INDIAN INSTITUTE OF SPACE SCIENCE AND TECHNOLOGY
THIRUVANANTHAPURAM

Assignment #2

Due on 27-03-2015

Akhil P M (SC14M044)

Assignment 2

1. Consider w$\in \mathbb{R}^n$. Let $f(x) = \langle w, x\rangle \forall x \in \mathbb{R}^n$. Find $||f||$.

We have

$$\begin{aligned} f(x) &= \langle w, x\rangle \quad \forall x \in \mathbb{R}^n \\ ||f(x)|| &= ||\langle w, x\rangle|| \\ &\leq ||w|| * ||x|| \\ \implies \quad sup\frac{||f(x)||}{||x||} &\leq ||w|| \\ \implies \quad ||f|| \leq ||w|| & \qquad (1) \end{aligned}$$

By the definition of norms. But we have

$$\begin{aligned} f(w) = \langle w, w\rangle \quad &= ||w||^2 \\ \implies \quad \frac{||f(w)||}{||w||} &= ||w|| \\ \leq sup\frac{||f(w)||}{||w||} &= ||f|| \\ \implies ||w|| \leq ||f|| & \qquad (2) \end{aligned}$$

combining (1) and (2) we will get $||f|| = ||w||$.

2. Let the data $\{(x_i, y_i$), i = 1,2,...,N}, $x_i \in \mathbb{R}^N, y_i \in \mathbb{R}$ be generated by a hyperplane. By kernel theory, the function that generates the data can be written as a linear combination of N training points and a bias term. However if the data is n dimensional the equation of the hyper plane consists of n+1 terms. Are these the same? Justify your answer.

No. In kernel methods we are getting a hyperplane in feature space which is linear. But the equation of the hyperplane consisting of n+1(n is the number of attributes) terms is obtained in the input space itself. In case of classification, the data will be always separable in feature space by that hyperplane, but in input space they may be non-separable with a hyperplane. Moreover, the representation of these hyperplanes are also different in terms of coefficients, number of terms in the equation etc. Hence it can be concluded that the two hyperplanes are different.

3. Consider the following data {[(1,1),-1], [(1,-1),1], [(-1,-1),-1], [(-1,1),1]}
(a) Plot the nonlinear boundary in the input space.
(b) Plot the linear boundary in RKHS space generated by the kernel $k(x, y) = \langle x, y\rangle^2$ and the representor of evaluation at the input points.

This is called XOR problem, a simplest linearly non-separable data set. In the input space the data is non-separable with a line. So we need minimum two lines as shown in fig.1

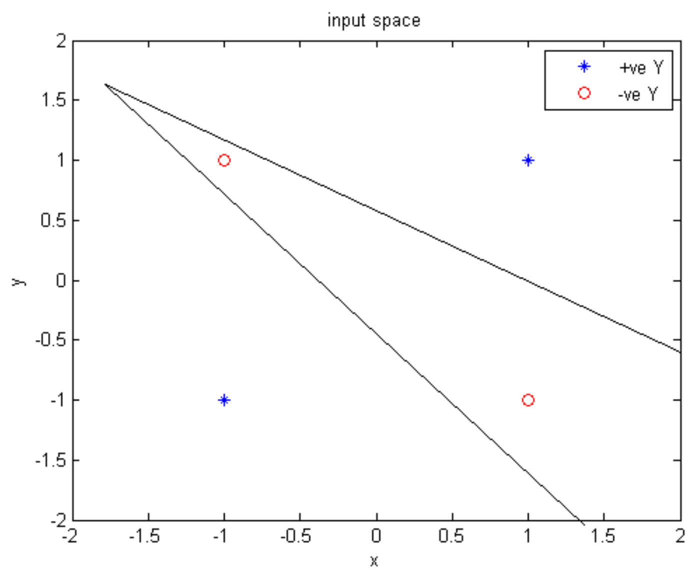

Figure 1: Points in input space

We will transform each point (x,y) in input space to the given feature space as $(x^2, y^2, \sqrt{2}xy)$. Then they become linearly separable.

