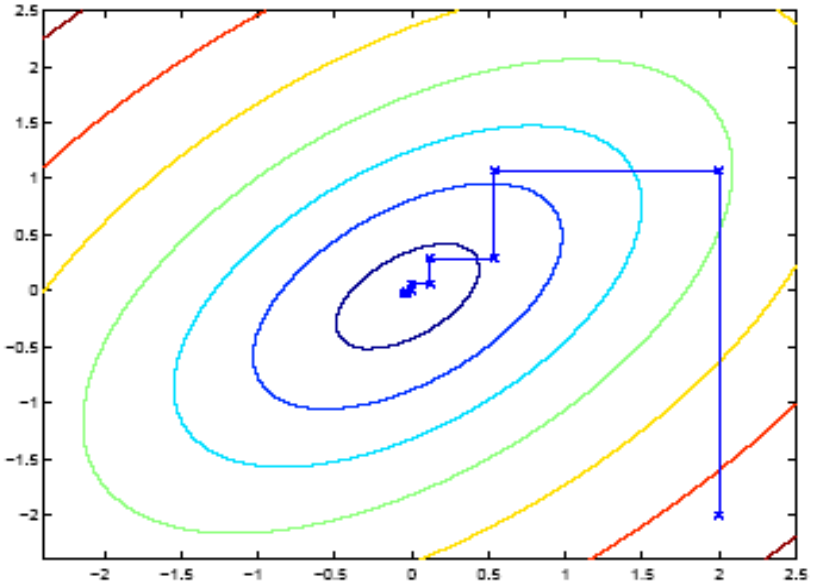
How to train SVM

Solving the quadratic programming optimization problem directly to train the SVM is very slow when the training data grows large.

Sequential minimal optimization (SMO) algorithm, due to John Platt.

First, let us introduce coordinate ascent algorithm:
Loop until convergence:
{
  For i=1, …, m
  {
    a_i:=argmax$_{ai}$$L(a_1,\ldots, a_{i-1}, a_i, a_{i+1},\ldots, a_m)$
  }
}



Coordinate ascent is ok?

$$\alpha_1 y^{(1)} = -\sum_{i=2}^{N} \alpha_i y^{(i)}$$

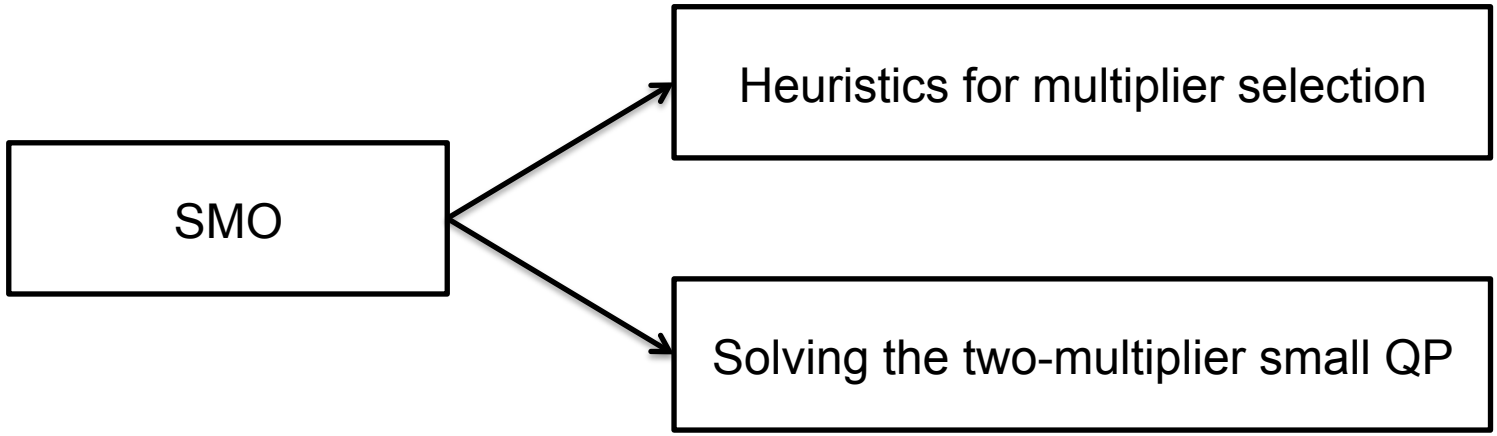
$$\alpha_1 = -y^{(1)} \sum_{i=2}^{N} \alpha_i y^{(i)}$$

Is it sufficient?

