

Coursework Report

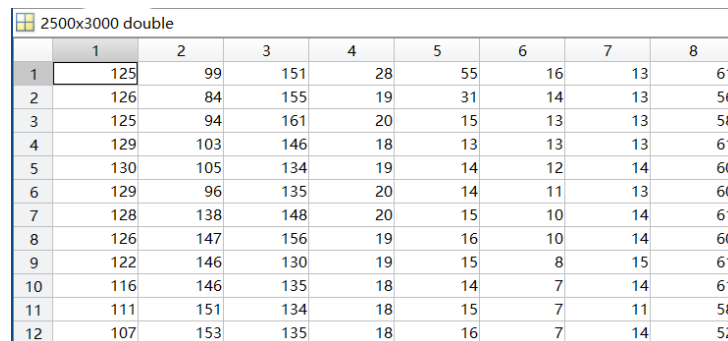
Chen Liao

20030694

Data Structure & Network Configuration

The main dataset structures are two matrixes named X and y. For X, this matrix has a size of $n \times 3000$ (n is depending on the resolution of each image) whose each column refers to all pixels in a certain image by numbers and whose each row represents different pixels in every images within the same certain position. For matrix y, which has a size of 3×3000 , mainly contains the symbols of the three categories: cats, dogs and pandas.

Originally, when each image was invoked in, the data structure of it was a three- dimensional array: $\text{width} \times \text{height} \times 3$. After the dimensional reduction, we merged the 3rd dimension into one single integer by converting the true color RGB image into greyscale image. Then, we standardized the sizes of all images into the size we want by the function “`imresize()`”. Subsequently, we use the function “`reshape()`” to reshape the image matrix from 2D to 1D (that is $\text{size}^2 \times 1$). Finally, we combine all the $n \times 1$ matrixes together to create a single $n \times 3000$ matrix as our final X dataset.



	1	2	3	4	5	6	7	8
1	125	99	151	28	55	16	13	61
2	126	84	155	19	31	14	13	56
3	125	94	161	20	15	13	13	58
4	129	103	146	18	13	13	13	61
5	130	105	134	19	14	12	14	60
6	129	96	135	20	14	11	13	60
7	128	138	148	20	15	10	14	61
8	126	147	156	19	16	10	14	60
9	122	146	130	19	15	8	15	61
10	116	146	135	18	14	7	14	61
11	111	151	134	18	15	7	11	58
12	107	153	135	18	16	7	14	52

When processing matrix y, initially, it is an 1D array with the size of 1×3000 . After vectorizing by the function “`ind2vec()`”, we got a vector set of 3×3000 . Actually, the inner structure of the vectorized matrix is as follows:

(1, 997)	1	(2, 1001)	1
(1, 998)	1	(2, 1002)	1
(1, 999)	1	(2, 1003)	1
(1, 1000)	1	(2, 1004)	1

This basically means the index relations between the ones and the positions they should appear:

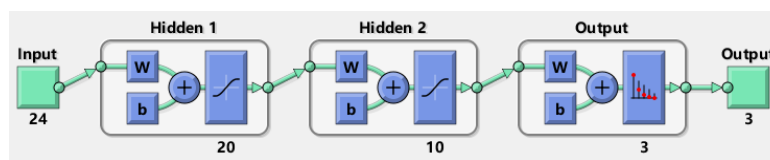
2018/2019 COMP1037 Coursework 2 – Machine Learning

As $(1, 1000) \rightarrow 100$

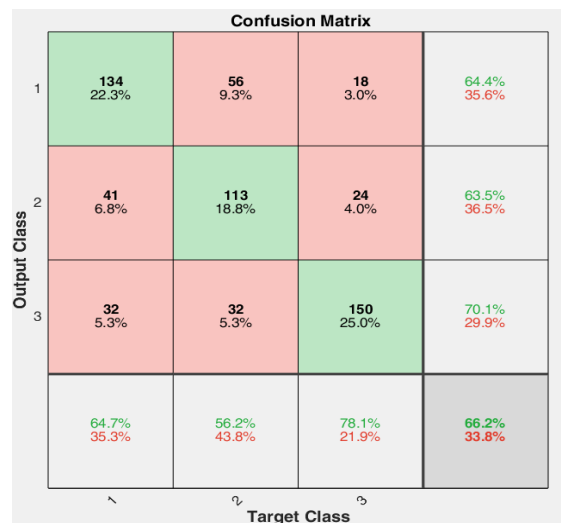
$(2, 1001) \rightarrow 010$

Thus, we got our second matrix y for the dataset.

For the network configuration, the pattern net with double hidden layers was adopted. There are 20 neurons in the first hidden layer and a 10 for the second one. For the input layer, the number of neurons there depends on the size of the input (the number of pixels per image). For the output layer, there are 3 output neurons because the data would be classified into 3 different categories.



Confusion Matrix & Performance Evaluation



The figure above is the confusion matrix with the best testing accuracy rate. The relationship between rows and columns of this matrix is basically the actual classification results of images by this nn versus the expected categories they should be in.

Take the first row as an example, all elements in the first row were actually categorized into the 1st class of the outputs by this nn. However, among the total 34.6% (22.3% + 9.3% + 3.0%) of all images (34.6% out of 3000), there are only 134 images were correctly categorized, with 56 which are supposed to be in the 2nd class and 18 in the 3rd class. The last column of the first row

2018/2019 COMP1037 Coursework 2 – Machine Learning

indicates the accuracy only within this row, which means that, among all elements in this row, 64.4% of them were accurately categorized.

Similarly, for the first column, all images in this column are supposed to be categorized in the 1st class. However, 41 of them were put into the 2nd class and 32 of them were recognized as elements in the 3rd class. The last row of the first column represents the classification accuracy only within the first column, obviously, 64.7% of the elements in the 1st class were accurately categorized.

