Mid-sem

EE698V – Machine Learning for Signal Processing

Instructions:

- Use Python 3 for coding
- Use only numpy library. No other libraries allowed.
- Submit your ipynb notebooks. Your submission will be evaluated based on the 4 functions explained below. You can define any number of extra functions, classes and variables you wish.
- The runtime of your code should be restricted to 5 minutes on Google colab CPU.
- For any questions, ask on MooKIT forum.

```
class MidSem:
     def __init__(self):
          ...
     def generateData(self, params):
          ...
          return X_ki
     def trainWithMSE(self, X_ki, Yd_k):
          ...
          return w
     def trainBP(self, X_ki, Nepochs=100, eta=0.1):
          ...
          return w
     def evaluate(self, X_i, t_i, w):
          ...
          return CM, accuracy, Y_i
```

Q1. Generate Data (10 marks)

Write the function "generateData" that generates random points that are Gaussian distributed. The input "params" is a list of lists: [[mu1, sigma1, N1], [mu2, sigma2, N2], [mu3, sigma3, N3], ...], where mu and sigma are numpy vectors of mean and covariance matrix of the Gaussian, respectively, and N is the number of samples from that Gaussian. The shape of mu is (M,) and that of sigma is (M,M), and len(params) is K. An example is like this

```
params = [[np.array([2.5,-1.3]), np.ones((2,2)),5], \
          [np.array([-1.9, 3.4]), .5*np.eye(2), 4]]
```

The output "X_ki" should again be a list of numpy arrays, each array being of shape (N,M). Here, k is the class index and i is the sample index for that class.

Hint: If you need data normalization, you can define class variables.

Q2. Linear Binary Classifier with Least Squared Solution (20 Marks)

Write the function "trainWithMSE" that trains a linear model **y=wx** using least squared solution. The input X_ki is defined above and Yd_k is a list of class targets with len(Yd_k) as 2; e.g. Yd=[-1,1].

The output **w** is a numpy array of **weights** (*+ bias)* of the linear model with shape (**M+1**,).

Q3. Linear Classifier with Softmax and Backpropagation (30 Marks)

Write the function "trainBP" that trains a model y=softmax(wx) using backpropagation. The input Nepochs (int) is the number of epochs for training and eta is the learning rate (float). Feel free to change their default values. Here, X_ki and w are as defined above. The targets Yd_k can be derived from index k in X_ki; there are K classes.

Q4. Evaluation (20 Marks)

Write the function "evaluate" that tests the trained model. The input X_i is a numpy array of shape (N,M), t_i is a numpy array of targets of shape (N,) and w is as defined above. The output CM is the confusion matrix, i.e., a numpy array of shape (K,K); accuracy is the accuracy of the model; and Y_i are the predictions corresponding to X_i, i.e., each row is y=wx or y=softmax(wx) and shape of Y_i is (N,) or (N,K) respectively.
Note: This function should be able to evaluate the models of both Q2 and Q3.