CSC 2545, Spring 2017
Kernel Methods and Support Vector Machines

Assignment 2

Due at the start of class, at 2:10pm, Thurs March 23.
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. Plots, other computer output, and explanations should be submitted electronically as a single pdf file called `results.pdf`. All computer problems are due on Friday March 31 at 11pm.

No more questions will be added

1. (7 points) Prove the following statements in the context of ν-SVC:

   (a) if $y_i(\omega \bullet x_i + b) > \rho$ then $\alpha_i = 0$

   (b) if $y_i(\omega \bullet x_i + b) < \rho$ then $\alpha_i = 1/m$

2. (15 points) Question 9.2 on page 274 of the text.

3. (10 points) Question 9.16 on page 276 of the text.

4. (5 points) Show that the following optimization problem is equivalent to a soft-margin SVM:

$$\min_{\omega,b} \; \|\omega\|^2/2 + C\sum_i h[y_i(\omega \bullet x_i + b)]$$

   where $h(z) = \max(0, 1-z)$ is the hinge loss function.

5. (30 points total) In this problem, we will formulate a soft-margin SVM without using a mapping to a higher-dimensional feature space. Instead, we will search directly for a decision function of the form $f(x) = \sum_i \beta_i k(x, x_i) + b$. In particular, consider the following optimization problem:

$$\min_{\beta,b} \; \sum_{i,j} \beta_i\beta_j k(x_i, x_j)/2 + C\sum_i h[y_i f(x_i)]$$

   where $f(x) = \sum_i \beta_i k(x, x_i) + b$ and h is the hinge loss function (and $C > 0$). As usual, the x_i range over the training data.

(a) (5 points) Assuming that $k(x, y)$ is a similarity measure, give an intuitive description of the term $\sum_{i,j} \beta_i \beta_j k(x_i, x_j)$ without referring to a feature space or maximizing a margin. How is this reflected in the properties of an SVM?

(b) (10 points) Introduce slack variables into the problem, derive the dual, and show that it is the same as the dual of a soft-margin SVM.

(c) (4 points) Under what conditions are the β_i uniquely determined by the solution to the dual problem? In this case, give a simple formula for β_i.

(d) (2 points) Do we need to assume that $k(x, y)$ is positive definite as in the standard formulation of an SVM? If so, how is it assumed?

(e) (4 points) Do we need to assume that $k(x, y)$ is symmetric as in the standard formulation of an SVM? If so, how is it assumed?

(f) (5 points) Show that the solution to the dual problem (as given on the bottom of page 205 in the text) does not depend on k being symmetric. That is, if k is not symmetric, then there is a symmetric k' that gives exactly the same solution for the dual variables.

6. (20 points) In this question, you will write Python programs to apply support vector regression (SVR) to a synthetic data set and visualize the effect of different parameter values. You will need to import the following Python modules:

```
from sklearn.svm import SVR
import numpy as np
import numpy.random as rnd
import matplotlib.pyplot as plt
```

Recall that support vector regression has two parameters, C (the weight of the error term) and ϵ (the width of the tube around the prediction function). The following Python commands set these parameters, and then fit an SVR to a data set using an RBF kernel (the default):

```
svr = SVR(gamma=1.0, C=1.0, epsilon=0.1)
svr.fit(X,y)
```

Here, `svr` is a Python object representing the SVR. The kernel is given by $k(x_1, x_2) = exp(-\gamma\|x_1 - x_2\|^2)$. The training data consists of a set of pairs $(x_1, y_1), ..., (x_m, y_m)$, which are stored in the matrix `X` and the vector `y`. Specifically, x_i is the i^{th} row of `X`, and y_i is the i^{th} element of `y`.

Finally, the following command uses the trained SVR to predict the y values at a list of points, `Xtest`:

```
svr.predict(Xtest)
```

Below, we use $\hat{f}(x)$ to denote the predicted value of y at point x. Since we are doing regression, these predictions are real numbers.

Given the correct y values, `Ytest`, the following command returns the prediction accuracy, as measured by the coefficient of determination. The prediction error is 1 minus the accuracy.

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

The coeffcient of determination is derived from mean-squared error. For more details, see

`http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html#sklearn.svm.SVR.score`

The first step below is to generate synthetic data. Each synthetic data point has the form (x, y) where x and y are random variables. In this assignment, x is uniformly distributed on the interval $[-4, 4]$, and $y = f(x) + \eta$, where $f(x) = \sin(x)/x + x/20$ and η is independent Gaussian noise with mean 0 and variance σ^2, that is, $\eta \sim \mathcal{N}(0, \sigma^2)$. Helpfully, the function `numpy.sinc` evaluates $\sin(x)/x$ (and returns 1 when $x = 0$). Note that $f(x)$ is the expected value of y for a given value of x.

