LSTM classification model based on min-max features and decision tree rules

HanDu

Reserch School of Computer Science.
The Australia National University,
Canberra ACT 2601
u7077100@anu.edu.au

Abstract: The interpretability of the neural network model is limited, and we cannot clearly explain the specific meaning of each layer inside the neural network. But we can adjust the input of the model to allow the neural network model to give us the data characteristics, thereby judge whether the features we give are valid and whether the model can perform well. I analyzed the features of thermal-RGB-components.xlsx [1] and extracted the corresponding min-max features in two groups according to the ten features of RGB_Five_Fist_Top_Components1-5 and Thermal_Five_Fist_Top_Components1-5. I also trained a decision tree model, using some interpretable features provided by the decision tree as the input component of the LSTM model. In the experimental part, I verified that the min-max feature and the rule feature extracted by the decision tree have improved significantly for the RNN model. I also compared RNN (Recurrent Neural Network) [2], GRU (Gated Recurrent Unit) [3] and the effect of the LSTM (Long Short Term Memory networks) [4] model and found that the LSTM model has the best effect, and then I conducted an in-depth adjustment of the LSTM to obtain the optimal parameters, reaching a maximum accuracy of 66.92%.

Keywords: Neural networks, Stress recognition, LSTM, Rule features, Decision tree

1 Introduction

This paper uses the data set provided by the paper [1], this data set contains 620 pressure identification data. Anxiety is an emotion when people face a negative environment. The continuation of this emotion will have a lot of negative effects on people's health. Therefore, identifying anxiety is a very important issue. Based on the data set provided by the paper, this paper proposes two ways to expand features: Firstly, I divide the features into two groups and extract min-max features from each group of features respectively; Secondly, after dividing the data into training set and test set, I trained a decision tree model and extracted some hidden features in the data through the rules provided by the decision tree. To identify the pressure data, I chose RNN variant models GRU and LSTM, and did three sets of experiments to explore whether these features and models are effective:

1. Compare the RNN model performance when using min-max features, using decision tree rules generated features, and using both types of features.
2. After determining the optimal feature combination, based on the optimal feature combination and the same model parameter conditions, compare the effects of RNN and its improved models GRU and LSTM.
3. Adjust the parameters of the optimal model obtained under the same conditions in the previous experiment, obtain the best model and calculate its effect.

The specific experimental process is shown in the figure:

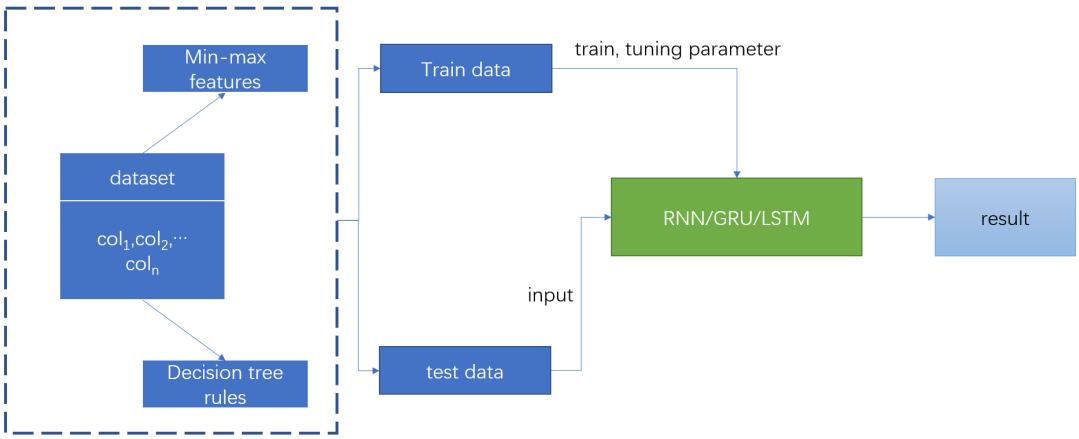

Figure 1 experimental process

After comparing some of the above experiments, I found that these two types of features have an improvement effect on the model, and LSTM performs best. After the model is adjusted, the model effect is also improved to a certain extent, indicating that LSTM is very suitable for this task.

