Assignment 4: Classifier Comparison

Pascal Schlachter & Mario Döbler

Institute for Signal Processing and System Theory
University of Stuttgart

1 Introduction

Machine learning can be divided into two categories, namely classification and regression. Classification is the task to categorize data into a predefined discrete number of classes while regression aims to estimate a continuous signal or parameter. In this assignment, we take a look at different classification methods and compare them. To do so, we first create some artificial datasets. Second, we evaluate the properties and performances of different classifiers, just like in the examples known from the lecture.

2 Datasets

To visualize the properties of different classifiers, we first create some artificial two-dimensional datasets. Here, we use binary datasets, meaning they contain two different classes. All datasets are illustrated in Figure 1.

2.1 Two moons

The first dataset consists of two moons. Thereby, each moon corresponds to one class. This dataset is visualized in Figure 1a.

The first moon is generated using the function

$$x_2 = -0.3 \cdot (x_1 - 2)^2 + 4 + 0.7n \ , \tag{1}$$

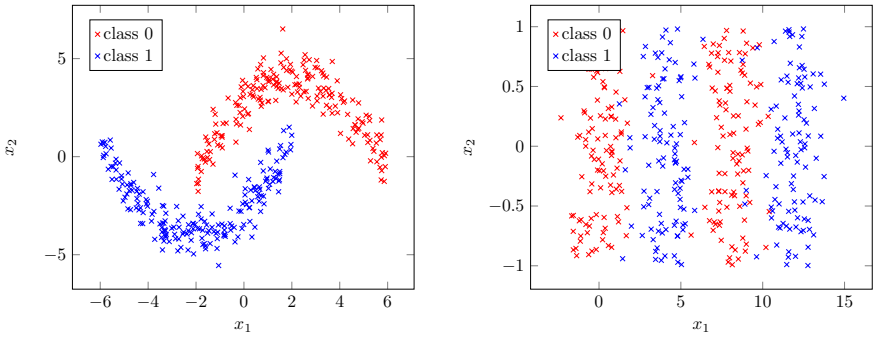
while the second moon is generated using

$$x_2 = 0.3 \cdot (x_1 + 2)^2 - 4 + 0.7n \ . \tag{2}$$

Thereby, $n \sim \mathcal{N}(0, 1)$ is a standard normal distributed random variable. The data generation is done by drawing values for x_1 using a uniform distribution in the range $[-2, 6]$ for the first moon and $[-6, 2]$ for the second moon.

2.2 Four parallel distributions

The second dataset consists of 4 similar distributions arranged in parallel. Thereby, two distributions are assigned to each class. This assignment is done such that the classes are alternating along x_1. This dataset is shown in Figure



(a) Two moons

(b) Four parallel distributions

(c) Four Gaussian distributions (XOR)

(d) Circular distributions

Figure 1: Visualization of the datasets. Here, each class consists of 200 samples.

1b.

Each of these distributions is generated by using a Gaussian distribution for x_1 and a uniform distribution for x_2. The uniform distribution ranges between -1 and 1. All Gaussian distributions have unit variance $\sigma^2 = 1$. To get the parallel arrangement, the means of the Gaussian distributions are chosen accordingly. Concretely, they are selected as 0, 4, 8 and 12, respectively.

2.3 Four Gaussian distributions (XOR)

This dataset consists of four Gaussian distributions, two for each class. Their means are arranged in a square. Thereby, the class assignment is done such that the diagonally opposite distributions belong to the same class.

All distributions have unit variance $\boldsymbol{C} = I$. The distributions of the first class have the means $(2, 2)$ and $(-2, -2)$, respectively, while the distributions of the second class have the means $(-2, 2)$ and $(2, -2)$.

2.4 Circular distributions

Finally, for the last dataset, one class builds a cluster while the other class distributes circularly around that cluster. This dataset is visualized in Figure 1d.

In python, this dataset can be generated using the function `make_circles` of the `sklearn.datasets` library (see `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_circles.html`). Here, we choose the scale factor between the inner and outer circle to 0, meaning all samples of the second class are mapped to the center point of the circle. Furthermore, we scale all data samples by factor 4 and corrupt them with standard normal distributed noise.

3 Classifiers

In this assignment, we focus on the nearest mean classifier, the k-nearest neighbor classifier, and the Gaussian mixture model classifier.

