KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Flexible software design for stochastic and kernel-based learning in advanced data-driven modelling

Vilens Jumutcs

Promotor:
Prof. dr. ir. Johan A.K. Suykens

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Engineering Science

May 2016

Flexible software design for stochastic and kernel-based learning in advanced data-driven modelling

Vilens JUMUTCS

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Engineering Science

May 2016

Preface

The work presented in this thesis is related to the research carried out during my doctoral studies at the STADIUS research group in the Department of Electrical Engineering. It has been a unique period in my life when personal and professional achievements were going hand in hand and I was paving my own path to the machine learning science.

This was an eternal quest for new ideas, inspiration and better understanding of how we can analyze and learn from data. A famous Greek philosopher Plutarch once said - **"The mind is not a vessel to be filled but a fire to be kindled"**. I think my mind had a perfect fuel to be ignited with the never-ending flow of encouragement and support from my supervisor, colleagues and most importantly - my beloved family.

During the entire course of my doctoral studies I met a lot of people who inspired me, helped me and collaborated with me. Among these precious people, first I would like to acknowledge and thank my promoter Professor Johan A.K. Suykens. He has been the best and the most helpful mentor in my life. I am sincerely thankful to him for these years of fascinating research, great scientific findings and endless discussions in Alma about science, Belgian weather and almost every aspect of life.

I would like to thank my current and former colleagues from the STADIUS research group: Raghvendra, Rocco, Xiaolin, Mauricio, Gervasio, Kim, Phillippe, Ricardo, Siamak, Antoine, Emanuele, Marco, Marko, Lynn, Bertrand, Michael, Oliver, Zahra, Yunlong, Carolina and Carlos. They always have been friendly and open-minded to me. Together we shared a lot of funny and creative moments, spent lovely hours discussing our research and common interests. I have to give my very special thanks to Raghvendra, who has become my closest friend. I will never forget my visit to India and our enlightened dances at the Isckon temple. I would like to thank Rocco, who always was helping me to find the right path in the twilight of the machine learning field. Finally, I am thankful to the administrative and technical staff at STADIUS, in particular to Ida, John, Wim, Elsy, Liesbeth and Maarten, for efficient management of all the bureaucratic and technical issues that I faced along the way.

I would also like to thank all members of the Examination Board for the guidance and patience: prof. dr. Moritz Diehl, prof. dr. ir. Hendrik Blockeel, prof. dr. ir. Panagiotis Patrinos, prof. dr. Barbara Hammer, prof. dr. Yurii Nesterov and the chairman prof. dr. ir. Paul Van Houtte.

I want to share my gratitude for everything I have achieved to my family, my sons and my wife for all their love, care and support.

Thank you a lot! Thank you for helping and guiding me all these years!

Vilens Jumutcs
Leuven, May 2016.

Abstract

Many practical problems and applications of data driven modelling are embedded into our daily lives and support decision making in various business and industry domains. Proliferation of data lakes and exponentially growing data volume is a source of new challenges in the machine learning and data governance fields. This immense amount of data has resulted in massive web-scale sources of information and very large datasets. The latter has led the industry and science to an emerging concept of Big Data. Effective analysis, understanding and learning from such sources of data using flexible software tools are among primary goals of this thesis, related to kernel-based and sparse linear modelling.

Kernel-based methods like Support Vector Machines (SVM) and Least-Squares Support Vector Machines (LSSVM) are among the most popular machine learning techniques for solving complex classification, regression, clustering or correlation analysis tasks. These techniques are capable of handling nonlinear mappings from the input space to the function (solution) space and dealing with non-stationary data. In this thesis we focus on the kernel-based methods for unsupervised and semi-supervised learning. Among unsupervised techniques we study density estimation problems for fault, anomaly and novelty detection. The latter problem is investigated by the Multi-Class Supervised Novelty Detection (SND) method by which we combine ideas of One-Class Support Vector Machines with a novel multi-class classification approach which is tackling discrimination between modelled *i.i.d.* distributions and corresponding densities. Within semi-supervised techniques we have developed an extension of the Kernel Spectral Clustering (KSC) framework tailored towards learning from both labeled and unlabeled data using very similar to the SND approach ideas.

Another contribution of this thesis is related to a study of a stochastic learning paradigm. In this particular setting one deals with randomly drawn subsamples of the entire dataset. With respect to a given subsample, we re-iteratively update our solution in order to achieve in expectation a nearly optimal result. We study a stochastic optimization and learning for classification, regression and clustering problems. Along these lines we develop several extensions of the Pegasos (Primal Estimated sub-GrAdient SOlver for SVM) algorithm and some specific reweighted modifications for dual averaging schemes with primal-dual iterate updates. Within the first approach we modify the Pegasos algorithm by incorporating novel and interesting loss functions, like pinball loss which is robust in the presence of outliers. Another studied modification of the Pegasos algorithm is represented by the usage of the Nyström approximated feature space where we obtain nonlinear decision boundaries. Within the second direction we implement very sparse linear classification and regression models using the regularized dual averaging framework. This framework embodies a notion of different regularization schemes, for instance elastic-net, l_1-regularization etc. As a novel contribution to this framework we propose reweighted l_1- and l_2-regularization schemes.

Currently there are many machine learning software packages available for the end user but the majority of such solutions is merely intended for the use of novice practitioners and cannot be adopted by the out-of-field scientists. One might consider the ultimate necessity for a simple self-explanatory software design and user-friendly usage patterns where sophisticated machine learning methods are wrapped by the out-of-box tuning, cross-validation and evaluation procedures. One of the essential contributions of this thesis is presented by the `SALSA.jl` (Software lab for Advanced machine Learning with Stochastic Algorithms in Julia) software library. It combines kernel-based and sparse linear modelling with stochastic learning methods. It uses advanced software design principles for elaborating scalable, robust and user-friendly black-box modelling library. The latter is one of the major contributions of this thesis.

Beknopte samenvatting

Veel praktische problemen en toepassingen van data-driven modelling zijn ingebed in ons dagelijks leven en helpen om beslissingen te maken met betrekking tot vele bedrijfs- en industrie-domeinen. Het profileren van data en de exponentiëel groeiende hoeveelheid gegevens is een bron van nieuwe uitdagingen in de machine learning en data governance velden. Deze immense hoeveelheid data heeft geresulteerd in enorme web-scale bronnen van informatie en zeer grote datasets. Dit laatste heeft in de industrie en de wetenschap geleid tot een nieuw concept, namelijk Big Data. Effectieve analyse, begrip en leren uit dergelijke gegevensbronnen met behulp van flexibele software tools zijn een aantal van de primaire doelstellingen van deze thesis, gerelateerd aan de kernel-gebaseerde en sparse lineaire modellering.

Kernel-gebaseerde methoden zoals Support Vector Machines (SVM) en Least-Squares Support Vector Machines (LSSVM) behoren tot de meest populaire machine learning technieken voor het oplossen van complexe classificatie, regressie, clustering of correlatieanalyse taken. In deze thesis richten we ons op de kernel-gebaseerde methoden voor unsupervised en semi-supervised leren. Als unsupervised technieken bestuderen we dichtheidschatting problemen voor de fout, anomalie en nieuwheid detectie. Het laatste probleem wordt onderzocht door de Multi-Class Supervised Novelty Detection (SND) methode, waarbij we ideeën van One-Class Support Vector Machines met een nieuwe multi-class classificatie benadering combineren om zo de discriminatie tussen gemodelleerd *i.i.d.* distributies en hun overeenkomstige dichtheden aan te pakken. Voor semi-supervised technieken hebben we een uitbreiding van het Kernel Spectral Clustering (KSC) framework afgestemd op het leren van zowel gelabelde als ongelabelde data ontwikkeld, gebruikmakend van ideeën vergelijkbaar met SND benaderingen.

Een andere bijdrage van deze thesis heeft betrekking tot een studie van een stochastisch leerparadigma. Met betrekking tot een bepaalde deelverzameling, wordt de oplossing op een re-iteratieve wijze ge-update om zo een vrijwel optimaal resultaat te bekomen. We bestuderen een stochastische optimalisatie en het leren van classificatie, regressie en clustering problemen. Langs deze lijnen ontwikkelen we een aantal uitbreidingen van het Pegasos (Primal Estimated sub-gradiënt Solver voor SVM) algoritme en een aantal specifieke herwogen aanpassingen voor dual averaging schemas met iteratieve primal-dual updates. In een eerste benadering passen wij het Pegasos algoritme aan door het opnemen van nieuwe en interessante verliesfuncties, zoals pinball loss welke robuust is in aanwezigheid van uitschieters. Andere bestudeerde aanpassingen aan het Pegasos algoritme worden weergegeven door het gebruik van de Nyström benaderende ruimte waar we lineaire beslissingsgrenzen verkrijgen. In de tweede richting implementeren we zeer schaarse lineaire classificatie en regressiemodellen gebruikmakend van regularized dual averaging framework. Dit kader belichaamt een notie van verschillende regularisatie modellen, bijvoorbeeld elastic-net, l_1-regularisatie enz. Als een nieuwe bijdrage aan dit kader stellen wij herwogen l_1- en l_2-regularisatie modellen voor.

Momenteel zijn er veel machine learning software pakketten beschikbaar voor de eindgebruiker, maar de meerderheid van dergelijke oplossingen is louter bedoeld voor het gebruik van beginnende gebruikers en kan niet door de wetenschappers out-of-field worden aangenomen. Men zou de noodzaak kunnen overwegen voor een eenvoudig vanzelfsprekend software design en gebruiksvriendelijke gebruikspatronen, waar geavanceerde machine learning methoden worden verpakt door procedures voor out-of-box tuning, cross-validatie en de evaluatie. Een van de belangrijkste bijdragen van deze thesis wordt gepresenteerd door de `SALSA.jl` (Software lab voor geavanceerde machine learning met Stochastic Algorithms in Julia) softwarebibliotheek. Het combineert kernel-gebaseerde en ijle lineaire modellen met stochastische leermethoden. Het maakt gebruik van geavanceerde software design principes voor de uitwerking van een schaalbare, robuuste en gebruiksvriendelijke black-box modeling bibliotheek. De laatste is een van de belangrijkste bijdragen van deze thesis.

Contents

Chapter 1

Introduction

1.1 General Background

Our generation is facing a unique challenge in conceiving, processing and learning from the unprecedented volumes of information generated each day. This information is barely structured and demands better machine learning tools and software packages to be processed and understood. In 2014 Google received over 4 million search queries from the 2.4 billion strong global internet population, Facebook users shared nearly 2.5 million pieces of content and email users sent over 200 million messages. And everything happened just in one minute. Petabytes of data are being generated every moment. A prominent source of such information is related to the social media and messengers. In Figure 1.1 one can find the most recent outlook on the numbers and processing throughputs of these media in 2015[1].

Our understanding and ability to learn from such volumes and variety of data sources stumbles upon our mathematical interpretation and analysis. Here comes the machine learning field of science and Big Data concepts [2]. In this thesis we focus on different aspects of machine learning methods applied to a diverse set of challenging problems, such as classification, clustering and novelty detection.

One of the prominent research directions in machine learning is related to kernel-based techniques for probability density estimation problems [103]. In this particular setting one is interested in detecting anomalies or novelties

[1] Refer to `https://www.domo.com/blog/2015/08/data-never-sleeps-3-0`

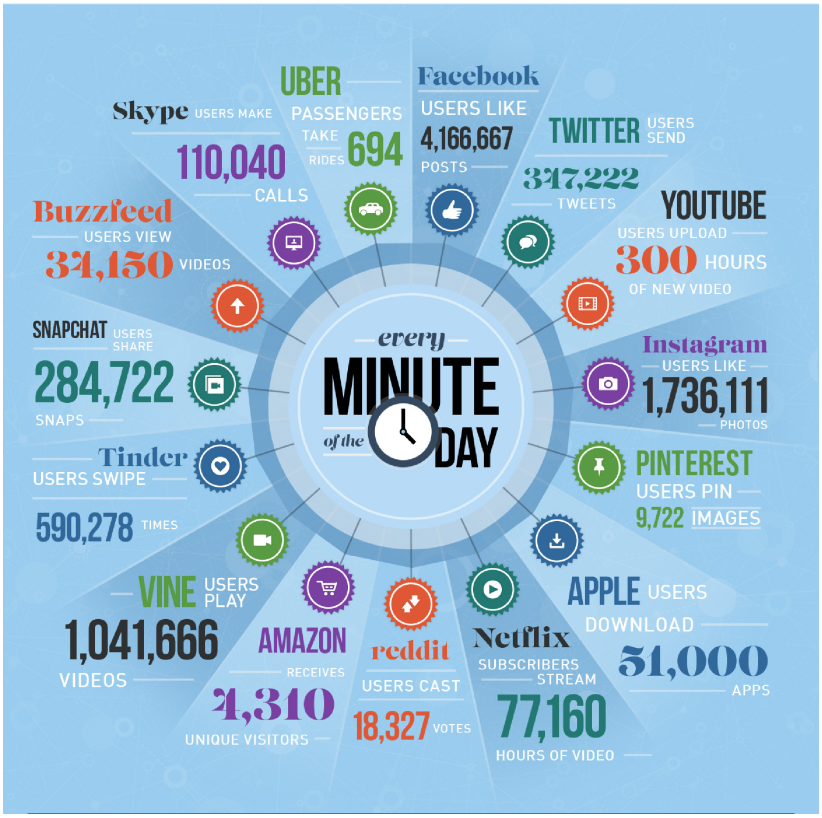


Figure 1.1: Social media and data volumes in 2015.

in data by estimating the support of a high-dimensional distribution which is induced by the kernel trick. The latter problem is interesting from many different perspectives. On the one hand it introduces a non-parametrized setting into the well-known field of statistics and on the other hand it extends ideas of Support Vector Machines [19, 125] beyond classification but tackles it as a typical classification problem where all unlabelled data points are discriminated against a "single-point" class: the origin.

Another interesting direction of machine learning research is allotted to a stochastic learning paradigm for Support Vector Machines [19, 125] and regularized parametric models in the primal. In this particular setting we are interested in learning iteratively from randomly drawn subsamples of the entire dataset. This can be particularly welcomed in the context of Big Data and online learning with large-scale data sources where we cannot observe all data at once and are obliged to delve into the batch (semi-stochastic) or single point[2] updates. A prominent and widely acknowledged example of such an algorithm is Pegasos (Primal Estimated sub-GrAdient SOlver for SVM) [107] which learns an SVM objective in the primal via proximal Stochastic Gradient Descent (SGD) steps.

Sparsity is considered to be another challenging machine learning topic. It can be tackled by applying sparsity inducing norms, such as l_1- or nuclear norm, or within the kernel-based approach by applying Nyström approximation [129] and the Fixed-Size technique [39]. The latter approach allows to reduce computational complexity of the kernel-based methods by selecting a small representative subsample and projecting original dataset onto the approximated and kernel-induced feature space. This method guarantees non-linear mappings and the reduced computational complexity in comparison to the full-scale[3] kernel techniques. Furthermore the Fixed-Size approach can be directly embedded into the stochastic learning paradigm by replacing an original input space with the approximated one [63].

Finally every machine learning scientist devotes a lot of time and attention to the design and implementation details of his or her software. We can allude here to the necessity of writing machine learning libraries as an open source software which implements flexible and extendable design principles. The latter aspect should be perceived as a cornerstone and a necessary ingredient to the successful adoption of machine learning software among out-of-field researchers and practitioners.

Open source machine learning software is gaining an increasing importance as an essential toolkit for many challenging scientific and industrial problems. On the other hand one of the major drawbacks of such a toolkit is the lack of easy-to-use and user friendly software packages. The latter boils down to the incomplete documentation, very narrow command-line interfaces or lack of abstraction which gives the desired level of flexibility in the implementation of proper extensions and plugins. In our thesis we aim at closing a gap between many sophisticated machine learning methods, scalable and robust open source software and out-of-field practitioners which are supposed to utilize this software in their daily routine activities.

Our attention is focused on the scalability of the aforementioned software. In the momentum of the Big Data era we have to take into account every implementation aspect which might be of huge importance in scaling up and out the way we process data. Many modern technological startups and mainstream

[2]We refer to an update based on a single data point $x_i \in \mathcal{X}^d$

[3]We refer to the $\mathcal{O}(dn^2)$ order complexity of computing dense kernel matrices

companies utilize distributed and complex MapReduce eco-systems, like Hadoop[4] [128] or Spark[5] [137], to handle this issue. Scalability and flexibility of the modern machine learning software are among our interests and key stories which are in a closer view of this thesis.

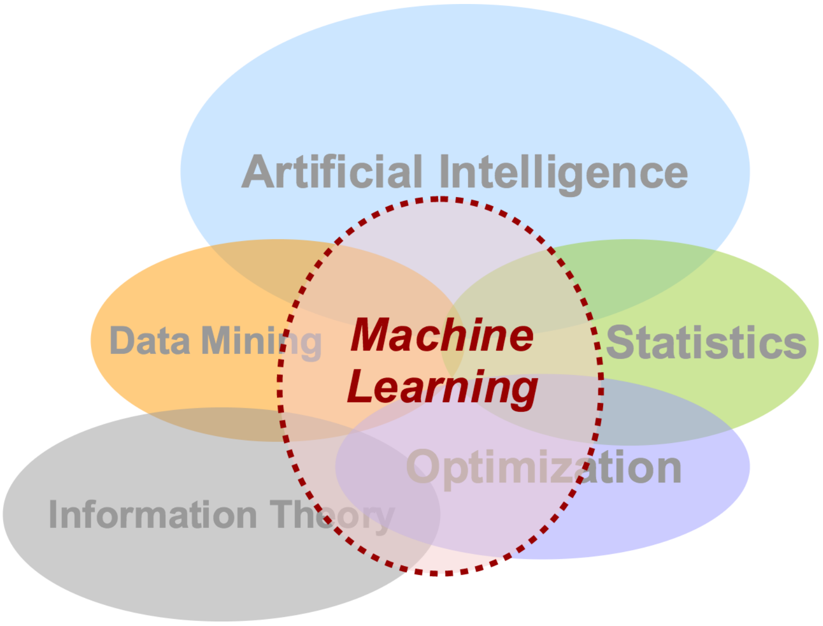


Figure 1.2: Machine Learning domain is overlapping with other related but different scientific domains.

1.2 Machine Learning Challenges in Data-Driven Modelling

Machine Learning can be considered as a core scientific domain in the applied mathematics. It is comprised of different sub-domains and overlaps with other related scientific domains, as shown in Figure 1.2. As we can notice Artificial Intelligence, Data Mining, Information theory as well as Statistics and Optimization share with the Machine Learning domain a considerable amount of findings and methodology which are essential for a successful and seamless

[4]See http://hadoop.apache.org
[5]See http://spark.apache.org

implementation in software and hardware. As an example we can mention the foundation of Statistical Learning Theory [125] by Vladimir Vapnik in the '60s and later developments of the soft-margin Support Vector Machines [19] which can be considered as one of the cornerstones and breakthroughs in the Machine Learning field.

1.2.1 Role of Unsupervised and Semi-Supervised Learning

In the last years kernel-based methods in unsupervised and semi-supervised learning were gaining importance and weight. This fact can be easily explained by the underlying assumptions and envisioned by our understanding of the connection between the distance in some feature space and inner product evaluated in Reproducing Kernel Hilbert Space (RKHS) [7]. Both these terms can be represented by the kernel function [3].

The Machine Learning domain is divided into several more specific sub-areas. We can mention an unsupervised learning problem and more specifically the Kernel Spectral Clustering (KSC) framework [5], Maximum Margin Clustering [136] and other kernel-based clustering approaches. On the other hand among other unsupervised techniques we can emphasize probability density estimation problem related to the novelty or anomaly detection. In the latter problem one seeks for a compact support of the underlying high-dimensional distribution in order to detect outliers in data [103]. Kernel-based approaches can be very efficient in estimating such a support in a non-parametric way. This is a promising research direction because of numerous applications and flexibility of the kernel expansion and RKHS framework [3] in modelling any kind of a function.

In retrospect, semi-supervised learning has received an unprecedented attention due to very limited and scarce labelling information available for learning a model. Nowadays petabytes of information are being generated every moment and only a few bytes are being correctly labelled by the human operator. Therefore kernel-based techniques utilizing both labelled and unlabelled samples are very important for obtaining a correct classification or clustering model. Here we can mention a semi-supervised extension of the KSC framework [87] and well-acknowledged Laplacian Support Vector Machines [14, 88] based on manifold regularization.

In Figure 1.3 we can see a diagram reflecting only a few out of many possible choices of kernel methods. We emphasize that kernel-based learning can be structured into several sub-domains: unsupervised, semi-supervised, supervised

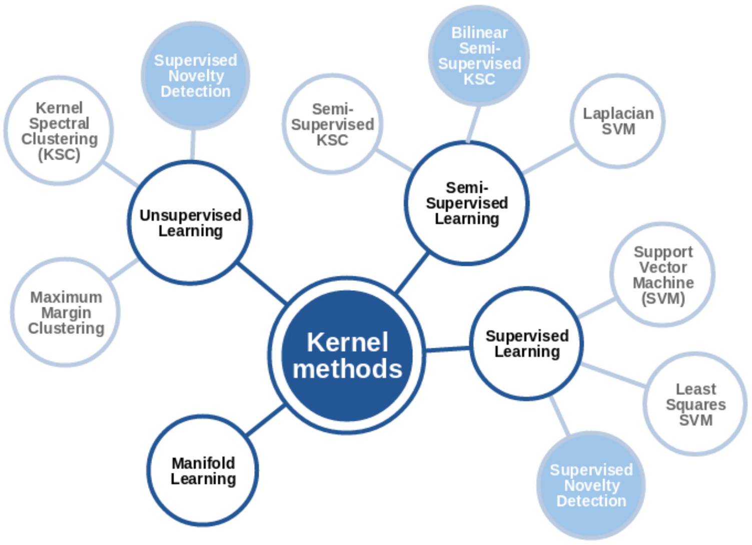

Kernel methods
Unsupervised Learning
Kernel Spectral Clustering (KSC)
Supervised Novelty Detection
Maximum Margin Clustering
Manifold Learning
Semi-Supervised Learning
Semi-Supervised KSC
Bilinear Semi-Supervised KSC
Laplacian SVM
Supervised Learning
Support Vector Machine (SVM)
Least Squares SVM
Supervised Novelty Detection

Figure 1.3: Kernel-based methods in machine learning. With the filled light blue circles we emphasize our contribution to the field.

and manifold learning[6]. We do highlight with the filled light blue circles our contribution described in Section 1.4. As we can notice from this Figure some of our contributions such as Supervised Novelty Detection [65] can be classified to several categories.

1.2.2 Role of Stochastic Learning for Big Data

Stochastic Learning can be considered as another promising direction in the Machine Learning domain. In particular we are interested in stochastic optimization when the number of data points (or the sample size) is too large and both computation- and memory-wise kernel-based techniques and other approaches become infeasible.

One of the oldest and most recognizable approaches in stochastic optimization is represented by Stochastic Gradient Descent (SGD) [20]. In Figure 1.4[7] we present a simplified outlook on how the SGD scheme works in case of unconstrained and convex optimization objective. As we can notice SGD

[6]typically used in the dimensionality reduction context
[7]Refer to `http://www.holehouse.org/mlclass/17_Large_Scale_Machine_Learning.html`.

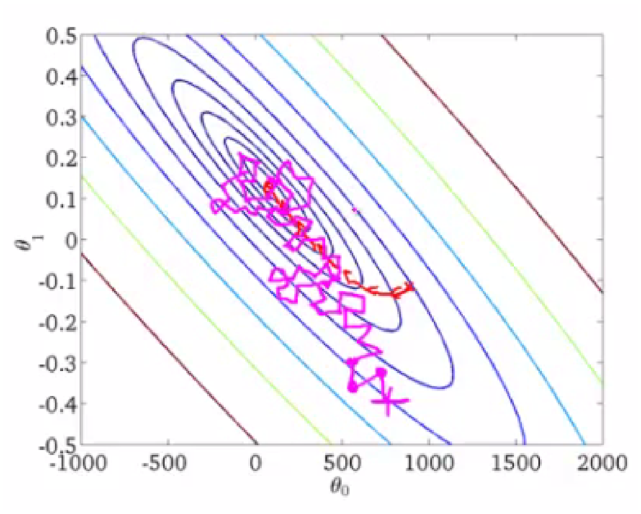


Figure 1.4: With Stochastic Gradient Descent (SGD) every descent step (given in pink) is much faster but every iteration is fitting only a single observation. In this figure we present a contour plot of a convex optimization objective evaluated in 2D domain with respect to arguments θ_0, θ_1. With red lines we depict a descent step evaluated with the full gradient information.

performs a lot of very simple descent steps. Each step is based on a single observation (data point). Therefore the total number of iterations is larger but computationally the overall computational cost is smaller. In stochastic optimization the convergence rates do not depend on the total number of data points. For instance we can reach an ϵ-optimal solution within $T \approx \frac{1}{\epsilon}$ iterations in case of the strongly convex optimization objective and we need $T \approx \frac{1}{\sqrt{\epsilon}}$ iterations in the general convex and non-smooth case.

Challenges of the Big Data world are around us. Stochastic optimization and learning is perceived as a cornerstone of the modern Machine Learning era, especially in such domains as Empirical Risk Minimization and Deep Learning. Stochastic Gradient Descent constitutes an essential component of learning schemes and algorithms in many practical scientific and industrial problems. Learning of interconnection weights in deep convolutional networks, learning from the streaming data, learning from images *etc.* Many of the aforementioned problems are difficult to handle without a stochastic learning setting.

From the very beginning the flexibility of the Support Vector Machines framework was related to the kernel trick and dual quadratic optimization objective. Indeed kernel-based techniques are flexible enough to approximate any kind of a function in RKHS. This motivation can be extended to the stochastic learning setting with the aforementioned SVM optimization objective in primal as well. The key ingredient to learning a non-linear (in the input space) solution via stochastic optimization is related to the Nyström method [129] and approximated feature maps.

1.2.3 Role of Sparsity in Parametric Models

For many years sparsity was tightly related to compressed sensing [43, 48] in the signal processing field. Nowadays we can easily stumble upon sparsity inducing regularization in many machine learning challenges. For instance learning from graphs and biological microarrays, understanding of protein-to-protein networks. All these problems involve sparse data and are believed to retain their solution or signal in a sparse basis.

The most well-studied example of sparsity inducing regularization is Least Absolute Shrinkage and Selection Operator or LASSO [53] which promotes a sparse statistical model with only a small number of non-zero parameters or weights; therefore, it is much easier to estimate and interpret such a model than a dense one. This particular operator represents the l_1-type of a penalty, *i.e.* sum of absolute values or $\|\cdot\|_1$ norm. In case of parametric models in Regularized Empirical Risk Minimization this implies finding an optimal trade-off between an induced sparsity in the recovered signal (solution) and an observed empirical loss (risk). The latter can be represented by the average mismatch between the predicted $\hat{y}$ outputs of the aforementioned statistical model and the true measured signal y.

Another interesting example of sparsity inducing regularization is referred to the l_0-type of a penalty or $\|\cdot\|_0$ pseudo-norm. This particular type of regularization can be approached through the reweighted l_1-norm induced optimization schemes [67, 70]. This regularization promotes the sparsest possible solution. To give a quantitative example please refer to Figure 1.5 where we present an evaluation of occurrence frequencies for different features of the Spambase UCI [49] dataset when using l_1 and the reweighted l_1 regularization.

Finally sparsity is important because of very evident computation- and memory-wise issues. Imagine a bag of words or n-gram model capturing some specific attributes or features of the text corpus. These models can span millions or even billions of entries but not all of them might be inherent to the presented corpus. If we evaluated a dense model in such a setting[8] we would spend hours

[8]For instance by assigning a diminishing weight to each element of the model.

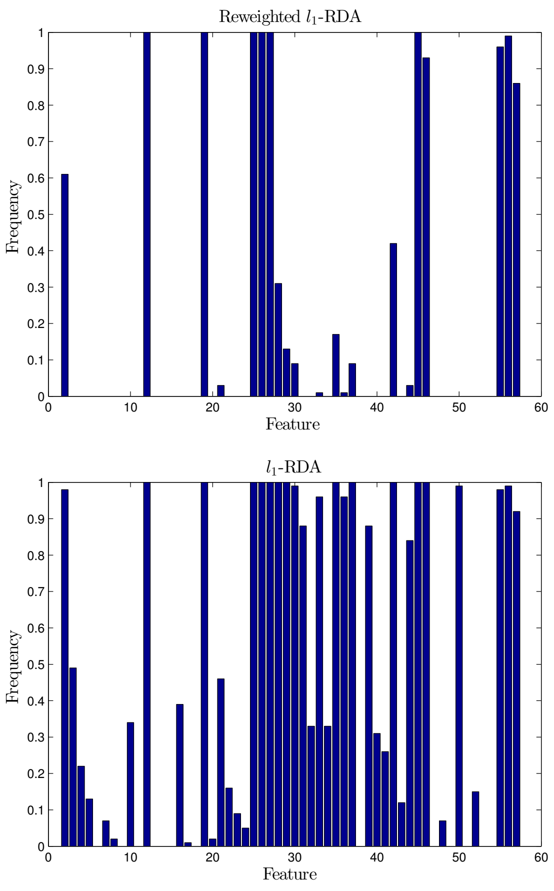
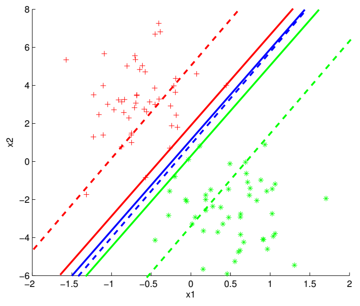


Figure 1.5: Frequency of being non-zero for the features of the Spambase dataset. In the top subfigure we present the results for the reweighted l_1-norm induced approach [67], while the bottom subfigure corresponds to the l_1-norm induced regularization [134].

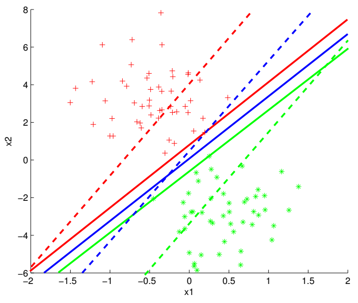
for a single full scan through the entire dataset.

1.2.4 Role of Different Loss Functions

Loss functions are considered as a cornerstone in the Regularized Empirical Risk Minimization [125] framework and the analysis of trade-offs between attained model complexities and observed (empirical) errors. Among the most popular loss functions for classification we can distinguish hinge loss and logistic loss [101]. The first one is heavily utilized for learning linear SVMs while the latter one is suitable for interpreting a classification decision from the probabilistic perspective, *i.e.* how likely is the obtained solution. Furthermore other loss functions are interesting from an emerging prospect of robustness. A pinball loss was well-known before in quantile regression [115] but recently rediscovered in the classification and SVM setting [58] as a promising candidate solution to handle outliers in data. This particular advantage can be explained by Figure 1.6 where we obtain different SVM decision boundaries when using hinge and pinball loss respectively. In Figure 1.6b we can notice that pinball loss is much less sensitive in the presence of outliers and preserves a good classification boundary. We can reason that other well-suited loss functions



(a)



(b)

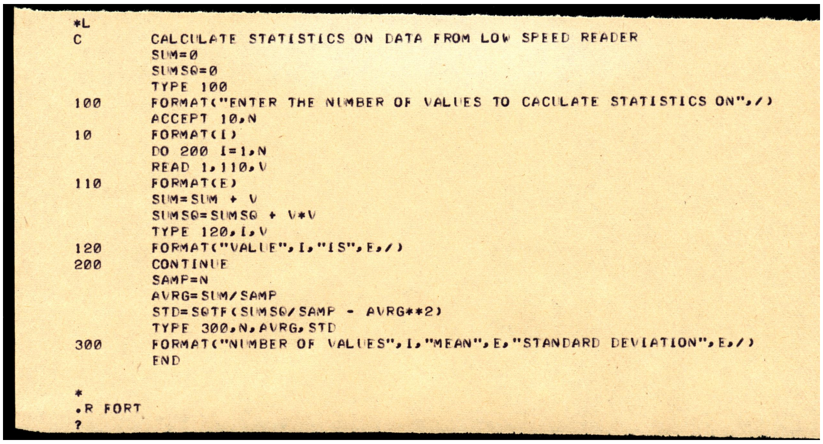
Figure 1.6: Points in two classes are marked by red crosses and green stars. The data in (a) and (b) come from the same distribution but the results of hinge loss SVM (the solid lines) differ significantly. By contrast, the results of pinball loss SVM (the dashed lines) are more robust to the presence of outliers and possible re-sampling.

might be successful in other more specific machine learning subdomains and provide a justified measure of empirical risk.

1.2.5 Flexible Software Design in Machine Learning

History of the scientific software dates back to the early days of computing. The language `Fortran` was developed at IBM in the mid 1950s, and became the first widely used high-level general purpose and scientific programming language. In

Figure 1.7[9] we present a small snippet of code in `Fortran` which is calculating statistics on data.



```
*L
C        CALCULATE STATISTICS ON DATA FROM LOW SPEED READER
         SUM=0
         SUMSQ=0
         TYPE 100
100      FORMAT("ENTER THE NUMBER OF VALUES TO CACULATE STATISTICS ON",/)
         ACCEPT 10,N
10       FORMAT(I)
         DO 200 I=1,N
         READ 1,110,V
110      FORMAT(E)
         SUM=SUM + V
         SUMSQ=SUMSQ + V*V
         TYPE 120,I,V
120      FORMAT("VALUE",I,"IS",E,/)
200      CONTINUE
         SAMP=N
         AVRG=SUM/SAMP
         STD=SQTF(SUMSQ/SAMP - AVRG**2)
         TYPE 300,N,AVRG,STD
300      FORMAT("NUMBER OF VALUES",I,"MEAN",E,"STANDARD DEVIATION",E,/)
         END

*
.R FORT
?
```

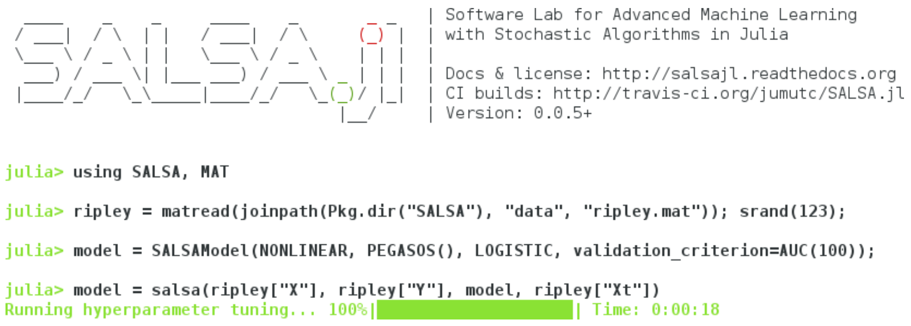
Figure 1.7: Printout of the code snippet calculating statistics on data in `Fortran`, the first high-level scientific programming language.

Since then hundreds of high-level languages have been developed. However, a few have stood the test of time. The same claim applies to the machine learning software as well. Since the times of `Weka` [55] many talented and visionary researchers and practitioners have tried to devise their own implementation of the most famous and cutting edge machine learning methods. Nowadays there are many available implementations, such as `scikit-learn` [99] or `Shogun` [112], which are tailored towards different machine learning sub-domains but only a few of them in practice are implemented based on flexible and extendable software design principles. The latter paves the way to a number of possible extensions and enhances the user experience of other software developers and out-of-field practitioners.

One of the promising directions in scientific computing and high-level, high-performance dynamic programming languages is called `Julia`. `Julia` language features a syntax that is familiar to users of other technical computing environments. It provides a sophisticated compiler, distributed parallel execution, numerical accuracy, and an extensive mathematical function library largely written in `Julia` itself. In Figure 5.1 we present a logo of the `Julia` technical and scientific computing language.

[9]Refer to `http://www.biosmuseum.com/en/2015/10/01/fortran`.

Figure 1.8: `Julia` language logo.



```
 ____    _    _     ____    _      _ _    | Software Lab for Advanced Machine Learning
/ ___|  / \  | |   / ___|  / \    (_) |   | with Stochastic Algorithms in Julia
\___ \ / _ \ | |   \___ \ / _ \   | | |   |
 ___) / ___ \| |___ ___) / ___ \ _ | | |   | Docs & license: http://salsajl.readthedocs.org
|____/_/   \_\_____|____/_/   \_(_)/ |_|   | CI builds: http://travis-ci.org/jumutc/SALSA.jl
                                |__/      | Version: 0.0.5+

julia> using SALSA, MAT

julia> ripley = matread(joinpath(Pkg.dir("SALSA"), "data", "ripley.mat")); srand(123);

julia> model = SALSAModel(NONLINEAR, PEGASOS(), LOGISTIC, validation_criterion=AUC(100));

julia> model = salsa(ripley["X"], ripley["Y"], model, ripley["Xt"])
Running hyperparameter tuning... 100%|████████████████████| Time: 0:00:18
```

Figure 1.9: Usage of `SALSA.jl` software library used with Ripley dataset.

One of the evident advantages of `Julia` is an emerging eco-system comprising of thousands solid software packages purely written in `Julia`. Another advantage is presented by the backbone of open source and well-recognized and tested numerical libraries, such as LAPACK [6] or BLAS [17]. Finally data structures, lexical scoping and mechanisms embedded into `Julia` allow an infinite number of combinations and usage patterns previously inherent only to purely Object Oriented or Functional Programming languages.

Using the aforementioned principles many researches and software developers have implemented useful and interesting libraries for machine learning and artificial intelligence fields. To give a good qualitative example in Figure 1.9 we present a snippet of our contribution to the open source software initiative, namely `SALSA.jl` software library and corresponding classification use case using Ripley dataset [121].

1.2.6 Scalability of the Machine Learning Software

Scalability of the Machine Learning software was always a major concern since we have started generating amounts of data that cannot fit into the memory of a single computer. There are multiple ways of approaching the scalability issue. One is related to cloud computing environments and the aforementioned `Hadoop` [128] and `Spark` [137] eco-systems which are implementing a well-acknowledged `MapReduce` scheme [40].

One of the machine learning libraries embedded into `Apache Spark` is `MLib` [89]. It implements a wide range of learning settings and includes several underlying statistical, optimization and linear algebra primitives. These primitives take a full advantage of the distributed environment and do parallelize numerical computations across available nodes in a cluster.

Another way of approaching the same issue is related to the multi-threaded compilation and execution of programs. This option is more suitable for supercomputers and it is less prone to the communication bottleneck problem. Many existing numerical computing libraries, such as LAPACK or BLAS, can be easily compiled with this option and can distribute computations across all available processors (cores).

Listing 1.1: Usage of SALSA.jl package with HIGGS Data Set [11]

```
julia> using SALSA;
julia> f = LabelledDelimitedFile("./Data/HIGGS.csv",false,',',1);
julia> @time w, b = pegasos_alg(loss_derivative(LOGISTIC),f,[],.1,22000,500,1e-5);
300.523824 seconds (1.06 G allocations: 232.014 GB, 14.80% gc time)
julia> size(f)
(11000000,28)
```

To illustrate an importance of scalability we present a quantitative experiment for the previously mentioned `SALSA.jl` software library when learning from the large-scale HIGGS Data Set [11]. As we can see from the code snippet presented in Listing 1.1 it contains 11 million data points and 28 dimensions. Just in a matter of few minutes we can learn a stochastic model with logistic loss by going though the entire sample (dataset) within 500 iterations and 22 thousand data points (batch size) per iteration. We learn our model sequentially in one thread.

Using this model (w, b) we can try to visualize our findings and predicted labels for the first 10 thousand samples in Figure 1.10. Subfigures represent the Multidimensional Scaling (MDS) approach [18] applied to this subsample in order to lower the dimensionality of the input space for visualization. As we can see even a linear model observing the entire sample only once can deliver a decent accuracy rate of more than 60%.

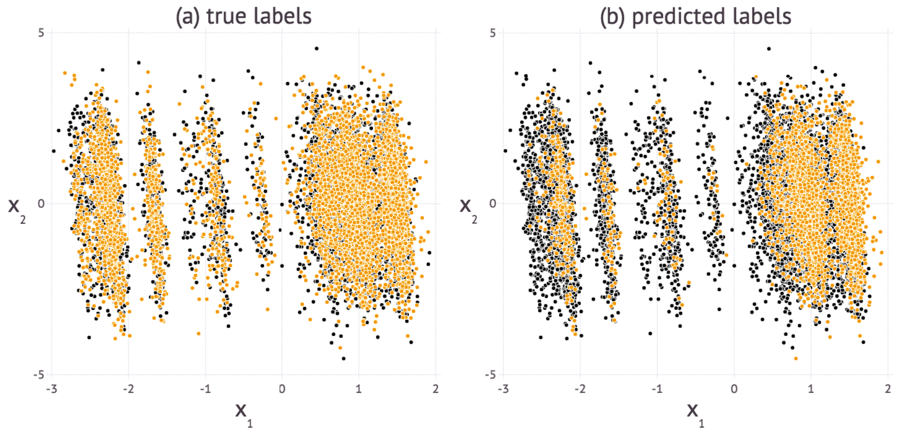


Figure 1.10: Distribution of (a) true and (b) predicted labels of the first 10000 observations in the HIGGS dataset. We train our Pegasos-based [107] model with logistic loss using the entire HIGGS sample of 11 million data points. MDS approach [18] was applied to reduce dimensionality of the problem.

1.3 Objectives and Motivations

We outline the objectives and motivations of this thesis below:

- The first objective is to design a novel kernel-based approach for novelty (outlier) detection which would take into account labelling information from several observed classes (distributions) and optimally model density and support of the aforementioned distributions to detect outliers in data. We aim at developing a unified kernel-based approach which is suitable for both: novelty detection and multi-class classification. We approach a data-driven model which exploits different aspects of the presented novelty detection problem in terms of the primal-dual optimization framework [22]. Another related objective is to extend our approach to the semi-supervised learning setting. We aim at learning from partially labelled data while being able to classify or cluster unlabelled data points.

- The second objective is to address a stochastic learning paradigm within the Regularized Empirical Risk Minimization [125] framework, and more specifically linear Support Vector Machines solvable by the Pegasos (Primal Estimated sub-GrAdient SOlver for SVM) algorithm [107]. We aim at developing more specific extensions for this algorithm which would be tailored towards new loss functions and adaptive coordinate-wise learning schemes. We focus on the stochastic learning paradigm because of an emerging Big Data concept and infeasibility of other learning schemes when dealing with large-scale datasets comprising of millions and billions of data points. Application of proximal Stochastic Gradient Descent (SGD) to the well-known machine learning problems guarantees convergence rates independent of the sample size. We can reach ϵ-optimal solution within a fixed computational budget of $T \approx \frac{1}{\epsilon}$ iterations in case of the strongly convex optimization objective.

- The third objective is related to a study of the sparsity inducing l_0 pseudo norm. Again we tackle our objective within the Regularized Empirical Risk Minimization [125] framework and regularized dual averaging schemes [134] applied via stochastic optimization. We aim at studying reweighted learning schemes which in a limit could approach the l_0-type of a penalty. This is in particular interesting when the l_1-norm regularization does not attain considerable recovery of a sparse pattern in the obtained solution or we are interested in the maximally compressed solution. The latter speeds up model evaluation and reduces the overall memory footprint.

- The fourth objective is devoted to an extension of the well-acknowledged K-Means [83] clustering algorithm which we tackle via the Regularized Empirical Risk Minimization framework. We opt for the compressed and sparsified solution (prototype vectors). We study the regularization applied to prototype vectors in order to reduce the effect of outliers in data. Another related objective is connected with the `MapReduce` [40] scheme which can be used for learning in parallel individual prototype vectors. We would like to achieve a considerable speed-up when the number of clusters to be located is large.

- Our final objective is related to the machine learning open source software. We aim at developing a unique blend of the latest methodology based on the Regularized Empirical Risk Minimization and stochastic algorithms together with the user-centric and user-friendly implementation and design strategies. These strategies are ubiquitous in software development industry but still are somewhat incomplete or missing in academia. We can notice a lot of advanced low-level machine learning libraries which are hardly conceived by the out-of-field scientists and practitioners. We want to bridge the gap between broader audience which is unfamiliar with programming or implementation details on the one hand and skilful machine learning practitioners and developers on the other. Furthermore our motivation is to enhance and foster the usage of machine learning software by adopting very easy to comprehend and utilize application programming interfaces and routines. In addition these interfaces should be tailored to the needs of the Big Data world.

