CONCORDIA UNIVERSITY

Department of Mathematics & Statistics

STAT 380 (ACTU491H, MACF 491H, MAST 679H)

Statistical Learning

Assignment 1

Class Notes

Frédéric Godin, Ph.D., FSA, FCIA

January 2024

1 Instructions

- The assignment is done in teams of **three or four** people. In some specific cases where it is impossible to form teams of at least three, a special permission can be granted by the instructor to form smaller teams. If this is your case, please write to the instructor (**frederic.godin@concordia.ca**) to get his approval. You should try to avoid as much as possible mixed teams containing both undergraduate and graduate students.
- You are allowed to discuss the content of the assignment with people from other teams. However, solutions must be designed separately. Copies of solution files (pdf report and R code) from different teams must be redacted separately. Plagiarism is formally prohibited. Please refer to Concordia's policy related to plagiarism if you are not sure you understand clearly what plagiarism is.
- The due date to hand in your assignment is **Saturday February 17, 2024 at 11h59 pm**.
- A zip file **".zip"** must be handed in by email. This file should contain (exactly)
    - A **pdf** report containing the answer to all of the questions. For some sections of the assignment, the report might simply need to refer to the R code. This file can be prepared among others by Microsoft Word or LaTeX. This pdf file does not need to contain the R code developed to obtain the answers.
    - A single file **"main.R"** which contains the integral *R* code needed to run all experiments which lead to your answers. All numerical experiments must be done with the *R* software.
- Ensure your document is clean and clearly written. Points can be deducted if the clarity is deemed inadequate. This includes putting comments within your *R* code to ensure readability.

- The zip file must be sent by email to the professor before the due date. The professor's email is frederic.godin@concordia.ca
- **Justify all your answers and present the integrality of the rationale leading to your answers.**
- **Ensure the name of all teammates and their Concordia ID numbers are written on the first page of your report**.

2 Assignment questions

Question 1 (35 points)

This question is related to the prediction of the weekly electricity consumption (called electricity load) in the Nord Pool electricity market (Scandinavian countries) through Fourier expansions and regression.

We shall attempt predicting the weekly load by using a predictor related to time which is meant to represent seasonality. Indeed, we shall use the following model:

$$\begin{aligned} L_t &= \beta_0 + \sum_{j=1}^{P} \beta_{2j-1} C_t^{(\sin,j)} + \sum_{j=1}^{P} \beta_{2j} C_t^{(\cos,j)} + \epsilon_t && (1) \\ C_t^{(\sin,j)} &= \sin\left(\frac{3\pi}{2} + \frac{2\pi j t}{365.25/7}\right), \quad C_t^{(\cos,j)} = \cos\left(\frac{3\pi}{2} + \frac{2\pi j t}{365.25/7}\right) && (2) \end{aligned}$$

where

- t: denotes the week i.e. $t = 0$ is the first week, $t = 1$ is the second week, etc,
- L_t: is the weekly load (electricity consumption) during week t,
- ϵ_t: is a noise term to represent errors from the model,
- $C_t^{(\sin,j)}$ and $C_t^{(\cos,j)}$: are the sinusoidal predictors representing the first terms of a Fourier series with respect to time.

Note that this model is a simplified version of a model found in the paper of Dupuis et al. (2016). [1]

The objective of the question is to find suitable values for P and parameters β_j, $j = 0, \ldots, 2P$, and then to make predictions with this optimized model.

Part (a), 5 points:

[1] Dupuis, D. J., Gauthier, G., Godin, F. (2016). Short-term Hedging for an Electricity Retailer. *Energy Journal*, **37**(2).

Load the content of the file `Loaddata.RData` into `R` with the function `load`. This should load the following two vectors:

- `Loadseries`: a time series of the weekly electricity consumption on the Nord Pool market (Scandinavian countries).
- `Weekseries`: the date corresponding to the first day of the week for each entry of the time series.

Part (b), 5 points:

Construct the predictors design matrix which you will store in the matrix `Designmatrix` which is associated with the model (1)-(2) when $P = 5$. The design matrix must have the `matrix` format and not a `data.frame` format.

