MapReduce

Bean：bean 类里定义了需要的实体类，主要用于接收 json 文件中的字段，并用阿里云提供的第三方 jar 包 json 将他们转化成实体类

CourseInfo：

```
package Bean;

import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;
import java.util.List;

import org.apache.hadoop.io.Writable;

public class CourseInfo implements Writable{
	private List<String> item;
	private String courseId;

	public CourseInfo() {
	}

	public CourseInfo(List<String> item,String courseId) {
		super();
		this.item = item;
		this.courseId = courseId;
	}

	public void setItem(List<String> item) {
		this.item = item;
	}

	public List<String> getItem(){
		return this.item;
	}

	public void setCourseId(String courseId) {
		this.courseId = courseId;
	}

	public String getCourseId() {
		return this.courseId;
	}
```

```
@Override
public void readFields(DataInput in) throws IOException {
	// TODO Auto-generated method stub
	while(in.readLine() != null) {
		this.item.add(in.readLine());
	}
	this.courseId = in.readLine();
}

@Override
public void write(DataOutput out) throws IOException {
	// TODO Auto-generated method stub
	for(int i = 0;i<this.item.size();i++)
		out.writeUTF(this.item.get(i));
	out.writeUTF(this.courseId);
}
```

（后面的 construct、get、set 方法不再做截图）

Prerequisite：

```java
package Bean;

import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;

import org.apache.hadoop.io.Writable;

public class Prerequisite implements Writable{
	private String conceptB;
	private String conceptA;
```

```java
@Override
public void readFields(DataInput in) throws IOException {
	// TODO Auto-generated method stub
	this.conceptA = in.readLine();
	this.conceptB = in.readLine();
}

@Override
public void write(DataOutput out) throws IOException {
	// TODO Auto-generated method stub
	out.writeUTF(this.conceptA);
	out.writeUTF(this.conceptB);
}
```

ProblemAct：

```java
package Bean;

import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;

import org.apache.hadoop.io.Writable;

public class ProblemAct implements Writable{
	private String problemId;
	private String studentId;
	private int label;
```

```java
@Override
public void readFields(DataInput in) throws IOException {
	// TODO Auto-generated method stub
	this.problemId = in.readLine();
	this.studentId = in.readLine();
	this.label = in.readInt();
}

@Override
public void write(DataOutput out) throws IOException {
	// TODO Auto-generated method stub
	out.writeUTF(this.problemId);
	out.writeUTF(this.studentId);
	out.writeInt(label);
}
```

ProblemActivity：

```java
package Bean;

import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;
import java.util.List;

import org.apache.hadoop.io.Writable;

public class ProblemActivity implements Writable{
        private String actId;
        private String problemId;
        private String studentId;
        private String time;
        private String content;
        private String courseId;
        private List<String> concept;
        private int label;
```

```java
  @Override
  public void readFields(DataInput in) throws IOException {
          // TODO Auto-generated method stub
          actId = in.readLine();
          problemId = in.readLine();
          studentId = in.readLine();
          time = in.readLine();
          content = in.readLine();
          courseId = in.readLine();
          while(in.readLine()!=null)
                  concept.add(in.readLine());
          label = in.readInt();
  }

  @Override
  public void write(DataOutput out) throws IOException {
          // TODO Auto-generated method stub
          out.writeUTF(actId);
          out.writeUTF(problemId);
          out.writeUTF(studentId);
          out.writeUTF(time);
          out.writeUTF(content);
          out.writeUTF(courseId);
          for(int i=0;i<concept.size();i++)
                  out.writeUTF(concept.get(i));
          out.writeInt(label);
  }
```

Map:主要做 json 文件的读入,并将读入的内容赋值到实体类中,运用 json.parseObject 实现

AnswerMap：

```java
package Map;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Mapper.Context;

import com.alibaba.fastjson.JSON;

import Bean.ProblemAct;

public class AnswerMap extends Mapper<LongWritable, Text, Text, IntWritable>{
        private IntWritable one = new IntWritable();
        private Text word = new Text();
        ProblemAct problemActTrain1 = new ProblemAct();
        ProblemAct problemActTrain = new ProblemAct();
        @Override
    protected void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
                problemActTrain1 = JSON.parseObject(value.toString(), ProblemAct.class);
                problemActTrain.setProblemId(problemActTrain1.getProblemId());
                problemActTrain.setStudentId(problemActTrain1.getStudentId());
                problemActTrain.setLabel(problemActTrain1.getLabel());
                word.set(problemActTrain.getProblemId());
                one.set(problemActTrain.getLabel());
                context.write(word,one);

//              context.write(problemActTrain,NullWritable.get());

        }
}
```

