negative class} \end{aligned} \right\} \text{ i.e. } y = \text{sign}(\langle \mathbf{w}, \mathbf{x} \rangle + b). \quad (2.20)$$

Combining the decision rule definition and the last inequality in (2.18), we can observe that the hyperplane offset from origin along \mathbf{w} is given by:

$$b = -c \|\mathbf{w}\| \Rightarrow \hat{b} \stackrel{\text{def}}{=} \frac{b}{\|\mathbf{w}\|} = -c \Rightarrow h_{\Theta}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + \hat{b}. \quad (2.21)$$

So far, we have analyzed the hyperplane qualities. Additionally, it is reasonable to require that the linear model defined by the hyperplane H also maximizes a geometric margin. We know, based on the previous discussion and remarks, this model quality reflects classifier robustness in the sense of its generalization performance. This ability is built by means of adapting the training data set $\mathbb{X}_{\text{train}}$ to perform a related inference process using a trained model on previously unseen samples.

In order to supervised learning in this part and mostly in the rest of this thesis, we distinguish two phases associated with developing ML models. Specifically, they are a training phase followed by a test phase. Since we will not run with semi-labeled or unlabelled samples in these phases, it seems to be reasonable denoting these unseen samples as the test data set \mathbb{X}_{test} for improving readability and comprehension of the following text. Further, recall that training $\mathbb{X}_{\text{train}}$ and unseen samples \mathbb{X}_{test} are independent and identically drawn from the same probability distribution $p(\mathbf{x}, y)$:

$$\mathbb{X}_{\text{train}} \stackrel{\text{iid}}{\sim} p(\mathbf{x}, y) \wedge \mathbb{X}_{\text{test}} \stackrel{\text{iid}}{\sim} p(\mathbf{x}, y). \quad (2.22)$$

Let still assume that the given training samples are linearly separable and let $\sigma \in \mathbb{R}$. We can now formulate finding a maximal-margin (linear) model as an optimization problem so that each training sample \mathbf{x} has the geometric margin $\gamma(\mathbf{w}, b, (\mathbf{x}, y))$ at least $\sigma \in \mathbb{R}^+$.

Formally, given $\mathbb{X}_{\text{train}}$, find σ^* , \mathbf{w}^* , b^* such that:

$$(\sigma^*, \mathbf{w}^*, b^*) = \arg \max_{\sigma, \mathbf{w}, b} \sigma \text{ s.t. } \begin{cases} 0 \leq \sigma \leq \frac{y[\langle \mathbf{w}, \mathbf{x} \rangle + b]}{\|\mathbf{w}\|}, \\ \forall (\mathbf{x}, y) \in \mathbb{X}_{\text{train}}. \end{cases} \quad (2.23)$$

Right now, let us take a closer look at this optimization problem. An associated objective function σ is linear and therefore convex. However, $\|\mathbf{w}\|$ in a denominator of an upper bound in (2.23) leads to non-convexity of this constraint and conduct the non-convex problem at all, see [33] for further details. Comparing to a convex optimization problem, it can be costly and thus generally harder to attain a global optima in this case. Recall, a convex optimization problem ensures that every local optimum is actually a global optimum. Thus, we introduce a chain of transformations and remarks using which we modify the non-convex problem (2.23) into a convex one.

First, we include $\|\mathbf{w}\| = 1$ as an auxiliary constraint into our optimization problem (2.23). This guarantees that a geometric margin equals a functional margin, which is at least σ for each sample in the training set $\mathbb{X}_{\text{train}}$. This modification results into the following optimization problem:

$$(\sigma^*, \mathbf{w}^*, b^*) = \arg \max_{\sigma, \mathbf{w}, b} \sigma \text{ s.t. } \begin{cases} 0 \leq \sigma \leq y[\langle \mathbf{w}, \mathbf{x} \rangle + b], \forall (\mathbf{x}, y) \in \mathbb{X}_{\text{train}}, \\ \|\mathbf{w}\| = 1. \end{cases} \quad (2.24)$$

However, the constraint $\|\mathbf{w}\| = 1$ is still non-convex. We can simply show that $\|\mathbf{w}\| = 1$ in \mathbb{R}^2 represents an unit circle. In general, it corresponds to a unit hypersphere in \mathbb{R}^n , and thus the constraint is non-convex. To rid of this constraint from the optimization problem, we can divide the objective function in (2.24) by the norm $\|\mathbf{w}\|$. Recall, the constraint $\|\mathbf{w}\| = 1$ ensures that a geometric margin equals a functional margin in (2.24). So if σ is a lower bound for a functional margin, then $\hat{\sigma} = \frac{\sigma}{\|\mathbf{w}\|}$ is a lower bound for a geometric margin in the following optimization problem:

$$(\hat{\sigma}^*, \mathbf{w}^*, b^*) = \arg \max_{\hat{\sigma}, \mathbf{w}, b} \hat{\sigma} \|\mathbf{w}\| \text{ s.t. } \begin{cases} 0 \leq \hat{\sigma} \leq y[\langle \mathbf{w}, \mathbf{x} \rangle + b], \\ \forall (\mathbf{x}, y) \in \mathbb{X}_{\text{train}}. \end{cases} \quad (2.25)$$

Employing our previous remark about scaling invariance (at the beginning of this section) associated with a geometric margin, we can impose $\sigma = 1$ as the scaling parameter value and, finally, apply the last sequence of modifications:

$$(\mathbf{w}^*, b^*) = \arg \max_{\mathbf{w}, b} \frac{1}{\|\mathbf{w}\|} \text{ s.t. } \begin{cases} y[\langle \mathbf{w}, \mathbf{x} \rangle + b] \geq 1, \\ \forall (\mathbf{x}, y) \in \mathbb{X}_{\text{train}}, \end{cases} \quad (2.26a)$$

$$\simeq \arg \min_{\mathbf{w}, b} \|\mathbf{w}\| \text{ s.t. } \begin{cases} y [\langle \mathbf{w}, \mathbf{x} \rangle + b] \geq 1, \\ \forall (\mathbf{x}, y) \in \mathbb{X}_{\text{train}}, \end{cases} \quad (2.26b)$$

$$\simeq \arg \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \text{ s.t. } \begin{cases} y [\langle \mathbf{w}, \mathbf{x} \rangle + b] \geq 1, \\ \forall (\mathbf{x}, y) \in \mathbb{X}_{\text{train}}. \end{cases} \quad (2.26c)$$

$$= \arg \min_{\mathbf{w}, b} \frac{1}{2} \langle \mathbf{w}, \mathbf{w} \rangle \text{ s.t. } \begin{cases} y [\langle \mathbf{w}, \mathbf{x} \rangle + b] \geq 1, \\ \forall (\mathbf{x}, y) \in \mathbb{X}_{\text{train}}. \end{cases} \quad (2.26d)$$

We transformed the initially stated non-convex optimization problem (2.23) into the convex quadratic program (QP) called *primal hard-margin SVM* that has a quadratic objective function $\frac{1}{2} \|\mathbf{w}\|^2$ and affine inequality constraints, which form a polyhedron set \mathcal{C} [37]. Using the fact that $y_i \in \mathbb{Y}$, we can interpret the set of constraints \mathcal{C} so that all training samples lie on, above or below margins:

$$\langle \mathbf{w}, \mathbf{x} \rangle + b = \pm 1 \quad (2.27)$$

for positive samples (Class A) or negative samples (Class B), respectively.

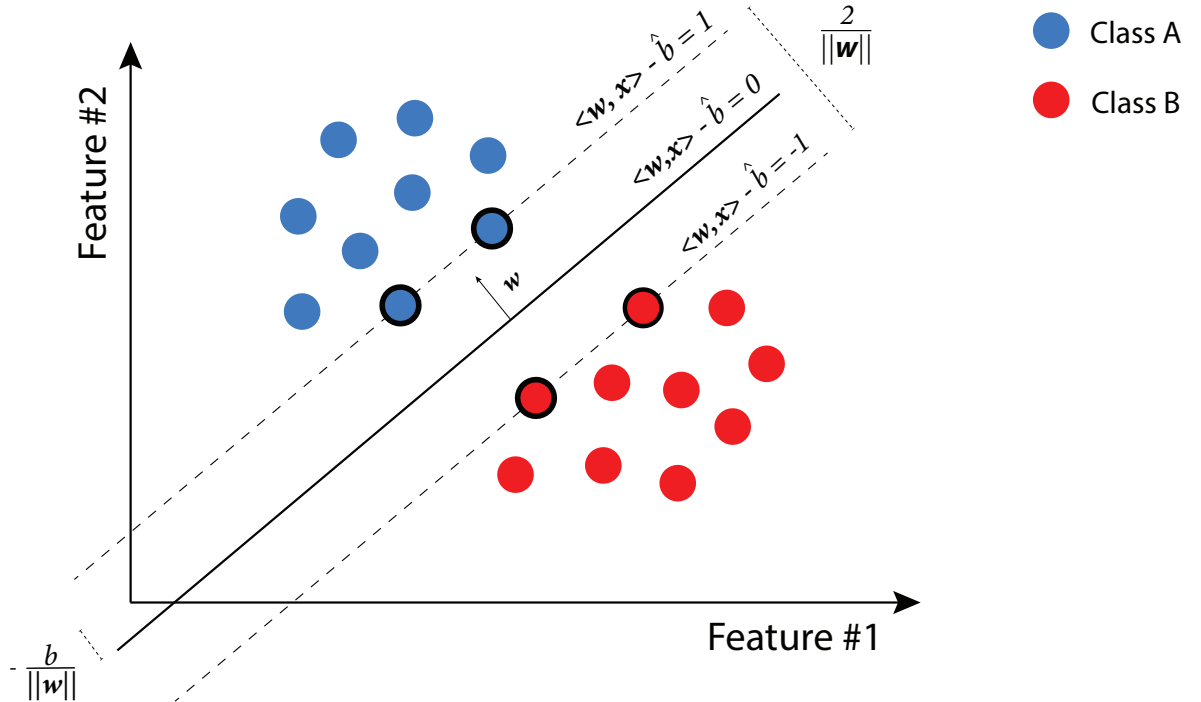


Figure 2.2: The example illustrates a simplified binary classification problem in 2D solved by a linear hard-margin SVM. Encircled samples represent support vectors. *Original image was downloaded from [38]. It was slightly modified for the purpose of this text.*

Samples that exactly lie on the margins (2.27) are called support vectors (SV) - depicted

as encircled samples lying on the dashed lines in Figure 2.2 (on the previous page). In this simplified example, the lines represent margins for training samples belonging to \mathbb{R}^2 .

Further, let $\mathbf{e} \in \mathbb{R}^m$ be an all-ones vector, $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m] \in \mathbb{R}^{n \times m}$ corresponds to training samples, and $\mathbf{Y}_d = \text{diag}(\mathbf{y}) \in \mathbb{R}^{m \times m}$. We can then rewrite the primal formulation above into a QP problem in a standard matrix form using a vector of parameters $\Theta = [\mathbf{w}, b] \in \mathbb{R}^{n+1}$ and matrices above as follows:

$$\left(\mathcal{P}_{\text{hard}}^{\text{SVM}}\right) : \Theta^* = \arg \min_{\Theta} \frac{1}{2} \Theta^T Q \Theta \quad \text{s.t.} \quad B \Theta \leq -\mathbf{e}, \quad (2.28)$$

where

$$Q = \begin{bmatrix} \mathbf{I}_n & \mathbf{O}_{n,1} \\ \mathbf{O}_{1,n} & \mathbf{O}_1 \end{bmatrix} \in \mathbb{R}^{(n+1) \times (n+1)}, \quad (2.29a)$$

$$B = \begin{bmatrix} -\mathbf{Y}_d \mathbf{X}^T, & -\mathbf{y} \end{bmatrix} \in \mathbb{R}^{m \times (n+1)}, \quad (2.29b)$$

$$\Theta = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} \in \mathbb{R}^{n+1}. \quad (2.29c)$$

By solving the optimization problem $\mathcal{P}_{\text{hard}}^{\text{SVM}}$ (2.28), we attain a linear model $h_{\Theta}(\mathbf{x})$, which maximizes geometric margin $\gamma(\mathbf{w}, b, \mathbb{X}_{\text{train}})$ and which is named as a maximal-margin classifier early in this text. According to the VC dimension definition, we know that such a classifier reflects robustness in the sense of its generalization performance, as we mentioned in our previous discussion.

Let us draw attention to the equation (2.27) associated with the margins equal $+1$ for positive samples and -1 for negative samples. Notice that this equation represents two parallel margins and solving the optimization problem $\mathcal{P}_{\text{hard}}^{\text{SVM}}$ results in the linear model represented by separating hyperplane H equidistant to these margins. To calculate the distance between them, let us take a difference vector \mathbf{d}_{sv} of support vectors \mathbf{x}_{sv}^+ and \mathbf{x}_{sv}^- . The first one support vector \mathbf{x}_{sv}^+ lies on the margin equal $+1$ and the second support vector \mathbf{x}_{sv}^- is associated with the margin, which equals -1 . Let $\hat{\mathbf{w}}$ be a normalized normal vector related to the separating hyperplane H . The distance between these margins is then determined as a scalar projection of the difference vector \mathbf{d}_{sv} onto the normal vector \mathbf{w} as follows (see also Figure 2.2):

$$\langle \mathbf{d}_{\text{sv}}, \hat{\mathbf{w}} \rangle = \langle \mathbf{x}_{\text{sv}}^+ - \mathbf{x}_{\text{sv}}^-, \hat{\mathbf{w}} \rangle = \frac{\langle \mathbf{x}_{\text{sv}}^+, \mathbf{w} \rangle}{\|\mathbf{w}\|} - \frac{\langle \mathbf{x}_{\text{sv}}^-, \mathbf{w} \rangle}{\|\mathbf{w}\|} = \frac{1-b}{\|\mathbf{w}\|} + \frac{1+b}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}. \quad (2.30)$$

Let us note that the problem (2.28) is infeasible if samples in $\mathbb{X}_{\text{train}}$ are not linearly separable. More detailed discussion about solvability of this primal problem can be found in [39].

2.3 SVM dual formulation

Each convex, and even non-convex, constrained optimization problem \mathcal{P} (called primal) can be transformed into its dual formulation \mathcal{D} . This typically provides an approach to solve the original problem \mathcal{P} in an easier way, e.g. by decreasing a computational cost arising from reduction variables or transforming it into an optimization problem with a more convenient structure, e.g affine inequality constraints are transformed to simple bounds.³

Note that the significant part related to benchmarks and applications presented in this thesis is focused on analyzing classification models trained using the PERMON toolbox [40]. More specifically, we employ its extension called PermonSVM [17]. It brings a high-level API⁴ for the out-of-the-box abstraction of the SVM formulations and internally sets up an optimization solver and objects used for training models. A solver interface implemented in a core module in PERMON is originally designed for QP problems with box and equality constraints. Let us refer to Chapter 6, where PermonSVM is introduced more in details.

Unless otherwise stated, let $\mathbf{z} \in \mathbb{R}^r$ be a variable. We can then write down a general formulation associated with a QP optimization problem that consists of the prescribed inequality constraints (a box type) and the equality constraints as follows:

$$\begin{aligned} \text{find } \mathbf{z}^* &= \arg \min_{\mathbf{z}} \frac{1}{2} \mathbf{z}^T \mathbf{A} \mathbf{z} - \mathbf{b}_{\text{rhs}}^T \mathbf{z} \\ \text{subject to } &\begin{cases} \mathbf{l}_b \leq \mathbf{z} \leq \mathbf{u}_b, \\ \mathbf{B}_E \mathbf{z} = \mathbf{c}_E. \end{cases} \end{aligned} \quad (2.31)$$

The vector $\mathbf{z}^* \in \mathbb{R}^r$ corresponds to an optimal solution of the constrained QP problem introduced in (2.31), where $\mathbf{A} \in \mathbb{R}^{r \times r}$ is a symmetric positive definite (SPD), or symmetric positive semidefinite (SPS), Hessian matrix. The vector $\mathbf{b}_{\text{rhs}} \in \mathbb{R}^r$ represents a right-hand side related to the quadratic objective function:

$$f_{\text{QP}}(\mathbf{z}) \stackrel{\text{def}}{=} \frac{1}{2} \mathbf{z}^T \mathbf{A} \mathbf{z} - \mathbf{b}_{\text{rhs}}^T \mathbf{z}. \quad (2.32)$$

As stated above, we consider the following types of constraints outlined for the QP problem introduced in the formulation (2.31):

- **box constraints** - these involve a lower bound vector $\mathbf{l}_b \in (\{-\infty\} \cup \mathbb{R})^r$ and an upper bound vector $\mathbf{u}_b \in (\mathbb{R} \cup \{+\infty\})^r$ such that the components corresponding to the solution vector \mathbf{z} must satisfy $\mathbf{l}_b \leq \mathbf{z} \leq \mathbf{u}_b$. If only a vector \mathbf{l}_b is given⁵, we refer these

³The dualization could also transform the original problem \mathcal{P} into the dual one \mathcal{D} having favourable spectral properties. This is not generally true in order to solve ML problems of the SVM type.

⁴Abbreviation API is commonly used in information technology and computer science. It stands for Application Program Interface.

⁵It means that all components of a vector \mathbf{u}_b , which are associated with an upper bound, equal $+\infty$.

constraints as a lower bound. Let us note that we do not consider an upper bound for the QP problems arising from ML models of an SVM type in this thesis.

- **equality constraints** - this constraint type is characterized using an equality constraints matrix $\mathbf{B}_E \in \mathbb{R}^{m_e \times r}$ and a vector $\mathbf{c}_E \in \mathbb{R}^{m_e}$, which is an equality constraint right-hand side. Note that setting the equality constraints is optional for a specification of an optimization problem in the PERMON toolbox.⁶

Now, let us take a look at the hard-margin SVM (primal) formulation $\mathcal{P}_{\text{hard}}^{\text{SVM}}$ that was previously defined in (2.28). As we can see, the problem consists of inequality constraints. Unfortunately, these inequality constraints take a different (say general) form than the QP API implemented in PERMON is designed for; we consider the box constraints as a special case of these inequality constraints. A useful tool for analyzing constrained optimization problems and relationships between primal \mathcal{P} and dual \mathcal{D} formulations is a Lagrangian function \mathcal{L} , which is also known as Lagrangian. Since reviewing the theory related to Lagrange duality used for convex optimization is out of the scope of this thesis, we recommend the book *Convex optimization* [42] written by Boyd and Vandenberghe for additional details.

The essential idea beyond the Lagrange duality is to take the constraints into account by augmenting an objective function, which is associated with the primal optimization problem \mathcal{P} , using a weighted sum of the functions related to constraints [37, 42]. Employing this approach, we set up a Lagrangian function, where these weights are represented by means of the Lagrange multipliers, which are typically called dual variables:

$$\{\lambda_1, \lambda_2, \dots, \lambda_{s-1}, \lambda_s\} \subset \{\mathbb{R}_+ \cup \{0\}\}^s. \quad (2.33)$$

Definition 4 (Lagrangian for an optimization problem with inequality constraints)

Let us consider an optimization problem with minimized function f_0 and inequality constraints represented using the constraint functions $f_{i=1, \dots, s} : \mathbb{R}^r \rightarrow \mathbb{R}$ such that:

$$\arg \min_{\mathbf{z}} f_0(\mathbf{z}) \text{ s.t. } f_i(\mathbf{z}) \leq 0 \text{ where } i \in \{1, \dots, s\}, \quad (2.34)$$

where $\mathbf{z} \in \mathbb{R}^r$ denotes a variable. Let Ω_I be a domain⁷ related to the inequality constraint functions and defined as follows:

$$\Omega_I \stackrel{\text{def}}{=} \bigcap_{i=1}^s \text{dom } f_i, \quad (2.35)$$

⁶More technical insights regarding the software architecture of the PERMON toolbox and implemented modules can be found in Václav's Hapla doctoral thesis [41].

⁷Note. We interpret the domain Ω_I defined in (2.35) as a feasible region, where all inequality constraints are satisfied.

where:

$$\text{dom } f_i = \{\mathbf{z} : f_i(\mathbf{z}) \leq 0\}. \quad (2.36)$$

Regarding the domain Ω_I (2.35), let us assume that this domain is non-empty, i.e. $\Omega_I \neq \emptyset$. We can then define a Lagrangian function $\mathcal{L} : \mathbb{R}^r \times \mathbb{R}^s \rightarrow \mathbb{R}$ corresponding to the optimization problem (2.34) such that:

$$\mathcal{L}(\mathbf{z}, \boldsymbol{\lambda}) \stackrel{\text{def}}{=} f_0(\mathbf{z}) + \sum_{i=1}^s \lambda_i f_i(\mathbf{z}). \quad (2.37)$$

A domain $\text{dom } \mathcal{L}$ associated with the Lagrangian function \mathcal{L} (2.37) equals:

$$\text{dom } \mathcal{L} = \mathbb{R}^s \times D_I, \quad (2.38)$$

where:

$$D_I = \{\boldsymbol{\lambda} : \lambda_i \geq 0, i = 1, 2, \dots, s\}, \quad (2.39)$$

is a set of the Lagrange multipliers.

Concerning Definition 4 and assuming that a number of the Lagrange multipliers equals to a number of training samples,⁸ a Lagrangian function, which corresponds to the primal problem $\mathcal{P}_{\text{hard}}^{\text{SVM}}$ (2.28), takes the following form:

$$\mathcal{L}(\boldsymbol{\Theta}, \boldsymbol{\lambda}) = \frac{1}{2} \boldsymbol{\Theta}^T \mathbf{Q} \boldsymbol{\Theta} + \boldsymbol{\lambda}^T (\mathbf{B} \boldsymbol{\Theta} + \mathbf{e}), \quad (2.40)$$

where $\boldsymbol{\lambda} \in \mathbb{R}^m$ is the vector that consists of the Lagrange multipliers, and Ω_I is defined as follows:

$$\Omega_I = \{\boldsymbol{\Theta} \in \mathbb{R}^{n+1} : \mathbf{B} \boldsymbol{\Theta} \leq -\mathbf{e}\}. \quad (2.41)$$

Equivalently, we can set up a Lagrangian for the optimization problem (2.26d), where we consider primal variables as components of the vector $\boldsymbol{\Theta} = [\mathbf{w}^T, b]^T$:

$$\begin{aligned} \mathcal{L}(\mathbf{w}, b, \boldsymbol{\lambda}) &= \frac{1}{2} \langle \mathbf{w}, \mathbf{w} \rangle + \sum_{i=1}^m \lambda_i (-y_i [\langle \mathbf{w}, \mathbf{x}_i \rangle + b] + 1) \\ &= \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^m \lambda_i (y_i [\langle \mathbf{w}, \mathbf{x}_i \rangle + b] - 1). \end{aligned} \quad (2.42)$$

Note, a vector $\boldsymbol{\Theta}$ denotes a vector of parameters associated with an ML model. In order to an application of SVM, it is better to take into account primal variables as components of

⁸Recall. We interpret inequality constraints in the last optimization problem introduced in the transformation chain (2.26d) such that all training samples lie on, above or below respective geometric margins. Thus, we have introduced the inequality constraint for each training sample in $\mathbb{X}_{\text{train}}$.

a vector Θ . Occasionally, it is suitable to represent primal variables as a vector Θ itself. It depends on a particular case. Therefore, we often refer to both forms in this thesis.

Recall that our original primal problem (2.28) is convex having a quadratic objective function and affine inequality constraints. We can directly show that the Lagrangian \mathcal{L} introduced in (2.42) is also convex with respect to the variable Θ :

$$\begin{aligned}\nabla_{\Theta} \mathcal{L}(\Theta, \lambda) &= Q\Theta + \lambda^T B, \\ \Delta_{\Theta} \mathcal{L}(\Theta, \lambda) &= Q,\end{aligned}\tag{2.43}$$

where Q is a symmetric positive semidefinite (SPS) matrix introduced in (2.29a).

Definition 5 (Lagrange dual function for inequality constraints) *A Lagrangian dual function or, shortly, a dual function $g(\lambda) : D_I \rightarrow \mathbb{R}$ is defined as a infimum of a Lagrangian $\mathcal{L}(z, \lambda)$ over a variable $z \in \mathbb{R}^r$ for $\lambda \in D_I$ such that:*

$$g(\lambda) \stackrel{\text{def}}{=} \inf_{z \in \mathbb{R}^r} \mathcal{L}(z, \lambda), \tag{2.44}$$

where Ω_I is a feasible set introduced in (2.35).

The concavity of the dual function $g(\lambda)$ is one of the key property used in convex optimization. We proved above that it does not depend on the convexity (or non-convexity) related to a primal functional \mathcal{P} . Considering this observation, we can provide the first insight into a solution of the primal problem. We take an advantage of it in the following text.

Further, we can effectively express a dual problem \mathcal{D} using a dual function previously defined in (2.44) such that:

$$d^* = \sup_{\lambda \geq \mathbf{o}} \inf_{\Theta \in \mathbb{R}^{n+1}} \mathcal{L}(\Theta, \lambda) = \sup_{\lambda \geq \mathbf{o}} g(\lambda), \tag{2.45}$$

and the primal problem \mathcal{P} by means of swapping infimum and supremum:

$$\inf_{\Theta \in \Omega_I} f(\Theta) = \inf_{\Theta \in \mathbb{R}^{n+1}} \sup_{\lambda \geq \mathbf{o}} \mathcal{L}(\Theta, \lambda) = p^*. \tag{2.46}$$

As we mentioned earlier in this section, a dual function yields a lower bound on the optimal value p^* associated with a primal problem. We can formalize this fact using the following inequality:

$$d^* = \sup_{\lambda \geq \mathbf{o}} \inf_{\Theta \in \mathbb{R}^{n+1}} \mathcal{L}(\Theta, \lambda) \leq \inf_{\Theta \in \mathbb{R}^{n+1}} \sup_{\lambda \geq \mathbf{o}} \mathcal{L}(\Theta, \lambda) = p^*. \tag{2.47}$$

A difference between the value p^* associated with primal problem \mathcal{P} and the value d^* related to a dual problem \mathcal{D} is called duality gap, i.e. $p^* - d^*$.

Recall. The vectors $\Theta \in \mathbb{R}^{n+1}$ and $\lambda \in \mathbb{R}_+^m$ are related to primal respective dual variables. In the following text, we briefly form the dual formulation. First, we state necessary conditions for a solution pair (Θ^*, λ^*) to be considered as locally optimal for a primal problem \mathcal{P} , and a dual problem \mathcal{D} either. These conditions are called the Karush-Kuhn-Tucker (KKT) conditions, and they establish a link between the primal and dual problems as follows:

$$\nabla_{\Theta} \mathcal{L}(\Theta^*, \lambda^*) = \mathbf{o} \quad (\text{stacionarity}) \quad (2.48a)$$

$$\nabla_{\lambda} \mathcal{L}(\Theta^*, \lambda^*) = B\Theta^* + e \leq \mathbf{o} \quad (\text{primal feasibility}) \quad (2.48b)$$

$$\lambda^* \geq \mathbf{o}, \quad (\text{dual feasibility}) \quad (2.48c)$$

$$(\lambda^*)^T (B\Theta^* + e) = 0. \quad (\text{complementarity}) \quad (2.48d)$$

Additionally, we can explicitly express the KKT conditions above for the SVM model parameters w^* and b^* so that:

$$\nabla_{w,b} \mathcal{L}(w^*, b^*, \lambda^*) = \mathbf{o}, \quad (\text{stacionarity}) \quad (2.49a)$$

$$\nabla_{\lambda} \mathcal{L}(w^*, b^*, \lambda^*) = \sum_{i=1}^m [1 - y_i (\langle w^*, x_i \rangle + b^*)] \leq 0, \quad (\text{primal feasibility}) \quad (2.49b)$$

$$\lambda^* \geq \mathbf{o}, \quad (\text{dual feasibility}) \quad (2.49c)$$

$$\sum_{i=1}^m \lambda_i^* [1 - y_i (\langle w^*, x_i \rangle + b^*)] = 0. \quad (\text{complementarity}) \quad (2.49d)$$

The first condition (2.48a) states that Θ^* , which denotes an optimal solution of a primal problem \mathcal{P} , is a minimizer of $\mathcal{L}(\cdot, \lambda^*)$ over Θ . The conditions (2.48b, 2.48c) bring out that both the primal and dual variables must satisfy to the constraints to their respective optimization problems. The last complementary condition (2.48d) represents a connection between the primal and dual variables. It implies that a product of the Lagrange multipliers and the corresponding primal constraints should be zero. This condition summarizes an idea that either the primal constraints are active (equal to zero), or the dual variables λ are zero for inactive constraints. We effectively use this condition for reconstructing a bias b from a dual solution later in the text.

Now, we determine a dual optimization problem related to $\mathcal{P}_{\text{hard}}^{\text{SVM}}$ (2.28) in a form where a Gram matrix⁹ is incorporated. First, we find out a stationary point of the Lagrangian (2.42) for a fixed λ . We do this by means of setting a gradient $\nabla_{\Theta} \mathcal{L}(\Theta, \lambda)$ to zero. In the case of an optimization problem related to SVM models, it is convenient to compute such gradient explicitly and component-wisely with respect to the SVM model parameters w and b . Note, the dual variable λ is fixed as we mentioned above.

⁹A Gram matrix is also known as a kernel matrix in machine learning community.

Recall, a stationary point of some function is a point, where all partial derivatives of this function equal 0. Let us start to compute a partial derivative of $\mathcal{L}(\mathbf{w}, b, \boldsymbol{\lambda})$ with respects to \mathbf{w} , and set this partial derivative to 0. Then, we obtain:

$$\frac{\partial \mathcal{L}(\mathbf{w}, b, \boldsymbol{\lambda})}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}} \left(\frac{1}{2} \|\mathbf{w}\|^2 \right) - \frac{\partial}{\partial \mathbf{w}} \sum_{i=1}^m \lambda_i [y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1] \quad (2.50a)$$

$$= \frac{\partial}{\partial \mathbf{w}} \left(\frac{1}{2} \|\mathbf{w}\|^2 \right) - \frac{\partial}{\partial \mathbf{w}} \sum_{i=1}^m \lambda_i y_i \langle \mathbf{w}, \mathbf{x}_i \rangle - \underbrace{\frac{\partial}{\partial \mathbf{w}} \sum_{i=1}^m \lambda_i y_i b}_{=0} + \underbrace{\frac{\partial}{\partial \mathbf{w}} \sum_{i=1}^m \lambda_i}_{=0} \quad (2.50b)$$

$$= \mathbf{w} - \sum_{i=1}^m \lambda_i y_i \mathbf{x}_i = \mathbf{o}, \quad (2.50c)$$

which implies the following relationship for the solution components \mathbf{w}^* and $\boldsymbol{\lambda}^*$:

$$\mathbf{w}^* \stackrel{\text{def}}{=} \sum_{i=1}^m \lambda_i^* y_i \mathbf{x}_i = \mathbf{X} \mathbf{Y}_d \boldsymbol{\lambda}. \quad (2.51)$$

Subsequently, a partial derivative of $\mathcal{L}(\mathbf{w}, b, \boldsymbol{\lambda})$ with respect to b and set to 0 results in:

$$\frac{\partial \mathcal{L}(\mathbf{w}, b, \boldsymbol{\lambda})}{\partial b} = \frac{\partial}{\partial b} \left(\frac{1}{2} \|\mathbf{w}\|^2 \right) - \frac{\partial}{\partial b} \left(\sum_{i=1}^m \lambda_i [y_i (\mathbf{w}^T \mathbf{x}_i + b)] - 1 \right) \quad (2.52a)$$

$$= \underbrace{\frac{\partial}{\partial b} \left(\frac{1}{2} \|\mathbf{w}\|^2 \right)}_{=0} - \underbrace{\frac{\partial}{\partial b} \sum_{i=1}^m \lambda_i y_i \langle \mathbf{w}, \mathbf{x}_i \rangle}_{=0} - \frac{\partial}{\partial b} \sum_{i=1}^m \lambda_i y_i b + \underbrace{\frac{\partial}{\partial b} \sum_{i=1}^m \lambda_i}_{=0} \quad (2.52b)$$

$$= \sum_{i=1}^m y_i \lambda_i = 0 \quad (2.52c)$$

implying:

$$\langle \mathbf{y}, \boldsymbol{\lambda}^* \rangle = 0. \quad (2.53)$$

Let us take into account a definition of a normal vector \mathbf{w} , which was previously introduced in (2.51), and the equality mentioned in (2.53). Inserting these relations back into a Lagrangian function (2.42), we get:

$$\mathcal{L}(\mathbf{w}^*, b^*, \boldsymbol{\lambda}^*) = \frac{1}{2} \langle \mathbf{w}^*, \mathbf{w}^* \rangle - \sum_{i=1}^m \lambda_i^* (y_i [\langle \mathbf{w}^*, \mathbf{x}_i \rangle + b^*] - 1) \quad (2.54a)$$

$$= \frac{1}{2} \|\mathbf{w}^*\|^2 - \sum_{i=1}^m \lambda_i^* (y_i [\langle \mathbf{w}^*, \mathbf{x}_i \rangle + b^*] - 1) \quad (2.54b)$$

$$= \frac{1}{2} \|\mathbf{w}^*\|^2 - \sum_{i=1}^m \lambda_i^* \left(y_i \left[\sum_{j=1}^m \lambda_j^* y_j \langle \mathbf{x}_j, \mathbf{x}_i \rangle + b^* \right] - 1 \right) \quad (2.54c)$$

$$= \frac{1}{2} \|\mathbf{w}^*\|^2 + \sum_{i=1}^m \lambda_i^* - \sum_{i=1}^m \sum_{j=1}^m \lambda_i^* y_i \lambda_j^* y_j \langle \mathbf{x}_j, \mathbf{x}_i \rangle + \underbrace{\sum_{i=1}^m b^* \lambda_i^* y_i}_{=0 \text{ from (2.53)}} \quad (2.54d)$$

$$= \frac{1}{2} \langle \mathbf{w}^*, \mathbf{w}^* \rangle + \sum_{i=1}^m \lambda_i^* - \sum_{i=1}^m \sum_{j=1}^m \lambda_i^* y_i \lambda_j^* y_j \langle \mathbf{x}_j, \mathbf{x}_i \rangle \quad (2.54e)$$

$$= \underbrace{\frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \lambda_i^* y_i \lambda_j^* y_j \langle \mathbf{x}_j, \mathbf{x}_i \rangle}_{\text{using definition of } \mathbf{w} \text{ in (2.51)}} + \sum_{i=1}^m \lambda_i^* - \sum_{i=1}^m \sum_{j=1}^m \lambda_i^* y_i \lambda_j^* y_j \langle \mathbf{x}_j, \mathbf{x}_i \rangle \quad (2.54f)$$

$$= \sum_{i=1}^m \lambda_i^* - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \lambda_i^* y_i \langle \mathbf{x}_i, \mathbf{x}_j \rangle \lambda_j^* y_j. \quad (2.54g)$$

Recall. We get (2.54g) by minimizing a Lagrangian function (2.42). Putting it together with constraints $\boldsymbol{\lambda} \geq 0$ and (2.53), we get the following dual optimization problem:

$$\boldsymbol{\lambda}^* = \arg \max_{\boldsymbol{\lambda}} \sum_{i=1}^m \lambda_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \lambda_i y_i \langle \mathbf{x}_i, \mathbf{x}_j \rangle \lambda_j y_j \text{ s.t. } \begin{cases} \mathbf{o} \leq \boldsymbol{\lambda}, \\ \langle \mathbf{y}, \boldsymbol{\lambda} \rangle = 0, \end{cases} \quad (2.55a)$$

$$= \arg \min_{\boldsymbol{\lambda}} \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \lambda_i y_i \langle \mathbf{x}_i, \mathbf{x}_j \rangle \lambda_j y_j - \sum_{i=1}^m \lambda_i \text{ s.t. } \begin{cases} \mathbf{o} \leq \boldsymbol{\lambda}, \\ \langle \mathbf{y}, \boldsymbol{\lambda} \rangle = 0. \end{cases} \quad (2.55b)$$

Finally, we can put a dual formulation (2.55b) into a matrix form such that:

$$(\mathcal{D}_{\text{hard}}^{\text{SVM}}) : \boldsymbol{\lambda}^* = \arg \min_{\boldsymbol{\lambda}} \frac{1}{2} \boldsymbol{\lambda}^T \mathbf{Q} \boldsymbol{\lambda} - \mathbf{e}^T \boldsymbol{\lambda} \text{ s.t. } \begin{cases} \mathbf{o} \leq \boldsymbol{\lambda}, \\ \mathbf{B}_E \boldsymbol{\lambda} = 0, \end{cases} \quad (2.56)$$

where

$$\mathbf{Q} = \mathbf{Y}_d^T \mathbf{X}^T \mathbf{X} \mathbf{Y}_d, \quad \mathbf{B}_E = [\mathbf{y}^T]. \quad (2.57)$$

After the dual problem $\mathcal{D}_{\text{hard}}^{\text{SVM}}$ (2.56) is solved, we need to obtain the original model parameters \mathbf{w}^* and b^* using the Lagrange multipliers $\boldsymbol{\lambda}^*$. Thus, we will introduce so-called dual-primal reconstruction formulas. The first one is associated with the normal vector \mathbf{w} of the hyperplane H . It directly arises from stationary condition (2.48a), specifically from the derivative of the Lagrangian with respects to \mathbf{w} (2.50a) such that the reconstruction formula has a form (2.51).

For reconstruction of the bias b^* , we can effectively use the definition of support vectors. Let us first analyze the complementary condition (2.48d) from the KKT system to determine which training samples from $\mathbb{X}_{\text{train}}$ are actually support vectors. Looking at this condition, we can see that either the Lagrange multiplier λ_i or the margin $y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle + b)$ has to equal

zero for all indexes $i \in \{1, 2, \dots, m\}$. If $\lambda_i = 0$ then $y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1$, it follows that related sample \mathbf{x}_i lies on or above geometric margin $\gamma(\mathbf{w}, b, \mathbb{X}_{\text{train}}) = 1$ and thus it can be a support vector. This does not really help us.

Let us focus on the second case. If $\lambda_i > 0$ then $y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) = 1$, it means that the sample \mathbf{x}_i lies on the margin and implies the sample \mathbf{x}_i is the support vector, which we denote as \mathbf{x}_{sv} . We can effectively use the definition of support vector \mathbf{x}_{sv} and related label y_{sv} for reconstruction of the bias b^* :

$$1 = y_{\text{sv}} [\langle \mathbf{w}^*, \mathbf{x}_{\text{sv}} \rangle + b^*], \quad (2.58a)$$

$$\frac{1}{y_{\text{sv}}} = \langle \mathbf{w}^*, \mathbf{x}_{\text{sv}} \rangle + b^*, \quad (2.58b)$$

$$b^* = \frac{1}{y_{\text{sv}}} - \langle \mathbf{w}^*, \mathbf{x}_{\text{sv}} \rangle, \quad (2.58c)$$

$$b^* = y_{\text{sv}} - \langle \mathbf{w}^*, \mathbf{x}_{\text{sv}} \rangle, \quad (\text{assuming } y_{\text{sv}} \in \{-1, 1\}) \quad (2.58d)$$

$$b^* = y_{\text{sv}} - \langle \mathbf{X} \mathbf{Y}_d \boldsymbol{\lambda}^*, \mathbf{x}_{\text{sv}} \rangle. \quad (2.58e)$$

In practice, a different reconstruction formula is often employed to determine a bias b^* associated with a hyperplane H . Instead of considering a particular support vector \mathbf{x}_{sv} , we run calculation over all support vectors so that:

$$b^* = \frac{1}{\text{card}(\mathbb{I}_{\text{SV}})} \left(\mathbf{X}_{*\mathbb{I}_{\text{SV}}}^T \mathbf{w} - \mathbf{y}_{\mathbb{I}_{\text{SV}}} \right)^T \mathbf{e}_{\mathbb{I}_{\text{SV}}}, \quad (2.59)$$

where \mathbb{I}_{SV} is the support vector index set, which is defined as follows:

$$\mathbb{I}_{\text{SV}} \stackrel{\text{def}}{=} \{j : 0 < \lambda_j, j = 1, 2, \dots, k\}. \quad (2.60)$$

The cardinality of \mathbb{I}_{SV} is denoted as $\text{card}(\mathbb{I}_{\text{SV}})$ and it represents a number of support vectors. A matrix $\mathbf{X}_{*\mathbb{I}_{\text{SV}}}$ denotes a submatrix of a matrix

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m] \quad (2.61)$$

with the column indices belonging to \mathbb{I}_{SV} . Additionally, $\mathbf{y}_{\mathbb{I}_{\text{SV}}}$ and $\mathbf{e}_{\mathbb{I}_{\text{SV}}}$ are subvectors of the vectors \mathbf{y} and \mathbf{e} , respectively. More detailed discussion about solvability of this dual problem can be found in [39].

Chapter 3

Soft-margin support vector machines

We have introduced a theoretical framework of maximal-hard-margin classifiers related to linearly separable training samples based on the Vapnik-Chervonenkis dimension in the Chapter 2 so far. However, this is not commonly useful for training models in practical applications, because real-world samples are not (inherently) linearly separable. This chapter outlines an approach that overcomes this issue. **It employs soft-margin SVMs for training linear models on data sets that consists of non-linearly separable samples.** There are discussed two variants associated with soft-margin SVMs, specifically ℓ_1 -loss and ℓ_2 -loss SVM in Section 3.1 and Section 3.2, respectively. In Section 3.3, relaxed-bias approaches for soft-margin SVMs above are introduced.

Recall. SVMs are basically a set of methods, which belong to supervised learning algorithms used for classification, regression, or outlier detection. Note that we focus only on classification models in this thesis. Considering their underlying structures, we can see SVM as a single-layer perceptron that finds a learning function that maximizes a geometric margin between training samples and a separating hyperplane. This implicit capability guarantees a generalization performance of a model. It was earlier proven using an upper bound of the Vapnik-Chervonenkis dimension. We can also describe this generalization performance by means of a particular case of Tikhonov regularization in the following form:

$$\arg \min_{f \in \mathcal{H}} m^{-1} \sum_{i=1}^m V(y_i, f(\mathbf{x}_i))^2 + \alpha \|f\|_{\mathcal{H}}^2. \quad (3.1)$$

There, \mathcal{H} is a hypothesis space of learning functions, $\|\cdot\|_{\mathcal{H}}$ represents a norm on this hypothesis space, $f : \mathbb{R}^n \rightarrow \mathbb{Y}$ denotes a learning function (model) that maps training samples from $\mathbb{X}_{\text{train}}$ to a label space \mathbb{Y} , and $m = \text{card}(\mathbb{X}_{\text{train}})$. A loss function is represented by means of $V : \mathbb{Y} \rightarrow \mathbb{Y}$ and $\alpha \in \mathbb{R}_+$ serves as a regularization parameter such that:

$$\alpha = \frac{1}{2C}. \quad (3.2)$$

Moreover, this theoretical framework provides us with an explanation related to the regularization perspective of the SVM models such that a trade-off between bias and variance is driven by parameter $C \in \mathbb{R}_+$.

3.1 Soft-margin ℓ_1 -loss linear classifiers

A concept of a maximal-margin classifier based on a soft-margin approach was introduced in 1995 by Cortes and Vapnik [43]. This approach was originally developed as a supervised binary classifier and is widely known as a soft-margin SVM in ML community. Recall. An underlying model related to a linear SVM is given by:

$$h_{\Theta}(\mathbf{x}) = \langle \Theta_0, \mathbf{x} \rangle + \Theta_1 = \langle \mathbf{w}, \mathbf{x} \rangle + \hat{b}, \quad (3.3)$$

where $\mathbf{w} \neq \mathbf{o}$ is a normal vector of the hyperplane H . A scalar $\hat{b} = \frac{b}{\|\mathbf{w}\|}$ is called a bias that determines a displacement of a hyperplane from an origin in the direction of \mathbf{w} . Let us denote \hat{b} as b for a convincing reading of the following text.

Let us first introduce a training set $\mathbb{X}_{\text{train}}$ as a set of sample-label pairs, where an ordering is not mandatory:

$$\mathbb{X}_{\text{train}} \stackrel{\text{def}}{=} \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}, \quad (3.4)$$

where $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\} \subset \mathbb{R}^n$.

Recall. The separating hyperplane H is defined using the geometric margins, depicted as dashed lines in Figure 3.1. The margins are determined using the locations of support vectors $(\mathbf{x}_{\text{sv}}, y_{\text{sv}})$, which are samples from a training data set $\mathbb{X}_{\text{train}}$ such that the components of \mathbf{x}_{sv} lie on the these margins. A width between them equals $\frac{2}{\|\mathbf{w}\|}$.

Unlike the assumptions on a training data set in the previous chapter, we do not now consider that the samples are linearly separable. Considering this assumption, a solution associated with a hard-margin problem (introduced in Section 2.2) could not generally exist. To sort out the issue related to the non-existence of a solution, we allow some training samples to be misclassified. For this, we employ a so-called *soft-margin approach*. The idea is based on adding an auxiliary (regularization) term:

$$C \sum_{i=1}^m \xi_i \quad (3.5)$$

to the hard-margin formulation,¹ and an additional relaxation of the constraints associated

¹The hard-margin formulation was introduced in Section 2.2 in a formulation (2.26d).

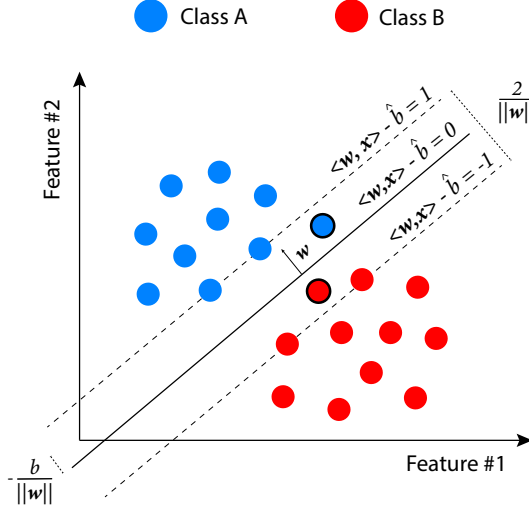


Figure 3.1: A case of penalized test samples: the encircled (test) samples set are correctly classified. However, they are on the wrong side of their related margin.

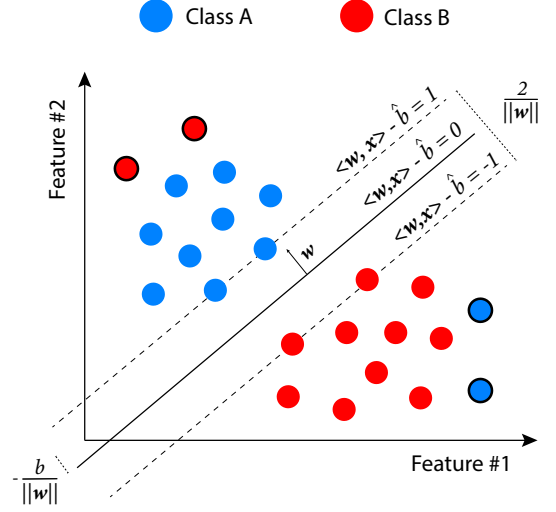


Figure 3.2: A case of misclassifying (test) samples: the encircled samples are not correctly classified.

with geometric margins such that:

$$(\mathbf{w}^*, b^*, \xi^*) = \arg \min_{\mathbf{w}, b, \xi} \frac{1}{2} \langle \mathbf{w}, \mathbf{w} \rangle + C \sum_{i=1}^m \xi_i \text{ s.t. } \begin{cases} y_i [\langle \mathbf{w}, \mathbf{x}_i \rangle + b] \geq 1 - \xi_i, \\ \xi_i \geq 0, i \in \{1, 2, \dots, m\}. \end{cases} \quad (3.6)$$

The term $C \sum_{i=1}^m \xi_i$ regularizes (possible) misclassification errors and restricts a complexity of a classifier in the sense of overfitting an underlying model, where ξ_i is a hinge loss function associated with each sample \mathbf{x}_i . We define this hinge loss function ξ_i for a sample \mathbf{x}_i such that:

$$\xi_i \stackrel{\text{def}}{=} \max(0, 1 - y_i [\langle \mathbf{w}, \mathbf{x}_i \rangle - b]). \quad (3.7)$$

Basically, a hinge loss function ξ_i quantifies an error between a predicted and correct classification of a sample \mathbf{x}_i . If the sample \mathbf{x}_i is correctly classified, a value of a related hinge loss function ξ_i equals 0. In order of sample misclassification, a value of ξ is proportional to the distance between a respective geometric margin and a misclassified sample. We can observe:

if $0 \leq \xi_i \leq 1$, then the i -th sample lies somewhere between a margin $\gamma = 1$ and separating hyperplane – illustrated in Figure 3.1; a sample is penalized in this case.

