ST4061 - Statistical Methods for Machine Learning II
ST6041 - Machine Learning and Statistical Analytics II

Exercises

Eric Wolsztynski
2021-2022 @UCC

Document version: 11 January 2022

I. Statistical Learning: data pre-processing

Exercise 1: effect of scaling (example with PCA and logistic regression)

First, create a 2-class scenario by removing the last 50 observations from the **iris** dataset, so as to keep only the *setosa* and *versicolor* species in the reduced dataset.

Note: you need to apply **droplevels()** to the dependent variable **Species** for this classification problem on the reduced dataset.

1. Perform PCA on the unscaled and scaled datasets respectively. Produce a 4-panel figure showing the corresponding scree plots and biplots of the first two principle components.
2. Perform logistic regression of the dependent variable (Species) on all other variables, in both unscaled and scaled versions of the dataset. Compare and discuss on the estimated model coefficients.
3. Fit a LASSO model to both unscaled and scaled versions of the dataset, again using Species as the dependent variable. Compare and discuss on the estimated model coefficients.

Exercise 2: imputing missing data (imputing predictor variables)

1. Consider the **lung** cancer dataset (**library(survival)**). Compare **meal.cal** values between male and female cohorts, and discuss your findings with respect to gender-specific data imputation.
2. Fit Cox proportional hazard models (**coxph()**) to the original, overall imputed and gender-specific imputed datasets, using the cohort-specific sample mean for each data imputation. Compare and discuss on model fitting outputs.
   NB - Example of R call to coxph() with lung dataset:
   coxph(Surv(time,status) ~ ., data=lung)

Exercise 3: imputing missing data (imputing the dependent variable)

1. Considering the **Hitters** dataset (**library(ISLR)**), fit a LASSO model on the subset of complete observations.
2. Replace all missing values for **Salary** (the dependent variable here) with the overall Salary mean in the full dataset, and fit a second LASSO model to this newly imputed dataset.
3. Compare and discuss on model fitting outputs. Can you think of an alternative approach to imputing the dependent variable?

Exercise 4: dataset randomization (example on 10-fold CV)

In this question we analyse linear regression of tree height (**Height**, in feet) with respect to tree diameter (**Girth**, in inches) based on **R**'s dataset **trees** of 31 felled black cherry trees. In particular we focus on the variability associated with the estimation procedure **lm(Height ~ Girth, data=trees)**. (Rings a bell?)

1. Implement 10-fold cross-validation of ordinary linear regression applied to the original dataset trees. Here we focus on the fitting part, so store the train-set regression coefficient estimates. Please run **set.seed(4060)** before you perform any other action for this cross-validation analysis. Quote the cross-validated estimate for the regression slope parameter.
2. Now implement 10-fold cross-validation of ordinary linear regression applied to a *randomized version* of the dataset trees (please used **set.seed(1)** before shuffling the dataset). Again, consider and store only the train-set regression coefficient estimates. Please run **set.seed(4060)** before you perform any other action for this cross-validation analysis. Quote the cross-validated estimate for the regression slope parameter.
3. Compare the sampling distributions of both cross-validation regression slope estimates, using an appropriate boxplot. Perform a two-sided, two-sample t-test to compare these sampling distributions. Next, perform an F-test (**var.test()**) to compare these sampling distributions. Comment on your findings.

Exercise 5: comparing Cross-Validation and Bootstrapping for validation

In this question we compare the statistical characteristics of prediction error estimates obtained from cross-validation and bootstrapping.

1. Go back over the R code for Exercise 4 above: run the code, and comment on the boxplot you obtain. Explain the difference observed between the CV and Bootstrapping outputs. What causes these differences?

2. Since we focus on validation, we must consider using the out-of-bootstrap sample points to evaluate prediction error. Start by evaluating roughly how many points would be in this out-of-bootstrap sample if applying bootstrapping to a sample comprising of 1,000 points.

3. Simulate a dataset of 1,000 data points as follows:

```
set.seed(1)
N = 1000
x = runif(N, 2, 20)
y = 2 + 5*x + rnorm(N)
```

Then apply CV and Bootstrapping to evaluate prediction error, performing R=33 repetitions of 3-fold CV on the one hand, and an appropriate number of bootstrap resample on the other hand. Use the out-of-bootstrap points for prediction validation. Compare the CV and Bootstrapping prediction error outputs for ordinary least squares regression of **y** onto **x**.

