Assignment 2 - CSC/DSC 265/465 - Spring 2017 - Due March 7

Q1. Suppose ϕ is a true density function on $\mathbb{R}$.

1. For $a \in \mathbb{R}$, $b > 0$, show that $\phi_{a,b}(x) = b^{-1}\phi((x-a)/b)$ is also a density function on $\mathbb{R}$.

2. Let $x_1, \ldots, x_n$ be any numbers, and let $h > 0$. Verify that f_h defined as

$$f_h(x) = \frac{1}{nh} \sum_{i=1}^{n} \phi\left(\frac{x - x_i}{h}\right)$$

is a density function. Then f_h is an approximation of the density from which $x_1, \ldots, x_n$ were sampled.

3. Let (x_i, y_i) be paired observations from model

$$y_i = g(x_i) + \epsilon_i, \;\; i = 1, \ldots, n, \tag{1}$$

where the ϵ_i are any zero mean error terms. The Nadaraya - Watson kernel regression estimate $\hat{y}_h(x) \approx g(x)$ is given by

$$\hat{y}_h(x) = \frac{\sum_{i=1}^{n} y_i K_h(x - x_i)}{\sum_{i=1}^{n} K_h(x - x_i)}, \tag{2}$$

wherever the expression is defined. Usually, $K_h(z) \propto \phi(z/h)$ where ϕ is a zero mean density, and $h > 0$ is the *bandwidth*.

   (a) Suppose $\phi(z) = I\{|z| \leq 1/2\}$, that is, the uniform density on interval $[-1/2, 1/2]$. Set $K_h(z) = \phi(z/h)$ in estimator (2). Let $N_h(x)$ be the set of indices from $\{1, \ldots, n\}$ defined by

$$N_h(x) = \{i : |x - x_i| \leq h/2\},$$

   and denote cardinality $n_h(x) = |N_h(x)|$. Express the $\hat{y}_h(x)$ explicitly in terms of (x_i, y_i), $i = 1, \ldots, n$, making use of $N_h(x)$ and $n_h(x)$.

   (b) Suppose the error terms ϵ_i in (1) are an *iid* sample from $N(0, \sigma^2)$. Give an explicit expression for the variance $V_h(x)$ and bias $B_h(x)$ of $\hat{y}_h(x)$.

   (c) We next consider a specific model. In (1) let

$$g(x) = \beta_0 + \beta_1 x$$

   on some interval $x \in [-M, M]$, $M > 0$, for two constants $\beta_0, \beta_1 \neq 0$. A *Poisson process* of rate λ on any interval $\mathcal{I} \subset \mathbb{R}$ is a random set of points $\mathcal{X} \subset \mathcal{I}$ which possesses the following properties (in addition to others):

      (i) The number of points N' from $\mathcal{X}$ in interval $[a, a+h] \subset \mathcal{I}$ has a Poisson distribution with mean λh.

      (ii) Given that there are N' points from $\mathcal{X}$ in interval $[a, a+h]$, these points form an *iid* sample from a uniform distribution on $[a, a+h]$.

   (See Section 7.1 from CSC/DSC 262/462 lecture notes). Then assume the predictor values $\mathcal{X} = (x_1, \ldots, x_N)$ are generated by a Poisson process on $[-M, M]$ with rate λ (this means that N is a Poisson random variable with mean $2M\lambda$). Then, once $\mathcal{X}$ is generated, responses y_i are given by

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i, \;\; i = 1, \ldots, N,$$

   where the error terms ϵ_i are an *iid* sample from $N(0, \sigma^2)$.

In general, the conditional expected value of X given event A, denoted $E[X \mid A]$, is the expected value of X under the distribution $P(X \in E \mid A)$. Accordingly, the quantity $E[B_h(x)^2 \mid n_h(x) = n]$ is the expected value of $B_h(x)^2$ calculated after assuming that $n_h(x) = n$, so that the property (ii) of a Poisson process given above may be applied. Then calculate

$$E[MSE_h(x) \mid n_h(x) = n] = E[V_h(x) \mid n_h(x) = n] + E[B_h(x)^2 \mid n_h(x) = n].$$

You can assume that M is large enough so that $[x - h/2, x + h/2] \subset [-M, M]$.

