Report for Optimization Techniques for Semi-Supervised Support Vector Machines

Zhiting Xu

December 15, 2008

1 Problem Description

Support Vector Machines(SVMs) has been showed that it is a powerful classifier in many problems. It can be viewed as a optimization problem and it has a nice property that the objective function is convex, which means its local minimizer is the global minimizer. However, it is usually expensive to get the labeled data, while it is much easier to get unlabeled data. As a result, Semi-Supervised Support Vector Machines(S^3VMs) were developed. One difficulty is that the formulation is a non-convex optimization problem, and thus varieties optimization techniques were proposed for this problem. Each technique has its own advantages and disadvantages, and [2] does a survey of optimization techniques for S^3VMs.

In that paper, for optimization techniques, they only consider the binary classification problem. In the semi-supervised learning for binary classification problem, the training set contains l labeled samples $\{(\vec{x}_i, y_i)\}_{i=1}^{l}$, $y_i = \pm 1$, and u unlabeled examples $\{\vec{x}_i\}_{i=l+1}^{n}$. The linear S^3VMs solve the following optimization problem:

$$\min_{(\vec{w},b),\vec{y}_u} I(\vec{w}, b, \vec{y}_u) = \frac{1}{2}||\vec{w}||^2 + C\sum_{i=1}^{l} V(y_i, o_i) + C^\star \sum_{i=l+1}^{n} V(y_i, o_i) \quad (1)$$

$$\text{s.t.} \quad \frac{1}{u}\sum_{i=l+1}^{n} \max(y_i, 0) = r \quad (2)$$

As it shows, S^3VMs solve over both the hyperplane parameters $(\vec{w}, b)$ and the labels for the unlabeled data $\vec{y}_u = [y_{l+1}, \ldots, y_n]^T$. The constrain is to avoid unbalanced solutions.

A widely choice for V is the Hinge loss $V(y_i, o_i) = \max(0, 1 - y_i o_i)^p$, and $p = 2$ is a widely choice for p.

There are generally two kinds of techniques to solve 1:

- **Combinatorial Optimization:** Fix $\vec{y}_u$ first, and then solve a standard SVM. Try to find the best $\vec{y}_u$. Formally, define

$$\vartheta(\vec{y}_u) = \min_{\vec{w},b} I(\vec{w}, b, \vec{y}_u) \quad (3)$$

  and solve

$$\min_{\vec{y}_u} \vartheta(y_u) \quad (4)$$

- **Continuous Optimization:** For a fixed $(\vec{w}, b)$, $\text{argmin}_y V(y, o) = \text{sign}(o)$, so just use $o_i = \vec{x}_i^T \vec{x}_i + b$ as the label of unlabeled point $\vec{x}_i$. So the objective function becomes:

$$\min_{(\vec{w},b)} \frac{1}{2}||\vec{w}||^2 + C\sum_{i=1}^{l} \max(0, 1 - y_i o_i)^2 + C^\star \sum_{i=l+1}^{n} \max(0, 1 - |o_i|)^2 \quad (5)$$

The last term in 5 is non-convex.

In the rest of the paper, several techniques of these two kinds of approaches were discussed.

2 Combinatorial Optimization

2.1 Branch-and-Bound (BB) for Global Optimization

Branch-and-Bound is a technique of global optimization for 4. The idea was to search over the $\vec{y}_u$ space with pruning.

BB searches on a binary tree, which nodes represent the set of points that are have labels (either the given labels or guessed labels), and thus for each node, it has an objective function value. A node's left child is its labeled points set union with a new point x_i with guessing label 1, while the right child is its set union with x_i with guessing label 0. Thus, the tree begin with the initial labeled points set, and gradually add unlabeled points to explore the whole tree.

One observation is crucial to prune the tree: if the set of labeled points on one node, n_1, is a subset of the set on another node n_2, and the objective function value of n_1 is larger than that of n_2, then there is no need to explore the children of n_1. In [1], they discussed how to choose candidate children to explore.

As BB approach searches for all possible $\vec{y}_u$, and only prunes bad subtrees, so it guarantees the global solutions. On the other hand, it is too expensive to search the tree if the unlabeled points set is large.

2.2 $\mathbf{S^3VM}^{light}$

$\mathrm{S^3VM}^{light}$ refers to implementation in the popular toolkits SVM^{light}. The initial labels of unlabeled points are given by SVM with balanced constraints; and for each iteration, it switches the labels of two points, if the following condition is satisfied:

$$y_i = 1, y_j = -1, V(1, o_i) + V(-1, o_j) > V(-1, o_i) + V(1, o_j) \qquad (6)$$

Thus after switching the labels, the objective function value is guaranteed to decrease. It is obvious that this approach converges to a local minimizer of 4. In an outer loop, $\mathrm{S^3VM}^{light}$ increases $C^\star$ to control the non-convex part of the objective function.

In practice, running time of $\mathrm{S^3VM}^{light}$ is acceptable, but it only converges to a local minimizer.

