Define a hyperplane

$$\mathbf{w}^T \phi(\mathbf{x}) + b = 0$$

such that

$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b$$

classifies $\mathbf{x}$.

The most important source for all of this today was Stephen Marsland, *Machine Learning: An Algorithmic Perspective*, 2015, pg 179–183. Notation adjusted to agree better with Bishop.

With constraint

$$\forall\ i \in [1, N], \quad t_i \left(\mathbf{w}^T \phi(\mathbf{x}_i) + b\right) \geq 1$$

maximize the margin between the hyperplane and the closest point by finding

$$\begin{aligned} & \text{argmax}_{\mathbf{w},b} \left\{ \frac{1}{||\mathbf{w}||} \min_i \left[t_i \left(\mathbf{w}^T \phi(\mathbf{x}_i) + b\right)\right] \right\} \\ = \quad & \text{argmax}_{\mathbf{w},b} \left\{ \frac{1}{||\mathbf{w}||} \right\} \\ = \quad & \text{argmin}_{\mathbf{w},b} \left\{ \tfrac{1}{2} ||\mathbf{w}|| \right\} \\ = \quad & \text{argmin}_{\mathbf{w},b} \left\{ \tfrac{1}{2} \mathbf{w}^T \mathbf{w} \right\} \end{aligned}$$

This *quadratic programming problem* has an equivalent *Lagrangian* function

$$\mathcal{L}(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}||\mathbf{w}|| - \sum_{i=1}^{N} a_i \left(t_i \left(\mathbf{w}^T \phi(\mathbf{x}_i) + b \right) \right)$$

where **a** is the vector of *Lagrangian multipliers*. This function has the *dual representation*

$$\tilde{\mathcal{L}}(\mathbf{a}) = \sum_{i=1}^{N} a_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} a_i a_j t_i t_j k(x_i, x_j)$$

which we want to maximize subject to constraints

$$0 \leq a_i [\leq C] \qquad \forall\ i \in [1, N]$$

$$\textstyle\sum_{i=1}^{N} a_i t_i = 0$$

where $k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$

Let $\mathbf{K}$ be the *kernel matrix* for data set $\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_N$:

$$\mathbf{K} = \begin{pmatrix} k(\mathbf{x_1}, \mathbf{x_1}) & k(\mathbf{x_2}, \mathbf{x_1}) & \cdots & k(\mathbf{x_N}, \mathbf{x_1}) \\ k(\mathbf{x_1}, \mathbf{x_2}) & k(\mathbf{x_2}, \mathbf{x_2}) & \cdots & k(\mathbf{x_N}, \mathbf{x_2}) \\ \vdots & & & \vdots \\ k(\mathbf{x_1}, \mathbf{x_N}) & k(\mathbf{x_2}, \mathbf{x_N}) & \cdots & k(\mathbf{x_N}, \mathbf{x_N}) \end{pmatrix}$$

Then

$$\tilde{\mathcal{L}}(\mathbf{a}) = \sum_{i=1}^{N} a_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} a_i a_j t_i t_j k(x_i, x_j)$$

becomes

$$\tilde{\mathcal{L}}(\mathbf{a}) = \mathbf{i}^T \mathbf{a} - \frac{1}{2} \mathbf{a}^T \mathbf{t} \mathbf{t}^T \mathbf{K} \mathbf{a}$$

where $\mathbf{i}$ is the identity vector.

A *quadratic programming* problem can be stated as, minimize

$$\frac{1}{2}\mathbf{x}^T\mathbf{P}\mathbf{x} + \mathbf{q}^T\mathbf{x}$$

subject to

$$\begin{aligned} \mathbf{G}\mathbf{x} &\leq \mathbf{h} \\ \mathbf{A}\mathbf{x} &= b \end{aligned}$$

Quadratic programming problem:

$$\min \quad \frac{1}{2}\mathbf{x}^T\mathbf{P}\mathbf{x} + \mathbf{q}^T\mathbf{x} \qquad \begin{aligned} \mathbf{G}\mathbf{x} &\leq \mathbf{h} \\ \mathbf{A}\mathbf{x} &= b \end{aligned}$$

