PROGRAMMING ASSIGNMENT 5

Understanding Data

Submission Date : 03/01/2020
Due Date : 12/01/2020 23:59
TA : Yunus Can Bilge

Accept your *5th Assignment*.

Introduction:

In this experiment, you will implement a simple Single Layer Neural Network in order to solve a real wold machine learning problem with a real world data. In this experiment, you will get familiar Python data science libraries such as; *Pandas*, *NumPy*, and *Matplotlib*. At the end, you will train the neural network to classify clinical cases to either *benign* or *malignant*.

Assignment:

Preliminary Notes

Figure 1: A comic about machine learning from [1].

In this assignment, you will use the technics of a subfield of computer science; which is machine learning. Machine learning can simlpy be defined as building a statistical model from a dataset in order to solve a real world problem.

There are types of machine learning. However in this assigment we basically use Supervised Learning. In supervised learning, the dataset consists of each sample's feature vector and label. A feature vector is a vector that each dimension describes a sample. For instance; if we

have a dataset that represents carbonated beverages, then the features can be fat, sodium, protein, calories etc. The label can be belong to finite set of classes 0, 1, ..., C.

In our dataset we have 9 dimensional feature vector and a class label which can be either 0 or 1. Two samples from our dataset are;

ClumpThickness	UniformityofCellSize	UniformityofCellShape	MarginalAdhesion	SingleEpithelialCellSize	BareNuclei	BlandChromatin	NormalNucleoli	Mitoses	Class
5	1	1	1	2	1	3	1	1	0
5	4	4	5	7	10	3	2	1	0

For more information about machine learning, please look at [4].

Part 1 - Reading Data

The first step in any machine learning problem is to obtain data. In this project, we use Breast Cancer Wisconsin Data Set which can be accessed at *breast-cancer-wisconsin.csv*. There are total of 699 samples in the dataset and each sample has 9 features and a class attribute. The features are; *ClumpThickness, UniformityofCellSize, UniformityofCellShape, MarginalAdhesion, SingleEpithelialCellSize, BareNuclei, BlandChromatin, NormalNucleoli, Mitoses*. Each attribute can get a value between 1 to 10. The class attribute can be either; *benign(0)* and *malignant(1)*.

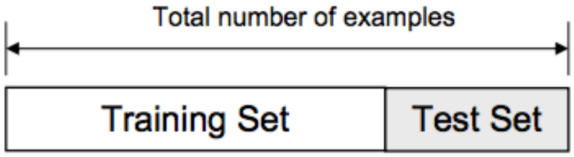

Figure 2: A visualisation of the splits taken from [2].

In this part;

1. **You will read the dataset (csv file) with Pandas library. *Pandas* is a data structure and data analysis tool for Python. [5 pts]**
2. **You are also expected to split your dataset into 2 sets; train and test sets. The training set is used to train the model. In other words, your model will learn from the training set and tune the weights. For the evaluation purposes; you will use the test set. You will obtain accuracy results on test set. You will use 80% of the data for training set and 20% of the data for test set. [5 pts]**
3. **There are some values missing in the dataset. You will have complete these missing values using one of the techniques that you learn in your class. [10 pts]**
4. **You are also expected to visualize pairwise correlations of the attributes of training set. You will use *matplotlib* for that task. [10 pts]**

Part 2 - Implementing a Single Layer Neural Network

Neural networks are simply modeled after the human neuron. Just as the brain is made up of numerous neurons, Neural Networks are made up of many activation units.

A unit that we use in this assignment is a binary linear classifier.

