Machine Learning Exercises

Jan-Philipp Kolb

30 5 2019

Part A - foundations in R

Exercise: Find R-packages

Go to https://cran.r-project.org/ and search for packages that can be used:

1) to reduce overfitting
2) for regression trees
3) for gradient boosting
4) for neural networks
5) for clustering

Exercise: load built-in data

Load the the built-in dataset `swiss`

1) How many observations and variables are available?
2) What is the scale level of the variables?

Interactive data table

3) Create an interactive data table

Exercise: random numbers

1) Draw 8 random numbers from the uniform distribution and save them in a vector `x`
2) Compute the logarithm of `x`, return suitably lagged and iterated differences,
3) compute the exponential function and round the result

```
## [1] 0.3 3.8 0.9 0.7 0.9 0.9 1.2
```

Exercises: Visualizing dataset to apply machine learning

- Exercise based on r-exercises - visualizing for ml

Exercise 1

Create a variable “x” and attach to it the input attributes of the “iris” dataset. HINT: Use columns 1 to 4.

Exercise 2

Create a variable “y” and attach to it the output attribute of the “iris” dataset. HINT: Use column 5.

Exercise 3

Create a whisker plot (boxplot) for the variable of the first column of the "iris" dataset. HINT: Use `boxplot()`.

Exercises

Exercise 4

Now create a whisker plot for each one of the four input variables of the "iris" dataset in one image. HINT: Use par().

Exercise 5

Create a barplot to breakdown your output attribute. HINT: Use plot().

Exercise 6

Create a scatterplot matrix of the "iris" dataset using the "x" and "y" variables. HINT: Use featurePlot().

Exercise 7

Create a scatterplot matrix with ellipses around each separated group. HINT: Use plot="ellipse".

Exercises

Exercise 8

Create box and whisker plots of each input variable again, but this time broken down into separated plots for each class. HINT: Use `plot="box"`.

Exercise 9

Create a list named "scales" that includes the "x" and "y" variables and set relation to "free" for both of them. HINT: Use `list()`

Exercise 10

Create a density plot matrix for each attribute by class value. HINT: Use `featurePlot()`.

Part B - Regression

Exercise: regression Ames housing data

1) Install the package `AmesHousing` and create a **processed version** of the Ames housing data with the variables `Sale_Price`, `Gr_Liv_Area` and `TotRms_AbvGrd`

2) Create a regression model with `Sale_Price` as dependent and `Gr_Liv_Area` and `TotRms_AbvGrd` as independent variables. Then create seperated models for the two independent variables. Compare the results. What do you think?

Exercise: ridge regression (I)

1) Load the `lars` package and the `diabetes` dataset
2) Load the `glmnet` package to implement ridge regression.

The dataset has three matrices x, x2 and y. x has a smaller set of independent variables while x2 contains the full set with quadratic and interaction terms. y is the dependent variable which is a quantitative measure of the progression of diabetes.

3) Generate separate scatterplots with the line of best fit for all the predictors in x with y on the vertical axis.

4) Regress y on the predictors in x using OLS. We will use this result as benchmark for comparison.

Exercise: ridge regression (I)

1) Load the `lars` package and the `diabetes` dataset
2) Load the `glmnet` package to implement ridge regression.

The dataset has three matrices x, x2 and y. x has a smaller set of independent variables while x2 contains the full set with quadratic and interaction terms. y is the dependent variable which is a quantitative measure of the progression of diabetes.

3) Generate separate scatterplots with the line of best fit for all the predictors in x with y on the vertical axis.

4) Regress y on the predictors in x using OLS. We will use this result as benchmark for comparison.

Exercise: ridge regression (II)

5) Fit the ridge regression model using the `glmnet` function and plot the trace of the estimated coefficients against lambdas. Note that coefficients are shrunk closer to zero for higher values of lambda.

6) Use the cv.glmnet function to get the cross validation curve and the value of lambda that minimizes the mean cross validation error.

7) Using the minimum value of lambda from the previous exercise, get the estimated beta matrix. Note that coefficients are lower than least squares estimates.

8) To get a more parsimonious model we can use a higher value of lambda that is within one standard error of the minimum. Use this value of lambda to get the beta coefficients. Note the shrinkage effect on the estimates.

Exercise: ridge regression (III)

9) Split the data randomly between a training set (80%) and test set (20%). We will use these to get the prediction standard error for least squares and ridge regression models.

10) Fit the ridge regression model on the training and get the estimated beta coefficients for both the minimum lambda and the higher lambda within 1-standard error of the minimum.

11) Get predictions from the ridge regression model for the test set and calculate the prediction standard error. Do this for both the minimum lambda and the higher lambda within 1-standard error of the minimum.

12) Fit the least squares model on the training set.

13) Get predictions from the least squares model for the test set and calculate the prediction standard error.

Exercises part c - trees and bagging

Exercise - `rpart` Kyphosis

Consider the Kyphosis data frame

1) Which variables are in the `kyphosis` dataset
2) Build a tree to classify Kyphosis from Age, Number and Start.

Consider the tree build above.

3) Which variables are used to explain Kyphosis presence?
4) How many observations contain the terminal nodes.

Consider the Kyphosis data frame.

5) Build a tree using the first 60 observations of kyphosis.
6) Predict the kyphosis presence for the other 21 observations.
7) Which is the misclassification rate (prediction error)

Exercise `rpart - iris`

Consider the `iris` data frame

1) Build a tree to classify Species from the other variables.
2) Plot the trees, add nodes information.