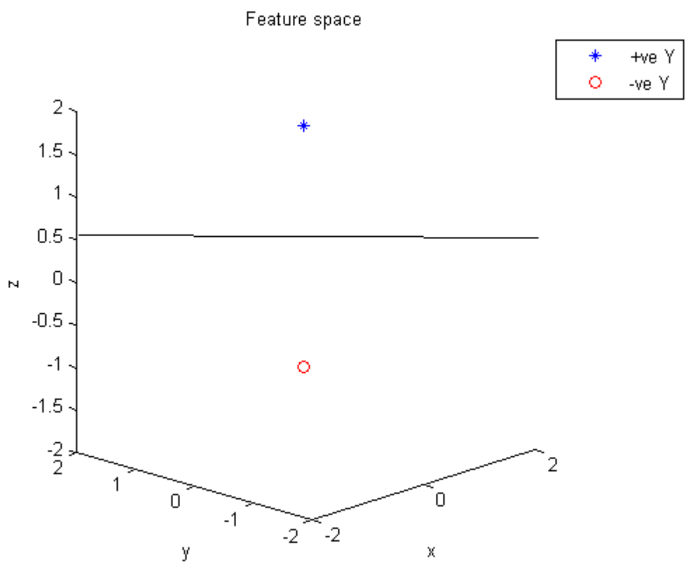

Figure 2: Points in feature space

4. Apply SVM classification on Data 1 and regression on Data 2 (find the attached documents).

(a) Apply direct method and an iterative technique to solve the problem.
(b) Find suitable kernel using cross validation techniques.
(c) Plot the decision boundary for classification and the SVM points.
(d) Plot the function that generates the data for the regression.

(e) Plot the value of primal and dual objective function against iteration.
(f) Assess the performance of the model.
Average accuracy is high for gaussian kernel compared to other kernels from the cross validation results.

Kernel Type	Avg. Accuracy	Avg. F-measure
Gaussian	.718	.687
Linear	.67	.644
Polynomial(d=7)	.707	.649

The decision boundary obtained with iterative methods is shown below.

