ANLY-601 Spring 2018

Assignment 7 — Revised for Clarity
Due May 7, 2018

You may use the text, your lecture notes and handouts, and any mathematics references that you need. You may ask me for clarification but you may not consult other people or other references (including the internet). If you use Mathematica for any of this, be sure to include the notebook in your solution so I can see what you did. Better yet, cut and paste from the Mathematica notebook into your solution so it all flows logically.

1. (10 points) **KL Divergence and Log-Likelihood**

   The Kulback-Liebler divergence is one of several measures of distance between *probability distributions*. Consider a true distribution $p(x)$ and a model $\hat{p}(x|\theta)$ with parameters θ, then the KL divergence between them is

$$\begin{aligned} d(p,\hat{p}) &\equiv -\int p(x)\,\ln\left(\frac{\hat{p}(x|\theta)}{p(x)}\right)\,dx \\ &= -\int p(x)\ln\hat{p}(x|\theta)\,dx + \int p(x)\,\ln p(x)\,dx \;\geq 0 \end{aligned} \tag{1}$$

   where the equality (in ≥ 0) is met if and only if $\hat{p}(x|\theta) = p(x)$. This exercise will show you how the KL divergence is related to data log-likelihood used in model fitting.

   Suppose you have a data set consisting of m samples $D = \{x_1, x_2, \ldots, x_m\}$ each of which is a scalar ($x \in R$). (Note, the subscript identifies the *sample*.) Assuming that these samples are statistically independent, the *log-likelihood of the data set* under the model is

$$L \;=\; \ln\hat{p}(D|\theta) \;=\; \sum_{a=1}^{m} \ln\hat{p}(x_a|\theta) \quad . \tag{2}$$

   Show that the *expected* log-likelihood of the dataset under the model is

$$E_D[L] \;=\; m\int p(x)\,\ln\hat{p}(x|\theta)\,dx \tag{3}$$

   where the expectation $E_D[\cdot]$ is respect to the distribution over all possible data sets. (To receive full credit, you must write E_D out as an integral over the distribution of data sets, and evaluate the integral.)

   Conclude that the KL divergence is

$$d(p,\hat{p}) \;=\; -\frac{1}{m}\,E_D[L] \;-\; H_p$$

   where the *differential entropy* of the true distribution is

$$H_p \;\equiv\; -\int p(x)\,\ln p(x)\,dx \quad .$$

   (Notice also that since the KL divergence is bounded below by zero, the expected log-likelihood is bounded above — a theoretical fitting bound $\frac{1}{m}\,E_D[L] \leq -H_p$.)

2. (10 points) **Interpolating and Smoothing Kernels**

In class we developed kernel density estimates. One can also use a kernel approach for regression, and this is closely related to Gaussian process regression and kriging (in geostatistics).

Suppose you have a data set consisting of (real) input/output pairs $(x_a, y_a), a = 1, \ldots N$. One can build an interpolating kernel model in a way very similar to a kernel density estimate. We will use symmetric kernels, as before

$$\kappa(x - y) \;=\; \kappa(y - x) \quad .$$

A common choice is the radial exponential kernel

$$\kappa(x - y) \;=\; \exp\left(-\frac{|x - y|^2}{r^2}\right)$$

where r is the (adjustable) kernel radius. (But you do not need to consider the kernel form in what follows.)

Our kernel model of $f(x)$ is

$$\hat{f}(x) \;=\; \sum_{a=1}^{N} c_a\, \kappa(x - x_a) \tag{4}$$

where the coefficients c_a need to be specified.

(a) **Find the coefficients** by minimizing the MSE cost

$$\mathcal{E}(c) \;=\; \frac{1}{N}\sum_{a=1}^{N}\Big(y_a \;-\; \hat{f}(x_a)\Big)^2 \;=\; \frac{1}{N}\sum_{a=1}^{N}\left(y_a \;-\; \sum_{b=1}^{N} c_b\, \kappa(x_a - x_b)\right)^2 \tag{5}$$

with respect to c_a, $a = 1, \ldots N$. That is, set

$$\frac{d\,\mathcal{E}}{dc_k} \;=\; 0$$

and solve for the c's. You should find

$$c_a \;=\; \Big(K^{-1}\, y\Big)_a \;=\; \sum_{b=1}^{N}\Big(K^{-1}\Big)_{ab}\; y_b \tag{6}$$

where K is the matrix with elements $K_{ab} \;=\; \kappa(x_a - x_b)$, K^{-1} is its matrix inverse, and y is the (column) vector with elements y_a, $a = 1, \ldots N$. The complete kernel interpolator is then

$$\hat{f}(x) \;=\; \sum_{a=1}^{N} c_a\, \kappa(x - x_a) \;=\; \sum_{a=1}^{N}\Big(K^{-1}\, y\Big)_a\; \kappa(x - x_a) \quad . \tag{7}$$

(b) **Show** that at each input point x_a, the interpolator passes through the corresponding output point $\hat{f}(x_a) = y_a$. (This form of interpolator assumes that the y_a are *noiseless* measurements of the underlying generating function $y_a = f(x_a)$.)

