NOVEL SUPPORT VECTOR MACHINES FOR DIVERSE LEARNING PARADIGMS

A Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at Virginia Commonwealth University.

by

GABRIELLA ANGELA MELKI

Ph.D. Candidate

Director: Alberto Cano,

Assistant Professor, Department of Computer Science,

Virginia Commonwealth University

Director: Sebastian Ventura,

Professor, Department of Computer Science & Numerical Analysis,

University of Córdoba

Virginia Commonwealth University

Richmond, Virginia

September 2018

Abstract

NOVEL SUPPORT VECTOR MACHINES FOR DIVERSE LEARNING PARADIGMS

This dissertation introduces novel support vector machines (SVM) for the traditional and non-traditional learning paradigms: *Online Learning*, *Multi-Target* Regression, *Multiple-Instance* classification, and *Data Stream* classification.

Three multi-target support vector regression (SVR) models are first presented. The first involves building independent, single-target SVR models for each target. The second builds an ensemble of random chains using the first single-target method as a base model. The third calculates the targets' correlations and forms a maximum correlation chain, which is used to build a single chained SVR model, improving the model's prediction performance, while reducing computational complexity.

Under the multi-instance paradigm, a novel SVM multiple-instance formulation and an algorithm with a bag-representative selector, named MIRSVM, are presented. The contribution trains the SVM based on bag-level information and is able to identify instances that highly impact classification, i.e. bag-representatives, for both positive and negative bags, while finding the optimal class separation hyperplane. Unlike other multi-instance SVM methods, this approach eliminates possible class imbalance issues by allowing both positive and negative bags to have at most one representative, which constitute as the most contributing instances to the model.

Add OLLAWV and DS Application

Rigorous experimental studies and statistical analyses over various metrics and datasets were conducted in order to comprehensively compare the proposed solutions against modern, widely-used methods from both paradigms. The experimental study and analysis confirms that the proposals achieve better performances and more scalable solutions than the methods compared, making them competitive in their respected fields.

CHAPTER 1

INTRODUCTION

In traditional classification and regression problems, learning algorithms attempt to correctly predict unknown samples by finding patterns between training samples and their outputs. Identifying these patterns is a non-trivial task due to many factors such as the high dimensionality of the data, as well as the dataset size.

Over the past decade, dataset sizes have grown disproportionately to the speed of processors and memory capacity, limiting machine learning methods to computational time. Many real-world applications, such as human activity recognition, operations research, and video/signal processing, require algorithms that are scalable and accurate, while being able to provide insightful information in a timely fashion.

More recently, these traditional methods have been extended to accommodate various types of data paradigms. Examples include *Multiple Target* (MT) learning and *Multiple Instance* (MI) learning. These emerging paradigms require algorithms to be robust and accommodate non-traditional data representations.

Multi-target learning is a challenging task that consists of creating predictive models for problems with multiple outputs. Specifically, MT learning is an approach to transfer learning that improves generalization by using domain information present in training data of multiple related targets as an inductive bias. This is done by learning in parallel, and what is learned for each target can be used to help other targets be learned better [2, 10, 42]. MT learning includes *multi-target regression* (MTR), which addresses the prediction of continuous targets, *multi-label classification* [49] which focuses on binary targets, and *multi-dimensional classification* which describes the prediction of discrete targets [5].

Multi-target prediction has the capacity to generate models representing a wide variety of real-world applications, ranging from natural language processing [22] to bioinformat-

ics [31]. A characteristic of the MT datasets used in these applications is that they are generated by a single system, indicating that the nature of the outputs captured has some structure. Even though modeling the multi-variate nature and possible complex relationships between the target variables is challenging, they are more accurately represented by a multi-target model [10, 16].

Several methods have been proposed for solving such multi-target tasks and can be categorized into two groups. The first being *problem transformation* methods in which the multi-target problem is transformed into multiple single-target (ST) problems, each solved separately using standard classification and regression algorithms. The second being *algorithm adaptation* methods which adapt existing traditional algorithms to predict all the target variables simultaneously [5]. Using *problem transformation* algorithms for a domain of t target variables, t predictive models must be constructed, each predicting a single-target variable. Prediction for an unseen sample would be obtained by running each of the t single-target models and concatenating their results. Conversely, when using *algorithm adaptation* algorithms for the same domain of t target variables, a single model would need to be constructed which would output all t predictions.