If $\xi_i > 1$, then i -th sample is misclassified – depicted in Figure 3.2.

The variable $C \in \mathbb{R}^+$ is a parameter that penalizes a misclassification error. A higher value of C increases the importance of minimizing the hinge loss functions ξ_i and the importance

of minimizing $\|\mathbf{w}\|$ at the expense of satisfying the margin constraint for fewer samples. On the other hand, it causes maximizing $\|\mathbf{w}\|$, i.e. minimizing the width of a margin, leading to poor generalization capabilities of a classification model.

A goal is to find a reasonable value of this parameter C such that a resulting model balances between robustness and complexity – a bias-variance trade-off. The parameter is user-defined or determined using hyperparameter optimization (HyperOpt) techniques, e.g. grid-search combined with cross-validation; these techniques are introduced in Chapter 4. Now, let us mention a few references where authors pointed out an importance of a parameter C :

- “In the support-vector networks algorithm one can control the trade-off between complexity of decision rule and frequency of error by changing the parameter C , ...” [43]
- “The parameter C controls the trade-off between errors of SVM on training data and margin maximization ($C \rightarrow \infty$ leads to hard-margin SVM).” [44, p. 82]
- “...the coefficient C affects the trade-off between complexity and proportion of nonseparable samples and must be selected by the user.” [45, p. 366]

The optimization problem introduced in (3.6) is commonly known as a primal ℓ_1 -loss SVM. Let $\Theta_\xi^* \in \mathbb{R}^{n+m+1}$ be a solution vector associated with an optimization problem (3.6). Now, let us introduce a primal ℓ_1 -loss SVM in a matrix form so that:

$$(\mathcal{P}_{\ell_1}^{\text{SVM}}) : \Theta_\xi^* = \arg \min_{\Theta_\xi} \frac{1}{2} \Theta_\xi^T \mathbf{Q}_{\ell_1} \Theta_\xi + \Theta_\xi^T \mathbf{q} \text{ s.t. } \mathbf{B}_{\ell_1} \Theta_\xi \leq -\mathbf{h}, \quad (3.8)$$

where

$$\mathbf{Q}_{\ell_1} = \begin{bmatrix} \mathbf{I}_{n,n} & \mathbf{O}_{n,m+1} \\ \mathbf{O}_{m+1,n} & \mathbf{O}_{m+1,m+1} \end{bmatrix}, \quad \mathbf{B}_{\ell_1} = \begin{bmatrix} \mathbf{B} & -\mathbf{I}_m \\ \mathbf{O}_{m,m+1} & -\mathbf{I}_m \end{bmatrix}, \quad \mathbf{B} = [-\mathbf{Y}_d \mathbf{X}^T, -\mathbf{y}], \quad (3.9)$$

and

$$\mathbf{q} = [\mathbf{o}_{m+1}^T, C \mathbf{e}_m^T]^T, \quad \mathbf{h} = [\mathbf{e}_m^T, \mathbf{o}_m^T]^T, \quad \Theta_\xi = [\mathbf{w}^T, b, \boldsymbol{\xi}^T]^T. \quad (3.10)$$

Looking at a primal formulation $\mathcal{P}_{\ell_1}^{\text{SVM}}$, we can see that inequality constraints take a form, which consists of an inequality matrix \mathbf{B} . Recall. The QP API implemented in PERMON is designed for box (also lower and upper bounds), and equality constraints. Thus, we need to transform a primal formulation $\mathcal{P}_{\ell_1}^{\text{SVM}}$ into a QP problem having those types of constraints. For this, we exploit an approach based on the Lagrange duality, which we effectively used earlier in Section 2.3.

First, we set up a Lagrangian function for a primal optimization problem in (3.6), where we consider primal variables as components of a vector Θ_ξ , i.e. \mathbf{w} , b , $\boldsymbol{\xi}$, and vectors of the

Lagrange multipliers $\boldsymbol{\lambda}$ and $\boldsymbol{\lambda}^\xi$ as follows:

$$\begin{aligned} \mathcal{L}_{\ell 1}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\lambda}, \boldsymbol{\lambda}^\xi) &= \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m \xi_i - \sum_{i=1}^m \lambda_i (y_i [\langle \mathbf{w}, \mathbf{x}_i \rangle + b] - 1 + \xi_i) \\ &\quad - \sum_{i=1}^m \lambda_i^\xi \xi_i. \end{aligned} \quad (3.11)$$

Additionally, we can express the KKT conditions exploiting \mathbf{w}^* , b^* , $\boldsymbol{\xi}^*$, and the vectors of the Lagrange multipliers $\boldsymbol{\lambda}^*$ and $\boldsymbol{\lambda}^{\xi*}$ so that:

$$\nabla_{\mathbf{w}, b, \boldsymbol{\xi}} \mathcal{L}_{\ell 1}(\mathbf{w}^*, b^*, \boldsymbol{\xi}^*, \boldsymbol{\lambda}^*, \boldsymbol{\lambda}^{\xi*}) = \mathbf{o}, \quad (\text{stacionarity}) \quad (3.12a)$$

$$\left. \begin{aligned} \sum_{i=0}^m (1 - \xi_i^* + y_i [\langle \mathbf{w}^*, \mathbf{x}_i \rangle + b^*]) &\leq 0, \\ -\boldsymbol{\xi}^* &\leq \mathbf{o}, \end{aligned} \right\} \quad (\text{primal feasibility}) \quad (3.12b)$$

$$\left. \begin{aligned} \boldsymbol{\lambda}^* &\geq \mathbf{o}, \\ \boldsymbol{\lambda}^{\xi*} &\geq \mathbf{o}, \end{aligned} \right\} \quad (\text{dual feasibility}) \quad (3.12c)$$

$$\left. \begin{aligned} \sum_{i=0}^m \lambda_i^* (1 - \xi_i^* + y_i [\langle \mathbf{w}^*, \mathbf{x}_i \rangle + b^*]) &= 0, \\ \sum_{i=0}^m \lambda_i^{\xi*} \xi_i^* &= 0. \end{aligned} \right\} \quad (\text{complementarity}) \quad (3.12d)$$

In the following text, we determine a dual optimization problem $\mathcal{D}_{\ell 1}^{\text{SVM}}$ corresponding to a primal problem $\mathcal{P}_{\ell 1}^{\text{SVM}}$. First, we identify a stationary point of the Lagrangian (3.11) for fixed $\boldsymbol{\lambda}$ and $\boldsymbol{\lambda}^\xi$. Recall a stationary point of a function is a point, where all partial derivatives of the function are equal to zero. To begin, we compute a partial derivative of $\mathcal{L}_{\ell 1}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\lambda}, \boldsymbol{\lambda}^\xi)$ with respects to \mathbf{w} , and set this partial derivative to 0. This yields:

$$\begin{aligned} \frac{\partial \mathcal{L}_{\ell 1}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\lambda}, \boldsymbol{\lambda}^\xi)}{\partial \mathbf{w}} &= \frac{\partial}{\partial \mathbf{w}} \left(\frac{1}{2} \|\mathbf{w}\|^2 \right) + \underbrace{C \frac{\partial}{\partial \mathbf{w}} \sum_{i=1}^m \xi_i}_{=0} \\ &\quad - \frac{\partial}{\partial \mathbf{w}} \sum_{i=1}^m \lambda_i (y_i [\langle \mathbf{w}, \mathbf{x}_i \rangle + b] - 1 + \xi_i) - \underbrace{\frac{\partial}{\partial \mathbf{w}} \sum_{i=1}^m \lambda_i^\xi \xi_i}_{=0} \end{aligned} \quad (3.13a)$$

$$\begin{aligned} &= \frac{\partial}{\partial \mathbf{w}} \left(\frac{1}{2} \|\mathbf{w}\|^2 \right) - \frac{\partial}{\partial \mathbf{w}} \sum_{i=1}^m \lambda_i y_i \langle \mathbf{w}, \mathbf{x}_i \rangle - \underbrace{\frac{\partial}{\partial \mathbf{w}} \sum_{i=1}^m \lambda_i y_i b}_{=0} \\ &\quad + \underbrace{\frac{\partial}{\partial \mathbf{w}} \sum_{i=1}^m \lambda_i}_{=0} - \underbrace{\frac{\partial}{\partial \mathbf{w}} \sum_{i=1}^m \lambda_i \xi_i}_{=0} \end{aligned} \quad (3.13b)$$

$$= \mathbf{w} - \sum_{i=1}^m \lambda_i y_i \mathbf{x}_i = \mathbf{o} \quad (3.13c)$$

which implies the following relationship for a normal vector \mathbf{w}^* of a hyperplane H :

$$\mathbf{w}^* = \sum_{i=1}^m \lambda_i y_i \mathbf{x}_i = \mathbf{X} \mathbf{Y}_d \boldsymbol{\lambda}^*. \quad (3.14)$$

Following this, we set the partial derivative of $\mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\lambda}, \boldsymbol{\lambda}^\xi)$ with respect to b to zero, which results in:

$$\frac{\partial \mathcal{L}_{\ell_1}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\lambda}, \boldsymbol{\lambda}^\xi)}{\partial b} = \underbrace{\frac{\partial}{\partial b} \left(\frac{1}{2} \|\mathbf{w}\|^2 \right)}_{=0} + C \underbrace{\frac{\partial}{\partial b} \sum_{i=1}^m \xi_i}_{=0} \quad (3.15a)$$

$$- \frac{\partial}{\partial b} \sum_{i=1}^m \lambda_i (y_i [\langle \mathbf{w}, \mathbf{x}_i \rangle + b] - 1 + \xi_i) - \underbrace{\frac{\partial}{\partial b} \sum_{i=1}^m \lambda_i^\xi \xi_i}_{=0}$$

$$= \underbrace{\frac{\partial}{\partial b} \sum_{i=1}^m \lambda_i y_i \langle \mathbf{w}, \mathbf{x}_i \rangle}_{=0} - \frac{\partial}{\partial b} \sum_{i=1}^m \lambda_i y_i b + \underbrace{\frac{\partial}{\partial b} \sum_{i=1}^m \lambda_i}_{=0} - \underbrace{\frac{\partial}{\partial b} \sum_{i=1}^m \lambda_i \xi_i}_{=0} \quad (3.15b)$$

$$= \sum_{i=1}^m y_i \lambda_i = \langle \mathbf{y}, \boldsymbol{\lambda} \rangle = 0. \quad (3.15c)$$

By computing the partial derivative of $\mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\lambda}, \boldsymbol{\lambda}^\xi)$ with respect to $\boldsymbol{\xi}$ and subsequently set it to zero, we derive:

$$\frac{\partial \mathcal{L}_{\ell_1}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\lambda}, \boldsymbol{\lambda}^\xi)}{\partial \boldsymbol{\xi}} = \underbrace{\frac{\partial}{\partial \boldsymbol{\xi}} \left(\frac{1}{2} \|\mathbf{w}\|^2 \right)}_{=0} + C \frac{\partial}{\partial \boldsymbol{\xi}} \sum_{i=1}^m \xi_i \quad (3.16a)$$

$$- \frac{\partial}{\partial \boldsymbol{\xi}} \sum_{i=1}^m \lambda_i (y_i [\langle \mathbf{w}, \mathbf{x}_i \rangle + b] - 1 + \xi_i) - \frac{\partial}{\partial \boldsymbol{\xi}} \sum_{i=1}^m \lambda_i^\xi \xi_i$$

$$= C \frac{\partial}{\partial \boldsymbol{\xi}} \sum_{i=1}^m \xi_i - \underbrace{\frac{\partial}{\partial \boldsymbol{\xi}} \sum_{i=1}^m \lambda_i y_i [\langle \mathbf{w}, \mathbf{x}_i \rangle + b]}_{=0} + \underbrace{\frac{\partial}{\partial \boldsymbol{\xi}} \sum_{i=1}^m \lambda_i}_{=0} \quad (3.16b)$$

$$- \frac{\partial}{\partial \boldsymbol{\xi}} \sum_{i=1}^m \lambda_i \xi_i - \frac{\partial}{\partial \boldsymbol{\xi}} \sum_{i=1}^m \lambda_i^\xi \xi_i$$

$$= C \mathbf{e} - \frac{\partial}{\partial \boldsymbol{\xi}} \sum_{i=1}^m \lambda_i \xi_i - \boldsymbol{\lambda}^\xi \quad (3.16c)$$

$$= C \mathbf{e} - \boldsymbol{\lambda} - \boldsymbol{\lambda}^\xi = \mathbf{o}, \quad (3.16d)$$

which implies the following relationship for the vector of Lagrange multipliers $\boldsymbol{\lambda}^\xi$:

$$\boldsymbol{\lambda}^\xi = C \mathbf{e} - \boldsymbol{\lambda}, \quad (3.17)$$

or:

$$C\mathbf{e} = \boldsymbol{\lambda}^\xi + \boldsymbol{\lambda}. \quad (3.18)$$

To derive the dual functional, we eliminate \mathbf{w} , b , and $\boldsymbol{\xi}$ from $\mathcal{L}_{\ell 1}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\lambda}, \boldsymbol{\lambda}^\xi)$ using (3.14), (3.15c), and (3.18) as follows:

$$\begin{aligned} \mathcal{L}_{\ell 1}(\mathbf{w}^*, b^*, \boldsymbol{\xi}^*, \boldsymbol{\lambda}^*, \boldsymbol{\lambda}^{\xi*}) &= \frac{1}{2} \|\mathbf{w}^*\|^2 + C \sum_{i=1}^m \xi_i^* - \sum_{i=1}^m \lambda_i^* (y_i [\langle \mathbf{w}^*, \mathbf{x}_i \rangle + b^*] - 1 + \xi_i^*) \\ &\quad - \sum_{i=1}^m \lambda_i^{\xi*} \xi_i^* \end{aligned} \quad (3.19a)$$

$$\begin{aligned} &= \frac{1}{2} \|\mathbf{w}^*\|^2 + C \sum_{i=1}^m \xi_i^* - \sum_{i=1}^m \lambda_i^* y_i [\langle \mathbf{w}^*, \mathbf{x}_i \rangle + b^*] + \sum_{i=1}^m \lambda_i^* \\ &\quad - \sum_{i=1}^m \lambda_i^* \xi_i^* - \sum_{i=1}^m \lambda_i^{\xi*} \xi_i^* \end{aligned} \quad (3.19b)$$

$$\begin{aligned} &= \frac{1}{2} \|\mathbf{w}^*\|^2 + \sum_{i=1}^m \underbrace{(\lambda_i^* + \lambda_i^{\xi*})}_{\text{using (3.18)}} \xi_i^* \\ &\quad - \sum_{i=1}^m \lambda_i^* y_i \underbrace{\sum_{j=1}^m \lambda_j^* y_j \langle \mathbf{x}_j, \mathbf{x}_i \rangle}_{\text{using definition of } \mathbf{w} \text{ in (3.14)}} + \sum_{i=1}^m \lambda_i^* y_i b^* + \sum_{i=1}^m \lambda_i^* \end{aligned} \quad (3.19c)$$

$$\begin{aligned} &\quad - \sum_{i=1}^m (\lambda_i^* + \lambda_i^{\xi*}) \xi_i^* \\ &= \frac{1}{2} \langle \mathbf{w}^*, \mathbf{w}^* \rangle + \sum_{i=1}^m \lambda_i^* - \sum_{i=1}^m \sum_{j=1}^m \lambda_i^* y_i \lambda_j^* y_j \langle \mathbf{x}_j, \mathbf{x}_i \rangle \\ &\quad - \underbrace{\sum_{i=1}^m \lambda_i^* y_i b^*}_{\substack{=0 \\ \text{from (3.15c)}}} \end{aligned} \quad (3.19d)$$

$$= \sum_{i=1}^m \lambda_i^* - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \lambda_i^* y_i \langle \mathbf{x}_i, \mathbf{x}_j \rangle \lambda_j^* y_j. \quad (3.19e)$$

It is important to note that the dual functional is independent of $\boldsymbol{\lambda}^\xi$.

Now, we can define constraints for the Lagrange multipliers $\boldsymbol{\lambda}$. According to their definition, a lower bound for the Lagrange multipliers is set to 0, i.e. $\boldsymbol{\lambda} \geq 0$ and $\boldsymbol{\lambda}^\xi \geq 0$. Following this and (3.18), we can establish an upper bound for $\boldsymbol{\lambda}$ as follows:

$$\boldsymbol{\lambda} \stackrel{(3.18)}{=} C\mathbf{e} - \boldsymbol{\lambda}^\xi \stackrel{(3.12c)}{\leq} C\mathbf{e}, \quad (3.20)$$

an thus:

$$\mathbf{o} \leq \boldsymbol{\lambda} \leq C\mathbf{e}. \quad (3.21)$$

The multiplier $\boldsymbol{\lambda}$ also satisfies the equality constraint arises from (3.15c):

$$\langle \mathbf{y}, \boldsymbol{\lambda}^* \rangle = 0. \quad (3.22)$$

Finally, we obtain the following dual optimization problem:

$$\boldsymbol{\lambda}^* = \arg \max_{\boldsymbol{\lambda}} \sum_{i=1}^m \lambda_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \lambda_i y_i \langle \mathbf{x}_i, \mathbf{x}_j \rangle \lambda_j y_j \quad \text{s.t.} \quad \begin{cases} \mathbf{o} \leq \boldsymbol{\lambda} \leq C\mathbf{e}, \\ \langle \mathbf{y}, \boldsymbol{\lambda} \rangle = 0, \end{cases} \quad (3.23a)$$

$$= \arg \min_{\boldsymbol{\lambda}} \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \lambda_i y_i \langle \mathbf{x}_i, \mathbf{x}_j \rangle \lambda_j y_j - \sum_{i=1}^m \lambda_i \quad \text{s.t.} \quad \begin{cases} \mathbf{o} \leq \boldsymbol{\lambda} \leq C\mathbf{e}, \\ \langle \mathbf{y}, \boldsymbol{\lambda} \rangle = 0. \end{cases} \quad (3.23b)$$

We can put a dual formulation (3.23b) into a matrix form such that:

$$(\mathcal{D}_{\ell_1}^{\text{SVM}}) : \boldsymbol{\lambda}^* = \arg \min_{\boldsymbol{\lambda}} \frac{1}{2} \boldsymbol{\lambda}^T \mathbf{Q}_{\ell_1} \boldsymbol{\lambda} - \mathbf{e}^T \boldsymbol{\lambda} \quad \text{s.t.} \quad \begin{cases} \mathbf{o} \leq \boldsymbol{\lambda} \leq C\mathbf{e}, \\ \mathbf{B}_E \boldsymbol{\lambda} = 0, \end{cases} \quad (3.24)$$

where:

$$\mathbf{Q}_{\ell_1} = \mathbf{Y}_d^T \mathbf{X}^T \mathbf{X} \mathbf{Y}_d, \quad \mathbf{B}_E = [\mathbf{y}^T]. \quad (3.25)$$

After a dual problem $\mathcal{D}_{\ell_1}^{\text{SVM}}$ (3.24) is solved,² we need to obtain primal variables associated with the parameters of a linear model \mathbf{w}^* , and b^* . The normal vector \mathbf{w}^* of hyperplane H can be determined from (3.14) such that:

$$\mathbf{w}^* = \mathbf{X} \mathbf{Y}_d \boldsymbol{\lambda}^*. \quad (3.26)$$

For reconstruction of the bias b , we need first to determine support vectors employing complementary conditions outlined in (3.12d). Let us recall them:

$$\lambda_i (1 - \xi_i + y_i [\langle \mathbf{w}, \mathbf{x}_i \rangle + b]) = 0, \quad i = 1, 2, \dots, m, \quad (3.27a)$$

$$\lambda_i^\xi \xi_i = \sum_{i=1}^m (C - \lambda_i) \xi_i = 0, \quad i = 1, 2, \dots, m. \quad (3.27b)$$

Looking at these conditions, we can see that we need to analyze three cases corresponding to the possible values of the Lagrange multiplier λ_i :

²Note, the Lagrange multipliers $\boldsymbol{\lambda}^\xi$ are determined by means of an equation (3.17) using the Lagrange multipliers $\boldsymbol{\lambda}$.

-
- When $\lambda_i = 0$, we can evaluate the complementary conditions (3.27) as follows:

- a condition (3.27a) is equivalent to $(C - 0)\xi_i = 0$ and hence $\xi_i = 0$,
- this implies that a condition (3.12b) yields $0(1 - 0 - y_i[\langle \mathbf{w}, \mathbf{x}_i \rangle + b]) = 0$, and then $y_i[\langle \mathbf{w}, \mathbf{x}_i \rangle + b] \geq 1$.

From the analysis above, we can conclude that a sample \mathbf{x}_i is correctly classified and thus lies on or above a respective geometric margin. Thus, a sample \mathbf{x}_i associated with a Lagrange multiplier λ_i could be a support vector.

- For $0 < \lambda_i < C$, we obtain:

- a condition (3.27b) is equivalent to $\underbrace{(C - \lambda_i)}_{>0}\xi_i = 0$ and hence $\xi_i = 0$,
- this consequently implies that the condition (3.27a) equals $\lambda_i(1 - 0 - y_i[\langle \mathbf{w}, \mathbf{x}_i \rangle + b]) = 0$, which results in $y_i[\langle \mathbf{w}, \mathbf{x}_i \rangle + b] = 1$.

Now, we can conclude that a sample \mathbf{x}_i is correctly classified and lies on a respective geometric margin. Thus, a sample \mathbf{x}_i associated with a Lagrange multiplier is a support vector.

- If $\lambda_i = C$:

- a condition (3.27a) is equivalent to $(C - C)\xi_i = 0$ and then $\xi_i \geq 0$ that follows from (3.12b),
- this implies that a condition (3.27a) equals $C(1 - \xi_i - y_i[\langle \mathbf{w}, \mathbf{x}_i \rangle + b]) = 0$, and then $y_i[\langle \mathbf{w}, \mathbf{x}_i \rangle + b] = 1 - \xi_i$.

Hence $y_i[\langle \mathbf{w}, \mathbf{x}_i \rangle + b] \leq 1$ and thus the sample \mathbf{x}_i is either misclassified or lying on the margin ($\xi_i = 0$).

To be really sure that we select a set of support vectors, we consider samples for which Lagrange multipliers falling within the range from 0 to C , i.e. $0 < \lambda_i < C$. Considering these support vectors, we can reconstruct a bias of a hyperplane H using the following reconstruction formula:

$$b^* = \frac{1}{\text{card}(\mathbb{I}_{\text{SV}})} \left(\mathbf{X}_{*\mathbb{I}_{\text{SV}}}^T \mathbf{w} - \mathbf{y}_{\mathbb{I}_{\text{SV}}} \right)^T \mathbf{e}_{\mathbb{I}_{\text{SV}}}, \quad (3.28)$$

where \mathbb{I}_{SV} is the support vector index set, which is defined as follows:

$$\mathbb{I}_{\text{SV}} \stackrel{\text{def}}{=} \{j : 0 < \lambda_j < C, j = 1, 2, \dots, m\}. \quad (3.29)$$

Now, let us analyze a rank of the Hessian matrix $\mathbf{Q}_{\ell_1} = \mathbf{Y}_d^T \mathbf{X}^T \mathbf{X} \mathbf{Y}_d$ in a dual ℓ_1 -loss SVM (3.24). For this, we need to first focus on the Gram matrix $\mathbf{X}^T \mathbf{X}$. Using the rank-nullity

theorem, we can obtain the following:

$$\text{rank}(\mathbf{X}^T \mathbf{X}) = \text{rank}(\mathbf{X}), \quad (3.30)$$

where \mathbf{X} is a data matrix. Analyzing this matrix \mathbf{X} , we can conclude that:

$$\text{rank}(\mathbf{X}) = \min\{n, N_g\}, \quad (3.31)$$

where N_g is a maximum number of linearly independent training samples, m and n is a number of training samples and their features, respectively. Thus, a Gram matrix is symmetric positive semidefinite (SPS). Further, the Hessian $\mathbf{Q}_{\ell 1}$ is also SPS either, since \mathbf{Y}_d is a diagonal matrix having -1 and 1 values on a main diagonal. More detailed discussion about solvability of the primal and dual $\ell 1$ -loss SVMs can be found in [39].

3.2 Soft-margin $\ell 2$ -loss linear classifiers

The Hessian matrix $\mathbf{Q}_{\ell 1}$ associated with a dual formulation of $\ell 1$ -loss SVM (3.24) is SPS in general, as we mentioned in Section 3.1. It implies that an underlying optimization problem has a non-unique solution. In this subsection, we modify the primal formulation $\mathcal{P}_{\ell 1}^{\text{SVM}}$ (3.6) in such way that the Hessian in a dual formulation becomes symmetric positive definite (SPD); we refer [46, 47] for additional details.

This adjustment involves substituting an $\ell 1$ -loss hinge function by means of an $\ell 2$ -loss (squared $\ell 1$ -loss) in the objective function so that a primal $\ell 1$ -loss SVM formulation (3.6) results into the following form:

$$(\mathbf{w}^*, b^*, \boldsymbol{\xi}^*) = \arg \min_{\mathbf{w}, b, \boldsymbol{\xi}} \frac{1}{2} \langle \mathbf{w}, \mathbf{w} \rangle + \frac{C}{2} \sum_{i=1}^m \xi_i^2 \text{ s.t. } \begin{cases} y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, \\ i \in \{1, 2, \dots, m\}. \end{cases} \quad (3.32)$$

Analyzing the resulting formulation, we can simply observe the term $\sum_{i=1}^m \xi_i^2 \geq 0$, which quantifies misclassification error, is always non-negative. Therefore, we do not consider $\xi_i \geq 0$ as a constraint. Let us note that the formulation (3.32) is called the primal $\ell 2$ -loss SVM problem.

Let $\boldsymbol{\Theta}_\xi \in \mathbb{R}^{m+n+1}$ be a vector of parameters defined as follows:

$$\boldsymbol{\Theta}_\xi = [\mathbf{w}^T, b, \boldsymbol{\xi}^T]^T. \quad (3.33)$$

Then, we can rewrite a primal $\ell 2$ -loss SVM (3.32) into a matrix form such that:

$$(\mathcal{P}_{\ell 2}^{\text{SVM}}) : \boldsymbol{\Theta}_\xi^* = \arg \min_{\boldsymbol{\Theta}_\xi} \frac{1}{2} \boldsymbol{\Theta}_\xi^T \mathbf{Q}_{\ell 2} \boldsymbol{\Theta}_\xi \text{ s.t. } \mathbf{B}_{\ell 2} \boldsymbol{\Theta}_\xi \leq -\mathbf{h}, \quad (3.34)$$

where the Hessian matrix $\mathbf{Q}_{\ell 2} \in \mathbb{R}^{(m+n+1) \times (m+n+1)}$ equals:

$$\mathbf{Q}_{\ell 2} = \begin{bmatrix} \mathbf{Q}_{n+1,n+1} & \mathbf{O}_{n+1,m} \\ \mathbf{O}_{m,n+1} & C\mathbf{I}_{m,m} \end{bmatrix}, \quad \mathbf{Q}_{n+1,n+1} = \begin{bmatrix} \mathbf{I}_{n,n} & \mathbf{O}_{n,1} \\ \mathbf{O}_{1,n} & \mathbf{O}_{1,1} \end{bmatrix}, \quad (3.35)$$

a matrix $\mathbf{B}_{\ell 2} \in \mathbb{R}^{m \times (m+n+1)}$ is defined as:

$$\mathbf{B}_{\ell 2} = \begin{bmatrix} \mathbf{B} & -\mathbf{I}_m \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} -\mathbf{Y}_d \mathbf{X}^T, -\mathbf{y} \end{bmatrix}, \quad (3.36)$$

and vector related to upper bound is given by:

$$\mathbf{h} = \mathbf{e}_m. \quad (3.37)$$

As in case of the $\ell 1$ -loss SVM, we derive dual formulation. Recall. Using the Lagrange duality, and, evaluating the KKT conditions, the primal formulation (3.32) transforms into the dual one. Let $\boldsymbol{\lambda}_{\ell 2} \in \{\mathbb{R}_+ \cup \{0\}\}^m$ a vector of the Lagrange multipliers, where m is a number of training samples. Then, a Lagrangian function corresponding to (3.32) takes the following form:

$$\mathcal{L}_{\ell 2}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\lambda}) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m \xi_i^2 - \sum_{i=1}^m \lambda_i (y_i [\langle \mathbf{w}, \mathbf{x}_i \rangle + b] - 1 + \xi_i) \quad (3.38)$$

and the corresponding KKT conditions are as follows:

$$\nabla_{\mathbf{w}, b, \boldsymbol{\xi}} \mathcal{L}_{\ell 1}(\mathbf{w}^*, b^*, \boldsymbol{\xi}^*, \boldsymbol{\lambda}^*) = \mathbf{0}, \quad (\text{stacionarity}) \quad (3.39a)$$

$$\sum_{i=0}^m (1 - \xi_i^* + y_i [\langle \mathbf{w}^*, \mathbf{x}_i \rangle + b^*]) \leq 0, \quad (\text{primal feasibility}) \quad (3.39b)$$

$$\boldsymbol{\lambda}^* \geq \mathbf{0}, \quad (\text{dual feasibility}) \quad (3.39c)$$

$$\sum_{i=0}^m \lambda_i^* (1 - \xi_i^* + y_i [\langle \mathbf{w}^*, \mathbf{x}_i \rangle + b^*]) = 0. \quad (\text{complementarity}) \quad (3.39d)$$

Using the Lagrange duality, and, evaluating the KKT conditions, we can transform a primal formulation (3.32) into the dual one. First, the partial derivatives with respect to \mathbf{w} and b remain same:

$$\frac{\partial \mathcal{L}_{\ell 2}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\lambda})}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^m \lambda_i y_i \mathbf{x}_i = \mathbf{0}, \quad (3.40)$$

and

$$\frac{\partial \mathcal{L}_{\ell 2}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\lambda})}{\partial b} = \sum_{i=1}^m y_i \lambda_i = \langle \mathbf{y}, \boldsymbol{\lambda} \rangle = 0. \quad (3.41)$$

By computing the partial derivative of $\mathcal{L}_{\ell 2}(\mathbf{w}, b, \boldsymbol{\lambda})$ with respect to $\boldsymbol{\xi}$ and subsequently

set it to zero, we derive:

$$\frac{\partial \mathcal{L}_{\ell 2}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\lambda})}{\partial \boldsymbol{\xi}} = \underbrace{\frac{\partial}{\partial \boldsymbol{\xi}} \left(\frac{1}{2} \|\mathbf{w}\|^2 \right)}_{=0} + \frac{\partial}{\partial \boldsymbol{\xi}} \sum_{i=1}^m \frac{C \xi_i^2}{2} - \frac{\partial}{\partial \boldsymbol{\xi}} \sum_{i=1}^m \lambda_i (y_i [\langle \mathbf{w}, \mathbf{x}_i \rangle + b] - 1 + \xi_i) \quad (3.42a)$$

$$= \sum_{i=0}^m C \xi_i - \underbrace{\frac{\partial}{\partial \boldsymbol{\xi}} \sum_{i=1}^m \lambda_i (y_i [\langle \mathbf{w}, \mathbf{x}_i \rangle] + b)}_{=0} + \underbrace{\frac{\partial}{\partial \boldsymbol{\xi}} \sum_{i=1}^m \lambda_i}_{=0} - \frac{\partial}{\partial \boldsymbol{\xi}} \sum_{i=0}^m \lambda_i \xi_i \quad (3.42b)$$

$$= \sum_{i=0}^m C \xi_i - \sum_{i=0}^m \lambda_i = 0 \quad (3.42c)$$

which implies the following relationship for the vector of Lagrange multipliers $\boldsymbol{\lambda}$ and $\boldsymbol{\xi}$:

$$C \boldsymbol{\xi} - \boldsymbol{\lambda} = \mathbf{0}. \quad (3.43)$$

By substituting (3.40), (3.41), and (3.43) back into the Lagrangian function (3.11), we get:

$$\mathcal{L}_{\ell 2}(\mathbf{w}^*, b^*, \boldsymbol{\xi}^*, \boldsymbol{\lambda}^*) = \frac{1}{2} \|\mathbf{w}^*\|^2 + \frac{C}{2} \sum_{i=0}^m (\xi_i^*)^2 - \sum_{i=0}^m \lambda_i^* (y_i [\langle \mathbf{w}^*, \mathbf{x}_i \rangle + b^*] - 1 + \xi_i^*) \quad (3.44a)$$

$$= \frac{1}{2} \|\mathbf{w}^*\|^2 + \frac{C}{2} \sum_{i=0}^m (\xi_i^*)^2 - \sum_{i=0}^m \lambda_i^* y_i [\langle \mathbf{w}^*, \mathbf{x}_i \rangle + b^*] + \sum_{i=0}^m \lambda_i^* - \sum_{i=0}^m \lambda_i^* \xi_i^* \quad (3.44b)$$

$$= \frac{1}{2} \|\mathbf{w}^*\|^2 + \frac{C}{2} \sum_{i=1}^m (\xi_i^*)^2 - \sum_{i=1}^m \lambda_i^* y_i \left[\sum_{j=1}^m \lambda_j^* y_j \langle \mathbf{x}_j, \mathbf{x}_i \rangle + b^* \right] + \sum_{i=0}^m \lambda_i^* - \sum_{i=0}^m \lambda_i^* \xi_i^* \quad (3.44c)$$

$$= \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \lambda_i^* y_i \lambda_j^* y_j \langle \mathbf{x}_j, \mathbf{x}_i \rangle - \sum_{i=1}^m \sum_{j=1}^m \lambda_i^* y_i \lambda_j^* y_j \langle \mathbf{x}_j, \mathbf{x}_i \rangle - \underbrace{\sum_{i=0}^m \lambda_i^* y_i b^*}_{=0} + \sum_{i=0}^m \lambda_i^* + \frac{C}{2} \sum_{i=0}^m (\xi_i^*)^2 - \sum_{i=0}^m \lambda_i^* \xi_i^* \quad (3.44d)$$

$$= -\frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \lambda_i^* y_i \lambda_j^* y_j \langle \mathbf{x}_j, \mathbf{x}_i \rangle + \sum_{i=0}^m \lambda_i^* + \frac{1}{2} \sum_{i=1}^m \underbrace{C \xi_i^*}_{=\lambda_i} \xi_i^* - \sum_{i=1}^m \lambda_i^* \xi_i^* \quad (3.44e)$$

$$= -\frac{1}{2} (\boldsymbol{\lambda}^*)^T \mathbf{Y}_d^T \mathbf{X}^T \mathbf{X} \mathbf{Y}_d \boldsymbol{\lambda}^* + \mathbf{e}^T \boldsymbol{\lambda}^* - \frac{1}{2} \sum_{i=1}^m \lambda_i^* \underbrace{\xi_i^*}_{=\lambda_i/C} \quad (3.44f)$$

$$= -\frac{1}{2} (\boldsymbol{\lambda}^*)^T \mathbf{Y}_d^T \mathbf{X}^T \mathbf{X} \mathbf{Y}_d \boldsymbol{\lambda}^* + \mathbf{e}^T \boldsymbol{\lambda}^* - \frac{1}{2C} (\boldsymbol{\lambda}^*)^T \boldsymbol{\lambda}^* \quad (3.44g)$$

$$= -\frac{1}{2} (\boldsymbol{\lambda}^*)^T \mathbf{Q} \boldsymbol{\lambda} + \mathbf{e}^T \boldsymbol{\lambda}^* - \frac{1}{2C} (\boldsymbol{\lambda}^*)^T \mathbf{I} \boldsymbol{\lambda}^* \quad (3.44h)$$

$$= -\frac{1}{2} (\boldsymbol{\lambda}^*)^T (\mathbf{Q} + C^{-1} \mathbf{I}) \boldsymbol{\lambda}^* + \mathbf{e}^T \boldsymbol{\lambda}^*. \quad (3.44i)$$

Recall. We get (3.44i) by minimizing a Lagrangian function (3.38). w.r.t. \mathbf{w} , b , $\boldsymbol{\xi}$. Putting it together with constraints $\boldsymbol{\lambda} \geq 0$ and (3.41), we get the following dual optimization problem:

$$\boldsymbol{\lambda}^* = \arg \max_{\boldsymbol{\lambda}} -\frac{1}{2} \boldsymbol{\lambda}^T (\mathbf{Q} + C^{-1} \mathbf{I}) \boldsymbol{\lambda} + \mathbf{e}^T \boldsymbol{\lambda} \quad \text{s.t.} \quad \begin{cases} \mathbf{0} \leq \boldsymbol{\lambda}, \\ \mathbf{B}_E \boldsymbol{\lambda} = 0, \end{cases} \quad (3.45)$$

which we can rewrite into minimization form such that:

$$(\mathcal{D}_{\ell_2}^{\text{SVM}}) : \boldsymbol{\lambda}^* = \arg \min_{\boldsymbol{\lambda}} \frac{1}{2} \boldsymbol{\lambda}^T (\mathbf{Q} + C^{-1} \mathbf{I}) \boldsymbol{\lambda} - \mathbf{e}^T \boldsymbol{\lambda} \quad \text{s.t.} \quad \begin{cases} \mathbf{0} \leq \boldsymbol{\lambda}, \\ \mathbf{B}_E \boldsymbol{\lambda} = 0, \end{cases} \quad (3.46)$$

where:

$$\mathbf{Q} = \mathbf{Y}_d^T \mathbf{X}^T \mathbf{X} \mathbf{Y}_d, \quad \mathbf{B}_E = [\mathbf{y}^T]. \quad (3.47)$$

Since the Hessian is regularized by the matrix $C^{-1} \mathbf{I}$, it avoids linear dependency of columns also arising from possible multicollinearity of the training samples. Hence, the matrix $\mathbf{Q} + C^{-1} \mathbf{I}$ becomes SPD. Practically, the optimization problem and the quality of its solution is data-driven, i.e. highly depends on the data nature. Therefore, we can say precisely, the associated optimization problem could be more computationally stable, and a convergence rate of an underlying solver could be faster, than in a case of the ℓ_1 -loss SVM. On the other hand, ℓ_1 -loss SVM could produce a more robust model in the sense of performance score, because using linear sum of ξ_i leads to catching the outliers during a training phase of a classifier. Then, we adapt the support vector index set \mathbb{I}_{SV} such that:

$$\mathbb{I}_{\text{SV}} = \{i : 0 < \alpha_i, i = 1, 2, \dots, m\} \quad (3.48)$$

for a reconstruction formula related to a bias b :

$$b^* = \frac{1}{\text{card}(\mathbb{I}_{\text{SV}})} \left(\mathbf{X}_{*\mathbb{I}_{\text{SV}}}^T \mathbf{w}^* - \mathbf{y}_{\mathbb{I}_{\text{SV}}} \right)^T \mathbf{e}_{\mathbb{I}_{\text{SV}}}. \quad (3.49)$$

The solution components of \mathbf{w}^* and $\boldsymbol{\xi}^*$ of the primal problem satisfy:

$$\mathbf{w}^* = \mathbf{X} \mathbf{Y}_d \boldsymbol{\lambda}^* \quad \text{and} \quad \boldsymbol{\xi}^* = C^{-1} \boldsymbol{\lambda}^*, \quad (3.50)$$

where $\boldsymbol{\lambda}^*$ is a unique solution associated with $(\mathcal{D}_{\ell_2}^{\text{SVM}})$ (3.46). More detailed discussion about solvability of the primal and dual ℓ_2 -loss SVMs can be found in [39].

3.3 Relaxed-bias classification

A standard approaches soft-margin SVM, specifically ℓ_1 -loss and ℓ_2 -loss SVMs introduced in Section 3.1 respective in Section 3.2, solve a problem of finding a classification model in a form of the maximal-margin hyperplane:

$$h_{\Theta}(\mathbf{x}) = \langle \Theta_0, \mathbf{x} \rangle + \Theta_1 = \langle \mathbf{w}, \mathbf{x} \rangle + b. \quad (3.51)$$

In the case of the relaxed-bias classification [48], we do not consider a bias b in a classification model, however we include it into a problem formulation by means of augmenting the vector \mathbf{w} and each sample \mathbf{x}_i with an additional dimension so that:

$$\hat{\mathbf{w}} = \begin{bmatrix} \mathbf{w} \\ B \end{bmatrix}, \quad \hat{\mathbf{x}}_i = \begin{bmatrix} \mathbf{x}_i \\ \gamma \end{bmatrix}, \quad (3.52)$$

where $B \in \mathbb{R}$ is a variable and $\gamma \in \mathbb{R}^+$ is a user defined constant (*bias*), which is typically set to 1. Let $p \in \{1, 2\}$ for purposes related to our SVM application, then the problem of finding a modified hyperplane:

$$h_{\hat{\Theta}}(\hat{\mathbf{x}}) = \langle \hat{\mathbf{w}}, \hat{\mathbf{x}} \rangle \quad (3.53)$$

can be formulated as a constrained optimization problem in the following primal formulation:

$$(\hat{\mathbf{w}}^*, \hat{\boldsymbol{\xi}}^*) = \arg \min_{\hat{\mathbf{w}}, \hat{\boldsymbol{\xi}}} \frac{1}{2} \langle \hat{\mathbf{w}}, \hat{\mathbf{w}} \rangle + \frac{C}{p} \sum_{i=1}^m \hat{\xi}_i^p \text{ s.t. } \begin{cases} y_i \langle \hat{\mathbf{w}}, \hat{\mathbf{x}}_i \rangle \geq 1 - \hat{\xi}_i, \\ \hat{\xi}_i \geq 0 \text{ if } p = 1, \quad i \in \{1, 2, \dots, m\}. \end{cases} \quad (3.54)$$

where $\hat{\xi}_i = \max\{0, 1 - y_i \langle \hat{\mathbf{w}}, \hat{\mathbf{x}}_i \rangle\}$ is a hinge loss function associated with augmented samples $\hat{\mathbf{x}}_i$. We can generally say that a minimizer associated with formulation (3.54) corresponds to an optimal rotation of separating hyperplane \hat{H} in the origin of one-dimension higher feature space \mathbb{R}^{n+1} .

If p equals 1, the formulation (3.54) is called a relaxed-bias ℓ_1 -loss SVM problem; for $p = 2$, we talk about relaxed-bias ℓ_2 -loss SVM. For both $p = 1$ and $p = 2$, we can dualize the primal formulation (3.54) using the Lagrange duality (introduced in previous sections) so that:

$$(\mathcal{D}_{\ell_{1\text{relaxed}}}^{\text{SVM}}) : \boldsymbol{\lambda}^* = \arg \min_{\boldsymbol{\lambda}} \frac{1}{2} \boldsymbol{\lambda}^T \hat{\mathbf{Q}} \boldsymbol{\lambda} - \mathbf{e}^T \boldsymbol{\lambda} \text{ s.t. } \mathbf{o} \leq \boldsymbol{\lambda} \leq C \mathbf{e}, \quad (p = 1) \quad (3.55)$$

and:

$$(\mathcal{D}_{\ell_{2\text{relaxed}}}^{\text{SVM}}) : \boldsymbol{\lambda}^* = \arg \min_{\boldsymbol{\lambda}} \frac{1}{2} \boldsymbol{\lambda}^T (\hat{\mathbf{Q}} + C^{-1} \mathbf{I}) \boldsymbol{\lambda} - \mathbf{e}^T \boldsymbol{\lambda} \text{ s.t. } \mathbf{o} \leq \boldsymbol{\lambda}, \quad (p = 2) \quad (3.56)$$

where:

$$\widehat{\mathbf{Q}} = \mathbf{Y}_d^T \widehat{\mathbf{X}}^T \widehat{\mathbf{X}} \mathbf{Y}_d, \quad (3.57)$$

and:

$$\widehat{\mathbf{X}} = [\widehat{\mathbf{x}}_1, \widehat{\mathbf{x}}_2, \dots, \widehat{\mathbf{x}}_m]. \quad (3.58)$$

In sense of equivalence of solutions, we can show a connection between the classification models attained by solving SVM-QP problem arising from the bias and relaxed-bias formulations. Let us write the separating hyperplane equation in a component-wise form such that:

$$\widehat{H} := \langle \widehat{\mathbf{w}}, \widehat{\mathbf{x}} \rangle = \underbrace{w_1 x_1 + w_2 x_2 + \dots + w_m x_m}_{=\langle \mathbf{w}, \mathbf{x} \rangle} + \underbrace{B\gamma}_{=:b}. \quad (3.59)$$

From (3.59), we can easily observe a hyperplane resulting from (3.54), and the hyperplanes after reconstruction from (3.55), (3.56) are equivalent (in the sense of models) to hyperplanes associated with the bias formulations (3.6), (3.32), and (3.24), (3.46), respectively.

Regularization perspective in the context of SVM provides a theoretical framework that gives us an explicit explanation of the relationship between robustness to perturbations uncertainty set (generalization ability) and regularization of weights in the sense of preventing overfitting. Mainly, the research on classifier regularization focuses on its effect to bounding the complexity of a function representing a classification model. Refer to [49], the SVM classifier asymptotically minimizes an upper bound of expected classification error that converges to the Bayes risk – arising from a consistency of SVM [50]. Therefore, it implies, the regularization term $R(f)$ bounds a gap between the classification error on the training and test dataset.

Particularly, relaxed-bias SVM can be considered as a special case of the Tikhonov regularization (3.1), in which, specifically, the hinge loss is used as loss function:

$$V(y_i, f(\mathbf{x}_i)) = (1 - y_i f(\mathbf{x}_i))_+, \quad (3.60)$$

where $(s)_+ = \max\{0, s\}$. Considering $R(f) = \lambda \|f\|^2$, the regularization problem (3.1) becomes:

$$f^* = \arg \min_{f \in \mathcal{H}} m^{-1} \sum_{i=1}^m (1 - y_i f(\mathbf{x}_i))_+^2 + \lambda \|f\|^2. \quad (3.61)$$

Then, multiplying by $\frac{1}{2\lambda}$ yields:

$$f^* = \arg \min_{f \in \mathcal{H}} \frac{C}{2} \sum_{i=1}^m (1 - y_i f(\mathbf{x}_i))_+^2 + \frac{1}{2} \|f\|^2, \quad (3.62)$$

where $C \stackrel{\text{def}}{=} \frac{1}{\lambda m}$ and $f(\mathbf{x}_i) := \langle \mathbf{w}, \mathbf{x}_i \rangle$. Analysing (3.62), we can observe, the related objective

function corresponds to objective function of SVM classification problem, namely no-bias ℓ_2 -loss SVM. Exploiting similarity of bias and relaxed-bias formulations, we attain equivalence between standard ℓ_2 -loss SVM and the Tikhonov regularization as well. For showing a connection between ℓ_1 -loss SVM formulations and the regularization point of view, we just use square root of loss function (3.60):

$$V^{-2}(y_i, f(\mathbf{x}_i)) = \sqrt{(1 - y_i f(\mathbf{x}_i))_+}. \quad (3.63)$$

Substituting loss function in (3.62) by the function (3.63) yields the standard ℓ_1 -loss SVM formulation. Using similar equivalence as in a case of standard and relaxed-bias ℓ_2 -loss SVM, we can also explain the regularization perspective straightforwardly. Thus, based on these observations, we can conclude, the SVM classifiers provide models that take advantages of implicit regularization properties so that trade-off between robustness and performance of a model is driven by the parameter C .

Chapter 4

Performance of classification models

Once a classifier has been trained, it is essential to understand how it represents and generalizes an associated problem. Recall, overall performance of a classification model is not typically reported on a training data set itself, because it has been fine-tuned using this data set and such evaluation approach could not estimate the generalization ability of a model in a meaningful way. Thus, it is required to test an attained model on a data set, which is independent of the training samples. This data set is commonly called a test data set, and we denote it as \mathbb{X}_{test} in the following text.

A common approach for making a test data set is based on splitting an input data set typically in ratio 2 : 1 (or 3 : 1), ensuring that labels in these two newly-emerged parts follow the same probability distribution. Afterwards, a larger part is used for training and the remaining one for evaluating the performance scores for an attained model.

