

CS373 HW4

Due Date: April 20, 11:59 PM

Instructions for submission: In this programming assignment you will implement the Perceptron and the Naive Bayes algorithms and evaluate them on the Yelp dataset. Instructions below detail how to turn in your code and assignment on `data.cs.purdue.edu`.

You need to implement this assignment from scratch in Python. Do not use any already implemented models like those in the scikit-learn library. Also, programs that print more than what is required will be penalized. Your code needs to run in the server: `data.cs.purdue.edu`. Check that it works there by executing your script from the terminal.

For the coding implementations, submit your code as a Python program. For all other parts write your answers in a single PDF. Name this **`yourusername_hw4.pdf`**.

As usual, label the plots with the question number. Your homework **must** contain your name and Purdue ID at the start of the file. If you omit your name or the question numbers for the plots, you will be penalized.

To submit your assignment, log into `data.cs.purdue.edu` (physically go to the lab or use ssh remotely) and follow these steps:

1. Make a directory named `yourusername-hw3` (all letters in lower case) and copy your PDF file and Python file inside it. To do it remotely, use:

```
scp ./path/to/your-file.pdf your-id@data.cs.purdue.edu:./remote/path/from-home-dir/
```

2. Go to the directory containing `yourusername-hw3` (e.g., if the files are in `/homes/dan/dan-hw3`, go to `/homes/dan`), and execute the following command:

```
turnin -c s373 -p hw3 yourusername-hw3
```

(e.g. Dan would use: `turnin -c s373 -p hw3 dan-hw3` to submit his work)

Note that `s373` is the course name for turnin. **It is not a typo.**

3. To overwrite an old submission, simply execute this command again.
4. To verify the contents of your submission, execute the following command:

```
turnin -v -c s373 -p hw3
```

0 Specification

0.1 Dataset Details

You will use `yelp_cat.csv` for this assignment. This data set is part of the Yelp database (similar to the one you worked with in hw2). The continuous variables have been removed and only categorical variables are retained. The dataset has 20,000 rows and 15 discrete attributes.

Features: Consider the first 14 discrete attributes in `yelp_cat.csv` for \mathbf{X} .

Class label: Consider the attribute `goodForGroups` as the class label, i.e., $Y \in \{0, 1\}$.

0.2 Code Details

You are only allowed to use Python 2.7. You need to submit **three** python scripts `vanilla.py`, `avg.py` and `nbc.py`. Each of your python scripts should take the following arguments:

1. *trainingDataFile*: corresponds to a subset of the Yelp data (in the same format as `yelp_cat.csv`) that should be used as the training set in your algorithm.
2. *testingDataFile*: corresponds to another subset of the Yelp data (in the same format as `yelp_cat.csv`) that should be used as the test set in your algorithm.

Note: Your perceptron models (`vanilla.py`, `avg.py`) also need to take an additional parameter corresponding to the maximum number of iterations it should run. More details mentioned in the perceptron section.

Your code should read in the training/test sets from the csv files, learn the respective model from the training set, apply the learned model to the test set, and evaluate the predictions with zero-one loss. Details on zero-one loss are given in the evaluation section.

Your input and output should look like this:

```
$ python vanilla.py train-set.csv test-set.csv 5
ZERO-ONE LOSS=0.2305
```

```
$ python avg.py train-set.csv test-set.csv
ZERO-ONE LOSS=0.2305
```

```
$ python nbc.py train-set.csv test-set.csv
ZERO-ONE LOSS=0.2305
```

Please stick to the given format. If the output is not in the correct format you **will** be penalized.

1 Perceptron (40 Points)

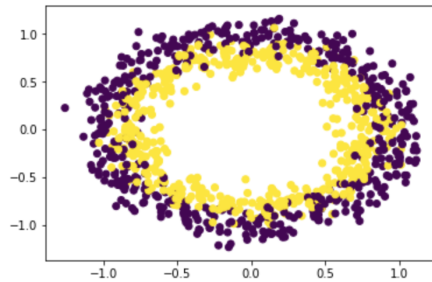
1.1 Perceptron Details (20 Points)

1. The perceptron model as we saw in class is given as:

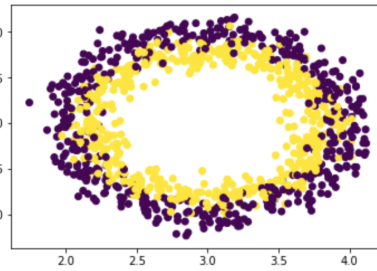
$$f(x) = \begin{cases} 1, & \sum w_j x_j \geq 0 \\ 0, & \sum w_j x_j < 0 \end{cases}$$

There is an important term called the *bias* missing from the equation above. Write the equation for the model with the bias term included. What is the significance of this term? Explain this using example plots for both cases.

