G53MLE - Artificial Neural Network Report

House Stark (Ting Pan, Junbin Wang, Yu Ding,Bolaji Abiodun,Bert Dzambulatov)

November 4, 2016

Introduction

Automatic recognition of human facial expressions is one of the most challenging tasks of our decade in computer science field. This report details how the task was implemented by using the Artificial Neural Network. In order to make our neural network perform well, we change the parameters of this neural network and train it repeatedly. For example, we change the number of neurons of hidden layer, the transfer function for the hidden layer and output layer, the ratio we divide the data, the number of iteration and so on. The accuracy of the Neural Network changed as we change the parameters and the report shows which combination of variables create the best Neural Network. The experiment figures are included in the appendix to show how we choose the parameters for the neural network. In addition to accuracy, we ensure the generalization and avoid the over-fitting of the neural network by early stopping and regularization.

1 Data Normalization and Balancing

Through applying the min-max function to the input data, the input data range is between -1 and 1. Hence, we conclude that normalization is not necessary in our case.

For cross validation, evenly distribution of the data is essential for the performance. We first grouped the samples with the same label. Then we evenly selected samples from each group to ensure that every validation set has similar data distribution. According to Figure 1, it is clear that network with balanced data perform much better than unbalanced data.

2 Parameters

The results of the ANN are dramatically influenced by the number of neurons, transfer function, training function and regularization coefficient and. They can be evaluated based on the maximum F1 measure.

- Neurons: We choose **20** for our neurons. The increase of the neurons improve the performance of the network. However, the growth gradually become slow as there are more neuron, especially after 15 neurons.
- Layer: We choose **1 layer** since the increase in the number of layers does not significantly improve the accuracy of the network.
- Regularization: We pick **0.04** as our regularization factor. The increase of the regularization coefficient can prevent over-fitting. However, under-fitting will happen if the value is too large. After experiment,
- Training function: Tried various training functions, We conclude that **trainlm** is the best option.
- Transform function: Experiment shows that the rank of the transform functions performance on 20 neurons is tansig, softmax, purlinm, logsig. Hence, we choose **tansig** in our case. However, we conculde softmax is suitable for classification problem because it has a stable and impressive performance even on a small number of neurons.

- Deviderand Ratio: With 10-fold cross validation we manually conduct the testing using the split validation set. Hence, our setting of the deviderand ratio is **80% for training, 10% for validation and 0% for testing** to ensures that more data can be trained.
- Learning Ratio: Experiment shows that learning ratio does not have substantial influence on the performance and a large learning rate might lead to a local optimal. Hence, we choose **0.01** as our learning rate.
- Epoch: A large epoch value does not substantially influence the training time since the training could be completed at the early stop position. However, when the value is too small, the network might not fully be trained. Hence, we set epoch to **1000** to ensure it is fully trained.

3 Difficulties encountered

The most difficulty encountered is the choice of parameters especially some coefficient. For example, when regularization factor is too large, it would lead to under-fitting. However, when it is not big enough, it also lower the performance. Similar situation happens when we start to choose number of neurons. To address this issue, we use control variate method to generate data and use graph to compare the result with different values. Through it we have more concrete and direct view for parameter choosing. Meanwhile, due to the mechanism of neural network, the random bias and weight might have quite influence on the final result. To handle this problem, we do multiple test for best performance.

4 Classification Results

Using the average cross validation each label has its own precision rate, recall rate and F1 measure derived from a confusion matrix. The reason the results had a good average, were because of the parameters. By using a combination of the tansig function, 20 neuros, a regularisation number of 0.04 and 1000 epochs were able to get an average F1 measure accuracy of 0.9301. The confusion matrix and average recall and precision rate are included in the appendix.