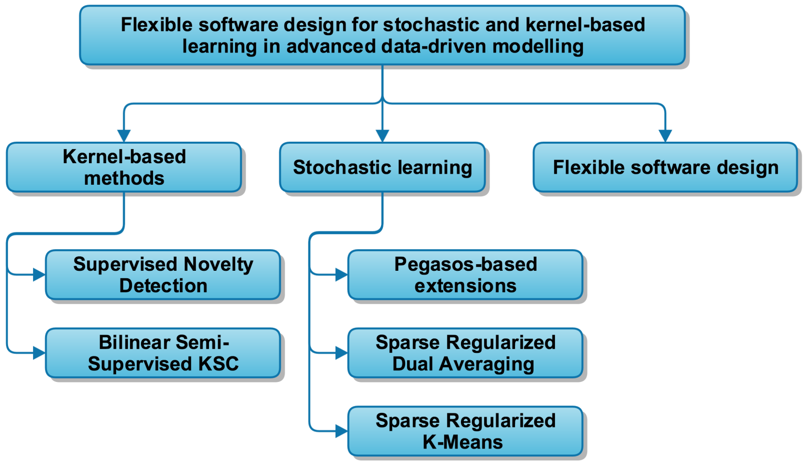

Flexible software design for stochastic and kernel-based learning in advanced data-driven modelling
Kernel-based methods
Stochastic learning
Flexible software design
Supervised Novelty Detection
Bilinear Semi-Supervised KSC
Pegasos-based extensions
Sparse Regularized Dual Averaging
Sparse Regularized K-Means

Figure 1.11: Outline of the contributions in this doctoral thesis.

1.4 Contributions of this thesis

The main contributions of this thesis are summarized as follows:

- **New regularization and coupling mechanisms for kernel-based methods in unsupervised and semi-supervised learning.** We study a novel regularization approach based on a coupling term between classifiers (classes) in primal. This approach extends the One-Class Support Vector Machine (SVM) setting for supervised and semi-supervised classification while keeping the nice properties of the novelty detection problem at hand. We address the latter problem by presenting a new class of SVM-like algorithms which helps to approach multi-class classification, semi-supervised classification and novelty detection from a new perspective. The introduced coupling term between classes leverages the problem of finding a good decision boundary while preserving the compactness of a support with the l_2-norm penalty. Additionally by utilizing the Kernel Spectral Clustering [5] framework we are able to extend our approach to the semi-supervised setting where only a few data points are provided with the labeling information. The related contributions are:

    1. Jumutc V., Suykens J.A.K., **"Multi-Class Supervised Novelty Detection"**, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 36, no. 12, Dec. 2014, pp. 2510 - 2523.

2. Jumutc V., Suykens J.A.K., **"New Bilinear Formulation to Semi-Supervised Classification Based on Kernel Spectral Clustering"**, in Proc. of the 2014 IEEE Symposium Series on Computational Intelligence (IEEE SSCI 2014), Orlando, Florida, Dec. 2014, pp. 41-47.

- **New algorithms for stochastic learning and linear SVM.** We present several extensions to the widely acknowledged Pegasos algorithm [107, 106]. It utilizes properties of hinge loss and theory of strongly convex optimization problems for fast convergence rates and lower computational and memory costs. First we adopt recently developed pinball loss SVM [58] for the Pegasos algorithm and show some advantages of using it in a variety of classification problems. Additionally we present extensions of the Pegasos algorithm applied to the kernel-induced and Nyström approximated [129] feature space which introduces non-linearity in the input space. This is done using a Fixed-Size kernel approach [39]. Second we devise a new weighted formulation of the Pegasos algorithm which favors from the coordinate-wise regularization hyperparameters. For both methods we provide theoretical justifications of the convergence and optimality. The related contributions are:

  1. Jumutc V., Huang X., Suykens J.A.K., **"Fixed-Size Pegasos for Hinge and Pinball Loss SVM"**, in Proc. of the 2013 International Joint Conference on Neural Networks (IJCNN 2013), Dallas, USA, Aug. 2013, pp. 1122-1128

  2. Jumutc V., Suykens J.A.K.„ **"Weighted Coordinate-Wise Pegasos"**, in Proc. of the 5th International Conference on Pattern Recognition and Machine Intelligence (PREMI 2013), Kolkata, India, Dec. 2013, pp. 262-269.

- **New sparsity inducing regularization for stochastic learning with SVM and other parametric models.** Recent advances in stochastic optimization and regularized dual averaging approaches revealed a substantial interest for a simple and scalable stochastic method which is tailored to some more specific needs. Among the latest one can find sparse signal recovery and l_0-based sparsity inducing approaches. These methods in particular can force many components of the solution shrink to zero thus clarifying the importance of the features and simplifying the evaluation. We concentrate on enhancing sparsity of the recently proposed l_1- and l_2-Regularized Dual Averaging (RDA) methods [134, 44] with a simple reweighting iterative procedure [68, 67] which in a limit applies the l_0-type of a penalty. We provide theoretical justifications of a bounded regret for a sequence of convex repeated games where every game stands for a separate reweighted RDA problem. Additionally we present a novel clustering approach based on the well-acknowledged K-Means algorithm which exploits some of the aforementioned schemes [44] for applying the regularization and inducing sparsity on the level of prototype vectors. The related contributions are:

1. V. Jumutc, J.A.K. Suykens, **"Reweighted stochastic learning"**, Neurocomputing Special Issue - ISNN2014, 2015. (In Press)
2. V. Jumutc, R. Langone, and J. A. K. Suykens, **"Regularized and sparse stochastic k-means for distributed large-scale clustering"**, to appear in 2015 IEEE International Conference on Big Data, 2015.

- **Flexible software design for machine learning software**. We present Software lab for Advanced machine Learning with Stochastic Algorithms (SALSA) which implements some of the well-known stochastic algorithms for machine learning [107, 134, 44] and our recent developments in [68, 67]. It was developed in the high-level technical computing language `Julia`. By this software package we address challenges in sparse linear modelling, linear and non-linear Support Vector Machines applied to large data samples with user-centric and user-friendly emphasis. We focus also on flexibility and extensibility of the package for other software developers and out-of-field practitioners. We have implemented SALSA to alleviate and enhance user experience with ubiquitous machine learning problems including classification, regression and clustering. It provides various interfaces and utilities for different pre- and post-processing routines, such as cross-validation, normalization *etc.* The related contributions are:

  1. Jumutc V., Suykens J.A.K., **"SALSA: Software Lab for Advanced Machine Learning with Stochastic Algorithms"**, Internal Report 15-179, ESAT-SISTA, KU Leuven (Leuven, Belgium), 2015. (Submitted to JMLR Software Section)

Chapter 2

Kernel-based Methods for Unsupervised and Semi-Supervised Learning

This chapter comprises previously published articles including:

1. Jumutc V., Suykens J.A.K., **"Multi-Class Supervised Novelty Detection"**, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 36, no. 12, Dec. 2014, pp. 2510 - 2523. (Section 2.1)

2. Jumutc V., Suykens J.A.K., **"New Bilinear Formulation to Semi-Supervised Classification Based on Kernel Spectral Clustering"**, in Proc. of the 2014 IEEE Symposium Series on Computational Intelligence (IEEE SSCI 2014), Orlando, Florida, Dec. 2014, pp. 41-47. (Section 2.2)

Keywords—novelty detection; One-Class SVM; classification; pattern recognition; labeling information;

2.1 Multi-Class Supervised Novelty Detection

2.1.1 Introduction

Novelty or anomaly detection is a widely recognized machine learning problem where one tries to find a compact support of some unknown probability distribution. Many existing methods, like One-Class SVM [103] or Bayesian approaches [41], heavily rely on the *i.i.d.* assumption and deal with unlabeled data. Contrary to these methods it was proposed recently [116] to approach novelty detection from a classification perspective. In this setting one tries to tackle density estimation via a weighted binary classification problem. However, while the results presented in [116] are consistent with those obtained by other works on Novelty Detection [24, 42], it is still unclear how these methods behave when the *i.i.d.* assumption does not hold or data are generated by a mixture of distributions. In this research we try to close the gap by answering some of the following questions. What if we model the support of each distribution (class) separately? How, in this case, are these models relating to each other? What is the optimal interpretation of such a problem?

In this paper we concentrate on presenting three different extensions of our previous method of Supervised Novelty Detection (SND) introduced in [66]. The first extension is formulated in terms of a QP problem with box constraints. The second one is a Least-Squares problem given by a linear Karush-Kuhn-Tucker (KKT) system. The third one is related to large-scale problems where one cannot approach the solution with standard QP solvers. In our previous research [66] we derived only the binary formulation of the SND method while in the current paper we extend it to the multi-class case. In this setting one is interested in obtaining decision functions for each class respectively while trying to keep the data description compact [122]. This merges together objectives of novelty detection and classification and reveals the importance of bringing them together. The outliers in this scheme can be identified as the data which are not covered by any of the classes related to the obtained decision functions.

To illustrate the practical importance of the Supervised Novelty Detection we apply it to data from AVIRIS (Airborne Visible/InfraRed Imaging Sensor) [97]. Some previous papers on anomalous change detection [117, 59] already exploited the importance of SVM-based approaches in hyperspectral analysis of infrared images. However we can extend this along the lines of classification and detect hyperspectral changes among different types of terrain while trying to automatically categorize the pixels according to these types. Another promising application of SND are Intrusion Detection Systems (IDS). Here the goal is to identify intruders which might be scattered between many existing user groups. We cannot rely then on the fact that all users are originated from the same underlying distribution. Therefore many existing approaches would fail to generalize under the *i.i.d.* assumption. One might consider intruders as a separate class and resolve the problem in a multi-class fashion. But this approach is not very practical because of the initial diversity of intruders and high risk of overfitting of the resulting classifier. Combining One-Class with Multi-Class SVM might not be an optimal solution because of an added complexity and intermediate difficulties with integration in the provided solution.

The remainder of this paper is structured as follows. Section 2.1.2 gives a general view of our approach and discusses some related methods proposed in the literature. Section 2.1.3 gives some conventional notations and reviews the binary case of the SND method. Section 2.1.4 outlines the multi-class QP and Least-Squares formulation while Section 2.1.5 extends the SND algorithm to large-scale problems with the newly derived optimization objective and provides theoretical bounds for convergence. Section 2.1.6 discusses some implementation and algorithmic issues. Section 2.1.7 provides the experimental setup and results. Finally Section 2.1.8 concludes the paper.

2.1.2 Problem statement and related work

Problem statement

Supervised Novelty Detection (SND) is designed for finding outliers in the presence of several classes/distributions. While being useful for detecting the outliers, the SND method can be effectively used for multi-class classification and it supplements the class of SVM-based algorithms. One can regard our approach as an extension of the original work by Schölkopf *et al.* [103] for One-Class SVM where one deals with the support of a high-dimensional distribution. Contrary to Schölkopf's approach we deal with labeled data and take the *i.i.d.* assumption for every class separately. We might also find some connections to [135] where the authors try to ablate outliers while trying to locate them with a new SVM objective reformulated in terms of a hinge loss. SND doesn't try to find outliers in the existing data pool of data. In general our objective is quite opposite. We try to find the support of each distribution per class such that we can identify outliers within our test or validation set while keeping a necessary discrimination between the observed classes. Moreover we can use outliers at the learning stage just by keeping their labels negative for all involved classes. This strategy helps to incorporate all available information at once.

Difference with other SVMs

We can think of SND as solving a density estimation problem for each involved distribution per class while trying to separate the classes as much as possible. In practice this results in finding an appropriate trade-off between the amount of errors, separation and compactness[1] of our model describing these particular distributions. The demonstrated problem is not of the same kind as other SVMs where one copes only with optimal separation (minimization of an average error) and the smoothness of the classifier. For instance, in Laplacian SVMs [88] one uses additional regularization to keep the values of the decision function for adjacent points similar but this regularization mostly affects unlabeled samples. In other methods [135] one is estimating outliers explicitly via a reformulated hinge-loss penalty. This setting is quite different from our objective of density estimation where we deal with the outliers either implicitly (see Section 2.1.6 for further remarks) or explicitly by setting all respective labels to -1's.

2.1.3 Binary case

Notation

We first introduce terminology and some notational conventions. We consider training data with the corresponding labeling given as a set of pairs

$$(x_1, y_1), ..., (x_n, y_n), x_i \in \mathcal{X}, y_i \in \{-1, 1\},$$

where n is the number of corresponding observations in the set $\mathcal{X}$. Let $\mathcal{X}$ be a compact subset of $\mathbb{R}^d$.

In Section 2.1.3 index i spans the range $\overline{1, n}$ if it is not declared explicitly. Greek letters $\alpha, \beta, \lambda, \xi$ without indices denote n-dimensional vectors, while in Section 2.1.4 Greek letters $\alpha, \beta, \lambda, \xi$ spanning only one index denote n-dimensional vectors. In Section 2.1.5 letters w and x denote d-dimensional vectors. Otherwise Greek letters denote constants or scalars throughout the paper.

Illustrative example

According to the classical work by Schölkopf *et al.* [103] in One-Class SVM we aim at mapping the data points into the feature space and separating them

[1] by that we mean finding the smallest unit ball in the feature space that captures all the data, see [103] for details

from the origin with maximum margin. From the joint perspective of density estimation for multiple distributions simultaneously we require more than only the compactness properties discussed in the previous section. From the model perspective we need a classification scheme which would preserve compactness and separation of distributions simultaneously. In our illustrative example we

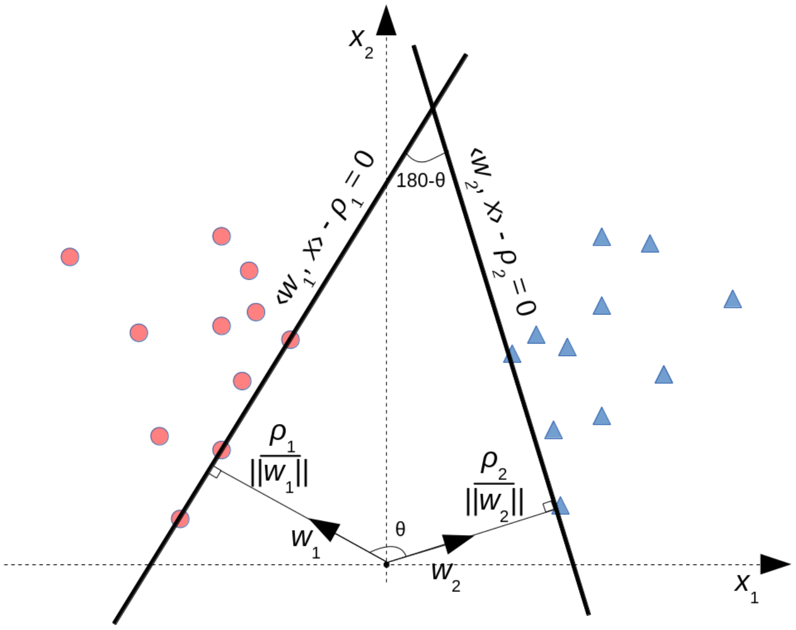
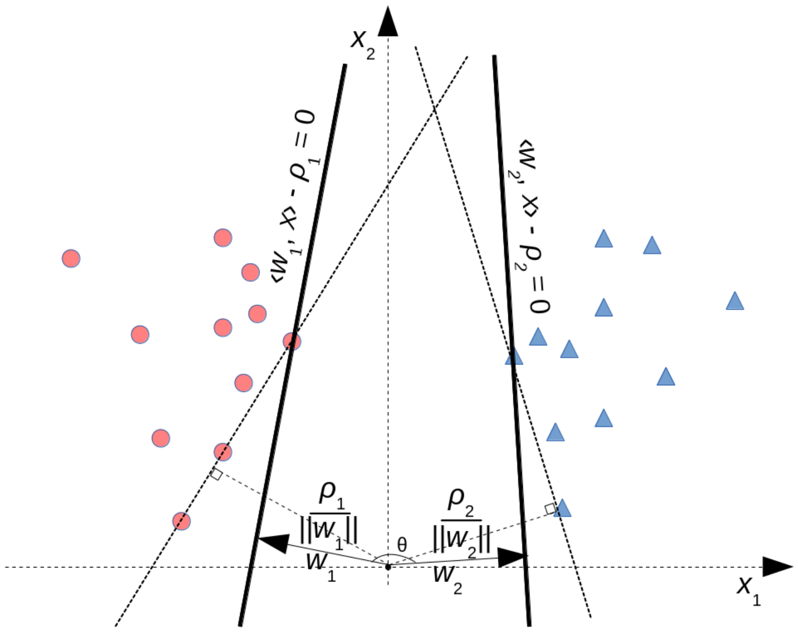


Figure 2.1: SND solution in the feature space. SND aims at separating training data by minimizing the inner product between the normal vectors w_1 and w_2 to the decision hyperplanes while maximizing the margins (distances) between these hyperplanes and the origin.

are emphasizing two core objectives of the SND method:

- maximizing margins $\frac{\rho_1}{\|w_1\|}$ and $\frac{\rho_2}{\|w_2\|}$,
- pushing θ closer to $180°$ angle (making $\cos\theta \simeq -1$).

If we take a look at the illustrative example in Figure 2.1 we can notice that these objectives are contradicting with each other. By making angle θ closer to 180 degrees we are making margins $\frac{\rho_1}{\|w_1\|}$ and $\frac{\rho_2}{\|w_2\|}$ smaller as it can be observed from Figure 2.2. This can be explained as well from the cosine perspective



x_2
$\langle w_1, x\rangle - \rho_1 = 0$
$\langle w_2, x\rangle - \rho_2 = 0$
$\frac{\rho_1}{\|w_1\|}$
$\frac{\rho_2}{\|w_2\|}$
θ
w_1
w_2
x_1

Figure 2.2: SND solution in the feature space if we are emphasizing the second objective, making $\cos\theta \simeq -1$.

$$\cos\theta = \frac{\langle w_1, w_2\rangle}{\|w_1\|\|w_2\|}$$

as we should maximize $\|w_1\|$, $\|w_2\|$ (denominator) and minimize $\langle w_1, w_2\rangle$ (numerator) in order to minimize the cosine and push angle θ closer to 180°. Following exactly this reasoning we present our binary QP problem in Section 2.1.3 where we trade-off the minimization of a coupling term $\langle w_1, w_2\rangle$ in the cosine, minimization of the l_2-norms for the normal vectors w_1 and w_2 and the training errors ξ_i. We maximize the ρ_1, ρ_2 values as well as they do enter the definition of the margins for both decision hyperplanes.

In Figure 2.3 we show some clear advantages of the SND approach over One-Class SVM. The latter is not capable of identifying an outlier if it is located on the line connecting centroids of each distribution. One-Class SVM treats all samples as being drawn from the same distribution under the *i.i.d.* assumption.

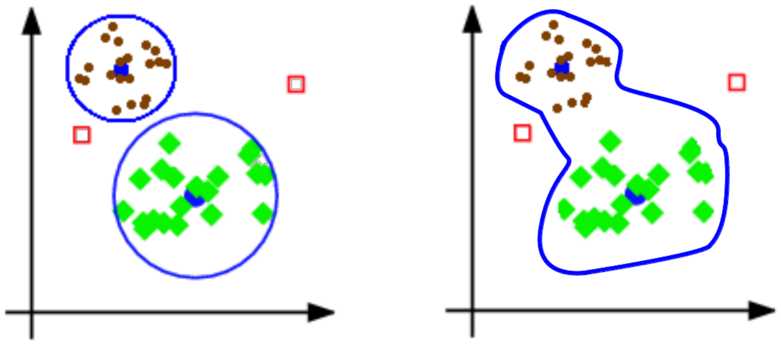


Figure 2.3: Qualitative figure illustrating the main difference between SND solution (left) and One-Class SVM solution (right) in the input space. SND can provide the better and more compact estimate of each distribution. If an outlier sample (marked with the red square) was located on the line connecting centroids of each distribution One-Class SVM method would not detect such an outlier.

Binary QP problem

For the completeness we recap in this section the binary formulation of our approach [66] and then continue with the generalized multi-class QP and Least-Squares problem in the next sections.

First we start with the initial set of constraints which clarify the nature of our optimization problem w.r.t. normal vectors w_1, w_2 and maximization of the ρ bias terms [103, 104]

$$\begin{aligned} \langle w_1, \Phi(x_i)\rangle &\geq \rho_1 - \xi_i^{(1)}, \quad \{x_i \in \mathcal{X} | y_i = 1\}, \\ \langle w_2, \Phi(x_i)\rangle &\geq \rho_2 - \xi_i^{(2)}, \quad \{x_i \in \mathcal{X} | y_i = -1\}, \end{aligned} \tag{2.1}$$

where $y_i \in \{-1, 1\}$. To make a link between the One-Class SVM formulation and our method we join the constraints in Eq.(2.1) and propose the following optimization problem

$$\begin{aligned} \min_{w_1, w_2 \in \mathcal{F}; \xi, \xi^* \in \mathbb{R}^n; \rho_1, \rho_2 \in \mathbb{R}} \quad & \tfrac{\gamma}{2}(\|w_1\|^2 + \|w_2\|^2) + \langle w_1, w_2\rangle \\ & + C\textstyle\sum_{i=1}^n (\xi_i + \xi_i^*) - \rho_1 - \rho_2 \end{aligned} \tag{2.2}$$

$$\begin{aligned} \text{s.t.} \quad & y_i(\langle w_1, \Phi(x_i)\rangle) \geq \rho_1 - \xi_i, & i \in \overline{1, n} \\ & y_i(\langle w_2, \Phi(x_i)\rangle) \geq \rho_2 - \xi_i^*, & i \in \overline{1, n} \\ & \xi_i \geq 0, \xi_i^* \geq 0, & i \in \overline{1, n} \end{aligned} \tag{2.3}$$

where γ and C are trade-off parameters. The decision functions are

$$\begin{aligned} f_{c_1}(x) &= \langle w_1, \Phi(x)\rangle - \rho_1, \\ f_{c_2}(x) &= \langle w_2, \Phi(x)\rangle - \rho_2. \end{aligned} \tag{2.4}$$

The final decision rule collects f_{c_1} and f_{c_2} as follows

$$c(x) = \begin{cases} \operatorname{argmax}_{c_i}\ f_{c_i}(x), & \text{if } \ \max_i f_{c_i}(x) > 0 \\ c_{out}, & \text{otherwise,} \end{cases} \tag{2.5}$$

where c_i is either the positive or negative class in the binary classification setting and c_{out} stands for the outliers' class.

Remark 1. *Here we should stress the main difference with the binary classification setting where labels y_i are strongly associated with classes c_i. Our decision rule implies a separate class which doesn't directly enter the formulation in Eq.(2.2) but is thoroughly used for determining tuning parameters and calculation of the performance measures for our method. These data are assigned to an outliers' class as it doesn't belong to any of the encoded classes and can be seen as an unsupervised counterpart of our algorithm that can enter the optimization objective but those y_i labels for all classes will be set to -1. This is different from Laplacian SVMs [88] and manifold regularization [14]. The data $\mathcal{Z}$ are a subset of $\mathcal{X}$ defined as follows*

$$z_1, \dots, z_m \in \mathcal{Z} \subseteq \{\mathcal{X} : y_i = -1, i \in \overline{1, n_c}\}, \tag{2.6}$$

where n_c gives the total number of classes. This setting explicitly follows the multi-class case of Section 2.1.4 and will be explained in detail in Section 2.1.6.

Using $\alpha_i, \lambda_i, \geq 0$ and $\beta_i, \beta_i^* \geq 0$ Lagrange multipliers we introduce the following Lagrangian

$$\begin{aligned} \mathcal{L}(w_1, w_2, \xi, \xi^*, \rho_1, \rho_2, \alpha, \lambda, \beta, \beta^*) &= \tfrac{\gamma}{2}(\|w_1\|^2 + \|w_2\|^2) \\ &+\langle w_1, w_2\rangle + C\textstyle\sum_{i=1}^n(\xi_i + \xi_j^*) \\ &-\textstyle\sum_{i=1}^n \alpha_i(y_i(\langle w_1, \Phi(x_i)\rangle) - \rho_1 + \xi_i) \\ &-\textstyle\sum_{i=1}^n \lambda_i(y_i(\langle w_2, \Phi(x_i)\rangle) - \rho_2 + \xi_i^*) \\ &-\textstyle\sum_{i=1}^n \beta_i\xi_i - \sum_{i=1}^{n_c} \beta_i^*\xi_i^* - \rho_1 - \rho_2. \end{aligned} \tag{2.7}$$

Before going to the final dual representation of Eq.(2.2) let Φ be a feature map $\mathcal{X} \to \mathcal{F}$ in connection to a positive definite Gaussian kernel [19, 102]

$$k(x, y) = \langle \Phi(x), \Phi(y)\rangle = e^{-\frac{\|x-y\|^2}{2\sigma^2}}. \tag{2.8}$$

By setting the derivatives of the Lagrangian with respect to the primal variables to zero, obtaining the saddle point conditions and substituting those into the Lagrangian one can directly obtain the matrix form of the corresponding Lagrangian to be maximized

$$\max_{\alpha,\lambda} \mathcal{L}_D(\alpha,\lambda) = \frac{\mu_1}{2}(\alpha^T G\alpha + \lambda^T G\lambda) - \mu_2(\alpha^T G\lambda), \tag{2.9}$$

$$\text{s.t.} \quad \begin{array}{l} C \geq \alpha_i \geq 0, \ \forall i \\ C \geq \lambda_i \geq 0, \ \forall i \\ y^T\alpha = 1, \\ y^T\lambda = 1, \end{array} \tag{2.10}$$

where y is a vector of labels, K is the kernel matrix of dimension $n \times n$ with $K_{ij} = k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$, $G = K \circ yy^T$, $\mu_1 = \frac{\gamma}{1-\gamma^2}$, $\mu_2 = \frac{1}{1-\gamma^2}$, and $\circ$ denotes component-wise multiplication. $\mathcal{L}_D$ is maximized and supplements the class of QP problems with box constraints. We can ensure the concavity of our dual objective in Eq.(2.9) by setting $\gamma > 1$. The latter condition is a straightforward consequence from the eigendecomposition of the matrix in the quadratic form of our optimization objective.

2.1.4 Multi-class case

Multi-class QP problem

In this subsection we develop a generic QP formulation for the multi-class setting of our algorithm which returns decision functions f_i for each of the involved target classes (distributions). These functions encode the support for each distribution and output positive values in a corresponding region capturing most of the data points drawn from it.

Combining ideas from One-Class SVM and our assumption we presented previously in Section 2.1.3 the following QP problem is formulated

$$\min_{w_i \in \mathcal{F}; \xi_i \in \mathbb{R}^n; \rho_i \in \mathbb{R}} \quad \begin{array}{l} \frac{\gamma}{2} \sum_{i=1}^{n_c} \|w_i\|^2 + \sum_{i,j=1; i \neq j}^{n_c} \langle w_i, w_j \rangle \\ + C \sum_{i=1}^{n} \sum_{j=1}^{n_c} \xi_{ij} - \sum_{i=1}^{n_c} \rho_i \end{array} \tag{2.11}$$

$$\text{s.t.} \quad \begin{array}{lll} y_{ij} \langle w_j, \Phi(x_i) \rangle \geq \rho_j - \xi_{ij}, & i \in \overline{1,n}, & j \in \overline{1,n_c} \\ \xi_{ij} \geq 0, & i \in \overline{1,n}, & j \in \overline{1,n_c} \end{array} \tag{2.12}$$

where $y_{ij} \in \{-1, 1\}$, γ and C are trade-off parameters and n_c is the number of classes. Here we observe that we are working with the set of indices $\mathcal{Y}$, where every entry $y_i \in \{-1, 1\}^{n_c}$. The decision functions are

$$f_{c_i}(x) = \langle w_i, \Phi(x) \rangle - \rho_i, \tag{2.13}$$

and the final decision rule is derived in Eq.(2.5). Using $\alpha_{ij}, \beta_{ij} \geq 0$ as Lagrange multipliers we introduce the following Lagrangian

$$\begin{aligned} \mathcal{L}(w, \xi, \rho, \alpha, \beta) = & \tfrac{\gamma}{2} \textstyle\sum_{i=1}^{n_c} \|w_i\|^2 + \sum_{i,j=1;i\neq j}^{n_c} \langle w_i, w_j \rangle \\ & + C \textstyle\sum_{i=1}^{n} \sum_{j=1}^{n_c} \xi_{ij} - \sum_{i=1}^{n} \rho_i - \sum_{i=1}^{n} \sum_{j=1}^{n_c} \beta_{ij} \xi_{ij} \\ & - \textstyle\sum_{i=1}^{n} \sum_{j=1}^{n_c} \alpha_{ij} (y_{ij} \langle w_j, \Phi(x_i) \rangle - \rho_j + \xi_{ij}). \end{aligned} \tag{2.14}$$

By setting the derivatives of the Lagrangian with respect to the primal variables to zero and defining $\eta = \gamma + n - 2$ we obtain

$$w_i = \frac{\eta \sum_{j=1}^{n} \alpha_{ji} y_{ji} \Phi(x_j) - \sum_{j=1}^{n} \sum_{p=1,p\neq i}^{n_c} \alpha_{jp} y_{jp} \Phi(x_j)}{(\eta + 1)(\gamma - 1)}, \tag{2.15}$$

$$C - \beta_{ij} - \alpha_{ij} = 0, \quad \forall i \in \overline{1, n} \quad \forall j \in \overline{1, n_c} \tag{2.16}$$

$$\textstyle\sum_{i=1}^{n} \alpha_{ij} = 1, \quad \forall j \in \overline{1, n_c}. \tag{2.17}$$

Substituting Eq.(2.15-2.17) into the Lagrangian and using the kernel trick with the expression given by Eq.(2.8) one can directly obtain the matrix form of the corresponding Lagrangian to be maximized

$$\max_{\alpha_i} \mathcal{L}_D(\alpha_i) = -\frac{1}{\mu} \sum_{i}^{n_c} \lambda_i^T K \alpha_i, \tag{2.18}$$

$$\begin{aligned} \text{s.t.} \quad & C \geq \alpha_{ij} \geq 0, \ \forall i \in \overline{1, n}, \ \forall j \in \overline{1, n_c} \\ & \textstyle\sum_{i=1}^{n} \alpha_{ij} = 1, \ \forall j \in \overline{1, n_c} \end{aligned} \tag{2.19}$$

where $\lambda_i = (\gamma + n - 2)(\alpha_i \circ y_i) - \sum_{j=1,j\neq i}^{n_c} (\alpha_j \circ y_j)$, $\mu = (\eta + 1)(\gamma - 1)$, K is a kernel matrix of size $n \times n$ and $\circ$ denotes component-wise multiplication. $\mathcal{L}_D$ is maximized and is almost identical to one defined in Eq.(2.9) if we take $n_c = 2$. The expression for f_i becomes

$$f_{c_i}(x) = \frac{\eta \sum_{j=1}^{n} \alpha_{ji} y_{ji} k(x_j, x) - \sum_{j}^{n} \sum_{p=1,p\neq i}^{n_c} \alpha_{jp} y_{jp} k(x_j, x)}{(\eta + 1)(\gamma - 1)} - \rho_i, \tag{2.20}$$

where $k(x, y)$ stands for our preferred kernel function in Eq.(2.8).

We can ensure the concavity of our dual objective in Eq.(2.18) by examining necessary conditions for the primal problem in Eq.(2.11) to be strictly convex. This can be done by applying the Gershgorin circle theorem to bound the minimal positive eigenvalue. It is very easy to verify when $\gamma > n_c - 1$ we have $\lambda_{min} > 0$.

Least-Squares problem

To obtain Least-Squares (LS-SND) formulation with equality constraints of our initial problem we reformulate Eq.(2.11) in terms of squared error residuals ξ_{ij}

$$\min_{w_i \in \mathcal{F};\xi_i \in \mathbb{R}^n;\rho_i \in \mathbb{R}} \quad \begin{array}{l} \frac{\gamma_1}{2}\sum_{i=1}^{n_c}\|w_i\|^2 + \sum_{i,j=1;i\neq j}^{n_c}\langle w_i, w_j\rangle \\ +\frac{\gamma_2}{2}\sum_{i=1}^{n}\sum_{j=1}^{n_c}\xi_{ij}^2 - \sum_{i=1}^{n_c}\rho_i \end{array} \tag{2.21}$$

$$\text{s.t.} \qquad y_{ij}\langle w_j, \Phi(x_i)\rangle = \rho_j - \xi_{ij}, \quad i \in \overline{1,n}, \quad j \in \overline{1,n_c}. \tag{2.22}$$

The Lagrangian for this problem is

$$\begin{aligned} \mathcal{L}(w_i,\xi,\rho,\alpha) = &\frac{\gamma_1}{2}\sum_{i=1}^{n_c}\|w_i\|^2 + \sum_{i,j=1;i\neq j}^{n_c}\langle w_i, w_j\rangle \\ &+\frac{\gamma_2}{2}\sum_{i=1}^{n}\sum_{j=1}^{n_c}\xi_{ij}^2 - \sum_{i=1}^{n_c}\rho_i \\ &-\sum_{i=1}^{n}\sum_{j=1}^{n_c}\alpha_{ij}(y_{ij}\langle w_j, \Phi(x_i)\rangle - \rho_j + \xi_{ij}), \end{aligned} \tag{2.23}$$

where the α_{ij} values are the Lagrange multipliers which can be both positive and negative now due to the equality constraints.

By substituting $\eta = \gamma_1 + n - 2$ the conditions for optimality now yield

$$w_i = \frac{\eta\sum_j^n \alpha_{ji}y_{ji}\Phi(x_j) - \sum_j^n\sum_{p=1,p\neq i}^{n_c}\alpha_{jp}y_{jp}\Phi(x_j)}{(\eta+1)(\gamma_1-1)}, \tag{2.24}$$

$$\alpha_{ij} = \gamma_2\xi_{ij}, \quad \forall i \in \overline{1,n} \quad \forall j \in \overline{1,n_c} \tag{2.25}$$

$$\sum_{i=1}^{n}\alpha_{ij} = 1, \quad \forall j \in \overline{1,n_c}. \tag{2.26}$$

By substituting the expressions for w_i and ξ_{ij} in our equality condition, applying the kernel trick in Eq.(2.8) and preserving matrices $G_{ij} = K \circ y_i y_j^T$ and constraints from Eq.(2.19) we can obtain the following linear Karush-Kuhn-Tucker (KKT) system of the form:

$$\Omega\alpha^\star = \theta, \tag{2.27}$$

which we solve in α_i and ρ_i, where

$$\Omega = \left(\begin{array}{ccc|ccc} 0 & \dots & 0 & 1_1^T & \dots & 0_n^T \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & 0_n^T & \dots & 1_n^T \\ \hline 1_n & \dots & 0_n & -\frac{\eta G_{11}^\star}{\mu} & \dots & \frac{G_{1n_c}}{\mu} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0_n & \dots & 1_n & \frac{G_{n_c1}}{\mu} & \dots & -\frac{\eta G_{n_cn_c}^\star}{\mu} \end{array}\right) \tag{2.28}$$

defining $G_{ij}^\star = G_{ij} + \frac{\mu}{\eta\gamma_2} I$ and

$$\alpha^\star = \left(\begin{array}{c} \rho_1 \\ \vdots \\ \rho_{n_c} \\ \hline \alpha_1 \\ \vdots \\ \alpha_{n_c} \end{array}\right) \qquad \theta = \left(\begin{array}{c} 1 \\ \vdots \\ 1 \\ \hline 0_n \\ \vdots \\ 0_n \end{array}\right) \tag{2.29}$$

and 1_n and 0_n denote vectors of length n. To clarify the structure of the matrix Ω we should refer to every part of this matrix separately. The upper-left submatrix is a square matrix of size $n_c \times n_c$ where all residuals are zeros. The upper-right and bottom-left matrices are block diagonal where every element on the diagonal is a vector 1_n. These matrices are identical but the upper-right matrix is transposed. The bottom-right matrix is a square matrix of size $nn_c \times nn_c$ where every element on the diagonal is of the form $-\frac{\eta}{\mu}(G_{ii} + I/\gamma_2)$ and any off-diagonal element is bound to matrix G_{ij} in the following form: $\frac{G_{ij}}{\mu}$. The final decision function and the decision rule are of the same form as in Eq.(2.20) and Eq.(2.5).

Remark 2. *Additionally we should emphasize that the Least-Squares form of our algorithm is of much less complexity than QP formulation and results in only one linear system of size $nn_c \times nn_c$. This drastically decreases computational costs for the cross-validation procedure which will be presented in Section 2.2.4 and mentioned in the description of Algorithms 2 – 3.*

2.1.5 Large-scale optimization problem

Algorithm

To cope with large-scale datasets we propose a scalable first-order optimization algorithm for the multi-class QP problem. The formulation is inspired by the Pegasos algorithm [107] and we provide theoretical justification along the lines of the Pegasos formulation.

Remark 3. *The large amount of variables significantly slows down every iteration of any QP solver and starting from several thousands of variables even our approach for tuning the parameters (see Section 2.2.4) becomes unfeasible. To tackle this problem one may study a scalable SMO-like method by Platt [100] or Nesterov's approach for convex optimization [94]. However we selected here a Pegasos-like implementation of the SND algorithm which makes use of the Nyström approximation of the RBF kernel [39, 129] and converges with the selected accuracy* ϵ *within* $\mathcal{O}(\frac{R^2}{\lambda\epsilon})$ *iterations. This result originally provided in [107] is much better than previously implemented approaches (e.g. SVM-Perf [62]) which like Pegasos make use of the subgradient descent but converge in* $\mathcal{O}(\frac{R^2}{\lambda\epsilon^2})$.

First we rewrite our optimization objective in Eq.(2.11) in terms of the hinge loss. Second we move the bias terms ρ_i into the hinge loss. Finally we optimize only over the weights w_i which are joint together as

$$w = \begin{pmatrix} w_1 \\ \vdots \\ w_{n_c} \end{pmatrix}$$

to be compatible with the original formulation of the Pegasos algorithm. We benefit from the convergence analysis provided in [107] and present our adjustments for the SND method in Theorem 3.

We derive an approximate instantaneous objective function in the primal for the SND method by

$$f(w; \mathcal{A}_t; B_i; \Gamma) = \frac{\lambda}{2} w^T \Gamma w + \frac{1}{m} \sum_{i=1}^{n_c} \sum_{(x,y)\in\mathcal{A}_t} \mathbb{L}(w; B_i; (x, y)), \tag{2.30}$$

where the hinge loss for the i-th class is given by

$$\mathbb{L}(w; B_i; (x, y)) = \max\{0, 1 - y(\langle w, B_i^T x\rangle + \rho_i)\}, \tag{2.31}$$

and $\mathcal{A}_t$ is our working subset (subsample) at iteration t and matrices Γ and B_i are of the special form

$$\Gamma = \begin{pmatrix} \gamma I_{11} & \dots & I_{1n_c} \\ \vdots & \ddots & \vdots \\ I_{n_c1} & \dots & \gamma I_{n_cn_c} \end{pmatrix}, \qquad B_i = \begin{pmatrix} 0 & \dots & I_i & \dots & 0 \end{pmatrix}. \tag{2.32}$$

In the above equations we expect w to be of dimension dn_c where d is our input dimension and n_c is the number of classes. Every identity matrix or zero matrix is of dimension $d \times d$ and $\rho_i \in \mathbb{R}$. Scalar m denotes the size of the working subset $\mathcal{A}_t$.

Here we should emphasize that we carry out optimization only *w.r.t.* w and we include ρ (which is part of the hinge loss) as a additional (last) element of vector w. This strategy, originally proposed in [107], allows us to rely on the strong convexity of the optimization objective.

Next we present a brief summary of the large-scale SND method in Algorithm 1 and continue with the analysis in the next subsection. Below we denote the whole dataset by $\mathcal{S}$.

Algorithm 1: Pegasos-based SND algorithm

```
Data: $\mathcal{S}, \gamma, \lambda, T, m$
1  Compute $\Gamma$ and $B_i$ matrices defined in Eq.(2.32)
2  Set $w^{(1)}$ randomly s.t. $\|w^{(1)}\| \leq \sqrt{n_c/\lambda(\gamma + n_c - 1)}$
3  for $t = 1 \to T$ do
4      Set $\eta_t = \frac{1}{\lambda t}$
5      Select $\mathcal{A}_t \subseteq \mathcal{S}$, where $|\mathcal{A}_t| = m$
6      $\mathcal{A}^+_{t(i)} = \{(x, y) \in \mathcal{A}_t : y(\langle w, B_i^T x\rangle) < 1\}, \forall i$
7      $w^{(t+\frac{1}{2})} = w^{(t)} - \eta_t(\lambda \Gamma w^{(t)} - \frac{1}{m}\sum_{i=1}^{n_c}\sum_{(x,y)\in\mathcal{A}^+_{t(i)}} y B_i^T x)$
8      $w^{(t+1)} = \min\left\{1, \frac{\sqrt{n_c/\lambda(\gamma+n_c-1)}}{\|w^{(t+\frac{1}{2})}\|}\right\} w^{(t+\frac{1}{2})}$
9  end
10 return $w^{(T+1)}$
```

The above algorithm is based on the Pegasos formulation but differs in the computation of the subgradient and the projection step. Now we can see that the subgradient

$$\nabla_t = \lambda \Gamma w^{(t)} - \frac{1}{m} \sum_{i=1}^{n_c} \sum_{(x,y_i) \in \mathcal{A}^+_{t(i)}} y_i B_i^T x \tag{2.33}$$

depends on the additional matrices Γ and B_i introduced in Eq.(2.32) and in projection step (8) we have slightly different rescaling term.

Analysis

In this subsection we present a convergence analysis which brings to our algorithm the same convergence bounds as in Pegasos. We extend the analysis presented in [107] to our instantaneous objective by presenting Theorem 3. But first we recap the important lemma from [107] which establishes necessary conditions for our theorem.

Lemma 1 (Shalev-Shwartz et al., 2007)**.** *Let $f^{(1)}, ..., f^{(T)}$ be a sequence of λ-strongly convex functions w.r.t. the function $\frac{1}{2}\|\cdot\|^2$. Let $\mathcal{B}$ be a closed convex set and define $\prod_{\mathcal{B}}(w) = arg\min_{w' \in \mathcal{B}} \|w - w'\|$. Let $w^{(1)}, \ldots, w^{(T+1)}$ be a sequence of vectors such that $w^{(1)} \in \mathcal{B}$ and for $t \geq 1$, $w^{(t+1)} = \prod_{\mathcal{B}}(w^{(t)} - \eta_t \nabla_t)$, where ∇_t is a subgradient of $f^{(t)}$ at $w^{(t)}$ and $\eta_t = 1/\lambda t$. Assume that for all t, $\|\nabla_t\| \leq G$. Then, for all $u \in \mathcal{B}$ we have*

$$\frac{1}{T} \sum_{t=1}^{T} f(w^{(t)}) \leq \frac{1}{T} \sum_{t=1}^{T} f(u) + \frac{G^2(1 + \ln(T))}{2\lambda T}.$$

Based on the above lemma, we are now ready to bound the average instantaneous objective of Algorithm 1.

Theorem 1. *Assume $\|x\| \leq R$ for all $(x, y) \in \mathcal{S}$. Let $w^* = arg\min_w f(w; \mathcal{A}_t; B_i; \Gamma)$ and let $c = \sqrt{\lambda n_c(\gamma + n_c - 1)} + n_c R$. Then, for $T \geq 3$ and $\gamma > n_c - 1$ we have*

$$\frac{1}{T} \sum_{t=1}^{T} f(w^{(t)}; \mathcal{A}_t; B_i; \Gamma) \leq \frac{1}{T} \sum_{t=1}^{T} f(w^*; \mathcal{A}_t; B_i; \Gamma) + \frac{c^2 \ln(T)}{\lambda T}.$$

Proof. To prove our theorem it suffices to show that all conditions of Lemma 1 hold. First we show that our problem is strongly convex. It is easy to verify that matrix Γ given in Eq.(2.32) is always positive definite if $\gamma > n_c - 1$ which implies that Bregman divergence is always bounded from below w.r.t to λ and 2-norm $\|\cdot\|$. Since $f^{(t)}$ is a sum of λ-strongly convex function $\frac{\lambda}{2} w^T \Gamma w$ and another convex function (hinge-loss), it is also λ-strongly convex. Next by assuming $\mathcal{B} = \{w : \|w\| \leq \sqrt{n_c/\lambda(\gamma + n_c - 1)}\}$ and the fact that $\|x\| \leq R$ we can bound subgradient ∇_t. The explicit form for the subgradient evaluated at point x is given in Eq.(2.33). Using the triangular inequality and denoting 2-norm by $\|\cdot\|$ one obtains

$$\begin{gathered}\|\nabla_t\| \leq \lambda\|\Gamma w\| + \textstyle\sum_i \|B_i^T x\| \leq \lambda\|\Gamma\|\|w\| + n_c\|x\| \leq \\ \leq \lambda(\gamma + n_c - 1)\|w\| + n_c R \leq \sqrt{\lambda n_c(\gamma + n_c - 1)} + n_c R.\end{gathered}$$

The upper bound on $\|\Gamma\|$ is derived using the Gershgorin circle theorem as follows:

$$\|\Gamma\| \leq \sqrt{\upsilon_{max}(\Gamma^*\Gamma)} = \upsilon_{max}(\Gamma) \leq D(\gamma, n_c - 1) = \gamma + n_c - 1,$$

where Γ^* is the conjugate transpose of Γ, υ_{max} is the maximum eigenvalue and $D(\gamma, n_c - 1)$ is the Gershgorin circle with the center γ and radius $n_c - 1$. The first equality follows from the block-wise structure of matrix Γ. The last inequality follows from the fact that diagonal elements of Γ are the same and equal to γ everywhere and the sum of off-diagonal elements is exactly $n_c - 1$, which is clear from the structure of Γ in Eq.(2.32). Finally we have to show that $w^* \in \mathcal{B}$. To do so, we derive the dual form of our objective in terms of the dual variables $\alpha_i \in [0,1]^n, i \in \overline{1, n_c}$ related to decision functions f_{c_i} in Eq.(2.13) such that we have the following mixed optimization objective

$$\max_{\alpha_i} \min_{w} \frac{1}{m} \sum_{i=1}^{n_c} \|\alpha_i\|_1 - \frac{\lambda}{2} w^T \Gamma w$$

and after assuming strong duality and the optimal solution *w.r.t* the primal variable w^* and dual variables α_i^* one gets

$$\frac{\lambda}{2} w^{*T} \Gamma w^* + \frac{1}{m} \sum_{i=1}^{n_c} \sum_{x \in \mathcal{S}} \mathbb{L}(w^*; x) = -\frac{\lambda}{2} w^{*T} \Gamma w^* + \frac{1}{m} \sum_{i=1}^{n_c} \|\alpha_i^*\|_1.$$

For simplicity we replace the notation for the hinge-loss with $\mathbb{L}(w^*; x)$. Rearranging the above, using the non-negativity of the hinge-loss and applying the Gershgorin circle theorem we obtain our bound: $\|w\| \leq \sqrt{n_c/\lambda(\gamma + n_c - 1)}$. Now we can plug-in everything back to inequality in Lemma 1 which completes the proof. $\square$

Fixed-Size approach

One of the crucial aspects in estimating the support of some unknown high-dimensional distribution is the non-linearity of the feature space where we are trying to find a solution. As it was discussed in [103] we cannot rely on the linear kernel in this case and should use the RBF kernel instead. To overcome restrictions of Algorithm 1 which operates only in the primal space we apply a Fixed-Size approach [39] to approximate the RBF kernel with some higher dimensional explicit feature vector.

