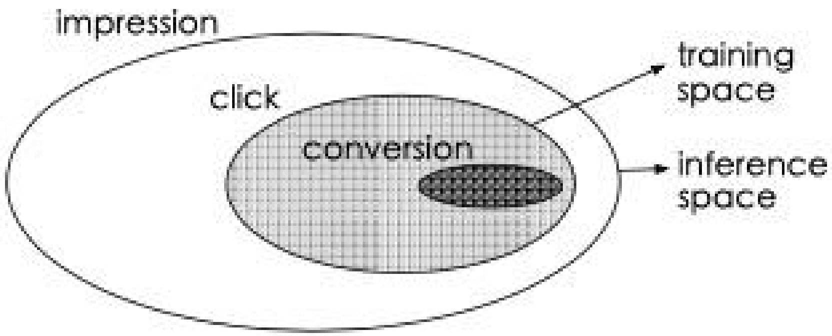
Entire Space Multi-Task Model: An Effective Approach for Estimating Post-Click Conversion Rate

何荣炜

2020-07-31



- Problem：
  - conventional CVR models are trained with samples of clicked impressions while utilized to make inference on the entire space with samples of all impressions. This causes a sample selection bias problem
  - Besides, there exists an extreme data sparsity problem, making the model fitting rather difficult.

- In other words, user actions follow a sequential pattern of impression → click → conversion. In this way, CVR modeling refers to the task of estimating the post-click conversion rate, i.e., pCVR = p(conversion | click,impression).

- sample selection bias (SSB) problem
- data sparsity (DS) problem.

- There are several studies trying to tackle these challenges.
  - hierarchical estimators on different features are built and combined with a logistic regression model to solve DS problem （However, it relies on a priori knowledge to construct hierarchical structures, which is difficult to be applied in recommender systems with tens of millions of users and items ）
  - Oversampling method copies rare class examples which helps lighten sparsity of data but is sensitive to sampling rates.
  - All Missing As Negative (AMAN) applies random sampling strategy to select un-clicked impressions as negative examples （but results in a consistently underestimated prediction ）

- In this paper， propose a novel approach named Entire Space Multi-task Model (ESMM) , which is able to eliminate the SSB and DS problems simultaneously.

- Both pCTCVR and pCTR are estimated over the entire space with samples of all impressions

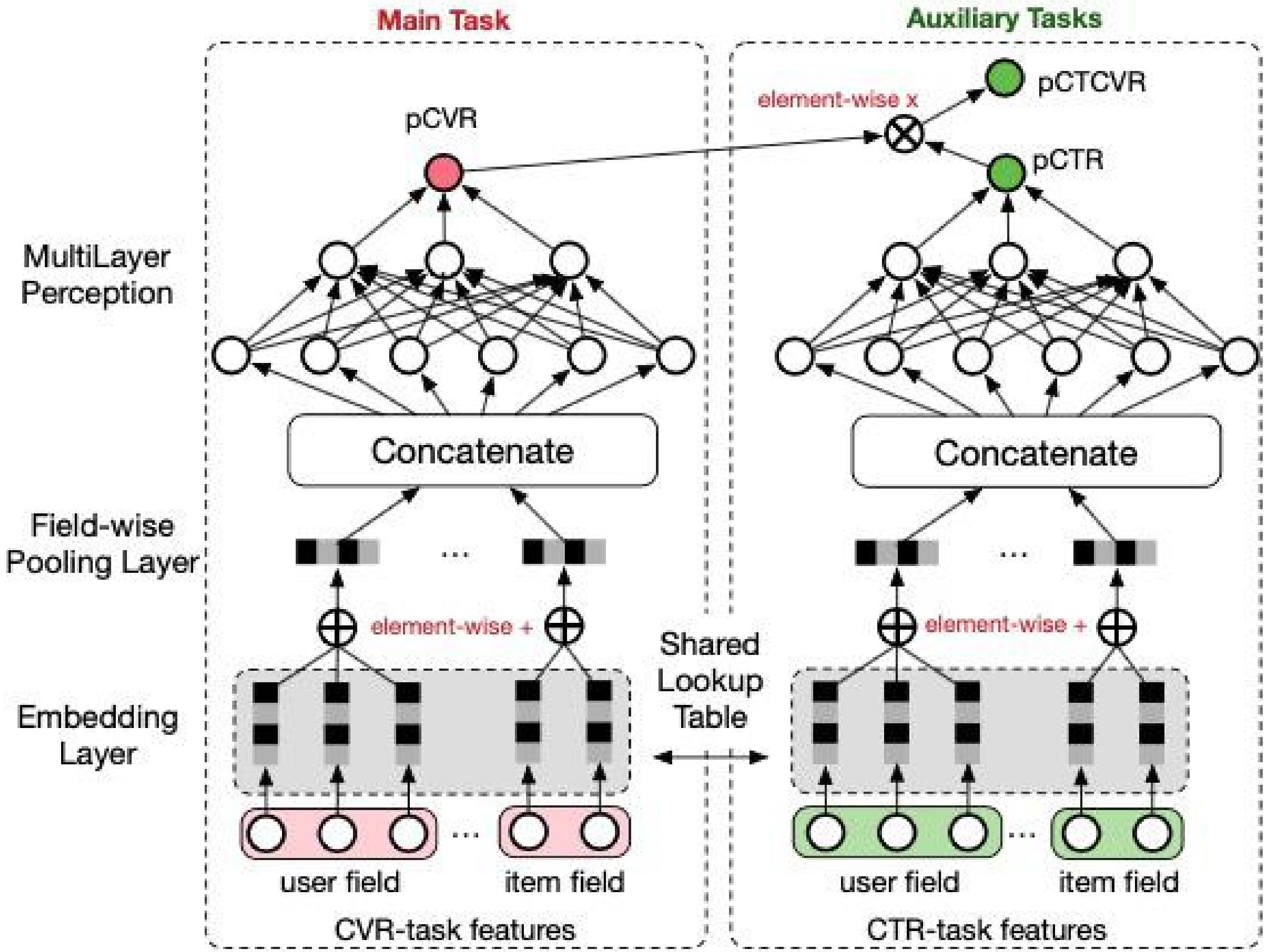
- Besides, parameters of feature representation of CVR network is shared with CTR network. The latter one is trained with much richer samples. This kind of parameter transfer learning helps to alleviate the DS trouble remarkably.

