PredictionIO를 활용한 머신러닝 서버 만들기

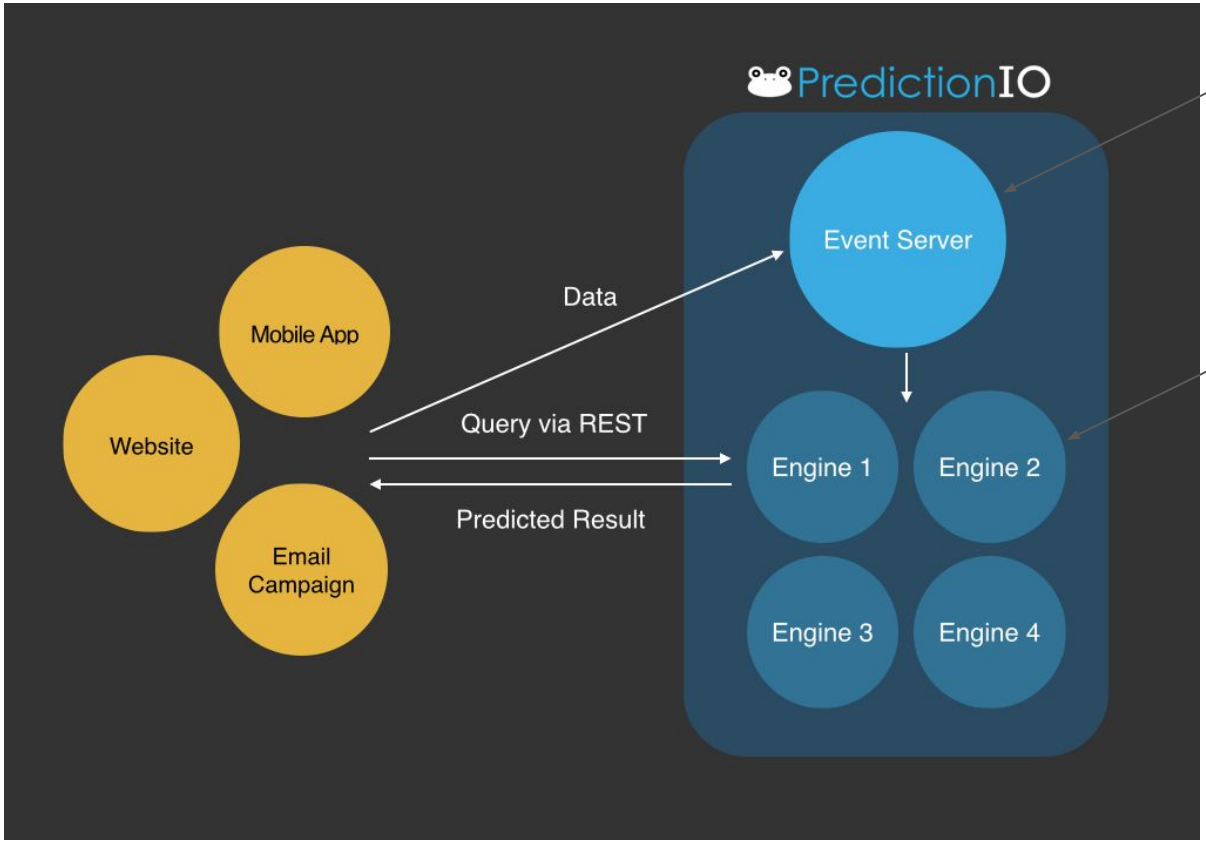
1. PredictionIO란?

- Apache incubating
- 오픈소스
- 머신러닝 서버
- http://predictionio.incubator.apache.org/

2. PredictionIO 특징

- 빌드, 배포 등 웹서비스 구축 절차가 편함
- 실시간 응답
- 머신러닝, 데이터 처리 라이브러리 지원(Spark MLlib, OpenNLP)
- 자신의 알고리즘도 추가 가능
- 데이터 관리 단순함

3. Overview



데이터

알고리즘

4. 설치하기

- 최소사양(중요함)
  - Apache Hadoop 2.4.0 (optional, required only if YARN and HDFS are needed)
  - Apache Spark 1.4.0 for Hadoop 2.4
  - Java SE Development Kit 7
  - 데이터 저장소
    - PostgreSQL 9.1 Or MySQL 5.1 Or(Apache HBase 0.98.6, Elasticsearch 1.4.0)
  - Spark는 standalone cluster mode 권장

