August 2020

Fast Machine Learning Algorithms for Massive Datasets with Applications in the Biomedical Domain

Ehsan Sadrfaridpour
Clemson University, esadrfa@g.clemson.edu

Fast Machine Learning Algorithms for Massive Datasets with Applications in the Biomedical Domain

---

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Biomedical Data Science and Informatics

---

by
Ehsan Sadrfaridpour
August 2020

Accepted by:
Dr. Ilya Safro, Committee Chair
Dr. Amy Apon
Dr. Alexander Herzog
Dr. Brian Dean
Dr. Lewis Frey

Abstract

The continuous increase in the size of datasets introduces computational challenges for machine learning algorithms. In this dissertation, we cover the machine learning algorithms and applications in large-scale data analysis in manufacturing and healthcare. We begin with introducing a multilevel framework to scale the support vector machine (SVM), a popular supervised learning algorithm with a few tunable hyperparameters and highly accurate prediction. The computational complexity of nonlinear SVM is prohibitive on large-scale datasets compared to the linear SVM, which is more scalable for massive datasets. The nonlinear SVM has shown to produce significantly higher classification quality on complex and highly imbalanced datasets. However, a higher classification quality requires a computationally expensive quadratic programming solver and extra kernel parameters for model selection. We introduce a generalized fast multilevel framework for regular, weighted, and instance weighted SVM that achieves similar or better classification quality compared to the state-of-the-art SVM libraries such as LIBSVM. Our framework improves the runtime more than two orders of magnitude for some of the well-known benchmark datasets. We cover multiple versions of our proposed framework and its implementation in detail. The framework is implemented using PETSc library which allows easy integration with scientific computing tasks. Next, we propose an adaptive multilevel learning framework for SVM to reduce the variance between prediction qualities across the levels, improve the overall prediction accuracy, and boost the runtime. We implement multi-threaded support to speed up the parameter fitting runtime that results in more than an order of magnitude speed-up. We design an early stopping criteria to reduce the extra computational cost when we achieve expected prediction quality. This approach provides significant speed-up, especially for massive datasets. Finally, we propose an efficient low dimensional feature extraction over massive knowledge networks. Knowledge networks are becoming more popular in the biomedical domain for knowledge representation. Each layer in knowledge networks can store the information from one or

multiple sources of data. The relationships between concepts or between layers represent valuable information. The proposed feature engineering approach provides an efficient and highly accurate prediction of the relationship between biomedical concepts on massive datasets. Our proposed approach utilizes semantics and probabilities to reduce the potential search space for the exploration and learning of machine learning algorithms. The calculation of probabilities is highly scalable with the size of the knowledge network. The number of features is fixed and equivalent to the number of relationships or classes in the data. A comprehensive comparison of well-known classifiers such as random forest, SVM, and deep learning over various features extracted from the same dataset, provides an overview for performance and computational trade-offs. Our source code, documentation and parameters will be available at `https://github.com/esadr/`.

Dedication

This dissertation is dedicated to my father, mother, Amirhosein and Behzad for their unconditional love and support.

I wish to express my gratitude to my advisor, Dr. Ilya Safro, who believed in me and supported me academically and financially to reach this point as a Ph.D. student. I would also like to thank my other committee members, Dr. Amy Apon, Dr. Alexander Herzog, Dr. Brian Dean, and Dr. Lewis Frey for providing valuable insights and guidance.

I am forever grateful to my loving parents Hamid and Zahra, who raised me and encouraged me to follow my dreams. My wonderful and supportive brothers Amirhosein and Behzad, have always inspired me through love, encouragement, and support, and I will be thankful to you.

I would like to thank Dr. Sekou Remy, who helped me at the beginning of my Ph.D. journey at Clemson University. I am thankful to my co-author, Dr. Talayeh Razzaghi, for great collaboration and incredible learning experience.

I want to thank the School of Computing staff, especially Ms. Christy Babb, Mr. Adam Rolling, Ms. Kaley Goodwin, Mr. Chuck Cook, and Mr. Sasha Kuksenok, for their unfailing support and assistance. I am thankful to research support staff for Palmetto Cluster for their support.

I would like to thank my friends and colleagues during my time in graduate school, Emmanuel John, Hayato Ushijima-Mwesigwa, Justin Sybrandt, Ruslan Shaydulin, Varsh Chauhan, Angelo Carrabba, Joey Liu, Korey Palmer, Zirou Qiu, and Ilya Tyagin.

This dissertation was supported in part by Biomedical Data Science and Informatics program at Clemson University and the National Science Foundation under Grants Nos. 1638321, and 1522751.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The advances in technologies empower us to collect more high-quality data from new sources such as wearable devices, sequencers, digitized systems, and many more. As the size of data grows, more efficient and robust machine learning and data analysis pipelines are required to maximize the utilization of this new data. Extracting meaningful information and patterns in raw data is valuable, but it is getting more complicated for big data. Big data has many challenges, such as volume, variety, and velocity, that require more efficient machine learning algorithms to improve the quality and reduce the computational cost and runtime. The imbalanced datasets include rare events that cause challenges for machine learning algorithms performance. Health-care, finance, and cybersecurity are some of the fields with large many imbalanced datasets. For instance, the number of samples (instances) of infected people or malicious network packets (cyber-attacks) is drastically less than the number of healthy people or regular network packets. Most of the time, researchers are not concerned with detecting healthy people or regular network packets. The rare events are the most important phenomena which need to be identified correctly and rapidly in a massive dataset. The development of efficient machine learning algorithms for such datasets is one of the main topics in this dissertation. In chapter 2, we cover our design and development of an efficient and robust multilevel framework for the classification of massive datasets with extremely imbalanced data. Furthermore, in chapter 3, we demonstrate an improvement in the performance and quality of the multilevel predictive framework.

Developing a knowledge network by linking information from multiple related sources can lead to better knowledge discovery. However, a manifold of data, huge volume, and heterogeneity of

nodes and relationships introduce modeling and computational challenges. In chapter 4, we propose a novel feature extraction algorithm along with a data pipeline for relationship type prediction. We applied our method for predicting the relationship types between concepts in biomedical literature. We developed multiple feature extraction and compared various classification algorithms. The probabilistic feature extraction proposed in this work has the potential to reduce similar large scale data science projects in many directions. The most important outcome is to significantly improve the quality of machine learning algorithms and reduce the computational cost of dimensionality reduction techniques.

Biomedical literature is an invaluable source of data; researchers have access to millions of published papers to learn and follow the latest innovations in their field. However, the volume of publications is overwhelming, and it is hard to keep track of all relevant papers. For new researchers in a field or researchers who start to work on a distant area of their earlier works, it is a tedious and challenging task to link all relevant concepts and their relationships. Moreover, some of the disciplines are not well connected, and they may not bridge their knowledge of distant concepts. The information retrieval systems only capable of providing results based on specific keyword searches, often return thousands of papers for a topic. The manual process of going through all the papers and extracting useful information is tedious. Literature based discovery or hypothesis generation systems provide fruitful hypotheses by finding the implicit connections in a large corpus of biomedical literature. Generated hypotheses can lead research in new directions with more promising results. The literature based discovery systems first introduced in 1986 by Smalheiser and Swanson [106] to bridge the concepts from distant domains. Similar systems have been developed and improved over the last two decades, but there is still a need for more functionality in the existing systems [34, 40]. Our proposed feature extraction method significantly improved the prediction quality for the relationship type prediction between biomedical concepts.

Chapter 2

Engineering fast multilevel support vector machines

2.1 Introduction

Support vector machine (SVM) is one of the most well-known supervised classification methods that has been extensively used in such fields as disease diagnosis, text categorization, and fraud detection. Training nonlinear SVM classifier (such as Gaussian kernel based) requires solving convex quadratic programming (QP) model whose running time can be prohibitive for large-scale instances without using specialized acceleration techniques such as sampling, boosting, and hierarchical training. Another typical reason of increased running time is complex data sets (e.g., when the data is noisy, imbalanced, or incomplete) that require using model selection techniques for finding the best model parameters.

The motivation behind this work was extensive applied experience with hard, large-scale, industrial (often highly heterogeneous) data sets for which fast *linear* SVMs produced extremely low quality results (as well as many other fast methods), and various nonlinear SVMs exhibited a strong trade off between running time and quality. It has been noticed in multiple works that many different real-world data sets have a strong underlying multiscale (in some works called hierarchical) structure [56, 49, 57, 98] that can be discovered through careful definitions of coarse-grained resolutions. Not surprisingly, we found that among fast methods the hierarchical nonlinear SVM was the best

candidate for producing most satisfying results in a reasonable time [3]. Although, several successful hierarchical SVM techniques [127, 39] have been developed since massive popularization of SVM, we found that most existing algorithms do not sustainably produce high-quality results in a short running time, and the behavior of hierarchical training is still not well studied. This is in contrast to a variety of well studied unsupervised multiscale clustering approaches [10, 72, 84].

In this chapter, we discuss several techniques for engineering multilevel SVMs demonstrating their (dis)advantages and generalizing them in a framework inspired by the algebraic multigrid and multiscale optimization strategies [11]. We deliberately omit the issues related to parallelization of multilevel frameworks as it has been discussed in a variety of works related to multilevel clustering, partitioning, and SVM QP solvers. Our goal is to demonstrate fast and scalable sequential techniques focusing on different aspects of building and using multilevel learning with regular and weighted SVM. Also, we focus only on nonlinear SVMs because (a) not much improvement can be introduced or required in practice to accelerate linear SVMs, and (b) in many hard practical cases, the quality of linear SVMs is incomparable to that of nonlinear SVMs. The most promising and stable version of our multilevel SVMs are implemented in PETSc [4] which is a well known scientific computing library. PETSc was selected because of its scalability of linear algebra computations on large sparse matrices and available software infrastructure for future parallelization. Our implementation also addresses a critical need [8] of adding data analysis functionality to broadly used scientific computing software.

2.1.1 Computational challenges

There is a number of basic challenges one has to address when applying SVM which we successfully tackle with the multilevel framework, namely, QP solver complexity for large-scale data, imbalanced data, and SVM model parameter fitting.

Large-scale data The baseline SVM classifier is typically formulated as a convex QP problem whose solvers scale between $\mathcal{O}(n^2)$ to $\mathcal{O}(n^3)$ [36]. For example, the solver we compare our algorithm with, namely, LibSVM [17], which is one of the most popular and fast QP solvers, scales between $\mathcal{O}(n_f {n_s}^2)$ to $\mathcal{O}(n_f {n_s}^3)$ subject to how effectively the cache is exploited in practice, where n_f and n_s are the number of features and samples, respectively. Clearly, this complexity is prohibitive for nonlinear SVM models applied on practical big data without using parallelization, high-performance

computing systems or another special treatment.

Imbalanced data The imbalanced data is one of the issues in which SVM often outperforms many fast machine learning methods. This problem occurs when the number of instances of one class (negative or majority class) is substantially larger than the number of instances that belong to the other class (positive or minority class). In multi-class classification, the problem of imbalanced data is even bolder and use of the standard classification methods become problematic in the presence of big and imbalanced data [65]. This may dramatically deteriorate the performance of the algorithm. It is worth noticing that there are cases in which correct classification of the smaller class is more important than misclassification of the larger class [105]. Fault diagnosis [124, 132], anomaly detection [53, 112], medical diagnosis [68, 6, 7] are some of applications which are known to suffer of this problem. Imbalanced data was one of our motivating factors because we have noticed that most standard SVM solvers do not behave well on it. In order to reduce the effect of minority class misclassification in highly imbalanced data, an extension of SVM, namely, the cost-sensitive SVM (whose extensions are also known as weighted or fuzzy SVM) [62], was developed for imbalanced classification problems. In cost-sensitive SVM, a special control of misclassification penalization is introduced as a part of the SVM model.

Parameter tuning The quality of SVM models is very sensitive to the parameters (such as penalty factors of misclassified data) especially in case of using kernels that typically introduce extra parameters. There are many different parameter tuning approaches such as [5, 129, 64, 18, 16, 2, 67, 58]. However, in any case, tuning parameters requires multiple executions of the training process for different parameters and due to the k-fold cross-validation which significantly increases the running time of the entire framework. In our experiments with industrial and healthcare data, not surprisingly, we were unable to find an acceptable quality SVM models without parameter fitting (also known as model selection [24, 128, 95]) which also motivated our work.

2.1.2 Related work

Multiple approaches have been proposed to improve the performance of SVM solvers. Examples include efficient serial algorithms that use a cohort of decomposition techniques [73], shrinking and caching [47], and fast second order working set selection [30]. A popular LibSVM solver [17]

implements the sequential minimal optimization algorithm. In the cases of simple data for which nonlinear SVM is not required such approaches as LibLINEAR [29] demonstrate excellent performance for linear SVM using a coordinate descent algorithm which is very fast but, typically, not suitable for complex or imbalanced data. Another approach to accelerate the QP solvers is a chunking [47], in which the models are solved iteratively on the subsets of training data until the global optimum is achieved.

A typical acceleration of support vector machines is done through parallelization and training on high-performance computing systems using interior-point methods (IPM) [69] applied on the dual problem which is a convex QP. The key idea of the primal-dual IPM is to remove inequality constraints using a barrier function and then resort to the iterative Newton's method to solve the KKT system of the dual problem. For example, in PSVM [130], the algorithm reduces memory use, and parallelizes data loading and computation in IPM. It improves the decomposition-based LibSVM from $O(n^2)$ to $O(np^2/m)$, where m is a number of processors (or heterogeneous machines used), and p is a column dimension of a factorized matrix that is required for effective distribution of the data. The HPSVM solver [60] is also based on solving the primal-dual IPM and uses effective parallelizm of factorization. The approach is specifically designed to take maximal advantage of the CPU-GPU collaborative computation with the dual buffers 3-stage pipeline mechanism, and efficiently handles large-scale training datasets. In HPSVM, the heterogeneous hierarchical memory is explored to optimize the bottleneck of data transfer. The P-packSVM [131] parallelizes the stochastic gradient descent solver of SVM that directly optimizes the primal objective with the help of a distributed hash table and sophisticated data packing strategy. Other works utilize many-core GPUs to accelerate the sequential minimal optimization [75], and other architectures [126].

One of the most well known works in which hierarchical SVM technique was introduced to improve the performance and quality of a classifier is [127]. The coarsesning consists of creating a hierarchical clustered representation of the data points that are merged pairwise using Euclidean distance criterion. In this work, only linear classifiers are discussed and no inheritance and refinement of model parameters was introduced. A similar hierarchical clustering framework was proposed for non-linear SVM kernels in combination with feature selection techniques to develop an advanced intrusion detection system [41].Another coarsening approach that uses k-means clustering was introduced in [42]. *In all these works, the quality of classifiers strictly depends on how well the data is clustered using a particular clustering method applied on it.* Our coarsening scheme is more gradual

and flexible than the clustering methods in these papers. Most of them, however, can be generalized as algebraic multigrid restriction operators (will be discussed further) in special forms. Also, in our frameworks, we emphasize several important aspects of training such as coarse level models, imbalanced coarsening, and parameter learning that are typically not considered in hierarchical SVM frameworks.

Multilevel Divide-and-Conquer SVM (DC-SVM) was developed using adaptive clustering and early prediction strategy [42]. It outperforms previously mentioned methods, so we compare the computational performance and quality of classification for both DC-SVM and our proposed framework. The training time of DC-SVM for *a fixed set of parameters* is fast. However, in order to achieve high quality classifiers a parameter fitting is typically required. While DC-SVM with parameter fitting is faster than state-of-the-art *exact* SVMs, it is significantly slower than our proposed framework. Our experimental results (that include the parameter fitting component) show significant performance improvement on benchmark data sets in comparison to DC-SVM.

In several works, a scalable parallelization of hierarchical SVM frameworks is developed to minimize the communication [125, 36, 25]. Such techniques can be used on top of our framework. Successful results obtained using hierarchical structures have been shown specifically for multi-class classification [21, 39, 52, 77]. Another relevant line of research is related to multilevel clustering and segmentation methods [56, 31, 98]. They produce solutions at different levels of granularity which makes them suitable for visualization, aggregation of data, and building a hierarchical solution.

2.1.3 Multilevel algorithmic frameworks

In this chapter, we discuss a practical construction of multilevel algorithmic frameworks (MAF) for SVM. These frameworks are inspired by the multiscale optimization strategies [11]. (We note that there exist several frameworks termed multilevel SVMs. These, however, correspond to completely different ideas. We preserve the terminology of multilevel, and multiscale optimization algorithms.) The main objective of multilevel algorithms is to construct a hierarchy of problems (coarsening), each approximating the original problem but with fewer degrees of freedom. This is achieved by introducing a chain of successive restrictions of the problem domain into low-dimensional or smaller-size domains and solving the coarse problems in them using local processing (uncoarsening) [66, 27]. The MAF combines solutions obtained by the local processing at different levels of coarseness into one global solution. Such frameworks have several key advantages that make them

attractive for applying on large-scale data: they typically exhibit linear complexity (see Sec. 2.3.3), and are relatively easily parallelized. Another advantage of the MAF is its heterogeneity, expressed in the ability to incorporate external appropriate optimization algorithms (as a refinement) in the framework at different levels. For example, if some SVM model selection technique is found to be particularly successful in parameter finding and obtaining high-quality solutions on some class of datasets, one can incorporate this technique at all levels of MAF and accelerate it by 1) applying it locally, 2) combining local solutions into global, and 3) inheriting parameters trained at coarse levels. These frameworks are extremely successful in various practical machine learning tasks such as clustering [72], segmentation [98], and dimensionality reduction [66].

The major difference between typical computational optimization MAF [88, 89, 38, 100], and those that we introduce for SVM is the output of the model. In SVM, the main output is the set of the support vectors which is usually much smaller at all levels of the multilevel hierarchy than the total number of data points at the corresponding levels. We use this observation in our methods by redefining the training set during the uncoarsening and making MAF scalable. In particular, we inherit the support vectors from the coarse scales, add their neighborhoods, and refine the support vectors at all scales. In other words, we improve the separating hyperplane throughout the hierarchy by gradual refinement of the support vectors until a global solution at the finest level is reached. In addition, we inherit the parameters of model selection and kernel from the coarse levels, and refine them throughout the uncoarsening.

2.1.4 Our contribution

We introduce novel methods of engineering fast and high quality multilevel frameworks for efficient and effective training of nonlinear SVM classifiers. We also summarize and generalize existing [80, 79] approaches. We discuss various coarsening strategies, and introduce the weighted aggregation framework inspired by the algebraic multigrid [11] which significantly improves and generalizes all of them. In the weighted aggregation framework, the data points are either partitioned in hierarchical fashion where small groups of data points are aggregated or split into fractions where different fractions of the same data point can belong to different aggregates. Without any notable loss in the quality of classifiers, multilevel SVM frameworks exhibit substantially faster running times and are able to generate *several* classifiers at different coarse-grained resolutions in one complete training iteration which also helps to interpret these classifiers qualitatively (see Section 2.3.4.8).

Depending on the size and structure of the training set, the resulting final decision rule of our multilevel classifier will be either exactly the same as in single SVM model or composed as voting of several smaller SVM models.

The proposed multilevel frameworks are particularly effective on imbalanced data sets where fitting model parameters is the most computationally expensive component. Our multilevel frameworks can be parallelized as any algebraic multigrid algorithm and their superiority is demonstrated on several publicly available and industrial data sets. The performance improvement over the best sequential state-of-the-art nonlinear SVM libraries with high classification quality is significant. For example, on the average, for large data sets we boost the performance 491 times over LibSVM and 45 times over the DC-SVM (which was chosen because of its superiority over other hierarchical methods mentioned above). On some large datasets, a full comparison was impossible because of infeasible running time of the competitive approaches which demonstrates superiority of the proposed method.

2.2 Preliminaries

We define the optimization problems underlying SVM models for binary classification. Given a set $\mathcal{J}$ that contains n data points $x_i \in \mathbb{R}^d$, $1 \leq i \leq n$, we define the corresponding labeled pairs (x_i, y_i), where each x_i belongs to the class determined by a given label $y_i \in \{-1, 1\}$. Data points with positive labels are called the *minority* class which is denoted by $\mathbf{C}^+$ with $|\mathbf{C}^+| = n^+$. The rest of the points belongs to the *majority* class which is denoted by $\mathbf{C}^-$, where $|\mathbf{C}^-| = n^-$, i.e., $\mathcal{J} = \mathbf{C}^+ \cup \mathbf{C}^-$. Solving the following convex optimization problem by finding w, and b produces a hyperplane with maximum margin between $\mathbf{C}^+$, and $\mathbf{C}^-$

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y_i(w^T\phi(x_i) + b) \geq 1 - \xi_i, \quad i = 1, \ldots, n \\ & \xi_i \geq 0, \quad i = 1, \ldots, n. \end{aligned} \tag{2.1}$$

The mapping of data points to higher dimensional space is done by $\phi : \mathbb{R}^d \to \mathbb{R}^p$ $(d \leq p)$ to make two classes separable by a hyperplane. The term slack variables $\{\xi_i\}_{i=1}^n$ are used to penalize misclassified points. The parameter $C > 0$ controls the magnitude of the penalization. The primal formulation is shown at (2.1) which is known as the *soft margin* SVM [122].

The weighted SVM (WSVM) addresses imbalanced problems with assigning different weights to classes with parameters C^+ and C^-. The set of slack variables is split into two disjoint sets $\{\xi_i^+\}_{i=1}^{n^+}$, and $\{\xi_i^-\}_{i=1}^{n^-}$, respectively. In WSVM, the objective of (2.1) is changed into

$$\text{minimize} \qquad \frac{1}{2}\|w\|^2 + C^+ \sum_{i=1}^{n^+} \xi_i^+ + C^- \sum_{j=1}^{n^-} \xi_j^-. \tag{2.2}$$

Solving the Lagrangian dual problem using kernel functions $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(x_i)^T \phi(x_j)$ produces a reliable convergence which is faster than methods for primal formulations (2.1) and (2.2). In our framework, we use the sequential minimal optimization solver implemented in LibSVM library [17]. The role of kernel functions is to measure the similarity for pairs of points x_i and x_j. We present computational results with the Gaussian kernel (RBF), $\exp(-\gamma||x_i - x_j||^2)$, which is known to be generally reliable when no additional assumptions about the data are known. Experiments with other kernels exhibit improvements that are similar to those with RBF if compared with regular (W)SVM solver with the same kernels. Technically, using another kernel requires only switching to it in the refinement at the uncoarsening stage (see Alg. 3) including parameter inheritance, if required. We note that some of our experimental datasets are not solved well with non-RBF kernels used in regular (W)SVM solver, so here we demonstrate the results only for RBF.

In order to achieve an acceptable quality of the classifier, many difficult data sets require reinforcement of (W)SVM with tuning methods for such model parameters as C, C^+, C^-, and kernel function parameters (e.g., the bandwidth parameter γ for RBF kernel function). This is one of the major sources of running time complexity of (W)SVM models which we are aiming to improve.

In our framework we use the adapted nested uniform design (NUD) model selection algorithm to fit the parameters [44] which is a popular model selection technique for (W)SVM. The main intuition behind NUD is that it finds the close-to-optimal parameter set in an iterative nested manner. The optimal solution is calculated in terms of maximizing the required performance measure (such as accuracy and G-mean). Although, we study binary classification problems, it can easily be extended to the multi-class classification using either directed multi-class classification or transforming the problem into multiple independent binary (W)SVMs that can be processed independently in parallel.

Two-level problem In order to describe the (un)coarsening algorithms, we introduce the two-level problem notation that can be extended into full multilevel hierarchy (see Figure 2.1). We will use subscript $(\cdot)_f$ and $(\cdot)_c$ to represent fine and coarse variables, respectively. For example, the data points of two consecutive levels, namely, fine and coarse, will be denoted by $\mathcal{J}_f$, and $\mathcal{J}_c$, respectively. The sets of fine and coarse support vectors are denoted by sv_f, and sv_c, respectively. We will also use a subscript in the parentheses to denote the level number in the hierarchy where appropriate. For example, $\mathcal{J}_{(i)}$ will denote the set of data points at level i.

Proximity graphs All multilevel (W)SVM frameworks discussed in subsequent sections are based on different coarsening schemes for creating a hierarchy of data proximity graphs. Initially, at the finest level, $\mathcal{J}$ is represented as two k-nearest neighbor (kNN) graphs $G^+_{(0)} = (\mathbf{C}^+, E^+)$, and $G^-_{(0)} = (\mathbf{C}^-, E^-)$ for minority and majority classes, respectively, where each $x_i \in \mathbf{C}^{+(-)}$ corresponds to a node in $G^{+(-)}_{(0)}$. A pair of nodes in $G^{+(-)}_{(0)}$ is connected with an edge that belongs to $E^{+(-)}$ if one of them belongs to a set of k-nearest neighbors of another. In practice, we are using *approximate* k-nearest neighbors graphs (AkNN) as our experiments with the *exact* nearest neighbor graphs do not demonstrate any improvement in the quality of classifiers whereas computing them is a time consuming task. In the computational experiments, we used FLANN library [70, 71]. Results obtained with other approximate nearest neighbor search algorithms are found to be not significantly different. Throughout the multilevel hierarchies, in two-level representation, the fine and coarse level graphs will be denoted by $G^{+(-)}_f = (\mathbf{C}^{+(-)}_f, E^{+(-)}_f)$, and $G^{+(-)}_c = (\mathbf{C}^{+(-)}_c, E^{+(-)}_c)$, respectively. All coarse graphs, except $G^{+(-)}_{(0)}$ are obtained using respective coarsening algorithm.

Multiple models In the proposed multilevel frameworks, when the data is too big, independent training of several subsets of the data will be performed. As a result, a training on k subsets will produce k models that will be denoted as $\{(\mathsf{sv}_f, C^+_f, C^-_f, \gamma_f)_i\}^k_{i=1}$ to avoid introducing additional index for each parameter.

2.3 Multilevel support vector machines

The multilevel frameworks discussed in this chapter include three phases (see Figure 2.1), namely, gradual training set coarsening, coarsest support vector learning, and gradual support vector refinement (uncoarsening). In the training set coarsening phase, we create a hierarchy of coarse

training set representations, $\mathcal{J}_{(i)}$, in which each next-coarser level $(i+1)$ contains a fewer number of points than in the previous level (i) such that the coarse level learning problem approximates the fine level problem. The coarse level training points are not necessarily the same fine level points (such as in [80]) or their strict small clusters (such as in [127]).

When the size of training set is sufficiently small to apply a high quality training algorithm for given computational resources, the set of coarsest support vectors and model parameters are trained. We denote by $M^{+(-)}$ the upper limit for the sizes of coarsest training sets which should depend on the ability of available computational resources to solve the problem exactly in a reasonable time. In the uncoarsening, both the support vectors and model parameters are inherited from the coarse level and improved using local refinement at the fine level. The uncoarsening is continued from the coarsest to the finest levels as is shown in Figure 2.1. Separate coarsening hierarchies are created for classes $\mathbf{C}^+$, and $\mathbf{C}^-$, independently.

The main driving routine, mlsvm-•, of a multilevel (W)SVM framework is presented in Algorithm 1. The SVM cost-sensitive framework is designed similarly with a parameter C, see Eq. (2.1). In Algorithm 1, the functions coarsen−•, uncoarsen−•, and refine−• are the building blocks of the multilevel framework discussed in this chapter. These functions will differ from multilevel framework to framework. The bullet "•" will be replaced with corresponding method names.

2.3.1 Iterative Independent Set Multilevel Framework

We describe the coarsening only for class $\mathbf{C}_f^+$ as the same process works for $\mathbf{C}_f^-$. The multilevel framework (mlsvm-IIS, Alg. 1) with iterative independent set coarsening applies several iterative passes in each of which a set of fine points is selected and added to the set of coarse points $\mathbf{C}_c^+$. In order to cover the space of points uniformly, this is done by selecting independent sets of nodes in G_f^+. The independent set is a set of vertices in a graph whose node-induced subgraph has no edges. We present this coarsening in details in [80].

Coarsening (coarsen-IIS in Alg. 1) We start with selecting a random independent set of nodes (or points), I_0, using one pass over all nodes (i.e., choose a random node to I_0, eliminate it with its neighbors from the graph, and choose the next node). The obtained independent set I_0 is added to the set of coarse points. Then, we remove I_0 from the graph and repeat the same process to find another independent set I_1 which is also added to the set of coarse points. The iterations are repeated until $\sum_k |I_k| \leq Q|\mathbf{C}_f^+|$, where Q is a parameter controlling the size of coarse level space.

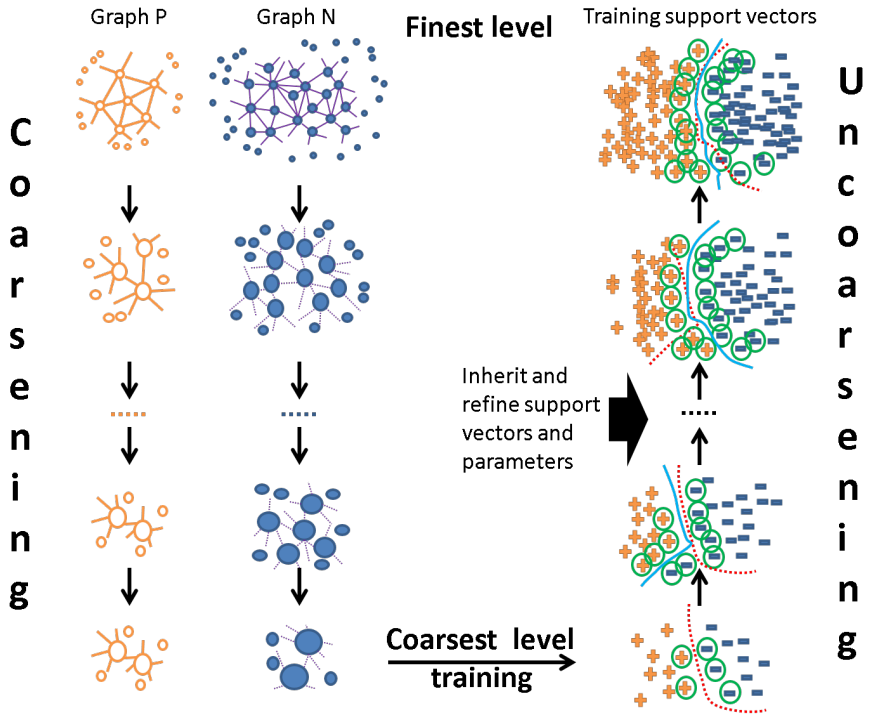


Figure 2.1: Multilevel SVM coarsening-uncoarsening framework scheme.