(d) Write an `R` program to calculate $E[MSE_h(x) \mid n_h(x) \geq 1]$ (note that $MSE_h(x)$ is not defined unless $n_h(x) \geq 1$). Set $\beta_1 = 1$, $\lambda = 10$. Fix $\sigma = 0.25$, and calculate $E[MSE_h(x) \mid n_h(x) \geq 1]$ for h on a grid defined by `seq(0.001,10,0.001)`. Plot $E[MSE_h(x) \mid n_h(x) \geq 1]$ against h, and identify h which minimizes $E[MSE_h(x) \mid n_h(x) \geq 1]$. Repeat for $\sigma = 1.0$.

HINT: Recall the law of total probability (Section 2.7 from CSC/DSC 262/462 lecture notes). This implies that the expected value of any random variable X may be calculated by:

$$E[X] = E[X \mid A_1]P(A_1) + \ldots + E[X \mid A_m]P(A_m),$$

where $A_1, \ldots, A_m$ are mutually exclusive events which partition the sample space, so that $P(A_1) + \ldots + P(A_m) = 1$. Then

$$E[MSE_h(x) \mid n_h(h) \geq 1] = \sum_{i=1}^{\infty} E[MSE_h(x) \mid n_h(x) = n]P(n_h(x) = n \mid n_h(x) \geq 1). \qquad (3)$$

If $p(n)$, $n = 0, 1, 2, \ldots$ is the PMF of the appropriate Poisson distribution, we have

$$P(n_h(x) = n \mid n_h(x) \geq 1) = p(n)/(1 - p(0)), \;\; n = 1, 2, \ldots.$$

Then $E[MSE_h(x) \mid n_h(h) \geq 1]$ can be evaluated numerically using (3), after combining the answer to part (c) with the `R dpois` function. Make sure that enough terms are included in any approximate summation of (3) to avoid truncation error.

Q2. For this problem use data set `Auto` from the `ISLR` package, which contains MPG; number of cylinders; model year; origin of car; and other information for 392 vehicles. Suppose a given application requires a prediction as to whether or not the model year of a vehicle is 1975 or later, bases on minimum technical specifications. The reasoning here is that a very low or very high MPG may be enough to give an accurate prediction (if not, more information would be collected).

(a) Create a new data frame containing an indicator function `Y` which equals 1 for model year ≥ 1975. Also retain only vehicles for which `origin == 1` (that is, we will only consider American cars).

(b) Fit a logistic regression model with binary response `Y` and predictor variable `mpg`. Report the standard coefficient table (ie. for each $\hat{\beta}_0$, $\hat{\beta}_1$ there are columms for Estimate, Standard Error, Z-score and P-value).

(c) Create a graph with the following elements.

(i) Each observed pair (mpg_i, Y_i) is plotted separately (the option `pch=3` works well for this).

(ii) The function $f(mpg) = E[Y \mid mpg] = P(1975+ \mid mpg)$ is plotted. Use the `predict()` function. This has several advantages. First, we don't need to rely on the fitted values to construct the plot, since `predict()` will calculate fitted values for any predictor values, using the `newdata` option (use a list or data frame with consistent variable names). So, we can use an evenly spaced grid for the `mpg` axis. Second, we can get standard errors for the fits using the `se=T` option. This gives approximate 95% confidence bands after adding $\pm 2SE$ to each fitted value. Usually, the confidence bounds are drawn using dashed lines. Use something like the following commands (data in data frame `auto2`) to add the fitted curve to the plot:

```
logistic = function(x) {(1+exp(-x))^(-1)}
fit0 = glm(Y ~ mpg, family='binomial',data=auto2)
mpg.range = seq(min(auto2$mpg),max(auto2$mpg),0.1)
pr = predict(fit0,newdata=list(mpg=mpg.range),se=T)
lines(mpg.range,logistic(pr$fit))
lines(mpg.range,logistic(pr$fit-2*pr$se.fit),lty=2)
lines(mpg.range,logistic(pr$fit+2*pr$se.fit),lty=2)
```

Note that `predict()` gives the linear predictor $\eta = X\beta$. You have to apply the logistic function yourself.

