Feed-forward Neural Networks with Genetic Algorithm and Network Reduction: A Case Study of Dermatologist-level Classification of Skin Cancer

Qian Wang

Research School of Computer Science, Australian National University

U6014726@anu.edu.au

Abstract. Skin cancer, which is mainly diagnosed visually, is the most common human malignancy. Diagnosis of skin lesions is a challenging task owing to the fine-grained variability in the appearance of skin lesions. Neural Networks (NNs) show potential for general and highly variable tasks across many fine-grained object categories. This paper implements a 3-layer feed-forward NN as a classifier for differential diagnosis of Erythemato-Squamous diseases. The genetic algorithm (GA) is employed to generate high-quality features used as inputs for the classifier, while by the network reduction technique excess hidden neurons are automatically detected and reduced after training. A dermatology database from UCI repository is utilized to train and test on this classifier. Our classifier performs a similar test accuracy solution (95.3%) compared with the work of Gvenir, H., etc. in 1998 (96.2%) on the same dataset, with efficacy improved on both storage and running time.

Keywords: Genetic algorithm, neural network reduction, differential diagnosis, feed-forward neural networks

1 Introduction

Skin is a highly dynamic tissue with continually renewing epidermis and ever-surveilling immune cells [1]. The differential diagnosis of some erythemato-squamous diseases is a significant problem in dermatology. This is the case with psoriasis, seborrhoeic dermatitis, lichen planus, pityriasis rosea, chronic dermatitis, and pityriasis rubra pilaris [2]. Difficulties for the differential diagnosis contain that these diseases may share similar clinical features, and may exhibit the characteristics of another disease at the beginning phase. Regarding the treatment, these diseases which are hard to distinguish visually may not have the same pharmacology. In some cases, wrong prescriptions may aggravate patients' conditions. Under these circumstances, finding out an efficient way to help diagnose dermatology diseases has become a crucial task.

The Dermatology Data Set (DDS) [3], which is available on the UCI repository, presents histopathological and clinical attributes (12 clinical and 22 histopatological features) data from 366 patients with dermatological conditions, specifically the erythemato-squamous diseases [2]. Concerning its sufficient patients' information and detailed collected pathological features, this dataset is widely used for the erythemato-squamous diseases classification. Some related dermatology classification work has been done on this dataset. For example, in [6], attempts of classifying skin diseases on DDS using decision trees are outlined, with a 5.5 ± 1.46 error rate. Three multi-classifier approaches are performed in [10], which are on the same dataset with an average accuracy of more than 98%. In [4], which is the paper we chose to compare with, a classification algorithm VFI5 (for Voting Feature Intervals) is developed and applied to this problem to differentiate a new case in the same domain.

The primary purpose of this study is to determine a classifier which shows higher accuracy on DDS but less storage and running time consumed, and which has the potential to be selected as an aid to avoiding misdiagnosing erythemato-squamous dermatological diseases. To build the classifier, we created a 3-layer feed-forward NN trained by back-propagation. We adopted GA to select optimized features as inputs of the NN, avoiding classification results being affected by noisy or irrelated data. We implemented a network reduction technique from one of Gedeon's papers [5] after the network training process completed. The main idea of this technique is to detect excessive hidden neurons and drop them. By utilizing this classifier, we received an average predicted accuracy of 95.3% on DDS, with 50% input-layer storage, 38% hidden-layer storage, 37% training time and 45% test time saved.

In this report, Section 2 introduces the data we chose from DDS and how we preprocessed the data before using it on our classifier. Section 3 shows principles and implementations of methods we used to build our classifier, while experiments and compared results are discussed in Section 4. In Section 5, we conclude our work, point out the limitations and describe our future work.

2 DDS and Data Preprocessing

In DDS, 34 features are included for diagnosis, along with 366 patients' conditions. In the range of this domain, the value of the family history feature is 1 if any of these diseases are observed in the family. Otherwise, it is

set to 0. Except for the family history feature and the age feature (which is linear with patients' actual age), all other features receive a degree from 0 to 3. In this case, 0 means that this feature does not exist, 3 represents the maximum number of possibilities, while 1 and 2 are relative intermediate values.

In some real-world domains, there always exist some missing values in the dataset, and that applies to the DDS as well. This dataset involves eight missing values describing the age of patients, which needs to be removed or fixed for classification. In the paper which we chose for results comparison, these missing values are automatically ignored by their algorithm, without making any sense. Since missing values need to be deleted or completed, referring to the compared paper, we decide to remove those eight missing values manually. Therefore, the current dataset consists of 358 cases, expressing the dermatological conditions of 358 patients with confirmed diseases. We adopt each column in the dataset as proposed inputs except the last one. These columns represent 34 features of the patients who are suffering erythemato-squamous diseases, while the last column identifies the diagnosis of each patient. We randomly separate the processed data into two groups, namely the training set and the test set, with the scale of 0.8 and 0.2 respectively.

