Uppsala University

Introduction to Machine Learning, Big Data, and AI

Assignment 1

General information

- The recommended tool in this course is R (with the IDE R-Studio). You can download R **here** and R-Studio **here**. You can use Python and Jupyter Notebooks, although the assignments may use data available only through the R package, a problem you would need to solve yourself.
- Report all results in a single, `*.pdf`-file. *Other formats, such as Word, Rmd, Jupyter Notebook, or similar, will automatically be failed.* Although, you are allowed first to knit your document to Word or HTML and then print the assignment as a PDF from Word or HTML if you find it difficult to get TeX to work.
- You should submit the report to **Studium**.
- To pass the assignments, *you should answer all questions not marked with **, although minor errors are ok.
- To get VG on the assignment, *all questions should be answered, including questions marked with a **, although minor errors are ok.
- A report that does not contain the general information (see the **template**), will be automatically rejected.
- When working with R, we recommend writing the reports using R markdown and the provided **R markdown template**. The template includes the formatting instructions and how to include code and figures.
- Instead of R markdown, you can use other software to make the pdf report, but you should use the same instructions for formatting. These instructions are also available in **the PDF produced from the R markdown template**.
- The course has its own R package `uuml` with data and functionality to simplify coding. To install the package just run the following:
  1. `install.packages("remotes")`
  2. `remotes::install_github("MansMeg/IntroML", subdir = "rpackage")`
- We collect common questions regarding installation and technical problems in a course Frequently Asked Questions (FAQ). This can be found **here**.
- Deadlines for all assignments are **Sunday 23.59**. See the course page for dates.
- If you have any suggestions or improvements to the course material, please post in the course chat feedback channel, create an issue **here**, or submit a pull request to the public repository.

1 General Questions

You will be able to answer the following questions based on the reading assignments for this assignment. See the course plan for detailed reading **here**.

1. Describe, with your own wards, what Efron (2020) see as the difference between the *traditional regression methods* and the *pure prediction algorithms*. (1-2 paragraphs)
2. In many machine learning applications, we use Stochastic Gradient Descent, even though other optimization algorithms that uses second order derivatives usually is better. Why is this the case? Describe with your own words (max 1 paragraph)
3. Goodfellow et al (2017) describe machine learning algorithms with tasks (T), performance (P) and experience (E). Describe these three concepts with your own words. (1-2 paragraphs)
4. Describe the free lunch theorem with your own words? (max 1 paragraph)

2 Basic, Stochastic, and Mini-Batch Gradient Descent

This assignment will study different ways to optimize common objective functions in many areas of Machine Learning, namely, stochastic gradient descent. Here we will test to implement these optimizers for a well-known model, logistic regression.

We are going to work with this data as a test case:

```
library(uuml)
data("binary")

binary$gre_sd <- (binary$gre - mean(binary$gre))/sd(binary$gre)
binary$gpa_sd <- (binary$gpa - mean(binary$gpa))/sd(binary$gpa)
X <- model.matrix(admit ~ gre_sd + gpa_sd, binary)
y <- binary$admit
```

2.1 Implement the gradient for logistic regression

The likelihood function for logistic regression is

$$L(\theta, \mathbf{y}, \mathbf{X}) = \prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{1-y_i}\,,$$

where

$$\log \frac{p_i}{1-p_i} = \mathbf{x}_i\theta\,,$$

and $\mathbf{x}_i$ is the ith row from the design matrix $\mathbf{X}$ and $\theta \in \mathbb{R}^P$ is a $1 \times P$ matrix with the parameters of interest.

Commonly, to find maximum likelihood estimates of θ, we usually use the log-likelihood as the objective function we want to optimize, i.e.

$$
\begin{aligned}
l(\theta, \mathbf{y}, \mathbf{X}) &= \sum_{i=1}^{n} y_i \log(p_i) + (1 - y_i) \log(1 - p_i) && (1) \\
&= \sum_{i=1}^{n} y_i \mathbf{x}_i \theta + \log(1 - p_i) && (2) \\
&= \sum_{i=1}^{n} y_i \mathbf{x}_i \theta - \log(1 + \exp(\mathbf{x}_i \theta)) \,. && (3)
\end{aligned}
$$

Although, in our case we instead want to minimize the negative log likelihood $\mathrm{NLL}(\theta, \mathbf{y}, \mathbf{X}) = -l(\theta, \mathbf{y}, \mathbf{X})$.

1. Derive the the gradient for $\mathrm{NLL}(\theta, \mathbf{y}, \mathbf{X})$ with respect to θ.
2. Implement the gradient as a function in R. Below are two examples of how it should work. Note, `l_grad(y, X, theta)` return the gradient of $(1/n)l(\theta, \mathbf{y}, \mathbf{X})$.

```
l_grad(y, X, theta = c(0,0,0))

## (Intercept)      gre_sd      gpa_sd
##     -0.1825      0.0857      0.0829
```

```
l_grad(y, X, theta = c(-1,0.5,0.5))

## (Intercept)      gre_sd      gpa_sd
##      0.0217     -0.0395     -0.0426
```

2.2 Implement Gradient Descent

We now have the primary tool for implementing gradient descent and stochastic gradient descent.

1. Implement the log-likelihood l or the negative log-likelihood NLL in R. Note that you will have to do gradient ascent if you use the log-likelihood.

```
l(y, X, theta = c(0,0,0))

## [1] -277.2589
```

```
l(y, X, theta = c(-1,0.5,0.5))

## [1] -244.5342
```

2. Run logistic regression in R to get an MLE estimate of `theta` using the `glm()` function. Print the three parameter estimates.

