Tutorials 09-11-2018 (M. Biehl)

Suggestions:

- work in groups (as formed for the other tutorials)
- all this should work in the python environments that you have been using; but you may also switch to matlab or whatever language you prefer (of course not everything is available ready-made in the same fashion there)

- consider one or several of the suggested data sets (see following slides)
  and/or any data set from the other tutorials
  and/or or data you are working with anyway

  Important: the data should have numerical features (as opposed to images, for instance) and it should be labeled as to define a classification (or regression) task

Tutorials 09-11-2018

- using a classification method of your choice, e.g. K-NN, SVM-linear kernel, LVQ / GMLVQ ...
  I focus here on classification, but you can also consider regression, e.g. simple linear regression, neural network regression

- implement possible preprocessing, training, validation etc. yourself if you want to get hands-on insight into how things work
  or
  used the provided (or linked to) implementations

- “play” with the data and algorithms by changing parameters etc. or follow one of the example workflows suggested below

suggested data sets:

Iris flower data set (Fisher, 1936)

http://archive.ics.uci.edu/ml/datasets/iris

a classical "toy data set", four-dim. feature vectors,
three-class problem
consider also binary sub-problems, e.g. class 1 vs. others...

UCI segmentation data set

http://archive.ics.uci.edu/ml/datasets/Image+Segmentation

a popular benchmark set, 19-dim. feature vectors, 7 clases
(disregard features 3,4,5 which are nearly constant,
perform z-score transformation)

In **J. Van der Plas'** book (thanks to Aleke for pointing this out)

http://www.astroml.org/book_figures/chapter9/index.html#book-fig-chapter9

you can find several examples for ML analysis of astronomical data, for instance:

http://www.astroml.org/examples/datasets/plot_rrlyrae_mags.html ,

concerning a binary classification of stars based on two dimensional features

This could make a nice example for SVM classification as illustrated also in the book

http://www.astroml.org/book_figures/chapter9/fig_rrlyrae_svm.html

but also for the application of other basic classifiers (see first link)

algorithm suggestions

- **Perceptron / Support Vector Machine**
  implement yourself, e.g. AdaTron with errors
  or
  use available SVM implementations, start linear
  first and extend to more complex kernel later

- **K-nearest neighbor classifier**
  implement yourself K-Nearest Neighbor classification

- **GMLVQ**: scikit/learn compatible implementation
  http://techfak.uni-bielefeld.de/~bpaassen/glvq.zip
  http://github.com/MrNuggelz/sklearn-lvq

other aspects: preprocessing, validation etc.

- **z-score transformation**
  rescale a given data set such that features have
  zero mean / unit variance

- **Receiver Operating Characteristics (ROC)** for two-class problems
  implement ROC computation for threshold classifiers
  e.g. SVM / Perceptron, use implementations in GMVQ toolbox
  etc.

- **Validation:** implement / use built-in procedures
  classical k-fold cross-validation
  repeated randomized splits

Possible mini-project 1

- consider the Iris data set or UCI segmetation data set
- split into training and test data or perform k-fold cross validation
- do the following with or without z-score transformation
  Note: obtain mean and standard deviation per feature only from the training data, but apply the transformation (subtraction of the mean, division by the standard deviation) also in the current validation set!

(*) perform a k-nearest neighbor classification, obtain the number of errors in the training set and in the validation set, perform averages over the validation runs (e.g. in cross-val.)

(**) vary k, repeat and compare by, for instance, plotting the mean training and validation error as a function of k

replace (*) and (**) according to other methods, e.g. SVM (consider several two-class problems to realize the multi-class system, use different kernels, LVQ or GMLVQ (play with #of prototypes, use or do not use relevance learning etc.)

Possible mini-project 2

- consider the Iris data set or UCI segmentation data set
- apply or do not apply z-score transformation (here: of the entire data set) in one or several of the following:
  - perform PCA, visualize the labeled data in the two leading components
  - perform k-means with different k, plot quantization error vs. k
    can you find a characteristic k (elbow?)
  - apply GMLVQ (see link) and visualize the data with respect to the leading eigenvectors of the relevance matrix

Possible mini-project 3

- consider the classification data set from van der Plas' book
- use linear SVM and SVM with more powerful kernels
- compare their performances as obtained in cross-validation
- obtain the ROC for training and validation sets, determine area under curve etc.

Feel free to mix/combine the above suggestions, come up with your own ideas, and ask lots of questions ☺