Our problem:

$$\max \quad \mathbf{i}^T\mathbf{a} - \frac{1}{2}\mathbf{a}^T\mathbf{t}\mathbf{t}^T\mathbf{K}\mathbf{a} \qquad \begin{aligned} 0 \leq a_i \quad &[\leq \quad C] \\ \textstyle\sum_{i=1}^{N} a_i t_i \quad &= \quad 0 \end{aligned}$$

We want to find $\mathbf{a}$. Let $\mathbf{P} = \mathbf{t}\mathbf{t}^T\mathbf{K}$, $q = \begin{pmatrix} -1 \\ -1 \\ \vdots \\ -1 \end{pmatrix}$, $\mathbf{A} = \begin{pmatrix} t_1 & 0 & \dots & 0 \\ 0 & t_2 & \dots & 0 \\ \vdots & & & \vdots \\ 0 & 0 & \cdots & t_n \end{pmatrix}$, and $b = 0$.

Quadratic programming problem:

$$\min \quad \tfrac{1}{2}\mathbf{x}^T\mathbf{P}\mathbf{x} + \mathbf{q}^T\mathbf{x} \qquad \begin{aligned} \mathbf{G}\mathbf{x} &\leq \mathbf{h} \\ \mathbf{A}\mathbf{x} &= b \end{aligned}$$

For *hard* margin classification ($0 \leq a_i$),

$$\mathbf{G} = -\mathcal{I} = \begin{pmatrix} -1 & 0 & \dots & 0 \\ 0 & -1 & \dots & 0 \\ \vdots & & & \vdots \\ 0 & 0 & \dots & -1 \end{pmatrix}$$

$$\mathbf{h} = \mathbf{0} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Quadratic programming problem:

$$\min \quad \frac{1}{2}\mathbf{x}^T\mathbf{P}\mathbf{x} + \mathbf{q}^T\mathbf{x} \qquad \begin{aligned} \mathbf{G}\mathbf{x} &\leq \mathbf{h} \\ \mathbf{A}\mathbf{x} &= b \end{aligned}$$

For *soft* margin classification ($0 \leq a_i \leq C$),

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & & & \vdots \\ 0 & 0 & \vdots & 1 \\ -1 & 0 & \dots & 0 \\ 0 & -1 & \dots & 0 \\ \vdots & & & \vdots \\ 0 & 0 & \dots & -1 \end{pmatrix} \qquad \mathbf{h} = \begin{pmatrix} C \\ C \\ \vdots \\ C \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Support vectors are $\{\mathbf{x}_i \mid a_i \neq 0\}$.
Weights are

$$\mathbf{w} = \sum_{i=1}^{N} a_i t_i \mathbf{x}_i = \sum_{i=1 | a_i \neq 0} a_i t_i \mathbf{x}_i$$

Intercept is

$$b = \frac{1}{|\{a_i \mid a_i \neq 0\}|} \sum_{j=1 | a_j \neq 0}^{N} \left(t_j - \sum_{i=1 | a_i \neq 0}^{N} a_i t_i \mathbf{x}_i^T \mathbf{x}_j \right)$$

To train a classifier for hard margin classification:

Given **data**, **targets**, and k,

Compute kernel matrix $\mathbf{K}$

Compute $\mathbf{P} = \mathbf{t}\mathbf{t}^T\mathbf{K}$

Assemble $\mathbf{q}$ vector of -1s

Assemble $\mathbf{A}$ matrix of t_i along the diagonal

Assemble $\mathbf{G}$ matrix of -1s along the diagonal

Assemble $\mathbf{h}$ verctor of 0s

Compute $\mathbf{a}$ vector by feeding $\mathbf{P}$, $\mathbf{q}$, $\mathbf{G}$, $\mathbf{h}$, $\mathbf{A}$, and $\mathbf{b=0}$ into QP solver

Select support vectors from $\mathbf{a}$ that are not zero

Compute b

(For soft margin, modify $\mathbf{G}$ and $\mathbf{h}$.)

To classify new data point $\mathbf{x}$, compute $\sum_{i=1|a_i \neq 0}^{N} a_i t_i k(\mathbf{x}_i, \mathbf{x}) + b$