- Review of formulas
- Using a QP solver
- Training and using a classifier

Let ϕ be a feature space mapping and k be a kernel function, $k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$.

General goal: Define a hyperplane $\mathbf{w}^T\phi(\mathbf{x}) + w_0 = 0$ such that $y(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x}) + w_0$. classifies $\mathbf{x}$.

I have changed the intercept from b (as it appears in the book and appeared in the original version of the handout) to w_0 so that it does not clash with b in the quadratic programming cannonical form below.

With constraint $\forall\ i \in [1, N], \quad t_i\left(\mathbf{w}^T\phi(\mathbf{x}_i) + w_0\right) \geq 1$, maximize the margin between the hyperplane and the closest point by finding

$$\underset{\mathbf{w},w_0}{\operatorname{argmax}}\left\{\frac{1}{||\mathbf{w}||}\min_i\left[t_i\left(\mathbf{w}^T\phi(\mathbf{x}_i) + w_0\right)\right]\right\} = \underset{\mathbf{w},w_0}{\operatorname{argmax}}\left\{\frac{1}{||\mathbf{w}||}\right\} = \underset{\mathbf{w},w_0}{\operatorname{argmin}}\left\{\frac{1}{2}||\mathbf{w}||\right\} = \underset{\mathbf{w},w_0}{\operatorname{argmin}}\left\{\frac{1}{2}\mathbf{w}^T\mathbf{w}\right\}$$

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

$$\mathcal{L}(\mathbf{w}, w_0, \mathbf{a}) = \frac{1}{2}||\mathbf{w}|| - \sum_{i=1}^{N} a_i\left(t_i\left(\mathbf{w}^T\phi(\mathbf{x}_i) + w_0\right)\right)$$

where $\mathbf{a}$ is the vector of *Lagrangian multipliers*.

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

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

The Lagrangian function has the *dual representation*

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

which we want to maximize subject to constraints $0 \leq a_i[\leq C]$, $\forall\ i \in [1, N]$ and $\sum_{i=1}^{N} a_i t_i = 0$. The targets $\mathbf{t}$ make a column vector, and so $\mathbf{t}\mathbf{t}^T$ is a square matrix of the same dimensionality as K. The $\circ$ operator indicates the *Hadamard product*, which is entry-wise multiplication of two matrices with the same dimension. The result is a matrix containing the kernel results each multiplied by the product of the corresponding targets:

$$\mathbf{t}\mathbf{t}^T) \circ \mathbf{K} = \begin{pmatrix} t_1 \cdot t_1 \cdot k(\mathbf{x_1}, \mathbf{x_1}) & t_2 \cdot t_1 \cdot k(\mathbf{x_2}, \mathbf{x_1}) & \cdots & t_N \cdot t_1 \cdot k(\mathbf{x_N}, \mathbf{x_1}) \\ t_1 \cdot t_2 \cdot k(\mathbf{x_1}, \mathbf{x_2}) & t_2 \cdot t_2 \cdot k(\mathbf{x_2}, \mathbf{x_2}) & \cdots & t_N \cdot t_2 \cdot k(\mathbf{x_N}, \mathbf{x_2}) \\ \vdots & & & \vdots \\ t_1 \cdot t_N \cdot k(\mathbf{x_1}, \mathbf{x_N}) & t_2 \cdot t_N \cdot k(\mathbf{x_2}, \mathbf{x_N}) & \cdots & t_N \cdot t_N \cdot k(\mathbf{x_N}, \mathbf{x_N}) \end{pmatrix}$$

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 $\mathbf{G}\mathbf{x} \leq \mathbf{h}$ and $\mathbf{A}\mathbf{x} = \mathbf{b}$.

In the original handout, $\mathbf{b}$ above was b, a scalar, and we vacillated on whether it should be a vector or scalar. In the cannonical form for quadratic programming, it is a vector. Additionally, $\mathbf{x}$ is an $n \times 1$ (column) vector, $\mathbf{P}$ is an $n \times n$ symmetric matrix, $\mathbf{G}$ is an $m \times n$ matrix, $\mathbf{h}$ is a $m \times 1$ (column) vector, $\mathbf{A}$ is a $\ell \times n$ matrix, and $\mathbf{b}$ is a $\ell \times 1$ (column) vector.

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

Notice that *in this problem*, $\mathbf{A}$ is a $1 \times n$ matrix, effectively a row vector, and $\mathbf{b}$ is a 1×1 column vector, effectively a scalar. *This was the source of ambiguity about whether b should be a vector or a scalar.* It is a vector in the general problem, but a scalar (as a degenerate vector) in this specific problem. Thanks to Drew and Haley for tracking this down. Also notice that $\mathbf{t}\mathbf{t}^T$ has a matrix result, but $(\mathbf{t}\mathbf{t}^T) \circ \mathbf{K}$ is is the Hadamard product.

For *hard* margin classification ($0 \leq a_i$), $\mathbf{G} = -\mathcal{I} = \begin{pmatrix} -1 & 0 & \dots & 0 \\ 0 & -1 & \dots & 0 \\ \vdots & & & \vdots \\ 0 & 0 & \dots & -1 \end{pmatrix}$, $\mathbf{h} = \mathbf{0} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$.

For *soft* margin classification ($0 \leq a_i \leq C$), $\mathbf{G} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & & & \vdots \\ 0 & 0 & \vdots & 1 \\ -1 & 0 & \dots & 0 \\ 0 & -1 & \dots & 0 \\ \vdots & & & \vdots \\ 0 & 0 & \dots & -1 \end{pmatrix}$, $\mathbf{h} = \begin{pmatrix} C \\ C \\ \vdots \\ C \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$.

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

$$w_0 = \frac{1}{|\{a_i \mid a_i \neq 0\}|} \sum_{j=1 | a_j \neq 0}^{N} \left(t_j - \sum_{i=1 | a_i \neq 0}^{N} a_i t_i k(\mathbf{x}_i, \mathbf{x}_j) \right)$$

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 w_0

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

To classify new data point $\mathbf{x}$, compute $\sum_{i=1 | a_i \neq 0}^{N} a_i t_i k(\mathbf{x}_i, \mathbf{x}) + w_0$

Next time: Lab work time (Fri); principal component analysis (Mon)

For Monday: Read Ch 12 intro and 12.1-12.2.2 (pg 559–580).