MSA220/MVE440 Exam June 2020
including exercise 3

Mathematical Sciences
Chalmers University of Technology and University of Gothenburg

Spring semester 2020
Deadline: 12th June 2020

1 General info

Examiner: Rebecka Jörnsten

Course coordinator: Felix Held

Official start date: 28th May 2020

Hard Deadline: 12th June 2020

Setup: Since this is a take-home exam you are allowed to freely use the course material and similar resources available to you. However, we require you to work on the exam **individually**.

How to submit: There are two separate assignments on Canvas where you upload

1. your **final report in PDF format**. Upon submission, the document will be automatically sent to Urkund[1]. Plagiarism is not allowed[2].
2. your **code as a ZIP file**.

You have to **submit both** for a valid submission!

Grading:

- Exercise 1 below determines pass or fail. This means that if you give a satisfactory answer to this part you are guaranteed a 3 at Chalmers and a G at University of Gothenburg (GU). However, if you fail this part, then the other two exercises will not count.
- The additional two exercises determine if you get a higher grade
  - For a 4 at Chalmers or a VG at GU you need to get the equivalent of one well worked-out answer to these exercises
  - For a 5 at Chalmers you need to get two well worked-out answers to these exercises

  By "equivalent of one well worked-out answer" we mean that you either do really well on only one additional exercise, or reasonably well on both exercises. See below for what is included in a well worked-out answer.

  Note that your answer to exercise 1 is only used to determine pass or fail and exercises 2 and 3 are the ones determining the higher grade.

[1] https://www.urkund.com/, a plagiarism checker that makes it very easy for us to see if your reports overlap, and to which extent

[2] Chalmers has a document that demonstrates how to avoid plagiarism (https://student.portal.chalmers.se/en/chalmersstudies/policy-documents/Documents/20090920_Academic_Honesty.pdf)

2 Formalities

2.1 Observe

- Answers are limited to **maximally 1500 Words** per exercise. Figure and table captions are considered separate.
- A **well worked-out answer** is one where
    1. you choose and use methods appropriately,
    2. you discuss your results correctly,
    3. and give clear and concise answers to the questions asked in the exercises below.
- Write your answers as clearly as possible!
- Motivate, motivate, motivate! Give clear motivations for why you use methods/do a certain analysis/approach the problem the way you do.
- Do not contradict your findings. Just because you expect to observe something does not mean that you actually do. If you end up in this situation, reason about why you get results different from your expectations.

2.2 Structure

The answers to the exercises should contain the following aspects in the order stated below.

1. **Methods:** Clearly describe your approach, simulation setup, chosen methods and models, ..., and motivate your choices. Also describe how you selected model parameters.

    Answer the questions: "**What** was done to answer the question(s) and **why** was it done this way?"

2. **Results:** Describe your findings short and concisely. Focus on results that are related to answering the question(s) in the exercise. If you attach figures and tables you have to refer to them in your text. Figure and table captions must explain all elements of a figure or table.

3. **Discussion:** Interpret your results in light of the exercise's question(s) and argue why and how your results support your answer.

3 Exercises

3.1 Passing grade exercise

This part of the exam determines **pass or fail**.

3.2 Classification and variable selection

For this exercise, you are provided with a high-dimensional dataset with binary labels (Data available on Canvas). You receive the training data in form of *(1)* a feature matrix $\mathbf{X}_1 \in \mathbb{R}^{n_1 \times p}$, where $n_1 = 323$ and $p = 800$, and *(2)* a vector of class labels $\mathbf{y}_1 \in \{0, 1\}^{n_1}$.

In addition, you are given a validation dataset, also consisting of a feature matrix $\mathbf{X}_2 \in \mathbb{R}^{n_2 \times p}$ and a vector of labels $\mathbf{y}_2 \in \{0, 1\}^{n_2}$ with $n_2 = 175$.

