Research Summary

SAP patents

1, Rating of city road segments for taxi hailing based on HANA technology, NO. 120542US01

- Goal: how to discover better locations for taxi requesters to hail their wanted taxis given historical data of taxis, start-destination pairs, time and season parameters of hailing events.
- Problem: It is a general problem in most of big cities to hail a taxi. Even if sometime just 100 meters away from their current location passengers can easily pick up a taxi, they still seem blind to their best options. Moreover, this issue is also dependent on seasons, week days as well as specific day time, which casts challenge to most citizens and visitors.
- How to solve the problem: We resort to massive taxi trace data (tens of millions per day) from taxi companies in a Chinese big city and personal location data from mobile app to help citizens and external visitors find the best nearby location to hail an open taxi given the aforementioned parameters. The location calculation has been done on SAP HANA in-memory calculation platform, which provide an instant insight to discover the best nearby road segments via HANA's high performance data mining capability.
- Difference from prior solution
  1. The calculation is not only based on historical taxi location data, but also integrates the real-time location data collected from mobile app. This can help improve accuracy of calculation results iteratively.
  2. The calculation is multi-dimensional which contains different measures, such as heading of taxis, pickups, taxi vacancy and ratio of demand and supply etc.
  3. The calculation has been executed on SAP HANA platform, which speeds up the whole process of data retrieval and analysis with less disk IO and optimized data indexes

2, Automatic category assignment and potential topic discovery for products based on Latent Dirichlet Topic algorithm, record ID. 83839165

- Goal: recognize the category of products on the product category tree when importing them into a system during the initialization phase
- Problem:
  - Joint training: how to train topic models and link prediction of products simultaneously?
  - Micro-Categories: how to establish product hierarchy tree with limited topic assignment on each product node?
  - Product graph and latent product relationship: how to identify complementary and substitute relationship between products?
- Algorithm:
  - $z_{d,j}$ is topic assignment for the j th word in document d (review documents for products)
  - $\theta_d \in$ Dirichlet Prior for 1...K topic assignment of each document, a multi-modal distribution: $f(x_1, ..., x_K; \alpha_1, ..., \alpha_K) = \frac{1}{\mathcal{B}(\alpha)} \prod_{i=1}^{K} x_i^{\alpha_i - 1}, \sum_{i=1}^{K} x_i = 1, \mathcal{B}(\alpha) = \frac{\prod_{i=1}^{K} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{K} \alpha_i)}$, having $\alpha = (\alpha_1, ..., \alpha_K), \Gamma(z+1) = \int_0^{+\infty} x^z e^{-x} dx = z\Gamma(z)$

$$p(\Gamma|\theta, \phi, z) = \prod_{d\in\Gamma} \prod_{i=1}^{N_d} \theta_{z_{d,j}} \times \phi_{z_{d,j}, w_{d,j}}$$

  - link prediction with topic models. Supposed product i, j have topic feature vector $\langle \theta_{i,1}, ..., \theta_{i,K} \rangle, \langle \theta_{j,1}, ..., \theta_{j,K} \rangle$ from the learnt document-topic model, their inter-relationship model can be established accordingly
    1. product pair (i, j)'s similarity:

$$\mathcal{F}_\beta(\theta_d) = \sigma(\langle \beta, \phi_\theta(i,j) \rangle), \phi_\theta(i,j) = (1, \theta_{i,1} \times \theta_{j,1}, ..., \theta_{i,K} \times \theta_{j,K})$$

    2. jointly-training model for the bi-direction graph having $\phi_\theta(i,j) = \phi_\theta(j,i)$:

$$\mathcal{L}(y, \Gamma|\beta, \theta, \phi, z) = \prod_{d\in\Gamma} \prod_{i=1}^{N_d} \theta_{z_{d,j}} \times \phi_{z_{d,j}, w_{d,j}} \prod_{(i,j)\in\xi} F_\beta(\phi_\theta(i,j)) \prod_{(i,j)\in\bar{\xi}} (1 - F_\beta(\phi_\theta(i,j)))$$

