Survey on Optimizations in SVM

Abstract

Support Vector Machine is an efficient classification tool. And training SVM for linear separable problem is equivalent to quadratic programming (QP) problem. However training SVM is complex in case of large scale training data sets. Three advanced optimization algorithms applied to training SVM are introduced in this paper, one is Least Square SVM, solving the optimization problem with least square methods; one is Linear SVM with Cutting Plane Method; and the other one is Stochastic Methods in SVM. Three methods has their own advantages and disadvantages, they're all widely used in specific areas. Comparison will be declared at the end.

Index Terms

support vector machine, optimization, least square method, cutting plane method, stochastic method

I. Introduction

Support vector machine (SVM) was developed by Cortes and Vapnik in 1995, which is a powerful tool for classification. It's widely used in areas like script recognition, image classification, word-sense disambiguation, with large number of training data, including examples and features, and it's some kind of supervised learning machine. The general form of SVM can be written as:

$$\min_{f} \quad \Omega(f) + C\sum_{i=1}^{l} l(f(\boldsymbol{x}_i), y_i), \tag{1}$$

The training of standard form of SVM for linear separable problem it's a quadratic programming problem[1].

As it's hard to train over large scale of training data, a lot of optimization methods were developed to accelerate the procedure, like Interior Point methods in SVM by Boyd and Vandenberghe in 2004, and decomposition methods such as SMO, SVM^{light}, which is widely used, and gradient descent methods applied to SVM. In this paper, three representative methods applied to SVM will be introduced, after the introduction of standard formulation of SVM.

II. Standard SVM

The standard SVM was referred in [1]. As a binary classifier, the complexity depends on the scale of training set, but not the dimentionality of fature space. Its sparseness also makes it working well over small mount of training data. And this section will take a short review.

Given function $\phi : \Re^n \to \Re^N$ mapping n-dimentional input space into N-dimentional feature space, a hyperplane: $\boldsymbol{w}^T\boldsymbol{x} + b = 0$ could be found to meet

$$\begin{cases} \boldsymbol{w}^T\phi(\boldsymbol{x}_k) + b \geq +1, & y_k = +1, \\ \boldsymbol{w}^T\phi(\boldsymbol{x}_k) + b \leq -1, & y_k = -1. \end{cases} \tag{2}$$

which means points in input space are separated into 2 groups by label y_k.

Given training set $S = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), ..., (\boldsymbol{x}_l, y_l)\},\ y_i \in \{-1, +1\}$, when it comes to getting the optimal hyperplane, or maximizing the margin between two group of support vectors, with soft-margin, which is the solution of the optimal problem:

$$\begin{aligned} \underset{\boldsymbol{w},b,\xi_i}{min} \quad & \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_i^l \xi_i \\ s.t. \quad & y_i\left(\boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b\right) \geq 1 - \xi_i, \\ & \forall i \in \{1, ..., l\}:\ \xi_i \geq 0. \end{aligned} \tag{3}$$

its dual problem is:

$$\begin{aligned} \underset{\alpha}{max} \quad & \sum_{i=1}^l \alpha_i - \frac{1}{2}\sum_{i=1}^l\sum_{j=1}^l \alpha_i\alpha_j y_i y_j \phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j), \\ s.t. \quad & \sum_{i=1}^l \alpha_i y_i = 0, \\ & \forall i \in \{1, ..., l\}:\ 0 \leq \alpha_i \leq C \end{aligned} \tag{4}$$

where C is constant, and it's hard-margin hyperplane when C is ∞ , $\alpha \geq 0$ are Lagrange multipliers.

After getting the solution α^* to problem (4), $\boldsymbol{w}$ can be delivered by $\boldsymbol{w} = \sum_{i=1}^l \alpha_i y_i \phi(x_i)$.

