Artif Intell Rev (2019) 52:803–855
https://doi.org/10.1007/s10462-018-9614-6

Problem formulations and solvers in linear SVM: a review

Vinod Kumar Chauhan[1] · **Kalpana Dahiya**[2] · **Anuj Sharma**[1]

Published online: 16 January 2018

Abstract Support vector machine (SVM) is an optimal margin based classification technique in machine learning. SVM is a binary linear classifier which has been extended to non-linear data using Kernels and multi-class data using various techniques like one-versus-one, one-versus-rest, Crammer Singer SVM, Weston Watkins SVM and directed acyclic graph SVM (DAGSVM) etc. SVM with a linear Kernel is called linear SVM and one with a non-linear Kernel is called non-linear SVM. Linear SVM is an efficient technique for high dimensional data applications like document classification, word-sense disambiguation, drug design etc. because under such data applications, test accuracy of linear SVM is closer to non-linear SVM while its training is much faster than non-linear SVM. SVM is continuously evolving since its inception and researchers have proposed many problem formulations, solvers and strategies for solving SVM. Moreover, due to advancements in the technology, data has taken the form of 'Big Data' which have posed a challenge for Machine Learning to train a classifier on this large-scale data. In this paper, we have presented a review on evolution of linear support vector machine classification, its solvers, strategies to improve solvers, experimental results, current challenges and research directions.

Keywords Support vector machines · Support vector classification · Linear SVM · SVM solvers · Optimization problem

✉ Anuj Sharma
anujs@pu.ac.in
https://sites.google.com/site/anujsharma25

Vinod Kumar Chauhan
vkumar@pu.ac.in; jmdvinodjmd@gmail.com

Kalpana Dahiya
kalpanas@pu.ac.in

[1] Department of Computer Science and Applications, Panjab University, Chandigarh, India

[2] University Institute of Engineering and Technology, Panjab University, Chandigarh, India

1 Introduction

Support vector machine (SVM, Boser et al. 1992; Cortes and Vapnik 1995) is a binary linear classification technique in Machine Learning, which separates the classes with largest gap (called optimal margin) between the border line instances (called Support Vectors). That's why it is known as optimal margin classifier. Figure 1, represents the geometrical view of SVM. SVM has been extended to multi-class problems using techniques like One-versus-One (Knerr et al. 1990; Friedman 1996; Kressel 1999), One-versus-Rest (Vapnik 1998; Crammer and Singer 2002; Weston and Watkins 1999) and Directed Acyclic Graph SVM (Platt et al. 2000) etc. For non-linearly separable data problems, SVM has been extended using Kernels. Kernels are mathematical functions that transform the data from given space (known as Input Space) to a new high dimensional space (known as Feature Space) where data can be separated with a linear surface (called hyperplane). Figure 2, represents the geometrical view of Kernels. Mathematically, a Kernel is a function that takes two arguments, apply a mapping on the arguments and then return the value of their dot product. Suppose x_1 and x_2 are two data points, ϕ is a mapping and K denotes Kernel which is given by

$$K(x_1, x_2) = \phi(x_1)^T \phi(x_2) \tag{1}$$

In some applications, the Input Space is rich in features and it is sufficient to take the mapping ϕ as an identity mapping, i.e., $\phi(x) = x$. Kernel where mapping is identity mapping, that is, Input Space and Feature Space are equal, is called linear Kernel and SVM using linear Kernel is called linear SVM. Mathematically, linear Kernel is given by

$$K(x_1, x_2) = x_1^T x_2 \quad \Longrightarrow \quad \phi(x) = x$$

Linear SVM is very efficient in high dimensional data applications. While their accuracy on test set is close to the non-linear SVM, it is much faster to train for such applications. For example, document classification applications have high dimensional Input Space and there is no need to add more features to the Input Space because it does not make much difference in the performance. So test accuracy of linear SVM is close to that of non-linear SVM (see, Gao and Sun 2010) but at the same time, training of linear SVM is much faster than non-linear SVM due to the difference in their computational complexities.

Linear SVM involves problem formulations, solvers to solve the problem and optimization strategies to make the solvers efficient. Generally, SVM problem is formulated as a convex problem (Crammer and Singer 2002; Cortes and Vapnik 1995) because there is no issue of local optimum in convex problems as every local optimum is a global optima. In formulating the problem, commonly L1 ($\|w\|$) and L2 regularization ($\|w\|^2$), L1 and L2 loss functions are used which are given below. So considering L1 and L2 regularization, L1 and L2 loss

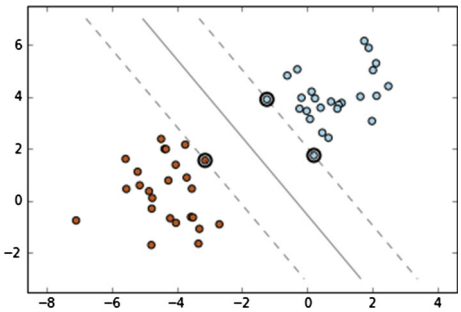

Fig. 1 An infinite number of classifiers can be drawn for the given data but SVM finds the classifier with largest gap between support vectors. Circles represent the support vectors

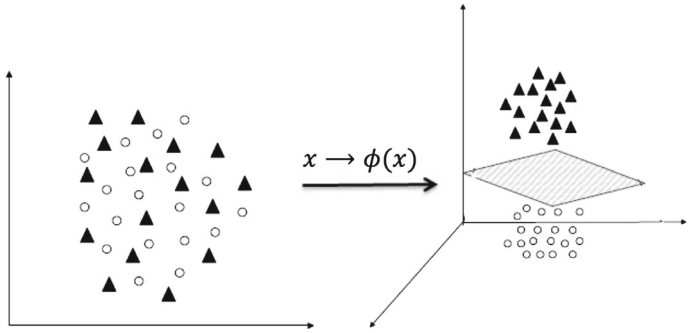

Fig. 2 Left part represents data in Input space and right part represents data in Feature space. Data has been transformed from Input space to Feature space using Kernels. Initially Input space is two dimensional and data is inseparable. Kernels transformed the data to three dimensional space where data is separable by a hyperplane

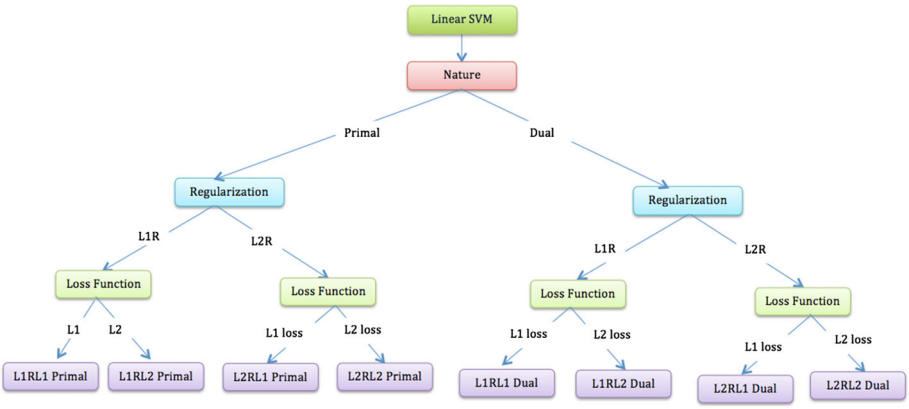

Fig. 3 Eight different problem formulations for the Linear SVM with L1-regularized (L1R), L2-regularized (L2R), L1 loss and L2 loss functions

functions and linear Kernel, there are eight different formulations for linear SVM as depicted in Fig. 3, which indicates the diverse range of possible problem formulations. L1 and L2 loss functions for parameter vector w and data point (x_i, y_i) are given below:

$$\text{L1-loss:} \quad \max\left(0, 1 - y_i w^T x_i\right), \quad \text{L2-loss:} \quad \left[\max\left(0, 1 - y_i w^T x_i\right)\right]^2 \tag{2}$$

For a particular SVM either primal or dual problems can be solved. Generally, when number of features (n) are much less than number of training points (l) then primal formulations of SVM (Lin et al. 2008) are solved otherwise dual formulations (Hsieh et al. 2008) are solved. This is because lesser the number of variables in the problem easier is the problem to solve.

Now a days, major challenge in SVM (which is shared with machine learning) is to deal with 'Big Data' problems and to handle such problems researchers are trying to exploit the platform/framework structures (like parallel and distributed computing) and proposing new problem formulations, solvers, optimization strategies and algorithms etc. The current research directions for handling big data problems in machine learning are: First, stochastic

approximation algorithms (Johnson and Zhang 2013; Konečný and Richtárik 2013; Defazio et al. 2014; Schmidt et al. 2016) which makes the learning algorithm scalable by removing the dependency of algorithm over number of data points by using only one data point during each iteration of the algorithm. Several variations of stochastic approximation, like mini-batch approach (Tak et al. 2013; Shalev-Shwartz and Zhang 2013a), have been proposed during last 5 years. Second, coordinate descent algorithms (Tseng 2001; Nesterov 2012; Beck and Tetruashvili 2013; Richtárik and Takáč 2014; Wright 2015; Lu and Xiao 2015; Shi and Liu 2016) which reduce large problem into a smaller subproblem by taking one feature as a variable and rest as constants. Extensive research have been done in this direction during last few years and researcher have proposed its several variations. Third, proximal algorithms (Parikh and Boyd 2014) which are used with first and second order methods. These algorithms are applicable to non-smooth problems with separable components. Fourth, parallel and distributed computing (Yang et al. 2016) which exploits the computing resources. Now a days, a lot of research is going in parallel and distributed computing since dataset sizes are going beyond the capacity of single computer and other approaches seem to be saturated. In addition to these research directions, their hybrid versions are also being studied very extensively, e.g., Xu and Yin (2015); Chauhan et al. (2017a).

Paper organization: Sect. 2 discusses literature, Sect. 3 discusses evolution of linear SVM problem formulations and Sect. 4 discusses problem solvers for SVM. Section 5 presents comparative study, Sect. 6 presents big data challenge, areas of improvement to tackle big data problems and research directions for linear SVM and Sect. 7 presents the concluding remarks.

Notations: $\langle ., . \rangle$ denotes dot product, $\forall$ means for all, $x_i \in \mathbb{R}^n$ denotes a training instance with n features and $i = 1, 2, \ldots, l$ where l denotes the number of training points. $y_i \in \{-1, +1\}$ denotes the labels of training instances. $m \in \mathbb{R}$ is used to denote the number of classes. $w \in \mathbb{R}^n$ and $b \in \mathbb{R}$ denote the parameters of the objective function. ξ denotes the slack variable, C denotes the penalty parameter, $\|.\|$ denotes Euclidean norm, $\mathbf{1}$ denotes a unit vector of arbitrary dimension and α denotes vector of dual variables.

2 Literature

Support vector machines (SVMs, Boser et al. 1992; Cortes and Vapnik 1995) is a classification technique in Machine Learning. The concepts used in SVMs, like hyperplane, optimal margin and Kernels etc., are several decades old but in the current form, SVM developments have started from Boser et al. (1992), namely, "A Training Algorithm for Optimal Margin Classifiers" in 1992 in COLT. This SVM formulation is also known as 'Hard Margin SVM'. Cortes and Vapnik (1995) improved the hard margin SVM. They considered the outliers and noise, and relaxed the margin to mis-classify some data points by introducing slack variables. This led to the 'Soft Margin SVM'. Till this time, SVM was used as a binary classifier, when Vapnik (1998) proposed a multi-class SVM using one-versus-rest (OVR) technique. Joachims (1999), proposed Transductive SVM for text classification. Suykens and Vandewalle (1999), proposed least squared version of SVM. Later Kressel (1999), proposed another multi-class SVM using one-versus-one (OVO) technique but both, OVR and OVO, were based on binary classification problems. Weston and Watkins (1999) proposed first multi-class SVM (WWSVM), which solves a single SVM problem and was not based on multiple binary classification problems. Platt et al. (2000) proposed Directed Acyclic Graph SVM (DAGSVM) for multi-class classification problems, which was similar to OVO technique in solving SVM problem but it employs an efficient, rooted acyclic binary tree

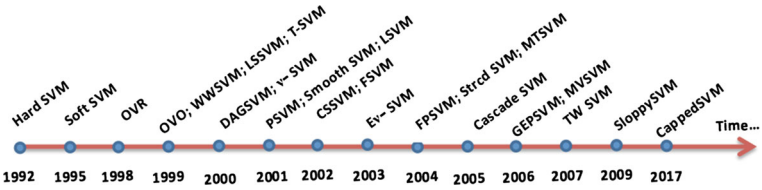

Fig. 4 Time line of SVM: OVR–one versus rest, T-SVM–transductive SVM, OVO–one-versus-one, WWSVM–Weston and Watkins SVM, LSSVM–least squares SVM, DAGSVM–directed acyclic graph SVM, LSVM–Lagrangian SVM, CSSVM–Crammer Singer SVM, FSVM–fuzzy SVM, PSVM–proximal SVM, FPSVM–fuzzy proximal SVM, Strcd SVM–structured SVM, MTSVM–multi-task SVM, GEPSVM–generalized eigenvalue proximal SVM, MVSVM–multi-view SVM, TW SVM–twin SVM

to find the label for the test data. Schölkopf et al. (2000), proposed a new ν-SVM, which removes parameter C and introduces parameter ν that roughly gives the fraction of support vectors. Fung and Mangasarian (2001) proposed Proximal SVM (PSVM) which uses the proximity of data point to the proximal planes to classify the data point. PSVM replaces bounding planes of the standard SVM with the proximal planes and uses proximal planes for classification instead of the separating hyperplane. Lee and Mangasarian (2001), introduced Smooth SVM which handles the non-differentiability issues of the standard SVM. Mangasarian and Musicant (2001), proposed Lagrangian SVM. Crammer and Singer (2002) proposed another multi-class classification SVM that solves a single SVM problem unlike OVO and OVR techniques. Later Lin and Wang (2002) proposed a new formulation called Fuzzy SVM (FSVM) which was further extended by Abe and Inoue (2002) and Abe (2015). Fuzzy SVM introduces fuzzy membership of data points to each of the classes and contributions of these points are used to find the separating plane. Fuzzy SVM is very useful in reducing the effect of outliers and noise. Prez-Cruz et al. (2003), extended ν-SVM (Schölkopf et al. 2000) to $E\nu$-SVM and helped to control margin, errors and consequently support vectors. Jayadeva et al. (2004), combined the idea of FSVM (Lin and Wang 2002) and PSVM (Fung and Mangasarian 2001) and proposed Fuzzy Proximal SVM (FPSVM). Here fuzzy membership is assigned to data points and points are classified on the basis of proximity to two parallel planes which are kept as far from each other as possible. Tsochantaridis et al. (2004) proposed Structured SVM which generalizes the multi-class SVM, namely, WWSVM and CSSVM, to a new formulation that has structured outputs like sequences, strings, graphs, labeled trees or lattices etc. Evgeniou and Pontil (2004) extended the standard SVM, i.e., single task learning SVM and proposed a new binary SVM for multi-task learning. Graf et al. (2005), proposed a parallel SVM known as Cascade SVM which deploys a divide and conquer strategy to solve the SVM problem. Cascade SVM divides the dataset into subsets and trains multiple SVMs in parallel on each subset. Mangasarian and Wild (2006), proposed Generalized Eigenvalue Proximal SVM (GEPSVM) with two non-parallel hyperplanes and the data points are classified according to the proximity to one of the two non-parallel planes. Farquhar et al. (2006) extended the idea of multi-view learning to SVM and proposed multi-view SVM. Jayadeva and Chandra (2007), extended Proximal SVM to Twin SVM which solves two small sized SVM problems and gives two non-parallel hyperplanes. Stempfel and Ralaivola (2009), proposed SloppySVM which can learn from the sloppily labeled data. Nie et al. (2017) have proposed CappedSVM using capped l_p-norm based loss function, to tackle the challenge of data outliers.

Figure 4, depicts the year-wise evolution of different SVMs which are not necessarily improvements over previous ones. The mathematical formulations and other details of dif-

ferent SVM evolutions and their solvers are discussed in the coming sections. The focus of this paper is on supervised linear Support Vector Classification (SVC), so only those SVM formulations are discussed which are used for classification and have major contributions. As far as linear and non-linear SVMs are concerned, they have same formulations except the Space in which they perform classification. Linear SVMs perform in the Input Space and non-linear SVMs perform in the Feature Space. The paper discusses only linear SVMs but the mathematical formulations for non-linear SVMs can be derived from the linear SVMs by using the following replacements:

$$x_1^T x_2 \mapsto K(x_1, x_2),$$

where x_1 and x_2 denote data points and K denotes the Kernel function.

3 Problem formulations

Problem formulation is the mathematical representation of the problem under study. SVM Classification problem is an optimization problem whose mathematical representation consists of an objective function and constraints (if any). Researchers have proposed a variety of ideas for SVM classification with different problem formulations. The evolution of the SVM is discussed in the following subsections:

3.1 Hard margin SVM (1992)

Although the concepts like hyperplane, optimal margin and Kernels are in use for several decades but Boser et al. 1992 are the first to formulate SVM in its current form. This SVM is a binary classifier which can work with only linearly separable data applications and presence of any noise or outliers strongly affects the margin. It does not allow mis-classification errors, that's why it is known as hard margin SVM. Hard margin SVM is represented by the Fig. 1.

Suppose the training set of instance-label pairs is denoted by (x_i, y_i), for $i = 1, 2, \ldots, l$ where $x_i \in \mathbb{R}^n$, $y_i \in \{+1, -1\}$, n is the number of features and l is the number of training data points. Then hard margin linear SVM solves the following optimization problem:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & y_i(w^T x_i + b) \geq 1, \ \forall i, \end{aligned} \tag{3}$$

where w, b are parameters. The decision function is:

$$h(x) = sign(w^T x + b).$$

Parameters w and b can be combined into a single parameter by adding a new feature to every instance for notational convenience as follow:

$$x_j^T \leftarrow \left[x_j^T, 1\right] \quad w^T \leftarrow [w^T, b]$$

So now Eq. (3) and decision function can be rewritten as:

$$\begin{aligned} \min_{w} \quad & \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & y_i w^T x_i \geq 1, \ \forall i \\ & h(x) = sign(w^T x) \end{aligned} \tag{4}$$

Equation (4) represents the primal form of the hard margin linear SVM. The dual form of the hard margin linear SVM, obtained using the Lagrangian Multipliers technique is:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\alpha^T Q\alpha - \mathbf{1}^T\alpha \\ \text{s.t.} \quad & \alpha_i \geq 0, \quad \alpha y = 0 \end{aligned} \tag{5}$$

where $Q_{ij} = y_i y_j x_i^T x_j$, α is vector of dual variables α_i, y is vector of output labels y_i and $\mathbf{1}$ is a unit vector. The decision function and relation between w and α are given as:

$$h(x) = sign\left(\sum_{i=1}^{l} y_i \alpha_i x_i^T x\right), \quad w = \sum_{i=1}^{l} y_i \alpha_i x_i .$$

3.2 Soft margin SVM (1995)

The problem with hard margin SVM is that if the dataset is not linearly separable then it becomes impossible to classify the data (that is, no separating hyperplane is found) and the presence of noise or outliers greatly affect the margin. Cortes and Vapnik (1995), introduced a slack variable into the hard margin SVM to allow some mis-classification errors. This new SVM is known as soft margin SVM. Figure 5, depicts an example of soft margin SVM.