3. directional flow for positive relationship in the direct graph to include additional feature dimensions in $\phi_\theta(i,j)$, such as product price, VAPR(sale velocity, acceleration of sales, profit and revenue), with $\phi_\theta(i,j) \neq \phi_\theta(j,i)$:

$$\mathcal{L}(y,\Gamma|\beta,\theta,\phi,z) = \prod_{d\in\Gamma}\prod_{i=1}^{N_d}\theta_{z_{d,j}} \times \phi_{z_{d,j},w_{d,j}} \prod_{(i,j)\in\xi} F_\beta^{\leftrightarrow}(\phi_\theta(i,j))\mathcal{F}_\eta^{\rightarrow}(\phi_\theta(i,j))(1-\mathcal{F}_\eta^{\rightarrow}(\phi_\theta(j,i))) \prod_{(i,j)\in\bar{\xi}}(1-F_\beta^{\leftrightarrow}(\phi_\theta(i,j)))$$

4. extend to multiple graphs: we need to aggregate $g \in G$ for all sorts of considered relationships

- Hierarchical product category tree assigned with latent topics
  1. each product can be represented by a path from the root node to the leaf or the intermediate node in the product category tree
  2. each topic is associated with a particular node in the category tree, with maximum 100 topics on each product node
  3. sparse representation of each product's topic: product words can only be drawn from topics related with nodes in that path
     After assigning products in training sets to relevant category tree nodes, the latent topic model has been linked with product category as well. Therefore, once learnt the latent topic of newly imported products, their potential category tree nodes could be identified as well
- Optimization and training: EM process
  1. update $\Theta^{(t)} = \arg\max_\Theta l(y,\Gamma|\beta,\eta,\theta,\phi,z^{(t-1)}), \Theta^{(t)} = (\beta,\eta,\theta,\phi)$ having the fixed $z_{d,j}^{(t)}$
  2. sample $z_{d,j}^{(t)}$ with probability $p(z_{d,j}^{(t)} = k) = \theta_{d,k}^{(t)}\phi_{k,w_{d,j}}^{(t)}$, where $\theta_{d,k}^{(t)}\phi_{k,w_{d,j}}^{(t)}$ have been learnt from step 1

- Summary:
  1. Problem: assign products to the category tree
  2. Data: collect a comprehensive product category tree. And on the specific tenant, we only have a subset of the product tree, short description of products as well as user behavior data (click A then click B, click A then not click B, which reflects the production relationship), together with additional product features
  3. Training model: use a jointly training model - a latent dirichlet model with a logistic regression model for product inter-relationship, a document-to-topic distribution by Dirichlet Model and a word-to-topic distribution. We multiple topic distributions together to represent the corpus likelihood of available product documents. For each product in product relationship graphic based on user interaction data, we use logistic regression for positive sample and negative counterpart. Naturally, it is also possible to extend the model by adding additional product feature dimensions. Ultimately, a EM(expectation-maximizing) process is used to firstly optimize all of latent model variables given the fixed topic assignment of document-topic distribution, then sample new document-topic assignment given the learnt parameters in the previous step.

3, Simulator of bundle clicking for validating Bandit strategies in A/B testing, NO. 210412US01

- Goal: prepare an offline simulator to validate how Bandit strategies impact the reward convergence trend of different Bandit arms
- Problem:
  - how to set up the main process of the simulator: product & product bundle, the decay model of customer interest, A/B arms
  - understand the mechanism of different Bandit algorithms
- Algorithm:
  - E-Greedy

```
if np.random.rand() > self.epsilon:
  return self.__max_reward_arm()
else:
  return int(np.random.uniform(0, len(self.arms)))
```

