CS 6362: Machine Learning
Assignment 2
Support Vector Machines

Due: 10/14/2019, at midnight

General Instructions:

If anything is ambiguous or unclear:

1. **Discuss possible interpretations with other students, your TA, and instructor. You may email your instructor or TA if you have specific questions or need clarification on questions.**
2. **Make assumptions, state them explicitly, and then use them to work out the problem**

Remember that after general discussions with others, you are required to work out the solutions by yourself. Do not plagiarize code from any available source. All submitted work must be your own work. Please refer to the Honor code for clarifications.

Scikit-learn contains a number of synthetic data sets. In this question, you will perform SVM experiments on one of them, called Moons (http://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_moons.html), which has two- dimensional input points belonging to two classes. You will need to import the following Python modules:

```
from sklearn.svm import SVC
from sklearn import datasets
import matplotlib.pyplot as plt
import numpy as  np
```

The following commands create and display a random sample of Moons data. Here, X is a list of two-dimensional points, and y is a list of corresponding class labels. Each label is either 0 or 1.

```
data  =  datasets.make_moons(n_samples=2000, noise=0.1)
X,y = data
plt.figure()
plt.suptitle('Moons Data Sample') colors = p.array(['r','b'])
plt.scatter(X[:, 0], X[:, 1], color=colors[y],s=3)
```

The goal in this assignment is to train a soft-margin SVM to separate the two classes

and to visualize the effect of different parameter values. (To make the problem more challenging, you will be using a noisier data sample and fewer training p o i n t s .)

To train an SVM on the above data, the following commands first set the parameters of the SVM, and then fit the SVM to the data using an RBF kernel (the default):

```
clf = SVC(gamma=1.0, C=1.0)
clf.fit(X,y)
```

Here, clf is a Python object representing the SVM. Its kernel is given by

$$k(x, y) = exp(-\gamma x - y^2).$$

To visualize the decision function of the trained SVM, import the Python module bonnerlib from the Brightspace assignment page. The following command then displays a scatter plot of the data superimposed on a contour plot of the decision function:

```
bonnerlib.dfContour(clf,data)
```

The plot is shown in Figure 1, attached. The solid black line represents the decision boundary, and the black dashed lines represent the two margins. In addition, the following command displays the decision function in 3D above a contour plot:

```
bonnerlib.df3D(clf,data)
```

Use this function to generate and study the 3D plot (see Figure 2).

Finally, the following commands use the trained SVM to make predictions and evaluate the decision function, respectively, at a list of points, Xtest. Note that the decision function is a real number while a prediction is a label, i.e., either 0 or 1.

```
clf.predict(Xtest)
clf.decision_function(Xtest)
```

Given the correct labels, Ytest, for these points, the following command returns the prediction accuracy, that is, the fraction of predicted labels that are the same as the correct labels. The prediction error is $1 - accuracy$.

```
clf.score(Xtest,Ytest)
```

In the questions below, hand in all plots you are asked to generate. Label each plot with the question number it refers to and any other pertinent information, such as the SVM parameter values used to generate it. It should be clear which question each plot is addressing. Comment the programs you are asked to write and hand them in. The code should be well-written and easy to understand. Explanations should be clear, complete and well-written. Points will be deducted for poor English and long, rambling answers.

(I) Generate two random samples of Moons data, a training set with 200 points and a test set with 2000 points. Both sets should have a noise level of 0.4 (so that the two moon classes overlap significantly). Use these two data sets throughout the rest of this question. In the questions below, you will be evaluating the training error and the test error of an SVM. The training error is the error on the training data, and the test error is the error on the test data.

