EECE 5698
Assignment 3: Parallel Classification

Preparation. Create a folder on discovery named after your username under the directory `/gss_gpfs_scratch`. Copy the directory

```
/gss_gpfs_scratch/EECE5698/Assignment3
```

to this folder, by typing:

```
cp -r /gss_gpfs_scratch/EECE5698/Assignment3 /gss_gpfs_scratch/$USER/
```

Make the contents of this directory private, by typing:

```
chmod -R o-rx /gss_gpfs_scratch/$USER/Assignment3
```

The directory contains the following python files:

```
LogisticRegression.py
ParallelLogisticRegression.py
SparseVector.py
helpers.py
```

as well as two directories called `mushrooms` and `newsgroups`.

The `mushrooms` dataset[1] contains the features of edible (label:$+1$) and poisonous (label:-1) mushrooms. The `newsgroups` dataset[2] contains the most indicative words in messages posted in two newsgroups: `sci.med` (label:$+1$) and `rec.sports.baseball` (label:-1). In this assignment, all features are *binarized*: the presense of a feature is indicated with a value 1.0, while the absense is indicated with an (implicit) value 0.0 in the corresponding feature vector.

You must:

1. Provide a report, in `pdf` format, outlining the answers of the questions below. The report should be type-written in a word processor of your choice (e.g., MS Word, Latex, etc.).
2. Provide the final code of files `LogisticRegression.py` and `ParallelLogisticRegression.py`.

The report along with your final code should be uploaded to Blackboard. Executions of the code to generate the requested output can be run on "local" mode, on a single compute node which you have reserved on the Discovery cluster. Use, e.g., a node in `ser-par-10g-3` or `ser-par-10g-4`, with 40 logical cores.

[1] https://archive.ics.uci.edu/ml/datasets/Mushroom
[2] http://ana.cachopo.org/datasets-for-single-label-text-categorization

Question 1: File `SparseVector.py` defines a new class called `SparseVector`.

(1a) Describe how this class relates to a standard `python` dictionary.

(1b) Give a few examples of functionalities/methods it shares with a standard dictionary, as well as ones that it has that are not present in a dictionary.

(1c) Launch the python interpreter, create two sparse vectors, add and multiply them with each other and with a scalar. Include the operations that you executed and the outcome of each execution in your report.

(1d) Edit `SparseVector.py` to create a new method for objects of class `SparseVector`, that computes a sparse vector's norm. The method should be called `norm`. It should take an integer $p \geq 1$ as an optional argument, and return the Minkowski p-norm of the sparse vector. If the argument p is omitted, `norm` should return the Euclidean ($p = 2$) norm. Include the code that you wrote in your report, along with an execution over the python interpreter showing a computation of the p-norm of a sparse vector of your choice, for $p = 1, 2$, and 3.

Question 2: Consider dataset $\mathcal{D}$ of n pairs $(x_i, y_i) \in \mathbb{R}^d \times \{-1, +1\}$, $i = 1, \dots n$, where $x_i \in \mathbb{R}^d$ are *features* and $y_i \in \{-1, +1\}$ are *binary labels*. Logistic regression attempts to find a vector $\beta \in \mathbb{R}^d$ such that, given an x_i, the *prediction*:

$$\hat{f}(x_i) = \begin{cases} +1, & \text{if } \beta^\top x_i \geq 0, \\ -1, & \text{if } \beta^\top x_i < 0, \end{cases}$$

agrees with the values y_i in the data set. Regularized logistic regression trains such a β from $\mathcal{D}$ by minimizing the loss function:

$$L(\beta) = \sum_{i=1}^{n} \ell(\beta; x_i, y_i) + \lambda \|\beta\|_2^2$$

over $\beta \in \mathbb{R}^d$,where

$$\ell(\beta; x_i, y_i) = \log\left(1 + e^{-y\beta^\top x}\right)$$

is known as the *logistic loss*.

2(a) Prove that, for all $x \in \mathbb{R}^d$ and all $y \in \{-1, +1\}$, the logistic loss $\ell(\beta; x, y)$ is a convex function of $\beta \in \mathbb{R}^d$.