- Softmax

$$probs = \frac{e^{\frac{r_i}{T}}}{\sum_i e^{\frac{r_i}{T}}},\ \text{choose the arm with} \arg\min_i \sum_i probs[i] > z$$

- UCB1 (Upper confidential bound): i is the arm index

$$\text{ucb}[i] = \text{reward}[i] + \sqrt{\frac{2\log(N)}{\text{count[i]}}}, \text{then select } \arg\max_i(ucb)$$

4, Reinforcement Learning Model for product recommendation considering balance between product profit and customer interests, NO. 210416US01

- Goal: establish a Reinforcement Learning (RL) Model to generate recommended product items, so our system can reach the balance between gaining product profit and satisfying customer interests
- Problem:
  - How to setup RL model four tuples (state, action, reward, transition)
  - How to integrate RL model with system
- Data:
  - Historical data preparation: statistics for clicked products' VAPR(sales velocity, acceleration of sales, profit, revenue), extra features(such as product price, customer trend and so forth)
  - Customer behavior data: customer's clicking on one page, product sorting on that page
- Algorithm:
  1. state definition: list of VAPR short-term features, customer purchase power, preference of item and product category with dimension K, $|s| = S$.
  2. action definition: returned product sorting, $\mu_\theta(s) = \langle \mu_\theta^1(s), ..., \mu_\theta^m(s) \rangle, \theta = (\theta_1, ..., \theta_m)$ is parameter vector of the actor, C_i is the sorting weight, the sorting weight $\mu_\theta(s)$ has **m product feature dimensions**, $|\mu_\theta(s)| = m$.

     $$\mu_\theta^i = \frac{C_i e^{\theta_i^T \phi(s)}}{\sum_{j=1}^m e^{\theta_j^T \phi(s)}}$$

     μ_θ is **the mapping from state s' projection** $\phi(s)$ to **the M sorting dimension**. In brief, we have the mapping functions ϕ, μ to convert state s via $(s, |s| = S) \xrightarrow{\phi} (\phi(s), |\phi(s)| = P) \xrightarrow{\mu} (\mu_\theta^i, |\mu_\theta^i| = M)$
  3. Reward function R and **reward shaping function** $\Phi(s)$ given customers' click and payment statistics, which naturally incorporates with prior knowledge of latent product distribution

     $$R(s, a, s') = \begin{cases} \text{normalized number of clicked products,} & \text{if clicking happens} \\ \text{normalized price of purchased products,} & \text{if purchase happens} \\ 0, & \text{otherwise} \end{cases}$$

     $$\Phi_{clk} = \sum_{i=1}^{K} y_i^c x_i^T \mu_\theta(s) - \ln(1 + e^{x_i^T \mu_\theta(s)})$$

     $$\Phi_{pay} = \sum_{i=1}^{K} y_i^p x_i^T \mu_\theta(s) - \ln(1 + e^{x_i^T \mu_\theta(s)}) + \ln Price_i$$

     where $x_i = (x_i^1, x_i^2, ..., x_i^m)$ is **the feature vector of product i with length M**, K is **the size of the rendered product list**.
     We also assume that **product i's click or payment probability** p_i **and its sorting score** $x_i^T \mu_\theta(s)$ fits for the relationship $\ln \frac{p_i}{1-p_i} = x_i^T \mu_\theta(s)$, where $|x_i| = |\mu_{\theta(s)}| = M$.
     then the final reward R(s, a, s') as follow:

     $$R(s, a, s') = R_0(s, a, s') + \Phi_{clk} + \Phi_{pay}$$

     After learnt μ_θ, we get $x_i^T \mu_\theta$ with K dimensions given different product features. With this sorting weight, we can decide how to sort products according to certain feature.
  4. Policy gradient

     $$\begin{aligned} \mathcal{J}(\mu_\theta) &= \int_S \int_S \sum_{t=1}^{\infty} \gamma^{t-1} p_0(s') T(s', \mu_\theta(s'), s) R(s, \mu_\theta(s)) ds' ds \\ &= \int_S \rho^\mu(s) R(s, \mu_\theta) ds = E_{s \sim \rho^\mu}[R(s, \mu_\theta(s))] \end{aligned}$$

     where $\rho^\mu(s) = \int_S \sum_{t=1}^{\infty} \gamma^{t-1} p_0(s') T(s', \mu_\theta(s'), s) ds'$