It is known that *algorithm adaptation* methods outperform *problem transformation* methods. The most valuable advantage of using multi-target techniques is that, not only are the relationships between the sample variables and the targets exploited, but the relationships between the targets amongst themselves are as well [2, 10]. Single-target techniques, on the other hand, eliminate any possibility of learning from the possible relationships between the target variables because a single, independent model is trained for each target separately. Another advantage of MT techniques is model interpretability [3]. A single multi-target model is highly more interpretable than a series of single-target models. Not only is a single MT model more interpretable, but it could also be considerably more computationally efficient to train, rather than training multiple single-target models individually [17].

Multi-instance learning (MIL) is a generalization of *supervised learning* that has been

recently been gaining interest because of its applicability to many real-world problems such as image classification and annotation [20], human action recognition [48], and drug activity prediction [14]. The difference between MIL and traditional learning is the nature of the data. In the multi-instance setting, a sample is considered a *bag* that contains multiple *instances* and is associated with a single label. The individual instance labels within a bag are unknown and bag labels are assigned based on a multi-instance assumption, or hypothesis. Introduced by Dietterich et. al. [14], the standard MI assumption states that a bag is labeled positive if and only if it contains at least one positive instance. Other hypotheses and frameworks have been proposed by Foulds and Frank [18] to encompass a wider range of applications with MI data, but for the scope of this thesis, we will focus on the standard MI assumption.

One of the major complexities associated with MIL is the ambiguity of the relationship between a bag label and the instances within the bag. This stems from the standard MI assumption, where the underlying distribution among instances within positive bags is unknown. There have been different attempts to overcome this complexity, such as "flattening" the MIL datasets, meaning instances contained in positive bags each adopt a positive label, allowing the use of classical supervised learning techniques [36]. This approach assumes that positive bags contain a significant number of positive instances, which may not be the case, causing the classifier to mislabel negative instances within the bag, decreasing the power of the MI model. To overcome this issue, a different MIL approach was proposed, where subsets of instances are selected from positive bags for classifier training [32]. One drawback of this type of method is that the resulting training datasets become imbalanced towards positive instances. Model performance further deteriorates when more instances are selected as subsets than needed [9].

Our MTR and MIL proposals aim to deal with these drawbacks that exist in both paradigms using support vector machines. Support vector machines (SVM), proposed by Cortes and Vapnik [13], represent popular linear and non-linear (kernelized) learning algorithms based on the idea of a large-margin classifier. They have been shown to improve

generalization performance for binary classification problems. SVMs are similar to other machine learning techniques, but literature shows that they usually outperform them in terms of scalability, computational efficiency, and robustness against outliers. They are known for creating sparse and non-linear classifiers, making them suitable for handling large datasets. A traditional approach for training SVMs is the *Sequential Minimal Optimization* (SMO) algorithm [35], a method for solving the L1-SVM's Quadratic Programming (QP) task.

In this thesis, various approaches are devised for solving machine learning problems for various types of learning paradigms using traditional support vector machine solvers. Specifically, novel algorithms are proposed for solving multi-target regression (a subset of multi-target learning) and multi-instance classification problems.

1.1 Contributions of the Proposal

The current leading MT models are based on ensembles of regressor chains, where random, differently ordered chains of the target variables are created and used to build separate regression models, using the previous target predictions in the chain. The challenges of building MT models stem from trying to capture and exploit possible correlations among the target variables during training, at the expense of increasing the computational complexity of model training. One of the contributions of this proposal aims to investigate the performance changes when building a regression model using two distinct *algorithm adaptation* chaining methods versus building independent single-target models for each target variable using a novel framework. Specifically, this MTR contribution includes:

- Evaluating the performance of a Support Vector Regressor (SVR) as a multi-target to single-target *problem transformation* method to determine whether it outperforms current popular ST algorithms. Its performance is analyzed as a base-line model for MT chaining methods due to the fact that ST methods do not account for any correlation among the target variables.