The formulation of SVM classifier is:

$$y(x) = sign\left(\sum_{i=1}^l \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b\right), \tag{5}$$

by replacing $\boldsymbol{w}^T\phi(\boldsymbol{x})$ with $\sum\limits_i \alpha_i y_i \phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x})$, in which $K(\boldsymbol{u}, \boldsymbol{v}) = \phi(\boldsymbol{u})^T\phi(\boldsymbol{v})$ is kernel function which transform lower dimentional input space into higher dimentional feature space in case that the problem is non-linear separable. $K(\boldsymbol{u}, \boldsymbol{v})$ could be linear kernel when $K(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^T\boldsymbol{v}$; or polynomial kernel when $K(\boldsymbol{u}, \boldsymbol{v}) = \left(\boldsymbol{u}^T\boldsymbol{v} + c\right)^d$; or Gaussian kernel, same with RBF, when $K(\boldsymbol{u}, \boldsymbol{v}) = e^{\frac{-||\boldsymbol{u}-\boldsymbol{v}||_2^2}{\sigma^2}}$; or any other kernel function[2].

It's known as kernel method with kernel function mapping low-dimentional space to high-dimentional space.

III. Least Square SVM

Least Square SVM was referred in [2], by Suykens and Vandewalle in 1999. This section will simply introduce this.

The standard formulation of Least Square Problem is:

$$\underset{\boldsymbol{x}}{min} \quad ||A\boldsymbol{x} + y||^2 \tag{6}$$

In Least Square SVM, the formulation of standard SVM is transformed into this form:

$$\begin{aligned} \underset{w,b,\xi_i}{min} \quad & J(\boldsymbol{w}, b, \boldsymbol{\xi}) = \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_i^l \xi_i^2 \\ s.t. \quad & y_i\left(\boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b\right) \geq 1 - \xi_i, \\ & \forall i \in \{1, ..., l\}:\ \xi_i \geq 0. \end{aligned} \tag{7}$$

by replacing ξ_i with ξ_i^2, while its Lagrangian is

$$L(\boldsymbol{w}, b, \xi, \alpha) = J(\boldsymbol{w}, b, \boldsymbol{\xi}) - \sum_{k=1}^{l} \alpha_k \left\{ y_k \left[\boldsymbol{w}^T \phi(\boldsymbol{x}_k) + b \right] - 1 + \xi_k \right\} \tag{8}$$

as C is constant, $\alpha \geq 0$ are Lagrange multipliers.

According to the optimal condition, get the following functions:

$$\begin{aligned}
\frac{\partial L}{\partial \boldsymbol{w}} &= 0 \ \rightarrow \ \boldsymbol{w} = \sum_{i=1}^{l} \alpha_i y_i \phi(x_i), \\
\frac{\partial L}{\partial b} &= 0 \ \rightarrow \ \sum_{i=1}^{l} \alpha_i y_i = 0, \\
\frac{\partial L}{\partial \xi_i} &= 0 \ \rightarrow \ \alpha_i = C\xi_i, \quad i = 1, ..., l, \\
\frac{\partial L}{\partial \alpha_i} &= 0 \ \rightarrow \ y_i \left[\boldsymbol{w}^T \phi(x_i) + b \right] - 1 + \xi_i = 0, \quad i = 1, ..., l.
\end{aligned} \tag{9}$$

which can be written as linear equations over $\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}$.

It's simple and easy to understand, while the complexity of training standard SVM is still a problem in case of big data.

IV. Linear SVM with Cutting Plane Method

Linear SVM is one of the SVMs with linear kernel function, which means $K(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^T \boldsymbol{v}$. An advanced optimization algorithm was introduced by Joachims in 2006, referring to [3]. And a simple review would be taken in this section.

And for simplicity, $\phi(\boldsymbol{x})$ is replaced with $\boldsymbol{x}$ in the following discussion.

