A First Glance at Support Vector Machines

Andreas Maletti [1]

Department of Computer Science
Dresden University of Technology

National Taiwan University, September 12, 2001
based on slides provided by Chih-Jen Lin, CS Dep. at NTU

[1] sponsored by DAAD/NSC

Itinerary

- Motivation and history
- Linear learning machines
- Feature spaces and kernels
- Performance considerations
- Optimization algorithms

Motivation

The generic problem: Classify a given input

1. two classes (binary classification)
2. *several, but finitely many classes (multi-class classification)*
3. *infinitely many classes (regression)*

Applications:

- Handwritten digits recognition
- Speech recognition
- Text classification
- Face recognition

The proposed solution: supervised learning, so given (non-trivial) training data in *different* classes (labels known) predict test data (labels unknown).

More formally: *Given a training set* $S \subseteq \mathbb{R}^n \times \{-1, 1\}$ *of correctly classified input data vectors* $\vec{x} \in \mathbb{R}^n$, *where every input data vector appears at most once in* S *and there exist input data vectors* $\vec{p}$ *and* $\vec{n}$ *such that* $(\vec{p}, 1) \in S$ *as well as* $(\vec{n}, -1) \in S$ *(non-trivial* S*), successfully classify unseen input data vectors.*

Contestants:

- Nearest Neighbor
- Neural Networks
- Decision Trees

Different approaches:

- unsupervised learning
- query learning
- reinforcement learning

Goal: Performing better than the competitors in relevant applications

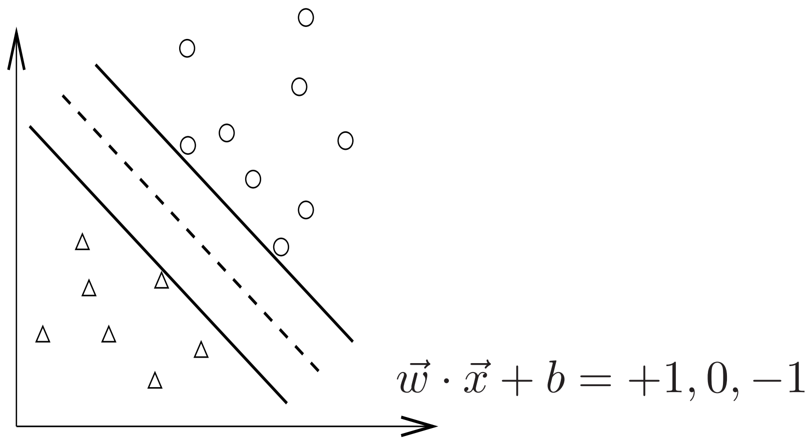
History

- Support Vector Machines are a rather new field of study
- Early development in Bell Labs from 1990 to 1995
- Proposed by Vapnik and co-workers in 1992
- Since then it is becoming more and more popular
- Is still a field of active research

Linear Learning Machines

Given: A training set S

Wanted: A hyperplane separating the input space into halves containing only elements of one class



Variables:

$\vec{x}$; $\vec{x}_i$	*input data vector* ($\vec{x} \in \mathbb{R}^n$); specific input data vector
y; y_i	*classifier* ($y \in \{1, -1\}$); classifier for $\vec{x}_i$, so $(\vec{x}_i, y_i) \in S$
$\vec{w}$	*weight vector* (normal vector) of a hyperplane ($\vec{w} \in \mathbb{R}^n$)
b	*bias* of a hyperplane ($b \in \mathbb{R}$)

Representation of a separating hyperplane: $\vec{w} \cdot \vec{x} + b = 0$

$$\vec{w} \cdot \vec{x}_i + b \begin{cases} > 0 & \text{, if } y_i = 1 \\ < 0 & \text{, if } y_i = -1 \end{cases}$$

Decision function: $f(\vec{x}) = \text{sgn}(\vec{w} \cdot \vec{x} + b)$

Goal: learn the coefficients $\vec{w}$ and b of the hyperplane

Problem: Many possible choices of $\vec{w}$ and b

Solution: Select $\vec{w}$ and b with the maximal margin (maximal distance to any input data vector)

Observations reveal (cf. Vapnik's statistical learning theory)

$$\vec{w} \cdot \vec{x}_i + b \begin{cases} \geq 1 & , \text{if } y_i = 1 \\ \leq -1 & , \text{if } y_i = -1 \end{cases} \tag{1}$$

