Exercise Set 1

- Answer anonymously, i.e., do not write your name to the answer sheet.
- Submit the answer via Moodle at latest on 8 November 2020 (Moodle submission will open during the week starting on 2 November).
- Your answer will be peer-reviewed by other students and you will review your answer and answers of 2 random peers during the week starting on 9 November.
- The assignment should be completed by one person, but discussions with others are encouraged. Your final solution must be your own.
- The language of the assignments is English.
- The submitted report should be in a single Portable Document Format (pdf) file.
- Answer to the problems in the correct order.
- Read the general instructions in Moodle before starting with the problems.
- Main source material: James et al., Chapters 1-3 and 5. Please feel to use other resources as well.
- Version history: v2: Task 4c clarified.

Problem 1

[5% points]

Objective: familiarity with command line tools

Unix command line tools are quite useful. Download the csv-file of Problem 2 below and do the following operations with the unix command line tools.

Tasks

a. Show the first and last lines by using `head` and `tail`.
b. Count the rows by using `wc`.
c. Change the file from csv (comma-separated values) format to tsv (tab-separated values) format by using `sed` or `tr`.
d. Separate the class variable (column "class4") by using `awk` and list the unique classes.

Hint: Useful commands: `head`, `tail`, `sort`, `uniq`. You can find the instructions in any unix system by man command (e.g., `man head`), or by using Google.

Instructions

Sufficient answer here is listing of commands and outputs for the tasks above.

Problem 2

[10% points]

Objective: familiarity with tools, basic description of the data set

This exercise relates to a data set about new particle formation (NPF) of which you will do your term project. Read the data description and download the dataset file `npf_train.csv` from the Moodle term project page.

The instructions below are in R, but you can do equivalent tasks with Python as well. Before reading the data into R, it can be viewed in Excel or a text editor.

Task a

Use the read.csv() function to read the data into R. Call the loaded data npf. Make sure that you have the directory set to the correct location for the data.

```
npf <- read.csv("npf_train.csv")
```

Task b

Look at the data using the `fix` function. In RStudio, instead of `fix`, you can use the nicer Data Viewer `View`.

```
fix(npf)
```

You should notice that the second column is the date and first column is the id. We don't really want R to treat this as data. However, it may be handy to have the dates. Try the following commands:

```
rownames(npf) <- npf[,"date"]
fix(npf)
```

You should see that there is now a row.names column with the name of each day recorded. This means that R has given each row a name corresponding to the appropriate id. R will not try to perform calculations on the row names. However, we still need to eliminate the first column in the data where the id is stored. Try

```
npf <- npf[,-1]
fix(npf)
```

Now you should see that the first data column is date. Note that another column labeled row.names now appears before the data column. However, this is not a data column but rather the name that R is giving to each row.

Task c

i. Use the summary() function to produce a numerical summary of the variables in the data set. We notice that the variable "partlybad" is always false and thereofore useless. We opt to remove it as well.

```
npf <- npf[,-3]
```

ii. Use the pairs() function to produce a scatterplot matrix of the first ten columns or variables of the data. Recall that you can reference the columns from 3 to 12 of a matrix A using A[,3:12].

```
pairs(npf[,3:12])
```

iii. Use the plot() function to produce side-by-side boxplots of event vs. nonevent days.

```
boxplot(RHIRGA84.mean ~ class4,npf)
```

iv. Create a new qualitative variable, called class2, which is "event" if there was a NPF event and "nonevent" otherwise.

```
npf$class2 <- factor("event",levels=c("nonevent","event"))
npf$class2[npf$class4=="nonevent"] <- "nonevent"
```

Use the summary() function to see how many event days there are. Now use the plot() function to produce side-by-side boxplots of RHIRGA84.mean versus event.

```
boxplot(CS.mean ~ class2,npf)
```

v. Use the hist() function to produce some histograms with differing numbers of bins for a few of the quantitative variables. You may find the command par(mfrow=c(2,2)) useful: it will divide the print window into four regions so that four plots can be made simultaneously. Modifying the arguments to this function will divide the screen in other ways.

vi. Continue exploring the data, and provide a brief summary of what you discover.

Problem 3

[10% points]

Objective: how to deal with probabilities, relevance of floating point arithmetics.