Sequential Minimal Optimization (SMO)

SMO Algorithm

Repeat until convergence
{
heuristically choosing two multipliers (α_i's)
optimize the dual problem regarding the selected multipliers (with all the other α_i's fixed)
}



Solving for Two Lagrange Multipliers

- Recall the linear equality constraint derived from KKT condition

$$\sum_{i=1}^{N} \alpha^*_i y^{(i)} = 0$$

- ***2* is the minimum number of multipliers chosen for optimization**
- If we optimize the dual objective with only one multiplier, we can no long satisfy the above constraint.
- Without loss of generality, assume that we are optimizing over alpha_1 and alpha_2, with all other alpha's fixed.

Illustration for the Constraint

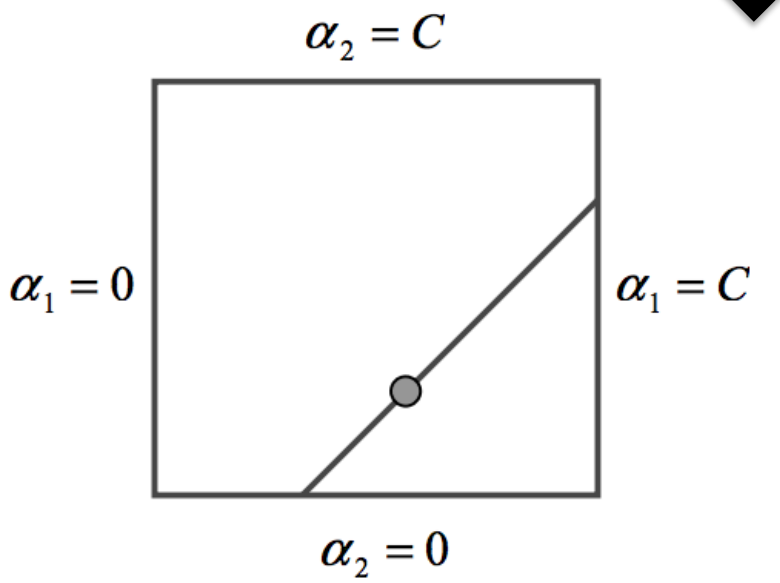
Constriant 1

$$\sum_{i=1}^{N} \alpha^*_i y^{(i)} = 0$$

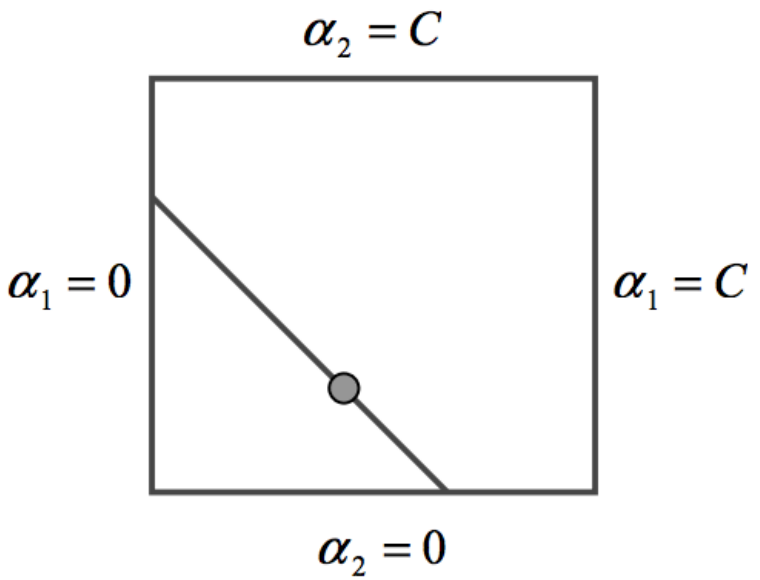
Constriant 2

$$0 \le \alpha_i \le C$$

Optimize over alpha_1 and alpha_2 with all other alpha's fixed

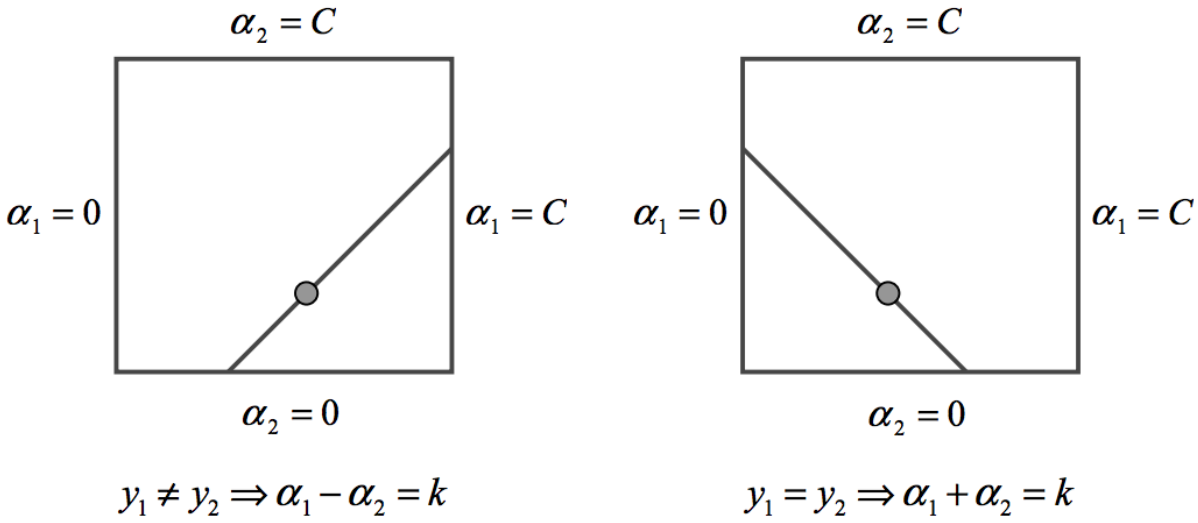


