Neural network classification on mushroom dataset with feature selection using evolutionary algorithm and auto-associative network

Yuhan Zhang, u6007076@anu.edu.au, canberra, australia

Abstract. The paper tries to determine if certain species of mushroom is poisonous based on the mushroom's physical features using an artificial neural network. This paper illustrated how to use evolutionary algorithm to do features selection on mushroom dataset, and compares the classification result with a standard neural network as well as a standard neural network with inputs compressed by an auto-associative network. Genetic algorithm is used with a hall of fame selection method. The genetic algorithm determines which subset of features has the best fitness as the input for mushroom type classification. Deap, an evolutional computation framework written in python, is used to implement the evolutionary algorithm in the paper. Inspired by auto-associative network for image compression, the paper employed auto-associative network to do a compression on the input the data, and then feed the compressed representation of the input data into a three layers neural network. Pytorch, a highly popular deep learning library written in python and maintained by facebook, is used to implement the auto-associative network and neural network classifier. This approach achieved a better result than feeding the input data naively into a three layer neural network without encoder to compress the data, which demonstrated that a compressed representation of input data will achieve better result than naïve input data. The paper also talks about other applications of auto-associative network and genetic algorithm.

Keywords: neural network, data compression, data preprocessing, auto-associative network, evolutionary algorithm, genetic algorithm

Introduction

Whether a mushroom is edible could sometimes be determined by observing its physical characteristics. The mushroom dataset provides a set of data with 8124 instances of mushrooms and their respective physical attributes recorded, also its edibility. By employing a machine learning method the inherent links between these physical characteristics and its edibility can be investigated. A model could be trained to predict the edibility of a certain instance of mushroom, based on the physical traits that it presents. Auto-associative network tries to apply a dimensional bottleneck to recreate a compressed representation of the input data (Kramer, 1992). This is done through a model trained to approximate identity mapping between its input and output, taking the hidden layer activation as the compressed representation.

This paper use genetic algorithm to determine which subset of features yield best classification result. Genetic algorithm is a technique in artificial intelligence encompasses the ideas of genetic mutation, recombination, and survival based on fitness (Butterfield & Ngondi, 2016). Genetic algorithm belongs to the larger class of evolutional computation (tutorialspoints, n.d.). In a genetic algorithm, a candidate population pool is maintained and evolved towards an overall better population. During the evolution crossover and mutations are applied when parents reproduce offspring. Some research (Eiben, 1994) has shown that more than two parents could generate offspring with higher quality chromosomes. A set of selection functions are used on each generation of population where most of them are used fitness value as criteria to allow individuals with best fitness to survive to next generation and has chance to breed offspring (Engelbrecht, 2002). By applying a genetic algorithm to extract an optimal set of feature from original dataset the size of neural network could be reduced, as fewer number of input neurons will be required in the input layer of neuron network.

Data set selection

The mushroom dataset is retrieved from UCI machine learning dataset repository (National Science Foundation). It contains descriptions of mushroom samples related to 23 different mushroom species of gilled mushrooms from the Agaricus and Lepiota Family. In term of mushroom edibility, the mushrooms are described as edible and poisonous. The mushroom dataset provides 8124 instances of mushrooms corresponding to 23 mushroom species, with regard to its 22 physical traits. It can be identified that These physical traits described the shape, size and color of mushroom's different physical parts like stalk, gill and cap. Some features like odor and habitat are also described. The values of these features are discrete labels. In the dataset there are 8124 entries, each corresponding a mushroom instance. Each instance is described with respects to 22 features. The value of those attributes in each instance is represented by the first letter of the label, for example, bell = b. The dataset was chosen because it has a respectable number of instances and attributes. The dataset is divided into training set and testing set, with the number of instances of mushrooms in training set being 6107, while the number the mushroom instances in the testing set being 2017, roughly a 3 to 1 split. The neural network

classified will be trained on the training set, and tested on the testing sets. The mushroom data instances in the testing set will not be feed into the neural network classifier before the testing phase begin.

