CSE474/574 Introduction to Machine Learning
Programming Assignment 2

Non-linear Models for Supervised Learning

Due Date: **April 13th 2020**
Maximum Score: 100

Note A zipped file containing skeleton Jupyter notebooks (`PA2-Part1.ipynb` and `PA2-Part2.ipynb`) and data is provided. Note that for each part, you need to write code in the specified function within the corresponding notebook file.

Evaluation For this assignment, we will evaluate your report. At the same time, we will also execute your submitted code to make sure that the findings documented in your report are observed in your code as well.

Submission You are required to submit a single file called ***pa2.zip*** using UBLearns. One submission per group is required. File ***pa2.zip*** must contain 3 files: ***report.pdf***, ***PA2-Part1.ipynb*** and ***PA2-Part2.ipynb***.

- Submit your report in a pdf format. Please indicate the **team members**, **group number**, and your **course number** on the top of the report.
- The notebooks should contain all implemented functions. Please do not change the names of the files.

Please make sure that your group is enrolled in the UBLearns system: You should submit one solution per group through the groups page. *If you want to change the group, contact the instructors.*

Part I - Sentiment Analysis (Total 65 Points)

In this part, you will classify movie reviews as positive or negative using the following two approaches and compare their accuracy.

Approach 1 - Word Count Vectorization

Task 1: Implement a simple hand-crafted classifier that can label a movie review as positive or negative.

- This part of the assignment is designed to help with the following:
  - Understanding the data and thus identifying the important features.
  - Building a basic intuition of what model should be employed to solve a problem.
- Steps (*Some of these are already implemented*):

1. To get started with the exercise, you will need to download the supporting files and unzip its contents to the directory you want to complete this assignment in. Navigate to the root directory for the sentiment analysis dataset. Two files are provided: `reviews.txt` and `labels.txt`. Each line in `reviews.txt` correspond to a textual review for a movie. The corresponding line in the label files will have the label - `NEGATIVE` and `POSITIVE`, which is a binary sentiment inferred by reading the review. The files contain reviews and labels for 25000 different movies.
2. Iterating through each review, obtain the count of each word, for positive and negative labels separately. These words can be referred to as the *vocabulary*[1]. You will note that in the code, we are ignoring words that occur fewer than 100 times in the entire training data set (training corpus).
3. Calculate a feature for each word, based on the word counts for the positive and negative reviews, that can indicate the sentiment of the review. See the notebook for more details. [**10 Points**]
   - *Hint*: You can make use of the positive-to-negative ratio of a word from either category. Thus, a neutral word would have a ratio of around 1, as it appears just as often in both the categories.
   - Convert the ratios into log values so that positive values are close to 1, negative values are close to -1 and neutral values are around 0.
4. Use the word feature, defined above, to develop a simple "classifier", that does not use a machine learning model. Instead, it should be a rule-based system that can assign a positive or negative label to each test review. Implement this logic in the `nonml_classifier` function in the notebook. [**15 Points**]

REPORT 1.

Describe your word features and the classification model and report the accuracy of your model on the test data. [**10 Points**]

Approach 2 - Neural Network Based Sentiment Classification

Using the previous approach, you are able to assign a sentiment to a review based on the positive or negative value of the words in a review. This is not a very sophisticated method to solve this particular problem but helps to build an intuition towards a more complex model we would use in Approach 2.

In the notebook, we have given you a code that allows you to create a single hidden layer, multi-layer perceptron (MLP), that is evaluated on the test review data.

REPORT 2.

Report the accuracy and running time of the vanilla neural network. Is it better or worse than your rule-based algorithm implemented in Approach 1? [**5 Points**]

Improving the Neural Network Performance

How can we make the neural network more accurate? In this part, we want you to explore different strategies. For each strategy you will have to report the final accuracy on the test data set and the running time.

- Making the model more complex: try increasing the "width" (number of units) in the hidden layer. Similarily, you can add more hidden layers to the architecture or increase the number of training epochs. [**10 Points**]
- Modifying inputs to the neural network. In the first approach, you had developed word-level features. Could those be used to prune away some neutral words? Will that have an impact on the performance, in terms of accuracy and/or time? Implement the `find_ignore_words` function that identifies a set of words that can be ignored from the vocabulary. Re-run your code, including generation of the data file, and study the performance of the system. [**15 Points**]

[1] Hint: You can make use of the Counter function from collections library in python3.