Part (c), 5 points:

Create a function `OLSSSE` which takes as inputs

- `yTrain`: a vector of responses from your training set,
- `XdesignTrain`: the predictors design matrix from your training set,
- `XdesignValid`: the predictors design matrix for observations in your validation set,
- `yValid`: a vector of responses from your validation set,

and which outputs

- `SSE`: the sum of squared prediction errors from your validation set when predictions were performed through an OLS regression whose parameters were estimated on the training set.

Part (d), 10 points:

You will now perform 6-fold cross-validation to identify the best value of P among $P = 1, 2, 3, 4, 5$. The dataset will be split as follows:

- the first subset corresponds to all data points observed in 2007,

- the second subset corresponds to all data points observed in 2008,
- $\cdots$
- the last subset corresponds to all data points observed in 2012,

Use your function `OLSSSE` from part (c) to fill a 5×6 matrix `SSEmat` containing as element (i, j) the out-of-sample sum of squared prediction errors for the subsample j and when $P = i$. Then sum the columns of this matrix to obtain a cross validation sum of squared errors (SSE) for each possible value of $P = 1, 2, 3, 4, 5$, which you should store in a vector `SSEtot`.

Write the values from `SSEtot` in your report. According this procedure, which is the best value of P?

Part (e), 5 points:

Use the optimal value of P you obtained in part (d) to retrain your model (i.e. re-estimate the parameters) on your full data sample. What are the parameter estimates you obtain (i.e. list them in a table)?

Use such re-estimated parameters (that you should store in a vector `OptPreds`) to perform predictions for each load value in the time series using your model (1)-(2).

Plot the time series of realized load values (the original time series) as a black curve, and then the time series of predicted load values as a blue curve. Add a legend to your plot to identify both curves.

Part (f), 5 points:

Using your model, what is the predicted value of the weekly load for the week starting on Monday August 13, 2012?

Question 2 (30 points)

This question is related to variables selection and regularization applied to linear regression models. We consider a situation where we need to predict the outstanding balance of a credit card user using attributes of the borrower. Understanding drivers of outstanding balances is relevant for lending institutions since it determines how much money will be lost if the borrower defaults on his payments.

The dataset used for the question is the `Credit` dataset from the `ISLR` package in `R`. The following steps must be followed to load the dataset:

- If you never installed the `ISLR` package before, you need to run the command `install.packages("ISLR")`.
- Then, once the package is installed, run the command `library(ISLR)`.
- The dataset is loaded and stored in the variable `Credit`. For instance, if you want some information on this dataset, you can use the command `?Credit`.

The response variable is the variable `Balance`, and all other variables are potential predictors.

In this question, we will perform 5-fold cross-validation for several models. The full dataset has a sample size of $n = 400$, and therefore the 5 data splits we will consider are:

- The first split corresponds to observations 1 to 80.
- The second split corresponds to observations 81 to 160.
- The third split corresponds to observations 161 to 240.
- The four split corresponds to observations 241 to 320.
- The fifth split corresponds to observations 321 to 400.

Note that for this question, you are allowed to use built-in functions from `R` such as `lm`, `regsubsets` and `glmnet`.

Note: for this question, you might want to refer to Labs from Chapter 6 of the ISL textbook.

Part (a), 5 points:

Calculate the 5-fold cross-validation root mean square error (RMSE) i.e. the square root of the MSE for a linear regression model where `Balance` is regressed on **all** other predictors.

Part (b), 5 points:

Run a linear regression of `Balance` on all other predictors **on the full dataset**. Determine which parameters are significant at the 5% confidence level.[2]

Then, calculate the 5-fold cross-validation RMSE for a linear regression model where `Balance` is regressed only on predictors which were deemed significant in the previous step.

Part (c), 5 points:

Load the `R` package called `leaps` through the command `library(leaps)`. Note that if you did not install this package previously, you must first enter the command `install.packages("leaps")` before entering `library(leaps)`. Use the `regsubsets` function to perform a forward variable selection (on the whole dataset) and identify the number of parameters (and the names of such parameters) which minimizes the Bayesian Information Criterion (BIC) metric.