The data set has 22 features, by observing if can be inferred that some groups of feature are correlated. Stalk color above the ring and stalk color below the ring may have some correlation as they are both properties relating to the stalk color of a certain type of mushroom. By using genetic algorithm maybe one of those two properties could be pruned if only model trained and tested using only the other properties yield a better accuracy and performance than having all two properties as input features. The nature that the dataset features are both diverse (22 features) and include features that potentially have inner correlation provides the ground for a genetic algorithm to do feature selection.

Model architecture design

A three-layer neural network is adopted to solve this classification problem. The topology of the neural network for classification of neurons can be described as follows: The neural network will be consisted of three layers, input layer, hidden layer and output layer. The number of neurons in the input layer will equal to the number of attributes each instance of mushroom has, which is 22 in the instance mushroom dataset. The number of neurons of hidden layer could be adjusted until the best performance is reached, using approach to determine neural network topology described in the paper network reduction techniques (T.D.Gedeon), the approach adopted in this paper is starting with a small amount of hidden neurons, for example, 5, and the case is that the neuron network will fail to learn, then increase the number of neuron by 1,2. Finding the net still could not learn very well, the number of hidden neurons is increased further by 10%. Finally the ideal size of hidden neurons found using this progressive approach is 50. For the output layer, the number of output neurons equates the number of classes that the mushroom is divided two. In this case, the number is 2(edible, poisonous). Full batch is used, which means the input data will have a batch size which is equal to the number of instances of mushrooms in the training dataset, in this case, is 6107. Standard feedforward and backpropagation approach is used to train the neural network, the cross-entropy loss is computed, and based on that loss, the gradients of all the weights existed in the network can be computed and deducted from those weights after multiplied by the learning rate. The neural network gradually converges after 2000 epochs of training as its weights being continued adjusted and loss being continuously minimized. Pytorch, the deep learning framework used in this paper, encapsulates a lot of math operations into methods, reducing the complexity of implementing a deep neural network architecture. The algorithm for feed forward and backpropagation training can be illustrated by the code below, which is taken from code repo for this paper.

```
#define neural network metadata
input_neurons = 22
hidden_neurons = 100
output_neurons = 2
learning_rate = 0.01
num_epoch = 1000
#initialize neural network
net = TwoLayerNet(input_neurons, hidden_neurons, output_neurons)

#define loss function
loss_func = torch.nn.CrossEntropyLoss()

# define optimiser
optimiser = torch.optim.SGD(net.parameters(), lr=learning_rate)

# store all losses for visualisation
all_losses = []

# start neural network training
for epoch in range(num_epoch):
    # Perform forward pass: compute predicted y by passing x to the model.
    Y_pred = net(X)

    # Compute loss
    loss = loss_func(Y_pred, Y)
    all_losses.append(loss.data[0])

    # Set all gradients to zero before computing gradients for this epoch
    net.zero_grad()

    # Calculate the gradients based on loss computed
    loss.backward()

    # Calling the step function to substract the quotient of learning rate and gradients from the weights
    optimiser.step()
```

Code Snippet 1. Feed forward and backpropagation algorithm implemented in pytorch framwork

The paper tries two major approaches to increase the prediction accuracy. One is to use auto-associative network to condense each input data instance into a compressed hidden representation, and use this hidden representation as input feeding into the neural network instead of the original inputs.

The auto-associative neural network has a standard three layers neural network topology. The difference between the auto associative network and a neural network classifier is that its input and output vectors are identical (Kramer, 1992). In an attemp to reconstruct the input data through a hidden layer, the hidden activation is taken which represent a compressed representation of original input.

```
Y_encoding = encoder(X)
Compressed_input = encoder.compressed
Y_pred = net(Compressed_input)
_, predicted = torch.max(F.softmax(Y_pred)
```

Code Snippet 2. Hidden activations taken as the compressed representation of the original input.

