MACHINE LEARNING

SHEET 2

02.12.2021

10:15 A.M. UNTIL 11:45 A.M. VIA ZOOM

Please prepare the exercises in order to present them in the meeting. Fill-in the exercises you solved in the questionnaire in StudIP until 01.12.2021 4:00 p.m.
Join the meeting via this Zoom link:
https://uni-trier.zoom.us/j/84882443567?pwd=S0xucnRjY2ZEMHNOalVkdllrVDRRZz09

TASK 1: LINEAR SUPPORT VECTOR MACHINES

Support Vector Machines (SVM) are a powerful machine learning toolkit for classification and regression problems. We will explore their capabilities using the OpenML dataset about forged banknotes (https://www.openml.org/d/1462).

a) Import the dataset from OpenML by using the respective scikit-learn methods.
b) Visualize the features by converting them to a two-dimensional format with the help of t-SNE. Does this dataset appear to be suitable for linear SVM classification?
c) Split the data into a training set and a test set with a test ratio of 0.2.
d) Setup a scikit-learn pipeline that does the following:
    1. Scale the features by using one of the feature scalers from scikit-learn. Which one did you choose and why?
    2. Train a linear SVM classifier with the hyperparameter *C=1*, *max_iter=10000*, and hinge loss.
e) What is the performance of this classifier on the test set regarding accuracy and the confusion matrix? Also compare it to the performance on the training set. Is overfitting or underfitting a problem?

TASK 2: NON-LINEAR SUPPORT VECTOR MACHINES

In the first exercise, we used a linear SVM classifier. We will now look at non-linear SVMs that are capable of fitting more complex data. We use the breast cancer dataset (https://scikit-learn.org/stable/datasets/index.html#breast-cancer-dataset) for this.

a) Fetch the dataset. Split it up into a training set and a test set with a test ratio of 0.2.
b) Build three scikit-learn pipelines (see below) that do the following:
    1. Scale the features by using one of the feature scalers from scikit-learn. Which one did you choose and why?
    2. Train an SVM classifier on the training data.

    The three pipelines only vary in the way they classify:

    Pipeline 1: A linear SVM classifier with *C=1*, *max_iter=1000*, and a hinge loss that acts as a baseline (similar to the one from Exercise 1).

    Pipeline 2: A linear SVM classifier with the same parameter settings as Pipeline 1. In contrast to the first one, this classifier should work on polynomial features up to a degree of 3.

DEPARTMENT OF BUSINESS INFORMATION SYSTEMS II
PROF. DR. RALPH BERGMANN
UNIVERSITY OF TRIER

Pipeline 3: An SVM classifier with a polynomial kernel (degree of 3), *C=1*, and *coef0=1*.

c) Compare the approaches in terms of accuracy on the test set. Explain the results.
d) Try to improve the results by performing a grid search with the best performing classifier. Keep the configuration that was used before and search for better parameter settings of *C* (range between 0.1 and 100) and *max_iter* (range between 100 and 10000). Use appropriate steps in the grid.

TASK 3: INSTANCE-BASED LEARNING

Instance-based learning belongs to the class of lazy learners where an example is classified according to other examples from the dataset at runtime. In this exercise, we will program a similarity-based k-nearest neighbor (KNN) classification with numpy.

a) Create an exemplary classification dataset with the respective scikit-learn method (https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_classification.html). It should have ten examples, five attributes, and two target classes.
b) Now compute the distance values between all attributes of all examples. Thereby, the distances are computed between pairs of examples where each attribute has its own distance value. The distance $d(v_1, v_2)$ of two attribute values belonging to the same attribute $v_1, v_2 \in A$ is computed according to the following formula:

$$d(v_1, v_2) = \frac{|v_1 - v_2|}{\max(A) - \min(A)}$$

In the formula, $\min(A)$ and $\max(A)$ are the minimum and maximum values, respectively, of all examples regarding this attribute.
c) Apply the following two similarity measures on different attributes of the distance matrix computed before:
   - Use the exponential similarity measure, i.e., $sim_i(v_1, v_2) = e^{d(v_1, v_2) \cdot (-0.1)}$, for all values of the first three attributes ($i = 1 \dots 3$).
   - Use the following binary measure for all values of the last two attributes ($i = 4,5$):

$$sim_i(v_1, v_2) = \begin{cases} 1, & d(v_1, v_2) < 0.3 \\ 0, & else. \end{cases}$$

d) Finish the KNN computations by averaging the values for the attribute similarities of all pairs of examples (resulting in a single pairwise similarity between the examples) and determining the k-nearest neighbors for the example number 3. Thereby, use $k = 3$.

TASK 4: DECISION TREES

Decision Trees are another popular class of ML algorithms. We will use the dataset about forged banknotes that was also used in Exercise 1.

a) Fetch the dataset from OpenML and split-up the data into a training and a test set (test ratio of 0.2).
b) Train a Decision Tree Classifier with *max_depth=2* and visualize the resulting tree. Explain the visualization and how this tree can be used for predictions.
c) Use a grid search to find the best parameter settings for the classifier. Search in depth values between 2 and 15, test both criteria (*entropy* and *gini*) for measuring the split quality, and test both criteria to select the split (*best* and *random*).
d) Decision Trees can also be used for regression problems. For the sake of simplicity, we simply use the original numeric target values of the dataset as numeric regression targets. Apply some gaussian noise (mean of 0 and standard deviation of 0.2) to make the target values more realistic.
e) Repeat the steps from c) to find a well performing predictor. Rate its quality. Please note that you have to search through different parameter values than in c) since the regressor expects different values.