Notation

- We assume the observed dataset to be $\mathcal{S} = \{(\boldsymbol{x}_i, y_i \rightarrow z_i)\}|_{i=1}^{N}$
- where X is feature space, Y and Z are label spaces, and N is the total number of impressions.

$$\underbrace{p(y=1, z=1|\boldsymbol{x})}_{pCTCVR} = \underbrace{p(y=1|\boldsymbol{x})}_{pCTR} \times \underbrace{p(z=1|y=1, \boldsymbol{x})}_{pCVR}.$$

Architecture overview of ESMM



i) help to model CVR over entire input space

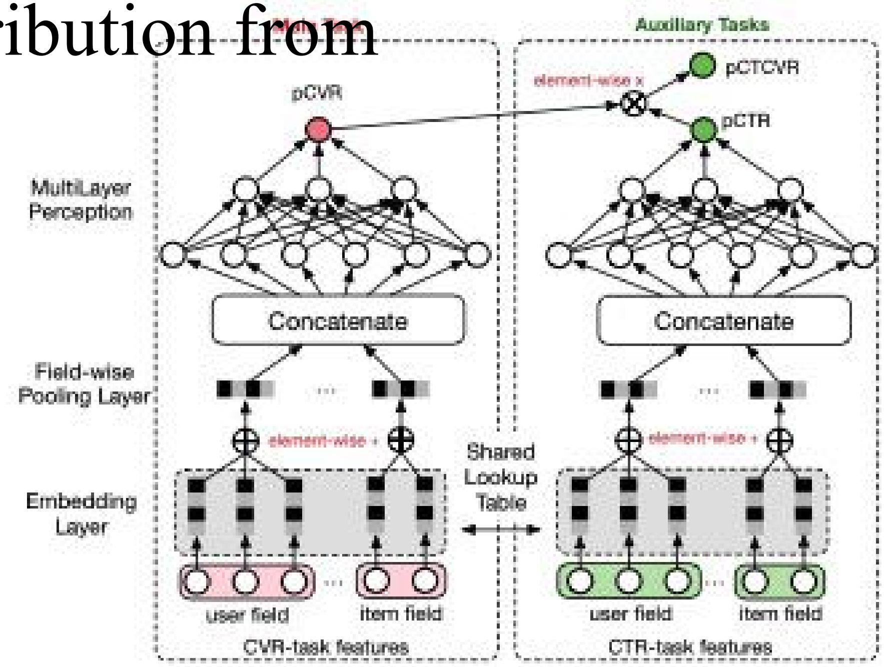
ii) provide feature representation transfer learning.