Scaling does not change the hyperplane, but it does change the *margin*, so adjust the scaling such that the closest points have functional margin 1 ($f(\vec{x}) = 1$)

$\Rightarrow$Maximize distance between $\vec{w} \cdot \vec{x} + b = \pm 1$

Distance between $\vec{w} \cdot \vec{x} + b = 1$ and -1 (closest points $\vec{x}^+$ and $\vec{x}^-$):

$$\frac{\vec{w} \cdot \vec{x}^+ + b - \vec{w} \cdot \vec{x}^- - b}{\|\vec{w}\,\|} = \frac{2}{\|\vec{w}\,\|} = \frac{2}{\sqrt{\vec{w} \cdot \vec{w}}}.$$

Since $\max \frac{2}{\|\vec{w}\,\|} \equiv \min \frac{\vec{w} \cdot \vec{w}}{2}$ we finally gain the **optimization problem:**

$$\begin{array}{lll} \text{target function:} & \min_{\vec{w},b} & \dfrac{\vec{w} \cdot \vec{w}}{2} \\ \text{conditions:} & & y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1, \qquad \text{from (1)} \\ & & \text{for } i = 1, \ldots, l. \end{array}$$

$\Rightarrow$ This optimization problem is the basic (primal) Support Vector Machine form.

Higher Dimensional Feature Spaces

Problem: We tried to find a linear separating hyperplane, but data may not be linear separable

Non-separable case: allow training errors ξ_i

$$\text{target function:} \quad \min_{\vec{w},b,\vec{\xi}} \quad \frac{\vec{w}\cdot\vec{w}}{2} + C\sum_{i=1}^{l}\xi_i$$

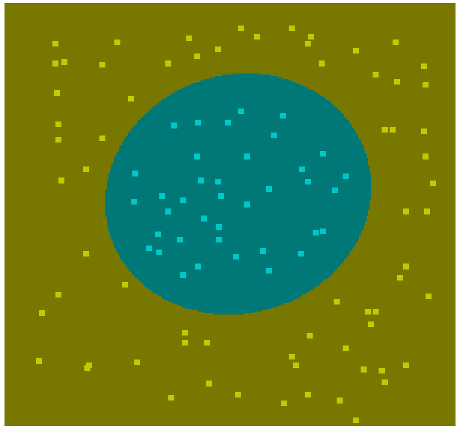
$$\text{conditions:} \quad y_i(\vec{w}\cdot\vec{x}_i + b) \geq 1 - \xi_i,$$

$$\xi_i \geq 0; \quad \text{for } i = 1, \ldots, l$$

If $\xi_i > 1$ **then** $\vec{x}_i$ not on the correct side of the separating plane

Parameter C: large penalty parameter, so most ξ_i are zero

Nonlinear case: linear separable in other spaces ?



Higher dimensional (maybe infinite) feature space

$$\phi(\vec{x}) = (\phi_1(\vec{x}), \phi_2(\vec{x}), \ldots).$$

Example: $\vec{x} \in R^3, \phi(\vec{x}) \in R^{10}$

$$\begin{aligned}\phi(\vec{x}) &= (1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_3, x_1^2, \\ &\quad x_2^2, x_3^2, \sqrt{2}x_1x_2, \sqrt{2}x_1x_3, \sqrt{2}x_2x_3)\end{aligned}$$

Why higher dimensional spaces: a classic result by Cover [1965]

A standard problem [Cortes and Vapnik, 1995]:

$$\begin{aligned} \text{target function:} \quad & \min_{\vec{w},b,\vec{\xi}} \quad \frac{\vec{w}\cdot\vec{w}}{2} + C(\sum_{i=1}^{l}\xi_i) \\ \text{conditions:} \quad & y_i(\vec{w}\cdot\phi(\vec{x}_i)+b) \geq 1-\xi_i, \\ & \xi_i \geq 0; \quad \text{for } i=1,\ldots,l \end{aligned}$$

Other variants (though similar); Example:

$$\begin{aligned} \text{target function:} \quad & \min_{\vec{w},b,\vec{\xi}} \quad \frac{\vec{w}\cdot\vec{w}}{2} + C(\sum_{i=1}^{l}\xi_i^2) \\ \text{conditions:} \quad & y_i(\vec{w}\cdot\phi(\vec{x}_i)+b) \geq 1-\xi_i, \\ & \xi_i \geq 0; \quad \text{for } i=1,\ldots,l \end{aligned}$$