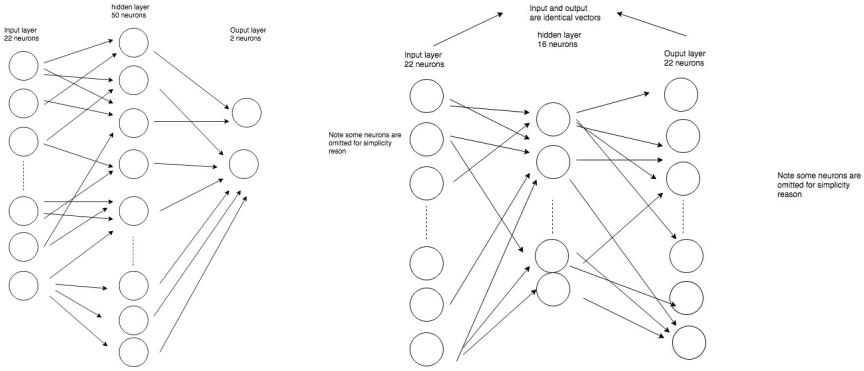

Fig. 1. Three layers neural network (left) and auto-associative network(right) topology.

The auto-associative network is firstly trained on the mushroom dataset until loss is minimized. Then classifier A is trained on the training set of mushroom data for 1000 epochs. At the same time, the training data along are fed into the encoder (auto-associative network), and hidden activations taken from the encoder along with the corresponding labels are fed into classifier B for training purpose. The training for classifier B will also last 1000 epochs.

After the training for both two classifiers complete, they are both tested on the testing dataset of mushroom data with the only difference that the input are firstly compressed by the encoder before being fed into classifier B for testing purpose. Then the accuracy and performance of two classifiers are compared. The training and testing process of the architecture can be illustrated by the figure below.

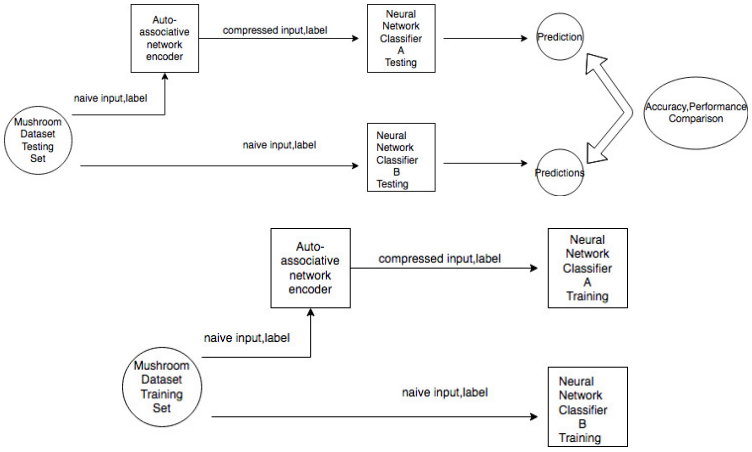

Fig. 2. Illustration of the training and testing process of the architecture.

The second approach is to use genetic algorithm to extract a subset of features from the original 22 features. Training and prediction on this subset of features may outperform the original dataset with full input features. Here an initial candidate population is initialized with chromosomes for each individual candidate being represented as a bit string.

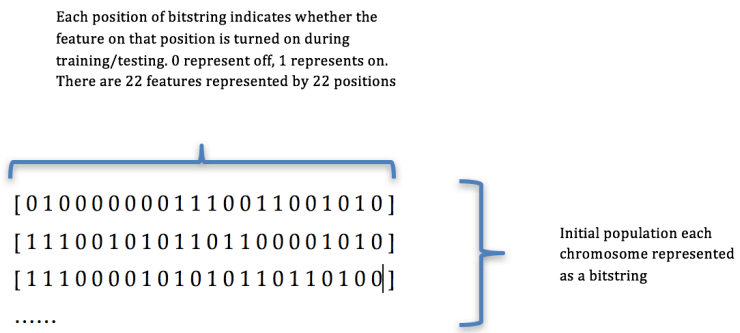

Fig. 3. How feature selection gets encoded in genetic algorithm and initialization of population.