PrerequisiteMap:

```java
package Map;

import java.io.IOException;

import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import com.alibaba.fastjson.JSON;

import Bean.Prerequisite;

public class PrerequisiteMap extends Mapper<LongWritable,Text,Prerequisite,NullWritable>{
        Prerequisite prerequisites = new Prerequisite();
        Prerequisite prerequisite = new Prerequisite();

        @Override
    protected void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
                // JSON -> prerequisites
                prerequisites = JSON.parseObject(value.toString(),prerequisites.getClass());
                // get key from prerequisites
                prerequisite.setConceptA(prerequisites.getConceptA());
                prerequisite.setConceptB(prerequisites.getConceptB());
                // write out
                context.write(prerequisite, NullWritable.get());
        }
}
```

ProblemActUnionMap：

```java
package Map;

import java.io.IOException;

import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import com.alibaba.fastjson.JSON;

import Bean.ProblemAct;

public class ProblemActUnionMap extends Mapper<LongWritable, Text, ProblemAct, NullWritable>{
        ProblemAct problemActTrain1 = new ProblemAct();
//      ProblemAct problemActTrain2 = new ProblemAct();
        ProblemAct problemActTrain = new ProblemAct();
        @Override
    protected void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
                problemActTrain1 = JSON.parseObject(value.toString(), ProblemAct.class);
//              problemActTrain2 = JSON.parseObject(value.toString(), ProblemAct.class);
                problemActTrain.setProblemId(problemActTrain1.getProblemId());
                problemActTrain.setStudentId(problemActTrain1.getStudentId());
                problemActTrain.setLabel(problemActTrain1.getLabel());
                context.write(problemActTrain,NullWritable.get());
//              problemActTrain.setProblemId(problemActTrain2.getProblemId());
//              problemActTrain.setStudentId(problemActTrain2.getStudentId());
//              problemActTrain.setLabel(problemActTrain2.getLabel());
//              context.write(problemActTrain,NullWritable.get());
        }
}
```

Reduce：主要对读入的数据做处理，可以通过调用 MapReduce 提供的各类 class（比如 sort、wordcount 等）、或者自己写对应的函数实现

AnswerReduce：统计数据集中各个问题正确回答的个数。（改编自 wordcount）

```java
package Reduce;

import java.io.IOException;
import java.util.Iterator;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.examples.WordCount;
import org.apache.hadoop.examples.SecondarySort.Reduce;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.Reducer.Context;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import com.alibaba.fastjson.JSONObject;

import Bean.ProblemAct;
import Map.AnswerMap;
import Map.ProblemActUnionMap;

public class AnswerReduce extends Reducer<IntWritable, ProblemAct, Text, IntWritable>{
        private static IntWritable linenum = new IntWritable(1);

    private IntWritable result = new IntWritable();
    public void Reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable, Text,
IntWritable>.Context context) throws IOException, InterruptedException {
        int sum = 0;
        IntWritable val;
        for(Iterator i$ = values.iterator(); i$.hasNext(); sum += val.get()) {
            val = (IntWritable)i$.next();
        }
        this.result.set(sum);
        context.write(key, this.result);
    }
```

ProblemActUnionReduce：

```
public class ProblemActUnionReduce extends Reducer<Text,Text, Text, NullWritable>{

	private static IntWritable linenum = new IntWritable(1);

	protected void reduce(ProblemAct key, Iterable<IntWritable> values, Context context, Object
JSONObject) throws IOException, InterruptedException {
		for (IntWritable val : values) {
			JSONObject json = new JSONObject();
			json.put("student_id",key.getStudentId());
			json.put("problem_id",key.getProblemId());
			json.put("label",key.getLabel());
	    context.write(new Text(json.toString()),null);
	    linenum = new IntWritable(linenum.get() + 1);
	}
	}
```

运行代码：将写好的 map 类、reduce 类汇总，并执行输出

Answer:

```
public static void main(String[] args) throws Exception{
	    Configuration conf = new Configuration();
    Job job = new Job(conf, "Answer");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(AnswerMap.class);
    job.setReducerClass(Reduce.class);
    job.setOutputKeyClass(IntWritable.class);
    job.setOutputValueClass(ProblemAct.class);
    FileInputFormat.addInputPath(job, new Path("localhost:9000/hadoop/usr/local/hadoop/input/"));
    FileOutputFormat.setOutputPath(job, new Path("localhost:9000/hadoop/usr/local/hadoop/output/"));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
```

ProblemActUnion：

```
public static void main(String[] args) throws Exception{
	Configuration conf = new Configuration();
Job job = new Job(conf, "Union");
job.setJarByClass(Sort.class);
job.setMapperClass(ProblemActUnionMap.class);
job.setReducerClass(Reduce.class);
job.setOutputKeyClass(IntWritable.class);
job.setOutputValueClass(ProblemAct.class);
FileInputFormat.addInputPath(job, new Path("localhost:9000/hadoop/usr/local/hadoop/input/"));
FileOutputFormat.setOutputPath(job, new Path("localhost:9000/hadoop/usr/local/hadoop/output/"));
System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```

运行结果：

ProblemActUnion：

