Exercise 1 - preliminary!!

Deadline: 27.10.2014

In this exercise, you will implement the nearest-neighbor classifier and estimate its possibilities to generalize to unseen data via cross-validation. We propose to test and evaluate your classifier on a data set of handwritten digits that is available in sklearn.

Data Description

Sklearn provides a small collection of datasets that are suitable for testing a classification algorithm (See `http://scikit-learn.org/stable/datasets/` for overview of the available datasets). For this exercise we want to stick to the problem of recognition of handwritten digits, witch is a typical application for machine learning. This specific dataset consists of 1797 small images.

1 Exploring the Data (3 points)

After following the instructions from Exercise 0 you should be able to execute

```
import vigra
import numpy
import sklearn
```

without errors. At first we want to get an overview over the digits dataset. The following lines show how to load the digits dataset from sklearn and how to extract the necessary data:

```
from sklearn.datasets import load_digits

digits = load_digits()

print digits.keys()

data          = digits["data"]
images        = digits["images"]
target        = digits["target"]
target_names  = digits["target_names"]

import numpy as np
print np.dtype(data)
```

Note that data is a vectorized version of the images. Explore the data further by testing its dimensionality (the function numpy.shape() might come in handy). Visualize one image of a '3'. One possibility to visualize an image is the imshow function from matplotlib.pyplot:

```
import numpy as np
import matplotlib.pyplot as plt

img = ...

assert 2 == np.size(np.shape(img))

plt.figure()
plt.gray()
plt.imshow(img, interpolation="nearest")
plt.show()
```

2 Nearest Neighbor Classifier

Overall Task: Write a nearest neighbor classifier that distinguishes the digit '3' from all other digits.

2.1 Splitting Data into a training- and a test set (1 point)

Sklearn provides a convenient function that separates your data into a training- and a test set.

```
from sklearn import cross_validation

X_all = data
y_all = target

X_train, X_test, y_train, y_test =\
    cross_validation.train_test_split(digits.data, digits.target,
                                      test_size=0.4, random_state=0)
```

2.2 Distance function computation using loops (2 points)

As a first step, write a python function dist_loop(training, test) which computes the Euclidian distance between all digits in the training and test set (in the feature space) using loops and measure the run time (suggested commands: time.time() from the package time). The input should be the n1 x d and n2 x d training and test matrices with d pixels per image and n1 respectively n2 samples in the training resp. test set. The output should be a n1 x n2 distance matrix. For the calculation of the Euclidian distance you might want to use numpu.square(), numpy.sum() and numpy.sqrt().

2.3 Distance function computation using vectorization (3 points)

Since loops can turn out to be rather slow in python, and since we may be in need of efficient code later on, write a second python function for computing the distance function which relies on vectorization and does not have a "for"loop, dist_vec(training, test) and compare the run time. The results of 2.2 and 2.3 shapuld be identical. numpy.tile() and numpy.tensordot() can help you with this task.

2.4 Write a nearest neighbor classifier (3 points)

Using the distance matrix from 2.2 or 2.3, now implement a nearest neighbor classifier and use it to distinct the digit '1' from the digit '3'. Therefore zou will need to restrict the distance Matrix to images of ones and threes. You can use Your function from 2.2 or 2.3 with only images from ones and threes. Compute the correct classification rate as the number of digits that were predicted correctly as three or one. Repeat the same experiment for ones and sevens. (suggested commands: numpy.argsort() or numpy.sort())

2.5 k-nearest neighbors (3 points)

Extend your classifier to take the opineon of more than one neighbor into account (take the majority vote from the k-nearest neighbours). Try varying the value for k (try the values 1, 3, 5, 9, 17 and 33) and compute the correct classification rate for the classification of ones against sevens. Describe the dependency of the classification performance on k. Explain your observation.

3 Cross-validation

In 2.4 we measured the performance of the nearest neighbor classifier on a predefined test set. To be able to do this we had to decrease the size of the training set and therefore may have missed some relevant training samples. Another way to estimate whether the trained classifier is able to generalize to unseen data is cross-validation. In this exercise we want you to implement a 5-fold cros validation scheme that can be test the nearest neighbor classifier from 2.4. Alternatively you can use the sklearn.neighbors.KNeighborsClassifier() that you train with the fit() function and from which you can get predictions via the predict() function (more details on `http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html`).

3.1 Cross-validation of nearest neighbor (4 points)

Write a function that is able to split the given annotated data in n parts of equal size. Use each of the k subparts one time as test set and the remaining parts as training set. Return the mean classification rate as well as the variance. Cross-validate the nearest neighbor classifier on the digits dataset for n=2,5,10.

4 Regulations

Please hand in the python code, figures and explanations (describing clearly which belongs to which). Non-trivial sections of your code should be explained with short comments, and variables should have self-explanatory names. Plots should have informative axis labels, legends and captions. Please enclose all results into a single .pdf document and hand in the .py files that created the results. Please email the solutions to `niko.krasowski@hci.iwr.uni-heidelberg.de` before the deadline. You may hand in the exercises in teams of maximally three people, which must be clearly named on the solution sheet (one email is sufficient). Discussions between different teams about the exercises are encouraged, but the code must not be copied verbatim (the same holds for any implementations which may be available on the WWW). Please respect particularly this rule, otherwise we cannot give you a passing grade. Solutions are due by email at the beginning of the next exercise. For each exercise there will be maximally 20 points assigned. If you have 50% or more points in the end of the semester you will allowed to take the exam.