Real Data Analysis at PNC

Zhifeng Wang

Department of Statistics
Florida State University

PNC Bank

- PNC is a Pittsburgh-based financial services corporation.
- I worked in
  Marketing Department
  $\Longrightarrow$ Decision, Analytics and Research (DAR)
  $\Longrightarrow$ Marketing Science Innovation (MSI)
- The DAR group is responsible for the design, implementation, and execution of analytics-driven marketing strategies.

Their Datasets

- The main datasets used in MSI group are customer-product datasets.
  - **customers**: one million
  - **features** ($\# \approx 300$): including customer information and all products of PNC, e.g., balance, account number, total assets
  - **format**: weekly or monthly
- Target: predict product ownership and product opening (binary classification), obtain possible drivers.

Questions

- Product **ownership**: one-month lagged data.
  - use the data of August to predict the ownership in September.
- Product **opening**: six-month lagged data.
  - use data from customers who didn't have the product in March to predict whether they will have in the next six months.

Challenges

- Redundancy: many features in the weekly data have no change for a long time.
- Imbalance:
  - $< 10\%$ positive response for product ownership
  - $\approx 1\%$ positive response for product opening

Cross-Sectional Data

- First, we used cross-sectional data to deal with ownership problem and test different classification models (packages).

	hh_id	rlb_location_key	time_period	rcb_consumer_hh	core_hh	wmg_hh	acorn_hh	hh_agr_type	core_agr_type	cls_dym_hh	...	od_la
0	2	2000222001002153429	2017-05-31	1	1	0	1	1E	1E	0	...	0
1	5	2000222001005672586	2017-05-31	1	1	0	1	1E	1E	0	...	0
2	8	2000269000000000535	2017-05-31	1	1	1	1	1E	1E	0	...	0
3	11	2000269000000000622	2017-05-31	1	1	0	1	1E	1E	0	...	0
4	12	2000269000000000631	2017-05-31	1	1	0	1	1E	1E	0	...	0