II. Regression

Exercise 1: tuning LASSO

1. Consider dataset **ISLR::Hitters** again.
2. Use **glmnet::glmnet()** to perform *regularization-based* and *LASSO* selections, using a grid 10^seq(10,-2,length=100) for λ-values.
3. Compare estimated model coefficients obtained from ridge regression and the LASSO for small enough values of λ. Discuss a possible strategy for final calibration of λ.

Exercise 2: tuning LASSO (continued)

1. Consider dataset **ISLR::Hitters** again, but split it now into a training set and a test set (say a 70%-30% split).
2. Implement tuning strategy discuss in previous exercise on the basis of this data split.
3. Compare output from ridge regression and the LASSO with ordinary least squares.

III. Classification

Exercise 1: kNN

1. Apply the **knn()** function to the **iris** dataset (let's only consider the original 3-class problem) and quote prediction error for *k=1*.
2. Compute the overall classification error rate for the test set with varying *k*.
3. Can you identify a 'better' value for *k*?

Exercise 2: logistic regression of iris data

1. Use a scrambled subsample **x** of the iris dataset using
   is = sample(1:150); x = iris[is[1:100],]
2. Recode **x$Species** into **is.virginica** with values in (0;1) (we're changing the problem formulation slightly, as in previous exercise).
3. Fit GLM model via
   fit <- glm(Species~., data=x, family=binomial(logit))
4. Use **fit** to predict Species of remaining data points:
   testset = iris[-is,]
   y <- testset[,1:4]
   pred <- predict(fit, newdata=y, type="response")
5. Assess prediction performance.

NB: source http://michael.hahsler.net/SMU/EMIS7332/R/

Exercise 3: LDA assumptions

- Problem 1: 2-class problem
  1. Recode variable **Species** in the iris dataset so that Species = 1 if virginica, 0 otherwise. This new classification problem consists in deciding whether a new flower is virginica or not, on the basis of four measurements: Sepal.Width, Sepal.Length, Petal.Width and Petal.Length.
  2. Explore the dataset in terms of predictors distributions w.r.t. Species**.**
  3. Try out Bartlett's test on these.
  4. Interpret the model fit summary.
- Problem 2: Repeat for the 3-class problem formulation.

Exercise 4: LDA of iris data

- Problem 1: 2-class problem
  1. Recode variable **Species** in the **iris** dataset so that Species = 1 if virginica, 0 otherwise, as in the previous exercise.
  2. Perform LDA and QDA on a random train set of 100 specimens.
  3. Assess prediction performance on the corresponding test set.
- Problem 2: 3-class problem
  1. Repeat for the 3-class problem formulation.

Exercise 5: comparing classifiers

1. Compare error rates for kNN, logistic regression, LDA and QDA classifiers for the **Default** and **Caravan** datasets from **library(ISLR)**.
2. Think about what drives these performances.
3. Repeat your analysis on standardized data and review your observations from (2)…
4. Use package **pROC** to obtain ROC curves for some of these classifiers, and compare the corresponding areas under the curve (AUCs).

Exercise 6: further practice

1. Recreate the ROC curve "by hand" for the classifiers used in Exercise 5, using the output from table().
2. Repeat Exercise 5 on the iris dataset.

