Emotion classification using facial expression features using DL approach and comparison with nn

U6919082

Zhijie Guo

Abstract

Based on the dataset face-emotion (The static database has been extracted
from the temporal dataset Acted Facial Expressions in the Wild (AFEW))[1], I did the classification based on the given features to predict the emotion. And I used the pruning method in the paper of Progress Image Compression, I implemented the pruning method(base on distinctiveness defined in[2], pruning by inspection is difficult even on small examples, some automatable process would be ideal. Properties such as *relevance*, *contribution* , *sensitivity* , *badness* , and *distinctiveness* have been described in detail elsewhere.)in the paper to increase the efficiency of training. I used a 3-layer neural network and multi-hidden layer nn to do the job separately and compared the outcome. I learned techniques from [5] and [6] and I also tried to do the rnn, but my hardware is not adequate for me to use tensorflow. My result is much worse than the result in the paper.

Introduction

Facial expression plays an vital role in our daily life, no matter whether you are with your friends or with strangers, as long as you are not alone, your facial expression need to be read and understand as a way of showing your emotions and make other people respond to it. Thus understanding the facial expression, the very first step, is what I am quite interested in. Base on the provided dataset, I decide to implement the function which classify the emotion of a face base on features. [1]The feature represents facial expressions, head poses, large age range, different face resolutions, occlusions, varied focus and close to real world illumination, which gives me quite a lot confidence that with the dataset I can perform a real-world class function. I implemented a neural network with input being the features in the dataset and output being the predicted value with single hidden layer(code learned from[3]). I used the dataset to train the model with optimizer being torch.optim.RMSprop(code learned in [4]) and loss function being Cross entropy(code learned in [5]) as loss function. And the speed of training can be improved by implementing the pruning method(base on distinctiveness defined in[2]).

Method

I used torch.nn.sequential to implement the 3-layer neural net work and multi-hidden layer neural network. I used a 3-layer neural network(10 input units for 10 features, 100 units for hidden layer and 7units for output layer to represent the 7 classes) with 1 hidden layer and the activation function of hidden layer and output layer are Sigmoid and Softmax separately, for the optimizer, I used torch.optim.RMSprop(code learned in [4], a bit faster so that the epochs can be lessened) and Crossentropy(code learned in [5], suitable for the classification problem) as loss function. While training, I used the distinctiveness defined in[2](Since all activations are constrained to the range 0 to 1, the vector angle calculations are normalised to 0.5, 0.5 to use the angular range of 0-180° rather than 0-90°.) as the standard of deciding if the node should be pruned and prune the hidden layer to increase the efficiency of training. And I did the similar thing to create and train the multi-hidden layer neural network. After training, I used self-defined accuracy function(probability that the prediction is correct, defined as (corrected predicted)/(number of all predictions)) to evaluate the performance of the 2 models.
For RNN, I read the paper[5](Recurrent neural networks (RNNs) are a powerful family of connectionist models that capture time dynamics via cycles in the graph. Unlike feedforward neural networks, recurrent networks can process examples one at a time, retaining a state, or memory, that reflects an arbitrarily long context window. While these networks have long been difficult to train and often contain millions of parameters, recent advances in network architectures, optimization techniques, and parallel computation have enabled large-scale learning with recurrent nets.) and got the basic idea how the network is going to work and how I should employ it. After that, I read paper [6](We present a simple regularization technique for Recurrent Neural Networks (RNNs) with Long Short-Term Memory (LSTM) units. Dropout, the most successful technique for regularizing neural networks, does not work well with RNNs and LSTMs.) and tried to do some regularization to the network to improve the performance. At last I learned how to build RNN from [8] https://www.jianshu.com/p/dfb7cf19f7b9.