```json
[
  {
    "student_id": "U_66",
    "problem_id": "P_a0f85c0032554e18b12167e664fc5e0f",
    "label": 1
  },
  {
    "student_id": "U_66",
    "problem_id": "P_548651d5d84243b7be436954c513d644",
    "label": 1
  },
  {
    "student_id": "U_66",
    "problem_id": "P_f028b11627024fd1912eafd979920286",
    "label": 1
  },
  {
    "student_id": "U_141",
    "problem_id": "P_6773b17fc36b4a68acc6f92bfbc36ecd",
    "label": 1
  },
  {
    "student_id": "U_143",
    "problem_id": "P_a5c0adc0f137489481668c4707381582",
    "label": 1
  },
  {
    "student_id": "U_143",
    "problem_id": "P_2ecdfd6c130a4384900f59e3ba63a664",
    "label": 1
  },
  {
    "student_id": "U_143",
    "problem_id": "P_f52bdc7e03cb42b2aecd3d03b35633ad",
    "label": 1
  },
```

Answer：

```
'P_c623a61a88154ab4a69a697c1ea55e2b': 110,
'P_cd21c1cdb2d54a3cbf716a2bca74d5a2': 114,
'P_7dbab9b976c34661a5278a9544f0422c': 118,
'P_90281b2918374a33a3fd660c0eb9e672': 122,
'P_a511a5048bae4a488fa5d1e07c747180': 126,
'P_25073478e7954a5b9f558d04b631e5e7': 130,
'P_ca9c815b2b3e4ae3b90f36e83acbd6ef': 134,
'P_d67e5b2e5bce4ebe9ed779b98eb9f62a': 138,
'P_835633b273b24973807cd5eee35f33dd': 142,
'P_c8a4b5a72f4d454d8fd19805f040907f': 146,
'P_4ba8ba6f0bd14f11b3d7bf2b89fd7b4a': 150,
'P_e670ae4e0ab6428885f6dea70b6058d5': 154,
'P_bd9d199a4d27470d921f8f2685d6bc02': 158,
'P_f6a9555585f543c8b5f137fa599444f9': 162,
'P_cfc263b47c2d4224b37a5b51a6db39d3': 166,
'P_cc75baba80124153ab498cb41e1b0cd0': 170,
'P_f8fc6298e31247a7aeaf288a629aab86': 174,
'P_33ee03934ddb45b3ab6c500b32b99504': 178,
'P_37dc889b8c3e476e9c55d1852831de65': 182,
'P_8982158634c84729a39ba5a80a17c4b2': 186,
'P_a88d0e94cafd49e6b1c5c8be36397237': 190,
'P_de3a7c1e3ba74350b0aaaddc2aad0832': 194,
'P_d9ffa67f921d47729ae3b6beaaa35227': 198,
'P_49c91a8acde14291b077795f9c8c3117': 202,
'P_7a508df6e0c24f30a694b6e927c72024': 206,
'P_c69585fa1a444af5b562872516c4425d': 210,
'P_8f85956f33134adaac6b8a7af2672a16': 214,
'P_184e8367583c48179fcec86353982285': 218,
'P_ae017f314ef44d60905a33cd6f5a97c7': 222,
'P_b6acaa16d61749c38c6113ae49f16343': 226,
'P_4d3060dea4b34b5e9de497089a418db3': 230,
'P_ddcc798d6ab640cbba0607af7ff86ec8': 234,
'P_6d5a72ff375e499cb41de3d712a00805': 238,
'P_5ef3b905854b4cc28237f6b7397082f5': 242,
'P_124845ff34eb406cb9b6d9b0878fedac': 246,
'P_52d2468622a74067ace70a3334923408': 250,
'P_daa9687c8f874e61be8cb6334c4a32be': 254,
'P_8bc1568fb7914d099a2521990741f9ba': 258,
'P_b742f699e71f40d8bed268d873dfec1a': 262,
'P_9a90cb57cc8c4b2d8f7523d123eb4d29': 266,
```

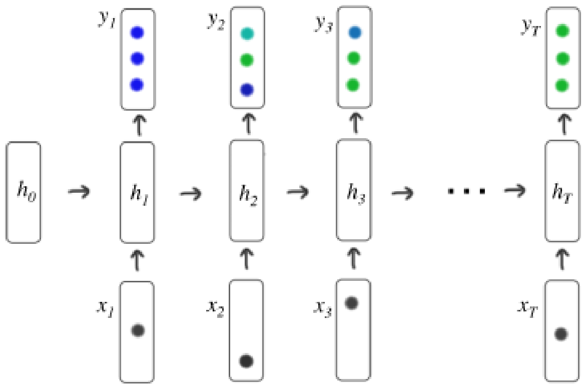
References:

[1].走向云计算之 MapReduce 应用案例详解
[2].Hadoop 使用 MapReduce 处理百万行 Json 格式数据
[3].使用 MapReduce 将 JSON 数据进行分类
[4]. MapReduce 输出 json 几种格式
[5]. MapReduce 实现数据去重、数据排序、求平均值、多列单行输出

模型

DKT：
参考论文 Deep Knowledge Tracing[1]搭建了 model 并进行了训练，该文中采用了传统的 RNN 网络结构，并对于最终的结果通过测量曲线下的面积（AUC）来进行判断。网络结构及相关公式如下所示：

$$\mathbf{h}_t = \tanh(\mathbf{W}_{hx}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h)$$
$$\mathbf{y}_t = \sigma(\mathbf{W}_{yh}\mathbf{h}_t + \mathbf{b}_y)$$



其中，输入x_t是单次 one-hot 编码或学生动作的压缩表示，而预测值y_t是代表使每个数据集练习正确的概率的向量。h_t 是 t 时刻的隐状态，也是网络的“记忆”，h_t 的计算依赖于前一个时刻的隐状态和当前时刻的输入：$h_t = f(Ux_t + Wh_{t-1})$，通过输入权重矩阵W_{hx}，循环权重矩阵W_{hh}，初始状态h_0和读出权重矩阵W_{yh}对模型进行参数化。潜在单位和读出单位的偏差由b_h和b_y给出。
具体的代码实现如下所示：