First we use an entropy based criterion to select the prototype vectors (small working sample of size $m \ll n$)[2] and construct kernel matrix K. Based on the Nyström approximation [129] an expression for the entries of the approximation of the feature map $\hat{\Phi}(x) : \mathbb{R}^d \to \mathbb{R}^m$, with $\hat{\Phi}(x) = (\hat{\Phi}_1(x), \ldots, \hat{\Phi}_m(x))^T$ is given by

$$\hat{\Phi}_i(x) = \frac{1}{\sqrt{\lambda_{i,m}}} \sum_{t=1}^{m} u_{ti,m} k(x_t, x),$$

where $\lambda_{i,m}$ and $u_{i,m}$ denote the i-th eigenvalue and the i-th eigenvector of K defined in Eq.(2.8). Using the above expression for $\hat{\Phi}(x)$ we can proceed with the original formulation of Algorithm 1 and find the solution of our problem in primal.

2.1.6 Algorithms and explanations

Coupling term and γ explained

To illustrate the importance of the coupling term $\langle w_i, w_j \rangle$ we implemented a toy example where initially the coefficient γ in Eq.(2.11, 2.21) is fixed and the other hyperparameters were obtained via the tuning procedure described in Section 2.2.4.

As we can see in Figure 2.4 the parameter γ directly affects the decision boundaries of the SND method as it increases from 1.1 in the topmost subfigure to 100 in the bottom one. To facilitate the reasoning of how γ value affects the coupling term and and the overall model consistency we provide each subfigure with the effective value of $\|w_1\|$, $\|w_2\|$ and $\cos\theta$ terms which are calculated *w.r.t.* our dual representation in Eq.(2.9) and the kernel expansion in Eq.(2.8) as

$$\cos\theta = \frac{\langle w_1, w_2 \rangle}{\|w_1\| \|w_2\|} = \frac{\alpha^T G \lambda}{\sqrt{(\alpha^T G \alpha)(\lambda^T G \lambda)}},$$

where $\|w_1\| = \sqrt{\alpha^T G \alpha}$, $\|w_2\| = \sqrt{\lambda^T G \lambda}$ and $G = K \circ yy^T$ relates to the matrix calculated from the training data. From examining Figure 2.4 one can observe that only carefully chosen parameter γ and a trade-off for $\langle w_1, w_2 \rangle$ term can

[2]see Section 4 of [39] for the details

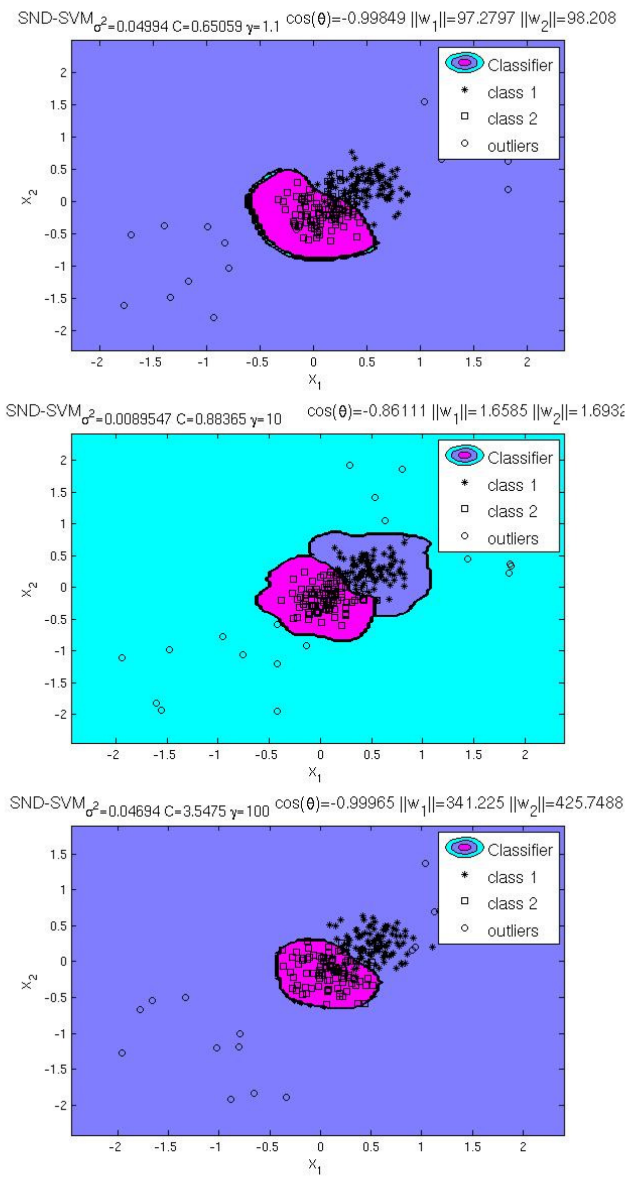


Figure 2.4: Decision boundaries of the SND method for varying values of the γ hyperparameter, illustrating the importance of small $cos\theta$ and minimized $\|w_1\|, \|w_2\|$.

bring necessary discrimination between classes while preserving the compactness of the support. This means that any over- or underestimation of γ parameter can lead to an unsatisfactory solution. The central subfigure of Figure 2.4 clearly indicates that a minimal $\cos\theta$ term doesn't ensure the best possible solution. This fact empirically illustrates our intuition and reasoning about the relation between the coupling term and margins as the top and bottom subfigures provide a good separation between classes but do not ensure the compact support for one of the distributions. We can see that $\|w_1\|$, $\|w_2\|$ are quite large (of 10^2 magnitude) and one of the classes almost completely covers the entire space.

Classification and novelty detection algorithms

In this section we present a general purpose algorithm for SND which can be applied both in classification and novelty detection settings.

To clarify how the SND method can be used in both settings: classification and novelty detection, we present a brief algorithmic summary for these settings in Algorithms 2–3. One should notice that the main difference between both algorithms is the cross-validation step, decision rule and the input data.

In the presented algorithms the "CrossvalidateSND" function stands for the tuning procedure which will be described in the next section. The crucial difference between Algorithm 2 and 3 is the usage of the data Z defined in Eq.(2.6). The SND model is tuned to perform novelty detection with respect to data Z and maximize the observed detection rate. In binary classification problem in Eq.(2.2) we cannot use data Z because of the labeling limitation on $y_i \in \{-1, 1\}$. We have to switch to the multi-class optimization objective in Eq.(2.11). Here we refer to Z as a matrix containing subset $\mathcal{Z} \subseteq \mathcal{X}$ which is labeled negatively everywhere, by taking $y_i = -1, i \in \overline{1, n_c}$. It can be used in the cross-validation procedure, such that we do care about maximizing detection rate of those samples along with minimization of the validation error for positively labeled samples. As a result of the "CrossvalidateSND" function we output the optimal parameters γ, C for the SND model and the optimal RBF kernel width σ. Finally $c(x)$ decision functions are defined by the means of the dual variables α_i, the primal variables ρ_i, the optimal parameters γ, σ and the labeling Y in Eq.(2.5) and Eq.(2.20). Here we can notice that for Algorithm 2 we are not giving any alternative decisions in $c(x)$ and are obliged to select between classes c_i.

Algorithm 2: SND for binary classification

```
input  : training data $X$ of size $l \times d$, class labels $Y$ of size $l \times n_c$
output : SND explicit decision rule
1 begin
2     $[\gamma, \sigma, C] \leftarrow$ CrossvalidateSND($X, Y$);
3     $[\alpha, \rho] \leftarrow$ ComputeSND($X, Y, \gamma, \sigma, C$);
4     $c(x) \leftarrow \text{argmax}_{c_i} f_{c_i}(x)$;
5 end
```

Algorithm 3: SND for novelty detection

```
input  : training data $X$ of size $l \times d$, outliers' data $Z$ of size $m \times d$, class
         labels $Y$ of size $l \times n_c$, $-1_z$ matrix of minus ones of size $m \times n_c$
output : SND explicit decision rule
1 begin
2     $[\gamma, \sigma, C] \leftarrow$ CrossvalidateSND($X, Y, Z, -1_z$);
3     $[\alpha, \rho] \leftarrow$ ComputeSND($[X; Z], [Y; -1_z], \gamma, \sigma, C$);
4     $c(x) \leftarrow \begin{cases} \text{argmax}_{c_i} f_{c_i}(x), & \text{if } \max_i f_{c_i}(x) > 0 \\ c_{out}, & \text{otherwise} \end{cases}$;
5 end
```

2.1.7 Empirical results

Experimental setup

In all our experiments for all tested SND and SVM models we use a 2-step procedure for tuning the parameters. This procedure consists of Coupled Simulated Annealing [133] initialized with 5 random sets of parameters for the first step and the simplex method [91] for the second step. After CSA converges to some local minima we select the tuple of parameters that attains the lowest error and start the simplex procedure to refine our selection. On every iteration step for CSA and simplex method we proceed with a 10-fold cross-validation. While being considerably faster than the straightforward grid search technique obtained parameters tend to vary more because of the randomness in initialization.

We selected the universal RBF kernel (see [114]) that is generally capable to separate all compact subsets and is suitable for many kinds of data. The choice of the RBF kernel was motivated by [103] where the authors explain an obvious advantage of it and that the data are always separable from the origin in the feature space (see Definition 1 in [103]). We tune the bandwidth of the RBF kernel in Eq.(2.8) with additional trade-off parameters for all methods using the tuning procedure described within the previous paragraph.

For the large-scale version of SND we use the Nyström approximation and the Fixed-Size approach [39] where the σ parameter was inferred via cross-validation procedure described above. The active subset was selected via maximization of the Renyi entropy. The size of this subset was set to be $\sqrt{n}$ for all methods that utilize Nyström approximation. Finally we fix the m parameter in Algorithm 1 to be $0.1|\mathcal{S}|$.

For the Toy Data (1) we performed 100 iterations with random sampling of size 100 according to the separate uniform distributions from intersecting intervals $[0, 1]$ and $[-0.5, 0.5]$, collected averaged error rates with corresponding standard deviations. For novelty detection we performed 100 iterations with random sampling from three different distributions[3] (see Figure 2.6) scaled to the range $[-1, 1]$ for all dimensions. For all toy datasets in every iteration we splitted all data points in proportion 80% to 20% into training and test counterparts. In novelty detection setting 15% of all data samples were generated as outliers. For all UCI datasets [49] (except for Arcene and large scale datasets) we used 5 independent 10-fold splittings and performed averaging and paired t-tests [38] for the comparison of errors. Arcene was split into training and validation datasets initially and we simply run the classification scheme 10 times. For the large scale datasets we run all methods 50 times with the random split in proportion of 50% by 50% for training and test data respectively. For the properties of UCI and toy datasets one can refer to the Table 2.1.

We implemented the original QP formulation of the SND method as an optimization problem using the Ipopt package (see [127]), which implements a general purpose interior point search algorithm. The Least-Squares version of SND was implemented using standard Matlab backslash operation. The large-scale version of SND and Pegasos were implemented in Matlab. LS-SVM with Fixed-Size approach is entirely implemented in Matlab as well. For learning C-SVM and One-Class SVM we used the LIBSVM package [27]. All experiments were run on Core i7 CPU with 8GB of RAM available under Linux CentOS platform.

Numerical results with UCI datasets

First we present some results for the classification setting where we can fairly compare our method to C-SVM [19] and LS-SVM [121]. Then we proceed

[3]Toy Data (2-4)

Table 2.1: Datasets

Dataset	# of attributes	# of classes	# of data points
Toy Data (1)	2	2	200
Toy Data (2-4)	2	2	150
Arcene	10000	2	900
Ionosphere	34	2	351
Parkinsons	23	2	197
Sonar	60	2	208
Zoo	17	7	101
Iris	4	3	150
Ecoli	8	5	336
TAE	5	3	151
Seeds	7	3	210
Arrhythmia	279	2	452
Pima	8	2	768
Madelon	500	2	2000
Red Wine	12	2	1599
White Wine	12	2	4898
Magic	11	2	19020

with some results for the large-scale UCI datasets. Then we continue with the novelty detection scheme in the presence of two and more classes and some number of outliers. Here we simply present preliminary results for different toy problems and report performance in terms of general test error and detection rate[4]. Finally in the next subsection we present real life example from anomalous change detection in AVIRIS (Airborne Visible/InfraRed Imaging Sensor) images [97].

Tables 2.2-2.3 present results for independent runs of QP and Least-Squares formulation of SND method in comparison to C-SVM and LS-SVM. All misclassification rates are collected on the identical test sets described in Section 2.2.4. Comparing the results in Tables 2.2-2.6 we can clearly observe that our method is quite comparable in terms of generalization error to C-SVM and LS-SVM. In Tables 2.5-2.6 we show p-values of a pairwise t-test which gives a clear evidence that generalization errors for SND and LS-SND are comparable to the corresponding values obtained for C-SVM and LS-SVM and there is no statistically significant difference in the mean values. However in

[4]we report the percentage of the detected outliers

Table 2.2: Averaged misclassification error on test data

Dataset	SND	C-SVM	LS-SVM
Toy Data (1)	0.1395± 0.097	0.1385± 0.078	**0.1325**± 0.085
Arcene	**0.1620**± 0.006	0.1730± 0.095	0.1810± 0.091
Ionosphere	0.0684± 0.043	0.0740± 0.031	**0.0483**± 0.030
Parkinsons	**0.0613**± 0.046	0.0721± 0.060	0.0621± 0.064
Sonar	**0.0962**± 0.069	0.1250± 0.105	0.1205± 0.101
Zoo	**0.0500**± 0.081	0.0733± 0.119	0.1071± 0.119
Iris	0.0467± 0.068	**0.0440**± 0.065	0.0493± 0.067
Ecoli	0.1263± 0.069	**0.1240**± 0.061	0.1562± 0.062
TAE	**0.4031**± 0.159	0.4346± 0.146	0.5545± 0.131
Seeds	0.0667± 0.060	**0.0650**± 0.050	0.0838± 0.073

Table 2.3: Averaged misclassification error on test data

Dataset	LS-SND	C-SVM	LS-SVM
Toy Data (1)	0.1425± 0.079	0.1450± 0.081	**0.1395**± 0.079
Ionosphere	0.0803± 0.033	0.0705± 0.044	**0.0541**± 0.034
Parkinsons	**0.0566**± 0.046	0.0664± 0.065	0.0647± 0.050
Sonar	0.1198± 0.059	**0.1173**± 0.074	0.1283± 0.054
Arrhythmia	**0.2193**± 0.050	0.2220± 0.050	0.2286± 0.061
Pima	0.2325± 0.039	**0.2308**± 0.043	0.2391± 0.039
Zoo	0.1487± 0.145	**0.0671**± 0.079	0.1518± 0.109
Iris	0.0667± 0.070	0.0427± 0.060	**0.0347**± 0.043
Ecoli	0.1586± 0.084	**0.1192**± 0.044	0.1376± 0.040
TAE	**0.4219**± 0.110	0.4300± 0.141	0.5655± 0.116
Seeds	0.0905± 0.063	**0.0629**± 0.049	0.0905± 0.063

Table 2.3 we can see that LS-SND algorithm almost in all cases is superior to LS-SVM and obtains lower generalization errors. In general we can observe better performance from QP versions of SVM but this can be easily explained by properties of hinge-loss which better deals with the outliers. The latter disadvantage can be easily handled with a weighted formulation of LS-SVM [119].

For the second part of our numerical experiments we applied a large-scale

Table 2.4: Averaged misclassification error on test data

Dataset	SND	Pegasos	NyFS-LSSVM
Pima	0.2885± 0.024	0.2866± 0.020	0.2333± 0.020
Madelon	0.4307± 0.022	0.4272± 0.017	0.4531± 0.014
Red Wine	0.2648± 0.016	0.2625± 0.014	0.2583± 0.014
White Wine	0.2747± 0.021	0.2715± 0.014	0.2381± 0.008
Magic	0.1474± 0.012	0.1576± 0.004	0.1375± 0.003

Table 2.5: P-values of a pairwise t-test on generalization error between SND and other methods

Dataset	to C-SVM	to LS-SVM
Toy Data (1)	0.87329	0.63883
Arcene	0.71842	0.52162
Ionosphere	0.73986	0.24175
Parkinsons	0.65938	0.97501
Sonar	0.47715	0.53844
Zoo	0.25673	0.011471
Iris	0.84167	0.84356
Ecoli	0.85788	0.02481
TAE	0.30483	1.9013e-09
Seeds	0.86329	0.20278

modification of the SND algorithm to five large UCI datasets and collected corresponding misclassification errors. Table 2.4 presents these results and we can see that almost everywhere NyFS-LSSVM [84] (Nyström Fixed-Size LS-SVM) method achieves better performance than SND or Pegasos algorithms. This can be simply addressed by the nature of NyFS-LSSVM method, which is an exact algorithm while Algorithm 1 and Pegasos are approximate algorithms. On the other hand SND and Pegasos are very similar in the achieved results but for the largest Magic dataset SND surprisingly achieves better performance with very high statistical significance (see Table 2.7). One of the major advantages of Pegasos-based algorithms is the price of every iteration/training which can be controlled by m parameter in Algorithm 1. The example of novelty detection problem solved by this large-scale algorithm one can observe in Figure 2.5. Table 2.8 represents a pivot table of the effective values for the l_2-norms

Table 2.6: P-values of a pairwise t-test on generalization error between LS-SND and other methods

Dataset	to C-SVM	to LS-SVM
Toy Data (1)	0.8265	0.79085
Ionosphere	0.2189	0.00016358
Parkinsons	0.33084	0.40091
Sonar	0.8537	0.44872
Pima	0.82858	0.40384
Sonar	0.8537	0.44872
Zoo	0.0006965	0.90418
Iris	0.007038	0.068409
Ecoli	0.0039443	0.11129
TAE	0.75031	6.5273e-09
Seeds	0.18541	1

Table 2.7: P-values of a pairwise t-test on generalization error between large-scale Pegasos-based SND and other methods

Dataset	to Pegasos	to NyFS-LSSVM
Pima	0.66776	9.5771e-22
Madelon	0.37543	1.4418e-08
Red Wine	0.45226	0.032591
White Wine	0.37445	9.4174e-20
Magic	9.3029e-08	1.0061e-07

Table 2.8: Effective values of the l_2-norms and the $cos\theta$ value between the corresponding normal vectors in Figure 2.5

Classes (c_i - c_j)	$\cos\theta$	norms ($\|w_i\|$, $\|w_j\|$)
c_1 - c_2	-0.3795	(0.5113, 0.4928)
c_1 - c_3	-0.4812	(0.5113, 0.5174)
c_2 - c_3	-0.4034	(0.4928, 0.5174)

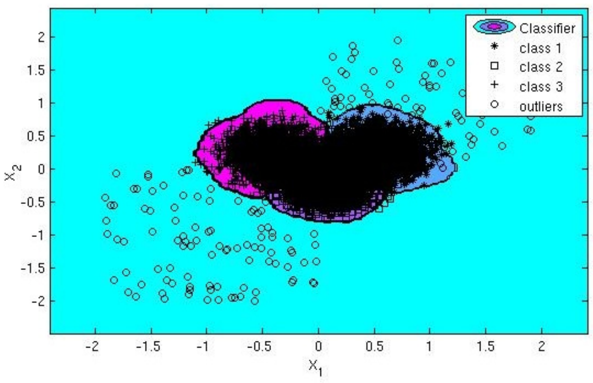


Figure 2.5: Pegasos-based SND method in a novelty detection scheme with 3 classes. Size of the toy dataset is 9200.

Table 2.9: Averaged misclassification error / (detection rate) for SND and One-Class SVM

Dataset	SND	One-Class SVM
Toy Data (2)	**0.0083** / (0.9746)	0.0233 / (1)
Toy Data (3)	**0.0113** / (1)	0.0233 / (1)
Toy Data (4)	**0.0366** / (0.8182)	0.0791 / (0.7808)

and the $cos\theta$ value between the corresponding normal vectors and decision boundaries (hyperplanes in the feature space) in Figure 2.5. This information helps us to understand the connection in a large-scale setting between the pairwise discrimination of classes and the corresponding compact support of the distributions from which these classes are drawn.

For the third part of our numerical experiments we have chosen to apply SND in an anomaly detection scheme in the presence of 2 or more classes. In this setting we cannot fairly compare our method to other SVM-based algorithms because of an obvious novelty of our problem. So we restrict ourselves to evaluating the SND algorithm for our 3 toy datasets and comparing it to One-Class SVM in terms of total misclassification error (assuming binary setting: non-outliers vs. outliers) and detection rate of outliers. From the Table 2.9 we can clearly conclude that SND provides better support for underlying distributions and gives comparable or even better detection rates. One can also observe decision boundaries of the SND method for several random runs on different toy problems

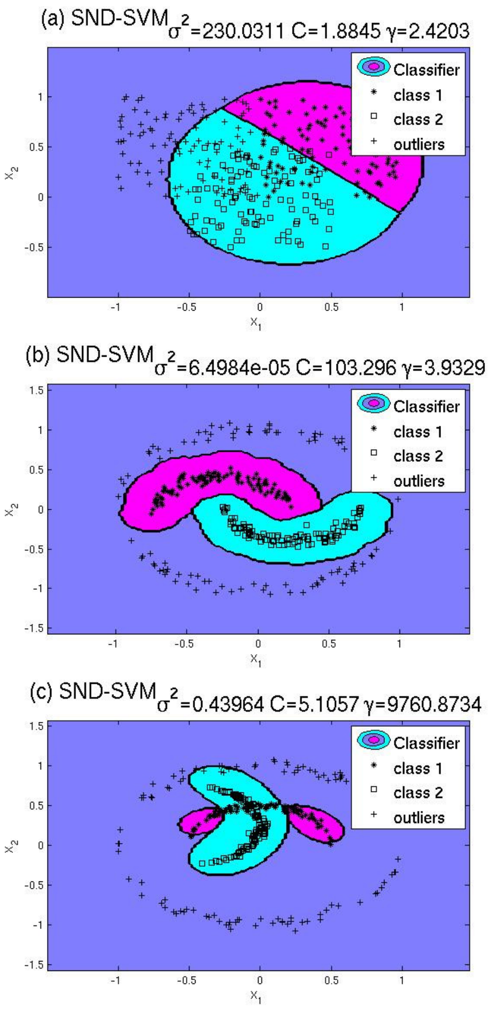


Figure 2.6: SND method in a novelty detection scheme with 2 classes. Subfigures (a) through (c) represent SND boundaries in the presence of outliers (+) and correspond to Toy Data (2) through (4).

(Toy Data (2-4)) in Figure 2.6. The latter figure provides a better view on SND properties and output decision boundaries in the presence of the scattered outliers. In Figures 2.7 and 2.8 we can see a comparison of the SND approach with One-Class SVM. In Figure 2.7 we use for One-Class training all data points available in both classes while in Figure 2.8 we try to find the support for each

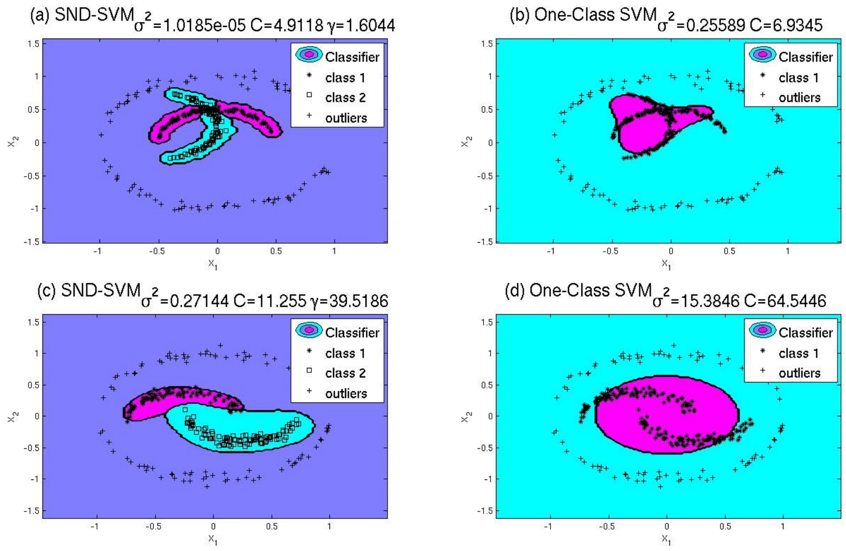


Figure 2.7: Comparison of SND (a,c) and One-Class SVM (b,d) in the novelty detection scheme.

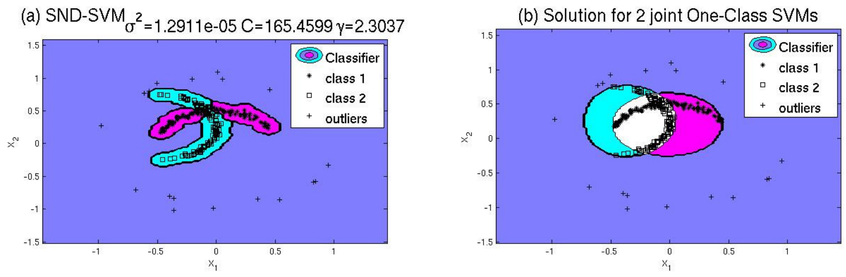


Figure 2.8: Comparison of SND (a) and two joint One-Class SVMs (b) in the novelty detection scheme showing a clear improvement of SND. White region depicts the area which belongs to the support of both One-Class SVMs simultaneously.

class/distribution separately. Here by the white color we denote intersecting regions of two separate One-Class SVM estimators. However One-Class SVM is able to capture many data points by the underlying support it still far from the correct density estimation.

Analyzing these figures one can clearly observe the importance of labeling to capture the different underlying distributions in the data. One of the key advantages of the SND approach is a better understanding and modelling of the support for a mixture of distributions where one possesses a certain amount of information about each distribution.

Real life example

To justify the practical importance of our method we applied the SND Algorithm 1 in the context of AVIRIS data (Airborne Visible/InfraRed Imaging Sensor) [97]. We took one of the high definition greyscale images and extracted two disjoint sub-images of sizes 205×236 and 283×281 pixels respectively. The first sub-image was used for training the SND algorithm while the second one for test purposes.

We extracted for every pixel its intensity and averaged intensity of the window of size 10×10 of surrounding pixels excluding the nearest 5×5 pixels. Finally we took these values along with pixel intensities as our 2-dimensional training/test datasets. We separated the training image by the average white color intensity of the mentioned window across all pixels. Finally we defined outliers as the white spots on the darker greyscale region[5] by taking pixels belonging to that segment of the processed image with intensities grater than 190. The setting is artificial but it will help us to evaluate our approach *w.r.t.* real life data.

We applied Algorithm 3 to the final training data of size 48380 and determined σ parameter of the RBF kernel, λ and γ parameters of Algorithm 1 using 10-fold cross-validation on training data as described in Section 2.2.4. On every step of Algorithm 3 the SND model was calculated via Algorithm 1 and non-linearity of the model was achieved applying the Fixed-Size approach described in Section 10.

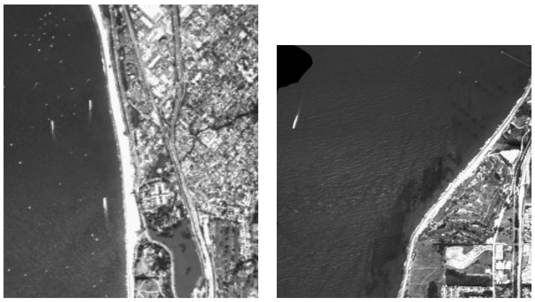


Figure 2.9: AVIRIS training (top) and test (bottom) images.

[5]these spots correspond to the tracks remained after the transition of the fast boats

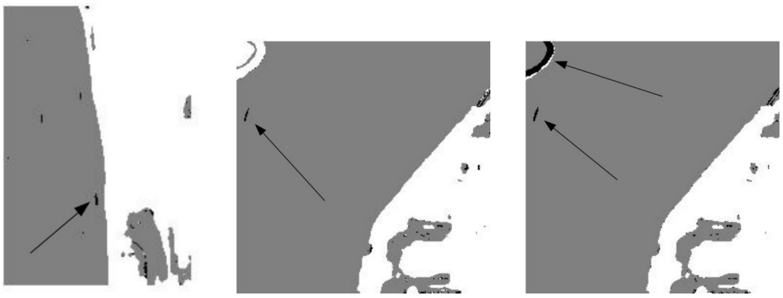
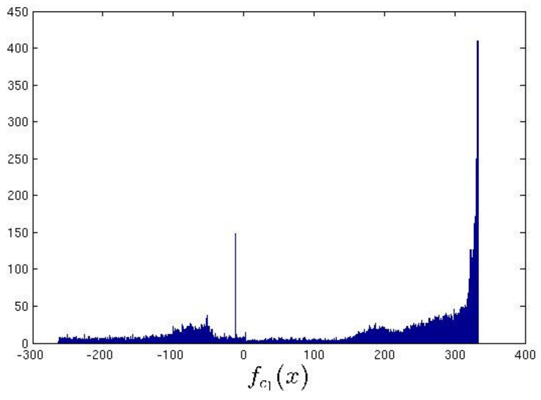


Figure 2.10: AVIRIS training image after preprocessing (left) and test image after evaluation by the SND algorithm (middle) and the Pegasos algorithm (right) with pointed out outliers.

In Figure 2.9 we can see these AVIRIS images while in Figure 2.10 we notice the same images but after the segregation to different terrains and detection of outliers by the SND and Pegasos[6] algorithms. As we can see our method is capable of good image segregation while being able to detect anomalous spots in the test image[7]. Both methods were able to detect outliers denoting pixels of interest[8] while Pegasos was much less accurate in estimating the densities of two classes and resulted in the increased number of the detected outliers[9]. These results can be extended to anomalous change detection when we consider the problem of finding anomalous changes in the obtained scenes of the same image.

In Figure 2.11 we can observe two histograms corresponding to the different decision functions obtained by SND Algorithm 3 which was evaluated on AVIRIS test image. Topmost image corresponds to the function which outputs positive values for the marine region and the bottom one outputs positive values for the land views. Analyzing these figures we can clearly notice some revealing patterns and distributions of output values. For instance in the images we can see two major peaks which obviously correspond to two classes. In general outliers are not concentrated as there are no intersecting peaks on both histograms. This fact corresponds to the intuition of [116] and validates the usefulness of the SND approach.

[6]we trained 2 Pegasos-based classifiers *w.r.t.* each class
[7]black pixels pointed by arrows in Figure 2.10
[8]big fast-boat transition track
[9]222 for SND and 507 for Pegasos



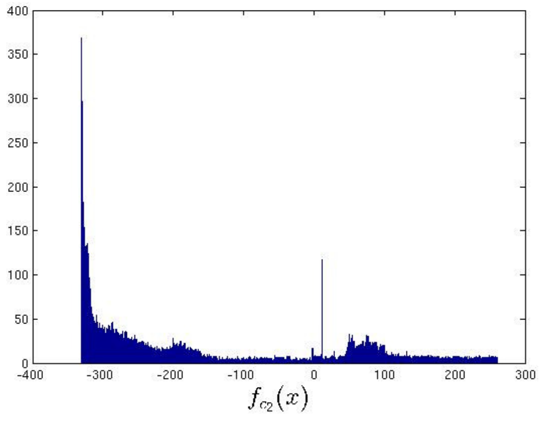


Figure 2.11: Histograms of the output values for two decision functions (see Eq.(2.13)) obtained by SND Algorithm 3 and evaluated on AVIRIS test image. Top image corresponds to the function which outputs positive values for the marine region and the bottom one outputs positive values for the land views.

2.1.8 Conclusion and future work

In this paper we approached the novelty detection problem and estimation of the support for a high-dimensional distribution from the new perspective of multi-class classification. This setting is mainly designed for finding outliers in the presence of several classes while being valuable as a general purpose classifier as well. The SND setting can be potentially extended for a semi-supervised case with and an intrinsic norm [14] applied in conjunction with coupling terms (see Eq.(2.11)). The latter formulation implies that we need only few labeled data points to approximate the coupling term fairly well and the other data can be involved in the manifold learning. We consider the latter approach as a promising extension of our method for future work. We demonstrated that the performance and obtained generalization errors are comparable or even less than for other SVMs. The experimental results verify the usefulness of our approach for both settings: classification and novelty detection.

2.2 New Bilinear Formulation to Semi-Supervised Classification Based on Kernel Spectral Clustering

2.2.1 Introduction

For many decades classification and Semi-Supervised Learning (SSL) were among the most popular machine learning topics [102, 121]. The substantial interest is driven by the simple and very important observation: the amount of unlabeled data is instantly growing while resources for labeling and preprocessing are limited and not always easily accessible. Many existing semi-supervised approaches [105, 138] aim at utilizing either labeled or unlabeled data. While both strategies have numerous drawbacks it was recently stressed [14] that preserving them jointly might be important for obtaining good generalization in the presence of major unlabeled counterpart.

Laplacian Support Vector Machine (LapSVM) [14, 88] is one of the state-of-the-art techniques which addresses this problem and provides a natural out-of-sample extension for unseen unlabeled data. It is based on implicit manifold regularization introduced by the normalized Laplacian matrix [111]. Another possible approach is related to the Spectral Clustering techniques [95, 36, 5] which make use of the eigenspectrum of the Laplacian matrix to divide a dataset into the clusters such that points within the same cluster are similar and points from other clusters are dissimilar to each other. However though these techniques are of great interest for machine learning practitioners they often lack proper model selection and verification procedures in semi-supervised setting and cannot be applied blindly. Recently a kernel-based extension for semi-supervised learning was proposed known as Semi-Supervised Kernel Spectral Clustering [4, 85]. The main idea of the latter approach is to formulate the semi-supervised learning problem as a combination of a weighted kernelized Principal Component Analysis (PCA) and minimization of an empirical error associated with the labeled counterpart.

For the roots of the bilinear term minimization which we utilize in our approach we refer the interested reader to the recently proposed Supervised Novelty Detection (SND) approach [64, 65]. SND can be effectively used for multi-class classification and it supplements the class of SVM-based algorithms intended for the detection of outliers in the presence of several classes (distributions). We should mention that the SND approach is related to other non-parallel SVM classifiers, like Twin SVM (TWSVM), GEPSVM [61, 110, 72] or Non-Parallel Semi-Supervised KSC [87] etc., but it has one crucial difference in its primal formulation, i.e. a bilinear term which couples two non-parallel classifiers. In this paper we extend ideas given by [65] to the semi-supervised setting and show that it can be quite beneficial to couple non-parallel classifiers in the presence of unlabeled information. We compare our results with the aforementioned approaches and present some convincing and challenging toy-problems where we are violating a commonly referenced low-density assumption (between clusters) and performing better than other approaches.

The remainder of this paper is structured as follows. Section 2.2.2 outlines some existing semi-supervised learning approaches. Section 2.2.3 gives an overview of our method and the resulting optimization problem. Section 2.2.4 provides the experimental setup and results, while Section 2.2.5 discusses some important issues and further directions of our research. Section 2.2.6 concludes the paper.

2.2.2 Semi-supervised learning

In this section we present a brief outlook on the existing state-of-the-art methods in semi-supervised learning. But first we present some preliminary notations which will be used across this paper.

Preliminaries

We first introduce terminology and some essential conventions. We consider training data with the corresponding partial labeling given as a set of pairs

$$(x_1, y_1), \ldots, (x_l, y_l), x_i \in \mathcal{X}, y_i \in \{-1, 1\},$$

where l is the number of corresponding labeled observations in the set $\mathcal{X}$ of size m. The unlabeled part is given by the set $\{x_{l+1}, \ldots, x_m\} = \mathcal{X}_u \in \mathcal{X}$. For simplicity we think of all our data as a compact subset of $\mathbb{R}^d$. Then define φ to be a feature map $\varphi : \mathbb{R}^d \to \mathbb{R}^h$ which is a mapping to a high-dimensional feature space of dimension h with the connection to a positive definite kernel [102]

$$k(x, y) = \langle \varphi(x), \varphi(y) \rangle, \tag{2.34}$$

where by using a kernel trick [102, 121] one can define a kernel matrix Ω as $\Omega_{ij} = k(x_i, x_j)$.

Further in the paper index i spans the range $\overline{1,m}$ if it is not declared explicitly. Greek letters α, λ without indices denote m-dimensional vectors.

Laplacian SVM [14]

The solution of the LapSVM problem proposed by Belkin *et al.* [14] is based on manifold learning. This problem explicitly incorporates the kernel matrix Ω and the Laplacian matrix L as follows:

$$\min_{\alpha\in\mathbb{R}^m;\xi\in\mathbb{R}^l} \gamma_A \alpha^T \Omega \alpha + \gamma_I \alpha^T \Omega L \Omega \alpha + \sum_{i=1}^{l} \xi_l \tag{2.35}$$

$$\begin{aligned} \text{s.t.} \quad y_i(\textstyle\sum_{j=1}^m \alpha_j k(x_j, x_i) + b) \geq 1 - \xi_i, &\quad i = 1, \ldots, l \\ \xi_i \geq 0, &\quad i = 1, \ldots, l \end{aligned} \tag{2.36}$$

This formulation directly implies regularization over two different terms and minimization of the empirical error ξ_i. The first term is establishing a regularization framework for function learning, given a kernel function $k(\cdot,\cdot)$, its associated Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}_k$ of functions $X \to \mathbb{R}$ with the corresponding norm $\|\cdot\|_A$ and a trade-off hyperparameter γ_A. The second term stands for the implicit manifold regularization with the corresponding norm $\|f\|_I = f^T L f$ defined as the Laplace-Beltrami operator [13] applied to the function f.

Semi-supervised KSC [4]

The following formulation to the semi-supervised learning problem is very close to the presented one in the next section. We provide it for the interested reader as a reference point in understanding our own extension to it. The proposed primal problem of Semi-Supervised Kernel Spectral Clustering (SemiKSC) in [4] is given by:

$$\begin{aligned} \min_{w\in\mathbb{R}^d;e\in\mathbb{R}^m} \quad & \tfrac{1}{2}\|w\|^2 + \tfrac{\rho}{2}\textstyle\sum_{i=1}^{l}(e_i - y_i)^2 - \tfrac{\gamma}{2} e^T V e \\ \text{s.t.} \quad & \Phi w + b 1_M = e, \end{aligned} \tag{2.37}$$

where $\Phi = [\varphi(x_1)^T; \ldots; \varphi(x_m)^T] \in \mathbb{R}^{m\times h}$ is a stacked feature map for all samples in $\mathcal{X}$, $e \in \mathbb{R}^m$ is the projection, b is the bias, V is the weighting diagonal matrix while γ and ρ are trade-off hyperparameters.

After eliminating primal variables and converting the above problem into a linear system of equations we obtain the following problem in terms of the dual variables $\alpha \in \mathbb{R}^m$:

$$(I_M - (\gamma D^{-1} - \rho A) M_S \Omega)\alpha = \rho M_S^T y, \tag{2.38}$$

where I_M is $m \times m$ identity matrix and $M_S \in \mathbb{R}^{m\times m}$ is a centering matrix defined as:

$$M_S = I_M - \frac{1}{c} 1_M 1_M^T (\gamma D^{-1} - \rho A)$$

with

$$A = \left[\begin{array}{c|c} 0_{p\times p} & 0_{p\times l} \\ \hline 0_{l\times p} & I_{l\times l} \end{array}\right] \in \mathbb{R}^{m\times m}$$

and $p = m - l$, $c = 1_M^T(\gamma D^{-1} - \rho A)1_M$. If we omit bias term b in the latent clustering model for training points, we can now write it in terms of the dual variables α:

$$e = M_S \Omega \alpha - \frac{\rho}{c} 1_M 1_M^T y.$$

This model defines the binary cluster membership by sign(e).

2.2.3 Bilinear Non-Parallel Kernel Spectral Semi-Supervised Learning

In this section we present an optimization problem in the primal form. Further we derive a dual formulation with respect to our constraints. Finally using Karush-Kuhn-Tucker (KKT) optimality conditions we boil-down the problem into a linear system of equations which can be efficiently solved without introducing extra costs.

Optimization problem

We start with a primal formulation for a simple binary semi-supervised classification problem where we introduce an additional bilinear term $\langle w_1, w_2\rangle$ between classifiers:

$$\begin{aligned} \min_{w_1,w_2\in\mathbb{R}^d; e_1,e_2\in\mathbb{R}^m} \quad & \tfrac{\gamma_1}{2}(\|w_1\|^2 + \|w_2\|^2) + \langle w_1, w_2\rangle \\ & + \tfrac{\gamma_2}{2} \textstyle\sum_{i=1}^{l} [(e_{1i} - y_i)^2 + (e_{2i} + y_i)^2] \\ & - \tfrac{\gamma_3}{2} e_1^T V e_1 - \tfrac{\gamma_4}{2} e_2^T V e_2 \end{aligned} \tag{2.39}$$

$$\begin{aligned} \text{s.t.} \quad & \Phi w_1 = e_1 \\ & \Phi w_2 = e_2 \end{aligned} \tag{2.40}$$

where $w_1, w_2 \in \mathbb{R}^h$ are the hyperplanes associated with each non-parallel classifier, $\Phi = [\varphi(x_1)^T; \ldots; \varphi(x_m)^T] \in \mathbb{R}^{m \times h}$ is a stacked feature map for all samples in $\mathcal{X}$, $e_1, e_2 \in \mathbb{R}^m$ are the projections, V is the weighting diagonal matrix and γ_1 through γ_4 denote trade-off hyperparameters. For the simplicity and because of the flexibility of RBF-kernel which is often used in the KSC-based models we omit in our formulation a bias term b.

In the primal optimization objective we have different terms which are related to either supervised or unsupervised counterpart. Terms $(e_{1i} - y_i)^2$ and $(e_{2i} + y_i)^2$ are trying to minimize empirical error *w.r.t.* each non-parallel classifier and provided set of labels y. Terms $e_1^T V e_1$ and $e_2^T V e_2$ are related to the KSC model [5] and are trying to explain (maximize) variance *w.r.t.* involved classes.

Before proceeding to the formulation of the dual problem we explain the importance of the weighting matrix V. It can be shown that if we take V as an inverse of a degree matrix[10]

$$V = D^{-1} = \text{diag}(\frac{1}{d_1}, \ldots, \frac{1}{d_m}),$$

where $d_i = \sum_j k(x_i, x_j)$, the overall problem can be related to the random walk algorithms in spectral clustering [95, 36].

The final decision function is

$$c(x) = \begin{cases} \text{argmax}_i \ \langle w_i, \varphi(x) \rangle, & \text{if} \quad \max_i \langle w_i, \varphi(x) \rangle > 0, \\ c_{out}, & \text{otherwise,} \end{cases} \tag{2.41}$$

where $i \in \{1, 2\}$ and c_{out} is representing the outliers' class. This decision rule can be advantageous when one tries to estimate a high-dimensional support of the underlying distributions (per class). This specific setting is discussed in detail in [65].