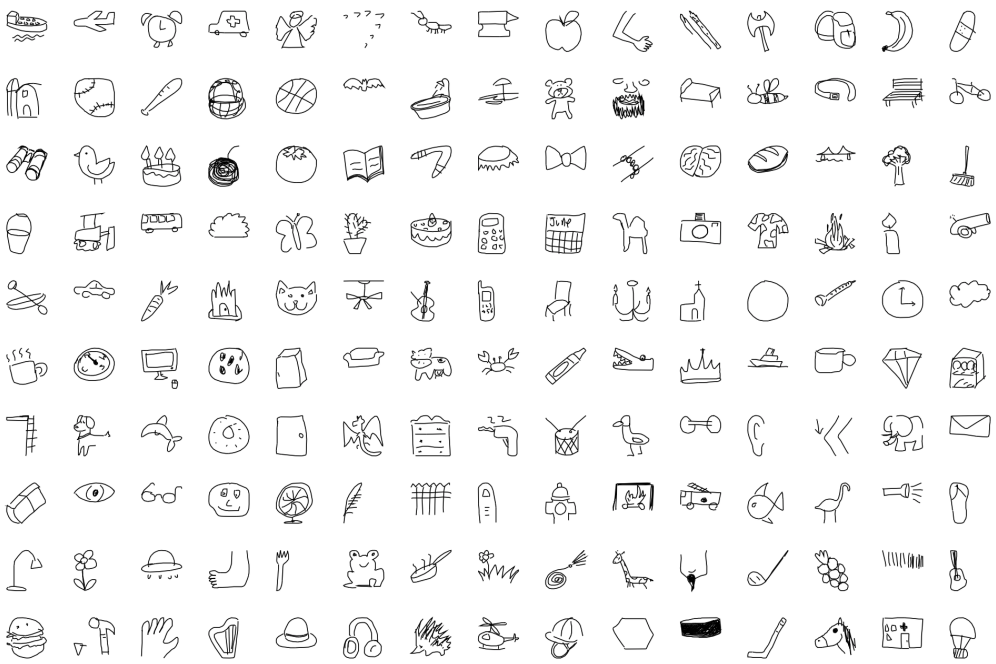

Figure 1: Sample sketches from the Quick Draw data set (Src: https://quickdraw.withgoogle.com/data)

REPORT 3.
Report the performance of the model (both in terms of accuracy and time) using different values for the hidden layer width, number of hidden layers, number of epochs and a set of ignored words. What has the most impact on the performance?

Part II - Image Classification on the AI Quick Draw Dataset (Total 35 Points)

The AI quick draw data set used for this part is created from the original Quick Draw dataset[2]. The original Quick Draw Dataset is a collection of 50 million drawings across 345 categories, contributed by players of the game Quick, Draw! (See Figure 1).

The drawings were captured as timestamped vectors, tagged with metadata including what the player was asked to draw and in which country the player was located. However, for this assignment, we have provided you with a subsampled Quick Draw data set with the following properties:

- 10 categories including 'apple', 'basketball', 'arm', 'horse', 'ant', 'axe', 'alarm clock', 'airplane', 'banana' and 'bed'.
- Each category includes 12,500 images with the size of 28×28. Therefore, in the data set, each sample is a 784 length vector.
- The data set has already been separated as training features, training labels, testing features andtesting labels. In the code, we iteratively use the load function provided by the pickle package to extract those data from the data set.
- *Note*: Since the data file is large, you will need to download it from the following link : `https://www.cse.buffalo.edu/ubds/docs/AI_quick_draw.pickle`. This file is 94 MB in size. If, due to versioning

[2]https://quickdraw.withgoogle.com/data

issues, the pickle file is not readable on your system, you may use the provided `create_data_set.py` to recreate the pickle file.

Tasks

You will first study the performance of a multi-layered perceptron on the provided data as you change the complexity of the model by changing the number of hidden layers. You will then study the performance of the model when the fidelity (resolution) of the input image changes. Use the provided `resize_images()` function to reduce the resolution of the image and then evaluate the performance (both accuracy and time) of the model on the test data set.

REPORT 4.

1. Run the evaluation of the 2 hidden layer neural network in the notebook - `PA2-Part2.ipynb` and report the test accuracy and the run time. [**5 Points**]
2. Compare the performance when the number of hidden layers are increased to 3 and 5. [**15 Points**]
3. Use the `resize_images()` function to reduce the resolution of the images to (20×20), (15×15), (10×10) and (5×5). Using the 2 hidden layer architecture, compare the performance at different resolutions, including the original (28×28) resolution, both in terms of test accuracy and time. [**15 Points**]