Technische Universität München
Fakultät für Informatik
Prof. Nassir Navab

SS 2016
Exercise Sheet 4
May 2nd, 2016

Exercises in Machine Learning for Medical Imaging

Exercise 1 SVMs on Toy Dataset

In this exercise, you will apply SVM classification to the 2-D points in the [`04-twomoons-dataset.zip`]. In Python, use panda e.g. [`import pandas as pd`] to read the files. In MATLAB [`csvread`]. The dataset is already split in training and testing data, where x stands for the point coordinates, and y for the labels. The general idea is to fit an SVM based on the training set, and then evaluate its performance both on the training and the test set. To this end:

- Read and visualise the dataset (In Python, `from matplotlib import pyplot as plt`),
- Create a function `svm_fit`, which takes a matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$ of samples, a vector $\mathbf{y} \in \mathbb{N}^n$ of outcomes for each sample, and a *kernel*. `svm_fit` should return the estimated predicted value and evaluate different evaluation criteria (*sensitivity, precision, specificity, Negative Predictive Value (NPV), accuracy and F1-measure*), on the training samples.
- Create a function `svm_predict` receiving the model trained above, a matrix $\mathbf{X}_t$ of samples for testing and their expected outcomes $\mathbf{y}_t$. (According to the platform and library used you may also need the matrix $\mathbf{X}$ and the kernel used during training). The function should return a vector of predicted outcomes and the different evaluation criteria on the testing samples.
- Plot with two curves the mean squared *train error* and *testing error* obtained:
  - for the `linear` and `RBF` kernel.
  - for different C values, i.e. $C \in \{0.001, 0.01, 0.1, 1, 10, 20, 50, 100, 200, 500, 1000, 10000\}$
  - for different values of the RBF kernel.
- Additionally, program a function that creates (animated) 2D plots of the points comparing the predicted vs. true values for every combination of parameters (see the example below)

In Python: Use `from sklearn import svm`. **With Matlab:** you can choose between the MATLAB embedded SVM library or by installing and downloading `libsvm`.

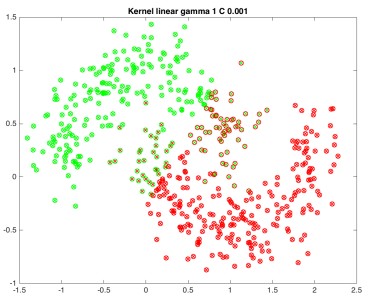

Figure 1: 2D points from the two moons test set. The colour of the circles stands for the target class, crosses show the predicted class. One such image should be created for each kernel and parameter set.

Exercise 2 Digit Recognition

This exercise focuses on the application of the well known Support Vector Machine (SVMs) to the recognition of handwritten digits using the popular MNIST dataset. The problem consists in assigning to each image a label among the 10 digits. With this goal in mind, we will follow a similar pipeline as above:

a) Read and visualise the dataset: Download the `cvs file`, containing the images and labels. Note that each row encodes the information for a single image. Each image is of size 28×28 and has been flattened to a row vector. In the training file, remove the first column of the training set file which contains the label assigned to each image. Visualise some of the images and their labels to verify that you are correctly reading the data, e.g. using openCV `import cv2`. Note that for the test set there is no access to the labels.

b) Dimensionality Reduction: apply PCA to both the training and testing images to reduce their dimensionality while preserving a given percentage (e.g. 80 %) of the variance of the dataset. In Python you may use the sklearn functionality: `from sklearn.decomposition import PCA`.

c) Crossvalidation: Based on the first part of the exercise, create a function that splits the training dataset in several (e.g. 5 folds) and uses a K-fold scheme to train and test the different splits of the data. Report the same evaluation criteria as before, but this time per fold and as well as in average. Scikit crossvalidation functionalities can be used but own implementation may give a better understanding.

d) Hyper-parameter optimisation: Use the crossvalidation function within a grid search strategy to choose adequate values for the variance preserved in PCA and for the C parameter of the SVM.

e) Validation test: Classify the images in the "test set" and visualise random images with their predicted label.

f) **Bonus 1:** You may want to extract features from the images (e.g. HOG, LBP, BRIEF) in addition to the dimensionality reduction.

g) **Bonus 2:** You can try to upload your results to the open challenge website (`https://www.kaggle.com/c/digit-recognizer`)