Training Algorithms for SVMs

Outline:

- Should one solve the primal or the dual problem?
- How to handle a large number of features?
- How to handle a large number of training examples?
- How to handle a large number of support vectors?

The Optimization Problem

Solve one of the following quadratic optimization problems:

$$\min P(\vec{w}, b, \vec{\xi}) = \frac{1}{2}\vec{w}\cdot\vec{w} + C\sum_{i=1}^{n}\xi_i$$

$$\text{s. t. } y_i[\vec{w}\cdot\vec{x}_i + b] \geq 1 - \xi_i \text{ and } \xi_i \geq 0$$

- $n + N + 1$ variables
- n linear inequality constrains
- no direct use of kernels
- size scales $O(nN)$

<= DUAL =>

$$\max D(\vec{\alpha}) = \left(\sum_{i=1}^{n}\alpha_i\right) - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j K(\vec{x}_i, \vec{x}_j)$$

$$\text{s. t. } \sum_{i=1}^{n}\alpha_i y_i = 0 \quad \text{and} \quad 0 \leq \alpha_i \leq C$$

- n variables
- 1 linear equality, $2n$ box constrains
- use of kernels natural
- size scales $O(n^2)$

=> positive semi-definite quadratic program with n variables

How to Tell that we Found the Optimal Solution?

Karush-Kuhn-Tucker conditions lead to the following criterion:

$$\text{maximize } D(\vec{\alpha}) = \left(\sum_{i=1}^{n}\alpha_i\right) - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j K(\vec{x}_i, \vec{x}_j)$$

$$\text{s. t. } \sum_{i=1}^{n}\alpha_i y_i = 0 \quad und \quad 0 \leq \alpha_i \leq C$$

is optimal

<=>

$$\forall i \begin{cases} (\alpha_i = 0) \Rightarrow y_i\left[\sum_{j=1}^{n}\alpha_j y_j K(\vec{x}_i, \vec{x}_j) + b\right] \geq 1 \\ (0 < \alpha_i < C) \Rightarrow y_i\left[\sum_{j=1}^{n}\alpha_j y_j K(\vec{x}_i, \vec{x}_j) + b\right] = 1 \\ (\alpha_i = C) \Rightarrow y_i\left[\sum_{j=1}^{n}\alpha_j y_j K(\vec{x}_i, \vec{x}_j) + b\right] \leq 1 \end{cases}$$

Chunking

Given: Quadratic program solver for “ small” problems.

Idea: Solve the quadratic program only for the support vectors!

Algorithm:

- Input: Training sample S, k
- initialize W to a size k random sample of examples from S
- repeat
  - solve quadratic program for W => hyperplane h
  - apply hyperplane h to the other examples in S
  - add (at most) k examples that violate KKT conditions to W
- until no more examples added to W
- return hyperplane h

Decomposition

Idea: Solve small subproblems until convergence (Osuna, et al.)!

$$
max \begin{bmatrix}1\\1\\1\\1\\1\\1\\1\end{bmatrix}^T \begin{bmatrix}\alpha_1\\\alpha_2\\\alpha_3\\\alpha_4\\\alpha_5\\\alpha_6\\\alpha_7\end{bmatrix} - \frac{1}{2} \begin{bmatrix}\alpha_1\\\alpha_2\\\alpha_3\\\alpha_4\\\alpha_5\\\alpha_6\\\alpha_7\end{bmatrix}^T \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} & k_{15} & k_{16} & k_{17} \\ k_{21} & k_{22} & k_{23} & k_{24} & k_{25} & k_{26} & k_{27} \\ k_{31} & k_{32} & k_{33} & k_{34} & k_{35} & k_{36} & k_{37} \\ k_{41} & k_{42} & k_{43} & k_{44} & k_{45} & k_{46} & k_{47} \\ k_{51} & k_{52} & k_{53} & k_{54} & k_{55} & k_{56} & k_{57} \\ k_{61} & k_{62} & k_{63} & k_{64} & k_{65} & k_{66} & k_{67} \\ k_{71} & k_{73} & k_{73} & k_{74} & k_{75} & k_{76} & k_{77} \end{bmatrix} \begin{bmatrix}\alpha_1\\\alpha_2\\\alpha_3\\\alpha_4\\\alpha_5\\\alpha_6\\\alpha_7\end{bmatrix}
$$

Time complexity: working set of size $2 \leq q \leq 100$ and f nonzero features:

