CS771 Assignment-1

Abhishek Soni (170034) **Aniket Gupta (170108)** **Bholeshwar Khurana (170214)**

Saurav Prakash (170646) **Sayak Chakrabarti (170648)**

Abstract

We aim to classify data-points efficiently using the Squared SVM Solver. The data consists of 20000 data-points divided into two classes. We use various optimization techniques to optimize the problem given to us to find the best fitting hyper-plane corresponding to it.

1 Introduction

In machine learning, support-vector machines (SVMs) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier.
Support Vector Machines (SVMs) are a way to set up a constrained optimization problem that attempts to find a hyper-plane with maximum margin from distinguishing class of data points.
To maximise the margin the model can be written as[1]:

$$\arg \max_{\mathbf{w}\in\mathbb{R}^d, b\in\mathbb{R}} \{\min_{i\in[n]} \frac{|\langle \mathbf{w}, \mathbf{x}^i\rangle + b|}{\|\mathbf{w}\|_2}\} \tag{1}$$

$$\text{s.t. } y^i(\langle \mathbf{w}, \mathbf{x}^i\rangle + b) \geq 0 \quad \forall\, i \in [n]$$

We solve this constrained problem in the following sections. We consider the following formulation of our problem for n binary labelled data points $(x^i, y^i)_{i\in[n]}$ where $\mathbf{x}^i \in \mathbb{R}^d$ and $y^i \in \{-1, 1\}$ using the squared hinge loss function:

$$\arg \min_{\mathbf{w}\in\mathbb{R}^d} \{\frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{n}([1 - y^i\langle \mathbf{w}, \mathbf{x}^i\rangle]_+)^2\} \tag{P1}$$

The above optimization problem (P1) can be rewritten as:

$$\arg \min_{\mathbf{w}\in\mathbb{R}^d,\, \boldsymbol{\xi}\in\mathbb{R}^n} \{\frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{n}\xi_i^2\} \tag{P2}$$

$$\text{s.t. } y^i\langle \mathbf{w}, \mathbf{x}^i\rangle \geq 1 - \xi_i \quad \forall\, i \in 1, 2 \ldots n$$

[1] From now on we ignore the bias b as this bias can be included into $\mathbf{w}$ by increasing its dimension by one and concatenating 1 to the end of $\mathbf{x}$ while increasing its dimension by one too

Preprint. Under review.

2 Expression for Lagrangian (Part 1)

We introduce the dual variables α_i's for each $i \in [n]$ and write the expression for Lagrangian. The Lagrangian is:

$$\mathcal{L}(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{n}\xi_i^2 + \sum_{i=1}^{n}\alpha_i(1 - \xi_i - y_i\langle \mathbf{w}, \mathbf{x}^i\rangle) \tag{2}$$

where $\mathbf{w} \in \mathbb{R}^d$, $\boldsymbol{\xi} = [\xi_1, \xi_2, \ldots, \xi_n] \in \mathbb{R}^n$ and $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_n] \in \mathbb{R}^n$

This is the Lagrangian of the problem. It takes care of all the boundary conditions and constraints. Violation of any constraint yields

$$\max_{\boldsymbol{\alpha}\in\mathbb{R}^n, \alpha_i \geq 0} \mathcal{L}(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}) = \infty$$

Hence solving

$$\min_{\mathbf{w}\in\mathbb{R}^d,\ \boldsymbol{\xi}\in\mathbb{R}^n} \{\max_{\boldsymbol{\alpha}\in\mathbb{R}^n, \alpha_i \geq 0} \mathcal{L}(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha})\} \tag{3}$$

is sufficient to obtain the best fitting hyper-plane for our classification problem.

3 The Dual Problem (Part 2)

The original optimization problem is called the primal problem, denoted by Equation (3). We convert it to a dual problem by switching the order of min and max.

$$\arg\max_{\boldsymbol{\alpha}\in\mathbb{R}^n, \alpha_i \geq 0}\{\arg\min_{\mathbf{w}\in\mathbb{R}^d,\ \boldsymbol{\xi}\in\mathbb{R}^n} \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{n}\xi_i^2 + \sum_{i=1}^{n}\alpha_i(1 - \xi_i - y_i\langle \mathbf{w}, \mathbf{x}^i\rangle)\} \tag{D2}$$

In our case, the dual problem is easier to solve. Hence we proceed with the optimization problem by solving this dual problem represented in (D2).