2.3 Deterministic Annealing S^{3}VM

Apply Deterministic annealing(DA) to S^{3}VM, the discrete label variables $\vec{y}_u$ are relaxed to real-valued variables $\vec{p}_u = (p_{l+1}, \ldots, p_{l+u})$ where p_i indicates the probability of $y_i = 1$. So 4 could be transferred as:

$$\begin{aligned} I'(\vec{w}, b, \vec{p}_u) &= E(I(\vec{w}, b, \vec{y}_u)) \\ &= \frac{1}{2}||\vec{w}||^2 + C\sum_{i=1}^{l} V(y_i, o_i) + C^\star(\sum_{i=l+1}^{n} p_i V(1, o_i) + (1-p_i)V(-1, o_i)) \end{aligned} \tag{7}$$

where E is expectation under the probabilities $\vec{p}_u$.

In addition, an entropy term $-H(\vec{p}_u)$ is added to the objective,

$$\begin{aligned} I''(\vec{w}, b, \vec{p}_u; T) &= I'(\vec{w}, b, \vec{p}_u) - TH(\vec{p}_u) \\ \text{where} H(\vec{p}_u) &= -\sum_i p_i \log p_i + (1-p_i)\log(1-p_i) \end{aligned} \tag{8}$$

The balance constraint used in DA is:

$$\frac{1}{u}\sum_{i=l+1}^{n} p_i = r \tag{9}$$

The degree of non-convexity decreases with the increasing of T. At any T, let $(\vec{w}_T, b_T, \vec{p}_{uT}) = \text{argmin}_{(\vec{w},b),\vec{p}_u} I''(\vec{w}, b, \vec{p}_u; T)$. There are two ways to perform the optimization here:

- *Alternating Minimization (DA):* For each iteration, it fixes $\vec{p}_u$ first, which leads to a standard SVM; and then fixes the $(\vec{w}, b)$, solves 8 with constraints 9, which leads to

  $$p_i = \frac{1}{1 + e^{g_i - v/T}} \tag{10}$$

  where $g_i = C^\star[V(1, o_i), V(-1, o_i)]$ and v, the Lagrange multiplier associated with the balance constraint.

  Repeat this until $\vec{p}_u$ stabilizes in a KL-divergence sense.

- *Gradient Methods (∇DA):* This approach substitutes the 10 as a function of $(\vec{w}, b)$, and thus solve a problem:

  $$\min_{\vec{w},b} \mathcal{S}(\vec{w}, b) = \min_{\vec{p}_u} I''(\vec{w}, b, \vec{p}_u; T) \tag{11}$$

  $\mathcal{S}$ could be minimized through optimization techniques mentioned in our course, though in this paper they used a conjugate gradient descent.

Both DA and ∇DA converge to a local minimizer, but the paper claims that ∇DA faster than DA.

2.4 Convex Relaxation

Consider the dual problem of S^{3}VM:

$$\min_{y_i} \max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K_{ij} \tag{12}$$
$$\text{s.t.} \sum_{i=1}^{n} \alpha_i y_i = 0, \alpha_i \geq 0$$

where $K_{ij} = \vec{x}_i^T \vec{x}_j + D_{ij}$ and D is a diagonal matrix given by $D_{ii} = 1/2C, i = 1, \ldots, l$ and $D_{ii} = 1/2C^\star, i = l+1, \ldots, n$.

Introducing an $n \times n$ matrix Γ, the problem can be written as:

$$\min_{\Gamma} \max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j \Gamma_{ij} K_{ij} \tag{13}$$
$$\text{s.t.} \sum \alpha_i y_i = 0, \alpha_i \geq 0, \Gamma = yy^T \tag{14}$$

where 13 is convex while 14 is not convex.

Relax the 14 through replacing it with:

$$\begin{aligned} &\Gamma \succeq 0 \\ &\Gamma_{ij} = y_i y_j, 1 \leq i, j \leq l \\ &\Gamma_{ii} = 1, l+1 \leq i \leq n \end{aligned} \tag{15}$$

which is a Semi-Definite Programming problem.

As this is a convex problem, the global minimizer could be found. However, SDP is very expensive to perform.

3 Continuous Optimization

Methods discussed in this section do not treat $\vec{y}_u$ as optimization variables; rather, they solve 5 using continuous optimization techniques. There are two issues that are common to these methods.

- **Balancing Constraint** These methods relax the balancing constraint as
$$\frac{1}{u} \sum_{i=l+1}^{n} \vec{w}^T \vec{x}_i + b = 2\tilde{r} - 1 \tag{16}$$
where $\tilde{r} = r$. To get an unconstrained optimization problem on $\vec{w}$, translate all the points so that $\sum_{i=l+1}^{n} \vec{x}_i = 0$ and fix $b = 2\tilde{r} - 1$.