Given optimization problem:

$$\begin{aligned}
&\underset{\boldsymbol{w}, \xi_i \geq 0}{min} \quad \frac{1}{2} \boldsymbol{w}^T \boldsymbol{w} + \frac{C}{l} \sum_{i=1}^{l} \xi_i \\
&s.t. \quad \forall i \in \{1, ..., l\} : \ y_i(\boldsymbol{w}^T \boldsymbol{x}_i) \geq 1 - \xi_i.
\end{aligned} \tag{10}$$

and

$$\begin{aligned}
&\underset{\boldsymbol{w}, \xi \geq 0}{min} \quad \frac{1}{2} \boldsymbol{w}^T \boldsymbol{w} + C\xi \\
&s.t. \quad \forall \boldsymbol{c} \in \{0, 1\}^l : \ \frac{1}{l} \boldsymbol{w}^T \sum_{i=1}^{l} c_i y_i \boldsymbol{x}_i \\
&\geq \ \frac{1}{l} \sum_{i=1}^{l} c_i - \xi.
\end{aligned} \tag{11}$$

The optimal solution $\boldsymbol{w}^*$ in (10) and (11) are equivalent according to the theorem referred in [3] (THEOREM 1 on page 218) when ξ^* in (11) equals to $\frac{1}{l} \sum_{i=1}^{l} \xi_i^*$ in (10). So it's feasible to solve problem (11), as its Wolfe Dual has sparseness properties[3]. Problem (10) is the standard form of soft-margin SVM, and problem (11) is the problem to solve next.

Algorithm is shown below. In Algorithm 1, W is set of constraints in (11). W starts with empty set, and in each iteration, it solve an optimal solution in W while W is merged with $\boldsymbol{c}$, a accuracy of current optimal $\boldsymbol{w}$ and ξ. The procedure stops when it satisfies:

$$\frac{1}{l}\sum_{i=1}^{l} c_i - \frac{1}{l}\sum_{i=1}^{l} c_i y_i \left(\boldsymbol{w}^T \boldsymbol{x}_i\right) \leq \xi + \epsilon \tag{12}$$

which means the solution is not violated with ϵ-accuracy.

As said in Joachims' work, time consuming is $O(dlC/\epsilon)$ for size of $|W|$ [4], in which d is the avarage number of non-zero elements in each feature vector, representing the sparsity of training set. Meanwhile, traditional complexity of the problem is $O(l^3)$. A widely used application of this method was SVM^{perf} by Joachims in 2005, and it's faster than SVM^{light}, a application developed also by Joachims in 1999[5] which introduced a SVM with decomposition techniques.

```
Algorithm 1 Cutting Plane Method in SVM
Input: $S = ((\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_l, y_l))$: training set;
    $C$: soft-margin parameter;
    $\epsilon$: accuracy;
Output: optimal $\boldsymbol{w}^*$
 1: $W := \emptyset$;
 2: repeat
 3:   $(\boldsymbol{w}, \xi) := \min_{\boldsymbol{w},\xi\geq 0} \ \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\xi \quad s.t. \quad \forall \boldsymbol{c} \in W : \frac{1}{l}\boldsymbol{w}^T \sum_{i=1}^{l} c_i y_i \boldsymbol{x}_i \geq \frac{1}{l}\sum_{i=1}^{l} c_i - \xi$
 4:   for $i = 1$ to $l$ do
 5:     if $y_i((w)^T x_i) < 1$ then
 6:       $c_i = 1$
 7:     else
 8:       $c_i = 0$
 9:     end if
10:   end for
11:   $W := W \cup \{\boldsymbol{c}\}$
12: until $\frac{1}{l}\sum_{i=1}^{l} c_i - \frac{1}{l}\sum_{i=1}^{l} c_i y_i \left(\boldsymbol{w}^T \boldsymbol{x}_i\right) \leq \xi + \epsilon$
13: return $(\boldsymbol{w}, \xi)$
```

V. Stochastic Methods

In this section, stochastic methods will be introduced, such as Stochastic Gradient Descent (SGD), by Bottou and LeCun in 2005 [6], [7]. and Stochastic Coordinate Descent method, Stochastic Mirror Descent method and many other methods.

The attractiveness is complexities of Stochastic Methods do not depend on the scale of training sets, or they're even negatively correlative, like SGD, which is to be introduced.