Algorithm 1 mlsvm-•$(\mathbf{C}_f^+, \mathbf{C}_f^-, G_f^+, G_f^-, M^+, M^-)$: multilevel (W)SVM main driving routine. The functions coarsen−•, uncoarsen−•, and refine−• are the building blocks of the multilevel framework. They will differ from multilevel framework to framework. "•" will be replaced by method names described in following sections.

if $|\mathcal{J}_f| \leq M^+ + M^-$ **then** ▷ Solve the problem exactly if the data is small
  $(\mathsf{sv}_f, C^+, C^-, \gamma) \leftarrow$ train (W)SVM model on $\mathcal{J}_f$ (including NUD)
else ▷ Create and solve a coarse problem. Then refine its solution at level f.
  if $|\mathbf{C}_f^+| \leq M^+$ **then** $\mathbf{C}_c^+ \leftarrow \mathbf{C}_f^+$; $G_c^+ \leftarrow G_f^+$
    else $(\mathbf{C}_c^+, G_c^+) \leftarrow$ coarsen−•$(\mathbf{C}_f^+, G_f^+)$
  if $|\mathbf{C}_f^-| \leq M^-$ **then** $\mathbf{C}_c^- \leftarrow \mathbf{C}_f^-$; $G_c^- \leftarrow G_f^-$
    else $(\mathbf{C}_c^-, G_c^-) \leftarrow$ coarsen−•$(\mathbf{C}_f^-, G_f^-)$
  $(\mathsf{sv}_c, \tilde{C}^+, \tilde{C}^-, \tilde{\gamma}) \leftarrow$ mlsvm-•$(\mathbf{C}_c^+, \mathbf{C}_c^-, G_c^+, G_c^-, M^+, M^-)$
  $\mathsf{sv}_f \leftarrow$ uncoarsen−•(sv_c) ▷ Project support vectors from c to f and add neighbors
  ▷ Get one or more (k) models if the data is too big at current level
  $\{(\mathsf{sv}_f, C^+, C^-, \gamma)_i\}_{i=1}^k \leftarrow$ refine−•$(\mathsf{sv}_f, \tilde{C}^+, \tilde{C}^-, \tilde{\gamma})$
if f is the finest level **then**
  Return k models $\{(\mathsf{sv}_f, C^+, C^-, \gamma)_i\}_{i=1}^k$
else if f is not the finest level and $k = 1$
  Return $(\mathsf{sv}_f, C^+, C^-, \gamma)_1$ ▷ Return a single model with updated parameters
else if f is not the finest level and $k > 1$
  ▷ Return all support vectors from all models and last inherited single parameter set
  Return $(\cup\{\mathsf{sv}_f$ from model $i\}_{i=1}^k, \tilde{C}^+, \tilde{C}^-, \tilde{\gamma})$

In our experiments, $Q = 0.5$. However, experimenting with different $Q \in [0.4, .., 0.6]$ does not affect the quality demonstrating the robustness of this parameter. For too small Q, the coarsening might be too fast and, thus, similar to clustering-based coarsening. The process for $\mathbf{C}_f^-$ is similar.

Coarsest level (line 2, Alg. 1) At the coarsest level ρ, when $|\mathcal{J}_{(r)}| \leq M^+ + M^- \ll |\mathcal{J}_{(0)}|$, we can apply an exact (or computationally expensive) algorithm for training the coarsest classifier. Typically, $|\mathcal{J}_{(\rho)}|$ depends on the available computational resources. However, one can also consider some criteria of separability between $\mathbf{C}_{(\rho)}^+$, and $\mathbf{C}_{(\rho)}^-$ [118], i.e., if a fast test exists or some helpful data properties are known. In all our experiments, we used a simple criterion limiting $|\mathcal{J}_{(\rho)}|$ to 500. Processing the coarsest level includes an application of NUD [44] model selection to get high-quality classifiers on the difficult data sets. To this end, we obtained a solution of the coarsest level, namely, $\mathsf{sv}_{(\rho)}$, $C_{(\rho)}^+$, $C_{(\rho)}^-$, and $\gamma_{(\rho)}$.

Uncoarsening Given the solution of coarse level c, the primary goal of the uncoarsening is to interpolate and refine this solution for the current fine level f. Unlike many other multilevel algorithms, in which the inherited coarse solution contains projected variables only, in our case, we inherit not only sv_c but also parameters for model selection. This is important because the model selection is an extremely time-consuming component of (W)SVM, and can be prohibitive at fine levels of the hierarchy. However, at the coarse levels, when the problem is much smaller than the original, we can apply much heavier methods for the model selection almost without any loss in the total complexity of the framework.

Algorithm 2 uncoarsen-IIS(sv_c): uncoarsening at level f

1: $(N_f^+, N_f^-) \leftarrow$ Find nearest neighbors of support vectors sv_c in G_f^+ and G_f^-
2: $T \leftarrow \mathsf{sv}_c \cup N_f^+ \cup N_f^-$ ▷ T is a new training set for refinement
3: **Return** T

The uncoarsening and refinement are presented in Algorithms 2 and 3, respectively. After the coarsest level is solved exactly and reinforced by the model selection (line 2 in Alg. 1), the coarse support vectors sv_c and their nearest neighbors (in our experiments no more than 5) in both classes (i.e., N_f^+ and N_f^-) initialize the fine level training set T (lines 1-2 in Alg. 2). This completes uncoarsen-IIS (the uncoarsening of sv_c), and T initializes sv_f.

In Alg. 3, the refinement first verifies if $|\mathsf{sv}_f|$ is still small (relatively to the existing computational resources, and the initial size of the data) for applying model selection, i.e., if it is less than a parameter Q_t, then we use coarse parameters $\tilde{C}^{+(-)}$, and $\tilde{\gamma}$ as initializers for the current level

Algorithm 3 refine-IIS($\mathsf{sv}_f, \tilde{C}^+, \tilde{C}^-, \tilde{\gamma}$): refinement at level f

1: **if** $|\mathsf{sv}_f| < Q_t$ **then**
2: $C^O \leftarrow (\tilde{C}^+, \tilde{C}^-)$; $\gamma^O \leftarrow \tilde{\gamma}$
3: $(\mathsf{sv}_f, C_f^+, C_f^-, \gamma_f) \leftarrow$ train (W)SVM using NUD (or similar technique) initialized with (C^O, γ^O)
4: **Return** sv_f, C_f^+, C_f^-, γ_f
5: **else**
6: $C_f^+ \leftarrow \tilde{C}^+$; $C_f^- \leftarrow \tilde{C}^-$; $\gamma_f \leftarrow \tilde{\gamma}$ ▷ Inherit the coarse parameters
7: $CL \leftarrow$ partition sv_f into K (almost) equal size clusters
8: $\forall k \in CL$ find P nearest opposite-class clusters
9: $\{(\mathsf{sv}_f, C_f^+, C_f^-, \gamma_f)_i\}_{i=1}^k \leftarrow$ train (W)SVMs on pairs of nearest clusters with inherited initial parameters C_f^+, C_f^-, γ_f, and generate k models
10: **Return** k models $\{(\mathsf{sv}_f, C_f^+, C_f^-, \gamma_f)_i\}_{i=1}^k$
11: **end if**

NUD grid search, and re-train (lines 2-3 in Alg. 3). Otherwise, the coarse $\tilde{C}^{+(-)}$, and $\tilde{\gamma}$ are inherited in $C_f^{+(-)}$, and γ_f (line 6 in Alg. 3). Then, being large for a direct application of the (W)SVM, T is partitioned into several equal size clusters (using fast solver of balanced k-partitioning [14]), and pairs of nearest opposite clusters are trained (see details in Section 2.3.4.6). The obtained K models are returned (lines 7-10 in Alg. 3). If the current level f is finest then we return all models (line 12 in Alg. 1) otherwise a returned union of support vectors and parameter initializations will pass to the next level (see line 16 in Alg. 1). *We note that partition-based retraining can be done in parallel, as different pairs of clusters are independent. Moreover, the total complexity of the algorithm does not suffer from reinforcing the partition-based retraining with model selection.*

This coarsening scheme is one of the fastest and easily implementable. While the entire framework (including uncoarsening) is definitely much faster than a regular (W)SVM solver such as LibSVM (which is used in our implementation as a refinement), it is not the fastest among the multilevel SVM frameworks. There is a typical trade-off in discrete multilevel frameworks [22, 90], namely, when the quality of coarsening suffers, the most work is done at the refinement. A similar independent set coarsening approach was used in multilevel dimensionality reduction [31]. However, in contrast to that coarsening scheme, we found that using only one independent set (including possible maximization of it) does not lead to the best quality of classifiers. Instead, a more gradual coarsening makes the framework much more robust to the changes in the parameters and the shape of data manifold.

2.3.2 AMG multilevel framework

The algebraic multigrid (AMG) (W)SVM multilevel framework (mlsvm-AMG, Alg. 1) is inspired by the AMG aggregation solvers for computational optimization problems such as [91, 59, 56, 92]. Its first version was briefly presented in [85]. The AMG coarsening generalizes the independent set and clustering [42] based approaches leveraging a high quality coarsening and flexibility of AMG which belongs to the same family of multiscale learning strategies with the same main phases, namely, coarsening, coarsest scale learning, and uncoarsening. However, instead of eliminating a subset of the data points, in AMG coarsening, the original problem is gradually restricted to smaller spaces by creating *aggregates of fine data points and their fractions* (which is an important feature of AMG), and turning them into the data points at coarse levels. The main mechanism underlying the coarsening phase is the AMG [114, 11] which successfully helps to identify the interpolation operator for obtaining a fine level solution from the coarse aggregates. In the uncoarsening phase, the solution obtained at the coarsest level (i.e., the support vectors and parameters) is gradually projected back to the finest level by interpolation and further local refinement of support vectors and parameters. A critical difference between AMG approach and the earlier work of Razzaghi et al. [80] is that in AMG approach the coarse level support vectors are not the original data points prolongated from the finest level. Instead, they are centroids of aggregates that contain both full fine-level data points and their fractions.

Framework initialization The AMG framework is initialized with $G_0^{+(-)}$ with the edge weights that represent the strength of connectivity between nodes in order to "simulate" the following interpolation scheme applied at the uncoarsening, in which strongly coupled nodes can interpolate solution to each other. In the classifier learning problems, this is expressed as a similarity measure between points. We define a distance function between nodes (or corresponding data points) as an inverse of the Euclidean distance. More advanced distance measure approaches such as [12, 19, 20, 48] are often essential in similar multilevel frameworks.

Coarsening Phase (see Algorithm 4, coarsen-AMG) We describe the two-level process of obtaining the coarse level training set $\mathbf{C}_c^+$ with corresponding G_c^+ given the current fine level G_f^+ and its training set (e.g., the transition from level f to c). The majority class is coarsened similarly.

The process is started with selecting seed nodes that will serve as centers of coarse level

nodes, i.e., the aggregates at level f. Coarse nodes will correspond to the coarse data points at level c. Structurally, each aggregate must include one full seed f-level point, and possibly several other f-level points and their fractions. Intuitively, it is equivalent to grouping points in $\mathbf{C}_f^+$ into many small subsets allowing intersections, where each subset of nodes corresponds to a coarse point at level c. During the aggregation process, most coarse points will correspond to aggregates of size greater than 1 (because, throughout the hierarchy, they accumulate many fine points and their fractions), so we introduce the notion of a *volume* $v_i \in \mathbb{R}_+$ for all $i \in \mathbf{C}_f^+$ to reflect the importance of a point or its capacity that includes finest-level aggregated points and their fractions. We also introduce the edge weighting function $w : E_f^+ \to \mathbb{R}_{\geq 0}$ to reflect the strength of connectivity and similarity between nodes.

In Algorithm 4, we show the details of AMG coarsening. In the first step (line 2), we compute the future-volumes ϑ_i for all $i \in \mathbf{C}_f^+$ to determine the order in which f-level points will be tested for declaring them as seeds, namely,

$$\vartheta_i = v_i + \sum_{j \in \Gamma_i \cap \mathbf{C}_f^+} v_j \cdot \frac{w_{ji}}{\sum\limits_{k \in \Gamma_j \cap \mathbf{C}_f^+} w_{jk}}, \tag{2.3}$$

where Γ_i is the neighborhood of node i in G_f^+. The future-volume ϑ_i is defined as a measure (that is often used in multilevel frameworks [90]) of how much an aggregate seeded by a point i may potentially grow at the next level c. This is computed in linear time.

We assume that in the finest level, all volumes are ones. We start with selecting a dominating set of seed nodes $S \subset \mathbf{C}_f^+$ to initialize aggregates. Nodes that are not selected to S remain in F such that $\mathbf{C}_f^+ = F \cup S$. Initially, the set F is set to be $\mathbf{C}_f^+$, and $S = \emptyset$ since no seeds have been selected. After that, points with $\vartheta_i > \eta \cdot \overline{\vartheta}$, i.e., those that are exceptionally larger than the average future volume are transferred to S as the most "representative" points (line 3). Then, all points in F are accessed in the decreasing order of ϑ_i updating S iteratively (lines 7-11), namely, if with the current S, and F, for point $i \in F$, $\sum_{j \in S} w_{ij} / \sum_{j \in \mathbf{C}_f^+} w_{ij}$ is less than or equal to some threshold Q[1], i.e., the point is not strongly coupled to already selected points in S, then i is moved from F to S.

The points with large future-volumes usually have a better chance to serve as seeds and become centers of future coarse points. Selecting too few seeds (and then coarse level points) causes "overcompressed" coarser level which typically leads to the classification quality drop. Therefore, in

[1]Similar parameter Q that controls the speed of coarsening appears in coarsen-IIS.

order to keep sufficiently many points at the coarse level, the parameter Q is set to 0.4-0.6. It has been observed that in most AMG algorithms, $Q > 0.6$ is not required (however, it depends on the type and goals of aggregation). In our experiments $Q = 0.5$, and $\eta = 2$. Other similar values do not significantly change the results.

Algorithm 4 coarsen-AMG($\mathbf{C}_f^+, G_f^+$): AMG coarsening

$S \leftarrow \emptyset, F \leftarrow \mathbf{C}_f^+$ ▷ start select seeds for coarse nodes
Calculate using Eq. (2.3) $\forall i \in F$ ϑ_i, and the average $\bar{\vartheta}$
$S \leftarrow$ nodes with $\vartheta_i > \eta \cdot \bar{\vartheta}$
$F \leftarrow V_f \setminus S$
Recompute ϑ_i $\forall i \in F$
Sort F in descending order of ϑ
for $i \in F$ **do**
    if $\left(\sum_{j \in S} w_{ij} / \sum_{j \in \mathcal{J}_f^+} w_{ij} \right) \leq Q$ **then**
        move i from F to S
    end if
end for ▷ end select seeds for coarse nodes
Build interpolation matrix P according to Eq. (2.4)
Build coarse graph G_c^+ with edge weights using Eq. (2.5)
Define volumes of coarse points using Eq. (2.6)
Compute coarse points $\mathbf{C}_c^+$ using Eq. (2.7)
Return $(\mathbf{C}_c^+, G_c^+)$

When the set S is selected, we compute the AMG interpolation matrix $P \in \mathbb{R}^{|\mathbf{C}_f^+| \times |S|}$ that is defined as

$$P_{ij} = \left\{ \begin{array}{cc} w_{ij} / \sum\limits_{k \in \Gamma_i} w_{ik} & \text{if } i \in F,\, j \in \Gamma_i \\ 1 & \text{if } i \in S,\, j = I(i) \\ 0 & \text{otherwise} \end{array} \right\}, \tag{2.4}$$

where $\Gamma_i = \{j \in S \mid ij \in E_f^+\}$ is the set of ith seed neighbors, and $I(i)$ denotes the index of a coarse point at level c that corresponds to the fine level aggregate around seed $i \in S$. Typically, in AMG methods, the number of non-zeros in each row is limited by the parameter called the interpolation order or caliber [11] (see further discussion about r and Table 2.6). This parameter, r, controls the complexity of a coarse-scale system (the number of non-zero elements in the matrix of coarse k-NN graph). It limits the number of fractions a fine point can be divided into (and thus attached to the coarse points). If a row in P contains too many non-zero elements then it is likely to increase the number of non-zeros in the coarse graph matrix. In multigrid methods, this number is usually

controlled by different approaches that measure the strength of connectivity (or importance) between fine and coarse variables (see discussion and implementation in [83]).

Using the matrix P, the aggregated data points and volumes for the coarse level are calculated. The edge between points $p = I(i)$ and $q = I(j)$ is assigned with weight

$$w_{pq} = \sum\nolimits_{k \neq l} P_{ki} \cdot w_{kl} \cdot P_{lj}. \tag{2.5}$$

The volume for the aggregate $I(i)$ in the coarse graph is computed by

$$\sum\nolimits_{j} v_j P_{ji}, \tag{2.6}$$

i.e., the total volume of all points is preserved at all levels during the coarsening. The coarse point $q \in \mathbf{C}_c^+$ seeded by $i = I^{-1}(q) \in \mathbf{C}_f^+$ is represented by

$$\sum_{j \in \mathcal{A}_i} P_{j,q} \cdot j, \tag{2.7}$$

where $\mathcal{A}_i$ is a set of fine points in aggregate i. This set is extracted from the column of P that corresponds to aggregate i by considering rows j with non-zero values.

The stopping criteria for the coarsening depends on the available computational resources that can be used in order to train the classifier at the coarsest level. In our experiments, the coarsening stops when the size is less than a threshold (typically, 500 points) that ensures a fast performance of the
LibSVM dual solver.

Uncoarsening (see Algorithm 4, uncoarsen-AMG) The uncoarsening of AMG multilevel framework is similar to that of the mlsvm-IIS. The main difference is in lines 1-2 in Alg. 2. Instead of defining the training set for the refinement at level f as

$$T \leftarrow \mathsf{sv}_c \cup N_f^+ \cup N_f^-,$$

all coarse support vectors are uncoarsened by adding to T all elements of the corresponding aggregates, namely,

$$T \leftarrow \emptyset; \quad \forall p \in \mathsf{sv}_c \quad \forall j \in \mathcal{A}_p \quad T \leftarrow T \cup j. \tag{2.8}$$

The rule in (2.8) means the following: 1) take all c-level support vectors p, 2) find all f-level points that are aggregated in c-level support vectors, and 3) add them to T. The basic refinement, refine-AMG, is similar to refine-IIS.

2.3.3 Complexity of multilevel framework

The complexity of MAF for (W)SVM consists of three parts, namely, generating approximated k-NN graphs of both classes, coarsening and uncoarsening. The complexity of generating approximate k-NN graphs is based on FLANN library implementation [70, 71] that was used in our experiments. It includes construction of a k-means tree that is leveraged to search for approximate nearest neighbors. The overall complexity of FLANN is $\mathcal{O}(|\mathcal{J}| \cdot d \cdot (\log n' / \log K))$ where d is the data dimensionality, n' is the number of inner nodes the k-means tree, and K is the number of clusters or branching factor for the k-means. When we compare the running time of 1 V-cycle of our solver and that of parallelized FLANN preprocessing, we observe that FLANN does not significantly increases the running time of the entire framework when we parametrize it to find 10 nearest neighbors.

In the coarsening phase, we need to consider the complexity of coarsening the approximated k-NN graphs of $\mathbf{C}^+$ and $\mathbf{C}^-$ including aggregation of the data points. The complexity of coarsening is similar to that of AMG applied on graph $G = (V, E)$ which is proportional to $|V| + |E|$, where $|E| \approx k|V|$, where k is the number of nearest neighbors. In our experiments, we found that no data set requires $k > 10$ to improve the quality of classification. Because we do not anticipate to obtain exceptionally high-degree nodes during the coarsening, we also do not expect to observe very fast increasing density of nonzero features (nnz) in data points. Thus, we bound the complexity of coarsening with $\mathcal{O}(\text{nnz}(\mathcal{J}))$ (or $\mathcal{O}(|\mathcal{J}|)$ for low dimensional data) without having hidden coefficients, in practice.

The complexity of the uncoarsening mostly depends on that of the underlying QP solver (call it QPS, such as LibSVM) applied at the refinement stage. Another factor that affects the complexity is the number of support vectors found at each scale which is typically significantly smaller than the number of data points. Typically, the complexity will be approximately $\mathcal{O}(nnz(\mathcal{J})) +$

$\mathcal{O}(\text{QPS}(p \text{ points})) \cdot |\text{support vectors}|/p$, where p is the number of parts, the set of support vectors is split to if partitioning is applied. Typically, if the application does not include very dense data, the component $\mathcal{O}(nnz(\mathcal{J}))$ is much smaller than $\mathcal{O}(|\mathcal{J}| \cdot d)$. Overall, the complexity of the entire framework is linear in the number of data points.

The computational time obtained in our experiments and the amount of work per unit is presented in Section 2.4. In particular, in Table 2.14 we demonstrate the computational time per data point and per feature value. In particular, in Figure 2.4, we present the change in running time while training the model with increasingly larger parts of the dataset.

2.3.4 Engineering multilevel framework

The AMG framework generalizes many multilevel approaches by allowing a "soft" weighted aggregation of points (and their fractions) in contrast to the "strict" clustering [42] and subset based aggregations such as our mlsvm-IIS [80]. In this section we describe a variety of improvements we experimented with to further boost the quality of the multilevel classification framework, and improve the performance of both the training and validation processes in terms of the quality and running time. All of them are applicable in both "strict" or "soft" coarsening schemes.

2.3.4.1 Imbalanced classification

One of the major advantages of the proposed coarsening scheme is its natural ability to cope with the imbalanced data in addition to the cost-sensitive and weighted models solved in the refinement. When the coarsening is performed on both classes simultaneously, and in a small class the number of points reaches an allowed minimum, this level is simply copied throughout the rest of levels required to coarsen the big class. Since the number of points at the coarsest level is small, this does not affect the overall complexity of the framework. Therefore, the numbers of points in both classes are within the same range at the coarsest level regardless of how imbalanced they were at the fine levels. Such training on the balanced data mitigates the imbalance effects and improves the performance quality of trained models.

2.3.4.2 Coarse level density problem

Both mlsvm-IIS and mlsvm-AMG do not change the dimensionality during the coarsening which potentially may turn into a significant computational bottleneck for a large-scale data. In

many applications, a high-dimensional data is sparse, and, thus, even if the number of points is large, the space requirements are still not prohibitive. Examples include text tf-idf data and categorical features that are converted into a binary representation. While the mlsvm-IIS coarsening selects *original* (i.e., sparse) points for the coarse levels, the mlsvm-AMG aggregates points using a linear combination such as in Eq. 2.7. Even when the original $j \in \mathcal{A}_i$ are sparse, the points at coarse levels may eventually become much denser than the original points.

The second type of coarse level density is related to the aggregation itself. When f-level data points are divided into several parts to join several aggregates, the number of edges in coarse graphs is increasing when it is generated by $L_c \leftarrow P^T L_f P$ (subject to L_c diagonal entry correction). Finest level graphs that contain high-degree nodes have a good chance to generate very dense graphs at the coarse levels if their density is not controlled. This can potentially affect the performance of the framework.

The coarse level density problem is typical to most AMG and AMG-inspired approaches. We control it by filtering weak edges and using the order of interpolation in aggregation. The weak edges not only increase the density of coarse levels but also may affect the quality of the training process during the refinement. In mlsvm-AMG framework, we eliminate weak edges between i and j if $w_{ij} < \theta \cdot \text{avg}_{ki}\{w_{ki}\}$ and $w_{ij} < \theta \cdot \text{avg}_{kj}\{w_{kj}\}$ where avg$\{\cdot\}$ is the average of corresponding adjacent edge weights. We experimented with different values of θ between 0.001 and 0.005 which was typically a robust parameter that does not require much attention.

The order of interpolation, r, is the number of nonzeros allowed per row in P. A single nonzero jth entry in row i, $P_{ij} = 1$, means that a fine point i fully belongs to aggregate j which leads to creation of small clusters of fine points without splitting them. Typically, in AMG methods, increasing r improves the quality of solvers making them, however, slower. We experiment with different values of r and conclude that high interpolation orders such as 2 and 4 perform better than 1. In the same time, we observed that there is no practical need to increase it more (see further discussion and example in Sec. 2.4.2).

2.3.4.3 Validation for model selection

The problem of finding optimal parameters (i.e., the model selection) is important for achieving a better quality on many data sets. Typically, this component is computationally expensive because repetitive training is required for different choices of parameters. A validation data is then

required to choose the best trained model. A performance of model selection techniques is affected by the quality and size of the validation data. (We note that the test data for which the computational results are presented remains completely isolated from any training and validation.)

The problem of a validation set choice requires a special attention in multilevel frameworks because the models at the coarse levels should not necessarily be validated on the corresponding coarse data. As such, we propose different approaches to find the most suitable types of validation data. We developed the following approaches to choose validation set for multilevel frameworks, namely, *coarse sampling* (CS), *coarse cross k-fold* (CCkF), *finest full* (FF), and *fine sampling* (FS).

CS: The data in $\mathcal{J}_{(i)}^{+}$ and $\mathcal{J}_{(i)}^{-}$ is sampled and one part of it (in our experiments 10% or 20%) is selected for a validation set for model selection. In other words, the validation is performed on the data at the same level. This approach is extremely fast on the data in which the coarsening is anticipated to be uniform without generating a variability in the density of aggregation in different parts of the data. Typically, its quality is acceptable on homogeneous data. However, qualitatively, this approach may suffer from a small size of the validation data in comparison to the size of test data.

CCkF: In this method we apply a complete k-fold cross validation at all levels using the coarse data from the same level. The disadvantage of this method is that it is more time consuming but the quality is typically improved. During the k-fold cross validation, all data is covered. With this method, the performance measures are improved in comparison to the CS but the quality of the finest level can degrade because of potential overfitting at the coarse levels.

FF: This method exploits a multilevel framework by combining a coarse training set $\mathcal{J}_{c}^{+(-)}$ with a validation set that is a whole finest level training set $\mathcal{J}_{(0)}^{+(-)}$. The idea behind this approach is to choose the best model which increases a required performance measure (such as accuracy, and G-mean) of coarse aggregates with respect to the original data rather than the aggregates. This significantly increases the quality of final models. However, this method is time consuming on very large data sets as all original points participate in validation.

FS: This method resolves the complexity of FF by sampling $\mathcal{J}_{(0)}^{+(-)}$ to serve as a validation set at the coarse levels. The size of sampling should depend on computational resources. However, we note that we have not observed any drop in quality if it is more than 10% of the $\mathcal{J}_{(0)}^{+(-)}$. Both FF and FS exhibit the best performance measures.

2.3.4.4 Underlying solver

At all iterations of the refinement and at the coarsest level we used LibSVM [17] as an underlying solver by applying it on the small subsets of data (see lines 3 and 9 in Alg. 3). Depending on the objective Eq. (2.1) or (2.2), SVM or WSVM solvers are applied. In this chapter we report the results of WSVM in which the objective Eq. (2.2) is given by

$$\mathsf{minimize} \qquad \frac{1}{2}\|w\|^2 + C\big(W^+ \sum_{i=1}^{n^+} \xi_i^+ + W^- \sum_{j=1}^{n^-} \xi_j^-\big), \tag{2.9}$$

where the optimal C and γ are fitted using model selection, and the class importance coefficients are W^+ and W^-. While for the single-level WSVM, the typical class importance weighting scheme is

$$W^+ = \frac{1}{|\mathcal{J}^+|}, \quad W^- = \frac{1}{|\mathcal{J}^-|}, \tag{2.10}$$

in MAF, the aggregated points in each class have different importance due to the different accumulated volume of finer points. The aggregated points which represent more fine points are more important than aggregated points which represent small number of fine points. Therefore, the MAF approach for calculating the class weights is based on sum of the volumes in each class, i.e.,

$$W^+ = \frac{1}{\sum\limits_{i\in\mathcal{J}^+} v_i}, \quad W^- = \frac{1}{\sum\limits_{i\in\mathcal{J}^-} v_i}. \tag{2.11}$$

This method, however, ignores the importance of each point in its class. We find that the most successful penalty scheme is the one that is personalized per point, and adapt (2.11) to be

$$\forall i \in \mathcal{J}^{+(-)} \quad W_i = W^{+(-)} \frac{v_i}{\sum\limits_{j\in\mathcal{J}^{+(-)}} v_j}. \tag{2.12}$$

In other words, we consider the relative volume of the point in its class but also multiply it by an inverse of the total volume of the class which gives more weight to a small class. This helps to improve the correctness of a small class classification.

2.3.4.5 Expanding training set in refinement

Typically, in many applications, the number of support vectors is much smaller than $|\mathcal{J}|$. This observation allows some freedom in exploring the space around support vectors inherited from coarse levels. This can be done by adding more points to the refinement training set in attempt to improve the quality of a hyperplane. We describe several possible strategies that one can follow in designing a multilevel (W)SVM framework.