In this chapter, we introduce a various metrics used for evaluating performance of classification models in Section 4.1, and techniques for parameter optimization based on grid-search combined with cross-validation are then outlined in Section 4.2.

4.1 Model performance metrics

Selecting appropriate evaluation metrics is a key factor for the ML applications, because monitored qualities of a model varies according to different classification problems, e.g. we desire that a model balances predictive relevance among classes. In this section, we introduce a few commonly used performance scores, namely accuracy, precision, sensitivity, specificity, F_1 , Intersection over Union (IoU), and Area Under Curve - Receiver Operating Characteristic (AUC ROC). These metrics are sufficient for most real-world classification problems.

4.1.1 Confusion matrix

We start with a definition of a confusion matrix at first. Note, this matrix is not a performance metric itself. However, it gives us the first insights into model quality. The matrix has a specific table layout $\mathbb{N}^{k \times k}$, where k is a number of classes, and it summarizes a classification model performance in relation to test, validation sample/label pairs, or another relevant data set, where labels are known. It features two dimensions: *rows are related to samples within a predicted class, and columns depict samples withing an actual class*. We illustrate this concept for a general binary classification problem in Table 4.1, where $k = 2$.

		Predicted	
		Class A	Class B
Actual	Class A	<i>True Positive (TP)</i>	<i>False Positive (FP)</i>
	Class B	<i>False Negative (FN)</i>	<i>True Negative (TN)</i>

Table 4.1: Illustration: *a layout of a confusion matrix associated with a binary classifier.*

For a more straightforward analysis of class confusion, i.e. sample mislabeling, a confusion matrix provides counts for *True Positives (TP)*, *False Positives (FP)*, *False Negatives (FN)*, and *True Negatives (TN)* in the following manner:

- **True Positive** samples are those labeled as *Class A* and predicted as *Class A*,
- **False Positive** samples are labeled as *Class A* and incorrectly predicted as *Class B*. It is also known as a false alarm, i.e. a Type I Error,
- **False Negatives** are associated with samples labeled as *Class B* and incorrectly predicted as *Class A*, referred to as a Type II Error,
- **True Negatives** are samples labeled as *Class B* and predicted as *Class B*.

Let us note that various performance scores are derived from a confusion matrix using its entries, specifically numbers of *TP*, *FP*, *FN*, and *TN*. We introduce some of these scores in the following text.

4.1.2 Accuracy

The ISO standard defines accuracy as a metric that encapsulates both random and systematic observational errors. Mathematically, it is expressed as a ratio of a number of correctly

classified samples to a total number of samples:

$$\text{accuracy} \stackrel{\text{def}}{=} \frac{TP + TN}{TP + FP + FN + TN} * 100\%. \quad (4.1)$$

Accuracy is a suitable metric for nearly balanced data sets, which are characterized by a roughly equal distribution of labels across each class. Nonetheless, this metric can be misleading when we deal with highly imbalanced data sets, i.e. data sets where one class prevails. Typically, a minority is being a positive class. To demonstrate this point, let us consider the following example: *In a data set, there are 10 samples belonging to Class A and 90 samples of Class B, and all samples in Class A are being misclassified, it results in a classification model accuracy of 90%.*

To survey this scenario in details, our initial step could involve examining a confusion matrix. However, when we need a single score to assess model quality, e.g. for running a parameter optimization (Section 4.2), this approach may not be practical. In such cases, alternative performance scores come into play. We will briefly discuss some of them in the upcoming subsection.

4.1.3 Evaluating model performance for imbalanced data sets

Let us begin with the following classification problem: *building a classifier for predicting whether a patient has a serious disease or not.* For testing a model performance, we employ a data set consisting of 100 patients, where 95 of them are healthy, and 5 persons have the disease. In this section, we introduce commonly used scores for evaluating such an imbalanced data set, i.e. precision, sensitivity (recall) and harmonic mean of these scores called F_1 .

In information retrieval, precision is a metric that assesses the relevance of retrieved information, and it is defined as the proportion of relevant instances among the retrieved instances. In the context of our example, precision indicates a proportion of the patients who are diagnosed as having the disease and indeed, they have this disease. It is mathematically defined as follows:

$$\text{precision} \stackrel{\text{def}}{=} \frac{TP}{TP + FP} * 100\%. \quad (4.2)$$

Consider a scenario where a model in our example predicts every patient as diseased. In this case, the precision is 5%.

Another metric, which we can use for evaluating predictive relevance of a model, is sensitivity (recall). In the context of AUC-ROC, discussed in Section 4.1.4, it is referred to *True Positive Rate (TPR)* either. Sensitivity is defined as the proportion of relevant retrieved instances over a total number of relevant instances, which in our example represents a proportion of patients who have the disease and are correctly diagnosed as having the disease.

We can mathematically express this as:

$$\text{sensitivity} \stackrel{\text{def}}{=} \frac{TP}{TP + FN} * 100\%. \quad (4.3)$$

Let us consider the same case as mentioned earlier, where the classifier predicts that every patient has the disease. In this case, sensitivity equals to 100%.

From the analysis above, we can conclude that precision assesses a model performance in terms of *false positives*, while sensitivity measures the performance of the model with respect to *false negatives*. This is illustrated in Figure 4.1.

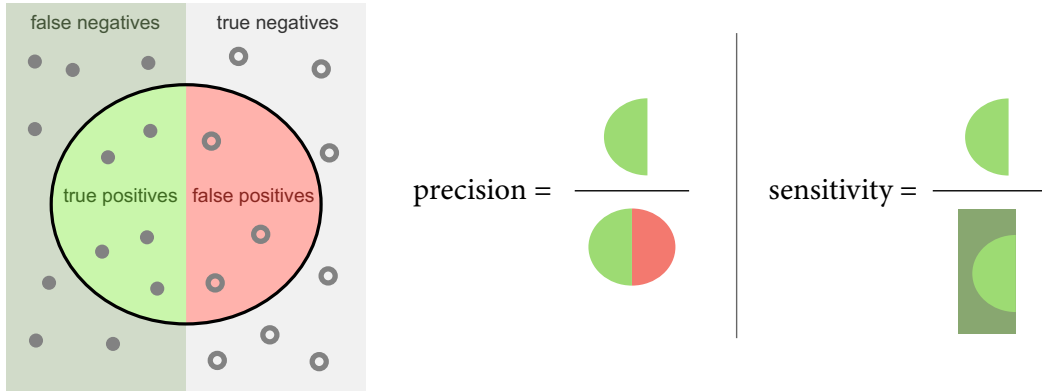


Figure 4.1: This illustration visually represents precision and sensitivity performance scores related to a classification model. Precision relates to the percentage of observations that are relevant, while sensitivity pertains to the total number of relevant observations that are correctly classified. *Original image was downloaded from [51]. It was slightly modified for the purpose of this text.*

We often require to have a single score that takes into account both precision and sensitivity for many practical applications. The first idea would be to use arithmetic mean of both scores; however it could misleads in some situations. Let us illustrate this point by means of the following example: *Consider precision is 5% and sensitivity is 100% from out previous case. Then, a overall score is 52.5%.*

A more suitable approach would be to use a harmonic mean of precision and sensitivity as opposed to a arithmetic mean. Mathematically, a harmonic mean can be considered as a reciprocal of a arithmetic mean of reciprocals of corresponding observations. In the field of data science, a harmonic mean of sensitivity and precision is commonly referred to F_1 and is defined such that:

$$F_1 \stackrel{\text{def}}{=} \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{sensitivity}}} = 2 * \frac{\text{precision} * \text{sensitivity}}{\text{precision} + \text{sensitivity}}. \quad (4.4)$$

Regarding our example, we obtain $F_1 = \frac{2 \cdot 100 \cdot 5}{100 + 5} = 9,52\%$. In practical applications, this score is often normalized rather than expressed as a percentage. If $F_1 \leq 0.5$, we categorize a classifier as a “random guess.” It implies that a classifier performs no better than simply flipping a coin.

4.1.4 Area Under Curve - Receiver Operating Characteristic

A Receiver Operating Characteristic (ROC) curve provides a visual representation related a diagnosing performance of a binary classifier. It displays a relative tradeoff between true positive and false positive samples. On the ROC curve, each point corresponds to a specific decision threshold with a TPR on y -axis and a false positive rate (FPR) on the x -axis as depicted in Figure 4.2. In addition, FPR is defined as follows:

$$FPR \stackrel{\text{def}}{=} 1 - \text{specificity} = 1 - \frac{TN}{TN + FP}, \quad (4.5)$$

where specificity (also known as a true negative rate) represents a probability that a negative sample is being classified as negative. As an ROC curve is constructed based on probability measurements, an Area Under a Curve (AUC) serves as an indicator of a classifier ability to effectively identify categories of samples, and is determined as area under ROC curve (Riemann integral), e.g. using the trapezoidal rule. This basically quantifies a degree of separability.

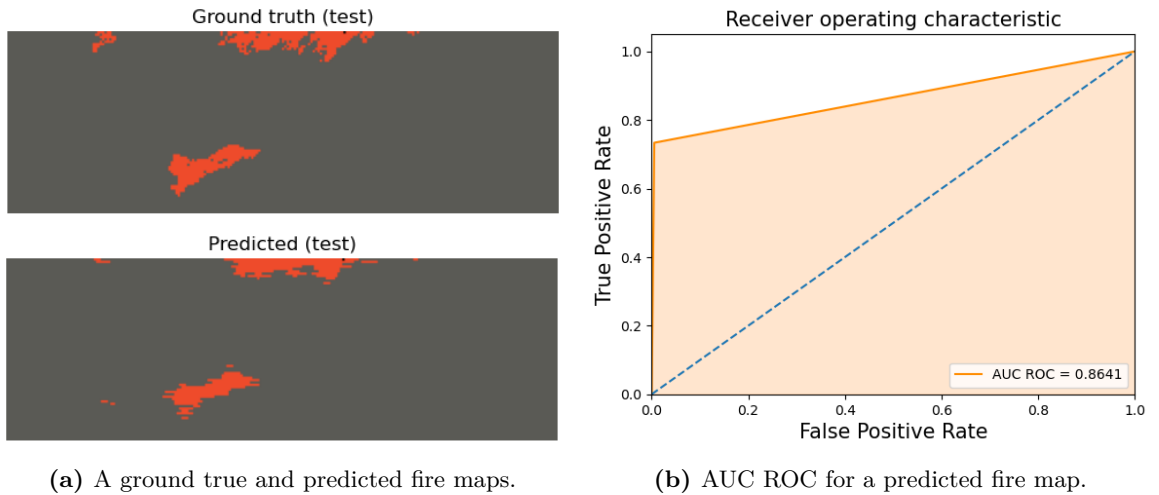


Figure 4.2: An evaluating performance of a semantic segmentation model used for localizing wildfires in Alaska regions. *This challenge is addressed as a part of a collaborative effort with researchers from Argonne and Oak Ridge National Laboratories within a broader natural hazard project [52] employing the PERMON toolbox [17, 40].*

4.1.5 Intersection over Union

Intersection over Union (IoU), also known as the Jaccard's index, is a metric primarily used in the computer vision field for evaluating a performance of a segmentation algorithm and object detection tasks. It measures the accuracy of an overlap between predicted and ground truth regions, or bounding boxes surrounding the objects in an image scene. Now, let us denote predicted regions as \mathbb{X}_{pred} , and \mathbb{X}_{gt} represents a ground truth.

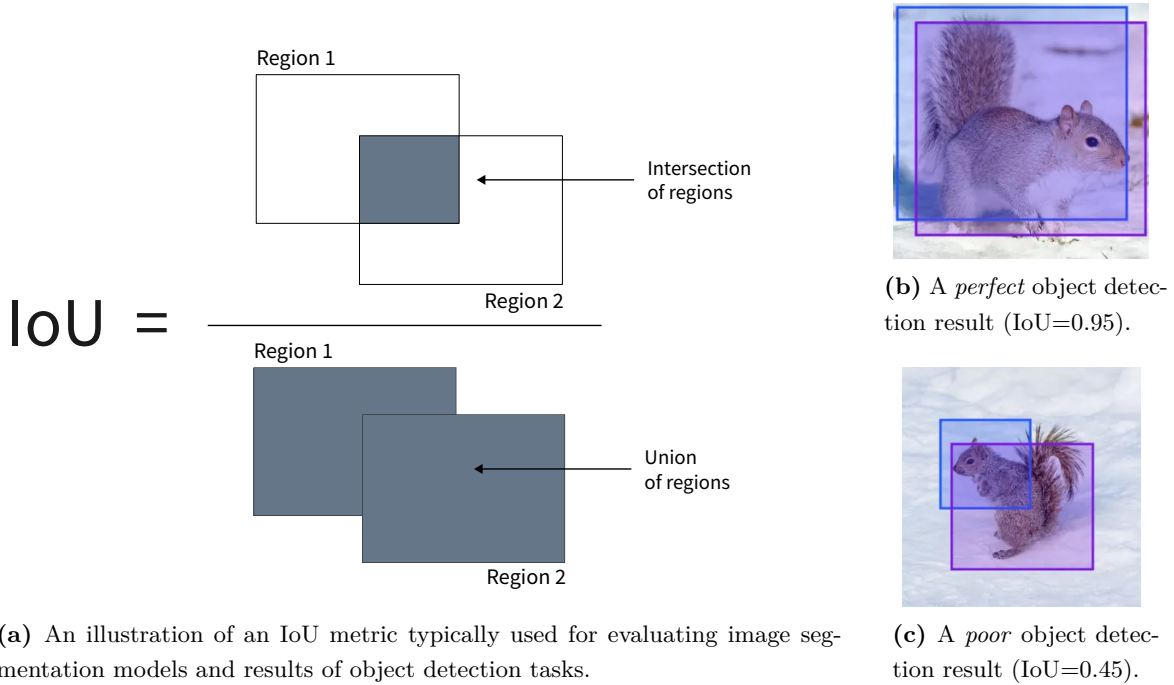


Figure 4.3: This example provides a visual representation of an IoU metric, accompanied by two scenarios related to a *perfect* and a *poor* object detection outcomes. To facilitate an evaluation of these results, a ground truth enclosed within a violet rectangle is used. Additionally, a detected object is outlined with a blue rectangle. *Original images were downloaded from [53]. We slightly modified them for the purpose of this text.*

Then, an IoU score is computed using the following formula:

$$\text{IoU} \stackrel{\text{def}}{=} \frac{\text{card}(\mathbb{X}_{\text{gt}} \cap \mathbb{X}_{\text{pred}})}{\text{card}(\mathbb{X}_{\text{gt}} \cup \mathbb{X}_{\text{pred}})}, \quad (4.6)$$

which is visualized in Figure 4.3a. For many real-world applications, the IoU score is determined slightly different. To account for binary image segmentation or an object detection task, the original formula (4.6) can be adapted using true positives, false positives, and false negatives such that:

$$\text{IoU} \stackrel{\text{def}}{=} \frac{TP}{TP + FN + FP}. \quad (4.7)$$

Note, the IoU score yields values between 0 and 1, where 0 means that predicted and ground truth regions have no overlap and 1 indicates perfect fit, i.e. regions are identical. In this text, we consider a results having IoU higher than 0.95 as excellent ones, and a good model typically has IoU higher than 0.7, an example depicted in Figure 4.3b. Any other scores are associated with poor results, visualized in Figure 4.3c.

4.2 Parameter optimization

A parameter optimization involves selecting the most suitable parameters for a learning algorithm during its training phase, intending to configure a model to improve its performance and prevent issues like underfitting or overfitting. In the context of SVMs, we aim to fine-tune a penalty parameter C to ensure that a model effectively addresses an associated problem and minimizes misclassification errors. A model performance is typically evaluated on independent samples, often utilizing a validation data set for this purpose.

4.2.1 Grid search

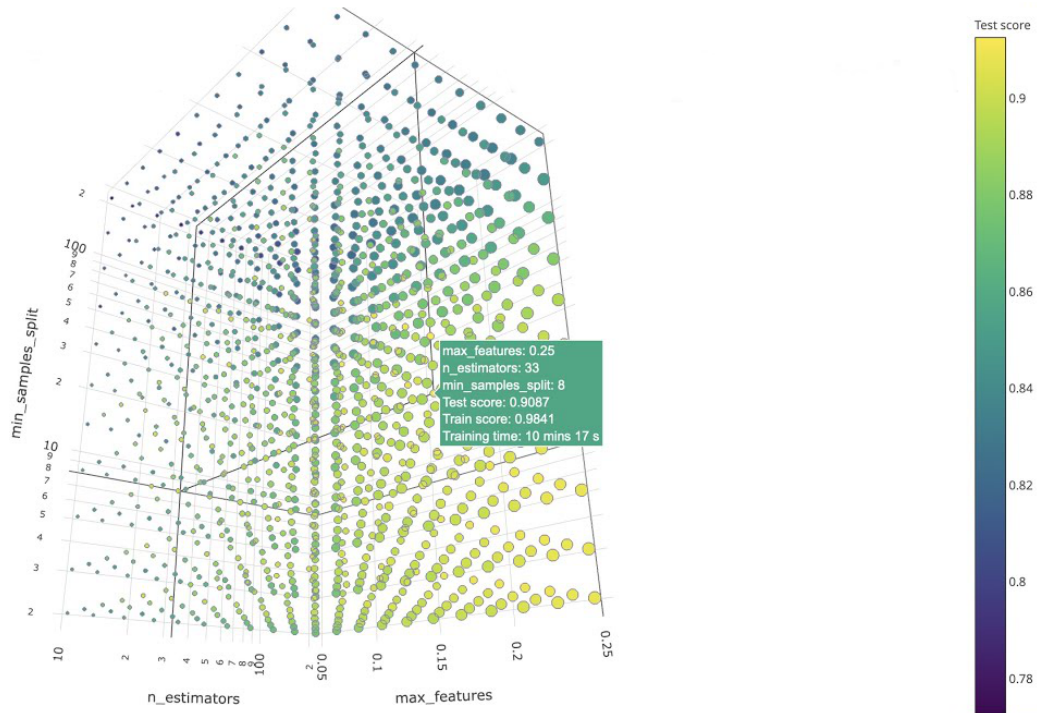


Figure 4.4: Visualization: *fine-tuning hyperparameters using a grid search approach [54].*

The traditional approach of performing parameter selection is based on a grid search approach. It involves an exhaustive search method that explores a predefined parameter set,

visualized in Figure 4.4. This set typically defines a reasonable region in the parameter space discretized using a uniform grid. A grid search then trains a model with each combination of the predefined values and evaluates a performance of a classifier on a held-out validation set. In practical implementations, a warm start of an underlying solver can significantly reduce computation time. It is worth mentioning other optimization techniques such as a randomized search [55] and the Bayesian optimization [56], which fall beyond the scope of this text.

4.2.2 Cross validation

For estimating a generalization process using grid-search, cross-validation techniques are frequently employed. The method called k -fold cross-validation belongs among the most widely used ones. In this approach, a dataset is randomly divided into k disjoint parts of roughly equal size. Sequentially, one of these partitions is kept aside as a validation data set for evaluating a model performance, while the remaining $k - 1$ parts are utilized for training. Afterwards, these k performance scores are aggregated and averaged to produce a single score.

Another widely used cross validation technique is called stratified cross validation, see publications [57, 58] for additional information. The stratification approach is implemented to prevent the misuse of class distributions so that each fold maintains a similar proportion of samples in each class, depicted in Figure 4.5. In order to training a classifier on a whole input data, nested cross-validation [59] is commonly employed for the best parameter setting.

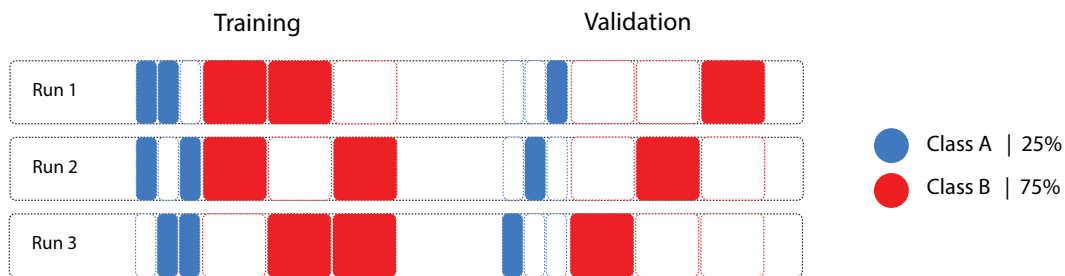


Figure 4.5: Stratified cross-validation performed on 3 folds.

Chapter 5

Deterministic solvers

This chapter introduces deterministic QP (Quadratic Programming) solvers, which we employ in our pipelines for training SVM models; we also outline their modifications and adaptations to improve the rate of convergence. Specifically, we used these algorithms to solve QP problems arising from QP-SVM formulations. The efficiency of the optimization algorithms is necessary to maximize utilization of modern hardware, which allows us to handle large-scale data sets in training pipelines designed to run on the fastest supercomputers in the world, e.g. Frontier¹ or Summit² operated by the Oak Ridge National Laboratory.

In this chapter, we will focus on algorithms developed by Prof. Dostál (VSB – TU Ostrava) such as Modified Proportioning with Reduced Gradient Projection (MPRGP), Semimonotonic augmented Lagrangian (SMALXE), their variants, or MPPCG developed by Kružík and firstly introduced in [60].

MPRGP is an efficient algorithm for solving convex QP problems with bound and box constraints. The theory guarantees R-linear convergence of the MPRGP algorithm for a fixed step length less than $\frac{2}{\|\mathbf{A}\|}$, where \mathbf{A} represents a Hessian matrix. However, this step-length is commonly really small in the case of the SVM problems, because a norm of the Hessian matrix \mathbf{A} is typically large, which results in many expansion steps. First, we investigate a selection of an optimal fixed step-length used in an expansion step on the convergence of the MPRGP algorithm in Section 5.5.1, where we explore also the effect of prolongation of this step length such that is greater than $\frac{2}{\|\mathbf{A}\|}$. However, the optimal value of the step-length can be different in each iteration. To overcome this issue, we introduce adaptive expansion strategies and study their advantages on publicly available data sets. We compare the results achieved using the adaptive expansion approaches with those attained using the MPPCG algorithm (mentioned above) in Section 5.3.

¹<https://www.olcf.ornl.gov/frontier/>

²<https://www.olcf.ornl.gov/summit/>

Since MPRGP and its variants are designed for the box or bound-constrained problems, they can be used only for solving relaxed-bias approaches related to SVM classification problems (introduced in Section 3.3). Thus, we introduce the SMALXE algorithm that we use to train models using the complete QP-SVM formulations (Section 3.1 and Section 3.2), i.e. with additional homogenized equality constraints in the dual formulations. SMALXE is a “pass-through” solver taking the case of equality constraints. In Section 5.4, we introduce two variants of this algorithm: SMALXE- ρ and SMALXE-M. We compare complete and relaxed-bias approaches for training SVM models on an initial data set for wildfire localization.³ The approaches for improving the rate of convergence of SMALXE-type algorithms based on normalizing equality constraint conclude this chapter.

5.1 Introduction

Performance of ML methods and learned models highly depends on the right data representation, e.g. using features extraction or selection. Roughly speaking, contemporary research on ML incorporates two mainstream directions, DNNs and representation learning (RL) based on conventional techniques. The key advantage of DNNs is incorporating feature selection inherently so that a series of hidden layers extracts abstract features. It allows the computer system to build complex concepts out of simpler ones hierarchically. On the other hand, designing a structure of DNNs and, afterwards, understanding resulting models are challenging. Currently, a general approach does not exist.

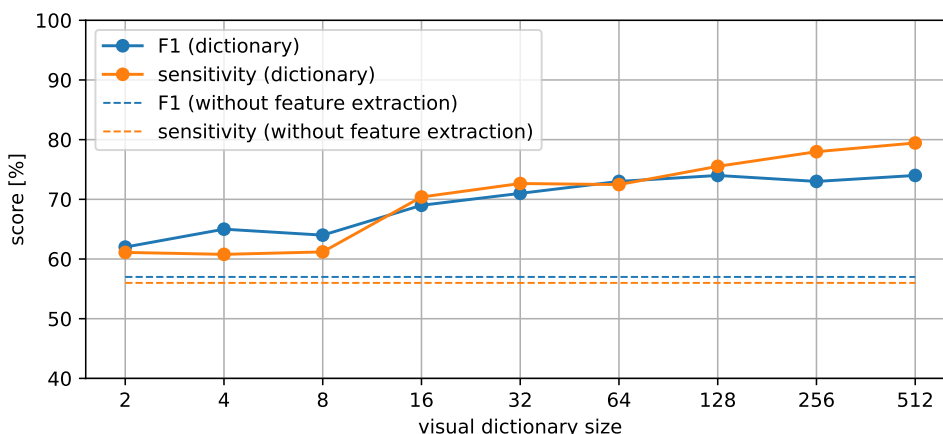


Figure 5.1: Oxford IIIT Pet Dataset: dictionary learning using SIFT descriptors (relaxed-bias ℓ_1 -loss SVM). Tested on SALOMON cluster (IT4Innovations).

³This application is introduced in Chapter 7 in more details, including description of workflow, large-scale benchmarks, etc.

Regarding feature extraction techniques, we study proper data representation using dictionary learning combining Scale-invariant feature transform (SIFT) / Speeded up robust features (SURF) extractors with vector quantification techniques in case of image processing in [62, 63], the effect of dictionary size on model performance score is depicted in Figure 5.1. For time series data, we observe applicability of PCA (Principal Component Analysis) later in Chapter 7 related to wildfire localization in Alaska.

Data transformation techniques are essential to force solver convergence and model performance. In the rest of chapter, we will introduce several active set algorithms, discuss optimal settings and tuning parameters, and examine the improvements in QP solvers. We effectively exploit the special structure of the dual QP-SVM problems to improve a convergence rate. Let us mention that these QP problems are characterized by a large active, one equality constraint and low precision in the sense of solution related to an optimization problem. As we mentioned at beginning of this chapter, we consider QP algorithms developed such as SMALXE, MPRGP [37, 61], their variants, or MPPCG [60].

Note that a QP problem with a box constraint, which arises from a dual relaxed-bias $\ell 1$ -loss SVM problem:

$$\left(\mathcal{D}_{\ell 1 \text{ relaxed}}^{\text{SVM}}\right): \arg \min_{\boldsymbol{\lambda}} \frac{1}{2} \boldsymbol{\lambda}^T \hat{\mathbf{Q}} \boldsymbol{\lambda} - \mathbf{e}^T \boldsymbol{\lambda} \text{ s.t. } \mathbf{o} \leq \boldsymbol{\lambda} \leq C \mathbf{e}, \quad (5.1)$$

or a QP problem with bound constraint as a dual relaxed-bias $\ell 2$ -loss SVM problem:

$$\left(\mathcal{D}_{\ell 2 \text{ relaxed}}^{\text{SVM}}\right): \arg \min_{\boldsymbol{\lambda}} \frac{1}{2} \boldsymbol{\lambda}^T \left(\hat{\mathbf{Q}} + C^{-1} \mathbf{I}\right) \boldsymbol{\lambda} - \mathbf{e}^T \boldsymbol{\lambda} \text{ s.t. } \mathbf{o} \leq \boldsymbol{\lambda}, \quad (5.2)$$

can be solved using the MPRGP algorithm developed. This algorithm belongs among active set based methods and has the rate of convergence given by the bound on the spectrum of the Hessian matrix. We implemented this algorithm into our software toolbox based on PETSc called PERMON [40, 17], introduced later in Chapter 6 in more detail.

The integral part of the MPRGP algorithm is the identification of the appropriate active set. To achieve this, MPRGP performs three types of steps:

1. a classical conjugate gradient (CG) step (solving a linear system),
2. a partial CG step to bound followed by an expansion step (expanding an active set),
3. a proportioning step (reducing an active set).

The theory guarantees R-linear convergence for a fixed step-length in an expansion step, which is less than $\frac{2}{\|\mathbf{A}\|}$, where \mathbf{A} represents a Hessian matrix. However, this step-length is commonly really small in the case of SVM problems, resulting in large number of expansion steps. It is suitable to find a fixed factor which could prolongate this step in the case of

our SVM problems, see our paper [64], and then examine an effect of adaptive expansion step-length computed from the free and reduced gradients, discussed later in this chapter.

If QP problem consists of an additional homogenized equality constraint $\mathbf{B}_E \mathbf{x} = \mathbf{o}$, e.g. a complete dual ℓ_2 -loss SVM (earlier introduced in Section 3.2):

$$\left(\mathcal{D}_{\ell_2}^{\text{SVM}}\right) : \arg \min_{\boldsymbol{\lambda}} \frac{1}{2} \boldsymbol{\lambda}^T \left(\mathbf{Q} + C^{-1} \mathbf{I}\right) \boldsymbol{\lambda} - \mathbf{e}^T \boldsymbol{\lambda} \text{ s.t. } \begin{cases} \mathbf{o} \leq \boldsymbol{\lambda}, \\ \mathbf{B}_E \boldsymbol{\lambda} = 0, \end{cases} \quad (5.3)$$

where $\mathbf{B}_E = [\mathbf{y}^T]$, we can add a penalized term $\rho \mathbf{B}_E^T \mathbf{B}_E$ being a part of the Hessian matrix:

$$\mathbf{Q} + C^{-1} \mathbf{I} + \rho \mathbf{B}_E^T \mathbf{B}_E, \quad (5.4)$$

where $\rho \in \mathbb{R}_0^+$, and solve a resulting QP problem using the MPRGP algorithm. Then, the convergence rate is sensitive to spectral properties of $\rho \mathbf{B}_E^T \mathbf{B}_E$, which involves setting the suitable value of a penalty ρ . Let us note that this penalty ensures sufficient fulfilment of the equality, i.e. a solution $\boldsymbol{\lambda}$ belongs $\text{Ker } \mathbf{B}_E$, and should not spoil significantly the conditioning of the Hessian.

A large penalty guarantees a more accurate fulfilment of this equality constraint, however the algorithm completely fails on the convergence rate. The spectral properties can be improved by multiplying this equality constraint by means of a transformation matrix \mathbf{T} defining the orthonormalization of rows of \mathbf{B}_E so that:

$$(\mathbf{T} \mathbf{B}_E)^T \mathbf{T} \mathbf{B}_E = \mathbf{B}_E^T \left(\mathbf{B}_E \mathbf{B}_E^T\right)^{-1} \mathbf{B}_E, \quad (5.5)$$

which is projector to $\text{Im } \mathbf{B}_E^T$ and the Hessian matrix with more favourable spectrum is then as follows:

$$\mathbf{Q} + C^{-1} \mathbf{I} + \rho \mathbf{B}_E^T \left(\mathbf{B}_E \mathbf{B}_E^T\right)^{-1} \mathbf{B}_E. \quad (5.6)$$

Another approach on how to enforce the equality constraint could be using a projector:

$$\mathbf{P} = \mathbf{I} - \mathbf{B}_E^T \left(\mathbf{B}_E \mathbf{B}_E^T\right)^{-1} \mathbf{B}_E \quad (5.7)$$

onto $\text{Ker } \mathbf{B}_E$, however, due to projections to the feasible set in expansion step performed by the MPRGP algorithm, it has to be equipped by penalized term $\rho \mathbf{B}_E^T \left(\mathbf{B}_E \mathbf{B}_E^T\right)^{-1} \mathbf{B}_E$ so that the Hessian has the following form:

$$\mathbf{P} \left(\mathbf{Q} + C^{-1} \mathbf{I}\right) \mathbf{P} + \rho \mathbf{B}_E^T \left(\mathbf{B}_E \mathbf{B}_E^T\right)^{-1} \mathbf{B}_E. \quad (5.8)$$

The SMALXE algorithm eliminates the requirement to set the proper penalty value ρ and extends the MPRGP algorithm by an outer loop updating the Lagrange multiplier for

this equality constraint. Both algorithms and their variants are discussed in this chapter and demonstrated on publicly available benchmarks from the LIBSVM dataset webpage.

5.2 The MPRGP algorithm

Recall that MPRGP represents an efficient algorithm for the solution of convex QP with box constraints:

$$\arg \min_{\mathbf{x}} f(\mathbf{x}) = \arg \min_{\mathbf{x}} \frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x} - \mathbf{x}^T \mathbf{b} \quad \text{s.t.} \quad \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}, \quad (5.9)$$

where $f(\mathbf{x})$ is the cost function, $\mathbf{A} \in \mathbb{R}^{n \times n}$ is positive semi-definite Hessian, $\mathbf{x} \in \mathbb{R}^n$ is the solution, $\mathbf{b} \in \mathbb{R}^n$ is the right hand side, $\mathbf{l} \in \mathbb{R}^n$ and $\mathbf{u} \in \mathbb{R}^n$ is the lower respectively the upper bound.

To describe the algorithm we first have to define a gradient splitting. Let $\mathbf{g} = \mathbf{A}\mathbf{x} - \mathbf{b}$ be the gradient. Then we can define a component-wise (for $j \in \{1, 2, \dots, n\}$) gradient splitting which is computed after each gradient evaluation. The free gradient is defined as:

$$g_j^f = \begin{cases} 0 & \text{if } x_j = l_j \text{ or } x_j = u_j, \\ g_j & \text{otherwise.} \end{cases} \quad (5.10)$$

The reduced free gradient is:

$$g_j^r = \begin{cases} 0 & \text{if } x_j = l_j \text{ or } x_j = u_j, \\ \min\left(\frac{x_j - l_j}{\bar{\alpha}}, g_j\right) & \text{if } l_j < x_j < u_j \text{ and } g_j > 0, \\ \max\left(\frac{x_j - u_j}{\bar{\alpha}}, g_j\right) & \text{if } l_j < x_j < u_j \text{ and } g_j \leq 0, \end{cases} \quad (5.11)$$

where $\bar{\alpha} \in (0, 2\|\mathbf{A}\|^{-1}]$ is used as an appriory chosen fixed step-length in the expansion step. Effectively, \mathbf{g}^f is the gradient on the free set and \mathbf{g}^r is the free gradient that is reduced such that a step in its opposite direction with the step-length $\bar{\alpha}$ does not leave the feasible set $\Omega = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\}$. A step in either of these directions can expand the active set, but cannot reduce it.

The chopped gradient is defined as

$$g_j^c = \begin{cases} 0 & \text{if } l_j < x_j < u_j, \\ \min(g_j, 0) & \text{if } x_j = l_j, \\ \max(g_j, 0) & \text{if } x_j = u_j. \end{cases} \quad (5.12)$$

A step in the direction opposite of \mathbf{g}^c may reduce the active set, but cannot expand it.

The next ingredient is the projection onto the feasible set Ω which is defined as

$$[P_\Omega(\mathbf{x})]_j = \min(u_j, \max(l_j, x_j)). \quad (5.13)$$

Finally, the projected gradient is defined as $\mathbf{g}^P = \mathbf{g}^f + \mathbf{g}^c$. Its norm decrease is the natural stopping criterion of the algorithm. These are all the necessary ingredients to summarise MPRGP in Algorithm 1.

Algorithm 1: MPRGP

Input: $\mathbf{A}, \mathbf{x}^0 \in \Omega, \mathbf{b}, \Gamma > 0, \bar{\alpha} \in (0, 2\|\mathbf{A}\|^{-1}]$

- 1 $\mathbf{g} = \mathbf{Ax}_0 - \mathbf{b}, \mathbf{p} = \mathbf{g}^f(\mathbf{x}^0), k = 0$
- 2 **while** $\|\mathbf{g}^P(\mathbf{x}^k)\|$ *is not small*:
- 3 **if** $\|\mathbf{g}^c(\mathbf{x}^k)\|^2 \leq \Gamma^2 \mathbf{g}^r(\mathbf{x}^k)^T \mathbf{g}^f(\mathbf{x}^k)$:
- 4 $\alpha_f = \max\{\alpha_{cg} : \mathbf{x}^k - \alpha_{cg}\mathbf{p}\}$
- 5 $\alpha_{cg} = \mathbf{g}^T \mathbf{p} / \mathbf{p}^T \mathbf{Ap}$
- 6 **if** $\alpha_{cg} < \alpha_f$:
- 7 // CG step
- 8 $\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_{cg}\mathbf{p}$
- 9 $\mathbf{g} = \mathbf{g} - \alpha_{cg}\mathbf{Ap}$
- 10 $\beta = \mathbf{g}^f(\mathbf{x}^{k+1})^T \mathbf{Ap} / \mathbf{p}^T \mathbf{Ap}$
- 11 $\mathbf{p} = \mathbf{g}^f(\mathbf{x}^{k+1}) - \beta\mathbf{p}$
- 12 **else:**
- 13 // Expansion step
- 14 $\mathbf{x}^{k+\frac{1}{2}} = \mathbf{x}^{k+1} - \alpha_f\mathbf{p}$
- 15 $\mathbf{g} = \mathbf{g} - \alpha_f\mathbf{p}$
- 16 $\mathbf{x}^{k+1} = P_\Omega(\mathbf{x}^{k+\frac{1}{2}} - \bar{\alpha}\mathbf{g}^f(\mathbf{x}^{k+\frac{1}{2}}))$
- 17 $\mathbf{g} = \mathbf{Ax}^{k+1} - \mathbf{b}$
- 18 $\mathbf{p} = \mathbf{g}^f(\mathbf{x}^{k+1})$
- 19 **else:**
- 20 // Proportioning step
- 21 $\alpha_{cg} = \mathbf{g}^T \mathbf{g}^c(\mathbf{x}^k) / \mathbf{g}^c(\mathbf{x}^k)^T \mathbf{Ag}^c(\mathbf{x}^k)$
- 22 $\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_{cg}\mathbf{g}^c(\mathbf{x}^k)$
- 23 $\mathbf{g} = \mathbf{g} - \alpha_{cg}\mathbf{Ag}^c(\mathbf{x}^k)$
- 24 $\mathbf{p} = \mathbf{g}^f(\mathbf{x}^{k+1})$
- 25 $k = k + 1$

Output: \mathbf{x}^k

5.3 Expansion strategies for the MPRGP algorithm

This section briefly summarizes theory introduced in our paper [60] dealing with modifications of the expansion step in the MPRGP algorithm. The original theory beyond this algorithm guarantees R-linear convergence for a fixed step length in expansion step such that its value goes from 0 to $\frac{2}{\|\mathbf{A}\|}$. However, this length is typically small in the case of the SVM problems, because a norm of \mathbf{A} is commonly large, which results in many expansion steps.

To reduce the number of expansion steps, we propose two alternatives to the original expansion. Additionally, we provide a comparison of the various choices for the search direction. All of these expansion steps perform a partial CG step to the bound or box, they differ in subsequent step employing these adaptive expansion step-lengths and directions. Another idea could be based performing the full CG step with a subsequent projection onto the feasible set. Therefore, we propose a projected CG step as another variant of the expansion.

Now, let us analyze the step lengths in a given direction to find the length, for which decreasing of the cost function is maximal. Recall that our cost function is:

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \mathbf{A} \mathbf{x} - \mathbf{x}^T \mathbf{b}. \quad (5.14)$$

and the expansion does a step in the \mathbf{g}^r direction with a fixed step length $\bar{\alpha} \in (0, 2\|\mathbf{A}\|^{-1}]$. Now, let us assume that a step performs in a direction \mathbf{d} , where \mathbf{d} is either \mathbf{g}^r or \mathbf{g}^f , and no active component is freed. We want to choose this step length such that the cost function decreases:

$$\begin{aligned} f(\mathbf{x}) - f(\mathbf{x} - \bar{\alpha}\mathbf{d}) &= f(\mathbf{x}) - \frac{1}{2}(\mathbf{x} - \bar{\alpha}\mathbf{d})^T \mathbf{A}(\mathbf{x} - \bar{\alpha}\mathbf{d}) + (\mathbf{x} - \bar{\alpha}\mathbf{d})^T \mathbf{b} = \\ &= f(\mathbf{x}) - \frac{1}{2}\mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{x}^T \mathbf{b} - \frac{1}{2}\bar{\alpha}^2 \mathbf{d}^T \mathbf{A} \mathbf{d} + \bar{\alpha} \mathbf{d}^T \mathbf{A} \mathbf{x} - \bar{\alpha} \mathbf{d}^T \mathbf{b} = \\ &= -\frac{1}{2}\bar{\alpha}^2 \mathbf{d}^T \mathbf{A} \mathbf{d} + \bar{\alpha} \mathbf{d}^T \mathbf{g} \geq 0, \end{aligned}$$

and after division by $\bar{\alpha} > 0$:

$$\frac{1}{2}\bar{\alpha} \mathbf{d}^T \mathbf{A} \mathbf{d} \leq \mathbf{d}^T \mathbf{g}. \quad (5.15)$$

Assuming \mathbf{d} is not in the null space of \mathbf{A} , we have $\mathbf{d}^T \mathbf{A} \mathbf{d} > 0$ and so we can divide the inequality by $\mathbf{d}^T \mathbf{A} \mathbf{d}$:

$$\bar{\alpha} \leq \frac{2\mathbf{d}^T \mathbf{g}}{\mathbf{d}^T \mathbf{A} \mathbf{d}}. \quad (5.16)$$

Because \mathbf{d} is either free gradient or reduced free gradient, we have:

$$\mathbf{d}^T \mathbf{d} \leq \mathbf{d}^T \mathbf{g} \leq \mathbf{g}^T \mathbf{g} \quad (5.17)$$

therefore:

$$1 \leq \frac{\mathbf{g}^T \mathbf{g}}{\mathbf{d}^T \mathbf{d}}, \quad (5.18)$$

and:

$$\bar{\alpha} \leq \frac{2\mathbf{d}^T \mathbf{d}}{\mathbf{d}^T \mathbf{A} \mathbf{d}} \leq \frac{2\mathbf{d}^T \mathbf{g}}{\mathbf{d}^T \mathbf{A} \mathbf{d}}. \quad (5.19)$$

We know that:

$$\mathbf{d}^T \mathbf{d} = \|\mathbf{d}\|^2 \quad \text{and} \quad \mathbf{d}^T \mathbf{A} \mathbf{d} = \left| \mathbf{d}^T \mathbf{A} \mathbf{d} \right| \leq \|\mathbf{A}\| \|\mathbf{d}\|^2, \quad (5.20)$$

therefore, using (5.19) we have:

$$\frac{2\mathbf{d}^T \mathbf{d}}{\mathbf{d}^T \mathbf{A} \mathbf{d}} \geq \frac{2 \|\mathbf{d}\|^2}{\|\mathbf{A}\| \|\mathbf{d}\|^2} = 2 \|\mathbf{A}\|^{-1} \geq \bar{\alpha}, \quad (5.21)$$

so that $\bar{\alpha} \leq 2 \|\mathbf{A}\|^{-1}$ is a step length ensuring the decrease of the cost function. Furthermore:

$$\frac{2\mathbf{d}^T \mathbf{g}}{\mathbf{d}^T \mathbf{A} \mathbf{d}} = \frac{2\mathbf{d}^T \mathbf{g}}{\mathbf{d}^T \mathbf{A} \mathbf{d}} \cdot 1 = \frac{2\mathbf{d}^T \mathbf{g}}{\mathbf{d}^T \mathbf{A} \mathbf{d}} \cdot \frac{\mathbf{d}^T \mathbf{d}}{\mathbf{d}^T \mathbf{d}} = \frac{2\mathbf{d}^T \mathbf{d}}{\mathbf{d}^T \mathbf{A} \mathbf{d}} \cdot \frac{\mathbf{d}^T \mathbf{g}}{\mathbf{d}^T \mathbf{d}}, \quad (5.22)$$

which in combination with (5.21) and (5.18) gives:

$$0 \leq \bar{\alpha} \leq 2 \|\mathbf{A}\|^{-1} \leq 2 \|\mathbf{A}\|^{-1} \frac{\mathbf{d}^T \mathbf{g}}{\mathbf{d}^T \mathbf{d}} \leq \frac{2\mathbf{d}^T \mathbf{g}}{\mathbf{d}^T \mathbf{A} \mathbf{d}} \quad (5.23)$$

Finally, we can consider the bounds:

$$0 \leq \bar{\alpha} \leq 2 \|\mathbf{A}\|^{-1} \frac{\mathbf{d}^T \mathbf{g}}{\mathbf{d}^T \mathbf{d}} \leq \frac{2\mathbf{d}^T \mathbf{g}}{\mathbf{d}^T \mathbf{A} \mathbf{d}} \quad (5.24)$$

as adaptive step lengths for an expansion step taking into account the actual situation. For testing purposes, let us consider the following notation:

- *fixed* $\bar{\alpha} = \alpha_u \|\mathbf{A}\|^{-1}$,
- *optapprox* $\bar{\alpha} = \alpha_u \|\mathbf{A}\|^{-1} \frac{\mathbf{d}^T \mathbf{g}}{\mathbf{d}^T \mathbf{d}}$,
- *opt* $\bar{\alpha} = \alpha_u \frac{\mathbf{d}^T \mathbf{g}}{\mathbf{d}^T \mathbf{A} \mathbf{d}}$,

where $\alpha_u \in (0, 2]$. In the derivation, we assumed that \mathbf{d} is not in the null space of \mathbf{A} . However, if \mathbf{d} is in the null space of \mathbf{A} we simply do not update $\bar{\alpha}$.

Since our goal is to expand the active set faster, it seems reasonable to replace the half step by the full CG step with a subsequent projection onto the feasible set. To be more specific, our expansion step becomes

$$\mathbf{x}^{k+1} = P_{\Omega}(\mathbf{x}^k - \alpha_{cg} \mathbf{p}),$$

followed by resetting $\mathbf{p} = \mathbf{g}^f$. Note, that realising the expansion in this way simplifies the implementation as we can always compute the CG step and then compute the gradient using CG recurrence when the step was feasible; otherwise, we project the solution onto the feasible set and recompute the gradient explicitly. Algorithm 2 illustrates the implementation. It replaces *if...else* block on lines 6–9 in Algorithm 1.

Algorithm 2: Projected CG

```

1  $\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_{cg}\mathbf{p}$ 
2 if  $\alpha_{cg} \leq \alpha_f$ :
3    $\mathbf{g} = \mathbf{g} - \alpha_{cg}\mathbf{A}\mathbf{p}$ 
4    $\beta = \mathbf{p}^T \mathbf{A}\mathbf{g}^c / \mathbf{p}^T \mathbf{A}\mathbf{p}$ 
5    $\mathbf{p} = \mathbf{g}^f - \beta\mathbf{p}$ 
6 else:
7    $\mathbf{x}^{k+1} = P_\Omega(\mathbf{x}^{k+1})$ 
8    $\mathbf{g} = \mathbf{A}\mathbf{x}^{k+1} - \mathbf{b}$ 
9    $\mathbf{p} = \mathbf{g}^f$ 

```

5.4 The SMALXE algorithm

In this section, we briefly introduce the SMALXE algorithm, which represents a class of the Semimonotonic Augmented Lagrangian algorithms for the QP problems that involves a linear equality constraint $\mathbf{B}_E\mathbf{x} = \mathbf{c}_E$; see [37] for further details. If there is no additional constraint, we talk about the SMALE algorithm, which typically employs conjugate gradients (CG) as inner solver. If we have an additional lower bound or box constraint, i.e. the QP problem in the following form:

$$\mathbf{x}^* = \arg \min_{\mathbf{x}} \frac{1}{2}\mathbf{x}^T \mathbf{A}\mathbf{x} - \mathbf{b}^T \mathbf{x} \text{ s.t. } \begin{cases} \mathbf{l}_b \leq \mathbf{x} \leq \mathbf{u}_b, \\ \mathbf{B}_E\mathbf{x} = \mathbf{c}_E, \end{cases} \quad (5.25)$$

we SMALBE or SMALSE algorithms, respectively. MPRGP is typically used as an inner solver in these algorithms. In the following text, we will refer the algorithms SMALE, SMALBE, and SMALSE by a common name SMALXE [41]. Inner solvers use the following stopping criterium:

$$\|\mathbf{g}^P\| \geq \min(M_k\|\mathbf{B}_E\mathbf{x}\|, \eta). \quad (5.26)$$

SMALXE algorithm has two basic variants, namely SMALXE- ρ and SMALXE-M. It is recommended in both variants to start with small penalty and increase ρ or reduce M by factor β if an increase of the Augmented Lagrangian in an outer loop is not sufficient, see the algorithm depicted in Figure 5.2.