Then redo another variable selection procedure using `regsubsets`, but this time using an exhaustive search (i.e. considering all possible subsets of predictors) instead of a forward approach. Are the two results the same? Which model is the best according to this exhaustive search approach?

Compare the 5-fold cross validation performance of the final models obtained through the exhaustive search and forward procedure.

[2]For the t-statistics and p-values to be valid, we would need to ensure that the Gauss-Markov assumptions are verified i.e. that the noise term ϵ is i.i.d. $N(0, \sigma^2)$. We assume here (possibly erroneously) that this assumption holds.

Part (d), 5 points:

Load the `R` package called `glmnet` through the command `library(glmnet)`. Note that if you did not install this package previously, you must first enter the command `install.packages("glmnet")` before entering `library(glmnet)`. You will attempt to make predictions using a ridge regression.

Enter the command `lambdagrid=10^seq(10,-2,length=100)`. This vector grid contains all the potential values that will be considered for the tuning parameter λ.

First use the function `model.matrix` to obtain a fully numerical design matrix for predictors (qualitative variables are transformed into dummies automatically). For more details, see section 6.6.1 of the ISL book.

Use the function `glmnet` to calculate the 5-fold RMSE of a ridge regression for all possible values for λ found in `lambdagrid`.

Note: when you use the `glmnet` function to estimate ridge regression parameters, ensure the input lambda=lambdagrid is passed to the function. Otherwise, the numerical algorithm could consider a range of values λ which is too narrow.

Note: use scaled predictors in the ridge regression (i.e. in `glmnet` use the option `standardize = TRUE`, which is the default option i.e. the option selected if you omit it).

Which value of λ is optimal in terms of minimization of out-of-sample prediction, and what is the associated RMSE?

Part (e), 5 points:

Redo exactly as in part (d), except you now use a lasso regression instead of a ridge regression.

What is the optimal value of λ and what is the associated out-of-sample RMSE?

Part (f), 5 points:

After all these tests, what would be the model you ultimately would suggest to use? What are the predictors and associated coefficients β associated with this model you would use to make predictions on new data?

Question 3 (35 points)

This question relates to the pricing of financial options. An option is a contract which allows buying (or selling) some financial asset at a given pre-determined date (the maturity) at a pre-determined price (the strike). The decision to buy or not at that date and price belongs to the holder of the option.

A key input to perform the pricing of an option is called its **implied volatility**. We shall not discuss option pricing theory here. For non-financial students who are interested in learning more about this topic, see

- Wikipedia: Black-Scholes option pricing model
- Wikipedia: Implied Volatility
- Investopedia: Implied Volatility

The implied volatility of an option depends on the characteristic of the option itself. For an option on a given asset, the implicit volatility will be a function of the strike price and of the time-to-maturity (i.e. the time difference in years between the option maturity and today).

There is a functional relationship between the price of an option and its implied volatility. Therefore, when an option trades on an exchange, the trading price is observed, which allows observing the implied volatility of this option. Note however that observed trading prices (and thus observed implied volatility) are noisy due to trading noise and market frictions.

In this question, we will attempt performing the prediction of the implied volatility for options that were not traded previously by using the information provided by previous trades.
More specifically, we will use as a response the implied volatility of an option which was traded recently, and we will use its strike and time-to-maturity as predictors. We will attempt predicting the implied volatility of a new option whose strike and time-to-maturity are given. This information is useful as such predicted implied volatility indicates at which price they should trade a given option.
First, load the content of the file `IV.RData` in `R`. using the function `load`.

Part (a), 3 points:

Create a function `dmvn` which calculates the pdf of a multivariate normal distribution. The function should take as inputs

- `xmat`: a matrix representing the set of points at which the pdf is calculated; each row represents one given data point,
- `muvec`: a column vector representing the mean of the distribution,
- `Sigmamat`: a matrix representing the covariance matrix of the distribution

and which outputs

- `pdfvec`: a vector containing the pdf for all data points.

Part (b), 10 points:

You will now make a prediction of the implied volatility for all possible combination of values for the time-to-maturity and strike in the following vectors:

- Time-to-maturity: `gridMaturity = seq(from=0, to = 1, by = 0.05)`

- Strike: `gridStrike = seq(from=850, to = 1150, by = 20)`.