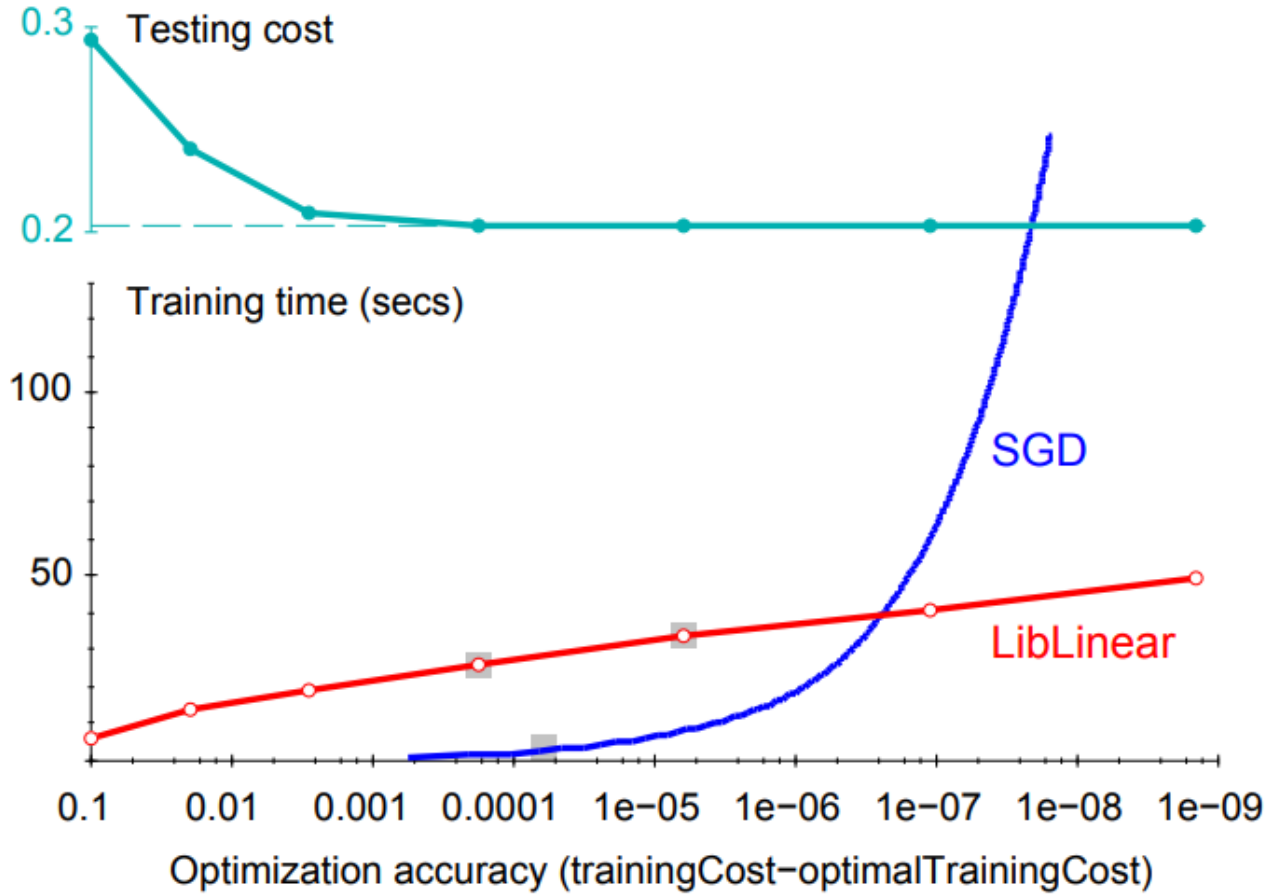
Models/Algorithms

- Logistic regression (with ℓ_2 penalty)
- SVM (SGD solver)
- kNN
- Random forest / extremely randomized forest
- Boosted trees (`xgboost`, `lightgbm`)
- Dense neural network (`keras`/`tensorflow`)
- *Stacking!*

Models/Algorithms

- Due to the imbalanced responses, we used AUC to evaluate prediction performance.
- Not surprisingly, boosted trees achieved the best prediction accuracy (got 0.92 AUC).
- The plain SVM solver is not scalable, we applied SGD to speed up computation in order to get a moderately accurate solution.

SVM Solvers



(picture from Leon Bottou's 2007 NIPS tutorial)

Some Packages

- `xgboost` and `lightgbm` are two popular packages for implementing boosted trees.
- `lightgbm` is developed by Microsoft (released last year), experiments show that it is often faster and a little bit more accurate than `xgboost`.
- `keras` is a Python Deep Learning library, which is high-level API running on top of `tensorflow`, `theano`, `CNTK` (now it supports `MXNet`!).

Dealing with Imbalance

- Weighted loss
  - was tried in all above methods except tree-based methods (weights proportional to the ratio).
  - improved the performance a lot.
  - the weights in tree-based methods seem pretty mysterious.
- Subsampling
  - oversample/downsample to balance data in subsets, then integrate the results from all subsets.
  - random forest can be implemented in an ensembled way with some mini-batch flavor.
  - performance is quite similar to that of using weighted losses.