You should hand in all plots and other computer output you are asked to generate. Label each piece of output with the question number it refers to and any other pertinent information, such as the SVR 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 synthetic data as described above, a training set with 20 points and a test set with 1000 points. Both sets should have a noise level of $\sigma = 0.1$. Use these two data sets throughout the rest of this question, where you will be evaluating the training error and the test error of support vector regression for various SVR parameter values.

(b) SVR with a RBF kernel requires three parameters, C, γ and ϵ. Write a Python program that carries out a stochastic search for the best values of these parameters. (Do *not* use Scikit-learn's build-in search facilities.) You should randomly generate 10,000 triples (C, γ, ϵ), where $\log(C)$, $\log(\gamma)$ and $\log(\epsilon)$ are chosen independently from uniform distributions. Choose the range of the uniform distributions so that C, γ and ϵ are spread over several orders of magnitude. You may find the function `rand` in `numpy.random` useful.

    For each random triple (C, γ, ϵ), fit an SVR to the training data and evaluate its test error. Save the triple that gives the lowest test error. Call these values C_0, γ_0 and ϵ_0. Report these values as well as the test error. In addition, using these values, generate a plot that shows all of the following on a single pair of axes:

    - $\hat{f}(x)$, the predicted value of y given x (in green)
    - $f(x)$, the expected value of y given x (in orange)

- the training data (as blue dots).

The result should look something like Figure 1 below.

(c) In this question you will fix the values of γ and ϵ 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 $\epsilon = \epsilon_0$, and use 100 different values of $log(C)$ equally spaced between $\log(C_0) - 4$ and $log(C_0) + 4$. For each value of C you will have to retrain and retest the SVR. Plot both curves on one set of axes, using blue for training error and orange for test error. The result should look something like Figure 2 below. Provide an intuitive explanation of any trends you observe.

(d) Generate 9 plots, each similar to that in Figure 1, for different values of C. The values of $log(C)$ should be equally spaced between $log(C_0) - 4$ and $log(C_0) + 4$, inclusive. Use $\gamma = \gamma_0$ and $\epsilon = \epsilon_0$ for each value of C. Display the 9 plots in a single figure in a grid, as in Figure 3 below. (You will have to use the Python function `plt.subplot`.) Provide an intuitive explanation of the changes you see.

(e) In this question you will fix the values of C and ϵ 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 $\epsilon = \epsilon_0$, and use 100 different values of $log(\gamma)$ equally spaced between $\log(\gamma_0) - 4$ and $log(\gamma_0) + 4$. For each value of γ you will have to retrain and retest the SVR. Plot both curves on one set of axes, using blue for training error and orange for test error. Provide an intuitive explanation of any trends you observe.

(f) Generate 9 plots, each similar to that in Figure 1, for different values of γ. The values of $log(\gamma)$ should be equally spaced between $log(\gamma_0) - 4$ and $log(\gamma_0) + 4$, inclusive. Use $C = C_0$ and $\epsilon = \epsilon_0$ for each value of γ. Display the 9 plots in a single figure in a grid pattern, as in Figure 3. Provide an intuitive explanation of the changes you see.

(g) In this question you will fix the values of C and γ and vary the value of ϵ. Generate two curves, one of test error vs $log(\epsilon)$, and one of training error vs $log(\epsilon)$. Use $C = C_0$ and $\gamma = \gamma_0$, and use 100 different values of $log(\epsilon)$ equally spaced between $\log(\epsilon_0) - 2$ and $log(\epsilon_0) + 2$. (Note the difference from parts (c) and (e) above.) For each value of ϵ you will have to retrain and retest the SVR. Plot both curves on one set of axes, using blue for training error and orange for test error. Provide an intuitive explanation of any trends you observe.

(h) Generate 9 plots, each similar to that in Figure 1, for different values of ϵ. The values of $log(\epsilon)$ should be equally spaced between $log(\epsilon_0) - 2$ and $log(\epsilon_0) + 2$, inclusive. (Note the difference from parts (d) and (f) above.) Use $C = C_0$ and $\gamma = \gamma_0$ for each value of ϵ. Display the 9 plots in a single figure in a grid pattern, as in Figure 3. Provide an intuitive explanation of the changes you see.

