MTAT.03.227 Machine Learning
Spring 2014 / Exercise session III
Nominal score: 10p
Maximum score: 15p
Deadline: 5th of March 8:15 EET

1. This exercise illustrates some basic properties of matrix algebra and why is it useful. You have to implement prediction of a linear model and its mean square error computation through linear algebra operations.

   (a) Load the data from the file `exercise-1.Rdata` and modify the file `basics-of-lm.R` in order to define design matrix X and predict outputs using the matrix equation $\boldsymbol{y} = X\boldsymbol{\beta}$. Test the correctness of the result against default prediction algorithm `predict.lm`. Next compute the mean square error with the matrix expression from the lecture slides. Compare the result with straightforward implementation. Are the results same? (**1p**)

   (b) Implement linear regression as function `FitLM` using the closed form solution form the lecture slides. Compare the result with the default implementation `lm`. Are the results same for the dataset `data`. See whether your implementation can also handle multivariate regression problems. If not modify the function accordingly. Generate a three dimensional input data consisting of 100 samples following the model $y = x_1 + 4x_2 - x_3 + \varepsilon$ where $\varepsilon_i \sim \mathcal{N}(0, sd = 0.05)$. Compare the results of your method and the default method `lm`. (**1p**)

   (c) Recall that linear regression tries to find parameters (β_0, β_1) that minimise the mean square error. A naive way to minimise the function is to try all possible values of β_i and then find the minimum value. This can be done by extensive tabulation consider a grid $\{0.0, 0.1, \ldots, 2.0\} \times \{-5.0, -4.9, \ldots, 5.0\}$ for the potential parameters. Draw the corresponding surface $\mathrm{MSE}(\beta_0, \beta_1)$ and confirm that the minimum is indeed in the location predicted by the linear regression formula. You can use `contour` and `wireframe(..., scales=list(arrows=FALSE))` commands for 3D plotting. In both methods the surface is specified by the value matrix. Arguments `row.values` and `column.values` allow you to prescribe scales for rows and columns. Draw an illustrative plot that shows the correctness of linear regression formula. (**1p**)

2. This exercise illustrates the meaning of confidence intervals. For this exercise, you should modify the contents of the file `confidence-intervals.R`. Two functions `GetConfidenceInterval` and `GenerateData` are defined in such a way that `GetConfidenceInterval` returns a correct confidence interval when the data is generated by `GenerateData`.

(a) Verify that the claim about confidence intervals is indeed correct. That is, if we apply the algorithm `GetConfidenceInterval` on large number of datasets generated by `GenerateData`, then the true value y_0 is inside the confidence interval in α fraction of of runs where α is the confidence level. Try four different parameter sets and draw also the illustrative plots with 100 runs for each parameter set. Interpret the results. How does the size confidence intervals depend on the number of observations and on the confidence level? Assume that the algorithm has returned a confidence interval, e.g. consider the first run of the algorithm in the first plot. What is the probability that y_0 is in the interval? (**1p**)

(b) In practise, one often needs to find a right trade-off between the length of the interval and the expected fraction of failures where the true value is outside the interval. Consider the following two-stage measurement scheme. You can first do low-precision measurements to determine the approximate value of y_0 and later use high precision measurement device to get the final measurement value. Each low precision measurement costs 0.05 € and the high precision measurement costs proportionally to the length of the initial interval estimate. A high-precision scan through a unit length interval costs 1 €. Describe at least two possible measurement strategies and use simulation to estimate their efficiency (average amount of money needed to get a successful high-precision measurement). (**1p**)

3. Although GNU R provides built-in support for the polynomial regression and other types of generalised linear regression, it is important to understand how these regression algorithms work. For that load the data form file `exercise-3.Rdata` and modify the file `multivariate-regression.R` to solve the following subtasks.

(a) Recall that multivariate linear regression fits models of type $y \sim x_1 + \cdots + x_k + 1$. What should x_1 and x_2 be if we want to fit a quadratic model $y \sim x^2 + x + 1$?. Define a new data matrix `quadratic.data` that is suitable to perform a quadratic regression and train the corresponding linear model. Extract coefficients and draw a plot which contains the data points and corresponding prediction. Repeat the same procedure in order to fit the polynomial with the degree 10. Compare the results with the built-in procedure. (**1p**)

Hints: You can use `lm(y ~., data)` to fit a linear model that predicts y form all other columns of the data frame. Note that the formula `y ~ poly(x, 3, raw = TRUE)` can be used in `lm` to determine coefficients of a cubic model.

(b) The dataset `weird.data` seems to capture linear relation which is corrupted by a periodic sinusoidal noise. More precisely, the noise component has basic frequency 2.5 Hz provided that `x` measured in seconds. Build an appropriate extended data matrix and perform a multivariate linear regression to identify the system. Draw the plot that shows the final fit of the model and interpret results. (**1p**)

Hint: Note that any sinusoidal noise component with frequency ω can be represented as $a_1 \sin(2\pi\omega x) + a_2 \cos(2\pi\omega x)$.