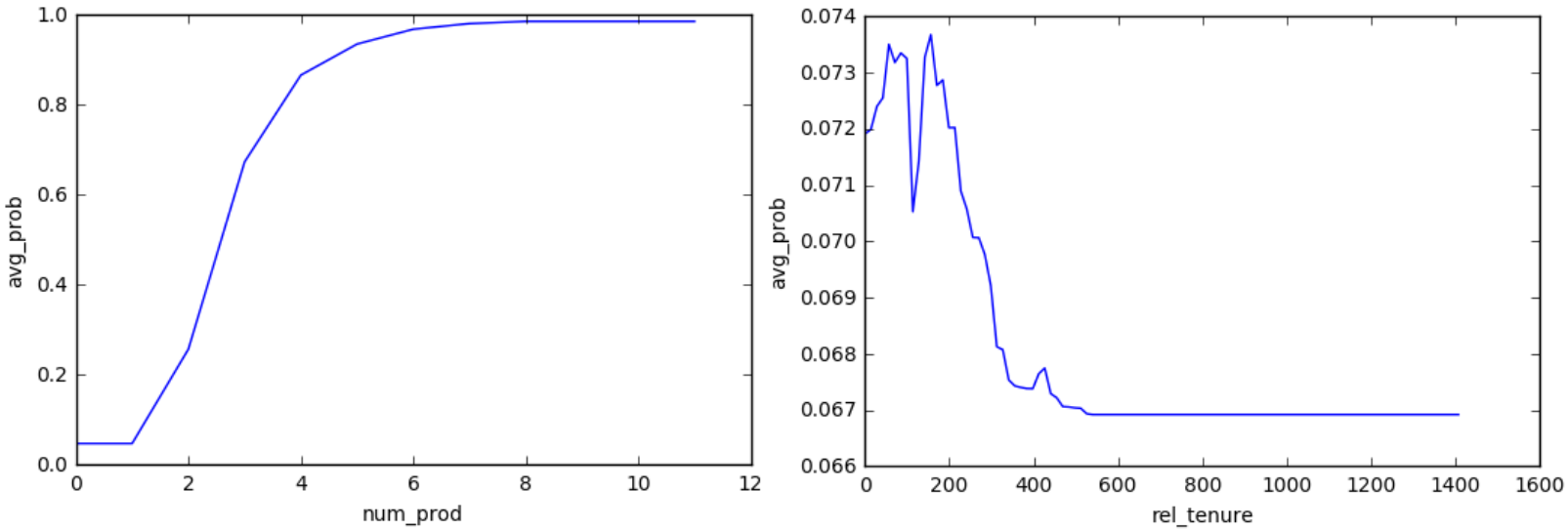
Related Results from Boosted Trees

- feature importance

	feature	score
0	num_prod	6371
1	rel_tenure	1397
2	dd_tenure	1302
3	cc_tenure	1198
4	sv_tenure	1113
5	age_hh	1042
6	mm_tenure	1015
7	dd_exp	890
8	num_acct	810
9	ploc_tenure	643
10	dd_rev	629

Related Results from Boosted Trees

- partial dependence plot



Some Practical Techniques

- Model stacking (in a cross-validated manner)
- Cat2Num
  - average response (may add some noise to avoid overfitting)
  - cross-validated average
  - replace by external related features (e.g., I replaced ZIP code with latitude and longitude)

Sequential Data

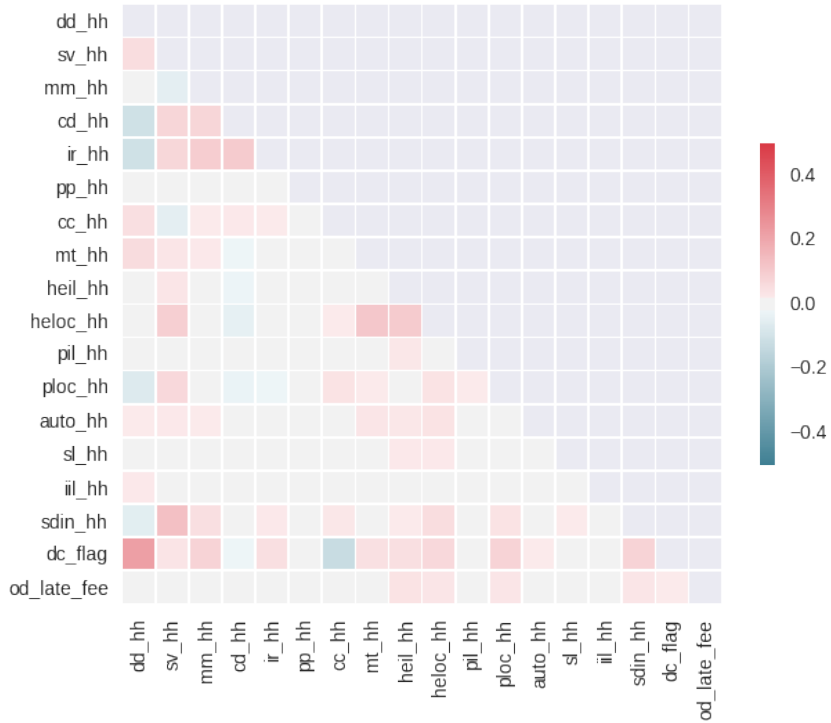
- The original dataset is formatted monthly (every customer has 5×12 records for this 5-year dataset).
- We would like to take advantage of the sequential data to improve prediction power.
- We used one-year monthly data to predict the product opening for the next 6 months.
- Ideas: RNN and CNN. (We do not know much about time series models)

