Computer Lab: Digit recognition

SD-TSIA 211

Radu Dragomir, Olivier Fercoq, Mohammed Abdullah, Victor Priser, Iyad Walwil

January 24, 2025

1 Submission and grading information

You can do the computer lab alone or in pairs. Please write a report and post it on `e-campus`. You can do it as a jupyter notebook or a pdf file.

Then, each of you will have to evaluate a couple of other students' reports and give comments.

Only the fact that you produce a report and evaluate your peers count in the final grade, so do not worry if you do not finish everything.

- 1 point for being present on the day of the lab on January 24th
- 1 point for submitting a report before February 5th
- 1 point for commenting 2 reports before February 20th

2 Dataset

We are going to use the MNIST dataset, which consists of images of written digits together with their label. As it is a very common dataset, it can be downloaded directly with a function of the Keras library.

The goal is to predict, given an image, the digit it contains. It is therefore a classification problem with 10 classes.

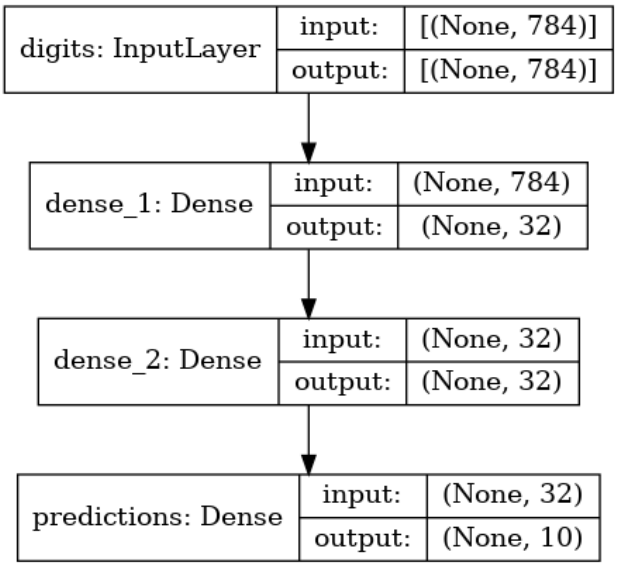
3 Dependencies

We will use the following packages: numpy, matplotlib, tensorflow and keras. You will need them to be installed in order to run the code.

On Télécom Paris's computers, you should have all the packages available by using `/cal/softs/anaconda/anaconda3/bin/jupyter notebook`

4 Model

We use a fully-connected neural network with three layers. Given an input image x, the neural network does a series of computations and returns a vector $\Phi_\theta(x) \in \mathbb{R}^{10}$, where



$[\Phi_\theta(x)]_d$ is the predicted probability that the image belongs to class $d \in \{0, 1, \ldots, 9\}$, and θ denotes the neural network parameters.

We train the neural network by solving the optimization problem

$$\min_\theta \frac{1}{n} \sum_{i=1}^{n} \ell(\Phi_\theta(x_i); y_i^{\text{true}}),$$

where n is the total number of training samples. The loss function ℓ is the categorical cross-entropy function defined as

$$\ell(\Phi_\theta(x_i); y_i^{\text{true}}) = -\sum_{d=0}^{9} y_{id}^{\text{true}} \log\left([\Phi_\theta(x_i)]_d\right),$$

where $y_{id}^{\text{true}} = 1$ if image i represents digit d and $y_{id}^{\text{true}} = 0$ otherwise.

The file `TP_MNIST_basic_functions.ipynb` or `TP_MNIST_basic_functions.py` provide the code for building the neural network model using Keras, defining the architecture and the loss function.

Question 4.1
How many optimization variables are we going to train using this model?

5 Stochastic gradient descent

We propose to implement SGD for training the model. Libraries such as Tensorflow allow to compute the model gradient using automatic differentiation. This requires two steps:

1. A forward propagation step, where the loss is computed on the current batch, and the intermediate layer activations are stored in `tf.GradientTape()`,

2. a backward propagation step, where the activations are used to compute the gradients via the chain rule.

An example code for computing the gradient of a batch of the data is provided.

Question 5.1
Using the helper code, implement stochastic gradient descent.

Question 5.2
Run it for one pass over the data (also called one epoch) and with a step size that satisfies the conditions required by theory.

Plot the objective value as a function of the iterations. Compare different step size choices. What do you observe?

Hint: Do not compute the objective value at each iteration, only from time to time (e.g., every 1000 iterations). This is sufficient for plotting purposes and will save considerable computing time.

6 Empirical risk minimization

When we want to put more energy on the database we have, we can run the algorithm for more than one epoch. In this case, we consider that we are solving the training problem on a finite sample and we try to minimize the empirical risk.

Question 6.1
Run stochastic gradient descent for 5 epochs and compare.

A commonly used technique in practice is **minibatching**: instead of computing the stochastic gradient on one data sample, we average it on a randomly selected batch of size B.

$$\begin{aligned} &\text{Select a batch } b_{k+1} \text{ of samples of size } B \\ &\theta_{k+1} = \theta_k - \frac{\gamma_k}{B} \sum_{i \in b_{k+1}} \nabla \ell(\Phi_{\theta_k}(x_i), y_i^{\text{true}}) \end{aligned}$$

Question 6.2
According to you, what is the advantage of such a scheme?

Question 6.3
Implement mini-batch stochastic gradient descent and compare.

7 Evaluation of the model

Question 7.1
Evaluate the accuracy of the models on the training set and on the test set. Compare the different methods.