Finding the Decision Function

Next problem: Finding $\vec{w}$ and b from the standard Support Vector Machine form

$\vec{w}$ is a vector in a high dimensional space $\Rightarrow$ perhaps infinite

Therefore we consider the dual problem:

$$
\begin{array}{rll}
\text{target function:} & \min_{\vec{\alpha}} & \dfrac{\vec{\alpha}^T \mathbf{Q} \vec{\alpha}}{2} - \displaystyle\sum_{i=1}^{l} \alpha_i \\
\text{conditions:} & & 0 \leq \alpha_i \leq C; \text{ for } i = 1, \ldots, l \\
& & \vec{y} \cdot \vec{\alpha} = 0,
\end{array}
$$

where $\mathbf{Q}_{ij} = y_i\, y_j\, \phi(\vec{x}_i) \cdot \phi(\vec{x}_j)$

$$\boxed{\vec{w} = \sum_{i=1}^{l} \alpha_i\, y_i\, \phi(\vec{x}_i)}$$

Remarks:

- Primal and dual: cf. optimization theory
- $\Rightarrow$ Infinite dimensional programming
- $\mathbf{Q}_{ij} = y_i\, y_j\, \phi(\vec{x}_i) \cdot \phi(\vec{x}_j)$ needs a closed form
- $\Rightarrow$ Efficient calculation of high dimensional inner products

  Example: $\vec{x}_i \in \mathbb{R}^3, \phi(\vec{x}_i) \in \mathbb{R}^{10}$

$$\begin{aligned} \phi(\vec{x}_i) \quad = \quad & (1, \sqrt{2}(\vec{x}_i)_1, \sqrt{2}(\vec{x}_i)_2, \sqrt{2}(\vec{x}_i)_3, (\vec{x}_i)_1^2, (\vec{x}_i)_2^2, (\vec{x}_i)_3^2, \\ & \sqrt{2}(\vec{x}_i)_1(\vec{x}_i)_2, \sqrt{2}(\vec{x}_i)_1(\vec{x}_i)_3, \sqrt{2}(\vec{x}_i)_2(\vec{x}_i)_3), \end{aligned}$$

  Then $K(\vec{x}_i, \vec{x}_j) = \phi(\vec{x}_i) \cdot \phi(\vec{x}_j) = (1 + \vec{x}_i \cdot \vec{x}_j)^2$.

Such a K-function is called **kernel function**, representing the inner product of two feature space vectors.

Popular methods (kernels) $\phi(\vec{x}_i) \cdot \phi(\vec{x}_j) =$

- $e^{-\gamma\|\vec{x}_i - \vec{x}_j\|^2}$ (Radial Basis Function),
- $(\frac{\vec{x}_i \cdot \vec{x}_j}{a+b})^d$ (Polynomial kernel),
- $\tanh(a\,\vec{x}_i \cdot \vec{x}_j + b)$

Decision function:

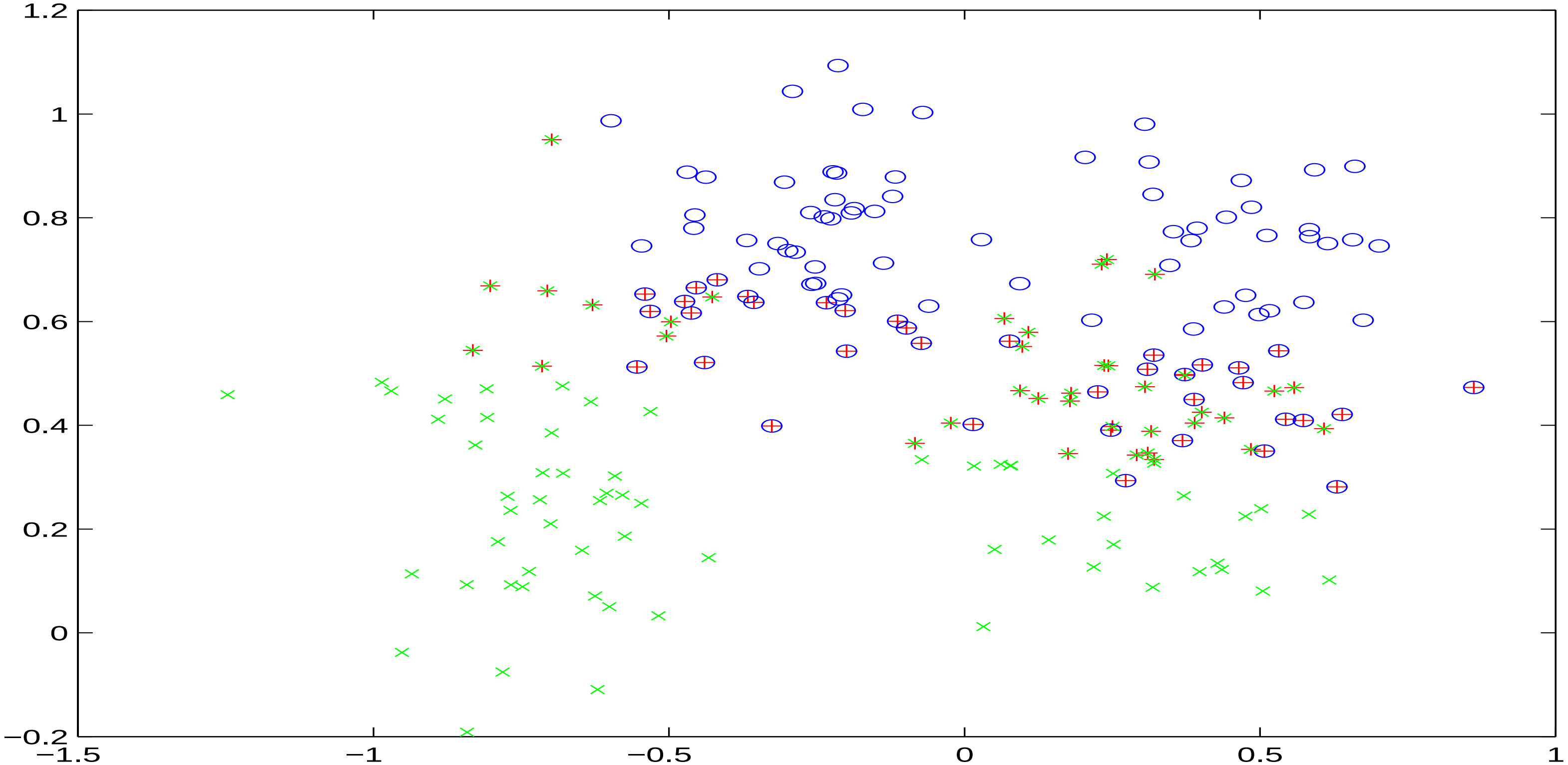
$$\mathrm{sgn}\Big(\vec{w} \cdot \phi(\vec{x}) + b\Big) \;=\; \mathrm{sgn}\Big(\sum_{i=1}^{l} \alpha_i\, y_i\, \phi(\vec{x}_i) \cdot \phi(\vec{x}) + b\Big)$$

$\Rightarrow$ No need to have $\vec{w}$

Only $\phi(\vec{x}_i)$ of $\alpha_i > 0$ used

$\alpha_i > 0 \Rightarrow$ support vectors

Support Vectors: More Important Data



Issues

Why is this good ? Statistical learning theory

- Solving large quadratic problems: dual variable α
- Multiple-class classifications
  - Several two-class problems or combined together
- Automatic model selection
  - select the best parameters (kernel type, C, etc)
- Comparisons with other methods
- Applications

Performance considerations

- Training errors not important; only test errors count
- If **Q** is positive definite, training can be fully separated
- l observations, $\vec{x}_i \in \mathbb{R}^n, i = 1, \ldots, l$, a learning machine:

$$\vec{x} \rightarrow f(\vec{x}, \vec{\alpha}), \quad f(\vec{x}, \vec{\alpha}) = 1 \text{ or } -1.$$

  $\Rightarrow$ Different $\vec{\alpha}$: different machines

- The expected test error (generalized error)

$$R(\vec{\alpha}) = \int \frac{1}{2}|y - f(\vec{x}, \vec{\alpha})| dP(\vec{x}, y)$$

  y: class of $\vec{x}$ (i.e. 1 or -1)

- $P(\vec{x}, y)$ unknown, empirical risk (training error):

$$R_{emp}(\vec{\alpha}) = \frac{1}{2l}\sum_{i=1}^{l} |y_i - f(\vec{x}_i, \vec{\alpha})|$$

- $\frac{1}{2}|y_i - f(\vec{x}_i, \vec{\alpha})|$: loss, choose $0 \leq \eta \leq 1$
  With probability at least $1 - \eta$:

$$R(\vec{\alpha}) \leq R_{emp}(\vec{\alpha}) + \sqrt{\frac{h(\log(2l/h) + 1) - \log(\eta/4)}{l}}$$