Soft margin SVM solves the following optimization problem:

$$\begin{aligned} \min_{w} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\xi_i \\ \text{s.t.} \quad & y_i w^T x_i \geq 1 - \xi_i, \quad \xi_i \geq 0, \ \forall i, \end{aligned} \tag{6}$$

where ξ_i is the slack variable to allow some misclassification and C is a penalty parameter. ξ_i is replaced by loss functions in SVM. L1 and L2 loss are two common loss functions in literature, given by Eq. (2). The decision function is given by

$$h(x) = sign(w^T x).$$

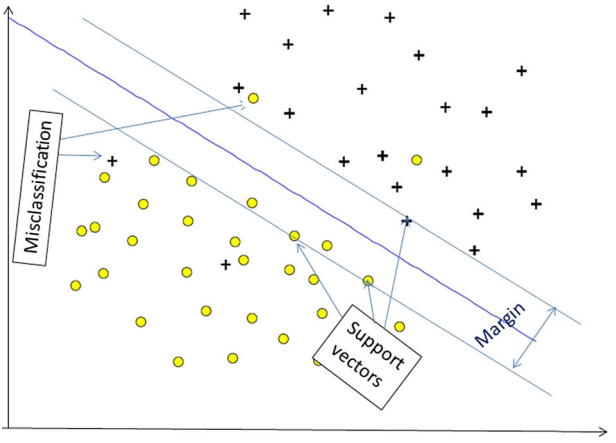

Fig. 5 An example of two dimensional inseparable data. Since data is not completely separable so SVM allows some mis-classifications errors to find the separating hyperplane

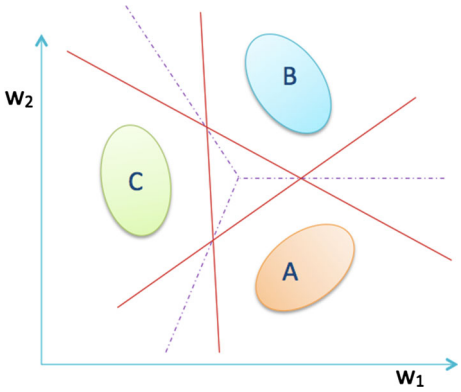

Fig. 6 One versus rest, multi-class classification technique: An example of three classes A, B and C in two dimensions

Equation (6) represents the primal form, the dual form of soft margin SVM is given by

$$\begin{aligned} \min_{\alpha} \quad & f(\alpha) = \frac{1}{2}\alpha^T \bar{Q}\alpha - \mathbf{1}^T\alpha \\ \text{s.t.} \quad & 0 \le \alpha_i \le U, \ \forall i, \end{aligned} \tag{7}$$

where $\bar{Q} = Q + D$, D is a diagonal matrix, $Q_{ij} = y_i y_j x_i^T x_j$,

$$U = \begin{cases} C & \text{for L1 loss SVM} \\ \infty & \text{for L2 loss SVM} \end{cases} \quad \text{and } D_{ii} = \begin{cases} 0 & \text{for L1 loss SVM} \\ \dfrac{1}{2C} & \text{for L2 loss SVM} \end{cases} \quad \forall i.$$

3.3 One-versus-rest (1998)

One-versus-rest (OVR, Vapnik 1998) technique is also called as one-versus-all and one-against-all. This is probably the earliest multi-class SVM implementation and is one of the mostly used multi-class SVM. In this technique, we solve m binary SVM problems where m is the number of classes. Each SVM classifier separates one class from the rest of the classes. In this way, multi-class classification problem is reduced to binary classification problem. OVR technique is pictorially represented in the Fig. 6.

$$\begin{aligned} \min_{w^i, b^i, \xi^i} \quad & \frac{1}{2}(w^i)^T w^i + C \sum_{j=1}^{l} \xi_j^i \\ \text{s.t.} \quad & (w^i)^T x_j + b^i \ge 1 - \xi_j^i, && \text{if } y_j = i, \\ & (w^i)^T x_j + b^i \le -1 + \xi_j^i, && \text{if } y_j \ne i, \\ & \xi_j^i \ge 0, && j = 1, 2, \ldots, l, \ i = 1, 2, \ldots, m. \end{aligned} \tag{8}$$

After solving, it needs m decision functions to predict the class of a test data point as given below:

$$(w^1)^T x + b^1, \quad (w^2)^T x + b^2, \quad \ldots \quad , \quad (w^m)^T x + b^m. \tag{9}$$

A test data point is assigned a particular class only when the data point is accepted by that class and rejected by all other classes. But this leads to a problem when a data point is accepted by more than one class and that leads to a common region in the feature space. Vapnik (1998) suggested that data point should be assigned to the class with highest value

of decision function, regardless of sign. The final output label is given to the class that has highest output value:

$$\text{class of x} \quad \equiv \underset{i=1,2,\ldots,m}{argmax} \quad ((w^i)^T x + b^i). \tag{10}$$

The main disadvantage of this technique is that it used to solve m different optimization problems and need to evaluate m decision functions for each test data point. But it is a simple technique.

3.4 Transductive SVM (1999)

Transductive SVM (T-SVM, Joachims 1999) is proposed for text classification problems. In Transductive SVM, the goal is to classify a given dataset with as few errors as possible without caring about the particular decision function. Standard SVM induces a decision function on training dataset and uses that decision function with the test dataset, but Transductive SVM considers the test dataset and try to minimize the misclassification errors for just those particular data points.

Transductive SVM for minimizing over $(y_1^*, y_2^*, \ldots, y_n^*, w, b, \xi_1, \xi_2, \ldots, \xi_n, \xi_1^*, \xi_2^*, \ldots, \xi_k^*)$ is:

$$\begin{aligned} \min_{w,b,\xi_i,\xi_j^*} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=0}^{n}\xi_i + C^*\sum_{j=0}^{k}\xi_j^* \\ \text{s.t.} \quad & y_i\left[w^T x_i + b\right] \geq 1 - \xi_i, \; y_j^*\left[w^T x_j^* + b\right] \geq 1 - \xi_j^*, \\ & \xi_i > 0, \quad \xi_j^* > 0, \quad i = 1, 2, \ldots, n, \quad j = 1, 2, \ldots, k \end{aligned} \tag{11}$$

where (x_i, y_i) are training data points with error ξ_i and (x_j^*, y_j^*) are test data point with error ξ_j^*.

3.5 Weston watkins SVM (1999)

Unlike OVR (Vapnik 1998) which solves many binary SVM problems, this technique (Weston and Watkins 1999) (WWSVM) solves only one SVM classifier. But it uses multiple decision functions to classify a data point just like OVR technique. This SVM problem formulation is as:

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}\sum_{k=1}^{m} w_k^T w_k + C\sum_{i=1}^{l}\sum_{k\neq y_i}\xi_i^k \\ & w_{y_i}^T x_i + b_{y_i} \geq w_k^T x_i + b_k + 2 - \xi_i^k \\ & \xi_i^k \geq 0, \quad i = 1, 2, \ldots, l, \quad k \in \{1, 2, \ldots, m\}\backslash y_i \end{aligned} \tag{12}$$

The decision function is:

$$\text{class of x} \quad \equiv \underset{k=1,2,\ldots,m}{argmax} \left(w_k^T x + b_k\right). \tag{13}$$

The corresponding dual problem is:

$$\begin{aligned} \min_{\alpha} \quad & \sum\left(\frac{1}{2}c_j^{y_i} A_i A_j - \sum_k \alpha_i^k \alpha_j^{y_i} + \frac{1}{2}\sum_k \alpha_i^k \alpha_j^k\right) x_i^T x_j - 2\sum_{i,k}\alpha_i^k \\ & \sum_{i=1}^{l}\alpha_i^k = \sum_{i=1}^{l} c_i^k A_i, \quad \forall i, \quad k = 1, 2, \ldots, m \end{aligned}$$

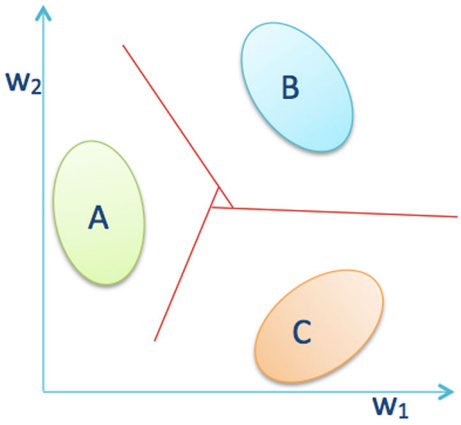

Fig. 7 One versus one multi-class classification technique: an example of three classes A, B and C in two dimensions

$$\text{where} \quad 0 \le \alpha_i^k \le C, \quad \alpha_i^{y_i} = 0, \quad A_i = \sum_{k=1}^{m} \alpha_i^k, \tag{14}$$

$$C_j^{y_i} = \begin{cases} 1, & \text{if } y_i = y_j \\ 0, & \text{if } y_i \neq y_j, \end{cases} \quad i = 1, 2, \ldots, l, \quad k = 1, 2, \ldots, m$$

$$w_k = \sum_{i=1}^{l} \left(c_i^k A_i - \alpha_i^k \right) x_i, \quad k = 1, 2, \ldots, m. \tag{15}$$

The decision function is:

$$\underset{k=1,2,\ldots,m}{argmax} \left(\sum_{i=1}^{l} \left(c_i^k A_i - \alpha_i^k \right) x_i^T x_j + b_k \right) \tag{16}$$

The primal formulations are equivalent to Bredensteiner and Bennett (1999) and Guermeur (2002).

3.6 One-versus-one (1999)

One-versus-one technique (OVO, Kressel 1999) is also called as one-against-one (OAO, 1-a-1) or pair-wise classification. In this classification technique, a binary SVM is trained for each possible pair of classes. This idea of pair-wise classification was proposed by Knerr et al. (1990) and later Friedman (1996), proposed 'Max Wins' algorithm which suggested that each one-vs-one classifier contributes one vote to its predicted class and final output is the class with the maximum number of votes. Kressel (1999), applied this idea to SVM. Suppose there are m classes then OVO technique would lead to $m(m-1)/2$ binary SVM classification problems, each one is trained on data from two classes. Figure 7, depicts the OVO multi-class SVM.

For training data from the ith and jth classes, the problem is given by

$$\min_{w^{i,j}, b^{i,j}, \xi^{i,j}} \frac{1}{2} (w^{i,j})^T w^{i,j} + C \sum_t \xi_t^{i,j} \tag{17}$$

$$\begin{aligned} &\text{s.t.} \quad ((w^{i,j})^T x + b^{i,j}) \geq 1 - \xi^{i,j}, \quad \text{for } y = i, \\ &((w^{i,j})^T x + b^{i,j}) \leq -1 + \xi^{i,j}, \quad \text{for } y = j, \\ &\xi_{i,j} \geq 0, \end{aligned} \tag{18}$$

where $t \in X_{i,j}$ and $X_{i,j}$ is the set of data points with labels i and j.

3.7 Least squares SVM (1999)

Least Squares SVM (LSSVM, Suykens and Vandewalle 1999) proposes a least squared version of SVM with equality constraints. Due to equality constraints, it solves a set of linear equations to find the solution, unlike standard SVM which solves quadratic programming. In LSSVM, number of support vectors are proportional to number of errors. LSSVM is given by

$$\min_{w,b,\xi} \quad \frac{1}{2}\|w\|^2 + \frac{C}{2}\sum_{i=1}^{l}\xi_i^2, \tag{19}$$

$$\text{s.t.} \quad y_i\left(w^T x_i + b\right) = 1 - \xi_i, \quad i = 1, 2, \ldots, l. \tag{20}$$

LSSVM has been extended by Li et al. (2006) which assigns a degree of importance to each training data point, this idea is similar to Fuzzy SVM (Lin and Wang 2002).

3.8 ν-SVM (2000)

ν-SVM (Schölkopf et al. 2000) proposes a new SVM formulation for classification and regression problems where a new parameter ν is introduced. ν roughly controls the fraction of support vectors. In addition to that, ν also helps to eliminate one of the other free parameters of the algorithm, ϵ (accuracy parameter) in regression and C (regularization coefficient) in the classification. Primal formulation for ν-SVM is:

$$\begin{aligned} \min_{w,b,\xi,\rho} \quad & \tau(w,\xi,\rho) = \frac{1}{2}\|w\|^2 - \nu\rho + \frac{1}{l}\sum_i \xi_i \\ \text{s.t.} \quad & y_i\left(x_i^T w + b\right) \geq \rho - \xi_i \\ & \xi_i \geq 0, \quad \rho \geq 0, \quad \nu \in [0,1] \end{aligned} \tag{21}$$

Dual problem for ν-SVM:

$$\begin{aligned} \max_{\alpha} \quad & W(\alpha) = -\frac{1}{2}\sum_{ij}\alpha_i\alpha_j y_i y_j x_i^T x_j \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq \frac{1}{l}, \quad 0 = \sum_i \alpha_i y_i, \quad \sum_i \alpha \leq \nu \end{aligned} \tag{22}$$

Decision function is:

$$h(x) = sign\left(\sum_i \alpha_i y_i x^T x_i + b\right) \tag{23}$$

On comparing with original soft margin dual problem, ν-SVM has two differences: Additional constraints and the linear term $\sum_i \alpha_i$ disappears from objective function in Eq. (22).

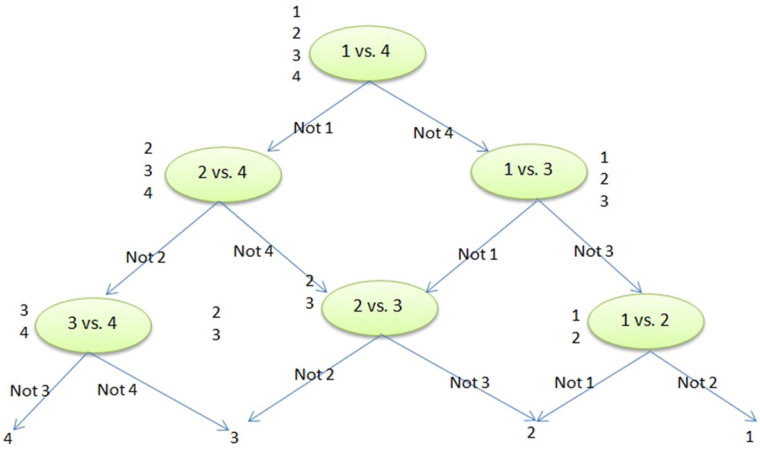

Fig. 8 DAGSVM: an example of classification with four classes 1, 2, 3 and 4. Suppose a test data point belonging to class 4 is being classified using DAGSVM. At root node, it evaluates a decision function for classes 1 versus 4 and move to left (not 1). Then it evaluates decision function for 2 versus 4 and move to left (not 2) and in the last it evaluates for 3 versus 4 and predicts label 4 as output. Thus for four classes (m), it needs three ($m-1$) evaluations of decision functions

3.9 DAGSVM (directed acyclic graph SVM, 2000)

DAGSVM problem (Platt et al. 2000) is a multi-class SVM which is similar to OVO technique and like OVO, it solves $m(m-1)/2$ binary SVM classifier for m classes. But to classify one test data point, it employs different strategy. It uses a rooted binary directed acyclic graph which has $m(m-1)/2$ internal nodes and m leaves. Each internal node represents a binary SVM classifier which tests two classes. The leaf nodes represent the output class labels. For m classes, it needs only $m-1$ evaluations of decision functions. For a test data point x, it starts at root node, tests two classes and move to the left or right node, keep evaluating classifiers until it reaches the leaf node which gives the output class. Figure 8, depicts an example of classification using DAGSVM. As it is clear from the figure, for four classes it needs three evaluations of decision functions.

3.10 Smooth SVM (2001)

Smooth SVM (Lee and Mangasarian 2001) proposes a new smooth formulation for SVM, to handle its non-differentiability issue. Smooth SVM is a strongly convex and infinitely differentiable. It enjoys globally, quadratically convergent solutions using a fast Newton-Armijo algorithm. It uses Newton's method for obtaining the step direction and Armijo rule for determining the step size.

Suppose there are l training points in n dimensional space represented by $l \times n$ matrix A with membership of each data A_i in classes $+1$ and -1 is represented by the diagonal $l \times l$ matrix D with values $+1$ or -1 on the diagonal. The standard SVM with some $\nu > 0$ and linear Kernel $A^T A$, is given by

$$\begin{aligned} \min_{(w,\gamma,y)\in\mathbb{R}^{n+1+l}} \quad & \nu \mathbf{1}^T y + \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & D(Aw - \mathbf{1}\gamma) + y \geq \mathbf{1}, \quad y \geq 0 \end{aligned} \tag{24}$$

where w, γ are parameters and y is the slack variable. The standard SVM is first modified to get the following SVM:

$$\begin{aligned} \min_{w,\gamma,y} \quad & \frac{\nu}{2} y^T y + \frac{1}{2}\left(\|w\|^2 + \gamma^2\right) \\ \text{s.t.} \quad & D\left(Aw - \mathbf{1}\gamma\right) + y \geq \mathbf{1}, \quad y \geq 0. \end{aligned} \tag{25}$$

From the constraints of problem (25), we get,

$$y = \left(\mathbf{1} - D\left(Aw - \mathbf{1}\gamma\right)\right)_+, \tag{26}$$

where $(.)_+$ function replaces negative components of a vector with zeros. So substituting the value of y in Eq. (25), equivalent SVM is obtained as:

$$\min_{w,\gamma} \quad \frac{\nu}{2} \| \left(\mathbf{1} - D\left(Aw - \mathbf{1}\gamma\right)\right)_+ \|^2 + \frac{1}{2}\left(\|w\|^2 + \gamma^2\right) \tag{27}$$

The objective function of this problem is not twice differentiable so x_+ is replaced with a smooth approximation, given by

$$p(x, \alpha) = x + \frac{1}{\alpha} log\left(1 + e^{-\alpha x}\right), \quad \alpha > 0, \tag{28}$$

where α is the smoothing parameter and e is the exponential constant. So using Eq. (28) in Eq. (27), it gives the Smooth SVM as:

$$\min_{w,\gamma} \frac{\nu}{2} \| p\left(\mathbf{1} - D\left(Aw - \mathbf{1}\gamma\right), \alpha\right) \|^2 + \frac{1}{2}\left(\|w\|^2 + \gamma^2\right) \tag{29}$$

The decision function for Smooth SVM is given by $sign(w^T x - \gamma)$.

3.11 Lagrangian SVM (2001)

Lagrangian SVM (LSVM, Mangasarian and Musicant 2001) is obtained by reformulating the standard primal linear SVM and taking an implicit Lagrangian of its dual form. This gives an unconstrained differentiable convex function. The dual of Eq. (25) is:

$$\min_{0 \leq u \in \mathbb{R}^l} \quad \frac{1}{2} u^T Q u - \mathbf{1}^T u \tag{30}$$

where u denotes dual variable and $w = A^T D u$, $y = u/\nu$, $\gamma = -\mathbf{1}^T D u$, $H = D\left[A \quad -\mathbf{1}\right]$, $Q = \dfrac{I}{\nu} + HH^T$ and I is the identity matrix.