This optimization problem, $\mathcal{L}(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha})$, is a convex function w.r.t. $\mathbf{w}$, so its partial derivative w.r.t. $\mathbf{w}$ vanishes at its global minima, i.e.

$$\frac{\delta\mathcal{L}(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha})}{\delta\mathbf{w}} = 0$$

$$\mathbf{w} - \sum_{i=1}^{n}\alpha_i y^i \mathbf{x}^i = 0$$

$$\mathbf{w} = \sum_{i=1}^{n}\alpha_i y^i \mathbf{x}^i \tag{4}$$

It is also convex w.r.t. $\boldsymbol{\xi}$. So derivative of $\mathcal{L}(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha})$ wrt $\boldsymbol{\xi}$ gives the global minima wrt $\boldsymbol{\xi}$. Hence,

$$\frac{\delta\mathcal{L}(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha})}{\delta\xi_i} = 0$$

$$2C\xi_i - \alpha_i = 0$$

$$\xi_i = \frac{\alpha_i}{2C}$$

So, we have

$$\boldsymbol{\xi} = \frac{\boldsymbol{\alpha}}{2C} \tag{5}$$

Now, we have

$$\|\mathbf{w}\|_2^2 = \sum_{i,j}\alpha_i\alpha_j y^i y^j \langle \mathbf{x}^i, \mathbf{x}^j\rangle$$

which on substituting ξ and after solving the min criterion could be written as,

$$\arg\max_{\beta_i\geq 0, \alpha_i \geq 0}\{\sum_{i=1}^{n}\alpha_i - \frac{1}{2}\|\mathbf{w}\|_2^2 - \sum_{i=1}^{n}\frac{\alpha_i^2}{4C}\}$$

Again on substituting the values of **w**, we have:

$$\arg \max_{\boldsymbol{\alpha}\in\mathbb{R}^n,\ \alpha_i\geq 0} \{\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y^i y^j \langle \mathbf{x}^i, \mathbf{x}^j\rangle + \frac{\alpha_i^2}{4C} + \sum_{i=0}^{n}\alpha_i(1-\frac{\alpha_i}{2C} - y^i\langle(\sum_{j=0}^{n}\alpha_j y^j \mathbf{x}^j), \mathbf{x}^i\rangle)\} \quad (6)$$

Now, looking at the i^{th} coordinate of Equation (6), we have

$$\frac{1}{2}\alpha_i^2\|\mathbf{x}^i\|_2^2 + \frac{1}{2}\alpha_i(\sum_{j\neq i}\alpha_j y^i y^j \langle \mathbf{x}^i, \mathbf{x}^j\rangle) + \frac{\alpha_i^2}{4C} + \alpha_i - \frac{\alpha_i^2}{2C} - \alpha_i^2\|\mathbf{x}^i\|_2^2 - \alpha_i(\sum_{j\neq i}\alpha_j y^i y^j \langle \mathbf{x}^i, \mathbf{x}^j\rangle) \quad (7)$$

Substituting

$$P = \sum_{j\neq i}\alpha_j y^i y^j \langle \mathbf{x}^i, \mathbf{x}^j\rangle$$

into Equation (7) we have the i^{th} coordinate gives

$$-\frac{\alpha_i^2}{4C} - \frac{\alpha_i^2\|\mathbf{x}^i\|_2^2}{2} + \alpha_i(1-P_i)$$

This is a convex function and the partial derivative wrt α_i gives us the global maxima. So, finding the value of α we will be able to find **w** and $\boldsymbol{\xi}$, thereby solving (D2).
Differentiating wrt α_i we have

$$-\alpha_i(\frac{1}{2C} + \|x^i\|_2^2) + (1-P_i) = 0$$

$$\alpha_i = \frac{(1-P_i)}{(\frac{1}{2C} + \|x^i\|_2^2)} \quad (8)$$

With this we can find the value of α and hence the values of **w** (from Equation (4)) and $\boldsymbol{\xi}$ (from Equation (5)).
Using these solutions obtained, we head towards the next section on implementation to find the actual values of our hyper-parameters best fitting for our dataset.