Fisrt of all, Surrogate Loss Function in soft-margin SVM should be introduced before further discussion. Given function $l_{0/1}$:

$$l_{0/1}(z) = \begin{cases} 0, & z < 0, \\ 1, & other. \end{cases} \tag{13}$$

and the target function in (3) can be written as:

$$\min_{\boldsymbol{w},b} \ \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{l} l_{0/1}(y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1), \tag{14}$$

by replacing ξ, as slack variables, with $l_{0/1}$, making classifier tolarent with small mount of samples that do not meet the constraints. While $l_{0/1}$ is not convex, continuous, or differentiable, hinge loss, exponential loss or logistic loss can be used to replace it[8].

$$\begin{aligned} \text{hinge loss}: \quad & l_{hinge} = max(0, 1 - z); \\ \text{exponential loss}: \quad & l_{exp} = exp(-z); \\ \text{logistic loss}: \quad & l_{log} = log(1 + exp(-z)). \end{aligned} \tag{15}$$

Let $Q(z, \boldsymbol{w}) = l(y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1)$, the update of w at each iteration in gradient descent method can be:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta\frac{1}{l}\sum_{i=1}^{l}\nabla_w Q(z_i, \boldsymbol{w}_t), \tag{16}$$

where η is the learning rate, each update of w in (16) makes it moving the direction of gradient descent. In case of inaccuracy at nearby area of w^*, it's necessary to make η a variable over t, with which it is second order gradient descent method.

As to SGD, the update of w at each iteration is:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t\nabla_{\boldsymbol{w}} Q(z_t, \boldsymbol{w}_t), \tag{17}$$

where the randomly picked z_t replacing the expectation of all elements. And because of that, noise of single element has been brought to the iteration. The convergence conditions of the algorithm[7] is:

$$\begin{aligned} \sum_t \eta_t^2 < \infty \\ \sum_t \eta_t = \infty \end{aligned} \tag{18}$$

and the fastest convergence is achieved when $\eta_t \propto t^{-1}$.

Same with second order gradient descent, the second order stochastic descent is:

$$w_{t+1} = w_t - \eta_t H_t \nabla_w Q(z_t, w_t), \tag{19}$$

where H_t decreases when t increases, same as second order gradient descent.

This method can get the optimal solution with ϵ-accuracy, without traversing all elements in training set. Because of that, the complexity of the algorithm is $\Omega(\frac{1}{\epsilon^2})$[7] iterations. A significant application and variant of SGD is PEGASOS[9].The complexity of PEGASOS is $O(dC/\epsilon)$ according to [9].

In PEGASOS, introduce an intermediate $w_{t+\frac{1}{2}}$ between two steps:

$$\boldsymbol{w}_{t+\frac{1}{2}} = \boldsymbol{w}_t - \eta_t \nabla_t, \tag{20}$$

where

$$\nabla_t = \frac{\boldsymbol{w}_t}{C} - \frac{1}{|A|} \sum_{(\boldsymbol{x},y)\in A_t} y\boldsymbol{x}. \tag{21}$$

in which A is a selected set, $A \in S$. And $\boldsymbol{w}_{t+1}$ can be got by scaling $w_{t+\frac{1}{2}}$:

$$\boldsymbol{w}_{t+1} = min\left\{1, \frac{1/\sqrt{\lambda}}{||\boldsymbol{w}_{t+\frac{1}{2}}||}\right\} \dot{w}_{t+\frac{1}{2}} \tag{22}$$