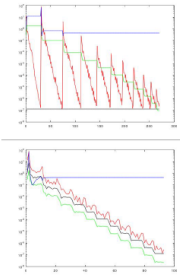
```

initialize:  $\mathbf{x}_0$  [o],  $\beta > 0$  [10],  $M_0 > 0$  [100||A||],  $\rho_0 > 0$  [2||A||],  $\eta > 0$  [0.1||b||]
 $\mu_0 = \mathbf{o}$ ,  $k = 0$ 
while  $\|\tilde{\mathbf{g}}^P(\mathbf{x}_k, \mu_k, \rho_k)\| > \varepsilon \|\mathbf{b}\| \vee \|\mathbf{B}\mathbf{x}_k\| > \varepsilon \|\mathbf{b}\|$ 
     $\mu_{k+1} = \mu_k + \rho_k \mathbf{B}\mathbf{x}_k$ 
    find  $\mathbf{x}_{k+1} \geq \mathbf{l}$  such that  $\|\tilde{\mathbf{g}}^P(\mathbf{x}_{k+1}, \mu_{k+1}, \rho_k)\| \leq \min(M_k \|\mathbf{B}\mathbf{x}_{k+1}\|, \eta)$  ...MPRGP
    if  $\tilde{L}(\mathbf{x}_{k+1}, \mu_{k+1}, \rho_k) \leq \tilde{L}(\mathbf{x}_k, \mu_k, \rho_{k-1}) + \frac{1}{2} \rho_k \|\mathbf{B}\mathbf{x}_{k+1}\|^2$ 
         $\rho_{k+1} = \beta \rho_k$  or  $M_{k+1} = M_k / \beta$ 
    end
     $k = k + 1$ 
end

```

$\min_{\mathbf{x} \geq \mathbf{l}} \frac{1}{2} \mathbf{x}^T (\mathbf{A} + \rho_k \mathbf{B}^T \mathbf{B}) \mathbf{x} - \mathbf{x}^T (\mathbf{b} - \mathbf{B}^T \mu_{k+1})$

SMALXE- ρ or SMALXE-M



$\tilde{L}(\mathbf{x}, \mu, \rho) = \frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x} - \mathbf{x}^T \mathbf{b} + \mu^T \mathbf{B} \mathbf{x} + \frac{1}{2} \rho \|\mathbf{B} \mathbf{x}\|^2$
 $\tilde{L}(\mathbf{x}, \mu, \rho) = \frac{1}{2} \mathbf{x}^T (\mathbf{A} + \rho \mathbf{B}^T \mathbf{B}) \mathbf{x} - \mathbf{x}^T (\mathbf{b} - \mathbf{B}^T \mu)$
 $\mathbf{g}(\mathbf{x}, \mu, \rho) = \mathbf{A} \mathbf{x} - \mathbf{b} + \mathbf{B}^T \mu + \rho \mathbf{B}^T \mathbf{B} \mathbf{x}$

 Figure 5.2: SMALXE-M and SMALXE- ρ variants.

Recommended variant is SMALXE-M, as it does not change the Hessian matrix and does not require recomputation of fixed step-length $\bar{\alpha}$ for the expansion. Further improvements could be achieved by reorthogonalizing rows of \mathbf{B}_E matrix

5.5 Benchmarks

5.5.1 Optimal fixed expansion step-length

In this section, we investigate a selection of an optimal fixed step-length and study an effect of prolongation this step length beyond interval $(0, 2\|\mathbf{A}\|^{-1}]$ as well. We earlier published a conference paper [64] on this topic and, here, we outline one selected result from this paper that we attained on the Australian data set. The Australian dataset (Australian Credit Approval) is public available on the LIBSVM dataset webpage [65] and concerns credit card applications. It contains 690 training samples with 14 features.

In the following experiments, we will focus on ℓ_1 -loss and ℓ_2 -loss relaxed-bias SVM, which we earlier introduced in Section 3.3. For the MPRGP solver, the relative norm used in the stopping criterion was set to 0.1, $\alpha_u = \{0 : 0.1 : 2.9, 3 : 1 : 10\}$, $C = 1.0$. The initial guess is set just under the upper bound, i.e. each component is set to $1 - 100\epsilon_m$, where $\epsilon_m \approx 2.2\text{e-}16$ is the machine precision.

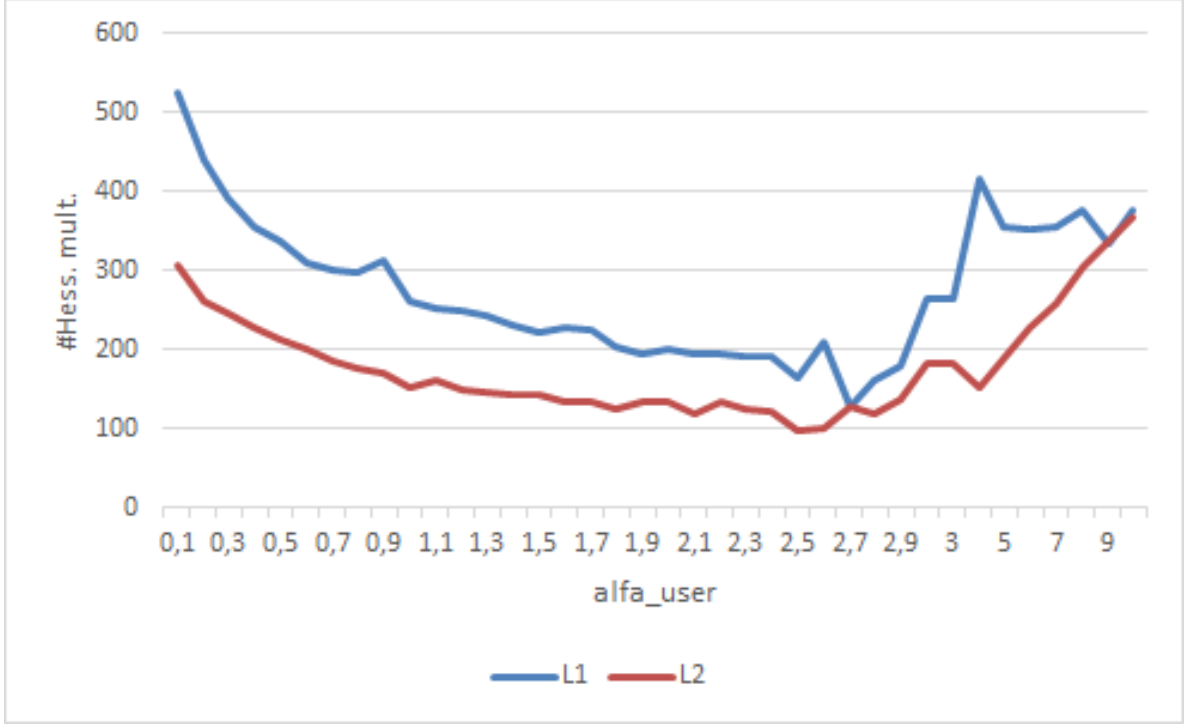


Figure 5.3: Australian dataset: Number of Hessian multiplications for ℓ_1 -loss and ℓ_2 -loss depends on varying α_u . *These results were published in our conference paper [64].*

The number of the Hessian multiplications related to ℓ_1 -loss and ℓ_2 -loss SVMs are depicted in Figure 5.3. For ℓ_1 -loss, we can see, the minimum of Hessian multiplication, 129, is obtained for $\alpha_u = 2.7$. It is a significant improvement compared to the optimum 195 Hessian multiplications for $\bar{\alpha} = 1.9\|\mathbf{A}\|^{-1} \in (0, 2\|\mathbf{A}\|^{-1}]$. For ℓ_2 -loss, the minimum is 96 Hessian multiplications for $\alpha_u = 2.5$ that is also an improvement compared to the optimum 125 Hessian multiplications for $\bar{\alpha} = 1.8\|\mathbf{A}\|^{-1} \in (0, 2\|\mathbf{A}\|^{-1}]$. We refer our conference paper [64] for additional details on this topic, including tables reporting number of Hessian multiplication, other experiments, and graphs.

5.5.2 Optimal adaptive expansion step-length

The optimal value of the step length can be different in each iteration. In this section, we outline experiments focusing on adaptive expansion strategies, which we earlier introduced in Section 5.3 and published in [60].

In the following experiments, we train classification models using relaxed-bias ℓ_1 -loss approach on the Australian data set, which we mentioned in Section 5.5.1. As in the previous section, we set the relative tolerance of MPRGP to 0.1 and an initial guess is set under an upper bound such that each component of \mathbf{x}_0 equal to $1 - 100\epsilon_m$, where ϵ_m represents the machine precision. The results are summarized in Figure 5.4, where opt and $optapprox$ being

prefixed by XY ; X denotes the vector used as the step-direction and Y is the vector used in the computation of the step-length.

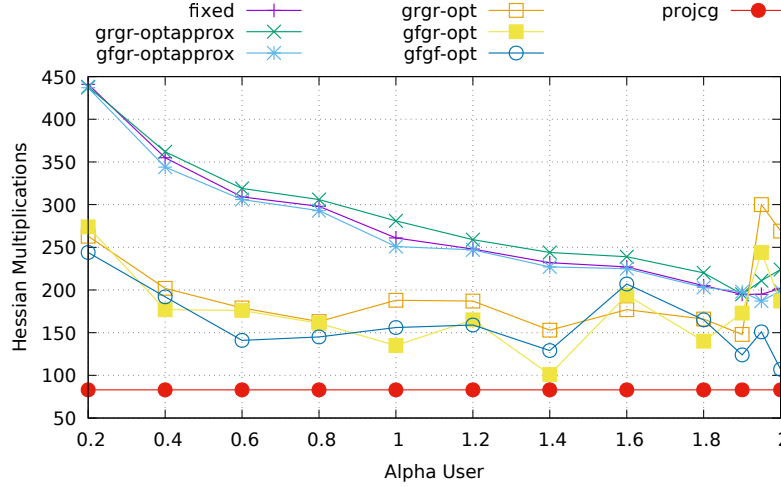


Figure 5.4: Australian dataset: Comparison of expansion strategies in the term of the number of the Hessian multiplications depending on α_u (Alpha user). *These results were published in our paper [60].*

The benchmarks showed that the optimal α_u for the *fixed* strategy is about 1.4. Using *optapprox* step length gives similar results to *fixed*. For most α_u the *opt* strategies outperform *fixed*. The drawback of *opt* is that there are relatively large jumps in the number of Hessian multiplications depending on α_u .

The *projcg* strategy consistently performed the best or was very close to the best. Moreover, it does not need an estimate of the maximal eigenvalue, nor a user-selected α_u . Additionally, the implementation of the algorithm is simplified. Therefore, we recommend using the projected CG step in place of the standard *fixed* step length expansion.

5.5.3 Comparison complete and relaxed-bias classification

This section deals with preliminary results on comparisons of approaches for training SVM models, specifically employing the complete and relaxed-bias ones. The concepts related to them, including problem formulations, were earlier introduced in Chapter 3. In the following experiments, we demonstrate these comparisons on the tasks of training semantic segmentation models associated with wildfires localization. In details, the application of wildfire localization is introduced later in this thesis in Chapter 7, including motivation, techniques for data processing, etc.

	#wildfires pixels	#background pixels	#attributes
training	46,851 (13.01%)	313,149 (86.99%)	133
test	28,351 (11.81%)	211,649 (88.19%)	133

Table 5.1: Description related to training data set and test one. Proportions of samples in each category are pointed out as percents.

In this section, we use a data set containing wildfires in the Alaska regions in 2004, their localization are depicted in Figure 5.5. Note that the results introduced in this section were earlier published in the following paper [66].

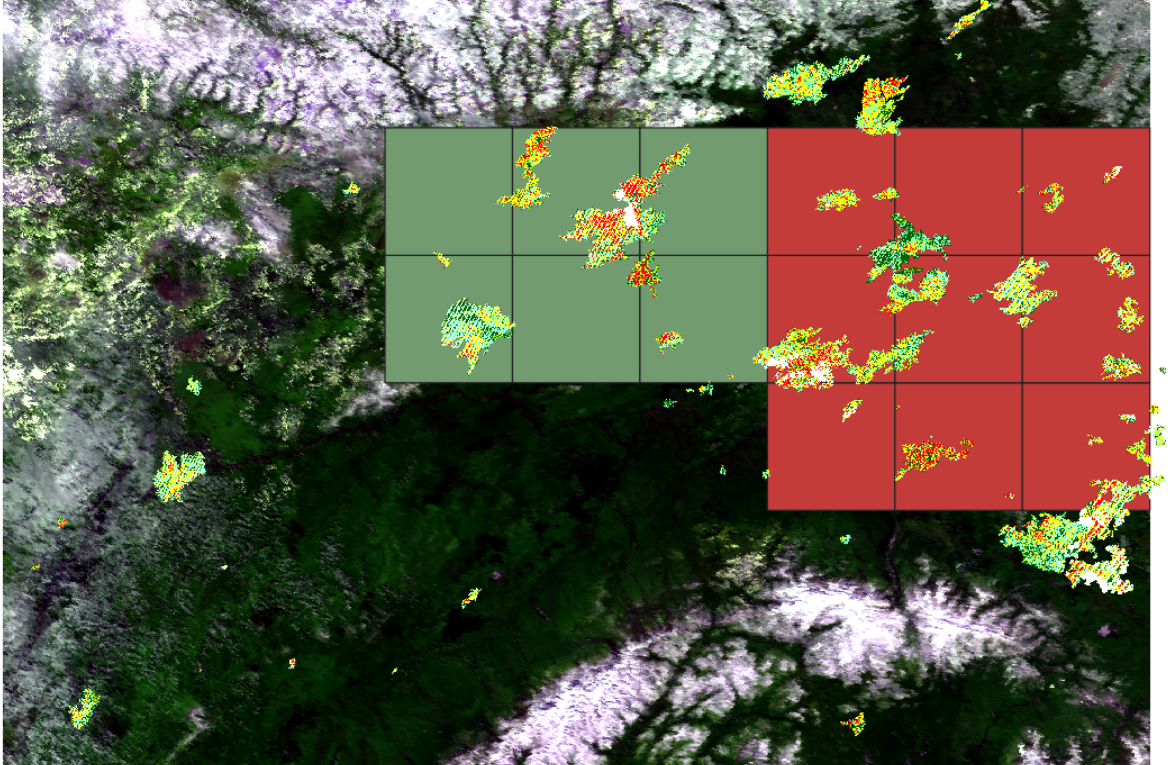


Figure 5.5: Alaska areas burned by wildfires. Red and green squares represent the training and test data sets, respectively. Wildfires localization are aggregated over 152 days from May to September 2004. *This visualization was created by Zachary Langford and was earlier published in [66].*

To describe regions affected by fire, we consider changes in normalized reflectance over time so that features corresponding to each pixel are represented by multidimensional time series related to the 7 spectral bands with an 8-day period. The pixels are then categorized using boundaries collected from Monitoring Trends in Burn Severity.⁴ Looking at Table 5.1,

⁴<https://www.mtbs.gov/>

we can see that the whole data set contains 600,000 pixels, which we split into training and test ones consisting of 360,000 and 240,000 samples, respectively.

loss func. (type)	\sum Hessian mult.	loss func. (lower better)	training time [s]
ℓ_1	34622	1.80e5	73.82
ℓ_2	51967	2.24e5	112.28

Table 5.2: Solutions related to the complete SVM formulations using SMALXE + MPRGP. A default stopping condition is used. Results are attained using 64 MPI processes on the KAROLINA supercomputer. Setting of an inner solver: $\Gamma = 100$, a relative tolerance $\text{rtol} = 1e - 2$ and $\text{divtol} = 1e10$ for an outer loop (SMALXE). Penalty $C = 1$.

We trained the following semantic segmentation models on the KAROLINA supercomputer,⁵ which is a combination of HPE Apollo 2000 and Apollo 6500 systems used for HPC workloads such as AI and other data-intensive applications. For training models, we used our in-house software called PermonSVM – introduced later in this thesis in Chapter 6.

First, we train models using the complete dual SVM formulations having SPS (ℓ_1 -loss) and regularized Hessian (ℓ_2 -loss), introduced in Section 3.1 and Section 3.2, respectively. As a solver, we employed the SMALXE-M algorithm (default in PERMON).

The approach above divides an optimization problem into two sub-problems such that equality and bound/box constraints are handled separately, one after another. An outer loop is performed using the augmented Lagrangians, and a bound or box constrained optimization problem is computed by means of a QP solver. We used the MPRGP algorithm, which was earlier introduced in Section 5.2. Computational demands are summarized in Table 5.2.

Further, we train models using the relaxed-bias approaches for both ℓ_1 -loss and ℓ_2 -loss dual SVM formulations, which we introduced earlier in Section 3.3. Recall, these models can be viewed as solutions related to formulations considered as a special case of the Tikhonov regularization (3.1) such that a bias term b is relaxed. Moreover, this approach simplifies the complete SVM formulations and it leads to optimization problems that are computationally cheaply to solve than the original ones (complete). The results are summarized in Table 5.3.

loss func. (type)	\sum Hessian mult.	loss func. (lower better)	training time [s]
ℓ_1	2962	2.28e4	22.67
ℓ_2	1025	3.03e4	6.96

Table 5.3: Attained results using 64 MPI processes (KAROLINA). Solver: MPRGP, an expansion step is performed using the projected CG step, $\Gamma = 100$ in proportion criterion, a relative tolerance was set to 0.1; penalty $C = 1$.

⁵<https://www.it4i.cz/en/infrastructure/karolina>

Looking at elapsed times in Table 5.2 and Table 5.3, we can see that training a model is 3.26 times slower using the complete ℓ_1 -loss formulation (dual) than in case of its relaxed approach. Nevertheless, we can observe that the value of a loss function is 7.89 times lower compared to complete and relaxed approaches.⁶ It is similar to training a model employing the complete ℓ_2 -loss formulation (dual) compared to its relaxed approach. A training time is 16.1 times slower, and a value associated with a loss function is 7.39 times higher for a complete formulation.

loss func. (type)	formulation (type)	precision [%]	sensitivity [%]	F_1
ℓ_1	complete	82.80	96.18	0.89
	relaxed-bias	84.12	94.58	0.89
ℓ_2	complete	82.98	95.63	0.89
	relaxed-bias	83.58	92.81	0.89

Table 5.4: The best performance scores related to models trained employing the complete and relaxed-bias SVM formulations (on the test data set).

The performance scores of models trained using complete and relaxed-bias formulations are summarized in Table 5.4. Comparing them, we can see that they do not significantly differ; however, a true positive rate (sensitivity) is slightly higher for the complete formulations. It means that models identify fire occurrences (true positives) better than the ones with a relaxed bias at the cost of decreasing precision, i.e. a false positive rate. Overall performance scores measured by F_1 score (a harmonic mean of precision and sensitivity) are the same for both relaxed-bias and complete models, see Chapter 4 for further information about evaluating performance scores of classification models.

5.5.4 SMALXE performance improvements

In previous section, we introduced preliminary results on comparison complete and relaxed-bias formulations for training classification models of SVM type, which we published in a conference paper [66]. In those experiments, we just focused on model quality not on effectiveness of training process itself (in the sense of convergence rate). In the following text, we introduce essential ideas to accelerate training processes, based on normalization equality constraint $\mathbf{B}_E = [\mathbf{y}^T]$ and, additionally, employing projector to $\text{Ker } \mathbf{y}^T$.

The presented results were computed in Octave on the small public available data set Heart from the LIBSVM collection [65]. The Heart dataset is based on data from the Cleveland Clinic Foundation. The samples belong to two classes: the presence or absence of heart disease. It contains 270 training samples with 13 features.

⁶Recall. The loss function quantifies misclassification error.

In the following experiments, we set penalty $\rho = \|\mathbf{A}\|$, $\mathbf{A} = \mathbf{Q} + C^{-1}\mathbf{I}$, misclassification penalty $C = 1$, an user defined α_u equals 1. We explore an effect of initial guess on convergence rate; we study two cases $\mathbf{x}_0 = \max(\mathbf{e}, \mathbf{l})$ and $\mathbf{x}_0 = \mathbf{l}$, where represents lower bound. The results attained without any additional normalization of \mathbf{B}_E or the projector are summarized in Table 5.5.

Variant	# Exp. type	# Out	# CG	# Exp	# Prop	# Hess. mult.
SMALSE-M	fixed	3 / 1	143 / 118	72 / 61	2 / 1	289 / 241
SMALSE- ρ	fixed	4 / 1	154 / 118	72 / 61	2 / 1	300 / 241

Table 5.5: SMALXE and MPRGP convergence (l_2 -loss SVM with equality) for Heart dataset: the Hessian matrix is $\mathbf{A} + \rho\mathbf{y}\mathbf{y}^T$, $\rho = \|\mathbf{A}\|$, $\alpha_u = 1$, $\mathbf{x}_0 = \max(\mathbf{e}, \mathbf{l})$ / $\mathbf{x}_0 = \mathbf{l}$, and $C = 1$.

Significant improvement is achieved by normalizing \mathbf{y} , i.e. a vector defining an equality constraint \mathbf{B}_E , so that the penalized term is the projector to $\text{Im } \mathbf{y}$. The condition number of the Hessian matrix (determinating the rate of convergence) is reduced from 89,465 to 929.44, i.e. by factor almost 97. The results are presented in Table 5.6.

Variant	# Exp. type	# Out	# CG	# Exp	# Prop	# Hess. mult.
SMALSE-M	fixed	4 / 1	98 / 72	49 / 49	2 / 1	198 / 171
SMALSE- ρ	fixed	4 / 1	105 / 72	51 / 49	1 / 1	208 / 171

Table 5.6: SMALXE and MPRGP convergence (l_2 -loss SVM with equality) for Heart dataset: the Hessian matrix is $\mathbf{A} + \rho\mathbf{y}_{norm}\mathbf{y}_{norm}^T$, $\rho = \|\mathbf{A}\|$, $\alpha_u = 1$, $\mathbf{x}_0 = \max(\mathbf{e}, \mathbf{l})$ / $\mathbf{x}_0 = \mathbf{l}$, and $C = 1$.

Next significant improvement consists in employing the projector to $\text{Ker } \mathbf{y}^T$, where the condition number of the Hessian matrix is 327.41, i.e. further reduction by factor almost 3. The results are outlined in Table 5.7.

Variant	# Exp. type	# Out	# CG	# Exp	# Prop	# Hess. mult.
SMALSE-M	fixed	3 / 1	55 / 53	41 / 40	1 / 1	138 / 134
SMALSE- ρ	fixed	4 / 1	113 / 53	65 / 40	1 / 1	244 / 134

Table 5.7: SMALXE and MPRGP convergence (l_2 -loss SVM with equality) for Heart dataset: the Hessian matrix is $\mathbf{PAP} + \rho\mathbf{y}_{norm}\mathbf{y}_{norm}^T$, $\rho = \|\mathbf{A}\|$, $\alpha_u = 1$, $\mathbf{x}_0 = \max(\mathbf{e}, \mathbf{l})$ / $\mathbf{x}_0 = \mathbf{l}$, and $C = 1$.

Distribution of eigenvalues for the Hessian matrices are depicted in Figure 5.6, Figure 5.7, Figure 5.8.

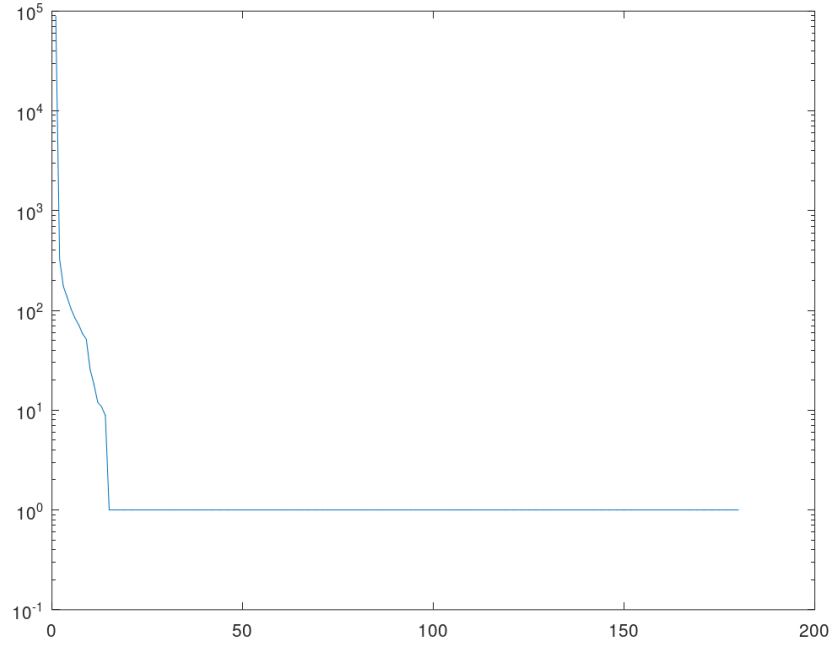


Figure 5.6: Eigenvalues of the Hessian $\mathbf{A} + \rho \mathbf{y} \mathbf{y}^T$, $C = 1$.

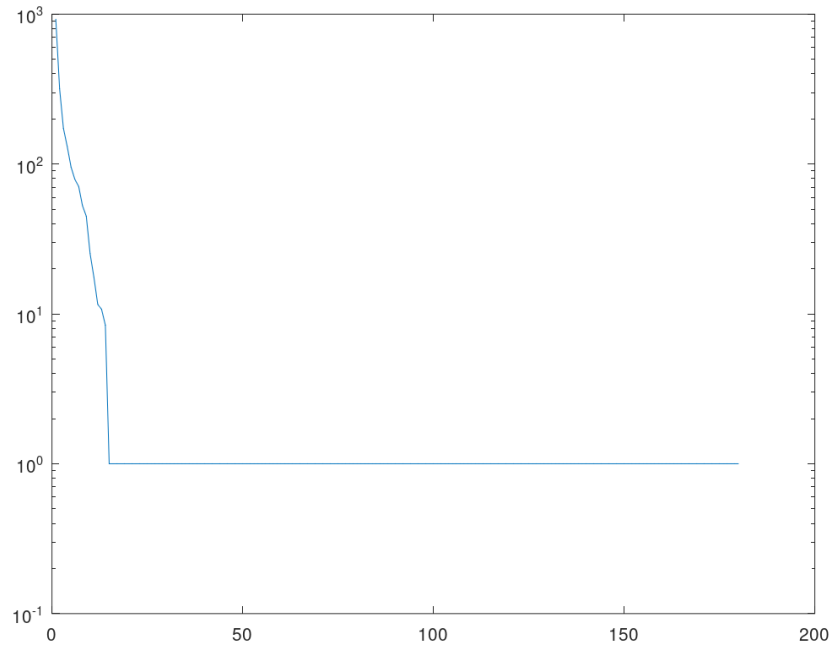


Figure 5.7: Eigenvalues for $\mathbf{A} + \rho \mathbf{y}_{norm} \mathbf{y}_{norm}^T$ for Heart dataset, $C = 1$.

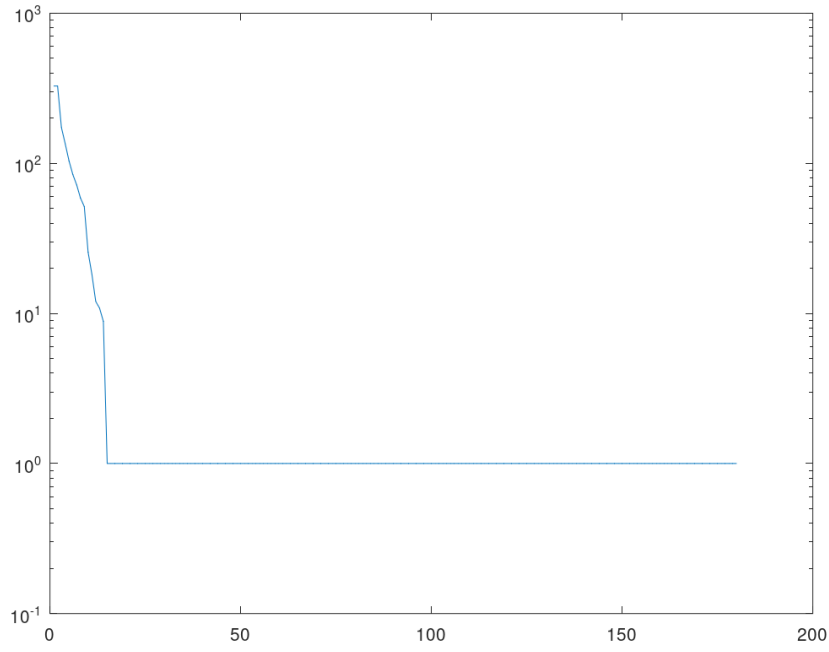


Figure 5.8: Eigenvalues for $PAP + \rho \mathbf{y}_{norm} \mathbf{y}_{norm}^T$ for Heart dataset, $C = 1$.

Chapter 6

PermonSVM: SVM implementation on top of PETSc

This chapter introduces scalable training SVM models implemented in PermonSVM, which is written on the top of the PETSc framework and the PERMON toolbox; the name PERMON stands for *Parallel, Efficient, Robust, Modular, Object-oriented, Numerical*. In the following text, we will introduce a basic functionality of PermonSVM and briefly comment on its scalability performance in Section 6.1. An efficient data loading using an HDF5 file format is mentioned in Section 6.2, where we also describe a structure of this file format in a nutshell. Two showcases of using PermonSVM API for C programming language are then demonstrated in Section 6.3.

Let us mention that the development of the PermonSVM module is an essential part of this thesis. You can see the contribution statistics on the GitHub site <https://github.com/permon/permonsvm/graphs/contributors>.

6.1 Introduction

PermonSVM package [17] is a part of the PERMON toolbox designed for usage in a massively parallel distributed environment containing hundreds of computational cores. Technically, it is an extension of the core PERMON package called PermonQP [40], from which it inherits basic data structures, initialization routines, build system, and utilizes computational and QP transformation routines, e.g. normalization of an objective function, dualization, etc.

Programmatically, a core functionality of the PERMON toolbox is written on the top of the PETSc framework [19, 67], follows the same design and coding style, making it easy-to-use for anyone familiar with PETSc. It is usable on all main operating systems and architectures consisting from smartphones through laptops to high-end supercomputers. It also supports computation on graphic cards (GPUs) [68] through CUDA [69] for GPUs manufactured by

NVidia¹, HIP [70] for AMD GPUs² or OpenCL [71], as well as hybrid MPI-GPU parallelism. A current version of PermonSVM (3.20) has the following features:

- distributed parallel (through MPI) I/O operations for formats such as:
 - SVMLight,
 - HDF5 - mentioned in Section 6.2,
 - PETSc binary file format,
- four problem formulations of classification problems such as the complete ℓ^1 -loss and ℓ^2 -loss SVM introduced in Chapter 2 respective in Chapter 3, and relaxed-bias formulations introduced in Section 3.3,
- two types of parallel cross-validation, namely k -fold and stratified k -fold cross-validation,
- a probability model based on Platt's scaling technique.

The resulting QP-SVM problem with the implicitly represented Hessian, i.e. a matrix $\mathbf{X}^T \mathbf{X}$ is not set up, is solved by PermonQP. The PermonQP package implements the QP algorithms such as MPRGP and SMAXE [37, 61, 41], which are developed and optimized by Prof. Dostál's group at the Department of Applied Mathematics, VŠB – Technical University of Ostrava. The effect of variants of parameters setting for these algorithm, and data transformations on the speed of model training is outlined in ???. Developing and optimizing these QP algorithms is another integral part of this thesis. This includes ingredients associated with an optimal initial guess, adaptive expansion steps, and variants of the SMALXE algorithm such as SMALXE-M and SMALSE- ρ ; you can find more information about these approaches in Chapter 5. The scalability of these algorithms and using PETSc became the main reasons for fruitful collaboration with world-leading research institutes, namely Argonne and Oak Ridge National Laboratories (USA), aimed at wildfires localization in Alaska, which is mentioned in Chapter 7.

Unlike standard ML libraries like scikit-learn [27], the PERMON toolbox provides interface functions to change underlying QP-SVM solver, monitoring performance scores simultaneously on training and test data sets, and tweaking the algorithms. Contemporary SVM solvers are commonly limited up to tens of thousands of samples in order to solve complete dual formulation. PermonSVM goes far beyond that since it has no intrinsic limitations imposed by a single node memory.

¹https://en.wikipedia.org/wiki/List_of_Nvidia_graphics_processing_units

²https://en.wikipedia.org/wiki/List_of_AMD_graphics_processing_units

The largest full dual problem successfully solved using PermonSVM was the benchmark of suspicious URL prediction [18] with more than 1.6 million training samples and over 3 million features. The related dataset is slightly unbalanced with a ratio of classes 1 : 2, approximately. We report the best-achieved scores³ on a test dataset in Table 6.1.

	Accuracy	Precision	Sensitivity	F_1
Suspicious URL prediction	96.41%	96.25%	93.16%	94.68%

Table 6.1: Attained the performance scores of a classification model on a test data set trained employing a full dual ℓ_2 -loss SVM formulation on the URL data set. Solver: MPRGP/SMALXE (default settings), $C = 1e - 5$.

The underlying MPRGP/SMALXE solver effectively scales up to 144 cores, depicted in Figure 6.1. **That makes PermonSVM a unique machine learning library.**

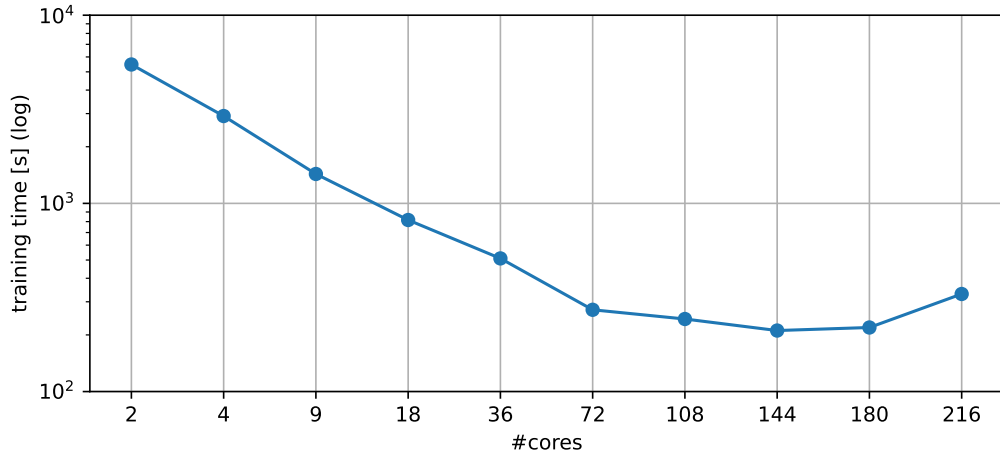


Figure 6.1: URL dataset, dual QP-SVM (ℓ_2 -loss), $C = 1e-5$, MPRGP strong parallel scalability. Tested on BARBORA cluster (IT4Innovations). Training times for each case are summarized in Table 6.2.

#cores	2	4	9	18	36	72	108	144	180	216
training time (s)	5471	2912	1433	816	510	272	243	211	219	330

Table 6.2: Training times of a classification models (a full dual ℓ_2 -loss SVM) on a different number of cores. Solver: MPRGP/SMALXE (default settings), $C = 1e - 5$.

³Recall. MPRGP/SMALXE are designed as deterministic solvers. However, floating point operations such as addition and multiplication are not associative. Thus, running training models on different numbers of cores gives us slightly different performance scores of models.

Our libraries are developed as an open-source project under the BSD 2-Clause Licence. We are grateful for the success that PermonSVM as well as the whole PERMON libraries are incorporated as external software associated with the PETSc framework, see Figure 6.2.

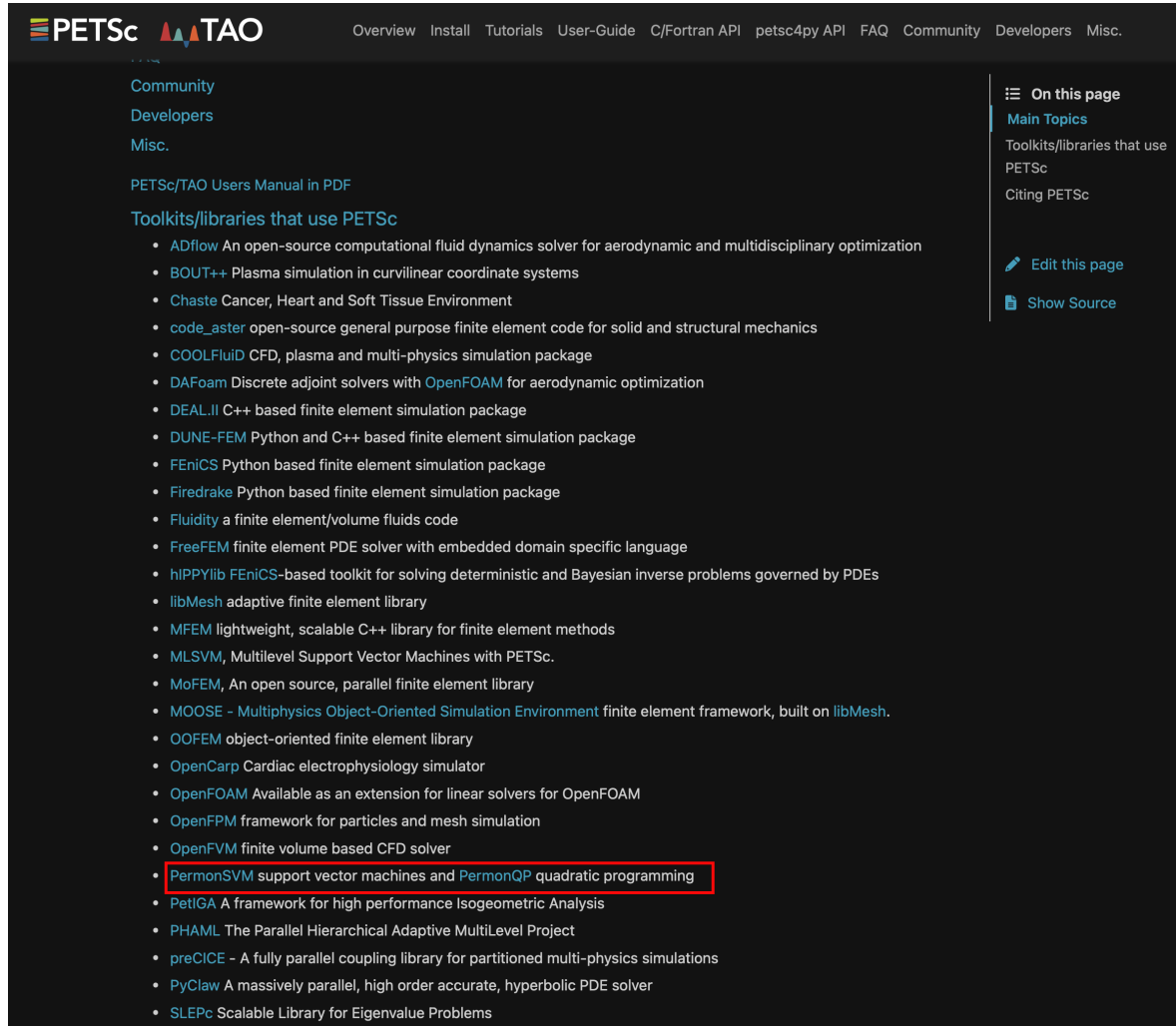


Figure 6.2: PermonSVM and PermonQP are mentioned as external software libraries, which use PETSc. On-line available on the following PETSc webpages <https://petsc.org/release/#toolkits-libraries-that-use-petsc>.

6.2 Parallel data loading

This section briefly introduces and provides short comments on the HDF5 file format, which is currently supported in the PETSc framework. Our tool PermonSVM uses it mainly for parallel I/O operations for dense and sparse samples stored as matrices; note that labels are then stored as a vector.

In the scientific community and industry, HDF5 is widely used for supercomputing or ML applications and is designed for keeping and organizing large volumes of data in N-dimensional arrays. The National Center for Supercomputing Applications developed it at the University of Illinois and is currently maintained by the HDF Group. This is a non-profit organization which ensures the continued development of the HDF5 file format and accessibility of data stored in HDF. The HDF5 format is based on two major types of objects:

- **datasets** represented by multidimensional data array,
- **groups** are container structures which can hold other groups or data arrays.

Metadata (light data) is then stored as attributes (user-defined and named) attached to datasets or groups. Resources in an HDF5 file can be accessed using POSIX-like syntax, such as `/group1/group2/dataset`. The great advantage of this file format is efficient parallel scalable I/O operations using MPI-IO. This is the main reason why we chose it for our ML applications, e.g. wildfires localization in Alaska introduced in Chapter 7. Moreover, it provides performance similar to that of employing loaders for the PETSc binary files. However, they are more flexible for other ML workflows or frameworks, e.g. PyTorch through the custom Dataset implementation, see <https://pytorch.org/vision/stable/datasets.html>.

I initially implemented a loader supporting the HDF5 format in PermonSVM during my research stay at the University of Edinburgh in 2018 (from February to May). After my return to the Czech Republic, Václav Hapla helped me to port this implementation to the PETSc framework [72, 73].

The new loader functionality was done for loading the HDF5 files having a structure of the MATLAB file (.mat) version 7.3 for sparse and, after a while, dense matrices. These MATLAB files are the HDF5 favoured file format. We mentioned it in the official PETSc reference manual and you can find a more related information on-line on the following PETSc webpages <https://petsc.org/release/manualpages/Mat/MatLoad/>, where you can also find a manual page for matrix loaders implemented in the PETSc framework.

6.3 Application programming interface

PERMON and PermonSVM are designed to use the same coding style as PETSc, as mentioned earlier in this chapter. This section introduces two typical showcases of PermonSVM using API for C programming language. The first illustrates a fundamental functionality for programming a classifier that predicts sample categories (a hard classifier); this simple program is in Code 6.1. The second one is presented in Code 6.2, which demonstrates a program of a probability classifier.

```
MPI_comm    comm = PETSC_COMM_WORLD;
SVM          svm;
PetscViewer v;

char         f_train[PETSC_MAX_PATH_LEN] = "url.h5";
char         f_test[PETSC_MAX_PATH_LEN] = "url.t.h5";

PetscCall( SVMCreate(comm,&svm) );
PetscCall( SVMSetType(svm,SVM_BINARY) ); /* Binary classifier */
PetscCall( SVMSetFromOptions(svm) );

PetscCall( PetscViewerHDF5Open(comm,f_train,FILE_MODE_READ,&v) );
PetscCall( SVMLoadTrainingDataset(svm,viewer) );
PetscCall( PetscViewerDestroy(&v) );

PetscCall( PetscViewerHDF5Open(comm,f_test,FILE_MODE_READ,&v) );
PetscCall( SVMLoadTestDataset(svm,v) );
PetscCall( PetscViewerDestroy(&v) );

PetscCall( SVMSetHyperOpt(svm,PETSC_TRUE) );
PetscCall( SVMSetNfolds(svm,3) );

PetscCall( SVMTrain(svm) );
PetscCall( SVMTest(svm) );

PetscCall( SVMDestroy(&svm) );
```

Code 6.1: Showcase of using PermonSVM API for training a binary classifier of SVM type.

One of the powerful features of the PETSc framework is a mechanism for runtime options. It enables users to modify the routine parameters and object options at runtime, making it desirable for many academic and industry applications. The PERMON and also PermonSVM inherit this functionality from the PETSc framework. In the case of PermonSVM, a user can set up for example a type of loss function, e.g. `-svm_loss_type L2`, set problem type such as a complete (introduced in Section 3.1) or relaxed-bias (introduced in Section 3.3) using `-svm_binary_mod 1`, i.e. a complete problem formulation, or set misclassification penalty C using `-svm_C`, setting up and others. Programmatically, the setting options for the SVM object and all inner objects, such as QPS or QPC, are called using a function called `SVMSetFromOptions`. You can find other examples in the

README in the PermonSVM GitHub repository, available on-line on the following link <https://github.com/permon/permonsvm/blob/master/README.md>.

```
MPI_comm      comm = PETSC_COMM_WORLD;
SVM            svm;
PetscViewer v;

char          f_train[PETSC_MAX_PATH_LEN] = "url.h5";
char          f_test[PETSC_MAX_PATH_LEN] = "url.t.h5";

PetscCall( SVMCreate(comm,&svm) );
PetscCall( SVMSetType(svm,SVM_PROBABILITY) ); /* Probability model */
PetscCall( SVMSetFromOptions(svm) );

PetscCall( PetscViewerHDF5Open(comm,f_train,FILE_MODE_READ,&v) );
PetscCall( SVMLoadTrainingDataset(svm,viewer) );
PetscCall( PetscViewerDestroy(&v) );

PetscCall( PetscViewerHDF5Open(comm,f_train,FILE_MODE_READ,&v) );
PetscCall( SVMLoadCalibrationDataset(svm,viewer) );
PetscCall( PetscViewerDestroy(&v) );

PetscCall( PetscViewerHDF5Open(comm,f_test,FILE_MODE_READ,&v) );
PetscCall( SVMLoadTestDataset(svm,v) );
PetscCall( PetscViewerDestroy(&v) );

PetscCall( SVMSetHyperOpt(svm,PETSC_TRUE) );
PetscCall( SVMSetNfolds(svm,3) );

PetscCall( SVMTrain(svm) );
PetscCall( SVMTest(svm) );

PetscCall( SVMDestroy(&svm) );
```

Code 6.2: Showcase of using PermonSVM API for training a probabilistic classifier.

Chapter 7

Wildfires localization in Alaska

Global climate change is increasing the frequency and intensity of ecological disturbance. This is particularly true in high latitudes. The scientists from the US space agency NASA in the ABoVE project¹ have long observed that these changes affect the frequency and intensity of natural disasters in the northern regions of Alaska. Wildfires are one important source of disturbance, and can significantly affect forest carbon balance. Despite their importance, it can be difficult to accurately quantify the effects of wildfire in places such as boreal forests that are far from human habitation and infrastructure. Data from remote sensing platforms and observatory networks can be of great use of this task, however these data sets can be vast, and analyzing them can require powerful computing resources and tools that are designed to fully utilize them. Text source: [66].

In this chapter, we will introduce a joint work with world-leading research institutes, aimed at the identification of natural hazards by means of employing remote sensing and ML techniques. Specifically, we focus on training semantic segmentation models using spatio-temporal satellite data sets. The main target is developing models for identifying and localizing wildfires in the boreal forests in Arctic regions in Alaska – an example is depicted on the next page in Figure 7.1. As the PERMON team, we cooperate on this application with the Argonne National Laboratory² (USA) represented by Dr Richard Mills,³ who is also a co-supervisor of this thesis, and Oak Ridge National Laboratory⁴ (USA), represented by Dr Jitendra Kumar⁵ and (formerly) Dr Zachary Langford⁶ (now at Databricks, Inc. at the time of this doctoral thesis submission).