Full disaggregation: This is a basic method presented in (2.8) in which all aggregates of coarse support vectors contribute all their elements to the training set. It is a default method in mlsvm-AMG.

k-distant disaggregation: In some cases, the quality can be improved by adding to T the k-distant neighbors of aggregate elements. In other words, after (2.8), we apply

$$\forall p \in T \quad T \leftarrow T \cup N_f^{+(-)}(p), \tag{2.13}$$

where $N_f^{+(-)}(p)$ is a set of neighbors of p in $G_f^{+(-)}$ depending on the class of p. Similarly, one can add points within distance k from the aggregates of inherited support vectors. Clearly, this can only improve the quality. However, the refinement training is expected to be increasingly slower especially when the $G_f^{+(-)}$ contains high-degree nodes. Even if the finest level graph nodes are of a small degree, this could happen at the coarse levels if a proper edge filtering and limiting interpolation order are not applied. In very rare cases, we observed a need for adding distance 2 neighbors.

Sampling aggregates: In some cases, the coarse level aggregates may become very dense which does not improve the quality of refinement training. Instead, it may affect the running time. One way to avoid of unnecessary complexity is to sample the elements of aggregates. A better way to sample them than a random sampling (after adding the seed) is to order them by the interpolation weights P_{ij}. The ascending order which gives a preference to the fine points that are split across more than one aggregate was the most successful option in our experiments. Fine non-seed points whose $P_{ij} = 1$ are likely to have high similarity with the seeds which does not improve the quality of the support vectors.

2.3.4.6 Partitioning in the refinement

When the number of uncoarsened support vectors at level f is too big for the available computational resources, multiple small-size models are trained and either validated or used as a final output. Small-size models are required for applying model selection in a reasonable computational time. For this purpose we partition the current level training sets $\mathbf{C}_f^{+(-)}$ (see lines 7-10 in Alg. 3) into k parts of approximately equal size using fast *graph partitioning* solvers [14]. Note that applying similar *graph clustering* strategies may lead to highly imbalanced parts which will make the whole process of refinement acceleration useless.

In both mlsvm-AMG and mlsvm-IIS, we leverage the graphs of both classes G_f^+ and G_f^- with the inverses of Euclidean distance between nodes playing the role of edge weights. After both graphs are partitioned, two sets of approximately equal size partitions, Π_f^+ and Π_f^- are created. For each part $\pi_i \in \Pi_f^+ \cup \Pi_f^-$ we compute its centroid c_i in order to estimate the nearest parts of opposite classes and train multiple models by pairs of parts.

The training by pairs of parts works as follows. For each c_i we find the nearest c_j such that i and j are in different classes and evaluate at most $|\Pi_f^+| + |\Pi_f^-|$ models for different choices of (π_i, π_j) pairs (without repetitions which often appear in practice making the process fast). The set of all generated models is denoted by $\mathcal{M}_f$. We note that the training of such pairs is independent and can be easily parallelized.

There are multiple ways one can test (or validate) a point using all models "voting". The simplest strategy which performs well on many data sets is a majority voting. However, the most successful way to generate a prediction was a voting by the relative distance from the test point t to the weighted center of the segment connecting c_i and c_j, namely,

$$x_{ij} = \frac{c_i \sum_{q \in \pi_i} v_q + c_j \sum_{q \in \pi_j} v_q}{\sum_{q \in \pi_i \cup \pi_j} v_q}, \tag{2.14}$$

where v_q is the volume of point q. For all pairs of nearest parts i and j, the label of t is computed as

$$\text{sign}\Big(\frac{\sum_{ij \in \mathcal{M}_f} l_{ij}(t) d^{-1}(t, x_{ij})}{\sum_{ij \in \mathcal{M}_f} d^{-1}(t, x_{ij})}\Big), \tag{2.15}$$

where $l_{ij}(t)$ is a label of ij model for point t, and $d(\cdot, \cdot)$ is a distance function between two points. We experimented with several distance functions to express the proximity of parts (i.e., the way we

choose pairs (π_i, π_j)) and $d(\cdot, \cdot)$, namely, Euclidean, exponential, and Manhattan. The quality of final models obtained using Euclidean distance was the highest.

If the partitioning refinement is applied at the finest level then Algorithm 1 outputs all generated finest level models, and the prediction works according to Eq. (2.15). Otherwise, if the partitioning refinement occurs in the middle levels then the next finer level will receive a union of all support vectors from the models (line 16 in Alg. 1) and model parameters inherited from last level in which a single model was trained. We note that it often might be the case that a partitioning refinement generates models with relatively small total number of support vectors such that at the next finer level, their union can be considered as an input to train a single model.

2.3.4.7 Model Selection

The MAF allows a flexible design for model selection techniques such as various types of parameter grid search [18], NUD [44] that we use in our computational experiments, and other search approaches [64, 5, 128]. A mechanism that typically works behind most of such search techniques evaluates different combinations of parameters (such as C^+, C^-, and γ) and chooses the one that exhibits the best performance measure. Besides the general applicability of model selection because the number of inherited and disaggregated support vectors (in the uncoarsening of mlsvm-IIS and mlsvm-AMG) is typically smaller than that of the corresponding training set, the MAF has the following advantages.

Fast parameter search: In many cases, there is no need to test all combinations of the parameters. The inherited c-level parameters can serve as a center point for their refinement only. For example, NUD suggests two-stage search strategy. In the first stage a wide range of parameters is considered. In the second stage, the best combination from the first stage is locally refined using a smaller search range. In MAF, we do not need to apply the first stage as we only refine the inherited c-level parameters. Other grid search methods can be adjusted in a similar way.

Selecting suitable performance measures for the best model: In MAF, a criterion for choosing the best model throughout the hierarchy is more influential than that at the finest level in non-MAF frameworks. Moreover, these criteria can be different at different levels. For example, when one focuses on highly imbalanced sets, a criteria such as the best G-mean could be more beneficial than the accuracy. We found that introducing 2-level criteria for imbalanced sets such as (a) choose the best G-mean, and (b) among the combinations with the best G-mean choose the best

sensitivity, performs particularly good if applied at the coarse levels when the tie breaker may be often required.

2.3.4.8 Models at different levels of coarseness

Over- and under-fitting are among the key problems of model selection and classifiers, in general. The MAF successfully helps to tackle them. Throughout the hierarchy, we solve (W)SVM models at different levels of coarseness. Intuitively, the coarsening procedure gradually creates generalized (or summarized) representations of the finest level data which results in generalized coarse hyperplanes which can also be used as *final solutions*. Indeed, at the finest level, rich data can easily lead to over-fitted models, a phenomenon frequently observed in practice [28]. In the same time, over-compressed data representation may lead to an under-fitted model because no fine details are considered. In a multilevel framework, one can use models from multiple levels of coarseness because the most correct validation is done against the fine level data in any case. Our experiments confirm that more than half of the best models are obtained from the coarse (but not coarsest) and middle levels which typically prevents over- and under-fitting.

If the best validation was obtained at the middle level and at this level the framework generated multiple models using partitioning refinement (see Section 2.3.4.6) then these multiple models will be the output of Alg. 1 and the prediction will work according to Eq. (2.15). In general, if the best models were produced by the finest and middle levels, we recommend to use the middle level model to avoid potential over-fitting. This recommendation is based on the observation that same quality models can be generated by different hyperplanes but finest models may contain a large number of support vectors that can lead to over-fitting. However, it is a general thought that requires further exploration. In our experiments, no additional parameters or conditions are introduced to choose the final model. We simply choose the best model among those generated at different levels.

2.4 Computational Results

We compare our algorithms in terms of classification quality and computational performance to the state-of-the-art sequential SVM algorithms LibSVM, DC-SVM, and fast Ensemble SVM. The DC-SVM is a most recent, fast, hierarchical approach that outperforms other hierarchical methods

which was the reason to choose it for comparison. The classification quality is evaluated using the following performance measures: sensitivity (SN), specificity (SP), geometric mean (G-mean), and accuracy (ACC), Precision (PPV), and F1, namely,

$$\text{SN} = \frac{TP}{TP+FN}, \quad \text{SP} = \frac{TN}{TN+FP}, \quad \text{G-mean} = \sqrt{\text{SP} \cdot \text{SN}},$$

$$\text{ACC} = \frac{TP+TN}{FP+TN+TP+FN}, \quad \text{Precision (PPV)} = \frac{TP}{TP+FP},$$

where TN, TP, FP, and FN correspond to the numbers of true negative, true positive, false positive, and false negative points. Our main metric for comparison is G-mean which measures the balance between classification quality on both the majority and minority classes. This metric is illuminating for imbalanced classification as a low G-mean is an indication of low-quality classification of the positive data points even if the negative points classification is of high quality. This measure indicates over-fitting of the negative class and under-fitting of the positive class, a critical problem in imbalanced datasets.

In all experiments the data is normalized using z-score. Each experimental result in the following tables represents an average over 100 executions of the same type with different random seeds. The computational time reported in all experiments contains generating the k-NN graph. The computational time is reported in seconds unless it is explicitly mentioned otherwise.

In each class, a part of the data is assigned to be the test data using k-fold cross validation. We experimented with k=5 and 10 (no significant difference was observed). The experiments are repeated k times to cover all the data as test data. The data randomly shuffled for each k-fold cross validation.The presented results are the averages of performance measures for all k folds. Data points which are not in the test data are used as the training data in $\mathcal{J}^{+(-)}$. The test data is never used for any training or validation purposes. The Metis library [50] is used for graph partitioning during the refinement phase. We present the details about data sets in Table 2.1. The imbalance of datasets is denoted by ϵ.

The Forest data set [32] has 7 classes and different classes are reported in the literature (typically, not the difficult ones). Class 5 is used in our experiments as the most difficult and highly imbalanced. We report our results on other classes which are listed in Table 2.2 for convenient comparison with other methods.

Table 2.1: Benchmark data sets.

Dataset	ϵ	n_f	$\lvert\mathcal{J}\rvert$	$\lvert\mathbf{C}^+\rvert$	$\lvert\mathbf{C}^-\rvert$
Advertisement	0.86	1558	3279	459	2820
Buzz	0.80	77	140707	27775	112932
Clean (Musk)	0.85	166	6598	1017	5581
Cod-rna	0.67	8	59535	19845	39690
EEG Eye State	0.55	14	14980	6723	8257
Forest (Class 5)	0.98	54	581012	9493	571519
Hypothyroid	0.94	21	3919	240	3679
ISOLET	0.96	617	6238	240	5998
Letter	0.96	16	20000	734	19266
Nursery	0.67	8	12960	4320	8640
Protein homology	0.99	74	145751	1296	144455
Ringnorm	0.50	20	7400	3664	3736
Twonorm	0.50	20	7400	3703	3697

Table 2.2: The Forest data set classes with $n_f = 54$ and $|\mathcal{J}| = 581012$

Class No	ϵ	$\lvert\mathbf{C}^+\rvert$	$\lvert\mathbf{C}^-\rvert$
Class 1	0.64	211840	369172
Class 2	0.51	283301	297711
Class 3	0.94	35754	545258
Class 4	1.00	2747	578265
Class 5	0.98	9493	571519
Class 6	0.97	17367	563645
Class 7	0.96	20510	560502

2.4.1 mlsvm-IIS results

The performance measures of single- (LibSVM) and multi-level (W)SVMs are computed and compared in Table 2.3. In our earlier work [80], it has been shown in that the multilevel (W)SVM produces similar results compared to the single-level (W)SVM, but it is much faster (see Table 2.4). All experiments on all data sets have been executed on a single machine Intel Core i7-4790, 3.60GHz, and 16 GB RAM. The framework ran in sequential mode with no parallelization using Ubuntu 14.04.5 LTS, Matlab 2012a, Metis 5.0.2, and FLANN 1.8.4.

2.4.2 mlsvm-AMG sparsity preserving coarsening

We have experimented with the light version of mlsvm-AMG in which instead of computing a linear combination of f-level points to get c-level points (see Eq. 2.7), we prolongate the seed to be a corresponding coarse point in attempt to preserve the sparsity of data points. In terms of quality of classifiers, the performance measures of this method are similar to that of mlsvm-IIS and in most

Table 2.3: Quality comparison using performance measures for multi- and single-level of (W)SVM. Each cell contains an average over 100 executions including model selection for each of them. Column "Depth" shows the number of levels. The best results are highlighted in bold font.

		Multilevel					Single-level			
	Dataset	ACC	SN	SP	G-mean	Depth	ACC	SN	SP	G-mean
SVM	Advertisement	0.94	0.97	0.79	**0.87**	7	0.92	0.99	0.45	0.67
	Buzz	0.94	0.96	0.85	**0.90**	14	0.97	0.99	0.81	0.89
	Clean (Musk)	1.00	1.00	0.99	**0.99**	5	1.00	1.00	0.98	**0.99**
	Cod-rna	0.95	0.93	0.97	0.95	9	0.96	0.96	0.95	**0.96**
	EEG Eye State	0.83	0.82	0.88	0.85	6	0.88	0.90	0.86	**0.88**
	Forest (Class 5)	0.93	0.93	0.90	0.91	33	1.00	1.00	0.86	**0.92**
	Hypothyroid	0.98	0.98	0.74	**0.85**	4	0.99	1.00	0.71	0.83
	ISOLET	0.99	1.00	0.83	**0.92**	11	0.99	1.00	0.85	**0.92**
	Letter	0.98	0.99	0.95	0.97	8	1.00	1.00	0.97	**0.98**
	Nursery	1.00	0.99	0.98	0.99	10	1.00	1.00	1.00	**1.00**
	Protein homology	1.00	1.00	0.72	0.85	18	1.00	1.00	0.80	**0.89**
	Ringnorm	0.98	0.98	0.99	**0.98**	6	0.98	0.99	0.98	**0.98**
	Twonorm	0.97	0.98	0.97	0.97	6	0.98	0.98	0.99	**0.98**
WSVM	Advertisement	0.94	0.96	0.80	**0.88**	7	0.92	0.99	0.45	0.67
	Buzz	0.94	0.96	0.87	**0.91**	14	0.96	0.99	0.81	0.89
	Clean (Musk)	1.00	1.00	0.99	**0.99**	5	1.00	1.00	0.98	0.99
	Cod-rna	0.94	0.97	0.95	**0.96**	9	0.96	0.96	0.96	**0.96**
	EEG Eye State	0.87	0.89	0.86	**0.88**	6	0.88	0.90	0.86	**0.88**
	Forest (Class 5)	0.92	0.92	0.90	0.91	33	1.00	1.00	0.86	**0.93**
	Hypothyroid	0.98	0.98	0.75	**0.86**	4	0.99	1.00	0.75	**0.86**
	ISOLET	0.99	1.00	0.85	**0.92**	11	0.99	1.00	0.85	**0.92**
	Letter	0.99	0.99	0.96	**0.99**	8	1.00	1.00	0.97	**0.99**
	Nursery	1.00	0.99	0.98	0.99	10	1.00	1.00	1.00	**1.00**
	Protein homology	1.00	1.00	0.87	**0.92**	18	1.00	1.00	0.80	0.89
	Ringnorm	0.98	0.97	0.99	**0.98**	6	0.98	0.99	0.98	**0.98**
	Twonorm	0.97	0.98	0.97	0.97	6	0.98	0.98	0.99	**0.98**

cases (see Tables 2.4-2.5) are faster. However, for Buzz and Cod-rna datasets, although mlsvm-AMG performs faster, it results in a lower sensitivity and specificity (see Table 2.5) for SVM, and higher sensitivity and specificity for WSVM (see Table 2.5) compared to mlsvm-IIS. For Protein dataset, the sensitivity and specificity are improved compared to mlsvm-IIS (see Table 2.5).

We perform the sensitivity analysis of the order of interpolation denoted by r (see Eq. 2.4), the maximum number of fractions a point in F can be divided into, and compare the performance measures and computational time in Table 2.6. As r increases, the performance measures such as G-mean are improving until they do not stop changing for larger r. For example, for Buzz dataset, the G-mean is not changing for larger $r = 6$. The presented results are computed without advancements FF and FS (see Section 3.3). Using these techniques, we obtain G-mean 0.95 with $r = 1$ for Buzz data set. Higher interpolation orders increase the time but produce the same quality on that data

Table 2.4: Comparison of computational time for single- (LibSVM) and multilevel (mlsvm-IIS and sparse mlsvm-AMG) solvers in seconds. Presented values include running time in seconds for both WSVM and SVM with model selection.

Dataset	mlsvm-IIS	Sparse mlsvm-AMG	Single-level
Advertisement	196	**91**	412
Buzz	2329	**957**	70452
Clean (Musk)	30	**6**	167
Cod-rna	172	**92**	1611
EEG Eye State	51	**45**	447
Forest (Class 5)	13785	**13328**	352500
Hypothyroid	**3**	**3**	5
ISOLET	69	**64**	1367
Letter	45	**18**	333
Nursery	63	**33**	519
Protein homology	**1564**	1597	73311
Ringnorm	**4**	5	42
Twonorm	**4**	**4**	45

set.

2.4.3 Full mlsvm-AMG coarsening

The best version of full mlsvm-AMG coarsening whose results are reported, chooses the best model from different scales (see Sec. 2.3.4.8). For this type of mlsvm-AMG, all experiments on all data sets have been executed on a single machine with CPU Intel Xeon E5-2665 2.4 GHz and 64 GB RAM. The framework runs in sequential mode. The FLANN library is used to generate the approximated k-NN graph for $k = 10$. Once it is generated for the whole data set, its result is saved and reused. In all experiments all data points are randomly reordered as well as for each k-fold, the indices from the original test data are removed and reordered, so *no order in which points are entered into QP solver affects the solution*. Each experiment includes a full k-fold cross validation. The average performance measures over 5 experiments each of which includes 10-fold cross validation are demonstrated in Tables 2.8 and 2.9. The best model among all the levels for each fold of cross validation is selected using *validation* data (see Sec. 2.3.4.7). Using the best model, the performance measures over the test data are calculated and reported as the final performance for this specific fold of cross validation.

For the purpose of comparison, the results of previous work using validation techniques CS, CCkF without partitioning the training data during the refinement [85] are presented in Table 2.7.

The results using validation techniques FF, FS with partitioning the training data during

Table 2.5: Performance measures of regular and weighted mlsvm-AMG. Column 'Depth' shows the number of levels in the multilevel hierarchy which is independent of SVM type.

	regular mlsvm-AMG				weighted mlsvm-AMG				Depth
Dataset	ACC	SN	SP	G-mean	ACC	SN	SP	G-mean	
Advertisement	0.95	0.99	0.64	**0.86**	0.95	0.99	0.64	**0.86**	2
Buzz	0.87	0.89	0.79	0.83	0.93	0.95	0.85	**0.90**	8
Clean	0.99	1.00	0.98	**0.99**	0.99	1.00	0.98	**0.99**	4
Cod-rna	0.86	0.85	0.88	0.87	0.89	0.89	0.90	**0.90**	6
EEG Eye State	0.87	0.88	0.85	**0.86**	0.87	0.88	0.85	**0.86**	4
Forest (Class 5)	0.97	0.98	0.79	0.88	0.96	0.97	0.82	**0.89**	9
ISOLET	0.99	1.00	0.83	**0.91**	0.99	1.00	0.83	**0.91**	3
Letter	0.99	0.99	0.95	**0.97**	0.99	0.99	0.93	0.96	5
Nursery	0.99	0.99	1.00	0.99	1.00	1.00	1.00	**1.00**	4
Protein homology	0.97	0.97	0.86	**0.91**	0.97	0.97	0.85	**0.91**	5
Ringnorm	0.98	0.98	0.98	**0.98**	0.98	0.99	0.98	**0.98**	3
Twonorm	0.98	0.97	0.98	**0.98**	0.98	0.97	0.98	**0.98**	3

Table 2.6: Sensitivity analysis of interpolation order r in mlsvm-AMG for Buzz data set.

	Metric	$r=1$	$r=2$	$r=3$	$r=4$	$r=5$	$r=6$	$r=10$
mlsvm-AMG SVM	G-mean	0.26	0.33	0.56	0.83	0.90	**0.91**	0.89
	SN	0.14	0.33	0.68	0.89	**0.98**	0.97	0.95
	SP	0.47	0.34	0.47	0.79	0.82	**0.86**	0.82
	ACC	0.21	0.33	0.64	0.87	**0.95**	**0.95**	0.94
mlsvm-AMG WSVM	G-mean	0.26	0.40	0.60	0.90	0.93	0.93	**0.94**
	SN	0.14	0.32	0.74	0.95	**0.98**	**0.98**	**0.98**
	SP	0.47	0.5	0.48	0.85	0.88	**0.89**	**0.89**
	ACC	0.21	0.35	0.69	0.93	0.96	**0.97**	**0.97**
	time(sec.)	389	541	659	957	1047	1116	1375

the refinement phase are presented in Tables 2.8, 2.9. We compare our performance and quality with those obtained by LibSVM, DC-SVM, and Ensemble SVM. All results are related to WSVM. The "Single level WSVM" column in Table 2.8 represents the weighted SVM results produced by LibSVM. The LibSVM solver is slow but it produces almost the best G-mean results over our experimental datasets except Advertisement, Buzz, and Forest. The DC-SVM [42] produces better G-mean on 4 datasets compare to LibSVM (see Table 2.8) but has lower G-mean on 4 other datasets. We choose DC-SVM not only because it has a hierarchical framework (with different principles of (un)coarsening) but also because it significantly outperforms other hierarchical techniques which are typically fast but not of high quality.

The mlsvm-AMG demonstrates significantly better computation time than DC-SVM on almost all datasets (see Table 2.8). Furthermore, mlsvm-AMG classification quality is significantly better on both Advertisement and Buzz datasets compared to LibSVM. In addition, the comparison

Table 2.7: Performance measures and running time (in seconds) for weighted single level SVM (LibSVM), and weighted mlsvm-AMG on benchmark data sets in [61] *without partitioning*.

	Single level WSVM					mlsvm-AMG				
Dataset	ACC	SN	SP	G-mean	Time	ACC	SN	SP	G-mean	Time
Advertisement	0.92	0.99	0.45	0.67	231	0.83	0.92	0.81	**0.86**	**213**
Buzz	0.96	0.99	0.81	0.89	26026	0.88	0.97	0.86	**0.91**	**233**
Clean (Musk)	1.00	1.00	0.98	**0.99**	82	0.97	0.97	0.97	0.97	**7**
Cod-RNA	0.96	0.96	0.96	**0.96**	1857	0.94	0.97	0.92	0.95	**102**
Forest	1.00	1.00	0.86	**0.92**	353210	0.88	0.92	0.88	0.90	**479**
Hypothyroid	0.99	1.00	0.75	0.86	3	0.98	0.83	0.99	**0.91**	**3**
ISOLET	0.99	1.00	0.85	0.92	1367	0.99	0.89	1.00	**0.94**	**66**
Letter	1.00	1.00	0.97	**0.99**	139	0.98	1.00	0.97	**0.99**	**12**
Nursery	1.00	1.00	1.00	**1.00**	192	1.00	1.00	1.00	**1.00**	**2**
Ringnorm	0.98	0.99	0.98	**0.98**	26	0.98	0.98	0.98	**0.98**	**2**
Twonorm	0.98	0.98	0.99	**0.98**	28	0.98	0.98	0.97	**0.98**	**1**

between DC-SVM and mlsvm-AMG shows that the latter has higher G-mean for Advertisement, Buzz, Clean, Cod, Ringnorm, and Twonorm datasets. A better performance of DC-SVM is observed on Forest dataset if mlsvm-AMG is applying partitioning, i.e., when the number of support vectors is big. However, in another version of multilevel framework with validation techniques CS, CCkF without partitioning the training data during the refinement, the G-mean raises to 0.90 (see Table 2.7). It is interesting to note that the dimensionality of Advertisement dataset is the main source of complexity for the parameter fitting in both LibSVM and mlsvm-AMG. All versions of multilevel SVMs produce G-mean 0.90 for this dataset which is significantly higher than that of LibSVM which is 0.67. The results for this dataset are not significantly different for DC-SVM which is, however, 3 times slower than full mlsvm-AMG and 6 times slower than sparse mlsvm-AMG.

The computational time in seconds is demonstrated in Table 2.9. Our experiments exhibit significant performance improvement.

2.4.3.1 Large datasets

Large datasets SUSY and Higgs are available at UCI repository [61]. The MNIST8M was downloaded from LibSVM data repository. Half of each class was randomly sampled to make classification more difficult. All the methods (our and competitors') are benchmarked using Intel Xeon (E5-2680v3) with 128Gb memory.

The experiments with DC-SVM have not been finished after 3 full days of running and its

Table 2.8: Performance measures for single level WSVM (LibSVM), DC-SVM and mlsvm-AMG on benchmark data sets using partitioning and FF, FS validation techniques.

	Single level WSVM				DC-SVM				mlsvm-AMG			
Datasets	ACC	SN	SP	G-mean	ACC	SN	SP	G-mean	ACC	SN	SP	G-mean
Advertisement	0.92	0.99	0.45	0.67	0.95	0.83	0.97	0.90	0.95	0.85	0.96	**0.91**
Buzz	0.96	0.99	0.81	0.89	0.96	0.88	0.97	0.92	0.94	0.95	0.94	**0.95**
Clean (Musk)	1.00	1.00	0.98	**0.99**	0.96	0.91	0.97	0.94	0.99	0.99	0.99	**0.99**
Cod-RNA	0.96	0.96	0.96	**0.96**	0.93	0.93	0.94	0.93	0.93	0.97	0.91	**0.94**
Forest	1.00	1.00	0.86	0.92	1.00	0.88	1.00	**0.94**	0.77	0.96	0.80	0.88
Letter	1.00	1.00	0.97	0.99	1.00	1.00	1.00	**1.00**	0.98	0.99	0.98	0.99
Nursery	1.00	1.00	1.00	**1.00**	1.00	1.00	1.00	**1.00**	1.00	1.00	1.00	**1.00**
Ringnorm	0.98	0.99	0.98	**0.98**	0.95	0.92	0.98	0.95	0.98	0.98	0.98	**0.98**
Twonorm	0.98	0.98	0.99	**0.98**	0.97	0.98	0.96	0.97	0.98	0.98	0.97	**0.98**

Table 2.9: Computational time in seconds for single level WSVM (LibSVM), DC-SVM and mlsvm-AMG.

Dataset	Single level WSVM	DC-SVM	mlsvm-AMG
Advertisement	231	610	**213**
Buzz	26026	2524	**31**
Clean (Musk)	**82**	95	94
Cod-RNA	1857	420	**13**
Forest	353210	19970	**948**
Letter	139	38	**30**
Nursery	192	49	**2**
Ringnorm	26	38	**2**
Twonorm	28	30	**1**

performance is not presented because of unrealistic slowness of the method. Therefore, it is not comparable with mlsvm-AMG on large datasets. The LibSVM performs slower than DC-SVM on these datasets and is also not presented. Although, fast linear SVM solvers are beyond the scope of this work, we compare the mlsvm-AMG with the LibLinear [29] that is significantly faster than both DC-SVM and LibSVM. We note that linear SVM solvers can also be used as the refinement in multilevel frameworks. However, in practice, we do not observe a need for this because nonlinear SVM refinement is already fast enough in our multilevel framework.

The results for performance measures and computational time are presented in Tables 2.11, and 2.12. The mlsvm-AMG produces higher G-means on SUSY, HIGGS, and 8 (out of 10) of classes in the MINST8M datasets. On classes 8, 5, and 9 of MNIST8M we have an improvement of 24%, 6% and 5%, respectively. On the average, the G-mean for all larger datasets are 5% higher for

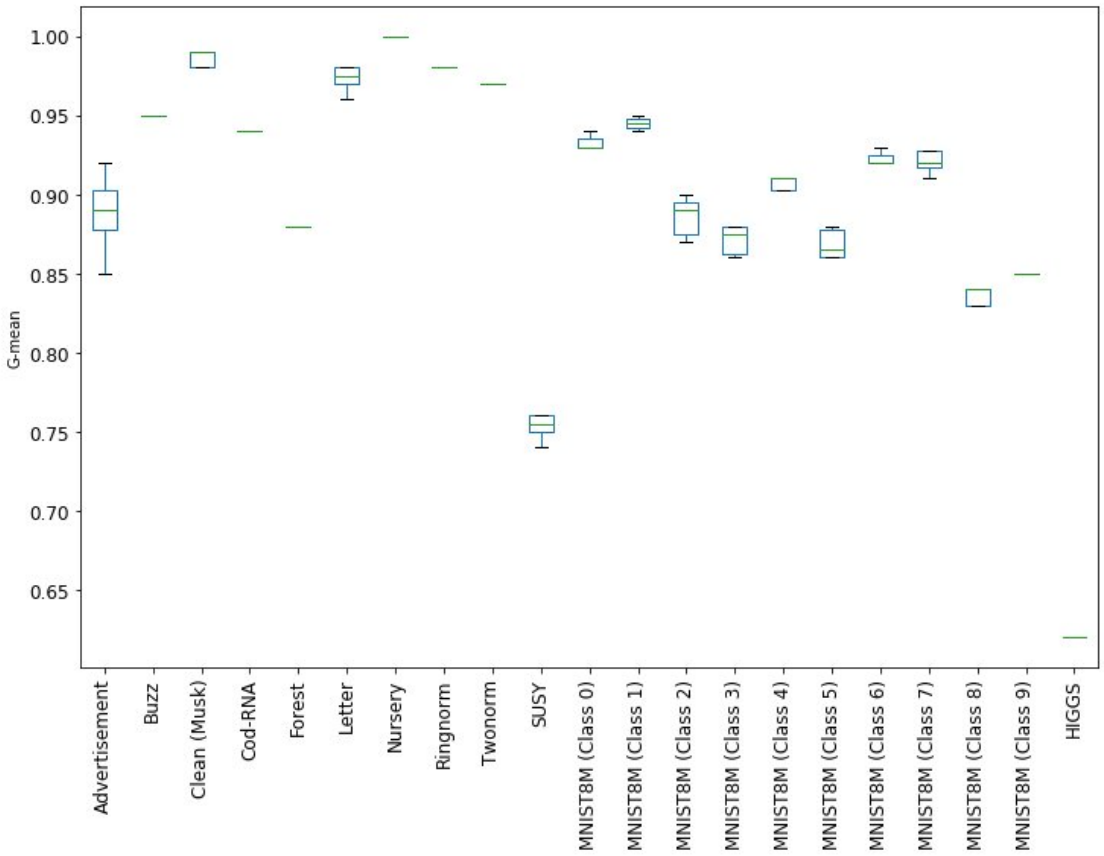


Figure 2.3: Each boxplot (horizontal axis) shows variability of the G-mean (vertical axis). A small standard deviation is observed in all cases.

mlsvm-AMG in comparison to LibLinear. The mlsvm-AMG is faster than LibLinear on SUSY and HIGGS datasets and slower on MNIST8M dataset. However, this slowness is eliminated if linear SVM solver is used in the refinement. The results for seven classes of Forest dataset are presented in Table 2.13. The statistics of G-mean variability is presented in Figure 2.3 which confirms the robustness of the proposed method.