(II) A soft-margin SVM with an RBF kernel requires two parameters, C and γ. Write a Python program that carries out an exhaustive grid search for the best values of these parameters. (Do *not* use Scikit-learn's build-in grid-search facility.) You should consider values of C and γ spread over several orders of magnitude. Choose them so that the values of log C are equally spaced, and likewise for log γ. You should use at least five values per order-of-magnitude (i.e., per factor-of-ten). (You may find the function numpy.linspace useful.) For each combination of values of C and γ, fit an SVM to the training data and evaluate its test error. Save the combination of C and γ that gives the lowest test error. Call these values C_0 and γ_0. Report these values as well as the test error. Generate and hand in a contour plot of the decision function with the decision boundary and margins highlighted.

(III) In this question you will fix the value of γ and vary the value of C. Generate two curves, one of test error vs $log(C)$, and one of training error vs $log(C)$. Use $\gamma = \gamma_0$, and use 100 different values of $log(C)$ equally spaced between $log(C_0) - 3$ and $log(C_0) + 3$. For each value of C you will have to retrain and retest the SVM. Plot both curves on one set of axes, using blue for the training error and green for the test error. You should find that the training error tends to decrease as C increases, and that the test error first tends to decrease and then increase, with its minimum very near C_0. Provide an intuitive explanation of this behavior.

(IV) Generate 7 contour plots of the decision function for different values of C. The values of $log(C)$ should be equally spaced between $log(C_0) - 3$ and $log(C_0) + 3$, inclusive. Use $\gamma = \gamma_0$ for each value of C. Display the 7 contour plots in a single figure in a grid pattern. (You will have to use the Python function plt.subplot (https://matplotlib.org/api/_as_gen/matplotlib.pyplot.subplot.html)) The plots should highlight the decision boundaries, but not the margins. Provide an intuitive explanation of the changes you see. (Feel free to include other figures to support your explanations.)

(V) In this question you will fix the value of C and vary the value of γ. Generate two curves, one of test error vs $log(\gamma)$, and one of training error vs $log(\gamma)$. Use $C = C_0$, and use 100 different values of $log(\gamma)$ equally spaced between $\log(\gamma_0) - 3$ and $log(\gamma_0)+3$. For each value of γ you will have to retrain and retest the SVM. Plot both curves on one set of axes, using blue for the training error and green for the test error. You should find that the training error tends to decrease

as γ increases, and that the test error first tends to decrease and then increase, with its minimum very near γ_0. Provide an intuitive explanation of this behavior.

(VI) Generate 7 contour plots of the decision function for different values of γ. The values of $log(\gamma)$ should be equally spaced between $log(\gamma_0) - 3$ and $log(\gamma_0) + 3$, inclusive. Use $C = C_0$ for each value of γ. Display the 7 contour plots in a single figure in a grid pattern, as in part (d). Provide an intuitive explanation of the changes you see. (Feel free to include other figures to butress your explanation.)

Useful links:

- Numpy libraries (Numerical Python): http://www.numpy.org/
- The SciPy libraries (Scientific Python): https://scipy.org/
- scikit-learn libraries (machine learning in Python): http://scikit-learn.org/stable/
- Tutorial on machine learning in scikit-learn:

  http://scikit-learn.org/stable/tutorial/basic/tutorial.html
- Support Vector Machines in scikit-learn:

  http://scikit-learn.org/stable/modules/svm.html#svm
- Getting started with SciPy: https://scipy.org/getting-started.html
- SciPy documentation: https://scipy.org/docs.html
- SciPy lecture notes by Valentin Haenel, Emmanuelle Gouillart, and Gael Varoquaux (eds): http://www.scipy-lectures.org/index.html

