Math Foundations of ML, Fall 2018

Homework #5

Due Friday October 12, at the beginning of class

As stated in the syllabus, unauthorized use of previous semester course materials is strictly prohibited in this course.

1. Using you class notes, prepare a 1-2 paragraph summary of what we talked about in class in the last week. I do not want just a bulleted list of topics, I want you to use complete sentences and establish context (Why is what we have learned relevant? How does it connect with other things you have learned here or in other classes?). The more insight you give, the better.

2. In this problem, we will solve a stylized regression problem using the data set `hw5p2_clusterdata.mat`. This file contains (noisy) samples of a function $f(t)$ for $t \in [0,1]$. The sample locations are in the vector `T`, the sample values are in `y`. If you plot these, you will see that the samples come in four clusters. We are going to use nonlinear regression using the orthobasis of Legendre polynomials — in the notes, these polynomials were defined on $[-1,1]$, we will use the analogous functions on $[0,1]$. To compute these, we can use the `legendreP` command in MATLAB[1]. If we define the function handle

   ```
   lpoly = @(z,p) sqrt(2)*sqrt((2*p+1)/2)*legendreP(p, 2*z-1);
   ```

   Then (for example) `lpoly(3,T)` will return samples of the 3rd order Legendre polynomial at locations in `T`. This command is unfortunately a but slow, but it will serve our purposes here.

   (a) Find the best cubic fit to the data using least-squares. That is, find $w_0, \ldots, w_3$ that minimizes

   $$\underset{\boldsymbol{w}}{\text{minimize}} \quad \sum_{m=1}^{M} (y_m - f(t_m))^2 \quad \text{where} \quad f(t) = \sum_{n=0}^{3} w_n v_n(t),$$

   where $v_n(t)$ is the nth order Legendre polynomial adapted to $[0,1]$. Let $\hat{\boldsymbol{w}}$ be the solution to the above, and $\hat{f}$ the corresponding cubic polynomial. Compute the *sample error*[2]

   $$\left(\sum_{m=1}^{M} (y_m - \hat{f}(t_m))^2 \right)^{1/2} = \|\boldsymbol{y} - \boldsymbol{A}\hat{\boldsymbol{w}}\|_2,$$

   where $\boldsymbol{A}$ is the matrix you set up to solve the least-squares problem. Plot your solution $\hat{f}(t)$ for $t \in [0,1]$, and overlay the sample values (t_m, y_m) — the sample values should not have lines connecting them[3].

[1] For Python, see https://docs.scipy.org/doc/scipy/reference/generated/scipy.special.legendre.html.

[2] Also called "training error".

[3] Use `plot(t,y,'o')` in MATLAB, for example.

(b) In this case, the function I happened to use to create the samples is

$$f_{\text{true}}(t) = \frac{\sin(12(t+0.2))}{t+0.2}.$$

Compute the (squared) generalization error

$$\left(\int_0^1 |\hat{f}(t) - f_{\text{true}}(t)|^2 \ \mathrm{d}t \right)^{1/2}.$$

You can either use some numerical integration package to do this (`integral()` in MATLAB), or you can simply sample the functions at 5000 points[4], take the sum of the squared difference, divide by 5000, then take the square root.

(c) Repeat parts (a) and (b) for polynomials of order $p = 5, 10, 15, 20, 25$ (note that the number of basis functions is always $N = p+1$). For each experiment, report the sample error, generalization error, and the largest and smallest singular value of the $\boldsymbol{A}$ matrix. Make a plot of the sample error versus N and the generalization error versus N. For what value of N does least-squares start to fall apart and why? Why does the sample error go down monotonically with N but the generalization error does not? Answer these questions, and for each value of p, make a plot of the sample values, $\hat{f}(t)$, and $f_{\text{true}}(t)$ overlaid on one another.

(d) Set $N = 25$. Plot the singular values of $\boldsymbol{A}$ for this N. Stabilize your least-squares solution by truncating the SVD; make a judgement call about what R' should be given your singular value plot, and explain your reasoning. Compute the sample and generalization errors.

(e) Again with $N = 25$, repeat part (d) and compute the truncated SVD estimate for $R' = 5, 6, \ldots, 25$. Plot the sample and generalization error versus R', and interpret in terms of noise error, approximation error, and null space error.

(f) Fix $N = 25$. Now we will consider the ridge regression estimate

$$\underset{\boldsymbol{w}}{\text{minimize}} \ \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{w}\|_2^2 + \delta \|\boldsymbol{w}\|_2^2$$

Again examining the plot of the singular values of $\boldsymbol{A}$, come up with a reasonable value of δ to use above. Perform the regression, make a plot of your result (again overlaid on the samples and $f_{\text{true}}(t)$, and report the sample and generalization error. Also comment on what happens to both the sample and generalization error as you sweep the value of δ — provide representative plots.

