Homework #4

Assign date: 2015-07-24

Due date: 2015-08-02, 6pm

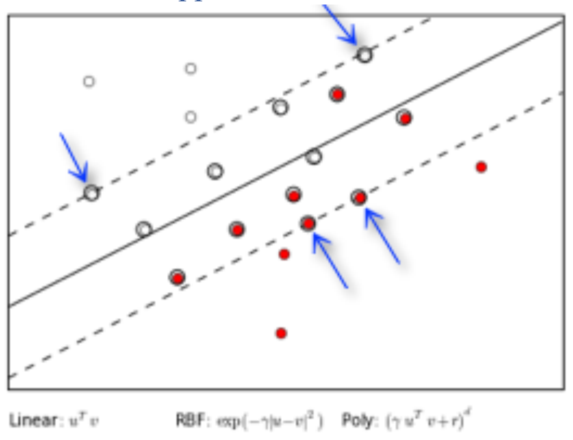
Submission:

1. **Please submit your results in email to the grader: 130301039@svuca.edu Chi Zhang**
2. **Please separate the written answers from the python code: you should submit 1 file in your email – cs596-29-hw4_yourID#.doc**
3. **30 pts per day will be deducted for late submission**

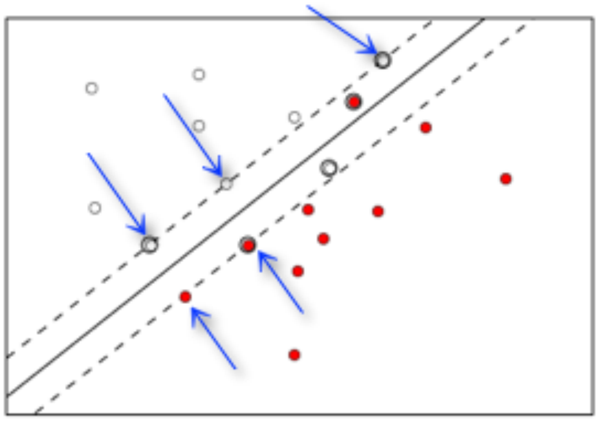
Problem 1) SVM classifier (15 pts.)

2 linear SVC, SVC-1 & SVC-2, are constructed with C=0.001 and C=1 to classify the data points. The results from svm-gui.py are shown in Figures 1 and 2. Answer the following questions.

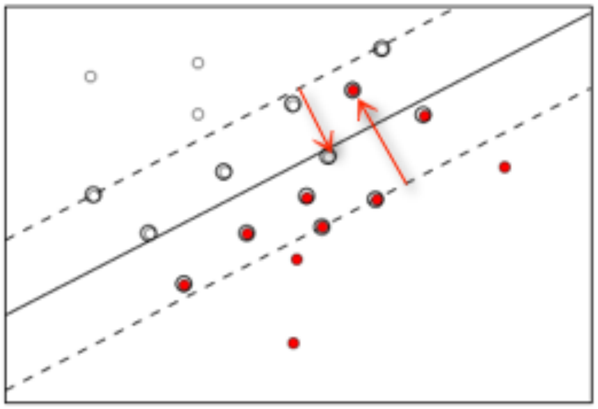
a) How many support vectors are identified for SVC-1? How many are for SVC-2?
Answer:
There are 4 support vectors in SVC-1.



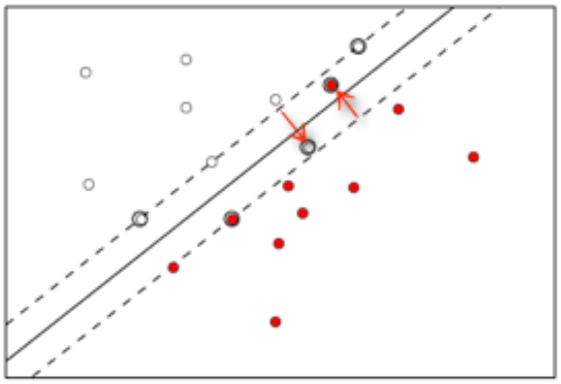
There are 5 support vectors in SVC-2.



b) How many data points are miss-classified by SVC-1? How many are by SVC-2?
Answer:
There are 2 misclassified points in SVC-1.