2 Feature extraction

2.1 Min-max features

RGB_Five_Fist_Top_Components1-5 and Thermal_Five_Fist_Top_Components1-5 are two different sets of features, but the values in each set of features are related. To reflect the correlation between the maximum and minimum values, I extracted each of the two groups. The min and max features are denoted as RGB_min, RGB_max, Thermal_min, Thermal_max, and the calculation formula is as follows:

$$\begin{cases} Thermal\_min = \min\{Thermal\_Five\_Fist\_Top\_Components\_i \mid i \in [1,5]\} \\ Thermal\_max = \max\{Thermal\_Five\_Fist\_Top\_Components\_i \mid i \in [1,5]\} \\ RGB\_min = \min\{RGB\_Five\_Fist\_Top\_Components\_i \mid i \in [1,5]\} \\ RGB\_max = \max\{RGB\_Five\_Fist\_Top\_Components\_i \mid i \in [1,5]\} \end{cases}$$

2.2 IF-THEN rules based on decision tree extraction

Since the decision tree can generate some interpretable rules, I decided to use it to construct some rules to extract features [6]. The decision tree is a machine learning model that is easy to overfit. To avoid overfitting the machine learning model and produce features with strong generalization ability, I made a 5-layer decision tree. The rules generated by the five-level decision tree are selected as my characteristics to enhance the generalization ability [7] [8]. Interpretable rules generated under the drawn decision tree:

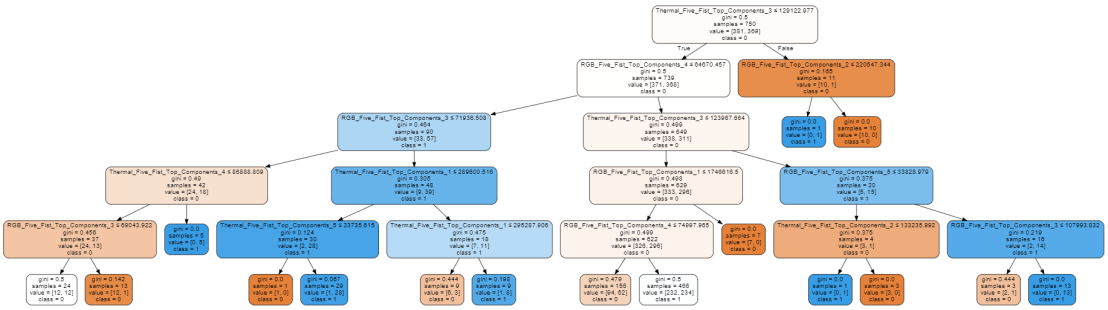

Figure 2 Decision Tree

Take the root node to two child nodes as an example. When the characteristics of a piece of data meet the condition Thermal_Five_Fist_Top_Components_3 $\leqslant$129122.977, the data should go to the left child node for the next judgment. If the condition is not met, it should go to the right child node for the next judgment.The last leaf nodes are specific categories, so if each piece of data is judged in the decision tree, specific categories can be obtained. As can be seen from the figure and the process, the decision tree has an IF-THEN process for the division of each feature, that is, if the conditions in the node are met, it will point to a True or False result. Also take the rule of the root node as an example, the rule I generated is:

IF Thermal_Five_Fist_Top_Components_3 ≤ 129122.977:
New_Feature = 1
ELSE:
New_Feature = 0

Through the tree above, I generated 15 rules in total and formed 15 new features.

3 Method

3.1 RNN and LSTM models

RNN has become the most used network model in recent years, but RNN is restricted by short-term memory, and there are problems of gradient disappearance and gradient explosion during training. This defect makes it difficult to train in long text. Models such as LSTM and GRU solve this problem to a certain extent [5].

