MATLAB Coding Guidelines at SPG

Mark Ryan Leonard & Adrian Šošić

November 26, 2017

This manuscript serves as a *guideline* for writing comprehensible MATLAB code. This presented list of rules is by no means complete and the used style may change from codebase to codebase as long as it is used consistently.

After some general remarks, we provide prototypical examples of a MATLAB function, a MATLAB script, and how to write pseudo-code in LaTeX.

1 General Remarks

- Give variables meaningful names, e.g. `nDims` instead of `d`.
- Arrange lines of code that implementing basic functionality in groups and explain the functionality of the resulting code segment. For example:

```
%%% generate random mixture weights %%%
weights = rand(nComponents, 1);
weights = weights / sum(weights);
```

2 Functions

- A function should contain a **docstring**, explaining the **purpose** of the function as well as its **interface**. See also `https://de.mathworks.com/help/matlab/matlab_prog/add-help-for-your-program.html`.
- In particular, the docstring should explain **all input/output variable types** and **what they represent**.
- If the function contains optional input/output arguments, their usage should be explained.

Example of a MATLAB function

```
function [samples, indicators] = sampleGMM(mus, sigmas, weights, nSamples)
% Generates samples from a Gaussian mixture model.
%
% [SAMPLES, INDICATOR] = sampleGMM(MUS, SIGMAS, WEIGHTS, NSAMPLES) creates
% a matrix SAMPLES containing a number NSAMPLES of random vectors from a
% Gaussian mixture model that is defined by the mean vectors in MUS, the
% covariance matrices in SIGMAS, and the relative mixture weights in
% WEIGHTS. The vector INDICATORS stores the mixture assignments.
%
% Input
% -----
% mus:
% An (nComp x nDims) matrix containing the mean values of the mixture
% components. "nDims" is the dimensionality of the sample space and "nComp"
% is the number of mixture components.
%
% sigmas:
% An (nDims x nDims x nComp) array containing the covariance
% matrices of the mixture components.
%
% weights:
% An (nComp x 1) or (1 x nComp) vector containing the mixing weights that
% sum to 1.
%
% nSamples:
% An integer specifying the number of samples to be generated.
%
%
% Output
% ------
% samples:
% An (nSamples x nDims) matrix containing the generated samples.
%
% indicators:
% An (nSamples x 1) vector of integers in {1,...,nComp} indicating the
% mixture assignments of all samples.
%
% See also gmdistribution


%%% dimension of the sample space and number of components %%%
[nComp, nDims] = size(mus);

%%% sample indicators %%%
indicators = randsample(1:nComp, nSamples, true, weights)';

%%% container variable to store the samples %%%
samples = zeros(nSamples, nDims);

%%% sample component-wise %%%
for c = 1:nComp
    %%% indices belonging to current component %%%
    inds = indicators == c;

    %%% generate the samples %%%
    samples(inds,:) = mvnrnd(mus(c,:), sigmas(:,:,c), sum(inds));
end
```

3 Scripts

Typically, scripts can be organized in a modular structure, where each part implements a certain functionality. In order to keep your code comprehensible, try to obey the following rules:

- Visually highlight the beginning of a new section (e.g. transitions between pre-processing, inference, post-processing and visualization).

- Use a dedicated section to define your settings. Sometimes, it is helpful to split this section into several subsections to define settings for each stage of the script individually.
- In particular, do **not** use hard-coded numbers in your code but define the corresponding variables in your settings.