```
class DKT(nn.Module):
    def __init__(self, input_dim, hidden_dim, layer_dim, output_dim):
        super(DKT, self).__init__()
        self.hidden_dim = hidden_dim
        self.layer_dim = layer_dim
        self.output_dim = output_dim
        self.rnn = nn.RNN(input_dim, hidden_dim, layer_dim, batch_first=True, nonlinearity='tanh')
        self.fc = nn.Linear(self.hidden_dim, self.output_dim)
        self.sig = nn.Sigmoid()

    def forward(self, x):
        h0 = Variable(torch.zeros(self.layer_dim, x.size(0), self.hidden_dim))
        # print(x)
        out, hn = self.rnn(x, h0)
        res = self.sig(self.fc(out))
        return res
```

因为该问题本质上是一个二分类问题（对于某一道题，学生 A 要么做对要么做错），因此采用 f1_score、recall_score 以及 precision_score 进行判断，相关公式及代码如下所示：

f1_score：

F1 值为精度（P）和召回率（R）的调和平均值。

$$\frac{2}{F_1} = \frac{1}{P} + \frac{1}{R} \Rightarrow F_1 = \frac{2PR}{P+R}$$

精度（P）定义：

$$R = \frac{tp}{tp + fp}$$

召回率（R）定义：

$$R = \frac{tp}{tp + fn}$$

tp--将正类预测为正类（true positive）：tp = sum(y_true & y_pred)

fn--将正类预测为负类（false negative）：fn = sum((y_true == 1) & (y_pred == 0))

fp--将负类预测为正类（false positive）：fp = sum((y_true == 0) & (y_pred == 1))

tn--将负类预测为负类（true negative）：tn = sum((y_true == 0) & (y_pred == 0))

recall_score：被预测正确的样本占样本总量的比例，即正确率

$$recall = \frac{tp}{tp + fn} \Rightarrow recall = \frac{tp}{p}$$

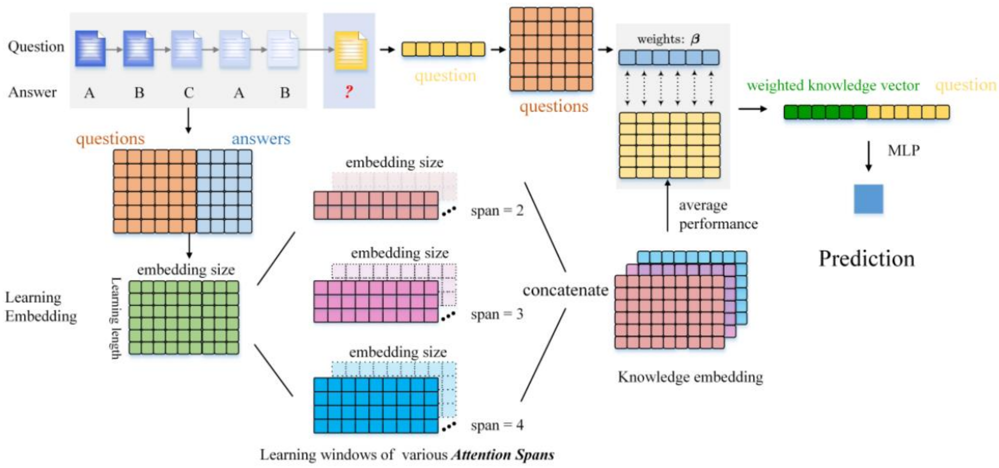
precision_score：

$$\text{Precision} = \frac{TP}{(TP + FP)}$$

具体实现可以通过调用 sklearn 中的 metrics 包实现，如下：

