Optimizing ad-auction reserve prices when the outcome is measured on an aggregate basis
Realtime media pricing

- An introduction to programmatic media selling
- The dynamic impression pricing problem
  - A Metropolis-Hasting approach
  - A Gaussian process modelling + UCB
- Comparison on a simple case
An introduction to programmatic media selling
The Real-Time Bidding revolution

- Since 2008 more and more ad spaces are sold in real-time during the loading of a web-page (~100ms)
- 30% of display ads impressions in France, 50% in the US
- An estimated $30Bn market for 2017
Main factors

- *Maturity* of the traditional/manual ad market: remnant inventory
- *Competition* organized through inefficient daisy-chains, harming user experience
- Ability to process *large amounts of data* and leverage it
- Real-time auction *technology*
- Long term trend toward *automation*
AlephD helps publishers in taking pricing decision

**Pricing individual RTB auctions in real-time**
- 2nd price auction
- How to bargain / set a reserve price in Real-time /
  10k times a sec?

**Pricing packaged impressions**
- Guaranteed price / volume
- Call options
- Quality grade

**Making sense of the impression market**
- Large number of heterogeneous assets
- High asymmetry of technology / information
- Availability known 100ms before sale
Auction mechanics

**Impression** \( i \in \mathcal{I} \)
- User
- Placement

**Advertisers** \( a_1, \ldots, a_n \)
- Bids \( b_{i,1}, \ldots, b_{i,n} \)

**Publisher (seller)**
- Reserve price \( r_{i,j} \)

**Auction**
- Winner
  \[
  a_w = \arg\max_{a_j} \{ b_{i,j} \mid b_{i,j} \geq r_{i,j} \}
  \]
- Winning bid
  \( b_{i,w} \)
- Closing price
  \[
  c_i = \max ( \{ b_{i,j} \mid b_{i,j} < b_{i,w} \} , r_{i,w} )
  \]
The second price auction mechanism

User visits site

Site won’t sell below floor

Advertisers bid

Winner pays 2nd price
The second price auction mechanism

User visits site

Site won’t sell below floor

Advertisers bid

Winner pays 2\textsuperscript{nd} price or floor
The second price auction mechanism

User visits site

Site won’t sell below floor

Advertisers bid

Impression lost or sold on other channel
Impressions sold as a uniform good
Impressions sold as a uniform good
Impressions sold one by one (i.e. in real-time)
The dynamic impression pricing problem
How to maximize the publisher revenue?

For each period $t$:

- the seller receives aggregate revenue data over the last period $c_t$,
- it can assign a reserve price $r_{t,a,g}$ for each buyer $\times$ tag ($r_t = (r_{t,a,g})$).

The task of the seller is to find the optimal reserve price policy:

\[
\max_{r_1,\ldots,r_T} \sum_{t=1}^{T} c_t(r_t)
\]

We assume that the revenues are independent draws from a gaussian random variable:

\[
c_t(r_t) = f(r_t) + \epsilon_t
\]
A Metropolis-Hasting approach

- An approach similar to simulated annealing is used
- Since the revenue function is non-negative no exponential
- We use an isotropic Gaussian perturbation

\[ r_{t+1}^\ast = r_t + \epsilon_t \]

\[ P (r_{t+1} = r_t^\ast) = \min \left( \left( \frac{c(r_{t+1}^\ast)}{c(r_t)} \right)^{\frac{1}{\theta}}, 1 \right) \]
A Gaussian process modelling + UCB

• We update a distribution in a functional space using bayes rule
• It can be done analytically using Gaussian processes and the kernel trick

\[ p(f) = \mathcal{N}(f|f_0, K_0) \]

\[ p(f(z)|x, \sigma^2) = \mathcal{N}(f|f_N(z), K_N(z, z)) \]

\[ f_N(z) = f_0(z) + K_0(z, x)(\sigma^2 I + K_0(x, x))^{-1}(y - f_0(x)) \]

\[ K_N(z, z) = K_0(z, z) - K_0(z, x)(\sigma^2 I + K_0(x, x))^{-1}K_0(x, z) \]
Figure 2.2: Panel (a) shows three functions drawn at random from a GP prior; the dots indicate values of \( y \) actually generated; the two other functions have (less correctly) been drawn as lines by joining a large number of evaluated points. Panel (b) shows three random functions drawn from the posterior, i.e. the prior conditioned on the five noise free observations indicated. In both plots the shaded area represents the pointwise mean plus and minus two times the standard deviation for each input value (corresponding to the 95% confidence region), for the prior and posterior respectively.