3. In this problem, we will solve a stylized regression problem using the data set `hw5p3_scatterdata.mat`. As before, this file contains (noisy) samples of a function $f(t)$ for $t \in [0, 1]$. The sample locations are in the vector `T`, the sample values are in `y`. If you plot these, you will see that the samples are scattered more or less evenly across the interval. We are going to use kernel regression to form the estimate; in particular, we will use

$$k(s,t) = e^{-|t-s|^2/2\sigma^2}.$$

The $f_{\text{true}}(t)$ is the same as in the last problem.

[4] 5000 might be overkill here, but these computations are cheap.

(a) Compute the kernel regression estimate with $\sigma = 1/10$ and $\delta = 0.004$. Plot you estimate overlaid on the data and $f_{\text{true}}(t)$. Compute the sample and generalization errors. Comment on why this choice of δ was a good one.

(b) Repeat part (a) with $\sigma = 1/2, 1/5, 1/20, 1/50, 1/100, 1/200$, producing plots and error estimates for each σ. Comment on how the number of data points we see would affect the right choice of σ.

4. Consider the set of bump basis vectors $\psi_1(t), \ldots, \psi_N(t)$, where

$$\psi_k(t) = g\left(t - k/N\right), \quad g(t) = e^{-200t^2} \tag{1}$$

Given a point t, define the nonlinear "feature map" as

$$\boldsymbol{\Psi}(t) = \begin{bmatrix} \psi_1(t) \\ \psi_2(t) \\ \vdots \\ \psi_N(t) \end{bmatrix}$$

Plot the feature map as a discrete set of coefficients[5] for $t = 1/3$ for $N = 10, 20, 50, 100, 200$. Compare to the radial basis kernel map

$$\boldsymbol{\Phi}_t(s) = k(s,t) = e^{-100|s-t|^2},$$

for $t = 1/3$ and $s \in [0,1]$. Discuss the relationship between kernel regression with a Gaussian radial basis function, and nonlinear regression using a basis of the form (1).

5. Let $\boldsymbol{A}$ be a banded tri-diagonal matrix:

$$\boldsymbol{A} = \begin{bmatrix} d_1 & c_1 & 0 & 0 & 0 & \cdots & 0 \\ f_1 & d_2 & c_2 & 0 & 0 & \cdots & 0 \\ 0 & f_2 & d_3 & c_3 & 0 & \cdots & 0 \\ 0 & 0 & f_3 & d_4 & c_4 & \cdots & 0 \\ \vdots & \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & & f_{N-2} & d_{N-1} & c_{N-1} \\ 0 & 0 & 0 & \cdots & & f_{N-1} & d_N \end{bmatrix}$$

(a) Argue that the LU factorization of $\boldsymbol{A}$ has the form

$$\boldsymbol{A} = \begin{bmatrix} * & 0 & 0 & 0 & \cdots & 0 \\ * & * & 0 & 0 & \cdots & 0 \\ 0 & * & * & 0 & \cdots & 0 \\ 0 & 0 & * & * & \cdots & 0 \\ \vdots & & & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & & * & * \end{bmatrix} \begin{bmatrix} * & * & 0 & 0 & 0 & \cdots & 0 \\ 0 & * & * & 0 & 0 & \cdots & 0 \\ 0 & 0 & * & * & 0 & \cdots & 0 \\ \vdots & & & & \ddots & \ddots & \\ 0 & 0 & \cdots & & & * & * \\ 0 & 0 & \cdots & & & 0 & * \end{bmatrix},$$

where $*$ signifies a non-zero term.

[5] In MATLAB, use `plot(1:N,Psit(1:N),'o')`.

(b) Write down an algorithm that computes the LU factorization of $\boldsymbol{A}$, meaning the $\{\ell_i\}, \{u_i\}$, and $\{g_i\}$ below

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ \ell_1 & 1 & 0 & 0 & \cdots & 0 \\ 0 & \ell_2 & 1 & 0 & \cdots & 0 \\ 0 & 0 & \ell_3 & 1 & \cdots & 0 \\ \vdots & & & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & & \ell_{N-1} & 1 \end{bmatrix} \begin{bmatrix} u_1 & g_1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & u_2 & g_2 & 0 & 0 & \cdots & 0 \\ 0 & 0 & u_3 & g_3 & 0 & \cdots & 0 \\ \vdots & & & & \ddots & \ddots & \\ 0 & 0 & \cdots & & & u_{N-1} & g_{N-1} \\ 0 & 0 & \cdots & & & 0 & u_N \end{bmatrix},$$

I will get you started:

$u_1 = d_1$
for $k = 2, \ldots, N$
    $g_{k-1} =$
    $\ell_{k-1} =$
    $u_k =$
end

(c) What is the computational complexity of the algorithm above? (How does the number of computations scale with N?) Once the LU factorization is in hand, how does solving $\boldsymbol{Ax} = \boldsymbol{b}$ scale with N?