# Pattern Recognition and Machine Learning : Assignment 5

- The assignment is **due** on **April 25**.
- Submit a soft copy of the code and report highlighting the observations and inferences before the deadline.

## Data-set description

- You have been provided training features, corresponding to three classes $\omega_1$, $\omega_2$ and $\omega_3$ in the files `Pattern1.mat`, `Pattern2.mat` and `Pattern3.mat` respectively. Each file contains 200 instances (training examples), of 120 feature dimensions.
- The features corresponding to 100 testing samples of $\omega_1$, $\omega_2$ and $\omega_3$ are contained in `Test1.mat`, `Test2.mat` and `Test3.mat` respectively.

## Task 1 : Single Layer Perceptron

1. By using the features contained in `Pattern1.mat` and `Pattern2.mat`, design a "batch-mode" perceptron classifier for the classes $\omega_1$ and $\omega_2$. How many iterations are required for convergence. Evaluate the performance on the test data `Test1.mat` and `Test2.mat` and report the accuracy.
2. Now consider building a perceptron for the classes $\omega_1$ and $\omega_3$. How many iterations are required for convergence in this case. Comment on the result. Evaluate the performance of this classifier on `Test1.mat` and `Test3.mat`.

## Task 2 : Multi-Layer Perceptron

In this task, I want you to design a multi layer perceptron by implementing the Backpropogation algorithm discussed in class. You may consider a single hidden layer comprising 20 nodes. The activation functions at the hidden nodes may be assumed to be sigmoidal.

- Plot a graph depicting the convergence of the error function with the number of iterations/epochs, during the training process.

- Evaluate the performance of the perceptron on the test-sets `Test1.mat`, `Test2.mat` and `Test3.mat`. Here are a few pointers to guide you through this task.

  1. It is very important to ensure that the mean squared error descends down to a steady value after sufficient number of iterations. You may however observe initial ripples/ spikes at earlier iterations, which is acceptable. However, the oscillatory behavior should subside with increasing number of epochs.
  2. Prior to training, I suggest you to rescale each feature of the training samples, so that they have zero mean and unit variance.
  3. Consider normalizing the weights at the end of each iteration/epoch. This is similar to the weight decay step.
  4. The weights may be initialized randomly from a uniform distribution `U[0,1]`.

## Task 3 : Support Vector Machines

- Build a SVM Classifier for the classes $\omega_1$ , $\omega_2$ and $\omega_3$ using a Radial Basis Function Kernel. Plot a graph depicting the recognition accuracy on the test data `Test1.mat`, `Test2.mat` and `Test3.mat` for different values of penalty factors $C$ and precisions $\gamma$ of the Radial Basis Function. The software package that I recommend for this task is `LibSVM`. On their webpage, download the version (either Matlab/Python), appropriate to the platform in which you are working.

## Task 4: Hidden Markov Models

(a) Train a Hidden Markov Model (HMM) Classifier for the classes $\omega_1$ , $\omega_2$ and $\omega_3$ using the Baum Welch Estimation Algorithm. Please note each of the classes need to be modelled by a separate HMM. You may consider adopting 6 states per class. The observations emitting from each state can be assumed to be generated from a GMM comprising 4 Gaussians.

(b) Using the trained models, obtained from part (a), test the performance of the classifier (using the forward algorithm) on the datasets `Test1.mat`, `Test2.mat` and `Test3.mat`. Report the recognition accuracies obtained. The software package that I recommend for this task is HMM Tool Box available in MATLAB. It can be downloaded from: `http://www.cs.ubc.ca/ murphyk/Software/HMM/hmm.html`