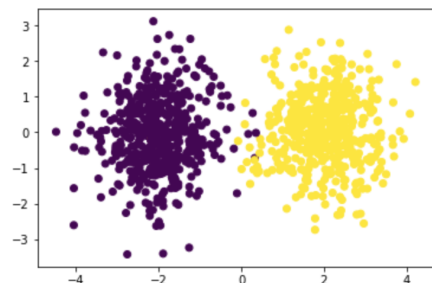
2. Below there are four figures that show the distribution of datapoints with binary classes. The two colors denote the different classes. For each of these, reason whether the following do would give a high (> 0.95) classification accuracy.
 - (i) a perceptron without bias
 - (ii) a perceptron with bias



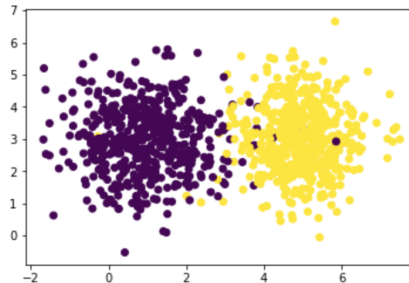
(a)



(b)



(c)



(d)

3. For the given Yelp data suggest a way of engineering a new feature from the given set(or subset) of features. Explain your answer and justify how this new feature might be capturing some form of complex behavior.
4. The pseudocode for the vanilla perceptron algorithm is given below (Algorithm 1). What do the weights and bias terms denote? Why do we need to update these in

a perceptron? How do line 8 and line 9 in the pseudocode achieve this? (**Hint:** Assume binary classes (0,1))

Algorithm 1 Vanilla Perceptron

```

1: function TRAIN(D, MaxIter)
2:    $w_i \leftarrow 0$ , for all  $i = 1, \dots, D$  ▷ Initialize Weights
3:    $b \leftarrow 0$  ▷ Initialize Bias
4:   for  $iter = 1, \dots, MaxIter$  do
5:     for all  $(x, y) \in \mathbf{D}$  do
6:        $error \leftarrow y - f(x)$ 
7:       if  $error$  then
8:          $b \leftarrow b + error$  ▷ Update Bias
9:          $w_i \leftarrow w_i + (error \times x_i)$ , for all  $i = 1, \dots, D$  ▷ Update Weights
10:    return  $b, w_0, \dots, w_D$ 
11: function PREDICT( $b, w_0, \dots, w_D, \hat{x}$ )
12:   return  $f(\hat{x})$ 

```

5. **Average Perceptron** In an average perceptron, instead of returning the updated weights/bias at the end of the training, we return the average of all the weights/bias calculated in each iteration for each data point.

- a) Write a pseudocode for the **Average Perceptron**. (**Hint:** You only need to add a few lines to the given pseudocode of the vanilla perceptron.)
- b) What is the advantage/disadvantage of doing this?

1.2 Perceptron Implementation (20 Points)

1. Implement the vanilla perceptron classifier. Name this file **vanilla.py**.
2. Implement the average perceptron classifier. Name this file **avg.py**.

Data: To make your Perceptron work, you will need to create binary features(similar to what you did in the last homework). You can assume you will not see any new attribute value other than what is in the data provided to you.

Hint: To create binary features, you could use `pd.get_dummies(X, columns=X.columns.values)`.

Please note here `X` needs to be a data frame. Once you have converted it to binary features, you can convert it to a matrix as you did in the last homework.

Note: As mentioned earlier, the two scripts need to take an additional argument *MaxIter* corresponding to the maximum number of iterations it should run. Set the default value of *MaxIter* = 2 in the case no **third** argument is specified.

2 Naive Bayes (40 Points)

Smoothing: For this assignment, you will use Laplace smoothing in the parameter estimation. For an attribute X_i with k values, Laplace correction adds 1 to the numerator and k to the denominator of the maximum likelihood estimate for $P(X_i = x_i|Y)$.