Consider the tree build before

3) Prune the the using median complexity parameter (cp) associated to the tree.
4) Plot in the same window, the pruned and the original tree.
5) In which terminal nodes is clasified each oobservations of `iris`?
6) Which Specie has a flower of `Petal.Length` greater than 2.45 and `Petal.Width` less than 1.75.

Exercise: random forests

Download and import example data

1) Download the Data from **here** and read in the `adult.csv` file with `header=FALSE`. Store this in df. Use `str()` command to see the dataframe.

2) Get the column names from the **meta data** and add them to the data frame. Notice the df is ordered - V1,V2,V3,... and so on.

Get an overview of the data

3) Use the `table` command to get the distribution of the `class` feature.
4) Make a binary variable `class`.
5) Use the `cor()` command to see the corelation of all the numeric and integer columns including the class column.

Exercise: random forests

Split the dataset

6) Split the dataset into Train and Test sample. You may use `caTools::sample.split()` and use the ratio as 0.7 and set the seed to be 1000.
7) Check the number of rows of Train and Test
8) We are ready to use decision tree in our dataset. Load the package `rpart` and `rpart.plot`
9) Use `rpart` to build the decision tree on the Train set. Include all features. Store this model in `dec`
10) Use `prp()` to plot the decision tree.

Exercise

1) use the `predict()` command to make predictions on the Train data. Set the method to `class`. Class returns classifications instead of probability scores. Store this prediction in pred_dec.
2) Print out the confusion matrix
3) What is the accuracy of the model. Use the confusion matrix.
4) What is the misclassification error rate? Refer to Basic_decision_tree exercise to get the formula.
5) Lets say we want to find the baseline model to compare our prediction improvement. We create a base model using this code

```
length(Test$class)
base=rep(1,3183)
```

- Use the table() command to create a confusion matrix between the base and Test$class.

6) What is the number difference between the confusion matrix accuracy of dec and base?
7) Remember the predict() command in question 1. We will use the same mode and same command except we will set the method to "regression". This gives us a probability estimates. Store this in pred_dec_reg
8) load the `ROCR` package.

Use the prediction(), performance() and plot() command to print the ROC curve. Use pred_dec_reg variable from Q7. You can also refer to the previous exercise to see the code.

9) `plot()` the same ROC curve but set colorize=TRUE
10) Comment on your findings using ROC curve and accuracy. Is it a good model? Did you notice that ROC prediction() command only takes probability predictions as one of its arguments. Why is that so?

ML Exercises part c3 - Boosting

eXtremely Boost your machine learning Exercises (Part-1)

- eXtreme Gradient Boosting is a machine learning model which became really popular few years ago after winning several Kaggle competitions.
- It is very powerful algorithm that use an ensemble of weak learners to obtain a strong learner.
- Its R implementation is available in xgboost package and it is really worth including into anyone's machine learning portfolio.

Boosting Exercises - first part

Exercise 1

Load xgboost library and download German Credit dataset. Your goal in this tutorial will be to predict Creditability (the first column in the dataset).

Exercise 2

Convert columns `c(2,4,5,7,8,9,10,11,12,13,15,16,17,18,19,20)` to factors and then encode them as dummy variables. HINT: use 'model.matrix()

Exercise 3

Split data into training and test set 700:300. Create xgb.DMatrix for both sets with Creditability as label.

Boosting Exercises - second part

Exercise 4

Train xgboost with logistic objective and 30 rounds of training and maximal depth 2.

Exercise 5

To check model performance calculate test set classification error.

Exercise 6

Plot predictors importance.

Boosting Exercises - third part

Exercise 7

Use `xgb.train()` instead of `xgboost()` to add both train and test sets as a watchlist. Train model with same parameters, but 100 rounds to see how it performs during training.

Exercise 8

Train model again adding AUC and Log Loss as evaluation metrices.

Exercise 9

Plot how AUC and Log Loss for train and test sets was changing during training process. Use plotting function/library of your choice.

Exercise 10

Check how setting parameter eta to 0.01 influences the AUC and Log Loss curves. image_pdf

ML Exercises part d - neural nets

Exercise neural networks (I)

Create example data

1) Set the seed to 42, draw 200 numbers from a unified random distribution between -10 and 10 and store them in a vector named `x`.
2) Create a vector named `y` containing the value of `sin(x)`.

Randomize initial weights

3) Create a vector of 10 random values, picked in the interval [-1,1].

Exercise neural networks (II)

Split the dataset

4) Neural networks have a strong tendency of overfitting data, they become really good at describing the relationship between the values in the data set, but are not effective with data that wasn't used to train your model. As a consequence, we need to cross-validate our model. Set the seed to 42, then create a training set containing 75% of the values in your initial data set and a test set containing the rest of your data.

Create a first model

5) Load the `nnet` package and use the function `nnet` to create a model. Pass your weights via the `Wts` argument and set `maxit=50`. Set `linout=T` and take some time to look at the structure of your model.

Exercise neural networks (IV)

Normilze

8) Now let us use neural networks to solve a classification problem, so let's load the iris data set! It is good practice to normalize your input data to uniformize the behavior of your model over different range of value and have a faster training. Normalize each factor so that they have a mean of zero and a standard deviation of 1, then create your train and test set.

Create model with `nnet`

9) Use the `nnet()` and use a hidden layer of ten neurons to create your model. We want to fit a function which have a finite amount of value as output. To do so, set the linout argument to true. Look at the structure of your model. With classification problem, the output is usually a factor that is coded as multiple dummy variables, instead of a single numeric value. As a consequence, the output layer have as one less neuron than the number of levels of the output factor.

Exercise neural networks (V)

prediction and confusion table

10) Make prediction with the values of the test set.

11) Create the confusion table of your prediction and compute the accuracy of the model.