Homework 4, ECE 6254, Spring 2014

Due: Wednesday March 5, at the beginning of class

Problems:

1. **Kernel Ridge Regression**

   In this problem you will implement the kernel ridge regression algorithm with a couple different kernels. Use the following Matlab code to generate training data for a nonlinear regression problem.

```
>> rand('state',0);
>> randn('state',0);
>> n = 100;
>> x = rand(1,n); % training inputs
>> y = sin(9*x) + sqrt(1/3)*randn(1,n); % training outputs
>> t = linspace(0,1,1001); % test inputs
```

   In the definition of the training outputs, the first term corresponds to the true function, while the second term is noise.

   You will be asked to implement two different methods for nonlinear regression. For each method, please turn in a (sub)plot showing the graphs of the true function and estimated functions, evaluated at the test points, as well as the original data. To avoid crowding, please generate a separate plot for each method, with one method per plot. Include a legend. For the second method, you may manually tune the bandwidth parameter until the estimate looks reasonable.

   For each method, report the mean squared error computed using the test points and the selected value of the bandwidth σ for the second method.

   (a) Implement kernel ridge regression using an inhomogeneous polynomial kernel of degree four, namely, $k(u, v) = (\langle u, v \rangle + 1)^4$.

   (b) Implement kernel ridge regression using the Gaussian kernel.
   Useful commands: `legend, hold on, plot, scatter`.

2. **SVMs for Handwritten Digit Classification**

   Download the file `mnist_49_3000.mat` from the course website and load it into Matlab. You can read more about this database at `http://yann.lecun.com/exdb/mnist/`. There are 3000 examples total, roughly half from each class.

   In this problem you will classify this data with support vector machines. Use the first 2000 data points for designing a classifier, and the last 1000 to compute the test error. Since SVMs involve a tuning parameter C, you will need to set this parameter. To do this, use the

"holdout" method: Decide on a finite set (sometimes called a "grid") of parameter values. For each of these values, train the SVM on the first 1000 data points, and estimate the error (called the "holdout error") using the rest of the training data that was held out. Select the parameter (or parameter combination, in the case of multiple parameters) having the lowest holdout error. You can use the `logspace` command to generate a logarithmic grid of values. *Note:* Once you have selected C, it may be tempting to "retrain" on all 2000 points. In this problem, however, you should only use the second batch of 1000 for selecting C.

To solve the SVM dual, I am providing you with the program `smo.m`, which implements the SMO algorithm described in class. Use this program, do not find other SVM software. The program has a tolerance parameter (the last argument) that specifies the stopping condition. I might suggest a value around 0.001 or 0.0001. Note that smaller values lead to slower convergence. Type `help smo` for details.

(a) Train an SVM using the kernels $k(x, x') = (\langle x, x' \rangle + 1)^p$, $p = 1, 2$, that is, the inhomogeneous linear and quadratic kernel. What is the test error in each case? For each kernel, report the test error and the number (out of 1000) of data points that are support vectors. The commands `find, sum, .*, .^` are helpful. Turn in your code.

(b) For each kernel, turn in a 4×5 subplot showing images of the 20 support vectors that violate the margin by the greatest amount (these are, in a sense, the "hardest" examples to classify), and explain how these are determined. Comment on the differences between the two kernels. The `sort` command will be helpful.

Above each subplot, using the `title` command, indicate the true label (4 or 9). To plot the i-th data point as an image, issue these commands:

```
>> imagesc(reshape(x(i,:),28,28)')
>> colormap(gray)
>> axis square
>> axis off
>> if y(i) == -1
>>      title('4')
>> else
>>      title('9')
>> end
```

(c) I hope you found that the linear kernel required a larger parameter C than the quadratic kernel. Can you explain why this might be?

3. **SVM with squared hinge loss**

Consider the following variation on the soft-margin hyperplane:

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{2n}\sum_{i=1}^{n} \xi_i^2 \\ \text{s.t.} \quad & y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, \text{for } i = 1, 2, \ldots, n \\ & \xi_i \geq 0, \text{for } i = 1, 2, \ldots, n \end{aligned}$$

where $C > 0$. Now we are penalizing margin violations quadratically instead of linearly. Let's call this hyperlane the soft-margin hyperplane with squared hinge loss.

(a) Argue that the constraints $\xi_i \geq 0$ can be dropped without affecting the solution.

(b) Determine the dual QP. Your result should look fairly similar to the dual of the soft-margin hyperplane with hinge loss.

(c) Argue that the above method can be kernelized, thus yielding a new method for nonlinear classification, which we could call the SVM with squared hinge loss.

(d) Explain how b^* can be recovered from the dual solution.

(e) Given an inner product kernel k, find an inner product kernel k' such that the SVM with squared hinge loss with kernel k have the same objective function.

(f) Is the SMO algorithm applicable for solving the dual? If not, can you suggest a simple alternative optimization strategy?