- Building an MT ensemble of randomly chained SVR models (SVRRC), an *algorithm adaptation* approach, inspired by the chaining classification method, Ensemble of Random Chains Corrected (ERCC) [40], to investigate the effects and advantages of exploiting correlations among target variables during model training, in the context of regression problems. The main issues to be investigated with this approach are the *randomness* of the created chains because they might not capture of correlations between the targets, as well as the time taken to build all the regressors in the ensemble.

- Proposing an MT *algorithm adaptation* model of SVRs that builds a unique chain, capturing the maximum correlation among target outputs, named SVR Correlation Chains (SVRCC). The advantages of using this approach include exploiting the correlations among the targets which leads to an improvement in model prediction performance, and a reduction in computational complexity because a single SVR-chain model is trained, rather than building an ensemble of 10 base regressors.

To address the limitations presented by MIL algorithms, this thesis proposes a novel SVM formulation with a bag-representative selector, called Multiple-Instance Representative Support Vector Machine (MIRSVM). The algorithm does not assume any distribution of the instances and is not affected by the number of instances within a bag, making it applicable to a variety of contexts. The key contributions of this work include:

- Reformulating the traditional primal L1-SVM problem to optimize over bags, rather than instances, ensuring all the information contained within each bag is utilized during training, while defining bag representative selector criteria.

- Deriving the dual multi-instance SVM problem, with the Karush-Kuhn-Tucker necessary and sufficient conditions for optimality. The dual is maximized with respect to the Lagrange multipliers and provides insightful information about the resulting sparse model. The dual formulation is kernelized with a Gaussian radial basis function, which calculates the distances between bag representatives.

- Devising a unique bag-representative selection method that makes no presumptions about the underlying distributions of the instances within each bag, while maintaining the default MI assumption. This approach eliminates the issue of class imbalance caused by techniques such as flattening or subsetting positive instances from each bag. The key feature of MIRSVM is its ability to identify instances (support vectors) within positive and negative bags that highly impact the model.

CHAPTER 2

BACKGROUND

Support vector machines represent a popular set of learning techniques that have been introduced under Vapnik-Chervonenkis theory of *structured risk minimization* (SRM) [6, 13, 24, 37, 38]. SRM is an inductive principle for the purpose of model selection. It minimizes the expected probability of error, resulting in a generalized model, without making assumptions about the data distribution [38, 46]. This is the basis for developing the maximal margin classifier [46]. Based on the work of Aizerman et. al. [1], Boser et. al. [6] generalized the linear algorithm to the non-linear case. Then, Cortes and Vapnik [13] proposed the soft margin SVM; a modification that not only allowed maximal margin classifiers to be applied to non-linearly separable data, but also introduced a regularization parameter to prevent overfitting and gauge generalizability. That same year, the algorithm was extended by Vapnik et. al. [47] to the regression case.

This chapter presents a theoretical background of support vector machines. First, the SVM paradigm is discussed in the context of classification, introducing the concepts of the linear *Hard Margin* classifier, and the nonlinear *Soft Margin SVM*. Next, the concept of Support Vector Regression (SVR) is introduced. Afterwards, popular methods for solving the SVM problem are presented and their advantages and problems are discussed.

2.1 Support Vector Machine Classification

Supervised learning is the process of determining a relationship $f(\boldsymbol{x})$ by using a training dataset $\mathcal{S} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_i, y_i), \ldots, (\boldsymbol{x}_n, y_n)\}$, which contains n inputs of d-dimensionality, $\boldsymbol{x}_i \in \mathbb{R}^d$, and their class labels y_i. In the case of binary classification, $y_i \in \{+1, -1\}$, where $+1$ and -1 are the two class labels.