```
def performance(gt, pred):
    fpr, tpr, thresholds = metrics.roc_curve(gt.detach().numpy(), pred.detach().numpy())
    auc = metrics.auc(fpr, tpr)
    f1 = metrics.f1_score(gt.detach().numpy(), torch.round(pred).detach().numpy())
    recall = metrics.recall_score(gt.detach().numpy(), torch.round(pred).detach().numpy())
    precision = metrics.precision_score(gt.detach().numpy(), torch.round(pred).detach().numpy())
    acc = metrics.accuracy_score(gt.detach().numpy(), torch.round(pred).detach().numpy())
    print('auc:' + str(auc) + '\nf1:' + str(f1) + '\nrecall:' + str(recall) + '\nprecision:' + str(precision) + '\nacc:' + str(acc))
```

CKT：(参考 git 上的 demo，模型代码与网络结构如下，地址在报告末尾给出)



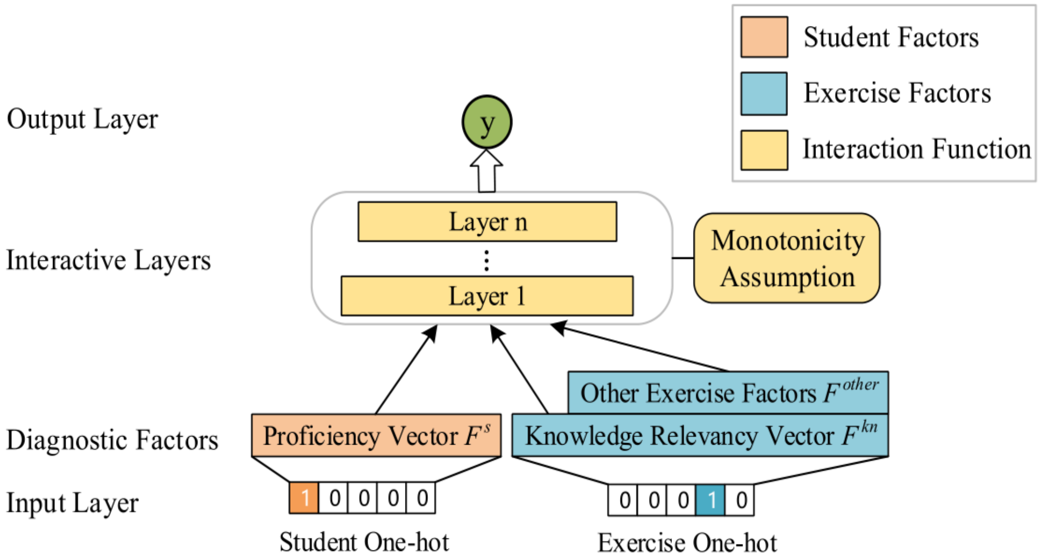

```
class CKT(object):

    def __init__(self, batch_size, num_skills, hidden_size):

        self.batch_size = batch_size = batch_size
        self.hidden_size = hidden_size
        self.num_skills = num_skills


        self.input_id = tf.compat.v1.placeholder(tf.int32, [None, 29], name="input_id")
        self.x_answer = tf.compat.v1.placeholder(tf.int32, [None, 29], name="x_answer")
        self.x_answer1 = tf.compat.v1.placeholder(tf.float32, [None, 29,2], name="x_answer1")
        self.valid_id = tf.compat.v1.placeholder(tf.int32, [None,], name="valid_id")
        self.target_correctness = tf.compat.v1.placeholder(tf.float32, [None], name="target_correctness")
        self.target_correctness1 = tf.compat.v1.placeholder(tf.float32, [None,2], name="target_correctness1")
        self.seq_length = tf.compat.v1.placeholder(tf.int32, [None,], name="seq_length")
        self.dropout_keep_prob = tf.compat.v1.placeholder(tf.float32, name="dropout_keep_prob")
        self.is_training = tf.compat.v1.placeholder(tf.bool, name="is_training")