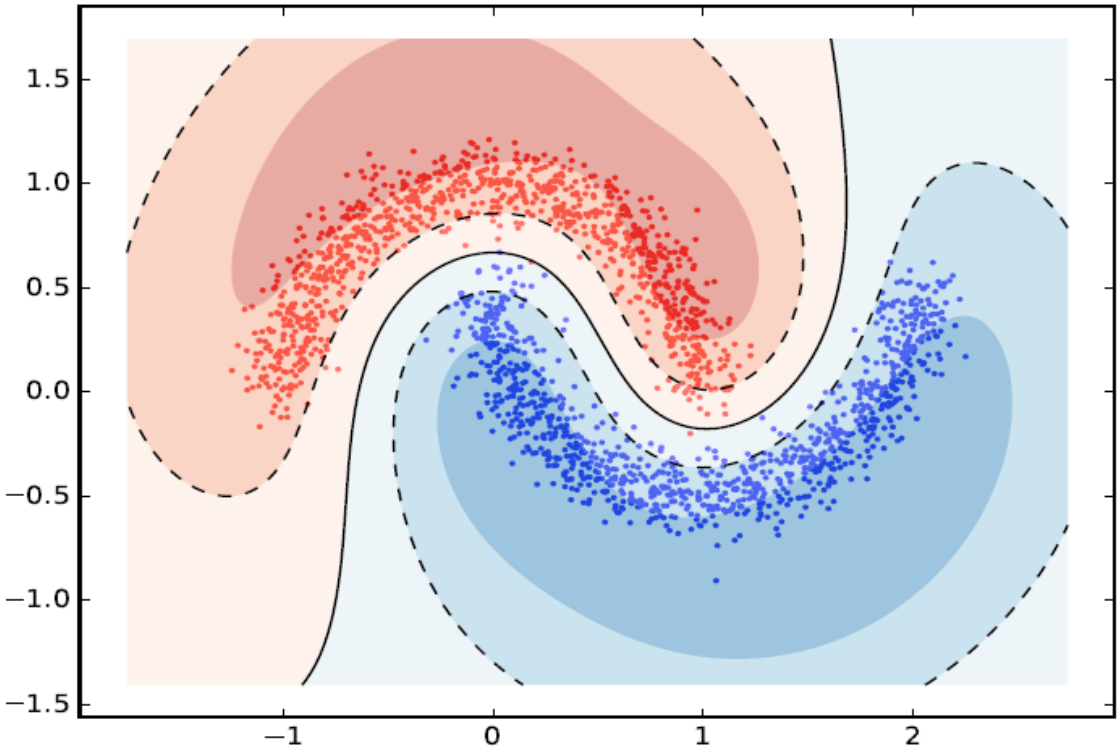

Figure 1

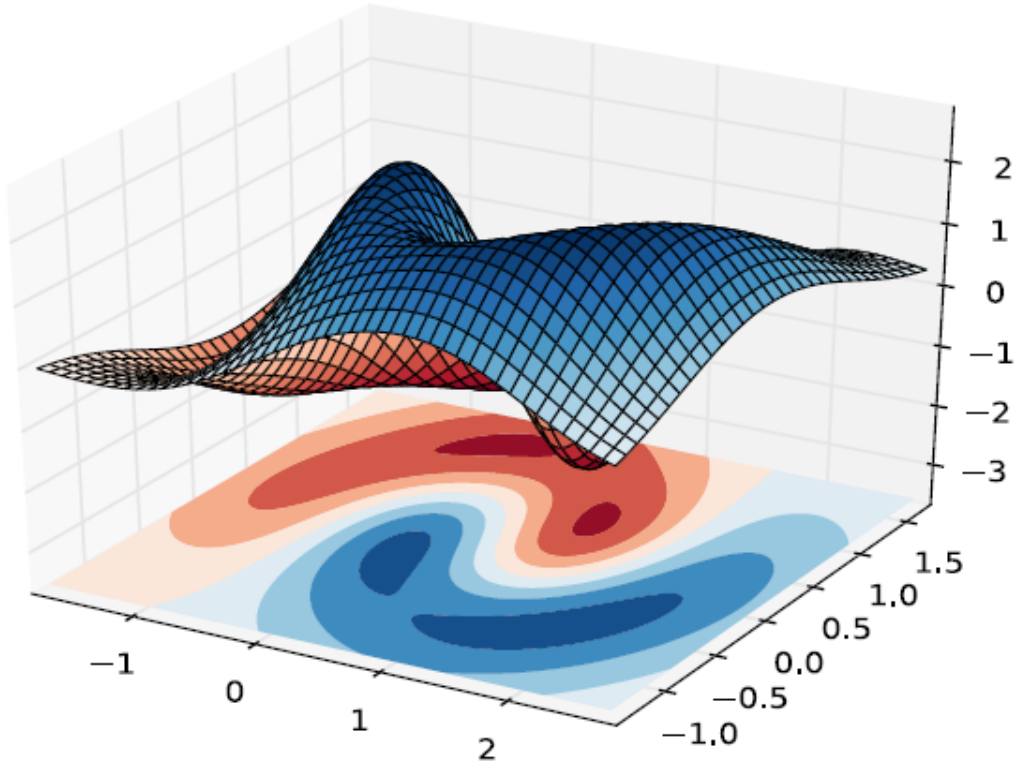
2
1
0
−1
−2
−3
−1
0
1
2
1.5
1.0
0.5
0.0
−0.5
−1.0