- h is the Vapnik Chervonenkis (VC) dimension
- A bound to judge the performance of a learning machine
- Independent of data distributions
- A good pattern recognition method: minimize both terms at the same time

- Support Vector Machine bound:
  Given $\vec{x}_1, \ldots, \vec{x}_l$

  $$\mathcal{F} = \{\vec{x} \to \vec{w} \cdot \vec{x} \mid \|\vec{w}\| \leq 1, \|\vec{x}\| \leq R\}$$

  With probability at least $1 - \eta$, if $\mathrm{sgn}(f) \in \mathrm{sgn}(\mathcal{F})$ has margin at least γ on all $\vec{x}_i$:

  $$\boxed{R(\vec{\alpha}) \leq R_{emp}(\vec{\alpha}) + \sqrt{\tfrac{c}{l}(\tfrac{R^2}{\gamma^2}\log^2 l + \log\tfrac{1}{\eta})}}$$

- γ^2: as large as possible

- Support Vector Machine:

$$\begin{aligned} \text{target function:} \quad & \min_{\vec{w},b,\vec{\xi}} \quad \frac{\vec{w}\cdot\vec{w}}{2} + C(\sum_{i=1}^{l}\xi_i) \\ \text{conditions:} \quad & y_i(\vec{w}\cdot\phi(\vec{x}_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0; \text{ for } i = 1, \ldots, l \end{aligned}$$

equivalent to

$$\min \frac{\vec{w} \cdot \vec{w}}{2} + \sum [-y_i(\vec{w} \cdot \phi(\vec{x}_i) + b) + 1]_+$$

$\sum_{i=1}^{l}$: training errors; SVM: search for a balance

- Continuous loss function ? Loss of sparsity: all $\alpha_i \neq 0$
- $\frac{\vec{w} \cdot \vec{w}}{2}$ usually called regularization term
- This kind of bounds are still very loose

Primal and Dual Relation

- Simplified primal:

$$\text{target function:} \quad \min_{\vec{w},b} \quad \frac{\vec{w}\cdot\vec{w}}{2}$$
$$\text{conditions:} \quad y_i(\vec{w}\cdot\vec{x}_i + b) \geq 1$$

- Simplified dual:

$$\text{target function:} \quad \min_{\vec{\alpha}} \quad \frac{\vec{\alpha}^T\mathbf{Q}\vec{\alpha}}{2} - \sum_{i=1}^{l}\alpha_i$$
$$\text{conditions:} \quad 0 \leq \alpha_i;\ \text{for } i = 1,\ldots,l$$
$$\vec{y}\cdot\vec{\alpha} = 0,$$

where $\mathbf{Q}_{ij} = y_i\, y_j\, \vec{x}_i \cdot \vec{x}_j$

Karush-Kuhn-Tucker (KKT) condition

- Given the optimization problem

$$\begin{aligned} \text{target function:} \quad & \min_{\vec{x}} \quad f(\vec{x}),\ \vec{x} \in \mathbb{R}^n \\ \text{conditions:} \quad & \qquad\quad g_i(\vec{x}) \geq 0;\ \text{for } i = 1, \ldots, m \\ & \qquad\quad h_j(\vec{x}) = 0;\ \text{for } j = 1, \ldots, l \end{aligned}$$

- Corresponding Lagrangian function

$$L(\vec{x}, \vec{\lambda}, \vec{\mu}) = f(\vec{x}) - \sum_i \lambda_i g_i(\vec{x}) + \sum_j \mu_j h_j(\vec{x})$$

where λ_i, μ_i: Lagrange multiplier

- **KKT-conditions:**

$$
\begin{aligned}
\frac{\partial L(\vec{x}^*, \vec{\lambda}^*, \vec{\mu}^*)}{\partial \vec{x}} &= 0 \\
\frac{\partial L(\vec{x}^*, \vec{\lambda}^*, \vec{\mu}^*)}{\partial \vec{\mu}} &= 0 \\
\lambda_i g_i(\vec{x}^*) &= 0; \text{ for } i = 1, \ldots, m \\
g_i(\vec{x}^*) &\geq 0; \text{ for } i = 1, \ldots, m \\
\lambda_i^* &\geq 0; \text{ for } i = 1, \ldots, m
\end{aligned}
$$