        self.global_step = tf.compat.v1.Variable(0, trainable=False, name="Global_Step")
        self.initializer=tf.compat.v1.keras.initializers.VarianceScaling()
```

NeuralCDM[2]:

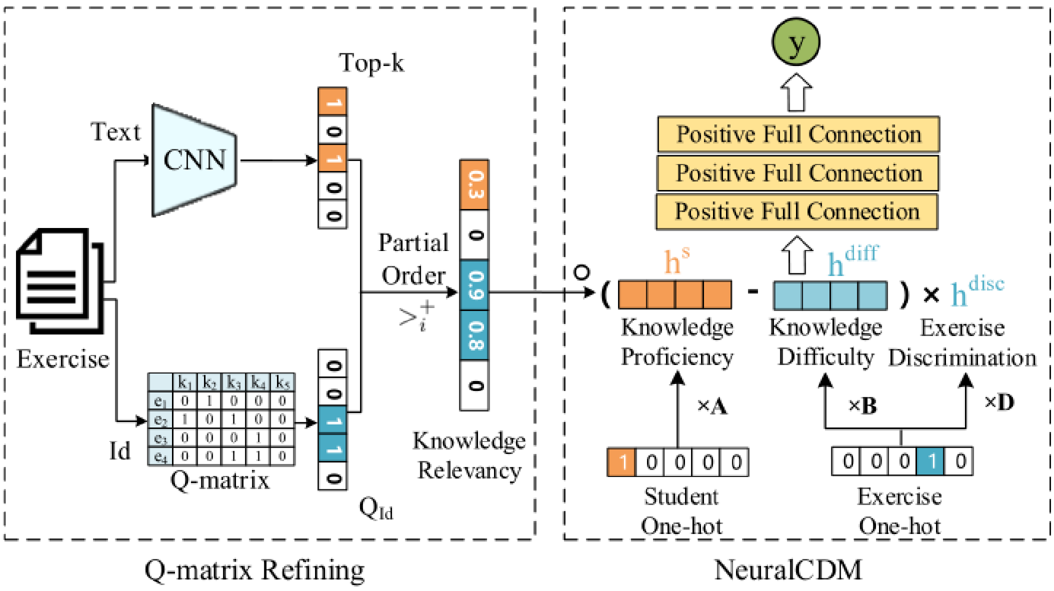
NeuralCD 分为三个部分：学生因素、试题因素和交互函数（在图中分别用不同颜色标注）。学生因素最基本的是知识点熟练度向量 F^S，试题因素则包括基本的知识点相关度向量F^{kn}和其他可选因素 F^{other}（如试题难度、区分度等）。

交互函数则由多层神经网络构成，这是与传统方法最大的不同点。NeuralCD 的输入为答题记录中学生和试题的 one-hot 向量，输出为学生的知识掌握程度。

神经网络结构如图所示：



其中，对于输入部分（F^S、F^{kn}、 F^{other}），论文中做了进一步的处理：



其中，学生的知识点熟练度向量（ F^S ）在此处具体为 $\boldsymbol{h}^s = \mathrm{sigmoid}(\boldsymbol{x}^s \times A)$，其中 $\boldsymbol{x}^s$是学生的 one-hot 向量,A 是所有学生的熟练度矩阵。试题因素中知识点相关度向量（ F^{kn} ）直接取自人工标注的 Q 矩阵，具体为$Q_e = \boldsymbol{x}^e \times Q$，其中 $\boldsymbol{x}^e$是试题的 one-hot 向量；此外还使用了试题的知识点难度向量 $\boldsymbol{h}^{diff} = \mathrm{sigmoid}(\boldsymbol{x}^e \times B)$表示试题对每个知识点考察的难度，以及试题区分度 $\boldsymbol{h}^{disc} = \mathrm{sigmoid}(\boldsymbol{x}^e \times D)$表示试题区分不同水平学生的能力。A、B、D 都是通过数据学习的可训练参数。这样，多层神经网络的输入层为：

$$\mathbf{x} = Q_e \circ \left(\mathbf{h}^s - \mathbf{h}^{diff}\right) \times h^{disc}$$

NeuralCDM 不追求模型的复杂性，而是为验证 NeuralCD 的有效性，因此对于输入做了较多的处理，并且其交互函数由最常见的多层全连接层构成。为满足单调性假设，我们限定每一层的权值为正（可部分为 0），这样能够保证$\frac{\partial y}{\partial h_i^s} \geq 0$，熟练度向量中值$h_i^s$的调整方向与输出

的预测值 y 的变化方向一致。最后，训练时采用预测值与真实得分的交叉熵：

$$\text{loss}_{CDM} = -\sum_{i}\left(r_i \log y_i + (1-r_i)\log(1-y_i)\right)$$

训练结束后，学生对应的$\mathbf{h}^s$即为该学生的诊断结果，每一维对应该学生在该知识点上的掌握程度（范围(0,1)）。

训练代码：

```python
def train():
    dataloader =TrainDataLoader()
    device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')

    model = NeuralCDM(params.student_num, params.exercise_num, params.knowledge_num)
    model = model.to(device)
    optimizer = optim.Adam(model.parameters(), lr=0.002)
    validate(model, 0)

    print('Training...')
    print('Device:', torch.cuda.get_device_name(0), torch.cuda.current_device(), 'count:', torch.cuda.device_count())
    loss_function = nn.NLLLoss()
    for epoch in range(params.epoch_num):
        dataloader.reset()
        running_loss = 0.0
        batch_count = 0
        while not dataloader.is_end():
            batch_count += 1
            input_stu_ids, input_exer_ids, input_knowledge_embs, labels = dataloader.next_batch()
            input_stu_ids, input_exer_ids, input_knowledge_embs, labels =input_stu_ids.to(device), input_exer_ids.to(device), input_knowled
            optimizer.zero_grad()
            output_1 = model(input_stu_ids, input_exer_ids, input_knowledge_embs)
            output_0 = torch.ones(output_1.size()).to(device) - output_1
            # output_0 = torch.ones(output_1.size()) - output_1

            output = torch.cat((output_0, output_1), 1)

            loss = loss_function(torch.log(output), labels)
            loss.backward()
            optimizer.step()
            model.apply_clipper()
```