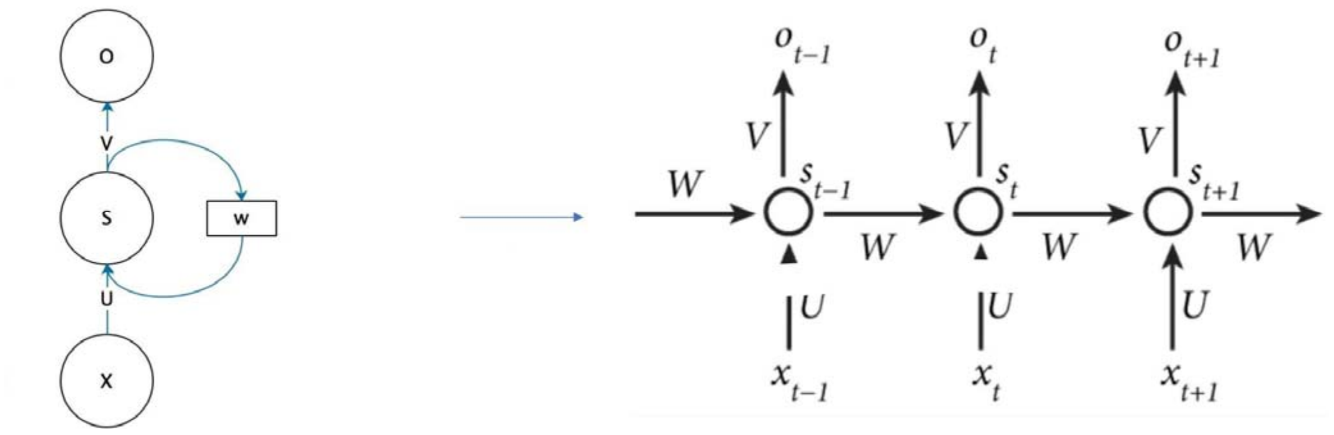


Image from: [9]https://zhuanlan.zhihu.com/p/30844905

Below is the result in the experiment in[1]

Emotion	Angry	Disgust	Fear	Happy	Neutral	Sad	Surprise
Precision	0.17	0.15	0.20	0.28	0.22	0.16	0.15
Recall	0.21	0.13	0.18	0.29	0.21	0.16	0.12
Specificity	0.48	0.66	0.64	0.51	0.61	0.60	0.66

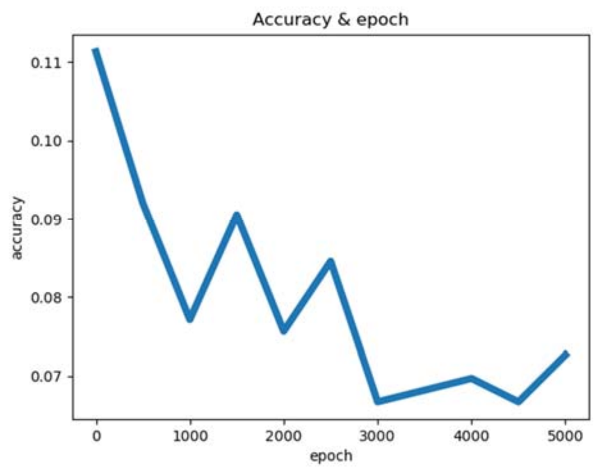
And my overall accuracy is:

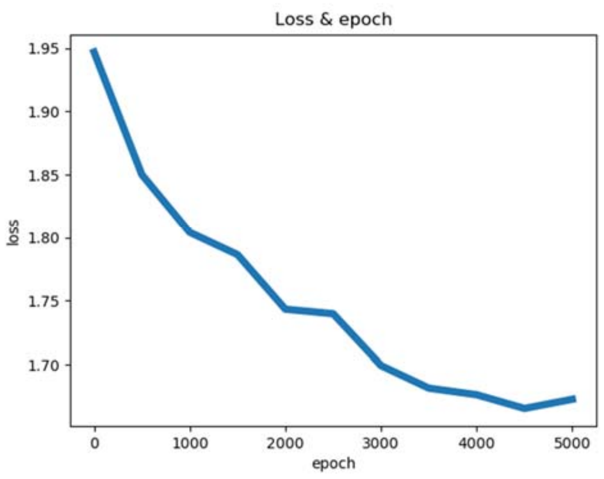
3-layer Neural network
0.14836795252225518

Multi-hidden layer Neural network
0.14836795252225518

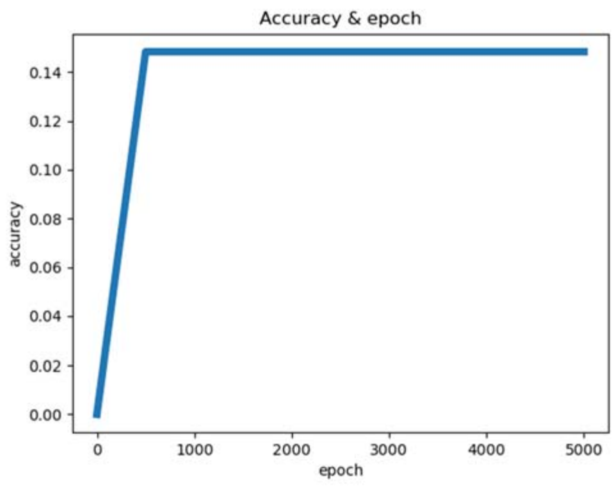
The accuracy and loss recorded every 500 epoches.