In machine learning we often have to deal with very small (or large) numbers. Take, for example, the probability density of the normal distribution given by $p(x) = \exp(-x^2/2)/\sqrt{2\pi}$ or in R, `p <- function(x) exp(-x^2/2)/sqrt(2*pi)`, which produces to very small numbers for any larger values of x.

Multiplication, division, and sums can easily lead to under or overflows. We can mitigate the problem by representing the probabilities as logarithms and then doing the multiplication, division, and summation using logarithmic values. I.e., instead of a probability p_i, where $i \in [n] = \{1, \ldots, n\}$, we use the natural logarithms $l_i = \log p_i$ to present the numbers. Denote the product of probabilities by $p_P = \prod_{i=1}^n p_i^{a_i}$ and sum by $p_S = \sum_{i=1}^n p_i$. Furthermore, denote $l_P = \log p_P$ and $l_S = \log p_S$.

Task a

Show the following equalities are true:

- $l_P = \sum_{i=1}^n a_i l_i$, and
- $l_S = \max_{j\in[n]} l_j + \log \sum_{i=1}^n e^{l_i - \max_{j\in[n]} l_j}$.

Task b

Implement the product and sum operations (i.e, formulas for l_P and l_S) as R or Python functions. Compute the value of the following expression by without the log presentation and by using the log representation and the functions you implemented above: $p(100)/(p(100) + p(100.01))$, where $p(x) = \exp(-x^2/2)/\sqrt{2\pi}$.

You should notice that without the "log trick" you should obtain a NaN value because of floating points underflows, but with the log trick you should obtain a correct answer which in this case should be about 0.731. Operations like this could appear later, e.g., in Naive Bayes classifier or other machine learning computations.

Task c

Read through the R documentation of numerical characteristics of your computer by running R command `help(.Machine)` and `help(Inf)` and experiment with R.

- What do expressions giving very small and large numbers evaluate to, e.g., `exp(-1000)` and `exp(1000)`?
- What is `1/Inf` and what does it evaluate to?
- Let `x` be the smallest positive number such that the expression `1+x!=1` is true. What is `x` called and what is its numerical value in your computer?

NB: Notice that Python floating point implementations behaves a bit differently. You can see this, e.g., by running:

```
import math
math.exp(1000)
```

Problem 4

[15% points]

Objective: theoretical properties of validation set methodology in model selection

The purpose of this exercise is to show some theoretical properties of a fundamental supervised learning problem. Most of the discussion here revolves around trying to estimate the "true loss" $E_{(x,y)\sim F}[L(y, f(x))]$, which is the central machine learning problem.

Consider the "usual" setup in supervised learning, where we are given a loss function $L(y, \hat{y})$ and n data points (x_i, y_i) where $i \in [n] = \{1, \ldots, n\}$. The data has been sampled i.i.d. from a fixed but unknown distribution $(x_i, y_i) \sim F$. Our objective is to find a function $\hat{y} = f(x)$ from some family of functions such the expected loss ("true loss") $E_{(x,y)\sim F}[L(y, f(x))]$ is as small as possible. We have K "learning algorithms" A_k, each of which takes n_{tr} data points as an input and outputs a function f_k, where $k \in [K]$, from the given concept class.

Example: The data could be real numbers, i.e., $(x, y) \in \mathbb{R}^2$, and the loss could be squared loss $L(y, \hat{y}) = (y - \hat{y})^2$ and the learning algorithms, e.g., linear models with polynomials of x of degree k (next problem!). A possible algorithm to find the function f_k would be, e.g., standard OLS linear regression. The basic binary classification case could be $(x, y) \in \mathbb{R} \times \{0, 1\}$ and 0-1 loss $L(y, \hat{y}) = I(y \neq \hat{y})$, where the indicator function $I(\Box)$ is 1 if $\Box$ is true and 0 otherwise.

In validation set methodology (see James et al., Sec. 5.1.1) you split the data to training, validation, and test sets, respectively. The training, validation, and test sets n_{tr}, n_{va}, and n_{te}, respectively, are given such that $n_{tr} + n_{va} + n_{te} \leq n$. Assume that you split your dataset in random into non-overlapping training, validation, and test set of the given sizes. Denote by $S_{tr} \subseteq [n]$, $S_{va} \subseteq [n]$, $S_{te} \subseteq [n]$ the indices of the data points in the training, validation, and test sets, respectively.