As the goal of feature selection is to select an optimal subset of features for training and testing, the chromosomes represent whether each feature in original dataset is turned on. After the initialization of population, those candidate population experiences reproduction with crossover and mutation, where a batch of offspring gets generated. Then offspring together with their parents becomes subject to selection based on fitness value. Here the selection methodology adopted is hall of fame, where in each generation best performing individuals are memorized and they form a pool of parents for crossover in the following generations (Nogueira, 2013). With each generation having population with a higher fitness value, the algorithm will stop if the difference between the sum of fitness value between two generation drops below a threshold.

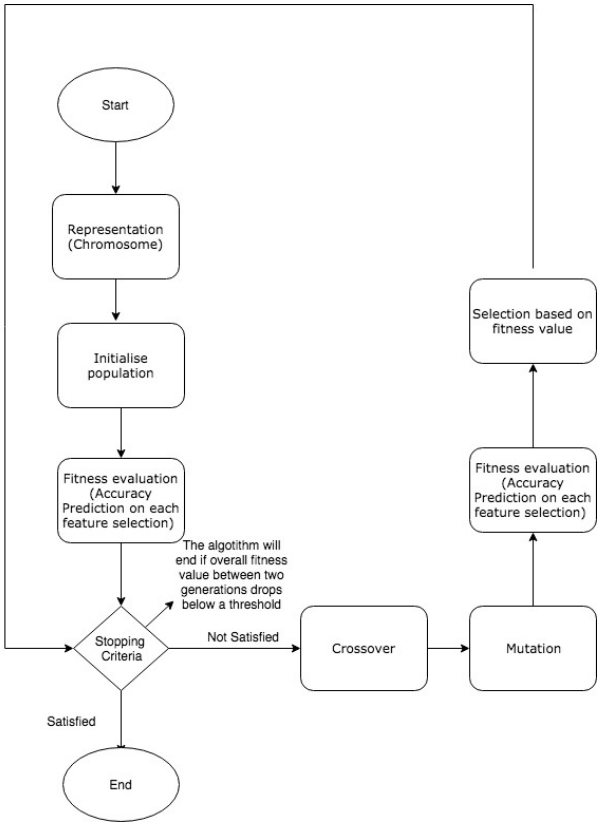

Fig. 4. Genetic algorithm for feature selection architecture design.

The aim of feature selection is to select an optimal set of features which produces highest prediction accuracy, so the fitness function adopted by our genetic algorithm is the accuracy of classification. To be more specific, the individual whose chromosome represents a features combination which produce higher prediction accuracy when that combination of features is used for training and testing has higher fitness value, which means it is more probable that this individual will survive in the selection phase of the genetic algorithm. There are other selection algorithms such as random selections where fitness value is neglected. Those selection methods are not suitable for this paper as a best performing feature set needs to be selected.

```
# Initialize variables to use eaSimple

 number_of_population = 100

 number_of_generation = 10

 pop = tb.population(n=number_of_population)

 hall_of_fame = tools.HallOfFame(number_of_population * number_of_generation)

 stats  = tools.Statistics(lambda individual: individual.fitness.values)

 #Reporting corresponding statistics in the computing process

 stats.register("average", numpy.mean)

 stats.register("standard", numpy.std)



 # Launch genetic algorithm

 pop = algorithms.eaSimple(pop, tb, cxpb=0.5, mutpb=0.2, ngen=number_of_generation, stats=stats, halloffame=hall_of_fame, verbose=True)
```

Code Snippet 3. Get Hall of fame population.