There are 2 misclassified points in SVC-2



c) What is most likely the C-value for SVC-1? What is the likely the C-value for SVC-2?
Answer:
On SVC-1, it looks like it uses a **small C value, educated guess of C-value = 0.001** (please refer to appendix for sample using C=0.001). Small C reduces the cost of misclassification and thus allows a pretty large minimum margin.

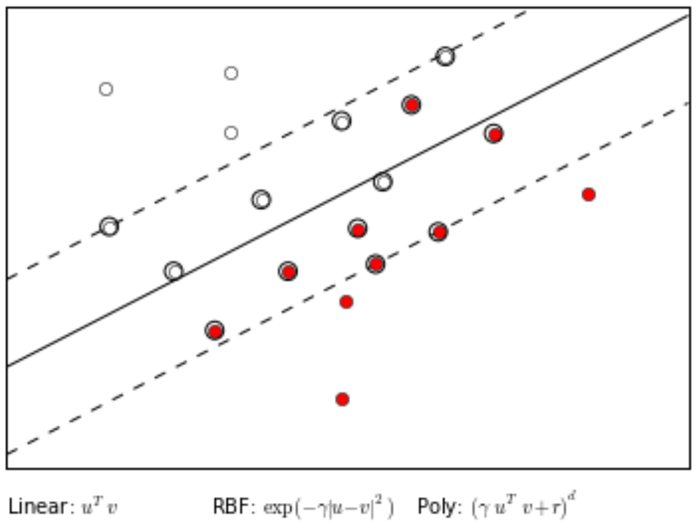


Figure 1. SVC-1

On SVC-2, it looks like it uses a **larger C value, educated guess of C-value = 0.1** (please refer to appendix for sample using C=0.1). Larger C penalizes the cost of misclassification more and thus allows pretty small minimum margin, but better variance.

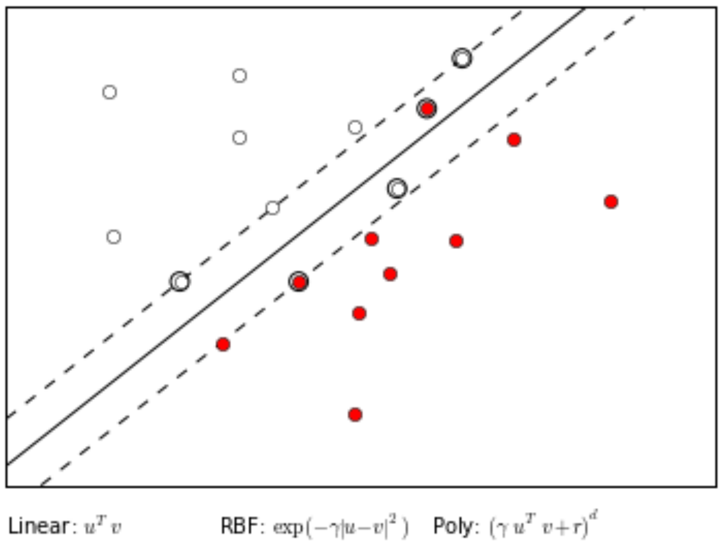


Figure 2. SVC-2

Problem 2) SVM classifier (10 pts.)

