CS4100/CS5100 Assignment 1

November 3, 2021

This assignment must be submitted by Tuesday 9 November, at 5pm. Feedback will be provided within ten working days of the submission deadline.

1 Instructions

In order to submit, copy all your submission files into a directory, e.g., `DA`, and create a compressed file, e.g., `DA.zip`. Upload `DA.zip` to Moodle through one of the following pages:

Assignment1 for CS5100
Assignment1 for CS4100

You should submit files with the following names:

- `GD.R` or `GD.m` should contain the source code for your program (these can be split into files such as `GD4.R` for task 4, `GD6.R` for task 6, etc.);
- optionally, `other.R` or `other.m`, which are all other auxiliary files with scripts and functions (please avoid submitting unnecessary files such as old versions, back-up copies made by the editor, etc.);
- `report.pdf` should contain the numerical results and a discussion about them.

The files you submit cannot be overwritten by anyone else, and they cannot be read by any other student. You can, however, overwrite your submission as often as you like, by resubmitting, though only the last version submitted will be kept. Submissions after the deadline will be accepted but they will be automatically recorded as late and are subject to College Regulations on late submissions. Please also note that all your submissions will be graded anonymously, so your name should not appear anywhere in your submission.

The deadline for submission is **Tuesday, 9 November, 5 pm**. An extension can only be given by the department office (but not by the lecturer).

> **Note:** All the work you submit should be solely your own work. Coursework submissions are routinely checked for this.

Learning outcomes assessed

The learning outcomes assessed are:

- develop, validate, evaluate, and use effectively machine-learning and statistical algorithms
- apply methods and techniques such as probabilistic forecasting, logistic regression, and gradient descent

- extract value and insight from data
- implement machine-learning and statistical algorithms in R or MATLAB

Marking criteria

To be awarded full marks you need both to submit correct code and to obtain correct results on the given data set. Even if your results are not correct, marks will be awarded for correct or partially correct code (up to a maximum of 70%). Correctly implementing gradient descent (Task 1) will give you at least 60%.

2 Tasks

1. Implement the logistic regression algorithm using gradient descent (as described in the course slides or in Sections 11.3–11.5 of The Elements of Statistical Learning), i.e. use gradient descent to train a learning machine $F : \mathcal{X} \times \Lambda \to [0, 1]$, where $\mathcal{X} = \mathbb{R}^d$ and $\Lambda = \mathbb{R}^{d+1}$, defined as

$$F(X, \lambda) = \sigma\left([1, X^T]\lambda\right), \quad \sigma(t) = \frac{1}{1 + e^{-t}}$$

and interpreted as *the probability of the object X to have label $Y = 1$*. Given a data set

$$\mathcal{D} = \{(x_n, y_n), x_n \in \mathbb{R}^d, y_n \in \{0, 1\}\}_{n=1}^N$$

The objective function, which your program should minimize, is the *negative log-likelihood*, i.e.

$$\ell(\lambda, \mathcal{D}_{train}) = -\sum_{n:y_n=1} \log F(x_n, \lambda) - \sum_{n:y_n=0} \log(1 - F(x_n, \lambda)) \quad (1)$$

$$= -\sum_{n=1}^{N} \log\left(y_n F(x_n, \lambda) + (1 - y_n)(1 - F(x_n, \lambda))\right) \quad (2)$$

Your program can be written either in R, MATLAB. or Python but *you are not allowed to use any existing implementations of logistic regression or gradient descent*. Code both the logistic regression model and the training algorithm from first principles, as explained in Section 3. Fix the number of attributes, d, so that you can use your algorithm on the following tasks.

2. Load the `Auto` data set as explained in the lab worksheets. Create a new variable `high` that takes values

$$\texttt{high} = \begin{cases} 1 & \text{if } \texttt{mpg} \geq 23 \\ 0 & \text{otherwise (i.e., if } \texttt{mpg} \leq 22). \end{cases}$$

Apply your program to the `Auto` data set and new variable to predict `high` given `horsepower`, `weight`, `year`, and `origin`. (In other words, `high` is

the label and `horsepower`, `weight`, `year`, and `origin` are the attributes.) Since `origin` is a qualitative variable, you will have to create appropriate dummy variables. Normalize the attributes, as described in Week 5's Worksheet and Week 4's Slides (see also Section 4.6.6 of An Introduction to Statistical Learning with Applications in R).

3. Split the data set randomly into two equal parts, which will serve as the training set and the test set. Use your birthday (in the format MMDD) as the seed for the pseudo-random number generator. The same training and test sets should be used throughout this assignment.