¹<https://above.nasa.gov>

²<https://www.anl.gov>

³<https://www.anl.gov/profile/richard-tran-mills>

⁴<https://www.ornl.gov>

⁵<https://www.ornl.gov/staff-profile/jitendra-kumar>

⁶<https://langfordzl.github.io>

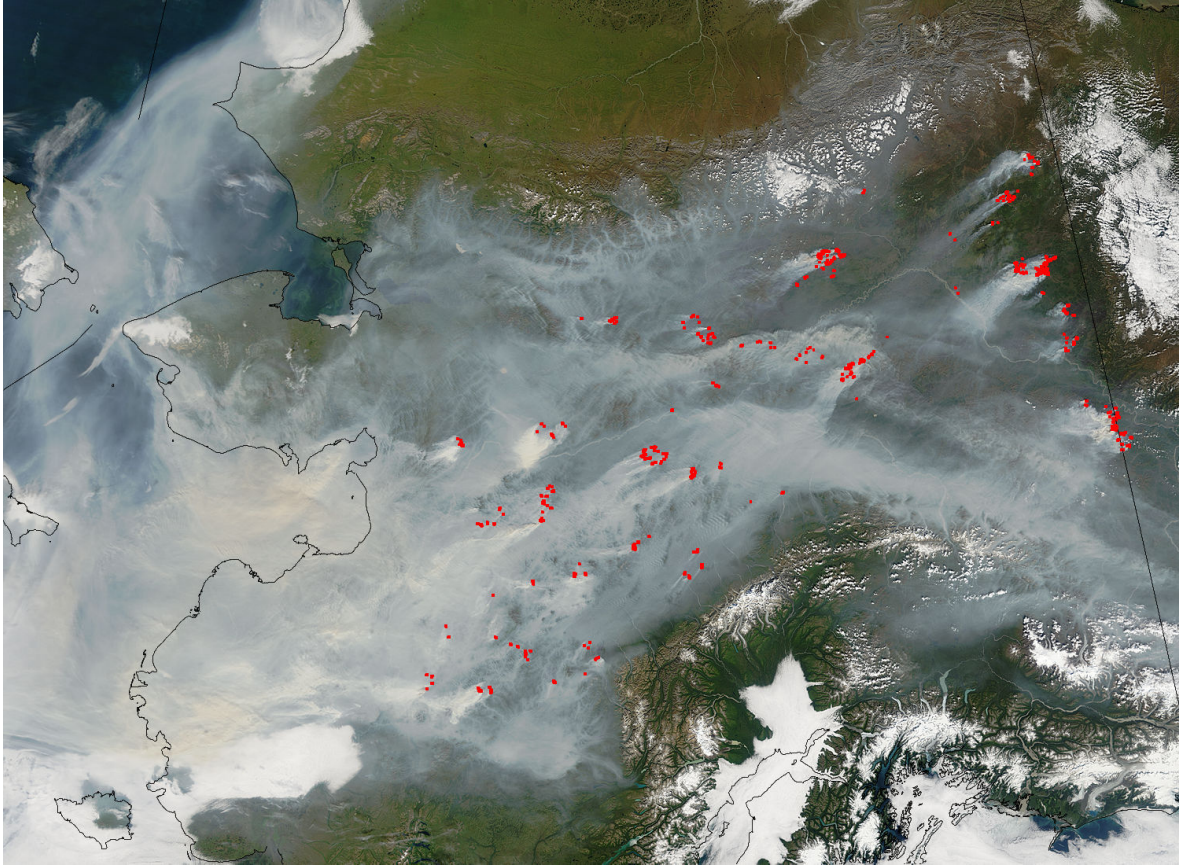


Figure 7.1: On August 14, 2005, Terra satellite (operated by NASA) captured this image of wildfires raging across the width of Alaska using the Moderate Resolution Imaging Spectroradiometer (MODIS) instrument. Smoke from scores of fires (marked in red) filled the state broad central valley and poured out to sea. More than a hundred fires were burning across the state as of August 14. *This image was downloaded from the NASA websites [74] using the web archive service (<https://web.archive.org>). A caption of this image was slightly modified.*

We have presented our results at the premier international conferences, namely: *the AGU Annual Meeting* (USA), *the IALE North America Annual Meeting* (USA), *the IEEE International Conference on Data Mining* (USA), *the SIAM Conference on Computational Science and Engineering* (NL). As an author of this thesis, I gave an invited talk at the Argonne National Laboratory. In this chapter, we summarize main highlights of our work, approaches, and results. Most of them were presented in the last conferences and talks in 2023.

7.1 Motivation

During our collaboration, we focus on efforts that are primarily motivated by using the PETSc framework [19, 67] and the PERMON extension called PermonSVM, which we earlier introduced in Chapter 6. This helps us to build an ML tool, which can run on modern, distributed memory parallel, GPU-accelerated HPC architectures [68]. Then, we are able to train ML models across a cluster or a supercomputer with many nodes, and nodes that employ GPUs to deliver most of their computational power, and scale to very large data sets. A parallel SVM implemented in PermonSVM enables us to train models on large remote sensing data sets, which is originally “*ultra-scale*,” for automated identification of areas burned by wildfires.

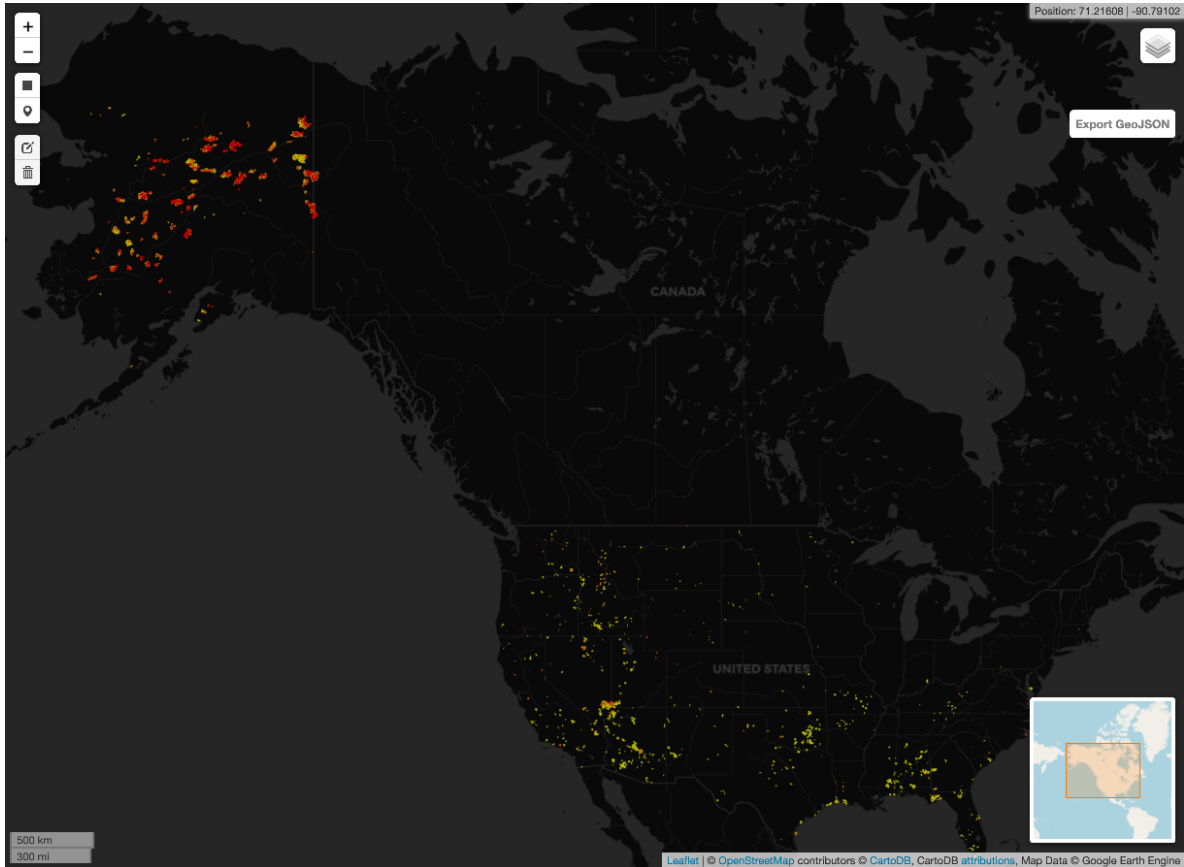


Figure 7.2: Visualization of aggregated wildfire localization over 2005 in Alaska and the continental USA using Monitoring Trends in Burn Severity (MTBS) product. Source: *the visualization was generated employing our in-house software for natural hazards based on ML techniques* [52].

Recall: Wildfires are a type of disturbance that can significantly affect forest carbon balance, change ecosystem composition, and increase the chances of further burning in subsequent years. Despite their importance, no one is synoptically mapping all fires across North

America. The Monitoring Trends in Burn Severity (MTBS) product [75] only maps fires over 1,000 acres in the Western US and over 500 acres in the East, see visualization of aggregated localization of wildfires over 2005 in Figure 7.2 (on the previous page). However, small fires are the most frequent and taken together with a larger ones may be the most important. Producing MTBS involves intensive human labour and is about 2 years behind. Therefore, using ML models could automate wildfire identification and additionally provide more complete coverage.

One of the real reasons why we need GPU-enabled, distributed-memory parallel machine learning tools arises from the nature of remote sensing data: *increasing abundance, fidelity, and richness of high-resolution spatio-temporal data sets from observatory networks and remote sensing platforms also pose true “big data” possibilities and challenges, especially when combined with Earth System Model (ESM) outputs.* Exascale Earth ESM simulations hold the potential for achieving “ultra-high” resolution and process representation combined with an increase in data volume, e.g. high-res CESM simulations on Sunway TaihuLight supercomputer⁷ produced 360 terabytes (TB) of output [76]. The Google Earth Engine platform [22] offers access to a repository of over 50 petabytes (PB) of satellite data available to analysis. Compared to large language models, the GPT-4 model [14] was trained on 45 TB of data volume.

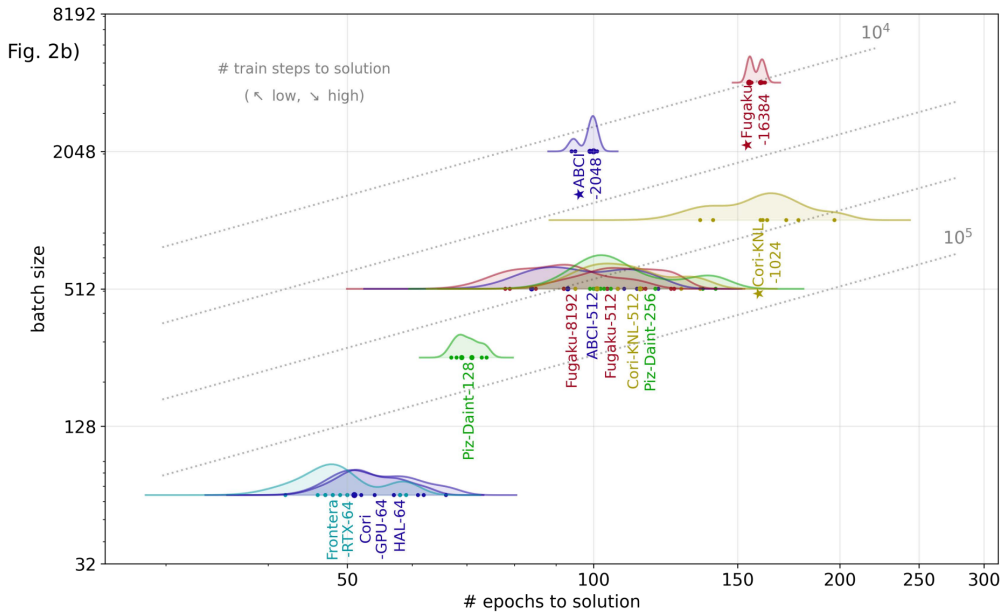


Figure 7.3: Epoch scaling for CosmoFlow deep learning benchmark. *This graph was adopted from Farrell et al. 2021, “MLPerf HPC: A Holistic Benchmark Suite for Scientific Machine Learning on HPC Systems” [77].*

⁷https://en.wikipedia.org/wiki/Sunway_TaihuLight

Performance portability in PETSc

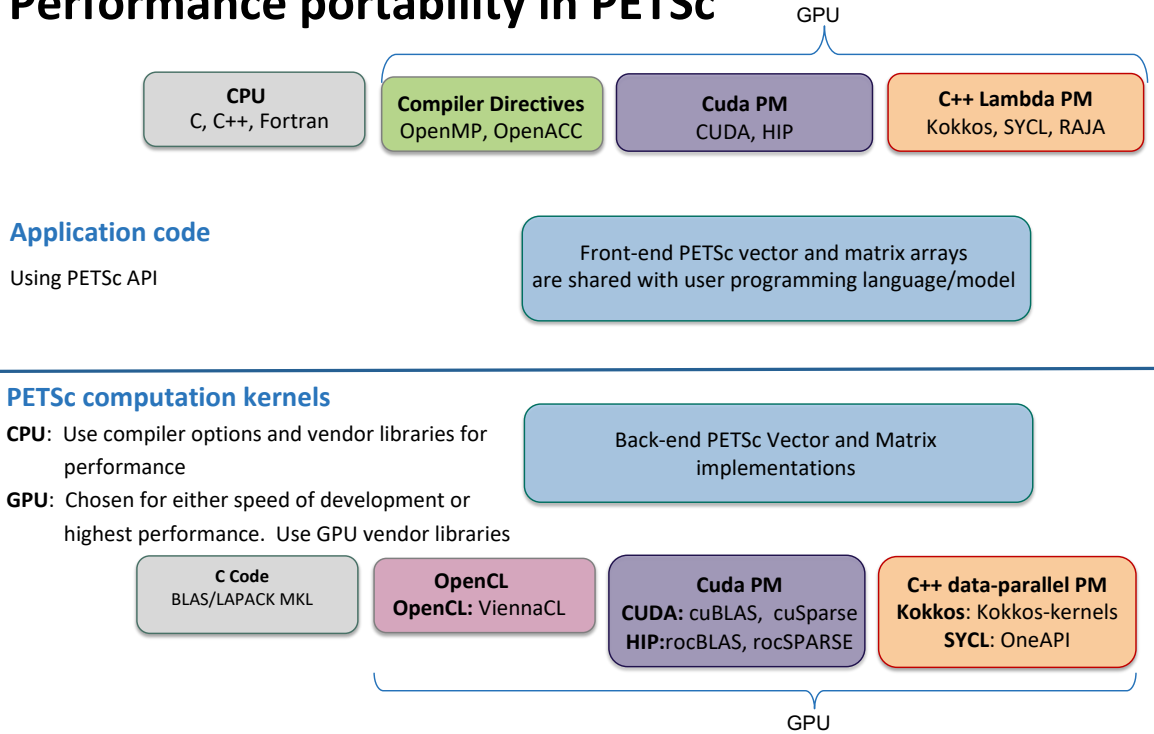


Figure 7.4: A conceptual diagram illustrating the separation between supported GPU programming models for user code and the PETSc backends. *An image source: R. T. Mills, et al. Toward performance-portable PETSc for GPU-based exascale systems [68].*

Existing tools are generally not designed to scale to such data set sizes and typically provide shared-memory task-level parallelism, or the level of high-performance orchestration, which is based on running multiple container instances of an application, e.g. using Kubernetes.⁸

Distributed memory approaches are usually “*data-parallel*” in the case of TensorFlow [78] and PyTorch [79] frameworks. This means that a model is replicated across many nodes, training samples are assigned to nodes, and training occurs using a local batch. Batch size limitations could be problematic, see Figure 7.3. Therefore, “*extremely*” large models using data sets in order of peta bytes could not be effective to train such models using widely-used frameworks for deep learning mentioned above, since they do not provide an infrastructure for model parallelism – major effort would be implement this mechanism.

So, we have ended up adopting a classical ML approaches. We have chosen SVM models, because it seems to work well enough for wildfire localization. The second reason is really fast prediction of wildfire localization, which is basically done using matrix-vector multiplication. Since SVM is basically simple model that are considered as a single layer perceptron, the data

⁸<https://kubernetes.io>

must be transformed using feature engineering approaches. These approaches are discussed later in this text in Section 7.2.

Thus, we decided to build our ML tools (including PERMON and PermonSVM) on the top of the PETSc framework. PETSc provides several building blocks useful for ML: *high-performance linear algebra, advanced numerical optimization methods, ODE integrators, etc.* It is designed for parallel scalability, and has been used in many groundbreaking supercomputer simulations including five that were awarded Gordon Bell Prizes, you can see list of awards won by PETSc or its user on the following page <https://petsc.org/release/miscellaneous/prizes/>. In recent years, PETSc has added robust support for GPU-accelerated architectures. It is simple to use them by employing the appropriate GPU back-ends in PETSc, see Figure 7.4 on the previous page.

7.2 Data processing

Since we investigate the abilities of standard ML models, it is necessary to (reasonably) represent data using feature extraction approaches. Extracting significant features from input data, which are multispectral satellite images in our case, is an essential part of data analysis. The analysis also includes inspection of relevant transformations of data into so-called latent spaces, which leads to an optimization in a change of basis, and this transformation appears as a projector of a specific type in a problem formulation.

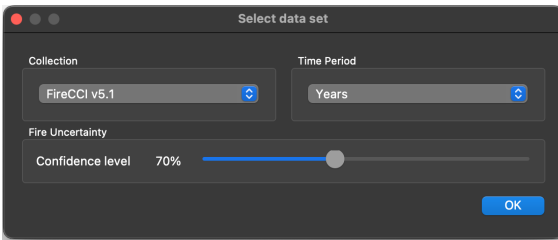
To obtain efficiently the satellite data, we have developed software in Python using Google Cloud and Google Earth Engine services for streaming and downloading data – graphical user interface of this tool is depicted in Figure 7.5. Equipped with a user-friendly graphical interface for selecting region(s) of interest (ROI), the software provides an visualization of accumulated wildfire localization over a specified period. It simplifies us manipulation with fire map collections and drawings of bounding box/boxes that are used to define ROI(s) for downloading satellite data and fire maps from Google Earth Engine. Software supports European Space Agency (ESA) FireCCI⁹ and MTBS fire map collections. We just tested the last one so far.

After downloading satellite images to local storage, we need to clean them first. In practice, this means removing the parts of the image parts that have missing values – note that this could be due to factors such as sensor errors, cloud cover at time of acquisition, etc. These areas are labelled by a value 6 in the MTBS collection. In the case of automated monitoring of wildfires, we also consider temporal information in addition to spatial information. In other words, each pixel is represented as a multidimensional time series. These time series describe the evolution of reflectance represented by the electromagnetic spectrum, the evolution of plant health using vegetation indices such as Normalized Difference Vegetation Index (NDVI) or Enhanced Vegetation Index (EVI), and temperature changes over a given period. The multispectral images are taken from MODIS instrument carried by Terra satellite¹⁰, where one pixel corresponds to a square area of the size $500 \times 500 \text{ m}^2$ (61.77 acres).

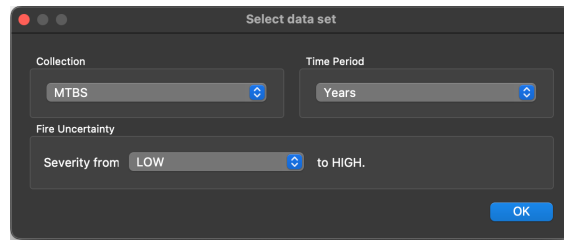
Recall. We train an image segmentation model using classical (standard) ML techniques. Thus, it is necessary to perform time series standardization and the so-called data transformation needed to extract some meaningful features. These transformations may include applying filtering techniques such as Savitzky-Golay or dimension reduction techniques. So far, we have tested dimension reduction using PCA (Principal Component Analysis) in upcoming experiments.

⁹<https://climate.esa.int/en/projects/fire/about/>

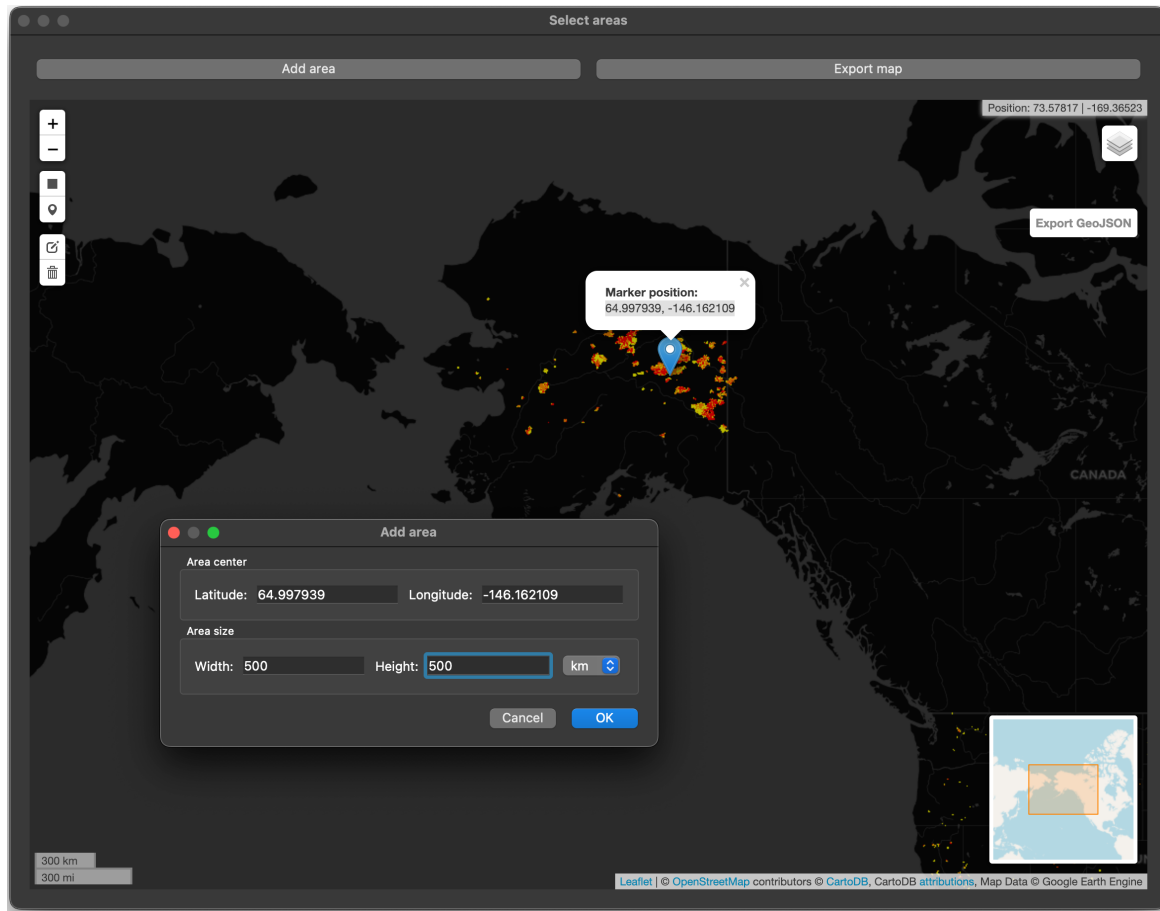
¹⁰<https://terra.nasa.gov>



(a) Dialog window for FireCCI data.



(b) Dialog window for MTBS data.



(c) Dialog window for region selection.

Figure 7.5: Graphical user interface of software for streaming and downloading data based on Google Cloud and Google Earth Engine.

7.3 Benchmarks

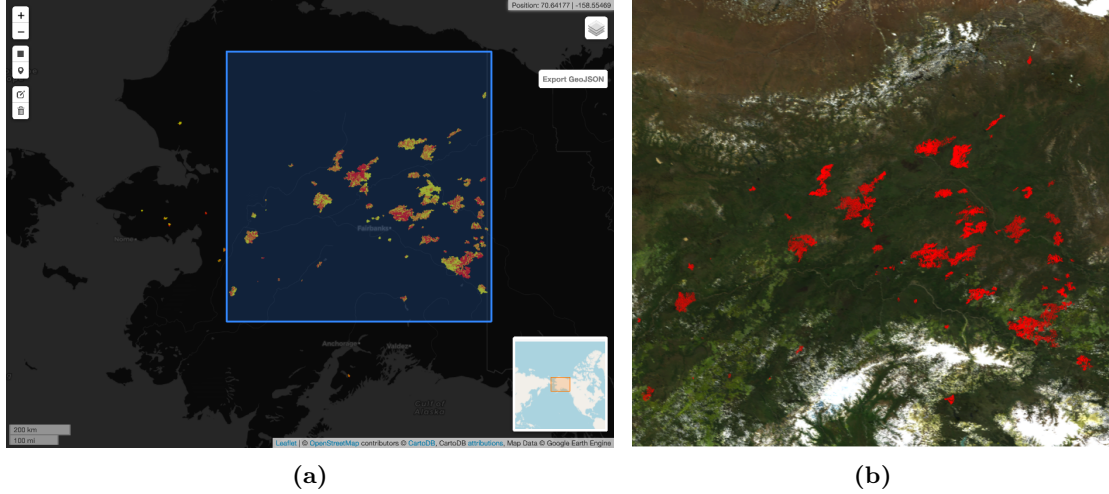


Figure 7.6: Alaska wildfire season in 2004 (aggregated data), area centered at $N65^{\circ} 44' 55.259''$ $E149^{\circ} 53' 50.859''$, area size $\approx 722,500 \text{ km}^2$, projection EPSG3338. Image size 1918×1780 (space domain).

This section presents the results of automated wildfire localization using semantic segmentation models. Recall. Semantic segmentation is a computer vision task associated with supervised image classification at a pixel level. It means that every pixel is assigned to its own category, creating a segmentation mask. A fire map represents this mask in our case.

Current state-of-the-art approaches are based on neural networks of U-Net type architectures processing RGB images – typically satellite images in the visible spectrum of light or aerial photos. There, the pre-trained encoder in U-Net extracts features and patterns from spatial images, and the decoder projects these lower-resolution features onto the pixel space in higher resolution to get a dense classification. The architecture associated with this type of neural network based on processing RGB images is discussed in more detail in [80].

In the following text, we will introduce different approaches based on employing standard classification models of the SVM type used for wildfire localization on multispectral satellite data including surface reflectance and temperature combined with vegetation indices. We consider both spatial and temporal information. Data were downloaded from the Google Earth Engine repositories using our in-house software, which was earlier mentioned in Section 7.2.

For our experiments, we downloaded satellite data captured in 2004 belonging to an area of size approximately equal $722,500 \text{ km}^2$, depicted in Figure 7.6. Note that the 2004 wildfire season was the worst on record in the U.S. state of Alaska in terms of area burned. About $27,000 \text{ km}^2$ of Alaska regions were affected by wildfires.

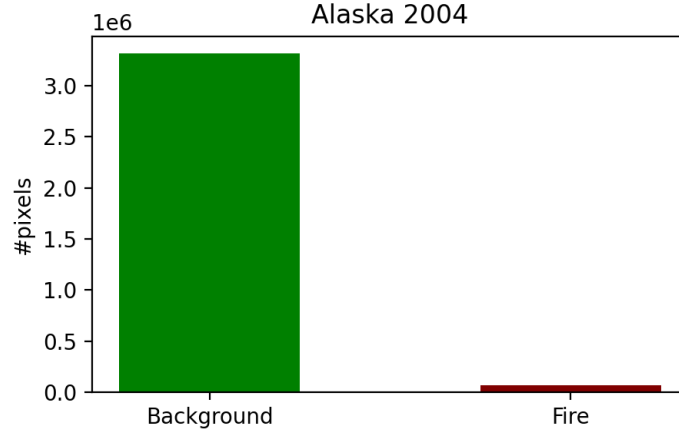


Figure 7.7: Alaska wildfires season 2004: highly unbalanced data set 3,317,870 (97.92%) of background pixels and 70,631 (2.08%) pixels are associated with wildfire regions.

As mentioned in the previous section, we used boundaries collected from the MTBS product for labelling pixels representing areas, which are affected by fire. Anyway, other pixels are considered as background. From it, we get highly unbalanced data set having 3,317,870 of background pixels (97.92%) and 70,631 of wildfire pixels (2.08%), see Figure 7.7.

Further, we split the image area (1918 x 1780 px) horizontally into training and test data sets in a ratio 3 : 1. A larger part is used for training models, and a smaller one is used for evaluating their performance. Time series length is 1 year (46 eight-day periods). Since we fuze data of reflectance (7 spectral bands), temperature, and 3 vegetation indices,¹¹ the dimension of the time series is then 11.

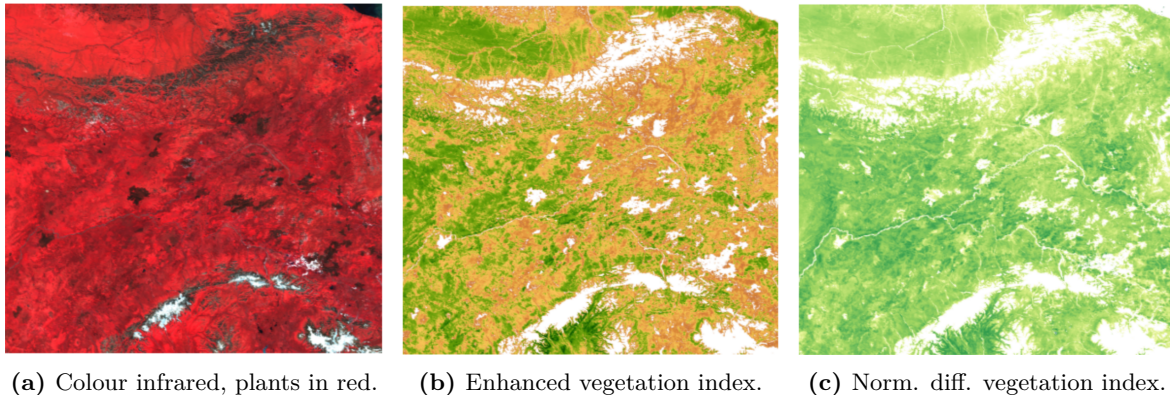


Figure 7.8: Vegetation distribution in Alaska (August 8, 2004).

¹¹Vegetation health is represented by NDVI, EVI and EVI2 indexes.

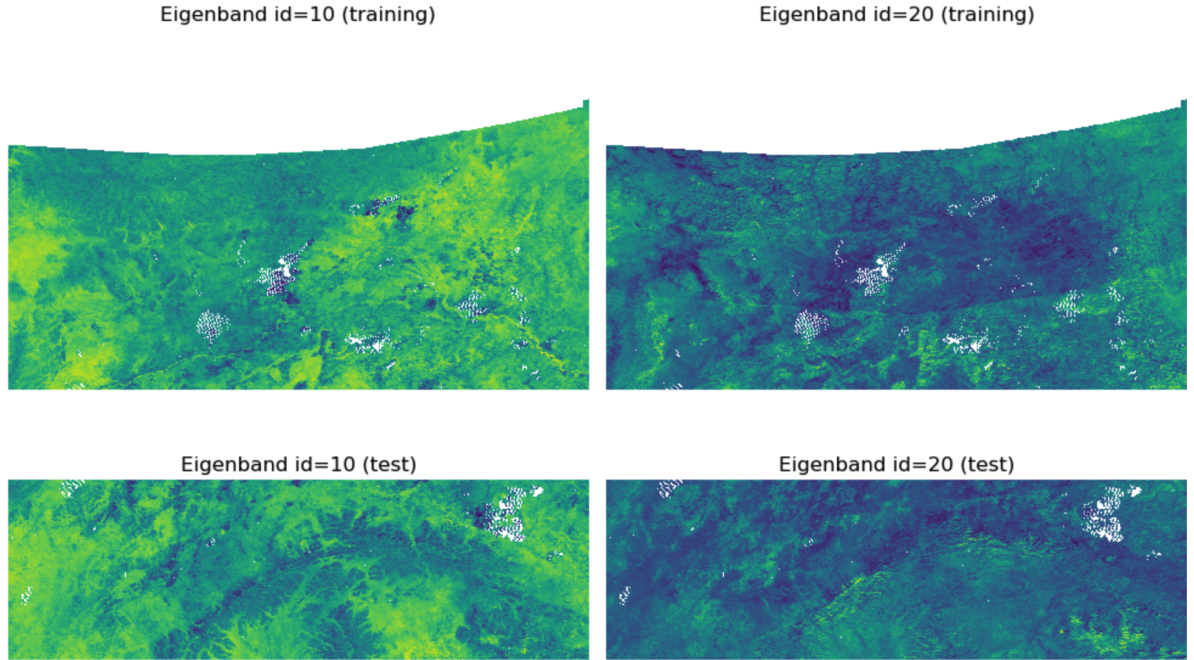


Figure 7.9: Example of eigenbands. White pixels are related to areas removed from the training and test data sets. These pixels are related to areas not monitored during the sensor issue (removed from MTBS), or the northeast part of Arctic Alaska.

We are deliberate about what data and features we are feeding to the SVM training procedure in the following experiments. First, we clean data such that we drop pixels not observed in MTBS (labelled by 6). By this approach, we also eliminate gap-filling NaNs in the MODIS products. Since pixels are represented by time series, we need to standardize them within each of the 11 features (convert to z-scores).

Then, we smoothed out the time series using the Savitzky–Golay (SG) filter. In the following experiments, we set polynomial order and window length to 2 respective 10 as parameters of the SG filter. We tested various combinations of a window length and a polynomial order, and it seems that these parameters are reasonable for our data. Last, we apply dimensionality reduction using PCA per each feature, retaining enough components to explain data ranging from 80% to 95% of variance, denoted as *var*. We get from 5 to 28 eigenbands per feature, see Table 7.1 and Table 7.2. In following experiments, we also study influence of each one transformation in our experiments. Example of two eigenbands are depicted in Figure 7.9.

We trained the series of semantic segmentation models employing our PermonSVM tool on NVidia DGX-2 machine at IT4Innovations, including fine-tuning performed by grid-search and cross-validation (HyperOpt) combining transformations mentioned above. DGX-2 is two PFlop/s system equipped with two high-end x86_64 processors (Intel Xeon Platinum, 8168) and 16 NVIDIA V100-HBM2 GPUs.¹² We utilized all 16 GPUs in the following experiments.

¹²The technical information about NVidia DGX-2 is mentioned in ???. Further information can be found on IT4Innovations webpages – <https://www.it4i.cz/en/infrastructure/nvidia-dgx-2>.

To avoid additional computational overhead, we used relaxed-bias approaches for both ℓ_1 -loss and ℓ_2 -loss formulations; their advantages are discussed in Section 5.5.3. We choose the MPRGP algorithm as a QP solver for optimization problems arising from QP-SVM formulations. An expansion of an active set was performed using a projected CG for reasonably represented data (reduced using PCA and smoothed out using the SG filter). Otherwise, a fixed expansion step is computed; $\Gamma = 10$ is used in the proportioning test. The best penalty C_{best} is selected using a HyperOpt approach (with enabled warmstart) from the following set:

$$\mathbb{S}_C = \{10^i \mid i \in \{-3.0, -2.9, -2.8, \dots, -2.2, -2.1, -2.0\}\}. \quad (7.1)$$

The volume of data used in the following experiments is nearly 67 GB, each combination of transformation is stored separately in HDF5 file and uploaded to cluster. The performance of attained models are summarized in Table 7.1 and Table 7.2. The visualization of wildfire predictions using the best attained model is depicted in Figure 7.10.

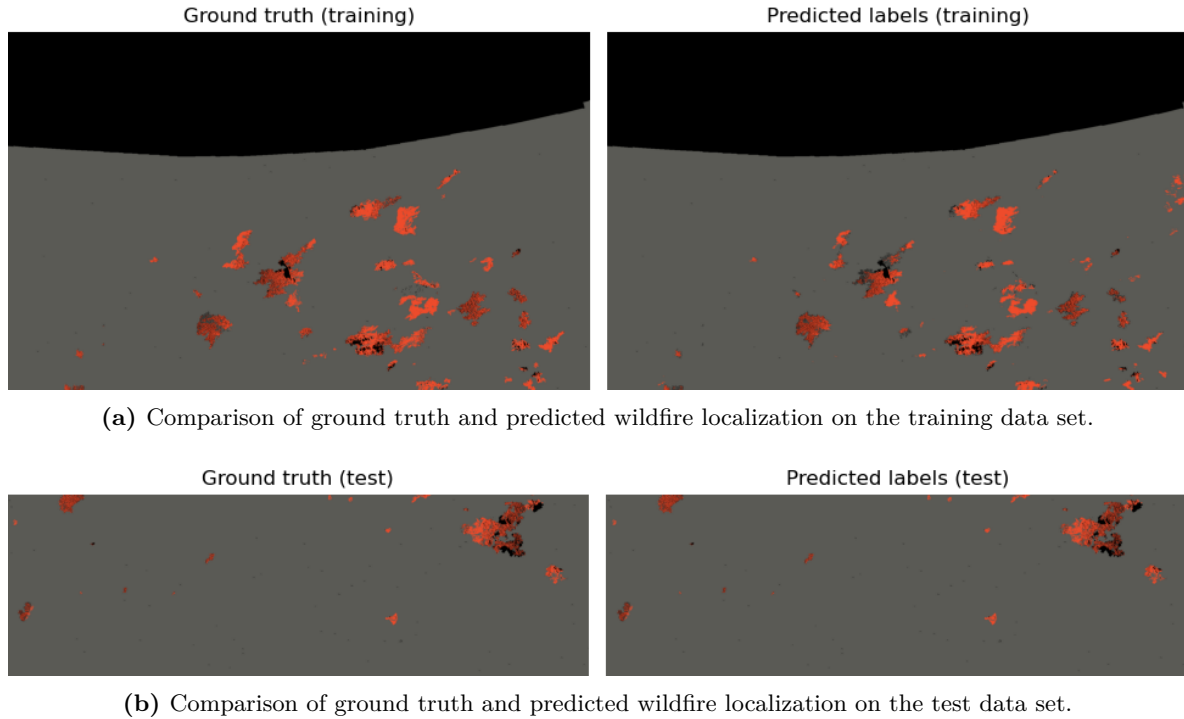


Figure 7.10: Side by side comparison of ground truth and predicted localization of wildfires on training and test data set obtained using the best model trained using ℓ_2 -loss SVM, which is mentioned in Table 7.2. Black pixels are related to the areas removed from training and test data sets.

transformation			#features (len×dim)	#Hessian mult. (train. time [s])	C_{best}	model scores (test)				training time + HyperOpt [s]	
z-score	PCA / var.	SG filter				sensitivity	precision	F_1	mIoU		
			46×11	DIVERGED							
×			46×11	DIVERGED							
		×	46×11	DIVERGED							
×		×	46×11	6808 (29.24)	5e−3	0.8672	0.5473	0.5575	0.4919	3279.65	
×	×	/ 0.80	17×11	1618 (2.29)	3e−3	0.8904	0.5467	0.5537	0.4865	2977.87	
×	×	/ 0.80	×	5 × 11	1993 (2.99)	8e−3	0.8903	0.8207	0.8520	0.7698	1915.12
×	×	/ 0.90		23 × 11	1032 (1.78)	1e−3	0.8634	0.5464	0.5558	0.4905	3566.99
×	×	/ 0.90	×	8 × 11	1792 (2.76)	8e−3	0.8956	0.6576	0.7219	0.6348	2156.82
×	×	/ 0.95		28 × 11	1967 (3.46)	8e−3	0.7269	0.8563	0.7768	0.6888	3950.33
×	×	/ 0.95	×	11 × 11	1503 (2.36)	3e−3	0.9266	0.5929	0.6408	0.5638	2821.76

Table 7.1: Comparison of attained models for wildfires localization trained using 16 GPUs NVIDIA Tesla V100- HBM2 (DGX-2, IT4Innovations), 16 MPI ranks. Solver: MPRGP, an expansion step is performed using the projected CG step, $\Gamma = 100$ in proportion criterion, $rtol = 0.1$, loss type ℓ_1 -loss, relaxed-bias formulation, single precision. The best penalty C was selected using hyper-parameter optimization, which was performed employing cross-validation combined with grid-search, from a set \mathbb{S}_C , earlier defined in (7.1).

transformation			#features (len×dim)	#Hessian mult. (train. time [s])	C_{best}	model scores (test)				training time + HyperOpt [s]	
z-score	PCA / var.	SG filter				sensitivity	precision	F_1	mIoU		
			46×11	DIVERGED							
×			46×11	3874 (14.01)	1e−3	0.7420	0.5614	0.5898	0.5301	7003.38	
		×	46×11	DIVERGED							
×		×	46×11	2541 (6.67)	1e−3	0.8722	0.5484	0.5598	0.4939	2739.81	
×	×	/ 0.80	17×11	455 (< 1)	1e−3	0.8770	0.7000	0.7598	0.6704	872.03	
×	×	/ 0.80	×	5×11	465 (< 1)	1e−3	0.8683	0.8587	0.8634	0.7840	481.15
×	×	/ 0.90	23×11	436 (< 1)	1e−3	0.9139	0.6144	0.6715	0.5904	1055.83	
×	×	/ 0.90	×	8×11	446 (< 1)	1e−3	0.9324	0.6913	0.7625	0.6723	771.88
×	×	/ 0.95	28×11	405 (< 1)	1e−3	0.9104	0.7625	0.8195	0.7318	1266.69	
×	×	/ 0.95	×	11×11	436 (< 1)	1e−3	0.9112	0.6786	0.7462	0.6569	1022.95

Table 7.2: Comparison of attained models for wildfires localization trained using 16 GPUs NVIDIA Tesla V100- HBM2 (DGX-2, IT4Innovations), 16 MPI ranks. Solver: MPRGP, an expansion step is performed using the projected CG step, $\Gamma = 100$ in proportion criterion, $\text{rtol} = 0.1$, loss type ℓ_2 -loss, relaxed-bias formulation, single precision. The best penalty C was selected using hyper-parameter optimization, which was performed employing cross-validation combined with grid-search, from a set \mathbb{S}_C , which was earlier defined in (7.1).

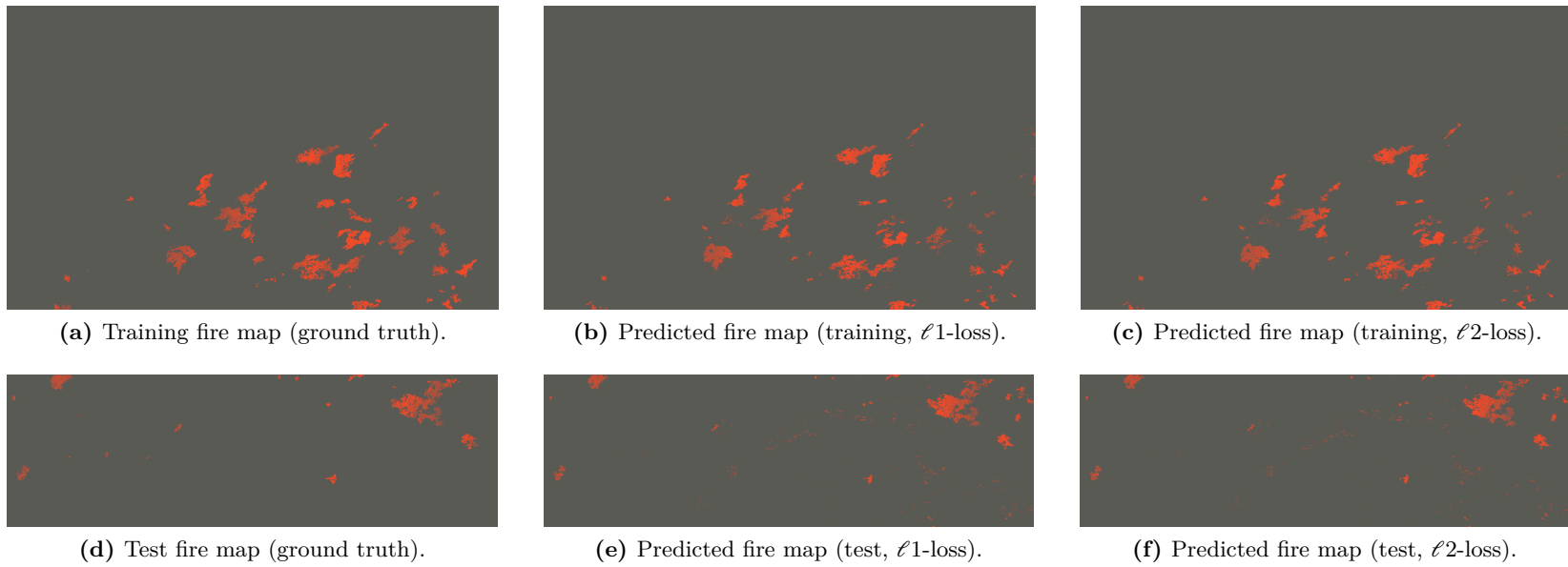


Figure 7.11: A side by side comparison of wildfire predicted by means of the best ℓ_1 -loss and ℓ_2 -loss models and the ground truth on the training and test data sets.

Comparing results summarized in Table 7.1 and Table 7.2, we can see that dimensionality reduction using PCA is necessary to train models for both ℓ_1 -loss and ℓ_2 -loss (relaxed-bias) formulations. Without this reduction in preprocessing step, the solver diverged in 3 and 2 cases for ℓ_1 -loss respective ℓ_2 -loss formulation, or we attained a model having performance scores around 0.5 in the F_1 and mIoU metrics. Recall: Models having these metrics around 0.5 are considered as “a random classifier” or “a random guess” in ML community. It means that these models showed no better accuracy than a coin flip.

Let us analyze the remaining models trained using the ℓ_1 -loss formulation. It seems reasonable to combine the SG filter with data reduction using PCA to speed up the training of models, which could perform reasonably well. We tested the following values 0.8, 0.9, and 0.95 for retained variance. As mentioned above, we fixed values for the SG filter such that a window length is 10 and polynomial order is 2.

Next, we can see that increasing variance causes the model to perform worse when data was smoothed using the SG filter. It seems we smoothed out data quite aggressively, and PCA prevents the classifier from overfitting, thus it is reasonable to drop out reasonable amount of information from data. When we only performed reduction data using PCA, the model performed better when we increased retained variance. However, the best-attained model performed worse than the previous case (PCA + SG filter) - difference is 8.1%. Developing of the mIoU score depending on retained variance for PCA and PCA combined with the SG filter, is depicted in Figure 7.12 for regularized ℓ_1 -loss SVM. Developing of the related training times are depicted in Figure 7.13, on the next page.

The best model was trained (including hyper parameter optimization) in 1915s, i.e. 31min 55s, and its performance scores are 0.76 (mIoU) and 0.85 (F_1). For this model, a retained variance was 0.8, and reflectance data was smoothed using the SG filter.

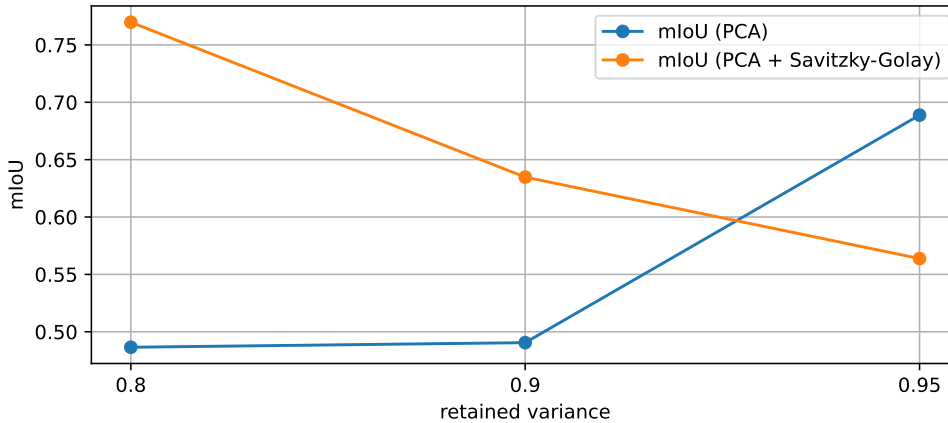


Figure 7.12: Developing of mIoU score depending on retained variance for PCA, and PCA combined with Savitzky-Golay filter (regularized ℓ_1 -loss SVM).

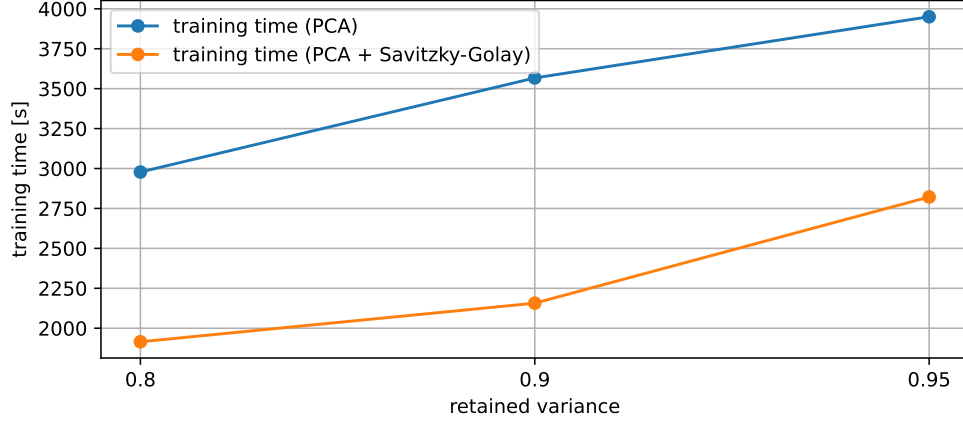


Figure 7.13: Developing of training time depending on retained variance for PCA, and PCA combined with Savitzky-Golay filter (regularized ℓ_1 -loss SVM).