According to KKT conditions:

$$\begin{aligned} & 0 \leq u \ \perp \ Qu - \mathbf{1} \geq 0. \\ & \text{Since,} \quad 0 \leq a \ \perp \ b \geq 0 \quad \Longleftrightarrow \quad a = (a - \alpha b)_+, \ \alpha > 0 \end{aligned} \tag{31}$$

So Eq. (31) can be written as:

$$Qu - \mathbf{1} = \left((Qu - \mathbf{1}) - \alpha u\right)_+. \tag{32}$$

This leads to a simple iterative LSVM algorithm as:

$$u^{k+1} = Q^{-1}\left(\mathbf{1} + \left(\left(Qu^k - \mathbf{1}\right) - \alpha u^k\right)_+\right), \ k = 0, 1, \ldots \tag{33}$$

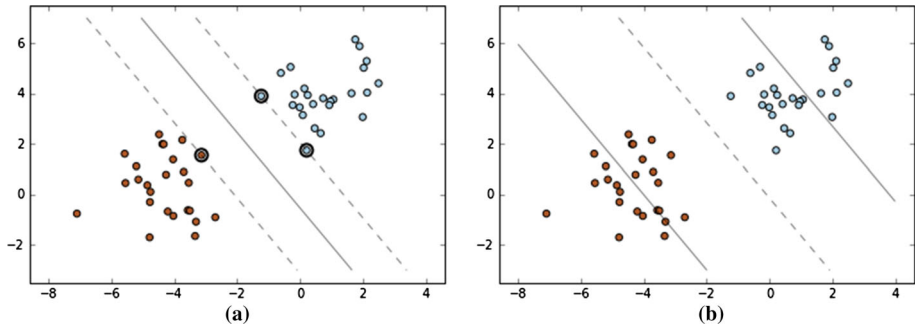

Fig. 9 Standard SVM versus Proximal SVM. **a** Standard SVM: separating plane with bounding planes. Circles represent support vectors. **b** Proximal SVM: separating plane with planes which are no longer bounding planes but proximal planes. The data points are clustered around the proximal planes

3.12 Proximal SVM (2001)

Proximal SVM (PSVM, Fung and Mangasarian 2001) classifies data points on the basis of proximity to the two parallel planes (called as proximal planes). Standard SVM classifies points using a separating hyperplane which assigns them to one of the half spaces generated by it (separating hyperplane). Figure 9, depicts the bounding planes in standard SVM and proximal planes of PSVM. The performance of PSVM on test dataset, is comparable to the standard SVM but its training speed on the training dataset, is considerably faster than the Standard SVM (Fung and Mangasarian 2001). Primal form of PSVM is given by

$$\begin{aligned} \min_{(w,\gamma,y)} \quad & \nu \frac{1}{2}\|y\|^2 + \frac{1}{2}\left(w^T w + \gamma^2\right) \\ \text{s.t.} \quad & D\left(Aw - \mathbf{1}\gamma\right) + y = \mathbf{1} \end{aligned} \tag{34}$$

where w, γ are parameters, A is $l \times n$ matrix representing l data points in n-dimensional real space $\mathbb{R}^n$, according to the membership of each point A_i in the class $A+$ or $A-$ as specified by a given $l \times l$ diagonal matrix D with $+1$ or -1 along its diagonal. $\nu > 0$, $\mathbf{1} =$ unit vector, y represents errors. The solution of this problem can be written directly in terms of problem data.

The planes $x^T w - \gamma = \pm 1$ are not boundary planes anymore but can be thought as proximal planes around which the points of each class are clustered and which are pushed as far apart as possible by the term $(w^T w + \gamma^2)$ in the objective function. The decision function for Proximal SVM $sign(w^T x - \gamma)$. Lagrangian with coefficient u is given by

$$L(w, \gamma, y, u) = \frac{\nu}{2}\|y\|^2 + \frac{1}{2}\left\|\begin{bmatrix} w \\ \gamma \end{bmatrix}\right\|^2 - u^T\left(D\left(Aw - \mathbf{1}\gamma\right) + y - \mathbf{1}\right) \tag{35}$$

Solving using KKT conditions gives

$$\begin{aligned} & w = A^T Du, \quad \gamma = -\mathbf{1}^T Du, \quad y = \frac{u}{\nu} \\ & u = \left(\frac{I}{\nu} + D\left(AA^T + \mathbf{1}\mathbf{1}^T\right)D\right)^{-1}\mathbf{1} = \left(\frac{I}{\nu} + HH^T\right)^{-1}\mathbf{1} \end{aligned} \tag{36}$$

where $H = D\,[A \quad -\mathbf{1}]$. Equation (36) gives the explicit solution to Eq. (34).

3.13 Crammer Singer SVM (2002)

Crammer and Singer (2002), proposed a multi-class SVM (CSSVM) which unlike OVR (Vapnik 1998) and OVO (Kressel 1999) trains a single SVM classifier and evaluates m (number of classes) decision functions to classify a test data point. The problem formulations are:

$$\begin{aligned} &\min_{w_k,\xi_i} \quad \frac{1}{2}\sum_{k=1}^{m} w_k^T w_k + C\sum_{i=1}^{l}\xi_i \\ &\text{s.t. } w_{y_i}^T x_i - w_k^T x_i \geq e_i^k - \xi_i, \quad i = 1,\ldots,l, \\ &\text{where} \quad e_i^k \equiv 1 - \delta_{y_i,k} \text{ and } \delta_{y_i,k} = \begin{cases} 1 & \text{if } y_i = k \\ 0 & \text{if } y_i \neq k \end{cases} \end{aligned} \tag{37}$$

Decision function is

$$\underset{k=1,2,\ldots,m}{argmax} \quad w_k^T x. \tag{38}$$

The main difference between Crammer Singer SVM (CSSVM) and other problems is that CSSVM does not need to explicitly write down constraints $\xi_i \geq 0$ as when $y_i = k$, $e_i^k = 0$, so Eq. (37) becomes $0 \geq 0 - \xi_i$ which is exactly $\xi_i \geq 0$. It contain only $\xi_i,\ i = 1, 2, \ldots, l,\ l$ slack variables, instead of using ξ_i^k as the gap between each pair of decision planes. Dual problem of CSSVM is:

$$\begin{aligned} &\min_{\alpha} \quad \frac{1}{2}\sum_{k=1}^{m}\|w_k\|^2 + \sum_{i=1}^{l}\sum_{k=1}^{m} e_i^k \alpha_i^k \\ &\text{s.t.} \quad \sum_{k=1}^{m}\alpha_i^k = 0, \quad i = 1,\ldots,l, \\ &\quad \alpha_i^k \leq C_{y_i}^k, \quad i = 1,\ldots,l,\ k = 1,\ldots,m, \end{aligned} \tag{39}$$

where $w_k = \sum_{i=1}^{l}\alpha_i^k x_i, \forall k$ and $C_{y_i}^k = \begin{cases} C & \text{if } y_i = k \\ 0 & \text{if } y_i \neq k \end{cases}$.

3.14 Fuzzy SVM (2002)

Fuzzy SVM (FSVM, Lin and Wang 2002) proposes a new SVM where each data point is assigned a fuzzy membership to the labeled class and reformulates the SVM problem where contributions of points are used to find the separating hyperplane. FSVM is useful in reducing the effect of outliers and noise. It is suitable for applications where data points have modeled characteristics. FSVM has been further extended to multi-class problems (see, Abe and Inoue 2002; Abe 2015).

Suppose labeled training data points are given with the associated fuzzy membership $(x_1, y_1, s_1), (x_2, y_2, s_2), \ldots, (x_l, y_l, s_l)$. $x_i \in \mathbb{R}^n$, $y_i \in \{-1, 1\}$ and fuzzy membership $\sigma \leq s_i \leq 1, i = 1, 2, \ldots, l$ and $\sigma > 0$.

$$\begin{aligned} &\min_{w,b,\xi_i} \quad \frac{1}{2}w^T w + C\sum_{i=1}^{l} s_i\xi_i \\ &\text{s.t. } y_i\left(w^T x_i + b\right) \geq 1 - \xi_i,\ \xi_i \geq 0,\ i = 1, 2, \ldots, l \end{aligned} \tag{40}$$

where smaller values of s_i reduces the effect of ξ_i and corresponding training point is considered as less important.

Dual form of Fuzzy SVM is:

$$\begin{aligned} \max_{\alpha} \quad & W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j x_i^T x_j \\ \text{s.t.} \quad & \sum_{i=1}^{l} y_i \alpha_i = 0, \quad 0 \le \alpha_i \le s_i C, \quad i = 1, 2, \ldots, l. \end{aligned} \tag{41}$$

3.15 *Eν*-SVM (2003)

The ν-SVM (Schölkopf et al. 2000) for classification (ν-SVC) has proposed a different formulation for solving SVM in which a new parameter ν has been introduced. ν gives an upper bound on the number of support vectors. ν also removes C which is a measure of trade off between the margin size and training errors. This is difficult to choose C priori. The value of ν cannot always take all possible values between 0 and 1 which limits the range of possible solutions. Either because the training set is non-separable or because the classes are unbalanced. So ν-SVM (Schölkopf et al. 2000) is extended to $E\nu$-SVM (Prez-Cruz et al. 2003), where a new problem is constructed using ν that helps to control the margin, errors and the support vectors. In $E\nu$-SVM, the value of ν cannot take full range of values from 0 to 1. For l training points, the value of ν can be defined as:

$$\nu_* = \lim_{C \to \infty} \frac{1}{lC} \sum_{i=1}^{l} \alpha_i^C, \quad \nu_* = \lim_{C \to 0} \frac{1}{lC} \sum_{i=1}^{l} \alpha_i^C \tag{42}$$

where $\nu_* > 0$ and $\nu_* \le 1$. Primal problem is given below:

$$\begin{aligned} \min_{\rho, w, b, \xi_i} \quad & -l\nu\rho + \sum_{i=1}^{l} \xi_i \\ \text{s.t. } y_i \left(x_i^T w + b \right) & \ge \rho - \xi_i, \quad \xi_i \ge 0, \ \forall i \\ & \frac{1}{2} \|w\|^2 = 1 \end{aligned} \tag{43}$$

Dual problem is:

$$\max_{\alpha_i, \lambda} \quad L_D = -\frac{1}{2\lambda} \sum_{i=1}^{l} \sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j x_i^T x_j - \lambda$$

$$\text{s.t.} \quad \nu = -\frac{1}{l} \sum_{i=1}^{l} \alpha_i, \quad \sum_{i=1}^{l} \alpha_i y_i = 0, \quad 0 \le \alpha_i \le 1, \ \forall i \tag{44}$$

$$\lambda = + \sqrt{\frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j x_i^T xj}. \tag{45}$$

3.16 Structured SVM (2004)

Structured SVM (Tsochantaridis et al. 2004; Joachims 2006; Joachims et al. 2009) generalizes the multi-class SVMs (Weston and Watkins 1999; Crammer and Singer 2002) to new

formulations that has structured outputs like sequences, strings, graphs, labeled trees or lattices. It uses the combined space with input and output features. The difference is in the outputs of SVM, the multi-class SVM has outputs like $k = 1, 2, 3, \ldots, m$ but Structured SVM has structured outputs. Structured SVM has applications in a number of areas like multi-label classification, supervised grammar learning and to label sequence learning etc.

Let the inputs be $x \in X$, outputs be $y \in Y$ (represents structured outputs) and training sample be $(x_1, y_1), (x_2, y_2), \ldots, (x_l, y_l)$. Let $\psi(x, y)$ represents the combined feature representation of inputs and outputs and $\delta\psi_i(y) \equiv \psi(x_i, y_i) - \psi(x_i, y)$.

The hard-margin problem is:

$$\begin{aligned} \text{SVM}_0 : \quad & \min_w \ \frac{1}{2}\|w\|^2 \\ & \forall i, \forall y \in Y \backslash y_i : \langle w, \delta\psi_i(y)\rangle \geq 1. \end{aligned} \tag{46}$$

where w is parameter vector and $\langle ., . \rangle$ represents the dot product. Problem with linear slack variable:

$$\begin{aligned} \text{SVM}_1 : \min_{w,\xi} \ & \frac{1}{2}\|w\|^2 + \frac{C}{l}\sum_{i=1}^{l}\xi_i, \quad \text{s.t. } \forall i,\ \xi_i \geq 0 \\ & \forall i, \forall y \in Y \backslash y_i : \langle w, \delta\psi_i(y)\rangle \geq 1 - \xi_i. \end{aligned} \tag{47}$$

Problem with quadratic slack variable (SVM_2) can be obtained when violations are penalized using $\frac{C}{2l}\sum_{i=1}^{l}\xi_i^2$.

For problems like language parsing, where $|Y|$ is large, zero-one loss function is not suitable. So for an arbitrary loss function δ, rescaling is used according to the loss incurred in each of the linear constraints. The violations are either multiplied by the loss, or equivalently, by the inverse loss. Thus the problem becomes:

$$\begin{aligned} \text{SVM}_1^{\Delta s} : \quad & \min_{w,\xi} \ \frac{1}{2}\|w\|^2 + \frac{C}{l}\sum_{i=1}^{l}\xi_i, \quad \text{s.t. } \forall i,\ \xi_i \geq 0 \\ & \forall i, \forall y \in Y \backslash y_i : \langle w, \delta\psi_i(y)\rangle \geq 1 - \frac{\xi_i}{\Delta(y, y_i)}. \end{aligned} \tag{48}$$

Dual problem SVM_0:

$$\begin{aligned} \max_{\alpha} \ & \sum_{i, y_i \neq y} \alpha_{iy} - \frac{1}{2}\sum_{i, y_i \neq y}\sum_{j, y_j \neq \bar{y}} \alpha_{iy}\alpha_{j\bar{y}}\left\langle \delta\psi_i(y), \delta\psi_j(\bar{y})\right\rangle \\ & \text{s.t.} \quad \forall i, \forall y \neq Y \backslash y_i : \quad \alpha_{iy} \geq 0. \end{aligned} \tag{49}$$

Dual problem for $\text{SVM}_1^{\Delta s}$ is same and need extra constraint as:

$$l\sum_{y_i \neq y}\frac{\alpha_{iy}}{\Delta(y, y_i)} \leq C, \quad \forall i \tag{50}$$

SVM_2 has same dual problem as SVM_0 after replacing inner product to

$$\left\langle \delta\psi_i(y), \delta\psi_j(\bar{y})\right\rangle + \delta_{i,j}\frac{l}{C\sqrt{\Delta(y_i, y)}\sqrt{\Delta(y_j, y)}} \text{where } \delta_{i,j} = 1 \text{ if } i = j, \text{ else } 0.$$

3.17 Multi-task SVM (2004)

Multi-task learning is the simultaneous learning of multiple, related but different tasks using shared representations, e.g., Isolet and SAD datasets are used for multi-task learning by Ji and Sun (2011). Multi-Task SVM (MTSVM, Evgeniou and Pontil 2004) extends the standard

SVM, i.e., single task learning SVM and proposes a new binary SVM for multi-task learning. Primal problem formulation for MTSVM is given by

$$\min_{w_0, v_t, \xi_{it}} \left\{ \sum_{t=1}^{T} \sum_{i=1}^{l} \xi_{it} + \frac{\lambda_1}{T} \sum_{t=1}^{T} \|v_t\|^2 + \lambda_2 \|w_0\|^2 \right\} \tag{51}$$

for all $i \in \{1, 2, \ldots, l\}$, $t \in \{1, 2, \ldots, T\}$ and $w_t = w_0 + v_t$, s.t. the following constraints

$$\begin{gathered} y_{it} \left(w_0 + v_t\right) \times x_{it} \geq 1 - \xi_{it} \\ \xi_{it} \geq 0, \end{gathered}$$

where T is number of learning tasks, l is number of data points, λ_1, λ_2 are positive regularization parameters and rest has usual meanings. And Dual problem formulation for MTSVM is given by

$$\max_{\alpha_{it}} \left\{ \sum_{i=1}^{l} \sum_{t=1}^{T} \alpha_{it} - \frac{1}{2} \sum_{i=1}^{l} \sum_{s=1}^{T} \sum_{j=1}^{l} \sum_{t=1}^{T} \alpha_{is} y_{is} \alpha_{jt} y_{jt} K_{st} \left(x_{is}, x_{jt}\right) \right\} \tag{52}$$

for all $i \in \{1, 2, \ldots, l\}$, $t \in \{1, 2, \ldots T\}$, s.t., the following constraints

$$0 \leq \alpha_{it} \leq C,$$

where

$$C = \frac{T}{2\lambda_1}, \quad \mu = \frac{T\lambda_2}{\lambda_1} \text{ and } K_{st}\left(x, z\right) = \left(\frac{1}{\mu} + \delta_{st}\right), \ s, t = 1, \ldots, T.$$

MTSVM, being a binary SVM, has been further extended to multi-class problems by Ji and Sun (2011, 2013).

3.18 Fuzzy proximal SVM (2004)

Fuzzy proximal SVM (FPSVM, Jayadeva et al. 2004) used the concept of FSVM (Lin and Wang 2002) and PSVM (Fung and Mangasarian 2001) to propose a new SVM where fuzzy membership is assigned to data points and data points are classified on the basis of proximity to two parallel planes (called proximal planes). Data points are clustered around the proximal planes which are kept as far from each other as possible.

Let the data to be classified be denoted by a set of l row vectors A_i $(i = 1, 2, \ldots, l)$ in n-dimensional real space $\mathbb{R}^n$, where $A_i = (A_{i1}, A_{i2}, \ldots, A_{in})$. Let $f_i \in \{-1, +1\}$ denotes the classes to which the data belongs, $w \in \mathbb{R}^n$ and $\gamma \in \mathbb{R}$. D is $l \times l$ diagonal matrix containing $+1$ or -1 as labels and y denotes error vector. Optimization problem is:

$$\min_{y, w, \gamma} \quad \frac{C}{2} \|Y\|^2 + \frac{1}{2} \left(w^T w + \gamma^2 \right) \tag{53}$$

$$\text{s.t.} \quad D\left(Aw - \mathbf{1}\gamma\right) + y = \mathbf{1} \tag{54}$$

where $Y = Sy$, $S = diag(s_1, s_2, \ldots, s_l)$ membership values are associated with a lower bound, i.e., $\sigma \leq s_i \leq 1, i = 1, 2, \ldots, l$.

Alternatively, Eq. (54) can be replaced by the constraint:

$$SD\left(Aw - \mathbf{1}\gamma\right) + Y = S\mathbf{1} \tag{55}$$

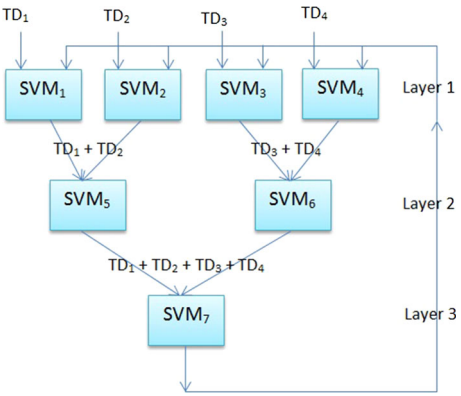

Fig. 10 Architecture of Cascade SVM: an example of Cascade SVM where the dataset is broken into four subsets and four SVMs are trained over them in the first layer. In second layer, support vectors of two SVMs are combined along with datasets and fed into a new SVM. Finally the results of SVM_5 and SVM_6 are combined along with datasets and fed to a new SVM_7 on the last layer. The support vectors from the SVM_7 are fed to the first layer and process is repeated until desired accuracy is obtained

Lagrangian is:

$$L(w, \gamma, Y, \beta) = \frac{C}{2} \|Y\|^2 + \frac{1}{2} \left\| \begin{bmatrix} w \\ \gamma \end{bmatrix} \right\|^2 - \beta^T \left[SD\left(Aw - \mathbf{1}\gamma\right) + Y - S\mathbf{1} \right] \tag{56}$$

where β is vector of Lagrangian multipliers.

Solving with KKT conditions, it gives

$$\beta = \left[I - SH \left(\frac{I}{C} + H^T S^T SH \right)^{-1} (SH)^T \right] CS\mathbf{1} \tag{57}$$

where $H = D\left[A \quad -\mathbf{1}\right]$.