5. Policy gradient theorem

$$\begin{aligned}\nabla_\theta J(\mu_\theta) &= \int_S \rho^\mu(s)\nabla_\theta\mu_\theta(s)\nabla_a Q^\mu(s,a|a=\mu_\theta(s))ds \\ &= E_{s\sim\rho^\mu}[\nabla_\theta\mu_\theta(s)\nabla_a Q^\mu(s,a|a=\mu_\theta(s))]\end{aligned}$$

6. State-action Pair

$$\theta_{t+1} \leftarrow \theta_t + \alpha_\theta \nabla_\theta\mu_\theta(s)\nabla_a Q^\mu(s,a|a=\mu_\theta(s))$$

   In general, We use **the norm of the expected TD update (NEU)** $\sqrt{\mathbb{E}[\delta\phi]^T\mathbb{E}[\delta\phi]}$ to measure if the learning algorithm approaches convergence.

7. **Linear Function Approximation (LFA)**

$$Q^\mu(s,a) \approx Q^w(s,a) = \phi(s,a)^T w, \phi(s,a) = a^T\nabla_\theta\mu_\theta(s)$$

   with this approximator, we can get update equations for $Q(s,a), \delta_{t+1}, \theta_{t+1}, w_{t+1}, v_{t+1}$
   To alleviate the issue of inaccuracy of LFA, we introduce advantage function for Q function $Q(s,a) = A^w(s,a) + V^v(s)$

$$\begin{gathered}Q(s,a) = A^w(s,a) + V^v(s) = (a-\mu_\theta(s))^T\nabla_\theta\mu_\theta(s)^T w + \phi(s)^T v \\ \delta_{t+1} = r_t + \gamma Q(s_{t+1}, \mu_\theta(s_{t+1})) - Q(s_t,a_t) = r_t + \gamma\phi(s_{t+1})^T v_t - ((a-\mu_\theta(s))^T\nabla_\theta\mu_\theta(s)^T w + \phi(s)^T v) \\ \theta_{t+1} = \theta_t + \alpha_\theta\nabla_\theta(s_t)(\nabla_\theta\mu_\theta(s_t)^T w_t) \\ w_{t+1} = w_t + \alpha_w\delta_{t+1}\phi(s_t,a_t) = w_t + a_w\delta_{t+1}(a_t^T\nabla_\theta\mu_\theta(s_t)) \\ v_{t+1} = v_t + a_v\delta_{t+1}\phi(s_t)\end{gathered}$$

8. **Deep deterministic policy gradient (DDPG)**
   8.1 System components
   - Actor network $\mu(s|\theta^\mu)$ and critic network $Q(s,a|\theta^Q)$
   - **Learning To Rank** (ltr) loss layer $L(a,X,Y)$ to measure output a vector of the actor network, a kind of cross-entropy loss given pointwise ltr. Here $X = [x_1, x_2, ..., x_n]$ is the normalized features of n products, $Y = [y_1, y_2, ..., y_n$ is the label of click or payment.

$$L(a,X,Y) = \frac{1}{n}\sum_i^n y_i\log(\sigma(a^T x_i)) + (1-y_i)\log(1-\sigma(a^T x_i))$$