4. Train your algorithm on the training set using independent random numbers in the range $[-0.7, 0.7]$ as the initial weights. Find the Error Rate of the trained model on the test set, i.e. compute

$$ER(\hat{f}, \mathcal{D}_{test}) = |\mathcal{D}_{test}|^{-1} \sum_{(x,y)\in\mathcal{D}_{test}} (y - \hat{y})^2, \quad \hat{y} = \arg\max\{1 - \hat{f}, \hat{f}\}$$

   Try different values of the learning rate, $10^{-10} < \eta \leq 10^{-1}$ and of the number of training steps, T, i.e. let your *stopping rule* be to stop after a given number of steps. Give a small table of test and training ER in your report.

5. Try different stopping rules, such as: stop when the value of the objective function does not change by more than 1% of its initial value over the last 10 training steps.

6. Run your logistic regression program for a fixed value of η and for a fixed stopping rule (producing reasonable results in your experiments so far) 100 times, for different values of the initial weights (produced as above, as independent random numbers in $[-0.7, 0.7]$). In each of the 100 cases compute the test ER and show it in your report as a box-plot.

7. Redo the experiments in items 4–5 modifying the training procedure as follows. Train $F(X, \lambda)$ 4 times using gradient descent with different values of the *initial weights* and then choose the prediction rule with the best *training* objective.

Feel free to include in your report anything else that you find interesting.

3 Details of the algorithm

The training set is $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, and we need to estimate the weights $\lambda = [\lambda_0, \ldots, \lambda_d]^T$. The negative log-likelihood evaluated on the training

set is

$$\ell(\lambda, \mathcal{D}_{train}) = -\sum_{n:y_n=1} \log F(x_n, \lambda) - \sum_{n:y_n=0} \log(1 - F(x_n, \lambda)) \tag{3}$$

$$= -\sum_{n=1}^{N} \log\left(y_n F(x_n, \lambda) + (1 - y_n)(1 - F(x_n, \lambda))\right) \tag{4}$$

$$F(x_n, \lambda) = \sigma(\lambda_0 + \sum_{i=1}^{d} x_{ni}\lambda_i) = \sigma\left([1, x_n^T]\lambda\right) = \sigma\left(\tilde{x}_n^T\lambda\right) \tag{5}$$

where $\tilde{x}_n = [1, x_n^T]^T$. The gradient of ℓ at λ is

$$\nabla\ell(\lambda, \mathcal{D}_{train}) = \left[\frac{\partial\ell(\lambda, \mathcal{D}_{train})}{\partial\lambda_0}, \ldots, \frac{\partial\ell(\lambda, \mathcal{D}_{train})}{\partial\lambda_d},\right]^T \tag{6}$$

$$\frac{\partial\ell(\lambda, \mathcal{D}_{train})}{\partial\lambda_i} = -\sum_{n:y_n=1} \frac{\sigma'(\tilde{x}_n^T\lambda)\tilde{x}_{ni}}{F(x_n, \lambda)} + \sum_{n:y_n=0} \frac{\sigma'(\tilde{x}_n^T\lambda)\tilde{x}_{ni}}{1 - F(x_n, \lambda)} \tag{7}$$

$$= -\sum_{n=1}^{N} \frac{y_n - (1 - y_n)}{y_n F(x_n, \lambda) + (1 - y_n)(1 - F(x_n, \lambda))}\sigma'(\tilde{x}_n^T\lambda)\tilde{x}_{ni} \tag{8}$$

where $\tilde{x}_n = [1, x_n^T]^T$ as above and $\sigma'(t) = \frac{d}{dt}\sigma(t) = \sigma(t)(1-\sigma(t))$. Try to obtain this formula using the standard differentiation rules given in the lecture Slides and in Computing Gradients: Rules and Examples.

The gradient descent algorithm is defined as follows.

- Choose a set of random weights $\lambda^{(0)}$, a learning rate, η, and a number of epochs, T_{epochs}
- for $t = 1, \ldots, T_{epochs}$ do the following:
  1. "Forward pass": compute

     $$F(x_n, \lambda^{(t-1)}) = \sigma([1, x_n^T]\lambda^{(t-1)}), \quad n = 1, \ldots, N$$

     and the value of the objective function at $\lambda = \lambda^{(t-1)}$, i.e.

     $$\ell(\lambda^{(t-1)}, \mathcal{D}_{train})$$

  2. "Backward pass": use $F(x_n, \lambda^{(t-1)})$, $n = 1, \ldots, N$, to compute the gradient of ℓ at $\lambda = \lambda^{(t-1)}$, i.e.

     $$\nabla\ell(\lambda^{(t-1)}, \mathcal{D}_{train})$$

  3. Compute the updated weights

     $$\lambda^{(t)} = \lambda^{(t-1)} - \eta\nabla\ell(\lambda^{(t-1)}, \mathcal{D}_{train})$$

     Note: λ is a $(d + 1)$-dimensional vector

- Let your parameter estimate be

$$\hat{\lambda} = \lambda^{(T_{epochs})}$$

and the corresponding *trained model*

$$\hat{f}(X) = F(X, \hat{\lambda}) = \sigma([1, X^T]\lambda^{(T_{epochs})})$$