3.19 Cascade SVM (2005)

Cascade SVM also called as parallel support vector machines (Graf et al. 2005) proposes an algorithm for solving an SVM problem, where algorithm can be parallelized using multiple SVMs. The dataset is broken into multiple subsets and an SVM is trained on each subset. The partial results (support vectors) from the different SVMs are combined together by taking two SVM at a time in multiple layers until a single set of support vectors is obtained, as shown in the Fig. 10. Single pass through this cascade of SVM produces very good accuracy but to obtain more accurate results, outputs from the last layer is fed back to the first layer. This process is repeated until desired accuracy is obtained. Since the output of first layer SVM is given to the second layer and so on, this is why, it is called as Cascade SVM. This is a divide and conquer strategy of solving the SVM problem, where in division step, it divides the problem into multiple subproblems and in conquer step, it combines the solutions from the subproblems and initializes a new SVM with that. Hsieh et al. (2014), has further extended the Cascade SVM to DC-SVM (Divide-and-Conquer SVM) which uses clustering of data in the division step so most of the support vectors of subproblems are also the support vectors of the combined problem.

3.20 GEPSVM (2006)

GEPSVM (Mangasarian and Wild 2006) is a new SVM problem formulation with two non-parallel hyperplanes. The data points are classified according to the proximity to one of the two non-parallel planes. This is different from the PSVM (Fung and Mangasarian 2001) and

FPSVM (Jayadeva et al. 2004) which also uses the proximity of data points to the proximal planes in classifying the data points, but these have parallel planes unlike GEPSVM. The non-parallel planes are the eigen vectors corresponding to the smallest eigen values of two related generalized eigen value problems.

Two non-parallel planes are:

$$x^T w^{(1)} + b^{(1)} = 0 \quad \text{and} \quad x^T w^{(2)} + b^{(2)} = 0 \tag{58}$$

GEPSVM minimizes the Euclidean distance of the planes form data points of classes $+1$ and -1, so the optimization problem of GEPSVM is:

$$\min_{w,b\neq 0} \frac{\|Aw + \mathbf{1}b\|^2 / \left\|[w, \quad b]^T\right\|^2}{\|Bw + \mathbf{1}b\|^2 / \left\|[w, \quad b]^T\right\|^2} \tag{59}$$

On simplification the above problem is reduced to

$$\min_{w,b\neq 0} \frac{\|Aw + \mathbf{1}b\|^2}{\|Bw + \mathbf{1}b\|^2} \tag{60}$$

Using Tikhonov regularization (Willoughby 1979; Evgeniou et al. 2000; Suykens and Vandewalle 1999),

$$\min_{w,b\neq 0} \frac{\left(\|Aw + \mathbf{1}b\|^2 + \delta \left\|[w, \quad b]^T\right\|^2\right)}{\|Bw + \mathbf{1}b\|^2}, \quad \delta > 0 \tag{61}$$

This in turn leads to Rayleigh Quotient (Parlett 1998) of the form

$$\min_{z\neq 0} \; z^T Gz / z^T Hz \tag{62}$$

where G, H are symmetric matrices in $\mathbb{R}^{(n+1)\times(n+1)}$ defined as:

$$G := [A \quad \mathbf{1}]^T [A \quad \mathbf{1}] + \delta I, \quad \text{for some} \quad \delta > 0,$$
$$H := [B \quad \mathbf{1}]^T [B \quad \mathbf{1}], \quad \text{and} \quad z := [w \quad b]$$

So the solution of problem (62) is:

$$Gz = \mu Hz, \quad z \neq 0$$

This gives one plane. Similarly second plane can be obtained by defining another problem by inter changing the roles of A and B in problem (59).

3.21 Multi-view SVM (2006)

Multi-view learning (Xu et al. 2013) is learning from single dataset using different feature sets, each representing one particular view, with a purpose to improve the generalization performance of the classifier. Multi-View SVM (MVSVM) was initially proposed by Farquhar et al. (2006) to learn two views by combining Kernel Canonical Correlation Analysis (KCCA) and SVM, and named it as SVM2K. SVM2K (Farquhar et al. 2006) uses following problem formulations. Suppose single dataset is represented by two views using feature projection ϕ_A with Kernel function K_A and feature projection ϕ_B with Kernel function K_B. Paired dataset using different views is given by set $S = \{(\phi_A(x_1), \phi_B(x_1), \ldots, \phi_A(x_l), \phi_B(x_l))\}$. Then the primal problem is given by

$$\begin{aligned}
\min_{w_A, w_B, \xi_i^A, \xi_i^B, \eta_i} \quad & \frac{1}{2}\|w_A\|^2 + \frac{1}{2}\|w_B\|^2 + C^A \sum_{i=1}^{l} \xi_i^A + C^B \sum_{i=1}^{l} \xi_i^B + D \sum_{i=1}^{l} \eta_i \\
\text{s.t.} \quad & |\langle w_A, \phi_A(x_i)\rangle + b_A - \langle w_B, \phi_B(x_i)\rangle - b_B| \le \eta_i + \epsilon \\
& y_i\left(\langle w_A, \phi_A(x_i)\rangle + b_A\right) \ge 1 - \xi_i^A \\
& y_i\left(\langle w_B, \phi_B(x_i)\rangle + b_B\right) \ge 1 - \xi_i^B \\
& \xi_i^A \ge 0, \quad \xi_i^B \ge 0, \quad \eta_i \ge 0, \quad \forall\, 1 \le i \le l.
\end{aligned} \tag{63}$$

Decision function is $h(x) = sign(f(x))$ where

$$f(x) = 0.5\left(\langle w_A, \phi_A(x)\rangle + b_A + \langle w_B, \phi_B(x)\rangle + b_B\right) \tag{64}$$

Dual problem is given by

$$\begin{aligned}
\max_{\alpha_i^A, \alpha_i^B, g_i^A, g_j^A, g_i^B, g_j^B} \quad & -\frac{1}{2}\sum_{i,j=1}^{l}\left(g_i^A g_j^A K_A(x_i, x_j) + g_i^B g_j^B K_B(x_i, x_j)\right) + \sum_{i=1}^{l}\left(\alpha_i^A + \alpha_i^B\right) \\
\text{s.t.,} \quad & g_i^A = \alpha_i^A y_i - \beta_i^+ + \beta_i^-, \; g_i^B = \alpha_i^B y_i + \beta_i^+ - \beta_i^-, \\
& \textstyle\sum_{i=1}^{l} g_i^A = 0 = \sum_{i=1}^{l} g_i^B, \\
& 0 \le \alpha_i^{A/B} \le C^{A/B} \\
& 0 \le \beta_i^{+/-}, \beta_i^+ + \beta_i^- \le D
\end{aligned} \tag{65}$$

with the functions

$$f_{A/B}(x) = \sum_{i=1}^{l} g_i^{A/B} K_A(x_i, x) + b_{A/B}. \tag{66}$$

MVSVM has been studied and further extended by researchers, like Xie and Sun 2015 have extended it to Twin SVM.

3.22 Twin SVM (2007)

Twin SVM (TWSVM, Jayadeva and Chandra 2007) is an extension of PSVM (Fung and Mangasarian 2001) via generalized eigen values. As the name suggest, TWSVM is a binary classifier that solves two related small sized SVM problems and give two non-parallel planes. A new test point is classified as per the closest plane. TWSVM is faster than standard SVM and give good generalization results (Jayadeva and Chandra 2007). It helps in automatically discovering two-dimensional projections of the data. Each of the non-parallel planes is closer to one of class and is as far as possible from the other class. TWSVM is comprised of two quadratic programming problems in which the objective function of one problem is represented by the data points of one class and constraints are represented by the data points of other class. This leads to two smaller SVM problems given below.

$$\begin{aligned}
\text{(TWSVM 1)} \quad & \min_{w^{(1)}, b^{(1)}, q} \quad \frac{1}{2}\left(Aw^{(1)} + \mathbf{1}_1 b^{(1)}\right)^T \left(Aw^{(1)} + \mathbf{1}_1 b^{(1)}\right) + C_1 \mathbf{1}_2^T q \\
\text{s.t.} \quad & -\left(Bw^{(1)} + \mathbf{1}_2 b^{(1)}\right) + q \ge \mathbf{1}_2, \quad q \ge 0 \\
\text{(TWSVM 2)} \quad & \min_{w^{(2)}, b^{(2)}, q} \quad \frac{1}{2}\left(Bw^{(2)} + \mathbf{1}_2 b^{(2)}\right)^T \left(Bw^{(2)} + \mathbf{1}_2 b^{(2)}\right) + C_2 \mathbf{1}_1^T q \\
\text{s.t.} \quad & \left(Aw^{(2)} + \mathbf{1}_1 b^{(2)}\right) + q \ge \mathbf{1}_1, \quad q \ge 0
\end{aligned} \tag{67}$$

where C_1, C_2 are penalty parameters and $\mathbf{1}_1$, $\mathbf{1}_2$ are unit vectors. First term represents distance of data points form hyperplane and second term represents errors. Here data points belonging

to classes $+1$ and -1 are represented with A and B. Let l_1 and l_2 denotes number of data points in A and B. So size of A and B is $l_1 \times n$ and $l_2 \times n$. Dual form is:

$$\begin{aligned} &\text{(DTWSVM 1)} \max_{\alpha} \quad \mathbf{1}_2^T \alpha - \frac{1}{2} \alpha^T G \left(H^T H\right)^{-1} G^T \alpha \\ &\text{s.t.} \qquad\qquad 0 \leq \alpha \leq C_1 \\ &\text{(DTWSVM 2)} \max_{\gamma} \quad \mathbf{1}_1^T \gamma - \frac{1}{2} \gamma^T P \left(Q^T Q\right)^{-1} P^T \gamma \\ &\text{s.t.} \qquad\qquad 0 \leq \gamma \leq C_2 \end{aligned} \tag{68}$$

where α and γ are dual variables.

$$\begin{aligned} \text{where} \quad H &= [A \quad \mathbf{1}_1], \quad G = [B \quad \mathbf{1}_2] \\ P &= [A \quad \mathbf{1}_1], \quad Q = [B \quad \mathbf{1}_2] \end{aligned}$$

TWSVM has been further extended to ν-TWSVM (Peng 2010) which introduces parameter ν into TWSVM for controlling the margin, errors and consequently support vectors. All extensions and applications of TWSVM can be found in Jayadeva et al. (2017).

3.23 SloppySVM (2009)

SloppySVM (Stempfel and Ralaivola 2009) proposes a new SVM which can learn from the sloppily labeled data. And to achieve this ability, SloppySVM formulation proposes a new non-convex objective functional which is a uniform estimate of noise free objective function of standard C-SVM.

Let $Y = \{-1, +1\}$ be the target space and X be the input space. Standard C-SVM problem with sample $S = \{(x_i, y_i)_{i=1}^l\}$ of size l is given by

$$\min_{w,b} \quad \frac{1}{2} \|w\|^2 + \frac{C}{l} \sum_{i=1}^{l} \xi \left(y_i w^T x_i\right) \tag{69}$$

where $C \in \mathbb{R}^+$ is a regularization parameter, $\gamma = y w^T x$ is the margin and $\xi(\gamma) = \max(0, 1 - \gamma)$ is hinge loss function.

Let η^+ is the probability of label +1 being represented as -1 and η^- is the probability of label -1 being represented as +1. Let $\eta = [\eta^+ \quad \eta^-]$ is the noise vector, $\sigma = [\sigma_1, \sigma_2, \ldots, \sigma_l]$ is a Rademacher vector, with $P(\sigma_i = -1) = \eta^{y_i} = 1 - P(\sigma_i = +1)$ and noisy version of the sample space is $S^\sigma = \{(x_i, y_i)\}_{i=1}^l$. Let $K_\eta = \dfrac{1}{1 - \eta^+ - \eta^-}$ and $\widehat{\xi}(\gamma, y) = K_\eta \left[\left(1 - \eta^{-y}\right) \xi(\gamma) - \eta^y \xi(-\gamma)\right]$. So the SloppySVM is:

$$\min_{w} \quad \frac{1}{2} \|w\|^2 + \frac{C}{l} \sum_{i=1}^{l} \widehat{\xi} \left(\sigma_i y_i w^T x_i, \sigma_i y_i\right). \tag{70}$$

3.24 Capped l_p-norm SVM (2017)

Capped l_p-norm SVM (CappedSVM, Nie et al. 2017) proposes a new SVM problem formulation using capped l_p-norm based loss function, to tackle the challenge of data outliers. Data outliers have large residue values so capped norm helps the CappedSVM to eliminate these outliers during the training process. CappedSVM is also extended to multi-class problems. Unlike one-vs-one, one-vs-rest which solve several binary problems, CappedSVM

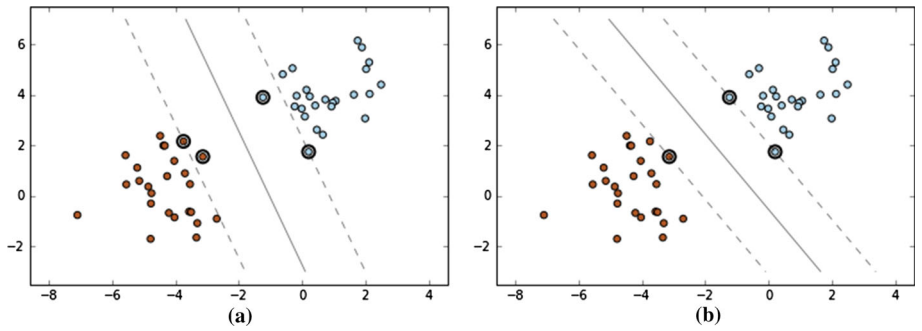

Fig. 11 Smaller the value of C, larger is the margin. **a** C = 0.1. **b** C = 10

solves single optimization problem, like, CSSVM. Authors have proved the utility of primal formulation and its binary formulation is given below (Nie et al. 2017):

$$\min_{w,b} \sum_{i=1}^{l} \xi_{ch}\left(w, b\,|x_i\right) + \gamma \left\|w\right\|_2^2 \tag{71}$$

Here γ is regularization parameter and $\xi_{ch}\left(w, b\,|x_i\right)$ is capped l_p-norm given by

$$\xi_{ch}\left(w, b\,|x_i\right) = \min\left(\epsilon, max\left(0, 1 - y_i w^T x_i\right)^p\right).$$

where ϵ is capping value and p is any real value. Multi-class problem for c classes is given by

$$\min_{W,b,M\geq 0} \quad \sum_{i=1}^{l} \min\left(\left\|W^T x_i + b - y_i - y_i \circ m_i\right\|_2^p, \epsilon\right) + \gamma \left\|W\right\|_F^2 \tag{72}$$

where m_i is a slack variable to encode the unilateral loss of x_i, $W \in \mathbb{R}^{n\times c}$, $b \in \mathbb{R}^{c\times 1}$, $m_i \in \mathbb{R}^{c\times 1}$, $M \in \mathbb{R}^{c\times n}$ with the i-th column as m_i and

$$\xi_{chm}\left(W, b|x_i\right) = \min\left(\min_{m_i\geq 0} \left\|W^T x_i + b - y_i - y_i \circ m_i\right\|_2^p, \epsilon\right),$$

is capped l_p-norm loss function for multi-class problem.

3.25 *C*-SVM

All primal SVMs which have C penalty parameter (also called regularization coefficient) are known as C-SVMs (Fan et al. 2008), e.g., SVMs represented by Eqs. (6) and (37) are C-SVM. The role of C is to control the trade off between margin size [represented by first part in Eqs. (6) and (37)] and classification errors [represented by second part in Eqs. (6) and (37)]. Importance of C: Larger the value of C, larger is the error part and smaller is the margin size. That means for larger values of C, over-fitting may occur, see Fig. 11. If value of C is not too large, smaller is the error part and larger is the margin size.

This is not a new SVM rather only a name given to existing SVMs, to distinguish them from the ν-SVM and its variations. In ν-SVMs, parameter C is removed and a more meaningful parameter ν is added (for ν and ν-SVM, see Sect. 3.8). Thus, in order to distinguish the

standard SVM from the ν-SVM, the former is called as C-SVM and it is considered under the evolution of SVM only for the clarity of readers.

3.26 Kernel SVM

For non-linearly separable data, soft margin classifiers does not generalize well and produces a lot of mis-classification errors. In such cases, Kernels are used, which transform the data from the Input Space to the Feature Space where the data is linearly separable. With Kernels, it does not need to calculate the transformations/mappings of data into the Feature Space rather it needs only the dot product of mappings (Eq. 1) so the overhead of using Kernels is much less than with other non-linear classifiers. Figure 2, represents the geometrical view of Kernel-SVM (Chang and Lin 2011) where a non-linearly separable data is separated using the Kernel.

Just like C-SVM, Kernel SVM is not a specific SVM but name given to a class of dual SVM formulations which uses non-linear Kernels so Kernel SVM has been covered here only for the clarify of readers (Table 1) .

4 Problem solvers

Once the SVM problem is formulated, the next task is to solve the problem using certain methods called SVM solvers. The method/solver to solve a particular SVM formulation depends upon a number of factors and some of these are:

(i) Problem formulation properties: convex and non-convex nature, strong convexity, smooth and non-smooth nature, primal or dual nature, constrained and unconstrained nature etc.
(ii) Dataset properties: #data_points >> #features or #features >> #data_points, sparsity, binary or multi-class data etc.
(iii) What do we expect: low or high accuracy solution, accuracy is important or time etc.

Different types of optimization problems, depending upon the nature of objective function and constraints, are depicted in Fig. 12. SVM problem is an optimization problem, which may be convex problem (Crammer and Singer 2002) or non-convex problem (Stempfel and Ralaivola 2009). It depends upon the problem formulation. It is easy to solve the convex optimization problem than non-convex problems because every local optimum is a global optimum in a convex optimization problem. Therefore, SVM problem is generally formulated as a convex optimization problem. Generally, our problem lies in the category of constrained convex optimization problem as per the Fig. 12. Convex optimization problem (COP) are of two types: unconstrained convex optimization problem (UCOP) and constrained convex optimization problem (CCOP). CCOP are further of three types: equality constrained (EC), inequality constrained (IC) and hybrid constrained (HC).