Figure 1: training data (blue), $f(x)$ (green), $\hat{f}(x)$ (orange)

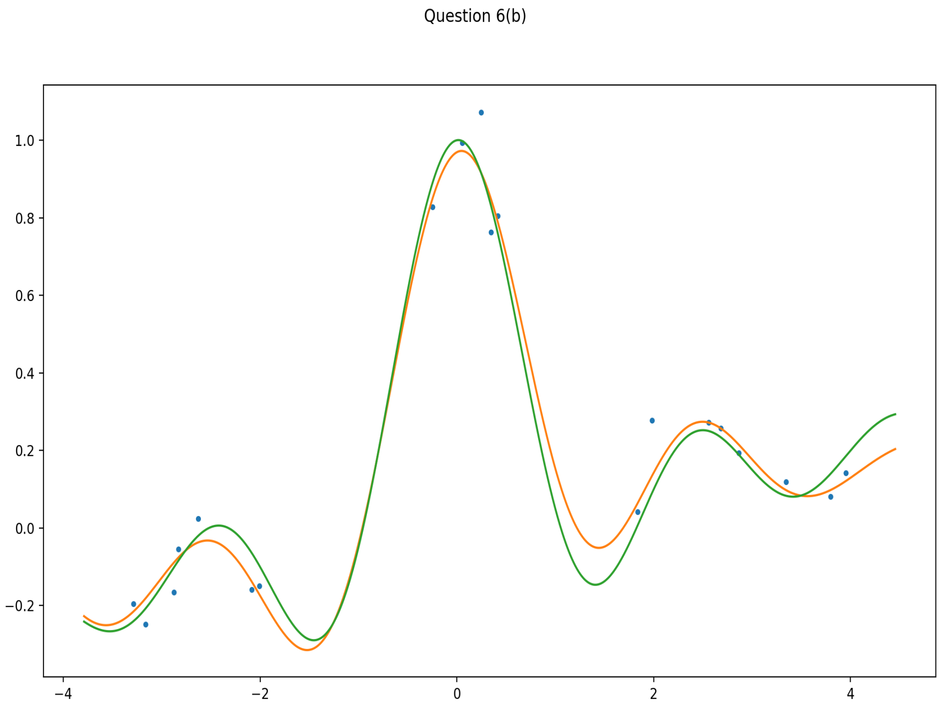

Figure 2: training error (blue) and testing error (orange)

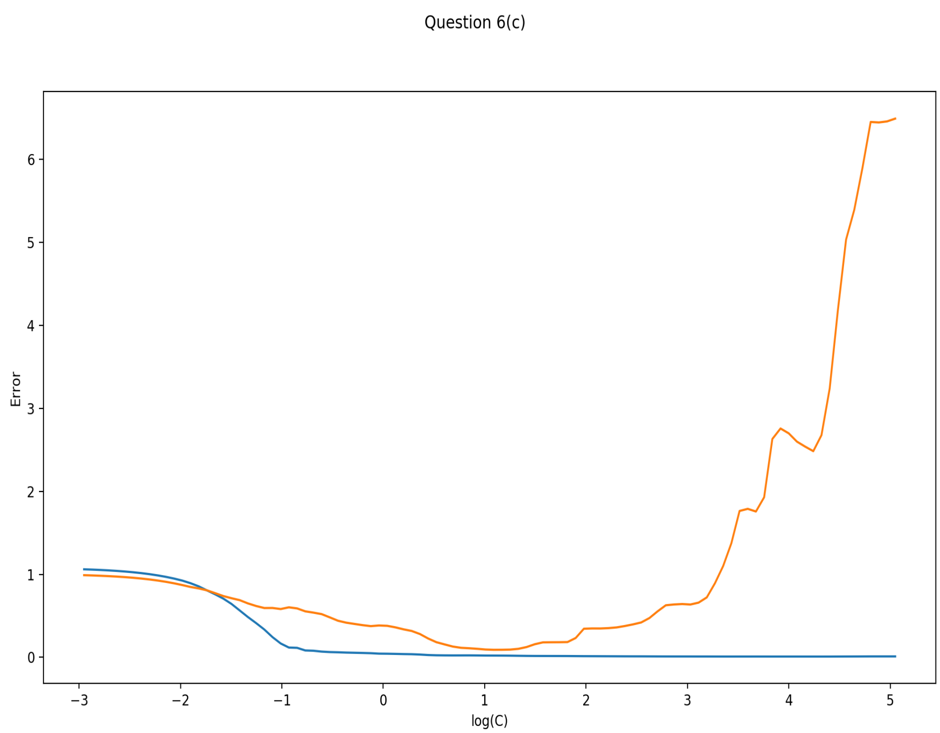

Figure 3: varying values of C