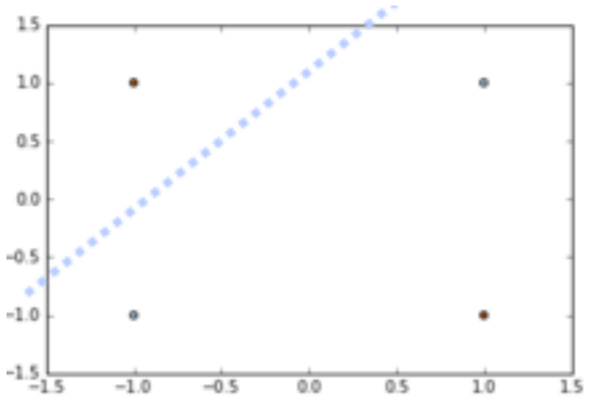
Construct a SVM classifier, either manually or using sk-learn, that can separate the 2 classes defined by the following table.

X_1	1	-1	-1	1
X_2	1	-1	1	-1
T =	-1	-1	1	1

Answer the following questions:

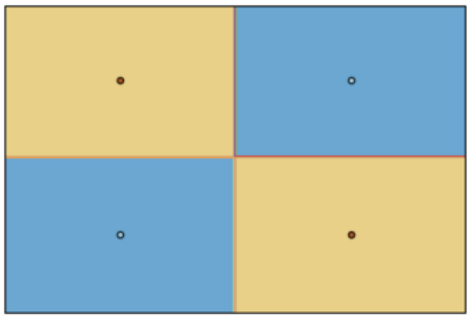
a) First with the linear SVC. What is the best accuracy that can be achieved?

Answer: Plotting of SVM classifier using Linear Kernel



Best accuracy we can achieve is 0.5.

b) Next try with RBF kernel. What is the best accuracy that you can achieve?
Answer: Plotting of SVM classifier using RBF Kernel



Best accuracy we can achieve is 1.0

Problem 3) SVM classifier (25 pts.)

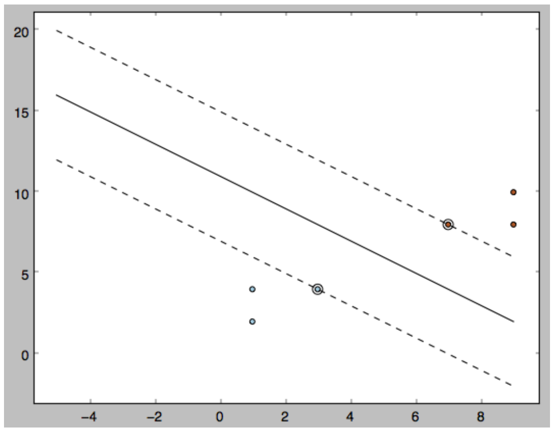
Construct a linear SVM classifier, either manually or using sk-learn, that can separate the 2 classes defined by the following table.

X_1	1	3	1	7	9	9
X_2	2	4	4	8	10	8
T =	0	0	0	1	1	1

Based on your classifier, answer the following questions:

a) What are the linear classifier coefficients?
b) What are the support vectors?
c) What is the margin?

Answer: This diagram plots the dataset and the SVM classifier, including support vectors and margin.



a) Linear classifier coefficients = [0.25 0.25]
b) Support vectors: [3,4] and [7,8]
c) Margin: 2.828

Problem 4) Bagging (25 pts.)

Create, either manually or using sk-learn, a Bagging ensemble based on single level decision tree (stump) classifier to classify the training data defined by the following table.

X_1	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
X_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
y =	1	1	1	0	0	0	1	1	1	1

To construct the ensemble, 9 classifiers will be created using the following sampling sequence:

```
[[5 0 3 3 7 9 3 5 2 4]
 [7 6 8 8 1 6 7 7 8 1]
 [5 9 8 9 4 3 0 3 5 0]
 [2 3 8 1 3 3 3 7 0 1]
 [9 9 0 4 7 3 2 7 2 0]
 [0 4 5 5 6 8 4 1 4 9]
 [8 1 1 7 9 9 3 6 7 2]
 [0 3 5 9 4 4 6 4 4 3]
```

[4 4 8 4 3 7 5 5 0 1]]