(d) Construct side-by-side boxplots of *mpg* for each of the six combinations of `Y` and `cylinders`. You can use the formula `mpg` $\sim$ `Y*cylinders` inside the `boxplot()` function, as long as you set the `data` option correctly. Examining the boxplot, would $mpg = 20$ be strong evidence that the model year was ≥ 1975 for a 4 cylinder vehicle? What about an 8 cylinder vehicle?

(e) Expand your fit to include the new predictor `cylinder` as a factor. Note that `cylinder` has mode `numeric`, so it must be explicitly converted to a factor. This can be done directly to the data frame, or within the model formula, using `Y` $\sim$ `mpg*as.factor(cylinders)`. This will essentially add two indicator functions to the model:

$$\begin{aligned} \texttt{as.factor(cylinders)6} &= I\{\texttt{cylinders == 6}\} \\ \texttt{as.factor(cylinders)8} &= I\{\texttt{cylinders == 8}\} \end{aligned}$$

These indicator variables will appear as separate terms, and as interactions with `mpg`. These will appear in the resulting coefficient table.

(f) Create a plot with the same axes as that of part (c):

(i) Include the observed pairs (mpg_i, Y_i), but use separate colors or symbols for each cylinder level.

(ii) Plot the same function $f(mpg) = E[Y \mid mpg] = P(1975+ \mid mpg)$ as in part (c) (ie, the original fit with `mpg` only), but without the confidence bounds.

(iii) For each cylinder level identify the `mpg` range.

(iv) Plot the function $f(mpg, cylinders) = E[Y \mid mpg, cylinders] = P(1975+ \mid mpg, cylinders)$ separately for each cylinder level 4,6,8. Use distinct colors or line types. Make sure each of the 4 curves is properly labeled (best to use the `legend()` function for this). No confidence bounds should be drawn.

(v) The `predict()` function should be used as in Part (c). However, for each cylinder level, only use the `mpg` range observed for that level. Note that the `newdata` object will need to include that cylinder level. For example, to draw the fitted curve for level `cylinders == 4`:

```
fit1 = glm(Y ~ mpg*as.factor(cylinders), family='binomial',data=auto2)
range.by.cylinder = tapply(auto2$mpg,auto2$cylinders,function(x) range(x))
mpg.col=2
mpg.type=4
mpg.range = seq(range.by.cylinder$'4'[1],range.by.cylinder$'4'[2],0.1)
ngrid = length(mpg.range)
pr = predict(fit1,newdata=list(mpg=mpg.range,cylinders = rep(mpg.type,ngrid)),se=F)
lines(mpg.range,logistic(pr$fit),col=mpg.col,lwd=2)
```

For `mpg` $= 20$, what is $P(1975+ \mid mpg)$ without knowing the cylinder level? What is $P(1975+ \mid mpg, cylinders)$ for `mpg` $= 20$, for each cylinder level. Does adding `cylinders` improve the prediction accuracy?

Q3. One question to consider when constructing classifiers is whether or not the method is *location-scale invariant*. This property holds if subjecting any subset of features in the training data to a linear transformation cannot change the prediction. For example, a classifier is location-scale invariant if changing the units of a feature from feet to inches, or from degrees fahrenheit to celsius, does not change the predictions (as long as the transformation is consistently applied to all classes and test data).

This is a crucial question, since if a classifier is not location-scale invariant, we need to decide how to make the units of each feature comparable. One way of doing this is to standardize quantitative features, typically by subtracting the mean then dividing by the standard deviation, so that each feature has mean zero and standard deviation 1. Doing this makes them essentially unitless. However, for some applications this may not be a suitable approach to this problem.

