Selected Topics in Neural Networks

Project Report

Different Approaches to Human Cancer Classification from GCM (Global Cancer Mapping) Dataset

Group 18:

Harsh Gaglani (201401415)

Purvik Shah (201401417)

Nirvisha Mankad (201401421)

Introduction:

The aim of classification of different human cancers based on the genomic data is to get precise information to help in accurate diagnosis which eventually leads to patient specific and disease specific treatment. Global Cancer Mapping Dataset is an oligonucleotide microarray data obtained from solid tumors of epithelial origin. It has 16063 features (genes) from 190 tumor samples (77 of these were normal samples.) The tumor samples span 14 different types of cancers (classes). Out of these 190 samples, 144 randomly selected samples are used for training. Thus, with 16063 features, 14 classes and only 144 randomly selected samples for training, this is a very sparse dataset. It is highly imbalanced too because in the randomly selected samples for training, samples from some classes will be more in number than samples from some other class. We were given a reduced set of 98 features and we have run different Neural Network algorithms to classify the samples in different classes (cancers) as precisely as possible.

Feature Reduction:

We tried reducing the features from 16,063 to 100 by ourselves as well. For that, we used DeeBNet (Deep Belief Network) toolbox in MATLAB. Deep Belief Networks (DBNs) are deep architectures that use stack of Restricted Boltzmann Machines (RBM) to create a powerful generative model using training data. DBNs can be used for many things such as feature extraction and classification. In the end, it was very computationally expensive to extract important features out of the large feature set. So, for our project we have used the reduced 98 features given to us.

Approaches:

We have implemented the following algorithms to classify the GCM data:

1) Radial Basis Function Neural Network (K-means clustering followed by pseudo inverse method; with and without gradient descent)

2) MultiLayer Perceptron (MLP) Neural Network with Least Square loss function

3) MultiLayer Perceptron (MLP) Neural Network with Cross Entropy loss function

4) Sparse Extreme Learning Machine (S-ELM)

We describe all the above approaches in brief below. The final results (accuracies) for all these approaches have been consolidated in a table, after the descriptions.

1) Radial Basis Function

We have used the gaussian function as our radial basis function. We first find centres for hidden neurons by using K-means clustering and then find the output weights using Pseudo-Inverse method. After initializing the output weights and hidden neuron centres, we have tried adding further learning of weights by gradient descent. We changed the number of hidden neurons in both cases in the range of 20 to 100 and found the optimum parameters. <u>We found 80 to be the optimum number of neurons.</u> We need so many hidden neurons because our data is very sparse; we have only 144 training samples for 14 classes. Following table shows the results we obtained for varied number of neurons.

RBF without gradient descent				
Number of Clusters - K	Overall testing accuracy	Average testing accuracy	Geometric testing accuracy	Number of classes with 0 correct classifications
20	17.391	15.331	0	6
40	60.87	53.81	0	2
60	80.435	79.422	59.604	0
80	82.209	79.643	60.318	0
100	78.261	77.619	59.364	0

It can be seen from the table that the testing accuracy increases as the number of neurons increases. But after a point, over learning occurs and the accuracy decreases.

We also tried to approximate the network weights (both input and output weights) better by using Gradient Descent algorithm in addition to the pseudo inverse method mentioned above. We tried different number of hidden neurons and different number of epochs to get the optimum parameters. We found 80 hidden neurons and 1000 epochs optimum for our dataset. We need so many hidden neurons because our data is very sparse and we have 14 classes. Our observations for various number of hidden neurons and number of epochs are shown in the tables below:

RBF with gradient descent (epo = 500)				
Number of Clusters - K	Overall testing accuracy	Average testing accuracy	Geometric testing accuracy	Number of classes with 0 correct classifications
20	45.652	35.119	0	3
40	63.043	60.884	28.542	3
60	78.261	75.119	0	2
80	78.261	78.095	0	1

RBF with gradient descent (epo = 1000)				
Number of Clusters - K	Overall testing accuracy	Average testing accuracy	Geometric testing accuracy	Number of classes with 0 correct classifications
20	45.652	35.119	0	3
40	69.565	68.452	0	1
60	78.261	74.405	55.554	0
80	84.783	79.167	59.537	0

RBF with gradient descent (epo = 2000)				
Number of Clusters - K	Overall testing accuracy	Average testing accuracy	Geometric testing accuracy	Number of classes with 0 correct classifications
20	63.043	58.333	0	1
40	84.783	83.929	0	1
60	78.261	75.289	0	1
80	58.696	61.726	0	2

Thus, on increasing epochs and number of hidden neurons, accuracy increases but after a point overlearning starts happening and testing accuracy decreases. RBF gives good accuracy for classification of cancer samples using the GCM dataset. The training and testing accuracies are mentioned in table-1.

