King Chung Ho (Johnny Ho) SID: 403833407
CS 260 Final Project Report

A. Introduction, discussion, and Summary

K-Nearest Neighbors (K-NN), and neural network (multi-layer perceptrons) have been implemented to classify the given dataset. 24 features have been generated from the given time-series data, and 14 of them are used in building the classifiers. Regularization has been added (weight decay) to neural network to avoid overfitting; each algorithm hyperparameters are obtained from cross validation. In addition to that, one unique contribution of this work is to implement a joint algorithm (autoencoder+KNN) which makes use of autoencoder (neural network) to generate new features from input and use K-NN as a classifier (with the new features).

Overall, the K-NN achieves the best performance in 10-fold cross-validation, with accuracy of 76.9% and f-measure of 76.9%, using Hamming distance measure. The 4-layers neural network (input-hiddenLayer-hiddenLayer-output, i.e., 14-6-3-2) achieves an accuracy of 71.8%, with a f-measure of 71.8%. The autoencoder+KNN achieves an accuracy of 71.8%, with a f-measure of 71.8%, using Chebychev distance measure. Based on the hyperparameters obtained from 10-fold cross-validation, leave-one-out cross-validation is performed for each classifier and the result is summarized below.

Although the performance of autoencoder+KNN is not the best, it shows some promising result. Neural network alone (**Accuracy:41.0%; F-measure:41.0%)** classifier or K-NN alone classifier (**Accuracy:48.7%; F-measure:47.4%)** performs worse than autoencoder+KNN (**Accuracy:71.8%; F-measure:71.8%)** with similar hyperparameters. This shows that autoencoder is capable to extract useful features (non-linear) from the input. In this project, 1-hidden layer autoencoder is used, and it is expected that multi-layer autoencoder [1] can obtain even more fundamental (and complex) features from the input. However, due to limited amount of data, multi-layer autoencoder is not implemented.

Comparing the cross validation result obtained from dataset 1 and dataset 2(LDL), the classifiers generally perform better with dataset 1. However, 4-layers neural network performs equally in these two datasets. This shows that multi-layer neural network is more robust.

Neural network (Convolutional) has been applied successfully in medical domains [2-4], however, the result is not as expected in this work. There are several possible reasons. First, dataset is small, and the heterogeneity of the dataset makes it difficult to build a good classifier. Second, there are many parameters. With small amount of data, and the fact that gradients diminish during backpropagation, it is hard to obtain a set of good weights for a neural network. Third, not enough features are generated. There are only 20+ features from the time-series data. Other methodology, such as fourier transform, can be applied to the data and obtain useful features for building classifiers.

All-in-all, future direction would be using different methods to extract useful features from the dataset, and then applying multi-layer autoencoder to extract complex but representative features. Then, we can use these features to build a K-NN or softmax classifier.

Additional Contributions

- In autoencoder, I added a sparsity penalty term, and desired average activation to control the hidden layer activations, which make the weights converge better.
- The codes are implemented in vectorised form, especially in neural network. This allows very fast hyperparameters optimization (e.g., it takes less than 10s to finish a 10-fold cross validation for a multi-layers neural network.).
- Softmax classifier is added to the end of neural network (instead of a node), which is a generalized form of logistic regression.

B. Methodology

B.0 Features generation

I obtained features from each time-series data (systolic & diastolic) as well as the difference (i.e., systolic-diastolic): Max, min, mean, standard deviation, skewness, kurtosis, number of peaks (max & min), approximate frequency. Not all of the features are used.

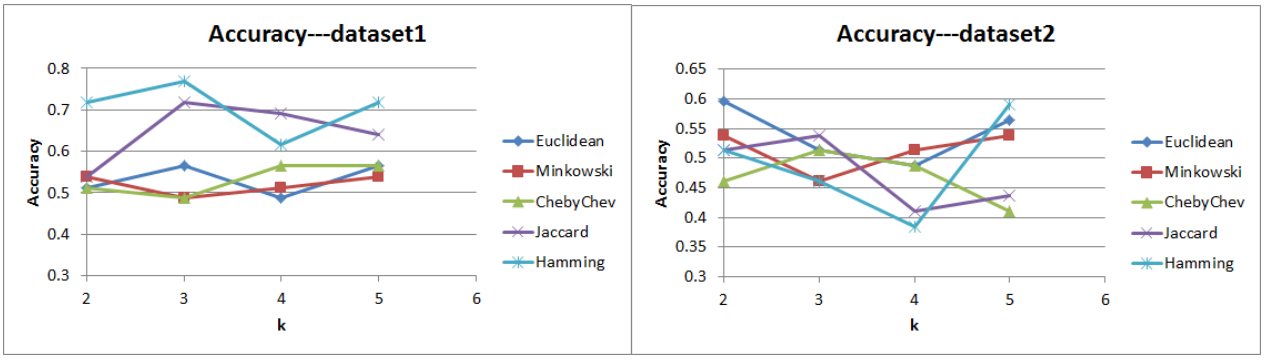
B.1 K-Nearest Neighbors (K-NN)