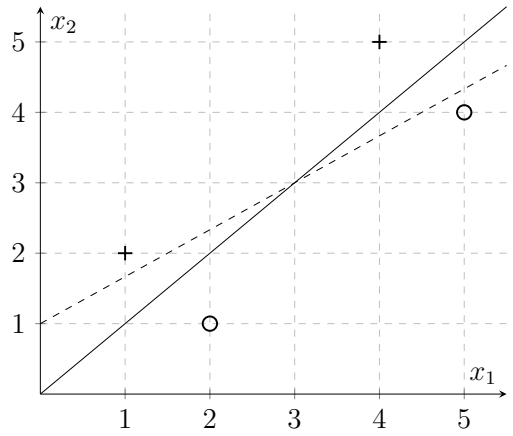

Fig. 2.1.: A 2-dimensional example of different possible separating hyperplanes that correctly classify all the toy data points.

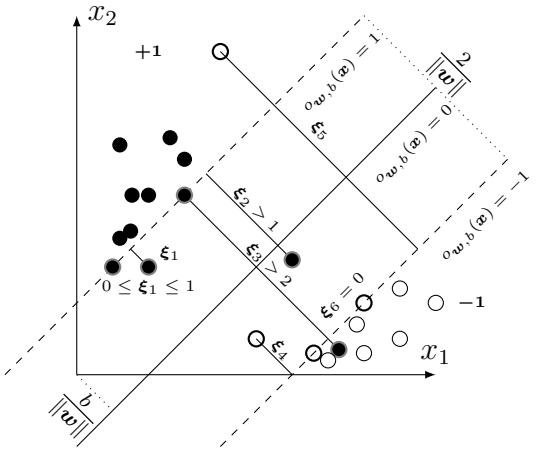

Fig. 2.2.: An illustration of the soft margin SVM solution on an example 2-dimensional non-linearly separable dataset.

The goal of the soft margin SVM classifier is to find a classification function

$$f(\boldsymbol{x}) = \text{SIGN } o_{\boldsymbol{w},b}(\boldsymbol{x}), \tag{2.1}$$

where $o_{\boldsymbol{w},b}(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x}_i + b$ is a linear decision (output) function representing an affine mapping function $o : \mathbb{R}^d \to \mathbb{R}$ and is parameterized by $\boldsymbol{w} \in \mathbb{R}^d$, the weight vector, and $b \in \mathbb{R}$, the bias term. In addition, $\boldsymbol{w}$ and b must satisfy the following,

$$y_i \left(\boldsymbol{w} \cdot \boldsymbol{x}_i + b\right) \geq 1 - \xi_i, \forall i \in \{1, \ldots, n\}, \tag{2.2}$$

where $\boldsymbol{\xi} \in \mathbb{R}^n$ are the non-negative slack variables that allow for some classification error to account for overlapping datasets. The minimal distance between points belonging to opposite classes and the hyperplane is defined as the *margin* and has a width equal to $\frac{2}{||\boldsymbol{w}||}$, which is why the $||\boldsymbol{w}||$ must be minimal in order to maximize the margin.

In the example shown in Figure 2.1, if the training data points are slightly moved, the solid line (with the larger margin) will still correctly classify all the instances, whereas the dotted line (with a much smaller margin, comparatively) will not. This illustrates that the

location of the hyperplane has a direct impact on the classifiers generalization capabilities. The hyperplane with the largest margin is called the *optimal separating hyperplane*. Figure 2.2 shows the optimal separating hyperplane for overlapping training data points, where the filled data points are from the $+1$ class and the non-filled data points are from the -1 class. The training data points on the separating hyperplane (the circled data points), whose decision function value equals $+1$ or -1, are called the *support vectors*.

The soft margin SVM problem is defined as follows,

$$\begin{aligned} \min_{(\boldsymbol{w},b)} \quad & \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{n}\xi_i \\ \text{s.t.} \quad & y_i\left(\boldsymbol{w}\cdot\boldsymbol{x}_i + b\right) \geq 1-\xi_i,\ \forall i \in \{1,\ldots,n\} \\ & \xi_i \geq 0,\ \forall i \in \{1,\ldots,n\}, \end{aligned} \tag{2.3}$$

where the penalty parameter $C \in \mathbb{R}$ controls the trade-off between margin maximization and classification error minimization, penalizing large norms and errors.