2.1 Naive Bayes Details (20 Points)

1. Write down the mathematical expression for $P(Y|X)$ given by the Naive Bayes Classifier(**Hint:** It's not just the Bayes equation).
2. Suppose your data has binary class labels , i.e $y \in \{0, 1\}$, write the expression for predicting the class for a given input row. You will use this expression in your implementation.
3. State the naive assumption that lets us simplify the expression $P(X|Y)P(Y)$. Is this assumption true for the given Yelp data? Explain why or why not?
4. What part of the expression corresponds to the class prior? Considering the entire Yelp data as the training dataset, calculate the maximum likelihood estimate for the class prior with and without smoothing. What is the effect of smoothing on the final probabilities?
5. Specify the full set of parameters that need to be estimated for the NBC model of the Yelp data. How many parameters are there?
6. For the Yelp data, explicitly state the mathematical expression for the maximum likelihood estimates (with smoothing) of the CPD parameters for the attribute `alcohol` conditioned on the the class label `goodForGroups`.
7. Consider the entire Yelp data as the training dataset and `goodForGroups` as the class label. Estimate the conditional probability distributions of the following attributes with and without smoothing:
 - a) stars
 - b) waiterService
 - c) caters
 - d) attire

What is the effect of smoothing (e.g., any difference compared to 4.)? Which attribute shows the most association with the class? Why?

2.2 Naive Bayes Implementation(20 Points)

1. Implement a Naive Bayes classifier in python. Name this file **nbc.py**.

Data: Since the Naive Bayes can handle multi-valued discrete attributes, there is no need to create binary features. Here you cannot assume that you may not see a new attribute value other than what is being provided to you.

Hint: To avoid underflow issues you could use **log** values.

Note: Your classifier should be able to handle attribute values that appear in test data but were not observed in the training data. One way to do this is have a separate label to accommodate not-in-training labels for all attributes that can have a variable number of values.

3 Analysis(20 Points)

Now you will evaluate your models **avg.py** and **nbc.py** using cross validation and learning curves. Cross-validation is a powerful tool to tune your hyperparameters and gauge how well your learned model generalizes. The basic idea behind cross-validation is that a subset of the data is kept aside(called the validation set) before the training begins. When we have a learned model, the validation set can be used to test the performance of our model.

In k -fold cross-validation, the data is divided into k subsets. In each iteration we use one such subset as the test set and train on the remaining $k - 1$ subsets as training set. This process is repeated k times. We then take the average error across all k trials. This gives a more accurate measure of model quality than if you were to hold out some fixed subset of the training data as validation set.

A learning curve is a way to plot the evolution of the loss as size of the training set changes. To evaluate the learned models you will use the following loss functions:

1. Zero-One Loss (for both models)
2. Squared Loss (for Naive Bayes)

Let $y(i)$ be the true class label for example i and let $\hat{y}(i)$ be the prediction for i . Then zero one loss for the test dataset T of n instances is:

$$Loss_{0/1} = \frac{1}{n} \sum_{i \in n} \begin{cases} 0 & \text{if } y(i) = \hat{y}_i \\ 1 & \text{otherwise} \end{cases}$$

Let p_i refer to the probability that the Naive Bayes classifier assigns to the true class of example i (i.e., $p_i := p(\hat{y}(i) = y(i))$). If $p_i > 0.5$, the prediction for example i will be correct (i.e., $\hat{y}(i) = y(i)$), but otherwise if $p_i < 0.5$, the prediction will be incorrect (i.e., $\hat{y}(i) \neq y(i)$). Then the squared loss for the test dataset T of n instances is:

$$Loss_{sq}(T) = \frac{1}{n} \sum_{i \in n} (1 - p_i)^2$$

In general you would use cross-validation to tune the hyperparameters of your model (e.g. *MaxIter* in the Perceptron model), but for the following problems you should use the default value.

1. For each % in [1, 10, 50, 70]:
 For i in [0...9]:

- Randomly sample % of the data to use as the training dataset.
 - Use the remaining $(100 - \%)$ of the data as the test dataset.
 - Learn a model from the training data and apply it to the test data.
 - Measure the loss on the test data.
- a) Record the **mean** zero-one loss observed across the ten trials for each training set size (i.e., sample %) for both Naive Bayes and Average Perceptron classifiers.
 - b) Record the **mean** squared loss across the ten trials for each training set size for the Naive Bayes classifier.
2. a) Plot a learning curve of training set size vs. zero-one-loss for both Naive Bayes and Perceptron.
- b) To the above plot add the baseline default error that would be achieved if you just predicted the most frequent class label in the overall data.
- c) Discuss your results:
- How the two classifiers compare with each other and with the baseline.
- How is zero-one loss impacted by training set size.
3. a) Plot a learning curve of training set size vs. square-loss for Naive Bayes.
- b) Discuss how zero-one loss performance compares to square-loss. Which do you think is a better metric and why?