4 Implementation (Part 3)

To solve our optimization problem, we implemented three methods:

- Mini-batch Stochastic Gradient descent on $P1$
- Coordinate Maximization on $D2$
- Coordinate descent on $P1$

Following is the description of each of these methods:

4.1 Mini-batch Stochastic Gradient descent

Mini-batch SGD is a modified version of Gradient descent, where instead of finding the gradient of $f(w)$ ($\nabla(f(w))$ on all the training set, we calculate the gradient using a subset of the training set. We call the size of this subset, the mini-batch size B. Note that $f(w)$ is the optimization problem here. It is called Stochastic because the subset is chosen randomly.

We implemented mini-batch SGD on the problem $P1$ (in python). Call the equation in $P1$ to be $f(w)$. We have to minimize $f(w)$. Since $f(w)$ has squared hinge loss function, so $f(w)$ is differentiable. We perform the following algorithm:

1. Initialize $\mathbf{w}^0 \in \mathbb{R}^d$ randomly.
2. At each step t, choose B random data-points without replacement, say $(x^{i_t^1}, y^{i_t^1})$, ..., $(x^{i_t^B}, y^{i_t^B})$. We obtain the gradient of $f(w)$ ($\boldsymbol{\nabla(f(w))}$ as:

$$\boldsymbol{\nabla(f(w))} \approx \mathbf{w} - 2C(\frac{n}{B})\sum_{b=1}^{B}([1-y^{i_t^b}\langle \mathbf{w}, \mathbf{x}^{i_t^b}\rangle]_+)y^{i_t^b}\mathbf{x}^{i_t^b}$$

3. Choose a step-length η_t
4. Update $\mathbf{w}$ as:
$$\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t - \eta_t \boldsymbol{\nabla}(\boldsymbol{f}(\boldsymbol{w})$$
5. Repeat Steps (2) to (4) till convergence.

We chose the mini-batch size (B) to be 10 and the step length function (η_t) to be $\eta/\sqrt{t+1}$. We used horizon = 1000000 (number of steps till convergence). We see in the next section how we arrived at these (hyper)parameters.

4.2 Coordinate Maximization

We apply coordinate maximization on the dual problem $D2$. In order to maximize $D2$, we concentrate at just the terms containing α_i and try to maximize these terms for all $i \in [n]$. We follow the following algorithm:

1. Initialize $\boldsymbol{\alpha} \in \mathbb{R}^n$ with all ones.
2. Initialize $\mathbf{w}$ according to the equation (4), i.e.
$$\mathbf{w}^0 = \sum_{i=0}^{n} \alpha_i y^i \mathbf{x}^i$$
3. At each step t, choose $\mathbf{x} \in \mathbb{R}^d$ as some random data-point from the training set.
4. At each step t, update α according to the equation (8) as:
$$\alpha_i^{t+1} \leftarrow \frac{(1 + \alpha_i^t \|\mathbf{x}^j\|_2^2 - y^j \langle \mathbf{w}, \mathbf{x}^j \rangle)}{(\|\mathbf{x}^j\|_2^2 + 1/2C)}$$
5. Since, we have the constraint $\alpha_i \in [0, C]$, so if $\alpha^{t+1} > C$, then $\alpha^{t+1} = C$ OR if $\alpha^{t+1} < 0$, then $\alpha^{t+1} = 0$
6. Update $\mathbf{w}$ as:
$$\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t + (\alpha^{t+1} - \alpha^t) y^j \mathbf{x}^j$$
7. Repeat Steps (3) to (6) till convergence.

In the step-3 above, we chose $\mathbf{x}$ (the coordinate along which to maximize) randomly, i.e. we chose the SCD approach. We chose the horizon = 1000000 (number of steps till convergence). We see in the next section for our choice of these (hyper)parameters.

