第1章　误差理论

陈路　CHENLU.SCIEN@GMAIL.COM
201800301206

本文首先简要总结误差理论的相关知识1，进而以二次方程求根问题为例讨论减少误差的具体方法，并以幂次运算问题为例讨论减少运算次数的具体策略。

1. 误差理论知识总结

2. 分析讨论题

2.1 二次方程求根

问题 1
求方程$x^2+(\alpha+\beta)x+10^9=0$的根，其中，$\alpha=-10^9,\beta=-1$，讨论如何设计计算格式才能有效地减少误差，提高计算精度.

$$0 \le P(v) \le 1.0 \text{ for each configuration } v \in V\text{; and} \tag{1}$$

$$\sum_{v\in V} P(v) = 1.0 \tag{2}$$

Referring to Equation 2 here.

2.2 幂次运算

问题 2
以计算x^{31}为例，讨论如何设计计算格式才能减少计算次数.

Referring to Table 1 (a) here.

2.3 Example Subsection

This is how you use math items inline: $(\theta_a \star \lambda_a)$, $(\theta_{ab} \star \lambda_b)$, $(\theta_{a\bar{b}} \star \lambda_{\bar{b}})$, $(\lambda_b \star \theta_{\bar{a}b})$, $(\lambda_{\bar{b}} \star \theta_{\bar{a}\bar{b}})$, and $(\lambda_{\bar{a}} \star \theta_{\bar{a}})$.

Referring to Figure 1.

```
import numpy as np
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Layer, Activation
import time
```

Table 1: Table (a) provides the prior probability of variable A and Table (b) provides the conditional probability of B given A.

(a)

A	$P(A)$
1	0.3
0	0.7

(b)

A	B	$P(B\|A)$
1	1	0.1
1	0	0.9
0	1	0.8
0	0	0.2

```python
import tensorflow as tf

f = open("results.csv", "w")


INPUT_SIZE = 10
OUTPUT_SIZE = INPUT_SIZE
nb_class = 3

batch_size = 128
nb_epoch = 40

np.random.seed(123)

X_train = np.random.rand(INPUT_SIZE, nb_class)
Y_train = np.random.rand(OUTPUT_SIZE, nb_class)

X_test = np.random.rand(INPUT_SIZE)
Y_test = np.random.rand(OUTPUT_SIZE)

for i in range(1,51):

    start_time = time.time()

    model = Sequential()
    model.add(Dense(INPUT_SIZE, input_shape=(nb_class,)))
    model.add(Activation('linear'))
    model.add(Dense(OUTPUT_SIZE))
    model.add(Activation('linear'))
    model.compile(loss='categorical_crossentropy', optimizer='rmsprop')

    final_time = time.time()
    diff_time = final_time - start_time

    f.write(str(i)+","+str(diff_time)+","+"\n")

f.close()
```

Table 2: Comparison of parallel and serial solutions for ACs with 50 runs each

Solution	Time average	Standard deviation
Serial	0.1610	0.0536
Parallel	0.0434	0.0082

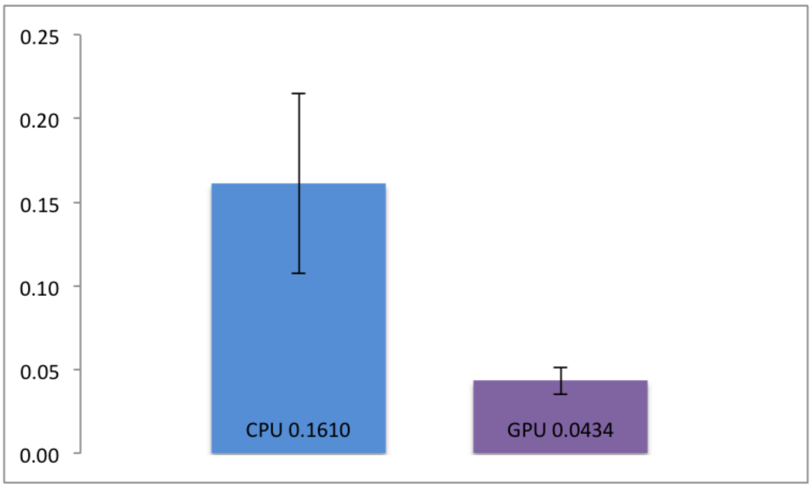

Figure 1: Comparison between CPU and GPU implementation of AC compiling.

3. Conclusion

In nec hendrerit arcu. Pellentesque leo libero, fringilla consectetur sapien eget, placerat finibus justo. Aliquam id sapien in eros lacinia euismod. Sed consectetur eros quis dui vestibulum pulvinar. Suspendisse ligula lacus, blandit interdum convallis sit amet, vulputate eget quam. Curabitur justo nunc, efficitur vitae fringilla eget, suscipit vitae sem. Etiam ultricies, nibh dictum convallis viverra, ipsum magna vehicula nibh, eu elementum purus est ac risus. Suspendisse sollicitudin auctor urna vel aliquet. In at eros et elit mollis lacinia. Fusce auctor leo id metus porttitor, vulputate semper erat congue. Donec rutrum erat non mauris convallis, id feugiat velit facilisis. Sed interdum magna sit amet mauris elementum pellentesque.