After changing the parameters there were different averages and accuracies depending on the parameters passed through. The best results were when we used a tansig function, 20 neurons, and 0.04 regularizations. There is a clear indication on graph 4 detailing the difference in the accuracies using different functions, and it is clear that the transig function is best suited to 20 neurons. The value from regularization comes from the graph 3 which shows 0.04 is the optimal value to avoid the trouble of overfitting and underfitting. In conclusion, for this specific data set these parameters can produce the best performance of neural network for the automatic recognition of human facial expressions.

5 Generalisation & Overfitting

With 10-fold cross validation, we can figure out the over-fitting when evaluating the trained model. Another technique that merits mention to improve the generalisation is adjusting the regularisation coefficient. We used this technique in our neural network and it has slightly improved our final performance.Moreover, we used standard early stopping method to avoid potential over-fitting during the training process. Manually operate the division ratio of ‘dividerand’ also increased our results and prevented over-fitting.

Appendix

A Average Confusion Matrix and Evaluation

	1	2	3	4	5	6
1	70	1	0	0	3	0
2	6	73	0	1	2	2
3	1	2	79	5	0	3
4	1	0	2	139	0	1
5	2	1	1	1	78	1
6	0	1	1	1	0	134
Precise for Label 1 = Purple / (Purple + Yellow)						
Recall for Label 1 = Purple / (Purple + Green)						

Table 1: Average Confusion Matrix

```
ann_eval_matrix =

    0.9459    0.8750    0.9091
    0.8690    0.9359    0.9012
    0.8778    0.9518    0.9133
    0.9720    0.9456    0.9586
    0.9286    0.9398    0.9341
    0.9781    0.9504    0.9640
    0.9286    0.9331    0.9301 (Average of 6 labels)
```

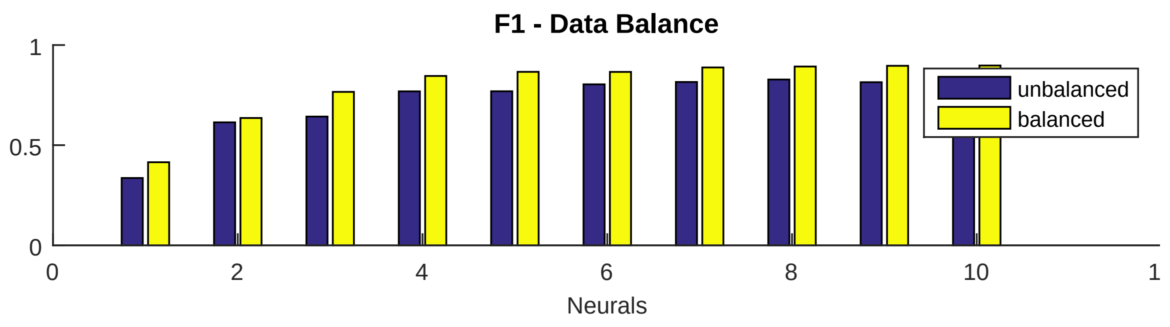
B Result of Each Validation

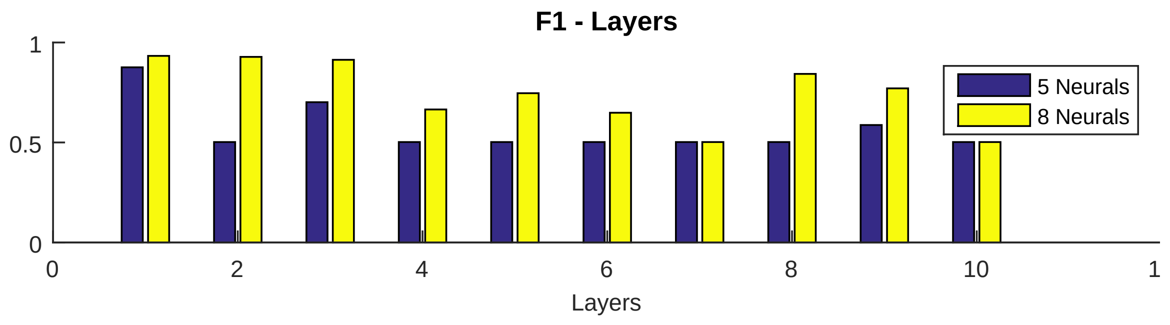
Each table shows classification results per cross-validation fold. Columns are split in recall, precision and F1 measure accordingly:

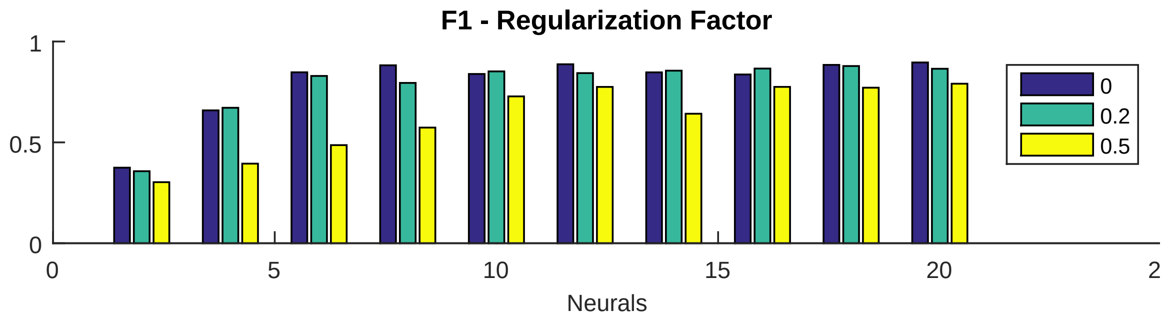
```
cross_valid_eval_matrixs(:,:,1) =

    0.8571    0.5455    0.6667
    0.3750    0.5000    0.4286
    0.4444    0.5714    0.5000
    0.7857    0.6875    0.7333
    0.5000    0.6667    0.5714
    0.8571    0.8571    0.8571
    0.6366    0.6380    0.6262


cross_valid_eval_matrixs(:,:,2) =

    1.0000    0.8889    0.9412
    1.0000    1.0000    1.0000
    1.0000    1.0000    1.0000
    0.9333    1.0000    0.9655
    1.0000    1.0000    1.0000
    1.0000    1.0000    1.0000
    0.9889    0.9815    0.9844


cross_valid_eval_matrixs(:,:,3) =

    0.8571    0.7500    0.8000
    0.7500    1.0000    0.8571
    1.0000    1.0000    1.0000
    1.0000    1.0000    1.0000
    1.0000    0.8889    0.9412
    1.0000    1.0000    1.0000
    0.9345    0.9398    0.9331


cross_valid_eval_matrixs(:,:,4) =

    1.0000    0.8889    0.9412
    0.8889    1.0000    0.9412
    0.8889    1.0000    0.9412
    1.0000    1.0000    1.0000
    1.0000    1.0000    1.0000
    1.0000    0.9333    0.9655
    0.9630    0.9704    0.9648


cross_valid_eval_matrixs(:,:,5) =

    0.8571    1.0000    0.9231
    0.8750    0.8750    0.8750
    1.0000    1.0000    1.0000
    1.0000    1.0000    1.0000
    1.0000    1.0000    1.0000
    1.0000    0.9333    0.9655
    0.9554    0.9681    0.9606


cross_valid_eval_matrixs(:,:,6) =

    1.0000    1.0000    1.0000
    1.0000    0.8889    0.9412
    0.8889    1.0000    0.9412
    1.0000    1.0000    1.0000
    1.0000    1.0000    1.0000
    1.0000    1.0000    1.0000
    0.9815    0.9815    0.9804
```