Below figures show the accuracy of the K-NN classifiers with different k values and distance measurement. The table shows the evaluation matrices. The K-NN classifier with 3 neighbors and Hamming distance performs the best.

Hyperparameters

k:2,3,4,5

Distance measurement: 'euclidean','minkowski','chebychev','jaccard','hamming'



	Set 1		Set2 (LDL)	
	10-fold cross validation	**Leave-one-out**	**10-fold cross validation**	**Leave-one-out**
Distance Measurement	Hamming	Hamming	Euclidean	Euclidean
Confusion Matrix	$\begin{bmatrix}15 & 4\\5 & 15\end{bmatrix}$	$\begin{bmatrix}12 & 6\\8 & 13\end{bmatrix}$	$\begin{bmatrix}16 & 16\\3 & 7\end{bmatrix}$	$\begin{bmatrix}14 & 13\\5 & 7\end{bmatrix}$
Accuracy	**0.769**	0.641	0.589	0.538
Sensitivity	0.750	0.600	0.842	0.737
Specificity	0.789	0.684	0.35	0.350
Precision	0.789	0.667	0.551	0.519
Recall	0.750	0.600	0.842	0.737
F-measure	**0.769**	0.632	0.667	0.609
MCC	0.539	0.285	0.220	0.094

B.2 Neural Network (multi-layer perceptrons)

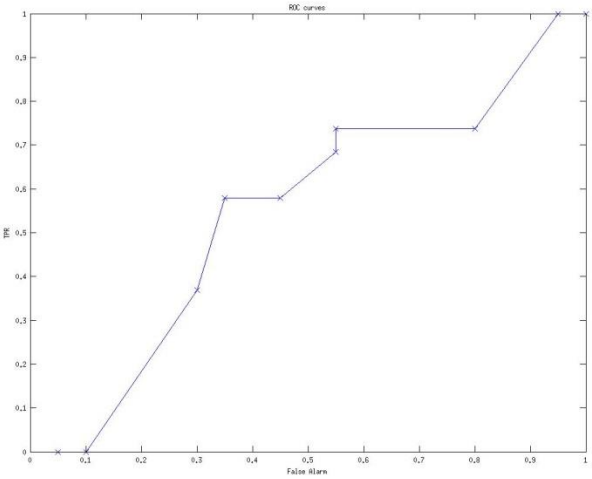
Below figures show the ROC curves of the best neural network classifier (left: correspond to dataset 1; right: correspond to dataset 2); the table shows the evaluation matrices.

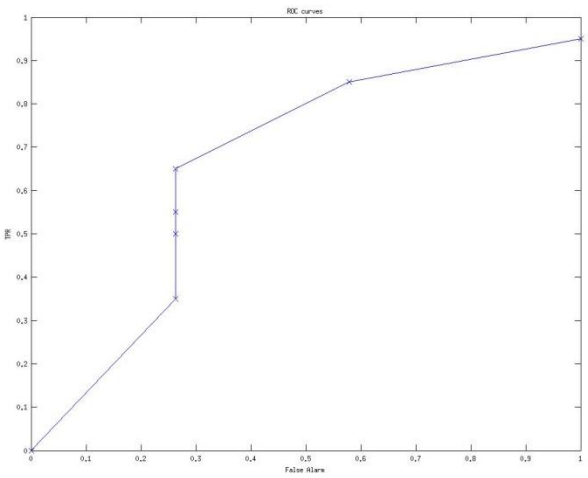
Different hidden layers structure has been tested. Multi-layers work better as it is able to separate data non-linearly. Softmax classifier is also added to the end of the neural network (instead of a node) for building a better classifier. The neural network with 2 hidden layers performs the best.

