KNN and MLP Data Classification on 2D and 5D Data Sets

Mohsen Derakhshan
Department of Mechanical Engineering, Temple University
mohsen.derakhshan@temple.edu

Introduction: In this project classification with a neural network and an optional non-neural network algorithm on two different data sets with 2D and 5D dimensions are investigated. Each data sets consist of three files: one large training file, a development file, and an evaluation file. It is supposed to train a classifier on the train data file, develop it on the development data file, and finally evaluate it on eval file. To have a better idea about data sets, the scatter plots of data sets is needed which is shown in Fig. 1 (for the 5D dataset, Fig.1 b, randomly 2 features are considered to obtain the scatter plot and since for this data set other combinations to provide similar information, therefore presenting them has no new information).

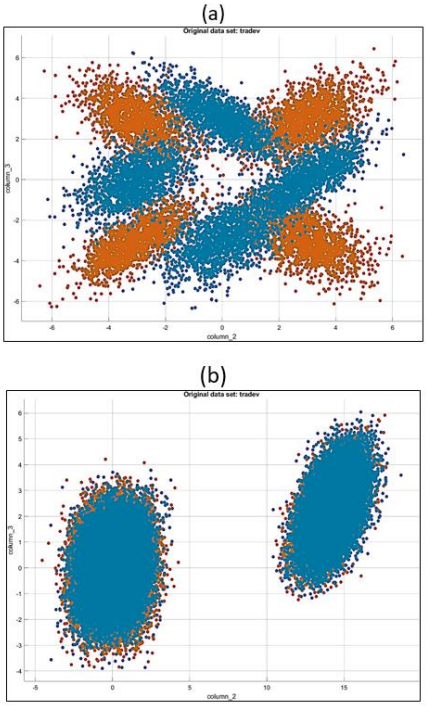

Figure 1: scatter plot of two features of (a) 2D data set and (b) 5D data set

According to the scatter plots in Fig. 1 it is obvious that the decision boundaries are nonlinear, so nonlinear classifier is required to have a better result. On the other hand, the main task of this project is to classify the evaluation data file of the 5D data set which does not have a label. So, it means that the combination of the supervised and unsupervised classifier should be used here to classify and labeled the evaluation data. For these reasons Random Forest Trees (RNF), K-Nearest Neighbors (KNN), and Support Vector Machines (SVM) are three main candidates. Alternatively, we can use a neural network algorithm for classifying these data sets.

K-Nearest Neighbors (KNN - MTLAB): For selecting one of the mentioned algorithms, all three algorithms (RFT, KNN, and SVM) are investigated in the MATLAB classification learning toolbar and by trying and error on their tunable parameters some kind of the range of their accuracy for classifying training data is obtained. Each of the algorithms has their own advantages and disadvantages, i.e. SVM with Gaussian kernel function seem to have a good fitting with this data (from scatter plot it seems that multivariable gaussian can have a good prediction of the data distribution) but in the other hand, it has two main parameters which required tuning and also the runtime of this algorithm is more than two others. According to my investigation, by changing the parameters, it seems that KNN and SVM have a better performance than RFT. The KNN has only one main tunable parameter and it has a shorter runtime which is its merit, but on the other hand, its performance is very sensitive to the number of the k, so finding the optimal k in this algorithm is one of the main issues. Since the KNN and SVM have almost the same accuracy in classifying training data, so the simplest and fasted one, KNN, is selected in this project (of course, even by exactly the same error rate of classification on training data for both algorithms, still there is no guaranty that which one has a better performance on dev and evaluate data).

Multilayer Perceptron (MLP - MATLAB): MLP, has main parts, an input layer, an output layer, a hidden layer (which can be one or several layers) some perceptron in these layers. In each of these parts, there are some tunable parameters that the performance of the net completely depends on them. The number of the layers, the number of the perceptron in each layer, some information about looping such as epoch, terminate condition, the number of iterations, the ratio of the train data to be considered as a dev, and test data and etc. are main variables. Definitely finding all the optimal value of the all these parameters is very hard and in some cases impossible, in this project by trying several different values for this parameters, I tried to find the best combination (in this problem we can use random based optimization algorithms like

BA, GA, PSO, … to find the optimal values). The value of the parameters and some reason behind them is mentioned in the following parts.

Results: For Non-Neural networked I used MATLAB classification toolbox, after trying all approaches as mentioned before, I select KNN. To increase the accuracy of the training, cross-validation by 5 fold is used (in this way I put all train and dev data in one matrix, and since the fold is 5 so the algorithm divided data into five groups and randomly select one of them as the dev data). By increasing the number of the K and checking the performance on dev data I selected k=27 for 2D and k=57 for 5D. generally, the range of error of the classification did not change a lot by changing the k (I mean statically the range of difference was not significant however I have to say it by providing some numbers!) so to reduce the cost of calculation it is better to avoid choosing large k. definitely, since I used cross-validation so the error of classification for train and dev are close to each other however it can be different for eval data.

For MLP and as the first important parameter, I tried 1, 2, 3, and four hidden layers for both 2D and 5D, and second, I tried several different from 3 to 250) perceptron in each layer. The general observation, Then the number of layers increased. When the number of layers has been increased, the error rates on train data significantly decreased but the error rates on development data increased. This observation clearly shows that increasing the number of layers will cause overfitting. It means that the classifier just adjusted to train data and has lost generalization. So, I see that two layers are a kind of optimized number for the hidden layer however for 2D data there was not a significant difference between one hidden layer and two. So, for 2D I set two hidden layers with 5 and 3 neurons in the first and second layers, respectively. For 5D, again I used 2 hidden layers with 150 and 75 neurons. The third parameter was the ratio of train data that the net use them as dev and test data. From my observation, I see that the classification is not very sensitive to this ratio, so I set 20% of the train data as dev and 10% for the test (in nntool of MATLAB you cannot adjust these value directly so I added some lines of code for this purpose). The result of the two classifications are presented in table 1.

Conclusions: Two different non-linear classification algorithms are used (KNN and MLP) to classified two sets of data (with 2 and 5 features respectively). Among non-neural network algorithms RFT, KNN, and SVM were representatives (because the decision boundary is very non-linear and we have to use a combination of supervised and unsupervised classifier). Since SVM and KNN had the same accuracy of classification for these data sets, so the KNN is selected because it has only one tunable parameter and also faster than SVM (RFT has a lower accuracy than these two algorithms upon my observations). In MLP I used two hidden layers for both data sets because however by increasing the numbers of later and neurons the error rate of MLP on train data decreases, but the error rate on dev data and evaluation data will increase and it clearly shows the overfitting. So for avoiding the overfitting and also reducing the calculation cost and time, I used 2 hidden layers and I avoid selecting a large number of neurons. For this problem, it is completely clear that the difference in the performance of the KNN and MPL on both data sets is not significant (statically the error rate of classification by these two methods is not significantly different however we can show it by calculating the level of confidence). So, since the KNN is much faster than MLP so I will select KNN as my classifier for this problem.

		error of classification (%)		
Algorithm		**Train**	**Dev Test**	**Eval**
2D	KNN	07.86	07.35	07.85
	MLP	07.91	08.40	08.10
5D	KNN	35.73	35.47	37.55
	MLP	36.63	36.92	37.25

Table 1. result of KNN and MLP on 2D and 5D data sets