Your **tasks for this exercise** are to

1. analyse the data with two different classification methods trained on the training dataset
2. compare the methods' performance on the validation dataset
3. determine the best predictors for classification and explain how confident you are in your selection

Some pointers

- Make sure to perform an exploratory data analysis first
- Make sure to address all three tasks mentioned above
- Make sure to motivate your choices and justify your results.
  - Why did you choose those methods?
  - If there are hyperparameters[3], how did you choose them?
  - Why do you believe the methods perform equal/one better than the other/...
- Pay attention to how you measure performance.
- Recommended approaches are
  - Sparse classification methods such as ℓ_1–/elastic net–/group lasso–regulated logistic regression or shrunken centroids.
  - Random Forests in the classification setting using its variable importance feature
  - Possibly even traditional variable selection approaches such as forward selection or best subset selection.

[3] There always are hyperparameters.

4 Additional higher grade exercises

Once you pass the exercise above, these exercises determine if you get a higher grade.

4.1 High-dimensional clustering

You just started to work as a data scientist at a startup trying to invent the future of sustainable robotic crop farming. One sub-project is the automatic detection of soil quality. During an initial exploration phase your colleagues assembled a dataset of soil samples from potential growing sights. The samples were analysed in a laboratory and a multitude of descriptors were extracted. You are given the dataset in form of a data matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$ with $n = 302$ soil samples and $p = 728$ descriptors (Data available on Canvas).

The company wants you to perform an exploratory data analysis and find clusters in the data to be able to comprehend it better. A pilot study indicated that some descriptors might be more informative about soil quality than others and management would like to know up to five variables that are most indicative of each found cluster.

Throughout this task, it is crucial that you motivate your decisions with clear numerical and/or graphical results. Otherwise management will not believe you.

4.2 Improving variable selection in the Lasso

Consider data coming from a sparse linear regression model

$$y_l = \boldsymbol{\beta}_{\text{true}}^\top \mathbf{x}_l + \varepsilon_l \quad \text{for all } l = 1, \dots, n \tag{1}$$

where

- errors ε_l are uncorrelated, have mean zero, and variance σ^2,
- $\boldsymbol{\beta}_{\text{true}}$ is a sparse vector, i.e. only s per-cent of elements are non-zero, where s is small[4],
- and $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$ is an *iid* sample.

Let $\mathcal{A} = \{i : \boldsymbol{\beta}_{\text{true}}^{(}i) \neq 0\}$ be the set of indices belonging to the non-zero elements of $\boldsymbol{\beta}_{\text{true}}$. When performing variable selection, we would hope that a variable selection method can recover the set $\mathcal{A}$ as accurately as possible.

The lasso estimator for a fixed $\lambda > 0$ was introduced in the lecture as the solution to the optimisation problem

$$\hat{\boldsymbol{\beta}}_{\text{Lasso}} = \arg\min_{\boldsymbol{\beta} \in \mathbb{R}^p} \frac{1}{2n} \sum_{l=1}^{n} \|y_l - \boldsymbol{\beta}^\top \mathbf{x}_l\|_2^2 + \lambda \|\boldsymbol{\beta}\|_1 \tag{2}$$

where $\mathbf{y} = (y_1, \dots, y_n)^\top \in \mathbb{R}^n$, $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)^\top \in \mathbb{R}^{n \times p}$, and $\mathbf{x}_l \in \mathbb{R}^p$. The lasso is able to perform variable selection by setting entries of $\hat{\boldsymbol{\beta}}_{\text{Lasso}}$ to exact zeros, thereby selecting only variables corresponding to non-zero entries.

[4] How small s should be is dependent on the eigenvalues of the feature matrix $\mathbf{X}$, p, and the method used for variable selection, but a sparsity level of 10% is reasonable for this exercise.

