Support vector machines unit:

- Linear programming (last week Wednesday)
- SVM concepts (last week Friday)
- Lab: SVM applications (Monday)
- The math of SVMs (**today**)
- SVM algorithms (Friday)

Today:

- Hard-margin
- Kernelized hard-margin
- Kernelized soft-margin

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

Given training data $\mathbf{X}, \boldsymbol{y}$, where $y_n \in \{-1, +1\}$, find $\boldsymbol{w}$, b, and r, specifically

$$\begin{aligned} &\text{maximize} && r \\ &\text{subject to the constraints} && \forall\ \boldsymbol{x_n}, y_n, \quad y_n(\boldsymbol{w}^T \boldsymbol{x_n} + b) \geq r \\ & && ||\boldsymbol{w}|| = 1 \\ & && r > 0 \end{aligned}$$

Or, equivalently

$$\begin{aligned} &\text{minimize} && \tfrac{1}{2}||w||^2 \\ &\text{subject to the constraints} && \forall\ \boldsymbol{x_n}, y_n, \quad y_n(\boldsymbol{w}^T \boldsymbol{x_n} + b) \geq 1 \end{aligned}$$

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

$$\frac{1}{2}\boldsymbol{x}^T\mathbf{Q}\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x}$$

subject to

$$\mathbf{G}\boldsymbol{x} \leq \boldsymbol{h}$$

$$\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$$

In this formula, let n be the number of variables, m be the number of inequality constraints, and ℓ be the number of equality constraints.

(Deisenroth et al combine the constrains into a single set of inequalities, $\mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}$.)

Define a hyperplane

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

such that

$$f(\boldsymbol{x}) = \boldsymbol{w}^T \phi(\boldsymbol{x}) + b$$

classifies $\boldsymbol{x}$.

ϕ is a feature map that projects the vectors $\boldsymbol{x}$ into higher dimensions. Assume k is a kernel function corresponding to ϕ for efficiently computing dot products in these higher dimensions.

With constraint

$$\forall\ n \in [0, N), \quad y_n \cdot \left(\boldsymbol{w}^T \phi(\boldsymbol{x}_i) + b \right) \geq 1$$

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

$$\begin{aligned} & \text{argmax}_{\boldsymbol{w},b} \left\{ \frac{1}{||\boldsymbol{w}||} \right\} \\ = \ & \text{argmin}_{\boldsymbol{w},b} \left\{ \frac{1}{2} ||\boldsymbol{w}||^2 \right\} \\ = \ & \text{argmin}_{\boldsymbol{w},b} \left\{ \frac{1}{2} \boldsymbol{w}^T \boldsymbol{w} \right\} \end{aligned}$$

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

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}||\boldsymbol{w}||^2 - \sum_{n=0}^{N-1} \alpha_n y_n \left(\boldsymbol{w}^T \phi(\boldsymbol{x}_n) + b \right)$$

where $\boldsymbol{\alpha}$ is the vector of *Lagrangian multipliers*. This function has the *dual representation*

$$\mathcal{D}(\boldsymbol{\alpha}) = \sum_{n=0}^{N-1} \alpha_n - \frac{1}{2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \alpha_i \alpha_j y_i y_j \cdot k(\boldsymbol{x_i}, \boldsymbol{x_j})$$

which we want to maximize subject to constraints

$$0 \leq \alpha_n \; [\leq C], \quad \forall \; n \in [0, N-1)$$

$$\textstyle\sum_{n=0}^{N-1} \alpha_n y_n = 0$$

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

Let $\mathbf{K}$ be the *kernel matrix* for data set $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots \boldsymbol{x}_{N-1}$:

$$\mathbf{K} = \begin{pmatrix} k(\boldsymbol{x_0}, \boldsymbol{x_0}) & k(\boldsymbol{x_1}, \boldsymbol{x_0}) & \cdots & k(\boldsymbol{x_{N-1}}, \boldsymbol{x_0}) \\ k(\boldsymbol{x_0}, \boldsymbol{x_1}) & k(\boldsymbol{x_1}, \boldsymbol{x_1}) & \cdots & k(\boldsymbol{x_{N-1}}, \boldsymbol{x_1}) \\ \vdots & & & \vdots \\ k(\boldsymbol{x_0}, \boldsymbol{x_{N-1}}) & k(\boldsymbol{x_1}, \boldsymbol{x_{N-1}}) & \cdots & k(\boldsymbol{x_{N-1}}, \boldsymbol{x_{N-1}}) \end{pmatrix}$$

Then

$$\mathcal{D}(\boldsymbol{\alpha}) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \cdot k(\boldsymbol{x_i}, \boldsymbol{x_j})$$

becomes

$$\mathcal{D}(\boldsymbol{\alpha}) = \mathbf{1}^T \boldsymbol{\alpha} - \frac{1}{2} \boldsymbol{\alpha}^T (\boldsymbol{y}\boldsymbol{y}^T \circ \mathbf{K}) \boldsymbol{\alpha}$$

where $\mathbf{1} = [1\,1\,1 \cdots 1]^T$ and $\circ$ indicates the Hadamard product..

