CSC 2545, Spring 2017
Kernel Methods and Support Vector Machines

Assignment 1

This assignment is due at the start of class, at 2:10pm, Thurs Feb 16.
No late assignments will be accepted.

The material you hand in should be legible, well-organized and easy to mark, including the use of good English. Short, simple answers and proofs will receive more marks than long, complicated ones. Up to 20% of your mark will be for presentation. Unless stated otherwise, you must justify your answers.

All computer problems are to be done in Python with Scikit-learn and should be properly commented and easy to understand. These programs should all be stored in a single file called `source.py`, which should be submitted electronically as described on the course web site. We should be able to import this file as a Python module and execute your programs.

No more questions will be added

1. (20 points) Question 7.7 on page 224 of the text.

2. (20 points) Question 7.13 on page 225 of the text. In addition to solving for the dual, show how to compute the bias term b in the decision function, $\omega \bullet x + b$.

3. (20 points) Argue that a soft-margin SVM (C-SVM) does not always have a unique solution for the bias term b in $\omega \bullet x + b$. Characterize those situations in which the solution is unique, and show how to compute b. Give a geometrical interpretation of those situations in which b is not unique. Show how to compute the range of possible b values. What does this say about the decision function and the decision boundary?

   Finally, give a specific example of when b is not unique. You should specify C and all the training examples (x_i, y_i). It is sufficient to consider a linear kernel (i.e., $k(x_i, x_j) = x_i \bullet x_j$) and to assume that the input vectors, x_i, are one dimensional. Your example should include points that are both inside and outside the margins.

4. (20 points total) Scikit-learn contains a number of synthetic data sets. In this question, you will perform SVM experiments on one of them, called Moons, 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 = np.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 points.)

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 the decision function of the trained SVM, download and import the Python module `bonnerlib` from the course web 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)
```

The 3D plot is shown in Figure 2, attached.

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 minus the 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.

(a) 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.

(b) 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 $\log \gamma$. 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.

(c) 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 a intuitive explanation of this behaviour.

(d) 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, as shown in Figure 3. (You will have to use the Python function plt.subplot.) 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 butress your explanation.)

(e) 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 behaviour.

(f) 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.)

Figure 1

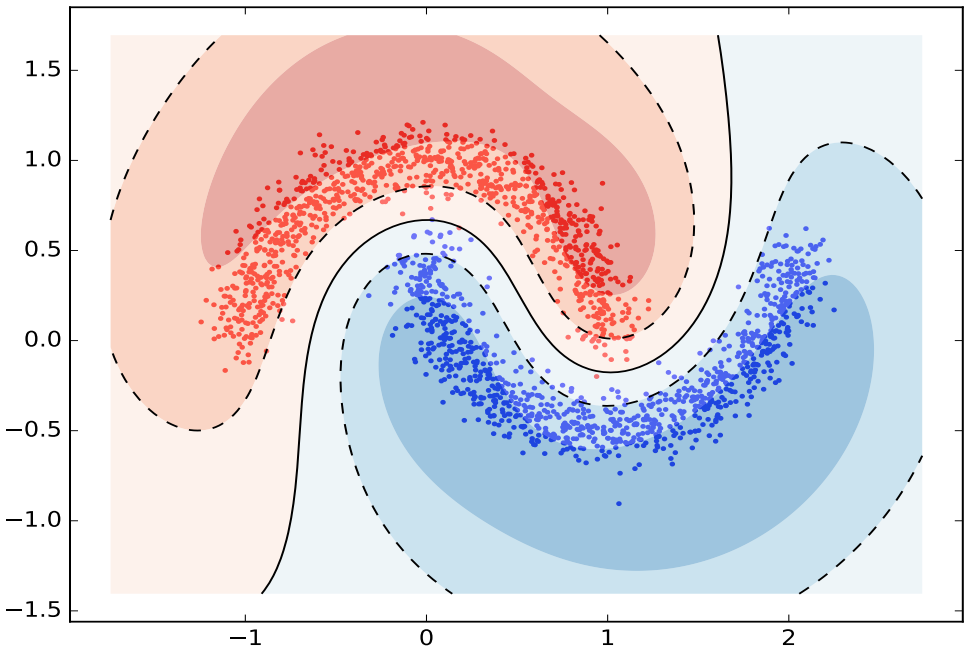

Figure 2

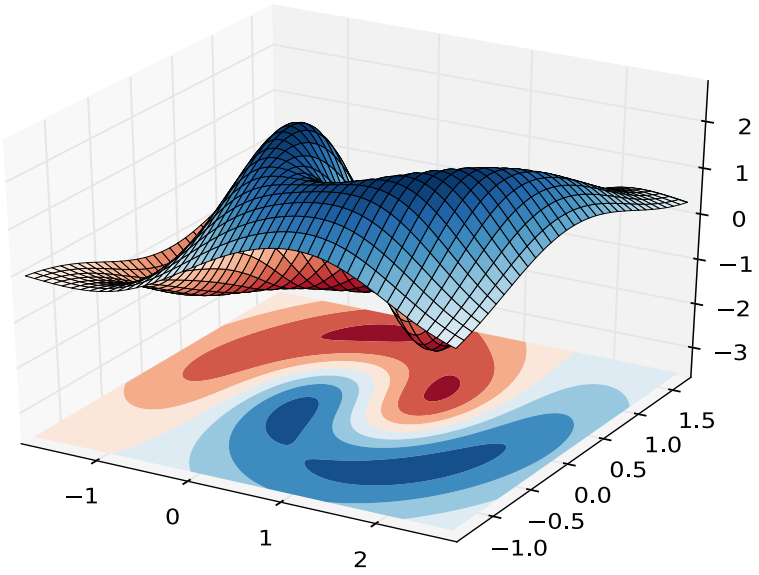

Figure 3. gamma = 2.154