LSTM further complicates the internal structure of the unit in RNN, adding three main mechanisms: forget gate, update gate, and output gate. Similar to a memory cell, it filters irrelevant information and retains useful information for a long time [5]. The specific structure inside the LSTM model unit is shown in the figure.

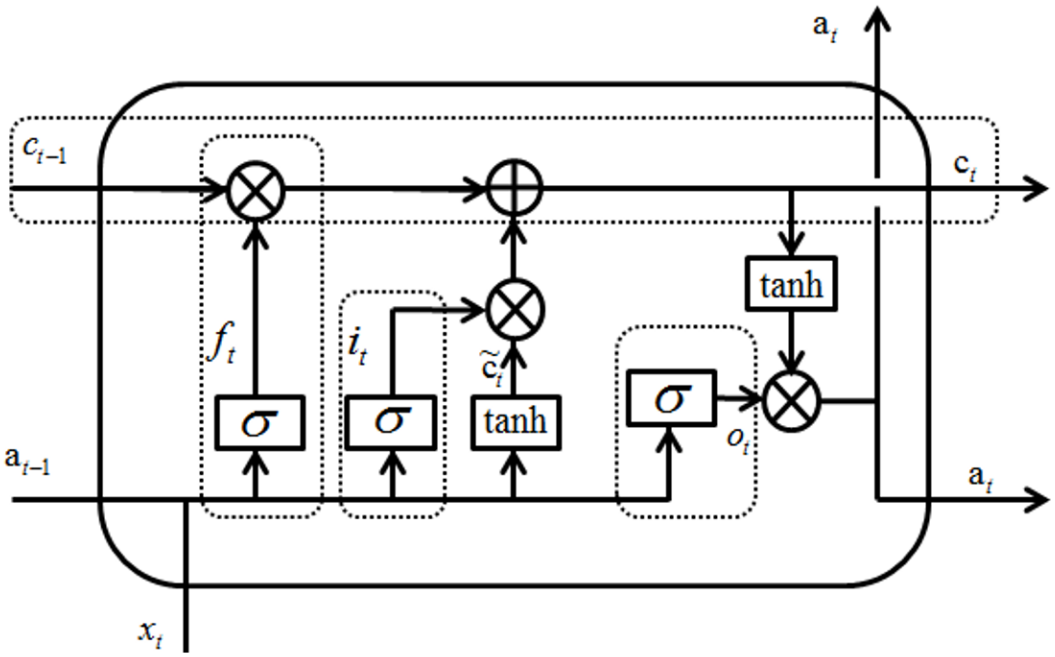

Figure 3 LSTM structure

Let t represent the time sequence, the cell state c_{t-1} and the hidden state a_{t-1} output by the previous sequential unit will be used as the input of the current sequential unit. In the current time sequence t, the unit mainly performs the following operations on the input information:

(1) Forget irrelevant information

After the current input is spliced with the hidden state of the t-1 timing output, the forgetting gate is used to determine the information that needs to be retained in the previous timing c_{t-1}, and the information that is not needed to be understood or the secondary information is deleted. The relevant formula for the forget gate is as follows:

$$f_t = \sigma(W_f \cdot [a_{t-1}, x_t] + b_f)$$

After the input is transformed by the linear equation and the sigmoid function, a vector with a range of (0,1) is obtained, corresponding to each feature in the cell state. In this way, the feature multiplied by 0 will be forgotten, and the feature multiplied by 1 will remain.

(2) Add new information

The update gate will extract useful information from the current input and join it. It is similar to the forget gate, acting as a filter and determining which features are added to the cell state through the sigmoid function. The expression to filter the input information is as follows:

$$i_t = \sigma(W_i \cdot [a_{t-1}, x_t] + b_i)$$

Then create a feature vector $\tilde{c}_t$ containing all the information in the current input:

$$\tilde{c}_t = \tanh(W_c \cdot [a_{t-1}, x_t] + b_c)$$

Then $i_t * \tilde{c}_t$ represents the feature vector after extracting useful information from the current input, and $f_t * c_{t-1}$ represents the feature vector after filtering out irrelevant information from the previous time sequence, and the current cell state c_t is obtained after addition . At this point you can ensure that important and non-redundant features are added to the cell state.