Each row corresponds to an iteration, and each item in a sequence represents the data point. For example, the 1st row [5 0 3 3 7 9 3 5 2 4] means that points 5 and 3 would be used more than once, and points 1, 6, 8 would not be used in the 1st iteration.

a) Please report the array of targets predicted by the ensemble based on the training data. An example table is given below.
Answer:

```
[
        [0 1 0 0 1 1 0 0 1 0]
        [1 1 1 1 1 1 1 1 1 1]
        [0 1 1 1 0 0 1 0 0 1]
        [1 0 1 1 0 0 0 1 1 1]
        [1 1 1 0 1 0 1 1 1 1]
        [1 0 0 0 1 1 0 1 0 1]
        [1 1 1 1 1 1 0 1 1 1]
        [1 0 0 1 0 0 1 0 0 0]
        [0 0 1 0 0 1 0 0 1 1]
]
```

b) Assuming majority vote, what is the final accuracy of your bagging ensemble for the training data?
Answer:

```
Bagging Round 1
X  0.1  0.3  0.4  0.4  0.4  0.5  0.6  0.6  0.8  1
y  1    1    0    0    0    0    0    0    1    1
⇨  y=1 if x≤0.35, else: 0

Bagging Round 2
X  0.2  0.2  0.7  0.7  0.8  0.8  0.8  0.9  0.9  0.9
y  1    1    1    1    1    1    1    1    1    1
⇨  y=1 if x>0.05, else: 0

Bagging Round 3
X  0.1  0.1  0.4  0.4  0.5  0.6  0.6  0.9  1  1
y  1    1    0    0    0    0    0    1    1  1
⇨  y=1 if x>0.75, else: 0

Bagging Round 4
X  0.1  0.2  0.2  0.3  0.4  0.4  0.4  0.4  0.8  0.9
y  1    1    1    1    0    0    0    0    1    1
⇨  y=1 if x≤0.35, else: 0

Bagging Round 5
```

```
X  0.1  0.1  0.3  0.3  0.4  0.5  0.8  0.8  1  1
y  1    1    1    1    0    0    1    1    1  1
⇨  y=1 if x≤0.35, else: 0
```

Bagging Round 6

```
X  0.1  0.2  0.5  0.5  0.5  0.6  0.6  0.7  0.9  1
y  1    1    0    0    0    0    0    1    1    1
⇨  y=1 if x>0.65, else: 0
```

Bagging Round 7

```
X  0.2  0.2  0.3  0.4  0.7  0.8  0.8  0.9  1  1
y  1    1    1    0    1    1    1    1    1  1
⇨  y=1 if x>0.55, else: 0
```

Bagging Round 8

```
X  0.1  0.4  0.4  0.5  0.5  0.5  0.5  0.6  0.7  1
y  1    0    0    0    0    0    0    0    1    1
⇨  y=1 if x>0.65, else: 0
```

Bagging Round 9

```
X  0.1  0.2  0.4  0.5  0.5  0.5  0.6  0.6  0.8  0.9
y  1    1    0    0    0    0    0    0    1    1
⇨  y=1 if x≤0.3, else: 0
```

Combining Classifiers Table:

Instead of using 1 and 0 as in the target table, I will replace 0 with -1 so the SUM of each target value is more disparity (either positive or negative) and thus easier to differentiate bags of negative votes and bag of positive votes.

		X									
Round	**Condition**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1**
1	y=1 if x≤0.35, else: 0	1	1	1	-1	-1	-1	-1	-1	-1	-1
2	y=1 if x>0.05, else: 0	1	1	1	1	1	1	1	1	1	1
3	y=1 if x>0.75, else: 0	-1	-1	-1	-1	-1	-1	-1	1	1	1
4	y=1 if x≤0.35, else: 0	1	1	1	-1	-1	-1	-1	-1	-1	-1
5	y=1 if x≤0.35, else: 0	1	1	1	-1	-1	-1	-1	-1	-1	-1
6	y=1 if x>0.65, else: 0	-1	-1	-1	-1	-1	-1	1	1	1	1
7	y=1 if x>0.55, else: 0	-1	-1	-1	-1	-1	1	1	1	1	1
8	y=1 if x>0.65, else: 0	-1	-1	-1	-1	-1	-1	1	1	1	1
9	y=1 if x≤0.3, else: 0	1	1	1	-1	-1	-1	-1	-1	-1	-1
SUM (or Vote)		1	1	1	-7	-7	-5	-1	1	1	1
Sign (negative=0, positive=1)		1	1	1	0	0	0	0	1	1	1
True Class		1	1	1	0	0	0	1	1	1	1