   8.2 Training process
   - Data preparation: we collect raw data from system, like VAPR, product features, customers' click and payment data, later convert them to transition trajectories, fitting for the MDP (state, action, next state, reward) tuples
   - Considering the limited training data, a replay buffer to augment the data size, like Dagger algorithm in imitation learning has been used for accumulating sufficient training samples.
     1. Sample action for a batch of states, use current DDPG network to retrieve relevant best action, then learn the next state and reward (stepping-out along the learnt best policy). All of those sampled trajectories will be appended into the shared replay buffer.
     2. When the size of the replay buffer reach the minimal sample size, we will train Q network and actor network in sequentially and monitor the trend of temporal different error (TD error).
   - Network internal design:
     1. trainable Q network: actor network $\text{argmax}_a[\mu_\theta(s)] = \text{argmax}_a[\{a\}] \rightarrow$ critic network $Q_\theta(s,\{a\}) = \{q\}$
     2. non-trainable target network: $Q_{\theta'}(s, \mu_{\theta'}(s)) = \{q'\}$
     3. compare one step of Q value from trainable part $\{reward\} + \gamma * \{q\}$ with non-trainable part $\{q'\}$
     4. prepare trainable operators: td error for q value, then copy Q network weights to target Q network

SAP research

1, SAP Nanjing Innovation Center - CONOP optimization

- Goal

1. **Toolkit development(VS project plugin + Profiling)** to analyze the gap between theoretical analysis and runtime performance, which helps us figure out the focus of performance improvement
2. Analysis of Simulate Annealing implementation of CONOP, especially convex constrains inside the non-convex internal loop, DP patterns and CPU cache hits analysis
3. Parallelization for the existing algorithm (inner loop parallelization, external loop sampling) and heuristic triggering of parallelization

2, SAP Nanjing Innovation Center - Smart traffic

- Goal: How to apply algorithms in real traffic scenarios (OpenGSM, public traffic data, mobile app statistics from citizens)
  1. Dynamic traffic zone extraction: map traffic statistic to bitmap and discover the boundary of different pixel clusters (Convert a clustering problem to image boundary discovery)
  2. Short-term congestion prediction: Markov Chain Reasoning based on neighborhood relationship of node(street corner) and edge(street segment)
  3. fake vehicle plate number discovery(time-location assumption) and plate number prediction for incomplete records(assumption for the same location under similar weather conditions)

3, SAP Upscale - Discovered potential product bundles from click data of customers based on Apriori and FP-Growth algorithms

- Goal: to discover associate items with traditional data mining algorithms
  1. Apriori: aggregate items by combination from top to bottom and make uses of parent-child relationship to filter non-frequent items
  2. FP-Growth
     - Prepare tree node and its frequency
     - Construct FP tree
     - Extract conditional pattern base(prefix path)

4, SAP Upscale - Developed product bundle recommendation via Collaborative Filtering Algorithm based on short-term customer interests

- Goal: how to use CF to discover product and customer feature given user-product rating matrix
  1. Loss function

$$\mathcal{J}(x,\theta) = \frac{1}{2}\sum_{(i,j):r(i,j)=1}((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}(x_k^{(i)})^2 + \frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}(\theta_k^{(j)})^2$$

  2. Optimize via gradient descent for $x_k^{(i)}$ and $\theta_k^{(j)}$

$$x_k^{(i)} -= \alpha\{\sum_{j:r(i,j)=1}((\theta^{(j)})^T x^{(i)} - y^{(i,j)})\theta_k^{(j)} + \lambda x_k^{(i)}\}$$
$$\theta_k^{(j)} -= \alpha\{\sum_{j:r(i,j)=1}((\theta^{(j)})^T x^{(i)} - y^{(i,j)})x_k^{(j)} + \lambda \theta_k^{(i)}\}$$

5, SAP Engagement Center - Analyzed system bottleneck and optimized the service based on payload statistics and services' dependency relationship

- Goal
  1. Construct the direct graph of services and relevant backend resources(DB, Kafka, etc.) based on their caller-callee relationship: static Spring boot's setting and dynamic web request routing
  2. Trace how requests have been passed through services and backend resources via unique request id and link requests
  3. Analyze cumulated request volumes, discover the high/low request volume nodes and optimize service dependency accordingly

