Data Classification using GMM and LSTM

Tongdi Zhou
Department of Electrical and Computer Engineering, Temple University
tzhou@temple.edu

Introduction: The goal of this final exam is to classify a two-class data set using a neural network approach and a non-neural network machine learning approach. The data set contains a 2D data set and a 5D data set. Figure 1 shows the scatter plot of the 2D training data. As the figure shows, the data is generated by two Gaussian mixture models each with 4 components. The 5D data is generated in a similar fashion. A straightforward way to classify the data is using the Gaussian mixture modeling maximum likelihood classification (GMM). For the neural network approach, we use a recurrent neural network with long short-term memory (LSTM) for classification. For the 2D data set, the training data set has 10,000 samples; the development data set as well as the evaluation data set have 2,000 samples. For the 5D data set, the training data set has 100,000 samples, while the development data set and the evaluation data set have 10,000 samples each. The overall strategy is to train the models on the training data set, to use the development data set as validation, and to evaluate the performance of the model using the evaluation data set.

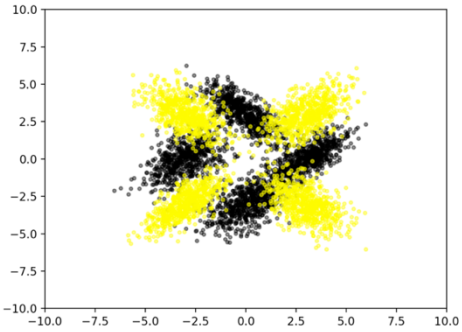

Figure 1. Scatter plot of the 2D training data set

GMM Description: To perform GMM maximum likelihood classification, we first need to estimate the Gaussian mixture models for each class using the training data set. The Statistical Learning Toolbox in MATLAB has built-in functions for estimating Gaussian mixture models. The estimation is done by an iterative Expectation-Maximization (EM) algorithm. The values of the parameters are initialed using a k-means clustering algorithm. We assume 4 components in the Gaussian mixture model, so the k value is set to be 4. After the initialization, in the E-step, the algorithm calculates the posterior probabilities of each Gaussian component for every data sample. The result of the E-step is a N-by-4 matrix where N is the number of training samples. The (i, j) element of the matrix contains the posterior probability that sample i is from Gaussian component j. Next in the M-step, the Gaussian component-wise posterior probabilities are used as weights to update the mean vectors and the covariance matrices of the Gaussian mixture models by maximum likelihood principle. The EM algorithm iterates over the E-step and the M-step until convergence is achieved. It should be noted that upon convergence, there is no guarantee that the solution is the global optimum. To prevent a suboptimal solution, the algorithm repeats the EM algorithm using a new set of initial values, and the solution with the maximum posterior probability is chosen to be the final result. In our experiments, the repetition number is set to be 50. The classification is performed once the Gaussian models are estimated. Each testing data point is passed into the probability density function of the Gaussian mixture model of two classes. The testing data point is assigned to the class with the higher likelihood. We run the entire process using the training data set for 100 times, and the Gaussian mixture models that produce the lowest error rate on the development data set are chosen to classify the evaluation data set.

LSTM Description: The long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture. Like RNNs, LSTMs have feedback connections and have the ability to store past information for future prediction. Unlike RNNs, LSTMs can deal with vanishing gradient problems that traditional RNNs usually have. The LSTMs have multiplicative gate units that can learn to open and close access to constant error flow. In LSTMs, there is a special structure called the memory cell. In each memory cell, there are four major components: the input gate, the forget gate, the output gate, and the neurons with self-

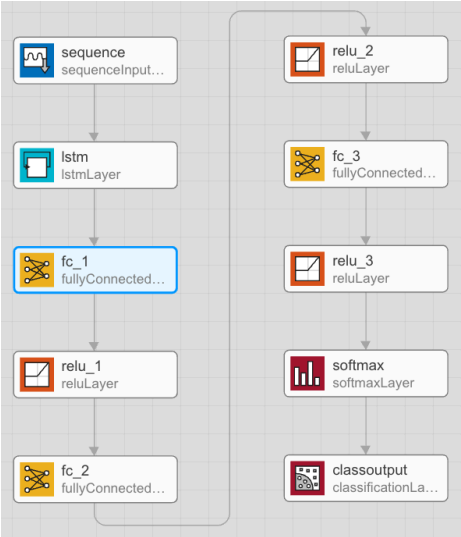

Figure 2 A recurrent neural network (RNN) with long short-term memory

feedback. MATLAB Deep Network Designer and MATLAB Deep Learning Toolbox are used for the experiment. Figure 2 shows the architecture of the entire neural network. The input sequence block takes sequences. The LSTM layer has 128 hidden units. The fully connected layers "fc_1" and "fc_2" both have 150 nodes. The "fc_3" has 2 nodes for the 2 classes.

We use a mini-batch approach for the neural network training. A mini-batch approach divides the training data set into small batches. The small batches are used to compute the error and update the network parameters. The mini-batch approach updates more frequently compared to regular approaches, which can result in a better convergence point. However, the mini-batch size should be carefully chosen. A small mini-batch size will have more noisy samples, while a large batch can significantly slow down the convergence. We find a mini-batch size of 1,000 is a good compromise.

Another important technique is to drop the learning rate during the training process. The learning rate determines the rate at which we update the parameters. At the beginning of the training, we would like a higher learning rate to make the neural network quickly converge to the vicinity of the optimum. Once we are close to the optimum, we need a lower learning rate to reach a high accuracy. Otherwise, the network will oscillate around the optimum without reaching it. Furthermore, there are two considerations on how to drop the learning rate. The first consideration is how often do we drop the learning rate. If the learning rate is dropped too infrequently, the network takes most of the time oscillating instead of approaching the optimum. On the other hand, if the learning rate is dropped too frequently, the learning rate can become very small before the network reaches the vicinity of the optimum, and the network could be stuck at a local optimum point. The second consideration is the drop factor at which we drop the learning rate. The drop factor is between 0 and 1, and the learning rate is multiplied with the drop factor at each rate drop. If the drop factor is too low, the learning rate is decreased too fast, and the network could be stuck in local optimum. If the drop factor is too high, the training takes a long time. We find at an initial learning rate of 0.01, we drop the learning rate every epoch at a drop factor of 0.9 is suitable for the data set.

Results: Table 1 shows the classification error rate of the 2D and the 5D data set using the Gaussian mixture model classifier (GMM) and the long short-term memory neural network (LSTM). We can see that the performance of the two methods are very similar. The difference between the results is not statistically significant. However, the computational burden of the GMM is much lower than the LSTM. This is expected since the data is generated with Gaussian mixture models. GMM should be able to model the data with high accuracy. However, this is not a fair comparison since we use the prior information that the data is generated from Gaussian mixture models with 4 components. If we do not have this prior information, GMM does not necessarily have the same performance as LSTM. We also need to estimate the number of components in the Gaussian mixture model, which brings additional computational burden.

Conclusions: The neural network method and non-neural network method have similar performance. The GMM should be used if the prior information is provided. Otherwise, LSTM can be applied to more generally cases.

	2D			5D		
Algorithm	**Train**	**Dev**	**Eval**	**Train**	**Dev**	**Eval**
GMM	8.00%	7.80%	8.20%	36.84%	36.00%	36.30%
LSTM	8.09%	8.25%	7.80%	36.78%	36.79%	36.74%

Table 1. Classification error rate of the 2D and the 5D data set using the Gaussian mixture model classifier (GMM) and the long short-term memory neural network (LSTM)