Hyperparameters

Hidden layers structure: [2],[4],[6],[8],[10],[12],[14],[16],[18],[20], [2 2],[4 4],[6 6],[8 8],[10 10],[12 12],[14 14], [4 2],[6 3],[8 4],[10 5],[12 6],[14 7], [8 4 2],[12 6 3],[2 2 2],[4 4 4]

Weight Decay parameter = 1e-2, 1e-3, 1e-4, 1e-5





	Set 1		Set2 (LDL)	
	10-fold cross validation	**Leave-one-out**	**10-fold cross validation**	**Leave-one-out**
Hidden Layers	Input-6-3-2	Input-6-3-2	Input-6-6-2	Input-6-6-2
Confusion Matrix	$\begin{bmatrix}14 & 5\\ 6 & 14\end{bmatrix}$	$\begin{bmatrix}10 & 7\\ 10 & 12\end{bmatrix}$	$\begin{bmatrix}16 & 8\\ 3 & 12\end{bmatrix}$	$\begin{bmatrix}14 & 5\\ 7 & 13\end{bmatrix}$
Accuracy	0.718	0.564	0.718	0.692
Sensitivity	0.700	0.500	0.842	0.737
Specificity	0.737	0.632	0.600	0.650
Precision	0.737	0.588	0.667	0.667
Recall	0.700	0.500	0.842	0.737
F-measure	0.718	0.541	0.744	0.700
MCC	0.437	0.133	0.454	0.388

B.3 Autoencoder + K-NN

Autoencoder is a type of neural network which is consist of three layers: input, a hidden layer, and output. Input and output are restricted to be the same, and the hidden layer becomes a new features set for the input data (similar to what PCA does except it is non-linearly dimension reduction if the size of hidden units is smaller than input size). In my work, autoencoder is used to extract features from input and the output of autoencoder is fed to K-NN for classification.

The autoencoder with 14 hidden units works the best with K-NN of 3 neighbors. Since the number of hidden units is more than the input units, we can called this over-complete hidden layer.

Hyperparameters

K=3 (obtained from K-NN result)

Hidden layer size = [2],[4],[6],[8],[10],[12],[14],[16],[18],[20]

Desired average activation of hidden unit= 0.1, 0.01, 0.001

Weight decay parameter = 3e-2, 3e-3, 3e-4, 3e-5

Weight of the sparsity penalty term = 3

	Set 1		Set2 (LDL)	
	10-fold cross validation	**Leave-one-out**	**10-fold cross validation**	**Leave-one-out**
Distance measurement	Chebychev	Chebychev	Minkowski	Minkowski
Hidden Layers	Input-14-2	Input-14-2	Input-18-2	Input-18-2
Confusion Matrix	$\begin{bmatrix}14 & 5\\ 6 & 14\end{bmatrix}$	$\begin{bmatrix}10 & 9\\ 10 & 10\end{bmatrix}$	$\begin{bmatrix}16 & 8\\ 3 & 12\end{bmatrix}$	$\begin{bmatrix}10 & 5\\ 9 & 15\end{bmatrix}$
Accuracy	0.718	0.513	0.667	0.641
Sensitivity	0.700	0.500	0.579	0.526
Specificity	0.737	0.526	0.750	0.750
Precision	0.737	0.526	0.688	0.667
Recall	0.700	0.500	0.579	0.526
F-measure	0.718	0.513	0.629	0.588
MCC	0.437	0.0263	0.334	0.284

References

1. Hinton, G. (2006). Reducing the Dimensionality of Data with Neural Networks. Science, 313(5786), 504-507. doi:10.1126/science.1127647
2. Roth, H. R., Lu, L., Seff, A., Cherry, K. M., Hoffman, J., Wang, S., ... & Summers, R. M. (2014). A new 2.5 D representation for lymph node detection using random sets of deep convolutional neural network observations. In Medical Image Computing and Computer-Assisted Intervention–MICCAI 2014 (pp. 520-527). Springer International Publishing.
3. Hoo-Chang Shin, Orton, M., Collins, D., Doran, S., & Leach, M. (2013). Stacked Autoencoders for Unsupervised Feature Learning and Multiple Organ Detection in a Pilot Study Using 4D Patient Data. IEEE Transactions On Pattern Analysis And Machine Intelligence, 35(8), 1930-1943. doi:10.1109/tpami.2012.277
4. Wang, H., Cruz-Roa, A., Basavanhally, A., Gilmore, H., Shih, N., & Feldman, M. et al. (2014). Mitosis detection in breast cancer pathology images by combining handcrafted and convolutional neural network features. Journal Of Medical Imaging, 1(3), 034003. doi:10.1117/1.jmi.1.3.034003