Introduction to Data Science 2018
Assignment 2

François Lauze, Thomas Hamelryck

Your solution to Assignment 2 must be uploaded to Absalon no later than Sunday March 4th, 2018, 12:00 (***i.e., by noon!***)

Guidelines for the assigment:

- **The assignments in IDS must be completed and written individually.** This means that your code and report must be written completely by yourself.
- Upload your report as a single PDF file (no Word) named `firstname.lastname.pdf`.
- Upload your Python code. If the code is in several files, upload them in a `.zip` archive.

Each question/exercise is worth 10 points, the total number of points for this assignment is 40.

Classification with nearest neighbors

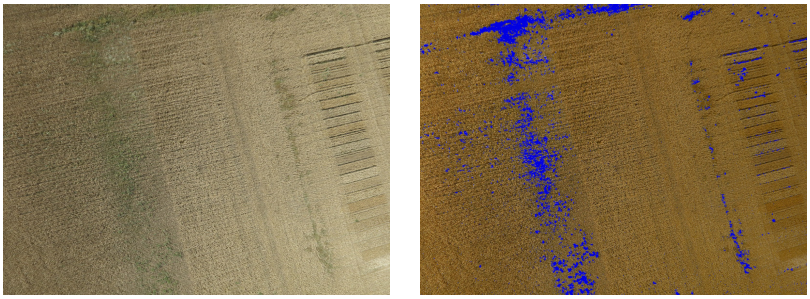
In the following, we will work with three fundamental data analysis techniques. We will perform classification with the *nearest neighbor classifier*, a non-linear, non-parametric method for classification. Then we will apply *cross-validation* for model selection and standard *data normalization* for preprocessing.

The data

The data for the following tasks are taken from a research project financed by Miljøstyrelsen and involving researchers from DIKU and PLEN/KU. Selected results from the project are described by Rasmussen et al. [2016] and Olsen et al. [2017]. While the problem setting is inspired by Olsen et al. [2017], the data were processed differently.

Introduction to the problem (not relevant for solving the exercises). Pesticide regulations and a relatively new EU directive on integrated pest management create strong incentives to limit herbicide applications. In Denmark, several pesticide action plans have been launched since the late 1980s with the aim to reduce herbicide use. One way to reduce the herbicide use is to apply site-specific weed management, which is an option when weeds are located in patches, rather than spread uniformly over the field. Site-specific weed management can effectively reduce herbicide use, since herbicides are only applied to parts of the field. This requires reliable remote sensing and sprayers with individually controllable boom sections or a series of controllable nozzles that enable spatially variable applications of herbicides. Preliminary analysis [Rasmussen et al., 2016] indicates that the amount of herbicide use for pre-harvest thistle (Cirsium arvense) control with glyphosate can be reduced by at least 60 % and that a reduction of 80 % is within reach. See Figure 1 for an example classification. The problem is to generate reliable and cost-effective maps of the weed patches. One approach is to use user-friendly drones equipped with RGB cameras as the basis for image analysis and mapping.

The use of drones as acquisition platform has the advantage of being cheap, hence allowing the farmers to invest in the technology. Also, images of sufficiently high resolution may be obtained from an altitude allowing a complete coverage of a normal sized Danish field in one flight.



a) b)

Figure 1: Example from another approach. a) An original image. b) Initial pixel based detection.

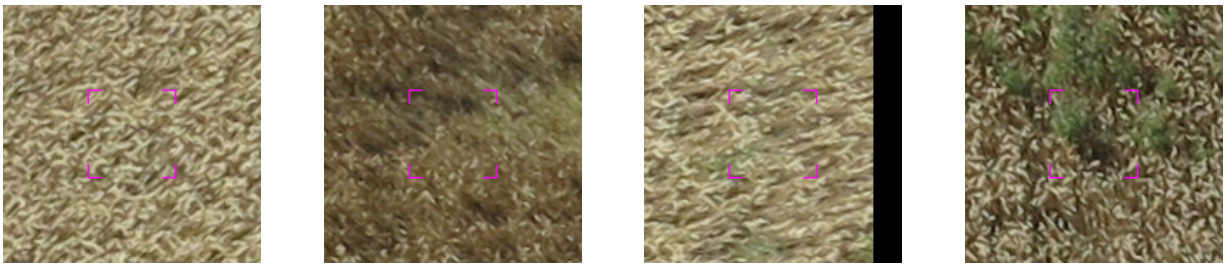


Figure 2: The two images on the left are classified as crop. The two images on the right are classified as weed. The classification of the middle two patches is debatable. The central area used for performance evaluation is indicated by the small magenta markers.

Your data is taken from a number of images of wheat fields taken by a drone carrying a 3K by 4K Canon Powershot camera. The flying height was 30 meters. A number of image patches, all showing a field area of 3×3 meters were extracted. Approximately half of the patches showed crop, the remaining thistles. For each patch only the central 1×1 meter sub-patch is used for performance measurement. The full patch was presented to an expert from agriculture and classified as showing either weed (class 0) or only crop (class 1).

In Figure 2 two patches classified as crop and two patches classified as weed are shown. Two of the patches are easy to classify (expert or not), while the remaining two less clearly belong to either of the classes.

For each of the cental sub-patches (here of size 100×100 pixels), 13 rotation and translation invariant features were extracted. In more detail, the RGB-values were transformed to HSV and the hue values were extracted. The 13 features were obtained by taking a 13-bin histogram of the relevant color interval.