Gradient based methods are generally used for solving the UCOP problems. CCOP can be converted to UCOP and then gradient based methods can be applied to them. In literature, generally gradient based methods (Zanni et al. 2006; Yc and Su 2016) are used for solving the SVM problem with occasional use of other methods like Interior Point Methods (Ferris and Munson 2002). All these methods are taken from the Mathematics, highlighting the inter-disciplinary nature of Machine Learning, so it is beyond the scope of this paper to discuss all methods for solving the optimization problems. This subsection gives an idea about SVM solvers and discusses some methods for solving SVM problems. Review of SVM solvers can

Table 1 Summary of SVM problem formulations

Name	Idea	Applications	Remarks
Hard SVM (Boser et al. 1992)	This SVM separates the data points using a hyperplane without allowing any mis-classification errors and that's why know as hard margin SVM. Because of hard margin, this is not applicable (in primal formulation) to problems where data can't be separated completely by the hyperplane	This is a standard SVM, used for general class of problems and is not suitable to specific requirements like noisy or outliers data applications	This was the first SVM
			This has been improved by Soft SVM so it is hardly used
Soft SVM (Cortes and Vapnik 1995)	This is an extension of hard margin SVM which introduced a slack variable into the hard margin SVM to allow some mis-classification errors. Thus, it can be applied to problems where data can't be separated completely by the hyperplane	Being extension of hard SVM, this is also a standard SVM, used for general class of problems and is not very good to specific requirements like noisy or outliers data applications	This is the most popular and widely used formulation of SVM
		This is used in large and sparse data applications and acts as a very good tool in image and document classification problems	
One-versus-Rest (OVR, Vapnik 1998)	This is a multi-class SVM formulation which reduces the multi-class problem to binary problem by considering one class at a time and separating it from the rest of the classes. Thus for m classes, it solves m binary SVM problems and evaluates same number of decision functions to find the class of a test point	This is used for multi-class data applications	This is a simple, one of the mostly used and probably the earliest multi-class SVM implementation but its disadvantage is that it needs to train and evaluate multiple SVMs and decision functions, respectively.
Transductive SVM (Joachims 1999)	Unlike, standard SVM, this considers both training and test data for training the model to reduce the test errors	This is used for text classification problems	This targets a particular dataset and try to have a minimum test error on that dataset. This is not suitable for other applications because of different goals

Table 1 continued

Name	Idea	Applications	Remarks
Weston Watkins SVM (Weston and Watkins 1999)	This is a multi-class SVM formulation which solves a single optimization problem and evaluates m (number of classes) decision functions	This is used for multi-class data applications	Unlike OVR, it solves a single optimization problem but, like OVR evaluates m (number of classes) decision functions to predict the class of a test data point
One-versus-One (OVO, Kressel 1999)	This is a multi-class SVM which trains a binary SVM for each possible pair of classes, leading to $\frac{m(m-1)}{2}$ binary SVM optimization problems	This is used for multi-class data applications	Unlike OVR, WWSVM it solves $\frac{m(m-1)}{2}$ optimization problems and evaluates same number of decision functions to predict the class of a test data point
Least Squares SVM (LSSVM, Suykens and Vandewalle 1999)	This proposes an SVM which can be put in the least squares form, with equality constraints	This is a standard SVM, used for general class of problems	Due to equality constraints, it solves a set of linear equations to find the solution, unlike, standard SVM which solves quadratic programming. Thus faster to solve it In LSSVM, number of support vectors are proportional to number of errors. LSSVM behaves like Hard SVM when data points are linearly independent (Ye and Xiong 2007) LSSVM has been extended by Li et al. (2006) which assigns a degree of importance to each training data point, this idea is similar to Fuzzy SVM (Lin and Wang 2002)
ν-SVM (Schölkopf et al. 2000)	This proposed a new SVM formulation for classification and regression problems where a new parameter ν replaces the existing C parameter in classification problem	This is successfully used by researchers for image boundary detection and in personal email assistant	ν roughly controls the fraction of support vectors and thus its value is limited to [0, 1], unlike, C which can take any positive value

Table 1 continued

Name	Idea	Applications	Remarks
			This is difficult to optimize as compared with standard SVM and is not scalable for large problems
Directed Acyclic Graph SVM (DAGSVM, Platt et al. 2000)	It uses a rooted binary directed acyclic graph which has $\frac{m(m-1)}{2}$ internal nodes and m leaves where m is the number of classes. Each internal node represents a binary SVM classifier which tests two classes. The leaf nodes represent the output class labels. For m classes, it needs only $m-1$ evaluations of decision functions	This is used for multi-class classification problems	This is similar to OVO technique and like OVO, it solves $\frac{m(m-1)}{2}$ binary SVM classifier but to classify a test data point, it uses a different strategy and needs only $m-1$ evaluations of decision functions
Smooth SVM (Lee and Mangasarian 2001)	This proposes a smooth variant of the standard SVM to handle the non-differentiability issue	This is a binary SVM which can use faster optimization methods available for smooth and strongly convex problems	Smooth SVM is strongly convex and infinitely differentiable so it enjoys globally, quadratically convergent solutions using a fast Newton-Armijo algorithm
Lagrangian SVM (LSVM, Mangasarian and Musicant 2001)	This is obtained by reformulating the standard primal linear SVM and taking an implicit Lagrangian of its dual form	This is a binary SVM	This gives an unconstrained differentiable convex function and thus easier to solve
Proximal SVM (PSVM, Fung and Mangasarian 2001)	This SVM finds two parallel hyperplane (called proximal hyperplanes), unlike, standard SVM which finds single. This classifies data points on the basis of proximity to the two parallel planes, unlike, standard SVM which assigns them to one of the half spaces generated by separating hyperplane	This is a binary SVM	Proximal hyperplanes are pushed much apart from each other because these are no longer bounding planes (see Fig. 9)
Crammer Singer SVM (CSSVM, Crammer and Singer 2002)	This is a multi-class SVM formulation which trains a single SVM classifier and evaluates m (number of classes) decision functions to classify a test data point	This is used for multi-class classification problems	The main difference between CSSVM and other problems is that CSSVM does not need to explicitly write down constraints

Table 1 continued

Name	Idea	Applications	Remarks
			Unlike OVO, OVR, DAGSVM (which solve multiple problems) and like WWSVM, it solves a single optimization problem but unlike DAGSVM, OVO and like WWSVM, it evaluates m (number of classes) decision functions to classify a test data point
Fuzzy SVM (FSVM, Lin and Wang 2002)	This assigns each data point a fuzzy membership to the labeled class and reformulates the SVM problem where contributions of points are used to find the separating hyperplane	It is suitable for fault diagnosis and applications where data points have modeled characteristics	FSVM is useful in reducing the effect of outliers and noise
			FSVM has been further extended to multi-class problems (see, Abe and Inoue 2002; Abe 2015)
$E\nu$-SVM (Prez-Cruz et al. 2003)	This extends ν-SVM by using the idea that ν can't take all values from 0 to 1, either because the training set is non-separable or because the classes are unbalanced	This is successfully used by researchers for image boundary detection and in personal email assistant	ν helps to control the margin, errors and the support vectors
Structured SVM (Tsochantaridis et al. 2004)	This problem formulation handles the structured outputs like sequences, strings, graphs, labeled trees or lattices	It has applications in a number of areas like multi-label classification, supervised grammar learning and to label sequence learning etc	This is different from all other SVMs because its outputs are structured
			This is a generalization of CSSVM and WWSVM
Multi-Task SVM (MTSVM) (Evgeniou and Pontil 2004)	MTSVM extends the standard SVM, i.e., single task learning SVM to multi-task learning	This is useful when multiple learning tasks, having some commonalities, need to be solved at the same time	MTSVM has been further extended to multi-class problems in Ji and Sun (2011, 2013)

Table 1 continued

Name	Idea	Applications	Remarks
Fuzzy Proximal SVM (FPSVM, Jayadeva et al. 2004)	This uses the concept of FSVM and PSVM to propose a new SVM where fuzzy membership is assigned to data points in the labeled class and data points are classified on the basis of proximity to two parallel planes	It is suitable for fault diagnosis and applications where data points have modeled characteristics	This is useful in reducing the effect of outliers and noise
Cascade SVM (Graf et al. 2005)	The dataset is broken into multiple subsets and an SVM is trained on each subset. The partial results (support vectors) from the different SVMs are combined together by taking two SVM at a time in multiple layers until a single set of support vectors is obtained, as shown in the Fig. 10	This is a binary SVM	This is based on divide and conquer strategy of solving the SVM problem
			Hsieh et al. (2014), has further extended the Cascade SVM to DC-SVM (Divide-and-Conquer SVM) which uses clustering of data in the division step so most of the support vectors of subproblems are also the support vectors of the combined problem
GEPSVM (Mangasarian and Wild 2006)	This proposes a new formulation with two non-parallel hyperplanes and data points are classified according to the proximity to one of the two non-parallel planes. The non-parallel planes are the eigen vectors corresponding to the smallest eigen values of two related generalized eigen value problems	This is a binary SVM	Unlike PSVM which have parallel proximal planes, it has non-parallel proximal planes

be found in Bottou and Lin (2007), Shawe-Taylor and Sun (2011) and for general discussion of optimization methods for large-scale machine learning, refer to Bottou et al. (2016).

Large-scale problems (SVMs) are solved using iterative methods and there is a trade off between the computational complexity of each iteration and number of iterations required by the method to solve problem up to the desired accuracy. Generally, smaller the complexity of each iteration, slower is the convergence and larger is the number of iterations taken by the

Table 1 continued

Name	Idea	Applications	Remarks
Multi-View SVM (MVSVM) (Farquhar et al. 2006)	MVSVM extends the idea of Multi-view learning to SVM	This is helpful in problems where data is obtained from different sources or obtained using different feature extractors and show heterogeneous properties	It aims to improve the generalization performance of the classifier through multiple views of the problem
			MVSVM has been further extended to Twin SVM by Xie and Sun (2015)
Twin SVM (TWSVM, Jayadeva and Chandra 2007)	This is an extension of PSVM via generalized eigen values which solves two related small sized SVM problems and give two non-parallel planes	This is a binary SVM	PSVM finds two parallel proximal planes but TWSVM, like, GEPSVM finds two non-parallel planes
			TWSVM has been extended to ν-TWSVM (Peng 2010), and been studied extensively. All extensions and applications can be found in the book (Jayadeva et al. 2017)
SloppySVM (Stempfel and Ralaivola 2009)	SloppySVM formulation proposes a new non-convex objective functional which is a uniform estimate of noise free objective function of standard C-SVM	This can be efficiently used only for sloppily labeled data, i.e., labeling of data can be wrong	This is a non-convex formulation so difficult to solve than standard SVM and thus this is rarely used
CappedSVM (Nie et al. 2017)	CappedSVM has proposed a new problem formulation using capped l_p-norm based loss function, to tackle the challenge of data outliers	It is useful for tackling the data outliers challenge, like in fault detection, and multi-class problems	It is available as binary and multi-class formulations
C-SVM (Fan et al. 2008)	This is a name given to standard SVM, which uses penalty parameter C, to distinguish it from the ν-SVM	This is a class of problems and have same applications as standard SVM like soft SVM	This is only a terminology and not a specific SVM formulation

method to solve the problem (Lin et al. 2008). For example, first order methods (like Gradient Descent method) have low iteration complexity (like in Gradient Descent, only gradient is calculated) but they converge very slowly and takes large number of iterations to reach the desired accuracy (Shalev-Shwartz et al. 2007). On the other hand, second order methods have high computational complexity (like in Newton method, it requires computing and inverting the Hessian matrix) of each iteration but they converge very fast and take lesser number of iterations (Lin et al. 2008) to reach the same level of accuracy.

Table 1 continued

Name	Idea	Applications	Remarks
Kernel SVM (Chang and Lin 2011)	This is a name given to a class of dual SVM problem formulations which uses non-linear Kernels	This is used for non-linearly separable data applications where it is not possible to find a separable hyperplane in the Input Space This is further very helpful when primal problem have issues, like, non-differentiability, and can't be solved efficiently	Any convex SVM problem can be converted into corresponding Kernel SVM

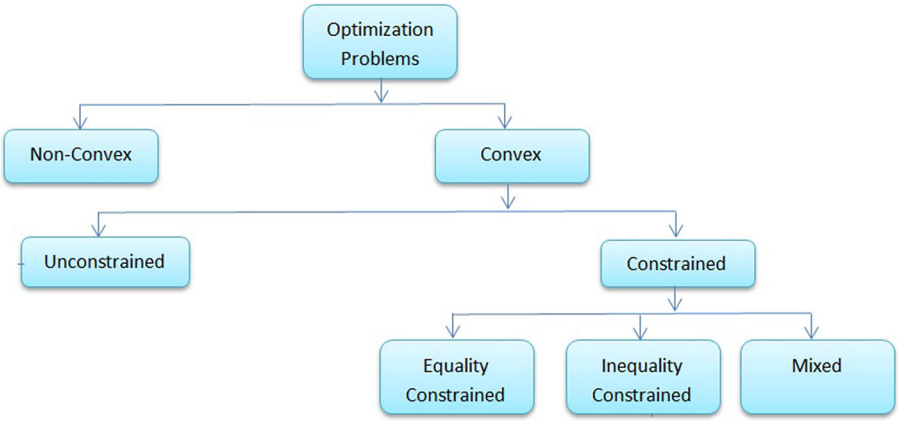

Fig. 12 Different types of Optimization problems

For an iterative method, solution in the $(k+1)$th iteration is given by

$$w_{k+1} = w_k + d_k s_k, \quad d_k > 0 \tag{73}$$

where s_k is the step direction and d_k is the step size along that direction. The process of finding d_k is also called as line search because of finding the value of step size along the step direction. The descent method is a method in which the value of a function decreases with every iteration, i.e., suppose function f is to be minimized then for $(k+1)$th iteration

$$f(w_{k+1}) < f(w_k)$$

except when w_k is optimal. s_k in a descent method must satisfy the following equation:

$$\nabla f(w_k)^T s_k < 0 \tag{74}$$

This means that the search direction should make an acute angle with the negative of gradient of function. Such a direction is known as a descent direction. A general algorithm for a descent optimization method is given by Algorithm 1. The step size and step direction can be found by separate methods or by a single method, and other exit criteria can be tried.

Algorithm 1 General Descent Method for Optimization

1: Initialization: Take initial solution w_0 and tolerance ϵ_0
2: **for** k=1,2,… **do**
3: Find descent direction s_k.
4: Line Search: Choose a step size $d_k > 0$.
5: Update solution using Eq. (73).
6: **if** $\|\nabla f(w_{k+1})\|_2 \leq \epsilon_0$ **then**
7: Exit.
8: **end if**
9: **end for**

Some methods for finding the step size are given below:

4.1 Exact line search method

In this method (Keerthi and DeCoste 2005; Lee and Roth 2015; Nie et al. 2017) of finding the step size (d), an exact optimization problem is solved to minimize f along the ray $\{w + ds | d > 0\}$:

$$d = \underset{v \geq 0}{argmin} \quad f(w + vs) \tag{75}$$

The limitation of exact line search method is that it takes a lot of overhead for large problems. So generally this method is not used for large problems.

4.2 Backtracking line search

Backtracking line search (Lin et al. 2007; Chang et al. 2008; Yuan et al. 2010; Lee and Roth 2015) is an inexact line search method where instead of solving the exact optimization problem, a set of values for step size, are tried along the ray $\{w+ds|d > 0\}$. The backtracking line search algorithm is given in Algorithm 2.

Algorithm 2 Backtracking Line Search Method

1: Initialization: Let step direction be s_k and take $d = 1$, $\alpha \in (0, 1)$ and $\beta \in (0, 1)$.
2: **while** $f(w_k + ds_k) \leq f(w_k) + \alpha d \nabla f(w_k)^T s_k$ **do**
3: $d = \beta d$
4: **end while**
5: Set $d_k = d$.

This method is called as backtracking line search method because it starts with unit size for step size and then decreases its value by a factor of β. Small value of β (0.1) corresponds to very crude search and large value (0.9) to less crude. The typical values of α are between 0.01 and 0.3, i.e., 1 to 30% decrease of prediction in f is accepted based on linear exploration.

4.3 Constant step size

This is the simplest method of finding the step length. In order to get the value of step size no optimization problem is solved rather a constant value (generally $d = 1$) is used as a step size. For larger value, the convergence is uncertain. For smaller value, the convergence is slower but certain in convex functions.

4.4 Lipschitz and strong convexity constants

Lipschitz (L) and strong convexity (μ) constants can also be used for calculating step sizes. These constants helps in easy calculation of step sizes but depends on correct estimation of L and μ. The main disadvantage of this method is that sometimes, these constants are not available. Using these constants, step size can be calculated as (Schmidt et al. 2016):

$$d = \frac{1}{L} \quad \text{and} \quad d = \frac{2}{L+\mu}, \tag{76}$$

These expressions for d, can change depending upon the implementation.

4.5 Trust region method

In trust region (TR) method (Lin et al. 2008; Fan et al. 2008; Lin et al. 2014; Tsai and Lin 2014), a step direction and a step size is selected, then if the step is successful in some sense then the step is accepted else rejected and another step direction and step size is found. Here an approximate model (m_k) of a function (f) to be minimized, is taken and solved to get the solution. The region around the current iterate in which the model has good approximation of function, is called as trust-region. A basic Trust Region algorithm is given by Algorithm 3. Typically used values are $\eta_v = 0.75$, $\eta_s = 0.25$, $\sigma_i = 2$ and $\sigma_d = 0.5$.

Algorithm 3 Basic Trust-Region Method

1: Given: Trust-region radius $\triangle_0 > 0$ and initial solution w_0.
2: Set $k = 0$, $\eta_v \in (0, 1)$, $\eta_s \in (0, \eta_v)$, $\sigma_i \geq 1$ and $\sigma_d \in (0, 1)$.
3: **repeat**
4: Build a quadratic model $m_k(w_k + p)$ for $f(w_k + p)$ as

$$m_k(w_k + p_k) = f(w_k) + g^T p_k + \frac{1}{2} p_k^T H p_k \tag{77}$$

where g and H are gradient and Hessian respectively.
5: Solve Trust-Region problem approximately for p_k,

$$\min_{\|p\| \leq \triangle_k} \quad m_k(w_k + p)$$

6: Find the ratio,

$$\rho_k = \frac{f(w_k) - f(w_k + p_k)}{f(w_k) - m_k(w_k + p_k)} \tag{78}$$

7: **if** $\rho_k \geq \eta_v$ ("very successful" step), **then**
8: set $w_{k+1} = w_k + p_k$ and $\triangle_{k+1} = \sigma_i \triangle_k$.
9: **else if** $\rho_k \geq \eta_s$ ("successful" step), **then**
10: set $w_{k+1} = w_k + p_k$ and $\triangle_{k+1} = \triangle_k$.
11: **else if** $\rho_k < \eta_s$ ("unsuccessful" step), **then**
12: set $w_{k+1} = w_k$ and $\triangle_{k+1} = \sigma_d \triangle_k$.
13: **end if**
14: k++
15: **until** convergence

Some optimization methods for finding the step direction are given below:

4.6 Gradient descent method

Gradient descent method (Cauchy 1847; Zanni et al. 2006; Yc and Su 2016) is also known as Steepest Descent (SD) method, is a first order optimization method. SD method is based on the observation that a function decreases fastest on moving along the direction of negative gradient of the function. In this method, the step direction is taken as negative of gradient, i.e.,

$$s_k^{GD} = -\nabla f(w_k) \tag{79}$$

and step size can be determined by either exact or backtracking line search. So substituting the values of step size and step direction into the Algorithm 1, it gives Gradient Descent Algorithm. SD method generally exhibits approximately linear convergence. On using backtracking line search, the parameters α and β have noticeable effect over the convergence rate but it does not have dramatic effect. Advantage of this method is its simplicity, it does not require second order derivative and every iteration is inexpensive. Disadvantages are: it is often slow, depends upon scaling and can't handle non-differentiable problems.

4.7 Newton method

For Newton's method (Chang et al. 2008; Lin et al. 2008; Mangasarian 2001), the step direction is given by

$$s_k^N = -\left(\nabla^2 f(w_k)\right)^{-1} \nabla f(w_k) \tag{80}$$

is called as Newton step which is a descent direction. When step size is taken as $d = 1$, then it is called as pure Newton method else called as damped Newton or guarded Newton method. On replacing the step direction and method to find step size, in Algorithm 1, it gives algorithm for the Newton method. In general, Newton method has quadratic rate of convergence when $d = 1$. Advantages of Newton method are that it is affine invariant to the choice of coordinates or condition number, scales with problem size, has fast convergence and it is independent of algorithm parameters. Disadvantages of Newton method are that it need second order derivative, high cost of computing and storing the Hessian and calculating step direction.

4.8 Gauss–Newton method

Gauss–Newton method (Gibson 2011) is an iterative optimization method for least squares problems which are of the form:

$$\min_w \quad f(w) = \frac{1}{2}\sum_{j=1}^{m} r_j^2(w) \tag{81}$$

where $w \in \mathbb{R}^n$ and $r_j : \mathbb{R}^n \to \mathbb{R}$ are residual functions.

Suppose $r : \mathbb{R}^n \to \mathbb{R}^m$ is a residual vector defined by

$$r(w) = (r_1(w), r_2(w), \ldots, r_m(w))$$

So Eq. (81) becomes:

$$\min_w \quad f(w) = \frac{1}{2} \parallel r(w) \parallel^2 \tag{82}$$

From Eq. (82), the gradient and Hessian can be expressed as:

$$\begin{aligned}\nabla f(w) &= \sum_{j=1}^{m} r_j(w)\nabla r_j(w) \\ &= J(w)^T r(w) \qquad (83)\end{aligned}$$

$$\nabla^2 f(w) = J(w)^T J(w) + \sum_{j=1}^{m} r_j(w)\nabla^2 r_j(w) \qquad (84)$$

For small value of $\nabla^2 r_j(w)$ or residuals ($r_j(w)$), Eq. (84) simplifies to,

$$\nabla^2 f(w) = J(w)^T J(w) \qquad (85)$$

Now using gradient and Hessian, for least squares problem, Eq. (80) gives Gauss–Newton step direction for the $(k+1)$th iteration as:

$$s_k^G = -\left(J(w_k)^T J(w_k)\right)^{-1} J(w_k)^T r(w_k) \qquad (86)$$

For small residual problem, the convergence rate is close to quadratic otherwise not even linear is guaranteed.