4. 설치하기

- 프로젝트 빌드

소스 다운로드
$ wget
http://mirror.navercorp.com/apache/incubator/predictionio/0.10.0-incubating/apache-predictionio-0.10.0-incubating.tar.gz

빌드하기

$ tar zxvf apache-predictionio-0.10.0-incubating.tar.gz

$ cd apache-predictionio-0.10.0-incubating

$ ./make-distribution.sh

빌드 성공하면 PredictionIO-0.10.0-incubating.tar.gz 생성됨

$ tar zxvf PredictionIO-0.10.0-incubating.tar.gz

4. 설치하기

- Dependencies (Apark Spark, HBase, Elasticsearch)

저장 폴더 생성

$ mkdir PredictionIO-0.10.0-incubating/vendors

Apache Spark 설치. 기존에 있으면 PredictionIO-0.10.0-incubating/conf/pio-env.sh 에서 SPARK_HOME만 변경하면 됨

$ wget http://d3kbcqa49mib13.cloudfront.net/spark-1.5.1-bin-hadoop2.6.tgz

$ tar zxvfC spark-1.5.1-bin-hadoop2.6.tgz PredictionIO-0.10.0-incubating/vendors

4. 설치하기

- Dependencies (Apark Spark, HBase, Elasticsearch)

Elasticsearch 설치

$ wget
https://download.elasticsearch.org/elasticsearch/elasticsearch/elasticsearch-1.4.4.tar.gz
$ tar zxvfC elasticsearch-1.4.4.tar.gz PredictionIO-0.10.0-incubating/vendors

PredictionIO-0.10.0-incubating/conf/pio-env.sh에서 저장소 설정 변경
$ vi PredictionIO-0.10.0-incubating/conf/pio-env.sh

PIO_STORAGE_SOURCES_ELASTICSEARCH_TYPE=elasticsearch
PIO_STORAGE_SOURCES_ELASTICSEARCH_HOSTS=localhost
PIO_STORAGE_SOURCES_ELASTICSEARCH_PORTS=9300

4. 설치하기

- Dependencies (Apark Spark, HBase, Elasticsearch)

HBase 설치

$ wget http://archive.apache.org/dist/hbase/hbase-1.0.0/hbase-1.0.0-bin.tar.gz
$ tar zxvfC hbase-1.0.0-bin.tar.gz PredictionIO-0.10.0-incubating/vendors

5. 실행하기

- Start PredictionIO and Dependent Services

```
$ PredictionIO-0.10.0-incubating/bin/pio-start-all
Starting Elasticsearch...
Starting HBase...
….

확인하기
$ PredictionIO-0.10.0-incubating/bin/pio status
```

6. 머신러닝 서버 만들기

- Engine Template 고르기
  - template gallery에서 template 구경
  - 대부분의 머신러닝 task들은 template으로 구현되어 있음
  - http://predictionio.incubator.apache.org/gallery/template-gallery

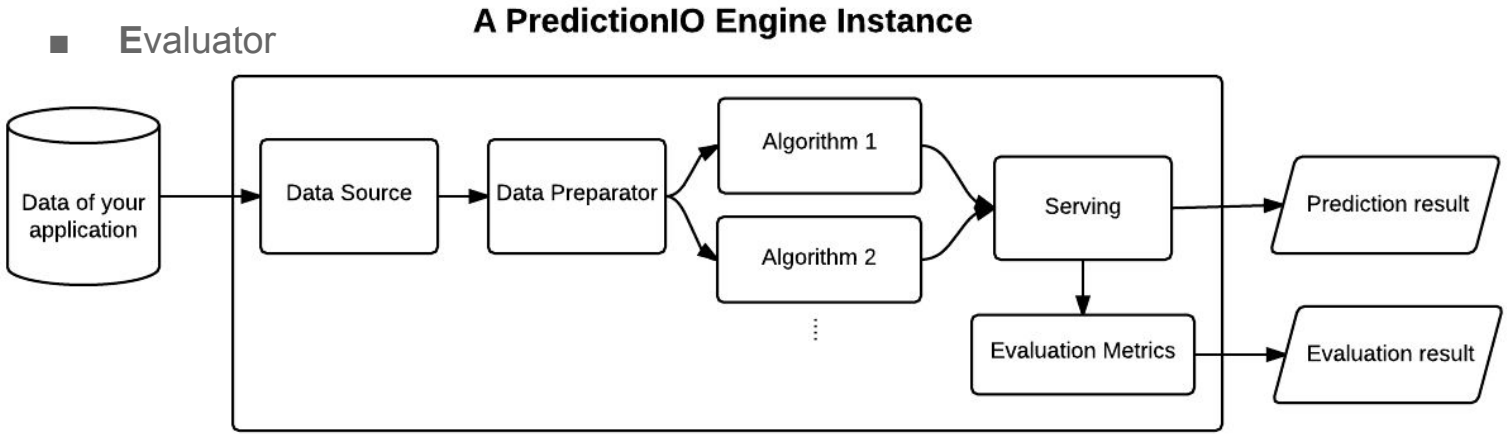
```
collaborative filtering Engine 다운로드
$ pio template get apache/incubator-predictionio-template-recommender
MyRecommendation
$ cd MyRecommendation
```

