Homework 7

36-350, Data Mining, Fall 2009

Due at the start of class, 6 November

Do not report any coefficients, p-values, etc., to more significant figures than you can justify.

1. Consider two random variables: X is uniformly distributed from -1 to $+1$; $Y = \tanh 5X + \epsilon$, where ϵ is Gaussian white noise with mean 0 and standard deviation 0.1.

   (a) What is the conditional expectation function ($r(x) = \mathbf{E}[Y|X = x]$)?

   (b) Plot the conditional expectation function. Include your code. (*Hint:* try using the `curve` function.)

   (c) Write a function which randomly generates n (X, Y) pairs and returns them in an $n \times 2$ matrix or data frame.

   (d) Test your simulation function by checking the marginal distribution of X values it gives.

   (e) Test your simulation function by making scatter-plots of the values it gives and checking that they fall near the conditional expectation curve $r(x)$.

   (f) Test your simulation function by plotting the residuals $Y - r(X)$ against X, and checking that the mean is near zero and the standard deviation is near 0.1 everywhere. (You should use a large n for this part.)

2. *Cross-validation for polynomial regression*

   (a) Write a function which takes as arguments an integer n and a proportion p, and returns np distinct numbers between 1 and n inclusive. If np is not an integer, round it down to the nearest integer, unless that would give zero, in which case set it to 1. Test that all the returned numbers are distinct (e.g., by using the `unique` function). *Hint:* try `sample`.

   (b) Write a function to do one cross-validation split on a data frame, fit a polynomial to the training data, and return the mean squared error on the testing data. The function should take as its arguments the data frame, the degree of the polynomial, and the fraction of the data

p to use for training. Include an option to also return the numbers of the rows used for the training data; test that your function works by comparing its output to manually fitting the same polynomial to the same training rows and predicting the same testing rows. *Hint:* look at the examples of polynomial fitting in the code accompanying Lecture 19.

(c) Write a function to estimate the generalization error of a polynomial by k-fold cross-validation. It should take as arguments a data frame, the degree of the polynomial, the training fraction p, and the number of folds k. It should return the average of the k testing MSEs.

(d) Generate a data set of size $n = 100$ from the model in problem 1. For polynomials of orders $d = 0$ through 10, calculate the in-sample MSE, and the cross-validated MSE (with $p = 0.9$, $k = 10$ — standard 10-fold cross-validation). Plot both sets of errors together as a function of d. What order polynomial should you use?

(e) Write a function to select which order of polynomial to use, up to a maximum which you give as an argument, by k-fold cross-validation. Which order does it select when run on the data from the previous part, with standard 10-fold cross-validation?

(f) Fit a polynomial of the order selected in the previous part to the whole data. Plot the estimated regression function $\widehat{r}(x)$ together with the data.

(g) Generate a new sample of size $n = 10^4$. What order of polynomial is selected by cross-validation? Should the selected order approach some limit as $n \to \infty$?

3. *Cross-validation for kernel regression* There are several packages for kernel regression. One of the most powerful ones is the `np` package on CRAN; download and install it.[1] The command

```
npreg(y~x1+x2,bws=c(0.1,2),ckertype="gaussian",data=big.frame)
```

regresses the variable y on x_1 and x_2, all values taken from the data frame `big.frame`. It uses a Gaussian kernel, with bandwidth 0.1 for the first input variable and bandwidth 2 for the second. The output is an object, like the output from `lm`, which you can print, plot, call `predict` on, etc. This problem develops cross-validation for bandwidth selection. The package actually includes a *very* sophisticated function, `npregbw`, to do bandwidth selection, but this way builds character.

(a) Write a function to split a data frame into training and testing sets, fit a Gaussian kernel regression with a given bandwidth to the training set, and return the mean squared error on the testing set. The

[1] There is a detailed description of the package online at `http://www.jstatsoft.org/v27/i05`, which may be helpful but is not required reading.

function should take as arguments the data frame, the bandwidth h, and the fraction of data p to go into the training set.

(b) Write a function to calculate the k-fold cross-validated MSE of a given bandwidth. The function should take as arguments the data frame, the bandwidth h, the training fraction p, and the number of folds k, and return the MSE.

(c) For the $n = 100$ data set from the previous problem, plot the in-sample and ten-fold cross-validated MSEs for $h = 0.05, 0.1, 0.15, \ldots 1.0$.

(d) Write a function to select the best bandwidth by cross-validation. It should take the same arguments as before, except that it needs a vector of bandwidths and not a single bandwidth; return the best bandwidth (and not its error or its position in the vector of options). Test that when given the same bandwidths as in the previous part, it selects the best.

(e) Plot the estimated regression function $\widehat{r}(x)$ from the best bandwidth along with the data. How does this differ from the selected polynomial?

(f) Repeat the cross-validated bandwidth selection with the $n = 10^4$ sample. (Allow at least 20 seconds of computing time per fold per bandwidth.) Does the selected bandwidth change? Plot the regression function with the selected bandwidth. How does this differ from the selected polynomial? From the true $r(x)$?