Image Classification

1 Introduction

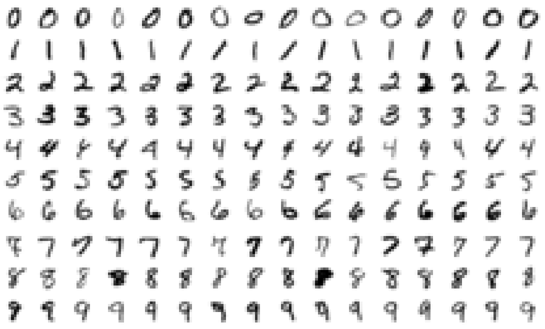

Figure 1: archive.ics.uci.edu

To cap off our time in this semester and before tackling the project presentations in our next session, let us dive into the world of machine learning together! We will focus on you getting to know the overall process of training a model to perform a specific task from scratch, which is why the tasks at hand will have more of a walk-through character than tasks in previous assignments. We won't focus too much on the theory behind why stuff works the way it does, but you are encouraged to try to find out yourself or ask your lecturer.

Applying Machine Learning to Images using scikit-learn

This lab session focuses on the application of supervised learning to images using scikit-learn, also called sklearn. More specifically, you are going to design an optical recognition system that recognizes hand written digits trained on the popular MNIST data set and puts out a numeric value based on the image it is fed.
Performing preprocessing, splitting the data into a train-, and test-set, training a model, making predictions on the test-set and finally evaluating the model's performance are the steps you are going to perform during our session.

Getting Started

To get started, please make sure to install scikit-learn into the environment you are about to use during this session. Once you have done that, we'll start by loading the dataset together and go from there!

simon.schindler@fh-salzburg.ac.at
maximilian.schirl@fh-salzburg.ac.at
martin.uray@fh-salzburg.ac.at

2 Lab Assignments

Start by using the following code to load the data:

```python
from sklearn.datasets import load_digits
# we load the data directly from the datasets module of sklearn
digits = load_digits()
```

Then, answer the following questions:

- What object type is `digits`?
- You will find out that `digits` is container-like (more specifically dict-like). Find out how to access its keys that are used to store data within the container.
- Print the description of the data set and read through it.

Plot an Image and its Label from the Data Set ✓

Now that you know how to access `images` and `target` in `digits`, access a single image and then plot this image using matplotlib's imshow(). Then, set the image `target` (may also be called label in related literature, denotes the true numeric value of handwritten digit in image in this case) as the plot title by accessing the `target` corresponding to the image you chose.

Preprocessing and Splitting the Data into a Train-, and Test-Set✓

As this dataset is wildly popular in the optical recognition domain and highly preprocessed already, preprocessing can be kept simple. The only thing you need to know is, that the simple classifier you are about to apply to the image data is not able to process two-dimensional data like that of an image. This is why you need to reshape the images contained in `digits.images` to be one-dimensional. Store those 1D representations in the variable `data`.

Then, to correctly apply supervised learning, you need to make sure that the `data` and labels your model is trained on is strictly kept separate from the data your model's performance is tested on. Conveniently, sklearn provides the handy function train_test_split() to do this for you. Apply the split while making sure that the data and corresponding targets are correctly matched (hint: this can be done by passing `data` and `digits.target` to train_test_split()). Store the train-, and test-set containing 1D-representations of the images to variables `X_train` and `X_test`, store the corresponding labels to `y_train` and `y_test`.

simon.schindler@fh-salzburg.ac.at
maximilian.schirl@fh-salzburg.ac.at
martin.uray@fh-salzburg.ac.at

Training a Support Vector Machine ✓

Next, you are going to set up the machine learning model that is used in your optical recognition system. More specifically, focus on the LinearSVC classifier provided by sklearn. If you want to learn more about SVMs and their functionality, you may start with this article on medium.com. Then, work through the following list to get your system up and running:

- **Instantiate the LinearSVC:** Store an instance of `LinearSVC` in variable `svm`. You do not need to set specific parameters to certain values, keeping all settings default is fine.
- **Start the training using the train-set:** this process is typically called *fitting* and is therefore usually started by calling the `fit()` method of a classifier. Consult the docs of `LinearSVC` to learn how to do this correctly. You may get some warnings here because we did not concern ourselves with tuning the `LinearSVC's` parameters. Ignore these warnings for now.

Evaluating the Support Vector Machine's Performance ✓

After training, we want to see how well our model is able to perform its task (in our case: recognizing handwritten digits from image data) on new data. This is where the test dataset comes into play!

- **Perform prediction on the test-set:** We ask our model to predict a label for each image in the test dataset `X_test`. This can be done by passing `X_test` to the `svm's` `predict()` method. Store the return value of `predict()` in the variable `y_pred`.
- **Evaluate the performance using typical performance measures:** Want to know how many of the handwritten digits were identified correctly? To get the answer, calculate and print out the accuracy of our result using the function `accuracy_score()`. You can find this function in the `metrics` module of sklearn.
- **Plot the confusion matrix:** Another way to evaluate your classifier's performance is the plotting of a confusion matrix. Sklearn provides the method ConfusionMatrixDisplay.from_predictions() to perform this task. Make sure to pass the labels for the test set as well as the corresponding predictions to this method! Also, please don't forget to set a title for your plot.

Recall and Summarize ✓

You reached the end of our short guided tutorial into the world of supervised learning. In your own words, give an overview of the tasks you completed up to now in the cell below and include:

- a description of the task you were asked to perform
- the idea behind the classifier (`LinearSVC`) you used (no details about the functionality are needed)
- an interpretation of the results of your evaluation

3 Homework

At the beginning of the next lab session, you will get a chance to indicate which of the **lab assignments** (✓) you completed and some of you will be asked to present their solution in class. Each student will be asked to present their solutions **at least twice** over the course of the semester. Other than that, no additional upload is needed. Please see the Course Syllabus for details on how this marking of assignments and their presentation affects your grade.