4.3 Coordinate descent

We apply stochastic coordinate descent on the Primal problem $P1$. In this method the value of $f(w)$ is minimized by movement of that coordinate of w along partial gradient of that coordinate of w. This coordinate is selected randomly in this method, hence the name Stochastic coordinate descent. The following algorithm is followed for this method:

1. Initialize $\mathbf{w}$ as $\mathbf{w}^0 \in \mathbb{R}^d$ with zeros.
2. Calculate the gradient of **f(w)** , i.e
$$\boldsymbol{\nabla}(\boldsymbol{f}(\boldsymbol{w})) = \mathbf{w} - 2C \sum_{i=1}^{n} ([1 - y^i \langle \mathbf{w}, \mathbf{x}^i \rangle]_+) y^i \mathbf{x}^i$$
3. Choose a step-length η_t
4. Choose a random coordinate w_i, and update that w_i as follows
$$w_i^{new} \leftarrow w_i - \eta \boldsymbol{\nabla}(\boldsymbol{f}(\boldsymbol{w}))_i$$
5. Repeat steps (2) to (4) till convergence.

We chose the step length function (η_t) to be $\eta/\sqrt{t}$ (where, t is the iteration number). We used horizon = 10000 (number of steps till convergence).
We see in the next section how we arrived at these hyper-parameters.

5 Hyper-parameter tuning (Part 4)

For each of the methods implemented by us, we tuned the hyper-parameters as follows:

5.1 Mini-batch Stochastic Gradient descent

We tried various batch sizes and got the best results for batch size 10. We chose two different step-length functions: $\eta_t = \eta/(t+1)$ and $\eta_t = \eta/\sqrt{t+1}$ and chose η from the set $\{5 * 10^{-4}, 2.5 * 10^{-4}, 1.25 * 10^{-4}, 6.25 * 10^{-5}, \dots\}$, i.e. we halved η after each iteration for sufficient number of iterations. We tried all possible pairs on 4-fold cross-validation for 100k iterations and got the following results:

η	**Step-length**	**Primal Objective function value**			
		Test-1	Test-2	Test-3	Test-4
5*10^{-4}	$\eta_t = \eta/(t+1)$	1614.24	1576.88	1611.36	1602.42
	$\eta_t = \eta/\sqrt{t+1}$	1475.85	1444.70	1514.35	1472.66
2.5*10^{-4}	$\eta_t = \eta/(t+1)$	1695.98	1644.82	1730.81	1731.16
	$\eta_t = \eta/\sqrt{t+1}$	1482.07	1455.14	1522.75	1485.66
1.25*10^{-4}	$\eta_t = \eta/(t+1)$	1817.74	1794.76	1874.10	1827.37
	$\eta_t = \eta/\sqrt{t+1}$	1491.681	1463.88	1527.11	1496.26
6.25*10^{-5}	$\eta_t = \eta/(t+1)$	1912.20	1881.01	1949.53	1947.59
	$\eta_t = \eta/\sqrt{t+1}$	1519.82	1490.41	1549.72	1527.56

We got the best results for the step-length function $\eta_t = \eta/\sqrt{t+1}$ and $\eta = 0.0005$.

5.2 Coordinate Maximization

We tried the following methods for Coordinate selection:

- **CCD:** Choose coordinate cyclically
- **SCD:** Choose coordinate stochastically(randomly)
- **Block CD:** Choose a small set of coordinates randomly at each step t

We tried each of these methods for 60k iterations on held-out validation and whole data-set, getting the following results:

Method	**Primal Objective function value**	
	Training on 16000 data-points (rest 4000 is validation set)	Training on whole data-set
CCD	54646.27	68367.49
SCD	5241.18	5225.92
Block CD	5233.43	5225.03

We got the best results for SCD and Mini Batch SGD but since SCD is faster than Block CD, so we used SCD in our model.

5.3 Coordinate Descent

We tried to maximize the value of Primal Objective function for the step length function $\eta/\sqrt{t}$, where t is the iteration number of solver. And the value of η is decided by the running the solver for 10000 iterations. Value of Primal Objective function for any test set indicates the value of the function on

the whole the training dataset, while training only on one-fourth part of data-set (Using 4-fold cross validation).

