Prediction模块分析

**1. 作用：**

预测障碍物的运动轨迹，每条轨迹都有一个概率值。

**2. 输入：**

（1） 车辆位置信息：/apollo/localization/pose（[pb\_msgs/LocalizationEstimate]）。

（2） 障碍物信息：/apollo/perception/obstacles（[pb\_msgs/ PerceptionObstacles]）。

**3. 输出：**

障碍物的运动轨迹。/apollo/prediction [pb\_msgs/PredictionObstacle]。

**4. 节点图：**



**5. 子模块：**

（1） 容器： 存储订阅话题中的数据。

（2） 评估器：对于任意一个障碍物，评估器预测路径和速度。一个评估器通过使用\_prediction/data/\_下的评估模型对每条路径给出一个概率值，实现评估。

（3） 预测器：预测器为障碍物预测通过，当前通道包含以下几种：

1） 车道序列：障碍物只能依据车道移动。

2） 自由移动：障碍物自由移动。

3） 区域引动：障碍物只能在一定的区域移动。

**6. 源码架构：**

（1） main.c:

启动/prediction节点。

（2） prediction.cc和prediction.h:

1） Name()函数：返回节点名字prediction。

2） Init()函数：

a） 使用配置文件prediction\_conf.pb.txt设置prediction\_conf\_，主要设置 preditor的产生的通道类型。

b) 使用配置文件adapter.conf设置adapter\_conf\_，设置节点话题类型。

c） 初始化AdapterManager，定义nodehandle和话题。

d） 初始化ContainerManager，每一个接受话题创建一个Container，用 于接受话题数据。

e） 初始化EvaluatorManager：注册一个Evaluator：MLP\_EVALUATOR； 根据prediction\_conf.pb.txt配置MLP\_EVALUATOR，似乎关注路上的 车辆，设置vehicle\_on\_lane\_evaluator\_。

f） 初始化PredictorManager：注册四个Predictor，并设置了 vehicle\_on\_lane\_predictor\_，vehicle\_off\_lane\_predictor\_和 pedestrian\_predictor\_。

g） 检测localization和perception节点是否准备好。

h） 设置localization和perception数据的回调函数OnLocalization和 OnPerception。

3） OnLocalization函数：

a）获取障碍物容器obstacles\_container。

b） 获取位置pose\_container。

c） 将新到的位置消息存入pose\_container。

d） 将位置转换为障碍物信息（不明确）

e） 更新障碍物信息。

4） OnPerception函数：

a） 获取障碍物容器obstacles\_container。

b) 将新到的障碍物信息存入obstacles\_container。

c） 运行Evaluator。

d） 运行Predictor。

e） 跟新待发布数据prediction\_obstacles header结构，发布消息。

**7. 评估器Evaluator:**

（1） 创建一个新的NewEvaluator：

1） data/mlp\_vehicle\_model.bin：利用深度学习实现的评估器核心部分。

2） feature.proto或lane\_graph.proto文件的配置输出，不清楚作用。

3） 在evaluator/vehicle/目录下，以Evaluator为基类实现一个新的评估器类 NewEvaluator。并参考mlp\_evaluator实现类。

4） prediction\_conf.pb.txt文件中指定所实现的新评估器类。

5） evaluator\_manager.h中修改默认使用的评估器类。

（2） evaluator\_manager.cc中的Run()函数：

1） 获取障碍物容器container。

2） 遍历所有障碍物，利用障碍物id和障碍物容器获取障碍物信息obstacle。

3） 针对lane上的障碍物，调用Evaluate进行评估。

（3） mlp\_evaluator.cc中的Evaluate()函数：

1） 以单个障碍物为参数。计算单个障碍物的feature，并计算其概率。

2） 每个obstacle\_ptr中包含多个lane\_graph\_ptr。

3） 利用obstacle\_ptr和lane\_graph\_ptr可计算出feature\_values

4） 由feature\_values计算概率值probability。

5） 将概率值probability设置到lane\_sequence\_ptr。

6） 进一步的分析需要理解feature和lane\_sequence概念，需要进入算法，暂 时评估器到这里。

**8. 预测器Predictor：**

（1） 功能：预测障碍物的未来轨迹。

（2） 创建一个新的预测器NewPredictor：

1） 在predictor/下新建目录vehicle。

2） 在vehicle目录下创建new\_predictor.h和new\_predictor.cc，主要是继承 与Predictor类定义并实现子类NewPredictor。具体实现可以参考vehicle 同级目录free\_move等。

3） 更新配置文件prediction\_conf.pb.txt，添加新预测器类型。

4） 更新manager，在文件predictor\_manager.h中修改默认预测器类型。

（3） prediction\_manager.cc中的run()函数：

1） 获取障碍物容器container。

2） 设置预测障碍物的时间戳。

3） 根据预测障碍物中的id和容器获取障碍物信息obstacle。

4） 根据预测障碍物中的类型设置预测器的类型predictor。

5） 传入障碍物信息，执行预测器predictor->Predict(obstacle)。

6） 将障碍物的所有轨道配置到预测到的障碍物中。并更新时间戳。

（4） 预测器函数Predict()函数：

1） 根据障碍物信息获取feature。

2） 由feature获取num\_lane\_sequence

3） 遍历num\_lane\_sequence，通过feature获取sequence。

4） 由sequence获取curr\_lane\_id

5） 由curr\_lane\_id通过DrawLaneSequenceTrajectoryPoints()函数获取 TrajectoryPoint。

6） 由TrajectoryPoint通过GenerateTrajectory()函数获取trajectory

7） 设置该trajectory的probability，存入trajectories\_向量

8） 和Evaluator类似，进一步分析需了解具体算法，暂时到这里。