# Deep Learning Project Report

Zhang Jiaheng 5140719075
Wan Cheng 5140309552
Xu Xiaojun 5140309536

2017.1.15

## Contents

# 1 Task

## 1.1 Task one

Before we write the program and do the experiment, we should make our goal. The aim of the program is to train the data, which represents four different tones in Chinese, and recognize them using neural network model. We need to deal with the original dataset and construct the neural network to train the data. In addition, for the test set, maybe there are some noises to affect the judgement of the network. So feature extraction, denoising and designing neural network model are the most important part of our program.

## 1.2 Task two

Another task for us is to compare the performance of our model in different library for deep learning. Our group chooses three different library: MxNet, Theano and CNTK. We use the same model and same data set to do experiment and see comparison between different toolkits.

# 2 Data processing

Since raw data is quite complicated and data size is small, we have to process data in advance in order to train a good model. A trivial step is to cut the head interval and tail interval whose frequency is zero. Now we focus on the remained data.

## 2.1 Feature extraction

Since the data size is small, how to extract the correct features becomes the key step to build up the system. After observing the data carefully, we think that the change in frequency is the major element to decide the tone of the input. So, we made several attempts to model this information:

### 2.1.1 Picking points on equal distance.

Our first attempt is to pick points on appropriate distance, and use their frequency and energy as input. For example, pick the points at distances 0%, 10%, 20%, ..., 90%, and 100%, and use their frequency and energy. Thus, it

forms a 22-dimensional vector and use this as the input. We hope that the network can learn the correlations between these points. For example, if the frequency at both ends is higher than the middle ones, then it tend to be the third-tone.

However, such method doesn't perform well. Even the training accuracy is below 80%. We think there're two reasons: 1. the data is noisy so merely picking one point cannot represent the information; 2. the dataset is small, so it's hard for the network to learn the complicated relations.

#### 2.1.2 Picking intervals on equal distance.

This is an improvement version of the first way. We'll take average value of all the points in a particular distance instead of picking a single point. For example, take average for the frequency and energy in intervals [0, 10%], [10%, 20%], ..., [90%, 100%], and this produces a 20-dimensional vector. We hope that such approach can help to denoise and produce a robust feature.

It turns out that the performance is slightly better than the previous one, reaching a training accuracy around 85%. But it's also not satisfying. So we would like to explore a method to produce more direct feature that can be used for learning.

#### 2.1.3 Least Square Method on equal distance.

We did survey on the Internet, and find an intersting idea for extracting feature. That is, to calculate the slope of the frequency of points in a particular distance. For example, calculate the slope of frequency of the points in [0, 20%], [20%, 40%], ..., [80%, 100%]. This results in a 5-d dimensional vector, which can be used for training. We think this could make sense because the slope is the direct metric for classifying tone.

This method does work. The training and first testing acc achieved higher than 97%. As for the second testing data, the acc is around 80%. So we need to combine it with the denoising method mentioned before to get the ideal result.

### 2.2 Denoising

Noise would interfere the energy and the frequency of the recorded data significantly. Even if we try our best to design sophisticated feature, the pre-

dicting accuracy of the second test data is not satisfiable. Therefore, we develop some methods to denoise the data.

### 2.2.1 Deleting the part of data with low energy

Since light sound might be covered by noise, we believe the data with low energy is highly unreliable. So the first step of denoising is to remove the data whose energy is lower than others' significantly. In our model, we consider the average of the energy $E_{\text{mean}}$ in each data is the standard energy of the pronunciation which the data present. And the data whose energy is lower than $\epsilon E_{\text{mean}}$ should be removed where $\epsilon$ is a parameter need to be tuned. We set $\epsilon = 0.2$.

The result shows that if we delete the data with low energy previously, the accuracy on the second test data could achieve about 88%.

### 2.2.2 Deleting the part of data with abnormal frequency

The other factor could be interfered by noise is frequency. Some part of data will be abnormal and we want to remove them too. In our model, we first statistic some properties on the trainning data, and then use them to filter the normal frequency. It should be mentioned that the data we deal with is what we get after deleting the data with low energy.

- **Step 1**: For the $i$th tone, suppose $\log f_{k,p+1} - \log f_{k,p}$ is i.i.d where $f_{k_p}$ is the $p$th frequency in the $k$th data. Here we use logarithm because the frequency is exponentially related to the pitch. Now we statistic the mean and the standard deviation of $\log f_{k,p+1} - \log f_{k,p}$ and denote them as $M_i$ and $\sigma_i$. Here is the result which we statictic from the whole trainning set.

| $i$ | $M_i$ | $\sigma_i$ |
|---|---|---|
| 1 | -0.00368890111513 | 0.0502888828252 |
| 2 | 0.00842593986922 | 0.0501864781776 |
| 3 | 0.00261934419705 | 0.0358012690812 |
| 4 | -0.0325935548158 | 0.0630625440446 |

Here $M_2$ is the largest and $M_4$ is the smallest, which is intuitively.

