Programming Homework, CSE 440
Spring 2019, Michigan State University,

Perceptron Models for Simple Image Classification

Sari Saba-Sadiya
CSE 440, Spring 2019

Michigan State University
sadiyasa@msu.edu

ABSTRACT. As seen in class, the perceptron is a simplified model of a biological neuron that can learn to perform classification tasks. While relatively simple, these classifiers perform surprisingly well even on non trivial tasks such as hand written digit recognition.

KEYWORDS: Perceptron Classifiers, UCI ML hand-written digits dataset.

Prelude

This document will be censored for academic honesty concerns, an uncensored version will be made available upon request after all projects are submitted. A censored text looks like this: [redacted].

For training and testing we will use the UCI ML hand-written digits datasets first published in [1, 2]. We will follow the version of multiclass perceptron that was presented in the *Perceptrons and Logistic Regression* lecture in class.

Introduction

In this homework you will implement a simple perceptron and train it to perform simple hand written digit recognition. This task involves recognizing the

digit presented in an 8×8 (gray scale) pixel array, as the digit can be anything between 0 and 9 simply guessing will result in a 10% accuracy on average. While non trivial, using machine learning one can train even a simple classifier to perform the task with over 85% accuracy.

0.1. Grading

Before we begin, I would like to offer a few words of encouragement:

- As long as your perceptron classifier (and the other classifiers you will use) achieves an at least 80% accuracy and runs in less than 3 minutes you will get full credit!

- No monkey business! Do not use `sklearn` or any off the shelf perceptron classifiers for phase 2 (see below). Also, sending your code to someone else / asking for someone else for his code will get you an academic suspension.

- My solution had less than 40 lines of code and gets 93% accuracy. This is not a complicated coding assignment. Just make sure to document your code.

There are three phases for this project: 1. getting and loading the dataset. 2. implementing, training and testing the perceptron. 3. testing some off the shelf classifiers and comparing them against your perceptron. The three following sections focus on the different phases respectively.

1. Packages, Data, and Setup

The package `sklearn` is a popular machine learning library for python. In addition to implementations of many algorithms and tools for statistical analysis this package also contains many small datasets of anything from boston house prices to the chemical structure of wine. We will start by loading the packages:

```
import numpy as np
import sklearn
from sklearn import datasets
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
```

The first line loads `numpy` which will be used for the perceptron, the following three lines load `sklearn` the datasets and methods that construct the confusion matrix (see wikipedia if you are unfamiliar) which we will use for analysis of the classifier results. The last two lines import methods for data visualization.

Next we will import the data and split it to training and testing sets. The following lines load the UCI ML hand-written digits datasets[1, 2] which contains 1797 images of hand written digits (each 8×8). We split this data to 1700 training images and the rest for testing, you are encouraged to change the training size and see what happens.

```
## load the data set
img,label=sklearn.datasets.load_digits(return_X_y=True)
TRAIN_SIZE = 1700
# split the data set
train_img,test_img = img[:TRAIN_SIZE],img[TRAIN_SIZE:]
train_label,test_label = label[:TRAIN_SIZE],label[TRAIN_SIZE:]
```

You can use `plt.imshow` to display one of the input arrays as an image. As can be seen below, while some of the digits are easily recognizable, others not so much.

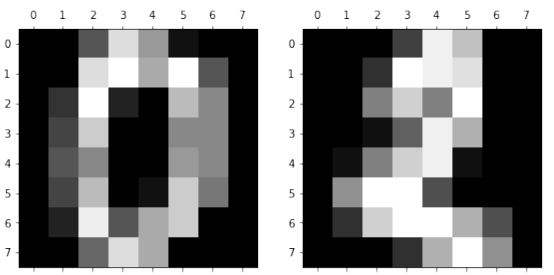

FIG. 1: *Hand written 0 and 2 from the dataset.*

Before we begin, ask yourself, what pairs of digits might be confusing? depending on the hand writing, 5 and 9 might appear similar, so might 3 and 8. We will see in the following section that this might present difficulties to our classifiers as well.

2. The Perceptron

As discussed in class different activation functions ($f(x) = x/tanh(x)/sigmoid(x)$), will produce different results. We begin by defining these functions before focusing on our preceptron.

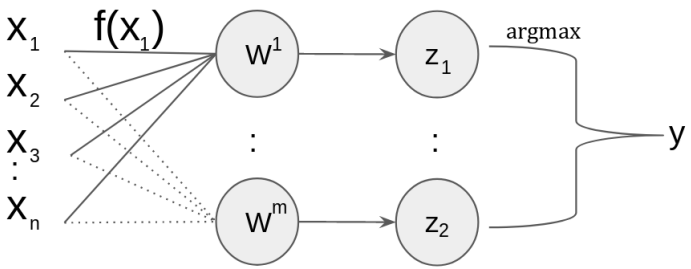

FIG. 2: *Our multiclass perceptron. Not that W^i is a matrix while x_i and z_i are elements in the vectors x and z.*

First we initialize the perceptron values, per the lecture slides we should make sure that the weights W are initialized to zeros. A multiclass perceptron is simply multiple binary peceptrons put together, one for each possible label. Hence we initialize 10 different W matrixies, each is $n \times n$ where $n = 8 \cdot 8 = 64$ is the size of our input (the handwritten digit images).