Here the number of generation in genetic algorithm is set to 20 and number of population is set to 200. The crossover probability is set 0.55 and mutation probability is set to 0.3. The deap framework already encapsulates most operations for the genetic algorithm to work, by simply calling the interface the algorithm will finally produce a hall of fame population with the best prediction accuracy.

Finally, it is important to note that for computation efficiency reason the classifier used in the fitness function of feature selection is Logistic Regression, which would produce a better predicting accuracy than neural network in less amount of computational time.

Model evaluation and result

The 8124 instances of data are split into two parts, training set, which contains 6750 instances of mushrooms, and testing set, 1374 instances. They are fed into the neural network and softmax is performed on the output result.

```
# convert three-column predicted Y values to one column for comparison
       _, predicted = torch.max(F.softmax(Y_pred), 1)

       # calculate and print accuracy
       total = predicted.size(0)
       correct = predicted.data.numpy() == Y.data.numpy()

       print('Epoch [%d/%d] Loss: %.4f  Accuracy: %.2f %%'
            % (epoch + 1, num_epoch, loss.data[0], 100 * sum(correct)/total))
```

Code Snippet 4. Softmax on prediction result and accuracy calculated

The output of softmax is then compared to the correct class label to calculate the accuracy.

	Without auto-encoder	With auto-encoder
Predicting Accuracy	70%	65%

Fig. 5. The performance evaluation for classifier with anto-encoder and classifier without auto-encoder when training number of epoch for encoder is 2000

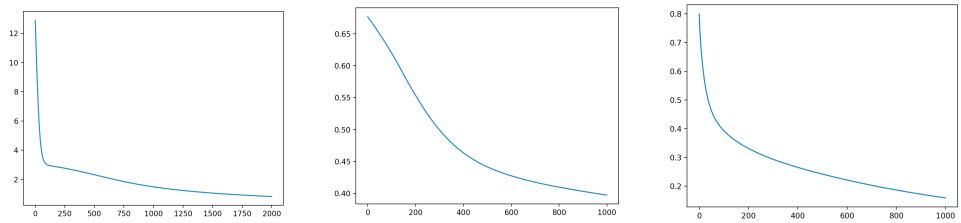

Fig. 6. When training number of epoch for encoder is 2000. The loss decreases as number of epochs increases, for encoder net(left), classifier A(middle), classifier B(right), respectively

From the evaluation result, it can be seen that when encoder is not sufficiently trained, in this case 2000 epochs, classifier with auto-encoder has less prediction accuracy than classifier without auto-encoder, with an average predicting accuracy of around 66%.

	Without auto-encoder	With auto-encoder
Predicting Accuracy	70%	71%

Fig. 7. The performance evaluation for classifier A and classifier B when training number of epoch for encoder is 8000.

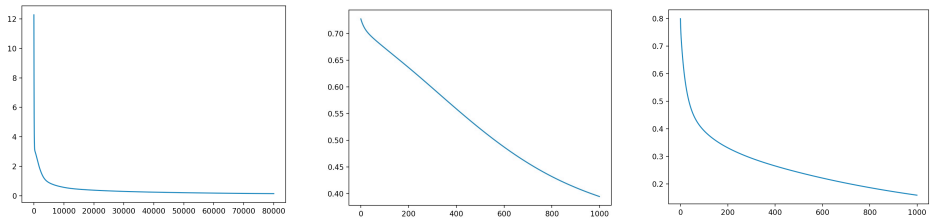

Fig. 8. When training number of epoch for encoder is 2000. The loss decreases as number of epochs increases, for encoder net(left), classifier A(middle), classifier B(right), respectively

When encoder is trained for 8000 epochs, the loss of encoder is sufficiently minimized. At this time, the classifier with auto-encoder has a slightly better prediction accuracy than classifier without auto-encoder, with an average accuracy of 71%. This proves that the approach of using an auto-associative encoder to compressed input for training can improve prediction accuracy.