6. 머신러닝 서버 만들기

- Recommendation Engine 설명
  - 데이터
    - 유저 평점
    - 유저 구매 item
    - https://raw.githubusercontent.com/apache/spark/master/data/mllib/sample_movielens_data.txt (user :: item :: rate, 구매한 사람은 평점 4점으로)
  - Apache Spark Collaborative Filtering 알고리즘 활용
    - http://spark.apache.org/docs/1.3.0/mllib-collaborative-filtering.html
  - 질의 입력
    - 유저 ID
    - item 번호
  - 추천 item 리스트, 스코어

6. 머신러닝 서버 만들기

- Recommendation Engine
  - DASE Components Explained (Recommendation)
    - **D**ata - includes Data Source and Data Preparator
    - **A**lgorithm(s)
    - **S**erving
    - **E**valuator



6. 머신러닝 서버 만들기

- Recommendation Engine
  - MyRecommendation/src/main/scala/**Engine.scala** 에 쿼리 포맷 클래스 선언

```
case class Query(
  user: String,
  num: Int
) extends Serializable
```

```
curl -H "Content-Type: application/json" \
-d '{ "user": "1", "num": 4 }' http://localhost:8000/queries.json
```

6. 머신러닝 서버 만들기

- Recommendation Engine
  - MyRecommendation/src/main/scala/**Engine.scala** 에 예측 결과 클래스 선언

```
case class PredictedResult(
  itemScores: Array[ItemScore]
) extends Serializable

case class ItemScore(
  item: String,
  score: Double
) extends Serializable
```

```
{
  "itemScores":[
    {"item":"22","score":4.072304374729956},
    {"item":"62","score":4.058482414005789},
    {"item":"75","score":4.046063009943821},
    {"item":"68","score":3.8153661512945325}
  ]
}
```

6. 머신러닝 서버 만들기

- Recommendation Engine
  - MyRecommendation/src/main/scala/**Engine.scala** 에 **DASE components** 정의

```
object RecommendationEngine extends IEngineFactory {
  def apply() = {
    new Engine(
      classOf[DataSource],
      classOf[Preparator],
      Map("als" -> classOf[ALSAlgorithm]),
      classOf[Serving])
  }
  ...
}
```

6. 머신러닝 서버 만들기

- Recommendation Engine
  - MyRecommendation/src/main/scala/**DataSource.scala**에서 **EventServer**에 있
    가져와서 **RDD**로 변환

```
def getRatings(sc: SparkContext): RDD[Rating] = {
  val eventsRDD: RDD[Event] = PEventStore.find(
    appName = dsp.appName,
    entityType = Some("user"),
    eventNames = Some(List("rate", "buy")), // read "rate" and "buy" event
    // targetEntityType is optional field of an event.
    targetEntityType = Some(Some("item")))(sc)
...
```

```
    val ratingsRDD: RDD[Rating] = eventsRDD.map { event =>
      val rating = try {
        val ratingValue: Double = event.event match {
          case "rate" => event.properties.get[Double]("rating")
          case "buy" => 4.0 // map buy event to rating value of 4
          case _ => throw new Exception(s"Unexpected event ${event} is read.")
        }
        Rating(event.entityId,event.targetEntityId.get,ratingValue)
      }
...
      }
      rating
    }.cache()
    ratingsRDD
  }
```

6. 머신러닝 서버 만들기

- Recommendation Engine
  - MyRecommendation/src/main/scala/**Preparator.scala**에서 데이터 전처리, 특징 추출

```
class Preparator
  extends PPreparator[TrainingData, PreparedData] {

  def prepare(sc: SparkContext, trainingData: TrainingData): PreparedData = {
    new PreparedData(ratings = trainingData.ratings)
  }
}

class PreparedData(
  val ratings: RDD[Rating]
) extends Serializable
```