4. Blind application of multivariate regression algorithms can lead to over- and under-fitting. Over-fitting is a phenomenon where the mean square error is much smaller on the training set than on the test set. As such it can be easily detected given enough samples. Under-fitting is much harder to detect as mean square errors for training and test sets are same while using more flexible model would greatly improve the prediction quality. Besides under-fitting we can also mis-fit the model due to few incorrect measurement commonly referenced as outliers. Load the data generation algorithms form the file `data-sources.R` and modify the file `basic-diagnostics.R` to solve the following subtasks.

   (a) A rule of thumb says that multivariate linear regression is successful if the residuals $y_i - \hat{y}_i$ are approximately normally distributed. If the distribution is non-symmetric and has few residuals which are large then the model mis-fits the data. Study the data distributions `FirstSource`, `SecondSource` and `ThirdSource` by sampling 100 data samples from each source. Use the `RemoveOutliers` function if you detect strong deviations from normal distributions. Measure the performance of original and improved models on training set and on test set. To form a test set draw another 100 samples form the data distribution. Draw a plot where the predictions of two models are contrasted with the actual measurements. Report normalised mean square error for all four combinations and interpret the results. (**1p**)

   (b) A standard way to detect nonlinearities in the data is a visual inspection. For univariate data, the procedure is straightforward, as the data naturally fits into two dimensional plot. For multivariate data, one can use `crPlots` and `ceresPlots` to get a virtual 2D projection. Ideally, you should see points that are almost linearly aligned without any clear pattern. Draw 100 samples form `SecondSource` and build three extended input datasets. The first with values x, x^2, y, the second with values x, x^3, y and the third with values x, x^2, x^3, y. Choose the best fit based on `crPlots`. (**1p**)

5. Linear regression methods can perfectly recover the underlying linear dependence $y = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$ provided that both input nor output measurements are noise-free. In the presence of noise, the recovery procedure is less successful and resulting coefficients $\hat{\boldsymbol{\beta}}$ drift away from the true values $\boldsymbol{\beta}$. Over-fitting can be viewed as a situation where the noise has bigger impact to the coefficients $\hat{\boldsymbol{\beta}}$ than the true signal and thus obtained estimates for coefficients are unreliable. The next exercise illustrates how over fitting and leverage are connected. Modify the file `overfitting.R` to solve the following subtasks.

(a) Recall that we can express $\hat{\boldsymbol{\beta}} = (X^T X)^{-1} X^T \boldsymbol{y}$ and thus we can decompose the impact of additive noise into second term

$$\hat{\boldsymbol{\beta}} = (X^T X)^{-1} X^T (\boldsymbol{y} + \boldsymbol{\varepsilon}) = (X^T X)^{-1} X^T \boldsymbol{y} + (X^T X)^{-1} X^T \boldsymbol{\varepsilon} \ .$$

Verify that this theoretical result holds also for the command `lm`. For that train three models `model.y`, `model.eps` and `model.ye` for relations $y \sim x + 1$, $\varepsilon \sim x + 1$ and $y + \varepsilon \sim x + 1$. Check whether `coeff(model.y) + coeff(model.eps) = coeff(model.ye)`. (**1p**)

(b) Provided that the linear relation described above holds, then the impact of the additive noise can be characterised by the term $\Delta\boldsymbol{\beta} = (X^T X)^{-1} X^T \boldsymbol{\varepsilon}$ which can be computed by fitting the linear model directly to the noise. Study how the normal noise $\mathcal{N}(0, 1)$ propagates for a linear model $y \sim x+1$ and polynomial model $y \sim x^9 + \cdots + x + 1$. Do this by repeatedly sampling the noise vector and solving the corresponding linear regression task. Draw boxplots for the components of inferred $\Delta\hat{\boldsymbol{\beta}}$ and interpret results. How reliable are coefficients estimates in the presence of noise $\mathcal{N}(0, sd = 0.01)$? How large should the coefficients of the 9th order polynomial be so that the error component is around 10% in the presence of noise $\mathcal{N}(0, sd = 0.01)$? (**1p**)

(c) Note that the error component can be further split and we can study how the error ε_i in single location x_i influences the coefficients $\hat{\boldsymbol{\beta}}$. The latter is known as leverage. Study and report how error in different points influences the coefficient vector for models $y \sim x + 1$ and polynomial model $y \sim x^9 + \cdots + x + 1$. Are there some points where the error has larger impact? Present results visually. (**1p**)

(d) Sometimes large changes in the coefficient vector $\boldsymbol{\beta}$ have only a marginal effect on predictions. Study how the error ε_i in single location x_i influences the prediction $\hat{y}(1)$ and $\hat{y}(2)$. Again, study models $y \sim x + 1$ and $y \sim x^9 + \cdots + x + 1$. Are there some points where the error has larger impact? Present results visually. (**1p**)

6. Study the Wine Quality Data Set from UCI Machine Learning repository `http://archive.ics.uci.edu/ml/datasets/Wine+Quality` in order to detect which attributes influence the goodness of wine. For that build various linear and nonlinear models for predicting the wine quality form other attributes. To get reliable results, split the data into training and test set and report prediction performance on both sets. Use various diagnostic measures d detect over- and under-fitting. (**5p**)