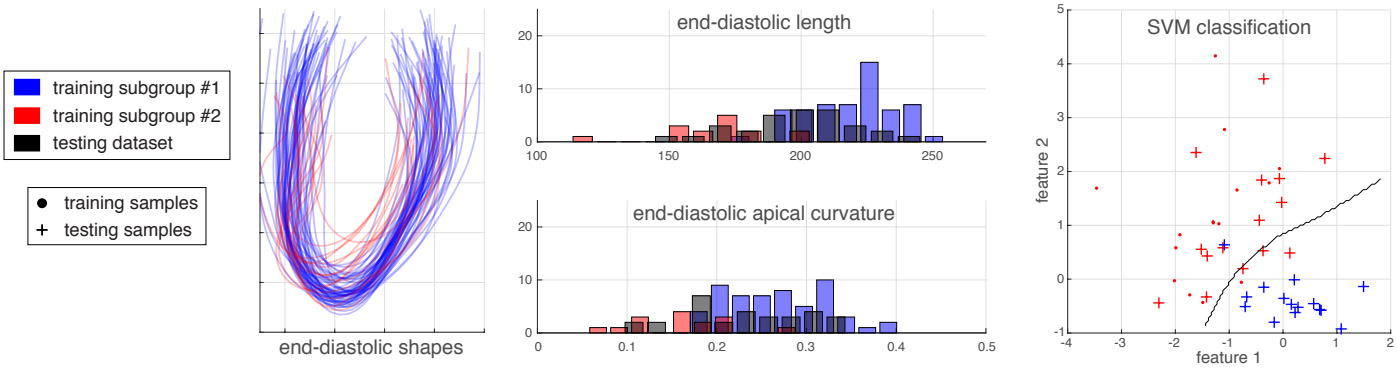
Hands-on: learning a diagnosis from cardiac function data.

The dataset in `cardiacShapes_multi.mat` corresponds to 101 acquisitions from 2D echocardiography, post-processed using commercial 2D speckle-tracking software (Echopac v.110.1.2, GE Healthcare). It consists of the following:

- `HR`: the heart rate of each acquisition,
- `shapeED` and `shapeES`: the myocardial centerline at end-diastole / end-systole, whose dimensions ($101 \times 2 \times 67$) stand for the number of subjects, the x-y coordinates, and the number of points along each curve,
- `sysDuration`: the relative duration of the systole vs. the whole cycle (in %),
- `labels`: the subgroup to which each subject belongs. Subjects are either healthy volunteers or patients with hypertrophic cardiomyopathy.



Objectives

30 subjects have masked labels (`NaN` values): the goal of this hands-on is to predict such labels from the remaining subjects' data. To do so, you are free to use standard statistics or machine learning algorithm(s).

Hints

• Launch the provided code in `ex_Launch.m`, which first plots both the shapes in each subgroup at end-diastole and end-systole. The code then computes 12 features, whose distribution is also plotted.

• Observe the shape data: do you already have the intuition of some subgroup differences? Which features would you consider relevant to compute, given the visualized shapes?

• Have a look at the code: can you identify which features are computed? how relevant do you think they are?

• Observe the feature distribution in each subgroup and check if some features seem more discriminative. Do they agree with your knowledge about the pathology? How challenging is the dataset for the masked cases?

• Fill in the code to launch a Matlab version of kernel SVM: which would be your training and testing samples? what would be the optimal values of the parameters `C` and `sigma`? you can use the `visualizeBoundary` function with 2 features to better observe the influence of these parameters.

• k-fold cross-validation should be preferred over manual tuning of your algorithm parameters to prevent overfitting: how to implement this?

• Once you have predictions on the unknown samples, have a look back at the data given your predictions: do they make sense?

Provided code

We provide an implementation of the kernel SVM algorithm, adapted from *Andrew Ng et al. Machine learning, Coursera online course, October 2013*, which can be used as follows:

- `model = svmTrain( X , y , C , @(x1,x2)gaussianKernel(x1,x2,sigma) );`
  for the *training*, where `X` and `y` stand for the training data and labels, with `C` and `sigma` the parameters to tune,
- `yPredicted = svmPredict( model , Xtest );`
  for the *testing*, where `Xtest` stands for the testing data,
- `visualizeBoundary( X , y , model );`
  for the *visualization* (2 features only !), where `X` and `y` stand for the training data and labels.

Note that dimensions of these objects are `X`: $N_{cases} \times N_{features}$, and `y`: $N_{cases} \times 1$, and that the values in `X` are first normalized before launching the learning.