Reading in the data. The training and test data are in the files `IDSWeedCropTrain.csv` and `IDSWeedCropTest.csv`, respectively, available on Absalon. Each line contains the features and the label for one patch. The last column corresponds to the class label.

This is one way to read in the data in Python:

```
import numpy as np
# read in the data
dataTrain = np.loadtxt('IDSWeedCropTrain.csv', delimiter=',')
dataTest = np.loadtxt('IDSWeedCropTest.csv', delimiter=',')
# split input variables and labels
XTrain = dataTrain[:, :-1]
YTrain = dataTrain[:, -1]
```

```
XTest = dataTest[:, :-1]
YTest = dataTest[:, -1]
```

Neartest neighbor classification

Exercise 1 (Neartest neighbor classification)**.** Apply a nearest neighbor classifier (1-NN) to the data. You are encouraged to implement it on your own. However, you can also use scikit-learn:

```
from sklearn.neighbors import KNeighborsClassifier
```

You are supposed to determine the classification accuracy of your model on the training and test data. One way to do this in Python is the following:

```
from sklearn.metrics import accuracy_score
# given classifier called knn, compute the accuracy on the test set
accTest = accuracy_score(YTest, knn.predict(XTest))
```

Deliverables. Source code; training and test results of your 1-NN classifier; discussion of the results.

You can get pretty accurate classification results on these data. Note that for other crop types and other drone flying heights the classification may be more challenging.

Hyperparameter selection using cross-validation

> As a general rule, you must not use the test data in the model building process at all (neither for training, data normalization, nor hyperparameter selection), because otherwise you may get a biased estimate of the generalization performance of the model.

The performance of the k nearest neighbor classifier (k-NN) depends on the choice of the parameter k determining the number of neighbors. Such a parameter of the learning algorithm (i.e., not a parameter of the resulting model) is called a *hyperparameter*.

Cross-validation is useful to determine proper hyperparameters. This is well supported in scikit-learn, for example:

```
from sklearn.model_selection import KFold
# create indices for CV
cv = KFold(n_splits=5)
# loop over CV folds
for train, test in cv.split(XTrain):
        XTrainCV, XTestCV, YTrainCV, YTestCV = XTrain[train],
        XTrain[test], YTrain[train], YTrain[test]
```

Exercise 2 (Cross-validation)**.** You are supposed to find a good value for k from $\{1, 3, 5, 7, 9, 11\}$. For every choice of k, estimate the performance of the k-NN classifier using 5-fold cross-validation. Pick the k with the lowest average 0-1 loss (classification error), which we will call k_{best} in the following. Only use the training data in the cross-validation process to generate the folds.

Deliverables. Source code; short description on how you proceeded; found parameter k_{best}.

After you have determined k_{best}, you can estimate the performance of the classifier using the test data `IDSWeedCropTest.csv`.

Exercise 3 (Evaluation of classification performance)**.** To estimate the generalization performance, build a k_{best}-NN classifier using the complete training data set `IDSWeedCropTrain.csv` and evaluate it on the independent test set `IDSWeedCropTest.csv` .

Deliverables. Training and test accuracy of k_{best}.

Data normalization

Data normalization is an important preprocessing step. A basic normalization is to generate zero-mean, unit variance input data, see the first three pages of Section 9.1 of our textbook [Abu-Mostafa et al., 2012].

Exercise 4 (Data normalization)**.** Center and normalize the data and repeat the model selection and classification process from Exercise 2 and Exercise 3. However, keep the general rule from above in mind.

You can implement the normalization yourself. First, compute the mean and the variance of every input feature (i.e. of every component of the input vector). Then, find the affine linear mapping that transforms the training input data such that the mean and the variance of every feature are zero and one, respectively, after the transformation.

You may also use scikit-learn. Here are three different ways how one could apply the preprocessing from scikit-learn, only one of which is correct:

```
from sklearn import preprocessing
# version 1
scaler = preprocessing.StandardScaler().fit(XTrain)
XTrainN = scaler.transform(XTrain)
XTestN = scaler.transform(XTest)
# version 2
scaler = preprocessing.StandardScaler().fit(XTrain)
XTrainN = scaler.transform(XTrain)
scaler = preprocessing.StandardScaler().fit(XTest)
XTestN = scaler.transform(XTest)
# version 3
XTotal = np.concatenate((XTrain, XTest))
scaler = preprocessing.StandardScaler().fit(XTotal)
XTrainN = scaler.transform(XTrain)
XTestN = scaler.transform(XTest)
```

Discuss which version is correct and why the other two are not.

Deliverables. Source code; discussion of the three normalization variants including conceptual arguments why two of them are flawed; parameter k_{best} found in the cross-validation procedure; training and test accuracy of k_{best}; short discussion comparing results with and without normalization

References

Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin. *Learning from data.* AMLbook, 2012.

S. Olsen, J. Nielsen, and R. J. Thistle detection. In *Scandinavian Conference on Image Analysis*, 2017. Submitted.

J. Rasmussen, J. Nielsen, S. I. Olsen, K. Steenstrup Petersen, J. E. Jensen, and J. Streibig. Droner til monitorering af flerårigt ukrudt i korn. Bekæmpelsesmiddelforskning 165, Miljøstyrelsen, Miljøministeriet, 2016.