In many cases, we observe a faster than linear behavior of our framework. An example is shown in Figure 2.4. When we use only a part of the dataset SUSY for training the model (horizontal axis), the computational time (vertical axis) is increasing slower than linearly. Such behavior can be observed when the number of support vectors is relatively small which is one of the main assumptions of this method. Another example with a larger number of features for MNIST8M is presented in Figure 2.5.

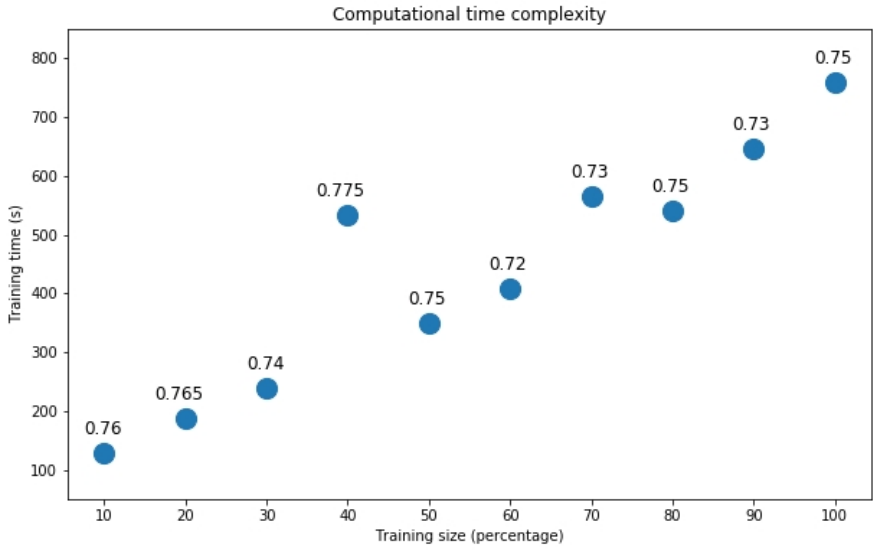


Figure 2.4: Scalability of mlsvm-AMG on growing training set of SUSY dataset. Each point represents the training time (vertical axis) when a certain part of the full training set (horizontal axis) is used. The numbers above points represent the G-mean performance measure. For example, if we use 60% of the training set to train the model, the running time is about 400 seconds, and the G-mean is 0.72.

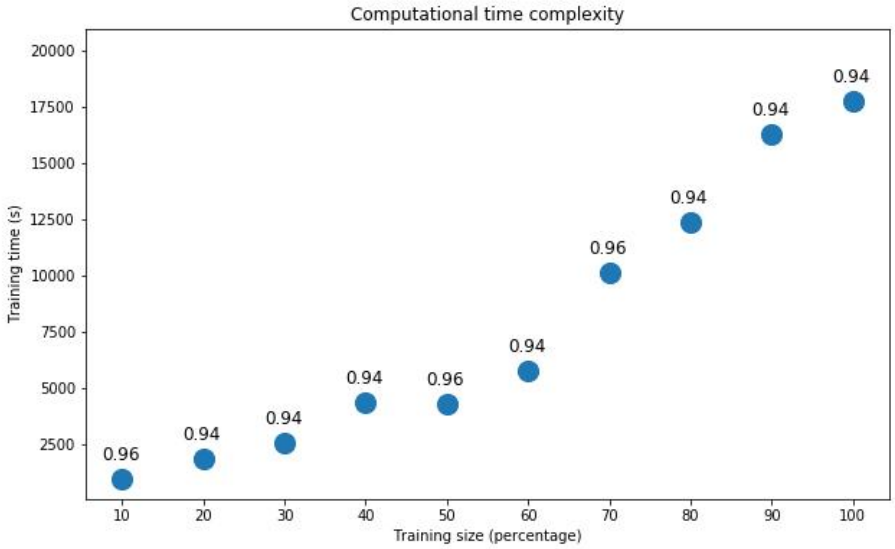


Figure 2.5: Scalability of mlsvm-AMG on growing training set of MNIST8M dataset using class 1. The 5M data points from the MNIST8M dataset are sampled to create a similar size comparison with SUSY dataset for a larger number of features.

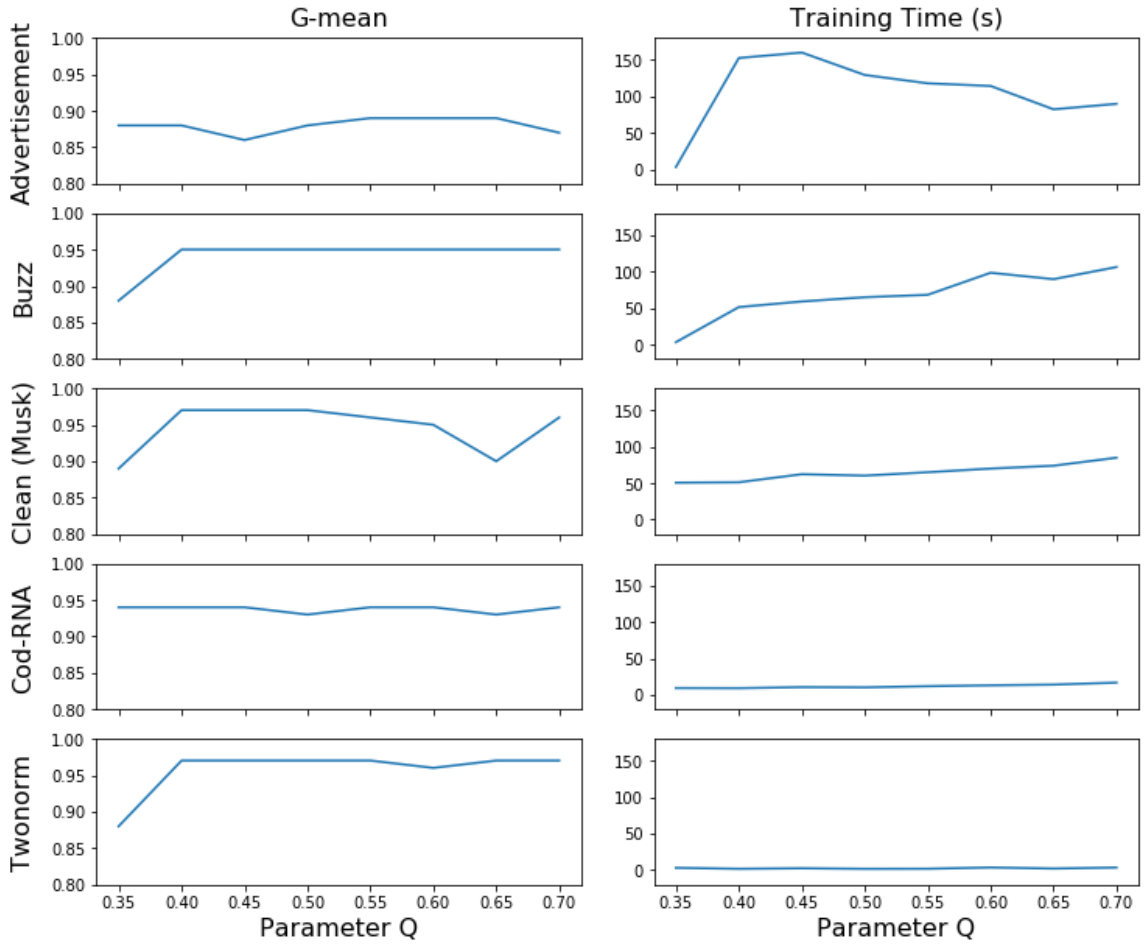


Figure 2.6: The mlsvm-AMG using parameter $Q \in [0.35, ..., 0.7]$ generates the best results on the benchmark data sets.

The robustness of parameter Q (see Alg. 4, line 8), which determines the size of the coarse level is also an important question. In AMG and AMG-inspired algorithms, a typical setting is to make $Q \in [0.4, \ldots, 0.6]$ unless a special reason for a faster aggregation allows more aggressive compression of the problem without significant loss in the solution quality. Here we observe that a similar range for Q is generally robust (see Fig. 2.6). In general, in multilevel learning, over-compression with too small Q is not recommended unless we know that a data is easily separable (or well clustered).

Finally, we present the computational time in terms of the amount of work per unit for all datasets in Table 2.14. In "$\frac{\mu s}{point}$" and "$\frac{\mu s}{value}$" columns, we present the computational time in

microseconds per data point and one feature value in data point, respectively.

Table 2.10: Larger benchmark data sets.

Dataset	ϵ	n_f	$\lvert\mathcal{J}\rvert$	$\lvert\mathbf{C}^+\rvert$	$\lvert\mathbf{C}^-\rvert$
SUSY	0.54	18	5000000	2287827	2712173
MNIST8M (Class 0)	0.90	784	4050003	399803	3650200
MNIST8M (Class 1)	0.89	784	4050003	455085	3594918
MNIST8M (Class 2)	0.90	784	4050003	402165	3647838
MNIST8M (Class 3)	0.90	784	4050003	413843	3636160
MNIST8M (Class 4)	0.90	784	4050003	394335	3655668
MNIST8M (Class 5)	0.91	784	4050003	365918	3684085
MNIST8M (Class 7)	0.90	784	4050003	399465	3650538
MNIST8M (Class 6)	0.90	784	4050003	422888	3627115
MNIST8M (Class 8)	0.90	784	4050003	394943	3655060
MNIST8M (Class 9)	0.90	784	4050003	401558	3648445
HIGGS	0.53	28	11000000	5170877	5829123

Table 2.11: Performance measures for single level WSVM (LibLinear), DC-SVM/LibSVM and mlsvm-AMG on larger benchmark data sets using partitioning and FF, FS validation techniques.

	LibLinear				DC-SVM and LibSVM	mlsvm-AMG			
Dataset	ACC	SN	SP	G-mean		ACC	SN	SP	G-mean
SUSY	0.69	0.61	0.76	0.68		0.75	0.71	0.78	**0.74**
MNIST8M (Class 0)	0.98	0.90	0.99	**0.95**		0.94	0.93	0.94	**0.95**
MNIST8M (Class 1)	0.98	0.93	0.99	**0.96**		0.94	0.95	0.94	0.95
MNIST8M (Class 2)	0.97	0.77	0.99	0.87		0.91	0.87	0.91	**0.89**
MNIST8M (Class 3)	0.96	0.70	0.98	0.83		0.88	0.86	0.89	**0.88**
MNIST8M (Class 4)	0.97	0.81	0.99	0.90	Stopped or failed after 3 days	0.91	0.92	0.91	**0.91**
MNIST8M (Class 5)	0.96	0.64	0.99	0.80	without any result	0.84	0.91	0.83	**0.86**
MNIST8M (Class 6)	0.98	0.86	0.99	0.92		0.94	0.90	0.94	**0.93**
MNIST8M (Class 7)	0.98	0.85	0.99	0.91		0.93	0.90	0.93	**0.92**
MNIST8M (Class 8)	0.92	0.36	0.98	0.60		0.82	0.87	0.81	**0.84**
MNIST8M (Class 9)	0.94	0.64	0.97	0.80		0.81	0.91	0.79	**0.85**
HIGGS	0.54	0.55	0.54	0.54		0.62	0.61	0.63	**0.62**

2.4.3.2 Disaggregation with neighbors

When the computational resources allow and the k-NN graph is not extremely dense, one may add neighboring nodes to the corresponding disaggregated support vector nodes. While this adds flexibility to train the models (with more added data points), in most cases, it is an unnecessary step that increases the running time. The Forest, Clean, and Letter are the three data sets which demonstrate an improvement on classification quality by adding the distance-1 neighbors. The results for including the distant neighbors for the Letter data set experimenting with multiple coarse

Table 2.12: Computational time in seconds for single level WSVM (LibLinear), DC-SVM/LibSVM and mlsvm-AMG on larger benchmark data sets

Dataset	LibLinear	DC-SVM and LibSVM	mlsvm-AMG
SUSY	1300		**1116**
MNIST8M (Class 0)	**1876**		11411
MNIST8M (Class 1)	**859**		15441
MNIST8M (Class 2)	**1840**		17398
MNIST8M (Class 3)	**2362**		10547
MNIST8M (Class 4)	**1448**	Stopped or failed after 3 days	13014
MNIST8M (Class 5)	**2360**	without any result	13353
MNIST8M (Class 6)	**1628**		10092
MNIST8M (Class 7)	**1747**		16789
MNIST8M (Class 8)	**2626**		17581
MNIST8M (Class 9)	**1650**		21611
HIGGS	4406		**3283**

Table 2.13: Performance measures and running time (in seconds) for all classes of Forest dataset using full mlsvm-AMG.

Dataset	ACC	SN	SP	G-mean	PPV	Time
Class 1	0.73	0.79	0.69	0.74	0.60	926
Class 2	0.70	0.78	0.62	0.70	0.67	215
Class 3	0.90	0.99	0.90	0.94	0.39	1496
Class 4	0.92	1.00	0.92	0.96	0.99	3231
Class 5	0.80	0.96	0.80	0.88	0.07	948
Class 6	0.86	0.95	0.95	0.90	0.17	2972
Class 7	0.91	0.87	0.91	0.89	0.28	2269

neighbor size reveal the largest improvement for $r = 1$ (see Figure 2.7).

2.4.3.3 Using partitioning in the refinement

When the training set becomes too big during the refinement (at any level), a partitioning is used to accelerate the performance. In Table 2.15, we compare the classification quality (G-mean), the size of training data, and the computational time. In columns "Partitioned" ("Full"), we show these three factors when (no) partitioning is applied. When no partitioning is used, we train the model with the whole training data at each level. The partitioning starts when the size of training data is 5000 points. Typically, at the very coarse levels the size of training data is small, so in the experiment demonstrated in Table 2.15, we show the numbers beginning level 5, the last level at which the partitioning was not applied. The results in this and many other similar experiments show significant improvement in computational time when the training data is partitioned with very

Table 2.14: Complexity Analysis

Dataset	$\lvert\mathcal{J}\rvert$	n_f	$\lvert\mathcal{J}\rvert \cdot n_f$	$\frac{\mu s}{point}$	$\frac{\mu s}{value}$
Nursery	13K	19	246.2K	232	12
Twonorm	7.4K	20	148K	405	20
Ringnorm	7.4K	20	148K	541	27
Letter	20K	16	320K	100	6
Cod-rna	59.5K	8	476.3K	100	13
Clean (Musk)	6.6K	166	1.1M	909	6
Advertisement	3.3K	1558	5.1M	31107	20
Buzz	140.7K	77	10.8M	1628	21
Forest	581K	54	31.4M	207	4
Susy	5M	18	90M	223	12
Higgs	11M	28	308M	298	11
mnist 4M	4.1M	784	3.2G	6673	9

minor loss in G-mean. The best level in the hierarchy is considered as the final level which is selected based on G-mean. Therefore, with no partitioning we obtain G-mean 0.79 and with partitioning it is 0.77 which are not significantly different results.

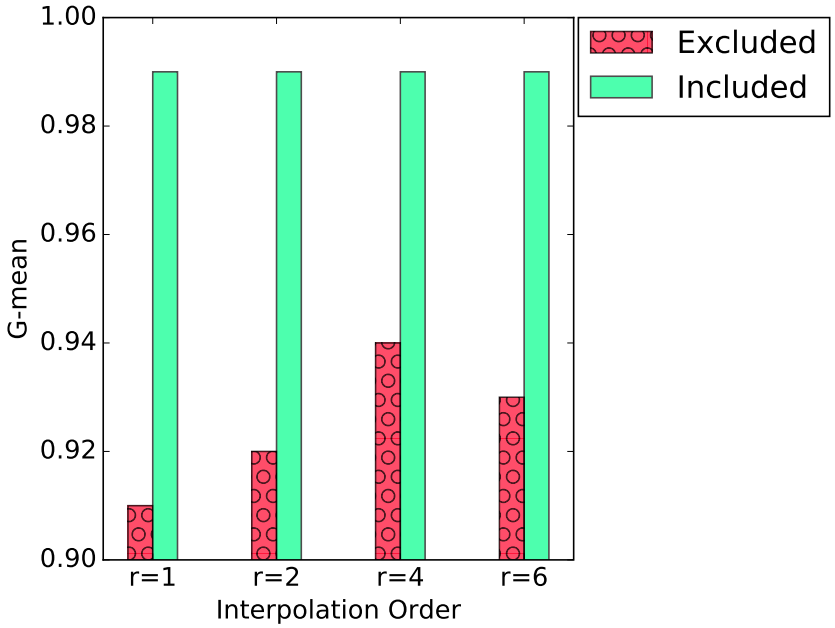
Table 2.15: The G-mean, training set size, and computational time are reported for levels 1-5 of Forest data set for Class 5. The partitioning is started with 5000 points.

	G-mean		Size of training set		Computational time	
Level	Full	Partitioned	Full	Partitioned	Full	Partitioned
5	0.69	0.69	4387	4387	373	373
4	0.68	0.72	18307	18673	1624	1262
3	0.79	0.77	47588	43977	6607	1528
2	0.79	0.72	95511	33763	17609	917
1	0.72	0.74	138018	24782	27033	576

2.4.3.4 Comparison with fast Ensemble SVM

A typical way to estimate the correctness of a multilevel solver is to compare its performance to those that use the local refinement techniques only. The EnsembleSVM [23] is a free software package containing efficient routines to perform ensemble learning with SVM models. The implementation exhibits very fast performance avoiding duplicate storage and evaluation of support vectors which are shared between constituent models. In fact, it is similar to our refinement and can potentially replace it in the multilevel framework. The comparison of our method with Ensem-

Figure 2.7: Effect of considering distance-1 disaggregation during the refinement phase on the G-mean for the Letter data set



bleSVM is presented in Table 2.16. While the running time is incomparable because of the obvious reasons (the complexity of EnsembleSVM is comparable to that of our last refinement only), the quality of our solver is significantly higher.

Table 2.16: Ensemble SVM on benchmark data sets

	Ensemble SVM				mlsvm-AMG			
Dataset	ACC	SN	SP	G-mean	ACC	SN	SP	G-mean
Advertisement	0.52	0.41	0.95	0.57	0.95	0.85	0.96	**0.91**
Buzz	0.65	0.36	0.99	0.59	0.94	0.95	0.94	**0.95**
Clean (Musk)	0.85	0.00	0.85	0.00	0.99	0.99	0.99	**0.99**
Cod-RNA	0.90	0.82	0.94	0.88	0.93	0.97	0.91	**0.94**
Forest	0.98	0.32	0.99	0.57	0.77	0.96	0.80	**0.88**
Letter	0.97	0.75	0.98	0.86	0.98	0.99	0.98	**0.99**
Nursery	0.68	1.00	0.68	0.82	1.00	1.00	1.00	**1.00**
Ringnorm	0.68	0.61	1.00	0.78	0.98	0.98	0.98	**0.98**
Twonorm	0.75	0.89	0.76	0.81	0.98	0.98	0.97	**0.98**

2.5 Conclusions

In this chapter we introduced novel multilevel frameworks for nonlinear support vector machines. and discussed the details of several techniques for engineering multilevel frameworks that lead to a good trade-off between quality and running time. We ran a variety of experiments to compare several state-of-the-art SVM libraries and our frameworks on the classification quality and computation performance.

The computation time of the proposed multilevel frameworks exhibits a significant improvement compared to the state-of-the-art SVM libraries with comparable or improved classification quality. For large data sets with more than 100,000 and up to millions of data points, we observed an improvement of computational time within an order of magnitude in comparison to DC-SVM and more two orders of magnitude in comparison to LibSVM. The improvement for larger datasets is even more significant. The code for mlsvm-AMG is available at https://github.com/esadr/mlsvm.

There exist several attractive directions for the future research. One of them is to study in-depth why generating models at the coarse scales eliminates the effects of over- and under-fitting, a phenomena that we observed in many data sets. Another research avenue is to develop an uncoarsening scheme which chooses an appropriate kernel type at the coarse levels (where the training set size is relatively small) and continues with the best choice to fine levels. Indeed, if we successfully fit the parameters of kernel at the coarse levels, why not to try to choose the kernel type as well? Another direction could be separating constraints from the objective and solving them in a combined multigrid framework similar to [82].

Appendix A: Summary of parameters

In Table 2.17, we mention recommended ranges of parameters for multilevel (W)SVM frameworks that we tested in our experiments.

Parameter	Reference	Description
r	Sec. 2.3.2	Recommended range $[1, .., 4]$. Almost all results were produced with $r = 1$ except Cod-RNA ($r = 2$) and SUSY ($r = 4$).
θ	Sec. 2.3.4.2	Recommended range $[0.001, .., 0.05]$. Almost all results were produced with $\theta = 0.05$ except Letter ($\theta = 0.005$) and Musk ($\theta = 0.001$) that produced slightly better results with less aggressive filtering.
d	Eq. (2.15)	Euclidean distance was used in all experiments.
Q_t	Alg. 3	Our simple single processor hardware allowed to start partitioning at 5000 data points. However, Q_t in a range $[3000, .., 5000]$ produced similar results.
η	Alg. 4	In all experiments $\eta = 2$.
K	Alg. 3	To preserve fast partitioning and training by parts, we used $K = \lfloor \lvert\mathcal{J}_{(i)}\rvert/1000 \rceil$ for all levels i. No difference when changing this value was observed.
M^+ and M^-	Alg. 1	In all experiments $M^+ = M^- = 300$.
$\lvert\mathcal{J}_{(\rho)}\rvert$	Alg. 1	The size of the coarsest level was always $\lvert\mathcal{J}_{(\rho)}\rvert = 500$ to maintain fast performance of model selection at the coarsest level.
Q	coarsen-IIS and coarsen-AMG (Alg. 4)	In all experiments $Q = 0.5$. No significant difference was observed for $Q \in [0.4, ..., 0.6]$, see Fig. 2.6.
C and γ	NUD in Alg. 3	The NUD model selection algorithm starts parameter search in range of $2^{-10} < C < 2^{10}$ and $2^{-10} < \gamma < 2^{10}$ for the RBF kernel using the standard 9-13 scheme described in [44].

Table 2.17: Recommended parameter values.

Appendix B: Standard deviation for **mlsvm-AMG**

Table 2.18: Standard deviations of the performance measures for mlsvm-AMG

Dataset	ACC	SN	SP	G-mean
Advertisement	0.01	0.04	0.01	0.02
Buzz	0.00	0.01	0.01	0.00
Clean (Musk)	0.00	0.00	0.00	0.00
Cod-RNA	0.00	0.00	0.00	0.00
Forest	0.01	0.01	0.01	0.00
Letter	0.00	0.01	0.00	0.00
Nursery	0.00	0.00	0.00	0.00
Ringnorm	0.00	0.00	0.00	0.00
Twonorm	0.00	0.00	0.01	0.00
SUSY	0.01	0.04	0.04	0.01
MNIST8M (Class 0)	0.01	0.00	0.01	0.01
MNIST8M (Class 1)	0.00	0.00	0.00	0.00
MNIST8M (Class 2)	0.02	0.02	0.02	0.02
MNIST8M (Class 3)	0.02	0.02	0.02	0.00
MNIST8M (Class 4)	0.02	0.01	0.02	0.01
MNIST8M (Class 5)	0.03	0.03	0.03	0.02
MNIST8M (Class 7)	0.02	0.02	0.02	0.02
MNIST8M (Class 6)	0.02	0.02	0.02	0.02
MNIST8M (Class 8)	0.03	0.03	0.04	0.01
MNIST8M (Class 9)	0.01	0.02	0.02	0.00
HIGGS	0.00	0.02	0.02	0.00

Chapter 3

AML-SVM: Adaptive Multilevel Learning with Support Vector Machines

3.1 Introduction

Support vector machine (SVM) is a widely used family of classification methods that leverage the principle of separating hyperplane. Technically, this is achieved by solving underlying regularized optimization model adjusting which can provide highly accurate and interpretable classification. Training linear SVM is very fast and can scale to millions of data points and features without using significant high-performance computing (HPC) resources. For problems that are not linearly separable, the nonlinear SVM uses the kernel trick by implicitly projecting the data into the higher-dimensional space to separate it by a hyperplane. Nonlinear SVM usually reaches higher prediction quality on complex datasets. However, it comes with a price tag of being not scalable in comparison to its linear version.

Solving the Lagrangian dual problem is typically the way to cope with regularized nonlinear SVM models with the underlying convex quadratic programming (QP) problem. In a number of libraries (such as LibSVM [17]) multiple methods have been implemented for solving both primal and dual problems. The complexity of the convex QP solvers for nonlinear SVM often scales between

$O(n^2 f)$ to $O(n^3 f)$ [36] , where n is the number of data points, and f is the number of features. Therefore, as n increases, the running time of the solver also increases, which hinders the usage of the nonlinear SVM for massive data sets.

For nonlinear SVM, optimal parameters (for kernel and regularization) are often required to achieve a more accurate decision boundary (hyperplane) for complex data sets. The parameter fitting (also known as model selection) often follows iterative schemes that (such as [123]) to train a better model. Parameter fitting is one of the most important factors that contributes to the slowness of nonlinear SVM training [104].

As the size of many datasets continues to grow due to the advancements in technologies such as high-throughput sequencing and the Internet of Things (IoT) [46], more scalable machine learning algorithms are required. Therefore, while a nonlinear SVM is fast on small datasets and can provide highly accurate prediction, more research is required to develop scalable nonlinear SVM solvers for massive datasets.

Our previous framework, Multilevel SVM (MLSVM), is a library that scales to millions of data points and exhibits up to two orders of magnitude faster running time compared to LIBSVM on reported benchmark datasets [86]. The MLSVM leverages multilevel algorithms that are inspired by the algebraic multigrid and some of its restricted versions [11]. These algorithms are known to be successful in accelerating computational optimization tasks without any loss in the optimization quality. Examples include hypergraph partitioning [99], image segmentation [98], and clustering [26]. Such multilevel methods have two essential phases, namely, coarsening and uncoarsening [11]. The coarsening phase gradually reduces the original problem size and generates a chain of smaller problems that approximate the original one. This is done by constructing a (possibly fuzzy) hierarchy of aggregated data points. The uncoarsening phase starts from the the smallest generated problem and gradually uses generated approximations to refine the solution of the original problem. During the uncoarsening phase, a refinement leverages a solution inherited from the previous coarser problem and is performed using a local processing solver to avoid any heavy computation.

3.1.1 Multilevel SVM

The multilevel SVM [87, 80, 86] is an approach to accelerate traditional SVM model solvers which also often improves the model quality as it exploits the geometry of data before starting the training. The key principles of multilevel SVM are summarized as follows.

1) Learning the separating hyperplane occurs within a small proximity to support vectors. The number of support vectors is typically much smaller than the original data size. Thus, after gradually aggregating the data points and training the coarsest SVM model, the data points that participate in the training at the next finer levels (i.e., during the uncoarsening) are only those that have been found within a limited distance neighborhoods of the aggregate centers. In other words, the multilevel SVM does not use the data points that are too far from the hyperplane in the training. The key idea here is to a) inherit the support vectors from the coarse level solution, b) disaggregate them into finer data points, and c) filter out some of them if their number is too big for the available computational resources.

2) Perform computationally expensive parts of the training only at the coarse levels. This is a principle of all multilevel algorithms and multiscale optimization strategies. The coarse levels that contain much smaller amount of data can be solved with computationally more expensive but higher quality algorithms. One of the major problems in nonlinear SVM training is the model selection which in most cases is essentially a process of (smart) try-and-train strategy with different sets of hyperparameters. In multilevel SVM, the model selection is applied only at the coarse levels.

3) The best final model is not necessarily the finest one. The multilevel SVM produces models at all levels of coarseness. Similar to many other methods the finest model can potentially suffer from overfitting and the coarsest one (the most aggregated) from underfitting. Multiple models of different coarseness provide with an opportunity to make a choice to avoid these two major problems and choose a model in between finest and coarsest levels.

We discuss several other principles in [86]. Among them is the one that discusses the way multilevel SVM copes with imbalanced data. While the MLSVM library [86] provided fast runtime and highly accurate predictions on massive datasets, we observed several unexpected results on benchmark datasets that motivated us to extend that work and introduce the Adaptive MLSVM in this chapter .