- **Step 2**: Now we use these properties to filter the data. For each data, we first guess it's tune is $i$. As we have assumed that $\log f_{k,p+1} - \log f_{k,p}$ is i.i.d, we can calculate the mean and the variance of $\log f_{k,p+t} - \log f_{k,p}$ should be $M_i$ and $\sqrt{t}\sigma_i$. We could accept the frequency with $M_i - 2\sqrt{t}\sigma_i \leq \log f_{k,p+t} - \log f_{k,p} \leq M_i + 2\sqrt{t}\sigma_i$. Otherwise, one of $f_{k,p+t}$ and $f_{k,p}$ must be removed. Here, for a fixed $p$, $f_{k,p}$ can follow with some previous frequency, and we select the nearest one. Let $l_i$ be the frequence subsequence of the data with the maximum length if we guess it's tune is $i$. And we select pick the longest subsequence from $\{l_1, l_2, l_3, l_4\}$ as the result of data processing.

Combing these two methods, the test accuracy could achieve about 95% and the highest accuracy is about 96.1%.

# 3 Network structure

Our network structure is very simple but performs pretty good. To be specific, our neural network have totally four layers. The input layer contains 14 vertexes, which repesents 7 means and 7 slopes. The first layer is a fully connected layer, which has four vertexes. The second layer is the relu function. The third layer is also fully connected. The last layer is the output layer, which uses softmax function. It looks like the figure 1.

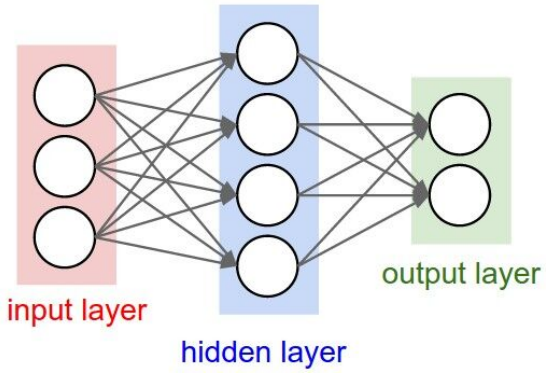


Figure 1: Our network.

# 4 Performance

## 4.1 MxNet

Normally, the algorithm needs around 1000 epoch to converge. When it converges, its training accuracy can arrive 98% to 99%. On the first testing set with 40 cases, the model can pass 39 or 40 cases. On the second testing set (test_new), the accuracy could arrive 92% to 96%.

## 4.2 Theano

Normally, the algorithm needs around 50000 epoch to converge and it does not have mini-batch. When it converges, its training accuracy can arrive 99%. On the first testing set with 40 cases, the model can pass 39 or 40 cases. On the second testing set (test_new), the accuracy could arrive 92% to 96%. And the model using Theano is not exactly same as others because Theano does not provide the network model. So we can only use the faster gradient function and set the original parameters to train the network. Besides, learning rate is different from models using CNTK and MxNet.

## 4.3 CNTK

Normally, the algorithm needs around 3000 epoch to converge. When it converges, its training accuracy can arrive 98% to 99%. On the first testing set with 40 cases, the model can pass 39 or 40 cases. On the second testing set (test_new), the accuracy could arrive 94% to 96%.

# 5 Comparison between different toolkits

## 5.1 Compiling time

| | MxNet | Theano | CNTK |
|---|---|---|---|
| Compiling time(s) | 6.6e-3 | 3.1e-1 | 1.6e-3 |

Table 1: Comparison of compiling time

Theano is well-known for its long compiling time. Since our model is very simple, it doesn't take a long time to compile the computational graph. Howev-

er, it can be seen that the compiling time for Theano is around 100 times longer than the other two.

## 5.2 Running time

| | MxNet | Theano | CNTK |
|---|---|---|---|
| Running time per epoch(s) | 1.67e-2 | 8.30e-5 | 2.75e-2 |
| Epoch number | 1000 | 50000 | 3000 |
| Total training time(s) | 16.72 | 4.15 | 82.36 |

Table 2: Comparison of running time

As for the training time, it can be seen that Theano has a greater advantage over the other two, with running time around 100 times faster than the other two. Since it uses SGD rather than Adam, it takes longer time to converge. But the speed is still faster than the other two. Also, the speed of MxNet is faster than CNTK.

## 5.3 Accuracy

| | MxNet | Theano | CNTK |
|---|---|---|---|
| Accuracy on test | 100% | 100% | 100% |
| Accuracy on test_new | 96.1% | 94.7% | 95.2% |

Table 3: Comparison of accuracy.

The accuracy is achieved by running each model three times and take the highest one (considering the randomness in the process of model training). MxNet achieves the highest testing accuracy. We owe this to its excellent training optimizer APIs. See next page for our testing accuracy curve.