Now given the input x and the weights W we want to predict the label. As seen in class, we only need to follow the equation:

$$y = \underset{i}{argmax}\{W^i \cdot f(x)\}$$

Problem:Prediction

Calculate $z = W^i \cdot f(x)$ and then $y = \underset{i}{argmax}\{z\}$. You may use any `numpy` or `math` function, or define your own functions.

Next we want to learn the correct weights. The method `one_update` will perform a single update given an input and the correct label. Following the slides we first use our predict method. If the image was labeled correctly we do not perform any changes. Otherwise we look at each W^i and update $\pm f(x)$ depending on whether $i = y$ or not.

Problem:Learn

Fill in the `one_update` method so that the perceptron successfully learns to classify the images.

The actual learning, or training, is done by simply iterating over all the training data and performing an update for each image label pair, the method `train` does exactly that using the `one_update` you have completed. Please look over it!

Finally the testing is done by iterating over all the testing images and counting how many were indded labeled correctly (meaning the label and prediction are the same).

Problem:Test

Fill in the `test` method. Ofcourse you may (and should) use other methods in the perceptron class.

2.1. Testing the Perceptron

One run over the data is often not enough for effective learning, instead we run multiple epochs. All we need is to initialize the perceptron function with the wanted activation function and train it the number of epochs we want. Note how different activation functions produce different results.

Problem:Teasting the perceptron

For each `perceptron.train(`▉,▉`)` and `perceptron.test(`▉,▉`)` fill in the training and testing sets we got from section 1. Hint: the 4 variables already exist, you only need to fill everything in.

2.2. Analysis of the results

The testing function gives us a list of the predicted labels, since we already have the true labels we can see how many times the digit 3 was misclassified as 8. Infact we can do this for all labels and get a 10×10 matrix. A perfect classifier will have non-zero elements only at the diagonal. Look at the following confusion matrix (this is not the one you will get).

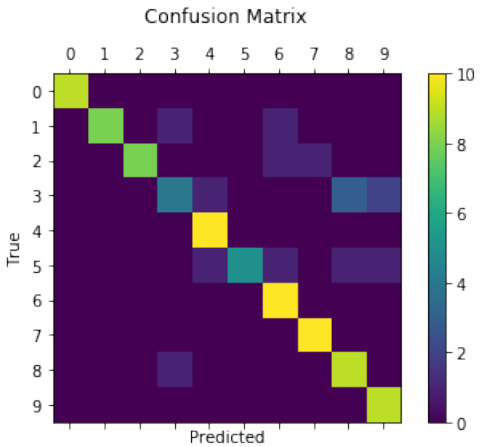

FIG. 3: *An example of a confusion matrix.*

The biggest value outside the diagonal is for the true label 3 and the predicted label 8. Meaning the most common mistake was misclassifying a 3 as an 8. What kind of mistakes do you get for your perceptron?

3. Off the shelf classifiers

While it is important that you are able to implement classifiers yourself, it is also importent that you learn how to utilize existing packadges and off the shelf methods. You can read about each of the algorithms and find variables online, we will focus on decision trees, Naive Bayes classifiers and an implementation of the Perceptron.

Problem:Off the shelf classifiers

For each `_classifier.fit(`▉`,`▉`)` and `_classifier.score(`▉`,`▉`)` fill in the training and testing sets we got from section 1.

Since the classifiers are already implemented all you have to do is test them. What classifiers do better than our perceptron? you can use the confusion matrix to investigate further what different errors they make,

4. Conclusion

Neural Networks are simply glorified perceptrons. As you saw in this work sheet you can implement simple NN and there are off the shelf implementation of NN and other classifiers that will solve any problem you might face easily. Do not be intimidated! this class might have seemed confusing at times but your work here and the other work sheets testifies to the fact that with good enough data and understanding of basic concepts you are able to use and modify machine learning algorithms to successfully get good predictions.

REFERENCES

[1] C. Kaynak, "Methods of combining multiple classifiers and their applications to handwritten digit recognition," Master's thesis, Institute of Graduate Studies in Science and Engineering, Bogazici University, 1995.

[2] E. Alpaydin and C. Kaynak, "Cascaded classifiers," *Kybernetika*, vol. 34, pp. 369–374, 1998.