- Convex programming: convex objective function and convex feasible region
- Linear constraints
- $\Rightarrow$ If there exist $\vec{\lambda}^*$ and $\vec{\mu}^*$ for some $\vec{x}^*$ and the conditions above are met, then $\vec{x}^*$ is an optimum.

necessary and sufficient condition

- The KKT condition of the dual:

$$\mathbf{Q}\vec{\alpha} - \vec{E} = -b\vec{y} + \vec{\lambda}$$
$$\alpha_i \lambda_i = 0$$
$$\vec{\lambda} \geq 0$$

- The KKT condition of the primal:

$$\vec{w} = \sum_{i=1}^{l} \alpha_i \vec{x}_i$$
$$\alpha_i(\vec{w} \cdot \vec{x}_i + by_i - 1) = 0$$
$$\vec{y} \cdot \vec{\alpha} = 0$$
$$\vec{\alpha} \geq 0$$

- Let $\lambda_i = y_i(\vec{w} \cdot \vec{x}_i + b) - 1$,

$$
\begin{aligned}
& (\mathbf{Q}\vec{\alpha} - \vec{E} + b\vec{y})_i \\
= & \sum_j y_i \, y_j \, \alpha_j \, \vec{x}_i \cdot \vec{x}_j - 1 + b y_i \\
= & y_i \, \vec{w} \cdot \vec{x}_i - 1 + y_i b \\
= & y_i(\vec{w} \cdot \vec{x}_i + b) - 1
\end{aligned}
$$

- The KKT of the primal is the same as the KKT of the dual (cf. strong duality theorem)

Large Dense Quadratic Programming

- $\min \frac{\vec{\alpha}^T \mathbf{Q} \vec{\alpha}}{2} - \sum_{i=1}^{l} \alpha_i$; for $\vec{y} \cdot \vec{\alpha} = 0$, $0 \leq \alpha_i \leq C$
- $\mathbf{Q}_{ij} \neq 0$, $\mathbf{Q}$: an l by l fully dense matrix
- 30,000 training points: 30,000 variables: $(30,000^2 \times 8/2)$ bytes $=$ 3GB RAM: still difficult
- Traditional methods: Newton, Quasi Newton cannot be directly applied
- Current methods:
  - Decomposition methods (Osuna et al. [1997], Joachims [1998], Platt [1998])
  - *Nearest point of two convex hulls (Keerthi et al. [1999a])*

Decomposition Methods: Avoid Memory Problem

- Working on a few variable each time
- It is like that for minimizing a function with 10 variables, you sequentially work on one variable in each iteration
- Working set B, $N = \{1, \ldots, l\} \backslash B$ fixed ; Size of B usually <= 100
- Sub-problem in each iteration:

$$\begin{aligned} \text{target function:} \quad & \min_{\vec{\alpha}_B} \quad \frac{\vec{\alpha}_B^T \mathbf{Q_{BB}} \vec{\alpha}_B}{2} + (\vec{E}_B + \mathbf{Q_{BN}} \vec{\alpha}_N^k) \cdot \vec{\alpha}_B \\ \text{conditions:} \quad & \vec{y}_B \cdot \vec{\alpha}_B = -\vec{y}_N \cdot \vec{\alpha}_N^k, \\ & 0 \le \vec{\alpha}_B \le C \end{aligned}$$

Decomposition Algorithm: a Framework

1. Given $q \ll l$, $\vec{\alpha}^1$: initial solution. $k \leftarrow 1$.
2. If $\vec{\alpha}^k$ an optimum, stop. Find a working set $B \subset \{1, \ldots, l\}$, $|B| = q$. Define $N \equiv \{1, \ldots, l\} \backslash B$, $\vec{\alpha}_B^k$ and $\vec{\alpha}_N^k$
3. Solve a sub-problem:

$$\begin{aligned} \text{target function:} \quad & \min_{\vec{\alpha}_B} \quad \frac{\vec{\alpha}_B^T \mathbf{Q_{BB}} \vec{\alpha}_B}{2} - (\vec{E}_B - \mathbf{Q_{BN}} \vec{\alpha}_N^k) \cdot \vec{\alpha}_B \\ \text{conditions:} \quad & 0 \leq (\vec{\alpha_B})_i \leq C; \text{ for } i = 1, \ldots, q, \\ & \vec{y}_B \cdot \vec{\alpha}_B = -\vec{y}_N \cdot \vec{\alpha}_N^k \end{aligned}$$

