Homework 12
MATH 504

1. - Section 5.4 in Elements of Statistical Learning covers spline fitting with penalty terms.
   - Sections 12.3 and 12.4 in Sauer cover SVD.

2. This problem repeats the spline regression problems of hw 11, but now we will use many knots. Specifically, consider the case 1000 knots equally spaced between the minimum and maximum x values (ages) of the dataset. As before, for this problem let's consider solely the female portion of the BoneMass dataset (file attached).

   B-splines are a choices of basis functions for splines that produce design/model matrices that are not badly conditioned. In the setting of 1000 knots, such a basis is essential. Choosing basis functions of the form $h_i(x) = ([x - \zeta_i]^+)^3$ as in the last homework will lead to huge condition numbers and the regression will not be possible. This is an example of the importance of a good basis in computation.

   To manipulate B-splines, R provides the function **splineDesign** as part of the **splines** package. Here are the two ways you will need to call **splineDesign** to apply spline regression with a penalty method.

   ```
   B <- splineDesign(knots=myknots, x=x, outer.ok=T)
   Bpp <- splineDesign(knots=myknots, x=mygrid, derivs=2,
   outer.ok = T)
   ```

   Above, B will be a design matrix, meaning that $B_{ij} = b_j(x_i)$ where $b_j(x)$ is the jth spline function in the B-spline basis and x_i is the ith x sample from the dataset. *Bpp* is similar, except that given the flag **derivs=2**, $Bpp_{ij} = b_j''(x_i)$ (i.e. the second derivative of $b_j(x)$). See below for the meaning of **mygrid**. Note: Using **splineDesign**, the dimension of the spline space is $K-4$, where K is the number of knots, rather than our usuual $K+4$. **splineDesign** places some constraints on the behavior of the splines at the end points that restrict the dimension to $K-4$ rather than $K+4$. This issue is minor and does not change the computations that must be done.. B will have dimension N by $K-4$ where N is the dimension of x. *Bpp* will have $K-4$ columns, but the number of rows will depend on the number of values in mygrid.

   (a) Consider minimizing the following penalized least squares, which we discussed in class

   $$\min_\alpha \sum_{i=1}^{N} |y_i - \sum_{j=1}^{K-4} \alpha_j b_j(x_i)|^2 + \rho \int_{x_{\min}}^{x_{\max}} \left((\sum_{j=1}^{K-4} \alpha_j b_j(z))'' \right)^2 dz \qquad (1)$$

   Show that this reduces to calculating

   $$\min_\alpha \|y - B\alpha\|^2 + \rho \alpha^T \Omega \alpha \qquad (2)$$

where B is precisely the matrix returned by **splineDesign** above and where Ω is a $K-4$ by $K-4$ matrix given by

$$\Omega_{k\ell} = \int_{x_{\min}}^{x_{\max}} b_k''(z) b_\ell''(z) dz \tag{3}$$

Show that the solution to the minimization is given by

$$\alpha = (B^T B + \rho\Omega)^{-1} B^T y \tag{4}$$

(b) Use R to calculate Ω through numerical integration. The matrix Bpp gives values of $b_j''(z)$ on a grid of z values. The vector **mygrid** (in code for Bpp above) gives the z values at which b_j'' is calculated. For simplicity, calculate Ω using Riemann integration, meaning that you will make the following approximation:

$$\int_{x_{\min}}^{x_{\max}} b_k''(z) b_\ell''(z) dz \approx \sum_{i=1}^{M} b_k''(z_i) b_\ell''(z_i) \Delta z \tag{5}$$

where the z_i are the points you specify in **mygrid**, $b_\ell''(z_i)$ is the i, ℓ entry of Bpp, and Δz is the distance between the points in **mygrid**. Make sure **mygrid** forms a dense grid of values so that you get accurate estimates using Riemann integration.

(c) Show that the matrix $B^T B$ is not invertible but that $B^T B + \rho\Omega$ is for $\rho > 0$. You can do this by just computing the determinant for various values of ρ in R or through theoretical arguments.

(d) Calculate α for (a) $\rho = .01$, (b) $\rho = 1$ and (c) $\rho = 100$. In each case plot the smoothing spline (hint: B provides you discretized versions of the $b_j(x)$, linearly combine them using α to form the fitted spline) and the data points. Comment on the fit in each case.

3. This problem will demonstrate an application of svd. In the problem below, the rows of the A matrix are formed from a single vector, with some added noise. The **rank** of a matrix is the dimension of the span of its row or column vectors (it turns out that the dimension is the same for row span and column span). The noise means that the A matrix has a 10 dimensional **rank**, but without the noise the A matrix would have a 1 dimensional **rank**. The svd allows us to compute the lower rank matrix and extract the underlying signals from A.

   (a) The script `signals.R` constructs a 500×10 matrix, A, which is saved to the file `A.txt`. Each row is of A has the form (q*sig + noise). sig is a fixed signal, where the signal is a 10 dimensional vector. Also saved, in the file `no_noise_A`, is a matrix with rows given by q*sig (i.e. no noise). Finally, the file `q` gives the q value used for each row. Look through `signals.R` and make sure you understand how A is constructed and the from of sig. Plot the values in the first row of A, the first row of `no_noise_A`, and the underlying signal. Can you tell what the signal is by looking at A? By averaging the columns of A?

(b) Perform an svd on A using R's **svd** function. Plot the singular values and comment on their values given what we know about A. Consider the approximation A_1 of A, where $A_1 = s_1 u^{(1)}(v^{(1)})^T$. Use `image(A)`, `image(no_noise_A)` and `image(A_1)` to visualize the three matrices and confirm that A_1 removes the noise from A. Compare the first row of A, `no_noise_A` , and A_1. Given a row of A in the form q*sig + noise, what role do $v^{(1)}, s_1$ and $u^{(1)}$ have in capturing q and sig?