- extracting subproblem: $O(q^2 f)$
- solving subproblem: $O(q^3)$
- updating large problem with result of subproblem: $O(nqf)$

Solving the QP for the Working Set

Special Case q=2 variables α_a and α_b [Platt/SMO]:

$$maxD(\vec{\alpha}) = \sum_{i \in \{a,b\}} \left(1 - \sum_{j \notin \{a,b\}} \alpha_j y_i y_j K(\vec{x}_i, \vec{x}_j)\right)\alpha_i - \frac{1}{2} \sum_{i \in \{a,b\}} \sum_{j \in \{a,b\}} \alpha_i \alpha_j y_i y_j K(\vec{x}_i, \vec{x}_j)$$

$$\alpha_a y_a + \alpha_b y_b = \sum_{i \notin \{a,b\}} \alpha_i y_i$$

$$0 \leq \alpha_a \leq C \qquad 0 \leq \alpha_b \leq C$$

- case 1: maximum at $min(\alpha_a, \alpha_a) = 0$
- case 2: maximum at $max(\alpha_a, \alpha_a) = C$
- case 3: maximum at interior point

$$\alpha_a = y_a \frac{y_a\left(1 - \sum_{j \notin \{a,b\}} \alpha_j y_a y_j K(\vec{x}_a, \vec{x}_j)\right) - y_b\left(1 - \sum_{j \notin \{a,b\}} \alpha_j y_b y_j K(\vec{x}_b, \vec{x}_j)\right) + (K(\vec{x}_b, \vec{x}_b) - K(\vec{x}_b, \vec{x}_a)) \sum_{i \notin \{a,b\}} \alpha_i y_i}{K(\vec{x}_a, \vec{x}_a) + K(\vec{x}_b, \vec{x}_b) - 2K(\vec{x}_b, \vec{x}_a)}$$

$$\alpha_b = y_b\left(-\alpha_a y_a + \sum_{i \notin \{a,b\}} \alpha_i y_i\right)$$

What Working Set to Select Next?

Solution: Select subproblem with q variables that minimizes

$$V(\vec{d}) = g(\vec{\alpha})^T \vec{d}$$

$$\text{subject to} \quad \begin{aligned} \vec{y}^T \vec{d} &= 0 \\ d_i \geq 0, &\ if(\alpha_i = 0) \\ d_i \leq 0, &\ if(\alpha_i = C) \\ -1 \leq &\ \vec{d} \leq 1 \\ |\{d_i \neq 0\}| &= q \end{aligned}$$

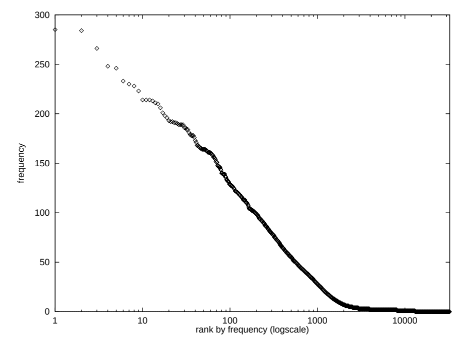
Efficiency: Selection linear in number of examples.

Convergence: Proofs by Chi-Chen Lin / Keerthi under mild assumptions.

Caching

Observation: Most CPU-time is spent on computing the Hessian!

Idea: Cache kernel evaluations.

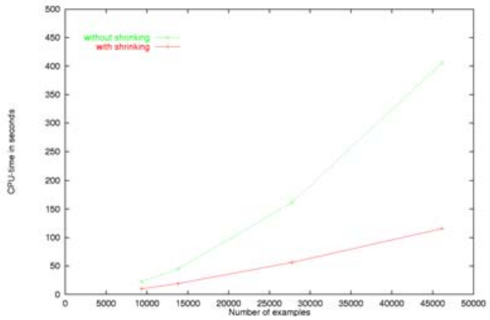


Result: A small cache leads to a large improvement.

Shrinking

Idea: If we knew the set of SVs, we could solve a smaller problem! (complexity per iteration from $O(nqf)$ to $O(sqf)$)

Algorithm:
- monitor the KKT-conditions in each iteration
- if a variable is " stuck at bound" , remove it
- do final optimality check



Other Training Algorithms

- ASVM restricted to linear SVMs with quadratic loss => fast for low dimensional data [Mangasarian & Musicant, 2000]
- Nearest Point Algorithm restricted to quadratic loss => compute distance between convex hulls [Keerthi et al., 1999]
- Kernel Adatron => very easy to implement [Friess et al., 1998]