Assignment 3

This assignment is due on Thursday December 18 at 3pm in my office (BA5230). 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.

1. Question 14.5 on page 453 of the text.

2. Question 14.13 on page 454 of the text. (The “covariance matrix” referred to in the question is actually the Gram matrix, K.)

3. SVM experiments.

   Getting started. Download and install the Spider, as described on the course web page. Start Matlab and execute the following commands:

   ```matlab
   clear classes;
d=gen(spiral({'m=300','n=2','noise=1'}));
plot(d);
   ```

   This generates and plots a data object, d, consisting of m=300 points distributed near the arms of a spiral with n=2 turns. Each point is in either red or blue. Try generating and plotting data for different values of m, n and noise.

   Re-execute the second command above (the one defining d), and then execute the following Matlab commands:

   ```matlab
   [r,a]=train(svm({kernel('rbf',1), 'C=1'}),d);
plot(a);
   ```

   This trains an svm on data d and plots the results. The svm uses C = 1 and a rbf kernel with σ = 1, that is, \( k(x,y) = \exp(-\|x - y\|^2/2\sigma^2) \). In the first command line, a is the trained support vector machine, and r is a data structure containing predictions.

   In the plot, the thick red line is the decision boundary, and the thin turquoise lines are the margins. Points with a turquoise outline are support vectors (points near the margin
have circular outlines, others have square outlines). Most of the data in this example is
near the decision boundary, so there are lots of support vectors.

Try training svms on spiral data using a variety of different values for $\sigma$ and $C$. To try a
polynomial kernel of degree $d$, use $\text{kernel}(\text{`poly',}d)$. In this case, $k(x, y) = (x \cdot y + 1)^d$.
To try a linear kernel, use $\text{kernel}(\text{`linear'})$. In this case, $k(x, y) = x \cdot y$.

For the purpose of programming, it is worth knowing that the above command line for
training an svm is equivalent to the following:

```plaintext
s = svm;
s.C = 1;
svm.child = kernel(\text{`rbf',}1);
[r,a] = train(s,d);
```

Here, $s$ specifies the svm, and the last line trains it on data $d$ to give prediction algorithm
$a$. The prediction algorithm can be applied to test data. For example,

```plaintext
dataTest = gen(spiral\{\text{`m=500',`n=2',`noise=1'}})
; tst = test\(a\),\text{dataTest})
; err = loss\(tst\),\text{class_loss'})
; display\(err\).Y)
```

The first line generate 500 test data points. The second line uses the prediction algorithm
$a$ to classify each of the test points, the third line computes the prediction error, and the
last line prints out the error. The parameter $\text{class_loss}$ means that error is measured
as the fraction of missclassified data points. See the Spider website for other measures of
error. The same functions can be applied to the training data to compute training error.

In the questions below, keep your explanations short and 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. You may find the Matlab functions $\text{xlabel}$, $\text{ylabel}$,
$\text{title}$, $\text{strcat}$ and $\text{num2str}$ useful for labelling the plots. Comment the programs you
are asked to write in parts (e) and (f) and hand them in. The code should be well-written
and easy to understand.

(a) **Training data.** Generate 300 data points using a spiral with two turns and a noise
level of 2. Plot the data points.

(b) **Linear kernel.** Train an svm on this data using a linear kernel with $C = 1$. Plot the
results and hand in the plot. What is the error on the training data? Why is this
value to be expected?

(c) **Polynomial Kernel.** Train an svm on this data using a polynomial kernel of degree 2
with $C = 1$. Plot the results and hand in the plot. What is the error on the training
data? Why is this value to be expected?
(d) **Testing data.** Generate 500 data points using a spiral with two turns and a noise level of 2. Plot the data points.

(e) **RBF kernel with various C values.**

Write a Matlab program to train and test an svm for a variety of different values of C using an rbf kernel with σ = 1. Use 100 different values of C ranging from C = 0.1 to C = 100. Choose them so that the values of log C are equally spaced. For each value of C, train the svm on the data in part (a) and test the svm on the data in part (d). Compute both the training error and the test error. In a single graph, plot the training error (as a blue curve) and the test error (as a red curve) as a function of C. Use the Matlab function `semilogx` to plot the C values on a log scale. You may find the Matlab functions `figure` and `hold` useful.

You should notice that the training error is always less than the test error. Why is this? You should also notice that the training error (in blue) tends to get smaller as C get bigger. Why is this? You should also notice that the testing error (in red) at first tends to get smaller as C gets bigger, but then tends to get big again. Why is this? Finally, you should notice that the training and test errors are most similar for small values of C. Why is this?

What is the minimal test error? For what value of C does this occur? Plot the decision boundary and margins for this value of C. Plot them again for C = 0.1 and C = 100. What is different about these figures? Explain the differences.

Train an svm using C = Inf. What is the training error? What is the test error? Plot the decision boundary and margins. Explain the figure and the errors.

(f) **RBF kernel with various σ values.**

Write a Matlab program to train and test an svm with C = 2 and an rbf kernel with a variety of different values for σ. Use 100 different values of σ ranging from σ = 0.01 to σ = 10. Choose them so that the values of log σ are equally spaced. For each value of σ, train the svm on the data in part (a) and test the svm on the data in part (d). Compute both the training error and the test error. In a single graph, plot the training error (as a blue curve) and the test error (as a red curve) as a function of σ. Use the Matlab function `semilogx` to plot the σ values on a log scale.

You should notice that the training error (in blue) tends to get bigger as σ get bigger. Why is this? You should also notice that the testing error (in red) at first tends to get smaller as σ gets bigger, but then tends to get big again. Why is this? Finally, you should notice that the training and test errors are most similar for large values of σ. Why is this?

What is the minimal test error? For what value of σ does this occur? Plot the decision boundary and margins for this value of σ. Plot them again for σ = 10, σ = 0.1 and σ = 0.01. What is different about these figures? Explain the differences. In particular, for σ = 0.01, where is the decision boundary?

---

No more questions will be added