In multilevel algorithms [83, 11], it is expected to observe that the quality of optimized objective is improved at each next finer level during the uncoarsening. As such, we expected to see a non-decreasing quality improvement across the uncoarsening. However, on some most difficult benchmarks, we observed that while the trained coarse models have exhibited reasonable to optimal prediction quality, some of the middle levels experienced a significant drop in the quality. In some cases, these middle level drops were continuing to the fine levels and in some were gradually improved

back to the high quality results. Although the observed quality drops were rare, they often led to accumulating and increasing problems during the uncoarsening. Indeed, low quality support vectors may potentially be disaggregated into the data points that are even more distant to the optimal hyperplane which gives too much freedom to the optimizer. Other stages in the MLSVM pipeline such as the filtering of data points after too aggressive disaggregation were also affected.

Our Contribution In this chapter , we propose the Adaptive Multilevel SVM, a successful approach to detect the problem of inconsistent learning quality during the uncoarsening and efficiently mitigate it. At each level of the multilevel hierarchy, we detect the problem of quality decrease by validating the model using the finest level data. In the adaptive multilevel SVM learning framework, we adjust the training data by filling the training data gap with the misclassified validation data points, retraining the model and improving the quality with a new set of support vectors.

Our exhaustive experimental results on the benchmark datasets demonstrate the proposed method recovers the multilevel framework from such quality drops and achieves higher quality in comparison to the non-adaptive multilevel SVM as well as other state of the art solvers. In addition, we speed up even more the runtime compared to non-adaptive multilevel SVM and reduce the prediction quality variance. Moreover, in the new version of the MLSVM library we implement the multi-thread support for parameter fitting to speed up the model training at each level. Our implementation is open-source and available at `http://github.com/esadr/amlsvm`.

3.2 Related Works

Improving the performance of SVM has been widely studied with a common goal of improving the training time without a significant decrease in prediction accuracy such as recent researches [94, 96, 121]. The complexity of nonlinear SVM on massive datasets is a potential target to develop scalable SVMs.

A challenge with solving the quadratic programming problem for the kernel SVM is the kernel matrix of size n^2, where n is the number of training data points. The matrix requires large memory, which is not feasible for massive datasets. The main categories of accelerated solvers follow one of the following approaches: a) target the solver performance directly, b) partition the original data, or c) considering alternative representation for the training data (including multilevel approach), and their combinations. Some of the improvements also rely on the advancement of software infrastructure, hardware, and distributed frameworks.

Solver performance: A Sequential Minimal Optimization (SMO)[75, 15] has been used to solve the underlying quadratic programming problem. SMO decomposes the restricted quadratic problem into a series of smallest possible subproblems and solved them analytically. A fast GPU implementation based on matrix matrix multiplication instead of matrix vector multiplication has been developed in [116]. The ThunderSVM [121] directly works on improving the performance of the solver using multiple cores and GPUs.

Partition original training data: Instead of solving the QP with a large number of data points, a set of smaller problems can be generated and solved independently. This reduces the running time of training and memory utilization required to store the kernel matrix. A drawback to this approach is the quality of partitioning the original data and the quality of the approach to combine all the trained models to drive a final model. Cascade SVM[36] partitions the training data evenly and trains a SVM model for each of partition. It only support linear SVM. An intrusion detection system is developed based on exploiting hierarchical clustering for reducing the number of training data points before training a SVM model [41] on millions of points. Ensemble SVM [23] samples the data and uses an ensemble of models for the final prediction. DC-SVM[42] used adaptive clustering for dividing the training data and relied on coordinate descent method to converge to a solution from multiple models. The latter models supports nonlinear SVM. A disadvantage for these approaches is relying on the partitioning or clustering that is sensitive to the order of the training

data, number of clusters. Assignment of a point to a partition or cluster is strict, which limits the points to move between clusters or participate in multiple clusters.

Graph representation of training data: There are two schemas for partitioning the data. The earlier approaches rely on the original data in the feature space, while newer approaches such as[80, 86, 94] rely on a graph representation of data. The graph representation of the training data is constructed as a preprocessing step for partitioning which is explained in detail in section 3.3.2. The graph representation provides the opportunity to leverage the multilevel paradigm that has been used successfully in a wide range of problems. We briefly mention some of the multilevel research related to graphs such as graph partitioning [93], graph clustering [27], and image segmentation [35]. The advantage of using a graph and multilevel paradigm such as Algebraic Multigrid (AMG) is to exploit more relaxed and less strict assignments of the data points to smaller aggregates compare to clustering, which has strict assignments and larger cluster sizes.

The position of AML-SVM: Our proposed framework (AML-SVM) has a robust coarsening, which allows partial participation of points in multiple aggregates through gradual assignment and relaxation steps. Our results demonstrate that a small training data at the coarsest level can be used to train a model with high accuracy. AML-SVM can directly leverage the performance improvements introduced by advanced solvers with multi-core or GPU support to achieve even faster runtime without any change in the coarsening or refinement process. The uncoarsening (refinement) phase as a general step in multilevel methods has the potential for improvement of initial solutions and carrying essential information such as support vectors, and parameters to other levels. The improvements proposed in this chapter reduce the variance between the quality of models at various levels and reduce the sensitivity of the framework to configuration parameters. Furthermore, these improvements, along with multiple core support for faster parameter fitting, can be used for ensemble models that we have not explored.

3.3 Preliminaries

Consider a set $\mathcal{J}$ of input samples that contains n data points denoted by x_i, where $x_i \in \mathbb{R}^d$, $1 \leq i \leq n$. Each data point x_i has a corresponding label $y_i \in \{-1, 1\}$. The SVM as a binary classification finds the largest margin hyperplane that separates the two classes of labeled data. The solution to the following optimization problem produces the largest margin hyperplane, which is defined by w, and b.

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y_i(w^T\phi(x_i) + b) \geq 1 - \xi_i, \quad i = 1, \ldots, n \\ & \xi_i \geq 0, \quad i = 1, \ldots, n. \end{aligned} \tag{3.1}$$

The misclassification is penalized using slack variables $\{\xi_i\}_{i=1}^{n}$. The parameter $C > 0$ controls the magnitude of penalty for miscalssified data points. The primal formulation in (3.1) is known as the *soft margin* SVM [122].

The SVM takes advantage of kernel $\phi : \mathbb{R}^d \to \mathbb{R}^p$ $(d \leq p)$ to map data points to higher-dimensional space. The kernel measures the similarity for pairs of points x_i and x_j. The Gaussian kernel (RBF), $\exp(-\gamma||x_i - x_j||^2)$, which is known to be generally reliable [113], is the default kernel in this chapter .

For imbalanced datasets, different weights can be assigned to classes using the *weighted* SVM (WSVM) to controllably penalize points in the minority class C^+ (e.g., rare events). The set of slack variables is split into two disjoint sets $\{\xi_i^+\}_{i=1}^{n^+}$, and $\{\xi_i^-\}_{i=1}^{n^-}$, respectively.

In WSVM, the objective of (3.1) is changed into

$$\text{minimize} \quad \frac{1}{2}\|w\|^2 + C\big(W^+\sum_{i=1}^{n^+}\xi_i^+ + W^-\sum_{j=1}^{n^-}\xi_j^-\big). \tag{3.2}$$

The imbalanced data sets contain significanlty fewer positively labeled data points. Hence, the data points in the *minority* class is denoted by $\mathcal{J}^+$, where size of minority class is $n^+ = |\mathcal{J}^+|$. The rest of the points belongs to the *majority* class which is denoted by $\mathcal{J}^-$, where $n^- = |\mathcal{J}^-|$, and $\mathcal{J} = \mathcal{J}^+ \cup \mathcal{J}^-$. We assign the class weight as the inverse of the total number of points in that class. For instance, the weight for minority class is $W^+ = \frac{1}{n^+}$

In the our multilevel framework, one of the basic steps is aggregating data points. The aggregation can be performed either on full points or their fractions (i.e., a data point can be split into fractions, and different fractions can contribute themselves to different aggregates). Therefore, a coarse level data point normally contains several finer points or their fractions. Because of this, a data point can be more or less representative in comparison to other data points. We define and use the volume property for each point and calculate it as number of partial or fully participated original points in that aggregate.

We denote v_i as volume for point x_i which represent the internal importance of point i among points in the same class. Moreover, we consider the class importance for the final weight of each data point. The calculation for both classes are the same, hence, we only present the positive class. We calculate the class weights based on sum of the volumes in the class, i.e.,

$$W^+ = \frac{1}{\sum\limits_{i \in \mathcal{J}^+} v_i}. \tag{3.3}$$

We calculate the final weight for each point as product of its volume weight times class weight

$$\forall i \in \mathcal{J}^{+(-)} \quad W_i = W^{+(-)} \frac{v_i}{\sum\limits_{j \in \mathcal{J}^{+(-)}} v_j}. \tag{3.4}$$

The instance weight based on volume and class, helps to improve prediction quality for the smaller class.

Solving the Lagrangian dual problem using kernel functions $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(x_i)^T \phi(x_j)$ produces a reliable convergence which is faster than methods for primal formulations (3.1) and (3.2). Therefore, we use the sequential minimal optimization solver implemented in LIBSVM library [17] and RBF kernel.

The parameter fitting is an iterative method to find the optimal parameters for a data set. For each dataset optimal misclassification penalty C, and kernel parameter (γ) are required. We use the adapted nested uniform design (NUD) model selection [44] as parameter fitting in the proposed framework. It finds the close-to-optimal parameter set in an iterative nested manner. The optimal solution is calculated in terms of maximizing the required performance measure (such as accuracy and G-mean). Although we study binary classification problems, it can easily be extended

to the multi-class classification. For example, the multi-class problem can be converted into multiple independent binary classification that can be processed independently in parallel.

3.3.1 Two-level problem

To describe the coarsening, uncoarsening and refinement algorithms, we introduce the two-level problem notation that can be extended into a full multilevel hierarchy (see Figure 3.1). In Algorithm 7, we will use subscript $(\cdot)_f$ and $(\cdot)_c$ to represent fine and coarse variables, respectively. For example, the data points of two consecutive levels, namely, fine and coarse, will be denoted by $\mathcal{J}_f$, and $\mathcal{J}_c$, respectively. The sets of fine and coarse support vectors are denoted by sv_f, and sv_c, respectively. We will also use a subscript in the parentheses to indicate the level number in the hierarchy where appropriate. For example, $\mathcal{J}_{(i)}$ will denote the set of data points at level i.

3.3.2 Proximity graphs

The original MLSVM framework [86] is based on the algebraic multigrid coarsening scheme for graphs (see coarsening in [90]) that creates a hierarchy of data proximity graphs. Initially, at the finest level, $\mathcal{J}$ is represented as two approximate k-nearest neighbor (kNN) graphs $G^{+}_{(0)} = (\mathbf{V}^{+}, E^{+})$, and $G^{-}_{(0)} = (\mathbf{V}^{-}, E^{-})$ for minority and majority classes, respectively, where each $x_i \in \mathbf{V}^{+(-)}$ corresponds to a node in $G^{+(-)}_{(0)}$.

A pair of nodes in $G^{+(-)}_{(0)}$ is connected with an edge that belongs to $E^{+(-)}$ if one of these nodes belongs to a set of k-nearest neighbors of another. Throughout the multilevel hierarchies, in two-level representation, the fine and coarse level graphs will be denoted by $G^{+(-)}_{f} = (\mathbf{V}^{+(-)}_{f}, E^{+(-)}_{f})$, and $G^{+(-)}_{c} = (\mathbf{V}^{+(-)}_{c}, E^{+(-)}_{c})$, respectively. For kNN, our experiments show $k = 10$, for all tested datasets, provides a good trade-off between computational cost of calculating kNN and average node degree for $G^{+(-)}$.

3.4 Adaptive Multilevel SVM

The multilevel support vector machine (MLSVM) [86] includes three main phases (see Figure 3.1), namely, (1) coarsening: gradual approximation of training set, (2) coarsest level solution: initial SVM training, and (3) uncoarsening: gradual support vector refinement at all levels. The coarsening phase creates a hierarchy of coarse training data $\mathcal{J}_{(i)}$, where i is the level number where $i = 1$ for the finest level and increases by one at each level of coarsening. The number of data points is decreasing at each level of coarsening, i.e., $|\mathcal{J}_{(i+1)}| < |\mathcal{J}_{(i)}|$. The coarsening continues until the aggregated training data reaches a certain threshold. The threshold depends on the available computational resources, which are capable of training a high-quality algorithm in a reasonable time. The threshold for the size of each class of data is denoted by M. At the coarsest level k, the runtime to train a SVM model on a small training set is fast since training size is limited to less than a threshold for each class. ($\mathcal{J}_{(k)} < 2 \times M$). The output of trained SVM with the parameter fitting includes $\mathsf{sv}_{(k)}$ and the optimal parameters $C_{(k)}, \gamma_{(k)}$. This output is inherited by the next finer level of the uncoarsening phase. At level l of the uncoarsening, we select the next action among the three choices based on $\mathcal{J}_{(l)}$ and $Q_{(l)}$, which are the size of training data and the quality of model on validation data at level l, respectively:

- Normal Refinement: The refinement continues by training new SVM model over training data based on decision boundary neighborhood at a coarser level.
- Early Stopping: We stop the refinement(uncoarsening) if the size of training data (both classes) is larger than threshold θ (line 7 in Algorithm 7).
- Detect and Recovery: If we detect a quality drop, we call the recovery algorithm described in Algorithm 8

The main parts of the framework are explained in Algorithms 5,6 and 7.

The algebraic multigrid (AMG) SVM multilevel framework is inspired by the AMG aggregation solvers for computational optimization problems such as [91, 59, 92]. More information is covered in [86, 85]. The AMG coarsening provides high quality coarsening. It aggregates both fine points and fractions of points, which is more flexible than strict (or matching based) coarsening techniques. The fine level solution is identified using the interpolation operator on the coarse aggregates which are the main part of the coarsening phase in the AMG [114]. The solution obtained

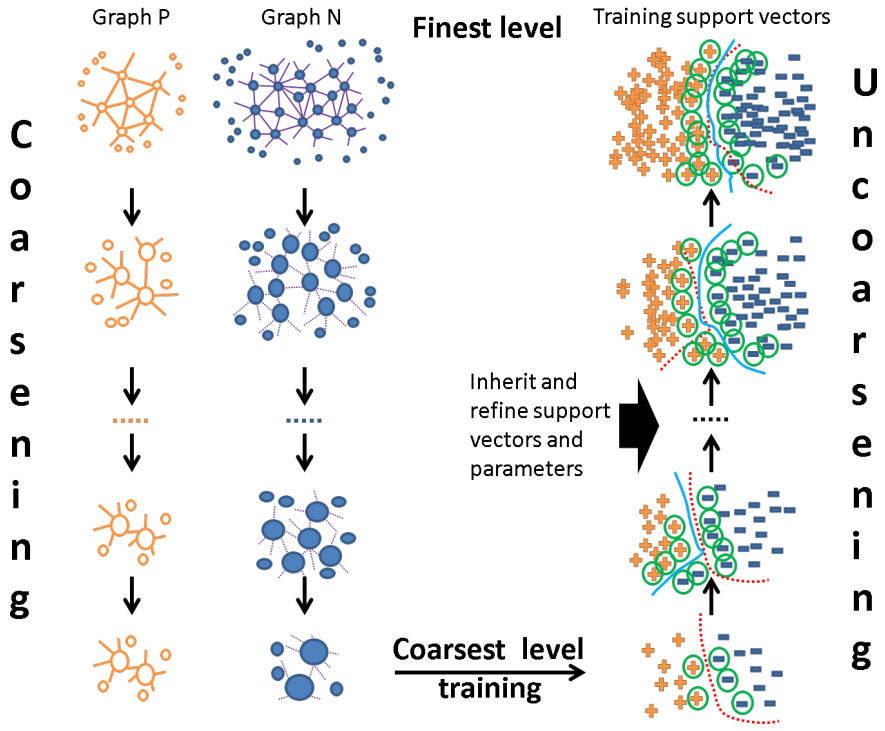


Figure 3.1: Adaptive multilevel learning framework scheme.

from the coarsest level is gradually projected back to the finer levels using the interpolation and refined locally.

3.4.1 Framework initialization

The AMG framework is initialized with $G_0^{+(-)}$ with the edge weights that represent the strength of connectivity between nodes to "simulate" the following interpolation scheme applied at the uncoarsening, in which strongly coupled nodes can interpolate solution to each other. In the classifier learning problems, this is expressed as a similarity measure between points. We define a distance function between nodes (or corresponding data points) as the inverse of the Euclidean distance in the feature space. For the completeness of this chapter we need to repeat the coarsening phase we defined in [86].

3.4.2 Coarsening Phase

In Algorithm 5, we demonstrate the process of obtaining the training set (aggregated points) for the next coarser level. The process for both class are the same, hence, only minority class is explained. The input to the coarsening algorithm includes the fine level graph G_f^+, and data points (training set) $\mathcal{J}_f^+$, for the minority class. The output includes the graph G_c^+ and data $\mathcal{J}_c^+$ at the coarser level for minority class. The same process applies to the majority class.

The process at level f starts with selecting seeds. Seeds are nodes that serve as centers of aggregates at level c. We introduce the notion of *volume* $v_i \in \mathbb{R}_+$ for all $i \in G_f^+$ that reflects the importance of each point during the coarsening phase. The volume is initialized to one for each node at the finest level. It increases during the coarsening with respect to the weighted sum of the volume of nodes that are aggregated together. (In some applications, so called anchor points can be used which can be reflected in the initial volumes.) After the coarse graph is constructed, the aggregated data points are calculated based on interpolation matrix P.

Seed selection starts with the future-volume ϑ_i computation which is a measure of how much an aggregate seeded by a node i may potentially grow at the next level c, and it is computed in linear time:

$$\vartheta_i = v_i + \sum_{j \in \Gamma_i \cap \mathbf{V}_f^+} v_j \cdot \frac{w_{ji}}{\sum\limits_{k \in \Gamma_j \cap \mathbf{V}_f^+} w_{jk}}, \tag{3.5}$$

where Γ_i is the neighborhood of node i in G_f^+. We start with selecting a dominating set of seed nodes $S \subset \mathbf{V}_f^+$ to initialize aggregates. Nodes that are not selected to S remain in F such that $\mathbf{V}_f^+ = F \cup S$. Initially, the set F is set to be $\mathbf{V}_f^+$, and $S = \emptyset$ since no seeds have been selected. After that, points with $\vartheta_i > \eta \cdot \overline{\vartheta}$, i.e., those that are exceptionally larger than the average future volume are transferred to S as the most "representative" points (line 3). Then, all points in F are accessed in the decreasing order of ϑ_i updating S iteratively (lines 6-11), namely, if with the current S, and F, for point $i \in F$, $\sum_{j \in S} w_{ij} / \sum_{j \in \mathbf{V}_f^+} w_{ij}$ is less than or equal to some threshold Q. The AMG interpolation matrix $P \in \mathbb{R}^{|G_f^+| \times |S|}$ is defined as

$$P_{ij} = \left\{ \begin{array}{cc} w_{ij} / \sum\limits_{k \in \Gamma_i} w_{ik} & \text{if } i \in F,\, j \in \Gamma_i \\ 1 & \text{if } i \in S,\, j = I(i) \\ 0 & \text{otherwise} \end{array} \right\}, \tag{3.6}$$

where $\Gamma_i = \{j \in S \mid ij \in E_f^+\}$ is the set of ith seed neighbors, and $I(i)$ denotes the index of a coarse point at level c that corresponds to the fine level aggregate seeded by $i \in S$.

The aggregated points and volumes at a coarser level are calculated using the matrix P. The edge between points $p = I(i)$ and $q = I(j)$ is assigned with weight

$$w_{pq} = \sum\nolimits_{k \neq l} P_{ki} \cdot w_{kl} \cdot P_{lj}. \tag{3.7}$$

The volume for the aggregate $I(i)$ in the coarse graph is computed by $\sum_j v_j P_{ji}$. i.e., the total volume of all points is preserved at all levels during the coarsening. The coarse point $q \in \mathbf{V}_c^+$ seeded by $i = I^{-1}(q) \in \mathbf{V}_f^+$ is represented by

$$\sum_{j \in \mathcal{A}_i} P_{j,q} \cdot j, \tag{3.8}$$

where $\mathcal{A}_i$ is a set of fine points in aggregate i. This set is extracted from the column of P that corresponds to aggregate i by considering rows j with non-zero values.

Algorithm 5 AMG coarsening for one class

$S \leftarrow \emptyset, \quad \mathbf{D}_f^+, \quad F \leftarrow \mathbf{V}_f^+$
Calculate using Eq. (3.5) $\forall i \in F$ ϑ_i, and the average $\bar{\vartheta}$
$S \leftarrow$ nodes with $\vartheta_i > \eta \cdot \bar{\vartheta}$
$F \leftarrow \mathbf{V}_f^+ \setminus S$
Recompute ϑ_i $\forall i \in F$
Sort F in descending order of ϑ
for $i \in F$
  if $(\sum_{j \in S} w_{ij} / \sum_{j \in \mathcal{J}_f^+} w_{ij}) \leq Q$
    move i from F to S
  end
end
Build interpolation matrix P according to Eq. (3.6)
Build coarse graph G_c^+ with edge weights using Eq. (3.7)
Define volumes of coarse data points
Compute coarse data points $\mathbf{D}_c^+$
return $(\mathbf{D}_c^+, G_c^+)$

The complete coarsening phase for both classes are demonstrated in Algorithm 6. The parameter M controls the size of each class at the coarsest level. In our experiments, the M is set to 300 which ensures training a SVM model using LIBSVM is fast.

Algorithm 6 Coarsening phase for both classes

if $|\mathbf{V}_f^+| \leq M$ & $|\mathbf{V}_f^-| \leq M$ **then** ▷ Solve exact problem
  $(\mathsf{sv}_f, C^+, C^-, \gamma) \leftarrow$ Train SVM model on $\mathcal{J}_f$ (+NUD)
else ▷ Start or continue to coarsen the problem
  if $|\mathbf{V}_f^+| \leq M$ **then**
    $\mathbf{D}_c^+ \leftarrow \mathbf{D}_f^+;\ G_c^+ \leftarrow G_f^+$
  else
    $(\mathbf{D}_c^+, G_c^+) \leftarrow$ coarse $(\mathbf{D}_f^+, G_f^+)$
  if $|\mathbf{V}_f^-| \leq M$ **then**
    $\mathbf{D}_c^- \leftarrow \mathbf{D}_f^-;\ G_c^- \leftarrow G_f^-$
  else
    $(\mathbf{D}_c^-, G_c^-) \leftarrow$ coarse $(\mathbf{D}_f^-, G_f^-)$

3.4.3 Uncoarsening

The primary goal of uncoarsening is to interpolate and refine the solution of coarse level c for the current fine level f. One of the advantages of multilevel framework is limiting the size of the projected information. We only inherit the optimal parameters C, γ and sv_c. The corresponding aggregates to sv_c are added to $\mathcal{J}_f$, namely, The training data at the finer level ($\mathcal{J}_f$) starts as an empty set, then for any aggregate p ($\mathcal{A}_p$) which is part of support vectors at the coarser level (sv_c), all points j in aggregates are added to the training data.

$$\mathcal{J}_f \leftarrow \emptyset; \quad \forall p \in \mathsf{sv}_c \quad \forall j \in \mathcal{A}_p \quad \mathcal{J}_f \leftarrow \mathcal{J}_f \cup j. \tag{3.9}$$

Algorithm 7 demonstrate the overall process for the refinement process. The $\tilde{C^+}, \tilde{C^-}, \tilde{\gamma}$ are the input parameters along with support vectors at coarser level sv_c. First, the training set is processed based on equation (3.9).

Algorithm 7 Uncoarsening (refinement) phase

Input: $(\mathsf{sv}_c, \tilde{C}^+, \tilde{C}^-, \tilde{\gamma})$ ▷ Solution from a coarser level
Create training set by Eq. (3.9)
While $(|\mathbf{D}^+| + |\mathbf{D}^-|) < \theta$ then
    Train SVM Models
    Evaluate performance qualities
    Call detect and recovery Algorithm
    Select best model based on G-mean, SN, and nSV
    ▷ Refine the solution
    $\{(\mathsf{sv}_f, C^+, C^-, \gamma)_i\}_{i=1}^k \leftarrow \text{refine}(\mathsf{sv}_f, \tilde{C}^+, \tilde{C}^-, \tilde{\gamma})$
return best model from k levels $\{(\mathsf{sv}_f, C^+, C^-, \gamma)_i\}_{i=1}^k$

3.5 Learning Approach

3.5.1 Imbalanced classification

In addition to the cost-sensitive and weighted models solved in the refinement, the multilevel learning framework can cope with the imbalanced data and mitigates its negative effect on the classification quality for the minority class. We make a balanced training set at the coarsest level by performing the coarsening for each class independently until the number of data points is smaller than M. While the minority class with a small number of points reaches this threshold in a few levels of coarsening, the majority class may take many more steps. The data for the minority class is preserved and transferred entirely to the coarser level. Therefore, at the coarsest level, the size of both classes are balanced. The importance of points and imbalanced size is enforced through the volume for each data point.

3.5.2 Parameter Fitting

We use the Nested Uniform Design (NUD) [44] in our computational experiments for parameter fitting. The framework is flexible to use various types of parameter fitting such as grid search [18]. In NUD, at first stage, multiple points are selected in the search space and models trained based on the parameters representing each of those point. In the following stages, the best point based on highest classification quality is selected as the center for the search. Multiple points around this are evaluated and the best results from all stages is the optimal parameters. In a similar approach, we pass the best parameters from a coarser level as the center for the first stage of NUD. Therefore, the inherited parameters from a coarser level stage is evaluated on the finer data.

3.5.3 Models at multiple scales

The obtained hierarchy normally includes multiple levels. At each level i, we train one or more models on $\mathcal{J}_i$ because in the case of training a model with parameter fitting, there are multiple models generated at one level. Each model is trained on a different set of parameters. As such we need to select the best model within each level and across various levels. Based on our focus on highly imbalanced data sets, we developed a 3-level hierarchical ranking method for selecting the optimal model. The performance measures are well-known metrics to evaluate the quality of classifiers. The geometric mean (G-mean) of the Recall and Specificity is a popular metric for the imbalanced data. The details on the metrics are explained in Section 3.6.1. The way we select the best model at each level is based on: (a) a model with the highest G-mean with significantly higher G-mean than rest of model, and (b) in case of a group of models with similar G-means, a model with the best sensitivity is selected, (c) if there is a tie between multiple models with similar G-mean and sensitivity, we select a model with the least number of support vectors for both classes among the previous group of models. When there is a tie, this ranking is useful. For instance, if there are multiple high-quality models, the one with a smaller number of support vectors has less burden in the uncoarsening. The lower number of aggregates leads to smaller training size at the finer level and higher computational performance of the SVM solver. In general, multilevel learning successfully mitigates the over- and underfitting problems through the hierarchy of models at various levels of coarseness. The overfitting potentially can happen at the finest level, and underfitting may occur at the coarsest level. The levels in the middle has a good trade-off between the bias and variance. We use the 3-level search based on validation set on multiple models that each of them was the best at their corresponding level. We report the final performance quality based on predictions on the test data.

3.5.4 Learning Challenge

In the hierarchical framework, it is expected that the first solution is only a reasonable (but not necessarily the best) approximation of the optimal solution. Therefore, extra steps are required to improve coarse solutions gradually. In the context of the classification task, the initial solution should provide a reasonable decision boundary. We can measure the quality of the initial solution with performance measures on validation data, which we cover in section 3.6.1. We can express it more accurately by

$$Q_{(1)} \leq Q_{(2)} ... \leq Q_{(k)},$$

when $Q_{(i)}$ is the classification quality at level i. We denote this expected trend with a continuous increase in quality as the **natural trend** and Figure 3.2 demonstrates it for the Ringnorm data set.

However, this is not what we observed over exhaustive experiments with various data sets. In our experimental results, there were some cases in which the classification quality was dropped significantly during the refinement. The consequences of unexpected phenomena are as follows.

The training process for the models at the following levels of refinement will be based on sub-optimal training data. As explained earlier, the training data at each level of refinement is filtered to contain only the neighbor points of SVs of a model at the coarser level. When the SVs are sub-optimal, the filtering adds the points which are not representative of the data for the SVM to learn an optimal decision boundary. Moreover, the number of SVs may increase dramatically without significant improvement in the classification quality. The increase in the number of SVs drastically increases the number of SV's neighbors, which forms a larger training data at the next level. Therefore, it reduces the performance of the rest of levels in the refinement.

We recognized a pattern that the quality improves during refinement but at some levels, declines significantly. We denote it as **unnatural trend** and Figure 3.3 demonstrates an example of the unnatural trend for the Cod-RNA dataset. Benefits of detecting and addressing the decrease in quality during the refinement are:

- Improving the classification quality of the finer levels and in general
- Improving the computational performance of the refinement process by improving the quality of training data set for the finer levels.

- Reducing the computational cost of refinement through the Early Stopping process when there is not a significant improvement in quality by training larger models
- Reducing the variance for the classification quality across levels
- Preserve smaller training set during the refinement

Intuitively, the coarsening procedure gradually creates approximated (or summarized) representations of the finest level (original) data. An initial model we train at the coarsest level, provides a large margin hyperplane which can be used as the *final solution*. On the one hand, at the finest level, rich data can easily lead to over-fitted models, a phenomenon frequently observed in practice [28]. On the other hand, over-compressed data representation may lead to an under-fitted model due to lack of detail information. Our experiments confirm that more than half of the best models are obtained from the coarse (but not coarsest) and middle levels which typically prevents over- and under-fitting. In general, if the best models were produced by the finest and middle levels, we recommend to use the model obtained at the middle levels to avoid potential over-fitting. This recommendation is based on the observation that same quality models can be generated by different hyperplanes but finest models may contain a large number of support vectors that can lead to over-fitting and slower prediction. However, this is a hypothesis that requires further exploration. In our experiments, no additional parameters or conditions are introduced to choose the final model. We simply choose the best model among those generated at different levels.