4. Set $\vec{\alpha}_B^{k+1}$ and $\vec{\alpha}_N^{k+1}$, $k \leftarrow k+1$ and goto Step 2.

- Submatrices $\mathbf{Q_{BB}}$ and $\mathbf{Q_{BN}}$ needed; calculated when needed: avoid the memory problem

- The objective function is decreasing ; Convergence was not fully understood
- Studies on convergence proofs: (Chang et al. [1999], Keerthi and Gilbert [2000], Lin [2000])
- Implementation: need knowledge of optimization
- Early implementation: Working set by heuristics; Stopping conditions not validated
- Starting from zero vector ; Efficient when the percentage of support vectors is small
- Still slow in some difficult cases

- Someone asked: the dual form is simple; why not solve it analytically ? Solve

$$\min \frac{\vec{x}^T \mathbf{A} \vec{x}}{2} - \vec{b} \cdot \vec{x}$$

by

$$\mathbf{A}\vec{x} = \vec{b} \Rightarrow \vec{x} = \mathbf{A}^{-1}\vec{b}$$

is not the end but just the beginning

$\Rightarrow$ Numerical analysis techniques are important

A Simple Implementation

- Consider $|B| = 2$, Sequential Minimal Optimization (SMO) by Platt [1998]
- Sub-problem analytically solved; no need to use optimization software
- Contained flaws; modified by Keerthi et al. [1999]
- KKT of the dual:

$$
\begin{aligned}
\mathbf{Q}\vec{\alpha} - \vec{E} &= -b\vec{y} + \vec{\lambda} - \vec{\mu} \\
\alpha_i \lambda_i = 0; &\quad \mu_i (C - \alpha_i) = 0 \\
\vec{\lambda} \geq 0; &\quad \vec{\mu} \geq 0
\end{aligned}
$$

- Equivalent to

$$(\mathbf{Q}\vec{\alpha} - \vec{E} + b\vec{y})_t \begin{cases} \geq 0 & \text{, if } \alpha_t < C \\ \leq 0 & \text{, if } \alpha_t > 0 \end{cases}$$

That is

$$(\mathbf{Q}\vec{\alpha} - \vec{E})_t \begin{cases} +b \geq 0 & \text{, if } y_t = 1\,;\ \alpha_t < C \\ -b \leq 0 & \text{, if } y_t = -1\,;\ \alpha_t > 0 \\ -b \geq 0 & \text{, if } y_t = -1\,;\ \alpha_t < C \\ +b \leq 0 & \text{, if } y_t = 1\,;\ \alpha_t > 0 \end{cases}$$

- That is

$$\max\Big(\max_{\alpha_t < C, y_t = 1} -\nabla f(\vec{\alpha})_t, \max_{\alpha_t > 0, y_t = -1} \nabla f(\vec{\alpha})_t\Big)$$
$$\leq b \leq \min\Big(\min_{\alpha_t < C, y_t = -1} \nabla f(\vec{\alpha})_t, \min_{\alpha_t > 0, y_t = 1} -\nabla f(\vec{\alpha})_t\Big)$$

This is how b is calculated

- $\vec{\alpha}$ not an optimal solution yet

$$\max\left(\max_{\alpha_t<C, y_t=1} -\nabla f(\vec{\alpha})_t, \max_{\alpha_t>0, y_t=-1} \nabla f(\vec{\alpha})_t\right)$$
$$> \min\left(\min_{\alpha_t<C, y_t=-1} \nabla f(\vec{\alpha})_t, \min_{\alpha_t>0, y_t=1} -\nabla f(\vec{\alpha})_t\right)$$

- Working set $\{i, j\}$

$$i \equiv \text{argmax}(\{-\nabla f(\vec{\alpha})_t \mid y_t = 1, \alpha_t < C\}, \{\nabla f(\vec{\alpha})_t \mid y_t = -1, \alpha_t > 0\}),$$
$$j \equiv \text{argmin}(\{\nabla f(\vec{\alpha})_t \mid y_t = -1, \alpha_t < C\}, \{-\nabla f(\vec{\alpha})_t \mid y_t = 1, \alpha_t > 0\})$$