```python
            running_loss += loss.item()

            if batch_count % 200 == 199:
                print('[%d, %5d] loss: %.3f' % (epoch + 1, batch_count + 1, running_loss / 100))
                running_loss = 0.0

        validate(model, epoch)
    return model
```

```python
def validate(model, epoch):
    dataloader = ValTestDataLoader('validation')
    # print('Validating...')
    dataloader.reset()
    model.eval()
    correct = 0
    exercise = 0
    batch_count = 0
    batch_avg_loss = 0
    pred = []
    label = []
    with torch.no_grad():
        while not dataloader.is_end():
            batch_count += 1
            input_stu_ids, input_exer_ids, input_knowledge_embs, labels = dataloader.next_batch()
            input_stu_ids, input_exer_ids, input_knowledge_embs, labels = input_stu_ids.to(device), input_exer_ids.to(device), input_knowlec
            output = model(input_stu_ids, input_exer_ids, input_knowledge_embs)
            output = output.view(-1)
            for i in range(len(labels)):
                if (labels[i] == 1 and output[i] > 0.5) or (labels[i] == 0 and output[i] < 0.5):
                    correct += 1
            exercise += len(labels)
            pred += output.to(torch.device('cpu')).tolist()
            label += labels.to(torch.device('cpu')).tolist()
        pred = np.array(pred)
        label = np.array(label)
        acc = correct / exercise
        rmse = np.sqrt(np.mean((label - pred) ** 2))
        auc = roc_auc_score(label, pred)
        print('epoch= %d, accuracy= %f, rmse= %f, auc= %f' % (epoch+1, acc, rmse, auc))
```

```python
# index = str(len(os.listdir('../log/')))
with open('../log/NeuralCDM_val.txt', 'a', encoding='utf8') as f:
    f.write('epoch= %d, accuracy= %f, rmse= %f, auc= %f\n' % (epoch + 1, acc, rmse, auc))
```

```python
def test(model):
    dataloader = ValTestDataLoader('test')
    print('Testing...')
    dataloader.reset()
    model.eval()
    correct = 0
    exercise = 0
    pred = []
    label = []
    with torch.no_grad():
        while not dataloader.is_end():
            input_stu_ids, input_exer_ids, input_knowledge_embs, labels = dataloader.next_batch()
            input_stu_ids, input_exer_ids, input_knowledge_embs, labels = input_stu_ids.to(device), input_exer_ids.to(device), input_knowlec
            output = model(input_stu_ids, input_exer_ids, input_knowledge_embs)
            output = output.view(-1)
            # print(labels, output)
            for i in range(len(labels)):
                # print(labels[i], output[i])
                if (labels[i] == 1 and output[i] > 0.5) or (labels[i] == 0 and output[i] < 0.5):
                    correct += 1
            exercise += len(labels)
            pred += output.tolist()
            label += labels.tolist()
        pred = np.array(pred)
        label = np.array(label)

        # print(correct, exercise)
        acc = correct / exercise
        rmse = np.sqrt(np.mean((pred - label) ** 2))
        auc = roc_auc_score(label, pred)
        print('accuracy= %f, rmse= %f, auc= %f' % (acc, rmse, auc))
        with open('../log/NeuralCDM_test.txt', 'a', encoding='utf8') as f:
            f.write('accuracy= %f, rmse= %f, auc= %f\n' % (acc, rmse, auc))
```

```python
        with open('../log/NeuralCDM_test.txt', 'a', encoding='utf8') as f:
            f.write('accuracy= %f, rmse= %f, auc= %f\n' % (acc, rmse, auc))

def saveModel(model, filename):
    f = open(filename, 'wb')
    torch.save(model, filename)
    f.close()

def loadModel(filename):
    f = open(filename, 'rb')
    model = torch.load(f)
    f.close()
    return model
```

```
accuracy= 0.730452, rmse= 0.433719, auc= 0.752963
accuracy= 0.730022, rmse= 0.433221, auc= 0.752583
```

模型结果

DKT：

