FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

Artificial Intelligence

Tarea 1: Evaluating Linear Models in a Multiclass Environment

Carla Vairetti & José Manuel Saavedra
Lab: Joaquín Curimil

Deadline: August 25th 2024

1 Objective

To understand linear models through a practical approach in the task of the classification of handwritten digits.

2 Description

Handwritten digit recognition is a traditional task in machine learning. Here, a machine learning model receives a digit image as input, and it should predict the corresponding class. This task has ten classes corresponding to the digits from 0 to 9. Figure 1 illustrates a sample of handwritten digits.

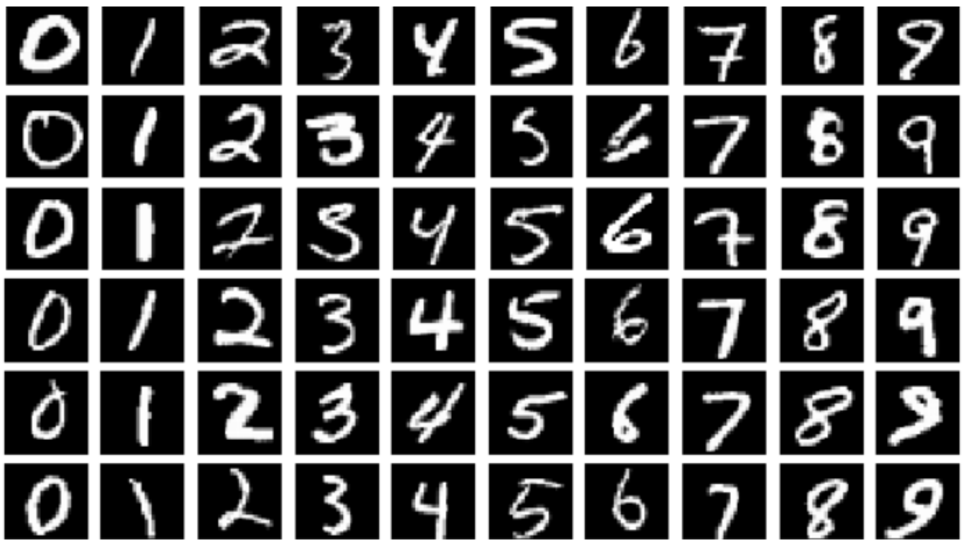
Figure 1: A sample of handwritten digits from the MNIST dataset.

2.1 Logistic Regression for Multiclasses

During our lessons, we learned the fundamentals of logistic regression as a linear classifier. We observed that this model tries to find a hyperplane to separate the feature space. Thus, the linear model is simply a binary classifier. However, now that we are in a multiclass environment, we should adapt or extend the binary classifier to a multiclass one.

A simple strategy to address this problem is through the **one-vs-all** method, where we train as many classifiers as classes we have. Each classifier will try to separate the target class from the others. Thus, in our problem, we must train ten classifiers under the one-vs-all strategy. To this end, each classifier is trained using n examples from the target class and n examples from others. Figure 2 illustrates the multiclass logreg for the handwritten digit problem.