4.9 Levenberg–Marquardt method

Levenberg (1944) introduced a parameter λ into the Hessian and step direction given by Eq. (86) changed to

$$s_k^L = -\left(J(w_k)^T J(w_k) + \lambda I\right)^{-1} J(w_k)^T r(w_k) \qquad (87)$$

where I is the identity matrix. Later (Marquardt 1963) replaced the identity matrix with diagonal matrix of Hessian matrix. So Eq. (87) gives Levenberg–Marquardt step direction as:

$$s_k^{LM} = -\left(H + \lambda\, diag(H)\right)^{-1} g \qquad (88)$$

where $H = J(w_k)^T J(w_k)$ and $g = J(w_k)^T r(w_k)$. λ is called the Levenberg–Marquardt parameter and is used to make the approximate Hessian $(H + \lambda\, diag(H))$ positive definite. This method is robust with respect to larger residual problems and poor initial solution. For $\lambda = 0$, Levenberg–Marquardt method behaves as Gauss–Newton method and for large value of λ, it behaves as Steepest Descent method.

Levenberg–Marquardt (LM) method can be extended as Levenberg–Marquardt Trust Region (LMTR, Moré 1978; Ranganathan 2004; Nocedal 1999; Chauhan et al. 2017b) method. In LM method λ is arbitrarily changed but in LMTR instead of changing the λ, trust region radius is changed. Trust Region method or line search methods are used to ensure sufficient decrease in the function. Trust Region method can be used with any method that uses quadratic model of the function for finding the solution.

4.10 Quasi-Newton method

Quasi-Newton method (Nocedal 1999; Stempfel and Ralaivola 2009) is an alternative to Newton method. In this method, instead of calculating the true Hessian ($\nabla^2 f_k$) an approximation

of Hessian (B_k) is used and updated at each step. So the Quasi-Newton search direction is given by

$$s_k^Q = -B_k^{-1} g \tag{89}$$

Advantage of Quasi-Newton method is that it avoids calculation of second order derivatives. It has super-linear rate of convergence. There are many methods to update the Hessian value at each step. Two of the most famous formula for updating B_k are:

Symmetric Rank (SR1, Nocedal 1999):

$$B_{k+1} = B_k + \frac{(y_k - B_k p_k)(y_k - B_k p_k)^T}{(y_k - B_k p_k)^T p_k} \tag{90}$$

where $y_k = \nabla f_{k+1} - \nabla f_k$ and $p_k = w_{k+1} - w_k$.

And BFGS (Stempfel and Ralaivola 2009), is named after its inventors, Broyden, Fletcher, Goldfarb, and Shanno:

$$B_{k+1} = B_k - \frac{B_k p_k p_k^T B_k}{p_k^T B_k p_k} + \frac{y_k y_k^T}{y_k^T p_k} \tag{91}$$

4.11 Subgradient method

Subgradient, subderivative, and subdifferential refers to the derivative of a non-differentiable convex function. Subgradient methods (Luss and d'Aspremont 2009; Shalev-Shwartz et al. 2007; Yuan et al. 2012) are optimization methods for solving the non-differentiable convex functions. The update rule for Subgradient method is given by

$$w_{k+1} = w_k - d_k g_k \tag{92}$$

where g_k is the subgradient of function f at w_k and for step size d_k.

Step size rule: (1) Fixed step: d_k is taken constant, (2) Fixed length: $d_k \|g_k\|_2$ is taken constant, (3) Diminishing: $d_k \to 0, \quad \sum_{k=1}^{\infty} d_k = \infty$.

Advantage of Subgradient method is that it leads to a very simple algorithm. Disadvantage of Subgradient method are that the convergence can be very slow and there is no good stopping criterion for it.

4.12 Conjugate gradient method

Conjugate gradient (CG) methods (Wen et al. 2003; Chapelle 2007; Lin et al. 2008; Fan et al. 2008) are the optimization methods which are less reliable than Newton method but can be used to solve very large problems. It is proposed by Hestenes and Stiefel (1952) for linear systems in 1950s but has been extended to non-linear systems by Fletcher and Reeves (1964) in the 1960s. Advantage of Conjugate method is that it does not need any matrix storage, it needs to calculate only objective function and its derivative which make it suitable for very large problems. It is faster than Steepest Descent method. Inexact and Truncated Newton methods use it to approximate the Newton step. Conjugate Gradient method can be accelerated using preconditioning.

According to Fletcher and Reeves (1964), step size is determined by any of the existing methods and step direction is found using the following formula and algorithm is given by Algorithm 4:

$$s_{k+1} = -\nabla f_{k+1} + \beta_{k+1} s_k \tag{93}$$

$$\text{where} \quad \beta_{k+1} = \frac{\nabla f_{k+1}^T \nabla f_{k+1}}{\nabla f_k^T \nabla f_k} \tag{94}$$

Algorithm 4 Fletcher and Reeves CG Method

1: Given: Initial solution w_0.
2: Calculate $f_0 = f(w_0)$, $\nabla f_0 = \nabla f(w_0)$. Let $s_0 = -\nabla f_0$ and k =0.
3: **while** $\nabla f_k \neq 0$ **do**
4: Compute the step size d_k (using some method) and find $w_{k+1} = w_k + d_k s_k$
5: Evaluate ∇f_{k+1}
6: Evaluate β_{k+1} using Eq. (94).
7: Evaluate p_{k+1} using Eq. (93).
8: k++
9: **end while**

FR method has many variations and Polak–Ribiere (PR) method (Stefano and Mikhail 2011) is one of the them. It varies only in updating value of β_{k+1}. So according to PR method the update formula is:

$$\beta_{k+1} = \frac{\nabla f_{k+1}^T (\nabla f_{k+1} - \nabla f_k)}{\nabla f_k^T \nabla f_k} \tag{95}$$

4.13 Truncated Newton method

Truncated Newton (TN) methods are an alternative to Newton method where Newton system is approximately solved leading to early termination. TN methods are less reliable than Newton method but these can handle very large problems. TN methods do not calculate Hessian so also known as Hessian-free optimization. Conjugate Gradient method and Preconditioned Conjugate Gradient (Lin et al. 2008) methods are examples of Truncated Newton methods.

4.14 Proximal gradient method

Proximal Gradient (PG) method (Xu and Yin 2015) is an optimization method for unconstrained problem where objective function can be split in two convex components. The proximity of a convex function $h(w)$ is given as:

$$prox_h(w) = \underset{u}{argmin} \quad \left(h(u) + \frac{1}{2}\|u - w\|_2^2 \right) \tag{96}$$

Suppose problem is given by

$$\min_w \quad f(w) = g(w) + h(w) \tag{97}$$

where g is convex, differentiable and $dom\ g = \mathbb{R}^n$. h is convex with inexpensive proximity. According to PG method the solution in the $(k+1)$th iteration is given by

$$w_{k+1} = prox_{d_k h}\left(w_k - d_k \nabla g\left(w_k\right)\right) \tag{98}$$

where $d_k > 0$ is step size, constant or determined by line search. Fast Proximal Gradient and Dual Proximal Gradient methods are the variations of PG method.

4.15 Recent algorithms

Generally, a learning algorithm is a combination of various methods (step size methods and step direction methods) and optimization strategies. Some examples of recently used algorithms are: SVM^{perf} (Joachims 2006), Pegasos (Shalev-Shwartz et al. 2007), TRON (Trust RegiOn Newton method, Lin et al. 2008), SDCA (Stochastic Dual Coordinate Ascent) method (Shalev-Shwartz and Zhang 2013b), SVM-ALM algorithms (Nie et al. 2014), SAG (stochastic average gradient) method (Schmidt et al. 2016), PCDM (Parallel Coordinate Descent Method, Richtárik and Takáč 2016), PCDM (Parallel Coordinate Descent Method, Richtárik and Takáč 2016) and CappedSVM (Nie et al. 2017) etc. It is not possible to discuss all the latest algorithms so only three of them are discussed below:

TRON In TRON, L2-regularized and L2-loss SVM problem is solved which is given below:

$$\min_{w} \quad f(w) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\left[\max\left(0, 1 - y_i w^T x_i\right)\right]^2 \tag{99}$$

Let quadratic model of $f(w)$ is:

$$m(s) = \nabla f(w)^T s + \frac{1}{2} s^T \nabla^2 f(w) s \tag{100}$$

TRON approximates $m(s)$ with trust region radius $\triangle$:

$$\min_{\|s\| \le \triangle} \quad m(s) \tag{101}$$

It uses the following ratio for updating the trust region radius:

$$\mu = \frac{f(w+s) - f(w)}{m(s)} \tag{102}$$

TRON uses modified Newton method with Conjugate Gradient and Trust Region method for each iteration. It uses Co-ordinate Descent strategy to make the problem light weight. The algorithm for TRON is given in Algorithm 5.

Algorithm 5 TRON for L2RL2 SVC (Lin et al. 2008)

1: Given: Initial solution w_0, trust-region radius $\triangle_0$ and η.
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: **if** Exit Criteria **then**
4: Stop.
5: **end if**
6: Solve the subproblem (101) to get solution s using Conjugate Gradient method.
7: Find the ratio μ, using Eq. (102).
8: **if** $\mu > \eta$ **then**
9: Update solution: $w = w + s$
10: **end if**
11: Adjust $\triangle$, according to μ.
12: **end for**

Pegasos Another example is Pegasos (Shalev-Shwartz and Singer 2006), it uses Subgradient method with Stochastic and Batch learning strategies. It solves L2RL1 SVM. Since L1-loss function is non-differentiable so it uses Subgradient Method. It can work for Batch

learning and Stochastic learning but here only Batch learning is used. This algorithm solves the L2RL1 loss SVM, given below:

$$\min_{w} \quad f(w; B) = \frac{1}{2}\|w\|^2 + C\sum_{i\in B}\max(0, 1 - y_i w^T x_i) \tag{103}$$

Algorithm 6 Pegasos for L2RL1 SVC (Shalev-Shwartz et al. 2007)

1: Given: Initial solution w, s.t. $w \leq \sqrt{Cl}$.
2: **for** $k = 1, 2, \ldots$ **do**
3: Let the training subset is B.
4: Calculate learning rate η.
5: Update solution using Eq. (104).
6: Use projection as given in Eq. (105), to ensure $w \leq \sqrt{Cl}$.
7: **end for**

where B is a training subset. Solution update for Eq. (103) is given by

$$w = w - \eta\nabla^s f(w; B) \tag{104}$$

where $\eta = (Cl)/k$, is the learning rate, k is iteration number, $\nabla^s f(w; B) = w - C\sum_{1-y_i w^T x_i>0} y_i x_i$, is the subgradient of f. Pegasos also uses projection of its solution as:

$$w = \min\left(1, \frac{\sqrt{Cl}}{\|w\|}\right) w \tag{105}$$

The algorithm for Pegasos is given by Algorithm 6. It takes $O(1/\epsilon)$ iterations to get ϵ accurate solution and total run time of algorithm is $O(d/(C\epsilon))$, where d is the bound on number of non-zeros features in each data point.

SAG (stochastic average gradient) method (Schmidt et al. 2016): This method is applicable to problems of the following type

$$\min_{w\in\mathbb{R}} \quad g(w) = \frac{1}{l}\sum_{i=1}^{l} f_i(w) = \frac{\lambda}{2}\|w\|^2 + \frac{1}{l}\sum_{i=1}^{l} l_i(w), \tag{106}$$

where $f_i(w)$ is finite, smooth and convex, and given by

$$f_i(w) = \frac{\lambda}{2}\|w\|^2 + l_i(w) \tag{107}$$

The $(k+1)$th iterate used by SAG is

$$w^{k+1} = w^k - \frac{d}{l}\sum_{i=1}^{l} y_i^k, \tag{108}$$

where y_i^k is given by

$$y_i^k = \begin{cases} f_i'(w^k) & \text{if } i = i_k, \\ y_i^{k-1} & \text{otherwise.} \end{cases} \tag{109}$$

Here i_k is randomly selected index during each iteration. Basic SAG algorithm is given by

SAG gives linear convergence and is suitable for large data applications.

Algorithm 7 Basic SAG Method (Schmidt et al. 2016)

Given: step size d.
Initialization: $s = 0$, $y_i = 0$ for $i = 0, 1, 2, \ldots, l$
for $k = 0, 1, 2, \ldots$ **do**
  Randomly select i from $\{i = 0, 1, 2, \ldots, l\}$
  $s = s - y_i + f_i^{'}(w)$
  $y_i = f_i^{'}(w)$
  $w = w - \frac{d}{l}s$
end for

5 Comparative study

In this section, we have discussed the comparative study of different SVMs. The results are discussed from the literature and from our experimental study.

5.1 Results from literature

According to literature, LIBLINEAR (Fan et al. 2008), Pegasos (Shalev-Shwartz et al. 2007) and SVM^{perf} (Joachims 2006) are the state of the art solvers with LIBLINEAR being the most efficient (Fan et al. 2008), popular and widely used. LIBLINEAR uses TRON (Trust Region Newton method, Lin et al. 2008) to solve L2 regularized L2 loss primal SVM, PCD (Primal Coordinate Descent, Chang et al. 2008) to solve L1 regularized L2 loss primal SVM and DCD (Dual Coordinate Descent, Hsieh et al. 2008) algorithm to solve L2-regularized L1- and L2-loss dual SVM. SVM^{perf} uses cutting plane technique to solve L2-regularized L1-loss primal SVM, and Pegasos uses an algorithm which alternates between gradient descent steps and projection steps to solve L2-regularized L1-loss primal SVM. The comparisons of these state of the art solvers have been presented in the Fig. 1 of Sect. 4 in Fan et al. (2008) (see latest version), which compares training time against testing accuracy for different linear solvers. Recently, Nie et al. (2014) have proposed an SVM-ALM algorithm to solve L2-regularized L1-, L2- and Lp norm loss primal problem with linear computational cost, to tackle the big data challenge. SVM-ALM is faster than LIBLINEAR, and experimental results are reported in Tables 1 and 2 of Sect. 4 in Nie et al. (2014). SVM-ALM (Augmented Lagrange Multipliers) uses gradient descent method for finding step direction and exact line search for finding the step size. The beauty of SVM-ALM algorithm is in transforming the Lp loss primal SVM problem using Augmented Lagrange Multipliers into a form which is very easy to solve. This highlights the point that it is not only a method but many other factors (see, beginning of Sect. 4 for such factors) which make an algorithm efficient.

5.2 Results from experimental study

5.2.1 Experimental setup and implementation details

The experiments have been performed on three datasets[1], named, adult (also known as a9a), mushroom and rcv1, on single node MacBook Air (8 GB 1600 MHz DDR3, 1.6 GHz Intel Core i5, 256 SSD). Each dataset has been divided randomly into 80 and 20% for training data and test data, respectively. The experiments compare seven prominent variants of SVM,

[1] datasets used in the experiments are available at link: https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/.

named, Soft SVM, Smooth SVM, Least Squares SVM, Twin SVM, Eν-SVM, Proximal SVM and Capped SVM, using two criteria of test accuracy versus epochs (one epoch refers to one pass through the dataset) and test accuracy versus training time (in seconds). For fair comparison, primal formulation of SVMs have been solved using same algorithmic structure/framework with differences only in the problem formulations and corresponding gradient values. The training algorithm uses mini-batch SGD (Stochastic Gradient Descent) method (Zhang 2004) to determine the step direction with decreasing step size, determined by $d_0/(1+d_0Ck)$, for the kth epoch with initial step size d_0 and regularization parameter C. The training algorithm uses mini-batch size of five randomly selected data points. The problem formulations for each SVM uses squared slack variable, i.e., squared error for fair comparison, e.g., using L2-loss in Soft SVM instead of L1-loss. Further, regularization parameter (C) has been set to 0.1 for all the C-SVMs, including both subproblems of Twin SVM and parameter ν has been set to 0.06 (mushroom and adult) and 0.006 (rcv1.binary) for ν-SVM. For Smooth SVM, the smoothing parameter (α) has been set to five and for Capped lp norm SVM, p is set to two and capping value ϵ is set to one. Since Twin SVM has two subproblems so the training time and epochs for Twin SVM has been taken as the sum of time and sum of epochs taken by subproblems, respectively.

5.2.2 Results and discussions

First, this is to be noted that there are several challenges in comparative study of different SVMs and these are: (a) different scope and applications areas of different SVMs, e.g., SloppySVM is useful for applications where data is sloppily labeled; Transductive SVM is specifically proposed for text classification problems; Fuzzy SVM and CappedSVM are suitable for applications where data contains noise and data outliers; Structured SVM is useful for applications where the output labels are structured like strings; MVSVM and MTSVM are used for multi-view learning and multi-task learning, respectively, etc. (b) Direct improvements of one over other, e.g., Soft SVM is an improvement over Hard SVM, Eν-SVM is an improvement over ν-SVM etc. (c) Binary and multi-class data, e.g., Soft SVM, Least Square SVM and Smooth SVM are mainly binary classifiers which have been extended to multi-class data using one-vs-one, one-vs-all etc. (d) every problem can't be solved using every solver. One interesting point about solving a specific SVM problem formulation is that being an optimization problem, each problem can be solved using a number of methods. Different factors play role in the selection of a solver for a specific SVM, as discussed at the beginning of Sect. 4. Thus, only prominent variants of binary primal SVMs have been studied experimentally using mini-batch SGD method. For the comparative study of multi-class SVMs, refer to Tables V.2–V.4 of Section V in Hsu and Lin (2002). One another interesting point is that from the experimental results, we can't generalize that a particular SVM is the best. This is as per the "no free lunch" theorem, and the point that there are several trade-offs and factors (see, Sect. 4) that we have to consider while solving SVMs.