Conclusion

Base on the result, there is 1 outcome difference (x=0.7, yellow highlighted) between the Combining result of bagging classifiers v/s the Ground Truth. **Thus the accuracy is 9/10 = 90%**.

c) Change the target vector to [1 1 1 0 0 0 0 0 1 1] but keep the X values the same. Rerun your bagging ensemble. What is the final accuracy of your bagging ensemble for the new training data?

Answer:

Target Table:

```
0  1  0  0  0  1  0  0  1  0
0  0  1  1  1  0  0  0  1  1
0  1  1  1  0  0  1  0  0  1
1  0  1  1  0  0  0  0  1  1
1  1  1  0  0  0  1  0  1  1
1  0  0  0  0  1  0  1  0  1
1  1  1  0  1  1  0  0  0  1
1   0  0  1  0  0  0  0  0  0
0   0  1  0  0  0  0  0  1  1
```

Bagging Analysis:

Bagging Round 1:

```
X  0.1  0.3  0.4  0.4  0.4  0.5  0.6  0.6  0.8  1
y  1    1    0    0    0    0    0    0    0    1
y=1 if x≤0.35, else: 0
```

Bagging Round 2:

```
X  0.2  0.2  0.7  0.7  0.8  0.8  0.8  0.9  0.9  0.9
y  1    1    0    0    0    0    0    1    1    1
y=1 if x>0.85, else: 0
```

Bagging Round 3:

```
X  0.1  0.1  0.4  0.4  0.5  0.6  0.6  0.9  1  1
y  1    1    0    0    0    0    0    1    1  1
y=1 if x>0.75, else: 0
```

Bagging Round 4:

```
X  0.1  0.2  0.2  0.3  0.4  0.4  0.4  0.4  0.8  0.9
y  1    1    1    1    0    0    0    0    0    1
y=1 if x≤0.35, else: 0
```

Bagging Round 5:

```
X  0.1  0.1  0.3  0.3  0.4  0.5  0.8  0.8  1  1
```

```
y  1    1    1    1    0    0    0    0    1  1
y=1 if x≤0.35, else: 0
```

Bagging Round 6:

```
X  0.1  0.2  0.5  0.5  0.5  0.6  0.6  0.7  0.9  1
y  1    1    0    0    0    0    0    0    1    1
y=1 if x≤0.35, else: 0
```

Bagging Round 7:

```
X  0.2  0.2  0.3  0.4  0.7  0.8  0.8  0.9  1  1
y  1    1    1    0    0    0    0    1    1  1
y=1 if x≤0.35, else: 0
```

Bagging Round 8:

```
X  0.1  0.4  0.4  0.5  0.5  0.5  0.5  0.6  0.7  1
y  1    0    0    0    0    0    0    0    0    1
y=1 if x≤0.25, else: 0
```

Bagging Round 9:

```
X  0.1  0.2  0.4  0.5  0.5  0.5  0.6  0.6  0.8  0.9
y  1    1    0    0    0    0    0    0    0    1
y=1 if x≤0.3, else: 0
```

Combining Classifiers Table:

Instead of using 1 and 0 as in the target table, I will replace 0 with -1 so the SUM of each target value is more disparity (either positive or negative) and thus easier to differentiate bags of negative votes and bag of positive votes.