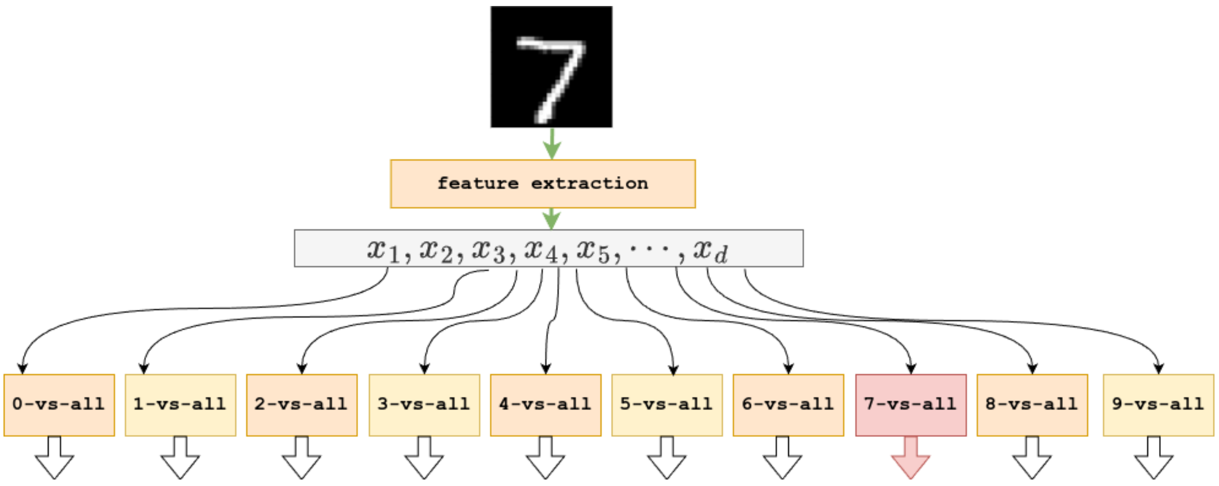

Figure 2: The one-vs-all model.

2.2 Defining the Inputs

The input to a model is represented by a vector x $\int d$. This representation is the key for a model to have a good performance. In the context of images, we need a feature extractor to convert the plain image to a vector in a d-dimensional space, where d can take many diverse values.

Coming up to our specific problem, we will experiment with three strategies to convert a digital image into a vector. At this moment, the detailed aspects of the feature extractor are not relevant; we only need to know the general idea of the strategies. Thus, below, we describe the three strategies you must try.

1. The image as a vector (plain representation). Just reshape the input image into a vector.
2. An histogram of orientations (HOG): this was the preferred feature extractor previous to the deep learning era. HOG computes the orientations of the contours in an image to build a histogram of orientations. In this task you can use the function HOG from the library sckit-learn `https://scikit-image.org/docs/stable/auto_examples/features_detection/plot_hog.html`
3. A deep feature vector (provided by deep learning model): in this case, we compute a feature vector by a machine learning model. To this end, you can use our model named **mnist_model** that can be downloaded from `https://www.dropbox.com/scl/fi/ovucxclytv4m72xnvgsb0/mnist_model.zip?rlkey=jzzaw6azetqmt2xrh0qigvxtu&st=hleis61y&dl=0`

   An example how to use this extractor can be found at `https://github.com/jmsaavedrar/machine_learning/blob/main/ssearch_emnist.py`.

2.3 Datasets

In this project we will use a subset of the MNIST dataset named MNIST-5000. This subset consists of 5000 images for training and 500 images for validation. In both cases, the images are well distributed among the ten classes.

You can download the MNIST-5000 dataset from `https://www.dropbox.com/scl/fi/tvrvsl2nd7e1upcmazgm1/MNIST-5000.zip?rlkey=j5a7rauyg83p0x6ybltev9pgj&st=bz50c9jf&dl=0`.

For training each logreg model, you must use the 1000 images for each digit and a random sample with 1000 images representing the other classes.

2.4 Allowed Code

You are limited to use the code from `https://github.com/jmsaavedrar/machine_learning`. You can adapt, modify or extend part of that repository to solve the task.

2.5 Metrics

For the three experiments you must report the following metrics over the **validation dataset**:

1. Overall accuracy.
2. Accuracy by classes: you must present a table and a bar graphic.
3. The corresponding confusion matrix.

3 The Report

Any tasks must be accompanied with a technical report describing the process to solve the proposed task. This report must be organized as follows:

1. **Abstract**: this section should summarize the work, highlighting the achieved results.
2. **Introduction**: this section should describe the problem and the related concepts and tools involved in the task. (10%)
3. **Development**: this section should describe how you solved the task. Provide details of the methods you implemented to solve the problem. Please avoid putting isolated screenshots of your code; we require a clear explanation of your process. (40%)
4. **Results and Discussion**: this is the document's most important section. Here, you must explain the results according to the task specifications. It is important to study the results and describe why you achieved poor or high performance. (40%)
5. **Conclusions**: here, you should present the conclusions highlighting the achievements. (10%)

4 Constraints

1. You are allowed to work in pairs.
2. You are allowed to use any part of the code described in Section 2.4
3. You must deliver the source code together with the corresponding report.

5 The Deadline

You need to deliver the source code along with the corresponding report through Canvas. The deadline is August 25th, entire day.

Please, take into account that any task without source code or report will be graded with 1.0.