- The loss function of ESMM is defined as Eq. It consists of two loss terms from CTR and CTCVR tasks which are calculated over samples of all impressions （ l(·) is cross-entropy loss function ）

$$
\begin{aligned}
L(\theta_{cvr}, \theta_{ctr}) = & \sum_{i=1}^{N} l\left(y_i, f(\boldsymbol{x}_i; \theta_{ctr})\right) \\
& + \sum_{i=1}^{N} l\left(y_i \& z_i, f(\boldsymbol{x}_i; \theta_{ctr}) \times f(\boldsymbol{x}_i; \theta_{cvr})\right)
\end{aligned}
$$

EXPERIMENTS

- BASE is the baseline model (The left part of Fig.2 illustrates this kind of architecture)
- AMAN applies negative sampling strategy
- OVERSAMPLING copies positive examples to reduce difficulty of training with sparse data
- UNBIAS follows to fit the truly underlying distribution from observations via rejection sampling
- DIVISION
- ESMM-NS



AMAN performs a little worse on CVR task,which may be due to the sensitive of random sampling.

OVERSAMPLING and UNBIAS show improvement over BASE model on both CVR and CTCVR tasks

ESMM achieves absolute AUC gain of 2.56% on CVR task. On CTCVR task with full samples, it brings 3.25% AUC gain

Table 1: Statistics of experimental datasets.

dataset	#user	#item	#impression	#click	#conversion
Public Dataset	0.4M	4.3M	84M	3.4M	18k
Product Dataset	48M	23.5M	8950M	324M	1774k

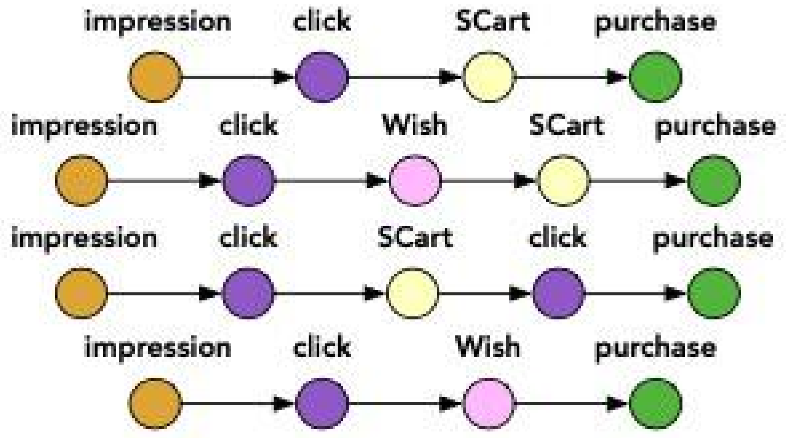
Table 2: Comparison of different models on Public Dataset.

Model	AUC(mean ± std) on CVR task	AUC(mean ± std) on CTCVR task
BASE	66.00 ± 0.37	62.07 ± 0.45
AMAN	65.21 ± 0.59	63.53 ± 0.57
OVERSAMPLING	67.18 ± 0.32	63.05 ± 0.48
UNBIAS	66.65 ± 0.28	63.56 ± 0.70
DIVISION	67.56 ± 0.48	63.62 ± 0.09
ESMM-NS	**68.25 ± 0.44**	**64.44 ± 0.62**
ESMM	**68.56 ± 0.37**	**65.32 ± 0.49**

FUTURE WORK

- In the future, we intend to design global optimization models in applications with multistage actions like request → impression → click → conversion.

