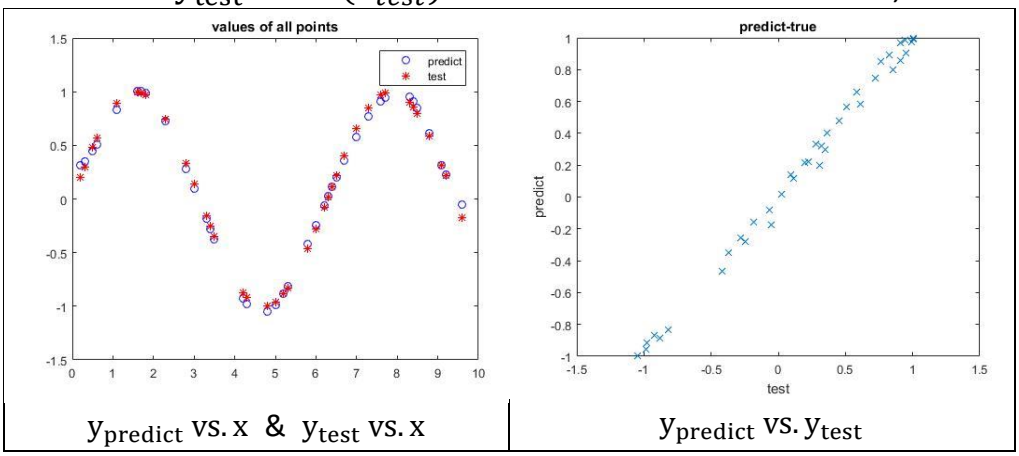
EE6550 Machine Learning, Spring 2016

Homework Assignment #5 Report

102061210 王尊玄

1. First, I tried predicting sine function to test ability of my SVR SMO algorithm. Given a set of number x range from 0 to 10 and their corresponding true value $\sin(x)$, SVR is trained. We further predict $y_{predict}$ another set of number x_{test} range from 0 to 10 and their true value $y_{test} = \sin(x_{test})$. Results are shown as follows,



$y_{predict}$ vs. x & y_{test} vs. x | $y_{predict}$ vs. y_{test}

It is obvious that my SVR works pretty well in predicting sine function. Also, by doing so, I make sure that my SVR may be correct.

2. Yacht: n-cross validation is used to find free parameter of SVR, (C,sigma) for Gaussian kernel and (C,d) for polynomial kernel. Cross validation error is defined as

$$\hat{R}_{CV} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{m_i}\sum_{j=1}^{m_i}\max(|y_{predict} - y_{test}| - eps, 0)$$

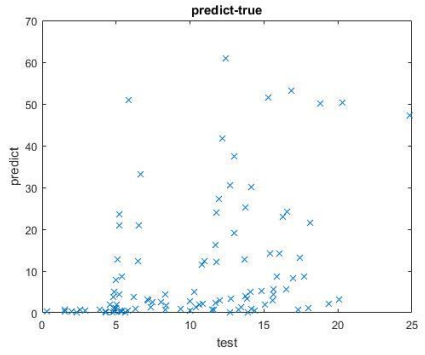
Results are shown as follows,

5-fold cross validation error using gaussian kernel					
sig\C	1	4	10	20	50
0.2	61.79328	68.59537	70.1968	70.1968	70.1968
0.5	61.9516	79.34429	77.59842	84.94468	88.68653
1.2	55.86639	69.17629	73.18767	75.59975	83.86079

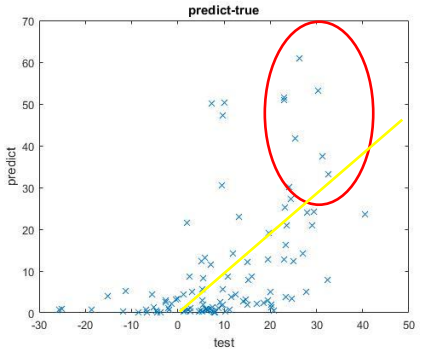
5-fold cross validatio error using polynomial kernel					
d\C	1	4	10	20	50
1	80.29071	83.42103	92.18268	83.1207	101.2454
2	94.18171	94.18171	94.18171	94.18171	94.18171
3	77.54331	77.54331	77.54331	77.54331	77.54331
4	150.728	150.728	150.728	150.728	150.728
5	90.24065	90.24065	90.24065	90.24065	90.24065

10-fold cross validation error using gaussian kernel					
sig\C	1	4	10	20	50
0.2	127.8059	116.4323	120.1016	120.1016	120.1016
0.5	130.4518	144.4981	137.2327	132.6587	132.6587
1.2	124.0776	127.177	159.5717	1668.182	160.2189

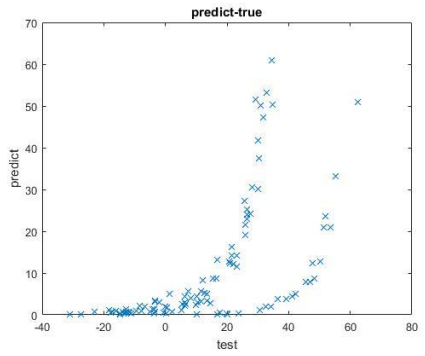
10-fold cross validation error using polynomial kernel					
d\C	1	4	10	20	50
1	115.803	140.9176	172.845	228.0354	222.7209
2	201.728	168.8997	168.8997	168.8997	168.8997
3	168.9428	168.9428	168.9428	168.9428	168.9428
4	236.6358	236.6358	236.6358	236.6358	236.6358
5	186.5039	186.5039	186.5039	186.5039	186.5039



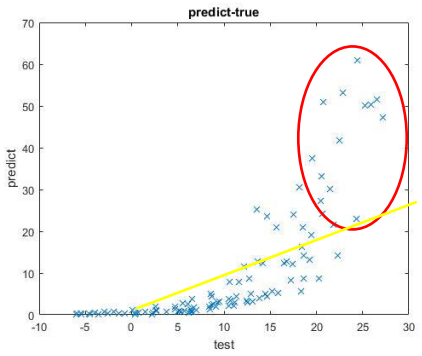
Gaussian (C,sigma)=(1,1.2)



Gaussian (C,sigma)=(4,0.2)



Polynomial (C,d)=(1,3)



Polynomial (C,d)=(1,1)

Performance is pretty bad on yacht set. It seems like Gaussian

kernel (C,sigma)=(4,0.2) and polynomial (C,d)=(1,1) have better performance. Also, we can find that higher value introduce more error. It can be seen from points in red circle, where in the same graph, yellow line specifies y=x, the correct prediction.

3. Normalization: Since the true value y or $\mathrm{y}_{\mathrm{test}}$ varies in a range with unbalanced distribution, I normalize y first before training and after prediction, I denormalize $\mathrm{y}_{\mathrm{predict}}$ and compare to $\mathrm{y}_{\mathrm{test}}$.

   Normalization is carried out as min-max scaling and standardization scaling. The result seems like min-max scaling works better, so in my code, min-max scaling is applied as parameter 'normOrNot' is set. However, standardization scaling can be still be found in my code 'normalization.m' and 'denormalization.m'.