Furthermore, in a multilevel framework, one can use many models across levels that are trained on diverse resolutions of the same data. A good representative validation set is essential for fair evaluation across models and MLSVM [86] has designed and experimented four methods for it. In this work, we used the FF method that provides similar quality as test data and requires low computational resources.

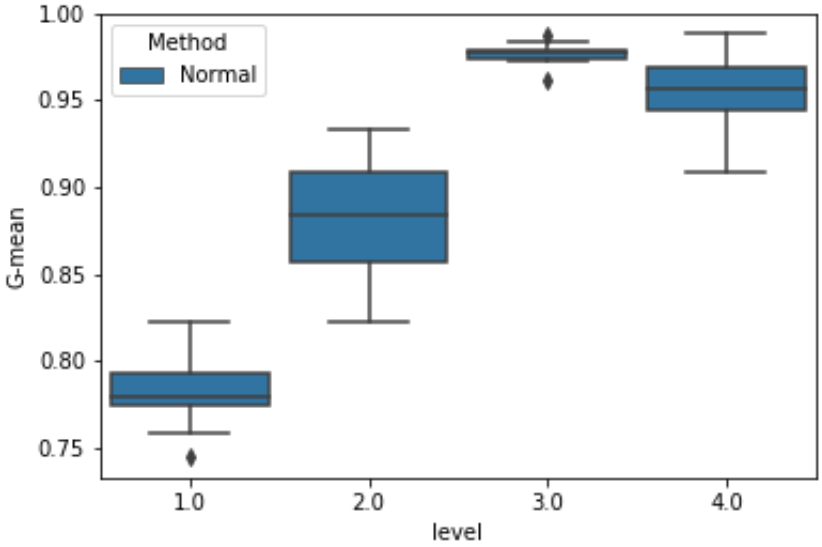


Figure 3.2: Natural trend, increasing classification quality (G-mean) during the refinement.

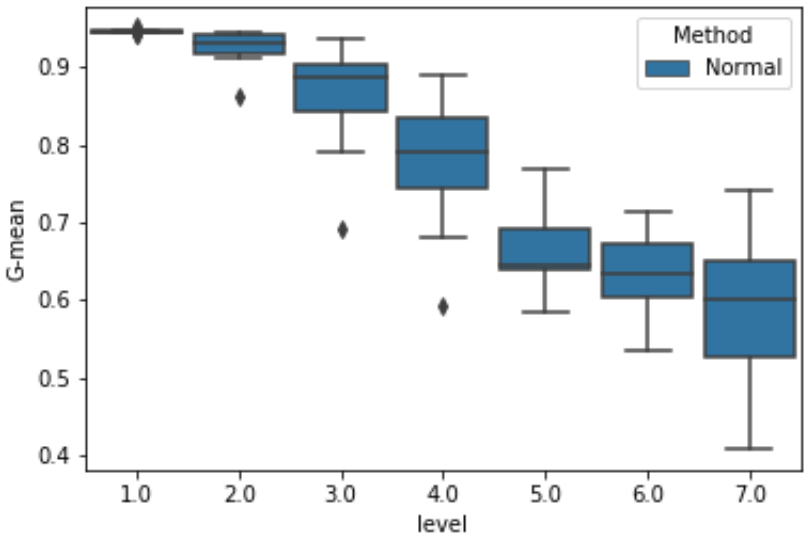


Figure 3.3: Unnatural trend, decreasing classification quality (G-mean) during the refinement.

3.5.5 Proposed Adaptive Method

The notation used in Algorithm 8 is explained in Table 3.1.

Notation	Description
k	Coarsest level identifier
c	Current level identifier
Q	Predictive Quality metric
δ	Parameter: Significant quality threshold
$\mathcal{J}$	Only neighbors of sv in the training set
$\mathcal{J}_{all}$	Complete training set at current level
$\mathcal{J}^a$	Augmented $\mathcal{J}$ (neighbors of sv)
A_i	Augmented point which is a NN of misclassified point i in $\mathcal{J}$
$\mathcal{V}$	Validation set
$\mathcal{F}_p$	Set of False Positive predictions in validation set
$\mathcal{F}_n$	Set of False Negative predictions in validation set
$NN_i^{+(-)}$	Nearest Neighbors in $\mathcal{J}_{all}^{+(-)}$ for data point i

Table 3.1: Notations in Algorithm 8

Algorithm 8 Detection and recovery of quality drop.

$Q_{(max)} = \max\ Q_{(i)}$

  where $i \in \{k, k-1, k-2, ..., c-1\}$

Evaluate performance qualities for all models (NUD)

$Q_{(c)} \leftarrow$ G-mean for the best model at current level

if $Q_{(c)} > Q_{(max)}$; then

  $Q_{(max)} = Q_{(c)}$

else

  if $Q_{(max)} - Q_{(c)} > \delta$; then

    $\mathcal{F}_p \leftarrow$ find false positive points in the validation set

    $\mathcal{F}_n \leftarrow$ find false negative points in the validation set

    for $i \in \{\mathcal{F}_p \cup \mathcal{F}_n\}$; then

      $A_i \leftarrow NN_i^+ \cup NN_i^-$ ▷ new points with respect to i

    $\mathcal{J}^a \leftarrow \mathcal{J} \cup A$

    Train new SVM models on $\mathcal{J}^a$ (NUD)

    Evaluate performance qualities for all new models

    $Q_{(c)}^a \leftarrow$ G-mean for the best of new models

  if $Q_{(c)}^a > Q_{(c)}$; then

    Return$\{\mathsf{sv}_c^a, C^a, \gamma^a\}$ ▷ Improved the quality

Return$\{\mathsf{sv}_c, C, \gamma\}$ ▷ No improvement

How to recover the quality drop? The drop in classification quality is equivalent to an increase in the number of points that are misclassified. Therefore, a classifier quality can be improved by training on more similar data to misclassified data points. The training data at each level is a sub-sample of data around the decision boundary (hyperplane) at the coarser level. The data is sampled from the neighborhood of SVs from the trained model at the coarser level. The points which are misclassified in the training data are not crucial for improvement since the SVM did not find a better solution for the decision boundary. However, the validation data points which are misclassified are the crucial points that need to be classified correctly in this step. It is worth to notice, this step does not rely on any information from the test data and only rely on the validation and training data. Increasing the number of data points usually help the classifier to learn a better decision boundary. A p,n number of neighbors of misclassified validation data points are found in the data from positive and negative class, respectively. The neighbor points are added to the training data, and the model is retrained. The p, n can start from 1 and increase to larger values. However, as these parameters increase, the number of data points that are added to training data increase with respect to the validation data set size and affect the computational performance in the current level and overall framework. We mention important considerations of earlier work in the MLSVM framework.

It is essential to mention that the validation data which is produced in the MLSVM is based on the sampling of the original training data at the finest level. Therefore, the size of the validation data set for large data sets is significantly larger than the training data at the coarser levels. For instance, a data set with $5M$ data points such as SUSY with k-fold cross-validation using $k = 5$ has $4M$ training data at the finest level. Using the validation sample rate of 10%, the validation size at all levels has $400k$ data points. At the coarsest level, the training size is less than $2 \times M$ (600 samples with default $M = 300$) , and it continues to grow slowly over the next few levels of refinement. Suppose the accuracy is 80% at a level. The maximum number of points that can be added to training data would be $400k * 20\% * (p + n)$ or $80k * (p + n)$. This would be an impractical training size. Therefore, we limit the number of p and n to one. In our experiments, many miclassified points have common neighbors which were not in the training. We only add distinct points into training set, hence, the increase is training size is much smaller.

The focus of MLSVM is on extremely imbalanced data sets. The majority of data points belong to the larger class in highly imbalanced data sets. Therefore, a more restrict sampling ratio

helps to reduce the number of data points from the majority or larger class in the validation data. However, the minority or smaller class typically has ten times fewer data points, and its size would not cause performance problems. Therefore, a more significant sample rate, such as 50% or more for minority class compares to 10% or less for majority class, both reduce the number of points in the validation data and make the validation data better representative sample to evaluate the quality of the classifier. A validation set is significant for the framework, which allows us to find the best model from many models at various levels. If the validation data is not representative, the quality of the model will severely affect the test data, which is the ultimate goal of training a classifier. The validation data size for a large data set with a sample ratio of 10% is still huge. Using smaller values such as 2%, the validation data was not representative of test data anymore in the past. With proposed new unbalanced sampling ratios for validation, the 2% validation sample ratio of majority class, and 50% sample ratio of the minority class, reduce the computational challenge.

3.6 Experimental Results

First, we report the comparison between the ML-SVM algorithm and the state-of-the-art SVM algorithms such as LIBSVM and DC-SVM in terms of classification quality and computational performance. Then, we compare the proposed AML-SVM method and ML-SVM method using new set of experiments with more details.

3.6.1 Performance measures

Performance measures are metrics which are used to evaluate the classification quality of the model. We report the accuracy (ACC), recall or sensitivity (SN), specificity (SP), and geometric mean (G-mean). They defined as

$$\text{ACC} = \frac{TP + TN}{FP + TN + TP + FN}, \quad \text{SN} = \frac{TP}{TP + FN},$$

$$\text{SP} = \frac{TN}{TN + FP}, \quad \text{G-mean} = \sqrt{\text{SP} \cdot \text{SN}}$$

Where TN, TP, FP, and FN correspond to the numbers of real negative, true positive, false positive, and false negative points. Our primary metric for comparison is G-mean, which measures the balance between classification quality on both the majority and minority classes. This metric is illuminating for imbalanced classification as a low G-mean is an indication of low-quality classification of the positive data points even if the negative points classification is of high quality. This measure indicates the over-fitting of the negative class and under-fitting of the positive class, a critical problem in imbalanced data sets. In both ML and AML -SVM frameworks, many models are trained. We need to provide one value for prediction as to the final model performance measure. Therefore, we evaluate all the models using the validation set and select the best model and report the performance measures of the selected model over the test (hold-out) set in the results. The detail of imbalanced data sets used in our experiment is presented in Table 3.2. The imbalance ratio of data sets is denoted by ϵ.

3.6.2 LIBSVM, DC-SVM, ML-SVM and AML-SVM Comparison

For self completeness of this chapter and convenience of the readers, we present all the relevant results from the ML-SVM chapter . The following results provides the clear comparison

Dataset	ϵ	n_f	$\lvert\mathcal{J}\rvert$	$\lvert\mathbf{C}^+\rvert$	$\lvert\mathbf{C}^-\rvert$
Advertisement	0.86	1558	3279	459	2820
Buzz	0.80	77	140707	27775	112932
Clean (Musk)	0.85	166	6598	1017	5581
Cod-rna	0.67	8	59535	19845	39690
Forest (Class 5)	0.98	54	581012	9493	571519
Letter	0.96	16	20000	734	19266
Nursery	0.67	8	12960	4320	8640
Ringnorm	0.50	20	7400	3664	3736
Twonorm	0.50	20	7400	3703	3697
HIGGS	0.53	28	11000000	5170877	5829123
SUSY	0.54	18	5000000	2287827	2712173

Table 3.2: Benchmark data sets.

between the state-of-the-art methods which set the base for next section which we provide more detailed results for comparing the ML-SVM and AML-SVM methods.

	LIBSVM		DC-SVM		ML-SVM		AML-SVM	
Datasets	G-mean	Time	G-mean	Time	G-mean	Time	G-mean	Time
Advertisement	0.67	231	0.90	610	**0.91**	213	0.90	**126**
Buzz	0.89	26026	0.92	2524	**0.95**	**31**	**0.95**	269
Clean (Musk)	**0.99**	82	0.94	95	**0.99**	94	0.93	**21**
Cod-RNA	**0.96**	1857	0.93	420	0.94	**13**	0.95	75
Forest	0.92	353210	**0.94**	19970	0.88	**948**	0.88	2149
Letter	0.99	139	**1.00**	38	0.99	30	0.98	**14**
Ringnorm	**0.98**	26	0.95	38	**0.98**	**2**	0.97	3
Twonorm	**0.98**	28	0.97	30	**0.98**	**1**	0.97	**1**

Table 3.3: Performance measures and time for LIBSVM, DC-SVM, ML-SVM and AML-SVM on medium size benchmark data sets.

	LIBSVM			DC-SVM			ML-SVM			AML-SVM		
Datasets	ACC	SN	SP	ACC	SN	SP	ACC	SN	SP	ACC	SN	SP
Advertisement	0.92	0.99	0.45	0.95	0.83	0.97	0.95	0.85	0.96	0.93	0.86	0.94
Buzz	0.96	0.99	0.81	0.96	0.88	0.97	0.94	0.95	0.94	0.94	0.97	0.93
Clean (Musk)	1.00	1.00	0.98	0.96	0.91	0.97	0.99	0.99	0.99	0.95	0.91	0.96
Cod-RNA	0.96	0.96	0.96	0.93	0.93	0.94	0.93	0.97	0.91	0.94	0.97	0.93
Forest	1.00	1.00	0.86	1.00	0.88	1.00	0.77	0.96	0.80	0.84	0.93	0.84
Letter	1.00	1.00	0.97	1.00	1.00	1.00	0.98	0.99	0.98	0.99	0.96	0.99
Ringnorm	0.98	0.99	0.98	0.95	0.92	0.98	0.98	0.98	0.98	0.97	0.98	0.96
Twonorm	0.98	0.98	0.99	0.97	0.98	0.96	0.98	0.98	0.97	0.97	0.98	0.96

Table 3.4: Rest of performance measures for LIBSVM, DC-SVM, ML-SVM and AML-SVM on medium size benchmark data sets.

3.6.3 Implementation

The ML-SVM was developed using C++ and PETSc library [4]. The following libraries are used for various parts of the framework. The FLANN [70] for finding approximate k-nearest

	LIBLINEAR				DC-SVM and LIBSVM	ML-SVM				AML-SVM			
Dataset	ACC	SN	SP	G-mean		ACC	SN	SP	G-mean	ACC	SN	SP	G-mean
HIGGS	0.54	0.55	0.54	0.54	Stopped or failed after 3 days	0.62	0.61	0.63	**0.62**	0.61	0.64	0.58	0.61
SUSY	0.69	0.61	0.76	0.68	Stopped or failed after 3 days	0.75	0.71	0.78	0.74	0.77	0.69	0.84	**0.76**

Table 3.5: Performance measures for LIBLINEAR), DC-SVM/LIBSVM, ML-SVM and AML-SVM on larger benchmark data sets.

Dataset	LIBLINEAR	DC-SVM and LIBSVM	ML-SVM	AML-SVM
HIGGS	4406	without any result	3283	**1488**
SUSY	1300	Stopped or failed after 3 days	1116	**881**

Table 3.6: Computational time in seconds for LIBLINEAR), DC-SVM/LIBSVM, ML-SVM and AML-SVM on larger benchmark data sets

neighbors before coarsening, and METIS [50] for graph partitioning during the refinement. The LIBSVM [17] as the underlying solver for SVM. The implementation is based on a single CPU core, and all the speedups are through algorithmic improvement without leveraging parallel processing. The proposed framework is developed based on the ML-SVM library. Moreover the OpenMP library, which has been used in other application to improve the performance [1], is used to speed up the parameter fitting at each level using multi-threading. Since the number of parameter fitting used for experiments is 9 in the first stage and 4 in the second stage of nested uniform design, we do not evaluate the speedups by increasing the number of threads. The coarsening implementation is sequential and based on a single CPU core. All experiments for data sets with less than 1M data points have executed on a single machine with CPU Intel Xeon E5-2665 2.4 GHz and 64 GB RAM. For larger data sets, we used one single machine with CPU Intel Xeon E5-2680v3 2.5 GHz and 128 GB RAM.

3.6.4 ML-SVM and AML-SVM Comparison

The same experiment setup and hardware configuration is used for the results in this section and next section. However, the purpose of experiments and the version of ML-SVM are different. The experiments are designed to have a better understanding of the multilevel framework and refinement process on each level of the hierarchical framework, while the earlier experiments are designed to compare the overall frameworks regardless of internal details.

In all experiments the data is normalized using z-score. The computational time reported in all experiments contains generating the k-NN graph. The computational time is reported in seconds

unless it is explicitly mentioned otherwise.

In each class, a part of the data is assigned to be the test data using k-fold cross validation. We experimented with $k = 5$ and 10 (no significant difference was observed). The experiments are repeated k times to cover all the data as test data. The data randomly shuffled for each k-fold cross validation.The presented results are the averages of performance measures for all k folds. Data points which are not in the test data are used as the training data in $\mathcal{J}^{+(-)}$. The test data is never used for any training or validation purposes.

We run 18 experiments per dataset using nested cross-validation with k-fold where $k \in \{5, 10\}$. with various configurations to evaluate the performance of the proposed method (AML). For each data set, we evaluated the combination of important parameters such as interpolation order (r), validation sample ratio (v), and where we stop to partition the data in ML method, which we define as stopping criteria for the AML method. We controlled all the other variables only to consider the effect of new method for an exactly similar configuration. Each combination of configuration provide slightly different results per dataset, and therefore, we evaluate 18 combinations of parameters and present the statistical information per dataset. The goal of the experiment is to have a well-designed experiment for comparing the current and new methods. We provide the related results from the MLSVM paper for convenient comparison of AML method with other states of the arts SVM solvers such as LIBSVM and DC-SVM. It is worth to mention, the results for comparing the ML and AML-SVM are based on a different set of new experiments on the newer version of the library, which has improved since the publication of the MLSVM paper. The current version of ML-SVM is v1.1.3, and the current version for AML-SVM is v1.2.2.

For each configuration, we present three sets of results over k-fold cross-validation.

- Classification quality: we plot the statistics of G-mean at each level of uncoarsening in Table 3.9, 3.10.
- Computational Performance: the average running time for both coarsening and uncoarsening framework is reported.
- An overall comparison only for the final results of the whole framework without details for levels are provided in Tables 3.3 to 3.7

Based on our observation for high quality model at earlier stages of the refinement, we evaluate the early stopping (termination) of the refinement if there exist a model that satisfies the

Dataset	ML	AML
Advertisement	67.6	126.0
Buzz	44.6	269.2
Clean	6.0	20.6
Cod-RNA	7.0	75.2
Forest	4993.3	2148.7
Letter	8.2	13.9
Ringnorm	3.1	3.1
Twonorm	0.8	1.4
HIGGS	6064.3	2134.4
SUSY	664.3	881.2

Table 3.7: Performance Time (seconds)

threshold quality. The results in the Table 3.8 demonstrates the predefined threshold, performance time for each method to reach an optimal solution, and the average number of levels for the refinement. The Time and Refinements are average across various configuration that results to fractions for number of refinement levels. The comparison between the time of the methods in Tables 3.8, 3.7 shows significant speed up for all data sets. There is an order of magnitude speed ups for Forest, SUSY, and Buzz datasets. The threshold which is set as expected results is in the range of 0.01 to 0.06 G-mean compared to best results in Tables 3.3, 3.5

	Threshold	Time		Refinements	
Dataset	G-mean	ML-SVM	AML-SVM	ML-SVM	AML-SVM
Advertisement	0.85	10.4	15.0	1.1	1.1
Buzz	0.90	21.6	22.2	1.0	1.0
Clean (Musk)	0.90	2.8	3.5	2.0	1.9
Cod-RNA	0.90	2.6	2.7	1.0	1.0
Forest	0.85	318.2	253.2	2.5	2.0
Letter	0.95	1.6	1.6	1.0	1.0
Ringnorm	0.95	2.7	2.5	3.0	3.0
Twonorm	0.95	0.6	0.6	1.0	1.0
HIGGS	0.60	408.2	457.5	1.0	1.0
SUSY	0.70	223.9	228.7	1.7	1.7

Table 3.8: Average time (seconds) and number of refinement levels for the early termination.

The left column in Table 3.9 shows the average results over 18 experiments, and the right column shows a selected sample (with k results) from those experiments. The number of levels in coarsening and uncoarsening is the same, and we use them interchangeably for the rest of this section. The Advertisement dataset has 3,279 data points and 1,558 features. It has a large number of features and a small number of data points. The number of levels in the coarsening phase, only depends on the number of data points. Therefore, there are only 3 levels in (un)coarsening phase

for the Advertisement dataset. Both ML-SVM and AML-SVM frameworks have identical results for the first and second levels. However, the AML-SVM achieved higher quality on the final level. All the qualities are reported based on G-mean performance measures.

The Buzz dataset has 140,707 data points, 77 features, and 7 levels of coarsening. The quality of both methods is comparable except for a significant drop on the 2nd level for the ML-SVM quality. The Clean dataset has 6,598 data points, 166 features, and 5 levels of coarsening. The ML-SVM's quality is descending from the 3rd level to the 5th level. However, while the AML-SVM has not decreased, in the 4th and 5th levels, it achieves the highest quality across all the levels. The Cod-RNA dataset is related to Breast Cancer. It has 59,535 data points and 8 features. It has a similar decreasing trend as the Clean dataset.

The Forest (Cover Type) dataset is one of the larger data sets with 581,012 data points and 54 features. Contrary to the earlier data sets, the quality (G-mean) at the first (coarsest) level is lower than most of the levels. The ML-SVM quality is increased in the 2nd level but continues to descend til level six. The quality at 7th level is increased with a slight decrease in the following levels. The AML-SVM recovers the descend at the third level and achieves the highest G-mean in third and fourth levels.

We designed early stopping for the AML-SVM to reduce the computational complexity. Therefore, as the size of training data reaches a threshold θ in Algorithm 7, the refinement process stops. The threshold is an optional input parameter for the framework. For these experiments, the θ is set to 5,000, therefore, the refinement stopped when the number of data points in the training data is reached to 5,000.

The Letter dataset includes 20,000 samples and 16 features of the 26 English letters. The ML-SVM has a descending trend for quality on the Letter data set similar to Clean and Cod-RNA data sets. However, AML-SVM has no decrease in quality. The Ringnorm dataset has 7,400 data points and 20 features. The ascending quality is observed for both methods without any drop in qualities. The Twonorm dataset has the same size and number of features as the Ringnorm [13]. The ML-SVM quality drops in the 3rd and 4th levels. The adaptive methods achieve high-quality results on all levels.

The HIGGS dataset has 11,000,000 data points and 28 features. The SUSY dataset has 5,000,000 data points and 18 features. The quality of ML-SVM for both HIGGS and SUSY data set has a similar trend of high quality at the coarsest levels following with lower quality in the middle

levels. At the finer levels, there is a slight improvement, but the variance of qualities is larger than the coarser levels. The adaptive method for the SUSY dataset achieves high quality up to level 6. In levels 7 and 8 the quality of adaptive method drops as well, and at level 8, the adaptive method stops. For the HIGGS dataset, the adaptive method has better quality compared to the normal method. At level 9, the adaptive method stops to reduce the computational complexity.

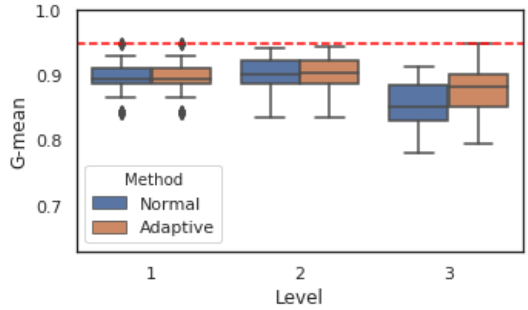
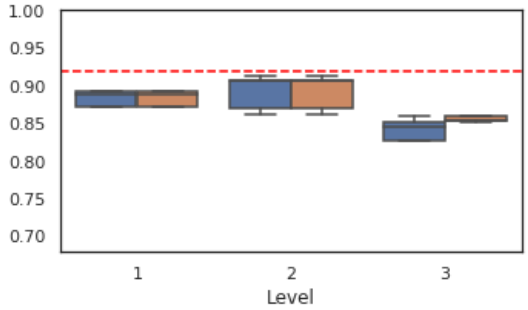
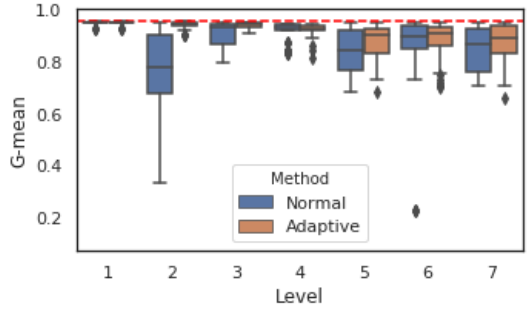
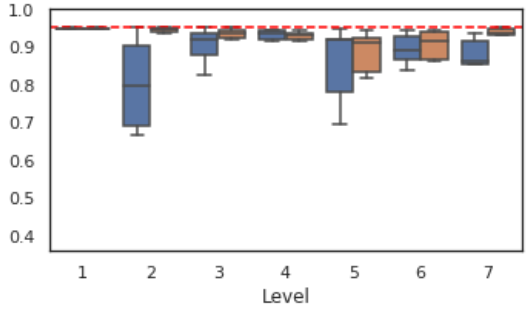
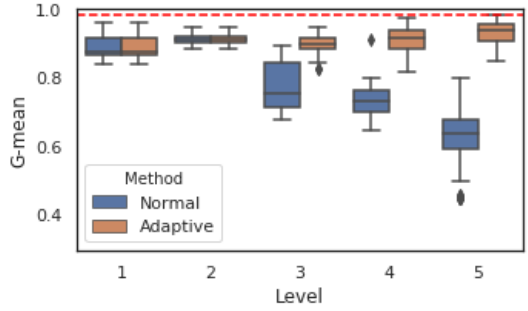
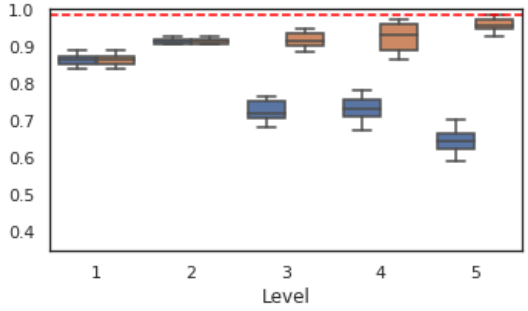
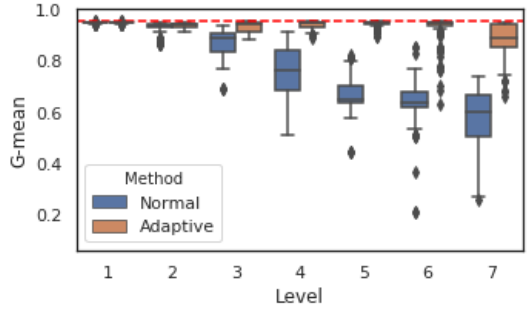
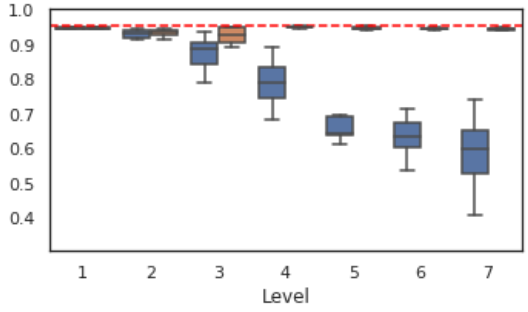
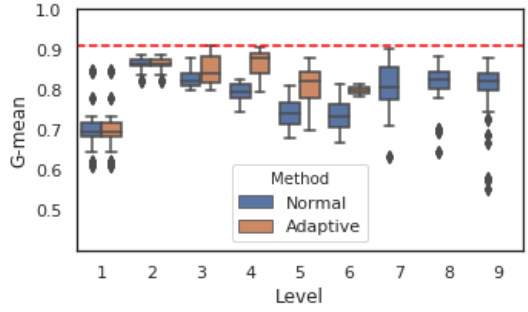
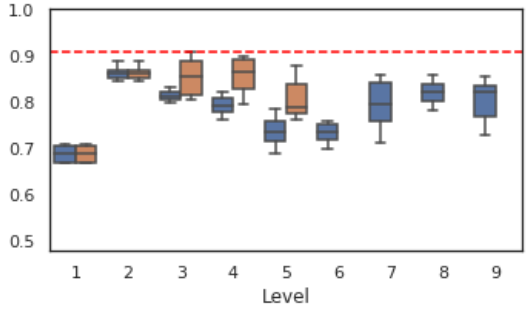
Name	Average	Sample
Advertisement		
Buzz		
Clean		
Cod-RNA		
Forest		