3 Principles and Implementations of the DDS Classifier

3.1 Feature Selection Using Genetic Algorithm

In a dataset, usually not all the features make equivalent contributions to the classification accuracy. Some features may be irrelative and may cause unnecessary biases. To prevent the prediction accuracy from being affected by those features, feature selection needs to be adapted to generate appropriate inputs before training the data. An exhaustive selection of features would evaluate lots of different combinations (2^{34} on DDS, where 34 is the number of features in DDS), which requires lots of computational work and is time-consuming. Therefore, we need inventive methods that allow performing feature selection in practice, one of which is GA.

In the early 1970s, John Holland, one of the founders of evolutionary computations, introduced the concept of GA, which is a heuristic approach used to find approximate solutions to solve problems through application of the principles from evolutionary biology, relying on biologically-derived techniques such as mutation, crossover and selection [11]. Combining GA with NNs can overcome several problems, such as the local limit value and the slow rate of convergence, thereby efficiently increasing the training speed of networks.

In this report, we implement GA to filter and optimize the input features before building our NN. The first step is to generate an initial population of individuals randomly. The numerical experiments show that increasing the size of the population of 5 to 30 chromosomes improves the test accuracy on DDS - from 88.6% to 96.1% on average. The further increase in the size of population (more than 40 chromosomes) leads only to an increase in computational time without test accuracy improved. Considering the consumed time and the test accuracy on DDS, we set the population size to 30.

After that, the performance of the population is evaluated by using an individual fitness function. Since we focus on the dermatology classification task, we use classification accuracy as our fitness function. We rank these individuals based on their fitness values, randomly pick members of the population to produce more like them, under the principle that individuals with higher ranks have more chances to be chosen. In the crossover process which plays a role in creating new individuals from old ones, we exchange 80% of the information between two parents for generating new individuals, where parents are randomly decided from the previously produced individuals. After that, we select an individual in the new population and alter some values in it, based on a mutation rate of 0.002. The crossover rate (0.8) and mutation rate (0.002) are determined by sensitivity analyses: we carried out multiple runs with different rates and compared the outcomes. After ten generations (further number of generations leads to increasing time without any improvements), with a series of iterative calculations with these operators, the optimal set of features is filtered.

3.2 Network Structure

In this report, we implement a 3-layer fully-connected feed-forward NN, which allows flexibility in dataset size and could continually learn as more data is added to the dataset. Our NN consists of an input layer, a hidden layer, and an output layer. The input layer contains optimized neurons we generated by GA, representing those best-fitted features in DDS. The output layer includes six neurons that determine the six erythemato-squamous diseases. The hidden layer consists of 51 hidden neurons (150% of maximum input features), the number of which was chosen from personal, prior knowledge of neural networks. The problem predetermines the number of input and output neurons whereas the number of hidden neurons can vary wildly, thus requiring further examination. All connections are from units in one level to the subsequent one, with no lateral, backward or multilayer connections [5].

After initializing our NN, 80% of preprocessed data then trains our feed-forward NN for 500 epochs through sigmoid activation function, which introduces non-linearity into our neural network model and normalizes the

output to [0,1]. We accept cross-entropy loss function for our network, for the reason that it performs better than other loss functions for discrete problems such as classification. Back-propagation is utilized for the process of weight updating. The Rprop optimization algorithm is employed in our network with a learning rate of 0.01, which is used as an iterative method for minimizing the loss function. It is one of the fastest weight update mechanisms in which the network is modified after each training sample is fed through the network and thus updating the network parameter in the direction in which the loss is minimized. Besides, for the reason that our dataset is not large enough (358 data in total), we adopt batch with the size of the whole dataset to update weights.

3.3 Network Reduction Based on Distinctiveness

Our NN is now implemented, using the selected features by GA as inputs. However, a network reduction technique based on distinctiveness [5] can be applied to our NN to optimize further the time and space usage, which has been proven to be particularly useful for determining an appropriate neural network size. The critical idea of this technique is by dropping the hidden units which are identified as unnecessary to our network to provide an equivalent solution with a smaller size of the network and less testing time without retraining.

In the network reduction paper [5], four types of undesirable hidden units are provided. In this report, two of them are selected as the principles to detect excessive hidden neurons, which are (1) two or more units may be too similar, in that they produce the same output for each pattern, and (2) groups of units together having no function can be two units complementary to each other in that for any pattern their outputs are inverses of each other; alternatively three or more units may produce no effect [5].