- ESM2 （2020）



Source code

```python
# user feature
bids = fc.categorical_column_with_hash_bucket("behaviorBids", 10240, dtype=tf.int64)
c1ids = fc.categorical_column_with_hash_bucket("behaviorC1ids", 100, dtype=tf.int64)
cids = fc.categorical_column_with_hash_bucket("behaviorCids", 10240, dtype=tf.int64)
sids = fc.categorical_column_with_hash_bucket("behaviorSids", 10240, dtype=tf.int64)
pids = fc.categorical_column_with_hash_bucket("behaviorPids", 1000000, dtype=tf.int64)
bids_weighted = fc.weighted_categorical_column(bids, "bidWeights")
c1ids_weighted = fc.weighted_categorical_column(c1ids, "c1idWeights")
cids_weighted = fc.weighted_categorical_column(cids, "cidWeights")
sids_weighted = fc.weighted_categorical_column(sids, "sidWeights")
pids_weighted = fc.weighted_categorical_column(pids, "pidWeights")

# item feature
pid = fc.categorical_column_with_hash_bucket("productId", 1000000, dtype=tf.int64)
sid = fc.categorical_column_with_hash_bucket("sellerId", 10240, dtype=tf.int64)
bid = fc.categorical_column_with_hash_bucket("brandId", 10240, dtype=tf.int64)
c1id = fc.categorical_column_with_hash_bucket("cate1Id", 100, dtype=tf.int64)
cid = fc.categorical_column_with_hash_bucket("cateId", 10240, dtype=tf.int64)
```

```python
pid_embed = fc.shared_embedding_columns([pids_weighted, pid], 64, combiner='sum', shared_embedding_collection_name="pid")
bid_embed = fc.shared_embedding_columns([bids_weighted, bid], 32, combiner='sum', shared_embedding_collection_name="bid")
cid_embed = fc.shared_embedding_columns([cids_weighted, cid], 32, combiner='sum', shared_embedding_collection_name="cid")
c1id_embed = fc.shared_embedding_columns([c1ids_weighted, c1id], 10, combiner='sum', shared_embedding_collection_name="c1id")
sid_embed = fc.shared_embedding_columns([sids_weighted, sid], 32, combiner='sum', shared_embedding_collection_name="sid")
global my_feature_columns
my_feature_columns = [matchScore, matchType, postition, triggerNum, triggerRank, sceneType, hour, phoneBrand, phoneResolution,
           phoneOs, tab, popScore, sellerPrefer, brandPrefer, cate2Prefer, catePrefer]
my_feature_columns += pid_embed
my_feature_columns += sid_embed
my_feature_columns += bid_embed
my_feature_columns += cid_embed
my_feature_columns += c1id_embed
```

```python
def build_mode(features, mode, params):
  net = fc.input_layer(features, params['feature_columns'])
  # Build the hidden layers, sized according to the 'hidden_units' param.
  for units in params['hidden_units']:
    net = tf.layers.dense(net, units=units, activation=tf.nn.relu)
    if 'dropout_rate' in params and params['dropout_rate'] > 0.0:
      net = tf.layers.dropout(net, params['dropout_rate'], training=(mode == tf.estimator.ModeKeys.TRAIN))
  # Compute logits
  logits = tf.layers.dense(net, 1, activation=None)
  return logits

def my_model(features, labels, mode, params):#特标模参
  with tf.variable_scope('ctr_model'):
    ctr_logits = build_mode(features, mode, params)
  with tf.variable_scope('cvr_model'):
    cvr_logits = build_mode(features, mode, params)

  ctr_predictions = tf.sigmoid(ctr_logits, name="CTR")
  cvr_predictions = tf.sigmoid(cvr_logits, name="CVR")
  prop = tf.multiply(ctr_predictions, cvr_predictions, name="CTCVR")
  if mode == tf.estimator.ModeKeys.PREDICT:
    predictions = {
      'probabilities': prop,
      'ctr_probabilities': ctr_predictions,
      'cvr_probabilities': cvr_predictions
    }
    export_outputs = {
      'prediction': tf.estimator.export.PredictOutput(predictions)
    }
    return tf.estimator.EstimatorSpec(mode, predictions=predictions, export_outputs=export_outputs)
```

```python
cvr_loss = tf.reduce_sum(tf.keras.backend.binary_crossentropy(y, prop), name="cvr_loss")
ctr_loss = tf.reduce_sum(tf.nn.sigmoid_cross_entropy_with_logits(labels=labels['ctr'], logits=ctr_logits), name="ctr_loss")
loss = tf.add(ctr_loss, cvr_loss, name="ctcvr_loss")

ctr_accuracy = tf.metrics.accuracy(labels=labels['ctr'], predictions=tf.to_float(tf.greater_equal(ctr_predictions, 0.5)))
cvr_accuracy = tf.metrics.accuracy(labels=y, predictions=tf.to_float(tf.greater_equal(prop, 0.5)))
ctr_auc = tf.metrics.auc(labels['ctr'], ctr_predictions)
cvr_auc = tf.metrics.auc(y, prop)
metrics = {'cvr_accuracy': cvr_accuracy, 'ctr_accuracy': ctr_accuracy, 'ctr_auc': ctr_auc, 'cvr_auc': cvr_auc}
tf.summary.scalar('ctr_accuracy', ctr_accuracy[1])
tf.summary.scalar('cvr_accuracy', cvr_accuracy[1])
tf.summary.scalar('ctr_auc', ctr_auc[1])
tf.summary.scalar('cvr_auc', cvr_auc[1])
if mode == tf.estimator.ModeKeys.EVAL:
  return tf.estimator.EstimatorSpec(mode, loss=loss, eval_metric_ops=metrics)

# Create training op.
assert mode == tf.estimator.ModeKeys.TRAIN
optimizer = tf.train.AdagradOptimizer(learning_rate=params['learning_rate'])
train_op = optimizer.minimize(loss, global_step=tf.train.get_global_step())
return tf.estimator.EstimatorSpec(mode, loss=loss, train_op=train_op)
```

