CS3317: Artificial Intelligence

Shanghai Jiao Tong University - 2022 Fall

Project 2: MNIST Classification

Introduction

MNIST (Mixed National Institute of Standards and Technology database) is a large handwritten digit dataset with a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28×28 pixels image. The dataset contains handwritten digits from 0-9.

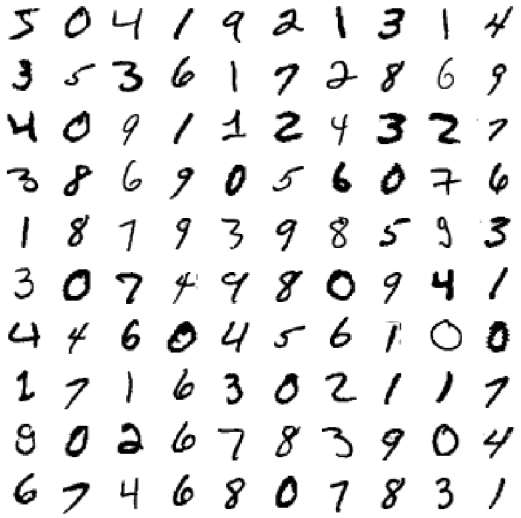

Fig 1: Some examples of MNIST.

In this project, we will classify the handwritten digits with SVM and neural networks. We do not limit the implementation environment to this project, but it might be more convenient to use python. If you are not familiar with python or lack some experience in using it, you can refer to the *guidelines.docx*. We provide some links that may be helpful. If you use python to do this project, you may need to install two python packages, sklearn and pytorch. You can install sklearn through your command line by entering "pip install -U scikit-learn" . Pytorch is a popular deep-learning framework. You can refer to the following links for installation and usage.

- (Tutorial) https://github.com/ShusenTang/Dive-into-DL-PyTorch
- (Build a network) https://zhuanlan.zhihu.com/p/99318332

If your computer does not support the GPU version, the CPU version is also enough to finish this project.

Requirements

1. Download the MNIST dataset and divide the dataset into the training set, validation set, and test set. You can download the dataset by searching on the internet or using the dataset provided in some python package, such as sklearn or Pytorch. You can also get the dataset through *get_data.py* provided in this project.

2. Use SVM to classify the MNIST examples. In class, we learn that SVM can be used for binary classification. But the MNIST dataset has ten classes, how do we use SVM to handle a classification task with more than 2 classes? There are two strategies, "One vs. Rest" and "One vs. One".

   "One vs Rest", abbreviated as "ovr", is to change an n-classification problem to n binary classification problems. We treat one class as positive and the rest classes as negative, then train a classifier for this class. Repeat this procedure for n times, we can get n classifiers. In testing stage, for a new sample, we input it into the n classifiers. The class which has the highest predicted score is the label of the new sample.

   "One vs One", abbreviated as "ovo", is to train classifiers for all possible class pairs. In an n-classification task, we have C_n^2 different class pairs. We train C_n^2 binary classifiers for all pairs. The predicted value of a new sample can be gained by a majority voting of those classifiers.

   "ovr" trains fewer classifiers, but its performance is influenced by the data imbalance. "ovr" is usually more accurate, but it takes a large amount of computation. For a convenient implementation, you can use sklearn.svm.SVC specifying the "multi_class" argument as "ovr" or "ovo". This project does not require you to try both strategies. You can choose one to finish. Optionally, you could compare the performance of "ovr" and "ovo".

   - Notice that there are many examples, and it might take a lot of time to train an SVM. You can select 1000 examples from each class (10000 examples in total) as the training set.
   - You should try different hyperparameters of the SVM, such as C and the types of the SVM kernel, and compare the results.

3. Use a deep neural network to classify the MNIST examples. The MNIST dataset contains 60000 examples as training data. You should re-split the training data into the training

set and validation set (50000 examples for training and 10000 examples for validating). It is recommended to use the data from Pytorch package. You can use it through torchvision.datasets.MNIST.

- Build an artificial neural network (ANN) to do the classification task. You can build a multilayer perceptron (a fully connected neural network with the input layer, hidden layer, and output layers, abbreviated as MLP) to do this task.
- Train your model on the training set and choose the best hyperparameters (epoch, learning rate, weight decay, the types of the optimizer, etc.) by validating the model on the validation set.
- Choose the hyperparameters with the best performance on the validation set and retrain your model on all the training data (training set+validation set, 60000 examples in total). Then test the model on the test set.

Submission

1. Please submit a file in ZIP or RAR format containing your **report** (Chinese or English) and **codes** on Canvas before **2022-12-18**, **23** : **59**. Name it as $StudentID\_Name$ (eg. 5180xxxxxxxx_ 张三).
2. Describe your implementation of SVM and the ANN, including the setting of hyperparameters, and report the experimental results.
3. Do not cheat. Using existing packages is allowed.