Assignment 1 - Foundations of Machine Learning CSCI3151 - Dalhousie University

Q1 (30%)
Gradient descent - Linear regression

In this question we are going to learn first hand how gradient descent works in the context of a concrete house pricing model. Note that we are not making use of a validation strategy here by using a test set or cross-fold validation—which is something that would be otherwise generally recommended. This exercise focuses on the inner workings of gradient descent applied to the quadratic cost function that was learned in the Linear regression class. It's important that the code provided is well documented and that no other external package is used (pandas, numpy are ok).

a) Using the house price prediction dataset (described here and downloadable from Brightspace as housePriceKaggleTrain) and two features, i.e. *1stFlrSF* and *2ndFlrSF*, we want to fit a linear regression model to predict *SalePrice*. We want to fit this model by applying a *hand made* gradient descent algorithm. We will use the basic cost function learned in the lecture.

The model should be as follows:
p: sale price (SalePrice)
t: first floor square feet (1stFlrSF)
s: second floor square feet (2ndFlrSF)
$p = t\,x_1 + s\,x_2 + b$

Your function should be able to return the updated weights x_1, x_2 and b after every iteration of the gradient descent algorithm.

Your function should be defined as follows:
```
def LRGradDesc(data, target, init_x1, init_x2, init_b, learning_rate_w, learning_rate_b, max_iter):
```

Note that there is a slight variation with the approach proposed in the lecture as here we use two different learning rates (one learning rate for the weight coefficients and another one for the bias).

And it should print lines as indicated below:
```
Iteration 0: init_x1, init_x2, init_b, initial_cost
Iteration 1: [x1 after first iteration], [x2 after first iteration], [b after first iteration], [cost after first iteration]
Iteration 2: [x1 after second iteration], [x2 after second iteration], [b after second iteration], [cost after second iteration]
…
Iteration max_iter: [x1 after max_iter iteration], [x2 after max_iter iteration], [b after max_iter iteration], [cost after max_iter iteration]
```

Note that you may want to print every 100 or every 1000 iterations if *max_iter* is a fairly large number (but you shouldn't have more iterations than the indicated in *max_iter*).

b) Compare this model with a solution computed in closed form or by using a machine learning library to compute the linear regression model.
c) Discuss how the choice of learning_rate affects the fitting of the model.

HINT: The learning rates for the weight coefficients should be about 1000 to 100000 times smaller than the one for the bias. In part c) you are encouraged to play with different learning rates.

Q2 (20%)
Classification using two standard techniques
In this question you will experiment with two traditional classification models on the pima indians data set. The models are: decision trees, and SVM. The objective of this question is to demonstrate your ability to compare different machine learning models, and derive a conclusion if possible.

a) For the SVM classifier, use the default parameters, and 5-fold cross validation, and report the overall accuracy and confusion matrix, as well as accuracy and confusion matrix for each fold. What is the standard deviation of accuracy over the folds?
b) for the Decision tree classifier, experiment with different numbers of max depths of the tree (in the range from 2 to 8). Use the default values for the remaining parameters. Evaluate each parameter selection using 5-fold cross validation, and report the overall accuracy and the confusion matrix. Only report the interesting parameter settings.
c) Summarize your findings from (a) and (b). What can you conclude from these two methods?

Q3 (20%)
Polynomial regression
This question aims at applying polynomial regression on a large data set and deciding the set of hyperparameters that best applies to this scenario.

a) Use the California Housing dataset from scikit-learn and apply a polynomial regression with the full set of features (no interactions). Experiment with polynomials of different degrees. Compare their performance with each other using a fixed train and validation partition (top 80% for training and the remaining 20% for validation). Discuss the interpretation of the results.
Use visualizations of appropriate quantities to make sense of the results and support your interpretation.

Q4 (30%)
k-Nearest Neighbors
In this question you will develop a kNN algorithm to be used for the automatic recognition of handwritten digits. You are allowed to use libraries for data manipulation and math operations (e.g. numpy, pandas) but no machine learning library to compute k-nearest neighbors (e.g. scikit-learn is not ok). If in doubt, please ask.

a) Write a k-NN algorithm that receives as input labeled training data and a set of unlabeled instances. The algorithm returns the most likely label (i.e. a digit) to each unlabeled instance. The algorithm should implement the Minkowski distance. The value for *k* (number of neighbors) and the parameter *p* are hyperparameters of the method that will be also passed as input.

Your function should be defined as follows:

```
def kNN(train_x, train_y, test_x, k, p):
```

The code should return a list or array of predictions for all the instances in *test_x*.

You will test your code using a reduced version of the MNIST database. The original MNIST dataset is a database of 60,000 handwritten digits for training and 10,000 for testing where each digit is represented by a 784-dimensional vector. The reduced version of this dataset can be found on Brightspace as reducedMNIST.zip.

b) Report the classification error and confusion matrix you obtain for different combinations of *p* and *k*. Remember that you must only use the “test_labels.csv” file to compare your predictions to. In addition, critically discuss your results, as you vary *p* and *k*.
c) What strategy would you follow to choose the parameters (k and p) for your final model, if you did not have the labels for the test set?

Submitting the assignment
1. Your assignment as a single .ipynb file including your answers should be submitted before the deadline on Brightspace.
Use markdown syntax to format your answers and to clearly indicate what question.
2. You can submit multiple editions of your assignment. Only the last one will be marked. It is recommended to upload a complete submission, even if you are still improving it, so that you have something into the system if your computer fails for whatever reason.
3. IMPORTANT: PLEASE NAME YOUR PYTHON NOTEBOOK FILE AS:
<LAST_NAME>-<FIRST_NAME>-Assignment-N.ipynb, for example
Soto-Axel-Assignment-1.ipynb
A penalty applies if the format is not correct.
4. The markers will enter your marks and their overall feedback on Brightspace, and they will upload your Python notebook file with comments on specific cells, as a new markdown cell below the cell being commented on.

Marking the assignment

Criteria and weights. Each criterion is marked by a letter grade. Overall mark is the weighted average of the grade of each criterion.

0.2 Clarity: All steps are clearly described. The origin of all code used is clearly. Markdown is used effectively to format the answer to make it easier to read and grasp the main points. Links have been added to all online resources used (markdown syntax is: [AnchorText](URL)).
0.2 Justification: Parameter choices or processes are well justified.
0.2 Results: The results are complete. The results are presented in a manner that is easy to understand. The answer is selective in the amount and diversity of the experimental results presented.
Only key results that support the insights are presented. There is no need to present every single experiment you carried out. Only the interesting results are presented, where the behaviour of the ML model varies.
0.4 Insights: The insights obtained from the experimental results are clearly explained. The insights are connected with the concepts discussed in the lectures.
The insights can also include statistical considerations (separate training-test data, cross-validation, variance).Preliminary investigation of the statistical properties of the attributes (e.g. histogram, mean, standard deviation) is included.