6, SAP Engagement Center - Conducted service exception discovery model based on Multivariate Gaussian distribution and analyzed efficiency of exception handling in the system

- Comments: this model can well handle with explicit exceptions collected from log as well as implicit exceptions caused by abnormal resource consumption or incorrect program logic
    - Part 1 - URL stability representation
        1. Version 1: oversimplified model via statistics vector
           Drawback: some statistics are not independent with each other, like CPU/Memory consumption. In another word, the latent variables under those statistics can strengthen or weaken with each other. Therefore, the ultimate learnt URL stability representation vectors can't fully represent its status.
        2. Version 2: stability co-occurrence matrix of Web URL resource
           Approach:
            - Static scanning to extract controller dependency for each Web URL resource
            - Construct co-occurrence exception matrix U of Web URL resources for a single Spring boot Service: $U(i, i)$ will store **exceptions without interdependencies**, while $U(i, j)$ will be fill with **the estimated average exception count for the remaining exceptions**
            - Decomposition of U to **the latent representation vector** for each Web URL resource with the fixed dimension length K
            - Construct Multi-Gaussian Distribution given latent representations of Web URL resources, assuming that each dimension is independent with each other
    - Part 2 - Analysis of exception handling model
        1. Collect statistics and convert them by time series for each URL resource. Supposed in a service with N URL resources, the feature vector for URL resource $\mathrm{URL}_i^{(t)} = \langle S_0^{(t)}, S_1^{(t)}, \ldots, S_N^{(t)} \rangle$, where $|S_i^{(t)}| = K, i \in [1, N]$
        2. For each time t, we use Multi-Gaussian Distribution to learn its $\mu^{(t)} = \langle u_1^{(t)}, u_2^{(t)}, \ldots, u_K^{(t)} \rangle$ as well as $\Sigma^{(t)} = \langle \sigma_1^{(t)}, \sigma_2^{(t)}, \ldots, \sigma_K^{(t)} \rangle$. About details of Multi-Gaussian Distribution, refer to The Multivariate Gaussian Distribution.
        3. The exception discovery problem can be formulated to find the most instable $\mu^{(t)}$ and $\Sigma^{(t)}$
    - Drawback:
        1. Latent variable vector of matrix U **mayn't have a clear latent semantic**
        2. The concatenation of $U_{i,j}$, where i, j is the considered factor pairs, may involve with multiple latent variables, which break our independency assumption of latent variables

7, SAP Nanjing Innovation Center - Analyzed public opinion for Nanjing 12345 hotline by leveraging multi-class classification algorithms and event extraction based on dependency parsing and temporal relationship analysis for events based on rules

- Goal:
    1. How to classify citizens' feedback to pre-defined classes
    2. How to extract concerned events and organize them chronologically
- Part 1 - classification model
    1. Tokenization, remove stop words, convert them within the fixed length given dictionary
    2. Algorithms
        - One-to-Rest SVM for multi-label classification
        - Use TF-IDF to construct feature vector for each sentence, then classify for multi-labels via K-mean/multi-class logistic regression
        - Use KNN for testing sample to measure euclidean distance with training sample
        - Bayesian classification via argmax for add-smoothing of K-classes
- Part 2 - event extraction model
    1. Simplify sentence based on keyword matching and extract event based on dependency parsing
    2. Temporal rule to link events together

8, SAP Nanjing Innovation Center - Analyzed clues for key groups with public security concerns

- Goal: how to discover the possible event transition path and extract its critical path
- Algorithms:
    1. Generate multiple community relationship graphs given available social relationships (family, students, gaming as well as information transition)

2. Sort graphs according to the sensitivity level of relevant public security events, use Floyd-Warshall algorithm to get the transition graphs in the Nth transition closure
3. Sum together for multiple transition closures to get the finalize influence graph in community
4. Use topological sort to discover critical paths for the transition DAG of each sensitive event