For eq. (2.16) the characteristic length-scale is around one unit. By replacing \( |x_p - x_q| \) by \( |x_p|/\lambda \) in eq. (2.16) for some positive constant \( \lambda \) we could change the characteristic length-scale of the process. Also, the overall variance of the magnitude random function can be controlled by a positive pre-factor before the exp in eq. (2.16). We will discuss more about how such factors affect the predictions in section 2.3, and say more about how to set such scale parameters in chapter 5.

Prediction with Noise-free Observations

We are usually not primarily interested in drawing random functions from the prior, but want to incorporate the knowledge that the training data provides about the function. Initially, we will consider the simple special case where the observations are noise free, that is we know \{ \( (x_i, f_i) \mid i = 1, ..., n \} \}. The joint prior distribution of the training outputs, \( f \), and the test outputs \( f^\ast \) according to the prior is

\[
\begin{bmatrix} f \\ f^\ast \end{bmatrix} \sim \mathcal{N} \left( \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} K(X, X) & K(X, X^\ast) \\ K(X^\ast, X) & K(X^\ast, X^\ast) \end{bmatrix} \right)
\]

(2.18)

If there are \( n \) training points and \( n^\ast \) test points then \( K(X, X^\ast) \) denotes the \( n \times n^\ast \) matrix of the covariances evaluated at all pairs of training and test points, and similarly for the other entries \( K(X, X), K(X^\ast, X^\ast) \) and \( K(X^\ast, X^\ast) \). To get the posterior distribution over functions we need to restrict this joint prior distribution to contain only those functions which agree with the observed data points. Graphically in Figure 2.2 you may think of generating functions from the prior, and rejecting the ones that disagree with the observations, algorithmic rejection
A Gaussian process modelling + UCB

- We use an isotropic Gaussian kernel as correlation structure

\[ K_0(z, x) = (K(z, x_1), \cdots, K(z, x_n)) \]

\[ K_0(x, x) = \begin{pmatrix} K_0(x_1, x_1) & \cdots & K_0(x_1, x_n) \\ \vdots & & \vdots \\ K_0(x_n, x_1) & \cdots & K_0(x_n, x_n) \end{pmatrix} \]

- The representation of the candidate function grows as O(n)
A Gaussian process modelling + UCB

Acquisition function: UCB

\[ \arg\max_{r_{t+1}} f_t(r_{t+1}) + \kappa \sqrt{K_t(r_{t+1}, r_{t+1})} \]

2 limit cases

- Kappa small: growing hyperplane search
- Kappa large: systematic coverage of the full domain


A Gaussian process modelling + UCB

Figure 5: Examples of acquisition functions and their settings. The GP posterior is shown at top. The other images show the acquisition functions for that GP. From the top: probability of improvement (Eqn (2)), expected improvement (Eqn (4)) and upper confidence bound (Eqn (5)). The maximum of each function is shown with a triangle marker.

Like other parameterized acquisition models we have seen, the parameter \( \alpha \) is left to the user. However, an alternative acquisition function has been proposed by Srinivas et al. [2010]. Casting the Bayesian optimization problem as a multi-armed bandit, the acquisition is the instantaneous regret function

\[
 r(x) = f(x^?) - f(x)
\]

The goal of optimizing in the framework is to find:

\[
 \min_{T} \sum_{x} r(x) = \max_{T} \sum_{x} f(x)
\]

where \( T \) is the number of iterations the optimization is to be run for.

Using the upper confidence bound selection criterion with \( \alpha_t = \sqrt{2 \tau} \) and the hyperparameter \( \tau > 0 \) Srinivas et al. define \( \text{GP-UCB}(x) = \mu(x) + \sqrt{2 \tau} \).
Kappa small: exploit
Kappa small: exploit
Kappa small: exploit

**UCB with small kappa**
- Little incentive to explore

**First line then hyperplane search**
- Start with one measurement (positive)
- Randomize direction
- Search greedily along line (symetry)
- Randomize direction
- Search in hyperplane generated by evaluation points
- Etc. Then stops on local maximum
Kappa large: explore
Kappa large: explore

Optimized function

- Posterior
- Acquisition
- Actual

Optimized function

- Posterior
- Acquisition
- Actual
Kappa large: explore

UCB with large kappa

• Little incentive to exploit

The algorithm will systematically cover the domain

• Finer and finer grained grid
Kappa medium
Kappa medium

Optimized function

-4 -3 -2 -1 0 1 2 3 4

0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0

-0.1

Optimized function

-4 -3 -2 -1 0 1 2 3 4

0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0

-0.1

Posterior
Acquisition
Actual
Works well

- Explore then end-up exploiting

**How to tune the parameters?**
Comparison on a simple case
Comparison on a simple case
Comparison on a simple case

50\textsuperscript{th} iteration distribution

Average over all iterations
Thank you

REAL TIME PREDICTION HAS A NAME

contact@alephd.com