[10]we assume degree of i-th data point in a neighborhood graph

Dual formulation

Using α and λ vectors as Lagrange multipliers we introduce the following dual optimization problem

$$\begin{aligned}\mathcal{L}(w_1, w_2, e_1, e_2, \alpha, \lambda) = & \tfrac{\gamma}{2}(\|w_1\|^2 + \|w_2\|^2) + \langle w_1, w_2 \rangle \\ & + \tfrac{\gamma_2}{2} \textstyle\sum_{i=1}^{l} [(e_{1i} - y_i)^2 + (e_{2i} + y_i)^2] \\ & - \tfrac{\gamma_3}{2} e_1^T V e_1 - \tfrac{\gamma_4}{2} e_2^T V e_2 \\ & + \alpha^T (e_1 - \Phi w_1) + \lambda^T (e_2 - \Phi w_2).\end{aligned} \tag{2.42}$$

By setting the derivatives of the dual problem with respect to primal and dual variables to zero we obtain the following Karush-Kuhn-Tucker (KKT) optimality conditions:

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial \alpha} = 0 \rightarrow e_1 - \Phi w_1 = 0, \\ \frac{\partial \mathcal{L}}{\partial \lambda} = 0 \rightarrow e_2 - \Phi w_2 = 0, \\ \frac{\partial \mathcal{L}}{\partial w_1} = 0 \rightarrow w_1 = \frac{1}{1-\gamma_1^2} \Phi^T (\lambda - \gamma_1 \alpha), \\ \frac{\partial \mathcal{L}}{\partial w_2} = 0 \rightarrow w_2 = \frac{1}{1-\gamma_1^2} \Phi^T (\alpha - \gamma_1 \lambda), \\ \frac{\partial \mathcal{L}}{\partial e_1} = 0 \rightarrow \alpha - \gamma_3 V e_1 + \gamma_2 G (e_1 - y^*) = 0, \\ \frac{\partial \mathcal{L}}{\partial e_2} = 0 \rightarrow \lambda - \gamma_4 V e_2 + \gamma_2 G (e_2 + y^*) = 0, \end{cases} \tag{2.43}$$

where defining $p = m - l$ as an auxiliary indexing upper bound G is given by

$$G = \left[\begin{array}{c|c} I_{l \times l} & 0_{l \times p} \\ \hline 0_{p \times l} & 0_{p \times p} \end{array}\right] \in \mathbb{R}^{m \times m}$$

and $y^* = [y_1, \ldots, y_l, 0, \ldots, 0]^T \in \mathbb{R}^m$, while $I_{l \times l}$ is an identity matrix.

Linear system

One can observe that our primal problem is non-convex but after eliminating the primal variables w_1, w_2 and e_1, e_2 we can directly work with the dual objective which is always concave in terms of the dual variables. Despite this fact it is more convenient to work with a linear system which we can obtain by plugging substituted primal variables back into the last two optimality conditions of Eq.(2.43) and deriving the following linear system of equations

$$\left[\begin{array}{c|c} A_d & A_{\text{off}} \\ \hline L_{\text{off}} & L_d \end{array}\right] \left[\begin{array}{c} \alpha \\ \hline \lambda \end{array}\right] = \gamma_2 \left[\begin{array}{c} y^* \\ \hline -y^* \end{array}\right], \tag{2.44}$$

where $A_d = \gamma_3\gamma_1\gamma_1'\Omega D^{-1} - \gamma_2\gamma_1\gamma_1' G + I$, $A_{\text{off}} = \gamma_2\gamma_1' G - \gamma_3\gamma_1'\Omega D^{-1}$, $L_d = \gamma_4\gamma_1\gamma_1'\Omega D^{-1} - \gamma_2\gamma_1\gamma_1' G + I$, $L_{\text{off}} = \gamma_2\gamma_1'\Omega - \gamma_4\gamma_1' G D^{-1}$, $\gamma_1' = 1/(1-\gamma_1^2)$, Ω is a full-rank kernel matrix and G, D^{-1} were given in the previous subsection, while I is an identity matrix of size $m \times m$.

By solving this linear system of equations we obtain key components of our final decision function defined in Eq.(2.41). If we take a look again at KKT conditions in Eq.(2.43) we can notice that we have a closed form solution for w_1 and w_2 in terms of our dual variables α and λ. By plugging these variables and taking into account definition of $\langle\varphi(x), \varphi(y)\rangle$ in Eq.(2.34) we obtain a kernel expansion for the decision function as

$$c(x) = \begin{cases} \text{argmax}_i \ f_i(x), & \text{if} \quad \max_i f_i(x) > 0 \\ c_{out}, & \text{otherwise}, \end{cases} \tag{2.45}$$

where $f_i(x)$ functions are defined as follows

$$\begin{aligned} f_1(x) &= \tfrac{1}{1-\gamma_1^2} \textstyle\sum_{i=1}^m k(x, x_i)(\lambda_i - \gamma_1\alpha_i), \\ f_2(x) &= \tfrac{1}{1-\gamma_1^2} \textstyle\sum_{i=1}^m k(x, x_i)(\alpha_i - \gamma_1\lambda_i). \end{aligned} \tag{2.46}$$

2.2.4 Experiments

Experimental setup

In all our experiments we tested all semi-supervised kernel-based models using a 2-step procedure for tuning the hyperparameters. This procedure consists of the Coupled Simulated Annealing [133] initialized with 5 and up to 30 random sets of parameters for the first step and the simplex method [91] for the second step. After CSA converges to some local minima we select the tuple of parameters that attains the lowest error and start the simplex procedure to refine our selection. On every iteration step for CSA and simplex method we proceed with a 5-fold cross-validation. Cross-validation is performed only with respect to the labeled information and classification accuracy. We do not impose any additional model selection criteria at this step, because most of them, like Silhouette index, Fisher index or Davies-Bouldin index [16], according to our empirical observations are resulting in worse generalization capabilities and might not reflect properly the non-linearity of clustering in some higher dimensional spaces and manifolds. In our experiments we use RBF-kernel related to Eq.(2.34) and given by

$$k(x,y) = \langle \varphi(x), \varphi(y) \rangle = e^{-\frac{\|x-y\|^2}{2\sigma^2}}.$$

We use the aforementioned cross-validation tuning procedure to estimate kernel bandwidth σ together with other trade-off hyperparameters.

In our toy examples we experiment with two challenging problems when data are drawn from the mixture of overlapping distributions. In this case we do violate the low-density assumption between clusters and semi-supervised algorithms based upon spectral clustering techniques and normalized cuts [111] might fail in this case. We artificially created two datasets, namely "spirals" and "half-moons". In the first one we have two spirals which are separable in the beginning and are merging in the end. For the second dataset we have two banana-shaped distributions which are overlapping at their tails. Additionally to the overlapping scenarios we present the results for the noiseless case where classes are clearly separable. In every dataset only 20% of points were labeled randomly.

For the numerical benchmark tests we took some recognized datasets in semi-supervised learning mentioned or created for [30]. Each of these datasets[11] is provided in 12 splits where each split is represented by the labeled points and the remaining unlabeled counterpart. We perform cross-validation using entire data (only labeled information is considered) for the split and then report the average misclassification rate for all datapoints as we are provided with the ground truth for them. We average the results across all 12 splits for each dataset. This experimental setup is used in the original book by Chapelle *et al.* [30] and is considered in other research papers as well [4, 87]. For our experiments we use the splits with 100 labeled datapoints only. We perform additional experiments on public UCI datasets [49] where we randomly select 100 points to remain labeled. We run all algorithms for UCI datasets 50 times and report average misclassification rate evaluating the methods on the entire dataset, where we know the ground truth for the unlabeled points. All datasets were normalized by mapping each dimension to zero mean and one standard deviation as $(\mu_i, \sigma_i) = (0,1), i \in \overline{1,d}$. For the properties of UCI and SSL benchmark datasets used in this paper one can refer to the Table 2.10.

We implemented Eq.(2.44) in MATLAB using the backslash operator. Because we are not using a bias term b in our method we decided to omit this term in the implementation of SemiKSC [4] approach as well, which would give us a

[11]i.e. BCI, Text, g241c and g241d

Table 2.10: Datasets

Dataset	# of attributes	# of classes	# of data points
Ionosphere	34	2	351
Parkinsons	23	2	197
Sonar	60	2	208
Ecoli	8	2	336
Arrhythmia	279	2	452
BCI	117	2	400
Text	11960	2	1500
g241c	241	2	1500
g241d	241	2	1500

fair comparison with the former. All experiments were run on Core i7 CPU with 8GB of available RAM.

Toy problems

In this subsection we present the results obtained on two artificial toy-problems. In Figure 2.12 we can see the classification boundaries of three methods applied to the "half-moons" problem. Figure 2.13 corresponds to the "spirals" dataset. In the noiseless case[12] all algorithms are performing reasonably good but SemiKSC approach is slightly inferior *w.r.t.* the other methods. Comparing Figures 2.12 - 2.13 (a-c) for the overlapping case we can clearly observe that Bilinear Semi-Supervised KSC (Bi-SemiKSC) is capturing the underlying manifolds better resulting in more precise and smoother classification boundaries. Although only 20% of datapoints are labeled in these examples Bi-SemiKSC method was able to converge to an acceptable solution while LapSVM is failing to generalize on Figures 2.12 - 2.13 (a-c). While for the "half-moons" problem original Semi-Supervised KSC (SemiKSC) is giving an acceptable result, for "spirals" problem it is obviously overfitting.

Numerical experiments

In this subsection we are analyzing the performance of four algorithms on the benchmark UCI and SSL datasets. We are comparing our Bilinear Semi-

[12]Figures 2.12 - 2.13 (d-f)

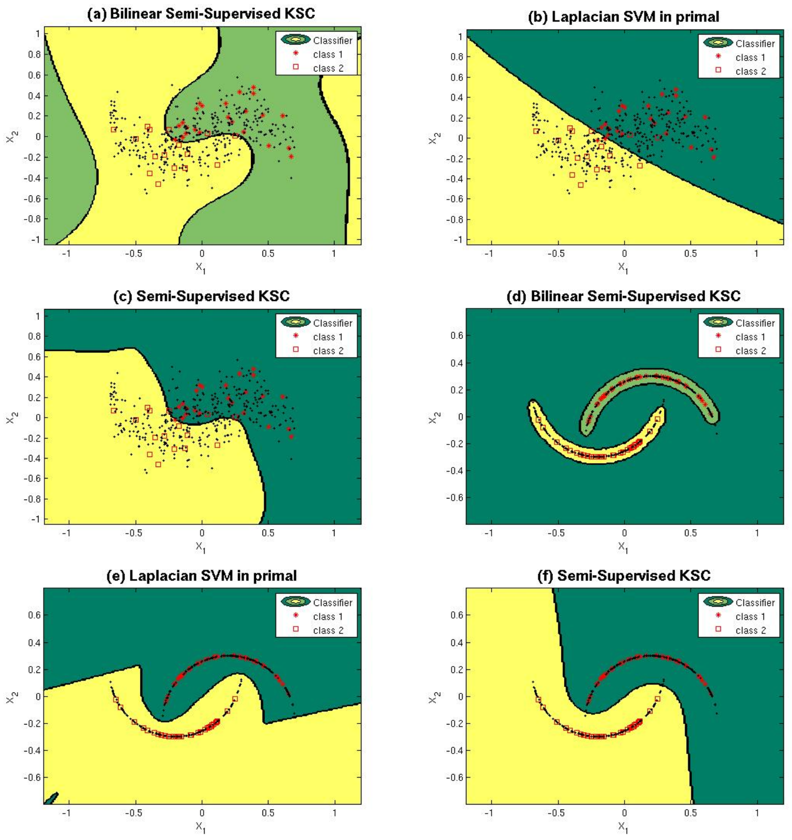


Figure 2.12: Different approaches applied to the "half-moons" problem. From left to right: (a) Bilinear Semi-Supervised KSC, (b) Laplacian SVM in primal, (c) Semi-Supervised KSC in an overlapping scenario. Bottom from left to right: (d) Bilinear Semi-Supervised KSC, (e) Laplacian SVM in primal, (f) Semi-Supervised KSC in a noiseless scenario. With small black dots we denote unlabeled datapoints. Bigger red stars and squares represent labeled samples from two classes.

Supervised KSC (Bi-SemiKSC) approach with LapSVM in primal (LapSVMp) [88], LapSVM in dual and the original Semi-Supervised KSC (SemiKSC) [4, 85]. As we can see from Tables 2.11 and 2.12 our approach is performing much better and the classification accuracy evaluated by the ground truth on the entire data suggests that Bi-SemiKSC with the help of an additional coupling term is able to find an appropriate embedding of data where classification is the most accurate.

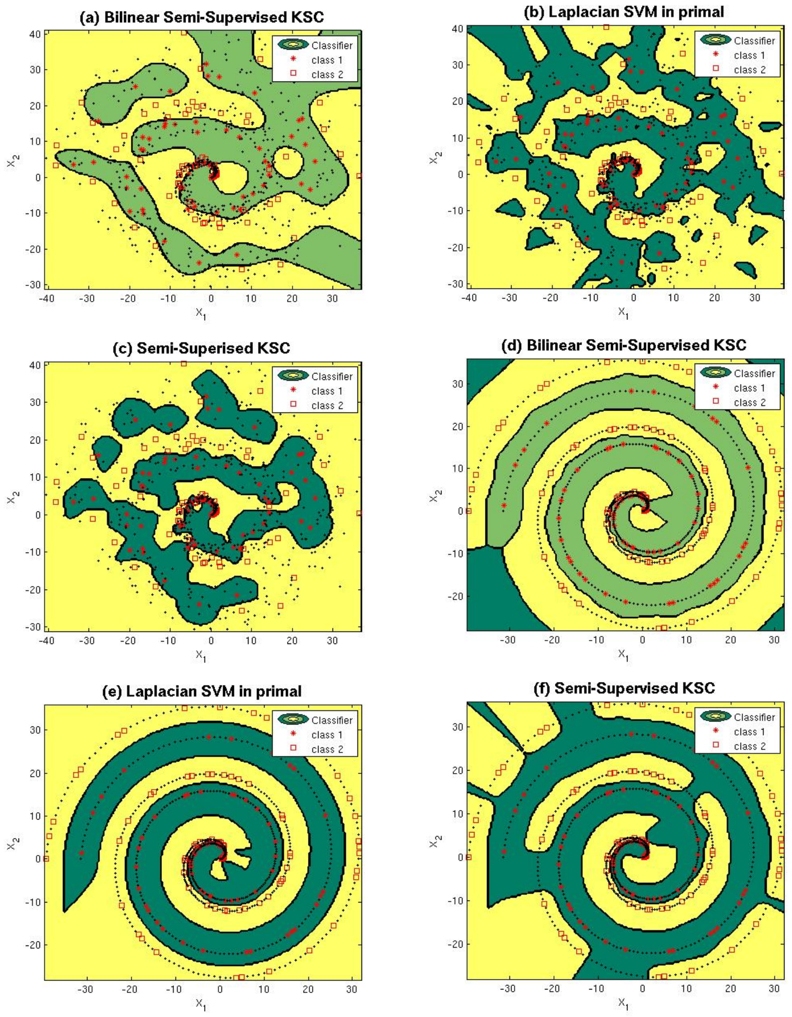


Figure 2.13: Different approaches applied to the "spirals" problem. Top from left to right: (a) Bilinear Semi-Supervised KSC, (b) Laplacian SVM in primal, (c) Semi-Supervised KSC in an overlapping scenario. Bottom from left to right: (d) Bilinear Semi-Supervised KSC, (e) Laplacian SVM in primal, (f) Semi-Supervised KSC in a noiseless scenario. With small black dots we denote unlabeled datapoints. Bigger red stars and squares represent labeled samples from two classes.

Table 2.11: Averaged misclassification rate for the benchmark SSL datasets

Dataset	Bi-SemiKSC	LapSVM [14]	LapSVMp [88]	SemiKSC [4]
g241c	**0.148**±0.044	0.232±0.052	0.248±0.063	0.194±0.102
g241d	**0.158**±0.034	0.278±0.062	0.276±0.053	0.191±0.069
BCI	**0.244**±0.056	0.376±0.049	0.324±0.057	0.315±0.078
Text	**0.382**±0.082	0.458±0.026	0.455±0.025	0.477±0.062

Additionally to the misclassification rate for UCI data we report in Table 2.13 p-values of a pairwise t-test on the same rate. We do not report these values for SSL data because we averaged the results only across 12 runs/splits which is quite a small number for a reliable t-test. By evaluating Table 2.13 we can conclude that attained p-values are very small and the means of distributions for misclassification rates differ statistically significantly between our approach and the corresponding competing approaches.

Table 2.12: Averaged misclassification rate for UCI datasets

Dataset	Bi-SemiKSC	LapSVM	LapSVMp	SemiKSC
Parkinsons	**0.039**±0.015	0.117±0.109	0.074±0.059	0.067±0.025
Sonar	**0.088**±0.018	0.131±0.085	0.122±0.086	0.102±0.053
Ionosphere	**0.099**±0.021	0.201±0.093	0.138±0.078	0.119±0.065
Ecoli	**0.036**±0.009	0.059±0.029	0.041±0.016	0.039±0.012
Arrhythmia	**0.257**±0.042	0.349±0.061	0.291±0.029	0.317±0.062

Additionally to the presented results we would like to compare some of the obtained results for SSL datasets with the reported values in the book of Chapelle *et al.*, Chapter 21.3 [30] and recently published paper on non-parallel KSC-based semi-supervised learning [87]. The experimental setup everywhere is the same but implementation details of cross-validation and hyperparameter tuning differ. Comparing our results with the book [30] we can clearly see that the best reported value for g241c dataset is 0.1741 which is by 2% worse than our result and it is related to Low-Density Separation technique (LDS) [28]. For BCI dataset our approach again is the best one and attained rates are much lower than those reported in the book. If we consider another relevant and non-parallel approach discussed in [87] we can see that authors report better result of 0.26 ± 0.01 for BCI dataset but it is still around 2% worse than we do.

Table 2.13: P-values of a pairwise t-test on misclassification rate between Bilinear SemiKSC and other methods

Dataset	to LapSVM	to LapSVMp	to SemiKSC
Parkinsons	2.4488E-06	0.00013319	1.1295E-09
Sonar	0.00064336	0.0077843	0.077444
Ionosphere	2.7248E-11	0.001315	0.044832
Ecoli	1.2486E-06	0.079189	0.11091
Arrhythmia	6.0693E-14	1.2068E-05	1.7722E-07

2.2.5 Discussion

Differences with other approaches

In this subsection we will briefly discuss some of the important differences between our approach and other non-parallel semi-supervised classifiers. As we aforementioned in Section I there exist several kernel-based non-parallel classifiers [61, 72] and only some of them explicitly utilize unlabeled information [87]. While being together with [4] and [87] based fundamentally on the same principles of KSC-implied modelling our method can be distinguished in a way we are coupling non-parallel classifiers. The reasoning and motivation behind this approach one can find in [65]. In brief it relates to the optimal separation between classes when *i.i.d.* assumption doesn't hold. From the modelling perspective in semi-supervised learning it might help to model each class and the underlying manifold better while preserving necessary discrimination between classes.

Future work

For the future research direction we are considering to explore different possible formulations of the Bi-SemiKSC model and apply them to large-scale datasets. While working on the millions of datapoints is not being directly feasibly in the dual formulation one can use Nyström approximation and Fixed-Size techniques [126, 39, 86] to go back for the primal formulation. This approach helps to devise a high-dimensional feature map which approximates well non-linearity while keeping first-order gradient-descent methods [107, 62] applicable and reasonably effective for optimizing the objective function in the primal.

2.2.6 Conclusion

In this paper we proposed a novel and promising way of handling Semi-Supervised Learning via incorporating a bilinear coupling term between non-parallel classifiers. This term enters a primal problem formulation and eventually boils-down to the coupled Lagrange multipliers comprising the solution of the dual problem. The fundamental part of our approach is a KSC-based model which helps to learn from the unlabeled datapoints and possesses a natural out-of-sample extension for the unseen ones. Our experimental validation for the artificial and real-life datasets confirms the usefulness and generalization capabilities of the proposed approach. We showed that even when a low-density assumption is broken we can learn from the unsupervised data and obtain acceptable results. For the future we consider extending our method to cope with large-scale data sources where proliferation of unlabeled information hampers the successful adoption of the supervised modelling techniques.

Chapter 3

Stochastic Learning for Support Vector Machines

This chapter comprises previously published articles including:

1. Jumutc V., Huang X., Suykens J.A.K., **"Fixed-Size Pegasos for Hinge and Pinball Loss SVM"**, in Proc. of the 2013 International Joint Conference on Neural Networks (IJCNN 2013), Dallas, USA, Aug. 2013, pp. 1122-1128. (Section 3.1)
2. Jumutc V., Suykens J., **"Weighted Coordinate-Wise Pegasos"**, in Proc. of the 5th International Conference on Pattern Recognition and Machine Intelligence (PREMI 2013), Kolkata, India, Dec. 2013, pp. 262-269. (Section 3.2)

Keywords—stochastic learning; loss functions; linear Support Vector Machines; Nytström approximation; Fixed-Size approach; proximal mapping;

3.1 Fixed-Size Pegasos for Hinge and Pinball Loss SVM

3.1.1 Introduction

Recent research in linear Support Vector Machines (SVM) [62, 107, 29] justified the importance of the first order approaches in bringing these machine learning techniques to large scale. While computing full gradients sometimes might be not feasible it was proposed recently to apply stochastic approximations [92] to the original optimization problem. The latter approach considerably saves memory and only to some degree increases the number of iterations to converge. In this paper we consider another aspect of learning SVM models in primal and particularly put our attention to using another loss function.

Pegasos [107] has become a widely acknowledged algorithm for learning linear SVM and has attracted research interest because of the strongly convex optimization objective and better convergence bounds. Pegasos utilizes hinge loss which replaces the original linear constraints while making the SVM objective unconstrained. With the proper projection step Pegasos achieves a solution of accuracy ϵ in $O(\frac{R^2}{\lambda\epsilon})$ iterations where λ is the regularization parameter. We would like to stress the fact that the hinge loss plays an important but not the essential role in establishing the results of Pegasos. Other authors [56] experimented with other types of loss (*e.g.* logarithmic loss) and optimization techniques.

In this paper we try to enrich the class of loss functions applicable for Pegasos with pinball loss [58] while establishing some essential theory to support our findings. We show some advantages and potential strengths of using the pinball loss within the Pegasos framework in a variety of classification problems. By the latter we assume achieving better classification performance and greater numerical stability in a partially or fully stochastic setting.

Together with the new loss function we employ in this paper Fixed-Size approach [129, 39, 121] which can be coupled with the Pegasos algorithm for learning linear classifiers in the induced feature space[1] for linearly non-separable cases. This setting generally enables us to solve any classification problem with the Pegasos algorithm and requires only one additional parameter for active subset selection. The details of this approach together with the corresponding algorithm will be given in Section IV.

[1]hereafter the induced feature space stands for the feature space of the RBF kernel approximated by the Fixed-Size approach.

This paper is organized as follows. Section 3.1.2 provides an introduction to hinge and pinball loss SVM and presents the corresponding properties of both loss functions. Sections 3.1.3–3.1.4 briefly outlines Pegasos optimization objective and presents our main results. Section 3.1.5 explains the Fixed-Size approach and outlines our algorithm for learning the Pagasos with the induced feature space. Experimental setup and numerical results are given in Section 3.1.6 while Section 3.1.7 concludes the paper.

3.1.2 Hinge and Pinball Loss SVM

The linear hinge loss SVM proposed by [125] takes the following formulation,

$$\min_{w} \quad \frac{\gamma}{2}\|w\|^2 + \frac{1}{m}\sum_{(x,y)\in\mathcal{S}} \mathbb{L}(w;(x,y)), \tag{3.1}$$

where $\mathcal{S}$ is the training dataset and

$$\mathbb{L}(w;(x,y)) = \max\{0, 1 - y\langle w, x\rangle\} \tag{3.2}$$

stands for the hinge loss with the (x,y) pair. In this setting the decision function is given by $\hat{y} = \text{sign}(\langle w, x\rangle)$. Hinge loss SVM has a nice geometrical interpretation for separable cases. It tries to find the largest margin between the instances of two classes. Consider a two dimensional example: points in two classes follow Gaussian distribution $\mathcal{N}(\mu_1, \Sigma)$ and $\mathcal{N}(\mu_2, \Sigma)$, respectively, where $\mu_1 = [0.5, -3]^T, \mu_2 = [-0.5, 3]^T$, and $\Sigma = \begin{bmatrix} 0.2 & 0 \\ 0 & 3 \end{bmatrix}$. Fig.3.1a illustrates one sampling by red crosses and green stars. For this data set, via hinge loss SVM (3.1), we find a decision function $\langle w, x\rangle$ which has the largest margin between $\langle w, x\rangle = \pm 1$ among all the functions satisfying $y\langle w, x\rangle \geq 1$. In Fig.3.1a, the obtained lines $\langle w, x\rangle = -1, 0$ and 1 are illustrated by red, blue, and green solid line, respectively. For this data set, hinge loss SVM in Eq.(3.1) performs well but it is easy to be affected by noise around decision boundary. That means that hinge loss SVM in Eq.(3.1) may have very different results for training sets drawn from the same distribution. To illustrate this point, another data set, which has the same distribution in Fig.3.1a, and the corresponding result of Eq.(3.1) are shown in Fig.3.1b. One can see that though the data come from the same distribution, the results of Eq.(3.1) can be significantly different.

The sensitivity to noise or the instability to re-sampling comes from the fact that in hinge loss SVM, the distance between two sets is measured by the nearest points. Hence, one way to overcome this weak point is to change the definition of distance between two sets. For example, if we use the distance of

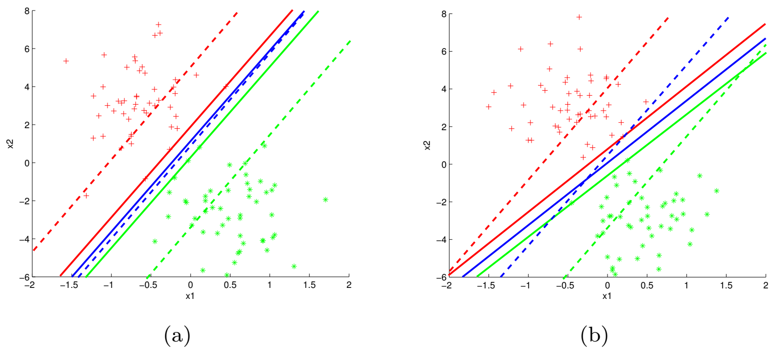


Figure 3.1: Points in two classes are marked by red crosses and green stars. The decision functions are shown by red, blue, and red lines, corresponding to $\langle w, x\rangle = -1, 0$, and 1, respectively. The data in (a) and (b) come from the same distribution but the results of hinge loss SVM (the solid lines) differ significantly. By contrast, the results of pinball loss SVM (the dashed lines) are more stable to re-sampling, which is suitable for stochastic gradient methods.

the nearest 30% points to measure the distance between two sets, the results are less sensitive, as shown by the dashed lines in Fig.3.1. Such distance is a kind of quantile value, which is closely related to pinball loss $\mathbb{L}_\tau$ defined by

$$\mathbb{L}_\tau(w;(x,y)) = \begin{cases} 1 - y\langle w, x\rangle & y\langle w, x\rangle \ \leq 1, \\ \tau(y\langle w, x\rangle - 1), & y\langle w, x\rangle \ > 1, \end{cases} \tag{3.3}$$

where the reasonable range of τ is $[0, 1]$ as explained in [58]. The pinball loss $\mathbb{L}_\tau$ has been applied for quantile regression, see, e.g. [71] [33] and [115]. Motivated by the relationship between pinball loss and quantile value, we proposed the following pinball loss SVM in [58],

$$\min_{w} \qquad \frac{\gamma}{2}\|w\|^2 + \frac{1}{m}\sum_{(x,y)\in\mathcal{S}} \mathbb{L}_\tau(w;(x,y)). \tag{3.4}$$

Hinge loss is a special case of pinball loss in Eq.(3.3) with $\tau = 0$. Accordingly, pinball loss SVM in Eq.(3.4) is an extension to hinge loss SVM. It has been shown in [58] that Eq.(3.4) and Eq.(3.2) have similar computational complexity and consistency property. Besides, the result of pinball loss SVM is less sensitive to noise around the boundary. Equivalently speaking, using pinball loss SVM, the results corresponding to different training data from the same distribution are stable. For the data in Fig.3.1, the decision functions obtained by Eq.(3.4) are shown by dashed lines. Compared with hinge loss SVM in Eq.(3.1), the results of pinball loss SVM are more stable for re-sampling from the same distribution, which is especially suitable for stochastic gradients methods.

3.1.3 Pegasos with Pinball Loss

In the original paper by Shalev-Shwartz *et al.* [107] the instantaneous optimization objective of Pegasos algorithm is given by

$$f(w;\mathcal{A}_t) = \frac{\lambda}{2}\|w\|^2 + \frac{1}{m}\sum_{(x,y)\in\mathcal{A}_t} \mathbb{L}(w;(x,y)), \tag{3.5}$$

where the hinge loss for the (x, y) pair is denoted by Eq.(3.2) and $\mathcal{A}_t$ stands for our working random subsample at iteration t. We can write down the subgradient of the instantaneous objective as follows

$$\nabla_t = \lambda w_t - \frac{1}{|\mathcal{A}_t|}\sum_{(x,y)\in\mathcal{A}_t^+} yx, \tag{3.6}$$

where $\mathcal{A}_t^+$ denotes the subset of $\mathcal{A}_t$ for which $\mathbb{L}(w;(x,y)) > 0$. Finally after taking the gradient descent step with learning rate given by $\eta_t = 1/(\lambda t)$ we need to project back our solution onto the set

$$\mathcal{B} = \{w : \|w\| \leq 1/\sqrt{\lambda}\}. \tag{3.7}$$

The latter effectively enables better convergence rates of the Pegasos algorithm.

3.1.4 The algorithm

In this part we provide the main results of using the pinball loss within the Pegasos algorithm. We start with redefining the instantaneous optimization objective by the means of a new loss function which we already defined in Eq.(3.3). Now we can see that the subgradient term in Eq.(3.6) can be refined in terms of a new loss function as follows

$$\nabla_t = \lambda w_t - \frac{1}{|\mathcal{A}_t|}\left[\sum_{(x,y)\in\mathcal{A}_t^+} yx - \sum_{(x,y)\in\mathcal{A}_t^-} \tau yx\right] \tag{3.8}$$

where ∇_t additionally depends on the τ parameter of the pinball loss and $\mathcal{A}_t^-$ stands for the subset of $\mathcal{A}_t$ where $y\langle w, x\rangle > 1$ (see Eq.(3.3)).

Next we present a brief summary of the modified Pegasos in Algorithm 4 and extend it with the pinball loss in Algorithm 5. Next we continue with the analysis of Algorithm 5 in the next subsection. In Algorithm 4 we can see a major "for" loop where gradient and projection steps are taking place and a minor "if" condition which terminates execution if the norm of the difference of two subsequent w vectors is less than ϵ. In Algorithms 4–5 we denote the whole dataset by $\mathcal{S}$ and at each iteration select randomly m samples for computation of the subgradient. Another important issue is related to the computation of the bias term. We should emphasize that the bias term ρ is not part of our instantaneous optimization objective and we perform computation of it just to return convenient and ubiquitous representation of SVM decision function by $\hat{y} = \text{sign}(\langle w, x\rangle + \rho)$ where ρ is returned in the Line 10 and 13 of Algorithm 4 together with w vector. We should note that the incorporation of the bias term is fairly straightforward via adding a constant $c_{d+1} = 1$ to the feature vector $x_i \in \mathbb{R}^d$.

Algorithm 4: Pagasos with hinge loss

```
Data: $\mathcal{S}, \lambda, T, k, \epsilon$
1  Select $w_1$ randomly s.t. $\|w^{(1)}\| \leq 1/\sqrt{\lambda}$
2  for $t = 1 \to T$ do
3      Set $\eta_t = \frac{1}{\lambda t}$
4      Select $\mathcal{A}_t \subseteq \mathcal{S}$, where $|\mathcal{A}_t| = k$
5      $\rho = \frac{1}{|\mathcal{S}|} \sum_{(x,y)\in\mathcal{A}_t} (y - \langle w_t, x\rangle)$
6      $\mathcal{A}_t^+ = \{(x,y) \in \mathcal{A}_t : y(\langle w_t, x\rangle + \rho) < 1\}$
7      $w_{t+\frac{1}{2}} = w_t - \eta_t(\lambda w_t - \frac{1}{k} \sum_{(x,y)\in\mathcal{A}_t^+} yx)$
8      $w_{t+1} = \min\left\{1, \frac{1/\sqrt{\lambda}}{\|w_{t+\frac{1}{2}}\|}\right\} w_{t+\frac{1}{2}}$
9      if $\|w_{t+1} - w_t\| \leq \epsilon$ then
10         return $(w_{t+1}, \frac{1}{|\mathcal{S}|} \sum_{(x,y)\in\mathcal{S}} (y - \langle w_t, x\rangle))$
11     end
12 end
13 return $(w_{T+1}, \frac{1}{|\mathcal{S}|} \sum_{(x,y)\in\mathcal{S}} (y - \langle w_t, x\rangle))$
```

By comparing Algorithm 4 and 5 we can see that the major difference is the computation of the gradient step which involves additional parameter τ and a new subset $\mathcal{A}_t^-$.

Algorithm 5: Pagasos with pinball loss

```
Data: $\mathcal{S}, \lambda, \tau, T, k, \epsilon$
1  Select $w_1$ randomly s.t. $\|w^{(1)}\| \leq 1/\sqrt{\lambda}$
2  for $t = 1 \to T$ do
3      Set $\eta_t = \frac{1}{\lambda t}$
4      Select $\mathcal{A}_t \subseteq \mathcal{S}$, where $|\mathcal{A}_t| = k$
5      $\rho = \frac{1}{|\mathcal{S}|} \sum_{(x,y) \in \mathcal{A}_t} (y - \langle w_t, x \rangle)$
6      $\mathcal{A}_t^+ = \{(x,y) \in \mathcal{A}_t : y(\langle w_t, x \rangle + \rho) < 1\}$
7      $\mathcal{A}_t^- = \{(x,y) \in \mathcal{A}_t : y(\langle w_t, x \rangle + \rho) > 1\}$
8      $w_{t+\frac{1}{2}} = w_t - \eta_t(\lambda w_t - \frac{1}{k} \left[\sum_{(x,y) \in \mathcal{A}_t^+} yx - \sum_{(x,y) \in \mathcal{A}_t^-} \tau yx\right])$
9      $w_{t+1} = \min\left\{1, \frac{1/\sqrt{\lambda}}{\|w_{t+\frac{1}{2}}\|}\right\} w_{t+\frac{1}{2}}$
10     if $\|w_{t+1} - w_t\| \leq \epsilon$ then
11         return $(w_{t+1}, \frac{1}{|\mathcal{S}|} \sum_{(x,y) \in \mathcal{S}} (y - \langle w_t, x \rangle))$
12     end
13 end
14 return $(w_{T+1}, \frac{1}{|\mathcal{S}|} \sum_{(x,y) \in \mathcal{S}} (y - \langle w_t, x \rangle))$
```

Analysis

In this subsection we present a convergence analysis which brings to our algorithm the same convergence bounds as in Pegasos. We extend the analysis presented in [107] to our new instantaneous objective by presenting Theorem 3. But first we recap the important lemma from [107] which establishes necessary conditions for our theorem.

Lemma 2 (Shalev-Shwartz et al., 2007)**.** *Let $f_1, \ldots, f_T$ be a sequence of λ-strongly convex functions w.r.t. the function $\frac{1}{2}\|\cdot\|^2$. Let $\mathcal{B}$ be a closed convex set and define $\prod_{\mathcal{B}}(w) = arg\min_{w' \in \mathcal{B}} \|w - w'\|$. Let $w_1, \ldots, w_{T+1}$ be a sequence of vectors such that $w_1 \in \mathcal{B}$ and for $t \geq 1$, $w_{t+1} = \prod_{\mathcal{B}}(w_t - \eta_t \nabla_t)$, where ∇_t is a subgradient of f_t at w_t and $\eta_t = 1/(\lambda t)$. Assume that for all t, $\|\nabla_t\| \leq G$. Then, for all $u \in \mathcal{B}$ we have*

$$\frac{1}{T} \sum_{t=1}^{T} f_t(w_t) \leq \frac{1}{T} \sum_{t=1}^{T} f_t(u) + \frac{G^2(1 + \ln(T))}{2\lambda T}.$$

Based on the above lemma, we are now ready to bound the average instantaneous objective of Algorithm 5.

Theorem 2. *Assume $\|x\| \leq R$ for all $(x, y) \in \mathcal{S}$. Let*

$$w^* = \arg\min_{\mathrm{w}} \left[\frac{\lambda}{2}\|\mathrm{w}\|^2 + \frac{1}{|\mathcal{A}_\mathrm{t}|} \sum_{(\mathrm{x},\mathrm{y}) \in \mathcal{A}_\mathrm{t}} \mathbb{L}_\tau(\mathrm{w}; (\mathrm{x}, \mathrm{y})) \right]$$

and let $c = (\sqrt{\lambda} + (\tau + 1)R)$. Then, for $T \geq 3$ we have

$$\frac{1}{T}\sum_{t=1}^{T} f(w_t; \mathcal{A}_t) \leq \frac{1}{T}\sum_{t=1}^{T} f(w^*; \mathcal{A}_t) + \frac{c^2 \ln(T)}{\lambda T}.$$

Proof. To prove our theorem it suffices to show that all conditions of Lemma 1 hold. First we show that our problem is strongly convex. It is easy to verify that the first term in Eq.(3.5) is strictly convex with respect to w. Since f is a sum of λ-strongly convex function $\frac{\lambda}{2}\|w\|$ and another convex function (pinball loss), it is also λ-strongly convex. Next by assuming $\mathcal{B} = \{w : \|w\| \leq 1/\sqrt{\lambda}\}$ and the fact that $\|x\| \leq R$ we can bound subgradient ∇_t. The explicit form for the subgradient evaluated at point x is given in Eq.(3.8). Using the triangular inequality and denoting 2-norm by $\|\cdot\|$ one obtains

$$\|\nabla_t\| \leq \lambda\|w\| + \tfrac{1}{|\mathcal{A}_t|}\textstyle\sum_i (1+\tau)\|x_i\| \leq \sqrt{\lambda} + (1+\tau)R.$$

Finally we have to show that $w^* \in \mathcal{B}$. To do so, we need to derive the dual form of our objective for the whole dataset $|\mathcal{S}| = n$ in terms of the dual variables $\eta_i \in [-\frac{\tau}{n}, \frac{1}{n}]$. By [58] we know that dual form of SVM with pinball loss can be cast as the same SVM objective but with $\zeta_i = \alpha_i - \beta_i$ and additional constraints $\alpha_i + \frac{1}{\tau}\beta_i = \frac{1}{n}, \forall i$, such that the final Lagrangian form is given by

$$\max_{\eta_i} \min_{w} \sum_{i=1}^{n} \zeta_i - \frac{\lambda}{2}\|w\|^2$$

and because strong duality holds for the optimal solution *w.r.t* the primal variable w^* and dual variables ζ_i^* one gets

$$\frac{\lambda}{2}\|w^*\|^2 + \frac{1}{n}\sum_{x \in \mathcal{S}} \mathbb{L}_\tau(w^*; x) = -\frac{\lambda}{2}\|w^*\|^2 + \sum_{i=1}^{n} \zeta_i^*.$$

Rearranging the above, using the non-negativity of the pinball loss and the bound on $\|\zeta\|_1 \leq 1$ we obtain our initial condition of set $\mathcal{B}$: $\|w\| \leq 1/\sqrt{\lambda}$. Now we can plug-in everything back to the inequality in Lemma 1, follow our initial bound on T and complete the proof by putting all together. □

3.1.5 Fixed-Size approach

Nyström approximation of the RBF kernel feature map

While linear SVM techniques operating in the primal space achieved good generalization capabilities in some specific application areas, like text categorization [51] and microarray analysis [80] one cannot in general deliver solutions with a low generalization error. To overcome restrictions of Algorithm 5 which operates only in the primal space we apply a Fixed-Size approach [121, 39, 129] to approximate the RBF kernel feature map with some higher dimensional explicit and approximate feature vector. In some of our previous works [65] we already exploited the Fixed-Size approach for achieving an approximation of the RBF kernel while bringing the algorithm to the large scale. The latter introduces desired level of the non-linearity in our solution.

First we use an entropy based criterion to select the prototype vectors (small working sample of size $m \ll n$)[2] and construct the RBF kernel matrix K with

$$K_{ij} = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}. \tag{3.9}$$

Based on the Nyström approximation [129] an expression for the entries of the approximation of the feature map $\hat{\Phi}(x) : \mathbb{R}^d \to \mathbb{R}^m$, with $\hat{\Phi}(x) = (\hat{\Phi}_1(x), \ldots, \hat{\Phi}_m(x))^T$ is given by

$$\hat{\Phi}_i(x) = \frac{1}{\sqrt{\lambda_{i,m}}} \sum_{t=1}^{m} u_{ti,m} k(x_t, x), \tag{3.10}$$

where $\lambda_{i,m}$ and $u_{i,m}$ denote the i-th eigenvalue and the i-th eigenvector of K defined in Eq.(3.9). Using the above expression for $\hat{\Phi}(x)$ we can proceed with the original formulation of Algorithms 4–5 and find the solution of our problem in primal.

Complete procedure with the Fixed-Size approach

In Algorithms 6–7 we present the complete outline of the Pegasos algorithm evaluated together with the Fixed-Size approach and two types of loss functions.

In Algorithm 6 "PegasosHL" function stands for the shortcut of Algorithm 4 and "ComputeNystromApprox" function denotes the Fixed-Size part where we first compute $m \times m$ RBF kernel matrix from the data points found by the maximization of Renyi entropy in "FindActiveSet" function and than we apply

[2]see Section 4 of [39] for additional details

Algorithm 6: Fixed-Size Pegasos with hinge loss

```
input  : training data $\mathcal{S}$, labeling $Y$, parameters $\lambda, T, k, \epsilon, m$
output : mapping $\hat{\Phi}(x), \forall x \in \mathcal{S}$, SVM model given by $w$ and $\rho$
1 begin
2     $\mathcal{S}_r \leftarrow$ FindActiveSet($\mathcal{S}, m$);
3     $\hat{\Phi}(x) \leftarrow$ ComputeNystromApprox($\mathcal{S}_r$);
4     $X \leftarrow [\hat{\Phi}(x_1)^T, \ldots, \hat{\Phi}(x_n)^T]$;
5     $[w, \rho] \leftarrow$ PegasosHL($X, Y, \lambda, T, k, \epsilon$);
6 end
```

Eq.(3.10) to derive our approximate feature map. Finally we stack our explicit feature vectors in matrix X and proceed to the actual Pegasos implementation.

Algorithm 7: Fixed-Size Pegasos with pinball loss

```
input  : training data $\mathcal{S}$, labeling $Y$, parameters $\lambda, \tau, T, k, \epsilon, m$
output : mapping $\hat{\Phi}(x), \forall x \in \mathcal{S}$, SVM model given by $w$ and $\rho$
1 begin
2     $\mathcal{S}_r \leftarrow$ FindActiveSet($\mathcal{S}, m$);
3     $\hat{\Phi}(x) \leftarrow$ ComputeNystromApprox($\mathcal{S}_r$);
4     $X \leftarrow [\hat{\Phi}(x_1)^T, \ldots, \hat{\Phi}(x_n)^T]$;
5     $[w, \rho] \leftarrow$ PegasosPBL($X, Y, \lambda, \tau, T, k, \epsilon$);
6 end
```

In Algorithm 7 "PegasosPBL" function stands for the shortcut of Algorithm 5 and the rest procedure is the same as in Algorithm 6.

3.1.6 Experiments

Setup

In all our experiments we use a 2-step procedure for tuning the λ parameter of Algorithm 5 and bandwidth σ of RBF kernel used to approximate our feature map in Eq.(3.10). This procedure consists of Coupled Simulated Annealing [133] initialized with 5 random sets of parameters for the first step and the simplex method [91] for the second step. After CSA converges to some local minima we select the tuple of parameters that attains the lowest error and start the simplex procedure to refine our selection. On every iteration step for CSA and simplex method we proceed with a 10-fold cross-validation. While being considerably faster than the straightforward grid search technique obtained parameters tend to vary more because of the randomness in initialization.

For both Pegasos-based algorithms (with hinge and pinball losses) we approximate RBF kernel with Fixed-Size approach [39] where the σ parameter was inferred via cross-validation procedure described above. The active subset was selected via maximization of Renyi entropy. The size of this subset m was set to be $\sqrt{n}$ for all UCI and toy datasets of size less than 10000 and $\sqrt{n}/5$ for larger datasets. Parameter ϵ in Algorithm 5 was to $1e-6$ and pinball loss related τ parameter was provided separately for each experiment. Finally we performed two different settings with the fixed m parameter in Algorithm 5. During the first setup (partially stochastic) we set k to be 10% of $|\mathcal{S}|$ and in the second one (fully stochastic) $k = 1$. We first consider the toy dataset described in Section II. Experiments with the toy dataset were performed with the increasing number of switched labels (error) subsequently set to 0.05, 0.15 and 0.35. All experiments were repeated 50 times with the random split to training and test sets in proportion 1:1. 10-fold cross-validation was performed only on training set. Descriptions of all public [49] and toy datasets we can find in Table 3.1.