The experimental results are represented by Figs. 13, 14, 15, which plot test accuracy versus time (in seconds) and test accuracy versus epochs. For mushroom dataset, Soft SVM, CappedSVM, Smooth SVM, Least Square SVM follow each other closely and give the best result on accuracy versus epoch. But for accuracy versus time, Soft SVM outperforms other SVMs, then followed by CappedSVM. For adult dataset, all the SVMs perform closely except the Twin SVM. Soft SVM outperforms other SVMs with a small margin. For rcv1.binary dataset, Soft SVM outperforms other SVMs, followed by Least Square SVM and then remaining SVMs. Thus, it is clear from the experiments that Soft SVM gives the best generalization results with respect to both training time and number of epochs, and perhaps that's why Soft

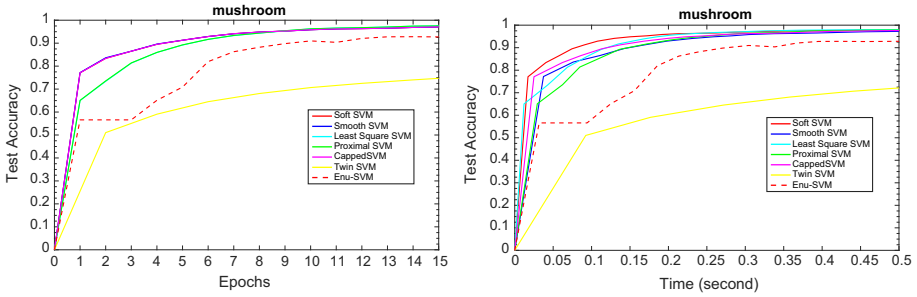

Fig. 13 Mushroom, *#instances: 8124, #features: 112*

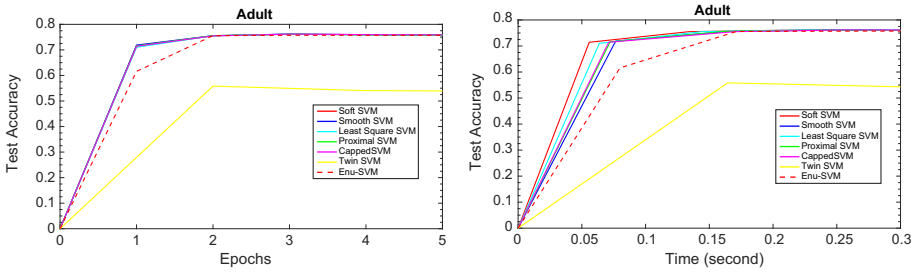

Fig. 14 Adult, *#instances: 32,561, #features: 123*

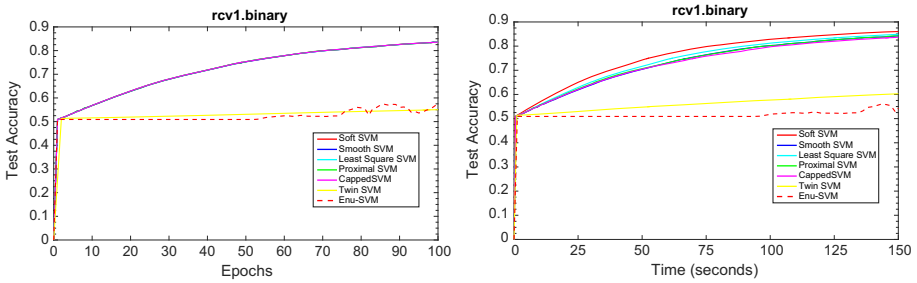

Fig. 15 rcv1, *#instances: 20,242, #features: 47,236.*

SVM is the most popular and widely used SVM problem formulation. Further, it is clear from the results that Twin SVM and Eν-SVM does not perform well on given datasets, and Twin SVM lags far behind on mushroom and adult datasets.

6 Current challenges and research directions

6.1 Big data challenge

With the advancements in the technology, data has taken the form of 'Big Data' and the meaning of 'big' in big data is continuously changing and will continue to change. Earlier, few hundred instances were considered to be a large dataset because of the limited computing resources and limited data available at that time. Later, meaning of large dataset changed to

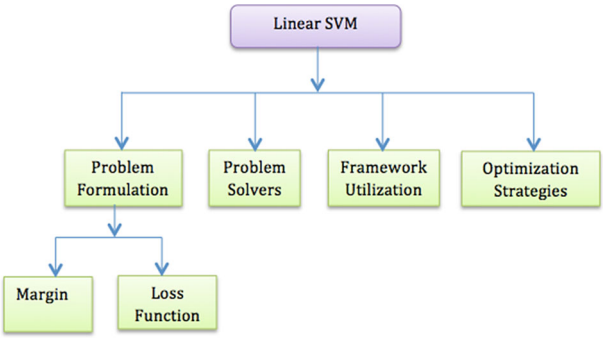

Fig. 16 Improvement areas for linear SVM classification

thousands of instances. Now, large dataset refers to millions of instances. In order to make use of this huge data, efficient machine learning algorithms are needed to handle these large datasets. Moreover, these algorithms are also needed to be scalable to deal with increasing sizes of datasets. *Thus now a days, the biggest challenge before SVC (which are shared with machine learning) is to develop efficient and scalable learning algorithms to deal with big data*.

To solve the big data challenge, we have identified different areas in SVC, most of which are also common with other areas in machine learning and these are discussed in the next subsection.

6.2 Areas of improvement

Classification is one technique in Machine Learning which is used to assign labels to test data points from a set of predefined values with the help of a model which is trained on labeled data. Mathematically, classification problem is an optimization problem where margin between different classes is maximized while keeping the classification errors minimum. While solving the classification problem, there are two objectives: first, training speed of SVM models on the training data set and second, accuracy of prediction of labels on test data set. So speed and accuracy are two goals while solving the classification problems but there is a trade off between the speed and accuracy of algorithms. If simple algorithms are used to increase the speed then it does not give good accuracy of classification. On the other hand when complex algorithms are used for training the model, it increases the accuracy but training speed decreases, e.g., Steepest Descent method is fast but its convergence rate is slow as compared with Newton method. It depends upon the application, whether to sacrifice speed or accuracy. Both training speed and test accuracy of an algorithm depends upon these factors: Problem Formulations, Problem Solvers, framework/platform utilization and optimization strategies, as depicted in Fig. 16. So to make Linear SVM classification efficient and scalable, the challenge is to handle these factors effectively.

6.2.1 Problem formulations

Problem formulation is very important in SVM, the solvers to be used in an SVM problem also depends on problem formulation to some extent. SVM can be formulated as convex (Lee and Mangasarian 2001; Crammer and Singer 2002; Sun and Xie 2016) as well as non-convex optimization problem (Stempfel and Ralaivola 2009). Generally SVM is formulated as a

convex optimization problem because there is no problem of local minimum as every local minimum is a global minimum. SVC optimization problem has two components: margin and loss functions/errors. The trade off between margin and errors is controlled by parameter C as discussed in C-SVM (Sect. 3.25).

(a) *Margin*
This component is used to control the distance between separating hyperplanes. Larger the margin, better is the generalization of classifier. Currently, L1R ($\|w\|$, Zhu et al. 2003; Mangasarian 2006) and L2R ($\|w\|^2$, Lin et al. 2008; Keerthi et al. 2008) are widely used in the literature for representing margin in the SVM problem.

(b) *Loss functions*
This component introduced in soft margin SVM as a slack variable. It helps to minimize the mis-classification errors of classifier. Commonly, L1 loss (Franc and Sonnenburg 2008; Teo et al. 2010; Shalev-Shwartz et al. 2007) and L2 loss (Hsieh et al. 2008; Lee and Lin 2013; Fan et al. 2008) functions (see, Eq. 2) are widely used in literature. Other loss functions (e.g., Liu et al. 2016) can be used, it depends upon the problem formulation (see Teo et al. 2007, for complete list of loss functions).
These two components combined together form the SVM problem (see, Sect. 3 for different problem formulations). So the current challenge is to make appropriate choices for margin and errors or propose new ones to have an effective SVM problem. Inappropriate formulations can lead to local minimum problems, differentiability issues, slower and less accurate methods.

6.2.2 Problem solvers

Problem Solvers, again play an important role to get faster and accurate results. There are many methods available for solving the optimization problems as discussed in Sect. 4. It depends upon problem formulation (to some extent) as which method is to be used for solving it. From literature, it is observed that different researchers have solved same problem with different solvers, has lead to different results (Lin et al. 2008; Keerthi and DeCoste 2005 solve the same problem). Today challenge is to find efficient solvers which could produce accurate and faster results.

6.2.3 Problem solving strategies/approaches

Approaches play a critical role in solving an SVM problem using an SVM solver. The problem formulation may be very good and problem solver may be very efficient too but still our algorithm may not perform well, due to lack of suitable optimization strategies e.g., Suppose there is large sparse data problem, if sparsity of problem is not utilized then the training of this optimization technique could be very slow. Similarly using decomposition methods, stochastic approximation and problem shrinking etc., can make the problem solving process faster. Some of the strategies for machine learning (SVM) problems are:

(a) *Exploit problem structures*
The problem structures like sparsity (Steinwart 2004; Lazar 2010; Cotter et al. 2013) of the problem can be exploited to make learning more efficient. There are many problems like document classification where most of the features are zero. So the problem solver should utilize this characteristic, while training the model which can help in faster training of the problem. Similarly, handling the data in memory also affects the SVM solvers like storing data feature wise (Fan et al. 2008) or instance wise.

(b) *Decomposition approaches*
According to this approach, the problem is decomposed into smaller problems and then subproblems are solved using some solver to get the solution of original problem (Kao et al. 2004; Lin 2001; Chang et al. 2000), e.g., Suppose there are 10,000 parameters in a problem, then it could be solved by first taking first 100 parameters, fixing the remaining and solving this subproblem, then next 100 parameters can be solved and repeating this process until all the parameters are solved. This constitutes one iteration, the process can be repeated until convergence. One extreme case of decomposition approach is Coordinate Descent approach (Hsieh et al. 2008; Chang et al. 2008; Fan et al. 2008; Wright 2015), according to this approach, problem is solved for one variable, keeping all other variables fixed. This approach of optimization helps in overcoming implementation issues of memory and CPU. No doubt this approach takes more iterations and slower convergence but it makes the problem feasible to implement because with large number of features (variables), it might not be possible to process the algorithm due to large memory and CPU requirements.
Coordinate descent (Hsieh et al. 2008; Chang et al. 2008; Fan et al. 2008; Wright 2015), SMO (Platt 1998) and Block Optimization (Xu and Yin 2013; Blondel et al. 2013) are three examples of decomposition approaches. This approach helps in solving the large problems. The general algorithm for Coordinate Descent approach is given in Algorithm 8. Advantages of Coordinate Descent approach: Parallel versions are available; No extra storage vector needed; No other pesky parameters are required here; It works well for very large problems and it's easy to implement. Disadvantages of Coordinate Descent approaches: Tricky if single variable optimization is hard; Convergence theory could be very complex; It can be slower near optimum.

Algorithm 8 Coordinate Descent Approach

1: Given: Initial solution $w_0 = \left(w_0^1, w_0^2, \ldots, w_0^n\right)$.
2: **while** Exit Criteria **do**
3: Choose index $i \in \{1, 2, \ldots, n\}$, for the w_k^i to be optimized.
4: Take w_k^i as a variable and rest variables as constants in objective function f.
5: Solve the subproblem for w_k^i using some solver:
$\min_{w_k^i} \quad f(w_k)$
6: Update w_k for the selected coordinate and keep other coordinates fixed.
7: **end while**

(c) *Stochastic approximation approach*
Generally the parameters are optimized over all the data instances in each iteration but in stochastic approximation (Shalev-Shwartz and Zhang 2013b; Bottou 2012; Wang et al. 2012; Bottou 2010; Ladicky and Torr 2011), the parameters are optimized over one instance at a time. Stochastic Approximation approach makes each iteration independent of data points and reduces the iteration complexity. Thus, this approach is very helpful in solving large-scale problems. This is because instead of going through all instances in the dataset, it takes only one instance during each iteration. There are other variations of this approach as 'mini-batch' approach (Tak et al. 2013; Shalev-Shwartz and Zhang 2013a) where instead of taking one instance during each iteration, it takes a subset of training instances.

(d) *Problem shrinking strategy*
While optimizing different variables (parameters), it is observed that some of the variables does not change or remain zero. Such variables can be left from the optimization process until rest of the variables reach their approximate optimum values. This is called shrinking strategy (Joachims et al. 1999; Fan et al. 2008; Keerthi and Decoste 2006). In this way, the number of variables to be minimized during intermediate iterations is reduced. This makes the training faster.
So the challenge is to devise new approaches and use existing ones in the problem solvers to make the learning faster and more accurate.

6.2.4 Platforms/frameworks

Platform/framework represents the environment that supports the execution of the SVM problem. By utilizing this environment, SVM problems can be further improved. From Fig. 16, it is clear that for large problems, speed and accuracy depends upon computing framework/platform also. Some examples of using the platform/framework are given below: (Yuan et al. 2012; Lin et al. 2014)

(a) *Parallel and distributed computing*
With the help of parallel computing (Collobert et al. 2002; Zanghirati and Zanni 2003; Zanni et al. 2006; Cao et al. 2006; Woodsend and Gondzio 2009; Manno et al. 2015), some tasks can be done in parallel on either multiple cores of a system or on clusters of computers. This makes the training process faster, but obviously inter-communication between different processing units is a bottleneck here.

(b) *In place memory calculations*
Reading and writing data, takes more time than the processing of data. So problems which need multi-stage processing, instead of reading and writing the data to disk after every stage, if the consecutive calculations are carried out in the memory then that could make the process faster while dealing with very large data problems. Apache Spark (Lin et al. 2014) uses the concept of in place memory calculations which makes it approximately 100 times faster than the Hadoop.
So now a days challenge is to exploit the existing frameworks to make algorithms more efficient.

6.3 Research directions

As per the Fig. 16, there are four areas of improvement which can help us in tackling big data problems. Now a days, platform utilization and optimization strategies, are the hot research topics in machine learning. Thus, the current research directions in linear SVM classification (which are shared with machine learning) are:

6.3.1 Stochastic approximation algorithms

For last 5 years, there have been extensive research in this area (see Sect. 6.2.3 for Stochastic Approximation Approach). Researchers have come up with a large number of algorithms using stochastic approximations (SA), especially with first order methods like gradient descent (GD) method (Shalev-Shwartz and Zhang 2013a, b; Li et al. 2014b; Defazio et al. 2014; Needell et al. 2014; Schmidt et al. 2016) as a solver. In addition, SA has also been used with second order methods like quasi-Newton (Byrd et al. 2016). SA with GD forms stochastic gradient descent (SGD) method and leads to variance problem in the gradient values. So,

the research around SGD has been focused around reducing variance (e.g., Chauhan et al. 2017a). On the other hand, research around SA with second order methods has been focused on incorporating curvature information in method for faster convergence. This approach is suitable to solve problems with large number of data points.

6.3.2 Coordinate descent algorithms

This is one of the decompositional approach (see Sect. 6.2.3) for dealing with big data problems. For last few years, there have been good work in this area also. Coordinate Descent (CD) approach formulates a subproblem by taking one parameter as a variable and rest as constants, which is much smaller and thus much easier to solve. A lot of algorithms have been proposed using CD approach (Wright 2015) and its variations like block coordinate descent (Lu and Xiao 2015), with first order and seconder methods. This is generally applied to high dimensional primal or dual problems.

6.3.3 Proximal algorithms

Proximal algorithms (PA, Parikh and Boyd 2014) are suitable for non-smooth, constrained, large-scale, or distributed convex problems and recently, they produced good results with high dimensional optimization problems. With PA, the main operation is to calculate the proximity operator (see Sect. 4.14) where we solve a small optimization problem. Current research has been involved in using PA for first order methods [like proximal gradient method (Xiao and Zhang 2014)], second order methods [like quasi-Newton methods (Becker and Fadili 2012)] and their accelerated versions (Zhang and Gu 2016).

6.3.4 Parallel/distributed algorithms

Parallel and distributed algorithms (Recht et al. 2011; Li et al. 2014a; De et al. 2015; Yang et al. 2016) utilize the computing resources to handle the big data problems (like CPU cores of a system and clusters of computers) and can work with problems where datasets are extremely large and can't fit onto a single machine. The major bottleneck in these algorithms is the communication overhead. Researchers have been working on it for a long but still there is lot of scope in this area. This seems to be the hottest area in machine learning to handle the big data problems and in the coming time, research will probably focus in this direction because of increasing dataset sizes which eventually can go beyond the capability of single machine.

6.3.5 Hybrid algorithms

Hybrid algorithms combine two or more of the above approaches. It is obvious that if we want to develop efficient and scalable algorithms to tackle big data problems, all these approaches have to be combined together. A lot of research is already going in this direction also, like, combination of stochastic approximation and coordinate descent approaches have been studied in Xu and Yin (2015), Chauhan et al. (2017a).

7 Conclusions

Linear SVM is a very good tool in machine learning because this is much faster than non-linear Kernel SVM and at the same time, provides similar accuracy for high dimensional problems like document classification. Researchers have proposed a variety of SVM problem formulations, along with solvers and optimization strategies to solve them. Some of them are improvements over the other, like, 'Soft SVM' is an improvement over the 'Hard SVM'. Some of the SVMs have different contexts of application, like, SloppySVM is useful for applications where data is sloppily labeled; Transductive SVM is specifically proposed for text classification problems; Fuzzy SVM and CappedSVM are suitable for applications where data contains noise and data outliers; Structured SVM is useful for applications where the output labels are structured like strings; MVSVM and MTSVM are used for multi-view learning and multi-task learning, respectively, etc. Some of the SVMs are binary classifiers and need techniques like one-versus-one and one-versus-rest etc. to handle the multi-class problems, e.g., Soft SVM, Fuzzy SVM and Twin SVM etc. Some SVMs can handle the multi-class data using direct formulations like Crammer Singer SVM and Weston Watkins SVM. Amongst all the SVMs, Soft margin SVM or Soft SVM is the most popular and widely used SVM formulation due to faster training and better generalization results. That's why when one refers to SVM that means the Soft SVM. According to literature, SVM-ALM is the fastest primal solver and LIBLINEAR is the most popular and widely used tool which provides primal and dual, both solvers and uses TRON, PCD (Primal Coordinate Descent) and DCD (Dual Coordinate Descent) methods.

Now a days, the biggest challenge before linear SVM classification (which is shared with machine learning) is to develop efficient and scalable learning algorithms, to deal with 'Big Data'. To solve this challenge, extensive research is going in four directions: stochastic approximation algorithms, coordinate descent algorithms, proximal algorithms and parallel and distributed algorithms. First two, stochastic approximation and coordinate descent algorithms, try to reduce the problem size and computations by considering fewer parameters or fewer instances during each iteration of learning algorithm. Proximal algorithms are used for solving large non-smooth convex problems with separable components and parallel and distributed algorithms try to exploit the computing resources to handle big data problems. Amongst these, parallel and distributed computing looks to have a great future scope.

Acknowledgements First author would like to thank Ministry of Human Resource Development, Government of INDIA, for providing fellowship (University Grants Commission–Senior Research Fellowship) to pursue his Ph.D. work.