The method you will use for this question is **local linear regression**. You will use $K = n$ i.e. all training points will be considered in your local regression predictions. Local regression weights should be calculated with Gaussian kernels (use the function `dmvn` developed previously). Three different values for the covariance matrix Σ of the Gaussian kernel should be considered:

$$\Sigma^{(1)} = \begin{bmatrix} 0.3 & 0 \\ 0 & 100 \end{bmatrix}, \quad \Sigma^{(2)} = \begin{bmatrix} 0.1 & 0 \\ 0 & 25 \end{bmatrix}, \quad \Sigma^{(3)} = \begin{bmatrix} 0.9 & 0 \\ 0 & 200 \end{bmatrix},$$

where the first component corresponds to the time-to-maturity and the second corresponds to the strike.

Build a list `fhatmatlist` of length 3 whose j^{th} element is the 21×16 sized matrix of implied volatility predictions when $\Sigma^{(j)}$ is used to construct weights. Note that the element (i, m) of the latter matrix is the implied volatility prediction when the time-to-maturity is the i^{th} element of `gridMaturity` and when the strike is the m^{th} element of `gridStrike`.

Part (c), 5 points:

Install `plot3D` (if this was not done previously). Load the package with the command `library(plot3D)`.

Then, using `R`, draw an image containing 4 subfigures.

- The first 3 subfigures should be drawn with the function `surf3D`. They should contain a 3-D plot (strike in x axis, time-to-maturity in the y axis and implied volatilities in the z axis) containing the implied volatility predictions for all combinations of values of the predictor lists `gridMaturity` and `gridStrike`. Recall that such predictions were stored in `fhatmatlist`. The j^{th} subfigure contains predictions using $\Sigma^{(j)}$ to compute weights.

- The last subfigure should contain a scatterplot of all data points from the dataset (strike in x axis, time-to-maturity in the y axis and implied volatilities in the z axis). It should be drawn with the function `scatter3D`.

Note: the function `mesh` should be useful to obtain inputs required to produce the first three plots.

Note: to stack the four plots in the same figure, you can use the following command: `par(mfrow=c(2,2), oma = c(1,0,0,1) + 0.1, mar = c(2,0,1,1) + 0.3)`.

Note: You might find the following inputs to the function `surf3D` useful to make your plots nice:

- `xlab`, `ylab`, `zlab`: puts labels on your axis,
- `theta`: rotation angle to view your plot, I suggest $theta = 40$,
- `bty`: set `bty="b2"` to display axis grids and labeling,
- `ticktype`: allows for explicit axis grading,
- `cex.axis`: axis numbers magnification,
- `cex.lab`: axis label magnification.

Part (d), 5 points:

Create a function `PredictNearestNeighbors` which calculates the K-nearest neighbors prediction for a set of new observations with the help of responses and predictors from some training data. The function should take as input

- `yval`: a vector containing training data responses,
- `xval`: a matrix containing training data predictors (each row corresponds to one observation, each column represents the value of a single predictor for all observations),

- `xpred`: a matrix containing predictors associated with the new observations whose response we are trying to predict (each row corresponds to one new observation, each column represents the value of a single predictor for all new observations),
- `NNnumber`: the number K in K-nearest neighbors (number of neighbors used to make predictions),

and which outputs

- `NNpreds`: the vector containing the response prediction for each new observation.

Part (e), 10 points:

Redo exactly as in part (b), but this time use the K-nearest neighbors to make predictions instead of the local regression. The three values for K you should consider are $K = 5$, $K = 15$, $K = 30$. The function `PredictNearestNeighbors` created in (d) should be used to make predictions.

Since the strike predictor is on a different scale than the time-to-maturity predictor, **divide all values of the strike predictor by $c = 400$ to make K-nearest neighbors predictions**.

Instead of storing your predictions in `fhatmatlist`, store your predictions in a new size-3 list called `fhatmatlistKNN` whose j^{th} element are predictions for the j^{th} possible value of K among $\{5, 15, 30\}$.

Part (f), 2 points:

Redo exactly what you did in part (c) i.e. drawing the four plots, but this time using the K-nearest neighbors predictions instead of the local regression ones.