Table 3.1: Involved datasets

Dataset	# of attributes	# of classes	# of data points
Toy Data	2	2	10000
Pima	8	2	768
White Wine	12	2	4898
Sambase	57	2	4601
Transfusion	5	2	748
Magic	11	2	19020
Shuttle	9	2	58000
Skin	4	2	245057

Results

In this subsection we give numerical results on running Pegasos-based Algorithm 5 with pinball loss in different settings and choices of user-defined parameter τ. In Table 3.2 we can see our results for toy dataset generated with different number of switched labels (artificial noise). From Table 3.2 we can notice that everywhere pinball loss incorporated within Algorithm 5 attains better classification rates and smaller errors which are very close to initially embedded noise.

Table 3.2: Test errors for Pegasos

Dataset (% of errors)	Hinge Loss	Pinball Loss $\tau = 0.1$	$\tau = 0.5$	$\tau = 1$
Toy Data (5%)	0.08262	**0.06908**	0.06926	0.07446
Toy Data (15%)	0.18753	**0.15843**	0.16141	0.16538
Toy Data (35%)	0.36094	0.31829	0.32335	**0.31571**

In Table 3.3 we can observe results for UCI datasets within partially stochastic setting where we set k to be 10% of $|\mathcal{S}|$. We can notice that we perform almost equally good as hinge loss while giving better classification rates for Pima, Wine and Skin datasets with $\tau = 1$.

Table 3.3: Test errors for Pegasos with $k = 10\%$ of $|\mathcal{S}|$ (partially stochastic)

Dataset	Hinge Loss	Pinball Loss $\tau = 0.1$	$\tau = 0.5$	$\tau = 1$
Pima	0.28885	0.28417	0.27797	**0.27625**
Spambase	**0.13496**	0.13992	0.15047	0.15923
Transfusion	**0.23684**	0.23904	0.23824	0.23989
White Wine	0.27281	0.27124	0.26340	**0.26274**
Magic	**0.17992**	0.19330	0.20912	0.21371
Shuttle	**0.01015**	0.01943	0.02515	0.02705
Skin	0.00944	0.00997	0.01039	**0.00859**

In Table 3.4 we can see that when going to a completely stochastic setting with $k = 1$ advantages of pinball loss become more apparent and degradation of performance is not so crucial in comparison with hinge loss. We can notice that for almost all UCI datasets optimal value of $\tau = 0.5$. We can conclude that pinball loss SVM is more stable for re-sampling as shown in Figure 3.1b. Comparing our approach to Pegasos with hinge loss we can see general sustainability in fully stochastic setting which leads to immediate and practical application-wise importance in online learning.

Table 3.4: Test errors for Pegasos with $k = 1$
(fully stochastic)

Dataset	Hinge Loss	Pinball Loss		
		$\tau = 0.1$	$\tau = 0.5$	$\tau = 1$
Pima	0.28896	0.29422	**0.28870**	0.29198
Spambase	0.21444	0.21229	**0.20816**	0.21903
Transfusion	0.23406	0.23465	**0.23396**	0.23465
White Wine	0.29607	0.29526	0.29694	**0.28898**
Magic	0.22667	**0.22385**	0.22481	0.22750
Shuttle	0.04505	0.04145	**0.03499**	0.03736
Skin	0.02705	0.02498	**0.02172**	0.02401

Convergence

In this subsection we present some plots which give a better understanding of the convergence properties for the Pegasos algorithm with pinball loss. On plots we see the convergence over time for Pegasos with hinge loss (blue) and pinball loss (red) incorporated into the instantaneous optimization objective. In general we should note that the pinball loss should give a higher value of optimization objective and thereafter we are more interested in the convergence rates. We experiment with different values of λ and k parameters in Algorithm 5.

In Figures 3.2–3.3 we could see that Pegasos with pinball loss gives more stable and even better results. In Figure 3.2 the pinball loss reaches smaller objective values and in Figure 3.3 the convergence to near-optimal solution is faster than the corresponding convergence for hinge loss. We also observe more numerical stability in convergence for pinball loss which is justified by a smaller number of sudden peaks in the objectivsubsectione value.

3.1.7 Conclusion

In this paper we proposed an extension of the Pegasos algorithm for pinball loss and showed that it results into better classification rates and better numerical stability while running with pinball loss incorporated into its instantaneous optimization objective. We showed that the Fixed-Size approach helps to deal with linearly non-separable cases while applying the same learning procedure used for the linear SVM. Extensive numerical experiments show certain

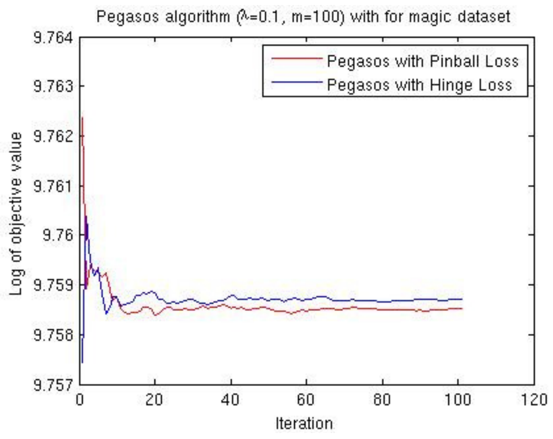


Figure 3.2: Convergence of Pegasos algorithm for Magic dataset in a short term (100 iterations) for hinge loss (blue) and pinball loss (red) respectively. In the experimental setup $\lambda = 0.1$ and $k = 100$.

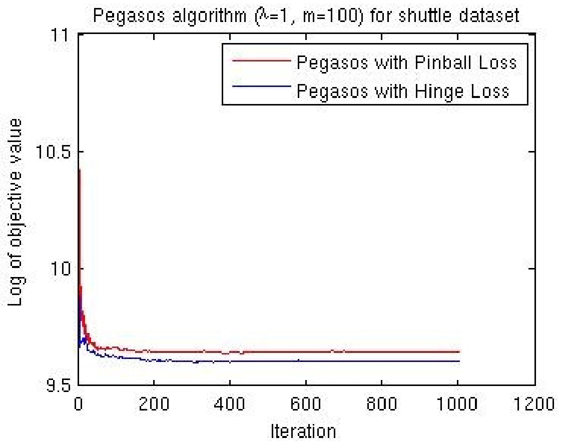


Figure 3.3: Convergence of Pegasos algorithm for Shuttle dataset in a long term (1000 iterations) for hinge loss (blue) and pinball loss (red) respectively. In the experimental setup $\lambda = 1$ and $k = 100$.

advantages of using the Pegasos algorithm with pinball loss and the Fixed-Size approach in the fully stochastic setting which in the turn implies a practical application-wise importance in online learning.

3.2 Weighted Coordinate-Wise Pegasos

3.2.1 Introduction

Recent advances in linear Support Vector Machines (SVM) and the first order stochastic optimization [62, 107, 29, 92] unlocked a room for improvement in the large-scale machine learning and black-box modelling. Usually computation of a full gradient for such large-scale problems is not bearable on a single machine, so one might consider to use a stochastic approximation to the original problem. In expectation [107, 92] such approach converges to the global optima if we consider strongly convex optimization objectives. Additionally the latter approach considerably saves the memory and only moderately increases the number of iterations to converge.

In this paper we propose a new weighted formulation of the Pegasos optimization objective and revise some of the algorithmic steps in order to maintain the consistency with the underlying theory. The key feature of the Pegasos [107] algorithm is a strongly convex optimization objective with the proper projection step. This combination helps to achieve a solution of accuracy ϵ in $O(\frac{R^2}{\lambda\epsilon})$ iterations where λ is the regularization parameter. On the other hand this approach imposes uniformed convergence speed for every single dimension. In general such uniform convergence is useful but doesn't reflect the importance and contribution of each dimension to the final classification result. The latter problem in Bayesian inference relates to Automatic Relevance Determination (ARD) [82, 123, 35]. We will compare this approach with the tuning via cross-validation for finding optimal λ_i hyperparameters.

The task of inferring optimal tuning parameters for our weighted coordinate-wise Pegasos is interesting by itself. In this paper we will highlight recent research in this subject and stress the difference in assumptions and behavior of Bayesian approach and tuning via global optimization techniques like Coupled Simulated Annealing (CSA) [133]. In kernel methods one can have three levels of inference. Two of them explicitly cope with the inference of hyperparameters like C in C-SVM [125], γ in LS-SVM [120] or σ which is relevant to the RBF kernel and non-linearity of the decision boundary. In our formulation of the Pegasos algorithm we deal with the linear approach and we are interested in obtaining the fittest regularization parameters λ_i *w.r.t* generalization and training speed-up. In our paper we will mostly consider only the first goal. In the end we will show convergence speed-up of the Weighted Coordinate-Wise Pegasos algorithm and explain the acquired results.

This paper is organized as follows. Section 3.2.2 outlines original and Weighted Coordinate-Wise Pegasos formulations, explains in detail our algorithm and gives a theoretical background, *i.e.* convergence proofs. Section 3.2.3 highlights Automatic Relevance Determination and Coupled Simulated Annealing used for tuning λ_i regularization hyperparameters. Experimental setup and numerical results are given in Section 3.2.4 while Section 3.2.5 concludes the paper.

3.2.2 Proposed method

Pegasos

In the Pegasos paper by Shalev-Shwartz *et al.* [107] authors propose to learn linear SVM model $\hat{y} = \text{sign}(w^T x + \rho)$ by the following instantaneous optimization objective

$$f(w; \mathcal{A}_t) = \frac{\lambda}{2}\|w\|^2 + \frac{1}{|\mathcal{A}_t|} \sum_{(x,y)\in\mathcal{A}_t} \mathbb{L}(w; (x, y)), \tag{3.11}$$

where λ is the regularization hyperparameter, $\mathbb{L}(w; (x, y))$ is the hinge loss for the (x, y) pair and $\mathcal{A}_t$ stands for our working random subsample at iteration t. The subgradient of the instantaneous objective is obtained as follows

$$\nabla_t = \lambda w_t - \frac{1}{|\mathcal{A}_t|} \sum_{(x,y)\in\mathcal{A}_t^+} yx, \tag{3.12}$$

where $\mathcal{A}_t^+$ denotes the subset of $\mathcal{A}_t$ for which $\mathbb{L}(w; (x, y)) > 0$. Finally we can ensure proper convergence of the algorithm after making the gradient descent step with the learning rate at iteration t given by $\eta_t = 1/(\lambda t)$ and projecting back the solution onto the set

$$\mathcal{B} = \{w : \|w\| \leq 1/\sqrt{\lambda}\}. \tag{3.13}$$

Optimization objective

Our approach is based on a simple assumption related to Automatic Relevance Determination. If we impose regularization differently for each dimension or data point we end up with the method which has a weighted impact of the features, dimensions or data samples on the classification output. In the case of the dual representation of SVMs we might talk about box constraints [125] which are related to the dual α unknowns. The larger is the upper bound of the particular box constraint the greater might be the influence of that data point on the decision boundary. On the other hand with ARD techniques and Bayesian inference one maximizes the posterior and the evidence can be used to assign a preference to alternative values of the hyperparameters [82].

To obtain such weighted formulation for the Pegasos algorithm first we propose a new optimization objective

$$f_{wcw}(w; \mathcal{A}_t) = \frac{1}{2} w^T \Lambda w + \frac{1}{|\mathcal{A}_t|} \sum_{(x,y)\in\mathcal{A}_t} \mathbb{L}(w; (x, y)), \tag{3.14}$$

where f_{wcw} stands for our new "weighted coordinate-wise" instantaneous optimization objective and it differs *w.r.t* original Pegasos objective only in the way we define our hyperparameters. Here Λ stands for the diagonal matrix with entries corresponding to coordinate-wise λ_i regularization hyperparameters.

Now we can see that the subgradient term in Eq.(3.12) becomes

$$\nabla_t = \Lambda w_t - \frac{1}{|\mathcal{A}_t|} \sum_{(x,y)\in\mathcal{A}_t^+} yx. \tag{3.15}$$

These modifications to the Pegasos algorithm do not guarantee a convergence without further modifications to the projection step in Eq.(3.13). In the next subsection we will take a closer look at our Weighted Coordinate-Wise Pegasos algorithm and present these missing details.

The algorithm

Hereby we present a brief summary of the modified Pegasos in Algorithm 8. Next we continue with the theoretical guarantees for convergence of this algorithm in the next subsection. In Algorithm 8 we can see a major "for" loop where gradient and projection steps are taking place and a minor "if" condition terminates an execution if the norm of the difference of two subsequent w vectors is less than ϵ. In Algorithm 8 we denote the whole dataset by $\mathcal{S}$ and at each iteration select randomly k samples for computation of the subgradient in Eq.(3.15). We should emphasize the importance of the projection step at the Line 9 where we are projecting our solution back on the ball with the radius $1/\sqrt{\lambda_{min}}$ and updated step size at the Line 4.

Analysis

This subsection presents a convergence analysis and theoretical background which brings our algorithm to the similar bounds as in the original paper of Shalev-Shwartz *et al.* [107]. We extend the analysis given in their paper to our new instantaneous objective by presenting Theorem 3.

Algorithm 8: Weighted Coordinate-Wise Pagasos

```
Data: $\mathcal{S}, \Lambda, T, k, \epsilon$
1  Set $\lambda_{min} = \min_i \Lambda_{ii}$
2  Select $w_1$ randomly s.t. $\|w^{(1)}\| \leq 1/\sqrt{\lambda_{min}}$
3  for $t = 1 \to T$ do
4      Set $\eta_t = \frac{1}{t}\Lambda^{-1}$
5      Select $\mathcal{A}_t \subseteq \mathcal{S}$, where $|\mathcal{A}_t| = k$
6      $\rho = \frac{1}{|\mathcal{S}|}\sum_{(x,y)\in\mathcal{S}}(y - \langle w_t, x\rangle)$
7      $\mathcal{A}_t^+ = \{(x,y) \in \mathcal{A}_t : y(\langle w_t, x\rangle + \rho) < 1\}$
8      $w_{t+\frac{1}{2}} = w_t - \eta_t(\Lambda w_t - \frac{1}{k}\sum_{(x,y)\in\mathcal{A}_t^+} yx)$
9      $w_{t+1} = \min\left\{1, \frac{1/\sqrt{\lambda_{min}}}{\|w_{t+\frac{1}{2}}\|}\right\} w_{t+\frac{1}{2}}$
10     if $\|w_{t+1} - w_t\| \leq \epsilon$ then
11         return $(w_{t+1}, \frac{1}{|\mathcal{S}|}\sum_{(x,y)\in\mathcal{S}}(y - \langle w_t, x\rangle))$
12     end
13 end
14 return $(w_{T+1}, \frac{1}{|\mathcal{S}|}\sum_{(x,y)\in\mathcal{S}}(y - \langle w_t, x\rangle))$
```

Theorem 3. *Let $\mathcal{B}$ be a closed convex set and define $\prod_{\mathcal{B}}(w) = arg\min_{w'\in\mathcal{B}} \|w - w'\|$. Let $w_1, \ldots, w_T$ be a sequence of vectors such that $w_1 \in \mathcal{B}$ and for $t \geq 1$, $w_{t+1} = \prod_{\mathcal{B}}(w_t - \eta_t \nabla_t)$, where ∇_t is a subgradient of Eq.(3.14) at w_t and $\eta_t = \frac{1}{t}\Lambda^{-1}$. Let w^* be the solution of the optimization problem in Eq.(3.14), $\lambda_{min} = \min_i \Lambda_{ii}$, $\lambda_{max} = \max_i \Lambda_{ii}$, $\lambda_{min} > 0$ and let $G = \lambda_{max}/\sqrt{\lambda_{min}} + R$. Assume $\|x\| \leq R$ for all $(x,y) \in \mathcal{S}$. Then, for $T \geq 1$ we have*

$$\|w_T - w^*\| \leq 2G\frac{\lambda_{max}}{\lambda_{min}^2}(\ln(T) + 1) + \frac{2R}{\lambda_{min}}.$$

Proof. To prove our theorem we refer to Lemma 4 in [106] and Section 9.1.2 in [22]. To apply results from Boyd's book [22] first we need to show that our problem is strongly convex. It is easy to verify that the first term in Eq.(3.14) is λ_{min}-strongly convex with respect to w. Since f is a sum of λ_{min}-strongly convex function and another convex function (hinge loss), it is also strongly convex. From Section 9.1.2 in [22] we know that one can bound the distance to the minimizer w^* of the objective in Eq.(3.14) by

$$\|w_T - w^*\| \leq \frac{2}{\lambda_{min}}\|\nabla f(w_T)\|.$$

The simplistic bound on $\|\nabla f(w_T)\|$ is derived using the upper-bound on the matrix-vector product as follows

$$\|\nabla f(w_T)\| \le \|\Lambda\|\|w_T\| + \frac{1}{|\mathcal{A}_T|}\sum_{(x,y)\in\mathcal{A}_T^+}\|yx\| \le \lambda_{max}\|w_T\| + R.$$

Next using the fact that there exists t-independent bound on $\|\nabla f(w)\| \le G$, $w_t = w_{t-1} - \eta_t \nabla_t$, $w_0 = 0$ and applying induction and the triangular inequality one can show that

$$\|w_T\| \le \sum_{t=1}^{T}\|\eta_t \nabla f(w_t)\| \le G\|\Lambda^{-1}\|\sum_{t=1}^{T}\frac{1}{t} \le \frac{G}{\lambda_{min}}(\ln(T)+1).$$

Next by assuming $\mathcal{B} = \{w : \|w\| \le 1/\sqrt{\lambda_{min}}\}$ and the fact that $\|x\| \le R$ we can derive t-independent bound on subgradient $\nabla f(w)$. The explicit form for the subgradient evaluated at intermediate solution w_t is given by Eq.(3.15). Using the triangular inequality, taking into account that Λ is a diagonal matrix one obtains

$$\|\nabla f(w)\| \le \|\Lambda\|\|w\| + \tfrac{1}{|\mathcal{A}_t|}\textstyle\sum_i \|x_i\| \le \lambda_{max}\|w\| + R \le \lambda_{max}/\sqrt{\lambda_{min}} + R.$$

Finally we have to show that $w^* \in \mathcal{B}$ and that there exist a universal upper bound for $\|w\|$. As a starting point we take Lemma 4 in [106]. First we need to derive the dual form of our instantaneous optimization objective for any subset $\mathcal{A}_t$ at iteration t. We use the fact that there exists a vector $\alpha^* \in [0,1]^m$ which maximizes our dual objective and duality gap equals to zero. Assuming these conditions and $|\mathcal{A}_t| = k$ one gets

$$\frac{1}{2}w^{*T}\Lambda w^* + \frac{1}{k}\sum_{(x,y)\in\mathcal{A}_t}\mathbb{L}(w^*;(x,y)) = -\frac{1}{2}w^{*T}\Lambda w^* + \frac{1}{k}\|\alpha^*\|_1.$$

Rearranging the above, using the non-negativity of the hinge loss, the bound on $\|\alpha\|_1 \le k$ and the fact that $w^{*T}\Lambda w^* \ge \lambda_{min}\|w^*\|^2$ we obtain our initial condition of set $\mathcal{B}$: $\|w\| \le 1/\sqrt{\lambda_{min}}$. Now we can plug-in everything back to the inequality in Theorem 1 and complete the proof by putting all together. □

3.2.3 Obtaining λ_i hypermarameters

In this section we focus on the problem of estimating λ_i hyperparameters. This problem is interesting by itself and gained a lot of attention within the framework of Automatic Relevance Determination in neural networks and kernel methods [82, 123, 35]. In our experiments with Weighted Coordinate-Wise Pegasos algorithm we will use two distinct strategies for obtaining these hyperparameters.

The first one is related to cross-validation and global optimization techniques, like Coupled Simulated Annealing [133]. Following this approach we evaluate 10-fold cross-validation for each iteration in every standard simulated annealing process. In CSA one couples together acceptance probability functions of these processes in order to control general statistical measures that may have crucial influence on the performance of the optimization.

The other technique is purely Bayesian and relies on the recent research in this area [130]. The main idea of the latter approach is to estimate these hyperparameters from the data by first marginalizing over the coefficients w and then performing what is commonly referred to as evidence maximization or type-II maximum likelihood [123, 81, 90]. Mathematically, in the context of Weighted Coordinate-Wise Pegasos this formulation is equivalent to minimizing

$$\mathcal{L}(\lambda) \triangleq -\log p(y;\lambda) \equiv \log|\Sigma_y| + y^T\Sigma_y^{-1}y, \tag{3.16}$$

where a flat hyperprior on λ_i hyperparameters is assumed and $\Sigma_y \triangleq \gamma I + \Phi\Lambda\Phi^T$. Here Φ denotes evaluated dataset and is given by the original mapping in the input space, *i.e.* $\phi(x) = x$, where $(x, y) \in \mathcal{S}$ as it is given in Theorem 1. In this setting we are following the proposed optimization via Iterative Re-Weighted Minimum l_1 in [130], where one is estimating the λ_i unknowns via alternating update rules which can be derived by decoupling $\mathcal{L}(\lambda)$ using upper bounding functions. This approach also relates to Sparse Bayesian Learning (SBL) that has been successful in a variety of applications [123].

3.2.4 Experiments

Setup

For the first part of our experiments we used a 2-step procedure for tuning the λ_i hyperparameters in Algorithm 8. This procedure consists of Coupled Simulated Annealing [133] initialized with 5 random sets of parameters for the first step and the simplex method [91] for the second step. After CSA converges to some local minima we select a tuple of λ_i hyperparameters which attains the lowest cross-validation error and start the simplex procedure to refine our selection. On every iteration step for CSA and simplex method we proceed with a 10-fold cross-validation.

For the second part of our experiments we used the ARD approach described in Section 3.2.3. For learning hyperparameters we used the procedure described in Section 2.1 of [130] and we defined $\gamma = 10^{-3}$ parameter related to our flat hyperprior in Eq.(3.16) (we can bind this parametrization to the numerical stability of the underlying approach). Finally we defined $\epsilon = 10^{-5}$ stopping criterion which controls the norm of the difference for two subsequent solutions. Once we drop below this threshold iterative re-weighting procedure halts. In this part we experimented mostly with small-scale UCI datasets given in Table 3.5 because of the computational burden of Σ_y matrix inversion and storage.

All experiments with large-scale UCI datasets were repeated 50 times with the random split to training and test sets in proportion 1:1. For the smaller UCI datasets we performed 50 iterations with splitting in proportion 1:9 taking 90% of the dataset for training. In the presence of 3 or more classes we performed binary classification where we learned to classify the first class versus all others. Only for Pen Digits dataset we performed several experiments where this setting was changed. For Algorithm 8 we fixed parameters: $T = 1000$ and $\epsilon = 10^{-5}$. Parameter k was set to $|\mathcal{S}|/10$ for all small-scale UCI datasets and we performed experiments with two different values of k for larger-scale UCI datasets. Description of all public UCI datasets [49] we can find in Table 3.5.

Table 3.5: Datasets

Dataset	# of attributes	# of classes	# of data points
Ionosphere	34	2	354
Parkinsons	23	2	197
Sonar	60	2	208
Iris	4	3	150
Ecoli	8	5	336
Red Wine	12	2	1599
White Wine	12	2	4898
Pen Digits	16	10	10992
Magic	11	2	19020
Shuttle	9	2	58000
Covertype	54	7	581012

Results

In this subsection we give numerical results on running Algorithm 8 within cross-validation and ARD settings. In Table 3.6 we can see our results for small-scale UCI datasets. Notation Pegasos$_{wcw}$ denotes our proposed approach with the tuning of λ_i hyperparameters via cross-validation and Pegasos$_{wcw/ard}$ stands for the same approach with hyperparameters obtained via the ARD method described in Section 3.2.3.

From Table 3.6 we can notice that Bayesian inference together with Algorithm 8 doesn't attain better results than the basic Pegasos algorithm. On the other hand Weighted Coordinate-Wise Pegasos performs better with respect to both of them.

Table 3.6: Test errors for small-scale datasets

Dataset	Pegasos	Pegasos$_{wcw}$	Pegasos$_{wcw/ard}$
Ionosphere	0.16889	**0.12714**	0.28922
Parkinsons	0.20832	**0.18274**	0.23253
Sonar	**0.23848**	0.28667	0.32176
Iris	0.34933	**0.30133**	0.61467
Ecoli	0.06500	**0.05710**	0.16299

In Table 3.7 we can observe results for large-scale UCI datasets within partially stochastic setting where we set k to be 10% of $|\mathcal{S}|$. We can notice that we perform equally better for almost all datasets in comparison with original Pegasos algorithm.

In Table 3.8 we can see that when going to a completely stochastic setting with $k = 1$ we are slightly deteriorating our performance. We can notice that our approach outperforms Pegasos algorithm for all datasets and it is well-suited for learning more accurately large scale linear SVMs via stochastic programming.

Convergence

In this subsection we present some plots which give a better understanding of the convergence properties for the Weighted Coordinate-Wise Pegasos algorithm. In Figure 3.4 we see the convergence over time for the basic Pegasos algorithm and our proposed approach. We ran algorithms with the optimal values of hyperparameters found via cross-validation procedure described in Section 4.2.5.

Table 3.7: Test errors for larger-scale datasets with $k = 10\%$ of $|\mathcal{S}|$ (partially stochastic)

Dataset	Pegasos	Pegasos$_{wcw}$
Magic	0.30550	**0.23607**
Shuttle	0.21450	**0.08549**
Red Wine	**0.26924**	0.27747
White Wine	0.32443	**0.30541**
Covertype	0.36466	**0.32807**
Pen Digits (1 vs all)	0.10432	**0.08984**
Pen Digits (2 vs all)	0.08448	**0.05133**
Pen Digits (5 vs all)	0.09609	**0.06830**
Pen Digits (6 vs all)	0.05835	**0.02896**

Table 3.8: Test errors for larger-scale datasets with $k = 1$ (fully stochastic)

Dataset	Pegasos	Pegasos$_{wcw}$
Magic	0.32288	**0.27743**
Shuttle	0.21751	**0.08598**
Red Wine	0.29552	**0.29224**
White Wine	0.32807	**0.30599**
Covertype	0.36474	**0.34962**
Pen Digits (1 vs all)	0.10409	**0.09255**
Pen Digits (2 vs all)	0.08437	**0.05317**
Pen Digits (5 vs all)	0.09635	**0.07059**
Pen Digits (6 vs all)	0.05863	**0.02847**

We can notice that Pegasos$_{wcw}$ attains faster convergence and lower objective values.

3.2.5 Conclusions and remarks

In this paper we proposed an extension of the Pegasos algorithm which is suitable for learning linear SVMs at the larger scale. It performs considerably better in terms of the generalization error. We presented two different approaches for learning λ_i regularization hyperparameters and showed that cross-validation

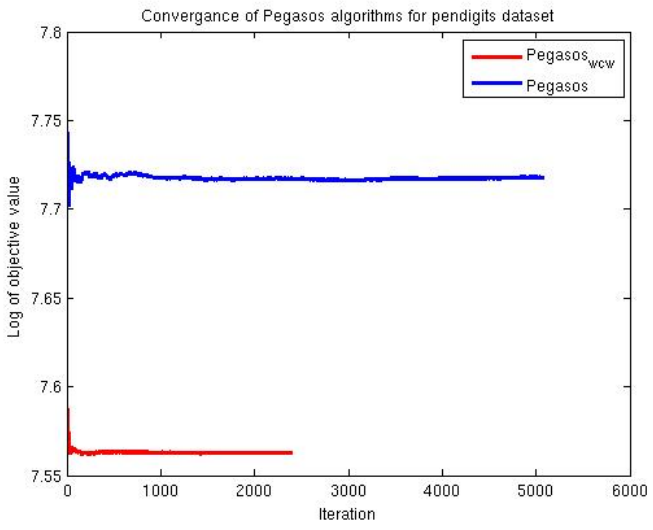


Figure 3.4: Convergence of Pegasos and Pegasos$_{wcw}$ algorithms for Pendigits dataset

together with CSA attains better results than a Bayesian approach. The latter results can be explained by the nature of ARD-related data-dependent prior distribution and inconsistencies between our assumptions and observed data. Finally we gave numerical results which demonstrated the merits of the proposed methods and verified the importance of the coordinate-wise parametrization for linear SVMs in terms of generalization error and training speed-up.

Chapter 4

New Sparsity Inducing Regularization for Stochastic Learning with SVMs and other Parametric Models

This chapter comprises previously published and currently in press articles including:

1. Jumutc V., Suykens J.A.K., **"Reweighted Stochastic Learning"**, Neurocomputing Special Issue - ISNN2014, 2015. (In Press) (Section 4.1)

2. Jumutc V., Langone R., Suykens J.A.K., **"Regularized and Sparse Stochastic K-Means for Distributed Large-Scale Clustering"**, in Proc. of the 2015 IEEE International Conference on Big Data (IEEE BigData 2015), Santa Clara, USA, Oct 29 – Nov 1, 2015. (Section 4.2)

Keywords—sparsity; l_0 penalty; regularization; stochastic learning; reweigthed iterate; regularized dual averaging; linear Support Vector Machines; proximal mapping; K-Means; distributed algorithms;

4.1 Reweighted Stochastic Learning

4.1.1 Introduction

In many domains dealing with online and stochastic learning, the input instances are of very high dimension, yet within any particular instance several features are non-zero. Therefore specific stochastic and online approaches crafted with sparsity inducing regularization are of particular interest for many machine learning researchers and practitioners. This paper investigates an interplay between Regularized Dual Averaging (RDA) approaches [134] (along with other techniques for solving linear SVMs in the context of stochastic learning [107]) and parsimony concepts arising from the application of sparsity inducing norms, like the l_0-type of a penalty.

One can see an increasing importance of correctly identified sparsity patterns and proliferation of proximal and soft-thresholding subgradient-based methods [134], [108], [45]. There are many important contributions of the parsimony concept to the machine learning field. One may allude to the understanding of the obtained solution and simplified or easy to extract decision rules [12, 96, 23]. On the other hand the informativeness of the obtained features might be useful for a better generalization on unseen data [12]. Approaches based on l_1-regularized loss minimization were studied in the context of stochastic and online learning by several research groups [134], [108], [32], [44] but we are not aware of any l_0-norm inducing methods which were applied in the context of Regularized Dual Averaging and stochastic optimization.

In this paper we are trying to provide a supplementary analysis and sufficient regret bounds for learning sparser linear Regularized Dual Averaging (RDA) [134] models from random observations. We extend and modify our previous research [68], [67] and present complementary proofs with fewer assumptions and discussion for the reported theoretical findings. We use sequences of (strongly) convex reweighted optimization objectives to accomplish this goal.

This paper is structured as follows. Section 4.1.2 describes previous work on l_0-norm induced learning and some existing solutions to stochastic optimization with regularized loss. Section 4.1.3 presents a problem statement for the reweighted algorithms. Section 4.1.3 introduce our reweighted l_1-RDA and l_2-RDA methods while Subsection on page 100 presents completely novel approach based on probabilistic reweighted Pegasos-like linear SVM solver. Subsections on pages 94 and 99 provide a theoretic background for our reweighted RDA approaches. Section 4.1.4 presents our numerical results and Section 4.1.5 concludes the paper.

4.1.2 Related work

Learning with $\|w\|_0$ pseudonorm regularization is a NP-hard problem [78] and can be approached via the reweighting schemes [31], [37], [131], [26] while lacking a proper theoretical analysis of convergence in the online and stochastic learning cases. Some methods, like [57], consider an embedded approach where one has to solve a sequence of QP-problems, which might be very computationally- and memory-wise expensive while still missing some proper convergence criteria.

In many existing iterative reweighting schemes [37], [75] the analysis is provided in terms of the Restricted Isometry Property (RIP) or the Null Space Property (NSP) [74], [37]. These approaches solely rely on the properties which are difficult to access beforehand in a data-driven fashion. This might be crucial if one decides to evaluate methods for their potential applicability. For instance in case of the Restricted Isometry Property, which is characterizing matrix Φ, one is interested to find a constant $\delta \in (0,1)$ such that for each vector w we would have:

$$(1-\delta)\|w\|_2 \leq \|\Phi w\|_2 \leq (1+\delta)\|w\|_2.$$

The RIP was introduced by Candes and Tao [25] in their study of compressed sensing and l_1-minimization. But it cannot be directly applied in the context of online and stochastic optimization because we cannot observe matrix Φ immediately. This fact directly impedes the successful implication of convergence guarantees based on the RIP or other related properties.

Other groups stemmed their research from the follow-the-regularized-leader (FTRL) family of algorithms [134], [44], [54] and complementary analysis for sparsity-induced learning. In primal-dual subgradient methods arising from this family of algorithms one aims at making a prediction $w_t \in \mathbb{R}^d$ on round t using the average subgradient of the loss function. The update encompasses a trade-off between a gradient-dependent linear term, the regularizer $\psi(w_t)$ and a strongly-convex term h_t for well-conditioned predictions. Our research is based on FTLR algorithms with primal-dual iterate updates, such as RDA [134], and corresponding theoretical guarantees are very much along the lines of the latter.

4.1.3 Reweighted Methods

Problem statement

In the stochastic Regularized Dual Averaging approach developed by Xiao [134] one approximates the loss function $f(w)$ by using a finite set of independent observations $\mathcal{S} = \{\xi_t\}_{1\leq t\leq T}$. Under this setting one minimizes the following optimization objective:

$$\min_w \frac{1}{T}\sum_{t=1}^{T} f(w,\xi_t) + \psi(w), \tag{4.1}$$

where $\psi(w)$ represents a regularization term. Every observation is given as a pair of input-output variables $\xi = (x, y)$. In the above setting one deals with a simple classification model $\hat{y}_t = \text{sign}(\langle w, x_t\rangle)$ and calculates the corresponding loss $f(w, \xi_t)$ accordingly[1]. It is common to acknowledge Eq.(4.1) as an online learning problem if $T \to \infty$.

The problem in Eq.(4.1) can be approached using a sequence of strongly convex optimization objectives. The solution of every optimization problem at iteration t is treated as a hypothesis of a learner which is induced by an expectation of possibly non-smooth loss function, *i.e.* $\mathbb{E}_\xi[f(w,\xi)]$. One can regularize it by a reweighted norm at each iteration t. This approach in case of satisfying the sufficient conditions will induce a bounded regret *w.r.t.* the loss function which is generating a sequence of stochastic subgradients endowing our dual space E^* [94].

For promoting sparsity we define an iterate-dependent regularization $\psi_t(w) \triangleq \lambda\|\Theta_t w\|$ which in the limit $(t \to \infty)$ applies an approximation to the l_0-norm penalty. At every iteration t we will be solving a separate convex instantaneous optimization problem conditioned on a combination of the diagonal reweighting matrices Θ_t. Specific variations of $\psi_t(w)$ for different norms (*e.g.* l_1- and l_2-norm) will be presented in the next subsections. By using a *simple dual averaging* scheme [94] we can solve our problem effectively by the following sequence of iterates w_{t+1} :

$$w_{t+1} = \arg\min_w \{\frac{1}{t}\sum_{\tau=1}^{t}(\langle g_\tau, w\rangle + \psi_\tau(w)) + \frac{\beta_t}{t}h(w)\}, \tag{4.2}$$

where $h(w)$ is an auxiliary 1-strongly convex smoothing term (proximal operator defined as $h(w) = \frac{1}{2}\|w - w_0\|$, where w_0 is set to origin), $g_t \in \partial f_t(w_t)$ represents a subgradient and $\{\beta_t\}_{t\geq 1}$ is a non-negative and non-decreasing sequence, which determines the boundedness of the regret function of our algorithms[2].

In detail Eq.(4.2) is derived using a different optimization objective where we have replaced static regularization term $\psi(w)$ in Eq.(4.1) with the iterate-dependent term $\psi_t(w)$. In the latter case our optimization objective becomes

[1]throughout this paper we will fix $f(w)$ to the hinge loss $f(w, \xi_t) = \max\{0, 1 - y_t\langle w, x_t\rangle\}$
[2]see Sections 12 and 12

$$\min_w \frac{1}{T}\sum_{t=1}^{T}\phi_t(w), \tag{4.3}$$

where composite function $\phi_t(w)$ is defined as $\phi_t(w) \triangleq f(w, \xi_t) + \psi_t(w)$. Using the aforementioned *dual averaging* scheme from [94] it is easy to show that the sequence w_t in Eq.(4.2) will approximate an optimal solution to Eq.(4.3) if we linearly approximate an accumulated loss function $f(w, \xi_t)$ from $\phi_t(w)$ and add a smoothing term $h(w)$. For exact details the interested reader can refer to Eq.(2.14) in [94] or in depth to Theorem 1 in [94].

Reweighted l_1-Regularized Dual Averaging

For promoting additional sparsity to the l_1-Regularized Dual Averaging method [134] we define $\psi_t(w) = \psi_{l_1,t}(w) \triangleq \lambda\|\Theta_t w\|_1$. Hence Eq.(4.2) becomes:

$$w_{t+1} = \arg\min_w\{\frac{1}{t}\sum_{\tau=1}^{t}(\langle g_\tau, w\rangle + \lambda\|\Theta_\tau w\|_1) + \frac{\gamma}{\sqrt{t}}(\frac{1}{2}\|w\|_2^2 + \rho\|w\|_1)\}. \tag{4.4}$$

For our reweighted l_1-RDA approach we set $\beta_t = \gamma\sqrt{t}$ and we replace $h(w)$ in Eq.(4.2) with the parameterized version:

$$h_{l_1}(w) = \frac{1}{2}\|w\|_2^2 + \rho\|w\|_1. \tag{4.5}$$

Each iterate has a closed form solution. Let us define $\eta_t^{(i)} = \frac{\lambda}{t}\sum_{\tau=1}^{t}\Theta_\tau^{(ii)} + \gamma\rho/\sqrt{t}$ and give an entry-wise solution by:

$$w_{t+1}^{(i)} = \begin{cases} 0, & \text{if } |\hat{g}_t^{(i)}| \le \eta_t^{(i)} \\ -\frac{\sqrt{t}}{\gamma}(\hat{g}_t^{(i)} - \eta_t^{(i)}\text{sign}(\hat{g}_t^{(i)})), & \text{otherwise} \end{cases}, \tag{4.6}$$

where $\hat{g}_t^{(i)} = \frac{t-1}{t}\hat{g}_{t-1}^{(i)} + \frac{1}{t}g_t^{(i)}$ is the i-th component of the averaged $g_t \in \partial f_t(w_t)$. *i.e.* $\hat{g}_t = \frac{1}{t}\sum_{\tau=1}^{t} g_\tau$.

Reweighted l_1-RDA Algorithm

In this subsection we will outline and explain our main algorithmic scheme for the Reweighted l_1-RDA method. It consists of a simple initialization step, drawing a sample $\mathcal{A}_t \subseteq \mathcal{S}$ from the dataset $\mathcal{S}$, computation and averaging of the subgradient g_t, evaluation of the iterate w_{t+1} and finally re-computation of the reweighting matrix Θ_{t+1}. By analyzing Algorithm 9 we can clearly see that it can operate in a stochastic ($k = 1$, where $k = |\mathcal{A}_t|$) and semi-stochastic mode ($k > 1$) such that one could draw only one or multiple samples from the dataset $\mathcal{S}$. We do not restrict ourselves to a particular choice of the loss function $f_t(w)$. In comparison with the l_1-RDA approach we have one additional input parameter ϵ, which should be tuned or selected properly as described in [26]. This additional hyperparameter ϵ controls the stability of the reweighting scheme and usually is set to a small number to avoid ill-conditioning of the matrix Θ_t.

Algorithm 9: Stochastic Reweighted l_1-Regularized Dual Averaging [67]

```
Data: $\mathcal{S}, \lambda > 0, \gamma > 0, \rho \geq 0, \epsilon > 0, T > 1, k \geq 1, \varepsilon > 0$
1  Set $w_1 = 0, \hat{g}_0 = 0, \Theta_1 = diag([1, \ldots, 1])$
2  for $t = 1 \to T$ do
3      Draw a sample $\mathcal{A}_t \subseteq \mathcal{S}$ of size $k$
4      Calculate $g_t \in \partial f_t(w_t; \mathcal{A}_t)$
5      Compute the dual average $\hat{g}_t = \frac{t-1}{t}\hat{g}_{t-1} + \frac{1}{t}g_t$
6      Compute the next iterate $w_{t+1}$ by Eq.(4.6)
7      Re-calculate the next $\Theta$ by $\Theta_{t+1}^{(ii)} = 1/(|w_{t+1}^{(i)}| + \epsilon)$
8      if $\|w_{t+1} - w_t\| \leq \varepsilon$ then
9          return $w_{t+1}$
10     end
11 end
12 return $w_{T+1}$
```

The time complexity of Algorithm 9 is driven by the computational budget T and subsample size k we use therein as our input parameters. In the worst case scenario the time complexity is of order $\mathcal{O}(dkT)$ where d is the input dimension.

Analysis for the Reweighted l_1-RDA method

In this section we will briefly discuss some of our convergence results and upper bounds for Algorithm 9. We concentrate mainly on the regret *w.r.t.* function $\phi_t(w) \triangleq f(w, \xi_t) + \psi_{l_1,t}(w)$ in Eq.(4.3) such that for all $w \in \mathbb{R}^n$ and iterates t we have:

$$\mathbf{R}_t(w) = \sum_{\tau=1}^{t}(\phi_\tau(w_\tau) - \phi_\tau(w)). \tag{4.7}$$

In Eq.(4.7) $\mathbf{R}_t(w)$ denotes an accumulated gap between function evaluations at the solution w_τ obtained in a closed form at iterate τ as for instance in Eq.(4.6) and any $w \in \mathbb{R}^n$. In detail we pay a fixed regret at each iterate τ if we take w_τ

instead of an optimal solution w^* for Eq.(4.3). From [94] and [134] we know that if we consider $\Delta\psi_{l_1,\tau} = \psi_{l_1,\tau}(w_\tau) - \psi_{l_1,\tau}(w)$ the following *gap sequence* δ_t holds:

$$\delta_t = \max_w\{\sum_{\tau=1}^{t}(\langle g_\tau, w_\tau - w\rangle \ + \Delta\psi_{l_1,\tau})\} \geq \sum_{\tau=1}^{t}(f(w_\tau) - f(w) + \Delta\psi_{l_1,\tau}) = \mathbf{R}_t(w), \tag{4.8}$$

which due to the convexity of f bounds the regret function from above [22]. Hence by ensuring the necessary condition of Eq.(49) in [134] we can show the upper bound on δ_t, which immediately implies the same bound on $\mathbf{R}_t(w)$.

Theorem 4. *Let the sequences $\{w_t\}_{t\geq1}$, $\{g_t\}_{t\geq1}$ and $\{\Theta_t\}_{t\geq1}$ be generated by the Algorithm 9. Assume $\psi_{l_1,t}(w_t) \leq \psi_{l_1,t}(w_{t+1})$, $\|g_t\|_* \leq G$, where $\|\cdot\|_*$ stands for the dual norm. Then for any fixed decision w:*

$$\mathbf{R}_t(w) \leq (\gamma D + \frac{G^2}{\gamma})\sqrt{t}, \tag{4.9}$$

where $h_{l_1}(w) \leq D$ holds for all $w \in \mathbb{R}^n$.

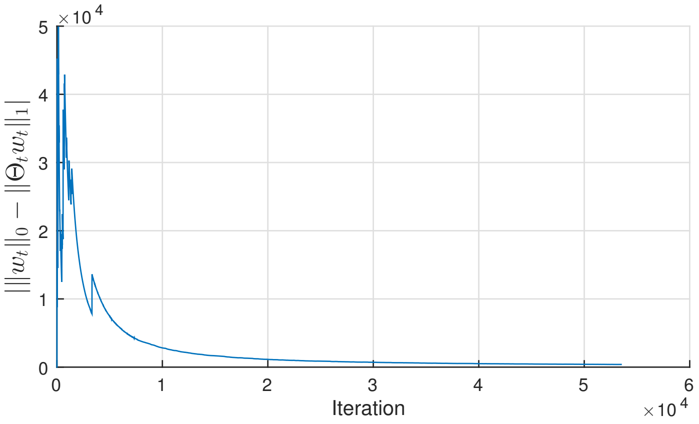


Figure 4.1: Difference between the reweighted l_1-norm and the true l_0-norm for CT slices dataset [49] at each iterate w_t.

Our intuition is related to the asymptotic convergence properties of an iterative reweighting procedure discussed in [57] where in the limit ($t \to \infty$) iterate Θ_t implies $\|\Theta_t w\|_1 \simeq \|w\|_{p_t}$ with $p_t \to 0$. Hereby we do not give any theoretical justification of the averaging effect implied by Eq.(4.2) on the approximation. Instead we present an empirical evidence of the convergence in Figure 4.1.