3. Implement the following gradient descent algorithms:

   (a) ordinary (full/batch) gradient descent

   (b) stochastic gradient descent

   (c) mini-batch (stochastic) gradient descent using ten samples to estimate the gradient

4. Try different learning parameters η and for roughly 500 epochs. When does the algorithm converge or diverge? Visualize the epochs (full data iterations, x-axis) and the log-likelihood value *for all observations* (y-axis). Show at least one plot per algorithm that converges for each parameter θ (i.e. three Figures), also include the true values you got from using the `glm()` function above. Describe your conclusions. Show the code (loop) you use.

3 Regularized Regression

The datasets `prob2_train` and `prob2_test` contains simulated data with 240 explanatory variables (`V1-V240`) and 1 numerical response variable (`y`). As per the dataset names, the first dataset contains training data and the second contains test data for this problem. To access the data, just run:

```
library(uuml)
data("prob2_train")
data("prob2_test")
dim(prob2_train)

## [1] 200 241
```

You should do the following and present the results in your report:

1. Fit a linear model to the training data. What are the results? Why does this happen?
2. Use `cv.glmnet` from the `glmnet` package to fit a linear lasso regression to the training data. Describe what the function does. Include the plot showing the MSE for different values of λ in your report and describe how to interpret it.
3. Have a look at the coefficients you get from the λ_{min} and λ_{1se} models. Describe the resulting models in the report, e.g., if any variables have been removed from the model. But please do not print all 240 coefficients!
4. Use `cv.glmnet` from the `glmnet` package to fit a linear ridge regression to the training data.
5. Use the models (lasso/ridge, λ_{min} or λ_{1se}) to make predictions for the test data. Present the MAE and RMSE of the four models in a table in your report. Discuss the results, comparing the interpretability and predictive performance of the models.

4 *Gradient Descent for penalized logistic regression

This assignment is the VG point assignment. If you do not want to get a VG-point, you can ignore this assignment. Here we use the same data as in task 1 above, namely

```
library(uuml)
data("binary")

binary$gre_sd <- (binary$gre - mean(binary$gre))/sd(binary$gre)
binary$gpa_sd <- (binary$gpa - mean(binary$gpa))/sd(binary$gpa)
X <- model.matrix(admit ~ gre_sd + gpa_sd, binary)
y <- binary$admit
```

In ridge regression we penalize the regression coefficients by λ. This gives the likelihood function for logistic regression with ridge penalty as

$$l_r(\theta, \mathbf{y}, \mathbf{X}, \lambda) = l(\theta, \mathbf{y}, \mathbf{X}) + \lambda \sum_{i=1}^{P} \theta_i^2 \,, \tag{4}$$

where $l(\theta, \mathbf{y}, \mathbf{X})$ is defined as in task 1 above. We also want to minimize the negative log likelihood $\mathrm{NLL}_r(\theta, \mathbf{y}, \mathbf{X}, \lambda) = -l(\theta, \mathbf{y}, \mathbf{X}, \lambda)$ here as well.

Note! In general, we should not regularize the intercept (see Hastie et al., 2009, Ch. 3.4). Hence below the gradient and log-likelihood only regularize the parameters `gre_sd` and `gpa_sd`.

1. Derive the the gradient for $\mathrm{NLL}_r(\theta, \mathbf{y}, \mathbf{X}, \lambda)$ with respect to θ.
2. Implement the gradient as a function in R. Below are two examples of how it should work. Note, `lr_grad(y, X, theta, lambda)` return the gradient of $(1/n)l(\theta, \mathbf{y}, \mathbf{X}, \lambda)$.

```
lr_grad(y, X, theta = c(0,0,0), lambda = 0)

## (Intercept)      gre_sd      gpa_sd
##     -0.1825      0.0857      0.0829
```

```
lr_grad(y, X, theta = c(0,0,0), lambda = 1)

## (Intercept)      gre_sd      gpa_sd
##     -0.1825      0.0857      0.0829
```

```
lr_grad(y, X, theta = c(-1,0.5,0.5), lambda = 1)

## (Intercept)      gre_sd      gpa_sd
##      0.0217      0.9600      0.9570
```

3. Implement the log-likelihood lr or the negative log-likelihood NLL_r in R.

```r
lr(y, X, theta = c(0,0,0), lambda = 0)

## [1] -277.26
```

```r
lr(y, X, theta = c(0,0,0), lambda = 1)

## [1] -277.26
```

```r
lr(y, X, theta = c(-1,0.5,0.5), lambda = 1)

## [1] -244.03
```

4. Run logistic regression with ridge penalty in R using `glmnet` estimate of `theta`. Set `lambda` to 1 and `alpha` to 0 to run a penalized logistic regression. You also need to remove the intercept from `X`. *Note that `glmnet`. Note! Your results might differ slightly due to a different implementation.*

5. Implement the following gradient descent algorithms for the penalized logistic objective:

   (a) ordinary (full/batch) gradient descent

   (b) mini-batch (stochastic) gradient descent using ten samples to estimate the gradient

6. Try different learning parameters η and different λ. When does the algorithm converge or diverge? Visualize the iterations (x-axis) and the log-likelihood (y-axis). Show at least one plot per algorithm that converges.