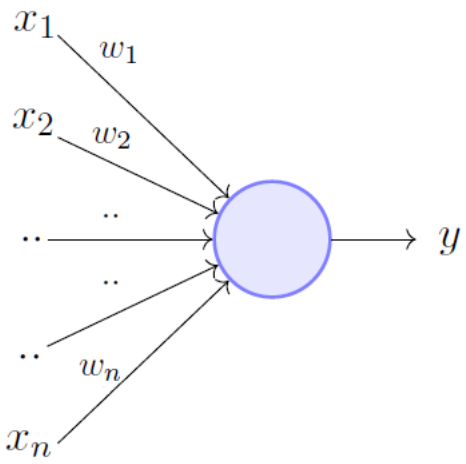

Figure 3: A binary linear classifier unit

This unit contains 5 parts;

- input values (x_1, x_2, x_3, ..., x_n) – n is the number of features/attributes
- weights (w_1, w_2, w_3, ... w_n)
- weighted sum

$$\sum_{i=1}^{n} x_i * w_i$$

- activation function/normalizing function (sigmoid)
- output (y)

Inputs to a unit is the features from training set. Weights are the parameters that you will learn. Activation functions modify the input data in order to learn complex relations between input and output data. In this assignment, the activation function squash the input data into a narrow range to make training more stable. Your task is to learn weights in order to correctly predict output y given input x.

In Neural Networks, you forward propagate to predict output and compare it with the real value and calculate the loss. Loss value tells us how good our model make its predictions which tells us how to change our prameters to make our model predict more accurate.

In order to minimize the loss, you propagate backwards with respect to each weight value by finding the derivative of the error. Thus, weights will be tuned. Neural networks are trained iteratively. After each iteration error is calculated via the difference between prediction and target.

As a/an normalizing/activation function you will use Sigmoid function;

$$h_\theta(x) = \frac{1}{1 + e^{-0.005*x}} \tag{1}$$

The derivative of the normalizing/activation function is;

$$\frac{\partial h_\theta(x)}{\partial x} = 0.005 * x * (1 - x) \tag{2}$$

In this part of the assignment, you will mainly use *numpy* library.

You will iterate forward and backward many times to learn from data and tune weights with respect to it.

Let's say we have the following matrices; The training data includes 4 samples and each sample has 3-dimensional features. Labels can be either 0 or 1. We start by randomly initializing weights and during the process we learn and tune the weights matrix.

training_inputs ; $\begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$ training_outputs / labels ; $\begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$ random initial weights ; $\begin{bmatrix} -0.16595599 \\ 0.44064899 \\ -0.99977125 \end{bmatrix}$

One iteration looks like this;

Forward Propagation;

```
outputs = training_inputs dot weights ## dot means dot product
outputs = sigmoid(outputs)
```

Backward Propagation;

```
loss = training_outputs - outputs
tuning = loss * sigmoid_derivative(outputs)
weights += training_inputs^T dot tuning   ## ^T means Transpose
```

Results on given matrices;

$outputs = training\_inputs \cdot weights$

$$\text{outputs} = \begin{bmatrix} -0.99977125 \\ -0.72507825 \\ -1.16572724 \\ -0.55912226 \end{bmatrix}$$

outputs = sigmoid(outputs)

$$\text{outputs} = \begin{bmatrix} 0.49875029 \\ 0.50090635 \\ 0.50145715 \\ -0.4993011 \end{bmatrix}$$

loss = training_outputs - outputs

loss =

$$\begin{bmatrix} -0.49875029 \\ 0.49909365 \\ 0.49854285 \\ -0.4993011 \end{bmatrix}$$

tunings = loss * sigmoid_derivative(outputs)

$$\text{arrangements} = \begin{bmatrix} -0.00062343 \\ 0.00062613 \\ 0.00062682 \\ -0.00062413 \end{bmatrix}$$

$weights+ = training\_inputs^T \cdot arrangements$

$$\text{updated weights after 1 iteration} = \begin{bmatrix} -0.16470304 \\ 0.44065099 \\ -0.99976586 \end{bmatrix}$$

In this part;

1. **you do not have to implement from scratch. You are expected to complete the given code. You have to implement forward and backward propagation using *numpy* library. [20 pts for forward propagation, 20 pts for backward propagation]**

