Assignment 05

In this assignment, you implement SVM with the German Credit Card practice project described at https://www.scriptedin.com/contests/view/20, The data sets can be downloaded, and are provided on HuskyCT. A template notebook is provided to you, also posted at. https://www.scriptedin.com/notebooks/view/196. For simplicity, no modeling tuning is needed.

The primal QP optimization problem for a non-separable case

$$\min_{w} \frac{1}{2} w^T w + C \sum_{i=1}^{N} \theta^{(i)}$$

Subject to

$$y^{(i)}\left(w^T x^{(i)} + b\right) - 1 + \theta^{(i)} \geq 0 \quad \text{and } \theta_i \geq 0 \ \forall \text{i}$$

And the corresponding dual optimization problem

$$\max_{\alpha} \left[\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{i=1}^{N} \alpha_i \alpha_j y^{(i)} y^{(j)} {x^{(i)}}^T x^{(j)} \right] =$$

Subject to

$$C \geq \alpha_i \geq 0, \ \ i = 1 \ldots N$$

$$\sum_{i=1}^{N} \alpha_i y^{(i)} = 0$$

C is set to a large number to penalize more to the points falling between the margins.

In the vectorized form:

$$\underset{\alpha}{arg\max} [\mathbf{1}^T * \boldsymbol{\alpha} - 0.5 \boldsymbol{\alpha}^T * diag(\boldsymbol{y}) * \boldsymbol{X} * \boldsymbol{X}^T * diag(\boldsymbol{y}) * \boldsymbol{\alpha}]$$

Subject to

$$\boldsymbol{I\alpha} \geq \mathbf{0}$$

$$\boldsymbol{\alpha}^T \boldsymbol{y} = 0$$

In which **1** is a column vector of all values 1, $\boldsymbol{\alpha}$ is a column vector of α_i, **y** is the column vector of the target variable in the training data set, and **X** is the training data set with all the variables on its columns and observations on its rows. **I** is the identity matrix.

The term

$$\boldsymbol{X} * \boldsymbol{X}^T$$

Can be denoted **K**, which is the linear kernel, so linear kernel also means no kernel. The above becomes

$$\underset{\alpha}{argmax}[\mathbf{1}^T * \boldsymbol{\alpha} - 0.5\boldsymbol{\alpha}^T * diag(\boldsymbol{y}) * \boldsymbol{K} * diag(\boldsymbol{y}) * \boldsymbol{\alpha}]$$

Subject to

$$\boldsymbol{I\alpha} \geq \mathbf{0}$$

$$\boldsymbol{\alpha}^T \boldsymbol{y} = 0$$

Other kernels include a polynomial kernel

$$\mathrm{K}\left(x^{(i)}, x^{(j)}\right) = \left(x^{(i)^T} x^{(j)} + 1\right)^d$$

Or a Gaussian kernel

$$\mathrm{K}\left(x^{(i)}, x^{(j)}\right) = exp\left(\frac{-\|x - x_k\|^2}{2\sigma^2}\right)$$

After getting $\boldsymbol{\alpha}$ from the optimization function, only $\alpha_i > 0$ is kept. $x^{(i)}$ corresponding to nonzero α_i are called support vectors.

A new observation x can be classified as follows:

$$g(x) = \sum_{j=1}^{N} \alpha_j y_j K\left(x, x^{(j)}\right) + b$$

b is calculated as follows. As any support vector satisfies $y_i g(x_i)=1$, so

$$y_i \left(\sum_{j=1}^{N_s} \alpha_j y_j K\left(x^{(i)}, x^{(j)}\right) + b \right) = 1$$

Knowing that $y_i^2=1$, multiply both sides with y_i

$$\sum_{j=1}^{N_s} \alpha_j y_j K\left(x^{(i)}, x^{(j)}\right) + b = y_i$$

$$b = y_i - \sum_{j=1}^{N_s} \alpha_j y_j K\left(x^{(i)}, x^{(j)}\right)$$

A stable solution is to average b over all support vectors:

$$b = \frac{1}{N_s}\left(\sum_{i=1}^{N_s}\left(y_i - \sum_{j=1}^{N_s} \alpha_i y_i K\left(x^{(i)}, x^{(j)}\right)\right)\right)$$

Note how to set up the quadratic programming function solvers.qp to solve the optimization problem (see the attached pdf for instructions & examples).

1. (3 points). Write a function namely “kernel”, which returns a linear, Gaussian, or polynomial kernel

2. (3 points). Write a function namely “intercept” to calculate the intercept b

Note that the average is by the number of support vectors N_s, not the number of instances N.

3. (2 points). Write a function namely “predict” to classify a vector x to the class of y=-1 or class of y=1 (y is the class label)

$$g(x) = \sum_{j=1}^{N} \alpha_j y_j K\left(x, x^{(j)}\right) + b$$

where

$$b = \frac{1}{N_s}\left(\sum_{i=1}^{N_s}\left(y_i - \sum_{j=1}^{N_s} \alpha_i y_i K\left(x^{(i)}, x^{(j)}\right)\right)\right)$$

The function computes the kernel (of which the type is up to your setting), then the sum for all the training instances, then b, then plugs them in g(x). Assign y = -1 if g(x) < 0, and vice versa y = 1.

4. (2 points). Write additional code to compute and display the confusion matrix, precision, and recall.

Submission on Husky includes a zip file of:

a. A notebook named as “group_xxx_assignment_yyy.ipynb”
b. A Word document/article with detailed explanation
c. All the other files

Also archive the notebook and the article (only after the deadline) to the project site.