Equation 2.3 can be rewritten as a regularized loss minimization problem by representing the constraints as the Hinge loss, given by:

$$L(y_i,\, o_{(w,b)}(\boldsymbol{x}_i)) = \max\left\{0, 1 - y_i o_{(w,b)}(\boldsymbol{x}_i)\right\}, \tag{2.4}$$

which penalizes errors satisfying the following: $y_i o_{(w,b)}(\boldsymbol{x}_i) < 1$ and is a crucial element that facilitates the SVM model's sparseness. The soft margin SVM represented as a regularized loss minimization problem becomes:

$$\min_{(\boldsymbol{w},b)\in\mathcal{H}_o\times\mathbb{R}} R \;=\; \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{n} L(y_i,\, o_{(w,b)}(\boldsymbol{x}_i)), \tag{2.5}$$

where $\mathcal{H}_o$ is a general Hilbert space. To handle cases when the data are non-linearly separable, while enhancing the classifier's generalization capabilities, a kernel function can be used [1], as shown in Equation 2.6:

$$\mathcal{K}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \langle \phi\left(\boldsymbol{x}_i\right),\, \phi\left(\boldsymbol{x}_j\right)\rangle, \tag{2.6}$$

where $\phi(\cdot)$ represents a mapping function from the original feature space to a higher dimensional space. The advantage of utilizing kernels is being able to calculate the inner product in the input space rather than in the very high feature dimensional space (including the infinite dimensional ones). The SVM model output, o shown in Equation 2.7, for a given input vector $\boldsymbol{x}$ is defined by the kernel as given below:

$$o(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}_i) + b, \tag{2.7}$$

where $\alpha_i \in \mathbb{R}$ are the coefficients, or weights, of the expansion in feature space, and $b \in \mathbb{R}$ is the so-called bias term. Note that if a positive definite kernel is used, there is no need for a bias term b, but b can nevertheless be used. The two terms, $\boldsymbol{\alpha}$ and b, parametrize the SVM model. A model is called *dense* if the absolute value of all its weights are greater than 0, while a *sparse* model would be one that contains some $\alpha_i = 0$. The level of sparseness may vary, but the sparser the model, the more scalable the applications.

2.2 Support Vector Regression

The support vector machine was applied to the regression case [15, 45], maintaining all the maximal margin algorithmic features. Unlike pattern recognition problems where the desired outputs y_i are discrete, for the regression case they are continuous, real-valued, function outputs. Given training dataset $\mathcal{S} = \big\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n) \in \mathbb{R}^d \times \mathbb{R}\big\}$, where $y_i \in \mathbb{R}$ is the continuous output of input $\boldsymbol{x}_i \in \mathbb{R}^d$, the goal is to learn a function $f(\boldsymbol{x})$ with at most ϵ deviation from the true targets y_i for all the training data, while being as flat as possible. This was introduced by Vapnik's linear loss function with ϵ-insensitivity zone, illustrated in Figure 2.3 and given by:

$$|y_i - o_{(w,b)}(\boldsymbol{x_i})|_\epsilon = \begin{cases} 0 & if |y_i - o_{(w,b)}(\boldsymbol{x_i})| \leq \epsilon \\ |y_i - o_{(w,b)}(\boldsymbol{x_i})| - \epsilon & \text{otherwise.} \end{cases} \tag{2.8}$$

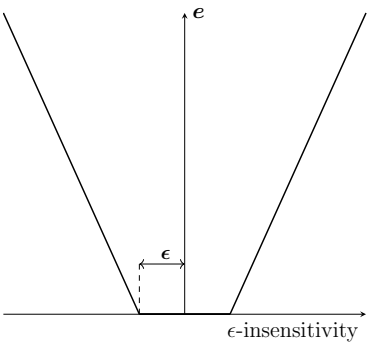

Fig. 2.3.: Vapnik's ϵ-insensitivity loss function.

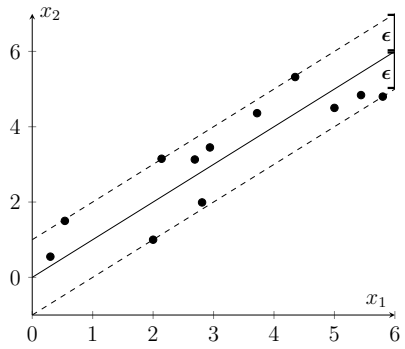

Fig. 2.4.: Support vector regression example solution.