		X									
Round	**Condition**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1**
1	y=1 if x≤0.35, else: 0	1	1	1	-1	-1	-1	-1	-1	-1	-1
2	y=1 if x>0.85, else: 0	-1	-1	-1	-1	-1	-1	-1	-1	1	1
3	y=1 if x>0.75, else: 0	-1	-1	-1	-1	-1	-1	-1	1	1	1
4	y=1 if x≤0.35, else: 0	1	1	1	-1	-1	-1	-1	-1	-1	-1
5	y=1 if x≤0.35, else: 0	1	1	1	-1	-1	-1	-1	-1	-1	-1
6	y=1 if x≤0.35, else: 0	1	1	1	-1	-1	-1	-1	-1	-1	-1
7	y=1 if x≤0.35, else: 0	1	1	1	-1	-1	-1	-1	-1	-1	-1
8	y=1 if x≤0.25, else: 0	1	1	-1	-1	-1	-1	-1	-1	-1	-1
9	y=1 if x≤0.3, else: 0	1	1	1	-1	-1	-1	-1	-1	-1	-1
SUM (or Vote)		5	5	3	-9	-9	-9	-9	-7	-5	-5
Sign (negative=0, positive=1)		1	1	1	0	0	0	0	0	0	0
True Class		1	1	1	0	0	0	0	0	1	1

Conclusion

Base on the result, there is 2 outcome difference (x=0.9 and 1.0, yellow highlighted) between the Combining result of bagging classifiers v/s the Ground Truth. **Thus the accuracy is 8/10 = 80%**.

Problem 5) Random Forest (25 pts.)

Use the same training data as in Problem-4. Create a Random Forest classifier using sklearn that has 9 estimators with all default values. Answer the following questions.

a) For targets = [1 1 1 0 0 0 1 1 1 1], what is the accuracy of the random forest?
Answer: by using sk-learn, the accuracy is flipped between **0.9 or 1.0**

```
$ python hw4_150201133.py
X:
[[ 0.1  0.1]
 [ 0.2  0.2]
 [ 0.3  0.3]
 [ 0.4  0.4]
 [ 0.5  0.5]
 [ 0.6  0.6]
 [ 0.7  0.7]
 [ 0.8  0.8]
 [ 0.9  0.9]
 [ 1.   1. ]]
Y:
[1, 1, 1, 0, 0, 0, 1, 1, 1, 1]
Accuracy: 1.0 (0.9 some times)
```

Note: Please refer to python implementation in file hw4_150201133.py

b) For targets = [1 1 1 0 0 0 0 0 1 1], what is the accuracy of the random forest?
Answer: by using sk-learn, the accuracy is flipped between **0.9 or 1.0**

```
$ python hw4_150201133.py
X:
[[ 0.1  0.1]
 [ 0.2  0.2]
 [ 0.3  0.3]
 [ 0.4  0.4]
 [ 0.5  0.5]
 [ 0.6  0.6]
 [ 0.7  0.7]
 [ 0.8  0.8]
 [ 0.9  0.9]
 [ 1.   1. ]]
Y:
[1, 1, 1, 0, 0, 0, 0, 0, 1, 1]
Accuracy: 1.0 (0.9 some times)
```

Note: Please refer to python implementation in file hw4_150201133.py

c) Give your opinions on why Random Forest is doing better or worse than Bagging.
Answer: Base on the results in Qn4 and Qn5, Random Forest seems to perform better than Bagging, at least 10-20% in our cases.
This could be due to one main difference between the Bagging and the Random Forest algorithm. In Bagging technique, it grows the trees in similar ways, influenced by 1 strong predictor and a couple of moderately strong ones. Thus the predictions will be fairly correlated. And averaging out those correlated variables won't help much to improve the variance in Bagging technique. In order to improve the variance situation, the Random Forest technique decorrelates the members of its ensemble, i.e. the trees. When building a Random Forest, at each split in the tree, the algorithm is not allowed to consider all available predictors, but only a subset of it. By doing this, the predictions will be more variant, and thus be more efficient, resulting in a more reliable and accurate trees.

d) Given the test data as follows, what are the prediction from the random forest classifier trained using the targets in (a)?