Online-predict process input and ouput

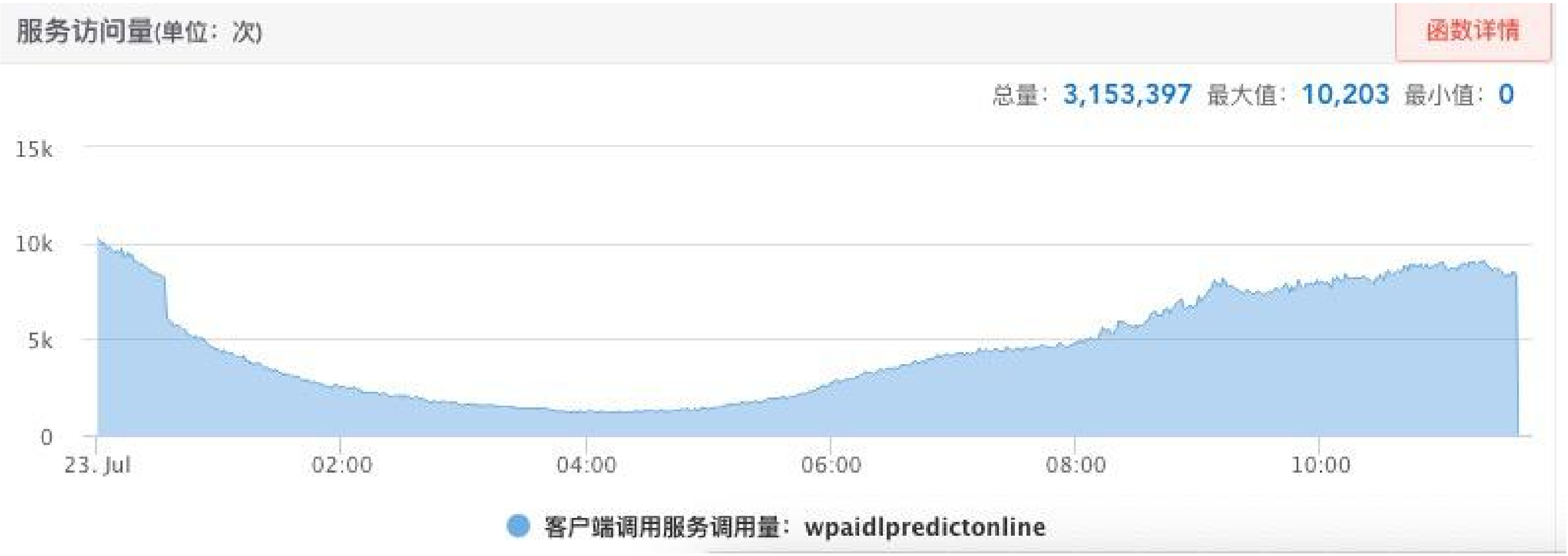
```java
}
List<ByteString> inputStrs = requestData.stream().map(o -> {

    String[] elems = o.toString().split("\\|", -1);
    Map<String, Feature> inputFeatures = new HashMap();

    buildFeature(FeatureType.INT_TYPE, "u_vpn", inputFeatures, elems[0]);
    buildFeature(FeatureType.INT_TYPE, "is_view", inputFeatures, elems[1]);
    buildFeature(FeatureType.INT_TYPE, "rcmd_r", inputFeatures, elems[2]);
    buildFeature(FeatureType.FLOAT_TYPE, "b_wl_sc", inputFeatures, elems[3]);
    buildFeature(FeatureType.FLOAT_TYPE, "b_gd_sc", inputFeatures, elems[4]);
    buildFeature(FeatureType.INT_TYPE, "b_sh_num", inputFeatures, elems[5]);
    buildFeature(FeatureType.INT_TYPE, "b_ph_times", inputFeatures, elems[6]);
    buildFeature(FeatureType.FLOAT_TYPE, "b_sh_r", inputFeatures, elems[7]);
    buildFeature(FeatureType.INT_TYPE, "u_rh_num", inputFeatures, elems[8]);
    buildFeature(FeatureType.INT_TYPE, "u_rp_num", inputFeatures, elems[9]);
    buildFeature(FeatureType.INT_TYPE, "u_ph_num", inputFeatures, elems[10]);
    buildFeature(FeatureType.FLOAT_TYPE, "u_rh_r", inputFeatures, elems[11]);
    buildFeature(FeatureType.FLOAT_TYPE, "u_rp_r", inputFeatures, elems[12]);
    buildFeature(FeatureType.FLOAT_TYPE, "u_fx3", inputFeatures, elems[13]);
    buildFeature(FeatureType.FLOAT_TYPE, "u_sc15", inputFeatures, elems[14]);
    buildFeature(FeatureType.FLOAT_TYPE, "u_sc3", inputFeatures, elems[15]);
    buildFeature(FeatureType.FLOAT_TYPE, "u_wl7", inputFeatures, elems[16]);
    buildFeature(FeatureType.FLOAT_TYPE, "u_fx7", inputFeatures, elems[17]);
    buildFeature(FeatureType.FLOAT_TYPE, "u_wl30", inputFeatures, elems[18]);
    buildFeature(FeatureType.FLOAT_TYPE, "u_dj15", inputFeatures, elems[19]);
    buildFeature(FeatureType.FLOAT_TYPE, "u_sc7", inputFeatures, elems[20]);
    buildFeature(FeatureType.FLOAT_TYPE, "u_fx15", inputFeatures, elems[21]);
    buildFeature(FeatureType.FLOAT_TYPE, "u_dh15", inputFeatures, elems[22]);
    buildFeature(FeatureType.FLOAT_TYPE, "u_dj3", inputFeatures, elems[23]);