To identify those hidden neurons which meet at least one of the two principles, after the data training activity completed, we construct one vector for each hidden unit with the same dimensionality. The length of the dimensionality is equal to the number of input patterns in the training set. Each component of the vector corresponds to the output activation of the unit. After constructing the vectors, angles between pairs of unit vectors are calculated. Instead of using the regular range of 0-90°, we normalize our vector angle calculations to (0.5, 0.5) to adopt the range of 0-180°. In our study, if the angle between two units is over 165°, we define them as complementary, and both of them are removed. For those pairs of units with an angle up to 15°, we consider them as too similar. With low angular separations, the averaging effect is insignificant, and the mapping from weights to pattern space remains adequate in that the error measure is no worse subsequently [5], which indicates that one of them needs to be deleted. Under this circumstance, the unit with a broader index is dropped, and the weight vector of the removed neuron is added to the remaining one.

We create two matrices to store the weight vectors of the remaining units, which involve the value of weights from the input to the hidden layer and the hidden to the output layer respectively. After dropping those unnecessary neurons, we build a new network automatically. The number of new hidden neurons is equal to the initial hidden neuron size minus the number of removed neurons, and the weights of the new network are equivalent to the values of the two matrices which consist of the weight vectors of the previous remaining neurons. After the new reduction network established, we do the test on the new network without retraining.

4 Experiment and Result Discussion

After having built our classifier, we evaluate it by passing the test data from DDS. The average classification accuracy, storage usage as well as training and test time are calculated on ten times tests. Since we separate the DDS into two subsets randomly, test sets for the ten runs are distinctive. Our network on the DDS has reached an average accuracy of 95.3%, with 1.960575s training time, 0.000614s test running time, 17 input neurons and 32 hidden neurons. Our experiments are compared with the test results of the traditional ANN, the reduced NN without accepting GA as feature selection, the GA improved NN without reduction, and the VFI5 classification algorithm in [4]. A line chart consists of test accuracies over ten runs for each model is shown at the end of this section, from which we could figure out that the average accuracies of these models are similar.

4.1 Result Comparison with Traditional ANN

We designed a base control experiment, where we utilized the traditional ANN as a classifier, without any pre-selected features or network reduction. All the hyperparameters are same except the number of the input neurons and hidden neurons, which equals to all the features in DDS (34) and the initialized number of hidden neurons (51) respectively. Ten runs are carried out using the traditional ANN, the average test accuracy of which is 95.38%, with 2.857632s training time and 0.001295s test time. We compared the results of our classifier with the traditional ANN, finding out that we reached a similar test accuracy with 37% training time and 52% test time saved. Since the traditional ANN employs all of the 34 features as input and all of 51 hidden neurons, our classifier saves 50% storage at the input layer and 38% at the hidden layer.

The possible reasons for this result may be that the features we generated by GA are of "high quality", by

which those making fewer contributions to the DDS classification accuracy are omitted; thus a similar accuracy reached with fewer input features used. Despite that, our classifier fails to improve the test accuracy as well, which might because no data in DDS is noise or could hurt the test accuracy, thus selecting features cannot help to improve the performance of test accuracy. Regarding the running time, reduced input neurons contribute to the training time decreased, because fewer features need to be trained. Another reason for the decreased time might be that we removed some redundant hidden neurons, which saves test time and space storage. Since the reduction is utilized after data training finished, no inappropriate action is made, and no useful neuron is dropped because of the insufficient training epochs, which in turn unable to hurt the accuracy.

To further exploring the influence of GA and network reduction on our classifier respectively, we did two further experiments on two distinct neural models, a classifier only processed with GA and a classifier only implemented with the reduction technique.

4.2 Result Comparison with ANN Processed by GA

We carried out ten runs with the neural model processed by GA without any reduction techniques applied, to see the influence of the distinctiveness network reduction technique on our classifier. All the other hyperparameters are placed the same between the two neural models. The average accuracy without reduction is 95.65%, with 0.000838s test time. We only consider these two issues here, for the reason that both of their input neurons are selected by GA, and training processes are the same, which are unnecessary to be calculated here.

From this experiment, we could figure out that the network reduction process saves 35.2% test time and 38% storage on the hidden layer, without losing the test accuracy. Since the test time is longer than using our classifier (0.000614s), we could interpret that the feature selection process by GA may contribute to the test time reducing as well. Besides, the test accuracy is still similar, which indicates those removed hidden neurons are redundant, thus deleting them will not lower the accuracy.