3.1 Nearest mean classifier

The nearest mean classifier is a simple multiclass classifier using template matching. For training, it estimates the class center for each class by using

$$\hat{\underline{\mu}}_j = \frac{1}{N_j} \sum_{y_n = \omega_j} \underline{x}_n \qquad 1 \leq j \leq c \; . \tag{3}$$

Thereby, $\{(\underline{x}_1, y_1), \ldots, (\underline{x}_N, y_N)\}$ denotes the training data, N_j is the number of training data of class ω_j and c describes the number of classes (here $c = 2$).

For inference, the nearest mean classifier calculates the distances to all means for each test sample. Thereby, either the Euclidean distance or the Mahalanobis

distance can be applied. In this assignment, we use the simpler Euclidean distance. Finally, each test instance is classified according to its closest mean.

3.2 *k*-nearest neighbor classifier

Like the nearest mean classifier, the k-nearest neighbor classifier is also a multiclass classifier using template matching. The classification of each test sample is done according to the most common class among its k nearest neighbors. In order to avoid a tie, k is chosen to be an odd number. To find the nearest neighbors, the distances to all training instances have to be calculated for each test sample. Again, either the Euclidean or the Mahalanobis distance can be applied. We still decide for the Euclidean distance in this assignment.

3.3 Gaussian mixture model classifier

The Gaussian mixture model classifier is a parametric method, meaning it estimates the likelihood of each class before applying Bayesian decision theory. Concretely, it assumes that the underlying data distribution of each class can be estimated by a combination of multiple Gaussian distributions.

For training, first the model order M_j has to be chosen for each class j. This parameter gives the number of Gaussian distributions used for the estimation. Thus, if all M_j are set to 1, the Gaussian mixture model classifier is equivalent to the Gaussian classifier. Second, the means and covariance matrices of all Gaussian modes have to be estimated for all classes, respectively. For this, the expectation maximization algorithm is used.

For inference, the estimated likelihood functions are used for applying Bayesian decision theory. In this assignment, we use maximum likelihood decision.

In python, the `sklearn.mixture` library already provides an implementation of the Gaussian mixture model called `GaussianMixture` (see `https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html`).

4 Metrics

To observe the performance of the classifiers, we use the confusion matrix. It is obtained by comparing the predicted labels to the true labels. For binary problems, four cases can occur, namely true negatives, false negatives, false positives or true positives. Their respective number of occurrences is counted and summed up in a matrix, the so-called confusion matrix.

In python, the confusion matrix can be calculated by using the function `confusion_matrix` of the `sklearn.metrics` library (see `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html`). However, note that the confusion matrix there is defined differently than in the lecture. In this assignment, we want to use the definition of the lecture, so we have to transpose the matrix which the function outputs.

5 Tasks

5.1 Theory

For the following tasks, we regard a binary dataset consisting of 20,000 samples each for training and test. Assume that both classes have the same amount of samples for both the training and test set.

5.1.1 Nearest mean classifier

1. What is the computational complexity of the training (how many summations and multiplications are necessary)?

2. How many distance calculations are necessary for the classification of each test sample? How many distance calculations are necessary for the classification of the whole test set?

5.1.2 k-nearest neighbor classifier

3. What is the computational complexity of the training?

4. How many distance calculations are necessary for the classification of each test sample? How many distance calculations are necessary for the classification of the whole test set?

5. Compare the computational complexity of the k-nearest neighbor classifier with that of the nearest mean classifier for training and inference, respectively. Where is a high computational complexity more costly, during training or during inference? Explain!

5.1.3 Gaussian mixture model classifier

6. For each dataset introduced in section 2, give the most suitable model order M_j for both classes ($j = 0, 1$), respectively.

7. Give the likelihood ratio test (LRT) of maximum likelihood (ML) decision. Turn it into the log-likelihood ratio test (LLRT).

5.2 Practice

1. Finish the implementation of the datasets and classifiers in `datasets.py` and `classifiers.py`. Make sure to choose an appropriate k for k-nearest neighbor and suitable model orders for the GMM classifier in `main.py`. You can find the code template in the Git repository (`https://github.tik.uni-stuttgart.de/iss/dpr-assignments-2021`).

2. For a dataset of your choice, set the size to 40,000 and the training split to 0.5.

   (a) Report how much time each classifier needs for inference. Which is the fastest, which the slowest?

(b) Calculate the true negative rate, false negative rate, false positive rate and true positive rate for each classifier from the confusion matrices. Report your results.

3. *(Optional)* Feel free to play around with different parameters to observe their influence on the results. For example, vary the number k of considered nearest neighbors, the model orders M_j of the Gaussian mixture model, or the size of the dataset.