$$y_1 \neq y_2 \Rightarrow \alpha_1 - \alpha_2 = k$$



$$y_1 = y_2 \Rightarrow \alpha_1 + \alpha_2 = k$$

Bounds for alpha_2



Assume we computer the new alpha_2 first. Given the above constraint, we can bound alpha_2 by the following: (L means lower bound and H means upper bound)

$$y_1 \neq y_2 => L = \max(0, \alpha_2 - \alpha_1), H = \min(C, C + \alpha_2 - \alpha_1)$$

$$y1 = y2 => L = \max(0, \alpha_2 + \alpha_1 - C), H = \min(C, \alpha_2 + \alpha_1)$$

Objective for alpha_2

- The objective function

$$L(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} y^{(i)} y^{(j)} \alpha_i \alpha_j \left\langle x^{(i)}, x^{(j)} \right\rangle$$

- Eliminate alpha_1 based on the constraint, and treat all other alpha's as constants

$$L(\alpha_2) = \frac{1}{2} \eta \alpha_2^2 + (y_2 (E_1^{\text{old}} - E_2^{\text{old}}) - \eta \alpha_2^{\text{old}}) \alpha_2 + \text{Const}$$

$$\text{s.t. } \mathrm{L} \le \alpha_2 \le H$$

- where

$$\eta = 2\mathbf{x}_1^T \mathbf{x}_2 - \mathbf{x}_1^T \mathbf{x}_1 - \mathbf{x}_2^T \mathbf{x}_2$$