```
Testing:    : 100%|██████████| 21/21 [00:03<00:00,  5.96it/s]
Testing:    : 100%|██████████| 17/17 [00:02<00:00,  5.98it/s]
Testing:    : 100%|██████████| 21/21 [00:03<00:00,  5.52it/s]
Testing:    : 100%|██████████| 21/21 [00:03<00:00,  5.43it/s]
auc:0.816997491455413 f1: 0.841505026177322 recall: 0.9194718618450354 precision: 0.7757270648317993


epoch: 10
Training:    : 100%|██████████| 99/99 [00:58<00:00,  1.69it/s]
Training:    : 100%|██████████| 78/78 [00:43<00:00,  1.79it/s]
Training:    : 100%|██████████| 79/79 [00:42<00:00,  1.86it/s]
Training:    : 100%|██████████| 83/83 [00:46<00:00,  1.80it/s]
Training:    : 100%|██████████| 79/79 [00:43<00:00,  1.81it/s]
Training:    : 100%|██████████| 78/78 [00:42<00:00,  1.84it/s]
Testing:    : 100%|██████████| 44/44 [00:07<00:00,  6.02it/s]
Testing:    : 100%|██████████| 22/22 [00:03<00:00,  6.15it/s]
Testing:    : 100%|██████████| 21/21 [00:03<00:00,  6.14it/s]
Testing:    : 100%|██████████| 17/17 [00:02<00:00,  5.99it/s]
Testing:    : 100%|██████████| 21/21 [00:03<00:00,  5.80it/s]
Testing:    : 100%|██████████| 21/21 [00:03<00:00,  5.83it/s]
auc:0.818016665288025 f1: 0.8405706292718873 recall: 0.9122978415681317 precision: 0.7793000470453476
```

CKT：

```
step 1: loss 0.247341 acc:0.945312
step 2: loss 0.396896 acc:0.921875
step 3: loss 0.238149 acc:0.953125
step 4: loss 0.330783 acc:0.9375
step 5: loss 0.352517 acc:0.929688
step 6: loss 0.723586 acc:0.882812
step 7: loss 0.446891 acc:0.890625
step 8: loss 0.430105 acc:0.914062
step 9: loss 0.240339 acc:0.9375
step 10: loss 0.333226 acc:0.921875
step 11: loss 0.22356 acc:0.953125
step 12: loss 0.205548 acc:0.953125
step 13: loss 0.246604 acc:0.9375
```

```
step 533: loss 0.120846 acc:0.960938
step 534: loss 0.145238 acc:0.960938
step 535: loss 0.102822 acc:0.96875
step 536: loss 0.112673 acc:0.96875
step 537: loss 0.110254 acc:0.976562
step 538: loss 0.0701208 acc:0.984375
step 539: loss 0.120761 acc:0.960938
step 540: loss 0.173894 acc:0.945312
step 541: loss 0.109045 acc:0.953125
step 542: loss 0.15224 acc:0.960938
step 543: loss 0.145357 acc:0.945312
step 544: loss 0.154636 acc:0.960938
step 545: loss 0.128378 acc:0.953125
```

```
Epoch 3 has finished!
running time analysis: epoch3, avg_time281.85999400000003
max: acc0.950195
```

NeuralCDM：

```
epoch: 6
Training...: 100%|██████████| 198/198 [00:55<00:00,  3.54it/s]
Testing...: 100%|██████████| 87/87 [00:10<00:00,  8.61it/s]
auc:0.9940988722111318
f1:0.976209769895268
recall:0.9802112438208112
precision:0.9722408332318188
acc:0.9684522119946474

epoch: 7
Training...: 100%|██████████| 198/198 [00:58<00:00,  3.37it/s]
Testing...: 100%|██████████| 87/87 [00:10<00:00,  8.24it/s]
auc:0.9945762162657876
f1:0.977104021894518
recall:0.9811016610887654
precision:0.9731388284021867
acc:0.9696382952840518
```

展示

后端 flask 代码

```python
@app.route('/predict', methods=['POST', 'GET'])
def predict():
    # result = request.form.get('result')
    # print(request.method)
    if request.method == 'GET':
        ques_index = int(request.args.get('quesIndex'))
        result = request.args.get('result')
        onehot = torch.zeros([params.NUM_OF_QUESTIONS * 2])

        if result == True:
            onehot[ques_index] = 1
        else:
            onehot[ques_index + params.NUM_OF_QUESTIONS] = 1

        info['steps_ans'] = torch.cat([info['steps_ans'].squeeze(0), onehot.reshape(-1, params.NUM_OF_QUESTIONS * 2)]).unsqueeze(0)

        print(info['steps_ans'], info['steps_ans'].shape)
        pred = model(info['steps_ans'])
        print(pred, pred.shape)
        print(pred[0][-1])
    # return render_template('index.html', index=ques_index, result=result)
    return jsonData(pred[0])
```

References:

[1]. Piech, C., et al., Deep Knowledge Tracing. 2015.

[2]. Wang, F., et al., Neural Cognitive Diagnosis for Intelligent Education Systems, in AAAI2020.

[3]. https://github.com/shshen-closer/MOOC_cube-Task2-TOP3