In the formula

$$\mathcal{D}(\boldsymbol{\alpha}) = \mathbf{1}^T\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^T(\boldsymbol{y}\boldsymbol{y}^T \circ \mathbf{K})\boldsymbol{\alpha}$$

the Hadamard product $\circ$ gives us

$$\boldsymbol{y}\boldsymbol{y}^T \circ \mathbf{K} = \begin{pmatrix} y_1 \cdot y_1 \cdot k(\boldsymbol{x_1}, \boldsymbol{x_1}) & y_2 \cdot y_1 \cdot k(\boldsymbol{x_2}, \boldsymbol{x_1}) & \cdots & y_N \cdot y_1 \cdot k(\boldsymbol{x_N}, \boldsymbol{x_1}) \\ y_1 \cdot y_2 \cdot k(\boldsymbol{x_1}, \boldsymbol{x_2}) & y_2 \cdot y_2 \cdot k(\boldsymbol{x_2}, \boldsymbol{x_2}) & \cdots & y_N \cdot y_2 \cdot k(\boldsymbol{x_N}, \boldsymbol{x_2}) \\ \vdots & & & \vdots \\ y_1 \cdot y_N \cdot k(\boldsymbol{x_1}, \boldsymbol{x_N}) & y_2 \cdot y_N \cdot k(\boldsymbol{x_2}, \boldsymbol{x_N}) & \cdots & y_N \cdot y_N \cdot k(\boldsymbol{x_N}, \boldsymbol{x_N}) \end{pmatrix}$$

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

$$\frac{1}{2}\boldsymbol{x}^T\mathbf{Q}\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x}$$

subject to

$$\mathbf{G}\boldsymbol{x} \leq \boldsymbol{h}$$

$$\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$$

In this formula, let n be the number of variables, m be the number of inequality constraints, and ℓ be the number of equality constraints.

(Deisenroth et al combine the constrains into a single set of inequalities, $\mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}$.)

Quadratic programming problem:

$$\min \quad \tfrac{1}{2}\boldsymbol{x}^T\mathbf{Q}\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x}$$

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

Our problem:

$$\max \quad \mathbf{1}^T\boldsymbol{\alpha} - \tfrac{1}{2}\boldsymbol{\alpha}^T(\boldsymbol{y}\boldsymbol{y}^T \circ \mathbf{K})\boldsymbol{\alpha}$$

$$\begin{aligned} 0 \leq \alpha_i \quad [\leq \quad C] \\ \textstyle\sum_{i=1}^{N} \alpha_i y_i \quad = \quad 0 \end{aligned}$$

We want to find $\boldsymbol{\alpha}$. Let $\mathbf{Q} = \boldsymbol{y}\boldsymbol{y}^T \circ \mathbf{K}$, $c = \begin{pmatrix} -1 \\ -1 \\ \vdots \\ -1 \end{pmatrix}$, $\mathbf{A} = \begin{pmatrix} y_1 & y_2 & \cdots & y_n \end{pmatrix}$, and $\boldsymbol{b} = \begin{pmatrix} 0 \end{pmatrix}$.

Quadratic programming problem:

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

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

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

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

Quadratic programming problem:

$$\min \quad \tfrac{1}{2}\boldsymbol{x}^T\mathbf{Q}\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x}$$

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

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

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

Support vectors are $\{\boldsymbol{x}_i \mid \alpha_i \neq 0\}$.

Weights are

$$\boldsymbol{w} = \sum_{n=1}^{N} \alpha_n y_n \boldsymbol{x}_n = \sum_{n=1 | \alpha_n \neq 0} \alpha_n y_n \boldsymbol{x}_n$$

Intercept is

$$b = \frac{1}{|\{\alpha_n \mid \alpha_n \neq 0\}|} \sum_{j=1 | \alpha_j \neq 0}^{N} \left(y_j - \sum_{i=1 | \alpha_i \neq 0}^{N} \alpha_i y_i \cdot k(\boldsymbol{x}_i, \boldsymbol{x}_j) \right)$$

To train a classifier for hard margin classification:

Given **data X**, **targets** $\boldsymbol{y}$, and **kernel function** k,

- Compute kernel matrix $\mathbf{K}$
- Compute $\mathbf{Q} = \boldsymbol{y}\boldsymbol{y}^T \circ \mathbf{K}$
- Assemble $\mathbf{c}$ vector of -1s
- Assemble $\mathbf{A}$ matrix of y_i along the diagonal
- Assemble $\mathbf{G}$ matrix of -1s along the diagonal
- Assemble $\mathbf{h}$ vector of 0s
- Compute $\boldsymbol{\alpha}$ vector by feeding $\mathbf{Q}$, $\mathbf{c}$, $\mathbf{G}$, $\mathbf{h}$, $\mathbf{A}$, and $\mathbf{b=0}$ into QP solver
- Select support vectors from $\boldsymbol{\alpha}$ that are not zero
- Compute b

(For soft margin, modify $\mathbf{G}$ and $\mathbf{h}$.)
To classify new data point $\boldsymbol{x}$, compute $\sum_{n=1|\alpha_n \neq 0}^{N} a_n y_n k(\boldsymbol{x}_n, \boldsymbol{x}) + b$