Machine Learning

Lab 2 – Classification Models & Model evaluations

Jay Morgan

November 2022

1 Introduction

Welcome to the second lab for the Machine Learning course. In this lab we're going to be implementing various methods to evaluate a logistic classifier.

This lab will build on what we created in the last lab, transforming the linear regressor into a logistic classifier.

1.1 How to answer the questions

In these lab sessions we will implement what we've learn during the theory lectures. These implementation will need to be using the Python Programming Language. You're allowed to use 3rd party libraries such as pandas, matplotlib, numpy. But it is not allowed to use a library that implements exactly machine learning model we're learning about. For example, in this lab, we're learning about linear models, so you cannot use a library that provides functionality for computing a linear model. If you're uncertain about whether a library can be used, please do ask!

I am going to give you the maximum flexibility with how you want to program these linear models: you can create a Python script, or if you like, you can use jupyter notebooks. In both cases, You should preface your implementation with which question you're answering with a comment (if you're writing a python script), or markdown (if you want to use jupyter) like this:

```
# Q1. Download, parse, and load the data into a pandas dataframe,
# print the first 5 rows to ensure the data is formated correctly
```

```python
import pandas as pd

def load_data(filepath: str) -> pd.DataFrame:
    # load_data logic
    return df
```

You are to work individually.

In summary:

- Work individually.
- Implementation should be using the Python programming language.
- You can use supplementary libraries such as pandas, matplotlib, numpy, but cannot use a library that provides a function call for the assignment.

1.2 Submission Procedure

To hand in your assignment, please zip all of your source code (**do not include the data**) into a zip-archive named using the following format:

`<first-name>-<surname>-machine-learning-lab-2.zip`

replacing `<first-name>` and `<surname>` with your name. Please send this assignment to my email address: `jay.morgan@univ-tln.fr` using the subject `Machine Learning Assignment 2` by the end of the lab session. I will accept late assignments, but will be deducted score accordingly.

2 Questions

2.1 Question 1

Copy the existing code from Lab 1, creating a new folder for this lab and pasting in the code for the linear regressor.

2.2 Question 2

Download the Iris dataset from `https://archive.ics.uci.edu/ml/machine-learning-databases/iris/`. The dataset can be downloaded from `iris.data`. Load the data into a pandas dataframe.

For this lab, we're going to be performing a binary classification problem, but this dataset has 3 classes: setosa, virginica, and versicolor. So we want to take this multi-class problem and transform it into a binary classification.

Create a new column for the dataset called `target`. The value of target will be 1 if the row contains a setosa flower, else the value is 0. There should be $\frac{1}{3}$ rows with the value of 1, the rest should be 0.

2.3 Question 3

For this question we want to take this dataset of 150 rows, and split it into a train, test, and validation dataset, using the following proportions for each split:

- Training: 70%
- Validation: 10%
- Testing: 20%

Sample data for each subset using stratified sampling. I.e. the training data should have roughly $\frac{1}{3}$ positive samples, the testing and validation dataset should also have roughly $\frac{1}{3}$ positive samples.

2.4 Question 4

Using the linear regression model you created in the previous lecture, transform it into a logistic regressor by applying the logistic function to the output of the model. The loss function for this model should be binary cross entropy.

Select two columns from the Iris dataset (i.e. petal length and petal width), and using these two columns, train a logistic regressor using gradient descent, measuring the gradient using finite differences approximation. This means that instead of having a single slope variable, we have multiple:

$$\hat{y} = \sigma(\beta_0 + \sum_{i=1}^{m} x_i \beta_i)$$

where $\hat{y}$ is the model's probability prediction, σ is the logistic/sigmoid function, β_0 is the intercept, β_i is the coefficient that modulates the x_i variable.

I've made a start for you, please fill in the '`#TODOs`':

```python
import numpy as np

def bce(y, yhat):
    # TODO: apply the binary cross entropy function returning the loss
    return loss

class LogisticRegressor:
    def __init__(self, n_features: int = 2):
        self.params = np.random.randn(n_features + 1)

    def logistic(self, x):
        # TODO: apply the logistic function
        return x

    def __call__(self, x, logits=False):
        y = self.params[0] + self.params[1:] @ x.T
        if not logits:
            y = self.logistic(y)
        return y

    def fit(train_x, train_y, valid_x, valid_y, epochs: int = 100, lr: float = 0.01):
        # TODO: train the model using gradient descent and finite-differences
        for epoch in range(1, epochs+1):
            for xi, yi in zip(train_x, train_y):
                # calculate loss and update model parameters using gradient descent
            for xi, yi in zip(valid_x, valid_y):
                # calculate validation loss (BUT DON'T UPDATE MODEL PARAMETERS!)

    def predict(x, logits):
        return self(x, logits=logits)
```

2.5 Question 5

As gradient descent is iterating, store (using class variables), the training and validation loss.

Visualise the training and validation loss. Is there a point at which the model begins to over fit? How do you know that the model is beginning to overfit by looking at these curves?

2.6 Question 6

Predict the class labels for the testing set.

For the testing set, calculate the:

- TP – number of true positives
- TN – number of true negatives
- FP – number of false positives
- FN – number of false negatives

2.7 Question 7

Calculate the precision and recall and F_1 score.

```
def precision(y, yhat):
    # calculate the precision and return it
    return

def recall(y, yhat):
    # calculate the recall and return it
    return

def f_beta(y, yhat, beta=1):
    pr = precision(y, yhat)
    rc = recall(y, yhat)
    # calculate the f_beta score and return it
    return


pr = precision(y, yhat>=0.5)
rc = recall(y, yhat>=0.5)
# ...
```

2.8 Question 8

Generate a report using the precision, recall and F_1 and confusion matrix. The report should be printed like:

```
|        |          | Predicted |          |
|        |          |  Positive | Negative |
| Actual | Positive |         5 |        2 |
|        | Negative |         3 |        1 |
```

- Precision: 0.6
- Recall: 0.6
- F_1 Score: 0.6

Replacing the scores with the correct numbers.

2.9 Question 9

Calculate the true-positive and false positive rate, and from these values generate a ROC curve.

```
def roc(y, yhat, threshold_step=0.01):
    # iteratively increase the threshold by threshold_step,
    # calculating the TP and FP rate for each iteration. This function
    # should return two lists, a list of TP rates, and a list of FP
    # rates.
    return tp, fp

tp, fp = roc(y, yhat)
# visualise the ROC curve here
```

2.10 Question 10

Now that you've created a logistic classifier for two features of the Iris dataset and have created some analytic results. Select another two columns (i.e. petal width and sepal length, or petal length and sepal width). Create a different logistic classifier using these new columns and create the same results as you did with questions 8 and 9.

Compare these two models trained with different columns. Which model is best, and why do we know that it's the best?

3 Marking Criteria

Criteria	**Marks**	0 % **No attempted**	0-30 % **Attempted**	30-80 % **Correct**	80-100 % **Good solution**	**Score**
Question 1	0					
Question 2	10					
Question 3	15					
Question 4	20					
Question 5	5					
Question 6	5					
Question 7	5					
Question 8	10					
Question 9	10					
Question 10	10					
Code comments are helpful	2					
Variable names are descriptive	2					
Functions include docstrings	2					
Functions are generic	4					
					Total	