IV. Tree-based methods

Exercise 1: growing and pruning a tree

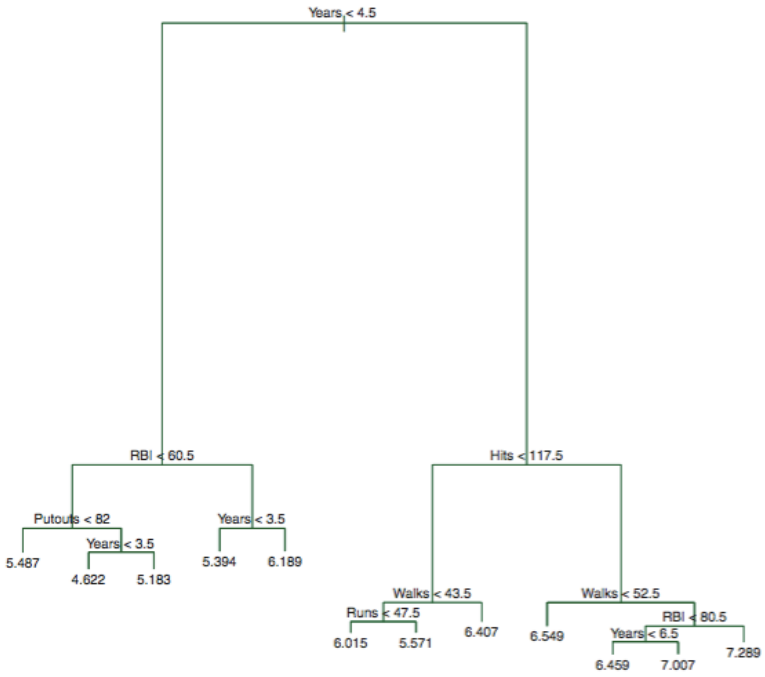
1. Consider dataset **ISLR::Carseats**, and load **library(tree).**
2. Let's consider running a classification tree first. Create a binary response variable as follows:
   High = ifelse(Carseats$Sales<=8), 'No', 'Yes')
   CS = data.frame(Carseats, High)
3. Grow a classification tree for this set using all variables except for **Sales**, and visualise.
4. Prune the tree using **cv.tree().**
5. Plot CV error as a function of tree size (number of terminal nodes) and of cost complexity α.
6. Work out how to use the cross-validated output…

Exercise 2: apply CV and ROC analysis

1. Consider dataset **ISLR::Carseats**, split into training and test sets as done before (say 50%-50% split).
2. Repeat the tree-based classification analysis of previous exercise in this split-set setting.
3. Evaluate misclassification error.
4. Plot the corresponding ROC.
5. Can you compute a cross-validated AUC?

Exercise 3: find the tree

Compute the following optimal tree for the full dataset **ISLR::Hitters** (no CV):



Exercise 4: grow a random forest

1. Recall **ISLR::Carseats** set with binary response **High**.
2. Grow an unpruned classification tree as before using **tree()**.
3. Fit **randomForest::randomForest()** (using default values) and compare classification rates from these two fits.
4. Perform bagging of trees, and compare to the other two models.
5. Now split into training and test sets as done before (say 50%-50% split), and compare test-set predictions for these three models.
6. What do you make of this?

Exercise 5: benchmarking

1. Add the CART and random forest methodologies to the 10-fold CV implementation of Exercise 5 of the "Classification" section, analysing the **Default** and **Caravan** datasets from library**(ISLR)**.
2. Compare all output predictions and discuss.

Exercise 6: variable importance from a random forest

1. Re-run the code of Exercise 4 to obtain a random forest and a bagging output for the 2-class classification problem with response variable **High**. (We do not worry about data splitting here.)
2. Extract a summary of variable importance from each fit and discuss.

Exercise 7: gradient boosting

1. Considering again the 2-class classification problem with response variable **High**, from the **ISLR::Carseats** dataset, split the data so as to save 100 observations as a validation set.
2. Using library **gbm**, fit a gradient boosting model to the training data, using an ensemble of 5,000 stumps. Inspect the output.
3. Generate predictions form this GB model to the validation set, as well as from a random forest fitted using 5,000 trees. Compare the two classifiers in terms of an ROC analysis.

Exercise 8: gradient boosting using caret

Considering the demo R script demonstrating how to use R's widely popular **caret** package. Update this script so as to include gradient boosting (using the **gbm** implementation).

V. Support Vector Machines

Exercise 1: using SVM

1. Consider, again, dataset **ISLR::Carseats**, and, again, modify this regression problem into a classification one by creating a binary response variable as follows:
   High = ifelse(Carseats$Sales<=8), 'No', 'Yes')
   CS = data.frame(Carseats, High)
2. Set the random seed to 4061 and randomly select 350 data points as your training set. Set aside the rest of the data as a validation set.
3. Run the following command, explain what happens, and fix the "problem":
   svmo = svm(x.train, y.train, kernel='polynomial')
4. Fit two SVM classifiers to the appropriately "massaged" training set, one using a linear kernel function another one using a polynomial kernel function.
5. Compare the two SVM outputs...
   a. ... visually;
   b. ... in terms of training fit;
   c. ... in terms of test set prediction accuracy;
   d. ... in terms of test set AUC performance.
6. Conclusions?

