The Optimization Problem

Solve one of the following quadratic optimization problems:

\[
\begin{align*}
\min \quad & P(w, b, \xi) = \frac{1}{2} w \cdot w + C \sum_{i=1}^{n} \xi_i \\
\text{s.t.} \quad & y_i (w \cdot x_i + b) \geq 1 - \xi_i \quad \text{and} \quad \xi_i \geq 0
\end{align*}
\]

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

\[ \leq \text{DUAL} \Rightarrow \]

\[
\begin{align*}
\max \quad & D(\alpha) = \frac{1}{2} \sum_{i=1}^{n} \alpha_i - \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \\
\text{s.t.} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0 \quad \text{and} \quad 0 \leq \alpha_i \leq C
\end{align*}
\]

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

\[ \Rightarrow \text{positive semi-definite quadratic program with} \ n \ \text{variables} \]

How to Tell that we Found the Optimal Solution?

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

\[
\begin{align*}
\text{maximize} \quad & D(\alpha) = \frac{1}{2} \sum_{i=1}^{n} \alpha_i \alpha_j - \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \\
\text{s.t.} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0 \quad \text{and} \quad 0 \leq \alpha_i \leq C
\end{align*}
\]

\[
\Rightarrow \begin{cases}
\hat{\alpha}_i (\alpha_i = 0) \hat{y}_i \left( \sum_{j=1}^{n} \hat{\alpha}_j \hat{y}_j K(x_i, x_j) + b \right) \geq 1 \\
\forall i \ (0 < \alpha_i < C) \hat{y}_i \left( \sum_{j=1}^{n} \hat{\alpha}_j \hat{y}_j K(x_i, x_j) + b \right) = 1 \\
\hat{\alpha}_i (\alpha_i = C) \hat{y}_i \left( \sum_{j=1}^{n} \hat{\alpha}_j \hat{y}_j K(x_i, x_j) + b \right) \leq 1
\end{cases}
\]

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 \Rightarrow 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.)!

\[
\begin{bmatrix}
1^T \\ 0^T \\ \alpha^T
\end{bmatrix}
\begin{bmatrix}
\alpha_1 \\ \alpha_2 \\ k_{11} k_{12} k_{13} k_{14} k_{15} k_{16} k_{17}
\end{bmatrix}
\begin{bmatrix}
\alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\
\alpha_6 \\ \alpha_7 
\end{bmatrix}
\]

\[\max_{\alpha} \quad \frac{1}{2} a \tilde{a} + a \hat{\alpha} \tilde{y} \tilde{y}^T K(x_i, x_j) + \frac{1}{2} \hat{\alpha} \hat{\alpha} \hat{\alpha} \hat{\alpha} \tilde{y} \tilde{y}^T K(x_i, x_j)
\]

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 \(\alpha_a\) and \(\alpha_b\) [Platt/SMO]:

\[
\begin{align*}
\max_{\alpha} & \quad a \tilde{a} - a \hat{\alpha} \tilde{y} \tilde{y}^T K(x_i, x_j) + \frac{1}{2} \hat{\alpha} \hat{\alpha} \hat{\alpha} \hat{\alpha} \tilde{y} \tilde{y}^T K(x_i, x_j) \\
\text{s.t.} & \quad \alpha_a + \alpha_b = 0
\end{align*}
\]

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

**What Working Set to Select Next?**

**Solution:** Select subproblem with \(q\) variables that minimizes

\[
V(d) = g(d)^T \hat{d}
\]

\[
\hat{d} = 0
\]

subject to \(d_i \geq 0, \text{if} (\alpha_i = 0)\)

\(d_i \leq 0, \text{if} (\alpha_i = C)\)

\(-1 \leq \hat{d} \leq 1\)

\(|\{d_i \neq 0\}| = q\)

**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]