X_1	0.9	0.8	0.7	0.6	0.4	0.3	0.2	0.1
X_2	0.1	0.2	0.3	0.4	0.6	0.7	0.8	0.9

Answer: The target prediction may differ in each execution:

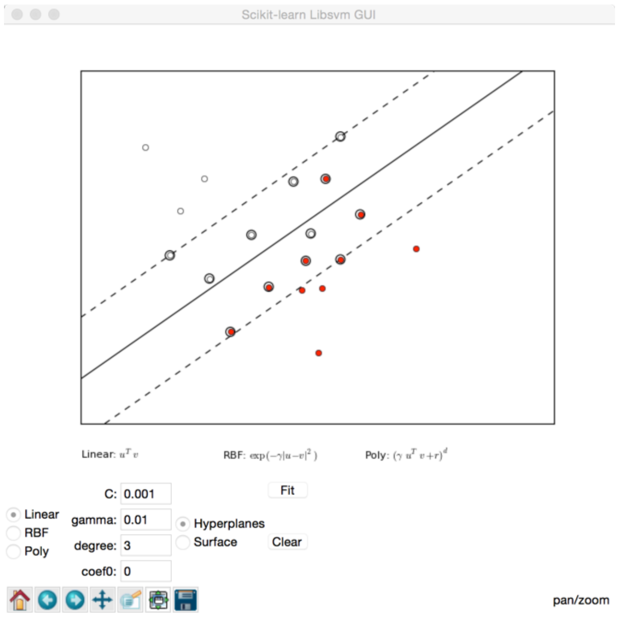
```
Y_predict:
[1, 1, 1, 0, 0, 1, 1, 1]
or
[0, 0, 0, 0, 0, 1, 1, 1]
or
[1, 1, 1, 0, 0, 0, 0, 0]
or
[1, 1, 0, 0, 0, 1, 1, 1]
or
[1, 1, 1, 0, 0, 0, 1, 1]
```

Note: Please refer to python implementation in file hw4_150201133.py

Appendix

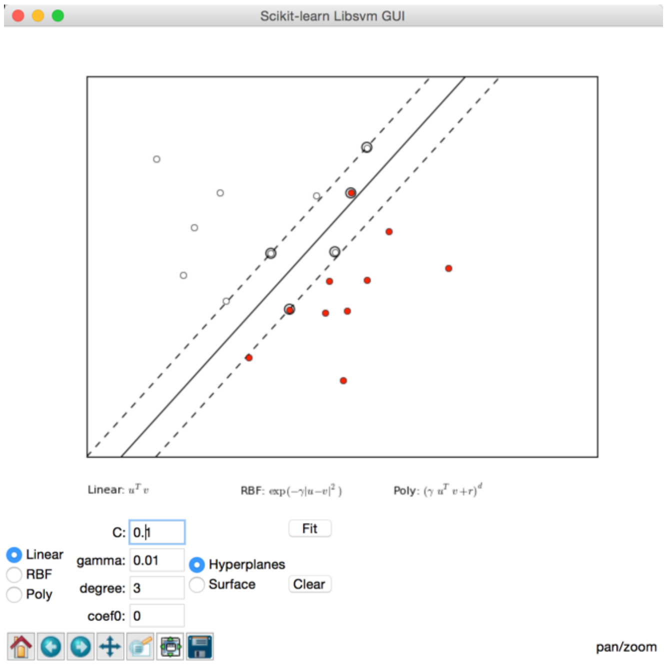
Question 1c)

C=0.001



C: 0.001
gamma: 0.01
degree: 3
coef0: 0
[✓ Linear] RBF Poly
[✓ Hyperplanes] Surface

C=0.1



C: 0.1
gamma: 0.01
degree: 3
coef0: 0
[✓ Linear] RBF Poly
[✓ Hyperplanes] Surface