4.3 Result Comparison with ANN Processed by Network Reduction

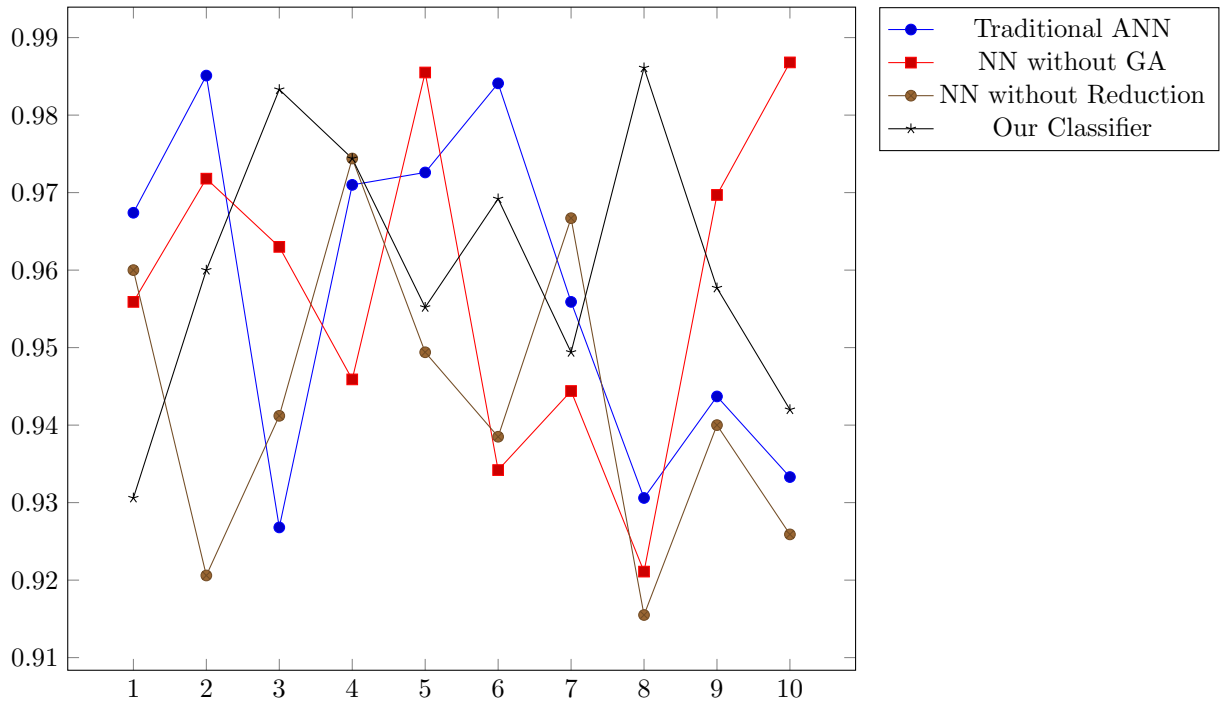
Another experiment is also taken, where the reduced ANN is used for ten runs on DDS. We developed this model to explore the influence of GA on our classifier. All the hyperparameters in this model are same as our classifier, except inputs of which is not selected by GA but adopting all features in DDS. Results of this experiment show that an average accuracy of 95.59% is reached, with the training time of 2.771871s and 0.000863s test time.

From the results, we could conclude that using GA as the feature selection method could save both the training time and the test time. Because GA selects those most relevant features as inputs, rather than the whole dataset, our neural model does not need to train or test the whole DDS, which saves both the training and the test time. However, the test accuracy is not improved after we applied GA, this may be due to the reason that there may not exist noisy features in DDS that could hurt the classification accuracy; hence the accuracy can hardly be improved.

4.4 Result Comparison with VFI5

Regarding comparing our results with others' related work in the same domain, we choose a paper [4] from Gvenir, H., etc. in 1998, in which they developed a classification algorithm VFI5 to determine a new case on the DDS. This study utilized all 34 features in the dataset to predict these six erythemato-squamous diseases. This work involved 366 instances in the dataset, missing values of which were ignored by their algorithm. In their study, they separated the DDS into ten disjoint subsets. They applied ten runs on the DDS, with each subset being the test set once. The performance of the VFI5 classifier was evaluated empirically concerning classification accuracy and running time [4]. This study reported the mean accuracy rates of 96.2%.