The loss is equal to 0 if the difference between the predicted and true output values is less than ϵ. Vapnik's ϵ-insensitivity function, shown in Equation 2.8, defines an ϵ-tube, illustrated in Figure 2.4. If the predicted value is within the tube, no loss is incurred [24]. Estimating a linear regression hyperplane is achieved by minimizing:

$$\min_{(\boldsymbol{w},b)\in\mathcal{H}_o\times\mathbb{R}} R \;=\; \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{n}(|y_i - o_{(w,b)}(\boldsymbol{x_i})|_\epsilon). \tag{2.9}$$

Equation 2.9 is equivalent to the following, where non-negative slack variables are introduced:

$$\begin{aligned} \min_{(\boldsymbol{w},b,\boldsymbol{\xi},\boldsymbol{\xi}^*)} \quad & \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \\ \text{s.t.} \quad & y_i - \boldsymbol{w}\cdot\boldsymbol{x}_i - b \le \xi_i + \epsilon,\ \forall i = \{1,\ldots,n\} \\ & \boldsymbol{w}\cdot\boldsymbol{x}_i + b - y_i \le \xi_i^* + \epsilon,\ \forall i = \{1,\ldots,n\} \\ & \xi_i, \xi_i^* \ge 0,\ \forall i = \{1,\ldots,n\}. \end{aligned} \tag{2.10}$$

Note that the constant C influences the trade-off between approximation error and the model generalizability, similar to the classification setting. The optimization function given in 2.10 can be solved more easily in its dual formulation and is key to extending the SVR to learn

from non-linear functions [37]. It is as follows:

$$\begin{aligned} \max_{(\boldsymbol{\alpha},\boldsymbol{\alpha}^*)} \quad & -\frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)\mathcal{K}(\boldsymbol{x}_i,\boldsymbol{x}_j) - \epsilon\sum_{i=1}^{n}(\alpha_i+\alpha_i^*) + \sum_{i=1}^{n} y_i(\alpha_i-\alpha_i^*) \\ \text{s.t.} \quad & \sum_{i=1}^{n}(\alpha_i-\alpha_i^*) = 0,\ \alpha_i,\alpha_i^* \in [0,C],\ \forall i = \{1,\ldots,n\}, \end{aligned} \tag{2.11}$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}^*$ correspond to the SVR dual variables.

2.3 Methods for Solving the SVM Problem

THIS MAYBE SHOULD GO IN THE SVM BACKGROUND taken from Pegasos paper. Before indulging into the detailed description and analysis of Pegasos, we would like to draw connections to and put our work in context of some of the more recent work on SVM. For a more comprehensive and up-to-date overview of relevant work see the ref- erences in the papers cited below as well as the web site dedicated to kernel methods at http://www.kernel-machines.org . Due to the centrality of the SVM optimization problem, quite a few methods were devised and analyzed. The different approaches can be roughly divided into the following categories. **taken from Pegasos paper.**

- **Interior Point (IP) methods:** IP methods (see for instance [7] and the references therein) cast the SVM learning task as a quadratic optimization problem subject to linear constraints. The constraints are replaced with a barrier function. The result is a sequence of uncon- strained problems which can be optimized very efficiently using Newton or Quasi-Newton methods. The advantage of IP methods is that the depen- dence on the accuracy is double logarithmic, namely, log(log(1/)). Alas, IP methods typically require run time which is cubic in the number of examples m. Moreover, the memory requirements of IP methods are O(m2) which renders a direct use of IP methods very difficult when the training set con- sists of many examples. It should be noted that there have been several attempts to reduce the complexity based on additional assumptions (see e.g. [15]). However, the dependence on m remains super

linear. In addition, while the focus of the paper is the optimization problem cast by SVM, one needs to bear in mind that the optimization problem is a proxy method for obtaining good classification error on unseen examples. Achieving a very high accuracy in the optimization process is usually unnecessary and does not translate to a significant increase in the generalization accuracy. The time spent by IP methods for finding a sin- gle accurate solution may, for instance, be better utilized for trying different regularization values.

