




HW 2

*Handed Out: October 25, 2021**Due: November 10, 2021*Instructions for submission

1. Conceptual Part: **Submit a single PDF on Gradescope with all your answers. You should have been added, but if not the Entry Code is N8NWDK. Make sure you select the page corresponding to the beginning of each answer, else points might be deducted.** Your homework must be typed and must contain your name and Purdue ID. If you are using a late day, please indicate it at the top of the document. Do not send the instructors emails about Late Days.
2. Coding part: To submit your assignment, log into `data.cs.purdue.edu` (physically go to the lab or use ssh remotely) and follow these steps:
 - (a) Place all of your code (`Execution.py`, `Perceptron.py`, `Logistic.py`, and any other files you used) in a folder titled `cs578-hw2`. **Make sure your folder is called this!**
 - (b) Change directory to outside of `cs578-hw2/` folder (run `cd ..` from inside `cs578-hw2/` folder)
 - (c) Execute the following command to turnin your code: `turnin -c cs578 -p hw2Fall2021 cs578-hw2`
 - (d) To overwrite an old submission, simply execute this command again.
 - (e) To verify the contents of your submission, execute this command: `turnin -v -c cs578 -p hw2Fall2021`.
Do not forget the `-v` option, else your submission will be overwritten with an empty submission.

Questions

1. Given n boolean variables (x_1, \dots, x_n) , we define our target classification function $f(\cdot)$ as $f(\cdot) = 1$ if at *least 3 variables are active*. For $n=5$ show how this function can be represented as (1) Boolean function and a (2) Linear function. 
2. Let CON_B be the set of all different monotone conjunctions defined over n boolean variables. What is the size of CON_B ? 
3. Let CON_L be the set of all linear functions defined over n boolean variables that are consistent with the functions in CON_B . We say that two functions $f_b \in CON_B$ and $f_l \in CON_L$ are consistent if $\forall x; f_b(x) = f_l(x)$. What is the size of CON_L ?

4. Define in one sentence: Mistake bound (your answer should include all the components described in class).
5. Suggest a mistake bound algorithm for learning Boolean conjunctions (*hint: recall the elimination algorithm for monotone conjunctions*). Show that your algorithm is a mistake bound algorithm for Boolean conjunctions.
6. Given a linearly separable dataset consisting of 1000 positive examples and 1000 negative examples, we train two linear classifiers using the perceptron algorithm. We provide the first classifier with a sorted dataset in which all the positive examples appear first, and then the negative examples appear. The second classifier is trained by randomly selecting examples at each training iteration. (1) Will both classifiers converge? (2) what will be the training error of each one of the classifiers? 
7. We showed in class that using the Boolean kernel function $K(x, y) = 2^{\text{same}(x, y)}$ we can express all conjunctions over the attributes of x and y (with both the positive and negative literals). This representation can be too expressive. Suggest a kernel function $K(x, y)$ that can express all conjunctions of size at most k (where k is a parameter, $k < n$).

Programming Assignment

In this part of the assignment you are required to implement the perceptron and logistic function and observe their performance on a binary sentiment classification task.

1. **Dataset/Task** The task will be a binary sentiment classification task where your models will take as input a document of text and then produce a binary output saying it's either positive (1) or negative (0). The dataset is essentially a table of two rows where the first row represents some text and the second row corresponds to its sentiment. Ideally, you want your trained pmodel to accurately differentiate between positive and negative sentiment by looking at words that are used in the text.

You will be given a single csv file that contains all the data corresponding to this task called data.csv. One component of this task is to split the data into a train and test data sets. The splits are determined by you. It is required that you use 5-fold cross validation when training your models. You should not use the **test set** in picking the optimal hyperparameter values. Simply report the performance of your models with most optimal hyperparameter values on the test set after observing the performance on the training datasets.

To program the splitting of the dataset, take a look at the function `split_dataset` in `Execution.py`. You will have to program how the original dataset will be split into train and test datasets. It is imperative that you maintain the input and outputs of the function as described in the program file. Additionally, each one of the dataset outputs must have column called 'Label' which has either a 0 or 1 to label negative or positive sentiment respectfully. For example, look at Fig. 1 below to see a sample format. It's important to keep this format because when your programs will be graded,

	Text	Label
0	spoiled rich kid kelley morse (chris klein) ...	0
1	the bond series is an island in the film world...	1
2	tarzan chad'z = good) 1999 , g , 90 minutes [...	1
3	a frequent error is the categorization of a te...	1
4	part one of " the strangest movies ever made "...	1

Figure 1: Sample pandas dataframe that contains the column ‘Label’ which represents the sentiment for each row in column ‘Text’.

the datasets will be passed in with this format and the ground-truth labels for every dataset will be extracted by simply calling the column ‘Label’ (look at Execution.py for more information).

2. **Feature Selection** An important task within this homework is to find an effective way to represent text into a meaningful quantitative representation to be used by the perceptron. One example way is to use a representation called **bag-of-words**. This representation simply represents a piece of text by the frequency of the words that appear. For example, take the two sentences below:

- John likes to play basketball.
- Mary prefers to play soccer.

