J. Platt
Microsoft Research

Fast Training of SVMs using Sequential Minimal Optimization

Presented by
Sudip Khanna & Ajit Datar

The Sequential Minimal Optimization (SMO) Algorithm

SMO solves the SVM QP problem by decomposing it into QP sub-problems and solving the smallest possible optimization problem, involving two Lagrange multipliers, at each step.

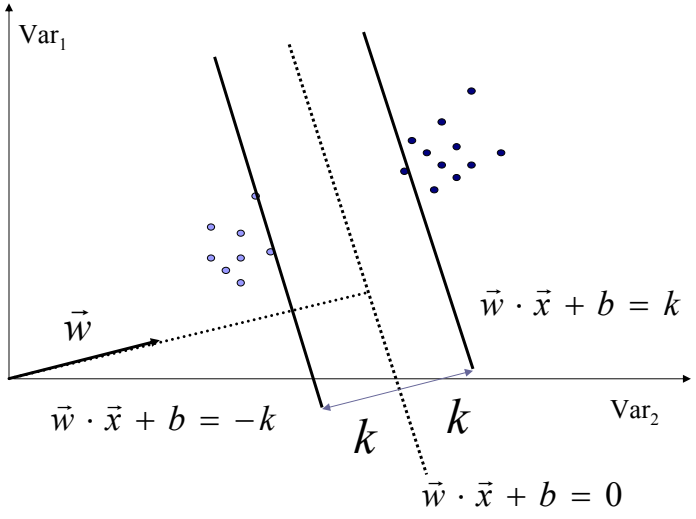
Part One - Background

- QP Problems
- SVM QP problem
- Lagrange Multipliers
- KKT Conditions

What's a QP problem?

Maximize/Minimize
a Quadratic Objective Function
subject to a Set of Linear Constraints

The SVM QP problem



The SVM QP problem

Maximize margin

$$\frac{2|k|}{\|w\|}$$

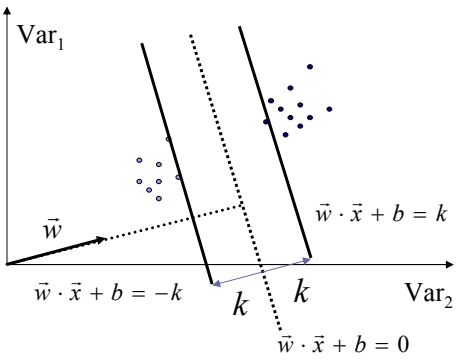
subject to

(w.x + b) ≥ k,

Vx of class1

(w.x + b) ≤ -k,

Vx of class2



The SVM QP problem

We can scale the data so that k = 1

The problem now reduces to

$$\min \frac{1}{2}\|w\|^2$$
$$s.t.\ y_i(w \cdot x_i + b) \geq 1,\ \forall x_i$$

where, y_i is the classification for example x_i (1 or -1)

The Lagrangian

The objective function and the constraints are combined in a single function

$L(\mathbf{w},b,\boldsymbol{\alpha}) = ½\|\mathbf{w}\|^2 - \Sigma\ \alpha_i\ (y_i\ .\ ((x_i.\mathbf{w})+b) - 1)$

where, Lagrange multiplier, $\alpha_i \geq 0$

For L to be maximized, only training examples with

$y_i\ .\ ((x_i.\mathbf{w})+b) - 1 = 0$ (support vectors)

will have $\alpha_i \neq 0$

BTW what's a Lagrange Multiplier?

- It is the ratio
  Gradient of Objective Function

  Gradient of Constraint Function
  At the solution of the problem
- In SVM context it allows us to simplify the constraints.
  When the problem is expressed with Lagrangian multipliers (α_i) the only constraints are non-negative α_i .

KKT conditions – what they mean

- The solution which satisfies the KKT conditions is an **optimal solution**
- In SVM equations this means
  $\Sigma\ \alpha_i\ y_i = 0$ should be satisfied
  => Only support vectors contribute to the constraints on the margin

Part Two: SMO algorithm

- Why do we need SMO?
  - Previous methods (chunking, decomposition)
- Numerical Vs Analytical Methods of optimization
- SMO: 3 part solution

Why do we need SMO?

- Current methods are based on Numerical Optimization
- Require calling library routines for solving optimization problems.
- Manipulation of large matrices => more numerical precision errors.
- Exponential memory requirements.

Previous methods

- Chunking: Optimizes chunks of examples at a time:
  - With non-zero Lagrange multipliers from last step
  - M worst examples (violators of KKT conditions)
- Decomposition: At each step add one violator example and optimize the new set.
  - Osuna's optimization: const size matrix, delete an example whenever a new one is added.