Figure 2

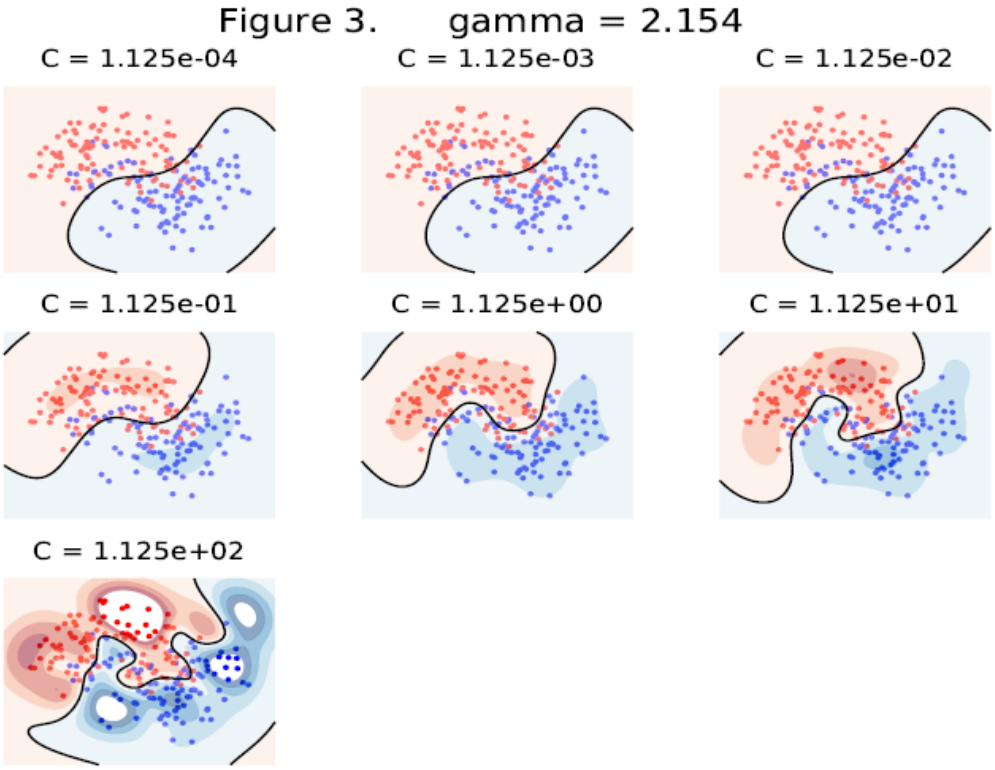
Figure 3. gamma = 2.154
C = 1.125e-04
C = 1.125e-03
C = 1.125e-02
C = 1.125e-01
C = 1.125e+00
C = 1.125e+01
C = 1.125e+02