Exercise 2: 3-class problem

1. Consider the iris dataset. Set the random seed to 4061 and randomly select 100 data points as your training set. Set aside the rest of the data as a validation set.
2. Fit SVMs using successively linear, polynomial and radial kernels.
3. Report the number of support vectors used in each model, and comment.
4. Evaluate overall test set prediction accuracy.
5. Using e1071::tune(), tune the SVM based on a radial kernel. Refit the SVM to the training data using optimal parameters obtained from this analysis, and compare test set prediction performance against the un-tuned radial-kernel SVM one. Comment...

Exercise 3: SVM using caret

1. Bring back the classification setting of Exercise IV.8 (based on the Hitters dataset). Set the random seed to 4061 and randomly select 70% of the sample as your training set. Set aside the rest of the data as a validation set.
2. Using the caret package, train a random forest, an SVM with linear kernel, and an SVM with radial kernel to the training data. Generate corresponding confusion matrix summaries, compare and comment.

Exercise 4: SVM-based regression

1. Consider the iris dataset, but now using Sepal.Length as the response variable. Set the random seed to 4061 and randomly select 100 data points as your training set. Set aside the rest of the data as a validation set.

2. Using the caret package, train an SVM with radial kernel to the training data using straightforward cross-validation.
3. Generate the corresponding test set predictions, evaluate the test set prediction MSE and plot the predicted test values against their actual values.

VI. Neural Networks

Exercise 1: neuralnet basics

See below example of a simple neural network:

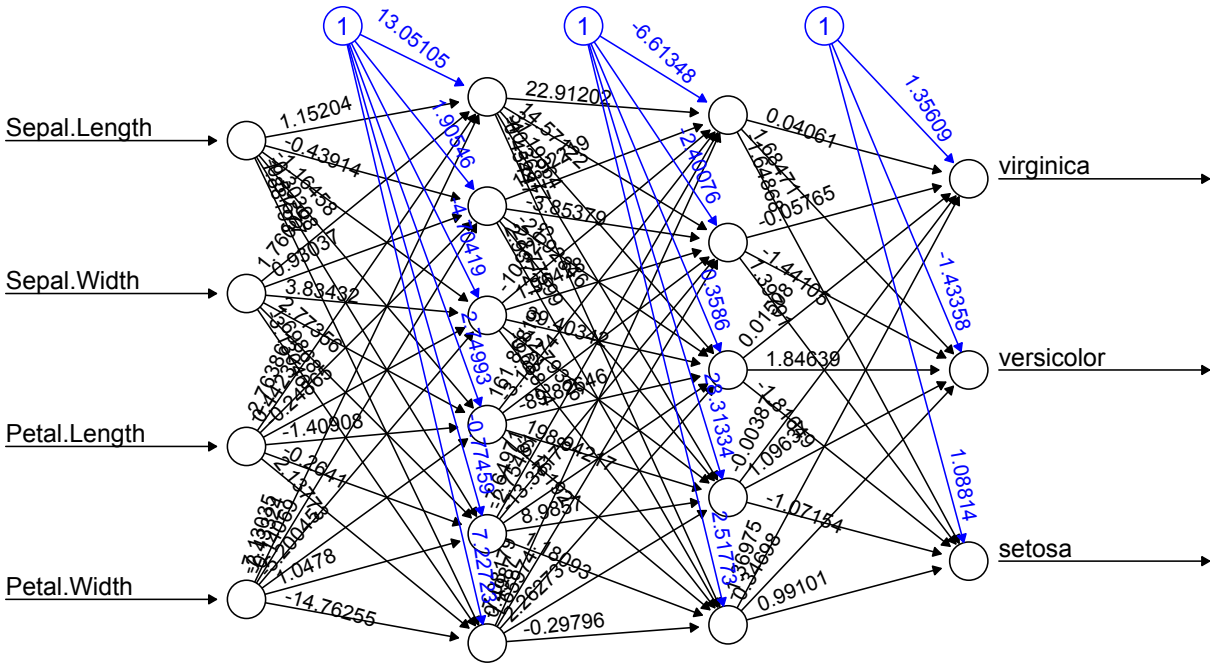
```
library(MASS)
library(neuralnet)
nms = names(Boston)[-14]
set.seed(4061)
f = as.formula(paste("medv ~", paste(nms, collapse = " + ")))
out.nn = neuralnet(f, data=Boston, hidden=c(10), rep=5, linear.output=FALSE)
```

1. What type of neural network does this code implement?
2. Reproduce the above analysis using the same neural network architecture but without using an activation function. Comment on your findings (no need to wait for the run to complete!).

