Online Parameter Selection for Web-based Ranking via Bayesian Optimization

Sep 12, 2019

Kinjal Basu
Staff Software Engineer, Flagship AI
Agenda

1. Problem Setup
   LinkedIn Feed

2. Reformulation as a Black-Box Optimization

3. Explore-Exploit Algorithm
   Thompson Sampling

4. Infrastructure

5. Results
LinkedIn Feed

**Mission:** Enable Members to build an active professional community that advances their career.
LinkedIn Feed

Mission: Enable Members to build an active professional community that advances their career.

Heterogenous List:
- Shares from a member’s connections
- Recommendations such as jobs, articles, courses, etc.
- Sponsored content or ads
Important Metrics

**Viral Actions (VA)**
Members liked, shared or commented on an item

**Job Applies (JA)**
Members applied for a job

**Engaged Feed Sessions (EFS)**
Sessions where a member engaged with anything on feed.
Ranking Function

- $m$ – Member, $u$ - Item

$$S(m, u) := P_{VA}(m, u) + x_{EFS} P_{EFS}(m, u) + x_{JA} P_{JA}(m, u)$$

- The weight vector $x = (x_{EFS}, x_{JA})$ controls the balance between the three business metrics: EFS, VA and JA.

- A Sample Business Strategy is

$$\text{Maximize.} \quad VA(x)$$
$$\text{s.t.} \quad EFS(x) > c_{EFS}$$
$$\quad JA(x) > c_{JA}$$
Major Challenges

- The optimal value of x (tuning parameters) changes over time
- Example of changes
  - New content types are added
  - Score distribution changes (Feature drift, updated models, etc.)

- With every change engineers would manually find the optimal x
  - Run multiple A/B tests
  - Not the best use of engineering time
Reformulation into a Black-Box Optimization Problem
Modeling The Metrics

- \( Y_{i,j}^k (x) \in \{0,1\} \) denotes if the \( i \)-th member during the \( j \)-th session which was served by parameter \( x \), did action \( k \) or not. Here \( k = \text{VA}, \text{EFS} \) or \( \text{JA} \).

- We model this data as follows
  \[
  Y_{i}^k \sim \text{Binomial} \left( n_i(x), \sigma(f_k(x)) \right)
  \]
  where \( n_i(x) \) is the total number of sessions of member \( i \) which was served by \( x \) and \( f_k \) is a latent function for the particular metric.

- Assume a Gaussian process prior on each of the latent function \( f_k \).
Reformulation

We approximate each of the metrics as:

\[
\begin{align*}
VA(x) &= \sigma(f_{VA}(x)) \\
EFS(x) &= \sigma(f_{EFS}(x)) \\
JA(x) &= \sigma(f_{JA}(x))
\end{align*}
\]

The original optimization problem can be written through this parametrization.

Maximize \( VA(x) \)  
\[ \text{s.t.} \quad EFS(x) > c_{EFS} \]
\[ \quad JA(x) > c_{JA} \]

Maximize \( \sigma(f_{VA}(x)) \)  
\[ \text{s.t.} \quad \sigma(f_{EFS}(x)) > c_{EFS} \]
\[ \quad \sigma(f_{JA}(x)) > c_{JA} \]

Maximize \( f(x) \)  
\[ \quad x \in X \]

**Benefit:** The last problem can now be solved using techniques from the literature of Bayesian Optimization.
Explore-Exploit Algorithms
Bayesian Optimization

A Quick Crash Course

- Explore-Exploit scheme to solve

\[
\begin{align*}
\text{Maximize} & \quad f(x) \\
\text{subject to} & \quad x \in X
\end{align*}
\]
Bayesian Optimization

A Quick Crash Course

• Explore-Exploit scheme to solve
  \[
  \text{Maximize } f(x) \\
  \text{ } x \in X
  \]
  
• Assume a Gaussian Process prior on \( f(x) \).

• Start with uniform sample
  get \((x, f(x))\)
  
• Estimate the mean function and covariance kernel
Bayesian Optimization

A Quick Crash Course

- Explore-Exploit scheme to solve
  \[
  \text{Maximize } f(x) \\
  x \in X
  \]
- Assume a Gaussian Process prior on \( f(x) \).

- Start with uniform sample
  - get \((x, f(x))\)
- Estimate the mean function and covariance kernel
- Draw the next sample \(x\) which maximizes an “acquisition function” or predictive posterior.
- Continue the process.
Bayesian Optimization
A Quick Crash Course

- Explore-Exploit scheme to solve
  \[
  \text{Maximize } f(x) \\
  x \in X
  \]
- Assume a Gaussian Process prior on \( f(x) \).

- Start with uniform sample get \((x, f(x))\)
- Estimate the mean function and covariance kernel
- Draw the next sample \(x\) which maximizes an “acquisition function” or predictive posterior.
- Continue the process.
Bayesian Optimization
A Quick Crash Course

- Explore-Exploit scheme to solve
  \[
  \text{Maximize} \quad f(x) \\
  x \in X
  \]
- Assume a Gaussian Process prior on \( f(x) \).

- Start with uniform sample
  get \((x, f(x))\)
  - Estimate the mean function and covariance kernel
  - Draw the next sample \( x \) which maximizes an “acquisition function” or predictive posterior.
  - Continue the process.
Bayesian Optimization
A Quick Crash Course

- Explore-Exploit scheme to solve
  \[ \text{Maximize } f(x) \]
  \[ x \in X \]
- Assume a Gaussian Process prior on \( f(x) \).