Algorithm 2 is the main procedure of PEGASOS, where η_t is the learning rate. As to the size of A, the method becomes SGD when $|A| = 1$, and becomes subgradient projection method when $A = S$ [9]. Relation between ϵ and T is:

$$\epsilon \leq \frac{cln(T)}{\delta\lambda T} \tag{23}$$

where $c = (\sqrt{\lambda} + R)^2$, in which R is the max norm of $\boldsymbol{x}$, $\forall(\boldsymbol{x}, y) \in S$; and δ is the probability that one can not get ϵ-accuracy within O(*) steps. It figures out that the more accuracy to achieve, the more time to consume.

```
Algorithm 2 PEGASOS
Input: $S = ((\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_l, y_l))$: training set;
    $\epsilon$: accuracy;
    $T$: loop time;
    $C$: soft-margin parameter
Output: optimal $\boldsymbol{w}^*$
1: get start point $w$, satisfying $||\boldsymbol{w}|| \leq 1/\sqrt{\lambda}$;
2: for t = 1 to T do
3:     Randomly choose $A \in S$
4:     $A^+ := \{(\boldsymbol{x}, y) \in A : y(\boldsymbol{w}\boldsymbol{x}) < 1\}$
5:     $\eta_t = 1/\lambda_t$
6:     $\boldsymbol{w}_{t+\frac{1}{2}} := (1 - \eta_t\lambda)\boldsymbol{w} + \frac{\eta_t}{|A_t|}\sum_{(\boldsymbol{x},y)\in A^+ y\boldsymbol{x}}$
7:     $\boldsymbol{w} := min\left\{1, \frac{1/\sqrt{\lambda}}{||w_{t+\frac{1}{2}}||}\right\} \cdot \boldsymbol{w}_{t+\frac{1}{2}}$
8: end for
9: return $\boldsymbol{w}$
```

VI. Discussion

Three kind of methods applied in SVM has been introduced above, and in this section a comparison between them is delivered below.

The complexities of three methods, as is introduced in the three sections above, are $O(l^3)$, $O(dlC/\epsilon)$, and $O(dC/\epsilon)$, and it's obvious to see that the SGD has the least complexity among the three. As the Least Square SVM is a simple variant of the standard SVM, SVM^{perf} and PEGASOS work better with large scale of training data.

While PEGASOS is highly proposed in this paper, which has the least runtime consuming as it's no reletive with the size of training data, but only the sparsity of samples and accuracy of results. PEGASOS is an advanced variant of SGD, and also a combination of SGD and subgradient projection method. Though it's a problem for SGD to get sparse solutions[10], variants of it has been developed by Langford in 2009 for this, and PEGASOS has no problem with that.

In this paper, we take a review of three optimization methods applied in SVM. While algorithms to optimize SVM are still developing, like on-line learning and kernel matrix, and the applications of SVMs are also spreading.

References

[1] C. Cortes and V. Vapnik, "Support-vector networks," Machine Learning, vol. 20, no. 3, pp. 273–297, 1995.

[2] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," Neural processing letters, vol. 9, no. 3, pp. 293–300, 1999.

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

[4] S. Shalev-Shwartz and N. Srebro, "Svm optimization: Inverse dependence on training set size," in Proceedings of the 25th International Conference on Machine Learning, ICML '08, (New York, NY, USA), pp. 928–935, ACM, 2008.

[5] T. Joachims, "Making large-scale SVM learning practical," in Advances in Kernel Methods - Support Vector Learning (B. Schölkopf, C. Burges, and A. Smola, eds.), ch. 11, pp. 169–184, Cambridge, MA: MIT Press, 1999.

[6] L. Bottou and Y. Le Cun, "On-line learning for very large data sets," Applied stochastic models in business and industry, vol. 21, no. 2, pp. 137–151, 2005.

[7] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in Proceedings of COMPSTAT'2010, pp. 177–186, Springer, 2010.

[8] Z. H. Zhou, Machine Learning. Tsinghua University Press, 2016.

[9] S. Shalev-Shwartz, Y. Singer, and N. Srebro, "Pegasos: Primal estimated sub-gradient solver for svm," in Proceedings of the 24th International Conference on Machine Learning, ICML '07, (New York, NY, USA), pp. 807–814, ACM, 2007.

[10] S. Shalev-Shwartz and A. Tewari, "Stochastic methods for l1-regularized loss minimization," Journal of Machine Learning Research, vol. 12, no. Jun, pp. 1865–1892, 2011.