We ran the Reweighted l_1-RDA algorithm on the CT slices dataset [49] only once using all data points in an online learning setting with the parameters of Algorithm 9 set as follows: $\lambda, \rho, \gamma = 1, \epsilon = 0.1$. The number of iterations corresponds to the total number of available examples in $\mathcal{S}$.

In the next theorem we will slightly relax the necessary condition in order to derive a new bound *w.r.t.* maximal discrepancy of $\psi_{l_1,t}$ function evaluations at subsequent w_t iterates.

Theorem 5. *Let the sequences $\{w_t\}_{t\geq 1}$, $\{g_t\}_{t\geq 1}$ and $\{\Theta_t\}_{t\geq 1}$ be generated by the Algorithm 9. Assume $\psi_{l_1,t}(w_t) - \psi_{l_1,t}(w_{t+1}) \leq \nu/t$ for some $\nu \geq 0$, $\|g_t\|_* \leq G$, where $\|\cdot\|_*$ stands for the dual norm. Then for any fixed decision w:*

$$\mathbf{R}_t(w) \leq \nu \log t + (\gamma D + \frac{G^2}{\gamma})\sqrt{t}, \tag{4.10}$$

where $h_{l_1}(w) \leq D$ holds for all $w \in \mathbb{R}^n$.

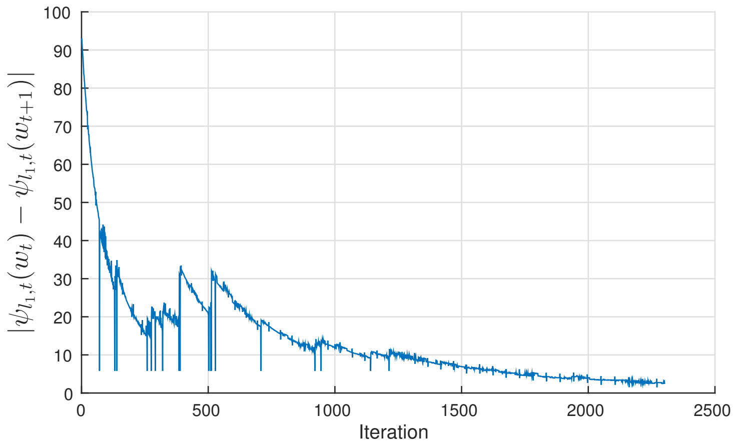


Figure 4.2: Near asymptotic convergence of the difference $|\psi_{l_1,t}(w_t)-\psi_{l_1,t}(w_{t+1})|$ at the first 2300 iterations for CT slices dataset.

From the above theorem it can be clearly seen that the dependence on the maximal discrepancy term is at most $\mathcal{O}(\log t)$ and is not linked directly to the non-strongly convex instantaneous optimization objective. Hence this discrepancy has less influence on the boundedness of the regret in comparison to the strong convexity assumption.

In Figure 4.2 we present an empirical evaluation of the near asymptotic convergence of $|\psi_{l_1,t}(w_t) - \psi_{l_1,t}(w_{t+1})|$ sequence, which verifies our assumption on the boundedness of this sequence in Theorem 5. The algorithmic setup is the same as in Figure 4.1. Proofs and lemmas related to Theorem 4 and Theorem 5 the interested reader can find in the Proofs for the Reweighted l_1-RDA method.

Reweighted l_2-Regularized Dual Averaging

Stemmed from the Section 4.1.3 this particular extension of the RDA approach [134] is dealing with the squared l_2 norm. For promoting additional sparsity we add the reweighted $\|\Theta_t^{1/2} w\|_2^2$ term such that we have $\psi_{l_2,t}(w) \triangleq \lambda\|w\|_2^2 + \|\Theta_t^{1/2} w\|_2^2$. At every iteration t we will be solving a separate λ-strongly convex instantaneous optimization objective conditioned on a combination of the diagonal reweighting matrices Θ_t.

To solve problem in Eq.(4.1) we split it into a sequence of separated optimization problems which should be cheap to compute and hence should have a closed form solution. These problems are interconnected through the sequence of dual variables $g_\tau \in \partial f(w, \xi_\tau), \tau \in \overline{1,t}$ and regularization terms which are averaged *w.r.t.* to the current iterate t.

Following the *dual averaging scheme* presented by Eq.(4.2) we can effectively solve our problem with a closed form solution. In our reweighted l_2-RDA approach we use a zero β_t-sequence[3] such that we omit the auxiliary smoothing term $h(w)$ in Eq.(4.2) which is not necessary since our $\psi_{l_2,t}(w)$ function is already smooth and strongly convex. Hence the solution for every iterate w_{t+1} in our approach is given by

$$w_{t+1} = \arg\min_w \{ \frac{1}{t} \sum_{\tau=1}^{t} (\langle g_\tau, w\rangle + \|\Theta_\tau^{1/2} w\|_2^2) + \lambda\|w\|_2^2 \}. \tag{4.11}$$

We will explain the details regarding recalculation of Θ_t and iterate w_{t+1} in the next subsection.

Reweighted l_2-RDA Algorithm

In this subsection we will outline and explain our main algorithmic scheme for the Reweighted l_2-RDA method. It consists of a simple initialization step, computation and averaging of the subgradient g_τ, evaluation of the iterate w_{t+1} and finally recalculation of the reweighting matrix Θ_{t+1}. In Algorithm 10 we

[3]we assume $\beta_0 = \lambda$ and $\beta_t = 0, t \geq 1$ for completeness

do not have any explicit sparsification mechanism for the iterate w_{t+1} except for the auxiliary function "`Sparsify`" which utilizes an additional hyperparameter ε and uses it to truncate the final solution w_t below the desired number precision as follows:

$$w_t^{(i)} := \begin{cases} 0, & \text{if } |w_t^{(i)}| \leq \varepsilon, \\ w_t^{(i)}, & \text{otherwise}, \end{cases} \tag{4.12}$$

where $w_t^{(i)}$ is i-th component of the vector w_t. In comparison with the simple l_2-RDA approach [134] we have one additional hyperparameter ϵ, which enters the closed form solution for w_{t+1} and should be tuned or adjusted *w.r.t.* the iterate t as described in [31] and highlighted in [26]. As in Section 4.1.3 this hyperparameter is introduced to stabilize the reweighting scheme and make Θ_t well-conditioned if some entries of w_t are zeros.

The time complexity of Algorithm 10 is the same as for Algorithm 9 with a small extra cost for sparsifying the final solution.

In Algorithms 9 and 10 we perform an optimization *w.r.t.* to the intrinsic bias term b, which doesn't enter our decision function

$$\hat{y} = \text{sign}(\langle w, x \rangle), \tag{4.13}$$

but is appended to the final solution w. The trick for including a bias term is to augment every input x_t in the subset $\mathcal{A}_t$ with an additional component which will be set to 1. This will alleviate the decision function with an offset in the input space. Empirically we have verified that sometimes this design has a crucial influence on the performance of a linear classifier.

Algorithm 10: Stochastic Reweighted l_2-Regularized Dual Averaging [68]

```
Data: $\mathcal{S}, \lambda > 0, k \geq 1, \epsilon > 0, \varepsilon > 0, \delta > 0$
1  Set $w_1 = 0, \hat{g}_0 = 0, \Theta_0 = diag([1, \dots, 1])$
2  for $t = 1 \to T$ do
3      Draw a sample $\mathcal{A}_t \subseteq \mathcal{S}$ of size $k$
4      Calculate $g_t \in \partial f(w_t, \mathcal{A}_t)$
5      Compute the dual average $\hat{g}_t = \frac{t-1}{t}\hat{g}_{t-1} + \frac{1}{t}g_t$
6      Compute the next iterate $w_{t+1}^{(i)} = -\hat{g}_t^{(i)} / (\lambda + \frac{1}{t}\sum_{\tau=1}^{t} \Theta_\tau^{(ii)})$
7      Recalculate the next $\Theta$ by $\Theta_{t+1}^{(ii)} = 1/((w_{t+1}^{(i)})^2 + \epsilon)$
8      if $\|w_{t+1} - w_t\| \leq \delta$ then
9          Sparsify($w_{t+1}, \varepsilon$)
10     end
11 end
12 return Sparsify($w_{T+1}, \varepsilon$)
```

Analysis for the Reweighted l_2-RDA method

In this subsection we will provide the theoretical guarantees for the upper bound on the regret of the function $\phi_t(w) \triangleq f(w, \xi_t) + \psi_{l_2,t}(w)$ as defined in Eq.(4.7). In this case we are interested in the guaranteed boundedness of the sum generated by this function applied to the sequences $\{\xi_1, \dots, \xi_t\}$ and $\{\Theta_1, \dots, \Theta_t\}$. In the next theorem we will provide the sufficient conditions for the boundedness of δ_t if the imposed regularization is given by the reweighted λ-strongly convex terms $\|\Theta_t^{1/2} w\|_2^2 + \lambda\|w\|_2^2$. Supplementary proofs and lemmas are provided in the Proofs for the Reweighted l_2-RDA method.

Theorem 6. *Let the sequences $\{w_t\}_{t\geq 1}$, $\{g_t\}_{t\geq 1}$ and $\{\Theta_t\}_{t\geq 1}$ be generated by Algorithm 10. Assume $\psi_{l_2,t}(w_t) \leq \psi_{l_2,t}(w_{t+1})$ and $\|g_t\|_* \leq G$, where $\|\cdot\|_*$ stands for the dual norm, and constant $\lambda > 0$ is given for all $\psi_{l_2,t}(w)$. Then for any fixed decision w:*

$$\mathbf{R}_t(w) \leq \frac{G^2}{2\lambda}(1 + \log t). \tag{4.14}$$

This theorem closely follows results presented in Section 3.2 of [134]. On the other hand our motivation and outline of the proof differs in many aspects. First we have to maintain a sequence of different regularization terms $\{\psi_{l_2,\tau}(w)\}_{1\leq\tau\leq t}$. Second averaging of this sequence is crucial for proving the boundedness of the conjugate support-type function $V_\tau(s_\tau)$ in [93, 94] for any $\tau \geq 1$.

Theorem 7 provides the necessary condition for deriving a new bound *w.r.t.* maximal discrepancy of $\psi_{l_2,t}$ function evaluations at subsequent w_t iterates.

Theorem 7. *Let the sequences $\{w_t\}_{t\geq 1}$, $\{g_t\}_{t\geq 1}$ and $\{\Theta_t\}_{t\geq 1}$ be generated by Algorithm 10. Assume $\psi_{l_2,t}(w_t) - \psi_{l_2,t}(w_{t+1}) \leq \nu/t$ for some $\nu \geq 0$, $\|g_t\|_* \leq G$, where $\|\cdot\|_*$ stands for the dual norm, and constant $\lambda > 0$ is given for all $\psi_{l_2,t}(w)$. Then for any fixed decision w:*

$$\mathbf{R}_t(w) \leq \frac{G^2}{2\lambda} + \frac{G^2 + 2\lambda\nu}{2\lambda}\log t. \tag{4.15}$$

The above bound boils down to the bound in Theorem 5 if we set ν to zero. In contrast with Theorem 5 here relaxation implies order $\mathcal{O}(\log t)$ dependency on the total number of iterations and perfectly fits within original bound in Theorem 6 implied by the strong convexity of the instantaneous optimization objective and the zero β_t-sequence.

Reweighted Dropout Pegasos

In this section we present another novel development based on the Pegasos algorithm [107] for solving linear SVMs. Our finding is motivated by [44], [113] and is rooted in the observation that more dominant features might require more frequent regularization based on the previous values (defined by w_{t-1}).

In our approach we maintain a vector of discrete Bernoulli variables of the same size as w_t, where every success probability $p_t^{(i)}$ of the Bernoulli distribution for feature i at round t is defined as follows:

$$p_t^{(i)} = \frac{w_{t-1}^{(i)} \cdot w_{t-1}^{(i)}}{1 + w_{t-1}^{(i)} \cdot w_{t-1}^{(i)}}. \tag{4.16}$$

Hence if feature i is updated more frequently and growing by modulus it has higher chances under the Bernoulli distribution for being regularized by the standard l_2-norm penalty [107]. The value of a discrete Bernoulli variable depends upon the weighted previous iterate w_{t-1}. After obtaining a draw of the particular Bernoulli distribution at round t we simply drop out from regularization features with zero-valued Bernoulli variables. This approach resembles "dropout" regularization applied in convolutional neural networks to prevent them from overfitting [113].

Algorithm 11: Reweighted Dropout Pegasos

```
Data: $\mathcal{S}, \lambda, T, k, \delta$
1  Select $w_0 = w_1$ randomly s.t. $\|w_1\| \leq 1/\sqrt{\lambda}$
2  for $t = 1 \to T$ do
3      Draw a sample $\mathcal{A}_t \subseteq \mathcal{S}$ of size $k$
4      Calculate $g_t \in \partial f(w_t, \mathcal{A}_t)$
5      Calculate $p_t^{(i)}$ by Eq.(4.16)
6      Draw a binary sample $r_t$ from $p_t$
7      Set $\eta_t = \frac{1}{\lambda t}$
8      $w_{t+\frac{1}{2}} = w_t - \eta_t(\lambda w_t \circ r_t - \frac{1}{k} g_t)$
9      $w_{t+1} = \min\left\{1, \frac{1/\sqrt{\lambda}}{\|w_{t+\frac{1}{2}}\|}\right\} w_{t+\frac{1}{2}}$
10     if $\|w_{t+1} - w_t\| \leq \delta$ then
11         return $w_{t+1}$
12     end
13 end
14 return $w_{T+1}$
```

Reweighted Dropout Pegasos has a very simple outline and can be summarized in Algorithm 11, where $\circ$ stands for the element-wise multiplication and g_t is a subgradient of an arbitrary convex loss function f_t. By analyzing Algorithm 11 we can see that one major deviation from the Pegasos algorithm is formulated in terms of a binary sample r_t which is drawn from the Bernoulli distribution p_t. This sample is used to drop out the regularization counterpart $\lambda\eta_t w_t$ at the step 8 for some particular dominant features.

4.1.4 Simulated experiments

Experimental setup

For all methods in our experiments with UCI datasets [49] for tuning (*e.g.* to estimate the ubiquitous λ hyperparameter or tuples of hyperparameters employed in Algorithms 9 and 10) we use Coupled Simulated Annealing [133] initialized with 5 random sets of parameters. These random sets are made out of tuples of hyperparameters linked to one particular setup of an algorithm. At every iteration step for CSA we proceed with a 10-fold cross-validation. Within the cross-validation we are promoting additional sparsity with a slightly modified evaluation criterion. We introduce an affine combination of the validation error and the obtained sparsity in proportion of 95% : 5% for initially non-sparse datasets and 80% : 20% for sparse datasets where sparsity is calculated as $\sum_i I(|w^{(i)}| > 0)/d$. This novel cross-validation criterion could be summarized as follows:

$$\text{criterion}_{cv}(X_{valid}, w) = (1-\kappa)\text{error}(X_{valid}, w) + \kappa \sum_i I(|w^{(i)}| > 0)/d, \quad (4.17)$$

where d is the input dimension, κ is the amount of introduced sparsity (0.05 vs. 0.20) and $\text{error}(X_{valid}, w)$ is implemented as a misclassification rate.

All experiments with large-scale UCI datasets [49] were repeated 50 times (iterations) with the random split to training and test sets in proportion 90% : 10%. At every iteration all methods are evaluated with the same test set to provide a consistent and fair comparison in terms of the generalization error and sparsity. In the presence of 3 or more classes we perform binary classification where we learn to classify the first class versus all others. For CT slices[4] dataset we performed a binarization of an output y_i by the median value. For URI dataset we took only "Day0" subset as a probe. For all presented stochastic algorithms we set $T = 1000$, $k = 1$, $\delta = 10^{-5}$ and other hyperparameters were determined using the cross-validation tuning procedure

[4]originally it is a regression problem

described above. All methods were using a hinge loss as f_t given in Eq.(4.1) and implemented in `julia` technical computing language[5]. Corresponding software can be found online at `www.esat.kuleuven.be/stadius/ADB/software.php` and `github.com/jumutc/SALSA.jl`.

Table 4.1: Datasets

Dataset	# attributes	# classes	# data points
Pen Digits	16	10	10992
Opt Digits	64	10	5620
CNAE-9	1080	9	857
Semeion	256	10	1593
Spambase	57	2	4601
Magic	11	2	19020
Shuttle	9	2	58000
CT slices	386	2	53500
Covertype	54	7	581012
URI	3231961	2	16000
RCV1	47152	4	804414

A slightly different approach we took for the RCV1 corpus data [79]. We adopted the experimental setup described in [44]. We ran all competing algorithms only once using cross-validation to search for the optimal trade-off hyperparameter λ. All other hyperparameters were set to the default values in order to obtain at least 50% sparsity. We set the T hyperparameter for all algorithms to the total number of training points, such that we could work within the online learning setting. There are 4 high-level categories: Economics, Commerce, Medical, and Government (ECAT, CCAT, MCAT, GCAT), and multiple more specific categories. We focus on training binary classifiers for each of these major categories. Originally RCV1 data is split into the test and training counterparts. We report our performance for both test and training data. One can find more information on the datasets in Table 4.1.

Numerical results

In this subsection we will provide an outlook on the performance of l_1-RDA [134], adaptive l_1-RDA$_{ada}$ [44], our reweighted l_1-RDA$_{re}$ [67] and l_2-RDA$_{re}$

[5]`http://julialang.org/`

Table 4.2: Generalization performance

Dataset	Generalization (test) errors in % for UCI datasets											
	l_1-RDA$_{re}$ [67]		l_2-RDA$_{re}$ [68]		l_1-RDA$_{ada}$ [44]		l_1-RDA [134]		Pegasos [107]		Pegasos$_{drop}$	
Pen Digits	**6.3**	(±2.1)	7.9	(±2.5)	6.9	(±2.3)	9.2	(±13)	6.3	(±1.9)	**6.1**	(±2.2)
Opt Digits	**3.9**	(±1.7)	4.8	(±2.1)	4.0	(±1.9)	4.4	(±1.9)	**3.4**	(±1.2)	5.3	(±6.4)
Shuttle	**5.3**	(±2.4)	6.9	(±2.7)	5.8	(±2.1)	5.6	(±2.0)	5.3	(±1.7)	**4.7**	(±1.4)
Spambase	11.0	(±3.0)	11.5	(±2.0)	**10.8**	(±1.7)	12.6	(±13)	10.0	(±1.7)	**9.4**	(±1.6)
Magic	22.7	(±2.4)	**22.2**	(±1.3)	22.4	(±1.7)	22.6	(±2.0)	**22.2**	(±1.1)	25.3	(±2.8)
Covertype	28.3	(±1.8)	27.0	(±1.4)	**25.3**	(±1.1)	26.6	(±2.6)	**27.6**	(±1.0)	28.2	(±2.6)
CNAE-9	2.0	(±1.4)	3.6	(±3.7)	**1.9**	(±1.4)	2.3	(±1.8)	1.2	(±1.1)	**0.9**	(±0.9)
Semeion	**8.9**	(±2.6)	13.3	(±18)	10.0	(±3.0)	11.6	(±13)	5.6	(±1.9)	**5.3**	(±1.8)
CT slices	**5.6**	(±1.4)	8.9	(±4.0)	8.4	(±2.8)	8.0	(±1.9)	**5.0**	(±0.7)	5.2	(±1.0)
URI	4.4	(±1.7)	5.2	(±3.0)	**4.0**	(±1.0)	4.8	(±2.5)	**4.3**	(±1.8)	8.4	(±6.0)

[68] methods as well as our novel Reweighted Dropout Pegasos algorithm (Pegasos$_{drop}$) together with original Pegasos [107] itself. In Table 4.2 one can see some generalization errors with standard deviations (in brackets) for all datasets.

In Table 4.2 we have highlighted dominant performances of sparsity inducing RDA-based algorithms and "non-sparse" Pegasos-based algorithms. Analyzing Table 4.2 we can conclude that for the majority of UCI datasets we are doing equally good *w.r.t.* the Adaptive l_1-RDA method and significantly better *w.r.t.* the original l_1-RDA approach. This fact can be understood from the similar (but different in theory) underlying principles of the reweighted and adaptive RDA approaches. Indeed both approaches are relying on the importance and hence accumulated information by the represented features. But in the adaptive RDA method we are "reweighting" our closed form solution at round t by the norm over historical subgradients while in the reweighted RDA approaches we are explicitly maintaining diagonal matrix Θ_t which directly preserves the weights and gives as in the limit the approximation for the l_0-norm. In hindsight we can evaluate performance of the competing approaches by taking a closer look at the boxplot of test error distributions in Figure 4.3. Analyzing performance of the Pegasos-based approaches we can see that for some datasets our Reweighted Dropout approach outperforms Pegasos.

Sparsity

In this subsection we will provide some of the findings which should highlight the enhanced sparsity of the reweighted RDA approaches. In Table 4.3 one can observe the evidence of an additional sparsity promoted by the reweighting procedures which in some cases significantly reduce the number of non-zeros in the obtained solution *w.r.t.* the adaptive and simple l_1-RDA approaches.

By analyzing Table 4.3 it is not difficult to imply that for almost all datasets

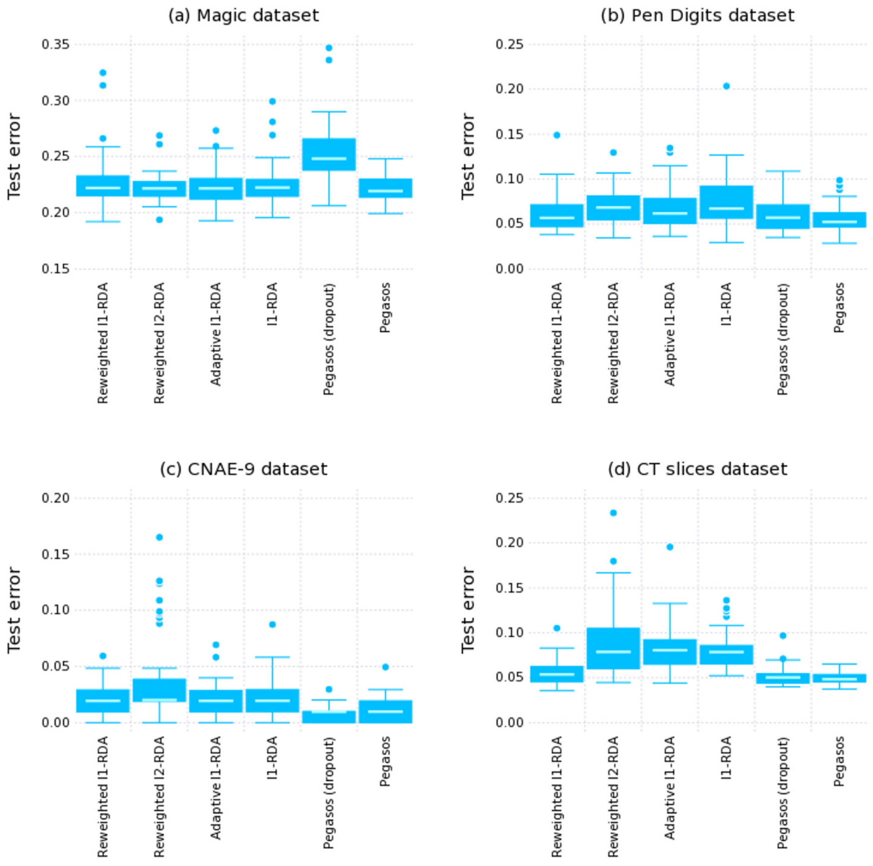


Figure 4.3: Comparison of the test error distribution over 50 runs for different UCI datasets.

we were able to find a good trade-off between sparsity and generalization error. For instance the Reweighted l_2-RDA method was able to find more sparsified solutions[6] with only a small increase in the generalization error for such datasets as Pen Digits, Spambase, Covertype or even better generalization, like for the Magic dataset. On the other hand the Reweighted l_1-RDA method was better in generalization for sparse datasets, like Semeion and CT slices, but less sparsifying than other RDA-based approaches. For one particular dataset (CNAE-9) the Reweighted l_1-RDA method was performing equally good in terms of generalization and sparsity. In hindsight we can evaluate the attained sparsity for different sparsity inducing methods and datasets over 50 trials in Figure 4.4.

By analyzing these distributions it is easy to verify that for some datasets, like (d) Semeion, most sparsity inducing methods are facing oversparsification issues, which in turn imply a considerable decay in generalization. For other presented datasets we are able to obtain more consistent performance *w.r.t.* other RDA-based approaches.

RCV1 dataset results

We present our results for RCV1 dataset [79] separately because of the different experimental setup and to concentrate on both training and test errors which are

[6] in comparison to other RDA-based approaches

Table 4.3: Sparsity $\sum_i I(|w^{(i)}| > 0)/d$

Dataset	Sparsity in % for UCI datasets					
	l_1-RDA$_{re}$ [67]	l_2-RDA$_{re}$ [68]	l_1-RDA$_{ada}$ [44]	l_1-RDA [134]	Pegasos [107]	Pegasos$_{drop}$
Pen Digits	98.6 (±4.6)	**26.1** (±19)	48.3 (±18)	40.5 (±18)	100 (±0.0)	100 (±0.0)
Opt Digits	94.0 (±4.1)	33.4 (±18)	36.6 (±12)	**32.5** (±10)	96.8 (±0.8)	94.8 (±13)
Shuttle	99.8 (±1.5)	**50.0** (±24)	52.8 (±20)	51.3 (±15)	100 (±0.0)	100 (±0.0)
Spambase	98.2 (±3.8)	**56.9** (±15)	57.7 (±17)	58.3 (±20)	100 (±0.0)	100 (±0.0)
Magic	93.2 (±12)	**30.8** (±7.2)	32.8 (±11)	37.2 (±15)	100 (±0.0)	100 (±0.0)
Covertype	92.8 (±14)	**8.0** (±5.4)	9.4 (±5.0)	12.4 (±7.1)	100 (±0.0)	98.1 (±14)
CNAE-9	**1.42** (±0.8)	2.86 (±3.7)	1.74 (±1.3)	1.74 (±1.3)	17.9 (±2.1)	14.3 (±1.8)
Semeion	4.82 (±6.7)	6.20 (±20.2)	1.33 (±3.2)	**0.11** (±0.8)	99.9 (±0.2)	99.8 (±0.2)
CT slices	84.7 (±18.7)	20.9 (±11.3)	**14.0** (±3.4)	14.4 (±4.2)	98.9 (±0.5)	98.8 (±0.4)
URI	0.06 (±0.06)	0.1 (±0.06)	0.04 (±0.05)	**0.03** (±.05)	1.4 (±0.07)	0.08 (±0.06)

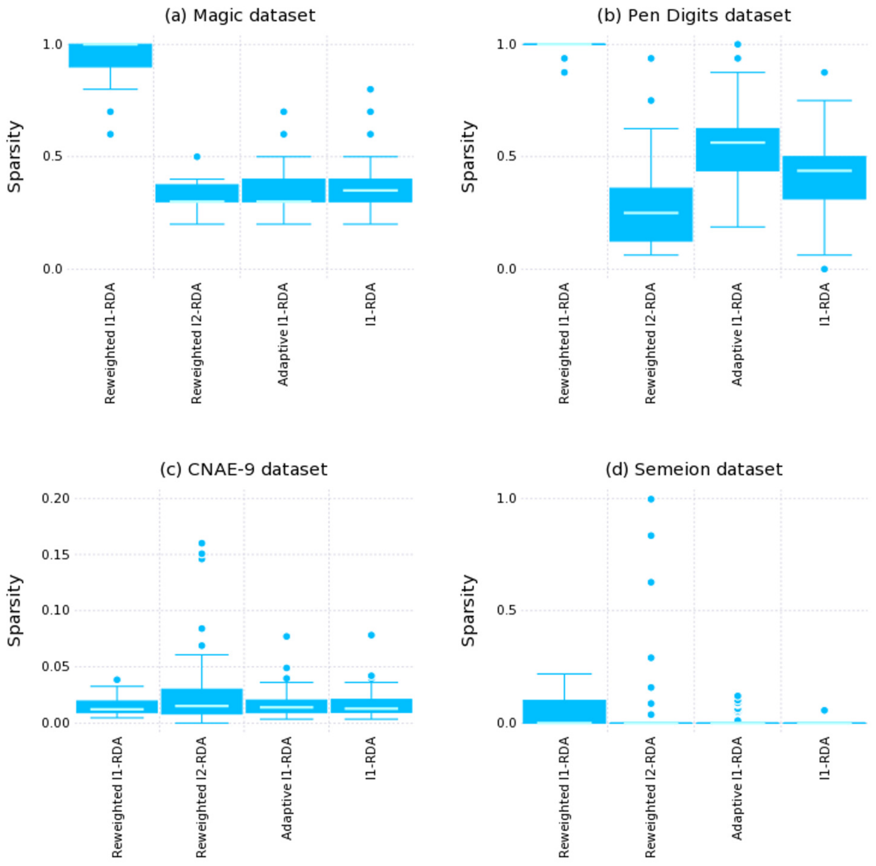


Figure 4.4: Comparison of the attained sparsity over 50 runs for different UCI datasets.

of the same interest in the online learning setup. We present both generalization- and sparsity-related performance in Table 4.4. Each row in the table represents the test and training errors (sparsity is given in brackets) of four different experiments in which we train our binary models *w.r.t.* one of the 4 major high-level categories, *i.e.*: Economics, Commerce, Medical, and Government (ECAT, CCAT, MCAT, GCAT respectively).

Numerical results are given only for l_1-RDA based approaches because we are interested in the comparison of sparsity-inducing methods within the same follow-the-regularized-leader (FTRL) framework, *i.e.* the l_1-norm regularization.

This approach gives a clear reference (*w.r.t.* the original l_1-RDA method) how good we can perform using adaptive [44] and reweighted [68] reformulations.

By analyzing Table 4.4 it is easy to verify that at least two approaches are performing equally good at each category *w.r.t.* the test error and there is no distinct winner in this case. Our results are very different from [44] but our experimental setup was slightly modified and we were using original TF-IDF features from [79]. In contrast the obtained training error and sparsity gives us more perspicuous outlook that the Reweighted l_1-RDA approach delivers better and faster learning rates in terms of generalization and sparsity while being relatively inferior only for one category, *i.e.* MCAT.

Stability

To test the stability of our approach (and specifically the Reweighted l_2-RDA method, which is less stable than the Reweighted l_1-RDA rival, due to the enforced and possibly unstable sparsification by `Sparsify(`w_t, ε`)` procedure in the end of Algorithm 10) we perform several series of experiments with UCI datasets to reveal the consistency and stability of our algorithm *w.r.t.* the obtained sparsity patterns. For every dataset first we tune the hyperparameters with all available data. We run our reweighted l_2-RDA approach and l_1-RDA [134] method 100 times in order to collect frequencies of every feature (dimension) being non-zero in the obtained solution. In Figure 4.5 we present the corresponding histograms. As we can see our approach results in much sparser solutions which are quite robust *w.r.t.* a sequence of random observations. l_1-RDA approach lacks these very important properties being relatively unstable under the stochastic setting.

Additionally we adopt an experimental setup from [32] where we create a toy dataset of sample size 10000, where every input vector a is drawn from a

Table 4.4: Performance on RCV1 dataset

Category	Training error (sparsity) (in %)			Test error in %		
	l_1-RDA$_{re}$	l_1-RDA$_{ada}$	l_1-RDA	l_1-RDA$_{re}$	l_1-RDA$_{ada}$	l_1-RDA
CCAT	**5.0** (32.7)	6.3 (47.5)	6.3 (**15.2**)	8.6	**7.9**	9.3
ECAT	**3.5** (24.9)	7.9 (28.5)	4.1 (**11.6**)	**6.5**	9.3	6.7
GCAT	**3.6** (**10.3**)	3.8 (40.0)	5.0 (11.7)	5.6	**5.0**	7.0
MCAT	6.6 (**9.5**)	3.8 (33.0)	**2.7** (10.3)	7.9	5.2	**4.7**

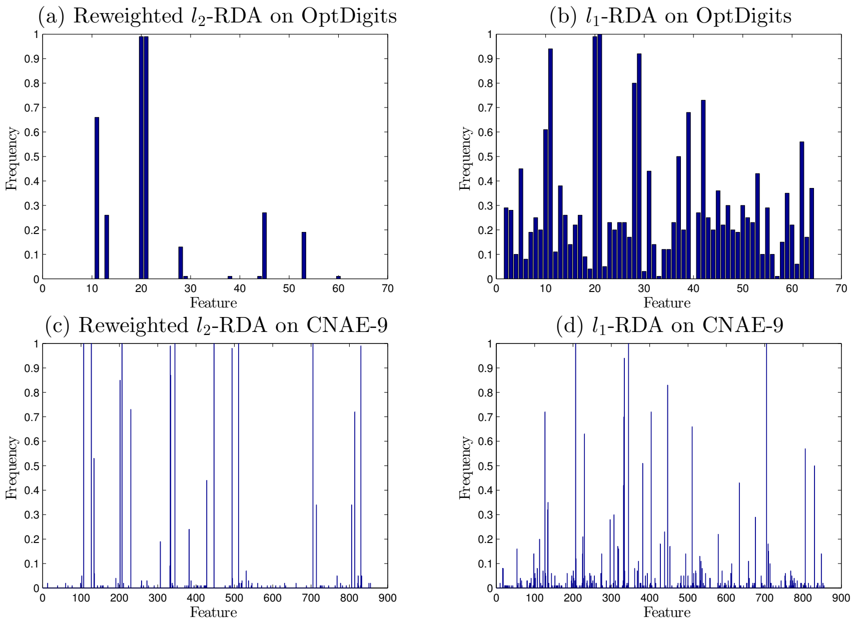


Figure 4.5: Frequency of being non-zero for the features of Opt Digits and CNAE-9 datasets. In the left subfigures (a,c) we present the results for the reweighted l_2-RDA approach, while the right subfigures (b,d) correspond to l_1-RDA method.

normal distribution $\mathcal{N}(0, I_{d\times d})$ and the output label is calculated as follows $y = \text{sign}(\langle w_*, a\rangle + \epsilon)$, where $w_*^{(i)} = 1$ for $1 \leq i \leq \lfloor d/2 \rfloor$ and 0 otherwise and the noise is given by $\epsilon \sim \mathcal{N}(0,1)$. We run each algorithm for 100 times and report the mean F1-score reflecting the performance of sparsity recovery. F1-score is defined as $2\frac{\text{precision}\times\text{recall}}{\text{precision}+\text{recall}}$, where

$$\text{precision} = \frac{\sum_{i=1}^{d} I(\hat{w}^{(i)}\neq 0, w_*^{(i)}=1)}{\sum_{i=1}^{d} I(\hat{w}^{(i)}\neq 0)}, \qquad \text{recall} = \frac{\sum_{i=1}^{d} I(\hat{w}^{(i)}\neq 0, w_*^{(i)}=1)}{\sum_{i=1}^{d} I(w_*^{(i)}=1)}.$$

Figure 4.6 shows that the reweighted l_2-RDA approach selects irrelevant features much less frequently as in comparison to l_1-RDA approach. As it was empirically verified before for UCI datasets we perform better both in terms of the stability of the selected set of features and the robustness to the stochasticity and randomness.

The higher the F1-score is, the better the recovery of the sparsity pattern. In Figure 4.7 we present an evaluation of our approach and l_1-RDA method *w.r.t.* to ability to identify the right sparsity pattern as the number of features

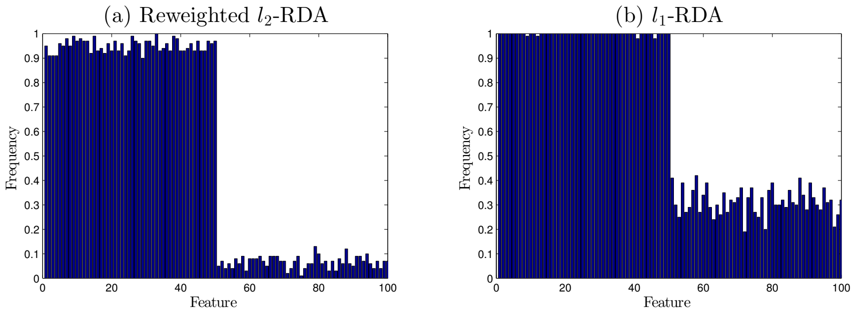


Figure 4.6: Frequency of being non-zero for the features of our toy dataset ($d = 100$). Only the first half of features do correspond to the encoded sparsity pattern. In the left subfigure (a) we present the results for the reweighted l_2-RDA approach, while the right subfigure (b) corresponds to l_1-RDA method.

increases. We clearly do outperform l_1-RDA method in terms of F1-score for $d \leq 300$. In conclusion we want to point out some of the inconsistencies discovered by comparing our F1-scores with [32]. Although the authors in [32] use a batch-version of the accelerated l_1-RDA method and a quadratic loss function they obtain very low F1-score (0.67) for the feature vector of size 100. In our experiments all F1-scores were above 0.7. For the dimension of size 100 our method obtains F1-score ≈ 0.95 while authors in [32] have only 0.87.

4.1.5 Conclusion

In this paper we studied reweighted stochastic learning in the context of dual averaging schemes and solvers for linear SVMs. We have presented two different directions for applying reweighting at each round t. The first approach helps to approximate very efficient l_0-type of a penalty using a reliable and proven dual averaging scheme [94]. We applied the reweighting procedure to different norms and elaborated two versions of the Regularized Dual Averaging method [134], namely Reweighted l_1- and l_2-RDA. The second approach stems from the Pegasos algorithm [107] and applies regularization based on resampling from the Bernoulli distribution where any success probability for any feature i depends upon the weighted value of the previous iterate $w_{t-1}^{(i)}$.

Our methods are suitable both for online and stochastic learning, while our numerical and theoretical results consider only the stochastic setting. We provided theoretical guarantees of the boundedness for the regret under different conditions. Experimental results validate the usefulness and promising capabilities of the proposed approaches in obtaining sparser, consistent and more

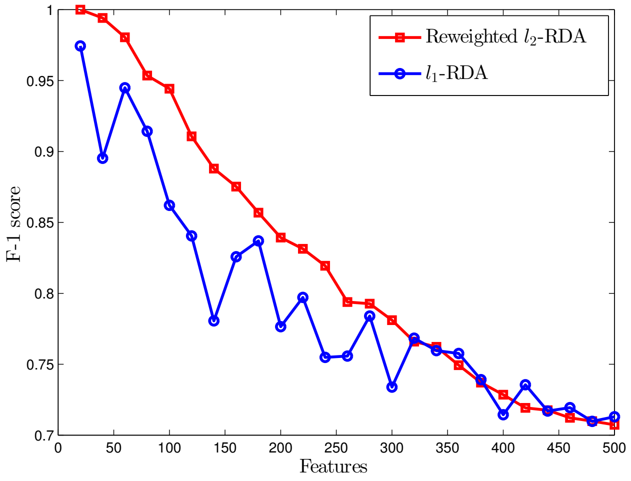


Figure 4.7: F1-score as the function of the number of features. We ranged the number of features from 20 to 500 with the step size of 20.

stable solutions while keeping the regret rates of the follow-the-regularized-leader methods at hand.

For the future we consider to improve our algorithms in terms of the accelerated convergence rate discussed in [32], [94] and develop some further extensions towards online and stochastic coordinate descent methods applied to the huge-scale[7] data.

4.2 Regularized and Sparse Stochastic K-Means for Distributed Large-Scale Clustering

4.2.1 Introduction

Clustering is considered as one of the cornerstones in the machine learning field. Many practical problems and applications of clustering are embedded into our daily lives and support decision making in various business domains. On the other hand the proliferation of data sources and exponentially growing data

[7]both in terms of dimensions and number of samples

volume are gaining more and more importance amid the greatest challenges in the machine learning and data governance fields [46, 2].

The K-Means algorithm [83] can be considered as one of the simplest and most scalable approaches which has been implemented and parallelized [34, 52] in numerous Big Data frameworks, like Mahout [98], Spark [137] etc. Despite its simplicity and obvious advantages it is known to be prone to instability due to the randomness in initialization [8]. One of the evident choices to stabilize performance of the K-Means algorithm is to apply stochastic learning paradigm. In this direction the interested reader can find only a few examples [73]. This particular scenario imposes stochasticity on the level of the re-current draw of some specific random variable which determines the segmentation and cluster memberships. In this setting one relies on the probabilistic measures dependent upon the distribution of per-sample distances to the centroids. Another way of approaching the same problem is a combination of stochastic gradient descent (SGD) and the K-Means optimization objective [21]. In the latter setting one seeks to find a new cluster centroid by observing one (or a small mini-batch) sample at iterate t and calculating the corresponding gradient descent step.

Another promising direction is the regularization with different norms. Recent developments [118, 132] indicate that this approach might be useful when one deals with high-dimensional datasets and seeks for a compressed (sparsified) solution. In [118] the authors propose to use an adaptive group Lasso penalty (variation of l_1-norm) [9] but obtain a solution per centroid in a conventional closed-form. To the best of our knowledge we are not aware of any K-Means algorithm combining together ideas of stochastic optimization with l_1-norm induced regularization applied to centroids through the dual averaging [93, 94, 68] scheme. In this paper we try to bridge the gap between regularized stochastic optimization and algorithmic schemes stemmed from the well-known and well-established K-Means approach. Additionally we devise an inherently distributed learning strategy where one finds a solution per prototype vector in parallel. This strategy requires only a limited number of outer synchronizations (iterations) to re-assign cluster memberships according to the proximity measure *w.r.t.* the prototype vectors (centroids).

This paper is structured as follows. Section 4.2.2 presents a problem statement for the regularized stochastic K-Means approach. Section 4.2.3 presents a stochastic strategy based on the Adaptive Dual Averaging [44] scheme. Section 4.2.4 refers the interested reader to the implementation details involving Big Data frameworks and concepts. Section 4.2.5 presents our numerical results while Section 4.2.6 concludes the paper.

4.2.2 Problem Statement and Proposed Method

To approach the well-established classical K-Means problem through the stochastic learning paradigm we approximate the K-Means optimization objective $f(w^{(i)}) = \frac{1}{2}\mathbb{E}_{x\in\mathcal{S}_i}\|w^{(i)} - x\|_2^2$ *w.r.t.* the i-th cluster ($i = 1, \dots, k$) by using a finite set of independent observations $\mathcal{S}_i = \{x_j\}_{1\le j\le N}$ belonging to this cluster. We add an additional regularization term $\psi(w^{(i)})$ as well. Under this setting one minimizes the following optimization objective for any i-th cluster with :

$$\min_{w^{(i)}} f(w^{(i)}) \triangleq \frac{1}{2N}\sum_{j=1}^{N}\|w^{(i)} - x_j\|_2^2 + \lambda\psi(w^{(i)}), \tag{4.18}$$

where $\psi(w^{(i)})$ represents a regularization term, λ is the trade-off hyperparameter and expectation is taken *w.r.t.* the set $\mathcal{S}_i$ with any $x_j \in \mathcal{S}_i$. The above optimization problem in Eq.(4.18) is a decoupled term of the global optimization objective involving all k clusters:

$$\min_{w^{(1)},\dots,w^{(k)}} \sum_{i=1}^{k}[\frac{1}{2N_i}\sum_{x\in\mathcal{S}_i}\|w^{(i)} - x\|_2^2 + \lambda\psi(w^{(i)})], \tag{4.19}$$

where $N_i = |\mathcal{S}_i|$ is the cardinality of the corresponding set $\mathcal{S}_i$. An entire superset $\hat{\mathcal{S}} = \{\mathcal{S}_i\}_{1\le i\le k}$ encompasses all samples from all k clusters encountered in Eq.(4.19). $\mathcal{S}_i$ subsets are disjoint and correspond to the individual non-overlapping clusters. We can implement Eq.(4.19) by the sequence of disjoint parallel optimization objectives learned via the stochastic optimization paradigm.

The core idea of the stochastic optimization paradigm is to optimize objective in Eq.(4.18) by the gradient descent step observing and taking at any step t some gradient $g_t \in \partial f(w_t^{(i)})$ *w.r.t.* only one sample x_t from $\mathcal{S}_i$ and the current iterate $w_t^{(i)}$ at hand. One usually draws a random sample from $\mathcal{S}_i$ until some ϵ-tolerance criterion is met or the total number of iterations is exceeded. It is common to acknowledge Eq.(4.18) as an online learning problem if $N \to \infty$.

In the above setting one deals with a simple clustering model $c(x) = \arg\min_i \|w^{(i)} - x\|_2$ and updates cluster memberships of the entire superset (dataset) $\hat{\mathcal{S}}$ after individual solutions $w^{(i)}$ (centroids) are found. We denote this update as an outer iteration (synchronization) and use it to fix $\mathcal{S}_i$ for learning each individual prototype vector $w^{(i)}$ in parallel.

