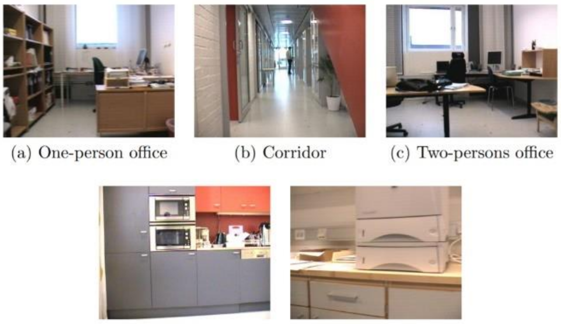
Module-Lab 10: Probabilistic Machine Learning

Intro: Let us work on a multi-class dataset related to mobile robotics application and study the performance of a simple Dynamic Bayesian Network (DBN) applied to such domain. The dataset we are going to explore is the IDOL (which stands for **I**mage **D**ataset for r**O**bot **L**ocalization) which is a dataset devoted to "semantic place classification". IDOL comprises 5 classes:

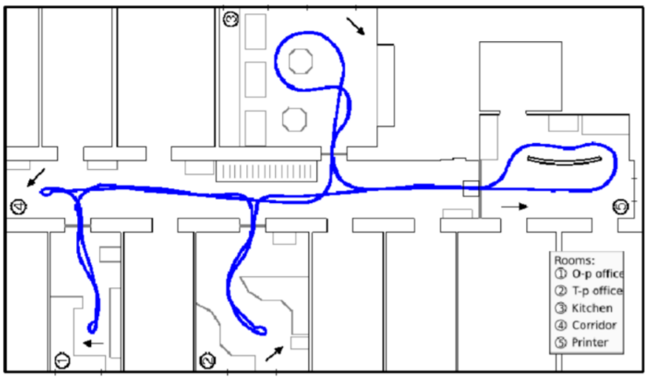
- One-person office [label: 1],
- Two-persons office [label: 2]
- Kitchen [label: 3]
- Corridor [label: 4]
- Printer area [label: 5]



(a) One-person office (b) Corridor (c) Two-persons office
(d) Kitchen (e) Printer area

Two mobile robots have been used to collect the data (images, 2D Lidar scans, and odometry data), namely a PeopleBot and a PowerBot (shown in the left-hand side Picture).

A "map" of the scenario where the dataset has been collected is shown below



Detailed information of the dataset can be found through the following links:
https://www.cas.kth.se/IDOL/
https://www.pronobis.pro/publications/luo2006idol2.pdf

Exercise 1

a) To simplify the problem, we shall use a "processed" test set containing the labels and the probabilistic outputs/scores of a classifier (in this case a SVM). The Matlab file comprising the dataset and the respective labels is ***IDOL_Exp1.mat***. Create a program in Matlab to open such file and then find the number of examples per class. Notice that the labels are {1, 2, 3, 4, 5} and that the dataset is a sequence i.e., the data has been collected along the time by a moving robot.

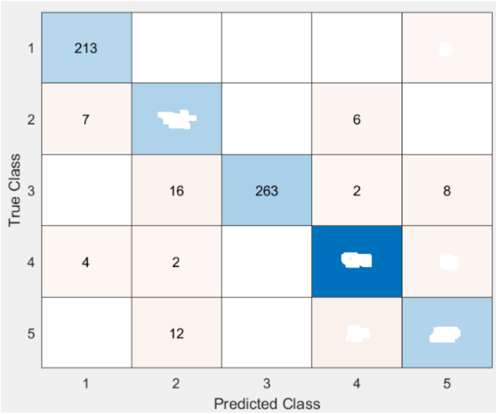
b) Now, create a plot to visualize the labels. You can use the following piece of code

```
L = Ytest1.label; %Labels on a test set
figure(1); cla; hold on
gcolor = ['r','k','b','g','m'];
for i=1:N
plot(i,L(i),'o','Color',gcolor(L(i)));
end
```

c) Using the SVM scores given by the variable `Ytest1.like` – which comes from a SVM with 'probabilistic' output - calculate the TP and FP per class, the Balanced Accuracy (BAcc), average Precision and average Recall as well; some of the results we shall find are:
TP1 = 213
…
TP3 = 263

…
FP1 = 11
FP2 = 30
…
BAcc = 93.12
Average Precision = 94.14
…

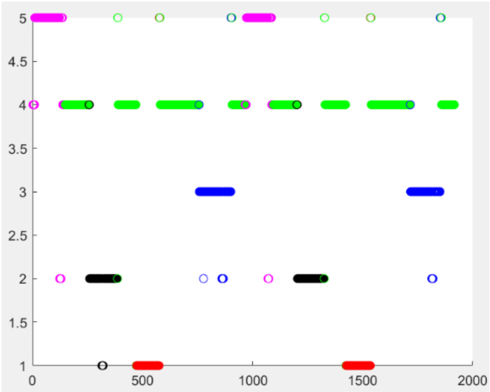
d) Using the Matlab function *confusionchart* obtain the confusion matrix.



e) Now, generate a graph to plot the predicted outputs $\hat{y}(x)$. You can use the following code:

```
figure(2); cla; hold on
gcolor = ['r','k','b','g','m'];
for i=1:N
plot(i,yhat(i),'o','Color',gcolor(L(i)));
end
```

The plot should look like this



Exercise 2

a) This part of the lecture is about DBN. Develop the necessary code and make use of the function *F_DBN_t2* (See the Appendix 1), in Matlab, to evaluate a DBN comprising 2 time-slices. Compare the performance measure against the results obtained above in 1.c) and 1.d).

b) Plot the predicted outputs as we have done in 1.e).

c) Develop a code to implement a DBN having <u>1 time-slice</u>. Compare the results.

<u>Hint</u>: for this DBN we shall get the following results
Acc = 94.78
BAcc = 93.79
Pre = 95.15
Rec = 93.79

Appendix 1

```
function Y = F_DBN_t2(X,AddSm)

  if nargin <2
    AddSm = 0.01; %Additive smoothing [*]
  end

        n = size(X.like,1);
        Y = X; nc = size(X.like,2);

        % % ---
        prio0 = ones(1,nc)/nc;
        like0 = X.like(1,:);

        post0 = like0.* prio0;
        N = sum(post0);
        post0 = post0/N;
        prio1 = post0;

        L1 = X.like(2,:) .* prio1 .* prio0;
        L = L1 .* like0 .* prio1 .* prio0; %
        N = sum(L);
        prio2 = L/N;

        for k=3:n
            L0 = X.like(k-2,:);
            L1 = X.like(k-1,:);
            L2 = X.like(k,:);

            L = L2 .* L1 .* L0 .* prio2 .* prio1 .* prio0; %
            N = sum(L);
            post2 = L/N;
        % ---
        Y.like(k,:) = post2;
        % % % =====================================================
        prio0 = prio1;
        prio0 = prio0 + AddSm; prio0 = prio0 / sum(prio0);

        prio1 = prio2;
        prio1 = prio1 + AddSm; prio1 = prio1 / sum(prio1);

        prio2 = post2;
        prio2 = prio2 + AddSm; prio2 = prio2 / sum(prio2);
        end

end %END Function
```

// ---

[*] More info regarding the smoothing parameter:
http://home.isr.uc.pt/~cpremebida/files_cp/Dynamic%20Bayesian%20Network%20for%20semantic%20place%20classification%20in%20mobile%20robotics_Preprint.pdf