最终报告

之前的工作我们已经对数据进行了一定的清洗和分析工作，现在我们需要做如下一些工作：

1. 特征选取（选取对预测的有更多决定作用的特征）
2. 选择模型进行预测

1. 特征选取

我们的目标是预测未来几天用户是否购买某个类目下的商品，经过简单的分析大概有 30%的购买是有过行为记录的，所以我们的重点是分析有过购买记录的用户的行为情况。

根据已有的用户行为数据（浏览，点击，加购物车，删购物车，关注，下单），使用滑动窗口进行简单的特征提取。

利用滑动窗口处理行为数据。用户在购买商品之前，会对商品有过不同的行为，近期的行为影响较大，远期的行为影响较少。建立在这个逻辑下，我们可以统计不同时间区间内用户的行为累积（包括用户和商品的购买率等累积特征）。示例代码如下：

① 统计时间区间内行为累积

```
def get_action_feat(start_date, end_date):
    dump_path = './cache/action_accumulate_%s_%s.pkl' % (start_date, end_date)
    if os.path.exists(dump_path):
        actions = pickle.load(open(dump_path))
    else:
        actions = get_actions(start_date, end_date)
        actions = actions[['user_id', 'sku_id', 'type']]
        df = pd.get_dummies(actions['type'], prefix='%s-%s-action' % (start_date, end_date))
        actions = pd.concat([actions, df], axis=1)  # type: pd.DataFrame
        actions = actions.groupby(['user_id', 'sku_id'], as_index=False).sum()
        del actions['type']
        pickle.dump(actions, open(dump_path, 'w'))
    return actions
```

② 统计距离预测开始日期前 1、2、3、5、7、10、15、21、30 天的行为累积

```
    actions = None
    for i in (1, 2, 3, 5, 7, 10, 15, 21, 30):
        start_days = datetime.strptime(train_end_date, '%Y-%m-%d') - timedelta(days=i)
        start_days = start_days.strftime('%Y-%m-%d')
        if actions is None:
            actions = get_action_feat(start_days, train_end_date)
        else:
            actions = pd.merge(actions, get_action_feat(start_days, train_end_date), how='left',
                               on=['user_id', 'sku_id'])
```

2. 模型选择

从简单到复杂，我们尝试了不同的方法来进行最后的购买预测。

（1） 查询统计的方法

根据有购买记录的用户购买之前的不同行为统计（下表所示），直接提取满足特定行为次数的（用户、商品）对作为最终结果。

	浏览	加入购物车	购物车删除	下单	关注	点击
最小值	0	0	0	1	0	0
均值	10.41	1.674	0.1781	1	0.105	27.21
最大值	266	50	38	1	4	665
中值	7	1	0	1	0	18
1分位点	3	1	0	1	0	8
3分位点	13	2	0	1	0	35

从该表中看出，浏览和点击对最终购买的影响较大，故查询了满足特定的浏览次数和点击次数但尚未购买的商品作为用户最终会购买的商品，结果分数为 0。

（2） 协同过滤

上述直接查询统计的方法，只考虑了用户单个行为，尝试通过商品推荐的思路，考虑用户之间的相关性，认为具有相似购买记录的用户也会购买另一用户评分最好的商品。所以将用户行为按照一定的权重进行累加作为用户对这个商品的喜爱程度（如：购买过的分值为 1，没有购买过的按照浏览、点击、加购、删购、关注顺序权重从大到小进行分值在 0 到 1 中的评分计算），使用 mathout 协同过滤库将（用户，商品，评分）三元组输入进行预测，最终分数为 0。

（3） xgboost 模型

根据用户历史行为记录预测时间段内可能的购买记录，用户行为的时间周期性变化应作为特征之一，而上述两种方法均未考虑这一重要特征。xgboost 模型具有速度快、效果好、功能多等特点，同时树模型对特征处理的要求不高效果不错，不管是类别特征，连续特征效果都很好。而且在以往 Kaggle 竞赛中，该模型取得较好成绩。使用滑动窗体提取行为特征，结合用户、商品行为购买率累计特征及商品属性、评价特征，将有购买记录的作为正样本输入 xgboost 逻辑回归模型进行训练预测，分数为 0.07。示例代码：

```
user_index, training_data, label = make_train_set(train_start_date, train_end_date, test_start_date, test_end_date)
X_train, X_test, y_train, y_test = train_test_split(training_data.values, label.values, test_size=0.2, random_state=0)
dtrain=xgb.DMatrix(X_train, label=y_train)
dtest=xgb.DMatrix(X_test, label=y_test)
param = {'learning_rate' : 0.1, 'n_estimators': 1000, 'max_depth': 3,
    'min_child_weight': 5, 'gamma': 0, 'subsample': 1.0, 'colsample_bytree': 0.8,
    'scale_pos_weight': 1, 'eta': 0.05, 'silent': 1, 'objective': 'binary:logistic'}
num_round = 283
param['nthread'] = 4
#param['eval_metric'] = "auc"
plst = param.items()
plst += [('eval_metric', 'logloss')]
evallist = [(dtest, 'eval'), (dtrain, 'train')]
bst=xgb.train(plst, dtrain, num_round, evallist)
sub_user_index, sub_trainning_data = make_test_set(sub_start_date, sub_end_date,)
sub_trainning_data = xgb.DMatrix(sub_trainning_data.values)
y = bst.predict(sub_trainning_data)
sub_user_index['label'] = y
pred = sub_user_index[sub_user_index['label'] >= 0.03]
pred = pred[['user_id', 'sku_id']]
pred = pred.groupby('user_id').first().reset_index()
pred['user_id'] = pred['user_id'].astype(int)
pred.to_csv('./sub/submission.csv', index=False, index_label=False)
```

3. 总结

对于此购买预测问题，最重要的两步是特征选取与模型选择。好的特征能简化后面的模型选取问题且效果不错，但对于此题目来说，所给的数据涉及到的特征太多（用户属性，用户行为历史记录，商品属性，商品评价），从这么多属性中选取有价值的属性难度较大，若有丰富的经验对有效特征的提取帮助较大，除此之外通过查询统计及相关性分析得到重要性较高的属性也是一种途径。而好的模型能自动学习提取特征，减少手工选取特征的困难，不过将问题进行抽象，选取适合该问题的模型是一个挑战。整体来说从这次实验来看，无论从特征选取还是模型选择都缺少相关积累和经验，所以效果不尽人意。