CS503– Machine Learning
Lab 3

Due on 29/3/2019 11.55pm

Instructions: Upload to your moodle account one zip file containing the following. Please do not submit hardcopy of your solutions. In case moodle is not accessible email the zip file to the instructor at ckn@iitrpr.ac.in. Late submission is not allowed without prior approval of the instructor. You are expected to follow the honor code of the course while doing this homework.

1. **This lab has to be completed individually. I and the TAs reserve the right to question you if required. If you are found unable to explain the code, you will receive no points for the lab.**

2. A neatly formatted PDF document with your answers for each of the questions in the homework. You can use latex, MS word or any other software to create the PDF.

3. Include a separate folder named as 'code' containing the scripts for the homework along with the necessary data files. Ensure the code is documented properly.

4. Include a README file explaining how to execute the scripts.

5. Name the ZIP file using the following convention rollnumberhwnumber.zip

---

In this lab you will be experimenting with support vector machines.

a. This is an ungraded Math warm up. Derive the dual formulation of the SVM for the non-separable case that uses Kernel function. The resulting quadratic optimization problem can be stated as

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(\mathrm{x_i}, \mathrm{x_j})$$

such that

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

$$\sum_{i=1}^{N} \alpha_i y_i = 0$$

where C is the box constraint for soft margin and K is the kernel function. As with the linear separable case, w can be expressed as

$$w = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{\phi}(\mathrm{x_i})$$

where ϕ is the feature transformation function that transforms the input $\mathrm{x_i}$into a high dimensional data point. The kernel function computes the inner product between the transformed data points in the high dimensional space. It can also be noted that

$$\alpha_i = 0 \Rightarrow y_i(w^T \boldsymbol{\phi}(\mathrm{x_i}) + w_0) > 1$$
$$\alpha_i = C \Rightarrow y_i(w^T \boldsymbol{\phi}(\mathrm{x}_i) + w_0) < 1$$
$$0 < \alpha_i < C \Rightarrow y_i(w^T \boldsymbol{\phi}(\mathrm{x_i}) + w_0) = 1$$

Support vectors correspond to the data points that satisfy the above equations. These data points can be used to determine the intercept w_0

b. We will be using CVXOPT package to execute the quadratic program solver. You can read more about this solver from this link. To solve the SVM dual objective, we have to rewrite the objective function in the form that is accepted by the solver. Your first step would to identify the different parameters of the solver - q, P, G, h, A, and b. Define how to compute these parameters from the expression in the SVM objective. This will help you to implement the fit function in the accompanying python script.
c. Once you have completed step b, you are ready to implement the SVM function. The first step is to implement the following functions in the accompanying script
    - lin_separable_data(): creates 100 instances of the 2-D positive and negative class that is linearly separable.
    - circular_data(): creates 100 instances of 2-D positive and negative examples that separable using a circle.
    - lin_separable_overlap_data(): creates 100 instances of 2-D positive and negative examples that have overlap between the classes.
d. Implement the function split_train_test(X1, y1, X2, y2): The function splits the data into train and test sets. X1, y1 are the instances and labels of class 1 and X2, y2 are the instances and labels of class 2. Approximately use 75% of the data for training and use the rest for testing.
e. The next step is to implement the different kernel functions.
    - linear_kernel(x1, x2): returns the linear kernel output for the points x1 and x2
    - polynomial_kernel(x1, x2, q): returns the value of the polynomial kernel of degree q for the points x1 and x2
    - gaussian_kernel(x1, x2, s): returns the value of the Gaussian kernel with width s for the points x1 and x2
f. Now we are ready to implement the function to train the SVM model. This needs to be implemented in the function named fit (self, X, y). The input to the function is the training data and label. The first step is to compute the Kernel matrix (Gram matrix). This matrix will have kernel function value between every pair of points. The next step is to compute

the parameters P, q, G, h, A, and b that will be sent as inputs to the solver. Use the expressions derived in part b to write the code. After you have have computed the values of these parameters, uncomment the line - #solution = cvxopt.solvers.qp(…). The solver will return the solution. Extract the values of the Lagrange multipliers out of it. (Note that internally the solver uses the symbol x for its variables. This will correspond to the alpha of the SVM objective function). This part is already implemented for you. Identify the support vectors by applying a threshold on the values of alpha. Due to numerical approximations and floating-point precision, it is not necessary that the alpha values of all the non-support vectors will be exactly 0. Once you have identified the support vectors compute the parameters of the line (hyperplane) using the expression for w. We will do this only for when dealing with linear kernel. The last step would be to compute the intercept term w_0.

g. Given an SVM model we now have to write the function for predicting the label for a test instance. You will implement this in the function predict(self, X). The function should return both the class label and the output of the SVM function for the instances in X.
h. Implement the functions
    - linear_svm()
    - kernel_svm()
    - soft_svm()

    The details of these functions are presented in the accompanying python script. Ignore point 7 for all the functions.
i. With this we should be able to uncomment the last three lines and execute the code to train and test the SVM models.
j. However, we are only computing the accuracy. We would like to visualize the separating line and the support vectors. Implement the plot_margin(X1_train, X2_train, model) function. Specifically, plot the three important lines (w^Tx+w_0 = 1, w^Tx+w_0 = -1, and w^Tx+w_0 = 0, lines) along with the training data. Color code the training data to indicate the positive and negative classes. Highlight the support vectors. This function will be useful for visualizing the output of the linearly dataset (linear separable and with overlap). For visualizing the contours of the kernel SVM implement the plot_contour(X1_train, X2_train, model) function. First create a meshgrid for a specific range. The number of points in the grid will depend upon the granularity of the grid. For each point in the grid compute the output of the SVM. Use the output of the SVM to plot the contours.
k. Call the plotting functions in the linear_svm(), kernel_svm() and soft_svm() functions to visualize the output of the SVM model.
l. Include in the report the plots for different datasets, and different choices of the kernel function and kernel function hyper parameters.
m. [Bonus question 20 points] Use this SVM implementation on the IRIS dataset [1]. Try different kernels and parameter settings. Document the results obtained.

An important aspect of machine learning is reproducibility of the results presented in a paper/report. Therefore, we will run your code to see if the results are closely matching with what you have presented in the report. Any deviation beyond a reasonable threshold will be considered as fudging of results and will invite severe penalty.

Reference

[1] https://archive.ics.uci.edu/ml/datasets/iris