2) MultiLayer Perceptron with Least Square Loss function

We used Unipolar Sigmoidal function as the activation function and Least Square loss function. The unipolar sigmoidal function is:

$$\varphi(v) = \frac{1}{1+\exp(-av)} \qquad \text{------uni-polar sigmoid}$$

and the least square loss function is:

$$\text{Least Square (LS): } \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C}(\hat{y}_{ij} - y_{ij})^2$$

We tried different number of epochs and different number of neurons to find the optimum parameters. <u>We found 60 optimum number of hidden neurons and 2000 optimum number of epochs.</u> A large number of epochs are needed because our data set is very sparse (144 training samples with 14 classes) and it needs these many iterations to learn enough to get even 70-75% accuracy. We could not try many varying number of epochs and number of hidden neurons because these many epochs were computationally very expensive.

3) MultiLayer Perceptron with Cross Entropy Loss function

We used Unipolar Sigmoidal function as the activation function and Cross Entropy loss function. The unipolar sigmoidal function is:

$$\varphi(v) = \frac{1}{1+\exp(-av)} \qquad \text{------uni-polar sigmoid}$$

And the cross entropy loss function is :

$$\text{Cross Entropy (CE): } \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C}\left[\frac{(1+y_{ij})}{2}\log\frac{1+\hat{y}_{ij}}{2} + \left(\frac{1-y_{ij}}{2}\right)\log\left(\frac{1-\hat{y}_{ij}}{2}\right)\right]$$

We tried different number of epochs, different number of hidden neurons and different learning rates and we got around 75-80% efficiency. The following table has the results obtained for different sets of network parameters that we tried:

Number of Clusters - K	Overall accuracy	Average Accuracy	Geometric accuracy	Number of classes with 0 correct classifications
MLP Cross Entropy Loss (epo = 1000)				
20	32.609	23.81	0	3
40	63.043	54.762	0	2
60	65.217	57.143	0	1
80	63.043	54.762	0	1

MLP Cross Entropy Loss (epo = 2000)				
20	45.652	34.524	0	1
40	78.261	72.619	0	1
60	82.609	77.381	0	2
80	82.609	76.19	0	2

MLP Cross Entropy Loss (epo = 3000)				
20	76.087	69.048	0	1
40	80.435	75	0	1
60	86.957	79.762	0	1
80	84.783	79.762	0	1

We got the highest results with 60 neurons and 3000 epochs at the learning rate of 0.001

4) Sparse Extreme Learning Machine (S-ELM)

Extreme Learning Machine is characterized by its fast learning. This algorithm works on a single layer feedforward neural network where the input weights and hidden neuron biases are randomly initialized and the output weights are analytically calculated using the Moore-Penrose Pseudo inverse. This is very fast because there is no back propagation involved and a single matrix inverse operation gives you the weights. We have implemented the sparse extreme learning machine described in the paper: “Performance enhancement of extreme learning machine for multi-category sparse data classification problems” by S. Suresh, S. Saraswathi, and N. Sundararajan. The ELM performs much better if we optimize its parameters. We vary the number of neurons from 5 to 50, in steps of 5 and for each given number of hidden neurons, we initialize the input weights and hidden neuron biases randomly 5 times. We find the best input weights and hidden neuron biases by using 5 x 2 cross validation. The average accuracy over these 10 (5 x 2) runs gives us the generalization accuracy for a specific combination of input weights and hidden neuron biases for a specific number of hidden neurons. We use this accuracy as a measure to select the best combination and the optimal number of hidden neurons. Using the best combination of input weights and hidden neuron biases, we train the network and find the output weights. The ELM algorithm, as given in the paper mentioned above, is:

1. Randomly select a set of hidden neurons [H_1, H_2, .. ,H_p].
2. For a given H_i, use 5 x 2-fold cross-validation to select W, B.
3. Develop a classifier model using the best W and B and calculate the training and cross-validation efficiencies.
4. Repeat the steps 2 and 3 for different values of H_i; i = 1,2, .. ,p and select the H_i for which the training and cross-validation efficiencies are high.
5. For the best H, W and B calculate the testing efficiency

The following table has details for the various number of hidden neurons, different input weights and hidden neuron bias combinations. For each number of hidden neurons, the 5 rows correspond to different input weight and hidden neuron bias initializations. The rows highlighted in yellow denote the best weights and bias initialization for the given number of hidden neurons. The row in green shows the optimum parameters.