References

Abe S (2015) Fuzzy support vector machines for multilabel classification. Pattern Recogn 48(6):2110–2117

Abe S, Inoue T (2002) Fuzzy support vector machines for multiclass problems. In: European symposium on artificial neural networks, pp 113–118

Beck A, Tetruashvili L (2013) On the convergence of block coordinate descent type methods. SIAM J Optim 23(4):2037–2060

Becker S, Fadili M (2012) A quasi-newton proximal splitting method. In: Advances in neural information processing systems 25: 26th annual conference on neural information processing systems 2012. Proceedings of a meeting held December 3–6, 2012, Lake Tahoe, Nevada, pp 2627–2635

Blondel M, Seki K, Uehara K (2013) Block coordinate descent algorithms for large-scale sparse multiclass classification. Mach Learn 93(1):31–52

Boser BE, Guyon IM, Vapnik VN (1992) A training algorithm for optimal margin classifiers. In: Proceedings of the 5th annual ACM workshop on computational learning theory, ACM Press, pp 144–152
Bottou L (2010) Large-scale machine learning with stochastic gradient descent. In: Lechevallier Y, Saporta G (eds) Proceedings of the 19th international conference on computational statistics (COMPSTAT'2010), Springer, Paris, pp 177–187. http://leon.bottou.org/papers/bottou-2010
Bottou L (2012) Neural networks: tricks of the trade. 2nd edn, Springer, Berlin, pp 421–436
Bottou L, Lin CJ (2007) Support vector machine solvers. Science 3(1):1–27
Bottou L, Curtis FE, Nocedal J (2016) Optimization methods for large-scale machine learning. arXiv:1606.04838
Bredensteiner EJ, Bennett KP (1999) Computational optimization: a tribute to olvi mangasarian, vol I, Springer, Boston, chap Multicategory Classification by Support Vector Machines, pp 53–79
Byrd RH, Hansen SL, Nocedal J, Singer Y (2016) A stochastic quasi-newton method for large-scale optimization. SIAM J Optim 26(2):1008–1031
Cao LJ, Keerthi SS, Ong CJ, Zhang JQ, Periyathamby U, Fu XJ, Lee HP (2006) Parallel sequential minimal optimization for the training of support vector machines. IEEE Trans Neural Netw 17(4):1039–1049
Cauchy AL (1847) Méthode générale pour la résolution des systèmes d'équations simultanées. Compte Rendu des S'eances de L'Acad'emie des Sciences XXV S'erie A(25):536–538
Chang CC, Hsu CW, Lin CJ (2000) The analysis of decomposition methods for support vector machines. IEEE Transactions on Neural Networks 11(4):1003–1008
Chang CC, Lin CJ (2011) LIBSVM: a library for support vector machines. ACM Trans Intell Syst Technol 2:27:1–27:27
Chang KW, Hsieh CJ, Lin CJ (2008) Coordinate descent method for large-scale l2-loss linear support vector machines. J Mach Learn Res 9:1369–1398
Chapelle O (2007) Training a support vector machine in the primal. Neural Comput 19(5):1155–1178
Chauhan VK, Dahiya K, Sharma A (2017a) Mini-batch block-coordinate based stochastic average adjusted gradient methods to solve big data problems. In: Zhang ML, Noh YK (eds) Proceedings of the ninth Asian conference on machine learning, PMLR, Proceedings of machine learning research, vol 77, pp 49–64. http://proceedings.mlr.press/v77/chauhan17a.html
Chauhan VK, Dahiya K, Sharma A (2017b) Trust region levenberg-marquardt method for linear svm. In: Ninth international conference on advances in pattern recognition (ICAPR-2017)
Yc C, Su C (2016) Distance-based margin support vector machine for classification. Appl Math Comput 283:141–152
Collobert R, Bengio S, Bengio Y (2002) A parallel mixture of svms for very large scale problems. Neural Comput 14(5):1105–1114
Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297
Cotter A, Shalev-Shwartz S, Srebro N (2013) Learning optimally sparse support vector machines. In: Proceedings of the 30th ICML, pp 266–274
Crammer K, Singer Y (2002) On the learnability and design of output codes for multiclass problems. Mach Learn 47(1995):201–233
De S, Taylor G, Goldstein T (2015) Variance reduction for distributed stochastic gradient descent. arXiv:1512.01708
Defazio A, Bach F, Lacoste-Julien S (2014) Saga: a fast incremental gradient method with support for non-strongly convex composite objectives. In: Proceedings of the 27th international conference on neural information processing systems, MIT Press, Cambridge, NIPS'14, pp 1646–1654
Evgeniou T, Pontil M (2004) Regularized multi–task learning. In: Proceedings of the tenth ACM SIGKDD international conference on knowledge discovery and data mining, ACM, New York, NY, USA, KDD '04, pp 109–117
Evgeniou T, Pontil M, Poggio T (2000) Regularization networks and support vector machines. In: Advances in computational mathematics, MIT Press, Cambridge, pp 1–50
Fan R, Chang K, Hsieh C, Wang X, Lin C (2008) LIBLINEAR: a library for large linear classification. JMLR 9:1871–1874
Farquhar JDR, Meng H, Szedmak S, Hardoon DR, Shawe-Taylor J (2006) Two view learning: Svm-2k, theory and practice. In: Advances in neural information processing systems, MIT Press, Cambridge
Ferris MC, Munson TS (2002) Interior-point methods for massive support vector machines. SIAM J Optim 13(3):783–804
Fletcher R, Reeves CM (1964) Function minimization by conjugate gradients. Comput J 7(2):149–154. https://doi.org/10.1093/comjnl/7.2.149
Franc V, Sonnenburg S (2008) Optimized cutting plane algorithm for support vector machines. In: Proceedings of the 25th international conference on machine learning, ACM, New York, NY, ICML '08, pp 320–327

Friedman J (1996) Another approach to polychotomous classification. Stanford University, Tech Rep, Dept Statistics

Fung G, Mangasarian OL (2001) Proximal support vector machine classifiers. In: Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data mining, ACM, New York, NY, KDD '01, pp 77–86

Gao Y, Sun S (2010) An empirical evaluation of linear and nonlinear kernels for text classification using support vector machines. In: 2010 seventh international conference on fuzzy systems and knowledge discovery, vol 4, pp 1502–1505, https://doi.org/10.1109/FSKD.2010.5569327

Gibson N (2011) Gradient-based methods for optimization. part i

Graf HP, Cosatto E, Bottou L, Durdanovic I, Vapnik V (2005) Parallel support vector machines: the cascade SVM. In: Advances in neural information processing systems pp 521–528

Guermeur Y (2002) Combining discriminant models with new multi-class svms. Pattern Anal Appl 5(2):168–179

Hestenes MR, Stiefel E (1952) Methods of conjugate gradients for solving linear systems. J Res Natl Bur Stand 49(6):409–436

Hsieh CJ, Chang KW, Lin CJ, Keerthi SS, Sundararajan S (2008) A dual coordinate descent method for large-scale linear svm. In: Proceedings of the 25th international conference on machine learning, ACM, New York, NY, ICML '08, pp 408–415

Hsieh CJ, Si S, Dhillon I (2014) A divide-and-conquer solver for Kernel support vector machines. JMLR W&Cp 32(1):566–574

Hsu CW, Lin CJ (2002) A comparison of methods for multiclass support vector machines. IEEE Trans Neural Netw 13(2):415–425. https://doi.org/10.1109/72.991427

Jayadeva, Khemchandani R, Chandra S (2004) Fast and robust learning through fuzzy linear proximal support vector machines. Neurocomput 61(C):401–411

Jayadeva KR, Chandra S (2007) Twin support vector machines for pattern classification. IEEE Trans Pattern Anal Mach Intell 29(5):905–910

Khemchandani R, Chandra S (2017) Twin support vector machines–models, extensions and applications, studies in computational intelligence, vol 659. Springer, Berlin

Ji Y, Sun S (2011) Multitask multiclass support vector machines. In: 2011 IEEE 11th international conference on data mining workshops, pp 512–518

Ji Y, Sun S (2013) Multitask multiclass support vector machines: model and experiments. Pattern Recogn 46(3):914–924

Joachims T (1999) Transductive inference for text classification using support vector machines. In: Proceedings of the sixteenth international conference on machine learning, Morgan Kaufmann Publishers Inc., San Francisco, CA, ICML '99, pp 200–209

Joachims T (2006) Training linear SVMs in linear time. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, NY, USA, KDD '06, pp 217–226

Joachims T, Dortmund U, Joachimscsuni-dortmundde T (1999) Making large-scale SVM learning practical. In: Advances in Kernel methods–support vector learning, pp 41–56

Joachims T, Finley T, Yu CNJ (2009) Cutting-plane training of structural svms. Mach Learn 77(1):27–59

Johnson R, Zhang T (2013) Accelerating stochastic gradient descent using predictive variance reduction. In: Burges CJC, Bottou L, Welling M, Ghahramani Z, Weinberger KQ (eds) Advances in neural information processing systems 26, Curran Associates, Inc., pp 315–323

Kao W, Chung K, Sun C, Lin C (2004) Decomposition methods for linear support vector machines. Neural Comput 16:1689–1704

Keerthi SS, DeCoste D (2005) A modified finite newton method for fast solution of large scale linear svms. JMLR 6:341–361

Keerthi SS, Decoste D (2006) Building support vector machines with reduced classifier complexity. JMLR 7:1493–1515

Keerthi SS, Sundararajan S, Chang KW, Hsieh CJ, Lin CJ (2008) A sequential dual method for large scale multi-class linear svms. In: Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining, ACM, New York, KDD '08, pp 408–416

Knerr S, Personnaz L, Dreyfus G (1990) Neurocomputing: algorithms, architectures and applications. Springer, Berlin, pp 41–50

Konečný J, Richtárik P (2013) Semi-stochastic gradient descent methods, 1:19. arXiv:1312.1666

Kressel UHG (1999) Advances in Kernel methods. MIT Press, Cambridge, MA, pp 255–268

Ladicky L, Torr P (2011) Locally linear support vector machines. ICML, Stockholm, pp 985–992

Lazar A (2010) A comparison of linear support vector machine algorithms on large non-sparse datasets. In: Proceedings–9th ICMLA

Lee CP, Lin CJ (2013) A study on l2-loss (squared hinge-loss) multi-class svm. Neural Comput 25:1302–1323
Lee CP, Roth D (2015) Distributed box-constrained quadratic optimization for dual linear SVM. In: Proceedings of the 32nd ICML, 37
Lee YJ, Mangasarian O (2001) Ssvm: a smooth support vector machine for classification. Comput Optim Appl 20(1):5–22
Levenberg K (1944) A method for the solution of certain problems in least squares. Quart Appl Math 2:164–168
Li L, Li Y, Su H, Chu J (2006) Intelligent computing: international conference on intelligent computing, ICIC 2006, Kunming, China, August 16–19, 2006. Proceedings, Part I, Springer Berlin Heidelberg, Berlin, Heidelberg, chap Least Squares Support Vector Machines Based on Support Vector Degrees, pp 1275–1281
Li M, Andersen DG, Park JW, Smola AJ, Ahmed A, Josifovski V, Long J, Shekita EJ, Su BY (2014a) Scaling distributed machine learning with the parameter server. In: Proceedings of the 11th USENIX conference on operating systems design and implementation, USENIX Association, Berkeley, CA, OSDI'14, pp 583–598
Li M, Zhang T, Chen Y, Smola AJ (2014b) Efficient mini-batch training for stochastic optimization. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining, ACM, New York, NY, KDD '14, pp 661–670
Lin CF, Wang SD (2002) Fuzzy support vector machines. IEEE Trans Neural Netw 13(2):464–471
Lin CJ (2001) On the convergence of the decomposition method for support vector machines. IEEE Trans Neural Netw 12(6):1288–1298
Lin CJ, Weng RC, Keerthi SS (2008) Trust region newton method for logistic regression. JMLR 9:627–650
Lin CY, Tsai CH, Lee CP, Lin CJ (2014) Large-scale logistic regression and linear support vector machines using spark. In: 2014 IEEE international conference on Big Data (Big Data) pp 519–528
Lin HT, Lin CJ, Weng RC (2007) A note on platt's probabilistic outputs for support vector machines. Mach Learn 68(3):267–276
Liu D, Shi Y, Tian Y, Huang X (2016) Ramp loss least squares support vector machine. J Comput Sci 14:61–68
Lu Z, Xiao L (2015) On the complexity analysis of randomized block-coordinate descent methods. Math Program 152(1):615–642
Luss R, d'Aspremont A (2009) Support vector machine classification with indefinite kernels. Math Program Comput 1(2):97–118
Mangasarian OL (2001) A finite newton method for classification problems. Tech. rep., Data Mining Institute, Computer Sciences Department, University of Wisconsin
Mangasarian OL (2006) Exact 1-norm support vector machines via unconstrained convex differentiable minimization. J Mach Learn Res 7:1517–1530
Mangasarian OL, Musicant DR (2001) Lagrangian support vector machines. JMLR 1:161–177
Mangasarian OL, Wild EW (2006) Multisurface proximal support vector machine classification via generalized eigenvalues. IEEE Trans Pattern Anal Mach Intell 28(1):69–74
Manno A, Palagi L, Sagratella S (2015) A class of parallel decomposition algorithms for SVMs training, pp 1–24
Marquardt D (1963) An algorithm for least-squares estimation of nonlinear parameters. SIAM J Appl Math 11:431–441
Moré JJ (1978) Numerical analysis. In: Proceedings of the biennial conference held at Dundee, June 28–July 1, 1977, Springer, Berlin, pp 105–116
Needell D, Srebro N, Ward R (2014) Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm. In: Proceedings of the 27th international conference on neural information processing systems, MIT Press, Cambridge, MA, NIPS'14, pp 1017–1025
Nesterov Y (2012) Efficiency of coordinate descent methods on huge-scale optimization problems. SIAM J Optim 22(2):341–362
Nie F, Huang Y, Wang X, Huang H (2014) New primal SVM solver with linear computational cost for big data classifications. In: Proceedings of the 31st international conference on international conference on machine learning, Vol 32, JMLR.org, ICML'14, pp II–505–II–513
Nie F, Wang X, Huang H (2017) Multiclass capped Lp-norm SVM for robust classifications. The 31st AAAI Conference on Artificial Intelligence (AAAI), San Francisco, USA
Nocedal WS (1999) Numerical optimization. Springer, New York
Parikh N, Boyd S (2014) Proximal algorithms. Found Trends Optim 1(3):127–239
Parlett B (1998) The symmetric eigenvalue problem. Society for Industrial and Applied Mathematics, Philadelphia, PA
Peng X (2010) A nu-twin support vector machine (nu-tsvm) classifier and its geometric algorithms. Inf Sci 180(20):3863–3875

Platt J (1998) Sequential minimal optimization: a fast algorithm for training support vector machines. Microsoft Res pp MSR–TR–98–14
Platt JC, Cristianini N, Shawe-Taylor J (2000) Large margin dags for multiclass classification. Adv Neural Inf Process Syst 12:547–553
Prez-Cruz F, Weston J, Herrmann D, Schölkopf B (2003) Extension of the nu-SVM range for classification. Nato Sci Series Sub Series iii Comput Syst Sci 190:179–196
Ranganathan A (2004) The levenberq-marquardt algorithm
Recht B, Re C, Wright S, Niu F (2011) Hogwild: a lock-free approach to parallelizing stochastic gradient descent. In: Shawe-Taylor J, Zemel RS, Bartlett PL, Pereira F, Weinberger KQ (eds) Advances in neural information processing systems 24, Curran Associates, Inc., pp 693–701
Richtárik P, Takáč M (2014) Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. Math Program 144(1–2):1–38
Richtárik P, Takáč M (2016) Parallel coordinate descent methods for big data optimization. Math Program 156(1):433–484
Schmidt M, Le Roux N, Bach F (2016) Minimizing finite sums with the stochastic average gradient. Math Program pp 1–30
Schölkopf B, Smola AJ, Williamson RC, Bartlett PL (2000) New support vector algorithms. Neural Comput 12(5):1207–1245
Shalev-Shwartz S, Singer Y (2006) Online learning meets optimization in the dual, pp 423–437
Shalev-Shwartz S, Zhang T (2013a) Accelerated mini-batch stochastic dual coordinate ascent. In: NIPS' 13 Proceedings of the 26th International Conference on Neural Information Processing Systems, Vol 1. pp 378–385
Shalev-Shwartz S, Zhang T (2013b) Stochastic dual coordinate ascent methods for regularized loss. JMLR 14(1):567–599
Shalev-Shwartz S, Singer Y, Srebro N (2007) Pegasos: primal estimated sub-gradient solver for SVM. In: Proceedings of the 24th international conference on machine learning, ACM, New York, NY, ICML '07, pp 807–814
Shawe-Taylor J, Sun S (2011) A review of optimization methodologies in support vector machines. Neurocomputing 74(17):3609–3618
Shi Z, Liu R (2016) A better convergence analysis of the block coordinate descent method for large scale machine learning. arXiv:1608.04826
Stefano M, Mikhail B (2011) Laplacian support vector machines trained in the primal. JMLR 12:1149–1184
Steinwart I (2004) Sparseness of support vector machinessome asymptotically sharp bounds. Adv Neural Inf Process Syst 16:1069–1076
Stempfel G, Ralaivola L (2009) Learning SVMs from sloppily labeled data. In: Proceedings of the 19th international conference on artificial neural networks: Part I, Springer, Berlin, ICANN '09, pp 884–893
Sun S, Xie X (2016) Semisupervised support vector machines with tangent space intrinsic manifold regularization. IEEE Trans Neural Netw Learn Syst 27(9):1827–1839
Suykens J, Vandewalle J (1999) Least squares support vector machine classifiers. Neural Process Lett 9(3):293–300
Tak M, Bijral A, Richtrik P, Srebro N (2013) Mini-batch primal and dual methods for SVMs. In: 30th international conference on machine learning, Springer, Berlin, pp 537–552
Teo CH, Smola A, Vishwanathan SV, Le QV (2007) A scalable modular convex solver for regularized risk minimization. In: Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining, ACM, New York, NY, KDD '07, pp 727–736
Teo CH, Vishwanthan S, Smola AJ, Le QV (2010) Bundle methods for regularized risk minimization. JMLR 11:311–365
Tsai CH, Lin CJCY (2014) Incremental and decremental training for linear classification. Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '14, pp 343–352
Tseng P (2001) Convergence of a block coordinate descent method for nondifferentiable minimization. J Optim Theory Appl 109(3):475–494
Tsochantaridis I, Hofmann T, Joachims T, Altun Y (2004) Support vector machine learning for interdependent and structured output spaces. In: Proceedings of the twenty-first international conference on machine learning, ACM, New York, NY, ICML '04, p 104
Vapnik VN (1998) Statistical learning theory, 1st edn. Wiley, New York
Wang Z, Crammer K, Vucetic S (2012) Breaking the curse of kernelization : budgeted stochastic gradient descent for large-scale SVM training. JMLR 13(1):3103–3131

Wen T, Edelman A, Gorsich D (2003) Contemporary mathematics. American Mathematical Society, Boston, MA, chap A Fast Projected Conjugate Gradient Algorithm for Training Support Vector Machines, pp 245–263

Weston J, Watkins C (1999) Support vector machines for multi-class pattern recognition. In: ESANN

Willoughby RA (1979) Solutions of ill-posed problems (a. n. tikhonov and v. y. arsenin). SIAM Rev 21(2):266–267

Woodsend K, Gondzio J (2009) Hybrid MPI/OpenMP parallel linear support vector machine training. JMLR 10:1937–1953

Wright SJ (2015) Coordinate descent algorithms. Math Program 151(1):3–34

Xiao L, Zhang T (2014) A proximal stochastic gradient method with progressive variance reduction. SIAM J Optim 24(4):2057–2075

Xie X, Sun S (2015) Multi-view twin support vector machines. Intell Data Anal 19(4):701–712

Xu C, Tao D, Xu C (2013) A survey on multi-view learning, pp 1–59. arXiv:1304.5634v1

Xu Y, Yin W (2013) A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. SIAM J Imaging Sci 6(3):1758–1789

Xu Y, Yin W (2015) Block stochastic gradient iteration for convex and nonconvex optimization. SIAM J Optim 25(3):1686–1716

Yang Y, Chen J, Zhu J (2016) Distributing the stochastic gradient sampler for large-scale lda. In: Proceedings of the 22Nd ACM SIGKDD international conference on knowledge discovery and data mining, ACM, New York, NY, KDD '16, pp 1975–1984

Ye J, Xiong T (2007) SVM versus least squares SVM. In: JMLR W&P, pp 644–651

Yuan GX, Chang KW, Hsieh CJ, Lin CJ (2010) A comparison of optimization methods and software for large-scale L1-regularized linear classification. JMLR 11:3183–3234

Yuan GX, Ho CH, Lin CJ (2012) Recent advances of large-scale linear classification. Proc IEEE 100(9):2584–2603

Zanghirati G, Zanni L (2003) A parallel solver for large quadratic programs in training support vector machines. Parallel Comput 29(4):535–551

Zanni L, Serafini T, Zanghirati G (2006) Parallel software for training large scale support vector machines on multiprocessor systems. JMLR 7:1467–1492

Zhang A, Gu Q (2016) Accelerated stochastic block coordinate descent with optimal sampling. In: Proceedings of the 22Nd ACM SIGKDD international conference on knowledge discovery and data mining, ACM, New York, NY, KDD '16, pp 2035–2044

Zhang T (2004) Solving large scale linear prediction problems using stochastic gradient descent algorithms. In: Proceedings of the twenty-first international conference on machine learning, ACM, New York, NY, ICML '04

Zhu J, Rosset S, Hastie T, Tibshirani R (2003) 1-norm support vector machines. In: Neural information processing systems, MIT Press, Cambridge, p 16