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t$$

(3) Output the hidden state of the current sequence

After calculating the current cell state, the output gate will filter c_t to select the information that needs to be kept in the hidden state. That is, the final output feature is determined by the vector o_t. The calculation method of the vector is as follows:

$$o_t = \sigma(W_o \cdot [a_{t-1}, x_t] + b_o)$$

Finally, the cell state is reduced to (-1, 1) through the tanh function, and the outer product is performed with the vector o_t to obtain the hidden state a_t of the time sequence t:

$$a_t = o_t * \tanh(c_t)$$

LSTM controls long-term memory through cell state and short-term memory through hidden state, which improves the defect that RNN is difficult to obtain long-distance dependent information. This model can often achieve better results in application scenarios with obvious timing characteristics.

3.2 Save the optimal model

To minimize the phenomenon of model overfitting, I adopted a mechanism similar to early stop. When training the model, save the model with the highest accuracy on the validation set and use it for testing in the test set [9]. This method can better prevent the model from continuing to train in the direction of more overfitting when it has some overfitting and save the model when the verification set has the best effect.

4 Results and Discussion

I used pytorch to build an RNN model, and based on this, I extended to the GRU and LSTM models. Since this is a binary classification task, I use accuracy as the criterion. Assuming that there are samples in the test set, and the correct number of samples predicted by the classification model is, the accuracy of the model is:

$$accuracy = \frac{M}{N}$$

I randomly divided 20% of the data as test data, so I have a total of 490 training samples and 130 test samples. To ensure that the variables in the experiment are unique, I preset several parameters:

Parameter	Value
num_epochs	1000
learning_rate	0.001
hidden_size	100
layer	1

To ensure the reproducibility of the experiment, I set a uniform random seed for numpy, pandas, and pytorch, with a value of 1.

4.1 Improvement of the model effect by features

In feature extraction, I extracted min-max features and dt rules (decision tree rules) respectively. To study whether these two features are helpful for model improvement, I analyzed the effectiveness of these features by comparing the accuracy of the RNN model on the test set under different feature extraction situations.

Feature	RNN acc
base	53.08%
Dt rules	55.38%
Minmax features	56.92%
Dt rules+minmax features	**58.46%**

According to the model effect, it can be found that both min-max features and dt rules have a certain improvement on the model effect, and when both are used at the same time, the model effect is the best, with an acc of 58.46%. This shows that whether it is based on the min-max feature extracted from the original data or the rule obtained based on the decision tree, the model effect can be improved.

4.2 RNN, GRU and LSTM comparison

Since the use of min-max features and dt rules in the previous experiment can better improve the model effect, in the subsequent experiments, I decided to use the data set obtained after the two feature extractions to conduct experiments. Both GRU and LSTM are based on the improved model structure of RNN and can better solve the problems of gradient

disappearance and gradient explosion, so I compared the effects of these three models in this experiment. The experimental results are shown in the figure below:

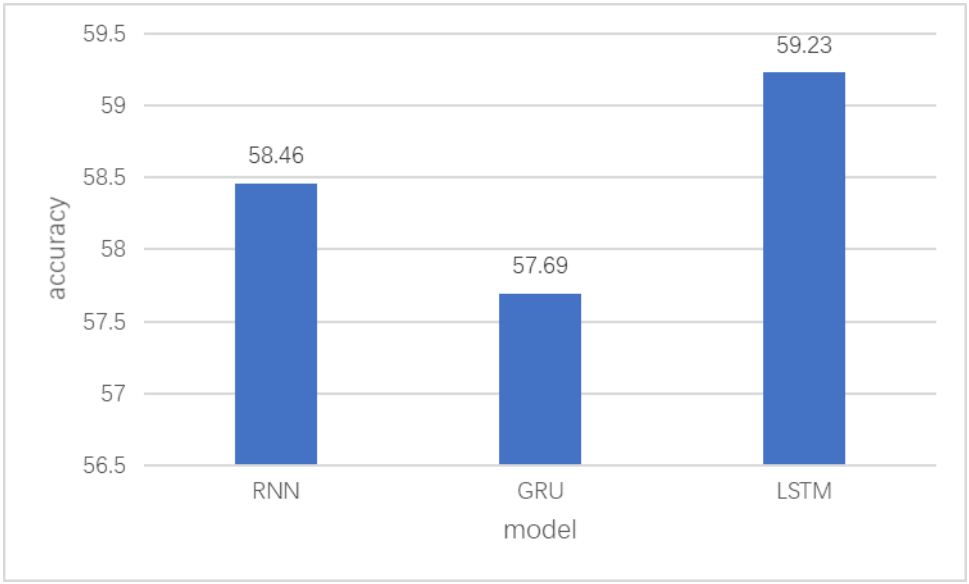

Figure 4 RNN, GRU and LSTM comparison

The effect of LSTM is the best, with an accuracy rate of 59.23%, while the effect of GRU is worse than that of RNN. This may be because GRU reduces the problem of gradient explosion and gradient disappearance to a certain extent, but it lacks a gating unit, and there may be some key features that have not been learned, so the effect is worse than RNN, only reaching 57.69%.

4.3 LSTM tuning comparison

LSTM performs well on this task, so I use the LSTM model to conduct some tuning experiments, hoping that the model can achieve the best possible effect on the data set. I tried to optimize the model effect by adjusting LSTM layers and hidden_size [10], the experimental results are as follows:

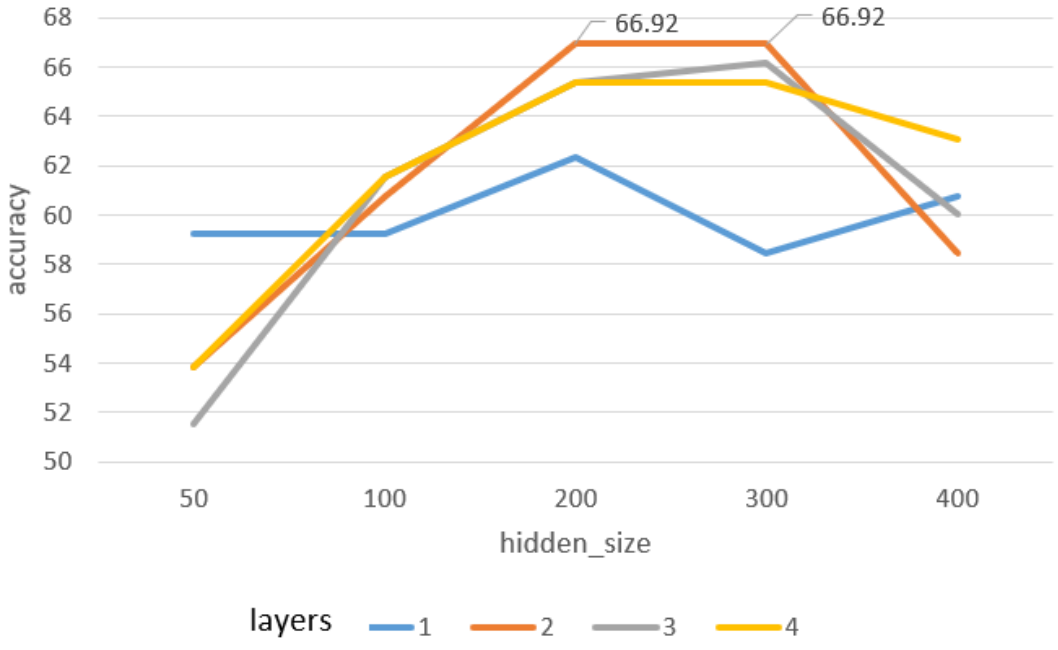

Figure 5 The effect of layers and hidden_size on LSTM performance

When a 1-layer LSTM is used, the effect of the model fluctuates between 58% and 63%, and the stability is strong. With 2-4 layers of LSTM and 50 neurons, the model effect is generally poor, but when the number of neurons increases

to 100 or 200, the model effect is significantly improved. This shows that within a certain range, the increase of neurons can better improve the model effect. However, when the number of neurons continues to increase to 300 and 400, the model effect not only does not improve, but even declines. This shows that even if the optimal model on the validation set for each epoch is saved, the overfitting phenomenon cannot be completely prevented. When the parameters and the number of layers reach a certain number, the model may still have a certain over-fitting phenomenon and reduce the effect of the model.