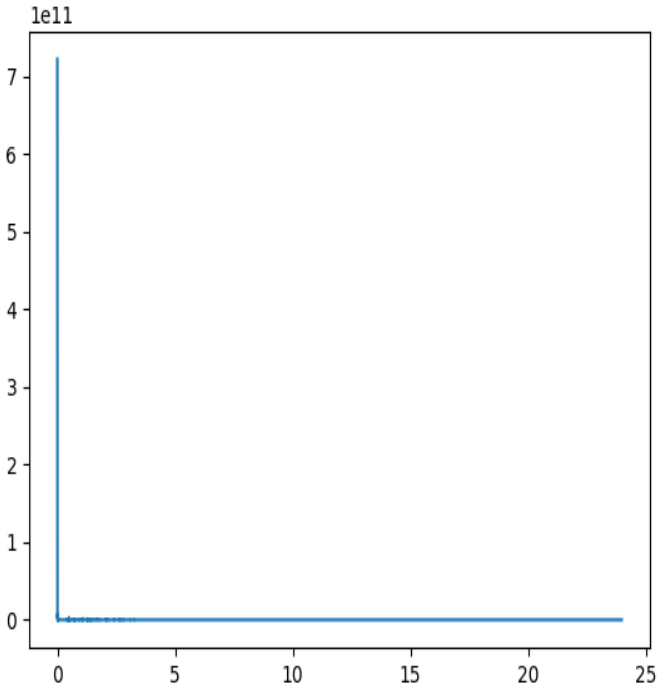
η	**Primal Objective function value**			
	Test Set 1	Test Set 2	Test Set 3	Test 4
0.0005	5697.84	5715.65	5682.88	5744.06
0.0006	5735.47	5732.90	5763.17	6008.29
0.0007	5629.10	5584.97	5676.94	5674.87
0.0008	5607.43	5431.14	5968.30	5546.35
0.0009	5532.43	5622.25	5597.94	5549.84

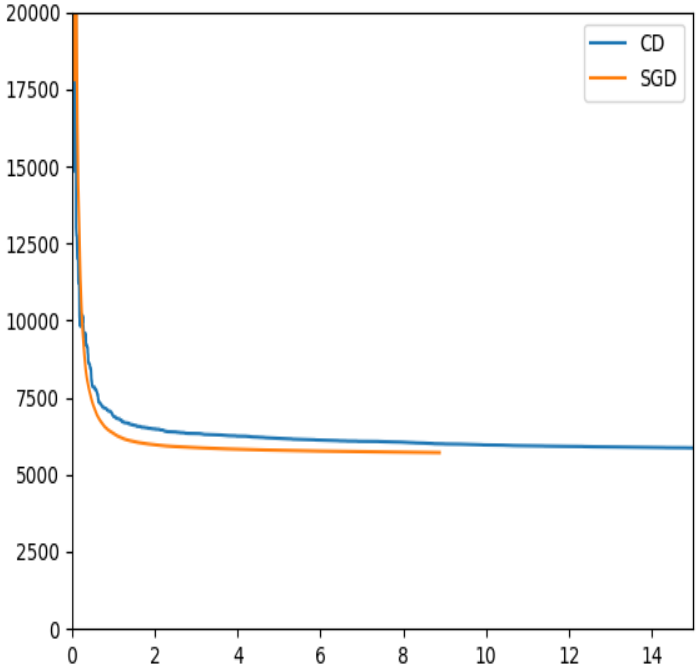
As η in the range $(0.0007, 0.0008)$ gave the lowest value of the function, running another loop for η in this range, yielded $\eta \approx 0.00078 - 0.00079$ as the best for this model.

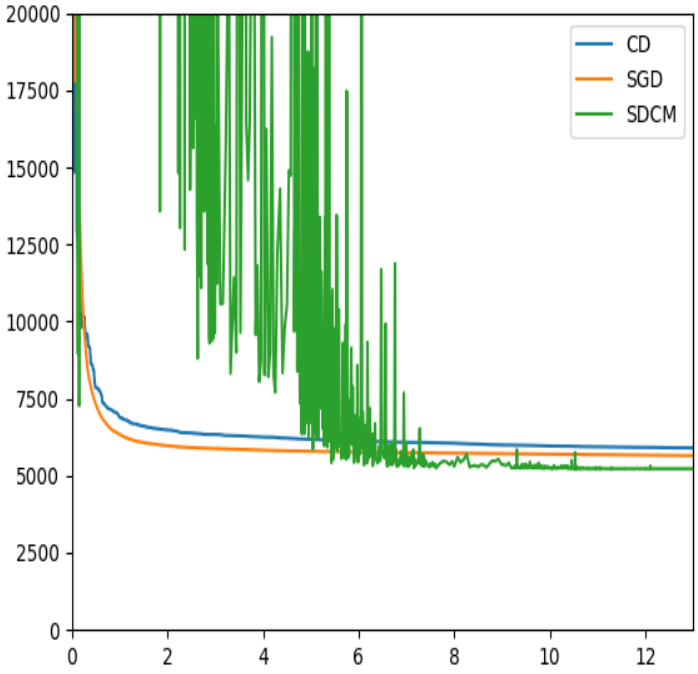
For this $\eta = 0.00078$ avg. value of primal objective function on all the separate test sets for 10 iteration yields a value of 5737.57.