First consider the case of only one learning algorithm, i.e., $K = 1$. Denote by f_1 the function output by this learning algorithm when input the data points in the training set. Assume that you pick data point (x_i, y_i) from the validation set in random, where $i \in S_{va}$. Denote the loss by a random variable $L_i^1 = L(y_i, f_1(x_i))$. This random variable is an unbiased estimator for the true loss $E_{(x,y)\sim F}[L(y, f_1(x))]$ (you can show this if you want!).

Task a

Assume that the random variable L_i^1 described above has a variance of σ^2. Consider the average loss on the validation set of size n_{va}, i.e., $L^1 = \sum_{j \in S_{va}} L_j^1 / n_{va}$. What is the variance of L^1, expressed in terms of n_{va} and σ^2?

Task b

In machine learning the fundamental problem is to select a model or learning algorithm, which is the main reason for using the validation set. Assume that you now have $K \geq 2$ learning algorithms. You can use your training set to find functions $f_1, \ldots, f_K$ by using your K different learning algorithms and compute the respective averaged validation losses $L^k = \sum_{j \in S_{va}} L(y_j, f_k(x_j))/n_{va}$, where $k \in [K]$. Denote by κ the index of the learning algorithm that gives you the smallest loss, i.e., $\kappa = \arg\min_{k \in [K]} L^k$.

Show that the average loss on the *test set* $\sum_{i \in S_{te}} L(y_i, f_\kappa(x_i))/n_{te}$ is an unbiased estimate of the true loss $E_{(x,y)\sim F}[L(y, f_\kappa(x))]$.

Task c

In task b above the loss L^κ is typically a biased (under-)estimate of the true loss. It is good to have a grasp of the amount of under-estimation, i.e., how large is the bias in L^κ? Lets do a small back-of-the-envelope computation.

Assume, for the sake of the argument, that all K algorithms are equally good (or bad) and that the expected loss for all the K models is zero, i.e., $E[L^k] = 0$ for all $k \in [K]$. Now, assume (again, just for the sake of the argument) that each of the validation losses L^k are mutually independent and they obey normal distribution with zero mean and unit variance. The expected validation loss of the chosen model is then given by $L_{va} = E[\min_{k \in [K]} L^k]$ (recall, we choose the model with the smallest validation loss!). Study the expected validation loss L_{va} numerically by using sampling. Plot the expectation as a function of K for values of K from 1 to 10000. Hint: You may want to use logarithmic x axis in your plot to get a straight-looking line. You can use suitably spaced values for K to have nice-looking plot (i.e., it is not necessary to compute all 10000 points!).

Task d (bonus)

This is a bonus task. You can do it if you want, but you will not get extra points for it.

In validation set methodology you are faced with a compromise: if your training set is large then you would expect to get a better model but less inaccurate estimate of the true loss and vice versa. What would be a smart choice for the sizes of these three sets?

Problem 5

[20% points]

Objective: learning linear regression, concrete use of validation set, k-fold cross validation

In this problem we study linear regression on a synthetically generated dataset, with underlying function $f(x) = 1 + x - x^2/2$. The idea here is also to apply the theory in the previous problem into practice.

Task a

Create a *training set* of 20 pairs (x_i, y_i), where $i \in S_{tr} = [20] = \{1, \dots, 20\}$, where x_i are sampled uniformly from interval $[-3, 3]$ (R function `runif`) and $y_i = f(x_i) + \epsilon_i$, where ϵ_i are i.i.d. normal random variables with zero mean and standard deviation of 0.4 (R function `rnorm`); this is your distribution F! Using the same procedure and distributions sample a *validation set* $S_{va} = \{21, \dots, 40\}$ of 20 pairs and a *test set* $S_{te} = \{41, \dots, 1040\}$ of 1000 pairs. In total, you should now therefore have 1040 pairs.

Task b

Lets choose $K = 11$ and let the learning algorithms to be OLS linear regression with polynomials of degree $k - 1$, where $k \in [K]$. More specifically, fit polynomials $\hat{y} = \sum_{p=0}^{k-1} w_p x^p$ to the training set (of 20 data items) for different orders $k \in [K]$ by using ordinary least squares (OLS) regression. For each 11 values of k produce a plot showing the points (x_i, y_i) in the training set and the fitted polynomial in interval $[-3, 3]$. Calculate and report the means squared error (MSE) on training set, where $\text{MSE}_{tr} = \sum_{i \in S_{tr}}^{n} (y_i - \hat{y}_i)^2 / n_{tr}$, and compare the MSE for the different orders of polynomials.