Now, we analyze the models trained using the ℓ_2 -loss SVM. Looking at the results summarized in Table 7.2, we can see a similar nature as in the case of training models using an approach based on the regularized ℓ_1 -loss SVM. The dimensionality reduction using PCA is necessary to train a model; otherwise, a solver diverges in 2 from 4 cases and these two models have performance scores around 0.5, which are basically random classifiers as we mentioned above.

Further, we can observe that PCA prevents overfitting for lower values of a retained variance when data is smoothed using the SG filter. This is similar to the ℓ_1 -loss case, discussed above. Otherwise, we trained models with the performance scores below 0.7 in the mIoU metric. The best model was trained (including hyper parameter optimization) in 481s, i.e. 8min 1s, and its performance scores are 0.78 (mIoU) and 0.86 (F_1). For this model, a retained variance was 0.8, and reflectance data was smoothed using the SG filter – same as for ℓ_1 -loss. Note that it is the best model in the sense of the mIoU score overall.

A visual comparison of prediction of wildfires localization using both ℓ_1 -loss and ℓ_2 -loss models with ground truth is depicted in Figure 7.11. Based on this visualization, we can conclude that these models can perform similarly.

Our model predictions may be more useful if we output probabilities of class membership $P(\text{class} \mid \text{input})$, rather than simply predicting class labels. This allows us to have evaluate confidence in our predictions. We are exploring using *the Platt scaling* (or *the Platt calibration*) to do this. Platt scaling constructs a logistic regression model (using maximum likelihood estimation) to map the SVM output to the posterior probability:

$$P(y = 1 \mid \mathbf{x}) \approx P_{A,B}(y = 1 \mid \mathbf{x}), \quad (7.2)$$

where the parameters A , B determine the slope of the sigmoidal curve respective lateral displacement and $P_{A,B}(y = 1 | \mathbf{x})$ is as follows:

$$P_{A,B}(y = 1 | \mathbf{x}) = \frac{1}{1 + e^{Ah_{\theta}(\mathbf{x}) + B}}. \quad (7.3)$$

To avoid overfitting, an additional training set (the calibration set), CA , of l samples is used:

$$CA := \{(h_{\theta}(\mathbf{x}_1), y_1), (h_{\theta}(\mathbf{x}_2), y_2), \dots, (h_{\theta}(\mathbf{x}_l), y_l)\}. \quad (7.4)$$

The parameters are determined by means of minimizing a binary cross-entropy so that:

$$(A^*, B^*) = \arg \min_{A,B} - \sum_{j=1}^l t_j \ln p_j + (1 - t_j) \ln(1 - p_j), \quad (7.5)$$

where $p_j = P_{A,B}(y_j = 1 | \mathbf{x}_j)$, and t_j is a target probability associated with the sample \mathbf{x}_j :

$$t_j = \begin{cases} \frac{N_p+1}{N_p+2} & \dots y_j = +1, \\ \frac{1}{N_n+2} & \dots y_j = -1, \end{cases} \quad (7.6)$$

where N_p and N_n is number of positive (wildfires) and negative (background) pixels. This optimization problem is not associated with QP, and we solve it using (quasi-)Newton method implemented in TAO component of PETSc. We introduce visualization of our preliminary results in Figure 7.14.

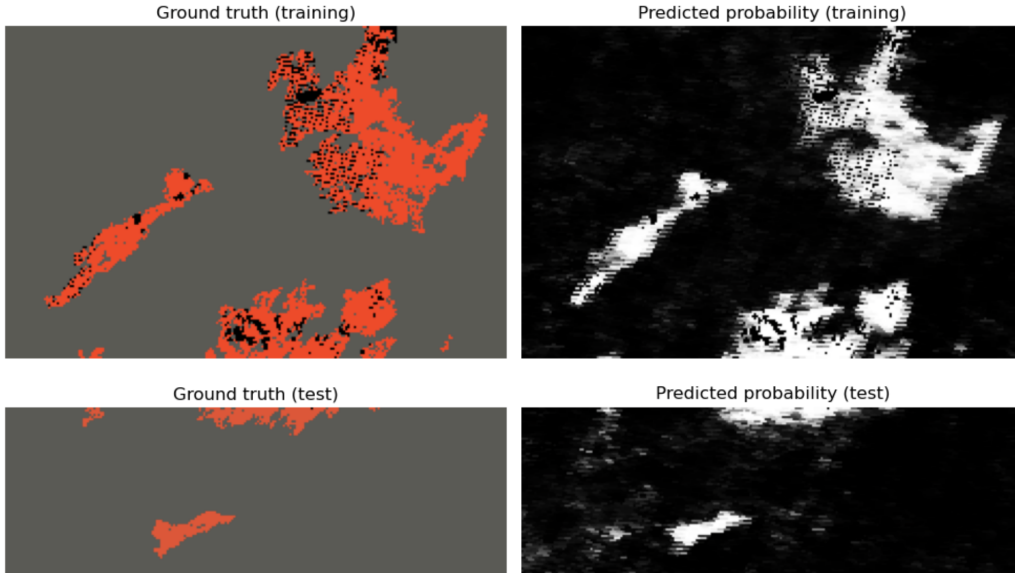


Figure 7.14: Visualization of preliminary results attained by Platt scaling approach.

Part II

Unsupervised learning

Chapter 8

Vector quantification

Robert Sokal and Peter Sneath introduced the concept of numerical taxonomy in the 1960s. After a while, ideas beyond this concept were published in the book called *Numerical Taxonomy. The Principles and Practice of Numerical Classification* [81]. The fundamentals of numerical taxonomy are essentially based on grouping biological systems into clusters such that entities are as similar as possible inside a cluster and various among clusters. Such an idea appears in clustering techniques used in modern data mining approaches. We can say that Sokal and Sneath laid down the foundations for clustering techniques firstly developed in the 1960s and early 1970s.

8.1 Introduction

Techniques of vector quantification [82] belong to a collection of coding techniques used for lossy data correction and compression. They seek a representative codevector of a cluster, thus, they can be directly used for a prototype-based clustering. In case of a fixed number of prototypes $k \leq k^*$, where k^* is an optimal number of prototypes (clusters), it corresponds to the well-known clustering methods of the k -means type, which are based on an idea of minimizing intra-cluster compactness.

The essential idea of k -means goes back to 1950s, when Hugo Steinhaus (a Polish mathematician) published paper, namely *Sur la division des corp materiels en parties* [83], where he formulated and discussed a solution of partitioning a composite material into k homogenous parts. Apparently, Steinhaus proposed a continuous formulation of the k -means algorithm in a multidimensional case at first. Independently on the Steinhaus' work, Stuart Lloyd an engineer at Bell Labs came up with a similar iterative algorithm as a technique for pulse-code modulation (PCM). Unlike Steinhaus, who formulated the problem in \mathbb{R}^3 , Lloyd introduced the method for \mathbb{R}^1 cases.

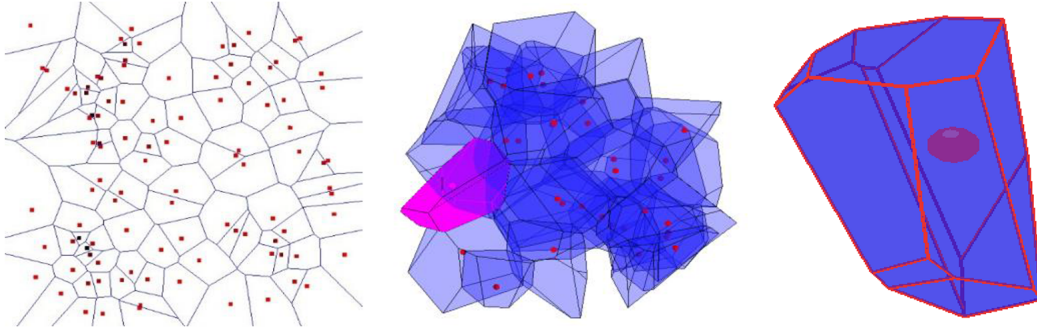


Figure 8.1: From left, visualisations of 2 and 3 dimensional Voronoi diagrams, where the red points within Voronoi cells represent centroids of clusters. Source: *Ying et al. Point Cluster Analysis Using a 3D Voronoi Diagram with Application in Point Cloud Segmentation* [85].

His idea firstly appeared in internal technical report in 1957 at Bell Labs. However, this methodology has not “come to light” until 1982. After almost 25 years, Lloyd published his original algorithm in paper *Least squares quantization in PCM* [84].

In the decades, Lloyd’s idea was many times modified into the contemporary k -means. Among fundamental works, we would mention MacQueen’s paper *Some methods for classification and analysis of multivariate observations* [86], where term k -means was firstly used. In this paper, MacQueen followed his previous work *The classification problem* [87] related to solutions of the Marschack’s economical problems [88, 89]. MacQueen’s solution was based on minimization of SSQ (Sum of Squared Distances) continuous criteria for multidimensional value distributions.

Further, Forgey proposed a modification of the k -means for clustering data in continuous space that uses a batch centroid model, where a centroid is considered as a geometric centre of a convex-shaped object [90]. This interpretation can take as a generalization of the mean. The similar principle was mentioned in the Lloyd’s work. It was defined for discrete data though. Therefore, it is why a standard k -means algorithm is sometimes referred to as Lloyd-Forgey. Through the literature, we can meet with various terms, names, and notations associated with the k -means. By the term k -means, we will call a general problem formulation in this text. Other terms are often linked to a specific approach for addressing this problem that include algorithms, their variations, and adaptations.

8.2 The k -means algorithm

The standard k -means algorithm [84] belongs to partitional clustering techniques. This basically means, the algorithm decomposes an initial set:

$$\mathbb{X} \stackrel{\text{def}}{=} \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\} \subset \mathbb{R}^n \quad (8.1)$$

into k clusters:

$$\mathbb{S} \stackrel{\text{def}}{=} \{\mathbb{S}_1, \mathbb{S}_2, \dots, \mathbb{S}_k\}, \text{ for all } \mathbb{S}_i \subset \mathbb{X}, i = 1, 2, \dots, k, \quad (8.2)$$

so that the clusters are convex and mutually disjoint:

$$\bigcap_{j=1}^k \mathbb{S}_j = \emptyset, \quad (8.3)$$

and satisfy:

$$\mathbb{X} = \bigcup_{j=1}^k \mathbb{S}_j. \quad (8.4)$$

A parameter k is appropriately chosen by a user or determined employing techniques for evaluation of clustering results, e.g. Calinski-Harabasz [91] or Davies–Bouldin index [92]. In addition, we consider that the sizes of the clusters $\mathbb{S}_1, \mathbb{S}_2, \dots, \mathbb{S}_k$ introduced in (8.2) follow:

$$\text{card}(\mathbb{S}_j) \geq 0 \text{ for all } j \in \mathbb{I}_{\mathbb{S}}, \quad (8.5)$$

where $\mathbb{I}_{\mathbb{S}} \stackrel{\text{def}}{=} \{1, 2, \dots, k\}$ is an index set associated with a set \mathbb{S} defined in (8.2), $\text{card}(\cdot)$ represents a set size (cardinality), and the condition (8.5) allows empty clusters.

Definition 6 (Clustering) Consider $\mathbb{I}_{\mathbb{X}} = \{1, 2, \dots, m\}$ as an index set related to an initial set \mathbb{X} defined in (8.1). We then define a clustering as a function ϕ , which maps indices of samples from $\mathbb{I}_{\mathbb{X}}$ to an index set $\mathbb{I}_{\mathbb{S}}$:

$$\phi : \mathbb{I}_{\mathbb{X}} \rightarrow \mathbb{I}_{\mathbb{S}}. \quad (8.6)$$

Using a mapping ϕ defined in (8.6), let us determine an optimal representative¹ $\boldsymbol{\mu}_j^* \in \mathbb{R}^n$ associated with samples belonging to a cluster \mathbb{S}_j such that:

$$\boldsymbol{\mu}_j^* = \arg \min_{\boldsymbol{\mu}} \sum_{i=1}^m \text{dist}(\mathbf{x}_i, \boldsymbol{\mu}) \chi_{\mathbb{S}_j}(\mathbf{x}_i), \quad (8.7)$$

where

$$\text{dist} : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R} \quad (8.8)$$

is a distance considered as a similarity measure and $\chi_{\mathbb{S}_j}(\cdot)$ represents an indicator function associated with a cluster \mathbb{S}_j . If a distance $\text{dist}(\cdot, \cdot)$ corresponds to a squared L_2 metric, then $\boldsymbol{\mu}_j^*$ becomes a mean of samples assigned to a cluster \mathbb{S}_j .

¹This representative is called a centroid in a case of the k -means algorithm.

In addition, the k -means algorithm minimizes a within-cluster variance. For now, let us consider a cluster \mathbb{S}_j , where $j \in \mathbb{I}_{\mathbb{S}}$. Then, we define a variance within a cluster \mathbb{S}_j as follows:

$$\text{Var}(\mathbb{S}_j) \stackrel{\text{def}}{=} \sum_{\mathbf{x} \in \mathbb{S}_j} \frac{\text{dist}(\mathbf{x}, \boldsymbol{\mu}_j^*)}{\text{card}(\mathbb{S}_j)}. \quad (8.9)$$

Formally, the standard k -means algorithm solves an underlying optimization problem so that it minimizes a residual sum of squares (RSS):

$$\arg \min_{\mathbb{S}} \sum_{j=1}^k \sum_{\mathbf{x} \in \mathbb{S}_j} \|\mathbf{x} - \boldsymbol{\mu}_j\|^2 = \arg \min_{\mathbb{S}} \sum_{j=1}^k \text{card}(\mathbb{S}_j) \text{Var}(\mathbb{S}_j). \quad (8.10)$$

A widely used stochastic solver for an optimization problem (8.10) is the Lloyd-Forgy algorithm [84]. It is based on an iterative refinement technique such that it proceeds by alternating between two steps, specifically *assignment* and *update*.

In its initial stage, the algorithm begins by a random selection of starting centroids. Following this, it proceeds to assign each sample to a cluster, where a squared distance to its associated centroid is minimal, and the memberships are programmatically stored in a vector \mathbf{u} :

$$\mathbf{u} = [u_1, u_2, \dots, u_m], \quad (8.11)$$

such that its components u_i , where $i = 1, 2, \dots, m$, are step-wisely associated with the input sample $\mathbf{x} \in \mathbb{X}$.

Afterwards, it updates the centroids as *a mean* of samples within the actual clusters. This is summarized in Algorithm 3.

Algorithm 3: RP (RANDOMPARTITION)

Input : $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$, $k \in \mathbb{N}$
1 $\{\mathbb{S}_1, \mathbb{S}_2, \dots, \mathbb{S}_k\}, \mathbf{u} \leftarrow \text{RANDOMASSIGNMENT}(\mathbb{X}, k);$
2 **for** $i = 1 \rightarrow k$ **do**
3 $\left| \begin{array}{l} \boldsymbol{\mu}_i \leftarrow \frac{1}{\text{card}(\mathbb{S}_i)} \sum_{\mathbf{x} \in \mathbb{S}_i} \mathbf{x}; \end{array} \right.$
Output: $\{(\mathbb{S}_1, \boldsymbol{\mu}_1), (\mathbb{S}_2, \boldsymbol{\mu}_2), \dots, (\mathbb{S}_k, \boldsymbol{\mu}_k)\}, \mathbf{u}$

Another commonly used approach for determining an initial guess is called the Forgy method. [90]. An essential idea beyond this method is to randomly choose k samples from a dataset \mathbb{X} and use them as an initial guess. You can find a pseudo code of this approach in Algorithm 4. Note that *the Multiply-With-Carry algorithm* [93] or *Mersenne Twister* [94] can be used as pseudo-random number generators in the both the random partition approach and the Forgy method.

Algorithm 4: FORGY

Input : $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}, k \in \mathbb{N}$
1 $\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_k\} \leftarrow \text{SELECTRANDOMSEEDS}(\mathbb{X}, k);$
2 **forall** $S_j^{(0)}$ **do**
3 $\mathbb{S}_j^{(0)} \leftarrow \emptyset;$
4 **for** $i = 1 \rightarrow m$ **do**
5 $j^* = \arg \min_{j=1,2,\dots,k} \|\boldsymbol{\mu}_j - \mathbf{x}_i\|;$
6 $\mathbb{S}_{j^*} \leftarrow \mathbb{S}_{j^*} \cup \{\mathbf{x}\};$
7 $[u]_i \leftarrow j^*;$
Output: $\{(\mathbb{S}_1, \boldsymbol{\mu}_1), (\mathbb{S}_2, \boldsymbol{\mu}_2), \dots, (\mathbb{S}_k, \boldsymbol{\mu}_k)\}, \mathbf{u}$

The k -means algorithm converges when the largest centroid displacement becomes smaller than a reasonably small square of $\varepsilon \in \mathbb{R}^+$:

$$\Delta^{(t)} \stackrel{\text{def}}{=} \max \left\{ \|\Delta \boldsymbol{\mu}_1^{(t)}\|^2, \|\Delta \boldsymbol{\mu}_2^{(t)}\|^2, \dots, \|\Delta \boldsymbol{\mu}_k^{(t)}\|^2 \right\} \leq \varepsilon^2, \quad (8.12)$$

where:

$$\Delta \boldsymbol{\mu}_i^{(t)} = \boldsymbol{\mu}_i^{(t)} - \boldsymbol{\mu}_i^{(t-1)} \quad (8.13)$$

is based on the normalized residual sum of squares (NRSS), and t represents a number of an actual iteration.

A second option for a stopping criterion can involve monitoring reassignments of samples. Let Δ be a change in a sample memberships after t -th iteration. This change can be expressed by means of a ratio of reassigned samples to the total number of samples in an initial set \mathbb{X} :

$$\Delta^{(t)} = \frac{\text{card}(\mathbb{I}_{\widehat{\mathbb{X}}^{(t)}})}{\text{card}(\mathbb{X})}, \quad (8.14)$$

where:

$$\mathbb{I}_{\widehat{\mathbb{X}}^{(t)}} \stackrel{\text{def}}{=} \left\{ j \in \mathbb{X} : u_j^{(t)} \neq u_j^{(t-1)} \right\}. \quad (8.15)$$

A sample assignment is practically implemented such that the memberships are stored in a vector \mathbf{u} . Now, we have all ingredients needed to introduce the Lloyd-Forgy algorithm; its pseudocode is summarised in Algorithm 5 on the next page.

Unfortunately, the Lloyd-Forgy algorithm does not guarantee convergence to a global optimum. Worse, it may fail to converge to a local minimum such that the algorithm yields a partial “optimal” solution in a stationary point of an objective function (8.10), we refer the following paper [95] for further details.

A commonly used strategy for overcoming the issue of achieving an optimal solution is based on the following approach: *the algorithm is executed several times using different initial*

guesses, which commonly leads to increasing the chance of attaining a local optimum. Among the achieved solutions, the best one is chosen such that the related RSS is minimal. Another way is to use more sophisticated initialization approaches.

Algorithm 5: STANDARDKMEANS (LLOYD-FORGY)

Input : $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$, $k \geq 2$, small $\varepsilon \in \mathbb{R}^+$, $m \in \mathbb{N}$

```

1  /* initialization */
2  INITIALIZE  $t \leftarrow 0$ ;
3  INITIALIZE  $\Delta \leftarrow \infty$ ;
4  if Forgy Strategy then
5      |  $\{(\mathbb{S}_1^{(0)}, \boldsymbol{\mu}_1^{(0)}), (\mathbb{S}_2^{(0)}, \boldsymbol{\mu}_2^{(0)}), \dots, (\mathbb{S}_k^{(0)}, \boldsymbol{\mu}_k^{(0)})\}, \mathbf{u}^{(0)} \leftarrow \text{FORGY}(\mathbb{X}, k)$ ;
6  else
7      |  $\{(\mathbb{S}_1^{(0)}, \boldsymbol{\mu}_1^{(0)}), (\mathbb{S}_2^{(0)}, \boldsymbol{\mu}_2^{(0)}), \dots, (\mathbb{S}_k^{(0)}, \boldsymbol{\mu}_k^{(0)})\}, \mathbf{u}^{(0)} \leftarrow \text{RP}(\mathbb{X}, k)$ ;
8   $\mathbb{M}^{(0)} \leftarrow \{\boldsymbol{\mu}_1^{(0)}, \boldsymbol{\mu}_2^{(0)}, \dots, \boldsymbol{\mu}_k^{(0)}\}$ ;
9  /* iteration */
10 while  $\Delta^{(t)} > \varepsilon \wedge t < m$  do
11     |  $t \leftarrow t + 1$ ;
12     |  $\mathbb{M}^{(t)} \leftarrow \emptyset$ ;
13     | /* update step */
14     | for  $i = 1 \rightarrow k$  do
15         |  $\boldsymbol{\mu}_i^{(t)} \leftarrow \frac{1}{\text{card}(\mathbb{S}_i^{(t-1)})} \sum_{\mathbf{x} \in \mathbb{S}_i^{(t-1)}} \mathbf{x}$ ;
16         |  $\mathbb{M}^{(t)} \leftarrow \mathbb{M}^{(t)} \cup \{\boldsymbol{\mu}_i^{(t)}\}$ ;
17         |  $\mathbb{S}_i^{(t)} \leftarrow \emptyset$ ;
18     | /* assignment step */
19     | for  $i = 1 \rightarrow m$  do
20         |  $j^* = \arg \min_{j=1,2,\dots,k} \|\boldsymbol{\mu}_j - \mathbf{x}_i\|$ ;
21         |  $\mathbb{S}_{j^*}^{(t)} \leftarrow \mathbb{S}_{j^*}^{(t)} \cup \{\mathbf{x}_i\}$ ;
22         |  $[u]_i \leftarrow j^*$ ;
23     | compute  $\Delta^{(t)}$  using (8.12) or (8.14);
Output:  $\mathbb{M}, \mathbf{u}, t, \text{RSS}$ 
    
```

8.3 The k-means++ algorithm

The Lloyd-Forgy approach does not guarantee an accuracy in the sense of finding an optimal solution as we mentioned in the previous Section 8.2. In many practical examples, the algorithm generates arbitrary bad clustering results. It arises from the fact that a ratio:

$$\frac{\phi(\mathbb{X})}{\phi_{\text{OPT}}(\mathbb{X})} \quad (8.16)$$

is unbounded even m (number of samples) and k (number of clusters) are fixed; ϕ_{OPT} represents an optimal solution (clustering). It is mainly a consequence of a uniformly random choice of initial guess.

Despite these facts, an implementation details of the Lloyd-Forgy algorithm, and its speed in the sense of convergence rate, are getting the algorithm very appealing in the various fields ranging from biology to computer graphics. The worst-case scenario of its running time is superpolynomial with factor $2^{\Omega(\sqrt{m})}$ [96]. However, it is considered to be of a linear complexity $\mathcal{O}(m)$ in practice, when a number of iterations and parameters are fixed. This makes the k -means algorithm very promising. Moreover, Duda et al. observed and published in [97] the following fact: “*The number of iterations is generally much less than the number of samples*”.

Algorithm 6: KMEANS++

Input : $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$, $k \geq 2$, small $\varepsilon \in \mathbb{R}^+$, $m \in \mathbb{N}$

```

1 /* initialization */
2 INITIALIZE  $t \leftarrow 0$ ;
3 INITIALIZE  $\Delta \leftarrow \infty$ ;
4  $\mu_1^{(0)} \leftarrow \text{SELECTRANDOMSEED}(\mathbb{X})$ ;
5 for  $j = 2 \rightarrow k$  do
6    $\mathbf{p} \leftarrow \mathbf{o}$ ;
7    $D \leftarrow 0$ ;
8   foreach  $i = 1 \rightarrow m$  do
9      $d \leftarrow \min(\|\mu_1^{(0)} - \mathbf{x}_i\|, \|\mu_2^{(0)} - \mathbf{x}_i\|, \dots, \|\mu_j^{(0)} - \mathbf{x}_i\|)$ ;
10     $\mathbf{p}(\mathbf{x}) \leftarrow d^2$ ;
11     $D \leftarrow D + [p]_i$ ;
12   foreach  $i = 1 \rightarrow m$  do
13      $P[\mathbb{X} = \mathbf{x}_i] \leftarrow \frac{[p]_i}{D}$ ;
14    $\mu_j^{(0)} \leftarrow \text{SECTDISTRIBUTEDRANDOMSEED}(\mathbb{X}, P)$ ;
15 /* assignment */
16 foreach  $\mathbf{x} \in \mathbb{X}$  do
17    $j^* = \arg \min_{j=1,2,\dots,k} \|\mu_j^{(0)} - \mathbf{x}\|$ ;
18    $\mathbb{S}_{j^*}^{(0)} \leftarrow \mathbb{S}_{j^*}^{(0)} \cup \{\mathbf{x}\}$ ;
19  $\mathbb{M}^{(0)} \leftarrow \{\mu_1^{(0)}, \mu_2^{(0)}, \dots, \mu_k^{(0)}\}$ ;
20 continue STANDARDKMEANS on line 10

Output:  $\mathbb{M}, \mathbf{u}, t, \text{RSS}$ 

```

A way how the issue associated with a partial solution could be overcome was proposed by David Arthur and Sergei Vassilvitskii in 2007. They developed a technique called k -means++ [98] aimed at improving an accuracy of an initial guess estimation. This method

adjusts the random initialization in such a way that the probability of selecting a particular sample as an initial centroid is directly proportional to its squared distance from the nearest already chosen centroid. Pseudocode of this method is outlined in Algorithm 6. By means of this modification, a solution returned by a minimizer is, at the most, worse by factor $8(\ln(k) + 2)$ than an optimal solution ϕ_{OPT} .

In this chapter, we summarized two commonly used variants related to algorithms of a k -means type, particularly the standard k -means employing the Lloyd-Forgy algorithm as an underlying solver and k -means++. However, there are other adaptations of these methods, which we briefly mention below.

The online version of the Lloyd-Forgy algorithm, where the means are immediately updated after point reassignments, is called the MacQueen algorithm [87]. The algorithm known as Partitioning Around Medoids (PAM) [99] uses a greedy search to determine an optimal representative of clusters from actual samples in an initial data set \mathbb{X} , and k -medians employs the Weiszfeld algorithm [100] for determining medians as the representatives of clusters. By collaborating with Pavel Skalný, we published a conference paper [101], where we proposed the “plus-plus” adaptations for the PAM and k -medians algorithms – we used a similar initialization strategy as in the case of k -means++ algorithm. The algorithm PAM and k -medians slightly differ in updating step, which arises from the fact that they used a different representative of clusters. For simplifying of this text, we mention this step in next section, where we also describe a parallel implementation of these methods, alongside k -means and k -means++.

8.4 Parallel implementation

The aim of this section is to describe a parallel implementation of the algorithm introduced in Sections 8.2 and 8.3. Specifically, we focus on parallelization of k -means, k -medians, PAM, and their “plus-plus” variants [101]. Recall that the last two algorithms differ in updating step, where another type of representative is computed. In collaboration with Pavel Skalný, we successfully implemented these approaches into software written in C++ programming language and designed for running in massively parallel distributed environment containing hundreds computational cores. For inter-process communication, we employ Message Passing Interface (MPI), which is currently a dominant protocol in high-performance computing (HPC) environments used for data exchange.

Using MPI-IO, input data $\mathbb{X} \subset \mathbb{R}^n$ are effectively distributed among multiple processes such that a load-balancer ensures that each process owns a roughly equal proportion of \mathbb{X} . By this approach, we prevent overloading some computational cores while other ones remain idle. This is one of the essential steps for optimizing an overall running time in a massively parallel distributed environment, e.g. the top-world supercomputers.

In the upcoming sections, we describe the concepts of parallelization strategies for the clustering algorithms mentioned above. While the fundamental principles are mostly similar, they differ in specific subprocedures. For example, the inter-process communication required for determining medoids is typically higher than for determining centroids of clusters; this is the same for “plus-plus” and standard initialization approaches.

8.4.1 Initialization step

Unless otherwise stated, let us refer centroids, medoids, and medians as cluster representatives, or simply representatives, for the reading convenience of this text. At first, we describe a parallel strategy for determining an initial guess using a standard (*random partition*) approach earlier introduced in Algorithm 3.

Here, a master (zero) process randomly selects k (related to a number of clusters) indices from an index set \mathbb{I}_X . These indices are associated with samples that were determined as initial representatives.² Then, a master process stores the selected samples belonging to its proportion of an initial set X to an array of cluster representatives, and sends indices of the remaining ones to the slave processes using MPI broadcast. Afterwards, it gathers the remaining selected samples from the slave processes. We use the implementations of *send* and *receive* procedures from the Boost framework [102] (C++ API):

- `world.recv(mpi::any_source, 0, ...)` in a root process,
- `world.send(0, 0, ...)` in slave processes,

where `world` is related to an MPI communicator object (`boost::mpi::communicator`). In the final step of the standard initialization of representatives, a master process broadcasts all initial representatives to other processes using MPI broadcast. Then, the samples are locally assigned to an appropriate cluster based on their minimal distances to representatives of those clusters, as we mentioned in Section 8.2.

The “plus-plus” initialization (Algorithm 6) related to the algorithms of “ k -means type” requires slightly more inter-process communication than the previously introduced parallel implementation of the standard version, i.e. *the random partition*. At first, a master process randomly selects one index from \mathbb{I}_X . Then, it gathers and broadcasts a sample associated with this index similarly to what was mentioned for the standard approach: *if a master process owns this sample, then the process **broadcasts** this sample to slave processes. Otherwise, it gathers this sample from the slave processes by means of **send/recv** procedures, and **broadcasts** this sample to all processes then.* Afterwards, probability weights $D(x)^2$ are computed locally, and step-wisely merged using the **send/recv** mechanism in a master process for a selection of a

²Note, we employ Mersenne Twister (MT97) [94] as a pseudo-random number generator in the software implementation.

next index associated with a next starting representative. The initialization process repeats until k representatives are not determined. Then, the samples are assigned to an appropriate cluster locally.

8.4.2 Update step

Once the initial guess including *representatives* and *memberships* (stored in \mathbf{u}) is determined, an algorithm recomputes cluster representatives such as *means*, *medians*, and *medoids* depending on a specific clustering method.

In the case of ***k*-means** (or ***k*-means++**), the cluster representative is a centroid, which is basically an arithmetic mean of samples belonging to a particular cluster. Parallelizing the update step for this method is quite straightforward. Each process maintains local sizes of clusters and sums of samples within these particular clusters, which are associated with a local proportion of an input data set \mathbb{X} . Then, the sizes and the sums are collectively reduced across all processes using `mpi::all_reduce()` – implemented in the Boost framework. Afterwards, the centroids are computed locally.

The *k*-medians algorithm employs the Weiszfeld algorithm [100] for determining a cluster representative as a geometric median. Let $\mu_j^{(t)}$ be a geometric median determined in an iteration t and associated with a cluster \mathbb{S}_j . Subsequently, the algorithm updates a median in an iteration $t + 1$ using the following formula:

$$\mu_j^{(t+1)} \stackrel{\text{def}}{=} \left(\sum_{\mathbf{x} \in \mathbb{S}_j} \frac{\mathbf{x}}{\|\mathbf{x} - \mu_j^{(t)}\|} \right) / \left(\sum_{\mathbf{x} \in \mathbb{S}_j} \frac{1}{\|\mathbf{x} - \mu_j^{(t)}\|} \right). \quad (8.17)$$

A practical parallel implementation of the update step in order to the *k*-median algorithm involving a parallelization of computing the equation (8.17). It follows a similar approach, which we implemented for the *k*-means algorithm: *each nominator and denominator are computed on a local proportion of an input data set and, then, they are reduced across all processes by mpi::all_reduce()*. Afterwards, medians are determined locally.

Finally, we will discuss the implementation details of parallelizing an update step associated with **the PAM algorithm**. Recall this algorithm finds an optimal representative among samples in an initial set \mathbb{X} . This basically means, the representatives are not computed as means, medians, or using any other approach based on samples within the actual clusters; they are just selected from an initial set \mathbb{X} .

Before updating sample assignments \mathbf{u} , the algorithm performs a greedy search, which is known as SWAP, to find the best representative within these clusters. Let us denote a set of non-medoid samples as \mathbb{O}_j associated with a cluster \mathbb{S}_j . We summarize an approach of finding the best representative of a cluster using SWAP in Algorithm 7.

Algorithm 7: SWAP

- 1 randomly select $\mathbf{o} \in \mathbb{O}_j$
 - 2 swap \mathbf{m}_j and \mathbf{o}
 - 3 recompute a cluster cost as a summation of distances between vectors within a same cluster and their respective medoid,
 - 4 if the a cluster cost increased, remove \mathbf{o} from \mathbb{O} , use original \mathbf{m}_j and continue with the step 1, otherwise finish.
-

A parallelization of SWAP can proceed as follows: *a master process selects an index of a non-medoid sample for each cluster. If a master process owns these samples, it broadcasts them to the slave processes. Otherwise, a master process gathers these samples from the slave processes by `send/recv` mechanism and, then, it distributes them to other processes by means of MPI broadcast. Afterwards, cluster costs are computed on a local proportion of \mathbb{X} , and then, they are reduced across all processes by `mpi::all_reduce()`. Step 4 is performed locally.*

8.5 Benchmarks

This section deals with introducing an application dealing with brittle and ductile fracture detection employing vector quantification techniques introduced in a previous Chapter 8, namely k -means and its variant k -means++, k -medians, and PAM. The results were attained during collaboration with Pavel Skalný (VSB-TU Ostrava), and were published in two papers [101, 103]. Note that evaluating the fracture resistance of a material (commonly steel) is crucial in civil engineering applications, e.g. building oil and gas pipelines in Siberia regions.

Recently, the Drop Weight Tear Test (DWTT) is typically used for testing this type of resistance. After realizing the test, an expert evaluates the material quality as a ratio between a ductile and brittle regions of a damaged area, depicted in Figure 8.2. Although expert's analysis has many advantages, human mistakes are also incorporated and expert's opinions may vary significantly. Therefore, other mostly automated techniques are developed for industrial applications.

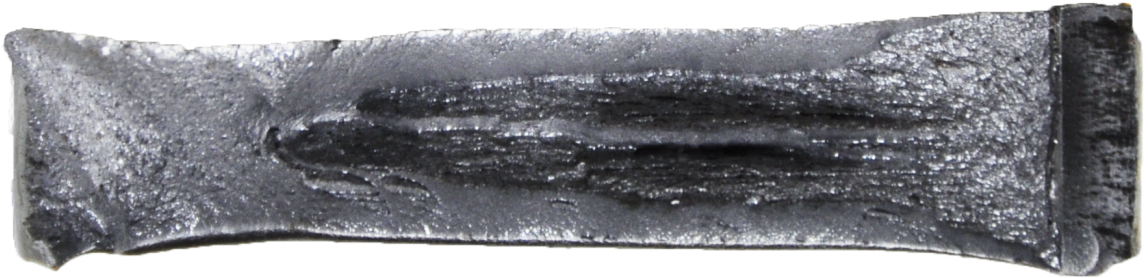


Figure 8.2: Photo depicting a fracture area of API 5L X-70 sheet steel (18.7 mm) after performing DWTT test. Published in [101].

3D scanning to capture a fracture area as a point cloud combined with computer analysis based on ML approaches represent common alternatives to expert evaluation for assessing material quality. To characterize a fracture area, fractal geometry can be used to determine feature descriptors. However, this approach has some disadvantages, e.g. characteristics must be calculated on a sufficiently large number of points, which are not always available.

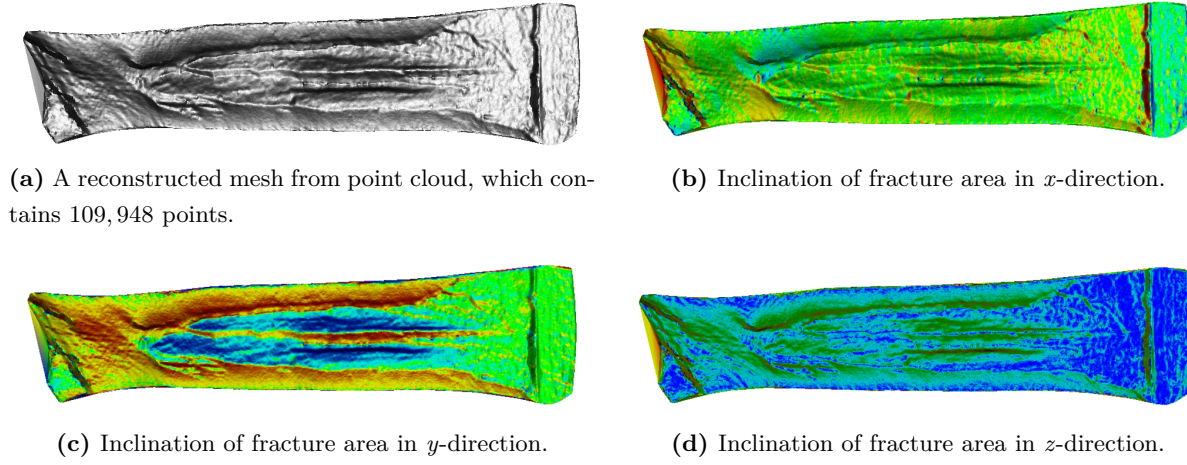


Figure 8.3: Reconstructed mesh and its features represented by normal vector in x , y , and z direction.

Thus, we decided to work on the concept where we locally evaluate the quality of the specimens using vector quantification techniques. For reconstructing an original fracture area from the point cloud, we used approach based on Delaunay triangulation [104]. As descriptors of fracture area, we choose normal vectors in each triangle of a reconstructed mesh, because they highly correspond to the brittle and ductile regions, see Figure 8.3.

Since clustering belongs to unsupervised learning, achieved results are evaluated using different metrics than in a previous application, where we employ supervised learning approaches. Typically used metrics for these purposes are the Calinski-Harabasz (CHI) [91] and Davies-Bouldin (DBI) [92] indexes. Further, the *CHI* index is defined as follows:

$$CHI \stackrel{\text{def}}{=} \frac{B(k)(n-k)}{W(k)(k-1)}, \quad (8.18)$$

where k is number of clusters and n is number of input vectors, $B(k)$ represents between-cluster sum of squares:

$$B(k) \stackrel{\text{def}}{=} \sum_{i=1}^k \text{card}(\mathbb{S}_i) \|\boldsymbol{\mu}_i - \boldsymbol{\mu}\|^2, \quad (8.19)$$

where $\boldsymbol{\mu}$ is an overall centroid of data, and $W(k)$ is within-cluster sums of squares:

$$W(k) \stackrel{\text{def}}{=} \sum_{i=1}^k \sum_{\mathbf{x} \in \mathbb{S}_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2. \quad (8.20)$$

The criterion based on *CHI* index is analogous to an F-ratio in ANOVA and the highest value determines the best clustering result [101]. *DBI* criterion is also based on a ratio of within-cluster and between-cluster distances and is defined as follows:

$$DBI = \frac{1}{k} \sum_{i=1}^k \max_{j=1,2,\dots,k, j \neq i} D_{ij}, \quad (8.21)$$

where:

$$D_{ij} = \frac{\bar{d}_i + \bar{d}_j}{d_{ij}}, \quad (8.22)$$

and \bar{d}_i is average distances between each point in a cluster \mathbb{S}_i and a centroid $\boldsymbol{\mu}_i$ associated with this cluster \mathbb{S}_i , which is defined such that:

$$\bar{d}_i = \frac{1}{\text{card}(\mathbb{S}_i)} \sum_{\mathbf{x} \in \mathbb{S}_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|, \quad (8.23)$$

it is similar for \bar{d}_j ; d_{ij} represents the distance between the centroids belonging cluster \mathbb{S}_i and \mathbb{S}_j , formally:

$$d_{ij} \stackrel{\text{def}}{=} \|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|. \quad (8.24)$$

Smaller value of *DBI* is acceptable as a better result value [101].

In our experiments, we compared *k*-means, PAM, and *k*-medians algorithms for detecting brittle and ductile areas of DWTT specimen. Since we study two regions, we set $k = 2$. We have implemented software in C++, which is designed for running computation in a parallel distributed environment containing hundreds of computational cores. Implementation details about this software were introduced in Section 8.4.

In our experiments, we study quality of commercially produced API 5L steel (thickness of a sheet is 18.7 mm). [101]. Note that the surface of mesh was constructed from point cloud produced using Limes Measurement Technique.³ This point cloud contains 109,948 points.

We ran the experiments on the Salomon supercomputer (retired now). Salomon consists of 1008 compute nodes. Each compute node contains two 2.5 GHz, 12-core Intel Xeon E5-2680v3 (Haswell) processors and 128 GB of memory. Compute nodes are interconnected by InfiniBand FDR56. Salomon has the peak performance around 2 petaFLOPS, you can find further information on the following website ⁴.

³<https://www.limes.com/en/>

⁴<https://www.it4i.cz/en/infrastructure/salomon>

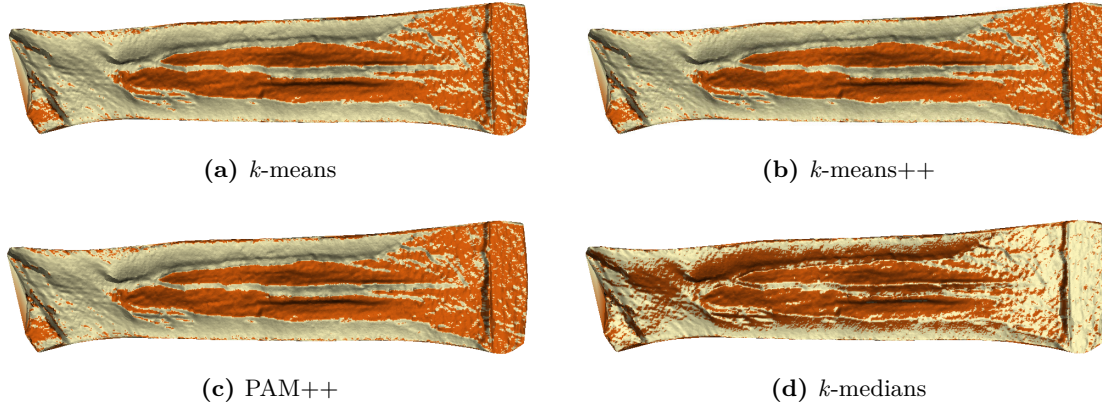


Figure 8.4: Visualization of 3 best results (*k*-means, *k*-means++, PAM++) and the worst one (*k*-medians). *These visualizations were published in [101].*

The results are summarized in Table 8.1. Analyzing them, we can conclude that the best ones are attained using *k*-means, *k*-means++, and PAM++ algorithms. Comparing with Figure 8.3, we can see that they also highly correspond to brittle and ductile fracture areas. Visualizations of representative results are depicted in Figure 8.4, where orange areas correspond to brittle fracture, and the ochre regions are associated with ductile fracture. Note that the worst result were attained by *k*-medians algorithm.

	CHI (higher better)	DBI (lower better)
k-means	122,630	0.8612
k-means++	122,630	0.8612
PAM	63,973	0.7372
PAM++	122,240	0.8607
k-medians	12,070	0.8619
k-medians++	109,070	0.8343

Table 8.1: Evaluation of attained results using Calinski-Harabasz (CHI) and Davies-Bouldin (DBI) criterions. *These results were published in [101].*

Now, let us briefly discuss the parallel scalability of these algorithms. The algorithms *k*-means, *k*-means++, *k*-medians, and *k*-medians++ converge in the order of tens of seconds on 2 MPI processes. When the number of MPI processes is higher than 2, then computational time increases because the amount of inter-process communication gets higher and computation runs slowly.

In the case of the PAM-type algorithms, they compute the total cluster cost in each update step. Thus, parallelization makes sense. Scalability graphs for PAM and PAM++ algorithms are depicted in Figure 8.5. These graphs show that the PAM algorithm scales fine up to 8 MPI processes, gets stuck between 8 and 16 processes, and scales well up to 48 processes,

which corresponds to 2 nodes. The PAM++ algorithm scales almost linearly up to 16 process, then the ratio between the amount of inter-process communication and computational times is higher, and the efficiency of this parallelization approach gets lower. However, it still scales up 120 MPI processes (5 nodes).

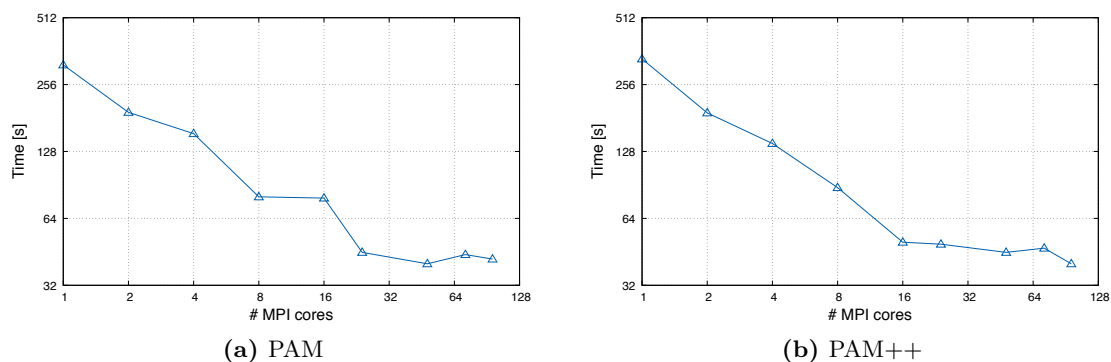


Figure 8.5: A strong-scalability tests of PAM and PAM++ algorithms, which were published in [101].

Chapter 9

Spectral clustering

This chapter focuses on another unsupervised learning technique used for clustering, which is based on spectral properties of a graph Laplacian matrix. This matrix will be properly defined later in this chapter in Section 9.1 on page 119. For the purpose of introducing this method, let us consider a graph Laplacian matrix as a matrix having similar properties as a discrete Laplace operator. Let us denote this matrix as $\mathbf{L} \in \mathbb{R}^{m \times m}$, where m represents number of samples.

A bottom line of spectral clustering methods is geometrically based on assuming that an initial data set:

$$\mathbb{X} \stackrel{\text{def}}{=} \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\} \subset \mathbb{R}^n \quad (9.1)$$

is sampled from a Riemannian manifold $\Omega \subset \mathbb{R}^n$ containing multiple connected components forming non-overlapping parts of Ω . In a traditional approach related to spectral clustering [105], samples belonging to a data set \mathbb{X} are projected onto a null space $\text{Null}(\mathbf{L})$, which is associated with a Laplace operator. It can be viewed as a discrete Laplace operator on a graph G that represents a part of Ω .

A dimension of this null space equals geometric multiplicity of a zero eigenvalue λ_0 corresponding to an operator \mathbf{L} . Then, such transformed samples are clustered by applying, for example, vector quantification techniques such as k -means and k -means++ earlier introduced in Section 8.2 respective in Section 8.3.

Definition 7 (Geometric multiplicity of eigenvalue) *Let $\mathbf{L} \in \mathbb{R}^{m \times m}$ and λ be an eigenvalue of \mathbf{L} . Let*

$$E_\lambda \stackrel{\text{def}}{=} \text{span}\{\mathbf{e} : (\mathbf{L} - \lambda \mathbf{I}) \mathbf{e} = \mathbf{0}\} \quad (9.2)$$

be an eigenspace corresponding to λ . A dimension of E_λ is called geometric multiplicity of an eigenvalue λ :

$$\dim E_\lambda = m_{\mathbf{L}}(\lambda). \quad (9.3)$$

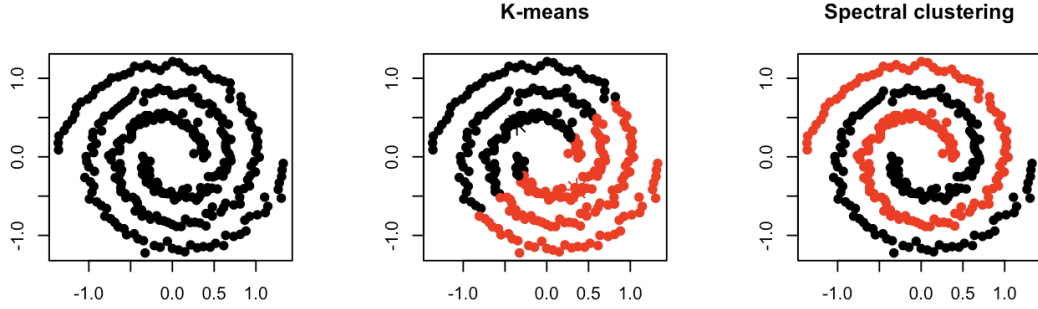


Figure 9.1: Visual comparison results attained by the k -means and spectral clustering methods on the two spiral problem. *The original image was downloaded from [107].*

Recall. Concerning a structure of an operator \mathbf{L} , it is obtained by a finite difference method [106] on a graph G capturing a geometry of a discretized manifold Ω .