- Start with uniform sample
  get \((x, f(x))\)
- Estimate the mean function and covariance kernel
- Draw the next sample \( x \) which maximizes an “acquisition function” or predictive posterior.
- Continue the process.
Thompson Sampling

- Consider a Gaussian Process Prior on each $f_k$, where $k$ is VA, EFS or JA
- Observe the data $(x, f_k(x))$
- Obtain the posterior of each $f_k$ which is another Gaussian Process
- Sample from the posterior distribution and generate samples for the overall objective function.
- We get the next distribution of hyperparameters by maximizing the sampled objectives (over a grid of QMC points).
- Continue this process till convergence.

\[
\begin{align*}
\text{Maximize} & \quad \sigma(f_{VA}(x)) \\
\text{s.t.} & \quad \sigma(f_{EFS}(x)) > c_{EFS} \\
& \quad \sigma(f_{JA}(x)) > c_{JA}
\end{align*}
\]
Infrastructure
Overall System Architecture
Offline System
The heart of the product

Tracking
- All member activities are tracked with the parameter of interest.
- ETL into HDFS for easy consumption

Utility Evaluation
- Using the tracking data we generate \( (x, f_k(x)) \) for each function \( k \).
- The data is kept in appropriate schema that is problem agnostic.

Bayesian Optimization
- The data and the problem specifications are input to this.
- Using the data, we first estimate each of the posterior distributions of the latent functions.
- Sample from those distributions to get distribution of the parameter \( x \) which maximizes the objective.
The Parameter Store and Online Serving

- The Bayesian Optimization library generates
  - A set of potential candidates for trying in the next round \((x_1, x_2, ..., x_n)\)
  - A probability of how likely each point is the true maximizer \((p_1, p_2, ..., p_n)\) such that \(\sum_{i=1}^{n} p_i = 1\)

- To serve members with the above distribution, each `memberId` is mapped to \([0,1]\) using a hashing function \(h\). For example, if \(\sum_{i=1}^{k} p_i < h(Kinjal) \leq \sum_{i=1}^{k+1} p_i\)

Then my feed is served with parameter \(x_{k+1}\)

- The parameter store (depending on use-case) can contain
  - \(<\text{ParameterValue}, \text{probability}>\) i.e. \((x_i, p_i)\) or
  - \(<\text{memberId}, \text{parameterValue}>\)
Online System
Serving hundreds of millions of users

Parameter Sampling
- For each member $m$ visiting LinkedIn,
- Depending on the parameter store, we either evaluate $<m, \text{parameterValue}>$
- Or we directly call the store to retrieve $<m, \text{parameterValue}>$

Online Serving
- Depending on the parameter value that is retrieved (say $x$), the member’s full feed is scored according to the ranking function and served

$$S(m, u) := P_{VA}(m, u) + x_{EFS} P_{EFS}(m, u) + x_{JA} P_{JA}(m, u)$$
Practical Design Considerations

- **Consistency in user experience.**
  - Randomize at member level instead of session level.

- **Offline Flow Frequency**
  - Batch computation where we collect data for an hour and run the offline flow each hour to update the sampling distribution.

- **Assume \((f_{VA}, f_{EFS}, f_{JA})\) to be Independent**
  - Works well in our setup. Joint modeling might reduce variance.

- **Choice of Business Constraint Thresholds.**
  - Chosen to allow for a small drop.
Results
Simulation Results

(a) Trimodal Shekel Function

(b) Decay of log relative square error
# Online A/B Testing Results

**Table 1: Online A/B results for Online Parameter Selection in LinkedIn Feed Ranking**

<table>
<thead>
<tr>
<th>Metric</th>
<th>Lift (%) vs Control $x_{c_1}$</th>
<th>Lift (%) vs Control $x_{c_2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Viral Actions</td>
<td>+3.3%</td>
<td>+1.2%</td>
</tr>
<tr>
<td>Engaged Feed Sessions</td>
<td>-0.8%</td>
<td>0%</td>
</tr>
<tr>
<td>Job Applies</td>
<td>+12.8%</td>
<td>+6.4%</td>
</tr>
</tbody>
</table>
Online Convergence Plots

(a) Iteration = 10

(b) Iteration = 15

(c) Iteration = 20

(d) Iteration = 25

(e) Iteration = 30

(f) Iteration = 35
Key Takeaways

- Removes the human in the loop: Fully automatic process to find the optimal parameters.
- Drastically improves developer productivity.
- Can scale to multiple competing metrics.
- Very easy onboarding infra for multiple vertical teams. Currently used by Ads, Feed, Notifications, PYMK, etc.

Future Direction

- Add on other Explore-Exploit algorithms.
- Move from Black-Box to Grey-Box optimizations
- Create a dependent structure on different utilities to better model the variance.
- Automatically identify the primary metric by understanding the models better.
Thank you
Appendix – Library API

- Problem Specifications

```json
{
    "treatmentModels": ["treatmentModel-1"],
    "controlModel": "controlModel-1",
    "exploreNumIterations": "6",
    "params": {
        "fieldName": "threshold",
        "parameterInfo": {
            "searchRange": {
                "low": "0.17",
                "high": "0.24"
            },
            "dataType": "Float"
        }
    }
}
```
Appendix – Library API

- Objective and Constraints

```json
"Objective": {
  "objectiveType": "max",
  "objectiveParts": [
    {
      "utilityName": "ClickRate",
      "ColumnNames": {
        "clickCount",
        "impressedCount"
      },
      "distribution": "gaussian"
    }
  ]

"Constraints": {
  "utilityName": "SendsByGenerated",
  "ColumnNames": {
    "sentCount",
    "generatedCount"
  },
  "distribution": "gaussian",
  "upperBound": {
    "multiplier": "Inf"
  },
  "lowerBound": {
    "multiplier": "1.0"
  }
}
```