2(b) Prove that

$$\nabla \ell(\beta; x, y) = -\frac{yx}{1.0 + e^{y\beta^\top x}}.$$

2(c) Suppose that $x_j = 0$ for some coordinate $j \in \{1, \dots, d\}$ of vector $x \in \mathbb{R}^d$. Prove that $\ell(\beta; x, y)$ and $\nabla \ell(\beta; x, y)$ do not depend on β_j, the j-th coordinate of $\beta \in \mathbb{R}^d$.

2(d) Prove that $L(\beta)$ is a convex function of $\beta \in \mathbb{R}^d$.

2(e) Suppose that there exists a vector $x' \in \mathbb{R}^d$ such that (i) $x' \neq \mathbf{0}$, and (ii) $x_i^\top x' = 0$ for all $i = 1, \dots, n$. Show that, when $\lambda = 0$, L is not a strictly convex function of $\beta \in \mathbb{R}^d$.

2(f) Prove that, if $\lambda > 0$, $L(\beta)$ is a strongly convex function of $\beta \in \mathbb{R}^d$.

Question 3: **(3a)** Implement the functions:

```
logisticLoss
gradLogisticLoss
gradTotalLoss
```

in program `LogisticRegression.py`, so that they act as specified in their docstring comments. **Do not use spark in this implementation**. You may use `estimateGrad` from file `helpers.py` to test the correctness of your gradient code. Include the three function definitions you wrote for these three functions in your report.

3(b) Implement the function `test` in `LogisticRegression.py`, so that it computes the accuracy, precision, and recall of β over a dataset. Include the final function definition in your report.

3(c) Run the code `LogisticRegression.py` on the `mushrooms` train and test datasets with three different values of λ, for a maximum of 20 iterations. Include the output of your execution in your report.

3(d) Recall that label $+1$ indicates edible mushrooms and label -1 indicates poisonous mushrooms. If you were to choose between (a) high precision and low recall, vs. (b) high recall and low precision, which would you prefer?

Question 4: Complete the missing code in program `ParallelLogisticRegression.py` **in Spark** so that it replicates exactly the same functionality as `LogisticRegression.py` (messages printed can differ in style/-format, but should contain the same information). You may import and reuse any of the following functions from `LogisticRegression.py`:

```
readBeta, writeBeta, gradLogisticLoss, logisticLoss, lineSearch
```

but you **may not** import any other function implemented in `LogisticRegression.py`. In particular, you should not include any for loop or list comprehension that iterates over the data in `dataRDD`.

(a) Include the code of every function you wrote in your report. All functions should contain docstring comments describing the the function in a single succinct sentence, followed by a detailed description of what the function computes, what are its inputs and outputs, etc.

(b) Compare the output of your program to the output of `LogisticRegression.py` when executed with $\lambda = 0$ over the `mushrooms` train and test datasets for at most 20 iterations. Include the output in your report. Which one is faster?

(c) Compare the output of your program to the output of `LogisticRegression.py` when executed with $\lambda = 0$ over the `newsgroups` train and test datasets for at most 20 iterations. Include the output in your report. Which one is faster?

Question 5: In what follows, run your code over the `newsgroups` train and test datasets. You need not run this until convegence: report how many iterations you used.

(a) Produce a plot for $\lambda = 0$ with time on the x-axis and the norm of the gradient $\|\nabla L\|_2$ on the y-axis, showing two curves: one for `LogisticRegression.py` and one for `ParallelLogisticRegression.py`, when executed over `newsgroups`. Produce 3 similar plots showing accuracy, precision, and recall on the test set, respectively, on the y-axis, for $\lambda = 0$.

(b) Produce a single plot showing accuracy on the test set as a function of time, for three different values of λ.

(c) For the value of λ and the number of iterations that gives you the highest accuracy, store the resulting β. Either in a table or a bar plot, report the 10 features of β with the most positive values. Similarly, report the 10 features of β with the most negative values.