Our average test accuracy on DDS is 95.3%, which is close to the accuracy of the paper. However, using our classifier, we cannot ignore those instances with missing values, so we removed these instances manually. Similarly, for the idea that we randomly separate the DDS into the training set and the test set with a scale of 8:2, the disjoint testing subsets for the ten runs cannot be guaranteed. Additionally, although in the study they evaluate the performance of the VFI5 classifier based on the accuracy and running time, there is no sign in the paper talking about the exact running time; hence no comparisons could be taken on the running time. Even if the running time was shown, it should be noted that the VFI5 classifier utilized a CPU from 1998, which is different from ours. The distinctive computational power means that it is not wise to compare the training and test time.



5 Conclusion and Future Work

This report implemented an NN classifier for the diagnosis of six erythemato-squamous diseases, with GA to generate input features and the distinctiveness network reduction to reduce hidden neurons. By applying GA and the reduction technique, we reached a similar accuracy (95.3%) compared with traditional ANN and the VFI5 classifier, with 50% input layer and 38% hidden layer storage saved. We also reduced 37% training time after applying GA, and the test time of our classifier is cut in half, with the help of both GA and network reduction. In this study, the high accuracy (95.3%) of prediction indicates that this classifier could be utilized to aid and enrich the dermatology diseases diagnosis, which is difficult to differentiate visually. It also means that patients could have a strong chance of being treated adequately with the accurate classifier.

Additionally, this classifier only provides similar performance on test accuracy without any improvements, which may be due to the reason that no features in DDS is useless or could hurt the classification. However, on some large datasets with non-negligible noise which affects the accuracy of test predictions, this classifier is expected to present better performance, with a higher test accuracy but less running time and storage needed.

Since our classifier made no contributions for improving the test accuracy and taking the characters of DDS into account (small size without significant noise), GA and network reduction may not be an ideal choice for DDS. Besides, although our experiments reached an average test accuracy of more than 95%, the performance of our classifier is not stable, and sometimes it is deficient (such as 70% accuracy). Even though this condition does not happen a lot, to aid the diagnosis of dermatology diseases, such kind of error is not permitted. Hence in the future, for this domain, rather than improving our classifier based on the current structure, we may consider other possible methods such as decision trees or k-means clustering.

This classifier may be applied to some other specific datasets with relatively larger size and non-eligible noise. However, with the size of dataset increases, the training time is growing, which means this classifier may get stuck and cannot deal with datasets in significant large size. For these cases, further improvements will be considered, such as changing the NN structure and trying some other powerful time-saving methods.

References

1. Tanaka, R. and Ono, M. (2013). Skin Disease Modeling from a Mathematical Perspective. *Journal of Investigative Dermatology*, 133(6), pp.1472-1478.
2. Barreto, A. (2014). Multivariate Statistical Analysis for Dermatological Disease Diagnosis. *IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*.
3. Bache K. and Lichman M. (2013) UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.
4. Gvenir, H., Demirz, G. and Iter, N. (1998). Learning Differential Diagnosis of Erythemato-squamous Diseases Using Voting Feature Intervals. *Artificial Intelligence in Medicine*, 13(3), pp.147-165.
5. Gedeon, TD. and Harris, D. (1991). Network Reduction Techniques. *Proceedings International Conference on Neural Networks Methodologies and Applications*, AMSE, vol. 1, pp. 119-126, San Diego.

6. Pappa, G. L., Freitas, A. A. and Kaestner, C. A. (2002, November). Attribute Selection with a Multi-objective Genetic Algorithm. In *Brazilian Symposium on Artificial Intelligence* (pp. 280-290). Springer, Berlin, Heidelberg.
7. Fidelis, M.V., Lopes, Heitor S. and Freitas, Alex A. (2000) Discovering Comprehensible Classification Rules with a Genetic Algorithm. In: *Evolutionary Computation*, 2000. Proceedings of the 2000 Congress on. IEEE, La Jolla, CA, USA pp. 805-810. ISBN 0-7803-6375-2. (doi:https://doi.org/10.1109/CEC.2000.870381)
8. Schiavo, R. A. Two Methods For Pruning Neural Networks: A Performance Evaluation.
9. Theclevermachine.wordpress.com. (2018). Machine Learning — The Clever Machine. [online] Available at: https://theclevermachine.wordpress.com/tag/machine-learning/ [Accessed 29 Apr. 2018].
10. Groccia M.C., Guido R., Conforti D. (2017). Multi-Classifier Approaches for Supporting Clinical Diagnosis. In: *Sforza A., Sterle C. (eds) Optimization and Decision Science: Methodologies and Applications.* ODS 2017. Springer Proceedings in Mathematics & Statistics, vol 217. Springer, Cham
11. Karthigayan, M., Rizon, M., Nagarajan, R., & Yaacob, S. (2008). Genetic Algorithm and Neural Network for Face Emotion Recognition. In *Affective Computing.* InTech.