E_i^{old} is the old training error on sample i

Derivatives for alpha_2

$$L(\alpha_2) = \frac{1}{2}\eta\alpha_2^2 + (y_2(E_1^{\text{old}} - E_2^{\text{old}}) - \eta\alpha_2^{\text{old}})\alpha_2 + \text{Const}$$

$$\text{s.t. } \mathrm{L} \le \alpha_2 \le H$$

First and second order derivatives w.r.t. alpha_2:

$$\frac{dL}{d\alpha_2} = \eta\alpha_2 + (y_2(E_1^{\text{old}} - E_2^{\text{old}}) - \eta\alpha_2^{\text{old}})$$

$$\frac{d^2L}{d\alpha_2^2} = \eta$$

We have eta <= 0. Proof: $\eta = -(\mathbf{x}_2 - \mathbf{x}_1)^T(\mathbf{x}_2 - \mathbf{x}_1) = -\| \mathbf{x}_2 - \mathbf{x}_1 \|^2 \le 0$

Optimal value for alpha_2

$$\frac{dL}{d\alpha_2} = \eta\alpha_2 + (y_2(E_1^{\text{old}} - E_2^{\text{old}}) - \eta\alpha_2^{\text{old}})$$

$$\frac{d^2L}{d\alpha_2^2} = \eta$$

- If eta < 0, we can simply set the first derivative to 0 and obtain:

$$\alpha_2^{\text{new}} = \alpha_2^{\text{old}} + \frac{y_2(E_2^{\text{old}} - E_1^{\text{old}})}{n}$$

- And then we clip alpha_2 such that it falls into the constraint area

$$\alpha_2^{\text{new,clipped}} = \max(L, \min(H, \alpha_2^{\text{new}}))$$

- If eta = 0, we should also check the end points for alpha_2 and decide the optimal one.
- Then we compute the new alpha_1 as

$$\alpha_1^{\text{new}} = \alpha_1^{\text{old}} + y_1 y_2(\alpha_2^{\text{old}} - \alpha_2^{\text{new, clipped}})$$

The whole algorithm

- Given alpha_1 and alpha_2

1. Compute $\eta = 2\mathbf{x}_1^T\mathbf{x}_2 - \mathbf{x}_1^T\mathbf{x}_1 - \mathbf{x}_2^T\mathbf{x}_2$
2. If eta < 0

$$\alpha_2^{\text{new}} = \alpha_2^{\text{old}} + \frac{y_2(E_2^{\text{old}} - E_1^{\text{old}})}{n}$$

$$\alpha_2^{\text{new,clipped}} = \max(L, \min(H, \alpha_2^{\text{new}}))$$

$$\alpha_1^{\text{new}} = \alpha_1^{\text{old}} + y_1 y_2(\alpha_2^{\text{old}} - \alpha_2^{\text{new, clipped}})$$

3. If eta = 0, check the end points and select the optimal one for alpha_1 and alpha_2
4. Update related variables E, b and **w**

How to select alpha_1 and alpha_2

- The SMO algorithm is guaranteed to converge.
- Properly choosing multipliers can speed up convergence.
- Recall the KKT condition

$$\alpha_i = 0 \Leftrightarrow y_i u_i \geq 1$$

$$0 < \alpha_i < C \Leftrightarrow y_i u_i = 1$$

$$\alpha_i = C \Leftrightarrow y_i u_i \leq 1$$

- The heuristic for choosing the multipliers is based on the violation of the KKT condition

Find the first alpha

```
repeat until convergence {    // alternate between STEP 1 and 2 until convergence
    STEP 1
    for each alpha_i {            // iterate over all alpha's
        if alpha_i violates the KKT condition {
            pick alpha_i as the first alpha;
            optimize;
        }
    }
    STEP 2
    while there exist unbounded alpha_i's (0 < alpha_i < C) {      // iterate over all unbounded alpha's
        for each unbounded alpha_i {
            if alpha_i violates the KKT condition {
                pick alpha_i as the first alpha;
                optimize;
            }
        }
    }
}
```

Find the second alpha

- Given the first alpha
- Look for the second alpha that maximizes |E_2 – E_1|
- If this does not make progress (special cases), it starts a sequential scan through all unbounded alpha's, starting at a random point
- If no progress still, it starts a sequential scan through all unbounded alpha's, starting at a random point

Thanks!

HP: http://keg.cs.tsinghua.edu.cn/jietang/

Email: jietang@tsinghua.edu.cn