- **Decomposition methods:** To overcome the quadratic memory requirement of IP methods, decomposition methods such as SMO [29] and SVM-Light [20] tackle the dual representa- tion of the SVM optimization problem, and employ an active set of constraints thus working on a subset of dual variables. In the extreme case, called row-action methods [8], the active set consists of a single constraint. While algorithms in this family are fairly simple to im- plement and entertain general asymptotic convergence properties [8], the time complexity of most of the algorithms in this family is typically super linear in the training set size m. Moreover, since decomposition methods find a feasible dual solution and their goal is to maximize the dual objective function, they often result in a rather slow convergence rate to the optimum of the primal objective function. (See also the discussion in [19].)

- **Primal optimization:** Most existing approaches, including the methods discussed above, focus on the dual of Eq. (1), especially when used in conjunction with non-linear kernels. However, even when non-linear kernels are used, the Representer theorem [23] allows us to re-parametrize w as w = iyixi and cast the primal objective Eq. (1) as an un- constrained optimization problem with the variables 1 , . . . , m (see Sec. 4). Tackling the primal objective directly was studied, for example, by Chapelle [10], who considered us- ing smooth loss functions instead of the hinge loss, in which case the optimization problem becomes a smooth unconstrained optimization

problem. Chapelle then suggested using var- ious optimization approaches such as conjugate gradient descent and Newtons method. We take a similar approach here, however we cope with the non-differentiability of the hinge- loss directly by using sub-gradients instead of gradients. Another important distinction is that Chapelle views the optimization problem as a function of the variables i. In contrast, though Pegasos maintains the same set of variables, the optimization process is performed with respect to w, see Sec. 4 for details.

- **Stochastic gradient descent:** The Pegasos algorithm is an application of a stochastic sub- gradient method (see for example [25,34]). In the context of machine learning problems, the efficiency of the stochastic gradient approach has been studied in [26,1,3,27,6,5]. In particular, it has been claimed and experimentally observed that, Stochastic algorithms yield the best generalization performance despite being the worst optimization algorithms. This claim has recently received formal treatment in [4,32]. Two concrete algorithms that are closely related to the Pegasos algorithm and are also variants of stochastic sub-gradient methods are the NORMA algorithm [24] and a stochas- tic gradient algorithm due to Zhang [37]. The main difference between Pegasos and these variants is in the procedure for setting the step size. We elaborate on this issue in Sec. 7. The convergence rate given in [24] implies that the number of iterations required to achieve -accurate solution is O(1/( )2). This bound is inferior to the corresponding bound of Pe- gasos. The analysis in [37] for the case of regularized loss shows that the squared Euclidean distance to the optimal solution converges to zero but the rate of convergence depends on the step size parameter. As we show in Sec. 7, tuning this parameter is crucial to the success of the method. In contrast, Pegasos is virtually parameter free. Another related recent work is Nesterovs general primal-dual subgradient method for the minimization of non-smooth functions [28]. Intuitively, the ideas presented in [28] can be combined with the stochastic regime of Pegasos. We leave this direction and other potential extensions of Pegasos for future

research.

- **Cutting Planes Approach:** Recently, Joachims [21] proposed SVM-Perf, which uses a cut- ting planes method to find a solution with accuracy in time O(md/(2)). This bound was later improved by Smola et al [33] to O(md/()). The complexity guarantee for Pe- gasos avoids the dependence on the data set size m. In addition, while SVM-Perf yields very significant improvements over decomposition methods for large data sets, our experiments (see Sec. 7) indicate that Pegasos is substantially faster than SVM-Perf.

Although support vector machines represent a major development in machine learning algorithms, in the case of large-scale problems (hundreds of thousands to several millions of samples), the design of SVM training algorithms still has room for improvement. So far, there have been various approaches for tackling large-scale SVM classification problems.

