# Problem Set 4
## Due on Monday, October 31, 2016 at 11:55 pm

## How to Submit

Create one archive file (.zip, **not** .rar) of your code and submit it via `ilearn.ucr.edu`. Supply your answers to question 1 as `q1.txt` or `q1.pdf`. Supply your code for question 2 as `learnsvm.m` and `q2.m`. You may also include other `.m` files containing functions you wrote that are helper functions to those above.

Do not supply any directories in your zip file. Each file should (in comments) list

- Your name
- Your UCR student ID number
- The date
- The course (CS 229)
- The assignment number (PS 4)

**Q1: Local optimization General SVM**

Recall that the dual formulation for support vector machine training is

$$\begin{aligned} \underset{a}{\text{maximize}} \; & -\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j K_{ij} + \sum_i \alpha_i \\ \text{s.t.} \; & \sum_i \alpha_i y_i = 0 \\ & C \geq \alpha_i \geq 0 \qquad\qquad \forall i \end{aligned}$$

The constraint that $\sum_i \alpha_i y_i = 0$ is problematic. This constraint forbids changing a single $\alpha$ value at a time. However, we can change two $\alpha$ values at the same time while keeping $\sum_i \alpha_i y_i$ a constant. In this problem, we explore this solution method.

**part a.** [2 points]

Consider changing $\alpha_i$ and $\alpha_j$ ($i$ and $j$ will be selected according to a method discussed later). Assume that the old value for $\alpha$ satisfied all of the constraints (it just might not maximize the objective). Therefore, if we keep the sum of $\alpha_i y_i + \alpha_j y_j$ the same before, after changing $\alpha_i$ and $\alpha_j$ the first constraint will still be maintained.

First write down the new value of $\alpha_j$ in terms of the new value of $\alpha_i$ and the old values of both. We will, therefore, search over values of $\alpha_i$ to maximize the objective, setting $\alpha_j$ as a function of $\alpha_i$ (and holding all other $\alpha$s constant).

Next write down the constraints on our choice of $\alpha_i$ to make sure that $0 \leq \alpha_i \leq C$ and $0 \leq \alpha_j \leq C$.

**part b.** [2 points] We now have a one-dimensional optimization problem (over $\alpha_i$) subject to the constraints you derived above. Our objective is to maximize

$$-\frac{1}{2}\sum_{i,j} a_i a_j y_i y_j K_{ij} + \sum_i a_i$$

For ease of notation, let $G$ be a matrix such that $G_{ij} = K_{ij} y_i y_j$. Then, the objective is

$$-\frac{1}{2}\alpha^\top G \alpha + \alpha^\top 1$$

where $\alpha$ is the vector of all $\alpha$s and 1 is the vector of 1s.

Given that all $\alpha_l$, for $l \neq i, l \neq j$, will be held fixed and that $\alpha_j$ is a function of $\alpha_i$ (as you derived above), the above expression is a function of the single scalar $\alpha_i$.

Assuming $\alpha_i$ were unconstrained, derive the formula for the extremum (maximum or minimum) of the objective with respect to $\alpha_i$.

You may find it helpful to divide $G$ and $\alpha$ into the elements corresponding indices $i$ and $j$ and the other indices. For instance, if $i = 1$ and $j = 2$, we could divide $G$ and $\alpha$ in the following way.

$$\alpha = \left[\begin{array}{c} a \\ \hline \tilde{a} \end{array}\right]$$

$$G = \left[\begin{array}{c|c} H & q^\top \\ \hline q & \tilde{H} \end{array}\right]$$

In this way, $a$ is a $2 \times 1$ vector of just the elements that change in $\alpha$ and $\tilde{a}$ is a $m-2 \times 1$ vector of all the other elements of $\alpha$. Similarly, $H$ is a $2 \times 2$ matrix of the elements in $G$ corresponding to rows and columns each from the set $\{i, j\}$. $q$ is a $m-2 \times 2$ matrix of the elements in which the row is not from this this set, but the column is. And, $\tilde{H}$ is a $m-2 \times m-2$ matrix of elements from $G$ for which neither the row nor the column is from this set.

**part c.** [2 points] Given the constraints from part a and the formula for the extremum from part b, write down pseudo-code for finding the optimal value for the pair $\alpha_i$ and $\alpha_j$, holding all other $\alpha$ values fixed.

**Q2: 2D SVM Results**

**part a.** [2 points]

Selecting any two $i$ and $j$ and executing the algorithm you derived above will improve (or at least not make worse) the objective, while maintaining the constraints. Applying this to random pairs of $\alpha$s can work fine. However, there are some standard heuristics for picking which $i$ and $j$ to use that make the algorithm faster. The supplied code `learnsvm.m` already have these heuristics implemented. All that is missing is the function `update`. Edit `learnsvm.m` to complete this algorithm.

**part b.** [2 points]

The supplied function `explot` demonstrates how to use the supplied function `plotclassifier` to plot both the training data and the resulting classification surface. Use `plotclassifier` to show the results of running your SVM code from above on two supplied 2D datasets: `example1.data` and `example2.data`. In each file, the first two columns are the $X$ values and the last is the $Y$ value.

In particular, write a function `q2` (that takes no arguments) that produces 4 figures, each with subplots (see the `subplot` command) as follows.

1. Using `example1.data`, plot 9 subplots (in a 3-by-3 grid). The top row is with the linear kernel. The second row is with a polynomial kernel with $c = 1$ and $d = 2$. The third row is with a polynomial kernel with $c = 1$ and $d = 5$.

   The first column is for $C = 0.1$. The second column is for $C = 1$. The third column is for $C = 10$.

2. Using `example1.data`, plot 9 subplots (in a 3-by-3 grid). All plots are with the "Gaussian" kernel:

   $$K(x, x') = e^{-\|x-x'\|^2/(2\sigma^2)}$$

   The top row is with $\sigma = 1$. The second row is with $\sigma = 5$. The third row is with $\sigma = 10$. Each column has a different $C$ value, the same as figure 1.

3. Same as figure 1, but with `example2.data`

4. Same as figure 2, but with `example2.data`

Label (using `title`) each subplot appropriately.