Since I've given you the answers, to get full credit your solution must *show all steps*. I suggest that your use component notation for all your matrix manipulations to be clear about the algebra.

(Note: If the data points y_a are noisy measurements of the generating function $y = f(x) + \epsilon$, $\epsilon \sim N(0, \sigma^2)$, the coefficients take the form $c = (K + \sigma^2 I)^{-1} y$ instead. The resulting model is called a *smoothing kernel*, and it does not pass through all the y_a. This form makes full contact with Gaussian process regression and smoothing splines.)

3. (10 points) **Bootstrap Variance Estimate**

Unlike the sample mean, one cannot write a closed-form expression for the variance of the sample *median* across data sets drawn from a distribution. This exercise will have you construct a bootstrap estimate for the variance of the sample median.

On the web page is a new dataset `dataForBootstrap.txt` that has 50 samples from a distribution. Calculate the median of the sample using whatever package you prefer. Next generate 10 bootstrap replicates. Calculate the median of each. Then compute from these 10 estimates of the median, the bootstrap variance of the median estimate. Repeat for

$$[10, 30, 100, 300, 1000, 3000, 10000, 30000]$$

replicates and report the variance for each case. Note that if your original data set has n samples, there are

$$\binom{2n-1}{n}$$

possible different bootstrap replicates; so you don't have to worry about running out.

Interpret your results. I suggest making a plot of the measured variance (of the median) as a function of the number of replicates using a log scale for the x-axis (number of replicates). What value would you quote for the variance of the median estimate ?

4. (25 points) **Overfitting in K-Means**

We discussed both soft clustering (with Gaussian mixture models fit by EM), and hard clustering (with K-means), and the relation between the former and latter algorithms. For the last homework assignment you fit Gaussian mixture models to a small, two-dimensional dataset, examined the log-likelihood on fitting and holdout sets, noting overfitting with increasing model sizes. (Although the log-likelihood per data point for the *fitting* data increases monotonically with increasing number of Gaussian bumps in the mixture model, the log-likelihood increases, then decreases on the holdout data.) This exercise has you explore the same dataset and concepts for K-Means.

Using any language or package you prefer, write or use an existing K-means algorithm (with Euclidean distance function) and fit a sequence of models to the first 750 vectors in the *toydata1.txt* dataset. Plot the mean square distance between the data points and the centroids of the cluster they belong to (the cost function minimized by K-means)

$$J = \sum_{i=1}^{K} \left(\frac{N_i}{N}\right) \frac{1}{N_i} \sum_{a=1}^{N_i} |x_a^{(i)} - m_i|^2$$

as a function of K. (Here K is the number of clusters, N is the total number of data points, N_i is the number of data points in cluster i, $x_a^{(i)}$ is the a^{th} data point in cluster i, and m_i is the centroid (or mean) of the data in the i^{th} cluster — see Lecture 13.)

Make such plots for both the *fitting data* (the first 750 data points) and the *test data* (the remaining 750 points). To facilitate comparing the curves of J vs L, plot both the training and test set curves on the same frame. Explore numbers of clusters K in the range 5–100. (You don't have to make models for each value in between — but use enough so you can see what's happening. I used 5,10,20,30,40,50,60,100 — but this is by no means a preferred set.) Since the K-means is a greedy algorithm (moves downhill in J), and the cost function has suboptimal local minima that the algorithm can converge to, it's a good idea to run several re-starts (with different initializations of the m_i) for each value of K. This will ensure a nice monotonically decreasing curve of J vs K on the training set. (I used 10 restarts for each L, and kept the best fit on the *training* data for each — running the test data through the corresponding models.)

(a) (10 points) For models fit to the training data, plot the cost J vs the number of clusters (means) K for both the training and test data. How do these plots compare (qualitatively) to what you found for the Gaussian mixture model, and what you're familiar with for regression and classification problems in general?

(b) (10 points) WHY does the K-means algorithm behave so different with regard to overfitting? Use diagrams and words to describe *clearly* what's going on with K-means in this regard. (It is your job to be clear — this is both an exercise in reasoning about the cost function J, and communicating your thoughts.)

(c) (5 points) Re-do the plots from part (a) using a *log scale* for both axes. (It doesn't matter if you use log base 10 or base 2 or base e, but *be clear* which you use.) You will see a roughly straight line for the training data curve.

$$\ln J \;=\; \ln b \;-\; m \ln L \quad .$$

Note that this is equivalent to

$$J \;\approx\; \frac{b}{L^m}$$

a power law for J vs L. Find b and m using the fitting curve values with $L \geq 20$ (using a linear fit for the log data). You should see $m \approx 2/D$ where D is the true dimensionality of the data.

(d) (10 points EXTRA CREDIT) Use simple mathematical arguments (dimensional analysis) to show that you should find $m = 2/D$. (Hints: The mean square error is proportional to the square of cluster length $J \propto r^2$, and the volume of each cluster is $\text{Vol}_{\text{cluster}} \approx \text{Vol}_{\text{data set}}/L$, while on dimensional grounds the volume of each cluster is $\text{Vol}_{\text{cluster}} \propto r^D$.)

Would you like an extra credit computational problems in either kernel regression or neural networks?