    buildFeature(FeatureType.FLOAT_TYPE, "u_dj30", inputFeatures, elems[24]);
    buildFeature(FeatureType.FLOAT_TYPE, "u_wl15", inputFeatures, elems[25]);
    buildFeature(FeatureType.FLOAT_TYPE, "u_dh3", inputFeatures, elems[26]);
    buildFeature(FeatureType.FLOAT_TYPE, "u_dh30", inputFeatures, elems[27]);
    buildFeature(FeatureType.FLOAT_TYPE, "u_dj7", inputFeatures, elems[28]);
    buildFeature(FeatureType.FLOAT_TYPE, "u_dh7", inputFeatures, elems[29]);
    buildFeature(FeatureType.FLOAT_TYPE, "u_sc30", inputFeatures, elems[30]);
    buildFeature(FeatureType.FLOAT_TYPE, "u_wl3", inputFeatures, elems[31]);
    buildFeature(FeatureType.FLOAT_TYPE, "u_fx30", inputFeatures, elems[32]);
    buildFeature(FeatureType.INT_TYPE, "ch_d", inputFeatures, elems[33]);

    Features featuresSerializeToString = Features.newBuilder().putAllFeature(inputFeatures).build();
    ByteString inputStr = Example.newBuilder().setFeatures(featuresSerializeToString).build().toByteString();
    return inputStr;
}).collect(Collectors.toList());

TensorShapeProto.Builder tensorShapeBuilder = TensorShapeProto.newBuilder();

tensorShapeBuilder.addDim(TensorShapeProto.Dim.newBuilder().setSize(requestData.size()));
TensorShapeProto shape = tensorShapeBuilder.build();
TensorProto proto = TensorProto.newBuilder()
        .setDtype(DataType.DT_STRING)
        .setTensorShape(shape)
        .addAllStringVal(inputStrs)
        .build();
```

```java
@Override
public Object predictOnlineAfter(PredictResponse response) {

    if (response == null) {
        return null;
    }

    TensorProto ctcvr = response.getOutputsOrDefault(OUTPUT_NAME_1, null);
    TensorProto ctr = response.getOutputsOrDefault(OUTPUT_NAME_2, null);
    TensorProto cvr = response.getOutputsOrDefault(OUTPUT_NAME_3, null);

    Map<String, Float> res = new HashMap<>();
    res.put("ctcvr", ctcvr == null ? 0.0f : ctcvr.getFloatVal(0));
    res.put("ctr", ctr == null ? 0.0f : ctr.getFloatVal(0));
    res.put("cvr", cvr == null ? 0.0f : cvr.getFloatVal(0));

    return res;

}
```