While algorithms of a k -means type minimize a within-cluster variance, an aim of spectral clustering is maximizing an internal cluster connectivity. Note that a cluster represents a component or merged various connected components associated with a graph G . Since vector quantification (employing k -means) is incorporated as a part of a spectral clustering, we can interpret a spectral clustering as a technique that could be used for outperforming k -means for cases of non-convex shaped clusters either, see Figure 9.1. Further, spectral clustering is commonly used as a technique for dimensionality reduction such that it embeds samples from an initial data set \mathbb{X} into a lower-dimensional feature space \mathbb{R}^p , where $m_{\mathbf{L}}(\lambda_0) \leq p < m$.

Formally, spectral clustering relaxes an NP-hard optimization problem¹ related to solve a graph-cut problem so that it balances a trade-off between internal and external cluster connectivity. Unless otherwise stated, we assume a graph is undirected and weighted such that a weights related to its edges measures similarity between samples. The radial basis function (RBF) is commonly exploited for determining such a level of similarity. In following Sections 9.1 and 9.2, we generally introduce two approaches related to a spectral clustering, specifically the unnormalized [105] and normalized [108] approaches, which we demonstrate on image segmentation problems in Section 9.3, including statistical estimating a number of zero eigenvalues in Section 9.3.1.

9.1 Unnormalized spectral clustering

Let $\mathbb{V} = \{v_1, v_2, \dots, v_m\}$ be a set of vertices representing samples $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\} \subset \mathbb{X}$, and let \mathbb{E} be a set of edges:

$$\mathbb{E} \subset \{\{v_k, v_l\} : v_k, v_l \in \mathbb{V}, k, l \in \{1, 2, \dots, m\}, k \neq l\}. \quad (9.4)$$

¹We assume a conjecture that $P \neq NP$, therefore a reasonable relaxation is necessary.

Then, we define an undirected graph G as a pair of edges \mathbb{E} and vertices \mathbb{V} such that:

$$G \stackrel{\text{def}}{=} (\mathbb{V}, \mathbb{E}). \quad (9.5)$$

Further, let $\mathbf{W} \in \mathbb{R}^{m \times m}$ be a weighed adjacency matrix corresponding to G , and let us consider that this matrix contains non-negative values (weights), see Definition 8.

Definition 8 (Weighed adjacency matrix) *A weighted adjacency matrix associated with a graph G is a matrix $\mathbf{W} \in \mathbb{R}_0^{m \times m}$, where $m = \text{card}(\mathbb{V})$, and entry $w_{ij} \geq 0$ represents a non-negative weight related to an edge connecting vertices v_i and v_j .*

Then, we define a cut of a graph G as a set of mutually disjoint non-empty subsets of \mathbb{V} :

$$\{\mathbb{A}_1, \mathbb{A}_2, \dots, \mathbb{A}_k\} \subset \mathbb{V}, \quad \bigcup_{i=1}^k \mathbb{A}_i = \mathbb{V} \wedge \bigcap_{i=1}^k \mathbb{A}_i = \emptyset \quad (9.6)$$

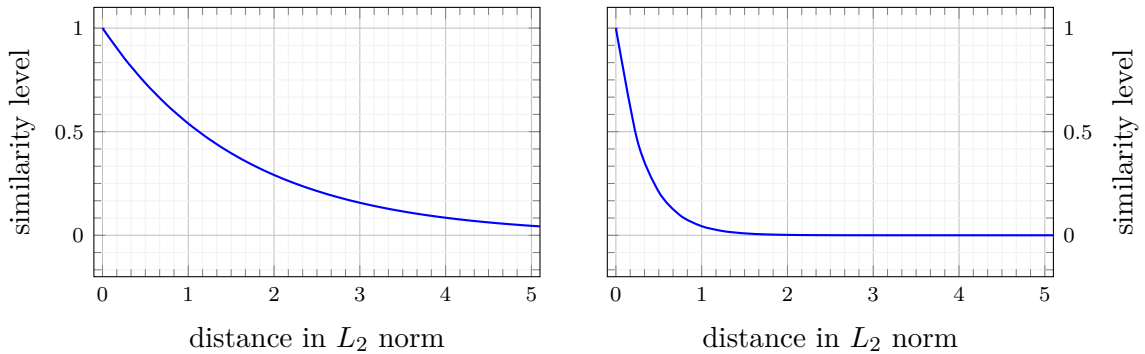
such that:

$$\text{cut}(\mathbb{A}_1, \mathbb{A}_2, \dots, \mathbb{A}_k) = \frac{1}{2} \sum_{i=1}^k W(\mathbb{A}_i, \mathbb{A}_i^c), \quad (9.7)$$

where:

$$W(\mathbb{A}_i, \mathbb{A}_i^c) = \sum_{k \in \mathbb{A}_i, l \in \mathbb{A}_i^c} w_{k,l}. \quad (9.8)$$

The set \mathbb{A}_i^c in (9.7) and (9.8) represents complement of the set \mathbb{A}_i in \mathbb{V} for $\forall i \in \{1, 2, \dots, k\}$, and w_{kl} is an weight of an edge connecting the vertices v_k and v_l .



(a) A plot depicts a radial basis function ($\sigma = 0.9$). (b) A plot depicts a radial basis function ($\sigma = 0.4$).

Figure 9.2: This example illustrates the influence of a parameter σ (standard deviation) on a shape of a radial basis function, which is defined in (9.9) on page 118.

In practical applications, the weight w_{kl} is commonly interpreted as a level of similarity between vertices v_k and v_l representing samples \mathbf{x}_k and $\mathbf{x}_l \in \mathbb{X}$, respectively. It is typically

modelled using a radial basis function (RBF) as follows:

$$w_{kl} = \exp\left(-\frac{\|\mathbf{x}_k - \mathbf{x}_l\|}{2\sigma^2}\right), \quad (9.9)$$

where a parameter σ denotes a standard deviation. We can consider σ as a scaling parameter that controls how rapidly similarity (represented by means of a weight w_{kl}) falls off with the Euclidean distance between samples \mathbf{x}_k and \mathbf{x}_l . In the definition of a graph-cut (9.7), a factor value $\frac{1}{2}$ prevents counting each edge twice in a cut.

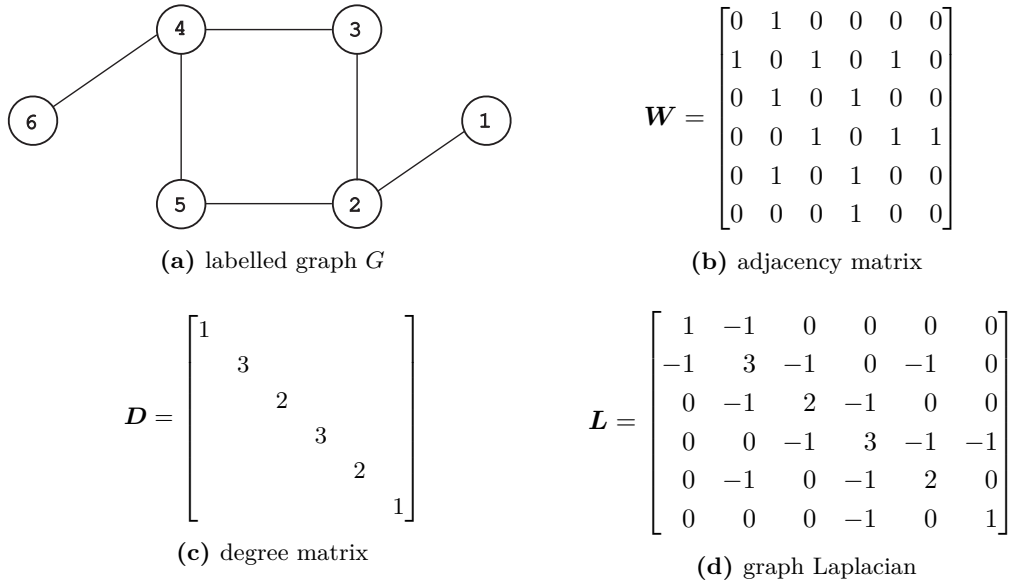


Figure 9.3: This showcase provides a simplified example of an undirected graph consisting of 6 vertices. A minimum degree of this graph is 1, and a maximal one equals 3. For this undirected graph, an adjacency matrix, a degree and a graph Laplacian matrices are outlined either.

Since we want to maximize internal cluster connectivity and, consequently, to minimize connections between the clusters $\mathbb{A}_1, \mathbb{A}_2, \dots, \mathbb{A}_k$, it leads to solving a minimal k -cut (min-cut) problem:

$$(\mathbb{A}_1^*, \mathbb{A}_2^*, \dots, \mathbb{A}_k^*) = \arg \min_{\mathbb{A}_1, \mathbb{A}_2, \dots, \mathbb{A}_k} \text{cut}(\mathbb{A}_1, \mathbb{A}_2, \dots, \mathbb{A}_k). \quad (9.10)$$

When a number of clusters $k = 2$, the min-cut problem (9.10) can be efficiently solved using the Stoer–Wagner algorithm [109] recursively. Nevertheless, the problem becomes NP-hard for any fixed $k \geq 3$. There exist various algorithms running in a polynomial-time $\sim \mathcal{O}(m^{2k})$ that provide a relaxed solution of the problem (9.10), see [110, 111, 112] for further details.

However, we do not typically attain a reasonable solution using the standard min-cut approach [113]. Since similarity between two vertices is inversely proportional to the distance

between them and a minimum cut (9.7) increases with the number of edges between the partitions, it typically tends to a trivial cut separating an individual vertex from the rest of a graph G . The way to exclude these singleton solutions is to explicitly ensure that the clusters are sufficiently large by considering their internal cluster connectivity. Hagen and Kahng introduced a technique called a ratio cut. It is defined as a fraction of a graph cut over cardinalities of its components [114]. First, let us define the “characteristic” vector $\mathbf{u} \in \mathbb{R}^m$ associated with some partition $\mathbb{A} \subset \mathbb{V}$ such that:

$$\mathbf{u} \stackrel{\text{def}}{=} [u_1, u_2, \dots, u_m], \text{ where } u_j = \begin{cases} \frac{1}{\sqrt{\text{card}(\mathbb{A}_i)}} & \text{if } v_j \in \mathbb{A}, \\ 0 & \text{if } v_j \in \mathbb{A}^c. \end{cases}, j \in \{1, 2, \dots, m\}. \quad (9.11)$$

Using the definition (9.11), we can introduce a ratio cut so that:

$$\text{Rcut}(\mathbb{A}_1, \dots, \mathbb{A}_k) = \sum_{i=1}^k \frac{\text{cut}(\mathbb{A}_i, \mathbb{A}_i^c)}{\text{card}(\mathbb{A}_i)} \quad (9.12a)$$

$$= \sum_{i=1}^k \mathbf{u}_i^T \mathbf{L} \mathbf{u}_i \quad (9.12b)$$

$$= \sum_{i=1}^k (\mathbf{U}^T \mathbf{L} \mathbf{U})_{ii} = \text{Tr}(\mathbf{U}^T \mathbf{L} \mathbf{U}), \quad (9.12c)$$

where $\text{Tr}(\cdot)$ denotes a trace of a matrix, $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_k]$, and $\mathbf{L} \in \mathbb{R}^{m \times m}$ is an unnormalized graph Laplacian matrix. The matrix \mathbf{L} is defined such that:

$$\mathbf{L} = \mathbf{W} - \mathbf{D}, \quad (9.13)$$

where $\mathbf{D} \in \mathbb{R}^{m \times m}$ is a diagonal matrix commonly called a degree matrix. It contains degrees of vertices in \mathbb{V} on a main diagonal:

$$\mathbf{D} = \begin{bmatrix} \deg(v_1) & & \\ & \ddots & \\ & & \deg(v_m) \end{bmatrix} \quad (9.14)$$

where a degree associated with a vertex v_i is defined as follows:

$$\deg(v_i) \stackrel{\text{def}}{=} \sum_{j=1}^m \mathbf{W}_{ij}. \quad (9.15)$$

The example of a labelled graph and associated matrices \mathbf{W} , \mathbf{D} , and \mathbf{L} is depicted in Figure 9.3.

Now, we can write a problem of minimizing a ratio cut as a constrained trace minimization so that:

$$(\mathbb{A}_1^*, \dots, \mathbb{A}_k^*) = \arg \min_{\mathbb{A}_1, \dots, \mathbb{A}_k} \text{Tr}(\mathbf{U}^T \mathbf{L} \mathbf{U}) \quad \text{s.t.} \quad \mathbf{U}^T \mathbf{U} = \mathbf{I}. \quad (9.16)$$

Since the formulation (9.16) takes the form of discrete energy minimization, it is known that obtaining a global solution is generally an NP-hard problem. Relaxing this problem to dispose of its discreteness, we allow the solution to take values in \mathbb{R} . It results in the following relaxed optimization problem:

$$\mathbf{H}^* = \arg \min_{\mathbf{H} \in \mathbb{R}^{n \times k}} \text{Tr}(\mathbf{H}^T \mathbf{L} \mathbf{H}) \quad \text{s.t.} \quad \mathbf{H}^T \mathbf{H} = \mathbf{I}. \quad (9.17)$$

By using the Rayleigh-Ritz theorem [115], the solution of problem (9.17) is given by setting the matrix \mathbf{H} as the matrix containing the first k eigenvectors of a graph Laplacian matrix \mathbf{L} as its columns [116]. Afterwards, vector quantification or other clustering techniques are commonly exploited for reconstructing the indicator vectors associated with the graph partitions from this real-valued solution. New representations of samples in null-space \mathbf{L} correspond to the rows of \mathbf{H} . It leads to the unnormalized spectral clustering introduced in the widely cited paper by Luxburg [105]. From another point of view, this approach can be interpreted as quantifying new representations of training vectors X that are projected onto the null-space of a (discrete) Laplace operator $\text{Null}(\mathbf{L})$.

9.2 Normalized spectral clustering overview

Studying publications dealing with the normalized approaches to spectral clustering, a reader should put an effort into distinguishing between the original versions of the algorithms, and their modifications or adaptations for particular problems. Note that there is no unique convention in the literature, which approach is precisely related to a normalized one.

We can meet with using different operators such as a symmetric normalized Laplacian matrix:

$$\mathbf{L}_{sym} = \mathbf{D}^{-\frac{1}{2}} (\mathbf{D} - \mathbf{W}) \mathbf{D}^{-\frac{1}{2}}, \quad (9.18)$$

its random walk version:

$$\mathbf{L}_{rw} = \mathbf{D}^{-1} (\mathbf{D} - \mathbf{W}), \quad (9.19)$$

or, alternatively, a Markov matrix:

$$\mathbf{P} = \mathbf{D}^{-1} \mathbf{W}. \quad (9.20)$$

Authors also exploit various proportions of spectra related to these operators, or a particular eigenvector for example the Fiedler vector. These approaches also differ in solving a

corresponding eigenproblem, e.g. a standard or generalized one, and decoding the indicator vectors from spectra, e.g. using vector quantification or bisection techniques. We would like to mention a highly cited technical report published by Verma and Meila [117]. It seems they proposed modifying an original Shi Malik algorithm [118, 119]. However, their approach is slightly different. Instead of the multiway bisection based on exploiting the Fiedler vector of \mathbf{L} that was obtained using a generalized eigenproblem:

$$\mathbf{L}\mathbf{e} = \lambda\mathbf{D}\mathbf{e}, \quad (9.21)$$

they processed the second dominant eigenvector of the matrix \mathbf{P} , i.e. the eigenvector corresponds to the second dominant eigenvalue. While Meila and Shi [120] proved that these approaches are equivalent in a manner of their clustering properties, the spectral properties of a matrix \mathbf{P} are additionally interpreted in the sense of random walk. Moreover, corresponding eigenvalues obtained using (9.21) and these associated with \mathbf{P} are from different ranges:

- the eigenvalues of \mathbf{P} lie in $[-1, +1]$,
- the eigenvalues attained using (9.21) belong to $[0, 2]$.

Since these methods are linked together, we think, a good way could be focusing on them chronologically.

The idea of the normalized spectral clustering and the corresponding preliminary results presented for image segmentation problems were first published by Shi and Malik in the previously mentioned conference paper related to the Conference on Computer Vision and Pattern Recognition [118] held in 1997. Then, they extended it to a journal paper, see [119], which could be considered as the prior work and, currently, one of the most widely cited papers associated with normalized spectral clustering.

In these papers, the authors discussed the disadvantages of minimal k -cut problem (9.10) for segmentation of real-world images. Specifically, they focused on the unnatural bias of partitioning out the singleton sets. We mentioned such an issue in the previous section. To overcome it, they proposed to incorporate a measuring of association among the vertices \mathbb{V} based on a volume of partition \mathbb{A}_i denoted as $\text{vol}(\mathbb{A}_i)$, for $i \in \{1, 2, \dots, k\}$:

$$\text{assoc}(\mathbb{A}_i, \mathbb{V}) \stackrel{\text{def}}{=} \sum_{u \in \mathbb{A}_i, t \in \mathbb{V}} \mathbf{W}(u, t) = \sum_{u \in \mathbb{A}_i} \text{deg}(u) = \text{vol}(\mathbb{A}_i), \quad (9.22)$$

into the graph cut formulation (9.7). This leads to a disassociation measure among graph partitions called the normalized cut (NCut), which is defined as follows:

$$\text{Ncut}(\mathbb{A}_1, \mathbb{A}_2, \dots, \mathbb{A}_k) \stackrel{\text{def}}{=} \sum_{i=1}^k \frac{\text{cut}(\mathbb{A}_i, \mathbb{A}_i^c)}{\text{assoc}(\mathbb{A}_i, \mathbb{V})} = \sum_{i=1}^k \frac{\text{cut}(\mathbb{A}_i, \mathbb{A}_i^c)}{\text{vol}(\mathbb{A}_i)}. \quad (9.23)$$

From a definition of an association measure in (9.22), we can see that a total cost of direct connections from vertices in the partition \mathbb{A}_i to all vertices in the graph G , note that including these in \mathbb{A}_i , is equal to the total degree of nodes belonging this partition. It arises from the fact that if two vertices are not adjacent, then the related weight is 0. Using this fact, the relation in (9.22) can be easily proven.

Comparing a ratio (9.12) and a normalized cut (9.23) approaches, both of them satisfy a min-max clustering principle.² Unlike a ratio cut, which minimizes the graph cut and simultaneously maximizes the cardinalities of desired components, minimizing a graph cut employing a normalized approach is proportional to maximizing the volume of the components. It tends that NCut could outperform RatioCut and provide a better quality solution in the sense of higher compactness of the partitions for some practical applications, e.g. segmentation-based object categorization [118, 119].

Recall. Compactness gives us an insight into how similar are samples belonging to the same group. In practice, it is better to compare compactness with other typically used metrics, such as the Davies-Bouldin criterium or the Silhouette index, which tests within-cluster consistency. Moreover, not always higher compactness of an overall solution is better or more reasonable than others. A counterexample could be finding defects (typically a small-size partition) in images or, generally, outliers in an initial dataset. Using a proper method is problem/data specific, and in-depth experiments provide us “a bigger picture” related to the quality of the achieved results. We want to point out this fact because finding the better size-balanced partitions could not be the best approach for all problem types. As we mentioned above, it highly depends on the problem and nature of the data either.

Originally, Shi and Malik [118, 119] considered just special cases of graphs such that the connectivity of these graphs is 1. They formulated a normalized cut as a trace minimization problem in the following form:

$$\mathbf{H}^* = \arg \min_{\mathbf{H} \in \mathbb{R}^{n \times k}} \text{Tr}(\mathbf{H}^T \mathbf{L} \mathbf{H}) \quad \text{s.t.} \quad \mathbf{H}^T \mathbf{D} \mathbf{H} = \mathbf{I}. \quad (9.24)$$

We can determine a solution of (9.24) as in the case of the unnormalized spectral clustering. Exploiting the Rayleigh-Ritz theorem, this solution is given by the matrix \mathbf{H} containing the first k generalized eigenvectors of a graph Laplacian matrix \mathbf{L} (9.21) as its columns.

Since these eigenvectors represent indicator vectors embedded into a real-value domain, we can use vector quantification techniques for reconstructing the indicator vectors, as in the case of unnormalized spectral clustering. This leads to the normalized spectral clustering presented in Luxburg [105]. Shi and Malik in their original approach, which is based on the

²The standard clustering paradigm mentioned in the previous sections: samples are reasonably divided into clusters so that extra-cluster similarities are minimized, and within-cluster similarities are maximized.

assumption that the connectivity of a graph G is 1, employed a bisection technique instead of vector quantification.

Let us further denote:

$$\widehat{\mathbf{H}} = \mathbf{D}^{\frac{1}{2}} \mathbf{H}, \quad (9.25)$$

then we can rewrite (9.24) as follows:

$$\widehat{\mathbf{H}}^* = \arg \min_{\widehat{\mathbf{H}} \in \mathbb{R}^{n \times k}} \text{Tr}(\widehat{\mathbf{H}}^T \mathbf{L}_{sym} \widehat{\mathbf{H}}) \quad \text{s.t.} \quad \widehat{\mathbf{H}}^T \widehat{\mathbf{H}} = \mathbf{I}, \quad (9.26)$$

where $\mathbf{L}_{sym} = \mathbf{D}^{-\frac{1}{2}} \mathbf{L} \mathbf{D}^{-\frac{1}{2}}$ and is called the symmetric normalized Laplacian.³ By this substitution and exploiting the Rayleigh-Ritz theorem, we can observe that we can equivalently solve the standard eigenproblem associated with \mathbf{L}_{sym} instead of a generalized eigenproblem arising from (9.24).

However, the features of the initial samples \mathbb{X} projected into $\text{Null}(\mathbf{L}_{sym})$ are proportional to square roots of degrees corresponding to vertices \mathbb{V} in the case of normalized spectral clustering as we can see from the relation (9.25). Therefore, $\widehat{\mathbf{H}}$ does not contain the actual indicator vectors related to the graph partitions $\mathbb{A}_1, \dots, \mathbb{A}_k$. Still, it contains non-uniformly scaled indicator vectors related to these partitions embedded into a real-value domain. Consequently, the new coordinates of vectors \mathbb{X} in $\text{Null}(\mathbf{L}_{sym})$ could be in different ranges and they need to be post-processed. We introduce two approaches on how to deal with this issue. Luxburg proposed the first one, and the second is based on obtaining the relaxed indicator vectors directly.

As in the case of the unnormalized spectral clustering, these new coordinates correspond to rows of $\widehat{\mathbf{H}}$. In [105], Luxburg proposed to normalize the new representations of training samples by L2 norm there, i.e. a row normalizing:

$$\widetilde{\mathbf{H}}_i = \frac{\widehat{\mathbf{H}}_i}{\|\widehat{\mathbf{H}}_i\|}. \quad (9.27)$$

In an ideal case, this normalization could not be necessarily required when degrees of vertices are sufficiently similar, because this could not affect the indicator vectors by scaling (9.25) and new representations could be used directly for subsequent processing. On the other hand, when degrees of vertices differ significantly, such normalization does not change the statistical distribution of the new representations of data but could affect outliers – typically samples for those corresponding vertices of graph G have low degrees. Luxburg discussed this problem from the perturbation theory point of view in [105].

³Note that \mathbf{L}_{sym} is symmetric even the graph G models non-symmetric relations, e.g. a case of directed graphs.

While the proposed normalizing affects outliers that could lead to misclassifying them, we can recover the relaxed indicator vectors from $\widehat{\mathbf{H}}$ directly. Let us consider an ideal case when columns of \mathbf{H} are piecewise constant,⁴ and degrees of all vertices V are non-zero. Then, we can directly obtain indicator vectors embedded into the real-value domain by the following multiplication:

$$\mathbf{H} = \mathbf{D}^{-\frac{1}{2}} \widehat{\mathbf{H}}. \quad (9.28)$$

After post-processing $\widehat{\mathbf{H}}$ proposed above, we can employ vector quantifying techniques for reconstructing discrete indicator vectors as in the earlier presented spectral clustering approaches.

9.3 Benchmarks

Image segmentation is a process of extracting meaningful information related to the structure of objects, discerning various parameters, or separating the regions of interest corresponding to foreground objects from the background of the image scene. This process is a crucial part of modern image processing methods such as object detection, localization, or other real-world applications.

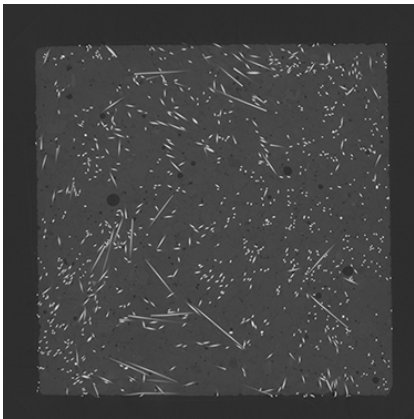


Figure 9.4: Example of transverse CT image of fibre-reinforced concrete.

In the most practical applications, regions of interest are considered as parts of an image that pixels have similar characteristics, e.g. grey level/-colour, or similar local gradients in pixel points.

In cooperation with Institute of Geonics of the Czech Academy of Sciences, we published paper *Advanced approach of material region detections on fibre-reinforced concrete CT-scans* [121], where we proposed exploiting unsupervised learning technique, namely *k*-means++, for detecting homogeneous parts, say areas of specific materials, in transverse CT (Computed Tomography) scans of fibre-reinforced concrete. Essentially, by this application, we go back to the Steinhaus' idea associated

with partitioning solid material. Moreover, we introduce a technique for reduction of air distribution that employs structural analysis based on connected components as small air bubbles are undesirable in practical computations, see the example of achieved results in Figure 9.5 on the next page – source image depicted in Figure 9.4.

⁴In practice, the values are affected by numerical errors which depend on employed eigensolver, numerical precision, scaling or normalizing eigenvectors during the computation.

In this chapter, we present a few results achieved exploiting image segmentation-based object categorization on benchmarks including volumetric images in Section 9.3.1 and photos in Section 9.3.2.

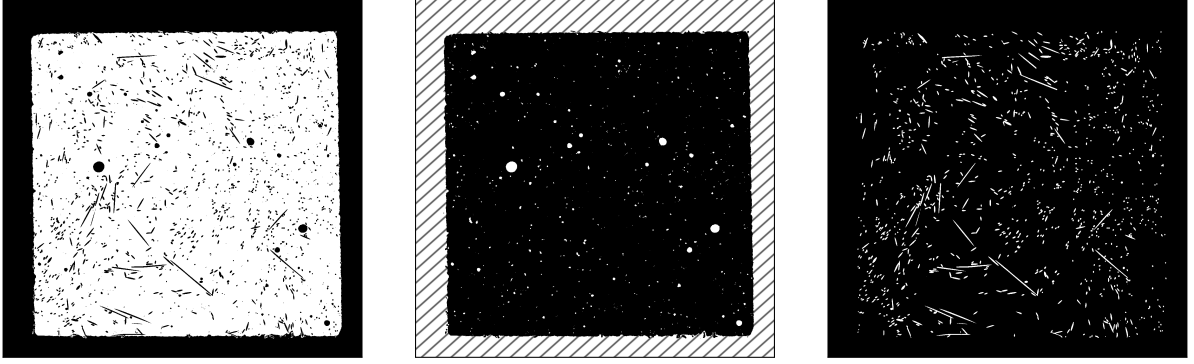


Figure 9.5: Binary masks of achieved material regions, namely from left concrete, air, fibres.

9.3.1 2-phase segmentation of volumetric images

In this section, we focus on the 2-phase image segmentation on volumetric (3D) images, for which we employ unnormalized and normalized spectral clustering methods and compare obtained results. We cooperate on this application with two experts Dr. Stanislav Harizanov and Prof. Svetozar Margenov from Institute for Parallel Processing, Bulgarian Academy of Sciences.⁵ This work is currently unpublished in any conference or journal paper; just part of it was outlined in the following preprint [122], where a stochastic approach based on Bartlett test for estimating dimension of $\dim \text{Null}(\mathbf{L})$, i.e. null-space of a graph Laplacian, is discussed.⁶

Volumetric images used for benchmarking the approaches are available on the following GitHub page <https://github.com/ml4py/specclus4py>, where the owners' consent is mentioned in Acknowledgements. The volumetric images are visualized in Figure 9.6. Recall. In a traditional approach related to spectral clustering [105], vectors belonging to an initial set \mathbb{X} are projected onto the null space $\text{Null}(\mathbf{L})$. A dimension of this null space equals:

$$\dim \text{Null}(\mathbf{L}) = m_{\mathbf{L}}(\lambda_0), \quad (9.29)$$

where $m_{\mathbf{L}}(\lambda_0)$ denotes a geometric multiplicity of a zero eigenvalue λ_0 corresponding to an operator \mathbf{L} .

⁵The IPP BAS website <http://www.bas.bg/clpp/en/indexen.htm>

⁶**Authorship note:** Since an author of this thesis is only author of the preprint [122], and the preprint is currently unpublished, a some sections of results are just copied and paste in this section.

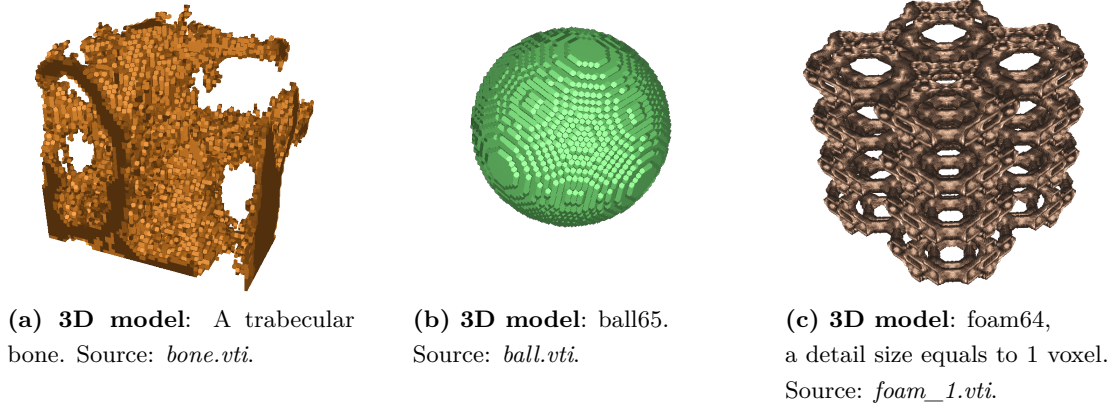


Figure 9.6: Visualization of volumetric images used in this section for benchmarking.

Then, such transformed samples are clustered, e.g. by applying vector quantification techniques namely k -means (Section 8.2) or k -means++ (Section 8.3). In many practical applications employing the spectral clustering, one assumes that an actual number of clusters k^* associated with an underlying model is:

$$k^* = \dim \text{Null}(\mathbf{L}). \quad (9.30)$$

Since the last step of spectral clustering is typically based on the vector quantification using the “ k -means like algorithms,” we can consider an underlying model as a special case of the Gaussian mixture model. Therefore, an optimal k^* could be estimated such that it minimizes the Bayesian information criterion [123] or the Akaike information criterion [124], discussed in [125], or seeking inflexion point in scree plot representing profile of eigenvalues [126]. A more practical way could be to determine $\dim \text{Null}(\mathbf{L})$ statistically based on testing equality variances of the smallest eigenvalues – presented in [127, 122] and will be discussed in the following text.

This test was firstly proposed by Bartlett [128] in the context of estimating the number of factors represented by means of eigenvectors and associated with the q largest eigenvalues of a covariance matrix $\mathbf{\Sigma} \in \mathbb{R}^{n \times n}$ in the sense of PCA. We refer the following comprehensive survey on significant results in this area [129].

The standard Bartlett test formulates the null-hypothesis H_0 that k eigenvalues associated with unselected eigenvectors of $\mathbf{\Sigma}$ have a small magnitude and small variance. Assuming this, taking the likelihood ratio statistic:

$$V_k = \prod_{i=q+1}^d \lambda_i / \bar{\lambda}_k, d = k + q, \bar{\lambda}_k = \frac{1}{k} \sum_{i=q+1}^d \lambda_i \quad (9.31)$$

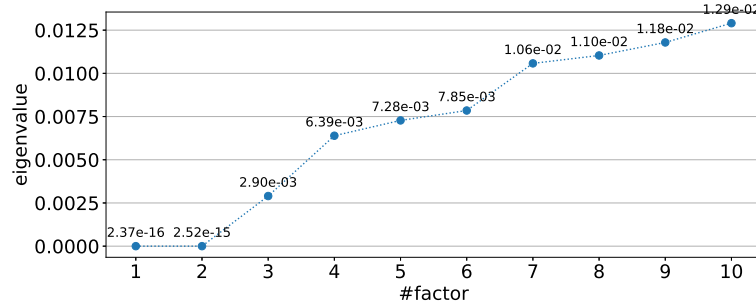


Figure 9.7: This screeplot illustrates a profile of the first 10 eigenvalues associated with the 3D unnormalized Laplacian for a trabecular bone depicted in Figure 9.6a. Finite differences were computed using 26 nearest neighbours, and similarity among them was determined using an RBF function (standard deviation $\sigma = 0.01$). We solved associated eigenproblem employing the SLEPc framework (solver type ARPACK, $\text{rtol}=1\text{e}-1$, $\text{nev}=10$).

and linkage factors into account [130, 131].

The authors showed that the statistic T defined as:

$$F = \left(n - q - \frac{k^2 + 1}{3k} - \frac{1}{6} + \sum_{i=1}^q \frac{\bar{\lambda}_k^2}{(\lambda_i - \bar{\lambda}_k)^2} \right), \quad (9.32)$$

$$T = -F \ln V_k,$$

follows χ^2 distribution with $\frac{1}{2}(k-1)(k+2)$ degrees of freedom. Principle of the test is to find the smallest acceptable a value of q so that $P(X < T) \leq 1 - \alpha_{crit}$, where α_{crit} is a significance level that typically chosen as 0.05 or 0.01.

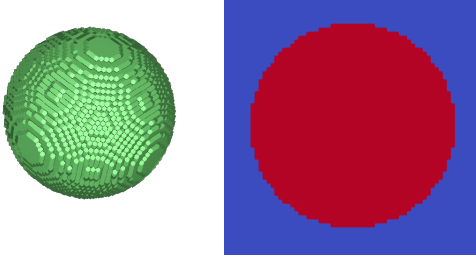
From a concept point of view, we can straightforwardly see similarities between the spectral clustering and PCA. Both involve eigen decomposition of the square symmetric matrix, however locations of relevant information are associated with opposite parts of spectra. This arises from the fact that the Laplacian matrix is considered as the scaled precision matrix \mathbf{Q} , which is pseudoinverse of covariance matrix:

$$\mathbf{L} = \delta \mathbf{Q} = \mathbf{\Sigma}^\dagger, \quad (9.33)$$

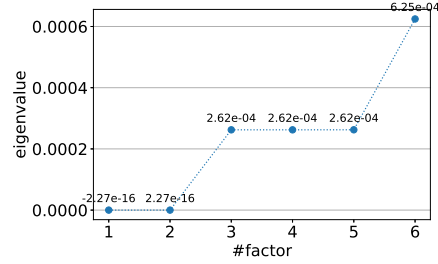
we refer the following paper [132] for further information. It seems to be reasonable to exploit $T \sim \chi^2$ for testing homoscedasticity of the smallest eigenvalues of \mathbf{L} , ergo estimating $\dim \text{Null}(\mathbf{L})$, with some minor modification of the original hypothesis test (9.32). This is originally proposed in Bruneau et. al. [127]. We proposed an additional correction of numerical errors, which could affect the test statistic. This post-processing approach is outlined in Algorithm 8.

Algorithm 8: EIGENVALSPROCESS

Input : Λ , err
 1 **for** $i = 1 \rightarrow d$ **do**
 2 **if** $err_i \geq 1$. **then**
 3 **break**;
 4 $exp \leftarrow \text{FLOOR}(\log_{10}(err_i));$
 5 $\hat{\lambda}_i \leftarrow \text{ROUND}(\lambda_i, -1 * exp);$
 6 **if** $\hat{\lambda}_i = 0$ **then**
 7 $\hat{\lambda}_i \leftarrow \text{POW}(10, exp);$
 8 **if** $\lambda_i < 0$ **then**
 9 $\hat{\lambda}_i \leftarrow -\hat{\lambda}_i;$
Output: $\hat{\Lambda} = \text{diag}(\hat{\lambda}_1, \hat{\lambda}_2, \dots, \hat{\lambda}_d)$



(a) From left: a visualization of a 3D model related to the ball65 dataset, an example of a horizontal slice associated with an underlying volumetric image ($65 \times 65 \times 65$ voxels).



(b) The first 7 eigenvalues associated with normalized Laplacian so that standard deviation of RBF is $\sigma = 0.01$, finite differences computed using $knn = 6$. Solver ARPACK, $rtol = 1e-4$, $nev = 10$).

Figure 9.8: An example of spectrum related to the normalized Laplacian affected by rounding-off errors caused by limits of floating point arithmetic so that it consists of a negative eigenvalue.

Another common issue goes from rounding-off errors caused by limits of floating-point arithmetic. Eigensolver could return some small negative eigenvalues very close to 0 even \mathbf{L} is a symmetric positive definite matrix. We show an example illustrating this in Figure 9.8. By similar comments as in the previous case, we cannot implicitly consider these eigenvalues as 0. On the other side, comparing negative eigenvalues causes non-defined value related to (9.32), specifically in the term $\ln V_k$. Since the test is based on analysing variances among eigenvalues, we can shift them to be positive without loss of generality so that:

$$\hat{\lambda}_i = \lambda_i + 2|\lambda_0|. \quad (9.34)$$

While spectral clustering provides regularization of noisy data implicitly⁷, we do not deal with meshes corrupted by random noise in following experiments. This setting helps us focus on object geometries, their representations in null spaces of related Laplacians, and we can then straightforwardly compare clustering results with ground truth by means of the Hamming distance $\|\cdot\|_H$ to analyse how precisely the merging works.

In the following experiments, we set a sufficiently high relative tolerance $rtol = 1e-3$ in case of the ARPACK eigensolver [136]. This value was chosen as a trade-off between numerical computation stability and a faster convergence rate. A high tolerance was set for the k -means solver as well. Specifically, this was 0.1. An initial guess was determined using the k -means++ initialization, and attempts for restarting the k -means solver was 5. Setting a number of eigenpairs to compute, i.e. -EPS_NEV option in the SLEPc framework [137], seems to be tricky, because not proper setting could cause a slower convergence rate or a solution diverging. A common recommendation is to set this value higher than a number of interesting eigenpairs. We tested various eigenpair counts, and it seems a good value for the following experiments is 50.

Computations run parallelly that 40 MPI processes on the Hugo multiprocessor system, technical parameters are outlined in Table 9.1. Since parallel addition/multiplication are not commutative and we run computation determining subset a spectra of lower magnitude that $rtol$ is high, the solver could not converge to the same solution for different runs. Thus, a test was repeated 100 times and reported observations are in a form $mean \pm std$. The results are summarized in Table 9.2 on page 131.

Processor	4× Intel Xeon Gold 6152 @ 2.1 GHz (22 cores)
RAM	1536 GB (48 × 32 GB) DDR4-2666 MHz
Graphical accelerator	NVidia GTX 1050 TI 4GB

Table 9.1: Parameters of Hugo multiprocessor system (Huawei FusionServer 2488H V5).

Analysing these results, we can see that an issue related to capture the geometry of the mesh arose in cases, where unnormalized Laplacian was build using 6 knn and normalized one of the same knn that the new representations of voxels were normalized. The first issue is a consequence of non-commutativity parallel addition/multiplication, and a high relative tolerance of the eigensolver, which converged to the solution that shrank a null-space dimension to 5 in 6 cases and new representations could not capture the geometry of the mesh properly. The second one is related to affecting outliers by additional normalization mentioned above. These affected outliers are depicted in Figure 9.9 (p. 131).

⁷Regularization properties arise from the Mumford and Shah regularization point of view mentioned [133] in author’s previous works [134, 135].

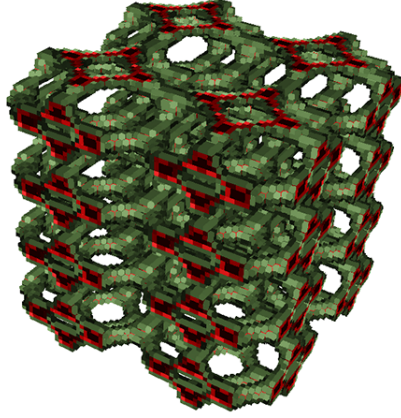


Figure 9.9: A visualization of a solution associated with the normalized Laplacian of the 6 knn, that coordinations of new representations were normalized. Misclassified voxels are red colored.

Comparing a dimension of a null-space related to the unnormalized and normalized Laplacian associated with 6 knn, we can see that $\dim \text{Null}(\mathbf{L}) > \dim \text{Null}(\mathbf{L}_{sym})$. It results that the unnormalized Laplacian is slightly connected with RatioCut whereas the normalized Laplacian to NCut [105]. Since RatioCut minimizes the graph cut and simultaneously maximizes cardinality of desired components and NCut is proportional to the volume of these components, normalized spectral clustering provides a smaller number of components related to the similarity graph G .

Increasing the size of the neighbourhood, σ_{max} and the number of iterations needed for converging eigensolver decreased. This follows a few dependent facts. By increasing the size of the neighbourhood, more voxels are incorporated there, and we need to filter out voxels from different groups more precisely by decreasing σ_{max} . It leads to maximizing internal cluster connectivity and minimizing intra-cluster variance, and, from the spectral properties of the operators, better separating zero eigenvalues from the rest of the spectra causing quicker convergence rate of the eigensolver. Thus, we basically reconstructed relaxed indicator vectors of components in cases of knn 18 and 26 for the both unnormalized and normalized Laplacians.

eigenvecs postproc.	operator			eig. solver #iters.	revr. Bartlett test		KMS		$\ \cdot\ _H$
	type	#knn	σ_{max}		p-val.	dim Null	#iters.	RSS _{Best}	
–	unnorm.	6	0.127	279 ± 60	$3.4e-4 \pm 7.3e-4$	7 ± 1	2	$5.4e+0 \pm 1.1e-1$	$1.2e4 \pm 5.1e4$
–	unnorm.	18	0.123	104 ± 15	$4.7e-2 \pm 2.5e-3$	2	2	$5.6e-8 \pm 3.3e-8$	0
–	unnorm.	26	0.122	103 ± 13	$3.9e-2 \pm 4.8e-3$	2	2	$1.8e-6 \pm 5.7e-8$	0
norm.	norm.	6	0.126	172 ± 9	$8.0e-5 \pm 2.6e-5$	≈ 7	4	$2.6e+4 \pm 7.1e+1$	16,896
recov.							2	$1.4e+0 \pm 7.1e-2$	0
norm.	norm.	18	0.122	71 ± 1	$2.9e-2 \pm 6.4e-3$	2	2	$4.9e-7 \pm 1.1e-7$	0
recov.							2	$2.5e-1 \pm 1.9e-1$	0
norm.	norm.	26	0.122	61 ± 1	0.0365	2	2	$1.2e-7 \pm 6.3e-8$	0
recov.							2	$5.1e-1 \pm 3.2e-1$	0

Table 9.2: Comparison of the results attained by means of the unnormalized and normalized Laplacians associated with the foam64 (1 voxel) dataset depicted in Figure 9.6c. The results are summarized from 100 runs of each setting and observations reported as *mean* \pm *std* if that makes a sense. Associated eigenproblems were solved employing the SLEPc framework such that solver type set to ARPACK, rtol=1e−3, nev=50.

9.3.2 Image segmentation-based object categorization

In the previous text, we introduced a direct application of the k -means to problem of image segmentation. However, in case of photos showing “real-world” scenes such as cats playing with a ball, a more natural way to formulate image segmentation is exploiting the minimization of the Mumford and Shah functional [133]:

$$\arg \min_{u(x,y), B} \left\{ \alpha \int_{\Omega_{roi}} [g(x,y) - u(x,y)]^2 dS + \beta \int_{\text{int}(\Omega_{roi})} [\nabla u(x,y)]^2 dS + \gamma |B| \right\}, \quad (9.35)$$

where $g(x,y) : \Omega \rightarrow \mathbb{R}^p$ is an input image function such that $p = 1$ in order to grayscale images and $p = 3$ in case of colour images. Ω denotes the domain of image function $g(x,y)$, $\Omega_{roi} \subset \Omega$ is region of interest, and $\text{int}(\Omega_{roi}) := \Omega_{roi} \setminus B$. Function $u(x,y)$ is solution of segmentation problem; typically, $u(x,y) \in C_{loc}^1$, $B = \partial\Omega$ is boundary among objects in image scene. Commonly, we suppose that B is piecewise smooth curve. The integrals are taken in the Lebesgue sense. Scalars α, β, γ are tuning parameters and their values are problem dependent.

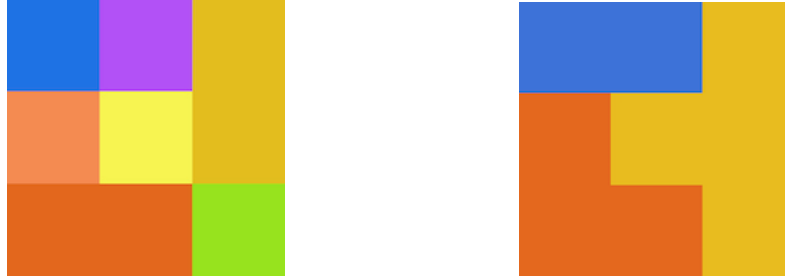


Figure 9.10: From left: piecewise smooth image function $g(x,y) : \Omega_{roi} \rightarrow \mathbb{R}^3$ and its C_{loc}^1 function approximation $u(x,y) : \Omega_{roi} \rightarrow \mathbb{R}^3$.

In paper [134], we formulated problem of image segmentation for piecewise smooth images, see Figure 9.10, and transformed the Mumford and Shah functional minimization problem as a minimization of an artificial quadratic term on a general open set, which directly leads to problem formulation as eigenproblem of graph Laplacian. Moreover, we showed fundamental connections between the Laplace-Beltrami equation on the Riemannian manifold and the eigenproblem. Typically, the similarities among graph vertices, which represent pixels, are modelled using an RBF function:

$$k(\mathbf{x}, \hat{\mathbf{x}}) = \exp\left(-\frac{\|\mathbf{x} - \hat{\mathbf{x}}\|^2}{2\sigma^2}\right), \quad (9.36)$$

where σ is standard deviation. In this paper, we also demonstrated a capability of spectral clustering, which is called image segmentation-based object categorization in image processing

communities, performed on real-world scenes, example is illustrated in Figure 9.11.



Figure 9.11: Cattedrale di Santa Maria del Fiore: the demonstration of image segmentation-based object categorization. [138]

Further, in paper [135], we generalized the problem above and showed a connection with the Helmholtz equation. Afterwards, we point out, in case of Neumann boundary condition $\frac{\partial u}{\partial \mathbf{n}}$ on $\partial\Omega$, the solution u of the Helmholtz equation is that $u \in L^2$. Additionally, we examined the connection between the Laplace operator and the graph Laplacian. For improving the numerical stability, we propose “a non-glueing image domain decomposition” based on the idea of not considering boundaries related to regions in the problem formulation.



Figure 9.12: Tatra T603: illustration of non-glueing image domain decomposition, presented in [135].

Chapter 10

Conclusions

The aim of this thesis is to study classical machine learning (ML) models and demonstrate their ability to handle complex and data-intensive applications, involving adaption of deterministic solvers to effectively train these models (primarily classification models) in parallel. Unlike deep neural networks, which are commonly investigated in ML research, classical models could use solvers based on deterministic approaches. This approach simplifies feeding samples to the training procedure, since the whole data set is used instead of batches. The training of classical models also offers clear explanation of attained model qualities and makes easier an analysis of the numerical behavior of the underlying solver.