Numerical Vs Analytical solvers

- Numerical
  - ✖ QP sub-problem solved iteratively
  - ✖ Subject to precision errors due to large matrices
  - ☑ Smaller number of QP sub-problems
- Analytical
  - ☑ QP sub-problem solved in fixed number of steps.
  - ☑ Small matrices => lesser precision errors.
  - ✖ Large number of smaller QP sub-problems

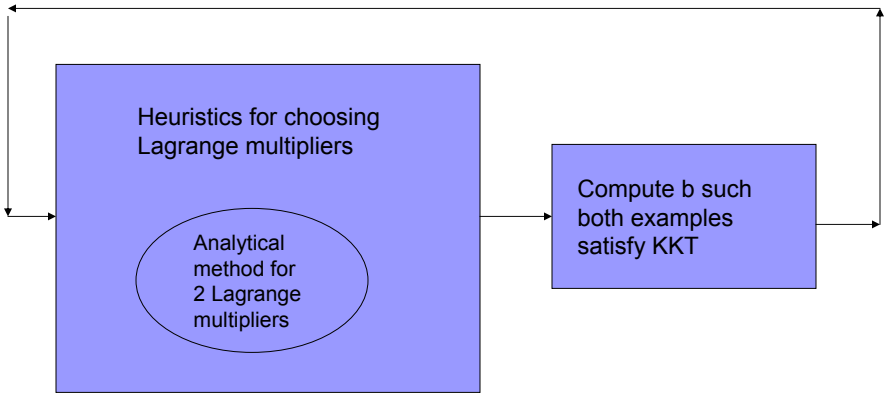
Sequential Minimal Optimization

- Not parallel
- Optimize in sets of 2 Lagrange multipliers

Optimize smallest possible sub-problem at each step.

Satisfy the constraints for the chosen pair of Lagrange multipliers.

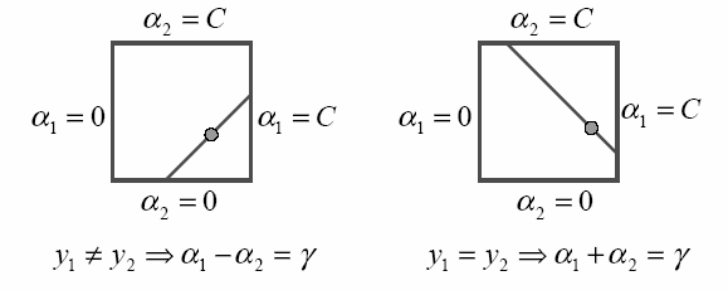
SMO components



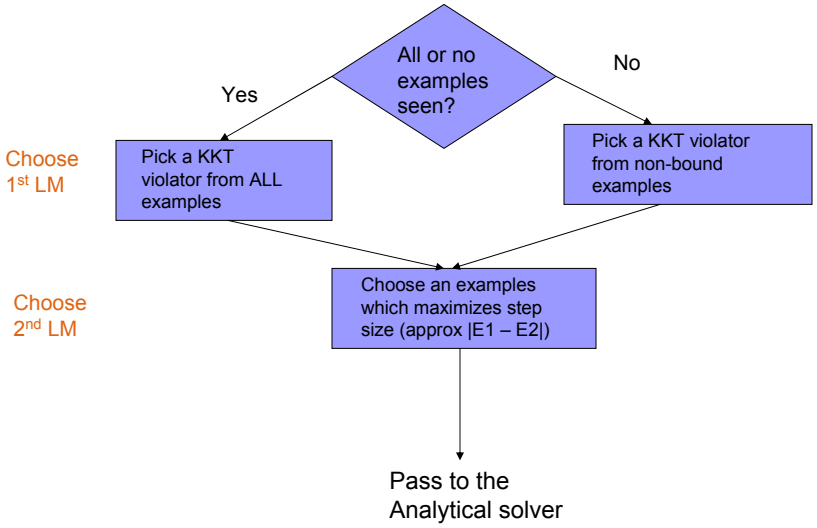
Do this until the entire training set obeys the KKT conditions

Solving for two Lagrange Multipliers

- Constraints on the Lagrange Multipliers
  - Bound constraints $0 \le \alpha_i \le C$
  - Linear equality constraint $\Sigma\ \alpha_i\ y_i = 0$



Choosing the multipliers



Calculating threshold ‘b’

- If the data is linearly separable, there is a unique value of b that maximizes margin
- b is recomputed after each step such that KKT conditions are fulfilled for both optimized examples

Speeding things up …

- Store the error associated with each example in cache
- Store and update a single weight vector which represents all examples
- SMO can take advantage of sparse input data

Relationship to previous algorithms

- Can be considered to be a special case of Osuna's algorithm
- Similar to Bregman methods of optimization

SMO & PCG Chunking on Adult dataset

Training-Set Size	SMO-Time (CPU sec)	PCG-Time (CPU sec)	SMO Iterations	PCG Iterations
1605	0.4	37.1	3230	1328
2265	0.9	228.3	4635	3964
3185	1.8	596.2	6950	6742
4781	3.6	1954.2	9847	10550
6414	5.5	3684.6	10669	12263
11221	17	20711.3	17128	25400
16101	35.3	N/A	22770	N/A
22697	85.7	N/A	35822	N/A
32562	163.6	N/A	44774	N/A

SMO & PCG Chunking on web dataset

Training-Set Size	SMO-Time (CPU sec)	PCG-Time (CPU sec)	SMO Iterations	PCG Iterations
2477	2.2	13.1	25296	1929
3470	4.9	16.1	46830	2379
4912	8.1	40.6	66890	4110
7366	12.7	140.7	88948	7416
9888	24.7	239.3	141538	8700
17188	65.4	1633.3	268907	27074
24692	104.9	3369.7	345736	32014
49749	268.3	17164.7	489302	63817

Conclusions

- SMO has potential for speed-up
- Scales well – memory footprint grows linearly with training set size
- Easier to implement – does not require a QP library