The classifier with auto-encoder takes a little bit longer than classifier B to train, this is the joint force from two aspects: One is that the input must be passed to an encoder before feeding into the classifier for training. This encoding process

takes time. The aspects is that because the input is compressed, this means the number connections between input layer and hidden layer in classifier with auto-encoder is less than that of classifier without one. The smaller network in makes it faster to be trained. However, the first force overtakes the second, concluded from the experiment result.

After genetic algorithm is run on the initialized population with a size of 200 for 20 generations, the individuals with best validation accuracies are selected from the hall of fame population as the candidate for the final feature selection strategy. the algorithm in this paper is designed in a way that the individual with largest number of features will be selected. The consideration behind this design principle is to maintain the original feature set to the largest possible extent. Among the four best performing individuals in the hall of fame, the individual which is selected as our feature selection strategy is the feature subset of { cap-shape,cap-surface,cap-color,bruises,odor,gill-attachment,gill-spacing,gill-size,gill-color,stalk-shape,stalk-root,stalk-surface-above-ring,stalk-surface-below-ring,stalk-color-below-ring,veil-type,veil-color,spore-print-color,population,habitat }, in other words, the feature subset of { ring-number, ring-type, and stalk-color-above-ring } is pruned from the origin feature set.

The training data and testing data are shuffled, but still retaining the train-test proportion applied in auto-associative network testing. Then the prediction accuracy between neural network with full feature set and neural network with feature selection using genetic algorithm are compared to identify if feature selection leads to a better prediction accuracy. The two neural network have a similar topology, with the only difference that the number of input neurons for the one with feature selection is 19 neurons. The two neural networks have same number of hidden layer neurons.

	With feature selection	Without feature selection	With Auto-associative Network
Predicting Accuracy	77%	70%	71%

Fig. 9. Comparison of prediction accuracy between neural network with feature selection on dataset and neural network without feature selection on mushroom dataset.

It can be seen that after feature selection, the predicting accuracy of neural network has been improved significantly. It can also be concluded from the experiment that applying a feature selection strategy using genetic algorithm yields better prediction accuracy than feeding a compressed representation as input using auto-encoder.

Conclusion and future work

An auto-associative network could be used to reduce the dimensionality of the input data. This compression mechanism could speed up classifier training and reduce the size of classifier. An increase in classification accuracy can also be identified in the classifier with compressed representation as its input. Feature selection leads to an optimized neural network size, and better predicting accuracy. Feature selection is a more effective prediction accuracy improvement strategy than auto-encoder. Some future work could be done when putting encoder could be located another physical software than the classifier. When the dataset is very large, this could potentially help speed up training by putting the encoder and classifier on different locations in a network and let the compressed data traverse the network, making encoding and classifier training two parallel processed. A drop out layer could be added to classifier to improve the classifier performance.

Bibliography

Butterfield, A., & Ngondi, E. G. (2016). *A Dictionary of Computer Science* . Oxford University Press.

Eiben, A. E. (1994). Genetic algorithms with multi-parent recombination. *Proceedings of the International Conference on Evolutionary Computation*.

Engelbrecht, A. P. (2002). *Computational intelligence : an introduction*.

Kramer, M. (1992, 4). Autoassociative neural networks. *Computers & Chemical Engineering, 16*(4).

National Science Foundation. (n.d.). *UCI Machine Learning Repository: data sets*. Retrieved from https://archive.ics.uci.edu/ml/datasets/Mushroom

Nogueira, M. e. (2013). An Analysis of Hall-of-Fame Strategies in Competitive Coevolutionary Algorithms for Self-Learning in RTS Games. (pp. 174-188). Springer, Berlin, Heidelberg.

T.D.Gedeon, D. (n.d.). network reduction techniques.

tutorialspoints. (n.d.). *Genetic Algorithms Introduction*. Retrieved from https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_introduction.htm