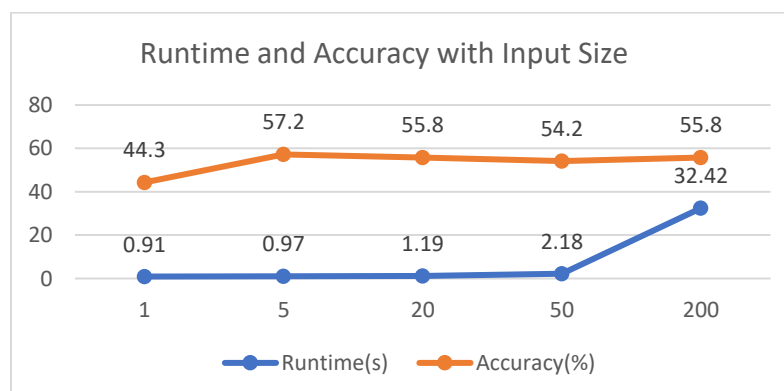
The last column of the last row indicates the total accuracy of this nn model, among all image data, 66.2% of them were correctly categorized and 33.8% of them were not.

To evaluate the overall performance of this neuron network, it could be noticed that the recognition ability of class 3 is comparatively higher than the abilities for other two classes, with the accuracy of 78.1%. This means that the boundary of class 3 with other class is relatively clear. Additionally, in the error cases, it could be concluded that when the elements in the other two class were classified incorrectly, they have a higher tendency to be classified into the other class rather than in class 3.

Parameter settings

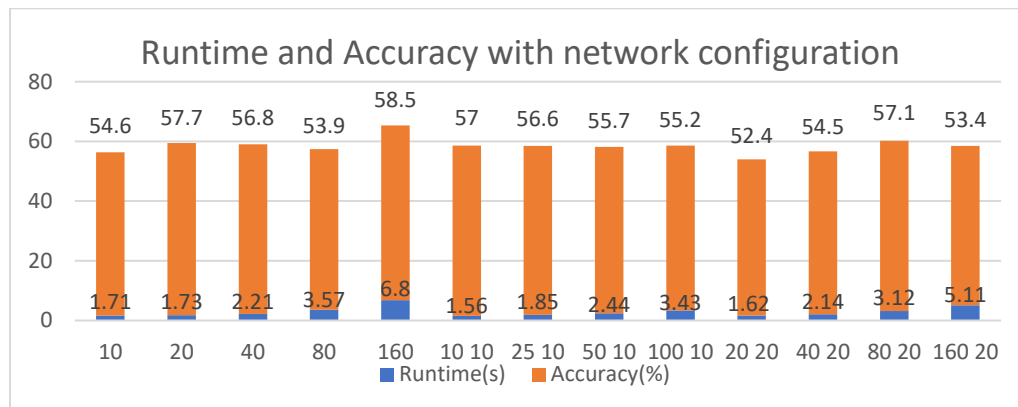
Parameters that I have tried in different settings are the size of the input images, the number of neurons in the hidden layers, the number of hidden layers, the ratio of the three element sets, the number of steps for the validation check and the model learning rate.

The following chart shows the relationship between program runtime, accuracy and the size of inputs when other parameters remain the same.



2018/2019 COMP1037 Coursework 2 – Machine Learning

As we can see from this chart, program runtime increase corresponding with the increase of the input size and after the input size of 50*50, the running time has a greater tendency of going upward than before 50*50. To conclude, the input size could influence the time cost of this program to great extent. Additionally, the accuracy of this program is relatively low when the input size is 1*1, after that, the accuracy would increase and fluctuate at about 56% under any other circumstances. That is to say, except the very extreme conditions, input size would have no such a significant influence on the total accuracy of this program.



As we can conclude from the chart above, when both within the single hidden layer network and double hidden layer network, the more complex the network is, the longer the running time will be. Comparatively, the number of hidden neurons is directly proportional to the program running time. For the different network configurations tested, the total accuracies are fluctuating in an average of 55.6%. To conclude, the network configuration does not show an obvious relationship with the accuracy. However, the more complex the network is, the more time it will cost to finish the program.

Other parameters were also adjusted for experimenting, such as the ratio of element sets and the learning rate. However, after experimenting, these parameters only influence the performance of the program to a very little extent.

Efficiency improvement

To improve the overall efficiency of this algorithm, we firstly adapt memory pre-allocation to improve the performance when reading in the images.

```
X = double(zeros(25, 3000));  
y = double(ones(1, 3000));
```

2018/2019 COMP1037 Coursework 2 – Machine Learning

Then, to improve the efficiency as well as avoiding over-simulating, we set validation set rate to be 0.15 and change the validation check max fails from the default to 12.

```
net.divideParam.valRatio = 0.15;

valData = X(:, rand_indices(1951:2400));
valLabels = y(:, rand_indices(1951:2400));

net.trainParam.max_fail = 12;
```

Thirdly, we adapt min-max normalization on the dataset X to convert each element in [0.0 – 1.0]:

```
for i = 1:m
    for j = 1:n
        X_(i, j) = (X(i, j)-xmin)/(xmax-xmin);
    end
end
```

Except the contribution for the improvement of efficiency, normalization also improve the accuracy by averaging the contribution of every feature value of the matrix.

Finally, we could use PCA(Principal Component Analysis) to implement dimensional reduction for a better efficiency.

```
X = pca(X);
X = X';
```

However, after experimenting, we noticed that because of the unavoidable loss of minor features, though the runtime of the program would be reduced, the accuracy of this algorithm would be affected. Therefore, the PCA was not invoked in the program eventually.