MACHINE LEARNING FROM DATA

Lab Session 7 – Neural Networks

1. Goal

The goal of this session is to

- Learn how to train a Neural Network
- Use a Neural Network to solve the MNIST classification problem

2. Instructions

- Download and uncompress the file Mlearn_Lab7.zip
- Answer the questions in the document Mlearn_Lab7_report_surname.pdf

3. Previous study

Read the slides corresponding to lecture 4.3: Neural Networks.

In the first part of the session, you will play with toy examples, using synthetic datasets with two classes and two features per class, to analyze the performance of a perceptron and a simple neural network.

In the second part, you will solve a multi-class classification problem using MNIST dataset.

4. Toy examples. Perceptron and MLP

Open Colab Notebook Mlearn_lab7.ipynb.
Read the notebook identifying the different sections in the code

Q1. Which are the default parameters used by the Perceptron class? (check scikit-learn documentation). Compare the performance of the Perceptron on the two toy examples (linearly and non-linearly separable). Compare the performance of the Perceptron when using the original 2D features and features augmented by interaction

Q2. Compare the performance of the Perceptron with 3D features (with interaction) and the Multi-Layer perceptron with 2D features (you can try to improve the performance by varying some hyperparameters).

5. MNIST dataset

5.1. Characteristics of the dataset

MNIST ("Modified National Institute of Standards and Technology") is a popular dataset in computer vision. Since its release in 1999, this classic dataset of handwritten images has served as the basis for benchmarking classification algorithms. As new machine learning techniques emerge, MNIST remains a reliable resource for researchers and learners alike. The goal is to correctly identify digits from a dataset of tens of thousands of handwritten images.

The dataset contains gray-scale images of hand-drawn digits, from zero through nine. There are 70.000 samples from the ten classes (c=10) (60.000 in the training set and 10.000 in the test set).

Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels. Each pixel has a single value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning brighter. This pixel-value is an integer between 0 and 255, inclusive.

5.2. Classification using a NN

You will first use a modest MLP with one hidden layer and 10 units in the hidden layer. This classifier will be trained using Stochastic Gradient Descent (SGD) without momentum. All the other parameters are set to their default values.

Train the classifier, classify the train and test set and compute the classification report and confusion matrices for the training and test sets.

Q3: Copy the global accuracy and the confusion matrix for the training and test set and analyze the results.

Next, you will compare two of the learning strategies or solvers provided by the scikit-learn library combining them with different schedules for the learning rate.
The solvers are: SGD, and Adam. SGD can be used with momentum or with nesterov_momentum
The learning rate schedulers for the weights update can be 'constant' (a constant learning rate given by the parameter 'learning_rate_init') , 'invscaling' (which gradually decreases the learning rate at each time step t using and inverse scaling exponent of power_t) and 'adaptive' (see documentation https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html#sklearn.neural_network.MLPClassifier)

Edit the notebook
Add the code to
- Train MLP with **at least** the following combinations of learning strategy, use of momentum or nesterov_momentum and learning rate schedule:
  1. SGD, 'constant'
  2. SGD with momentum, 'constant'
  3. SGD with nesterov, 'constant'
  4. SGD, 'invscaling'
  5- SGD with momentum, 'invscaling'
  6. SGD with nesterov, 'invscaling'
  7. Adam
- Show the final training set scores and loss values.
- Plot all the loss curves in a single figure

Q4. Analyze and compare the methods based on the loss curves.

Finally, you will try to optimize some of the MLP hyperparameters.

Edit the code
Define a set of values for the hyperparemeters, train and evaluate the performance on a validation set usign GridSearchCV using ShuffleSplit with 1 split.

Q5. Analyze the results provided by `grid_search.cv_results_`
Mention if you find significant differences in performances for some of the hyperparameters.

Q6. Classify the training and test sets with the best hyperparameters. Compute the classification reports, accuracy, error and confusion matrices for the training and test sets. Discuss the results.