According to the above experimental results, when the number of layers is 2 and the number of neurons is 200 or 300, the model achieves the best effect, and the accuracy rate reaches 66.92%.

5 Conclusion and Future Work

5.1 Conclusion

Although it is difficult to know the internal behavior of neurons and clearly explain the meaning of each layer in the neural network, I have extracted features that are meaningful to humans by processing and analyzing the data and input them into the neural network. The neural network continuously learns and fits the features, so that it has the ability to predict. Such work can increase the interpretability of the model at the input level. It can be observed whether the model effect is improved, stable, and generalized from the change of the characteristics of the input layer, the number of hidden layers, and the number of neurons, to analyze some universal laws. From this perspective, my research endows the neural network with external interpretability, and try to discover some meaningful features and viewpoints. By adjusting the features, comparing different models, and tuning parameters, I found some meaningful points:

1. The features extracted from the original data set and the features extracted from the if-else rule based on the decision tree are meaningful and can improve the effect of the model.
2. On this task, the LSTM model performs better than the RNN and GRU models.
3. The number of LSTM layers and the number of hidden layer neurons have certain influence on the model, too small will lead to poor model learning ability and too large will lead to model over-fitting. So, we need to take a moderate value. The more suitable number of neurons I found was 200-300.

5.2 Future Work

The feature extraction method in this paper mainly comes from the min-max feature of the original data set and the feature extracted by the decision tree, both of which have a certain interpretability. In the future, I will try to analyze and extract some more effective features from a more professional perspective. On this basis, I will try to use more neural network models for testing and confirm whether these newly extracted features and newly selected models are effective, so as to test the validity of the features and the generalization of the model.

References

[1] Irani R , Nasrollahi K , Dhall A , et al. Thermal super-pixels for bimodal stress recognition[C]// International Conference on Image Processing Theory Tools & Applications. IEEE, 2017.

[2] Zaremba W, Sutskever I, Vinyals O. Recurrent neural network regularization[J]. arXiv preprint arXiv:1409.2329, 2014.

[3] Chung J, Gulcehre C, Cho K H, et al. Empirical evaluation of gated recurrent neural networks on sequence modeling[J]. arXiv preprint arXiv:1412.3555, 2014.

[4] Zheng S, Ristovski K, Farahat A, et al. Long short-term memory network for remaining useful life estimation[C]//2017 IEEE international conference on prognostics and health management (ICPHM). IEEE, 2017: 88-95.

[5] Sundermeyer M, Schlüter R, Ney H. LSTM neural networks for language modeling[C]//Thirteenth annual conference of the international speech communication association. 2012.

[6] Swain P H, Hauska H. The decision tree classifier: Design and potential[J]. IEEE Transactions on Geoscience Electronics, 1977, 15(3): 142-147.

[7] Dietterich T. Overfitting and undercomputing in machine learning[J]. ACM computing surveys (CSUR), 1995, 27(3): 326-327.

[8] Schaffer C. When does overfitting decrease prediction accuracy in induced decision trees and rule sets?[C]//European Working Session on Learning. Springer, Berlin, Heidelberg, 1991: 192-205.

[9] Mexicano A, Rodríguez R, Cervantes S, et al. The Early Stop Heuristic: a new convergence criterion for k-means[C]//AIP conference proceedings. AIP Publishing LLC, 2016, 1738(1): 310003.

[10] Xue G, Pan Y, Lin T, et al. District heating load prediction algorithm based on feature fusion LSTM model[J]. Energies, 2019, 12(11): 2122.