Task c

Use the 11 polynomials you found above and compute and report the MSE of the polynomials on the validation and test sets as well.

Task d

Now, choose the polynomial degree κ that has the smallest MSE on the validation set. How do the MSE on the training, validation, and test sets compare?

Train a new regressor of degree κ on the combined training and validation set and report the MSE on the test set.

Task e

Now, instead of using simple training-validation set split combine the training and validation sets and use 10-fold cross validation to select a model based on the training+validation set (see James et al., Sec. 5.1.3). Report the cross-validation loss (Eq. (5.3) of James et al.) for different polynomial orders as well as the losses on the test set. How do the losses compare with the your previous results? Based on your cross-validation result, which polynomial degree should you choose?

Read about the bias-variance trade-off for k-fold cross-validation. (James et al., Section 5.1.4).

Problem 6

[20% points]

Learning objectives: bias and variance and model flexibility

Read Section 2.2 of James et al.

Consider the bias variance decomposition in the context of model selection.

Task a

Provide a sketch of typical (squared) bias, variance, training error, test error, and Bayes (or irreducible) error curves, on a single plot, as we go from less flexible statistical learning methods towards more flexible approaches. The x-axis should represent the amount of flexibility in the method, and the y-axis should represent the values for each curve. There should be five curves. Make sure to label each one.

Explain why each of the five curves has the shape displayed.

Task b

Test the bias variance tradeoff in practice with the data generation process described in Problem 5, at point $x = 0$. Generate 1000 new training sets of 20 pairs and train a polynomial regressor $g(x)$ on each of the training sets. For each of the 1000 training sets generate additionally a test data point at $x = 0$, i.e., $(0, y_0)$, where $y_0 = f(0) + \epsilon$ and ϵ is a random variable with zero mean and variance of $\sigma^2 = 0.4^2$.

According to Eq. (2.7) of James et al., the expected (expectation over your 1000 data sets, denoted by E_D) squared loss at $x = 0$, or $E_D[(y_0 - g(0))^2]$, can then be decomposed as a sum of irreducible error (here σ^2), the bias term $(E_D[g(0)] - f(0))^2$, and the variance term $E_D[(g(0) - E_D[g(0)])^2]$. Compute and plot these 4 terms (squared loss, irreducible error, bias term, variance term) at $x = 0$ as a function of polynomial degrees from 0 to 10 by using your 1000 data sets (i.e., you should end up with 4 curves). Check that the terms sum to squared loss for different polynomial degrees, i.e., verify Eq. (2.7) of James et al. Do the terms behave as you would expect from the discussion in the task a above?

Problem 7

[15% points]

Objective: properties of estimators

Lets continue with the linear regression and the data you created in Problem 5. So far, we have tried only to estimate the loss (with the purpose of choosing the best model). In machine learning it is also important to be able to estimate model parameters. For example, in the linear regression the parameters would be the regression coefficients, i.e., w_p in Problem 5 above.

Lets consider the simplest case of 0-degree polynomial ($k = 1$ in Problem 5) where you want to fit the constant line $\hat{y} = w_0$ to the data. In other words, lets first just compute mean of the data $\hat{y} = w_0 = \sum_{i \in S_{tr}} y_i / n_{tr}$.

Task a

What is the t-statistics (James et al., Sec. 3.1.2) and the corresponding 95% confidence intervals for the mean? Can you conclude that the true mean of the data is non-zero, by using the 20 data points in the training set of Problem 5 alone?

Task b

Study the coefficients $w_0, w_1, \ldots$ of the polynomials of different degree (from Problem 5) with R (or, e.g., with corresponding Python SciPy functions) as in Sec. 3.6.2-3 of James et al. Are the estimated coefficients and confidence limits consistent with what you know about the data generating process, i.e., that the data has been generated by using a degree-2 polynomial $1 + x - x^2/2$ plus random noise?

Problem 8

[5% points]

Objectives: self-reflection, giving feedback of the course

Tasks

- Write a learning diary of the topics of lectures 1-4 and this exercise set.

Instructions

The length of your reply should be 1-3 paragraphs of text. Guiding questions: What did I learn? What did I not understand? Was there something relevant for other studies or (future) work?