The first attempts at speeding up the training time and decreasing algorithm memory requirements were aimed at decomposing the underlying SVMs quadratic programming problem. First, Boser et al. [6] implemented Vapnik's *chunking* method. *Sequential Minimal Optimization* by Platt [35] and its improvement by Keerthie et al. [28] are an alternative approach to decomposing the QP problem, and are implemented in popular, widely used software package LIBSVM [11]. SMO is an iterative procedure that divides the SVM dual problem into a series of sub-problems, which are solved analytically by finding the optimal $\boldsymbol{\alpha}$ values that satisfy the Karush-Kuhn-Tucker conditions [8]. Although SMO is guaranteed to converge, heuristics are used to choose $\boldsymbol{\alpha}$ values in order to accelerate the convergence rate. This is a critical step because the convergence speed of the SMO algorithm is highly dependent on the dataset size and SVM hyperparameters [37].

Some advancements in handling large scale problems are based on a geometric interpretation of SVM problem. Some of these *geometric* SVMs include approaches that use convex hulls [4] and minimum enclosing balls such as *Core Vector Machines* (CVM) [44]. Tsang et al. [43] then improved the scalability of CVMs by introducing *Ball Vector Ma-*

chines (BVM) which do not require a QP solver. Other geometric approaches include the novel algorithms introduced by Strack [41], known as the *Sphere Support Vector Machine* (SphereSVM) and *Minimal Norm Support Vector Machine* (MNSVM), which utilize the connection between minimal enclosing balls and convex hull problems, while demonstrating a high capability for learning from large datasets. The *Non-Negative Iterative Single Data Algorithm* (NNISDA) [50] is an efficient approach for solving the SVM problem, shown to be faster than SMO and equal in terms of accuracy [27]. NNISDA is an iterative algorithm that finds a solution to the L2-SVM using coordinate descent, inspired by *Iterative Single Data Algorithm* (ISDA) [21] which was originally introduced by Kecman et al. [25].

Recently, several authors have proposed the use of a standard stochastic gradient descent (SGD) approach for SVMs to optimize large-scale learning problems [19, 29, 37, 38, 39]. Kivinen et al.[30] and Bousquet and Bottou [7] showed that stochastic algorithms can be both the fastest, and have the best generalization performances. Shalev-Shwartz and Ben-David [38] have also demonstrated that the basic SGD algorithm is very effective when data are sparse, taking less than linear $[O(d)]$ time and space per iteration to optimize a system with d parameters. It can greatly surpass the performance of more sophisticated batch methods on large data sets. The previously mentioned approaches are extended variants of a classic kernel perceptron algorithm [12].

Notable representatives of this method of learning include the *Naïve Online R Minimization Algorithm* (NORMA) by Kivinen et al. [30] and the *Primal Estimated Sub-Gradient SOlver for SVM* (PEGASOS) by Shalev-Shwartz et al. [39]. NORMA is an online kernel based algorithm designed to utilize SGD for solving the SVM problem, exploiting the kernel trick in an online setting. It can be regarded as a generalization of the kernel perceptron algorithm with regularization [30]. PEGASOS solves the primal SVM problem using stochastic sub-gradient descent, implementing both linear and non-linear kernels, and showed that the algorithm does not directly depend on the size of the data, making it suitable for large-scale learning problems.

A more recent approach, named *OnLine Learning Algorithm* (OLLA) [23] is a unification, simplification, and expansion of the somewhat similar approaches presented in [19, 29, 37, 38, 39] and [26, 33, 34]. This algorithm is unique because it is not only designed to optimize the SVM cost function, but also the cost functions of several other popular nonlinear (kernel) classifiers using SGD in the primal domain. Collobert and Bengio [12] provided justification for not using regularization, and thus OLLA was designed to handle cost functions with and without the regularization term. Comparisons of performances of OLLA with the popular SMO algorithm highlighted the merits of OLLA in terms of speed, as well as accuracy, when the number of samples was increased, making it suitable for large-scale learning. Comparisons using various different classifiers against SMO were also shown [23], but for the scope of this proposal the L1-SVM was mentioned.

Although the SGD approaches mentioned above have many merits when it comes to solving large-scale machine learning problems, stochastic procedures also have their disadvantages. One of them stems from the lack of meaningful stopping criteria. The only specified stopping criteria is a user defined input for the number of iterations, which gives rise to the question of what it should be set to. Another disadvantage of kernelized online algorithms is that the training time for each update increases superlinearly with the number of samples.