Assignment 2. Binary Classification with Logistic Regression

Due Date: October 20, 11:30 pm

If you submit the assignment after the deadline, the following penalty is applied:

- ◇ 10% penalty if the submission is before October 27, 11:30 pm (if the mark before applying the penalty is 78 out of 100, after applying the penalty it is 78 - 7.8 = 70.2 out of 100);
- ◇ 50% penalty if the submission is after October 27, 11:30 pm and before Dec. 6, 11:30 pm.

Description:

In this assignment you are required to experiment with logistic regression for binary classification. You will use the “Wisconsin breast cancer” data set, which is available in scikit-learn. The two classes are ”malignant” (the **positive** class) and “benign” (the **negative** class) (if the classes are labeled differently, please change their labels). The number of features is 30.

After you separate the test and training data, you have to standardize the features, in the training data and then apply the same transformations to the features in the test data. Then you have to do the following.

- ◇ Train a linear classifier using your implementation of logistic regression. Train your model using two ways: batch gradient descent (BGD) and stochastic gradient descent (SGD). (The weights obtained in the two cases should be the same or very close.) Then compute the misclassification rate and the F1 score on the test data (using the Bayes classifier). Additionally, you have to plot the “precision/recall” (PR) curve (“recall” is on the horizontal axis and “precision” is on the vertical axis) and the ROC curve (see Topic 4). For this you have to consider all possible thresholds β (see slide 16 in Topic 5) for your classifier based on your test data and compute the PR point and the ROC point for each β. More clarification on how to find all possible values of β are given at the end of this assignment.
- ◇ Do the same as above using the implementation of logistic regression provided in scikit-learn.

Whenever you use randomization in your code, use as seed for the pseudo number generator a number formed with the last 4 digits of your student ID, in any order.

You have to write a report to present your results and their discussion. For the two logistic regression models (yours and the one using scikit learn) you have to specify the vectors of parameters and to include a plot with the two PR curves and a plot with the two ROC curves. For both models specify the weight vector $\mathbf{w}$ and the train and test misclassification rates and F1 scores. For your model, you must also plot learning curves. For each method of training (i.e, BGD and SGD) select at least three values of the learning rate: one for which the learning algorithm does not converge (α_1), one for which the learning is too slow (α_2) and one for which the learning is fast enough (α_3). Then plot the three learning curves in the same figure (you should have one figure for BGD and one for SGD). Include in your report the above values of the learning rates. Your model should be trained with α_3. Specify in the report the number of iterations (for BGD)/epochs (for SGD) to obtain convergence with α_3.

How do the results obtained with your implementation compare with the results obtained using scikit-learn? Which of the two models is the best overall? To decide this, use first the misclassification rate as the metric, next use the F1 score as the metric. Is the winner the same in the two cases? Then compare the two models using the PR curves, then the ROC curves. Is the PR curve of one model better than for the other model? Is the ROC curve of one model better than for the other model?

Besides the report, you have to submit your numpy code. The code has to be modular and use vectorization whenever is possible (for instance, for computing the gradient for BGD training and for computing the errors and the cost). Write a function for each of the main tasks. Also, write a function for each task that is executed multiple times. The code should include instructive comments.

You may use the following code for feature normalization:

```
import numpy as np
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Note that the normalization is performed based on the training data. Then the same transformations are applied to the test data in order to be able to use the trained predictor on the test data.

Submission Instructions:

- Submit the report in pdf format, the Python file containing your code (extension .py), and a short demo video. The video should be 1 min or less. In the video, you should scroll down your code briefly explaining it, show that it runs and that it outputs the results for each part of the assignment. The main Python file in the project should be clearly distinguishable. Some feedback might be written on your report, therefore, please DO NOT ZIP YOUR FILES.
  Naming convention:
  "studentMacId_studentNumber_A2_report", "studentMacId_studentNumber_A2_code".

Notes

◇ **Computation of cost function with numpy.** Recall that the cost function that is minimized in logistic regression is the average cross-entropy loss over the training data, which can be written as follows (slide 21 in Topic 5):

$$\mathcal{C}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} \ell_{lce}(\mathbf{w}\bar{\mathbf{x}}^{(i)}, t^{(i)}),$$

where

$$\ell_{lce}(z,t) = t\log(1+e^{-z}) + (1-t)\log(1+e^{z}).$$

To compute the logarithms in the above formula use the function `numpy.logaddexp`. The result of `numpy.logaddexp(x1,x2)` is `log(exp(x1))+log(exp(x2))`. This is in order to avoid problems when the values $x1$ and $x2$ are too small or too large. Thus, to compute $\ell_{lce}(z,t)$, use `t*numpy.logaddexp(0,-z) + (1-t)*numpy.logaddexp(0,z)`

◇ **Instructions to obtain the thresholds β:**
The classifier that minimizes the misclassification rate is

$$\hat{C}(\mathbf{x}) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases},$$

where $z = \mathbf{w}^T\mathbf{x}$. Other classifiers can be obtained by replacing 0 by some other thresholds β (see slide 16 in Topic 5)

$$\hat{C}(\mathbf{x}) = \begin{cases} 1 & \text{if } z \geq \beta \\ 0 & \text{if } z < \beta \end{cases}.$$

To obtain all possible different classifiers for the test set, it is sufficient to consider as values for β, the values $z[0], z[1], \cdots, z[M-1]$, obtained on the test examples (where M is the size of the test set). For this, first sort the array $\mathbf{z}$. Then take in turn $\beta = z[0]$, $\beta = z[1]$, and so on, up to $\beta = z[M-1]$.