Number of hidden neurons	Training accuracy	Testing accuracy		Number of hidden neurons	Training accuracy	Testing accuracy		Number of hidden neurons	Training accuracy	Testing accuracy
5	0.3375	0.3713		25	0.51458	0.44522		45	0.55764	0.44304
5	0.34458	0.3713		25	0.51278	0.44522		45	0.56347	0.46348
5	0.26069	0.30783		25	0.53028	0.46304		45	0.55847	0.4613
5	0.30875	0.35087		25	0.50278	0.42391		45	0.55417	0.44435
5	0.2575	0.31957		25	0.53403	0.45565		45	0.56014	0.44696
10	0.33764	0.35		30	0.52611	0.42652		50	0.57431	0.47174
10	0.38417	0.38565		30	0.53681	0.45957		50	0.58542	0.4713
10	0.37972	0.36783		30	0.52528	0.4413		50	0.57306	0.45913
10	0.36694	0.36957		30	0.55847	0.45217		50	0.57542	0.47087
10	0.40014	0.39783		30	0.51319	0.44		50	0.57208	0.45957
15	0.46444	0.41261		35	0.53889	0.44652				
15	0.45167	0.39957		35	0.53222	0.44348				
15	0.44431	0.41826		35	0.54986	0.46217				

15	0.46278	0.43565		35	0.53639	0.47				
15	0.41222	0.39652		35	0.55903	0.44652				
20	0.49528	0.4187		40	0.545	0.45				
20	0.49778	0.43609		40	0.56028	0.45652				
20	0.47736	0.43783		40	0.54569	0.47478				
20	0.47625	0.39913		40	0.57097	0.4613				
20	0.47889	0.4313		40	0.54389	0.47				

Thus, we get optimum number of hidden neurons as 40. These accuracies are quite low because we break our already small training data into smaller sets for cross validation. But it gives better accuracy when trained with all the training data, as shown in table 1. The algorithm also gives slightly different accuracies on different runs because initially the weights are initialized randomly.

Results:

Algorithm	No. of hidden neurons	No. of epochs	Overall accuracy		Average Accuracy		Geometric accuracy	
			Training Accuracy	Testing Accuracy	Training Accuracy	Testing Accuracy	Training Accuracy	Testing Accuracy
RBF without gradient descent	80	-	97.917	82.209	97.778	79.643	97.594	60.318
RBF with Gradient Descent	100	1000	100	82.435	100	80.69	100	62.311
MLP with LS loss function	60	2000	73.611	60.87	66.071	52.381	0	0
MLP with Cross entropy	50	3000	97.917	86.957	97.321	79.762	97.179	0
Sparse - ELM	40	-	94.444	80.435	92.857	77.381	92.06	70.974

Table 1

Thus, RBF with/without gradient descent, MLP with cross entropy and Sparse-ELM give good results for classification of cancers based on the sparse GCM dataset. The geometric accuracy is very difficult to improve because there are only 144 training samples for classification in 14 classes. In that sense, S-ELM gives best performance because it has the highest geometric accuracy. Thus, we can classify such sparse data with good accuracy by using algorithms like RBF and ELM.

References

1) Vasily Sachnev, Saras Saraswathi, Rashid Niaz, Andrzej Kloczkowski and Sundaram Suresh Multi-class BCGA-ELM based classifier that identifies biomarkers associated with hallmarks of cancer. BMC Bioinformatics. 2015 16:166
2) S. Suresh, S. Saraswathi, and N. Sundararajan. 2010. Performance enhancement of extreme learning machine for multi-category sparse data classification problems. *Eng. Appl. Artif. Intell.*23, 7 (October 2010), 1149-1157. DOI=http://dx.doi.org/10.1016/j.engappai.2010.06.009
3) Basic ELM code taken from http://www.ntu.edu.sg/eee/icis/cv/egbhuang.htm.
   Authors: MR QIN-YU ZHU AND DR GUANG-BIN HUANG