Further work: Adapt this code to a split training-test set scenario and play with it!

Exercise 2: plot a neural net

Obtain a figure similar to the one below, for the iris dataset (using all 150 data points when fitting the model).



Exercise 3: neuralnet v nnet

Load dataset **MASS::Boston** and perform a 70%-30% split for training and test sets respectively. Use **set.seed(4061)** when splitting the data and also every time you run a neural network.

1. Compare single-layer neural network fits from the **neuralnet** and **nnet** libraries. Can you explain any difference you may find?
2. Change the "threshold" argument value to 0.001 in the call to **neuralnet**, and comment on your findings (this run might take a bit more time to converge).

Exercise 4: nnet for regression

Fit a single-layer feed-forward neural network using **nnet** to the following dataset:

```
dat = na.omit(Hitters)
set.seed(4061)
itrain = sample(1:n, round(.7*n), replace=FALSE)
dat.train = dat[itrain,]
```

Report on fitted values. Good luck.

Exercise 5: neural networks using caret

Consider the data of Exercise 4 above.

1. Fit a single-layer (10-neuron) FFNN to it using **caret**, with decay set to 0.1
2. Now tune the single-layer FFNN to the same data and compare.
3. Now use a 3-layer FFNN to fit the same data, comparing **neuralnet**'s and **caret**'s implementations. Fine-tune this 3-layer network using the **caret** package.
4. Use the **mlpML** implementation from caret as another NN model for comparison.

Exercise 6: Olden index

Use **NeuralNetTools::olden()** to compute variable importance for the following datasets, fitting a 7-neuron single-layer FFNN (**nnet**) each time:

1. The iris dataset (use the full set);
2. The Boston dataset.

Compare with variable importance assessment obtained from a random forest.

VII. Model selection

Exercise 1: best subset selection

Consider dataset **ISLR::Hitters** and remove all NA's from it (**na.omit(Hitters)**).

1. Use **leaps::regsubsets()** to perform *best subset* selection. What is the size of the largest model considered by default?
2. Interpret the standard regsubsets() output.
3. Plot the RSS as a function of number of variables.
4. Increase the desired size of the final model to 19, and find out which submodel yields to:
   - smallest RSS
   - highest adjusted R^2

   (try **summary(…)$which** and **names(which(summary(…)$which[…,]))**)

Exercise 2: stepwise selection

Consider dataset **ISLR::Hitters** again, removing observations with missing data.

1. Use **leaps::regsubsets()** to perform *forward* and *backward* selections.
2. Compare plots of RSS and Adjusted R^2 w.r.t. model size.
3. Compare the covariates found in the 4-variable models selected via each approach.
4. Explore final model selections in any other way you like.

Exercise 3: predict from leaps::regsubsets()

Consider dataset **ISLR::Hitters** again, removing observations with missing data, and using a train/test split with 150 observations in the training set.

1. Use **leaps::regsubsets()** to perform *forward* selection.
2. Try generating test-set predictions using:
   - predict()
   - your own way...

Exercise 4: fit and predict from stats::step()

Perform stepwise selections on the full Hitters dataset (once you removed observations with missing values, as in Exercise 2) using function **step()** from R's base package **stats**.

1. Compare the backward selection outputs from step() and regsubsets().
2. Quote the model fit summary for the final model selected via backward selection using step().

Exercise 5: predict from stats::step()

Consider dataset **ISLR::Hitters** again, removing observations with missing data, and using a train/test split with 150 observations in the training set, as in Exercise 3.

1. Use **stats::step()** to perform *forward* selection.
2. Try generating test-set predictions using **predict()**.