- **Primal Optimization** Methods in this section cannot directly use dual-based SVM software, so following primal methods are used:

– Find $\vec{z}_i$ such that $\vec{z}_i \cdot \vec{z}_j = k(\vec{x}_i, \vec{x}_j)$, and replace x_i in 5 by $\vec{z}_i$ and solve a linear SVM.
– Set $\vec{w} = \sum_{i=1}^{n} \beta_i \phi(\vec{x}_i)$ where ϕ denotes a higher dimensional feature map associated with the nonlinear kernel. Substitute this form in 5, and the problem is with β as the variables.

3.1 Concave Convex Procedure(CCCP)

The idea of CCCP is to decompose a non-convex function f into a convex part f_{vex} and a concave part f_{cave}. The update for x_{t+1} is $\text{argmin}_{\vec{x}} f_{vex}(\vec{x}) + \nabla f_{cave}(\vec{x}_t) \cdot \vec{x}$

For S^{3}VM in form 5, the first two terms are convex and split the last into a convex part and a concave part: $\max(0, 1-|t|) = \max(0, 1-|t|)^2+2|t|-2|t|$. The loss function $\tilde{L}(t)$ could be defined based on this form. So we have Algorithm 1.

Algorithm 1 CCCP for S^{3}VM

Starting point: Use the $\vec{w}$ obtained from the supervised SVM solution.
repeat
$y_i \leftarrow \text{sign}(\vec{w} \cdot \vec{x}_i + b), l+1 \le i \le n$
$(\vec{w}, b) = \text{argmin} \frac{1}{2}||\vec{2}||^2 + C \sum_{i=1}^{l} \max(0, 1-y_i(\vec{w} \cdot \vec{x}_i + b))^2 + C^* \sum_{i=l+1}^{n} \tilde{L} + \nabla(-2|t|) \cdot (\vec{w}, b)$
until convergence of y_i,$l+1 \le i \le n$,l

3.2 ∇S^{3}VM

This method is to minimize objective function 5 directly by gradient descent with the replacing of $\max(0, 1-|t|)^2$ with $\exp(-st^2)$. That is, the optimization problem becomes

$$\min_{\vec{w},b} \frac{1}{2}||\vec{w}||^2 + C \sum_{i=1}^{l} \max(0, 1-y_i(\vec{w} \cdot \vec{x}_i + b))^2 + C^* \sum_{i=l+1}^{n} \exp(-s(\vec{w} \cdot x_i + b)^2) \tag{17}$$

In the outer loop, this method increases C^* gradually.

3.3 Continuation S^{3}VM(cS^{3}VM)

This method's idea is that first, smooth the objection function enough to get a global minimizer on the smoothed function, and use that point as the starting point for the next iteration. Then smooth less to the objective function and optimize it again. Repeat this process until there is no smoothing.

Both ∇S^{3}VM and cS^{3}VM has a complexity of $O(n^3)$, which is relative expensive compared with linear SVM.

3.4 Newton S^{3}VM

As mentioned in last subsection, ∇S^{3}VM and cS^{3}VM are more expensive than SVM, which has a computing complexity of $O(n_{sv}^3 + n^2)$, where n_{sv} is the number of support vectors, which is relatively small. Newton S^3VM uses a new objective function and applies Newton method with Hessian Modification (which is named as Levenberg-Marquardt method in this paper, while text book names it as Modified Cholesky Factorization).

They use method 2 described in the beginning of this section. That is, set $\vec{w} = \sum_{i=1}^{n} \beta_i \phi(\vec{x}_i)$, and let ℓ_L be the loss function of the labeled points, and ℓ_U be the loss function of the unlabeled points. Replace $\vec{w}$ by β as the variables in 5, and get the objective function:

$$\min_{\beta} \frac{1}{2}\beta^T K\beta + C\sum_{i=1}^{l} \ell_L(y_i(K_i^T\beta + b)) + C^* \sum_{i=l+1}^{n} \ell_U(K_i^T\beta + b) \qquad (18)$$

where K is the kernel matrix and K_i is the i^{th} column of K.

4 Experiments and Conclusion

The experiments in this paper show that S^{3}VMs don't always perform better than SVMs; actually, in many cases, S^{3}VMs are worse than SVMs. But the author argues that this because the data points are on manifolds, which is a hard task for S^{3}VMs.

According to the results, ∇S^3VM and cS^{3}VM achieves the lowest objective values; however, lower objective values does not guarantee better performance on classification. The results of unlabeled errors show that the performance depends on the specified data set; for some data set, some algorithms perform better. However, the SVM^{light} seems to be the most robust one.

References

[1] O. Chapelle, V. Sindhwani, and S. S. Keerthi. Branch and bound for semi-supervised support vector machines. In J. Platt T. Hofmann Sch02lkopf, B., editor, *Twentieth Annual Conference on Neural Information Processing Systems (NIPS 2006)*, pages 217–224, Cambridge, MA, USA, 09 2007. MIT Press.

[2] O. Chapelle, V. Sindhwani, and S. S. Keerthi. Optimization techniques for semi-supervised support vector machines. *Journal of Machine Learning Research*, 9:203–233, 02 2008.