6. 머신러닝 서버 만들기

- Recommendation Engine
  - MyRecommendation/src/main/scala/**ALSAlgorithm.scala**에서 훈련. **pio train** 명령 시 호출됨

```
def train(sc: SparkContext, data: PreparedData): ALSModel = {
  ...
  // Convert user and item String IDs to Int index for MLlib
  val userStringIntMap = BiMap.stringInt(data.ratings.map(_.user))
  val itemStringIntMap = BiMap.stringInt(data.ratings.map(_.item))
  val mllibRatings = data.ratings.map( r =>
    // MLlibRating requires integer index for user and item
    MLlibRating(userStringIntMap(r.user), itemStringIntMap(r.item), r.rating)
  )
```

6. 머신러닝 서버 만들기

- Recommendation Engine
  - MyRecommendation/src/main/scala/**ALSAlgorithm.scala**에서 훈련. **pio train** 명령 시 호출됨

```
    val seed = ap.seed.getOrElse(System.nanoTime)
    val m = ALS.train(
      ratings = mllibRatings,
      rank = ap.rank,
      iterations = ap.numIterations,
      lambda = ap.lambda,
      blocks = -1,
      seed = seed)
...
```

6. 머신러닝 서버 만들기

- Recommendation Engine
  - MyRecommendation/src/main/scala/**ALSAlgorithm.scala**에서 **predict,** MatrixFactorizationModel

```
def predict(model: ALSModel, query: Query): PredictedResult = {
  // Convert String ID to Int index for Mllib
  model.userStringIntMap.get(query.user).map { userInt =>
    val itemIntStringMap = model.itemStringIntMap.inverse
    // recommendProducts() returns Array[MLlibRating], which uses item Int
    // index. Convert it to String ID for returning PredictedResult
    val itemScores = model.recommendProducts(userInt, query.num)
      .map (r => ItemScore(itemIntStringMap(r.product), r.rating))
    new PredictedResult(itemScores)
  }.getOrElse{  logger.info(s"No prediction for unknown user ${query.user}.")
    new PredictedResult(Array.empty)}
}
```

6. 머신러닝 서버 만들기

- Recommendation Engine
  - MyRecommendation/src/main/scala/**Serving.scala**에서 결과값 **PredictedResult**에 맞게 리턴

```
class Serving
  extends LServing[Query, PredictedResult] {

  override
  def serve(query: Query,
    predictedResults: Seq[PredictedResult]): PredictedResult = {
    predictedResults.head
  }
}
```

6. 머신러닝 서버 만들기

- Engine.scala

```
object RecommendationEngine extends IEngineFactory {
  def apply() = {
    new Engine(
      classOf[DataSource],  //
      classOf[Preparator],
      Map("als" -> classOf[ALSAlgorithm]),    //적용할 알고리즘 명시
      classOf[Serving])
  }
```

6. 머신러닝 서버 만들기

- App ID 와 Access Key 생성

MyApp1으로 App 생성
$ pio app new MyApp1

App list 확인
$ pio app list

6. 머신러닝 서버 만들기

- Data Import

```
$ cd MyRecommendation
$ curl
https://raw.githubusercontent.com/apache/spark/master/data/mllib/sample_
movielens_data.txt --create-dirs -o data/sample_movielens_data.txt
$ python data/import_eventserver.py --access_key $ACCESS_KEY

(app list에서 Access key 확인)
```

6. 머신러닝 서버 만들기

- Engine.json 수정

```
{
...
    "params" : {
      "appName": "INVALID_APP_NAME"    ← 내 App 이름
...
  "algorithms": [
    {
      "name": "als",
      "params": {   ← 파라미터 설정가능
        "rank": 10,
        "numIterations": 20,
        "lambda": 0.01,
        "seed": 3
...
```

6. 머신러닝 서버 만들기

- Deploy

빌드하기
$ pio build --verbose

훈련하기
$ pio train

배포하기
$ pio deploy

6. 머신러닝 서버 만들기

- Query

```
$ curl -H "Content-Type: application/json" -d '{ "user": "1", "num": 4 }'
http://localhost:8000/queries.json
```


```
{
  "itemScores":[
    {"item":"22","score":4.072304374729956},
    {"item":"62","score":4.058482414005789},
    {"item":"75","score":4.046063009943821},
    {"item":"68","score":3.8153661512945325}
  ]
}
```