Table 3.9: Comparing the prediction quality across levels, part 1

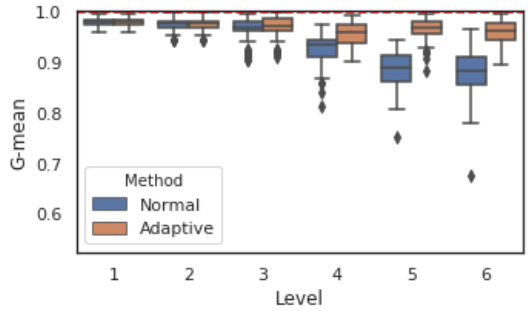
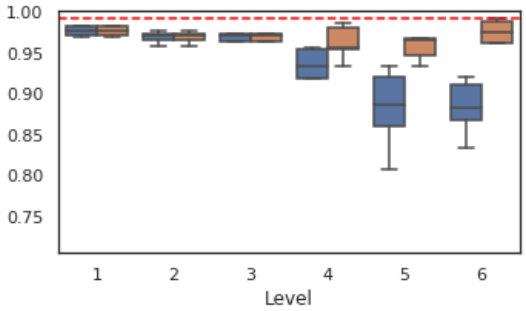
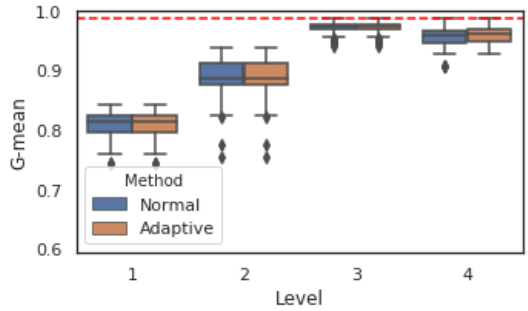
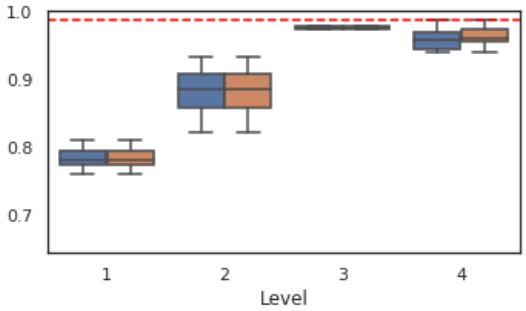
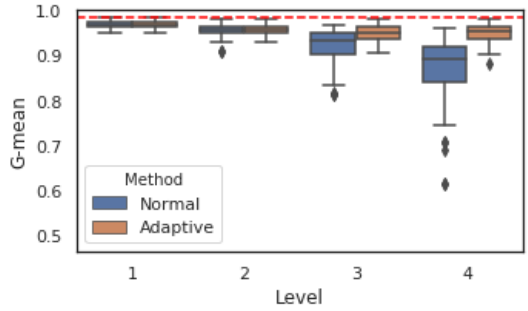
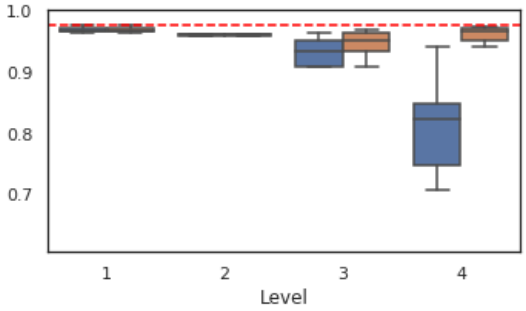
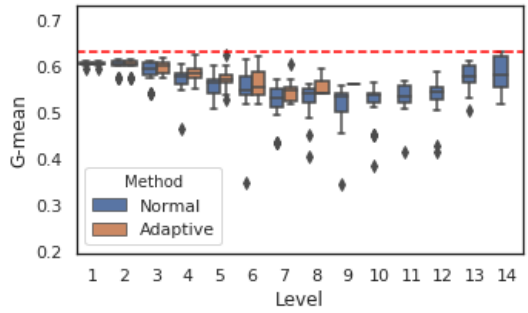
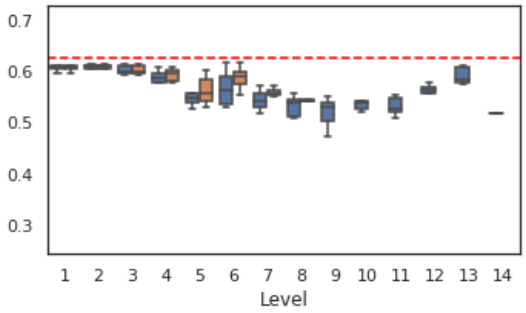
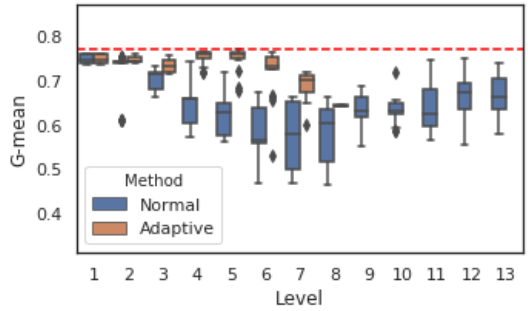
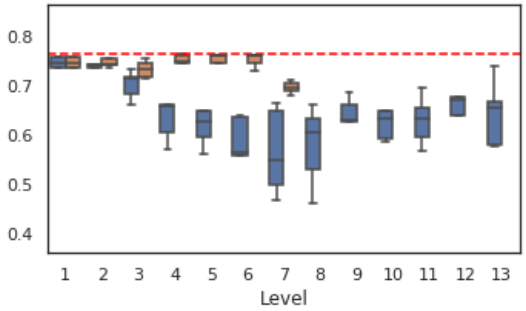
Name	Average	Sample
Letter		
Ringnorm		
Twonorm		
HIGGS		
SUSY		

Table 3.10: Comparing the prediction quality across levels, part 2

3.7 Conclusion

In this chapter , we introduced a novel adaptive recovery technique for under/over-fitting challenges in hierarchical frameworks which performs well on nonlinear support vector machine. Our results based on 18 distinct configurations of essential parameters for both coarsening and uncoarsening phases show that the adaptive approach is less sensitive to changes in configurations. One advantage is more convenience to train a high-quality classifier for researchers as they explore fewer configurations for their custom data sets. Another advantage is less computational cost and time for evaluating multiple configurations. The adaptive method has reduced the prediction quality variance across various levels in the multilevel framework, which provides a more robust solution. While we did not explore ensemble approaches over many models, we train many models at each level (parameter fitting) and across levels. Therefore, the lower variance between these models would be helpful for future work as well.

On massive data sets, the training size may increase significantly on the finer levels that dramatically affect the computational performance. Our results with earlier versions show that quality achievement is not significant in many cases. Therefore, our proposed early stopping functionality improves computational performance as it prevents computationally expensive training at the finer levels. We developed multi-threading support for parameter fitting, which utilizes more processing power on single computing node and increases the overall performance.

The proposed approach can be extended to a hierarchical learning framework with a clear objective for quality. Our exhaustive results demonstrate the ability of classifiers to learn a high-quality model at the coarsest level. It emphasizes that the small aggregated training data at the coarse levels of the framework is a good representative of the whole dataset. For large-scale problems, sampling is a good approach to reduce the computational cost for intermediate tasks such as feature selection or dimensionality reduction. The proposed framework is a useful alternative for simple sampling of data.

Chapter 4

Leading Medical Research using Knowledge Network Analysis

4.1 Introduction

One of the challenges with large heterogeneous datasets is an issue of extracting effective features for the data analysis tasks. In particular, this problem is becoming extremely sensitive with the constantly growing size of data fused using various sources of information such as those that are typical in social networks and biomedical domain. In this chapter, we propose a probabilistic feature extraction method that provides a low-dimensional representation of data with minimal computational cost. We demonstrate the applicability of the proposed methods in the literature-based biomedical discovery domain [103] in which properly designed expert systems play a crucial role to accelerate scientific discovery.

Biomedical science is one of the information most reachest domains in which information is generated by such sources as scientists, automated experiments, personal and lab devices, clinical trials, and healthcare providers. A significant part of this information is summarized and published in biomedical journals and conferences which, in turn, helps other researchers to further generate new knowledge that will be published in the literature and other related datasets. Therefore, the size of many datasets in the biomedical domain persistently grows to an overwhelming scale [51], and the scientific literature is one of them. Typically, the analysis of scientific literature is tightly related

to high-dimensional space of heterogeneous features that can include words, n-grams, predicates, categories, authors, type of organizations, ontology and other meta or real information. Such feature spaces cause numerous problems with numerical instability, modeling, and computational challenges for the downstream machine learning methods in expert systems.

4.1.1 Information Retrieval and Literature Based Discovery Systems

The classical Information Retrieval (IR) techniques work well to respond to a search query and finding related explicit connections in articles. Furthermore, some IR techniques can find limited implicit connections among documents such as Latent Dirichlet Allocation (LDA) [120], which can model and extract topics from a large set of documents. However, the IR methods are insufficient to fully extract and predict the implicit relations between concepts and keywords in a massive set of documents. Interpretable methods are required to explain the implicit connections. There is a growing interest in research for methods to utilize the vast corpus of biomedical literature to generate new knowledge. The existing Literature Based Discovery (LBD) [103, 34, 40] approaches leverage this information to find implicit relationships. The LitLinker [76] express it as "tools to help researchers capture new knowledge that bridges gaps across distinct sections of the literature." The LDB systems have shown to be useful in many domains such as drug target discovery, cancer, gene and disease connections [33].

One of the first LBD approaches was proposed by Swanson [106, 107]. A triplet base structure was intriduced to build a Knowledge Graph (KG) for LBD. Entities and their connections are encoded into a KG, which can be used for IR or Knowledge Discovery Swanson proposed the ABC paradigm to find relations between concepts or entities which are not mentioned together in literature. In other words, given A, B, and C are biomedical concepts, the $(A - C)$ connection (expressed in co-occurrence in the same literature) has not been mentioned, while both $(A - B)$, and $(B - C)$ have been. The triplet base structure, $A - B - C$, is widely used to model and further expand the Swanson's ABC paradigm.

The MOLIERE system [110] (and its recent next generation system AGATHA [111]) is an example of LDB that processes multiple sources of information and generates a hypothesis about the relation between two disconnected entities in the knowledge graph with high accuracy. However, one critical problem of MOLIERE and other similar LBD systems is lack of accompanying methods to predict the type of relations of new hypotheses. Various relationship types link the internal

entities of the knowledge graph to each other and then used to analyze biologically meaningful pathways. Our contribution is developing an efficient low-dimensional feature extraction method that significantly improves the machine learning pipeline's performance for predicting the relation type of new hypothetical links in knowledge graphs.

4.1.2 Big Data Challenges

There are multiple challenges in predicting the right relation between two concepts such as vast volume of literature, variety of types and relationships especially if full texts are analyzed [109] using computationally heavy tasks [37]. We mention some of the datasets here but more details are provided in Section 4.2.1. MEDLINE is the most popular source of data for literature collection in biomedical domain. It has more than 28 million papers. The Unified Medical Language System (UMLS) has a Semantic network which has 54 relationships and 15 semantic groups. The Semantic MEDLINE Database (SemMedDB), contains more than 96 million predications extracted from articles in MEDLINE with relationship types or (predicate) between UMLS concepts. Moreover, many other data sets exist, which can be linked to some of the publications to enrich knowledge. There have been researchers and commercial products to link a variety of datasets [117].

4.2 Background

First, the necessary terms or concepts are explained and then the datasets which are popular in biomedical domain are introduced.

- **Concept** is a term or phrase which refers to a biomedical entity. Some data sources have a limited set of concepts, and some use well-known concepts from other datasets.
- **Semantic Predicate** is a triple of subject-relation-object extracted from biomedical texts. For instance, *Infection-CAUSES-Guillain-Barre syndrome* is a predicate where the *Infection* is the subject, *CAUSES*is the relation and *Guillain-Barre syndrome* is the object.
- **Knowledge Graph (KG)** is an integration of multiple resources to form a better understanding of the relation between concepts or entities in various resources. We use the Knowledge network and knowledge graph terms interchangeably in this manuscript.

4.2.1 Datasets

A variety of datasets provide the required information for an HG system. We explain the related essential datasets in the following sections.

4.2.1.1 Unified Medical Language System

Unified Medical Language System (UMLS) is a biomedical knowledge-base as a set of files and softwares which connect manifold of biomedical vocabularies and standards. Therefore, it is a core part for many KD or HG systems [9]. There are three knowledge sources for UMLS:

- **Metathesaurus** is the most significant component in UMLS which includes sizable biomedical thesaurus organized by concept and meaning. It links alternative names and views of the same concept from nearly 200 various vocabularies. Moreover, it provides hierarchies, definitions, and other attributes.
- **Semantic Network** consists of two parts:
    1. Nodes or Vertices: All concepts in the UMLS Metathesaurus are categorized consistently by subject categories or semantic types. Semantic types are nodes in the semantic net-

work, like semantic types related to *Fish oil* ["biologically active substance," "lipid," "pharmacologic substance"].

2. Edges or Relations: The links between semantic types are the edges in semantic network which are called Semantic Relations. At present, there are 54 semantic relations such as "isa", "surrounds", "prevents", and "treats".

Furthermore, the Semantic Network includes semantic groups that is a set of grouping concepts according to their semantics types. There are 15 semantic groups such as *Chemicals and Drugs* which includes semantic types ["Amino Acid, Peptide, or Protein", "Antibiotic", "Biologically Active Substance", "Biomedical or Dental Material", "Chemical", "Chemical Viewed Functionally" and many other semantic types]. The full list of semantic groups are available at NLM website.

- **SPECIALIST Lexicon and Lexical Tools** provides a manifold of tools which are helpful for a Natural Language Processing (NLP) specialist. Some of the tools are spell checkers, a dataset of n-grams, prefix and sub-term finder, and Part of Speech (POS) tagger.

4.2.1.2 MEDLINE

MEDLINE is one of the most popular sources of data in the biomedical domain, with more than 28 million citations. An article or paper in MEDLINE has a title, abstract and indexing terms based on MeSH terms. There are a limited number of full-text articles indexed in MEDLINE accessible through PubMed Central (PMC). Today, the PMC has more than 5.5 million full-text records of biomedical and life science research from late 1700s to present. There are more than 7 thousands journals which provide content to PMC.

4.2.1.3 MOLIERE

The MOLIERE is a hypothesis generation system which uses the closed discovery model to generate new knowledge discovery. The MOLIERE has an aggregate multi-layered knowledge network from multiple sources, which consists of papers, terms, phrases, and various types of links between them. As a closed discovery model, MOLIERE queries two UMLS concepts in its KG based on the source and target concepts and returns a related set of nodes, which is equivalent to a cloud or path over entities that connect source and target. The cloud includes papers, related abstracts,

and titles as intermediate data as one of the data sources in this chapter. MOLIERE ranks and validates the hypothesis on a massive scale with high confidence using different similarity metrics in embedding space, on graph structures and topic modeling correlations [110, 108]. Besides the intermediate clouds for each query, the information provided by MOLIERE systems such as topics and validation information is not used in our analysis. So, it makes PLBD generalizable to any other HG system, providing a set of entities that relate the two concepts.

4.2.1.4 Semantic MEDLINE database

Semantic MEDLINE database (SemMedDB) is a repository of all semantic predications extracted from the MEDLINE articles using SemRep [81]. The subject and object in a triples refer to the UMLS Metathesaurus concepts and the predicate refers to the UMLS semantic relations. The list of all predicates are listed in [54]. There are 54 semantic relation types in SemMedDB and UMLS's Semantic network. The main categories of relation types are clinical medicine (e.g. TREATS, DIAGNOSES), substance interactions (e.g. INTERACTS_WITH, INHIBITS, STIMULATES), genetic etiology of disease (e.g. ASSOCIATED_WITH, CAUSES), and pharmacogenomics (AFFECTS, DISRUPUTS). The SemMedDB and SemRep are both supported by National Library of Medicine. The SemMedDB has two releases per year and has been used as a knowledge resource in hypothesis generation and literature based discovery and clinical decision-making support systems [55]. The lastest release of SemMedDB is semmedVER31_R (Processed up to June 30 2018), which has 96,363,098 predicates extracted from 28,429,379 citation in the MEDLINE using semrep version 1.7.

4.3 Related Work

Over the last two decades, many HG systems have been developed with common goals, such as finding an implicit relationship between concepts. Their design and functionality depend on their specific goals, the volume of information, diversity of data sources, and analysis algorithms. We categorize the systems into two groups based on the volume of information they process and explore. Some systems consider a specific domain, for instance, particular diseases or genes, while others are not limited to a fixed set of concepts.

4.3.1 Limited Domain or Volume

We define the category for systems that either focus on a specific domain or use a very limited volume of information. The first subgroup of this category focuses on a narrow fields, such as drug-diseases or gene-drug indirect relations. The second group is not limited to one domain but focuses on a small volume of data or highly restricted validation set. A typical validation approach for such systems is to replicate Swanson's experiments. For instance, a specialized HG in drug and disease relations at the protein level is developed by [117]. They used a commercial knowledge graph platform with 176 knowledge sources for drug efficacy screening and leveraged the predicates to improve their results. Their analysis considers three relationship scenarios; self relation, direct relation, and indirect relations. The indirect relation with an intermediate node is limited only to one node in the middle. They extract 1.58 million triples for the Euretos knowledge platform. Using binary features for these three relation scenarios and random forest as a classifier, they achieved an area under the ROC curve 78.1% for a limited set of drugs and diseases.

4.3.2 General Domain

The limited domain or volume approach is known for finding and replicating the Swanson's original experiment. However, they are not directly generalizable to other domains due to the heterogeneous nature of other datasets required to form a coherent KG. Furthermore, as some of these systems only limit the search to a narrow set of concepts, fields, and semantics, they may not find potentially related concepts that are not added to their KG or search scope. An essential HG's goal is to help researchers explore implicit connections from other parts of the literature to their work, which is hard to expose otherwise. However, the limited domain HGs typically do not analyze

the massive volume of data from multiple sources. For instance, [76] developed LitLinker, a system that incorporates knowledge-based methodologies, text mining, and data mining approaches to find new, potential causal links between biomedical terms. They identify correlations between concepts using data mining techniques. Also, they used these correlations for open discovery and provided ranking for the related terms.

Earlier, we mentioned closed discovery as one type of knowledge discovery that uses two concepts a and c as an input. Open discovery is another type of knowledge discovery that relies only on one concept instead of two; given an a, the system finds all potential c. Therefore, the system searches and finds all related concepts $C \in \{c_1, c_2, ..., c_n\}$. Each query may have a various number of related concepts in C. Some systems provide a rank for items in C. Most of the closed discovery HG systems can be modified to perform an open discovery [34].

LitLinker uses associations rule mining to find new related concepts in C. It starts with concept a and uses association rule to find many related concepts b where $(a \rightarrow b)$. Then for each concept b, the related concept c given $(b \rightarrow c)$ are found. It forms a large number of indirect relations from $a \rightarrow c$. The manual step to finding optimal support and confidence threshold parameters for pruning is challenging.

4.4 Problem Definition

There are many works on link prediction in biomedical domain knowledge graphs, but predicting the relationship type results requires more effort. The challenges to classify the relationship type accurately include but not limited to: (a) heterogeneity of edge or relation in knowledge graphs; (b) sparse and high dimensional representation; (c) multiple types of links between two entities in the graph; and (d) existance of extremely rare types of relations, i.e., imbalanced classes. This work focuses on developing a feature extraction method that performs well on relationship type prediction.

The vast number of unique entities, along with various relationship types, explode the total number of combinations. The memory of typical computer systems is less than hundreds of gigabytes, and the GPU memory is much smaller. For massive problem, either the analysis of a task is infeasible, or a large cluster of powerful computers is required. For the latter, each specific task requires an efficient distributed version of its algorithm as well.

The quality of the training data is an essential parameter for the quality and performance of Machine learning algorithms. While the training data is not always structured, the representation of the data is a crucial part of all the following tasks, such as visualization, pattern recognition, and finally, interpretation of the models or outcomes. Considering all the possible combinations of relationships between entities introduces multiple adverse effects with storage, high dimensional data, and lack of interpretability of the final machine learning models.

In the following section, we explain some of the approaches we explored and the insight we learned during this process. We hope this information saves time and effort for researchers who are working on similar problems. Many of the existing implementations on graph analytic do not scale to large weighted networks, especially with heterogeneous nodes or edge types. Therefore, there are potential research opportunities for developing well-known algorithms for graph analytic using distributed frameworks such as Hadoop, Dask, and Spark.

4.5 Methods

We started by extracting all the triplets and the year they mentioned for the first time in the SemMedDB database. There are 54 predicates (relation types) in SemMedDB, where only some of them have enough samples. Hence, we keep the 12 predicates which have more than 1% sample ratio in the data. The list of predicate are available in Table 4.1. The related queries between year 2013 to 2017 are processed using MOLIERE system. We used a set of PubMed identifier (PMID) from the output of query over MOLIERE system. Next, we map the PMID in the set to triplets from the SemMedDB. Five fields for each triplet are extracted which are {Subject Semantic Type, Subject CUI, predicate, Object Semantic Type, Object CUI}

4.5.1 Feature Extraction

- Semantic Types Only (STO)

  We consider a unique {*subject semantic type, predicate, object semantic type*} from the SemMedDB to be one feature. We calculate the Term Frequency Inverse Document Frequency (TF-IDF) for each feature in all queries for the training data. The total number of features considering the semantics types are 26k.

- CUI-Predicate-CUI (CPC)

  SemMedDB has information for the unique subject and unique object CUI in a predicate that is not part of STO. We expand the features to consider all the five values as a feature; {*subject semantic type, subject_CUI, predicate, object semantic type, object_CUI*}. We calculated the TF-IDF value similarly, and the number of features is 4,838,789.

- Second-Order Markov Chain over whole SemMedDB (SOMCS)

  All the unique triplets of {*subject semantic type, predicate, object semantic type*} are extracted from the SemMedDB [55]. There are 26k unique combinations for the selected 12 predicates. The probability for each predicate considering a specific semantic type for a subject and an object are processed using second-order Markov chain [97, 78]. There are 133 semantics types, denoted by N. Functions $f(s), g(s,o), h(s,o,r)$ are defined as

$$f(s) = \begin{cases} 1 & \text{if } s = SST_i \\ 0 & \text{else} \end{cases} \tag{4.1}$$

$$g(s,o) = \begin{cases} P\left(SST_i\right) & \text{if } s = SST_i \quad \text{and} \quad o = OST_j \\ 0 & \text{else} \end{cases} \tag{4.2}$$

$$h(s,o,r) = \begin{cases} P\left(SST_i, OST_j\right) & \text{if } s = SST_i \quad \text{and} \quad o = OST_j \quad \text{and} \quad r = R_k \\ 0 & \text{else} \end{cases} \tag{4.3}$$

The following equations explain the processing of the probabilities for a predict at three steps.

$$P\left(SST_i\right) = \frac{\sum_{i=1}^{N} f(s)}{N} \tag{4.4}$$

$$P\left(SST_i, OST_j\right) = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} g(s,o)}{\sum_{i=1}^{N} f(s)} \tag{4.5}$$

$$P\left(SST_i, OST_j, R_k\right) = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{12} h(s,o,k)}{\sum_{i=1}^{N}\sum_{j=1}^{N} g(s,o)} \tag{4.6}$$

, where SST_i denotes i^{th} Subject Semantic Type, OST_j denotes j^{th} Object Semantic Type, and R_k denotes k^{th} relationship type. The 12 probability values for the list of selected predicates for each pair of (SST_s, OST_t) form a vector. We create a matrix of all probabilities considering only the two query concepts (entities) from the year 2013 to 2018 by processing all the clouds. The matrix (training data), is used as the input for machine learning models.

- Second-Order Markov Chain over MOLIERE clouds (SOMCM)

  A set of PMID, $\mathcal{S}_p$, is one of the outputs MOLIERE generates for a query. We matched the corresponding set of predicates from SemMedDB, $\mathcal{S}_s$, to set of PMID $\mathcal{S}_p$. Next, we calculate the probability of a relation type over set $\mathcal{S}_s$. SOMCM has extra information related to the abstracts compared to SOMCS. Using equations 4.1 to 4.6, we calculate the probability of each predicate.

4.5.2 Preprocessing Data

We normalized the STO and CPC data using max value to preserve the sparsity of the matrix data. The elements in the "Second-Order Markov Chain" data are probabilities in the range $(0, 1)$; hence, no normalization is required.

4.5.3 Dimensionality Reduction

We used Fast Randomized SVD library [115] to reduce the high dimensional data (4,838,789 features) constructed by triplets of "SubjectCUI, Predicate, ObjectCUI", to l number of dimensions. We experimented variety of lower dimension where $l \in \{250, 500, 2000, 5000\}$.

4.5.4 Random Forest

The Random Forest (RF) is a well-known classification technique that is fast and embarrassingly parallel. The RF over the "Second-Order Markov Chain" data results in high accuracy. We used the Scikit-learn library [74] for RF method.

4.5.5 Support Vector Machines

We evaluated the performance of predictive models using the Support Vector Machines (SVM) for the STO and CPC data. The LIBLINEAR [29], LIBSVM [17], and MLSVM [86, 87] libraries are tested for training predictive models. The LIBLINEAR and MLSVM are fast and scalable to large datasets while LIBSVM is extremely slow on large dataset which is not practical for the full size of training data.

So, we sampled 10% of the training data and applied the SVM. However, we report the results over the complete validation set. For test data, we randomly queried 1500 pairs of UMLS concepts per class. We continue the experiments with the same 10% since other classifiers such as random forest and deep learning, achieved high-quality classification using this, but their classification quality was not significantly improved by using the whole training data. The results section covers various sample sizes and the quality of the models. The SVM was initially designed as a binary classification algorithm, and later multi-class support was introduced in various research [43, 102, 63].

4.5.6 Deep Learning

We developed various models to make a comprehensive comparison of algorithms due to the popularity of deep learning (DL) models. One of the useful advantages of Deep Learning libraries such as Keras is the functional API, which allows the extension of the model by merging multiple models without training from scratch. The embedding or lower dimensionality representation of the data can is helpful for data with a large number of features or sparse features. The code for the

various models which was designed and developed using Keras and Tensorflow will be accessible in the GitHub repository. For running the experiments, we used a machine with Intel(R) Xeon(R) Gold 6148 CPU @2.40GHz CPU, with two GPUs: Tesla V100 and 376 GB of memory.

4.6 Results

4.6.1 Data

The distribution of data across more than 50 classes are highly skewed and there are many classes with few samples. Therefore, we selected 12 classes with the most number of predicate (samples). We have assigned an Id to each class and the description on Table 4.1 demonstrate the predicate's description.

Class	Description	Number of Records
0	LOCATION_OF	2,357,400
1	AFFECTS	1,529,440
2	TREATS	1,376,228
3	COEXISTS_WITH	1,424,424
4	INTERACTS_WITH	1,234,449
5	PROCESS_OF	906,981
6	CAUSES	798,913
7	PART_OF	1,379,950
8	USES	1,355,271
9	ASSOCIATED_WITH	647,512
10	AUGMENTS	559,060
11	DISRUPTS	414,266

Table 4.1: List of predicate and corresponding class identifier.

The best results are achieved based on fusing three the STO, SOMCS and SOMCM feature sets. Figure 4.1 depicts the Receiver Operating Characteristic (ROC) curve results for all 12 classes.

The average Area Under the ROC curve (AUC) is 0.97 which provide strong prediction power to the model. More detail on classification per class is provided as a confusion matrix in the Appendix 4.9

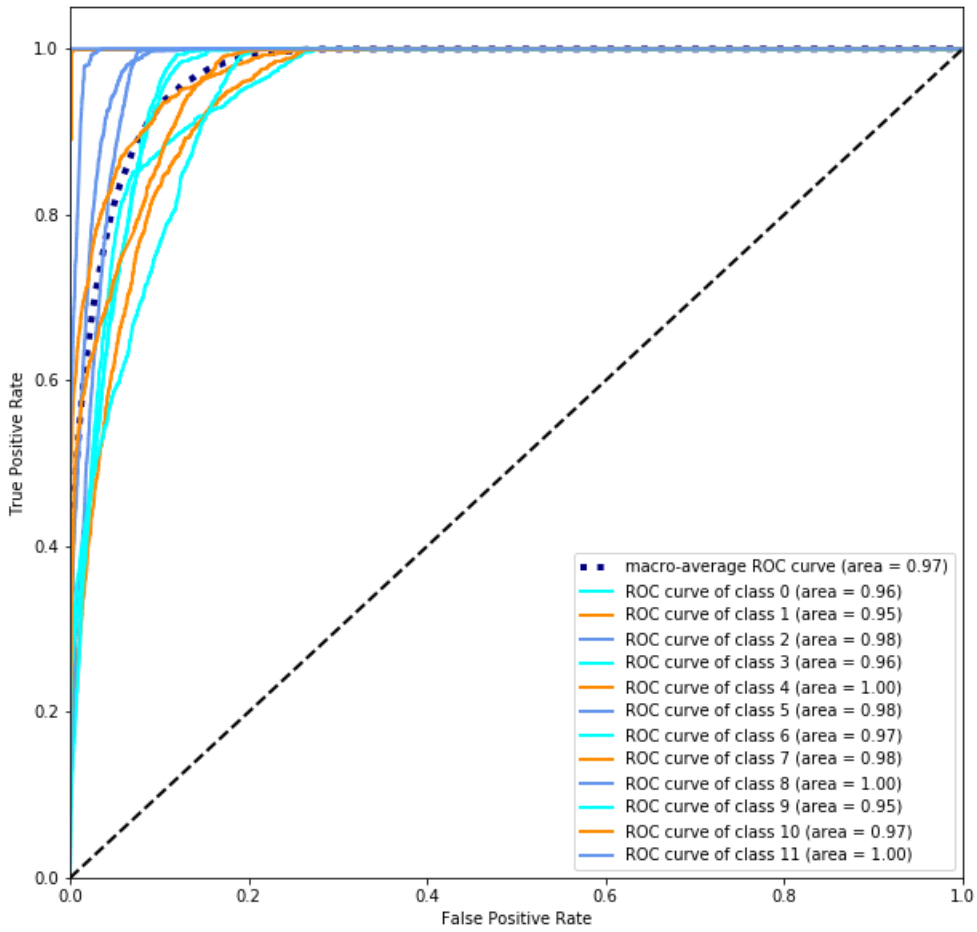


Figure 4.1: ROC curve for 12 classes

4.7 Discussion

While it seems there are a large corpus of information in MEDLINE, it is not close to all the published literature in the medical world internationaly. There are many journals which are not indexed in MEDLINE now [45]. There are many other datasets which can be aggregate to knowledge network such as PubChem [119], Gene-Disease Association [101]. While it is important to consider more data to increase the probability of finding more information and emerging patterns, as the number of heterogeniouse sources of data fuse together, larger volume and complexity emerged on the knowledge graph. Therefore, fast and scalable approaches are required to process the massive volume of heterogenous data in a reasonable time and computational cost. The proposed approach leverage probabilist approach to tackle the problem and prone highly unlikely outcomes to facilitate less computation on machine learning approach and improvement on the final quality. The proposed approach relies on smart and innovative reduction of the original problem that results the following invaluable outcomes.