Sequential Data

- To apply CNN, we transformed the data into a 3D matrix, each customer has a data matrix (like an image) with shape 12 (months) $\times$ 326 (features).
  - used a typical CNN architecture ([conv + relu + max-pool] $\times$ 2 + Dense $\times$ 2) with one-dim filters (e.g., with shape 1×3)
- We also tried a `SimpleRNN` model with input size 12×326.
- This was the work in the last week, didn't finish yet.

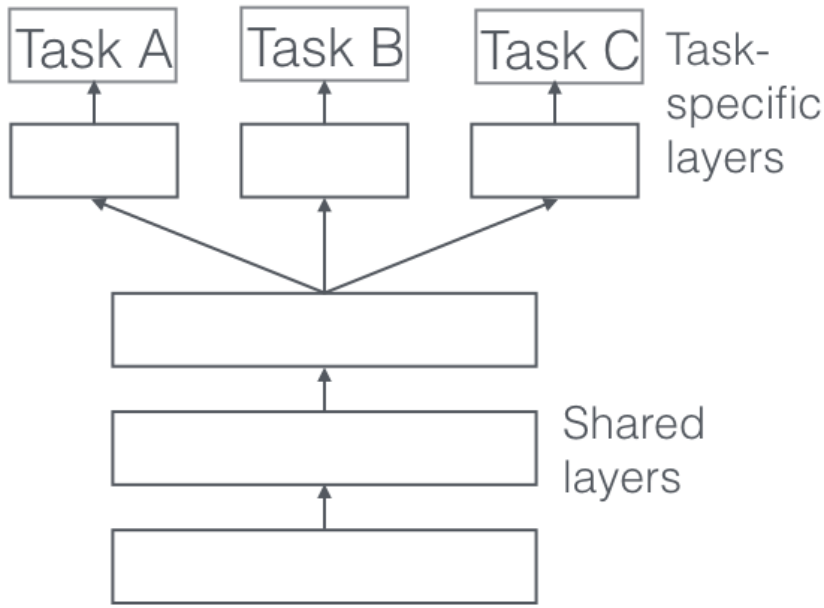
Multi-task Networks

- Another problem is to use very few features (only 20) to predict 18 binary outcomes (product ownership).
- We found that these responses have some mild correlation, we tried to use multi-task networks to take some information from different responses.



Multi-task Networks

- Multi-task learning got a better prediction accuracy for some responses that are difficult to predict independently.



Some Programming Techniques

- `RandomForestClassifier` in `sklearn` has a *warm_start* option, which can automatically ensemble your trees trained from different subsets.
- Parallelization
  - Extracting eligible data for the problems of product ownership and opening is really time-consuming, we finally could finish this process in 30 minutes.
  - By using `multiprocessing` package, we further reduced the processing time to less than 5 minutes.
  - `RandomForestClassifier` also supports parallel computing, just need to set $n\_jobs = -1$.

multiprocessing

```
import multiprocessing

def worker(num):
    """thread worker function"""
    print 'Worker:', num
    return

if __name__ == '__main__':
    jobs = []
    for i in range(5):
        p = multiprocessing.Process(target=worker, args=(i,))
        jobs.append(p)
        p.start()
```