Api调用

```java
/** 获取ESMM精排结果
 * @param requestDTO  参数
 * @param preRecommendations 待排房源集合
 * */
public List<Map<String, Object>> getESMMRankingResult(String userId, List<Map<String, Object>> preRecommendations){
    List<String> propIds = new ArrayList<>();
    for(Map<String, Object> prop : preRecommendations) {
        propIds.add(prop.get("pro_id").toString());
    }

    // 调用esmm模型预测ctr, cvr
    List<Object> dataList = getFeatureCols(userId, propIds);
    @SuppressWarnings("unchecked")
    Map<String, List<Float>> result = (Map<String, List<Float>>) wpaiService.tensorflowServingPredictOnline(2575, dataList);
    logger.info("Wpai2575 " + " result = " + result);
    if (result != null) {
        @SuppressWarnings("unchecked")
        List<Float> ctrs = result.getOrDefault("ctr", Collections.EMPTY_LIST);
        @SuppressWarnings("unchecked")
        List<Float> cvrs = result.getOrDefault("cvr", Collections.EMPTY_LIST);

        for(int i =0;i<preRecommendations.size();i++) {
            Float ctr = ctrs.get(i);
            Float cvr = cvrs.get(i);
            Map<String, Object> propInfo = preRecommendations.get(i);
            Double simScore = Double.parseDouble(propInfo.get("sim_score").toString()) + ctr + cvr*2;
            propInfo.put("sim_score", simScore);
        }
    }

    return preRecommendations;
}

/** 获取预测特征列
 * @param userId 用户设备id  (a-ajk : i, i-ajk:udid2)
 * @param propIds 房源id集合
 * */
public List<Object> getFeatureCols(String userId, List<String> propIds){
    // 获取用户特征列
    String userRowKey = "uid_fea:" + userId;
    Map<String, String> userWtableRes = wtableService.mGetValue("133496842", 2, Arrays.asList(userRowKey), "148973");
    String userFeatures = userWtableRes.getOrDefault(userRowKey, "");

    String userFeaturesCol = getFeatureMapValue(userFeatures, "fea", "0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|1|0|0|0");

    // 获取房源特征列
    List<String> propRowKeys = new ArrayList<String>();
    String contextRowKeys = "context_fea:"+userId;
    for(String propId : propIds) {
        propRowKeys.add("prop_fea:" + propId);
    }
    Map<String, String> propWtableRes = wtableService.mGetValueWithBatch("133496842", 2, propRowKeys, "148973", 88);
    Map<String, String>  contextWtableRes = wtableService.mGetValueWithBatch("133496842", 2, Arrays.asList(contextRowKeys), "148973", 88);

    // 构造全特征列
    List<Object> dataList = new ArrayList<>();
    for(String propId : propIds) {
        String propFeatures = propWtableRes.getOrDefault("prop_fea:"+propId, "");
        String propFeaturesCol = getFeatureMapValue(propFeatures, "fea", "3.6|7.8|3|1|2|0.8333333333333334|6|1|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0");

        String contextFeatures = contextWtableRes.getOrDefault("context_fea:"+userId, "");
        String contextFeaturesCol = getFeatureMapValue(contextFeatures, "fea", "0|0|0|0|0|0|0|3|0");

        dataList.add(propFeaturesCol + "|" + userFeaturesCol + "|" + contextFeaturesCol);
    }

    return dataList;
}
```

性能与监控


服务访问量(单位：次)
函数详情
总量：3,153,397 最大值：10,203 最小值：0
15k
10k
5k
0
23. Jul
02:00
04:00
06:00
08:00
10:00
客户端调用服务调用量：wpaidlpredictonline


访问量耗时(单位：耗时)
函数详情
平均值：5.77 最大值：9.2 最小值：0
15
10
5
0
23. Jul
02:00
04:00
06:00
08:00
10:00
客户端调用服务调用量耗时：wpaidlpredictonline

- Q&A