- The prediction quality improved significantly compared to tackling the original problem.
- The computation is significantly low and embarrassingly paralle without GPU requirements.
- The framework is clear, understandable, interparatble and eacy to build and utilize for similar problems.
- This approach highlights the importance of human (researcher) collaboration with AI to achieve simpler and more efficient algorithms for hard problems which are not well-known or studied heavyly.

4.8 Appendix: Datasets

The list of 12 selected predicates are as follow.

LOCATION_OF, AFFECTS, TREATS, COEXISTS_WITH, INTERACTS_WITH, PROCESS_OF, CAUSES, PART_OF, USES, ASSOCIATED_WITH, AUGMENTS, DISRUPTS.

4.9 Appendix: Results

Figure 4.2 demonstrate the normalized confusion matrix for the test data.

The X-axis is the predicted labels based on the outcome of the model, and Y-axis is the ground truth (True) label based on domain expert labels.

The diagonal of the matrix shows the correct classification result for each class. The off-diagonal values represent the misclassified labels.

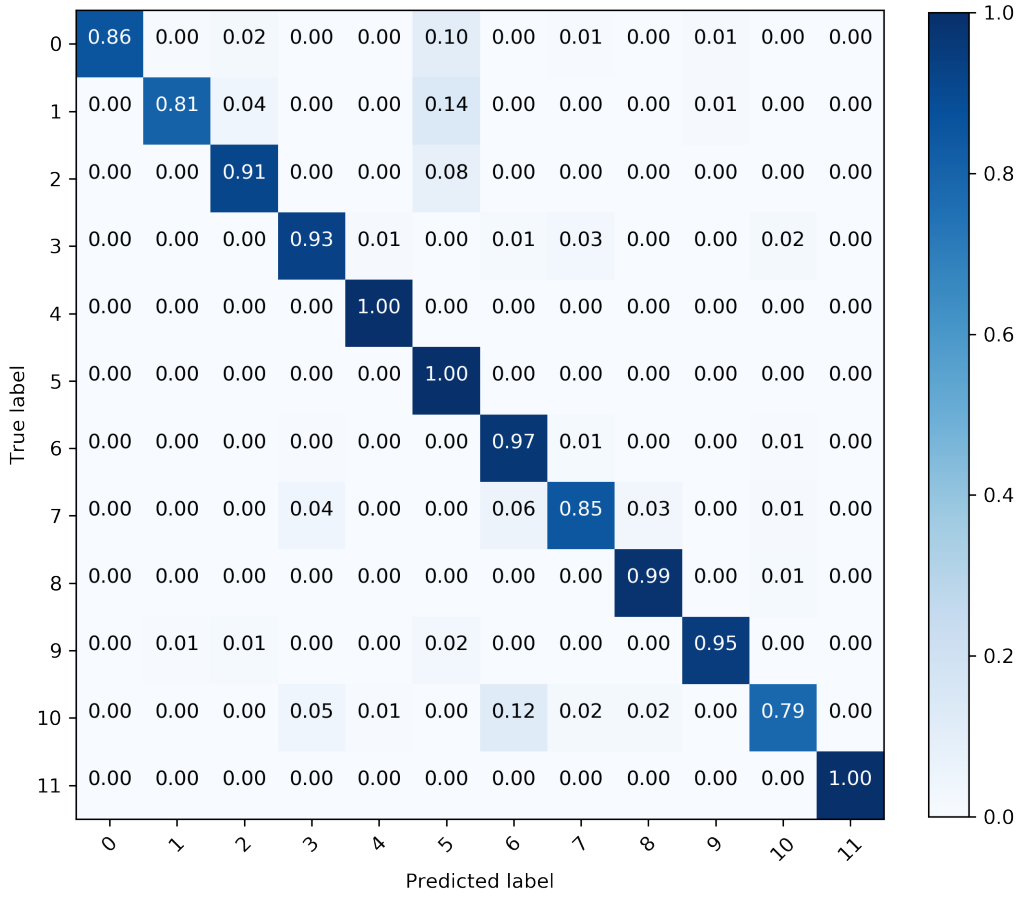


Figure 4.2: Normalized confusion matrix

Bibliography

[1] A. Ahmadi, S. Jin, M. C. Smith, E. R. Collins, and A. Goudarzi. Parallel power flow based on openmp. In *2018 North American Power Symposium (NAPS)*, pages 1–6, 2018.

[2] Senjian An, Wanquan Liu, and Svetha Venkatesh. Fast cross-validation algorithms for least squares support vector machine and kernel ridge regression. *Pattern Recognition*, 40(8):2154–2162, 2007.

[3] S Asharaf and M Narasimha Murty. Scalable non-linear support vector machine using hierarchical clustering. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 1, pages 908–911. IEEE, 2006.

[4] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Karl Rupp, Barry F. Smith, Stefano Zampini, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.7, Argonne National Laboratory, 2016.

[5] Yukun Bao, Zhongyi Hu, and Tao Xiong. A pso and pattern search based memetic algorithm for svms parameters optimization. *Neurocomputing*, 117:98–106, 2013.

[6] Omid Bazgir, Seyed Amir Hassan Habibi, Lorenzo Palma, Paola Pierleoni, and Saba Nafees. A classification system for assessment and home monitoring of tremor in patients with parkinson's disease. *Journal of medical signals and sensors*, 8(2):65, 2018.

[7] Omid Bazgir, Zeynab Mohammadi, and Seyed Amir Hassan Habibi. Emotion recognition with machine learning using eeg signals. In *2018 25th National and 3rd International Iranian Conference on Biomedical Engineering (ICBME)*, pages 1–5. IEEE, 2018.

[8] Michael Berry, Thomas E. Potok, Prasanna Balaprakash, Hank Hoffmann, Raju Vatsavai, and Prabhat. Machine Learning and Understandingfor Intelligent Extreme Scale Scientific Computing and Discovery. Technical Report 15-CS-1768, ASCR DOE Workshop Report, 2015.

[9] Olivier Bodenreider. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl_1):D267–D270, 2004.

[10] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Gorke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering*, 20(2):172–188, 2008.

[11] A. Brandt and D. Ron. Chapter 1 : Multigrid solvers and multilevel optimization strategies. In J. Cong and J. R. Shinnerl, editors, *Multilevel Optimization and VLSICAD*. Kluwer, 2003.

[12] J Brannick, M Brezina, S MacLachlan, T Manteuffel, S McCormick, and J Ruge. An energy-based amg coarsening strategy. *Numerical linear algebra with applications*, 13(2-3):133–148, 2006.

[13] Leo Breiman. Bias, variance, and arcing classifiers (technical report 460). *Statistics Department, University of California*, 1996.

[14] Aydin Buluç, Henning Meyerhenke, Ilya Safro, Peter Sanders, and Christian Schulz. Recent advances in graph partitioning. *Algorithm Engineering: Selected Results and Surveys. LNCS 9220, Springer-Verlag*, 2016.

[15] Li Juan Cao, S Sathiya Keerthi, Chong-Jin Ong, Jian Qiu Zhang, and Henry P Lee. Parallel sequential minimal optimization for the training of support vector machines. *Neural Networks, IEEE Transactions on*, 17(4):1039–1049, 2006.

[16] Gavin C Cawley and Nicola LC Talbot. On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, 11(Jul):2079–2107, 2010.

[17] 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*, 2011.

[18] Olivier Chapelle, Vladimir Vapnik, Olivier Bousquet, and Sayan Mukherjee. Choosing multiple parameters for support vector machines. *Machine learning*, 46(1):131–159, 2002.

[19] Jie Chen and Ilya Safro. Algebraic distance on graphs. *SIAM J. Scientific Computing*, 33(6):3468–3490, 2011.

[20] Jie Chen and Ilya Safro. A measure of the local connectivity between graph vertices. *Procedia Computer Science*, 4:196–205, 2011.

[21] Sungmoon Cheong, Sang Hoon Oh, and Soo-Young Lee. Support vector machines with binary tree architecture for multi-class classification. *Neural Information Processing-Letters and Reviews*, 2(3):47–51, 2004.

[22] Cédric Chevalier and Ilya Safro. Comparison of coarsening schemes for multilevel graph partitioning. *Learning and Intelligent Optimization*, pages 191–205, 2009.

[23] Marc Claesen, Frank De Smet, Johan AK Suykens, and Bart De Moor. Ensemblesvm: a library for ensemble learning using support vector machines. *Journal of Machine Learning Research*, 15(1):141–145, 2014.

[24] Kristof Coussement and Dirk Van den Poel. Churn prediction in subscription services: An application of support vector machines while comparing two parameter-selection techniques. *Expert systems with applications*, 34(1):313–327, 2008.

[25] Changjianand Li Wanliand Tan Ludanand Peng Yuxing Cui, Lijuanand Wang. *Multi-Modes Cascade SVMs: Fast Support Vector Machines in Distributed System*, pages 443–450. Springer Singapore, Singapore, 2017.

[26] Pasqua D'Ambra, Luisa Cutillo, and Panayot S Vassilevski. Bootstrap amg for spectral clustering. *Computational and Mathematical Methods*, 1(2):e1020, 2019.

[27] I. Dhillon, Y. Guan, and B. Kulis. A fast kernel-based multilevel algorithm for graph clustering. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'05)*, pages 629–634. ACM Press, 2005.

[28] Tom Dietterich. Overfitting and undercomputing in machine learning. *ACM computing surveys (CSUR)*, 27(3):326–327, 1995.

[29] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874, 2008.

[30] Rong-En Fan, Pai-Hsuen Chen, and Chih-Jen Lin. Working set selection using second order information for training support vector machines. *The Journal of Machine Learning Research*, 6:1889–1918, 2005.

[31] Haw-ren Fang, Sophia Sakellaridi, and Yousef Saad. Multilevel manifold learning with application to spectral clustering. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 419–428. ACM, 2010.

[32] A. Frank and A. Asuncion. UCI machine learning repository. *[http://archive. ics. uci.edu/ml]. irvine, ca: University of california," School of Information and Computer Science"*, vol. 213, 2010.

[33] Raoul Frijters, Marianne van Vugt, Ruben Smeets, René van Schaik, Jacob de Vlieg, and Wynand Alkema. Literature mining for the discovery of hidden connections between drugs, genes and diseases. *PLOS Computational Biology*, 6(9):1–11, 09 2010.

[34] Vishrawas Gopalakrishnan, Kishlay Jha, Wei Jin, and Aidong Zhang. A survey on literature based discovery approaches in biomedical domain. *Journal of Biomedical Informatics*, 93:103141, 2019.

[35] Leo Grady and Eric L. Schwartz. Isoperimetric graph partitioning for image segmentation. *IEEE Trans. on Pat. Anal. and Mach. Int*, 28:469–475, 2006.

[36] Hans P Graf, Eric Cosatto, Leon Bottou, Igor Dourdanovic, and Vladimir Vapnik. Parallel support vector machines: The cascade SVM. In *Advances in neural information processing systems*, pages 521–528, 2004.

[37] Christopher Gropp, Alexander Herzog, Ilya Safro, Paul W Wilson, and Amy W Apon. Clustered latent dirichlet allocation for scientific discovery. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 4503–4511. IEEE, 2019.

[38] William W Hager, James T Hungerford, and Ilya Safro. A multilevel bilinear programming algorithm for the vertex separator problem. *Computational Optimization and Applications*, 69(1):189–223, 2018.

[39] Pei-Yi Hao, Jung-Hsien Chiang, and Yi-Kun Tu. Hierarchically svm classification based on support vector clustering method and its application to document categorization. *Expert Systems with applications*, 33(3):627–635, 2007.

[40] Sam Henry and Bridget T. McInnes. Literature based discovery: Models, methods, and trends. *Journal of Biomedical Informatics*, 74:20 – 32, 2017.

[41] Shi-Jinn Horng, Ming-Yang Su, Yuan-Hsin Chen, Tzong-Wann Kao, Rong-Jian Chen, Jui-Lin Lai, and Citra Dwi Perkasa. A novel intrusion detection system based on hierarchical clustering and support vector machines. *Expert Systems with Applications*, 38(1):306 – 313, 2011.

[42] Cho-Jui Hsieh, Si Si, and Inderjit Dhillon. A divide-and-conquer solver for kernel support vector machines. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 566–574, Bejing, China, 22–24 Jun 2014. PMLR.

[43] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *IEEE transactions on Neural Networks*, 13(2):415–425, 2002.

[44] C.M. Huang, Y.J. Lee, D.K.J. Lin, and S.Y. Huang. Model selection for support vector machines via uniform design. *Computational Statistics & Data Analysis*, 52(1):335–346, 2007.

[45] Sun Huh. Promotion to medline, indexing with medical subject headings, and open data policy for the journal of educational evaluation for health professions. *Journal of Educational Evaluation for Health Professions*, 13, 2016.

[46] Hosagrahar Visvesva Jagadish. Big data and science: Myths and reality. *Big Data Research*, 2(2):49–52, 2015.

[47] Thorsten Joachims. Making large scale svm learning practical. Technical report, Universität Dortmund, 1999.

[48] Emmanuel John and Ilya Safro. Single-and multi-level network sparsification by algebraic distance. *Journal of Complex Networks*, 5(3):352–388, 2017.

[49] George Karypis, Eui-Hong Han, and Vipin Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, 1999.

[50] George Karypis and Vipin Kumar. *MeTiS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices, Version 4.0*. University of Minnesota, Minneapolis, MN, 1998.

[51] Andrej Kastrin and Dimitar Hristovski. Disentangling the evolution of medline bibliographic database: A complex network perspective. *Journal of biomedical informatics*, 89:101–113, 2019.

[52] Latifur Khan, Mamoun Awad, and Bhavani Thuraisingham. A new intrusion detection system using support vector machines and hierarchical clustering. *The VLDB Journal*, 16(4):507–521, October 2007.

[53] Wael Khreich, Eric Granger, Ali Miri, and Robert Sabourin. Iterative boolean combination of classifiers in the roc space: an application to anomaly detection with hmms. *Pattern Recognition*, 43(8):2732–2752, 2010.

[54] Halil Kilicoglu, Graciela Rosemblat, Marcelo Fiszman, and Thomas C Rindflesch. Constructing a semantic predication gold standard from the biomedical literature. *BMC bioinformatics*, 12(1):486, 2011.

[55] Halil Kilicoglu, Dongwook Shin, Marcelo Fiszman, Graciela Rosemblat, and Thomas C Rindflesch. Semmeddb: a pubmed-scale repository of biomedical semantic predications. *Bioinformatics*, 28(23):3158–3160, 2012.

[56] Dan Kushnir, Meirav Galun, and Achi Brandt. Fast multiscale clustering and manifold identification. *Pattern Recognition*, 39(10):1876–1891, 2006.

[57] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th annual international conference on machine learning*, pages 609–616. ACM, 2009.

[58] Stefan Lessmann, Robert Stahlbock, and Sven F Crone. Genetic algorithms for support vector machine model selection. In *Neural Networks, 2006. IJCNN'06. International Joint Conference on*, pages 3063–3069. IEEE, 2006.

[59] Sven Leyffer and Ilya Safro. Fast response to infection spread and cyber attacks on large-scale networks. *Journal of Complex Networks*, 1(2):183–199, 2013.

[60] Tao Li, Xuechen Liu, Qiankun Dong, Wenjing Ma, and Kai Wang. Hpsvm: Heterogeneous parallel svm with factorization based ipm algorithm on cpu-gpu cluster. In *2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, pages 74–81. IEEE, 2016.

[61] M. Lichman. UCI machine learning repository, 2013.

[62] Chun-Fu Lin and Sheng-De Wang. Fuzzy support vector machines. *Neural Networks, IEEE Transactions on*, 13(2):464–471, 2002.

[63] Hsuan-Tien Lin, Chih-Jen Lin, and Ruby C Weng. A note on platt's probabilistic outputs for support vector machines. *Machine learning*, 68(3):267–276, 2007.

[64] Shih-Wei Lin, Zne-Jung Lee, Shih-Chieh Chen, and Tsung-Yuan Tseng. Parameter determination of support vector machine and feature selection using simulated annealing approach. *Applied soft computing*, 8(4):1505–1512, 2008.

[65] Victoria López, Sara del Río, José Manuel Benítez, and Francisco Herrera. Cost-sensitive linguistic fuzzy rule based classification systems under the mapreduce framework for imbalanced big data. *Fuzzy Sets and Systems*, 258:5–38, 2015.

[66] PietroGiorgio Lovaglio and Giorgio Vittadini. Multilevel dimensionality-reduction methods. *Statistical Methods & Applications*, 22(2):183–207, 2013.

[67] Jan Luts, Fabian Ojeda, Raf Van de Plas, Bart De Moor, Sabine Van Huffel, and Johan AK Suykens. A tutorial on support vector machine-based methods for classification problems in chemometrics. *Analytica Chimica Acta*, 665(2):129–145, 2010.

[68] Maciej A Mazurowski, Piotr A Habas, Jacek M Zurada, Joseph Y Lo, Jay A Baker, and Georgia D Tourassi. Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance. *Neural networks*, 21(2):427–436, 2008.

[69] Sanjay Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on optimization*, 2(4):575–601, 1992.

[70] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09)*, pages 331–340. INSTICC Press, 2009.

[71] Marius Muja and David G Lowe. Scalable nearest neighbor algorithms for high dimensional data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2227–2240, 2014.

[72] Andreas Noack and Randolf Rotta. Multi-level algorithms for modularity clustering. In Jan Vahrenhold, editor, *Experimental Algorithms*, volume 5526 of *Lecture Notes in Computer Science*, pages 257–268. Springer Berlin Heidelberg, 2009.

[73] Edgar Osuna, Robert Freund, and Federico Girosi. An improved training algorithm for support vector machines. In *Neural Networks for Signal Processing [1997] VII. Proceedings of the 1997 IEEE Workshop*, pages 276–285. IEEE, 1997.

[74] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[75] John C Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in kernel methods*, pages 185–208. MIT press, 1999.

[76] Wanda Pratt and Meliha Yetisgen-Yildiz. Litlinker: capturing connections across the biomedical literature. In *Proceedings of the 2nd international conference on Knowledge capture*, pages 105–112. ACM, 2003.

[77] Raphael Puget and Nicolas Baskiotis. Hierarchical label partitioning for large scale classification. In *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on*, pages 1–10. IEEE, 2015.

[78] Adrian E. Raftery. A model for high-order markov chains. *Journal of the Royal Statistical Society. Series B (Methodological)*, 47(3):528–539, 1985.

[79] Talayeh Razzaghi, Oleg Roderick, Ilya Safro, and Nicholas Marko. Multilevel weighted support vector machine for classification on healthcare data with missing values. *PloS one*, 11(5):e0155119, 2016.

[80] Talayeh Razzaghi and Ilya Safro. Scalable multilevel support vector machines. In *International Conference on Computational Science (ICCS), Procedia Computer Science*, volume 51, pages 2683–2687. Elsevier, 2015.

[81] Thomas C Rindflesch and Marcelo Fiszman. The interaction of domain knowledge and linguistic structure in natural language processing: interpreting hypernymic propositions in biomedical text. *Journal of biomedical informatics*, 36(6):462–477, 2003.

[82] Dorit Ron, Ilya Safro, and Achi Brandt. A fast multigrid algorithm for energy minimization under planar density constraints. *Multiscale Modeling & Simulation*, 8(5):1599–1620, 2010.

[83] Dorit Ron, Ilya Safro, and Achi Brandt. Relaxation-based coarsening and multiscale graph organization. *Multiscale Modeling & Simulation*, 9(1):407–423, 2011.

[84] Randolf Rotta and Andreas Noack. Multilevel local search algorithms for modularity clustering. *Journal of Experimental Algorithmics (JEA)*, 16:2–3, 2011.

[85] Ehsan Sadrfaridpour, Sandeep Jeereddy, Ken Kennedy, Andre Luckow, Talayeh Razzaghi, and Ilya Safro. Algebraic multigrid support vector machines. *accepted in European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN), arXiv preprint arXiv:1611.05487*, 2017.

[86] Ehsan Sadrfaridpour, Talayeh Razzaghi, and Ilya Safro. Engineering fast multilevel support vector machines. *Machine Learning*, pages 1–39, 2019.

[87] Ehsan Sadrfaridpour, Jeereddy Sandeep, Ken Kennedy, Andre Luckow, Talayeh Razzaghi, and Ilya Safro. Algebraic multigrid support vector machines. In *ESANN 2017 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, Bruges, Belgium, 2017. ESANN.

[88] Ilya Safro, Dorit Ron, and Achi Brandt. Graph minimum linear arrangement by multilevel weighted edge contractions. *Journal of Algorithms*, 60(1):24–41, 2006.

[89] Ilya Safro, Dorit Ron, and Achi Brandt. A multilevel algorithm for the minimum 2-sum problem. *J. Graph Algorithms Appl.*, 10(2):237–258, 2006.

[90] Ilya Safro, Dorit Ron, and Achi Brandt. Multilevel algorithms for linear ordering problems. *ACM Journal of Experimental Algorithmics*, 13, 2008.

[91] Ilya Safro, Peter Sanders, and Christian Schulz. Advanced coarsening schemes for graph partitioning. In Ralf Klasing, editor, *Experimental Algorithms*, pages 369–380, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[92] Ilya Safro and Boris Temkin. Multiscale approach for the network compression-friendly ordering. *J. Discrete Algorithms*, 9(2):190–202, 2011.

[93] Peter Sanders and Christian Schulz. Engineering multilevel graph partitioning algorithms. In *European Symposium on Algorithms*, pages 469–480. Springer, 2011.

[94] Sebastian Schlag, Matthias Schmitt, and Christian Schulz. Faster support vector machines. In *2019 Proceedings of the Twenty-First Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 199–210. SIAM, 2019.

[95] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[96] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011.

[97] Claude Elwood Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.

[98] Eitan Sharon, Meirav Galun, Dahlia Sharon, Ronen Basri, and Achi Brandt. Hierarchy and adaptivity in segmenting visual scenes. *Nature*, 442(7104):810–813, 2006.

[99] Ruslan Shaydulin, Jie Chen, and Ilya Safro. Relaxation-based coarsening for multilevel hypergraph partitioning. *Multiscale Modeling & Simulation*, 17(1):482–506, 2019.

[100] Ruslan Shaydulin and Ilya Safro. Aggregative Coarsening for Multilevel Hypergraph Partitioning. In Gianlorenzo D'Angelo, editor, *17th International Symposium on Experimental Algorithms (SEA 2018)*, volume 103 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 2:1–2:15, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[101] U Martin Singh-Blom, Nagarajan Natarajan, Ambuj Tewari, John O Woods, Inderjit S Dhillon, and Edward M Marcotte. Prediction and validation of gene-disease associations using methods inspired by social network analyses. *PloS one*, 8(5), 2013.

[102] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information processing & management*, 45(4):427–437, 2009.

[103] Scott Spangler. *Accelerating Discovery: Mining Unstructured Information for Hypothesis Generation*, volume 37. CRC Press, 2015.

[104] Carl Staelin. Parameter selection for support vector machines. *Hewlett-Packard Company, Tech. Rep. HPL-2002-354R1*, 2003.

[105] Yanmin Sun, Mohamed S Kamel, Andrew KC Wong, and Yang Wang. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12):3358–3378, 2007.

[106] Don R Swanson. Fish oil, raynaud's syndrome, and undiscovered public knowledge. *Perspectives in biology and medicine*, 30(1):7–18, 1986.

[107] Don R Swanson. Migraine and magnesium: eleven neglected connections. *Perspectives in biology and medicine*, 31(4):526–557, 1988.

[108] J. Sybrandt, M. Shtutman, and I. Safro. Large-scale validation of hypothesis generation systems via candidate ranking. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 1494–1503, Dec 2018.

[109] Justin Sybrandt, Angelo Carrabba, Alexander Herzog, and Ilya Safro. Are abstracts enough for hypothesis generation? In *2018 IEEE International Conference on Big Data (Big Data)*, pages 1504–1513. IEEE, 2018.

[110] Justin Sybrandt, Michael Shtutman, and Ilya Safro. Moliere: Automatic biomedical hypothesis generation system. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1633–1642. ACM, 2017.

[111] Justin Sybrandt, Ilya Tyagin, Michael Shtutman, and Ilya Safro. Agatha: Automatic graph-mining and transformer based hypothesis generation approach. *arXiv preprint arXiv:2002.05635*, 2020.

[112] Mahbod Tavallaee, Natalia Stakhanova, and Ali Akbar Ghorbani. Toward credible evaluation of anomaly-based intrusion-detection methods. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 40(5):516–524, 2010.

[113] Francis EH Tay and Lijuan Cao. Application of support vector machines in financial time series forecasting. *Omega: The International Journal of Management Science*, 29(4):309–317, 2001.

[114] Ulrich Trottenberg and Anton Schuller. *Multigrid*. Academic Press, Orlando, FL, 2001.

[115] Andrew Tulloch. Fast randomized svd. *Blog Post*, 2014.

[116] Jan Vaněk, Josef Michálek, and Josef Psutka. A gpu-architecture optimized hierarchical decomposition algorithm for support vector machine training. *IEEE Transactions on Parallel and Distributed Systems*, 28(12):3330–3343, 2017.

[117] Wytze J Vlietstra, Rein Vos, Anneke M Sijbers, Erik M van Mulligen, and Jan A Kors. Using predicate and provenance information from a knowledge graph for drug efficacy screening. *Journal of biomedical semantics*, 9(1):23, 2018.

[118] Lei Wang. Feature selection with kernel class separability. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(9):1534–1546, 2008.

[119] Yanli Wang, Tugba Suzek, Jian Zhang, Jiyao Wang, Siqian He, Tiejun Cheng, Benjamin A Shoemaker, Asta Gindulyte, and Stephen H Bryant. Pubchem bioassay: 2014 update. *Nucleic acids research*, 42(D1):D1075–D1082, 2014.

[120] Xing Wei and W Bruce Croft. Lda-based document models for ad-hoc retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 178–185. ACM, 2006.

[121] Zeyi Wen, Jiashuai Shi, Qinbin Li, Bingsheng He, and Jian Chen. Thundersvm: A fast svm library on gpus and cpus. *The Journal of Machine Learning Research*, 19(1):797–801, 2018.

[122] Qiang Wu and Ding-Xuan Zhou. Svm soft margin classifiers: linear programming versus quadratic programming. *Neural computation*, 17(5):1160–1187, 2005.

[123] Taijia Xiao, Dong Ren, Shuanghui Lei, Junqiao Zhang, and Xiaobo Liu. Based on grid-search and pso parameter optimization for support vector machine. In *Proceeding of the 11th World Congress on Intelligent Control and Automation*, pages 1529–1533. IEEE, 2014.

[124] Z Yang, WH Tang, Almas Shintemirov, and QH Wu. Association rule mining-based dissolved gas analysis for fault diagnosis of power transformers. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 39(6):597–610, 2009.

[125] Yang You, James Demmel, Kenneth Czechowski, Le Song, and Richard Vuduc. Ca-svm: Communication-avoiding support vector machines on distributed systems. In *Parallel and Distributed Processing Symposium (IPDPS), 2015 IEEE International*, pages 847–859. IEEE, 2015.

[126] Yang You, Haohuan Fu, Shuaiwen Leon Song, Amanda Randles, Darren Kerbyson, Andres Marquez, Guangwen Yang, and Adolfy Hoisie. Scaling support vector machines on modern hpc platforms. *Journal of Parallel and Distributed Computing*, 76:16–31, 2015.

[127] Hwanjo Yu, Jiong Yang, and Jiawei Han. Classifying large data sets using svms with hierarchical clusters. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 306–315. ACM, 2003.

[128] XiaoLi Zhang, XueFeng Chen, and ZhengJia He. An aco-based algorithm for parameter optimization of support vector machines. *Expert Systems with Applications*, 37(9):6618–6628, 2010.

[129] Ligang Zhou, Kin Keung Lai, and Lean Yu. Credit scoring using support vector machines with direct search for parameters selection. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 13(2):149–155, 2009.

[130] Kaihua Zhu, Hao Wang, Hongjie Bai, Jian Li, Zhihuan Qiu, Hang Cui, and Edward Y Chang. Parallelizing support vector machines on distributed computers. In *Advances in Neural Information Processing Systems*, pages 257–264, 2008.

[131] Zeyuan Allen Zhu, Weizhu Chen, Gang Wang, Chenguang Zhu, and Zheng Chen. P-packsvm: Parallel primal gradient descent kernel svm. In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*, pages 677–686. IEEE, 2009.

[132] Zhi-Bo Zhu and Zhi-Huan Song. Fault diagnosis based on imbalance modified kernel fisher discriminant analysis. *Chemical Engineering Research and Design*, 88(8):936–951, 2010.