

D7041E “Applied artificial intelligence”

(Please report any found inconsistencies to Evgeny as soon as you find them)

LAB 1: Fundamentals of Machine Learning

1. Introduction

In this lab we will work with data-driven classification problem. The task is to assign an input data one label from a finite set of categories. The challenge here is that there is no model describing each category available in advance. That is we cannot hard-code it in an algorithm, instead we need to learn this model. In this lab you will work with two instance of this problem – the image classification problem and classification of flower types. In particular we will classify hand-written digits from the publicly available dataset MNIST and feature description from IRIS dataset.

2. Fundamentals of data pre-processing and basic skills in Jupyter

To master basic data pre-processing skills collect your own dataset of images (take at least 5 photos of 5 categories of objects)

Task 1.1 Import your dataset into Jupyter environment:

1. Make an $(n, x, y, 3)$ array containing all the samples of your dataset. Here n is the number of samples, x and y is the resolution of your photos and 3 is the rgb components.
2. Write a function for plotting a sample `PlotSample`, which takes the index of a sample as an argument and plots it using the functionality of the **matplotlib** library.

3. For a sample of your choice determine (and plot) the distribution of values across all features.
 - What are the minimum, maximum, mean and standard deviation values?

3. Nearest Neighbour classifier: skills of working with datasets and hyperparameters

The first approach you will test has little to do with learning, however we will use it for mastering the skills of working with datasets as well as a baseline for benchmarking performances of more advanced approaches.

MNIST data set:

One commonly used toy image classification dataset is the MNIST dataset. This dataset consists of 60,000 tiny images that are 28 pixels high and wide. Each image is labeled with one of 10 classes (“0”, “1”, “2”, “3”, “4”, “5”, “6”, “7”, “8”, “9”). These 60,000 images are partitioned into a training set of 50,000 images and a test set of 10,000 images.

Task 2.1

You are provided with an implementation of a special case of the K-Nearest Neighbors classifier (i.e. 1-NN) using L1 norm. The 1-NN classifier takes a test image, compares it to every single one of the training images, and predicts the label of the closest training image.

1. Run the provided code for 1-NN classifier. What is the accuracy of the method?
2. Modify the code of the NearestNeighbor example so that it uses L2 norm for classification. How the classification accuracy has changed now?
3. The accuracy of the NN classifier that you have observed in the previous two tasks is VERY low. This is due to the bug in the code – find it and fix it. HINT: check the distances computed by Python and those computed by hand for several pairs of images.
4. Extend the code of the NearestNeighbor example with necessary methods, which implement k-NN classifier. That is instead of finding the single closest image in

the training set, you will find the top k closest images, and have them vote on the label of the test image.

Task 2.2. Hyperparameters, cross-validation

The k -NN classifier requires a setting for k as well as there is an option of choosing one or another distance function (e.g. L1 or L2 norm). These choices are examples of so called **hyperparameters**, which needs to be optimized for best classification accuracy.

In machine learning one should avoid adjusting these parameters based on the test data! This is rather obvious since in this case your algorithm will be optimized for the particular data set. A typical approach for choosing test parameters is to have yet other data set for validation purposes. In some cases, when the dataset is small, a technique called cross-validation is used. It is illustrated in the figure below.

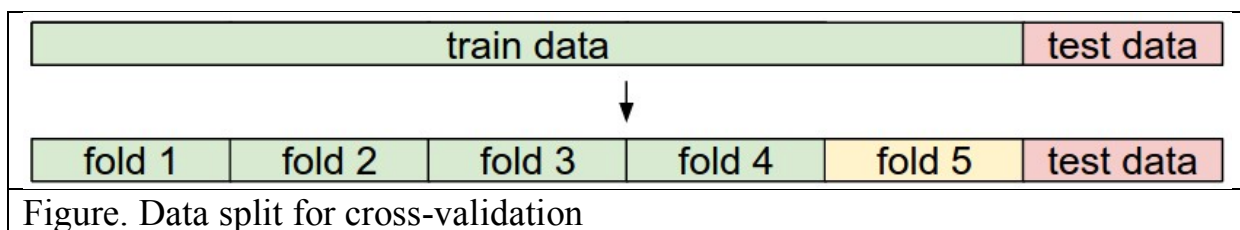


Figure. Data split for cross-validation

The training set is split into folds. In the figure above 5 folds are displayed. One fold for example fifth (shown by yellow color) is chosen for validation while other four for training. In cross-validation one iterates the validation fold when choosing the optimal values for hyper parameters. Typical number of folds you can see in practice would be 3-fold, 5-fold or 10-fold cross-validation.

In the very end once the model is trained and the best values of hyperparameters were determined, the model is evaluated a single time on the test data (red).

1. Extend the code of the NearestNeighbor example with necessary methods to find the best value of k using 3-fold cross-validation method. Use L2 norm as the distance function.
2. What is the classification accuracy on the test data for the best value of k ?

3. Support Vector Machines

Support Vector Machines is one of the most widely used Machine Learning techniques for solving classification problems. In this part of the lab you will use implementation of SVM from scikit-learn Python library to understand the fundamentals of using SVM.

Task 3.1 Load Iris dataset, split the dataset into the training and the test parts in proportion 80:20. Train SVM with linear, polynomial and RBF kernels for multiclass classification. In this task your solution could be inspired by the following blog:

<https://towardsdatascience.com/support-vector-machines-svm-clearly-explained-a-python-tutorial-for-classification-problems-29c539f3ad8>

1. Construct confusion matrices for one-vs-one and one-vs-rest training approaches and each kernel type;
2. Using which kernel the best accuracy and F1 score is achieved?
3. Extract the support vectors for each class in one-vs-rest training case.
4. Plot the decision boundary for features 2 vs. 3 and 3 vs. 4.

Congrats, you have just become familiar with fundamentals of the cutting edge learning techniques for data-driven classification! Well done!