CS534 — Implementation Assignment 3 — Due 11:59PM Nov 7th, 2020

General instructions.

1. Please use Python 3 (preferably version 3.6+). You may use packages: Numpy, Pandas, and matplotlib, along with any from the standard library (such as 'math', 'os', or 'random' - for example).

2. You should complete this assignment alone. Please do not share code with other students, or copy program files/structure from any outside sources like Github. Your work should be your own.

3. Your source code and report will be submitted through Canvas.

4. You need to follow the submission instructions for file organization (located at the end of the report).

5. Please run your code before submission on one of the OSU EECS servers (i.e. babylon01.eecs.oregonstate.edu). You can make your own virtual environment with the packages we've listed in either your user directory or on the scratch directory. If you're unfamiliar with any of this process, or have limited access, please contact one of the TA's.

6. Be sure to answer all the questions in your report. You will be graded based on your code as well as the report. In particular, **the clarity and quality of the report will be worth 10 pts**. So please write your report in clear and concise manner. Clearly label your figures, legends, and tables. It should be a PDF document.

7. In your report, the **results should always be accompanied by discussions** of the results. Do the results follow your expectation? Any surprises? What kind of explanation can you provide?

Perceptron and Kernels
(total points: 90 pts + 10 report pts)

Data. This dataset consists of health insurance customer demographics, as well as collected information related to the customers' driving situation. Your goal is to use this data to predict whether or not a customer may be interested in purchasing vehicular insurance as well (this is your "Response" variable). The dataset description (dictionary) is included. **Do not use existing code from outside sources for any portions of this assignment. This would be a violation of the academic integrity policy.**

The data is provided to you in both a training set: **pa2_train.csv**, and a validation set: **pa2_dev.csv**, with an X and y for both (X being features, y being labels). You have labels for both sets of data. The only preprocessing you will need to do is the normalization of the numeric/ordinal features (see next paragraph).

Preprocessing Information In order to train on this data, we have pre-processed it into an appropriate format. This is done for you in this assignment to ensure results are similar across submissions (easier to grade). You should be familiar with this process already from the last assignment. In particular, we have treated [**Gender, Driving_License, Region_Code, Previously_Insured, Vehicle_Age, Vehicle_Damage, Policy_Sales_Channel**] as categorical features. We have converted those with multiple categories (some that originally contained textual descriptions) into one-hot vectors. Note that we left **Age** as an ordinal numeric feature. You are to leave these as is and not modify further for this assignment, but understand the process. **The numeric and ordinal features [Age, Annual_Premium, Vintage] have already been scaled to the range of [0, 1].** The data also includes an intercept term in the last column. Additionally, the dataset should be relatively class balanced (close to the same number of +1's and -1's for Response). This was not the case in the raw data, so we downsampled for easier training purposes. There are other ways to handle class imbalance, beyond the scope of this assignment, but it is a common problem in real-world data.

General comments about training. For all parts, we have specified $maxiter = 100$ as the upper limit on the number of training iterations. This is probably not sufficient for convergence but the online algorithms are slower due to the fact that you can not use matrix operations to process all examples at once. If you find that your algorithm needs more than 100 iterations to converge to a good solution, feel free to use higher values if you have time. But if running time is a concern, stick with the 100 limit. You may wonder why should one use online algorithms if the run time is so bad. There are several situations for which online algorithms are practically useful. Sometimes we receive data instances one at a time and does not allow storing. Online learning would be appropriate for such setting. In other applications, e.g., structured prediction problems in Natural language processing, the feature set is ginormous and super sparse, it makes sense to process one example at a time.

1 Part 1 (35 pts) : Average Perceptron.

For this part you will implement and experiment with average perceptron, which is described in Algorithm **??**.

Algorithm 1: Average Perceptron

Input: $\{(\mathbf{x}_i, y_i)_{i=1}^N\}$(training data), *maxiter* (maximum of iterations)
Output: online perceptron $\mathbf{w}$, average perceptron $\bar{\mathbf{w}}$
Initialize $\mathbf{w} \leftarrow 0$;
Initialize $\bar{\mathbf{w}} \leftarrow 0$;
Initialize example counter $s \leftarrow 1$;
while *iter* $<$ *maxiter* **do**
  for *each training example* $\mathbf{x}_i$ **do**
    if $y_i \mathbf{w}^T \mathbf{x}_i \leq 0$ **then**
      $\mathbf{w} \leftarrow \mathbf{w} + y_i \mathbf{x}_i$
    end
    $\bar{\mathbf{w}} \leftarrow \frac{s\bar{\mathbf{w}}+\mathbf{w}}{s+1}$;
    $s \leftarrow s + 1$;
  end
end

Algorithm 1 returns two solutions, $\mathbf{w}$ is the solution of the vanilla online perceptron and $\bar{\mathbf{w}}$ is the solution of average perceptron. Note that the running average $\bar{\mathbf{w}}$ always gets updated every time an example is processed, regardless whether the current $\mathbf{w}$ correctly classifies it or not. If you are interested, there is a slightly different but equivalent version of this algorithm that only updates $\bar{\mathbf{w}}$ when $\mathbf{w}$ is updated, presented in Chapter 3 of A course in machine learning (Algorithm 7). This can lead to faster running time. You can choose to implement either version.