- The sub-problem

$$\min_{\alpha_i,\alpha_j} \quad \frac{1}{2}\begin{bmatrix}\alpha_i & \alpha_j\end{bmatrix}\begin{bmatrix}\mathbf{Q}_{ii} & \mathbf{Q}_{ij} \\ \mathbf{Q}_{ji} & \mathbf{Q}_{jj}\end{bmatrix}\begin{bmatrix}\alpha_i \\ \alpha_j\end{bmatrix} + (\mathbf{Q}_{i,N}\vec{\alpha}_N - 1)\vec{\alpha}_i$$
$$+(\mathbf{Q}_{j,N}\vec{\alpha}_N - 1)\alpha_j$$
$$y_i\alpha_i + y_j\alpha_j = -\vec{y}_N \cdot \vec{\alpha}_N^k,$$
$$0 \leq \alpha_i, \alpha_j \leq C$$

- Substitute

$$\alpha_i = y_i(-\vec{y}_N \cdot \vec{\alpha}_N - y_j\alpha_j)$$

into the objective function ; An one-variable optimization problem

- If without considering $0 \leq \alpha_j \leq C$:

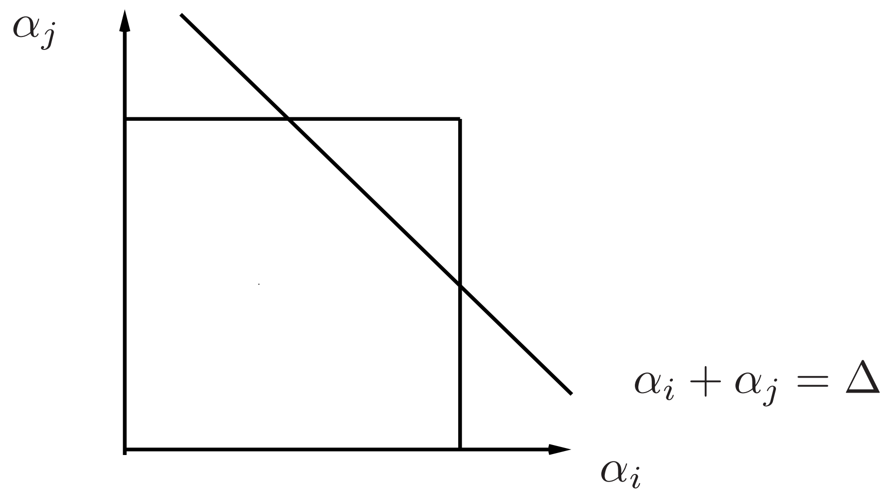
$$\alpha_j^{new} = \begin{cases} \alpha_j + \frac{-G_i - G_j}{Q_{ii}+Q_{jj}+2Q_{ij}} & , \text{if } y_i \neq y_j, \\ \alpha_j + \frac{G_i - G_j}{Q_{ii}+Q_{jj}-2Q_{ij}} & , \text{if } y_i = y_j, \end{cases} \tag{2}$$

where

$$G_i \equiv \nabla f(\alpha)_i \text{ and } G_j \equiv \nabla f(\alpha)_j.$$

- If outside $[0, C]$, clipped into the feasible region ; If $y_i = y_j$ and $C \leq \alpha_i + \alpha_j \leq 2C$,

$$L \equiv \alpha_i + \alpha_j - C \leq \alpha_j^{new} \leq C \equiv H$$



Hence if

$$\alpha_j + \frac{-G_i - G_j}{Q_{ii} + Q_{jj} + 2Q_{ij}} \leq L,$$

$\alpha_j^{new} \equiv L$ and

$$\alpha_i^{new} = \alpha_i + \alpha_j - \alpha_j^{new} = C. \tag{3}$$

- The stopping criteria

$$\max\Big(\max_{\alpha_i<C, y_i=1} -\nabla f(\alpha)_i, \max_{\alpha_i>0, y_i=-1} \nabla f(\alpha)_i\Big)$$
$$\leq \min\Big(\min_{\alpha_i<C, y_i=-1} \nabla f(\alpha)_i, \min_{\alpha_i>0, y_i=1} -\nabla f(\alpha)_i\Big) - \epsilon$$

- Computational Complexity: $O(l)$ in each iteration for finding two indices of the working set
- Implementation tricks: cache for recently used Q_{ij} and others

References

- Chih-Jen Lin: Support Vector Machines - Theory and Practice
- Bernhard Schölkopf, Christopher J.C. Burges, Alexander J. Smola: Advances in Kernel Methods - Support Vector Learning
- Nello Cristianini, John Shawe-Taylor: An Introduction to Support Vector Machines and other kernel-based learning methods