Example of a MATLAB script

```matlab
% This script demonstrates the use of the "sampleGMM" function. It
% generates a number of samples from a Gaussian mixture model with
% specified parameters using a single function call. For a number of
% dimensions smaller or equal 3, the samples are subsequently visualized.

%%% clean up %%%
clear; clc; close all;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                             % Settings %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

nSamples    = 1000;     % number of samples to be generated
nComponents = 5;        % number of mixture components
nDims       = 2;        % dimensionality of the samples
variance    = 0.1;      % variance of the (isotropic) components


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                             % Sampling %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% generate random mixture components %%%
mus     = randn(nComponents, nDims);
sigmas  = variance * repmat(eye(nDims), 1, 1, nComponents);

%%% generate random mixture weights %%%
weights = rand(nComponents, 1);
weights = weights / sum(weights);

%%% generate the samples %%%
[samples, indicators] = sampleGMM(mus, sigmas, weights, nSamples);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                             % Visualization %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% visualization only possible for up to three dimensions %%%
if nDims > 3
    disp('Visualization only possible for up to three dimensions.')
    return
end

%%% figure environment %%%
figure(), hold on, grid on

%%% plot the samples of each mixture component %%%
```

```
for c = 1:nComponents

    %%% corresponding samples %%%
    inds = indicators == c;

    %%% call corresponding plotting routine %%%
    switch nDims
        case 1
            stem(samples(inds), c * ones(sum(inds), 1));
            set(gca, 'ytick', [])
        case 2
            scatter(samples(inds,1), samples(inds,2));
            view(2)
        case 3
            scatter3(samples(inds,1), samples(inds,2), samples(inds,3));
            view(3)
    end
end

%%% annotate plot %%%
xlabel('first dimension')
if nDims >= 2
    ylabel('second dimension')
    if nDims == 3
        zlabel('third dimension')
    end
end
title(sprintf('Samples from a Gaussian mixture model with %d components',
   nComponents))
```

4 Pseudo-code

Every algorithm that you develop or modify should be included in your thesis as a pseudo-code. A pseudo-code can be written in LaTeX as follows:

```
\usepackage{amsmath,algorithm,algorithmic}
[...]
\begin{algorithm}
\caption{M-Estimator}
  \begin{algorithmic}[t]
    \STATE{\textbf{Input}: data $\pmb{x}$, threshold $\tau$}
    \STATE{\textbf{1. Set $j = 1$, initialize $\hat{\theta}_0$ and estimate $\
       hat{\sigma}$:}}
    \STATE{\hspace{4.5mm}$\hat{\theta}_0 = \text{median}(\pmb{x})$}
    \STATE{\hspace{4.5mm}$\hat{\sigma} = 1.483 \  \text{median}\left(|\pmb{x} -
       \text{median}(\pmb{x})|\right)$}
    \WHILE{$\frac{\left| \hat{\theta}_{j} - \hat{\theta}_{j-1} \right|}{\hat{\
       sigma}} > \tau$}
      \STATE{\textbf{2. Computation of the weights:}}
      \STATE{\hspace{4.5mm}\textbf{for }$i = 1, \ldots, L$}
      \STATE{\hspace{4.5mm}\hspace{4.5mm}$w_{ji} =W \left( \frac{x_i - \hat{\
          theta}_{j} }{\hat{\sigma}}\right)$}
      \STATE{\hspace{4.5mm}\textbf{end for}}
      \STATE{\textbf{3. Calculate the estimate $\hat{\theta}_{j+1}$:}}
      \STATE{\hspace{4.5mm}$\hat{\theta}_{j+1} = \frac{\sum_{i = 1}^L w_{ji} x_i
          }{\sum_{i=1}^L w_{ji}}$}
      \STATE{\textbf{4. Set $j = j + 1$}}
    \ENDWHILE
    \STATE{\textbf{Output}: $\hat{\theta}_{j}$}
\end{algorithmic}
\label{alg:m_alg}
\end{algorithm}
```

Algorithm 1 M-Estimator

```
Input: data $\boldsymbol{x}$, threshold $\tau$
1. Set $j = 1$, initialize $\hat{\theta}_0$ and estimate $\hat{\sigma}$:
   $\hat{\theta}_0 = \text{median}(\boldsymbol{x})$
   $\hat{\sigma} = 1.483\ \text{median}\left(|\boldsymbol{x} - \text{median}(\boldsymbol{x})|\right)$
while $\frac{|\hat{\theta}_j - \hat{\theta}_{j-1}|}{\hat{\sigma}} > \tau$ do
   2. Computation of the weights:
      for $i = 1, \ldots, L$
         $w_{ji} = W\left(\frac{x_i - \hat{\theta}_j}{\hat{\sigma}}\right)$
      end for
   3. Calculate the estimate $\hat{\theta}_{j+1}$:
      $\hat{\theta}_{j+1} = \frac{\sum_{i=1}^{L} w_{ji} x_i}{\sum_{i=1}^{L} w_{ji}}$
   4. Set $j = j + 1$
end while
Output: $\hat{\theta}_j$
```