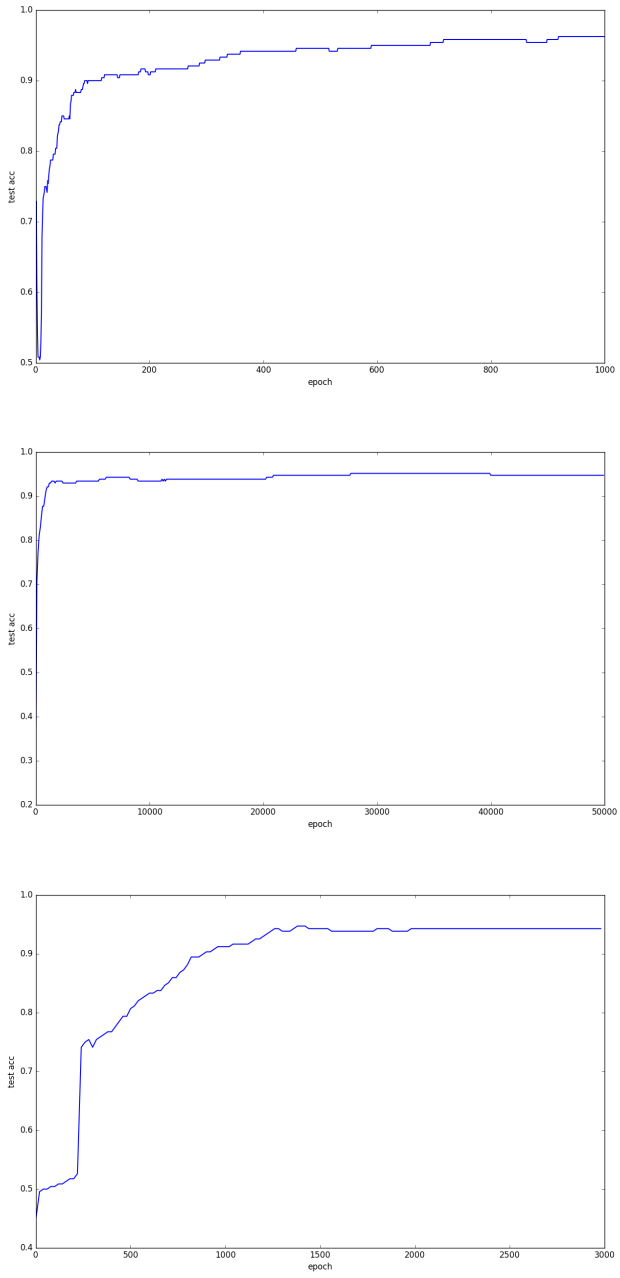


Figure 2: The testing accuracy curve. The above one is MxNet, center one Theano and bottom one CNTK.

### 5.4 APIs

We think that MxNet and CNTK are similar in the APIs they offer in the situation of our case. They both provides some integrated functions for neural network. These functions can make our code simple and readable.

Theano, on the other hand, is a rather basic toolkit. It doesn't offer APIs for neural networks, such as fully connected layer or various optimizers, and we have to manually operate gradient descent on it. But this also means that it offers more room to do different kind of things, and provides faster running time. In addition, its installation process is the easiest among the three.

## 6 Strengths and Weaknesses of our model

### 6.1 Strengths

- The feature we extract is very accurate and reasonable. We observe the training dataset combined with life experience. The change in frequency leads to a fundamental difference in tone.
- The method we use to eliminate noise is also effective. We pioneered a dynamic programming approach to solve the problem, which achieved very good results.
- Although our model is very simple, it performs pretty good in practice. The accuracy is very high and over 95%. And our epoch is very small and the accuracy converges very quickly because of the simple network.

### 6.2 Weaknesses

- In order to extract the most important features, we dropout much data that is useless in our eyes. However, it is possible that these data contains important information in some sense but we ignore them.
- Although our model can even achieve 97% accuracy for test data, it is unstable.

# 7 Future work

- We can try other method we have learned in data science to eliminate noises such as PCA, wavelet and Fourier transform.
- In the beginning of our experiment, we wanted to take the logarithm of frequency to train the data but failed. However, The frequency should be exponentially related to the pitch. Pitch is a physical property so it should make sense. So it is valuable to explore the problem deeply.
- We can use more complex and deeper neural network, maybe it can achieve the better accuracy.

# 8 Conclusion

We established a model for Chinese tone classification. Our model includes denoising, feature extraction and a shallow neural network. The model can achieve accuracy 96% even for the data with significant noise. So we think that our model is quite robust.

We conclude that when facing small dataset, the key point of machine learning is the part of feature extraction, but not the network structure. We've tried complex network structure but the performance is worse, because it overfits. As a result, we mainly focus on how to get appropriate features that both represent the data property and reduce the noise. Actually, it works!

We also do comparision between different toolkits. We found that different toolkits will focus on different aspects. MxNet and CNTK mainly focus on the utility of neural network and thus provides useful and beautiful APIs. Theano, on the other hand, can handle a larger variety of tasks and is more flexible. As for performance, CNTK has the fastest compiling time, Theano has the fastest running time and MxNet has the highest accuracy. People should choose the suitable toolkit depending on their usage.

# 9 Reference

http://cs231n.stanford.edu/
https://www.google.com/imghp?hl=zh-cn
http://www.siat.cas.cn/xscbw/xsqk/201202/W020120214579607833211.pdf