Unfortunately, the paper Zou (2006)[5] showed that the lasso is not capable of accurately recovering the subset $\mathcal{A}$ even in quite simple setups[6] and for $n \to \infty$ To solve this problem, the paper introduced the *adaptive lasso* which solves the problem

$$\hat{\boldsymbol{\beta}}_{\text{AdaLasso}} = \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\arg\min} \frac{1}{2n} \sum_{l=1}^{n} ||y_l - \boldsymbol{\beta}^\top \mathbf{x}_l||_2^2 + \lambda \sum_{i=1}^{p} w_i |\boldsymbol{\beta}^{(i)}| \tag{3}$$

By choosing the weights w_i in a smart way the adaptive lasso can exactly recover the subset $\mathcal{A}$ for $n \to \infty$. In the paper it was shown that the following procedure leads to useful weights

1. Given a feature matrix $\mathbf{X}$ and response vector $\mathbf{y}$, compute either
   - $\hat{\boldsymbol{\beta}}_{\text{OLS}}$, the least squares estimates for $\mathbf{X}$ and $\mathbf{y}$, if $p < n$, or
   - $\hat{\boldsymbol{\beta}}_{\text{Ridge}}$, the coefficients from a cross-validated ridge regression on $\mathbf{X}$ and $\mathbf{y}$, if $p \geq n$
2. Then set

$$w_i = \frac{1}{|\hat{\boldsymbol{\beta}}^{(i)}_{\text{Ridge}}|^\gamma} \quad \text{or} \quad w_i = \frac{1}{|\hat{\boldsymbol{\beta}}^{(i)}_{\text{OLS}}|^\gamma} \quad \text{for all } i = 1, \dots, p \tag{4}$$

   for $\gamma > 0$ and use these weights to compute the adaptive lasso estimate in Eq. (3).

There are two ways how you can perform the computation of the adaptive lasso.

1. Some packages, such as the R package `glmnet` offer a specific input option for weights. In `glmnet` this option is called `penalty.factor`. Assigning the weight vector $\mathbf{w} = (w_1, \dots, w_p)$ to this option solves the problem in Eq. (3) instead of in Eq. (2).
2. Alternatively, the data can be modified. Given a matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$ of features and a vector $\mathbf{y}$ of responses, perform the following.
   (a) Define $\mathbf{X}' = \mathbf{X}\,\text{diag}(\mathbf{w})^{-1}$, where $\mathbf{w} = (w_1, \dots, w_p)$ is the vector of weights from the ridge regression estimates and $\text{diag}(\mathbf{w})$ is a diagonal matrix with the elements of $\mathbf{w}$ on the diagonal, i.e. divide column $\mathbf{X}^{(:,i)}$ element-wise by the weight w_i.
   (b) Use $\mathbf{X}'$ and $\mathbf{y}$ to perform standard lasso regression to get an estimate $\hat{\boldsymbol{\beta}}'_{\text{Lasso}}$.
   (c) Compute $\hat{\boldsymbol{\beta}}_{\text{AdaLasso}} = \text{diag}(\mathbf{w})^{-1}\hat{\boldsymbol{\beta}}'_{\text{Lasso}}$.

Tasks for this exercise

Perform a simulation study and do the following

1. Explore the capabilities of the lasso and the adaptive lasso to recover the set of true non-zero coefficients $\mathcal{A}$ for small n to large n (for fixed p)[7]
2. Make sure your results are reliable, i.e. base them on repeated simulations
3. Quantify the differences between the two methods with suitable performance metrics and discuss!

Note on hyperparameters

The described procedure for the adaptive lasso contains two or three hyperparameters, depending on whether the OLS or ridge regression estimate is used. The penalisation parameter in the initial ridge regression can be chosen separately once for each simulated dataset. The parameter λ in Eq. (3) and the exponent $\gamma > 0$ have to be selected jointly since they strongly influence each other.

[5] `http://users.stat.umn.edu/~zouxx019/Papers/adalasso.pdf`

[6] Note that this does not mean that the lasso is useless. Often the coefficients that are chosen to be non-zero in $\hat{\boldsymbol{\beta}}_{\text{Lasso}}$ even though they are zero in $\hat{\boldsymbol{\beta}}_{\text{true}}$ will still be found to be close to zero, which is often enough for practical purposes. Here however, we are interested in exact recovery of the non-zero coefficients in $\hat{\boldsymbol{\beta}}_{\text{true}}$.

[7] To avoid ending up in computational run-time nirvana consider something like $p = 50$, $s = 10\%$, signal-to-noise ratio around 1, and $n > 50$. Go easy on increasing n, this parameter will have the largest impact on how slow your computations go.