Perform the following experiments:

(a) Apply your implemented algorithm to learn from the training data with *maxiter* $= 100$. Plot the train and validation accuracy of $\mathbf{w}$ (online perceptron) and $\bar{\mathbf{w}}$ (average perceptron) at the end of each training iteration.

(b) What are your observations when comparing the training accuracy and validation accuracy curves of the average perceptron with those of the online perceptron? What are your explanation for the observations?

(c) Focusing on average perceptron, use the validation accuracy to decide the best number of iterations to stop.

2 Part 2 (55 pts). Perceptron with Polynomial Kernel.

The online/average perceptron in Algorithm 1 are linear models. In this part we will consider kernelized perceptron, as described in Algorithm 2.

Algorithm 2: Kernelized Perceptron

Input: $\{(\mathbf{x}_i, y_i)_{i=1}^N\}$(training data), *maxiter* (maximum iterations), κ (kernel function)
Output: $\alpha_1, \ldots, \alpha_N$
Initialize $\alpha_i \leftarrow 0$ for $i = 1, ..., N$;
for $i = 1, ..., N$, $j = 1, ..., N$ **do**
  $K(i, j) = \kappa(\mathbf{x}_i, \mathbf{x}_j)$; // Compute the Gram Matrix
end
while *iter* $<$ *maxiter* **do**
  for *each training example* $\mathbf{x}_i$ **do**
    $u \leftarrow \sum_j \alpha_j K(i, j) y_j$; // make prediction for $\mathbf{x}_i$
    if $uy_i \leq 0$ **then**
      $\alpha_i \leftarrow \alpha_i + 1$; // Mistake on example i
    end
  end
end

In this assignment, we will implement the kernelized perceptron with polynomial kernel with degree p:

$$\kappa(x_1, x_2) = (1 + x_1^T x_2)^p \tag{1}$$

Part 2a. (40 pts) With your implementation of Algorithm 2, perform the following experiments:

(a) Apply the kernelized perceptron with different p values in [1, 2, 3, 4, 5] with $maxiter = 100$. Note that $p = 1$ will return to the vanilla online perceptron, so you should expect similar behavior compared to part 1.

(b) For each p value, at the end of each training iteration use the current model (aka the current set of α's) to make prediction for both the training and validation set. Record and plot the train and validation accuracy as a function of training iterations.

(c) Record the best validation accuracy achieved for each p value over all iterations. Plot the recorded best validation accuracy versus p. How do you think p is affecting the train and validation accuracy and what is your explanation for the observation?

(d) What is the asymptotic runtime of your algorithm in terms of the number of training examples n? Try to make your implementation as efficient as possible. For the p value chosen above, plot the empirical runtime of your algorithm as a function of the iterations.

Part 2b. (15 pts) Algorithm 2 implements the online kernelized perceptron. Now please modify it to implement batch kernelized perceptron with tunable learning rate and perform the following experiments:

(a) Apply your batch kernel perceptron with the best p value selected in part 2a. You should tune your learning rate as well as the number of iterations to maximize the validation accuracy. Report the configuration of learning rate and iteration number with the best validation accuracy you achieve.

(b) Record and plot the training accuracy and validation accuracy as a function of the iterations. Comparing these curves with the ones acquired with the same p value in part 2a, what do you observe? What are your explanations for the observation?

(c) What is the asymptotic runtime of your algorithm in terms of the number of training examples n? Try to make your implementation as efficient as possible. Plot the empirical runtime of your algorithm as a function of the iterations. How does it compare to the runtime of the online algorithm?

Submission. Your submission should include the following:
1) Your source code. **One file for each Part**. The files should be named (for example) **part1.py**, and should run with simply **python part1.py**. For part 2, please include a separate file for part a and b respectively. You do not need to generate plots in the submission code, please just include those in your report;
2) Your report (see general instruction items 6 and 7 on page 1 of the assignment), which should begin with a general introduction section, followed by one section for each part of the assignment;
3) Please submit the report PDF, along with a .zip containing the code to Canvas. The PDF should be outside the .zip so it's easier to view the report.