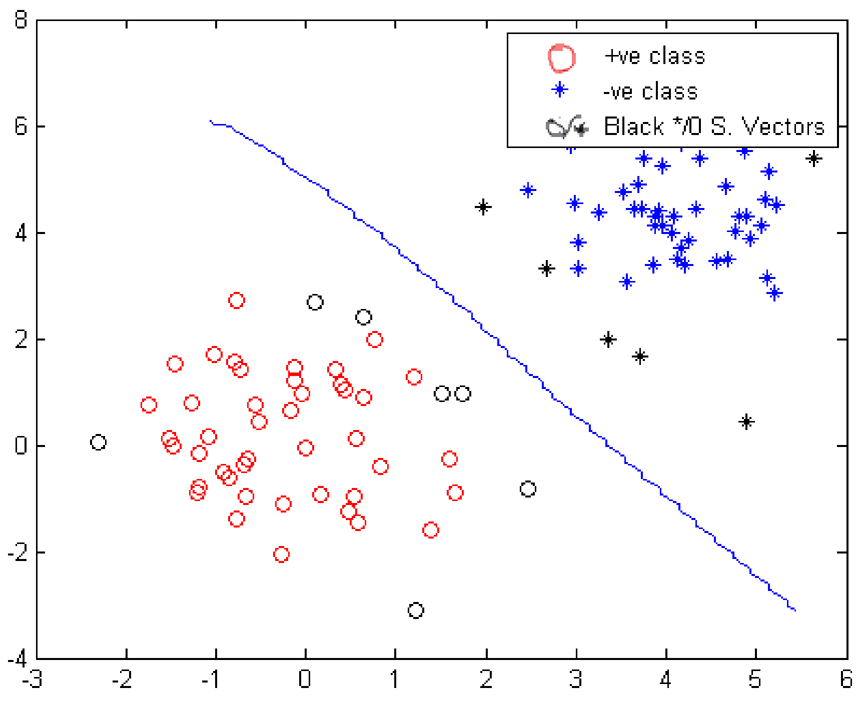

Figure 3: Decision Boundary for Classification

The plot of primal objective v/s dual objective is shown below

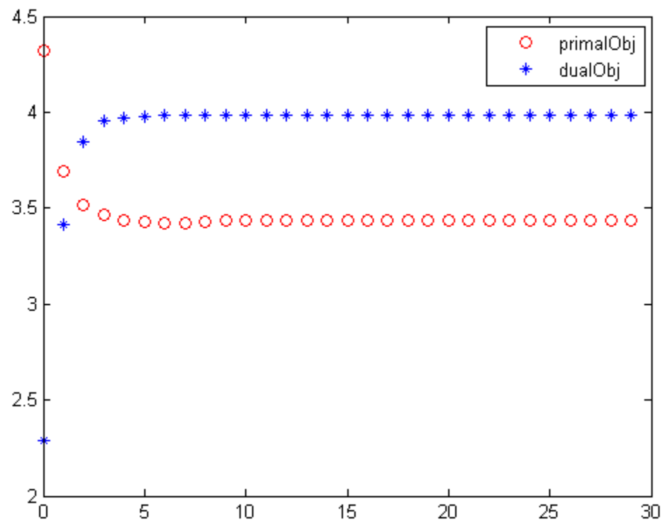

Figure 4: Primal v/s Dual objective Plot

Classification gives full accuracy with gaussian kernel.
no of misclassifications :0
accuracy :1.000
precision :1.000
recall/sensitivity :1.000
F-Measure :1.000

Optimal Parameter Values
C=.7
gama=.09
No of support points=13
The function that generates the regression is approximated as shown in the following fig.5
No of support vectors:174
no of iterations:5000
Average RMSE:0.4486
Polynomial kernel is giving more accuracy for regression(d=3)

5. Discuss the scalability of kernel methods

A bottleneck in scaling up kernel methods comes from the storage and computation cost of the dense kernel matrix, K. Storing the matrix requires O(n^2) space, and computing it takes O($n^2 d$) operations, where n is the number of data points and d is the dimension.
A common numerical linear algebra approach is to approximate the kernel matrix using low-rank factorizations, K $\approx A^T A$, with A $\in R^{r*n}$ and rank $r \leq n$. This low-rank approximation allows subsequent kernel algorithms to directly operate on A, but computing the approximation requires $O(nr^2 + nrd)$ operations.Thus, in order for kernel methods to achieve the best generalization ability, low-rank approximation based approaches immediately become impractical for big datasets because of their O($n^3 + n^2 d$) preprocessing time and O(n^2)storage[1].
Random feature approximation is another popular approach for scaling up kernel methods. The method directly approximates the kernel function instead of the kernel matrix using explicit feature maps. The advantage of this