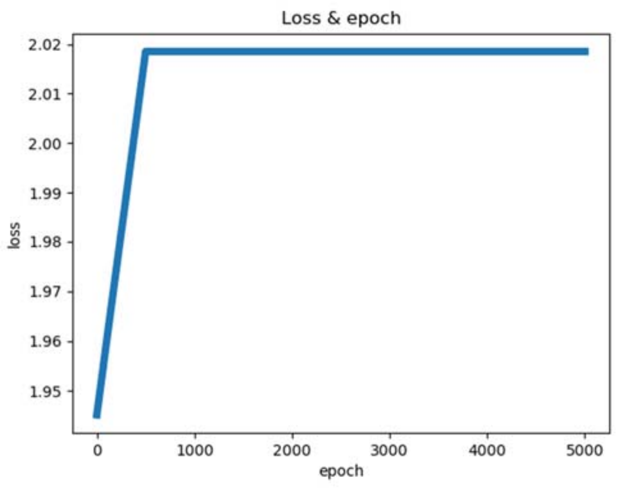
3-layer Neural network





Multi-hidden layer Neural network





Conclusion and Future Work

The result above show that multi-hidden layer neural network is not better than single hidden layer neural network, but recurrent network is much better than both of them because there is a difference between RNN and NN, which is RNN enables the network to memorize some previous experience and thus accuracy is improved.

RNN (recurrent neural network) is composed of input layer, hidden layer and output layer. The traditional neural network has a disadvantage that the neurons in the same layer do not transmit to each other, which leads to poor performance in some language translation. When translating, the next word is related to the previous one, and the language is related. The accuracy of translation is very low. RNN enables neurons in the hidden layer to communicate with each other, and stores the previous output results in the hidden layer in the form of information. When the next word is translated, the previous output also has an impact on it, which links up the word translation. Thus in some extent, RNN enables the network to memorize some previous experience and thus accuracy is improved.

For future work, I can try some other deep learning approaches or evolutionary algorithms.

Reference

--[1] A. Dhall, R. Goecke, S. Lucey, and T. Gedeon. Acted Facial
Expressions in the Wild Database. In Technical Report, 2011. 1, 2

--[2] PROGRESSIVE IMAGE COMPRESSION T.D. Gedeon1 & D. Harris2

[3]http://cache.baiducontent.com/c?m=9f65cb4a8c8507ed19fa950d100b92235c4380146d8b804b2281d25f93130a1c187bbae06279555dce87616402aa4a5feef5356537747af1c4969c0f80fbc4276dc33035005adb0143890eaebb5153c737e45dfede18f0caf62592dec5a3d94324bd44750c9783814d0164dd1efb0345e0b1e93f022c66adec40728e2f605d953431b6508ee5&p=882a9644d18702fc57efdb26460e80&newp=826ed51481904ead34bd9b7d0c1183231610db2151d6d7136b82c825d7331b001c3bbfb42328130fd3c67b620bae4a5de1f03573370923a3dda5c91d9fb4c57479e4&user=baidu&fm=sc&query=pytorch%D4%F5%C3%B4%B4%EE%BD%A8%C9%F1%BE%AD%CD%F8%C2%E7&qid=c7a4b908002c8939&p1=2

--[4] https://blog.csdn.net/devcy/article/details/89335575

--[5] https://blog.csdn.net/geter_CS/article/details/84857220

--[6] A Critical Review of Recurrent Neural Networks for Sequence Learning Zachary C. Lipton University of California, San Diego zlipton@cs.ucsd.edu

--[7] RECURRENT NEURAL NETWORK REGULARIZATION Wojciech Zaremba ∗ New York University woj.zaremba@gmail.com Ilya Sutskever, Oriol Vinyals Google Brain {ilyasu,vinyals }@google.com

--[8] https://www.jianshu.com/p/dfb7cf19f7b9

--[9]https://zhuanlan.zhihu.com/p/30844905