The main content of this text is organized into two parts based on the concept and purpose of the introduced models. The first part focuses on supervised learning on classification tasks, while the second part discusses and presents results related to unsupervised learning. In the second part, the author outlines the first results of research on ML, in which he was involved during first and second years of his doctoral studies.

The theoretical background associated with supervised learning, which are represented by means of SVM models, was introduced in two separate chapters 2 and 3. The first of them was dedicated to hard-margin approaches for linearly separable training samples. Even though this approach is not practically useful, it provides a fundamental framework for introducing soft-margin approaches later in Chapter 3. First, Chapter 2 provided a motivation in Section 2.1, where a possible poor generalization ability of a simple perceptron was discussed. To overcome this issue, a classification approach based on maximal-margin classifiers was introduced in Section 2.2. This section sets up a definition related to a maximal margin in the form of a normalized functional margin concerning $\|\mathbf{w}\|$, where \mathbf{w} represents a normal vector of a separating hyperplane. Then, a framework based on the Vapnik-Chervonenkis dimension (VC dimension) for quantifying a classifier performance was outlined, and it was shown that a classifier having this dimension reasonably small attains a good generalization ability in the sense of the bias-variance tradeoff. Based on this observation, the primal hard-margin SVM

was then formulated. Section 2.3 briefly introduced a theory beyond the Lagrange duality, which was applied to the formulation of dual-hard margin SVM later in this section.

However, real-world data are mainly not linearly separable. The approach based on soft-margin for SVMs solves this issue so that a reasonable number of training samples can be misclassified so that misclassification error is quantified using the hinge loss function. The soft-margin SVMs were then introduced in Chapter 3, including ℓ_1 -loss and ℓ_2 -loss formulations in both the primal and dual forms for complete and relaxed-bias approaches. The main difference between the ℓ_1 -loss and ℓ_2 -loss formulations lies in properties of the Hessian matrices and sparsity of a classification models:

- the ℓ_1 -loss SVM has SPS Hessian, which could cause a slow convergence rate, on the other hand, it induces model sparsity, shrinks coefficients associated with some features to zero, and does a sort of “automatic” feature selection,
- the ℓ_2 -loss SVM has SPD Hessian, which could result in faster model training, however, a model is not sparse.

In the case of the relaxed-bias classification, bias b is not considered in a classification model, however, it is included in the problem by means of augmenting the vector \mathbf{w} and each sample with an additional dimension, discussed in Section 3.3. This approach results in the dual formulation without a homogenized equality constraint, and arising optimization problem could be cheaper to solve. The benchmark section, where these approaches were compared to each other, concluded this chapter.

When a classifier has been trained, it is essential to understand how it represents and generalizes the associated problem. The metrics for and approaches evaluating the performance of classifiers were introduced in Chapter 4, including techniques for parameter selection.

An optimization of underlying quadratic programming (QP) solvers for effectively training SVM models was outlined, discussed, and tested in Chapter 5. The ingredients for optimal initial guess, adaptive expansion step length for the MPRGP algorithm, and variants of the SMALXE algorithm such as SMALXE-M and SMALXE- ρ were introduced. The theory guarantees R-linear convergence of the MPRGP algorithm for a fixed step length less than $\frac{2}{\|\mathbf{Q}\|}$, where \mathbf{Q} represents a Hessian matrix. However, this step-length is commonly really small in the case of the SVM problems (caused by a large norm of \mathbf{Q}), which results in many expansion steps. The advantage of adaptive expansion methods on convergence rate was also studied in this chapter. The presented results are done in PermonSVM, Octave, and Python.

PermonSVM is a software package that was primarily developed for training SVM models on supercomputer systems, and was introduced in Chapter 6. It provides an implementation to train SVM models in parallel, and supports multi-node multi-GPU approach for AMD and NVidia graphic cards using HIP respective CUDA through backends implemented in PETSc. SVM packages in commonly used ML libraries are limited to tens of thousands of samples to

solve complete dual formulations. The largest complete dual problem presented in this thesis had more than 1.6 million training samples and over 3 million features, and it was related to suspicious URL prediction. The problem was successfully solved in 211s on 144 CPU cores (Barbora supercomputer at IT4Innovations) and the attained model achieved 94.68% in F1 metrics. **That makes PermonSVM a unique machine learning library.** The effective loading of data from a parallel file system was also discussed. For this purpose, a loader of the HDF5 format inspired by MATLAB v7.3 structure was implemented in the PETSc library, this loader was introduced in more detail in Section 6.2.

The development of PermonSVM, the scalability of QP solvers implemented in PermonQP used for training models, and employing PETSc as a building block became the main reasons for collaboration with world-leading research institutes, namely the Argonne and Oak Ridge National Laboratories (USA), aimed at wildfires localization in Alaska. Just for clarity, it took almost 4 years to be contacted by Dr. Richard Mills (Argonne National Laboratory), and another 1 year than he became a co-supervisor of this thesis. The results of this cooperation were presented at the premier international conferences: *the AGU Annual Meeting* (USA), *the IALE North America Annual Meeting* (USA), *the IEEE International Conference on Data Mining* (USA), and *the SIAM Conference on Computational Science and Engineering* (NL). The largest area used in benchmarks covered approximately 722 thousand km², corresponding to 42% of Alaska area. For training using classical models on such large data set, it was necessary to completely redesign workflow, which currently supports functionality for fusing multispectral-temporal satellite images (reflectance, temperature, and vegetation indexes), data transformations and filtering, and finally retrieving data sets (training, calibration, and test), and visualization functionality to validate each step in data processing. The workflow (over 10 thousand lines of code in Python) currently can handle satellite data from MODIS (Moderate Resolution Imaging Spectroradiometer) instrument aboard the Terra satellite, which NASA operates. Further, the MTBS product can be used for labeling regions on satellite images that were affected by fire. The best model achieved performance around 0.78 and 0.86 in mIoU and F1 metrics, respectively. Details about this application, implementation details were introduced in Chapter 7. Moreover, two articles in form of interview were published in *Academik* or *Universitas* journals; the articles are written in Czech and the links on them are outlined in Appendix C.4.

The last two Chapters 8 and 9 in this thesis focused on the topics of unsupervised learning, mostly clustering approaches. Chapter 8 covered vector quantification methods represented by the Lloyd-type algorithms such as k-means, k-means++, and their variants, while Chapter 9 introduced spectral clustering. The theoretical background presented there is mostly in the form of a brief review, and the approaches presented on two applications. The first one showed the detection of brittle and ductile fracture on a steel sample, which has been damaged by the drop-weight tear test, using vector quantification techniques, and the second

one employs spectral clustering for image segmentation on photos and volumetric data. Moreover, there were outlined parallel implementations of these methods, and, in Section 9.3.1, a statistical approach based on the Bartlett’s test of homogeneity of variances for estimating the multiplicity of zero eigenvalue of the Laplace matrix was discussed as well.

10.1 Overview of author’s contribution

From the author’s point of view, the main contribution of this thesis is to create applications involving and connecting knowledge from multiple research fields, including parallel programming for distributed model training, big data analysis, statistics, optimization, geoinformatics, remote sensing, and environmental engineering. The greatest achievement presented in this thesis is designing and implementing a workflow for wildfire localization in Arctic Alaska using multispectral-temporal satellite images and classical models, including SVM and XGBoost. This workflow incorporates various libraries and technologies, e.g. *branca*, *GDAL*, *Google EE Python API*, *folium*, *OpenCV*, *pandas*, *RAPIDS*, *scikit-learn*, *scipy*, and *PermonSVM* implemented for distributed model training on supercomputer systems. Gratefully, this workflow is being used and adopted by researchers from world-leading research institutes, Argonne and Oak Ridge National Laboratories (USA). The second significant achievement mentioned in this work is implementation of *PermonSVM* (and the whole *PERMON* toolbox) and mention it as external software based on the *PETSc* framework on the official *PETSc* web pages.

For effective training of models, the QP solvers implemented in the *PermonQP* package have been adopted, which involves optimal settings of parameters, selecting suitable initial guess, using adaptive expansion approaches, or testing variants of the *SMALXE* algorithms. Other adaptations (directly related to solver modification) include effectively implementing a warm-start approach for optimal parameter selection using grid-search combined with cross-validation. This approach is based on scaling a solution attained for the previous combination of parameters. Last, a stopping criterium considering a duality gap between primal and dual functional was developed for ℓ^2 -loss SVM formulation. These modifications and adaptations done on QP solvers are other core contributions of this doctoral thesis.

Further, the author attempted to provide a view of SVM formulations, including dualization using the Lagrange duality, and, additionally, introduced both primal and dual formulations in a matrix form beside standardly used formulations in “a sum form.”

The contributions to a part related to unsupervised learning may involve parallel implementation of vector quantification algorithms demonstrated on an application focused on detecting brittle and ductile regions of steel sheet fractured by drop-weight tear test. The next one could be an improved approach for estimating the multiplicity of zero eigenvalues used in spectral clustering and image segmentation (volumetric data and photos) using spectral clustering.

The list of author’s publications is mentioned in Appendix A, research projects can be found in Appendix B, and other activities (involving supervising students and software development) in Appendix C.

10.2 Possible directions of further research

The application of wildfire localization involves many subtasks, which offer room for further improvement, development, and research. One possible direction could be using hybrid ML models, combining classical models such as SVM or XGBoost with deep neural networks. Since the semantic segmentation models used for wildfire localization are basically time series classifiers, we can increase their complexity by combining them with FCN-LSTM [139], i.e. fully convolutional recurrent neural network, or UNet modified for processing a series of images, e.g. LSTM-UNet [140, 141]. In these approaches, the neural networks serve as a feature extractor and SVM, or XGBoost, is then used as a classification layer. This could also require a modification of the training phase associated with these classical models to a mini-batch approach and redesigning the QP solvers, such as MRPGP or SMALXE, to a stochastic manner. Other research could require the optimal selection of unknown parameters, e.g. using the Bayesian optimization [142] instead of an essential grid-search.

Bibliography

- [1] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436, 2015.
- [2] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. The MIT Press, Jan. 2017. 800 pages. ISBN: 0262035618.
- [3] A. L. Katole, K. P. Yellapragada, A. K. Bedi, S. S. Kalra, and M. S. Chaitanya. Hierarchical deep learning architecture for 10k objects classification, 2015. DOI: 10.5121/csit.2015.51408. eprint: arXiv:1509.01951.
- [4] Y. Bengio, A. Courville, and P. Vincent. Representation learning: a review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, Aug. 2013. DOI: 10.1109/tpami.2013.50.
- [5] T. M. Mitchell. *Machine Learning*. McGraw-Hill Education, 1997. ISBN: 0070428077.
- [6] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas. Supervised machine learning: a review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160:3–24, 2007.
- [7] T. Hastie, R. Tibshirani, and J. Friedman. Unsupervised learning. In *The elements of statistical learning*, pages 485–585. Springer, 2009.
- [8] G. Brockman, V. Cheung, L. Pettersson, et al. OpenAI Gym, 2016. eprint: arXiv:1606.01540.
- [9] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 4:237–285, May 1996. DOI: 10.1613/jair.301.
- [10] X. Zhu. Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison*, 2(3):4, 2006.
- [11] H. Mao, M. Alizadeh, I. Menache, and S. Kandula. Resource management with deep reinforcement learning. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks - HotNets '16*. ACM Press, 2016. DOI: 10.1145/3005745.3005750.

- [12] I. Arel, C. Liu, T. Urbanik, and A. Kohls. Reinforcement learning-based multi-agent system for network traffic signal control. *IET Intelligent Transport Systems*, 4(2):128, 2010. DOI: 10.1049/iet-its.2009.0070.
- [13] G. Brockman and I. Sutskever. Introducing OpenAI, Dec. 2015. URL: <https://openai.com/blog/introducing-openai/>.
- [14] OpenAI. GPT-4 Technical Report, 2023. DOI: 10.48550/ARXIV.2303.08774.
- [15] A. Ramesh, M. Pavlov, G. Goh, et al. Zero-shot text-to-image generation, 2021. DOI: 10.48550/ARXIV.2102.12092.
- [16] OpenAI. SORA: creating video from text. Webpage: <https://openai.com/sora>, Feb. 2024. Accessed on-line: February 18, 2024.
- [17] PERMON. PermonSVM, 2023. URL: <http://github.com/permon/permonsvm>.
- [18] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Identifying suspicious urls: an application of large-scale online learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09. ACM, June 2009. DOI: 10.1145/1553374.1553462.
- [19] S. Balay, S. Abhyankar, M. Adams, et al. PETSc/TAO Users Manual. Technical report ANL-21/39 - Revision 3.20, Argonne National Laboratory, 2023. DOI: 10.2172/1968587.
- [20] P. Visualization. Branca (spinoff from Folium). <https://python-visualization.github.io/branca/>, 2024. Accessed on-line: February 18, 2024.
- [21] E. Rouault, F. Warmerdam, K. Schwehr, et al. Gdal, 2024. DOI: 10.5281/ZENODO.5884351.
- [22] N. Gorelick, M. Hancher, M. Dixon, et al. Google earth engine: planetary-scale geospatial analysis for everyone. *Remote Sensing of Environment*, 202:18–27, Dec. 2017. ISSN: 0034-4257. DOI: 10.1016/j.rse.2017.06.031.
- [23] Filipe, F. Anema, R. Story, et al. Python-visualization/folium: v0.15.1, 2023. DOI: 10.5281/ZENODO.594439.
- [24] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.
- [25] T. pandas development team. Pandas-dev/pandas: pandas, 2024. DOI: 10.5281/ZENODO.3509134.
- [26] R. D. Team. *RAPIDS: Libraries for End to End GPU Data Science*. 2023. URL: <https://rapids.ai>.
- [27] F. Pedregosa, G. Varoquaux, A. Gramfort, et al. Scikit-learn: machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

-
- [28] P. Virtanen, R. Gommers, T. E. Oliphant, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature Methods*, 17(3):261–272, Feb. 2020. ISSN: 1548-7105. DOI: 10.1038/s41592-019-0686-2.
- [29] J. Brownlee. Ordinal and one-hot encodings for categorical data. Online, Aug. 2020. URL: <https://machinelearningmastery.com/one-hot-encoding-for-categorical-data>. Accessed on-line: October 31, 2023.
- [30] C. M. Bishop. *Pattern recognition and machine learning*. Information Science and Statistics. Springer Science+Business Media, LLC, New York, NY, 2019. 758 pages. ISBN: 0-387-31073-8.
- [31] K. Fukushima. Visual feature extraction by a multilayered network of analog threshold elements. *IEEE Transactions on Systems Science and Cybernetics*, 5(4):322–333, 1969. DOI: 10.1109/tssc.1969.300225.
- [32] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer New York, 2000. DOI: 10.1007/978-1-4757-3264-1.
- [33] A. Ng. CS229 Lecture notes - Supervised learning. Technical report, 2012. URL: https://cs229.stanford.edu/main%5C_notes.pdf.
- [34] B. Schölkopf, A. Smola, K.-R. Müller, C. Burges, and V. Vapnik. Support vector methods in learning and feature extraction. *Ninth Australian Congress on Neural Networks (to appear)*, 1998.
- [35] C. J. Burges. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998. DOI: 10.1023/a:1009715923555.
- [36] Jayadeva. Learning a hyperplane classifier by minimizing an exact bound on the VC dimension1. *Neurocomputing*, 149:683–689, Feb. 2015. DOI: 10.1016/j.neucom.2014.07.062.
- [37] Z. Dostál. *Optimal Quadratic Programming Algorithms, with Applications to Variational Inequalities*, volume 23. SOIA, Springer, New York, US, 2009.
- [38] Larhmam. Support vector machine. https://en.wikipedia.org/wiki/Support_vector_machine, Feb. 2024. Accessed on-line: February 23, 2024.
- [39] C.-J. Lin. Formulations of support vector machines: a note from an optimization point of view. *Neural Computation*, 13(2):307–317, Feb. 2001. ISSN: 1530-888X. DOI: 10.1162/089976601300014547.
- [40] PERMON. PermonQP, 2018. URL: <https://github.com/permon/permon>.
- [41] V. Hapla. Massively parallel quadratic programming solvers with applications in mechanics. *Doctoral thesis*, 2016. URL: <http://hdl.handle.net/10084/112271>.
-

- [42] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. ISBN: 0521833787.
- [43] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 1995.
- [44] M. Rychetsky. *Algorithms and Architectures for Machine Learning Based on Regularized Neural Networks and Support Vector Approaches (Berichte Aus Der Informatik)*. Shaker Verlag GmbH, Germany, 2001. ISBN: 3826596404.
- [45] V. Cherkassky and F. M. Mulier. *Learning from Data: Concepts, Theory, and Methods*. Wiley-IEEE Press, 2007. ISBN: 0471681822.
- [46] M. Pecha and D. Horák. Analyzing l1-loss and l2-loss support vector machines implemented in PERMON toolbox. In *Lecture Notes in Electrical Engineering*, pages 13–23. Springer International Publishing, Apr. 2019. DOI: 10.1007/978-3-030-14907-9_2.
- [47] C.-P. Lee and C.-J. Lin. A study on l2-loss (squared hinge-loss) multiclass SVM. *Neural Computation*, 25(5):1302–1323, May 2013. DOI: 10.1162/neco_a_00434.
- [48] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *Proceedings of the 25th international conference on Machine learning - ICML '08*. ACM Press, 2008. DOI: 10.1145/1390156.1390208.
- [49] S. J. W. Suvrit Sra Sebastian Nowozin, editor. *Optimization for Machine Learning*. MIT Press Ltd, Sept. 2011. 512 pages. ISBN: 0262537761.
- [50] I. Steinwart. Support vector machines are universally consistent. *Journal of Complexity*, 18(3):768–791, Sept. 2002. DOI: 10.1006/jcom.2002.0642.
- [51] Walber and M. us (username). Diagram representation of precision and recall. <https://commons.wikimedia.org/wiki/File:Precisionrecall.svg#filehistory>, 2021. Accessed on-line: November 17, 2023.
- [52] Marek Pecha, Bohdan Rieznikov, Vít Wandrol, Jana Rušajová, Zachary Langford, Richard Mills, Ivo Wandrol. Natural hazard project for analysing wildfires and earthquakes using ML techniques, 2024. URL: <http://github.com/natural-hazards>.
- [53] D. Shah. Intersection over union (iou): definition, calculation, code. <https://www.v7labs.com/blog/intersection-over-union-guide>, May 2023. Accessed on-line: October 18, 2023.
- [54] B. Ergen. Building a custom grid search for your custom model. <https://www.linkedin.com/pulse/building-custom-grid-search-your-model-bünyamin-ergen/>, Apr. 2023. Accessed on-line: October 18, 2023.
- [55] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(2):281–305, 2012.

-
- [56] P. I. Frazier. A tutorial on bayesian optimization, 2018. eprint: [arXiv:1807.02811](https://arxiv.org/abs/1807.02811).
- [57] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In pages 1137–1143. Morgan Kaufmann, 1995.
- [58] L. Breiman and P. Spector. Submodel selection and evaluation in regression. the x-random case. *International Statistical Review / Revue Internationale de Statistique*, 60(3):291, Dec. 1992. DOI: [10.2307/1403680](https://doi.org/10.2307/1403680).
- [59] S. Varma and R. Simon. *BMC Bioinformatics*, 7(1):91, 2006. DOI: [10.1186/1471-2105-7-91](https://doi.org/10.1186/1471-2105-7-91).
- [60] J. Kružík, D. Horák, M. Čermák, L. Pospíšil, and M. Pecha. Active set expansion strategies in mprgp algorithm. *Advances in Engineering Software*, 149:102895, Nov. 2020. ISSN: 0965-9978. DOI: [10.1016/j.advengsoft.2020.102895](https://doi.org/10.1016/j.advengsoft.2020.102895).
- [61] Z. Dostál. Scalable algorithms for contact problems, New York, NY, 2017.
- [62] V. Dorňák, M. Čermák, M. Pecha, and L. Pospíšil. Sift feature extraction applied in svm classification. In *AIP Conference Proceedings*. AIP Publishing, 2022. DOI: [10.1063/5.0081373](https://doi.org/10.1063/5.0081373).
- [63] V. Dorňák, L. Pospíšil, M. Pecha, and M. Čermák. Image features classification using bayesian model. In *INTERNATIONAL CONFERENCE OF NUMERICAL ANALYSIS AND APPLIED MATHEMATICS ICNAAM 2021*. AIP Publishing, 2023. DOI: [10.1063/5.0162064](https://doi.org/10.1063/5.0162064).
- [64] D. Horak, J. Kruzik, M. Pecha, V. Hapla, and M. Cermak. Choosing optimal expansion step-length of mprgp algorithm. In *Civil-Comp Proceedings*, PARENG 2019. Civil-Comp Press. DOI: [10.4203/ccp.112.18](https://doi.org/10.4203/ccp.112.18).
- [65] Libsvm data: classification (binary class). URL: <https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html>.
- [66] M. Pecha, Z. Langford, D. Horák, and R. Tran Mills. Wildfires identification: semantic segmentation using support vector machine classifier. In *Programs and Algorithms of Numerical Mathematics 21*, PANM 21. Institute of Mathematics, Czech Academy of Sciences, Apr. 2023. DOI: [10.21136/panm.2022.16](https://doi.org/10.21136/panm.2022.16).
- [67] S. Balay, W. Gropp, L. C. McInnes, and B. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.
-

- [68] R. T. Mills, M. F. Adams, S. Balay, J. Brown, and A. Dener. Toward performance-portable PETSc for GPU-based exascale systems. *Parallel Computing*, 108:102831, 2021. ISSN: 0167-8191. DOI: <https://doi.org/10.1016/j.parco.2021.102831>. URL: <https://www.sciencedirect.com/science/article/pii/S016781912100079X>.
- [69] J. Nickolls, I. Buck, M. Garland, and K. Skadron. Scalable parallel programming with cuda: is cuda the parallel programming model that application developers have been waiting for? *Queue*, 6(2):40–53, Mar. 2008. ISSN: 1542-7749. DOI: 10.1145/1365490.1365500.
- [70] I. Advanced Micro Devices. Amd rocmTMdocumentation. <https://rocm.docs.amd.com/en/latest/>, 2023. Accessed on-line: November 17, 2023.
- [71] J. E. Stone, D. Gohara, and G. Shi. Opencl: a parallel programming standard for heterogeneous computing systems. *Computing in Science & Engineering*, 12(3):66–73, May 2010. ISSN: 1521-9615. DOI: 10.1109/mcse.2010.69.
- [72] V. Hapla and M. Pecha. HDF5 viewer for AIJ matrices implemented into PETSc. on-line, 2018.
- [73] V. Hapla, M. G. Knepley, M. Afanasiev, et al. Fully parallel mesh i/o using petsc dmpex with an application to waveform modeling. *SIAM Journal on Scientific Computing*, 43(2):C127–C153, Jan. 2021. ISSN: 1095-7197. DOI: 10.1137/20m1332748.
- [74] T. N. Aeronautics and S. A. (nasa.gov)). Wildfires ranging across the width of alaska captured by terra satellite. (WebArchive) https://www.nasa.gov/vision/earth/lookingatearth/alaskan_fire.html, Nov. 2005. Accessed on-line: December 23, 2023.
- [75] U. F. S. G. Survey. Usda forest service/u.s. geological survey. <https://mtbs.gov>, Oct. 2023. Accessed on-line: December 23, 2022.
- [76] P. Chang, S. Zhang, G. Danabasoglu, et al. An unprecedented set of high-resolution earth system simulations for understanding multiscale interactions in climate variability and change. *Journal of Advances in Modeling Earth Systems*, 12(12), Dec. 2020. ISSN: 1942-2466. DOI: 10.1029/2020ms002298.
- [77] S. Farrell, M. Emani, J. Balma, et al. Mlperf hpc: a holistic benchmark suite for scientific machine learning on hpc systems. In *2021 IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments (MLHPC)*. IEEE, Nov. 2021. DOI: 10.1109/mlhpc54614.2021.00009.
- [78] Martín Abadi, Ashish Agarwal, Paul Barham, et al. TensorFlow: large-scale machine learning on heterogeneous systems, 2015. URL: <https://www.tensorflow.org/>. Software available from tensorflow.org.

-
- [79] A. Paszke, S. Gross, S. Chintala, et al. Automatic differentiation in pytorch, 2017.
- [80] M. Hasal, M. Pecha, J. Nowaková, et al. Retinal vessel segmentation by u-net with VGG-16 backbone on patched images with smooth blending. In *Advances in Intelligent Networking and Collaborative Systems*, pages 465–474. Springer Nature Switzerland, 2023. DOI: 10.1007/978-3-031-40971-4_44.
- [81] D. Schlee, P. H. A. Sneath, R. R. Sokal, and W. H. Freeman. Numerical taxonomy. the principles and practice of numerical classification. *Systematic Zoology*, 24(2):263, Jan. 1975. DOI: 10.2307/2412767.
- [82] S.-E. Qian. Vector Quantization Data Compression. In *Optical Satellite Data Compression and Implementation*. Society of Photo-Optical Instrumentation Engineers. DOI: 10.1117/3.1002297.ch4.
- [83] H. Steinhaus. Sur la division des corp materiels en parties. *Bull. Acad. Polon. Sci*, 1(804):801, 1956.
- [84] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, Mar. 1982. DOI: 10.1109/tit.1982.1056489.
- [85] S. Ying, G. Xu, C. Li, and Z. Mao. Point cluster analysis using a 3d voronoi diagram with applications in point cloud segmentation. *ISPRS International Journal of Geo-Information*, 4(3):1480–1499, Aug. 2015. ISSN: 2220-9964. DOI: 10.3390/ijgi4031480.
- [86] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.
- [87] J. MacQueen. The classification problem. *Western Management Science Institute Working Paper*, 5, 1962.
- [88] J. Marschak. Towards an economic theory of organization and information. *Decision processes*, 3(1):187–220, 1954.
- [89] J. Marschak. Remarks on the Economics of Information. Technical report, Cowles Foundation for Research in Economics, Yale University, 1959.
- [90] E. W. Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics*, 21:768–769, 1965.
- [91] T. Calinski and J. Harabasz. A dendrite method for cluster analysis. *Communications in Statistics - Theory and Methods*, 3(1):1–27, 1974. DOI: 10.1080/03610927408827101.
- [92] D. L. Davies and D. W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227, Apr. 1979. DOI: 10.1109/tpami.1979.4766909.
-

- [93] G. Marsaglia and A. Zaman. A New Class of Random Number Generators. *The Annals of Applied Probability*, 1(3):462–480, Aug. 1991. DOI: 10.1214/aoap/1177005878.
- [94] M. Matsumoto and T. Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30, Jan. 1998. DOI: 10.1145/272991.272995.
- [95] S. Z. Selim and M. A. Ismail. K-means-type algorithms: a generalized convergence theorem and characterization of local optimality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(1):81–87, Jan. 1984. DOI: 10.1109/tpami.1984.4767478.
- [96] D. Arthur and S. Vassilvitskii. How slow is the k-means method? In *Proceedings of the twenty-second annual symposium on Computational geometry - SCG '06*. ACM Press, 2006. DOI: 10.1145/1137856.1137880.
- [97] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley John + Sons, Oct. 2000. ISBN: 0471056693.
- [98] D. Arthur and S. Vassilvitskii. K-means++: the advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007. ISBN: 978-0-898716-24-5.
- [99] L. Kaufman and P. J. Rousseeuw. *Partitioning Around Medoids (Program PAM)*. Wiley, Mar. 1990, pages 68–125. DOI: 10.1002/9780470316801.ch2.
- [100] E. Weiszfeld. Sur le point pour lequel la somme des distances de n points donnés est minimum. *Tohoku Mathematical Journal, First Series*, 43:355–386, 1937.
- [101] M. Pecha and P. Skalný. Identifying the ductile fracture in steel materials via Lloyd type algorithms. *Civil-Comp Proceedings*, 111, 2017.
- [102] Boost. Boost C++ Libraries, 2017. URL: <http://www.boost.org/>.
- [103] P. Skalný and M. Pecha. Identification of the ductile fracture area of X70 steel from DWTT broken specimens. In *METAL 2017 - 26th International Conference on Metallurgy and Materials, Conference Proceedings*, volume 2017-January, pages 589–594, 2017.
- [104] D.-T. Lee and B. J. Schachter. Two algorithms for constructing a delaunay triangulation. *International Journal of Computer & Information Sciences*, 9(3):219–242, 1980.
- [105] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, Aug. 2007. DOI: 10.1007/s11222-007-9033-z.

-
- [106] R. J. LeVeque. *Finite difference methods for ordinary and partial differential equations. Steady-state and time-dependent problems*. Society for Industrial and Applied Mathematics, Philadelphia, Pa., 2007. 341 pages. ISBN: 9780898716290. Includes bibliographical references and index.
- [107] D. Shah. Spectral clustering: A comprehensive guide for beginners. <https://www.analyticsvidhya.com/blog/2021/05/what-why-and-how-of-spectral-clustering/>, May 2024. Accessed on-line: February 21, 2024.
- [108] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: analysis and an algorithm. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 849–856. MIT Press, 2002.
- [109] M. Stoer and F. Wagner. A simple min-cut algorithm. *Journal of the ACM (JACM)*, 44(4):585–591, July 1997. DOI: 10.1145/263867.263872.
- [110] D. R. Karger and C. Stein. A new approach to the minimum cut problem. *Journal of the ACM (JACM)*, 43(4):601–640, July 1996. DOI: 10.1145/234533.234534.
- [111] M. Thorup. Minimum k-way cuts via deterministic greedy tree packing. In *Proceedings of the fourtieth annual ACM symposium on Theory of computing - STOC 08*. ACM Press, 2008. DOI: 10.1145/1374376.1374402.
- [112] A. Gupta, E. Lee, and J. Li. Faster exact and approximate algorithms for k-cut. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, Oct. 2018. DOI: 10.1109/focs.2018.00020.
- [113] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1101–1113, 1993. DOI: 10.1109/34.244673.
- [114] L. Hagen and A. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(9):1074–1085, 1992. DOI: 10.1109/43.159993.
- [115] G. H. Golub. *Matrix Computations*. J. Hopkins Uni. Press, 2013. ISBN: 1421407949.
- [116] E. Kokiopoulou, J. Chen, and Y. Saad. Trace optimization and eigenproblems in dimension reduction methods. *Numerical Linear Algebra with Applications*, 18(3):565–602, Sept. 2010. DOI: 10.1002/nla.743.
- [117] D. Verma and M. Meila. A Comparison of Spectral Clustering Algorithms. Technical report, 2003.
- [118] J. Shi and J. Malik. Normalized cuts and image segmentation. In *Proceedings IEEE Conference Computer Vision and Pattern Recognition*, pages 731–737, 1997.

- [119] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000. DOI: 10.1109/34.868688.
- [120] M. Meila and J. Shi. Learning segmentation by random walks. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 873–879. MIT Press, 2001.
- [121] M. Pecha, M. Čermák, V. Hapla, D. Horák, and J. Tomčala. Advanced approach of material region detections on fibre-reinforced concrete ct-scans. *Advances in Electrical and Electronic Engineering*, 15(2):223–229, 2017. DOI: 10.15598/aeet.v15i2.2319.
- [122] M. Pecha. General technique for estimating number of groups for spectral clustering, Jan. 2021. DOI: 10.36227/techrxiv.13553705.v1.
- [123] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, Mar. 1978. DOI: 10.1214/aos/1176344136.
- [124] H. Akaike. Information theory and an extension of the maximum likelihood principle. In *Springer Series in Statistics*, pages 199–213. Springer New York, 1998. DOI: 10.1007/978-1-4612-1694-0_15.
- [125] P. Bruneau and B. Otjacques. A probabilistic model selection criterion for spectral clustering. *Intelligent Data Analysis*, 22(5):1059–1077, Sept. 2018. ISSN: 1088467X, 15714128. DOI: 10.3233/IDA-173607.
- [126] R. B. Cattell. The scree test for the number of factors. *Multivariate Behavioral Research*, 1(2):245–276, Apr. 1966. DOI: 10.1207/s15327906mbr0102_10.
- [127] P. Bruneau, O. Parisot, and B. Otjacques. A heuristic for the automatic parametrization of the spectral clustering algorithm. In *2014 22nd International Conference on Pattern Recognition*, pages 1313–1318, 2014. DOI: 10.1109/ICPR.2014.235.
- [128] M. S. Bartlett. Properties of sufficiency and statistical tests. *Proceedings of the Royal Society of London. Series A - Mathematical and Physical Sciences*, 160(901):268–282, May 1937. DOI: 10.1098/rspa.1937.0109.
- [129] R. Muirhead. Latent roots and matrix variates: a review of some asymptotic results. *The Annals of Statistics*, 6(1):5–33, Jan. 1978. DOI: 10.1214/aos/1176344063.
- [130] D. N. Lawley. Tests of significance for the latent roots of covariance and correlation matrices. *Biometrika*, 43(1/2):128, June 1956. DOI: 10.2307/2333586.
- [131] A. T. JAMES. Tests of equality of latent roots of the covariance matrix. *Multivariate analysis-II*:205–218, 1969.

-
- [132] C. Zhang and D. Florencio. Analyzing the optimality of predictive transform coding using graph-based models. *IEEE Signal Processing Letters*, 20(1):106–109, Jan. 2013. DOI: 10.1109/lsp.2012.2230165.
- [133] D. Mumford and J. Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on Pure and Applied Mathematics*, 42(5):577–685, July 1989. DOI: 10.1002/cpa.3160420503.
- [134] M. Pecha and M. Čermák. Segmentations for piecewise smooth pictures in PERMON. In *AIP Conference Proceedings*, volume 1738, 2016. DOI: 10.1063/1.4952143.
- [135] M. Pecha, P. Jirutková, and M. Cermak. Fundamental improvements of the piecewise semi-smooth Laplace-Beltrami operator numerical stability. In *AIP Conference Proceedings*, volume 1863, 2017. DOI: 10.1063/1.4992519.
- [136] R. B. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK users’ guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*. SIAM, 1998.
- [137] V. Hernandez, J. E. Roman, and V. Vidal. SLEPc: a scalable and flexible toolkit for the solution of eigenvalue problems. *ACM Trans. Math. Software*, 31(3):351–362, 2005.
- [138] M. Pecha. *Image segmentation techniques in the HPC environment and their applications*. Master’s thesis, 2016.
- [139] N. Teimouri, M. Dyrmann, and R. N. Jørgensen. A novel spatio-temporal fcn-lstm network for recognizing various crop types using multi-temporal radar images. *Remote Sensing*, 11(8):990, Apr. 2019. ISSN: 2072-4292. DOI: 10.3390/rs11080990.
- [140] F. Xu, H. Ma, J. Sun, et al. Lstm multi-modal unet for brain tumor segmentation. In *2019 IEEE 4th International Conference on Image, Vision and Computing (ICIVC)*. IEEE, July 2019. DOI: 10.1109/icivc47709.2019.8981027.
- [141] L. Yin, L. Wang, T. Li, et al. U-net-lstm: time series-enhanced lake boundary prediction model. *Land*, 12(10):1859, Sept. 2023. ISSN: 2073-445X. DOI: 10.3390/land12101859.
- [142] A. H. Victoria and G. Maragatham. Automatic tuning of hyperparameters using bayesian optimization. *Evolving Systems*, 12(1):217–223, May 2020. ISSN: 1868-6486. DOI: 10.1007/s12530-020-09345-2.
-

Appendix A

List of author's publications

Related to the topic of the thesis

1. **M. Pecha**, V. Hapla, D. Horák, and M. Čermák. Notes on the preliminary results of a linear two-class classifier in the PERMON toolbox. AIP Conference Proceedings, volume 1978, 2018. (conference paper indexed on Scopus and WoS)
<https://doi.org/10.1063/1.5043965>.
2. J. Kružík, **M. Pecha**, V. Hapla, D. Horák, and M. Čermák. Investigating convergence of linear SVM implemented in PermonSVM employing MPRGP algorithm. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 11087 LNCS:115–129, 2018. (conference paper indexed on Scopus and WoS)
https://doi.org/10.1007/978-3-319-97136-0_9.
3. **M. Pecha**, P. Jirutková, and M. Čermák. Fundamental improvements of the piecewise semi-smooth Laplace-Beltrami operator numerical stability. In AIP Conference Proceedings, volume 1863, 2017. (conference paper indexed on Scopus and WoS)
<https://doi.org/10.1063/1.4992519>.
4. P. Skalný and **M. Pecha**. Identification of the ductile fracture area of X70 steel from DWTB broken specimens. In METAL 2017 - 26th International Conference on Metallurgy and Materials, Conference Proceedings, volume 2017-January, pages 589–594, 2017. (conference paper indexed on Scopus and WoS)
5. **M. Pecha**, M. Čermák, V. Hapla, D. Horák, and J. Tomčala. Advanced approach of material region detections on fibre-reinforced concrete ct-scans. Advances in Electrical and Electronic Engineering, 15(2):223–229, 2017. (journal paper, JSC type)
<https://doi.org/10.15598/aeed.v15i2.2319>.

6. **M. Pecha** and M. Čermák. Segmentations for piecewise smooth pictures in PERMON. In AIP Conference Proceedings, volume 1738, 2016. (conference paper indexed on Scopus and WoS)
<https://doi.org/10.1063/1.4952143>.
 7. **M. Pecha** and D. Horák. Analyzing l1-loss and l2-loss support vector machines implemented in PERMON toolbox. In Lecture Notes in Electrical Engineering, pages 13–23. Springer International Publishing, Apr. 2019. (conference paper)
https://doi.org/10.1007/978-3-030-14907-9_2.
 8. M. Hasal, **M. Pecha**, J. Nowaková, D. Hernández-Sosa, V. Snášel, and J. Timkovič. Retinal vessel segmentation by u-net with VGG-16 backbone on patched images with smooth blending. In Advances in Intelligent Networking and Collaborative Systems, pages 465–474. Springer Nature Switzerland, 2023. (conference paper)
https://doi.org/10.1007/978-3-031-40971-4_44.
 9. J. Kružík, D. Horák, M. Čermák, L. Pospíšil, and **M. Pecha**. Active set expansion strategies in MPRGP algorithm. Advances in Engineering Software, 149:102895, Nov. 2020. issn: 0965-9978. (journal paper, IF 4.8)
<https://doi.org/10.1016/j.advengsoft.2020.102895>.
 10. S. Crisci, J. Kružík, **M. Pecha**, and D. Horák. Comparison of active-set and gradient projection-based algorithms for box-constrained quadratic programming. Soft Computing, 24(23):17761–17770, Oct. 2020. ISSN: 1433-7479. (journal paper, IF 4.1)
<https://doi.org/10.1007/s00500-020-05304-w>.
 11. V. Dornák, M. Čermák, **M. Pecha**, and L. Pospíšil. SIFT feature extraction applied in SVM classification. In AIP Conference Proceedings. AIP Publishing, 2022. (conference paper indexed on Scopus)
<https://doi.org/10.1063/5.0081373>.
 12. V. Dornák, L. Pospíšil, **M. Pecha**, and M. Čermák. Image features classification using Bayesian model. IN AIP Conference Proceedings. AIP Publishing 2023. (conference paper indexed on Scopus)
<https://doi.org/10.1063/5.0162064>.
 13. **M. Pecha**. Balancing predictive relevance of ligand biochemical activities. In Uncertainty and Imprecision in Decision Making and Decision Support: New Advances, Challenges, and Perspectives. Springer International Publishing, 2022, pages 338–348. ISBN: 9783030959296. (conference paper indexed on Scopus and WoS)
https://doi.org/10.1007/978-3-030-95929-6_26.
-

-
14. **M. Pecha**, Z. Langford, D. Horák, and R. Tran Mills. Wildfires identification: semantic segmentation using support vector machine classifier. In Programs and Algorithms of Numerical Mathematics 21, PANM 21. Institute of Mathematics, Czech Academy of Sciences, Apr. 2023. (conference paper)
<https://doi.org/10.21136/panm.2022.16>.
 15. D. Horák, J. Kružík, **M. Pecha**, V. Hapla, and M. Čermák. Choosing optimal expansion step-length of MPRGP algorithm. In Civil-Comp Proceedings, PARENG 2019. Civil-Comp Press. (conference paper)
<https://doi.org/10.4203/ccp.112.18>.

Others

1. J. Tomčala, M. Čermák, **M. Pecha**, and D. Horák. The fatigue damage software parallelization. In AIP Conference Proceedings, volume 1978, 2018. (conference paper indexed on Scopus and WoS)
<https://doi.org/10.1063/1.5043966>.
2. J. Tomčala, J. Papuga, D. Horák, V. Hapla, **M. Pecha**, and M. Čermák. Steps to increase practical applicability of PragTic software. Advances in Engineering Software, 2018. (journal paper, IF 4.8)
<https://doi.org/10.1016/j.advengsoft.2018.06.009>.
3. J. Papuga, R. Halama, M. Fusek, J. Rojíček, F. Fojtík, D. Horák, **M. Pecha**, J. Tomčala, M. Čermák, V. Hapla, R. Sojka, and J. Kružík. Efficient lifetime estimation techniques for general multiaxial loading. In AIP Conference Proceedings, volume 1863, 2017. (conference paper indexed on Scopus and WoS)
<https://doi.org/10.1063/1.4992518>.
4. J. Tomčala, **M. Pecha**, and J. Papuga. Parallelization of fatigue damage calculation. In Civil-Comp Proceedings, 111, 2017.
5. D. Horák, V. Hapla, J. Kružík, R. Sojka, M. Čermák, J. Tomčala, **M. Pecha**, and Z. Dostál. A note on massively parallel implementation of FETI for the solution of contact problems. Advances in Electrical and Electronic Engineering, 15(2):230–236, 2017. (journal paper, JSC type)
<https://doi.org/10.15598/aeer.v15i2.2321>.

Preprint

- **M. Pecha**. General technique for estimating number of groups for spectral clustering.
<https://doi.org/10.36227/techrxiv.13553705.v1>.
-

APPENDIX A. LIST OF AUTHOR'S PUBLICATIONS

Appendix B

Projects and grants

- Motivational scholarships of VŠB–TUO (2019)
- Matematické modelování a vývoj algoritmů pro výpočetně náročné inženýrské úlohy V
EN: *Mathematical modelling and algorithm development for computationally intensive engineering problems V*
(SP2019/84)
- HPC-EUROPA3 with the support of the EC Research Innovation Action under the H2020 Programme
(INFRAIA-2016-1-730897)
- Matematické modelování a vývoj algoritmů pro výpočetně náročné inženýrské úlohy IV
EN: *Mathematical modelling and algorithm development for computationally intensive engineering problems IV*
(SP2018/84)
- Support for Science and Research in the Moravia–Silesia Region 2017 (RRC/10/2017)
- Matematické modelování a vývoj algoritmů pro výpočetně náročné inženýrské úlohy III
EN: *Mathematical modelling and algorithm development for computationally intensive engineering problems III*
(SP2017/122)
- PERMON toolbox development III (SP2017/169)
- Matematické modelování a vývoj algoritmů pro výpočetně náročné inženýrské úlohy II
EN: *Mathematical modelling and algorithm development for computationally intensive engineering problems II*
(SP2016/108)

APPENDIX B. PROJECTS AND GRANTS

- PERMON toolbox development II (SP2016/178)
- CZ: *Účinné metody odhadu životnosti pro obecné víceosé namáhání*
EN: *Efficient method to estimating lifetime for general multi-axial loading*
(GA15-18274S)
- ml4py - distributed machine learning tools (dev/stage) I – IV
Projects for computational resources within the Open Access Grant Competition announced by IT4Innovations (principal investigator)
- the Strategy AV21 (VP30 Dynamic Planet Earth)
Principal investigator of a topic related to applying machine learning approaches for automatic detection of seismic events, leading team of 5 people

Appendix C

Other activities during PhD studies

C.1 Supervising students

- Image feature extraction applied in classification techniques

Co-supervising bachelor thesis.

Supervisor: doc. Ing. Martin Čermák, PhD

Student: Vojtěch Dornák

Year of submission: 2019

The dean of the Faculty of Electrical Engineering and Computer Science, VŠB-TUO awarded this thesis.

Available online: <https://dspace.vsb.cz/handle/10084/136146>

- Using of cluster analysis for data segmentation

Supervising the high-school student Petr Stádník in years 2019 – 2020. His work was submitted for Czech Contest for Young Scientist (SOČ) 2020 and it was awarded:

- 1st place in the Moravian-Silesian region contest (Computer Science)
- 14th place in the national contest (Computer Science)

https://www.soc.cz/dokumenty/vysledkove_listiny/2019-20.pdf (pp. 86)

- Modelling biochemical activities of compounds employing supervised learning

Supervising the secondary school student Martin Birošček in years 2018 – 2020. His work was submitted for Czech Contest for Young Scientist (SOČ) 2020 and it was awarded:

- 1st place in the Moravian-Silesian region contest (Mathematics and statistics)

- 8th place in the national contest (Mathematics and statistics)
https://www.soc.cz/dokumenty/vysledkove_listiny/2019-20.pdf (pp. 8)

- **Algorithms used for classifying visual signals using methods for extracting significant features**

Co-supervising diploma thesis.

Supervisor: Ing. Lukáš Pospíšil, PhD

Student: Bc. Vojtěch Dornák

Year of submission: 2021

The dean of the Faculty of Electrical Engineering and Computer Science, VŠB-TUO awarded this thesis.

Available online: <https://dspace.vsb.cz/handle/10084/143861>

- **Analysis of an impact of used matrix format to the efficiency of matrix by vector multiplication**

Co-supervising bachelor thesis.

Supervisor: doc. Ing. David Horák, PhD

Student: Vojtěch Blana

Year of submission: 2023

Available online: <https://dspace.vsb.cz/handle/10084/150394>

- **Classification of Seismic Events Using Recurrent Neural Networks**

Supervising bachelor thesis.

Student: Bohdan Rieznikov

Expected year of submission: 2025

C.2 Internships and short research stays

- 2018

- **The University of Edinburgh (UK, Edinburgh)**

- Hosts: Dr Hall and Prof. Gondzio

- Departments: School of Mathematics and Edinburgh Parallel Computing Centre
(3 months)

- 2019

- **Università degli Studi di Napoli Federico II (IT, Naples)**

Host: Prof. Gerado Toraldo

Department: Dipartimento di Matematica e Applicazioni “Renato Caccioppoli”
(1 month)

– **Bulgarian Academy of Science (BG, Sofia)**

Host: Prof. Svetozar Margenov

Department: Institute for Parallel Processing
(2 weeks)

• **2023**

– **Argonne National Laboratory (USA, Illinois, Lemont)**

Host: Dr Richard Tran Mills

Department: Laboratory for Applied Mathematics, Numerical Software, and Statistics (LANS) at Department of Computing, Environment and Life Sciences (CELS)
(2 weeks)

C.3 Software development

- M. Pecha and PERMON team. PermonSVM, 2024, GitHub repository:
<http://github.com/permon/permonsvm>.
- M. Pecha, B. Rieznikov, V. Wandrol, J. Rušajová, Z. Langford, R. Mills, I. Wandrol. Natural hazard project for analysing wildfires and earthquakes using ML techniques, 2024. It will be released during 2024 on the following GitHub repository:
<http://github.com/natural-hazards>.
- V. Hapla and M. Pecha. HDF5 viewer for AIJ matrices implemented into PETSc, 2018.
- M. Pecha. ParallelClus, a parallelization of R CLUS package, 2019, GitHub repository:
<https://github.com/mpecha/cluster/tree/mpecha/feature-openmp>.
- M. Pecha and J. Papuga. A parallel extension of the PragTic (PermonFatigue). Available on-line on the following page <http://permon.it4i.cz/fatigue.htm>.

C.4 Media

- An article in the Akademik magazine (VŠB-TU Ostrava) where I described our approaches to machine learning implemented in a natural hazard project (written in Czech). Available on-line on the VŠB - TU Ostrava website: <https://www.fei.vsb.cz/cs/o-fakulte/novinky/detail-aktuality/?reportId=45579>
-

APPENDIX C. OTHER ACTIVITIES DURING PHD STUDIES

- An interview in the Universitas magazine. (written in Czech)
<https://www.universitas.cz/osobnosti/11322-strojove-uceni-pomaha-lokalizovat-lesni-pozary-na-aljasce>