For this problem use data set `fgl` from the `MASS` package. This is a forensic application. The observations consist of fragments of broken glass. The `type` column gives the type of glass. The `RI` column gives refractive index (but see description from `help(fgl)`). The remaining 8 columns are percentages by weight of oxides. The row totals of these 8 columns are approximately 100%.

The object is to build a classifier which predicts glass type from the measured `RI` and chemical composition of a glass fragment. We will consider three questions: (*i*) What form of classifier should be used? (*ii*) Which features should be included? (*iii*) How should the features be standardized (if this is needed)?

(a) For this exercise we will only consider 4 glass types: window float glass (`WinF`), window non-float glass (`WinNF`), vehicle window glass (`Veh`), and vehicle headlamps (`Head`). Create a subset of the data accordingly.

(b) We will consider three types of classifiers: KNN, LDA and QDA. Prove that QDA (and therefore LDA) is location-scale invariant. To do this, suppose a p-dimensional feature vector $\dot{x}$ is transformed to

$$\dot{y} = A\dot{x} + b$$

where $\dot{x}$ and $\dot{y}$ are interpreted as $p \times 1$ column vectors, A is an invertible $p \times p$ matrix, and b is a $p \times 1$ column vector. Then each class-dependent mean vector and covariance matrix must be transformed accordingly. An important observation is that neither A nor b depend on the class.

(c) Is the KNN classifier, assuming Euclidean distance is used, *location-scale invariant*? Why or why not?

(d) For any classifier that is not location-scale invariant, we will use the following strategy. We are given an n-dimensional class vector `Y` and an $n \times 9$ feature matrix `X`. First, standardize only the `RI` column to zero mean and unit variance. Then multiply the standardized `RI` column by a selected scale value α. Use the resulting feature matrix for the classification, and capture the classification error CE. Vary α, selecting the value yielding the minimum CE.

Then, write a main function which accepts a class vector and a data set of features, and which applies 3 classification methods to the data, giving an estimated CE for each:

(i) LDA, CE estimated using LOO cross-validation;

(ii) QDA, CE estimated using LOO cross-validation;

(iii) KNN, allowing K to vary over $1, 2, \ldots, 25$. For each model, estimate CE using LOO cross-validation. Select K yielding the minimum CE. In the case of ties, select the smallest K yielding the minimum CE.

Make sure the scale procedure just described is used for any classifier which is not location-scale invariant. Use α values generated by `RI.scale = c(1:150)/10`. The function should return enough information to identify the classifer with the smallest CE. The minimum output would be the minimum CE and the associated classifier, including the parameter K if that classifier is KNN, and any relevant scale factor α. Of course, more information can be returned.

(e) The main function will be used within the following heuristic, which sequentially removes features from the initial feature set $B = (1, 2, \ldots, 9)$ (ie the full feature set).

```
Blist is a list of Index Subsets
Blist = NULL

CElist is a list of Classification Errors
CElist = NULL

CEarray is a numerical array with elements addressed by index

# Initial feature set

B = (1,...,9)
CE = minimum possible CE using B

# Store classification results

Blist = append(Blist,B)
CElist = append(CElist,CE)

while length(B) > 1 {

     For each index I in B {

          # Remove index I from B, copy into B'

          B' = B - {I}
          CEarray[I] = minimum possible CE using B'
     }

     I* = Index from B which minimizes CEarray[I]

     # Update and store current feature set

     B = B - {I*}
     Blist = append(Blist,B)
     CElist = append(CElist,CEarray[I*])

}
```

Then select the feature set from `Blist` with the smallest CE (stored in `CElist`).

(f) Identify the exact classifier selected by the algorithm, including K and α, if needed. Report CE, and construct a confusion matrix.

(g) Use the `pairs` function to construct all pairwise scatterplots among the selected features (`RI` need not be scaled, if selected). Use symbols and/or colors to distinguish the four classes. Also, indicate which predictions are correct (for example, use colors to distinguish the classes, and symbols to indicate the prediction success). What does this plot suggest regarding why the classification method selected by the heuristic might be prefered?