4.2.3 l_1-Regularized Stochastic K-Means

Method

In this section we present a learning scheme induced by the l_1-norm regularization and corresponding dual averaging approaches [134] with adaptive primal-dual iterate updates [44]. This scheme allows sparsification of the prototype vectors and selection of the most important set of features. We begin with redefining our optimization objective in Eq.(4.18) in terms of a new $\psi(w^{(i)})$ function:

$$\min_{w^{(i)}} f(w^{(i)}) \triangleq \frac{1}{2N}\sum_{j=1}^{N} \|w^{(i)} - x_j\|_2^2 + \lambda\|w^{(i)}\|_1. \tag{4.20}$$

By using a *simple dual averaging* scheme [94] and adaptive strategy from [44] we can solve our non-smooth problem effectively by the following sequence of iterates $w_{t+1}^{(i)}$:

$$w_{t+1}^{(i)} = \arg\min_{w^{(i)}}\{\frac{\eta}{t}\sum_{\tau=1}^{t}\langle g_\tau, w^{(i)}\rangle + \eta\lambda\|w^{(i)}\|_1 + \frac{1}{t}h(w^{(i)})\}, \tag{4.21}$$

where $h_t(w^{(i)})$ is an adaptive strongly convex proximal term, g_t represents a gradient of the $\|w^{(i)} - x_t\|^2$ term *w.r.t.* only one randomly drawn sample $x_t \in \mathcal{S}_i$ and current iterate $w_t^{(i)}$ while η is a fixed step-size.

In the regularized Adaptive Dual Averaging (ADA) scheme [44] one is interested in finding a corresponding step-size for each coordinate which is inversely proportional to the time-based norm of the coordinate in the sequence $\{g_t\}_{t\geq 1}$ of gradients. This needs a careful design of an auxiliary adaptive term $h_t(w^{(i)}) = \langle w^{(i)}, H_t w^{(i)}\rangle$, where H_t depends on the aforementioned norm across each q-th coordinate in $\{g_t\}_{t\geq 1}$ sequence.

We can summarize a coordinate-wise update of the $w_t^{(i)}$ iterate in the adaptive dual averaging scheme as:

$$w_{t+1,q}^{(i)} = \text{sign}(-\hat{g}_{t,q})\frac{\eta t}{H_{t,qq}}[|\hat{g}_{t,q}| - \lambda]_+, \tag{4.22}$$

where $\hat{g}_{t,q} = \frac{1}{t}\sum_{\tau=1}^{t} g_{\tau,q}$ is the coordinate-wise mean across $\{g_t\}_{t\geq 1}$ sequence, $H_{t,qq} = \rho + \|g_{1:t,q}\|_2$ is the time-based norm of the q-th coordinate across the same sequence and $[x]_+ = \max(0, x)$.

Analyzing Eq.(4.22) we can find two crucial hyperparameters. The first one is λ and it trades off the importance of l_1-norm regularization in Eq.(4.20) while the second one (η) is necessary only for the proper convergence of the entire sequence of $w_t^{(i)}$ iterates.

Algorithm 12: l_1-Regularized Stochastic K-Means

```
Data: $\hat{\mathcal{S}}, \lambda > 0, \eta > 0, \rho > 0, T \geq 1, T_{out} \geq 1, k \geq 2, \epsilon > 0$
1  Initialize $\mathbf{W}_0$ randomly for all clusters $(1 \leq i \leq k)$
2  for $p \leftarrow 1$ to $T_{out}$ do
3      Initialize empty matrix $\mathbf{W}_p$
4      Partition $\hat{\mathcal{S}}$ by $c(x) = \arg\min_i \|\mathbf{W}^{(i)}_{p-1} - x\|_2$
5      for $\mathcal{S}_i \subset \hat{\mathcal{S}}$ in parallel do
6          Initialize $w^{(i)}_1$ randomly, $\hat{g}_0 = 0$
7          for $t \leftarrow 1$ to $T$ do
8              Draw a sample $x_t \in \mathcal{S}_i$
9              Calculate gradient $g_t = w^{(i)}_t - x_t$
10             Find the average $\hat{g}_t = \frac{t-1}{t}\hat{g}_{t-1} + \frac{1}{t}g_t$
11             Calculate $H_{t,qq} = \rho + \|g_{1:t,q}\|_2$
12             $w^{(i)}_{t+1,q} = \text{sign}(-\hat{g}_{t,q})\frac{\eta t}{H_{t,qq}}[|\hat{g}_{t,q}| - \lambda]_+$
13             if $\|w^{(i)}_t - w^{(i)}_{t+1}\|_2 \leq \epsilon$ then
14                 Append($w^{(i)}_{t+1}$, $\mathbf{W}_p$)
15                 return
16             end
17         end
18         Append($w^{(i)}_{T+1}$, $\mathbf{W}_p$)
19     end
20 end
21 return $\hat{\mathcal{S}}$ partitioned by $c(x) = \arg\min_i \|\mathbf{W}^{(i)}_{T_{out}} - x\|_2$
```

Algorithm

In this section we present an outline of our distributed stochastic l_1-regularized K-Means algorithm. Carefully going line by line we can notice that at the first line we start with initialization of a random matrix[8] $\mathbf{W}_0$ which serves as a proxy for the first partitioning of $\hat{\mathcal{S}}$. After initialization we perform T_{out} outer synchronization iterations where based on previously learned individual prototype vectors $w^{(i)}$ we recompute cluster memberships and re-partition $\hat{\mathcal{S}}$ (line 4).

After partitioning is done we run in parallel the Adaptive RDA scheme for our l_1-regularized optimization objective in Eq.(4.20) and concatenate the result with $\mathbf{W}_p$ by the `Append` function. When we exceed the total number of outer iterations

[8] of size $d \times k$, where d is the input dimension and k is the number of clusters.

T_{out} we exit with the final partitioning of $\hat{\mathcal{S}}$ by $c(x) = \arg\min_i \|\mathbf{W}_{T_{out}}^{(i)} - x\|_2$ where i denotes the i-th column of $\mathbf{W}_{T_{out}}$.

In Algorithm 12 the iterate $w_t^{(i)}$ has a closed form solution and depends on the dual average (and the sequence of gradients $\{g_t\}_{t\geq 1}$). Another important notice is the presence of some additional hyperparameters: the fixed step-size η and the additive constant ρ for making $H_{t,qq}$ term non-zero. Bringing additional degrees of freedom to the algorithm might be beneficial from the generalization perspective but it is compensated by the increased computational cost of cross-validation needed to estimate these degrees (hyperparameters).

4.2.4 Implementation Details

Learning of Prototype Vectors

In this subsection we will give a brief outlook on the implementation details of our Algorithm 12 involving Big Data frameworks and concepts like a Map-Reduce scheme [40]. Using the suggested architecture it is easy to extend our approach to the terascale data. In Figure 4.8 we show a schematic visualization of the Map-Reduce scheme for Algorithm 12. As we can notice the Map-Reduce scheme is needed to parallelize learning of individual centroids (prototype vectors) using our RDA-based approach in Algorithm 12. Each outer p-th iteration we `Reduce()` all learned centroids to the matrix $\mathbf{W}_p$ and re-partition the data again with `Map()`. After we reach T_{out} iterations we stop and re-partition the data according to the final solution and proximity to the prototype vectors.

Parallel Computing

We have implemented all our routines in `Julia` technical computing language[9]. In this subsection we will explain briefly how `Julia` is performing parallel computing and how we seamlessly managed to distribute computational burden without involving actual cluster setup and explicit usage of any MPI (Message Passing Interface) routines. An interested user may refer to `Julia` documentation[10] but in short `Julia` relies on the built-in routines defined in the base implementation of the language itself. Corresponding routines ensure that `Julia` workers instantiated at each node will communicate and pass messages to each other through the SSH connection (multiple options are supported).

[9]See `http://julialang.org/`

[10]See `http://julia.readthedocs.org/en/latest/manual/parallel-computing/`

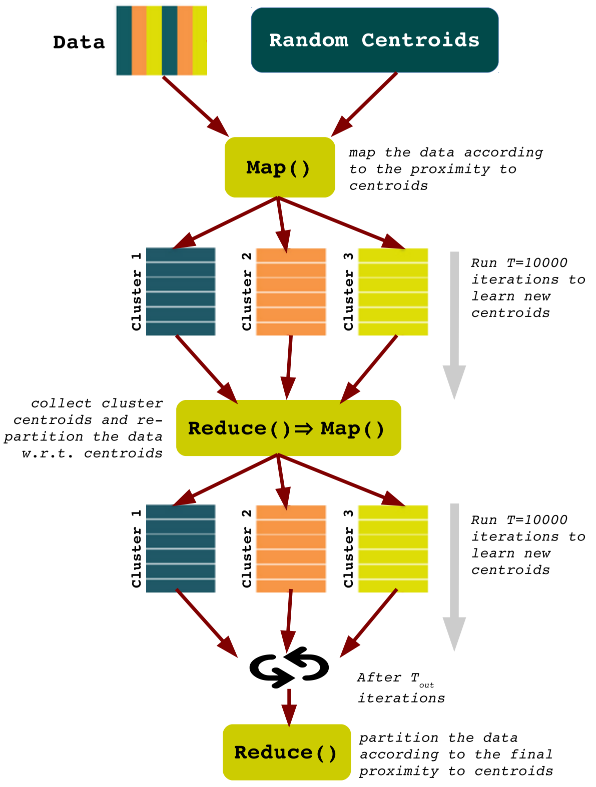

Data
Random Centroids
Map()
map the data according to the proximity to centroids
Cluster 1
Cluster 2
Cluster 3
Run T=10000 iterations to learn new centroids
collect cluster centroids and re-partition the data w.r.t. centroids
Reduce()⇒ Map()
Cluster 1
Cluster 2
Cluster 3
Run T=10000 iterations to learn new centroids
After T_out iterations
Reduce()
partition the data according to the final proximity to centroids

Figure 4.8: Schematic visualization of the Map-Reduce scheme for Algorithm 12.

Because of the independent learning of individual prototype vectors we used internal `@parallel (op)` macro command embedded into `Julia` language. This mechanism manages `for` loop in parallel and applies the reduce `op` operation to fold the results into a single output.

4.2.5 Experiments

Experimental Setup

In this section we describe our experimental setup. For all methods in our experiments we use UCI datasets [49] and datasets in [50]. Description of these datasets the interested user can find in Table 4.5. We compare our Stochastic Regularized K-means clustering with the randomized K-Means approach [83] and Proximal Plane Clustering (PPC) [109]. For all methods we know an exact number of clusters and set it as an input to all methods. In this setting K-Means approach does not require any tuning and for our approach and PPC we experiment with the range of $\{10^i | i = -2, -1, ..., 2\}$ for the trade-off hyperparameter[11].

All experiments were repeated 20 times (iterations) on a multicore machine[12]. We use Variation of Information (VI), Rand index and Adjusted Rand Index (ARI) as our performance measures for the comparison *w.r.t.* the ground truth. For our approach and PPC at each iteration we collect an average and the best measure across the aforementioned range of the hyperparameters. In the end for all measures we report an average, standard deviation and the best attained value across all 20 iterations. We report an average execution time for each method as well. For sparse datasets we additionally calculate sparsity as $\sum_{ij} I(|\mathbf{W}_{T_{out}}^{(ij)}| > 0)/(dk)$, where d is the input dimension, k is the total number of clusters and $\mathbf{W}_{T_{out}}^{(ij)}$ refers to the i-th column and j-th row of $\mathbf{W}_{T_{out}}$ which was explained in Section 21

For all presented stochastic algorithms we set $T_{out} = 20$, $T = 10000$, $\epsilon = 10^{-5}$. For Algorithm 12 we fixed $\eta = 1$ and $\rho = 0.1$. For PPC and randomized K-Means we set the number of outer iterations $T_{out} = 20$ to be the same as for our methods. All datasets are normalized. K-Means implementation was taken from `github.com/JuliaStats/Clustering.jl`. All methods were implemented in `Julia` technical computing language. Corresponding software can be found online at `www.esat.kuleuven.be/stadius/ADB/software.php` and `github.com/jumutc/SALSA.jl`.

Numerical Results

We present an exhaustive comparison with various algorithms in Table 4.6. All competitive algorithms have different modelling assumptions but we

[11] for our method in Eq.(4.20): λ, for PPC: c

[12] 20 cores were used to parallelize computations

Table 4.5: Datasets

Dataset	# attributes	# clusters	# data points
Magic	11	2	19020
Shuttle	9	2	58000
Skin	4	2	245057
Covertype	54	7	581012
Poker Hand	11	10	1025010
Higgs	28	2	11000000

have selected K-Means and PPC for a main comparison because of a small computational burden. For K-Means the time complexity is of order $\mathcal{O}(dNT_{out})$ while for PPC it is of order $\mathcal{O}(d^3NT_{out})$ if we distribute learning of individual prototype vectors or proximal hyperplanes. In the Proximal Plane Clustering approach we have to perform eigendecomposition of the linear combination of two co-variance matrices [109] so our cost is dominated by d if $d \gg N$.

If we compare execution times of all approaches we can notice that our methods are not the fastest ones because of the absence of a closed-form solution at hand. Instead our stochastic approaches utilize a fixed-size budget for learning individual prototype vectors. This budget is defined as follows: $\mathcal{B} = T_{out} \times T$. The latter implies the time complexity of order $\mathcal{O}(d\mathcal{B})$.

In Figure 4.9 we present a clustering visualization for the K-Means algorithm (left) and our l_1-Regularized Stochastic K-Means method (right) learned with Algorithm 12 as a qualitative example. As we can easily notice there is much less ambiguity for the bottom-right cluster in the case of l_1-Regularized Stochastic K-Means approach than for the classical K-Means algorithm. Additionally we can see slightly smaller number of different clusters merged together.

By analyzing Table 4.6 it is easy to verify that our l_1-Regularized Stochastic K-Means algorithm outperforms other approaches in terms of the scored performance metrics (VI, Rand index and ARI) especially for the best achievable scores. On average l_1-norm regularization helps to implicitly apply feature selection procedure while learning prototype vectors (centroids). In Table 4.7 we provide the obtained sparsity $\sum_{ij} I(|\mathbf{W}^{(i)}_{T_{out}}| > 0)/(dk)$ of the l_1-Regularized Stochastic K-Means approach on various datasets. We can observe that for some datasets like Covertype or Shuttle we can get quite sparsified results while for other datasets (Skin) the effective reduction is zero.

Table 4.6: Performance for large-scale datasets

Dataset		l_1-Regularized K-Means			K-Means [83]			PPC [109]		
		VI	Rand index	ARI	VI	Rand index	ARI	VI	Rand index	ARI
Magic	average	1.302	0.511	**0.017**	1.322	0.505	0.006	**1.234**	**0.512**	0.009
	std	0.051	0.016	0.026	0.000	0.000	0.000	0.188	0.016	0.017
	best	1.077	**0.588**	**0.154**	1.322	0.505	0.007	**0.728**	0.545	0.075
	time			22.515			0.040			0.032
Shuttle	average	**1.325**	**0.581**	**0.212**	1.477	0.538	0.206	1.491	0.558	0.125
	std	0.296	0.065	0.114	0.133	0.041	0.056	0.275	0.056	0.136
	best	**0.668**	0.709	0.440	1.183	0.648	0.359	0.800	0.684	0.401
	time			38.843			0.181			0.442
Skin	average	1.089	0.527	0.018	1.128	0.505	-0.030	**1.016**	**0.561**	**0.033**
	std	0.144	0.073	0.150	0.000	0.000	0.000	0.172	0.060	0.087
	best	**0.350**	**0.897**	**0.776**	1.128	0.505	-0.030	0.682	0.687	0.350
	time			145.724			0.283			0.220
Covertype	average	2.336	0.568	0.056	**2.334**	0.588	0.066	2.361	0.554	0.048
	std	0.451	0.076	0.027	0.129	0.015	0.024	0.463	0.067	0.030
	best	1.363	0.620	0.115	2.137	0.607	0.098	**1.263**	0.603	**0.143**
	time			214.742			3.790			14.133
Poker Hand	average	**3.220**	0.552	0.00029	3.282	**0.554**	0.00017	3.275	**0.554**	0.00015
	std	0.135	0.005	0.001	0.001	0.000	0.000	0.020	0.001	0.000
	best	**2.630**	**0.555**	**0.003**	3.278	0.554	0.001	3.144	0.554	0.002
	time			245.969			4.036			14.027
Higgs	average	1.202	0.504	0.006	**1.156**	**0.505**	**0.008**	1.321	0.501	0.002
	std	0.088	0.002	0.003	0.002	0.000	0.000	0.098	0.001	0.002
	best	1.132	**0.505**	0.008	1.153	**0.505**	0.008	**0.867**	**0.505**	**0.010**
	time			268.148			3.237			1.916

4.2.6 Conclusion

In this paper we presented a novel clustering approach based on the well-established methods of K-Means and regularized stochastic optimization. We devised a distributed algorithm where individual prototype vectors (centroids) are regularized and learned in parallel by the Adaptive Dual Averaging (ADA) scheme. In this scheme one needs to set carefully the step-size related to the smoothness and Lipschitz continuity of the optimization objective. Our comprehensive experimental studies with different large-scale datasets indicate the usefulness of the proposed methods for learning better prototype vectors

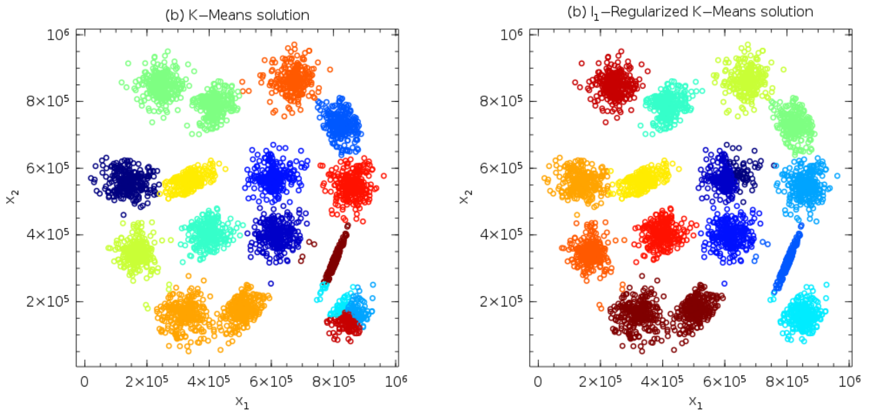


Figure 4.9: Clustering visualization for the (a, left) K-Means algorithm and (b, right) l_1-Regularized Stochastic K-Means algorithm on S1 dataset [50].

Table 4.7: Attained sparsity of the l_1-Regularized Stochastic K-Means

Dataset	average	std	minimum
Magic	0.762	0.299	0.050
Shuttle	0.651	0.253	0.111
Skin	1.000	0.000	1.000
Covertype	0.650	0.388	0.026
Poker Hand	0.812	0.276	0.230

while being able to perform feature selection by l_1-norm minimization at hand. In hindsight we have successfully applied the `MapReduce` scheme to distribute learning of individual prototype vectors. We have implemented all our routines in the inherently parallel and distributed `Julia` language which fits the requirements of the high-level and high-performance scientific computing applied to Big Data.

Chapter 5

Flexible Software Design & Julia Technical Computing Language

This chapter comprises currently submitted articles including:

1. Jumutc V., Suykens J.A.K., "**SALSA: Software Lab for Advanced Machine Learning with Stochastic Algorithms**", Internal Report 15-179, ESAT-SISTA, KU Leuven (Leuven, Belgium), 2015.

Keywords—stochastic learning; machine learning software; open source software design; application programming interface (API); `Julia` scientific computing language;

5.1 SALSA: Software Lab for Advanced Machine Learning with Stochastic Algorithms

5.1.1 Introduction

Stochastic algorithms can be considered as one of the cornerstones in learning from large data samples [108, 107]. Stochastic Gradient Descent (SGD) has pioneered and inspired a series of innovative machine learning (ML) methods applicable to datasets with millions and billions of instances [1]. The stochastic learning paradigm provides a theoretically well-grounded and reliable way of reaching an optimal solution within a fixed computational budget, for instance total number of iterations.

Many modern technological startups and mainstream companies utilize distributed and complex MapReduce eco-systems, like Hadoop[1] or Spark[2], to apply inherently sequential stochastic algorithms to their terascale data sources. This can be achieved by inherent parallelism at the cross-validation level or averaging schemes [1]. Nevertheless learning and understanding of low-level instrumentation and implementation details of the aforementioned eco-systems takes a lot of time. The latter prevents immediate and successful adaptation by the out-of-field scientists and practitioners. On the other hand the `Julia` language[3] is designed to address the requirements of high-performance numerical and scientific computing and can be easily launched and adapted by everyone who is familiar with Matlab[4] or any other scientific computing language. The `Julia` eco-system also features out-of-box cluster setup routines to seamlessly launch parallel computations[5]. In Figure 5.1 we present a logo of the `Julia` technical and scientific computing language.

In order to offer the scientific community a machine learning tool which is easy to learn and apply to any-scale data we have devised and implemented SALSA. In comparison to existing software packages, like LIBLINEAR [47] or Vowpal Wabbit [76], which are written in pure C++ and oriented to high-performance and high-throughput operations, SALSA embraces high-performance features of the `Julia` eco-system with the interpretability and flexibility of the language itself. We provide users of various technical and mathematical backgrounds with interfaces and APIs to find the easiest learning curve to their data.

[1]See `http://hadoop.apache.org`
[2]See `http://spark.apache.org`
[3]See `http://julialang.org`
[4]See `http://mathworks.com/products/matlab`
[5]See `http://julia.readthedocs.org/en/latest/manual/parallel-computing`

Figure 5.1: `Julia` language logo.

5.1.2 Description

SALSA (Software lab for Advanced machine Learning with Stochastic Algorithms) is a native `Julia` implementation of the well-known stochastic algorithms [134, 107, 44] for Regularized Empirical Risk minimization [124] and sparse linear modelling [53]. It aims at bridging a gap between sophisticated machine learning concepts and user-friendly environment with API which guides the user through many intermediate selection steps. The API features low- and high-level routines for both an experienced and a novice user. SALSA is managed and versioned by the package manager embedded in `Julia`.

The SALSA software package can be used as a framework and a library. As a framework it embraces all steps of creating a valid ML model, such as cross-validation and hyperparameter tuning. As a library it can be used for embedding separate core algorithmic and preprocessing routines into other frameworks as well. Nice interoperability between the `Julia` eco-system and other languages (such as C/C++, Python etc.) ensures seamless and quick integration.

In comparison to other frameworks [47, 76] the SALSA package features most of the underlying functionality, including several loss functions (hinge loss, logistic loss, least squares loss *etc.*) and various regularization schemes (l_2-norm, elastic-net regularization *etc.*), and in the meanwhile complements some unique features related to l_0 regularization [70], Nyström approximation of feature maps [129] and cross-validation with different criteria. SALSA features also a separate clustering sub-routine which implements a Regularized Stochastic K-Means approach [69].

The package is intended to be helpful both in research and real-life applications. To foster a quick knowledge transfer we have devised a separate question and answer (Q&A) table which guides and helps a novice user to select an appropriate machine learning model which fits a given dataset (see Listing 5.2).

The `SALSA` software library embraces many of our previously presented methods in Chapters 3–4 such as the Reweighted l_1- and l_2-Regularized Dual Averaging [68, 67, 70], regularized stochastic K-Means approach [69] as well as different versions of the Pegasos algorithm [107] enhanced by different loss functions and the Nytröm approximation [63].

5.1.3 API usage

In this section we present a brief outlook on SALSA's available API and possible use cases. In Listing 5.1 we present the most basic machine learning classification example. It suffices to load the data or to define an appropriate file wrapper[6] and then execute one of SALSA's many interface aliases. The resulting *model* and *model.output* contains a lot of useful information and a summary of the obtained hyperparameters, estimated linear model and cross-validation score. For the detailed and more exhaustive examples please refer to `http://salsajl.readthedocs.org/en/latest/examples.html`.

Listing 5.1: Basic classification example

```
# load packages
using SALSA, MAT
# Load Ripley data
data = matread(joinpath(Pkg.dir("SALSA"),"data","ripley.mat"))
# Train and cross-validate a classification model
model = salsa(data["X"], data["Y"], data["Xt"])
# Evaluate a prediction for data["Xt"]
@printf "Accuracy: %.2f" mean(model.output.Ytest .== data["Yt"])
# Or map (normalize) data beforehand and make a prediction
prediction = map_predict(model, data["Xt"])
@printf "Accuracy: %.2f" mean(prediction .== data["Yt"])
```

In Listing 5.2 we present another interesting and promising use-case. In this example we guide a novice user or any other out-of-field researcher/practitioner through many intermediate steps to select an appropriate machine learning model which fits any type of the dataset provided to *salsa_qa(...)* routine. In this example we suggest a default choice which is appropriate in the given context. Listing 5.2 presents only a small snippet of the selection process and the entire Q&A graph is not given for brevity.

[6]See documentation at `http://salsajl.readthedocs.org`

Listing 5.2: Q&A example

```
using SALSA

model = salsa_qa(readcsv(joinpath(Pkg.dir("SALSA"),"data","iris.data.csv")))

Do you have any target variable of interest in X (or ENTER for default 'yes')? [y/n]:

Please provide the column number of your target variable (or ENTER for default last column):

Is your problem of the classification type (or ENTER for default 'yes')? [y/n]:

Please select a loss function from options (or ENTER for default)
    1 : SALSA.PINBALL (Pinball (quantile) Loss, i.e. $l(y,p) = \begin{cases} 1-yp, & \text{ifyp} \leq 1, \\ \tau(yp-1), & \text{otherwise} \end{cases}$ )
    2 : SALSA.HINGE (Hinge Loss, i.e. $l(y,p) = \max(0, 1-yp)$) (default)
    3 : SALSA.LEAST_SQUARES (Squared Loss, i.e. $l(y,p) = \frac{1}{2}(p-y)^2$)
    4 : SALSA.LOGISTIC (Logistic Loss, i.e. $l(y,p) = \log(1+\exp(-\text{yp}))$)
    5 : SALSA.MODIFIED_HUBER (Modified Huber Loss, i.e. $l(y,p) = \begin{cases} -4yp, & \text{if yp} < -1 \\ \max(0,1-yp)^2, & \text{otherwise} \end{cases}$ )
    6 : SALSA.SQUARED_HINGE (Squared Hinge Loss, i.e. $l(y,p) = \max(0,1-yp)^2$)

Please select a cross-validation (CV) criterion from options (or ENTER for default)
    1 : SALSA.AUC (Area Under ROC Curve with 100 thresholds)
    2 : SALSA.MISCLASS (Misclassification Rate) (default)
    3 : SALSA.MSE (Mean Squared Error)
```

5.1.4 Benchmark tests

In this section we present a benchmark test for the SALSA package and a comparison of our platform with other state-of-the-art libraries, such as LIBLINEAR [47] and Vowpal Wabbit (VW) [76]. In Table 5.1 we report training and test errors (misclassification rates) for two UCI datasets[7] taken from the LIBSVM repository[8]. For the RCV1 dataset we report a test error while for Covtype a training error. In brackets an interested reader can find execution times. All experiments were performed on a single thread of a multi-core machine with the Core i7 processor and 8GB of RAM. We present results for different regularization schemes[9] applied jointly with the logistic loss function. For Vowpal Wabbit (VW) and SALSA we present the evaluation in the online learning setting as well. For LIBLINEAR an online learning option is not provided and only specific solvers are available (for instance Conjugate Gradient (CG) solver). We report only the best achievable misclassification rate within the regularization path of the trade-off hyperparameter. The range of the hyperparameter was fixed to $[10^{-7} \dots 10^{2}]$.

[7]See http://archive.ics.uci.edu/ml

[8]See http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html

[9]$\|w\|_1$ stands for l_1-norm regularization, $\|w\|_2$ represents l_2-norm regularization

Table 5.1: Benchmark tests

Dataset		VW (online)		VW (CG)		LIBLINEAR		SALSA (online)		SALSA (stoch.)	
		$\|w\|_1$	$\|w\|_2$	$\|w\|_1$	$\|w\|_2$	$\|w\|_1$	$\|w\|_2$	$\|w\|_1$	$\|w\|_2$	$\|w\|_1$	$\|w\|_2$
RCV1	error	0.041	0.041	0.519	0.519	0.041	0.038	0.039	0.049	0.041	0.046
	time	(0.3s)	(0.4s)	(7.7s)	(8.1s)	(0.6s)	(0.9s)	(43.7s)	(15.3s)	(11.6s)	(5.3s)
Covtype	error	0.396	0.445	0.245	0.245	0.243	0.244	0.487	0.398	0.253	0.253
	time	(1.2s)	(1.2s)	(7.6s)	(7.5s)	(19.8s)	(3.4s)	(18.4s)	(8.5s)	(6.6s)	(6.6s)

As we can figure out from Table 5.1 the obtained misclassification rates are very similar for all approaches. Another observation is that Vowpal Wabbit library completely fails for the entire regularization path in the online case for the Covtype dataset, and with the CG solver for RCV1 dataset.

In comparison to dedicated and performance oriented libraries, like LIBLINEAR or VW, SALSA's timing results might not be very impressive because we take into account completely different learning and optimization approaches as a reference point here. In the SALSA package we bounce into the fixed computational budget in case of not reaching a termination threshold for subsequent iterates. For particularly big datasets stochastic approaches might significantly reduce computational burden bringing desired levels of accuracy within a fixed and constrained budget.

5.1.5 Conclusion

SALSA is an open source platform and a library for stochastic learning applied to the most common machine learning problems such as classification, clustering, and regression. It also provides an API for developers and practitioners which alleviates understanding of stochastic algorithms while simplifying the use of machine learning methods for large-scale problems. SALSA source code is hosted on GitHub. Full documentation which includes an API reference, use-cases and mathematical background can be found at `http://salsajl.readthedocs.org`. SALSA contains a runnable test suite on each commit to the GitHub repository via a continuous integration server. Finally, SALSA is released under the GNU General Public License version 3.0.

Chapter 6

Conclusions and Future Work

6.1 General Conclusions

In this thesis we have explored many different aspects of the machine learning field applied in the context of modern software design principles and the development of open source software. We have devised and explored a novel machine learning methodology which is applicable in various domains and can be effectively implemented using an emerging technical computing language `Julia`.

In Chapter 2 we have presented two related kernel-based techniques which extend concepts of unsupervised and semi-supervised learning within the classification and novelty detection domain. First we presented the problem of finding a support of unknown high-dimensional distributions in the presence of labeling information, called Supervised Novelty Detection. The One-Class Support Vector Machine is a widely used kernel-based technique to address this problem. However with the latter approach it is difficult to model a mixture of distributions from which the support might be constituted. We addressed this issue by presenting a new class of SVM-like algorithms which helped to approach multi-class classification and novelty detection from a new perspective. We introduced a new coupling term between classes which leverages the problem of finding a good decision boundary while preserving the compactness of a support with the l_2-norm penalty. We presented our optimization objective in the primal and then derived a dual Quadratic Programming (QP) formulation of the problem. Next we proposed a Least-Squares formulation which resulted in a linear system which drastically reduced computational costs. Finally we have derived a

Pegasos-based formulation which can effectively cope with large datasets that cannot be handled by many existing QP solvers. We validated the usefulness and practical importance of the proposed methods both in classification and novelty detection settings. The second part of Chapter 2 is devoted to a novel semi-supervised classification approach which combines bilinear formulation for non-parallel binary classifiers based upon Kernel Spectral Clustering. The cornerstone of our approach is a bilinear term introduced into the primal formulation of semi-supervised classification problem. In addition we performed separate manifold regularization for each individual classifier. The latter relates to the Kernel Spectral Clustering unsupervised counterpart which helps to obtain more precise and generalizable classification boundaries. We derived the dual problem which can be effectively translated into a linear system of equations and then solved without introducing extra costs. We showed the usefulness and reported considerable improvements in performance with respect to other semi-supervised approaches, like Laplacian SVMs and other KSC-based models.

In Chapter 3 we presented a stochastic learning setting for learning linear Support Vector Machines in primal. First we extended a widely acknowledged algorithm for learning linear SVMs, namely Pegasos. It utilizes properties of hinge loss and theory of strongly convex optimization problems for fast convergence rates and lower computational and memory costs. We have adopted the recently proposed pinball loss for the Pegasos algorithm and emphasized some advantages of using it in a variety of classification problems. We presented a derivation of the Pegasos optimization objective with respect to pinball loss and analyzed its properties and convergence rates. Additionally we presented extensions of the Pegasos algorithm applied to the kernel-induced and Nyström approximated feature maps which introduce non-linearity in the input space. This was done using a Fixed-Size kernel method approach. We gave experimental results for publicly available UCI datasets to justify the advantages and the importance of pinball loss for achieving a better classification accuracy and greater numerical stability in the partially or fully stochastic setting. Secondly in Chapter 3 we devised a new weighted formulation of the Pegasos algorithm which favored from the different coordinate-wise λ_i regularization parameters. Together with the proposed extension we gave a brief theoretical justification of its convergence to an optimal solution and analysed at a glance its computational costs. We demonstrated the merits and the importance of our approach for achieving a better classification accuracy and convergence rates in the partially or fully stochastic setting.

In Chapter 4 we presented an extension to the Regularized Empirical Risk Minimization framework which bears upon ideas of sparser norms and regularization based on the reweighted algorithmic schemes. First we studied reweighted schemes for stochastic learning (specifically in the context of classification problems) based on linear SVMs and dual averaging methods with primal-dual iterate updates. All these methods favour properties of a convex and composite optimization objective. The latter consists of a convex regularization term and loss function with Lipschitz continuous subgradients, *e.g.* l_1-norm ball together with hinge loss. Some approaches approximate in a limit the l_0-type of a penalty. In our analysis we focused on a regret and convergence criteria of such an approximation. We derived our results in terms of a sequence of convex and strongly convex optimization objectives. These objectives are obtained via the smoothing of a generic sub-differential and possibly non-smooth composite function by the global proximal operator. We reported an extended evaluation and comparison of the reweighted schemes against different state-of-the-art techniques and solvers for linear SVMs. We have shown that reweighted schemes can outperform state-of-the-art traditional approaches in terms of generalization error as well. The second part of Chapter 4 is devoted to a novel clustering approach based on the stochastic learning paradigm and regularization with l_1-norms. Our approach is an extension of the widely acknowledged K-Means algorithm. We introduced a simple regularized dual averaging scheme for learning prototype vectors (centroids) with l_1-norms in a stochastic mode. In our approach we distributed the learning of individual prototype vectors for each cluster, and the re-assignment of cluster memberships is performed only for a fixed number of outer iterations. The latter approach is exactly the same as in original K-Means algorithm and aims at re-shuffling the pool of samples per cluster according to the learned centroids. We reported an extended evaluation and comparison of our approach with respect to various clustering techniques like randomized K-Means and Proximal Plane Clustering. Our experimental studies have indicated the usefulness of the proposed methods for obtaining better prototype vectors and corresponding cluster memberships while being able to perform feature selection by l_1-norm minimization.

Finally in Chapter 5 we presented SALSA (Software lab for Advanced machine Learning with Stochastic Algorithms) which is an implementation of the well-known and presented in Chapters 3– 4 stochastic algorithms for Machine Learning developed in the high-level technical computing language `Julia`. The SALSA software package is designed to address challenges in sparse linear modelling, linear and non-linear Support Vector Machines applied to large data samples with user-centric and user-friendly emphasis. We implemented SALSA to alleviate and enhance user experience with ubiquitous Machine Learning problems including classification, regression and clustering. We provided users with different levels of the technical and mathematical background with various interfaces and APIs to find the most suitable way of learning from their data. SALSA is an open source project available at `http://github.com/jumutc/SALSA.jl` under the GNU General Public License version 3.0. It is freely accessible online for collaborative development and support, issue tracking and further modifications.

Thus, in summary we proposed several novel kernel-based and linear machine learning approaches suitable for classification, regression and clustering problems. In the end we aggregated and implemented most of them in a novel machine learning library, namely SALSA. We demostrated that machine learning problems and methodology can be effectively tackled by flexible software design principles embedded into the high-level technical computing language `Julia`.

6.2 Future Work

Some possible future research directions may consider extending the ideas of unsupervised and semi-supervised learning delivered by the Kernel Spectral Clustering framework into the Deep Learning [15, 77] domain. We hope for interesting and promising extensions of Deep Learning architectures which could be applied for learning cluster memberships immediately by feeding the data to the neural network or creating the corresponding pre-trained instance of the network.

Another possible direction of research can be originated in Big Data applications and research. This is an emerging field with a lot of difficult challenges and problems deserving a very careful attention. For instance semi-supervised learning within the Big Data field is quite unexplored and possesses a huge potential benefit if tackled correctly. As we can see an implicit manifold regularization and clustering principles play an important role in semi-supervised learning. One of the obvious research directions would be to explore these principles in Deep Artificial Neural Networks.

Another interesting research direction can be devoted to devising better optimization strategies for stochastic learning approaches when dealing with novel loss functions or regularization schemes. Group LASSO [10], graph-based LASSO [60] and other types of more specific LASSO regularization schemes [53] deserve better theoretical and empirical verification in the stochastic learning domain.

Finally implementation of scientific and machine learning software libraries implies another promising research direction. From the theoretical Computer Science perspective we opt for more elaborate, flexible and user- as well as developer-oriented and friendly application programming interfaces (APIs) and modules!

Appendix A

Proofs and theorems

A.1 Proofs for the Reweighted l_1-RDA method

A.1.1 Proof of Theorem 4

We start with redefining the conjugate-type functions $V_t(s)$ and $U_t(s)$ in [134] by replacing $\psi(w)$ in each of them with the sum of our reweighted l_1 functions $\psi_{l_1,t}(w) \triangleq \lambda \|\Theta_t w\|_1$, such that:

$$U_t(s) = \max_{w \in \mathcal{F_D}} \{\langle s, w - w_0 \rangle - \sum_{\tau=1}^{t} \psi_{l_1,\tau}(w)\}, \tag{A.1}$$

$$V_t(s) = \max_{w} \{\langle s, w - w_0 \rangle - \sum_{\tau=1}^{t} \psi_{l_1,\tau}(w) - \beta_t h_{l_1}(w)\}, \tag{A.2}$$

where we define the domain of any $\psi_{l_1,\tau}(w)$ function to be identical and set as $\mathcal{F}_D = \{w \in \mathrm{dom}\psi_{l_1} | h_{l_1}(w) \leq D\}$ (*i.e.* the domain of a simple non-reweighted l_1-norm as in [134]), $\{\beta_t\}_{t\geq 1}$ is a non-negative and non-decreasing sequence and $h_{l_1}(w)$ is a 1-strongly convex function defined in Eq.(4.5).

Next we refer to the important Lemma 9 in [134], as for any $s \in E^*$ and $t \geq 0$ we have: $U_t(s) \leq V_t(s) + \beta_t D$, because using the definition of $\mathcal{F}_D$ we can bound

$U_t(s)$ as follows:

$$\begin{aligned} U_t(s) &= \max_{w \in \mathcal{F}_{\mathcal{D}}} \{\langle s, w - w_0 \rangle - \sum_{\tau=1}^{t} \psi_{l_1,\tau}(w)\} \\ &= \max_{w} \min_{\beta \geq 0} \{\langle s, w - w_0 \rangle - \sum_{\tau=1}^{t} \psi_{l_1,\tau}(w) + \beta(D - h_{l_1}(w))\} \\ &\leq \min_{\beta \geq 0} \max_{w} \{\langle s, w - w_0 \rangle - \sum_{\tau=1}^{t} \psi_{l_1,\tau}(w) + \beta(D - h_{l_1}(w))\} \\ &\leq \max_{w} \{\langle s, w - w_0 \rangle - \sum_{\tau=1}^{t} \psi_{l_1,\tau}(w) + \beta_t(D - h_{l_1}(w))\} \\ &= V_t(s) + \beta_t D. \end{aligned} \tag{A.3}$$

□

We will need this lemma to bound the regret as it will be shown below. Before bounding the regret we need to adjust one more lemma from [134]:

Lemma 3. *For each $t \geq 1$ we have $V_t(-s_t) + \psi_{l_1,t}(w_t) \leq V_{t-1}(-s_t)$.*

Proof.

$$\begin{aligned} V_{t-1}(-s_t) &= \max_{w} \{\langle s_t, w - w_0 \rangle - \sum_{\tau=1}^{t-1} \psi_{l_1,\tau}(w) - \beta_{t-1} h_{l_1}(w)\} \\ &\geq \langle s_t, w_{t+1} - w_0 \rangle - \sum_{\tau=1}^{t-1} \psi_{l_1,\tau}(w_{t+1}) - \beta_{t-1} h_{l_1}(w_{t+1}) \\ &\geq \{\langle s_t, w_{t+1} - w_0 \rangle - \sum_{\tau=1}^{t} \psi_{l_1,\tau}(w_{t+1}) - \beta_t h_{l_1}(w_{t+1})\} + \psi_{l_1,t}(w_{t+1}) \\ &= V_t(-s_t) + \psi_{l_1,t}(w_{t+1}) \geq V_t(-s_t) + \psi_{l_1,t}(w_t). \end{aligned}$$

□

In this proof the first line follows from the definition of $V_t(s)$. The second line is immediately implied by the maximization objective. On the third line we used the main property of $\{\beta_t\}_{t\geq 1}$ sequence to be non-decreasing and non-negative. We derived an equality on the final line by noticing that the expression in curly brackets is exactly $V_t(-s_t)$ (*i.e.* the solution of the maximization problem defined in $V_{t-1}(-s_t)$ is exactly w_{t+1} by our algorithmic design in Eq.(4.2). And the final inequality follows from our assumption $\psi_{l_1,t}(w_t) \leq \psi_{l_1,t}(w_{t+1})$.

In the next part we will finally bound our regret function defined in Eq.(4.7). From [94] and [134] we know that if we consider $\Delta\psi_{l_1,\tau} = \psi_{l_1,\tau}(w_\tau) - \psi_{l_1,\tau}(w)$ the following *gap sequence* δ_t:

$$\delta_t = \max_w\{\sum_{\tau=1}^{t}(\langle g_\tau, w_\tau - w\rangle + \Delta\psi_{l_1,\tau})\} \geq \sum_{\tau=1}^{t}(f(w_\tau,\xi_\tau) - f(w,\xi_\tau) + \Delta\psi_{l_1,\tau}) = \mathbf{R}_t(w) \tag{A.4}$$

bounds the regret function from above due to the convexity of f [22]. Next we can derive an upper bound on δ_t using the properties of the conjugate-type functions $V_t(s)$ and $U_t(s)$.

If we add and subtract the $\sum_{\tau=1}^{t}\langle g_\tau, w_0\rangle$ term from the definition in Eq.(A.4)[1] then for any $w \in \mathbb{R}^n$ we get:

$$\begin{aligned}\delta_t &= \sum_{\tau=1}^{t}(\langle g_\tau, w_\tau - w_0\rangle + \psi_{l_1,\tau}(w_\tau)) + \max_w\{\langle s_t, w_0 - w\rangle - \sum_{\tau=1}^{t}\psi_{l_1,\tau}(w)\} \\ &\leq \sum_{\tau=1}^{t}(\langle g_\tau, w_\tau - w_0\rangle + \psi_{l_1,\tau}(w_\tau)) + V_t(-s_t) + \beta_t D, \end{aligned} \tag{A.5}$$

where $s_t = \sum_{\tau=1}^{t} g_\tau$. The last maximization term in Eq.(A.5) is exactly $U_t(-s_t)$ so it can be bounded by Eq.(A.3).

Applying well-known results on the boundedness of the conjugate-type functions from [93, 94] for any $\tau \geq 1$ and in view of Lemma 3 we have:

$$\begin{aligned} V_\tau(-s_\tau) + \psi_{l_1,\tau}(w_\tau) &\leq V_{\tau-1}(-s_\tau) \\ &= V_{\tau-1}(-s_{\tau-1} - g_\tau) \\ &\leq V_{\tau-1}(-s_{\tau-1}) + \langle -g_\tau, \nabla V_{\tau-1}(-s_{\tau-1})\rangle + \frac{\|g_\tau\|_*^2}{2\beta_{\tau-1}} \\ &= V_{\tau-1}(-s_{\tau-1}) + \langle -g_\tau, \hat{w}_\tau - w_0\rangle + \frac{\|g_\tau\|_*^2}{2\beta_{\tau-1}}, \end{aligned} \tag{A.6}$$

[1] for derivations below we have to substitute general index t with the running index $\tau \in \overline{1,t}$

where $\|\cdot\|_*$ is a dual norm, second inequality is due to the result of [93]: $V_\beta(s+g) \leq V_\beta(s) + \langle g, \nabla V_\beta(s)\rangle + \|g\|_*/2\sigma\beta,\ \forall s, g \in E^*$, the fact that $\sigma = 1$ for $h_{l_1}(w)$ and

$$\nabla V_{\tau-1}(-s_{\tau-1}) = \hat{w}_\tau - w_0,\ \forall \tau \geq 1$$

where $\hat{w}_\tau \triangleq \arg\min_w \{\langle s_{\tau-1}, w\rangle + \sum_{\kappa=1}^{\tau-1} \psi_{l_1,\kappa}(w) + \beta_{\tau-1} h_{l_1}(w)\},\ \forall \tau \geq 1$.