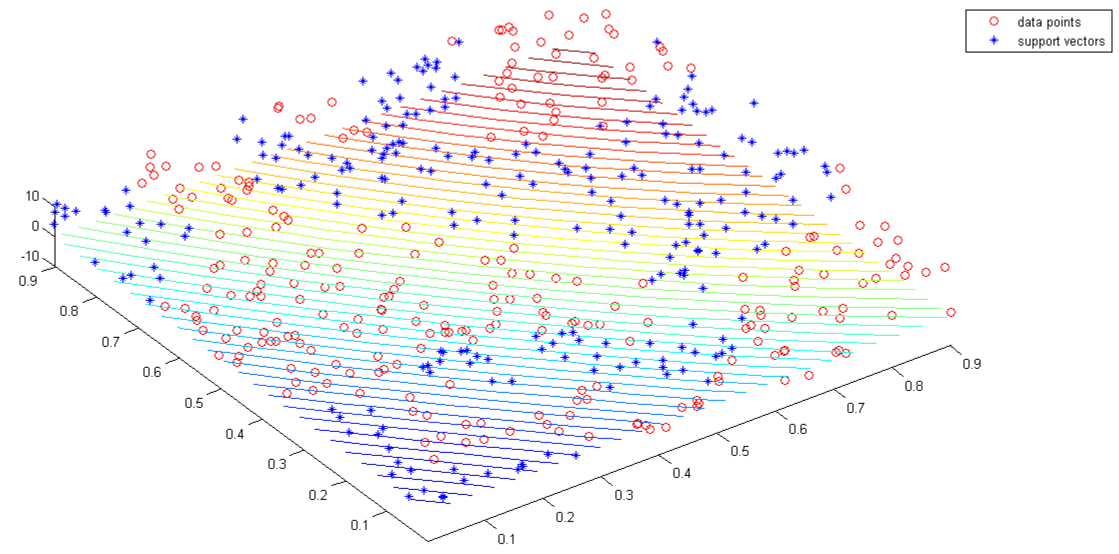

Figure 5: Regression Plot

approach is that the random feature matrix for n data points can be computed in time $O(nrd)$ using $O(nr)$ storage, where r is the number of random features. Subsequent algorithms then only need to operate on an $O(nr)$ matrix. Similar to low-rank kernel matrix approximation approach, the generalization ability of this approach is of the order O($\frac{1}{\sqrt{r}} + \frac{1}{\sqrt{n}}$), which implies that the number of random features also needs to be O(n)[1].
Another approach that addresses the scalability issue rises from the optimization perspective. One general strategy is to solve the dual forms of kernel methods using the block-coordinate descent.Each iteration of this algorithm only incurs O(nrd) computation and $O(nr)$ storage, where r is the block size. A second strategy is to perform functional gradient descent based on a batch of data points at each epoch. Thus, the computation and storage in each iteration required are also $O(nrd)$ and $O(nr)$, respectively. A serious drawback of these approaches is that, without further approximation, all support vectors need to be stored for testing, which can be as big as the entire training set[1].
Doubly stochastic functional gradient is a recent approach proposed by Anant Raj et.al in 2014. This method relies on the fact that most kernel methods can be expressed as convex optimization problems over functions in the reproducing kernel Hilbert spaces (RKHS) and solved via functional gradient descent.
Given a positive definite kernel k(x,x') and the associated reproducing kernel Hilbert space $\mathscr{H}$, the goal is to optimise a function $f^* \in \mathscr{H}$ by minimising the expected loss with a regularisation term

$$f^* = \arg\min_{f \in \mathscr{H}} \mathbb{E}_{(x,y)}[l(f(x), y)] + \frac{\lambda}{2}||f||^2_{\mathscr{H}}$$

Given a datapoint, the stochastic functional gradient w.r.t. f $\in \mathscr{H}$ is defined by

$$l'(f(x), y)k(x,.) + \lambda f(.),$$

for a positive kernel the doubly stochastic functional gradient is given by

$$l'(f(x), y)\phi_\omega(x)\phi_\omega(.) + \lambda f(.)$$

Updating f via the doubly stochastic gradient,

$$f_i = f_{i-1} - \gamma_i [l'(f(x), y)\phi_\omega(x)\phi_\omega(.) + \lambda f(.)],$$

where γ_i is the step-size. The representer theorem says we can write down f_i as a function of the following form

$$f_i(.) = \sum_{j=1}^{i} \alpha_j \phi_{\omega_j}(.)$$

where $\alpha_i = -\gamma_i[l'(f_{i-1}(x_i), y_i)\phi_{\omega_i}(x_i)]$ and $\alpha_j = (1-\gamma_i\lambda)\alpha_j$ for $j = 1, \cdots, i-1$. In this method is that we do not need to save ω_i for each datapoint. Rather, we use a seed (i) from a pseudo-random number generator for training α_i, which we use for testing. So, the only thing one needs to store is the pseudo-random seed. Thus the memory requirement is just $O(n)$. The authors claim that this method finds the optimal function with rate $O(\frac{1}{t})$ and achieves a generalisation bound of $O(\frac{1}{\sqrt{t}})$.

References

[1] Bo Dai, Bo Xie, Niao He, Yingyu Liang, Anant Raj, Maria-Florina Balcan, Le Song, "Scalable Kernel Methods via Doubly Stochastic Gradients ", *Advances in Neural Information Processing Systems* 27, 3041-3049, 2014.