6 Comparison of the three methods (Part 5)

The following Primal Objective value vs Time plot compares the three methods:







We chose SDCM over the other methods because it is giving the minimum value of the Primal Objective function.

7 Code for the best method (Part 6)

The code for the best method can be found at **https://cse.iitk.ac.in/users/asoni/submit.py** .

8 Bonus : CSVM with the positivity constraint (Part 7)

Adding the constraint that

$$\xi_i \geq 0$$

we can modify the problem obtained in part 3 as follows:

$$\arg \max_{\beta_i \geq 0, \alpha_i \geq 0} \{\arg \min_{\mathbf{w} \in \mathbb{R}^d,\ \boldsymbol{\xi} \in \mathbb{R}^n} \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{n} \xi_i^2 - \sum_{i=1}^{n} \beta_i \xi_i + \sum_{i=1}^{n} \alpha_i (1 - \xi_i - y_i(\langle \mathbf{w}, \mathbf{x}^i \rangle + b))\} \quad \text{(F)}$$

Solving the inner primal problem using the First order Conditions on **w**,ξ and b

$$\frac{\delta \mathcal{L}(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha})}{\delta \mathbf{w}} = 0$$

$$\mathbf{w} - \sum_{i=0}^{n} \alpha_i y^i \mathbf{x}^i = 0$$

$$\mathbf{w} = \sum_{i=0}^{n} \alpha_i y^i \mathbf{x}^i \tag{9}$$

and also,

$$\frac{\delta \mathcal{L}(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha})}{\delta \xi_i} = 0$$

$$2C\xi_i - \beta_i - \alpha_i = 0$$

$$\xi_i = \frac{\alpha_i + \beta_i}{2C}$$

So, we have

$$\boldsymbol{\xi} = \frac{\boldsymbol{\alpha} + \boldsymbol{\beta_i}}{2C} \tag{10}$$

and also,

$$\frac{\delta \mathcal{L}(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha})}{b} = 0$$

Thus,

$$\sum_{i=0}^{n} \alpha_i y^i = 0 \tag{11}$$

Now, we have

$$\|\mathbf{w}\|_2^2 = \sum_{i,j} \alpha_i \alpha_j y^i y^j \langle \mathbf{x}^i, \mathbf{x}^j \rangle$$

Replacing the values of **w** and $\boldsymbol{\xi}$ in (F) after solving the min criterion,and on simplifying we have:

$$\arg \max_{\beta_i \geq 0, \alpha_i \geq 0} \{\sum_{i=1}^{n} \alpha_i - \frac{1}{2}\|\mathbf{w}\|_2^2 - \sum_{i=1}^{n} \frac{(\alpha_i + \beta_i)^2}{4C}\}$$

Now the solution of this equation involves maximizing the included function which has the term

$$-\sum_{i=1}^{n} \frac{(\alpha_i + \beta_i)^2}{4C}$$

This expression however being the negative square of a real number is always less than or equal to zero and thus for the optimal constraint, we will set all of β_i's to 0 as they are not part of the function anywhere else thus maximizing the function's value. Thus the final optimization problem reduces to

$$\arg \max_{\beta_i \geq 0, \alpha_i \geq 0} \{\sum_{i=1}^{n} \alpha_i - \frac{1}{2}\|\mathbf{w}\|_2^2 - \sum_{i=1}^{n} \frac{\alpha_i^2}{4C}\}$$

which in fact is independent of all β_i's and is same as the original optimization problem without the ξ positivity constraint. Hence there is no change in the optimization problem.