Next we rearrange and sum up everything *w.r.t.* the index τ:

$$\sum_{\tau=1}^{t} (\langle g_\tau, \hat{w}_\tau - w_0\rangle + \psi_{l_1,\tau}(w_\tau)) + V_t(-s_t) \leq V_0(-s_0) + \sum_{\tau=1}^{t} \frac{\|g_\tau\|_*^2}{2\beta_{\tau-1}}. \tag{A.7}$$

We can further simplify this equation by assuming $V_0(-s_0) = V_0(0) = 0$ and from the first-order optimality conditions of the function f as follows:

$$0 \leq \langle g_\tau, \hat{w}_\tau - w_\tau\rangle \Longrightarrow \langle g_\tau, w_\tau\rangle \leq \langle g_\tau, \hat{w}_\tau\rangle.$$

Substituting all of above into Eq.(A.7) and taking into account Eq.(A.5) we get:

$$\mathbf{R}_t(w) \leq \delta_t \leq \beta_t D + \sum_{\tau=1}^{t} \frac{\|g_\tau\|_*^2}{2\beta_{\tau-1}}. \tag{A.8}$$

To obtain bound in Theorem 4 we assume $\|g_t\|_* \leq G$ and set $\beta_t = \gamma\sqrt{t}$, while keeping $\beta_0 = \beta_1 = \gamma$:

$$\mathbf{R}_t(w) \leq \gamma\sqrt{t}D + \frac{G^2}{2\gamma}(1 + \sum_{\tau=1}^{t-1} \frac{1}{\sqrt{\tau}}) \leq (\gamma D + \frac{G^2}{\gamma})\sqrt{t}, \tag{A.9}$$

where $\sum_{\tau=1}^{t-1} \tau^{-\frac{1}{2}} \leq 1 + \int_1^t \tau^{-\frac{1}{2}} d\tau = 2\sqrt{t} - 1$.

□

A.1.2 Proof of Theorem 5

The main part of the proof is exactly the same as for Theorem 4. We will refer to the parts of the proof of Theorem 4 as we will need them.

First we need to adjust Lemma 1 and introduce the dependency on the sufficient conditions of Theorem 5.

Lemma 4. *For each $t \geq 1$ we have $V_t(-s_t) + \psi_{l_1,t}(w_t) \leq V_{t-1}(-s_t) + \nu/t$.*

Proof.

$$\begin{aligned}
V_{t-1}(-s_t) &= \max_w \{\langle s_t, w - w_0\rangle - \sum_{\tau=1}^{t-1} \psi_{l_1,\tau}(w) - \beta_{t-1} h_{l_1}(w)\} \\
&\geq \langle s_t, w_{t+1} - w_0\rangle - \sum_{\tau=1}^{t-1} \psi_{l_1,\tau}(w_{t+1}) - \beta_{t-1} h_{l_1}(w_{t+1}) \\
&\geq \{\langle s_t, w_{t+1} - w_0\rangle - \sum_{\tau=1}^{t} \psi_{l_1,\tau}(w_{t+1}) - \beta_t h_{l_1}(w_{t+1})\} + \psi_{l_1,t}(w_{t+1}) \\
&= V_t(-s_t) + \psi_{l_1,t}(w_{t+1}).
\end{aligned}$$

And to get the final result we use our assumption in Theorem 5 that we have $\psi_{l_1,t}(w_t) - \psi_{l_1,t}(w_{t+1}) \leq \nu/t$ for any $t \geq 1$, $\nu \geq 0$, hence:

$$V_{t-1}(-s_t) \geq V_t(-s_t) + \psi_{l_1,t}(w_{t+1}) \Longrightarrow V_{t-1}(-s_t) + \nu/t \geq V_t(-s_t) + \psi_{l_1,t}(w_t).$$

□

Finally to derive our bound in Theorem 5 we note that Eq.(A.6) in view of Lemma 4 is given as follows:

$$V_\tau(-s_\tau) + \psi_{l_1,\tau}(w_\tau) \leq V_{\tau-1}(-s_{\tau-1}) + \nu/\tau + \langle -g_\tau, \hat{w}_\tau - w_0\rangle + \frac{\|g_\tau\|_*^2}{2\beta_{\tau-1}}. \tag{A.10}$$

Next we rearrange and sum up everything *w.r.t.* the index τ:

$$\sum_{\tau=1}^{t}(\langle g_\tau, \hat{w}_\tau - w_0\rangle + \psi_{l_1,\tau}(w_\tau)) + V_t(-s_t) \leq V_0(-s_0) + \nu \log t + \sum_{\tau=1}^{t} \frac{\|g_\tau\|_*^2}{2\beta_{\tau-1}}, \tag{A.11}$$

where we used $\sum_{\tau=1}^t \frac{1}{\tau} \leq \int_1^t \tau^{-1} d\tau = \log t$ to sum up ν/τ term. Next keeping in mind our initial bound on $U_t(s) \leq V_t(s) + \beta_t D$ and taking into account Eq.(A.5), $V_0(-s_0) = V_0(0) = 0$ and the first-order optimality conditions of the function f we get:

$$\mathbf{R}_t(w) \leq \delta_t \leq \nu(1 + \log t) + \beta_t D + \sum_{\tau=1}^{t} \frac{\|g_\tau\|_*^2}{2\beta_{\tau-1}}, \tag{A.12}$$

where one part of the right-hand side of inequality is exactly the same as in Eq.(A.8). So we can immediately substitute it with the right-hand side of Eq.(A.9) and get:

$$\mathbf{R}_t(w) \leq \nu \log + (\gamma D + \frac{G^2}{\gamma})\sqrt{t}. \tag{A.13}$$

□

A.2 Proofs for the Reweighted l_2-RDA method

A.2.1 Proof of Theorem 6

We stem our result from the proof of the Theorem 4 by redefining the conjugate-type functions $V_t(s)$ and $U_t(s)$ in [134] by replacing $\psi(w)$ in each of them with the sum of our reweighted l_2 functions $\psi_{l_2,t}(w) \triangleq \|\Theta_t^{1/2} w\|_2^2 + \lambda\|w\|_2^2$, such that:

$$U_t(s) = \max_{w\in\mathcal{F_D}} \{\langle s, w - w_0\rangle - \sum_{\tau=1}^{t} \psi_{l_2,\tau}(w)\},$$

$$V_t(s) = \max_{w}\{\langle s, w - w_0\rangle - \sum_{\tau=1}^{t} \psi_{l_2,\tau}(w) - \beta_t h(w)\}, \tag{A.14}$$

where we define the domain of $\psi_{l_2,\tau}(w)$ as $\mathcal{F}_D = \{w \in \mathrm{dom}\psi_{l_2} | h(w) \leq D\}$ (*i.e.* the domain of a simple non-reweighted l_2-norm as in [134]), $\{\beta_t\}_{t\geq 1}$ is a non-negative and non-decreasing sequence and $h(w)$ is a 1-strongly convex function (*i.e.* smoothing term).

Next we refer to the important Lemma 9 in [134], as for any $s \in E^*$ and $t \geq 0$ we have: $U_t(s) \leq V_t(s) + \beta_t D$, because using the definition of $\mathcal{F}_D$ we can bound $U_t(s)$ as follows:

$$\begin{aligned} U_t(s) &= \max_{w\in\mathcal{F_D}} \{\langle s, w - w_0\rangle - t\psi_{l_2,1}(w)\} \\ &= \max_{w} \min_{\beta\geq 0}\{\langle s, w - w_0\rangle - t\psi_{l_2,1}(w) + \beta(D - h(w))\} \\ &\leq \min_{\beta\geq 0} \max_{w}\{\langle s, w - w_0\rangle - t\psi_{l_2,1}(w) + \beta(D - h(w))\} \\ &\leq \max_{w}\{\langle s, w - w_0\rangle - t\psi_{l_2,1}(w) + \beta_t(D - h(w))\} \\ &= V_t(s) + \beta_t D. \end{aligned}$$

We will need this lemma to bound the regret as it will be shown below.

Lemma 5. *For each $t \geq 1$ in Algorithm 10 we have $V_t(-s_t) + \psi_{l_2,t}(w_t) \leq V_{t-1}(-s_t)$.*

Proof.

$$
\begin{aligned}
V_{t-1}(-s_t) &= \max_w \{\langle s_t, w - w_0\rangle - \sum_{\tau=1}^{t-1} \psi_{l_2,\tau}(w) - \beta_{t-1} h(w)\} \\
&\geq \langle s_t, w_{t+1} - w_0\rangle - \sum_{\tau=1}^{t-1} \psi_{l_2,\tau}(w_{t+1}) - \beta_{t-1} h(w_{t+1}) \\
&\geq \{\langle s_t, w_{t+1} - w_0\rangle - \sum_{\tau=1}^{t} \psi_{l_2,\tau}(w_{t+1}) - \beta_t h(w_{t+1})\} + \psi_{l_2,t}(w_{t+1}) \\
&= V_t(-s_t) + \psi_{l_2,t}(w_{t+1}) \geq V_t(-s_t) + \psi_{l_2,t}(w_t).
\end{aligned}
$$

□

In the above proof we used the same evidence as for Lemma 3 but with respect to $\psi_{l_2,1}(w)$. The final inequality follows from our assumption $\psi_{l_2,t}(w_t) \leq \psi_{l_2,t}(w_{t+1})$.

In the next part we will finally bound our regret function defined in Eq.(4.7) but with respect to $\psi_{l_2,t}(w)$. From [94] and [134] we know that if we consider $\Delta\psi_{l_2,\tau} = \psi_{l_2,\tau}(w_\tau) - \psi_{l_2,\tau}(w)$ the following *gap sequence* δ_t:

$$
\begin{aligned}
\delta_t &= \max_w \{\sum_{\tau=1}^{t} (\langle g_\tau, w_\tau - w\rangle \ + \Delta\psi_{l_2,\tau})\} \\
&\geq \sum_{\tau=1}^{t} (f(w_\tau, \xi_\tau) - f(w, \xi_\tau) + \Delta\psi_{l_2,\tau}) = \mathbf{R}_t(w) \qquad \text{(A.15)}
\end{aligned}
$$

bounds the regret function from above due to the convexity of f [22]. Next we can derive an upper bound on δ_t using the properties of the conjugate-type functions $V_t(s)$ and $U_t(s)$.

If we add and subtract $\sum_{\tau=1}^{t}\langle g_\tau, w_0\rangle$ term from the definition in Eq.(A.15) then for any $w \in \mathbb{R}^n$ we get:

$$
\begin{aligned}
\delta_t &= \sum_{\tau=1}^{t} (\langle g_\tau, w_\tau - w_0\rangle \ + \psi_{l_2,\tau}(w_\tau)) + \max_w \{\langle s_t, w_0 - w\rangle - \sum_{\tau=1}^{t} \psi_{l_2,\tau}(w)\} \\
&\leq \sum_{\tau=1}^{t} (\langle g_\tau, w_\tau - w_0\rangle \ + \psi_{l_2,\tau}(w_\tau)) + V_t(-s_t) + \beta_t D, \qquad \text{(A.16)}
\end{aligned}
$$

where $s_t = \sum_{\tau=1}^{t} g_\tau$. The last maximization term on the first line in Eq.(A.16) is exactly $U_t(-s_t)$, which was bounded earlier.

Using the well-known results on the boundedness of the conjugate support-type functions from [93, 94] for any $\tau \geq 1$ we have:

$$\begin{aligned} V_\tau(-s_\tau) + \psi_{l_2,\tau}(w_\tau) &\leq V_{\tau-1}(-s_\tau) \\ &= V_{\tau-1}(-s_{\tau-1} - g_\tau) \\ &\leq V_{\tau-1}(-s_{\tau-1}) + \langle -g_\tau, \nabla V_{\tau-1}(-s_{\tau-1})\rangle \\ &+ \frac{\|g_\tau\|_*^2}{2(\beta_{\tau-1} + \lambda(\tau-1))} \\ &= V_{\tau-1}(-s_{\tau-1}) + \langle -g_\tau, \hat{w}_\tau - w_0\rangle \\ &+ \frac{\|g_\tau\|_*^2}{2(\beta_{\tau-1} + \lambda(\tau-1))}, \end{aligned} \tag{A.17}$$

where $\|\cdot\|_*$ is a dual norm, the second inequality is due to the result of [93]: $V_\beta(s+g) \leq V_\beta(s) + \langle g, \nabla V_\beta(s)\rangle + \|g\|_*/(2\sigma\beta)$, $\forall s, g \in E^*$, where $\sigma\beta$ term refers to the smoothing part, which is quite different in case of $V_t(s)$ in Eq.(A.14). Based on the result in [93] and Lemma 1 in [94], the fact that $\sigma = 1$ in $h(w)$ and our particular choice of the smoothing strongly-convex term $\sum_{\tau=1}^{t} \psi_{l_2,\tau}(w) + \beta_t h(w)$ in Eq.(A.14) we have the Lipschitz continuous gradient of $V_t(s)$ with constant $\frac{1}{\beta_t + \lambda t}$, such that:

$$\|\nabla V_t(s_1) - \nabla V_t(s_2)\| \leq \frac{1}{\beta_t + \lambda t}\|s_1 - s_2\|_*, \quad \forall s_1, s_2 \in E^* \tag{A.18}$$

On the other hand we know a closed form for such a gradient in case of Eq.(A.17) and in view of conjugate support-type function $V_t(s)$ in Eq.(A.14):

$$\nabla V_{\tau-1}(-s_{\tau-1}) = \hat{w}_\tau - w_0, \ \forall \tau \geq 1$$

where $\hat{w}_\tau \triangleq \arg\min_w \{\langle s_{\tau-1}, w\rangle + \sum_{\kappa=1}^{\tau-1} \psi_{l_2,\kappa}(w) + \beta_{\tau-1} h(w)\}$ for $\forall \tau \geq 1$.

Next we rearrange and sum up everything *w.r.t.* index τ:

$$\begin{aligned} &\sum_{\tau=1}^{t}(\langle g_\tau, \hat{w}_\tau - w_0\rangle + \psi_{l_2,\tau}(w_\tau)) + V_t(-s_t) \\ &\qquad \leq V_0(-s_0) + \sum_{\tau=1}^{t} \frac{\|g_\tau\|_*^2}{2(\beta_{\tau-1} + \lambda(\tau-1))}. \end{aligned} \tag{A.19}$$

We can further simplify this equation by assuming $V_0(-s_0) = V_0(0) = 0$ and by using the first-order optimality conditions of the function f as follows:

$$0 \leq \langle g_\tau, \hat{w}_\tau - w_\tau \rangle \Longrightarrow \langle g_\tau, w_\tau \rangle \leq \langle g_\tau, \hat{w}_\tau \rangle.$$

Substituting all of above into Eq.(A.19) and taking into account Eq.(A.16) we get:

$$\mathbf{R}_t(w) \leq \delta_t \leq \beta_t D + \sum_{\tau=1}^{t} \frac{\|g_\tau\|_*^2}{2(\beta_{\tau-1} + \lambda(\tau - 1))}. \tag{A.20}$$

To obtain the bound in Theorem 6 we assume $\|g_t\|_* \leq G$ and set $\beta_{t\geq 1} = 0$, while keeping $\beta_0 = \lambda$:

$$\mathbf{R}_t(w) \leq \frac{\|g_1\|_*^2}{2\lambda} + \sum_{\tau=1}^{t-1} \frac{\|g_{\tau+1}\|_*^2}{2\lambda\tau} \leq \frac{G^2}{2\lambda}(1 + \int_1^t \tau^{-1} d\tau) = \frac{G^2}{2\lambda}(1 + \log t). \tag{A.21}$$

□

A.2.2 Proof of Theorem 7

The main part of the proof is exactly the same as for Theorem 6. We will refer to the parts of the proof of Theorem 6 as we will need them.

First we need to adjust Lemma 5 and introduce the dependency on the sufficient conditions of Theorem 6.

Lemma 6. *For each $t \geq 1$ in Algorithm 10 assuming that $\psi_{l_2,t}(w_t) \leq \psi_{l_2,t}(w_{t+1}) + \nu/t$ for some $\nu \geq 0$ we have $V_t(-s_t) + \psi_{l_2,t}(w_t) \leq V_{t-1}(-s_t) + \nu/t$.*

Proof.

$$\begin{aligned}
V_{t-1}(-s_t) &= \max_w \{\langle s_t, w - w_0 \rangle - \sum_{\tau=1}^{t-1} \psi_{l_2,\tau}(w) - \beta_{t-1} h(w)\} \\
&\geq \langle s_t, w_{t+1} - w_0 \rangle - \sum_{\tau=1}^{t-1} \psi_{l_2,\tau}(w_{t+1}) - \beta_{t-1} h(w_{t+1}) \\
&\geq \{\langle s_t, w_{t+1} - w_0 \rangle - \sum_{\tau=1}^{t} \psi_{l_2,\tau}(w_{t+1}) - \beta_t h(w_{t+1})\} + \psi_{l_2,t}(w_{t+1}) \\
&= V_t(-s_t) + \psi_{l_2,t}(w_{t+1}),
\end{aligned}$$

and to get the final result we use our assumption:

$$V_{t-1}(-s_t) \geq V_t(-s_t) + \psi_{l_2,t}(w_1) \Longrightarrow V_{t-1}(-s_t) + \nu/t \geq V_t(-s_t) + \psi_{l_2,t}(w_t).$$

□

In the above proof we used the same evidence as for Lemma 4 but with respect to $\psi_{l_2,1}(w)$.

Finally to derive our bound in Theorem 7 we note that Eq.(A.17) in view of Lemma 6 is given as follows:

$$\begin{aligned} V_\tau(-s_\tau) + \psi_{l_2,\tau}(w_\tau) &\leq \nu/\tau + V_{\tau-1}(-s_{\tau-1}) \\ &+\langle -g_\tau, \hat{w}_\tau - w_0\rangle + \frac{\|g_\tau\|_*^2}{2(\beta_{\tau-1} + \lambda(\tau-1))}. \end{aligned} \tag{A.22}$$

Next we rearrange and sum up everything *w.r.t.* index τ:

$$\begin{aligned} \sum_{\tau=1}^{t}(\langle g_\tau, \hat{w}_\tau - w_0\rangle + \psi_{l_2,\tau}(w_\tau)) + V_t(-s_t) &\leq \nu \log t \\ &+V_0(-s_0) + \sum_{\tau=1}^{t} \frac{\|g_\tau\|_*^2}{2(\beta_{\tau-1} + \lambda(\tau-1))}. \end{aligned} \tag{A.23}$$

The inequality above is almost the same as Eq.(A.19) except for an additional $\nu \log t$ term. Next keeping in mind our initial bound on $U_t(s) \leq V_t(s) + \beta_t D$ and taking into account Eq.(A.16), $V_0(-s_0) = V_0(0) = 0$ and the first-order optimality conditions of the function f we get:

$$\mathbf{R}_t(w) \leq \delta_t \leq \nu \log t + \beta_t D + \sum_{\tau=1}^{t} \frac{\|g_\tau\|_*^2}{2(\beta_{\tau-1} + \lambda(\tau-1))}, \tag{A.24}$$

where one part of the right-hand side of inequality is exactly the same as in Eq.(A.20), so that we can immediately substitute it with the right-hand side of Eq.(A.21) and get:

$$\mathbf{R}_t(w) \leq \frac{G^2}{2\lambda} + \frac{G^2 + 2\lambda\nu}{2\lambda} \log t. \tag{A.25}$$

□

Bibliography

[1] Alekh Agarwal, Olivier Chapelle, Miroslav Dudík, and John Langford. A reliable effective terascale linear learning system. *J. Mach. Learn. Res.*, 15(1):1111–1133, 2014.

[2] Divyakant Agrawal, Sudipto Das, and Amr El Abbadi. Big data and cloud computing: Current state and future opportunities. In *Proceedings of the 14th International Conference on Extending Database Technology*, EDBT/ICDT '11, pages 530–533, New York, NY, USA, 2011. ACM.

[3] Mauricio A. Álvarez, Lorenzo Rosasco, and Neil D. Lawrence. Kernels for vector-valued functions: A review. *Found. Trends Mach. Learn.*, 4(3):195–266, March 2012.

[4] Carlos Alzate and J. A. K. Suykens. A semi-supervised formulation to binary kernel spectral clustering. In *IJCNN*, pages 1–8, 2012.

[5] Carlos Alzate and Johan A. K. Suykens. Multiway spectral clustering with out-of-sample extensions through weighted kernel pca. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(2):335–347, 2010.

[6] E. Anderson, Z. Bai, J. Dongarra, A. Greenbaum, A. McKenney, J. Du Croz, S. Hammerling, J. Demmel, C. Bischof, and D. Sorensen. LAPACK: A portable linear algebra library for high-performance computers. In *Proceedings of the 1990 ACM/IEEE Conference on Supercomputing*, Supercomputing '90, pages 2–11, Los Alamitos, CA, USA, 1990. IEEE Computer Society Press.

[7] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404, 1950.

[8] David Arthur and Sergei 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, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[9] Francis Bach, Rodolphe Jenatton, and Julien Mairal. *Optimization with Sparsity-Inducing Penalties (Foundations and Trends in Machine Learning)*. Now Publishers Inc., Hanover, MA, USA, 2011.

[10] Francis R. Bach. Consistency of the Group Lasso and Multiple Kernel Learning. *J. Mach. Learn. Res.*, 9:1179–1225, June 2008.

[11] P Baldi, P Sadowski, and D Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5, 2014.

[12] Nahla H. Barakat and Andrew P. Bradley. Rule extraction from support vector machines: A review. *Neurocomputing*, 74(1-3):178–190, 2010.

[13] Mikhail Belkin and Partha Niyogi. Towards a theoretical foundation for laplacian-based manifold methods. *J. Comput. Syst. Sci.*, 74(8):1289–1308, 2008.

[14] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

[15] Yoshua Bengio and Xavier Glorot. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of AISTATS 2010*, volume 9, pages 249–256, May 2010.

[16] James C. Bezdek and Nikhil R. Pal. Some new indexes of cluster validity. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 28(3):301–315, 1998.

[17] L. S. Blackford, J. Demmel, J. Dongarra, I. Duff, S. Hammarling, G. Henry, M. Heroux, L. Kaufman, A. Lumsdaine, A. Petitet, R. Pozo, K. Remington, and R. C. Whaley. An updated set of basic linear algebra subprograms (BLAS). *ACM Transactions on Mathematical Software*, 28:135–151, 2001.

[18] I. Borg and P.J.F. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer, 2005.

[19] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, COLT '92, pages 144–152, New York, NY, USA, 1992. ACM.

[20] Léon Bottou. Online algorithms and stochastic approximations. In David Saad, editor, *Online Learning and Neural Networks*. Cambridge University Press, Cambridge, UK, 1998. revised, oct 2012.

[21] Léon Bottou. Large-Scale Machine Learning with Stochastic Gradient Descent. In Yves Lechevallier and Gilbert Saporta, editors, *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT 2010)*, pages 177–187, Paris, France, August 2010. Springer.

[22] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[23] C. J. C. Burges. Simplified support vector decision rules. In L. Saitta, editor, *Proceedings of the 13th International Conference on Machine Learning*, pages 71–77. Morgan Kaufmann, San Mateo, CA, 1996.

[24] Colin Campbell and Kristin P. Bennett. A Linear Programming Approach to Novelty Detection. In *Advances in Neural Information Processing Systems 13*, pages 395–401. MIT Press, 2001.

[25] E.J. Candes and T. Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *Information Theory, IEEE Transactions on*, 52(12):5406–5425, 2006.

[26] Emmanuel Candès, Michael Wakin, and Stephen Boyd. Enhancing sparsity by reweighted l_1 minimization. *Journal of Fourier Analysis and Applications*, 14(5):877–905, 2008.

[27] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/cjlin/libsvm.

[28] O. Chapelle and A. Zien. Semi-supervised classification by low density separation. In *Proceedings of the International Workshop on Artificial Intelligence and Statistics*, 2005.

[29] Olivier Chapelle. Training a support vector machine in the primal. *Neural Computation*, 19:1155–1178, 2007.

[30] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. *Semi-Supervised Learning*. MIT Press, 2006.

[31] R. Chartrand and Wotao Yin. Iteratively reweighted algorithms for compressive sensing. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2008*, pages 3869–3872, March 2008.

[32] Xi Chen, Qihang Lin, and Javier Peña. Optimal regularized dual averaging methods for stochastic optimization. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, editors, *NIPS*, pages 404–412, 2012.

[33] Andreas Christmann and Ingo Steinwart. How SVMs can estimate quantiles and the median. In *NIPS*, pages 305–312, 2007.

[34] Cheng-tao Chu, Sang K. Kim, Yi-an Lin, Yuanyuan Yu, Gary Bradski, Kunle Olukotun, and Andrew Y. Ng. Map-Reduce for machine learning on multicore. In B. Schölkopf, J.C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 281–288. MIT Press, 2007.

[35] Wei Chu, S. Sathiya Keerthi, and Chong Jin Ong. Bayesian trigonometric support vector classifier. *Neural Comput.*, 15(9):2227–2254, September 2003.

[36] F. R. K. Chung. *Spectral Graph Theory.* American Mathematical Society, 1997.

[37] Ingrid Daubechies, Ronald DeVore, Massimo Fornasier, and C. Sinan Güntürk. Iteratively reweighted least squares minimization for sparse recovery. *Comm. Pure Appl. Math.*, 63(1):1–38, January 2010.

[38] H. A. David and Jason L. Gunnink. The paired t test under artificial pairing. 51(1):9–12, February 1997.

[39] K. De Brabanter, J. De Brabanter, J. A. K. Suykens, and B. De Moor. Optimized fixed-size kernel models for large data sets. *Comput. Stat. Data Anal.*, 54(6):1484–1504, June 2010.

[40] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, January 2008.

[41] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[42] M. J. Desforges, P. J. Jacob, and J. E. Cooper. Applications of probability density estimation to the detection of abnormal conditions in engineering. In *Proceedings of Institute of Mechanical Engineers*, volume 212, pages 687–703, 1998.

[43] David L. Donoho. Compressed sensing. *IEEE Trans. Inform. Theory*, 52:1289–1306, 2006.

[44] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July 2011.

[45] John Duchi and Yoram Singer. Efficient online and batch learning using forward backward splitting. *J. Mach. Learn. Res.*, 10:2899–2934, December 2009.

[46] Adil Fahad, Najlaa Alshatri, Zahir Tari, Abdullah Alamri, Ibrahim Khalil, Albert Y. Zomaya, Sebti Foufou, and Abdelaziz Bouras. A survey of clustering algorithms for big data: Taxonomy and empirical analysis. *IEEE Trans. Emerging Topics Comput.*, 2(3):267–279, 2014.

[47] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, 2008.

[48] Simon Foucart and Holger Rauhut. *A Mathematical Introduction to Compressive Sensing.* Birkhäuser Basel, 2013.

[49] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[50] Pasi Fränti and Olli Virmajoki. Iterative shrinking method for clustering problems. *Pattern Recogn.*, 39(5):761–775, May 2006.

[51] Ya Gao and Shiliang Sun. An empirical evaluation of linear and nonlinear kernels for text classification using support vector machines. In *FSKD*, pages 1502–1505, 2010.

[52] Attila Gürsoy. Data decomposition for parallel k-means clustering. In Roman Wyrzykowski, Jack Dongarra, Marcin Paprzycki, and Jerzy Wasniewski, editors, *PPAM*, volume 3019 of *Lecture Notes in Computer Science*, pages 241–248. Springer, 2003.

[53] T. Hastie, R. Tibshirani, and M. Wainwright. *Statistical Learning with Sparsity: The Lasso and Generalizations.* Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 2015.

[54] Elad Hazan, Amit Agarwal, and Satyen Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2-3):169–192, December 2007.

[55] G. Holmes, A. Donkin, and I. H. Witten. WEKA: a machine learning workbench. In *Proc. of the Second Australian and New Zealand Conference on the Intelligent Information Systems*, pages 357–361, August 1994.

[56] Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S. Sathiya 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, pages 408–415, New York, NY, USA, 2008. ACM.

[57] Kaizhu Huang, Irwin King, and Michael R. Lyu. Direct zero-norm optimization for feature selection. In *ICDM*, pages 845–850, 2008.

[58] Xiaolin Huang, Lei Shi, and Johan A. K. Suykens. Support vector machine classifier with pinball loss. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(5):984–997, 2014.

[59] Patrick C. Hytla, Russell C. Hardie, Michael T. Eismann, and Joseph Meola. Anomaly detection in hyperspectral imagery: comparison of methods using diurnal and seasonal data. *Journal of Applied Remote Sensing*, 3(1):033546–033546–30, 2009.

[60] Laurent Jacob, Guillaume Obozinski, and Jean-Philippe Vert. Group lasso with overlap and graph lasso. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 433–440, New York, NY, USA, 2009. ACM.

[61] Jayadeva, R. Khemchandani, and Suresh Chandra. Twin support vector machines for pattern classification. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(5):905–910, May 2007.

[62] Thorsten Joachims. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 217–226, New York, NY, USA, 2006. ACM.

[63] Vilen Jumutc, Xiaolin Huang, and Johan A. K. Suykens. Fixed-size pegasos for hinge and pinball loss svm. In *IJCNN*, pages 1–7. IEEE, 2013.

[64] Vilen Jumutc and Johan A. K. Suykens. Supervised novelty detection. In *CIDM*, pages 143–149. IEEE, 2013.

[65] Vilen Jumutc and Johan A. K. Suykens. Multi-class supervised novelty detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(12):2510–2523, 2014.

[66] Vilen Jumutc and Johan A. K. Suykens. New bilinear formulation to semi-supervised classification based on kernel spectral clustering. In *CIDM*, pages 41–47. IEEE, 2014.

[67] Vilen Jumutc and Johan A. K. Suykens. Reweighted l_1 dual averaging approach for sparse stochastic learning. In *ESANN*, 2014.

[68] Vilen Jumutc and Johan A. K. Suykens. Reweighted l_2-regularized dual averaging approach for highly sparse stochastic learning. In *Advances in Neural Networks - ISNN 2014 - 11th International Symposium on Neural Networks, ISNN 2014, Hong Kong and Macao, China, November 28- December 1, 2014. Proceedings*, pages 232–242, 2014.

[69] Vilen Jumutc and Johan A. K. Suykens. Regularized and sparse stochastic k-means for distributed large-scale clustering. Technical Report 15-126, Department of Electrical Engineering (ESAT-STADIUS), KU Leuven, June 2015.

[70] Vilen Jumutc and Johan A. K. Suykens. Reweighted stochastic learning. *Neurocomputing (In Press)*, 2015.

[71] Roger Koenker. *Quantile Regression.* Econometric Society Monographs. Cambridge University Press, 2005.

[72] M. Arun Kumar and Madan Gopal. Least squares twin support vector machines for pattern classification. *Expert Syst. Appl.*, 36(4):7535–7543, 2009.

[73] Balázs Kövesi, Jean-Marc Boucher, and Samir Saoudi. Stochastic k-means algorithm for vector quantization. *Pattern Recognition Letters*, 22(6/7):603–610, 2001.

[74] Ming-Jun Lai and Yang Liu. The null space property for sparse recovery from multiple measurement vectors. *Applied and Computational Harmonic Analysis*, 30(3):402–406, 2011.

[75] Ming-Jun Lai, Yangyang Xu, and Wotao Yin. Improved iteratively reweighted least squares for unconstrained smoothed l_q minimization. *SIAM J. Numerical Analysis*, 51(2):927–957, 2013.

[76] John Langford, Lihong Li, and Alex Strehl. Vowpal Wabbit, 2007.

[77] Hugo Larochelle, Yoshua Bengio, Jérôme Louradour, and Pascal Lamblin. Exploring strategies for training deep neural networks. *J. Mach. Learn. Res.*, 10:1–40, June 2009.

[78] Jorge López Lázaro, Kris De Brabanter, José R. Dorronsoro, and Johan A. K. Suykens. Sparse LS-SVMs with l_0-norm minimization. In *ESANN*, pages 189–194, 2011.

[79] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, 5:361–397, December 2004.

[80] Juntao Li, Yingmin Jia, Junping Du, and Fashan Yu. A new support vector machine for microarray classification and adaptive gene selection. In *Proceedings of the 2009 conference on American Control Conference*, ACC'09, pages 5410–5415, Piscataway, NJ, USA, 2009. IEEE Press.

[81] David J.C. MacKay. Bayesian interpolation. *Neural Computation*, 4:415–447, 1991.

[82] David J.C. MacKay. Comparison of approximate methods for handling hyperparameters. *Neural Computation*, 11:1035–1068, 1999.

[83] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[84] Raghvendra Mall and Johan A. K. Suykens. Sparse reductions for fixed-size least squares support vector machines on large scale data. In *PAKDD (1)*, pages 161–173, 2013.

[85] S. Mehrkanoon, C. Alzate, R. Mall, R. Langone, and J. A. K Suykens. Multi-class semi-supervised learning based upon kernel spectral clustering. *IEEE Transactions on Neural Networks and Learning Systems, in press*, 2014.

[86] S. Mehrkanoon and J. A. K Suykens. Large scale semi-supervised learning using KSC based model. In *International Joint Conference on Neural Networks (IJCNN 2014), accepted*, 2014.

[87] Siamak Mehrkanoon and J. A. K Suykens. Non-parallel semi-supervised classification based on kernel spectral clustering. In *IJCNN*, pages 1–8. IEEE, 2013.

[88] Stefano Melacci and Mikhail Belkin. Laplacian support vector machines trained in the primal. *Journal of Machine Learning Research*, 12:1149–1184, 2011.

[89] Xiangrui Meng, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, D. B. Tsai, Manish Amde, Sean Owen, Doris Xin, Reynold Xin, Michael J. Franklin, Reza Zadeh, Matei Zaharia, and Ameet Talwalkar. MLlib: Machine Learning in Apache Spark, May 2015.

[90] Radford M. Neal. *Bayesian Learning for Neural Networks.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1996.

[91] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[92] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM J. on Optimization*, 19(4):1574–1609, January 2009.

[93] Yurii. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, 2005.

[94] Yurii Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical Programming*, 120(1):221–259, 2009.

[95] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems*, 14:849–856, 2001.

[96] Haydemar Núnez, Cecilio Angulo, and Andreu Català. Rule extraction from support vector machines. In *In Proceedings of European Symposium on Artificial Neural Networks*, pages 107–112, 2002.

[97] Natascha Oppelt and Wolfram Mauser. The Airborne Visible / Infrared Imaging Spectrometer AVIS: Design, Characterization and Calibration. *Sensors*, 7(9):1934–1953, 2007.

[98] Sean Owen, Robin Anil, Ted Dunning, and Ellen Friedman. *Mahout in Action.* Manning Publications Co., Greenwich, CT, USA, 2011.

[99] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[100] J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, pages 42–65. MIT Press, Cambridge, MA, 1998.

[101] Lorenzo Rosasco, Ernesto De Vito, Andrea Caponnetto, Michele Piana, and Alessandro Verri. Are loss functions all the same? *Neural Comput.*, 16(5):1063–1076, May 2004.

[102] Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors. *Advances in kernel methods: support vector learning.* MIT Press, Cambridge, MA, USA, 1999.

[103] Bernhard Schölkopf, John C. Platt, John C. Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Comput.*, 13(7):1443–1471, July 2001.

[104] Bernhard Schölkopf, Alex J. Smola, Robert C. Williamson, and Peter L. Bartlett. New support vector algorithms. *Neural Comput.*, 12(5):1207–1245, May 2000.

[105] Matthias Seeger. Learning with labeled and unlabeled data. Technical report, 2001.

[106] Shai Shalev-Shwartz and Yoram Singer. Logarithmic regret algorithms for strongly convex repeated games. Technical report, The Hebrew University, 2007.

[107] Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 807–814, New York, NY, USA, 2007.

[108] Shai Shalev-Shwartz and Ambuj Tewari. Stochastic methods for l_1 regularized loss minimization. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 929–936, New York, NY, USA, 2009. ACM.

[109] Yuan-Hai Shao, Lan Bai, Zhen Wang, Xiang-Yu Hua, and Nai-Yang Deng. Proximal plane clustering via eigenvalues. In *ITQM*, volume 17 of *Procedia Computer Science*, pages 41–47. Elsevier, 2013.

[110] Yuan-Hai Shao, Chun-Hua Zhang, Xiao-Bo Wang, and Nai-Yang Deng. Improvements on twin support vector machines. *IEEE Transactions on Neural Networks*, 22(6):962–968, 2011.

[111] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, 2000.

[112] Sören Sonnenburg, Gunnar Rätsch, Sebastian Henschel, Christian Widmer, Jonas Behr, Alexander Zien, De Fabio Bona, Alexander Binder, Christian Gehl, and Vojtech Franc. The SHOGUN machine learning toolbox. *J. Mach. Learn. Res.*, 11:1799–1802, August 2010.

[113] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[114] Ingo Steinwart. On the influence of the kernel on the consistency of support vector machines. *Journal of Machine Learning Research*, 2:67–93, 2001.

[115] Ingo Steinwart and Andreas Christmann. Estimating conditional quantiles with the help of the pinball loss. *Bernoulli*, 17(1):211–225, 2011.

[116] Ingo Steinwart, Don Hush, and Clint Scovel. A classification framework for anomaly detection. *J. Machine Learning Research*, 6:211–232, 2005.

[117] Ingo Steinwart, James Theiler, and Daniel Llamocca. Using support vector machines for anomalous change detection. In *IGARSS*, pages 3732–3735, 2010.

[118] Wei Sun and Junhui Wang. Regularized k-means clustering of high-dimensional data and its asymptotic consistency. *Electronic Journal of Statistics*, 6:148–167, 2012.

[119] J. A. K. Suykens, J. De Brabanter, L. Lukas, and J. Vandewalle. Weighted least squares support vector machines: robustness and sparse approximation. *Neurocomputing*, 48(1):85–105, October 2002.

[120] J. A. K. Suykens, T Van Gestel, J De Brabanter, B De Moor, and Joos Vandewalle. *Least Squares Support Vector Machines.* World Scientific, Singapore, 2002.

[121] J. A. K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Process. Lett.*, 9(3):293–300, June 1999.

[122] David M. J. Tax and Robert P. W. Duin. Support vector data description. *Mach. Learn.*, 54(1):45–66, January 2004.

[123] Michael E. Tipping. Sparse bayesian learning and the relevance vector machine. *J. Mach. Learn. Res.*, 1:211–244, September 2001.

[124] Vladimir Vapnik. Principles of risk minimization for learning theory. In *Advances in Neural Information Processing Systems 4, NIPS Conference, Denver, Colorado, USA, December 2-5, 1991]*, pages 831–838, 1991.

[125] Vladimir Vapnik. *Statistical learning theory.* Wiley, 1 edition, September 1998.

[126] Andrea Vedaldi and Andrew Zisserman. Efficient additive kernels via explicit feature maps. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(3):480–492, 2012.

[127] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program.*, 106(1):25–57, May 2006.

[128] Tom White. *Hadoop: The Definitive Guide.* O'Reilly Media, Inc., 1st edition, 2009.

[129] Christopher K. I. Williams and Matthias W. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13, (NIPS) 2000, Denver, CO, USA*, pages 682–688, 2000.

[130] David Wipf and Srikantan Nagarajan. A New View of Automatic Relevance Determination. In J. C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1625–1632. MIT Press, Cambridge, MA, 2008.

[131] David P. Wipf and Srikantan S. Nagarajan. Iterative reweighted l_1 and l_2 methods for finding sparse solutions. *J. Sel. Topics Signal Processing*, 4(2):317–329, 2010.

[132] Daniela M. Witten and Robert Tibshirani. A framework for feature selection in clustering. 105(490):713–726, June 2010.

[133] Samuel Xavier-De-Souza, Johan A. K. Suykens, Joos Vandewalle, and Désiré Bollé. Coupled simulated annealing. *IEEE Trans. Sys. Man Cyber. Part B*, 40(2):320–335, April 2010.

[134] Lin Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *J. Mach. Learn. Res.*, 11:2543–2596, December 2010.

[135] Linli Xu, Koby Crammer, and Dale Schuurmans. Robust support vector machine training via convex outlier ablation. In *AAAI*, pages 536–542, 2006.

[136] Linli Xu, James Neufeld, Bryce Larson, and Dale Schuurmans. Maximum margin clustering. In *Advances in Neural Information Processing Systems 17*, pages 1537–1544. MIT Press, 2005.

[137] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, pages 10–10, Berkeley, CA, USA, 2010. USENIX Association.

[138] Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

Curriculum vitae

Vilen Jumutc was born in Daugavpils, Latvia. He received his B.Sc. and M.Sc. degrees in computer science from the Riga Technical University in 2007 and 2009 respectively. During his Masters, he worked on Support Vector Machines and Artificial Neural Networks. Together with colleagues at the Biomedical Research and Study Center in Riga he devised several kernel-based methods for some specific problems in cancer diagnostics based on protein micro-arrays.

While finishing his studies at Riga Technical University he was working as a software engineer in the outsourcing company embedding and implementing for customers the latest and bleeding-edge technologies in Computer Science.

He started his doctoral studies at the Department of Electrical Engineering (ESAT-STADIUS) of KU Leuven in April 2012 under the supervision of professor Johan A.K. Suykens. His interests included large-scale optimization problems, kernel methods, semi-supervised learning and convex optimization. During his studies he published two first author peer-reviewed journal papers, seven first author international conference papers and co-authored several others.

He has accomplished a number of additional duties as a professional researcher and doctoral student, including teaching, supervision of master students, reviewing for several journals, participation in conferences and contribution to the development of scientific open source software. He was a co-organizer of TCMM 2014[2] which took place in Leuven.

[2] International Workshop on Technical Computing for Machine Learning and Mathematical Engineering

List of publications

International Journal Papers:

- **Jumutc V.**, Suykens J.A.K., "Multi-Class Supervised Novelty Detection", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 36, no. 12, Dec. 2014, pp. 2510 - 2523.
- **Jumutc V.**, J.A.K. Suykens, "Reweighted Stochastic Learning", Neurocomputing Special Issue - ISNN2014, 2015. (In Press)

International Conference Papers:

- **Jumutc V.**, Langone R., Suykens J.A.K., "Regularized and Sparse Stochastic K-Means for Distributed Large-Scale Clustering", in Proc. of the 2015 IEEE International Conference on Big Data (IEEE BigData 2015), Santa Clara, USA, Oct 29 – Nov 1, 2015.
- Mall R., **Jumutc V.**, Langone R., Suykens J.A.K., "Representative Subsets For Big Data Learning using kNN graphs", in Proc. of the IEEE BigData (BigData), Washington DC, U.S.A., Oct. 2014, pp. 37-42.
- **Jumutc V.**, Suykens J.A.K., "New Bilinear Formulation to Semi-Supervised Classification Based on Kernel Spectral Clustering", in Proc. of the 2014 IEEE Symposium Series on Computational Intelligence (IEEE SSCI 2014), Orlando, Florida, Dec. 2014, pp. 41-47.
- **Jumutc V.**, Suykens J.A.K., "Reweighted l2-Regularized Dual Averaging Approach for Highly Sparse Stochastic Learning", in Proc. of the 11th

International Symposium on Neural Networks (ISNN 2014), Hong-Kong & Makao, People's Republic of China , Nov. 2014, pp. 232-242.

- **Jumutc V.**, Suykens J.A.K., "Reweighted l1 Dual Averaging Approach for Sparse Stochastic Learning", in Proc. of the 22th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2014), Bruges, Belgium, Apr. 2014, pp. 1 - 6.
- **Jumutc V.**, Suykens J., "Weighted Coordinate-Wise Pegasos", in Proc. of the 5th International Conference on Pattern Recognition and Machine Intelligence (PREMI 2013), Kolkata, India, Dec. 2013, pp. 262-269.
- **Jumutc V.**, Huang X., Suykens J.A.K., "Fixed-Size Pegasos for Hinge and Pinball Loss SVM", in Proc. of the 2013 International Joint Conference on Neural Networks (IJCNN 2013), Dallas, USA, Aug. 2013, pp. 1122-1128.
- **Jumutc V.**, Suykens J.A.K., "Supervised Novelty Detection", in Proc. of the IEEE Symposium Series on Computational Intelligence (SSCI 2013), Singapore, Singapore, Apr. 2013, pp. 143 - 149.

Forthcoming Papers & Submissions:

- **Jumutc V.**, Suykens J.A.K., "SALSA: Software Lab for Advanced Machine Learning with Stochastic Algorithms", Internal Report 15-179, ESAT-STADIUS, KU Leuven (Leuven, Belgium), 2015, submitted for publication.
- **Jumutc V.**, Langone R., Suykens J.A.K., "Regularized and Distributed Stochastic K-Means in the Presence of Outliers", submitted for publication.

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF ELECTRICAL ENGINEERING
STADIUS
Kasteelpark Arenberg 10, box 2446
B-3001 Leuven
vilen.jumutc@esat.kuleuven.be
http://homes.esat.kuleuven.be/ vjumutc