Helper Links

- Transpose - `https://chortle.ccsu.edu/VectorLessons/vmch13/vmch13_14.html`
- Transpose - `https://docs.scipy.org/doc/numpy/reference/generated/numpy.transpose.html`
- Dot product - `https://www.mathsisfun.com/algebra/vectors-dot-product.html`
- Dot product - `https://docs.scipy.org/doc/numpy/reference/generated/numpy.dot.html`

Part 3 - Plot Loss and Accuracy in each Iteration for Test Set

In order to plot a graph you will use matplotlib as in Part 1. In test set you will get prediction by the following formula;

```
test_output = sigmoid( test_inputs dot weights ) \# weights = learned weights
```

You are expected;

1. **to predict the final value(either 0 or 1) for each row(sample) in your test set. You will map each test_output row into 0 or 1 by checking if the value is greater than 0.5 or else. If the value is greater than 0.5 then your prediction is 1 and else is 0. [10 pts]**

Accuracy will be calculated as; $accuracy = TruePositiveCount(TP)/TotalNumberofSamples$

TP = is an outcome that model predicts the true class correctly.

For instance; the predictions matrix and label matrixes are for 4 sample;

predictions matrix; $\begin{bmatrix}1\\0\\0\\1\end{bmatrix}$ label matrix; $\begin{bmatrix}1\\0\\1\\0\end{bmatrix}$ TP count = 2 and Total Number of Samples = 4

Accuracy = 2/4 = 50%

Therefore you will;

1. **plot accuracy change in test set for each iteration. You will also expected to plot loss change in each iteration. Your loss matrix will have more than one values in order to plot a loss value you will average the loss matrix in each iteration. [20 pts]**

For instance; A plot for accuracy and loss change on train and test data for 7 iterations. Important to note that, in our assignment we only expect training loss plot and test accuracy plot.

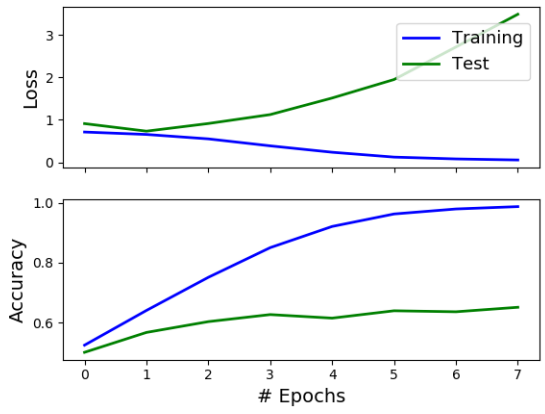

Figure 4: Train/Test accuracy and loss plots

You can use the given code for all 3 parts of the assignment. It is important to note that the given code usage is optional.

The aim of this assignment is;

1. Learning to use libraries

2. Learning to understand and analyse a given problem
3. The resulting accuracy percantage is not important. The important part is to understand the problem and implementing it.

Academic Integrity

All work on assignments must be done individually unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) will not be tolerated. In short, turning in someone else's work, in whole or in part, as your own will be considered as a violation of academic integrity. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else.

Important Notes

- Do not miss the submission deadline.
- Compile your code before submitting your work to make sure it compiles without any problems with *Python 2.7*, *numpy version 1.16.4*, *matplotlib version 2.2.4*, and *pandas version 0.24.2*.
- Save all your work until the assignment is graded.
- The assignment must be original, individual work. Duplicate or very similar assignments are both going to be considered as cheating. You can ask your questions via Piazza and you are supposed to be aware of everything discussed on Piazza. You cannot share algorithms or source code. All work must be individual! Assignments will be checked for similarity, and there will be serious consequences if plagiarism is detected.
- You must submit your work with the file hierarchy as stated below:

→ <assignment5.py>

References

1. `https://xkcd.com/1838/`
2. `https://towardsdatascience.com/train-test-split-and-cross-validation-in-py`
3. `https://towardsdatascience.com/perceptron-learning-algorithm-d5db0deab975`
4. The Hundred-Page Machine Learning Book by Andriy Burkov, http://themlbook.com/wiki/doku.php
5. `https://ml-cheatsheet.readthedocs.io/en/latest/nn_concepts.html`