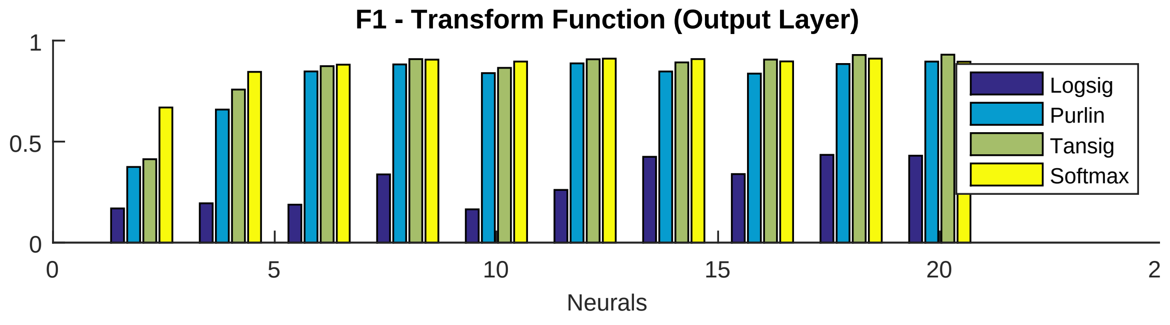
```
cross_valid_eval_matrixs(:,:,7) =

    0.8750    0.8750    0.8750
    1.0000    1.0000    1.0000
    0.6667    1.0000    0.8000
    1.0000    0.8750    0.9333
    1.0000    0.9000    0.9474
    1.0000    1.0000    1.0000
    0.9236    0.9417    0.9260


cross_valid_eval_matrixs(:,:,8) =

    1.0000    1.0000    1.0000
    0.8750    1.0000    0.9333
    0.8889    0.8889    0.8889
    1.0000    1.0000    1.0000
    0.8750    0.8750    0.8750
    1.0000    0.9333    0.9655
    0.9398    0.9495    0.9438


cross_valid_eval_matrixs(:,:,9) =

    1.0000    1.0000    1.0000
    1.0000    1.0000    1.0000
    1.0000    1.0000    1.0000
    1.0000    0.9375    0.9677
    1.0000    1.0000    1.0000
    0.9231    1.0000    0.9600
    0.9872    0.9896    0.9880


cross_valid_eval_matrixs(:,:,10) =

    1.0000    1.0000    1.0000
    0.8750    1.0000    0.9333
    1.0000    1.0000    1.0000
    1.0000    1.0000    1.0000
    0.8750    1.0000    0.9333
    1.0000    0.8750    0.9333
    0.9583    0.9792    0.9667
```

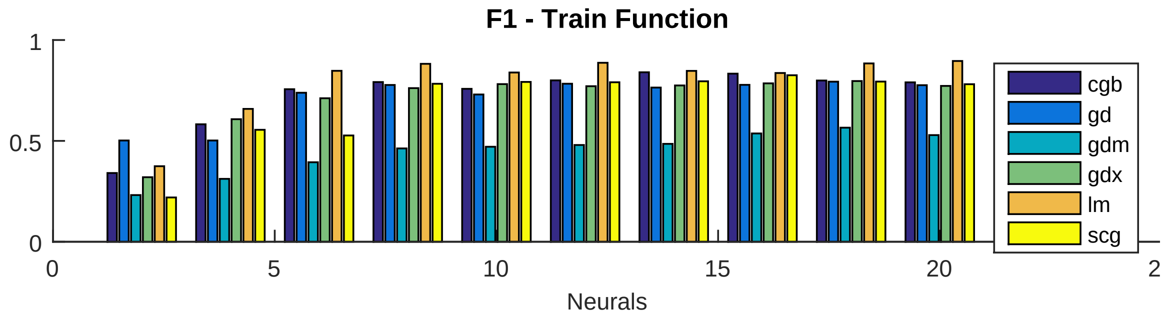
C Experiment Graphs


F1 - Data Balance
Neurals
unbalanced
balanced


F1 - Layers
Layers
5 Neurals
8 Neurals


F1 - Regularization Factor
Neurals


F1 - Transform Function (Output Layer)
Neurals
Logsig
Purlin
Tansig
Softmax


F1 - Train Function
Neurals
cgb
gd
gdm
gdx
lm
scg