Now, we build a vocabulary of words. So, our vocabulary would be the following set: $\{John, Mary, likes, prefer, to, play, soccer, basketball\}$. Since we have 8 distinct words in our vocabulary, we can use 8-dimensional vectors where each dimension corresponds to a word and the value in each dimension represents the frequency. For example,

- (1, 0, 1, 0, 1, 1, 0, 1)
- (0, 1, 0, 1, 1, 1, 1, 0).

If we take the first vector, we see that there is a 1 in the 1st, 3rd, 5th, 6th, and 8th dimensions which corresponds to 1 word of each John, likes, to, play, and basketball. This is one such representation to translate text into a vector representation.

While bag-of-words is a pretty simple approach, there are several drawbacks such that it doesn’t preserve sentence structure and syntax that could be useful as features in a model. In fact, if the features extracted from the text are not meaningful, then the performance of the machine learning model can suffer greatly. Therefore, it is highly important to construct and preserve relevant features that will be used as input to the perceptron model. For example, another widely used technique is called **n-gram** modeling which you can learn more from reading section 3.1 in the following pdf: <https://web.stanford.edu/~jurafsky/slp3/3.pdf>.

You are encouraged to explore and try out several different feature engineering processes and find the one that maximizes the performance of your model. It is highly encouraged if you want to pursue other feature processing techniques that are not listed as examples here.

3. **Implementation** In this assignment, you will be implementing two different models and comparing their performance on the sentiment classification task. The program `Perceptron.py` is where you will be implementing the perceptron algorithm. The program `Logistic.py` is where you will be implementing and training the logistic function. Both `Perceptron.py` and `Logistic.py` are setup to do the following things:

- Train the given model on the training data.
- Predict labels on some test set.

You will have to train and implement both models separately using the programs listed above. Program the perceptron and its training algorithm on the binary sentiment classification task. The perceptron contains two hyperparameters that you must experiment with: the learning rate and maximum number of iterations. The learning rate controls how large each update affects the weight vector, and maximum iterations controls how many examples the training algorithm cycles through. For the logistic function, you will train it via stochastic gradient descent using logistic loss (same as binary cross-entropy). You will have two hyperparameters when training the logistic function: the learning rate and max number of epochs. The latter hyperparameter controls how many epochs stochastic gradient descent should run for before terminating.

You have the freedom to modify most of the starter code and implement it how you feel is optimal. However, there are certain requirements, which are essentially the same for both programs, that you must follow that are detailed explicitly in the comments of the starter code. As a high level overview, you may not change the inputs to the `__init__` function and the input parameters and return statements of either `train()` or `predict()` may not change. You can customize what code to place inside these functions but these requirements must hold. The reason for this is to streamline the process of evaluating your programs for test time on datasets that your model hasn't seen before.

It is highly important that the features you use are consistent for both the perceptron and the logistic function to yield a fair comparison.

The starter code and the comments contain explicit instructions on the requirements and what pieces of the program you can modify.

4. **Evaluation** There is another program called `Execution.py` that is given to you to evaluate the performance of your model on the datasets. This program creates both models, trains them on the train data set, produces predicted labels, and evaluates their accuracy compared to the ground-truth.

In this program, there is one function which you must program which is `split_dataset`. The function `split_dataset` takes in the full csv dataset file and you must specify how to split it into train and test sets.

The primary way you will be evaluating the performance of both your models will be through classification accuracy which is already implemented in `Execution.py`. When you run `Execution.py`, it will produce these accuracies on the train and test sets. Please look at the code for further information.

You can simply run `Execution.py` from the command line in the following way:

```
python3 Execution.py
```

It is important to note that the `python` command assumes that a `python3` interpreter is being used. Additionally, it is important that `Execution.py`, `Perceptron.py`, `Logistic.py`, and the `.csv` files stay in the same folder because `Execution.py` will load the data files assuming it's in the same directory.

It is important to not alter the code for specific sections in this program because these exact functions will be used to evaluate your code and model. More detailed information is contained in the comments of the starter code.

5. **Report** You will need to submit a report detailing the results of your algorithm. Please have the following sections labeled in the order specified. Below are the following requirements for the report:
 - (a) Explain how you split the dataset and why you selected those splits for the training and test sets.
 - (b) Explain your feature engineering processes, the different strategies you used, and which method you ultimately used. Also, give an explanation and a simple example of how your strategy worked in processing the raw dataset. Explain how you handled unseen features.
 - (c) It's important to show how you selected these hyperparameter values. For each model, plot the values of learning rate (x-axis) vs. accuracy (y-axis) for a fixed number of max iterations (or number of max epochs for the logistic function). Since you will be conducting 5-fold cross validation, average the accuracy across the 5 folds and plot that across different learning rates you have picked. You should have multiple of such plots for different number of max iterations (or number of max epochs for the logistic function).
 - (d) Report the optimal values for the hyperparameters and explain how you arrived at those values. Report the accuracy of your perceptron and logistic function on the test set with those selected hyperparameter values. Finally, compare and evaluate the training and performance of both models against each other.