Optimization and Pricing of RTB Algorithms

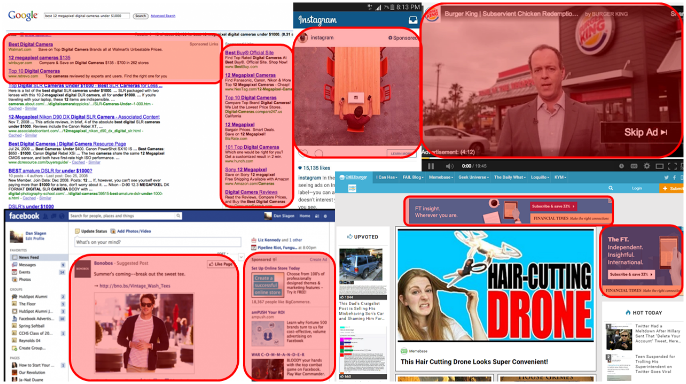
The Programmatic Company

Joaquin Fernandez-Tapia

June 15, 2016

Real-Time Bidding

When users consume online services, the available slots to push advertising are auctioned in real-time. This is called **R**eal-**T**ime **B**idding, or RTB.


Google
Best Digital Camera
12 megapixel cameras $135
Top 10 Digital Cameras
Instagram
Sponsored
Burger King | Subservient Chicken Redemptio...
Skip Ad
facebook
Bonobos - Suggested Post
Summer's coming--break out the sweet tee.
FT insight. Wherever you are.
UPVOTED
HAIR-CUTTING DRONE
This Hair Cutting Drone Looks Super Convenient!
The FT. Independent. Insightful. International.
HOT TODAY

RTB permit a better targeting, leveraging contextual data and algorithmic capabilities in order to optimize a **K**ey **P**erformance **I**ndicator, or KPI.

RTB Algorithms

- An RTB algorithm participates to a stream of auctions. For each auction, it responds a **bid** consistent with its optimization goals.
- The mechanics of RTB auctions is known as second-price auction: the best bidder wins and he pays the price of second-best bid.
- The goal of the algorithm is, under a limited budget, to optimize a KPI which is only known at the end of the ad-buying period.

Approach

Mathematics is a lost art so I will use code notation to explain the ideas.

States

At any time, the state of the algorithm is represented by a key-value store with keys `nbAuct`, `spent`, `imps`, `convs`, such as:

- `nbAuct` counts the number of auctions up to the current time
- `spent` is the current spent
- `imps` is the current number of impressions obtained
- `clicks` is the current number of clicks obtained

Auctions

We suppose the existence of a function called `getNewPriceToBeat()` which returns, for each auction, **the best bid among the other players** (hence the price to beat and the price we pay if we win).

We also suppose the existence of a function called `userClicks()` which returns `True`, for each auction such that **we win the auction and the user clicked in our banner**, and `False` for any other case.

Transitions

Right after an auction, the transition function takes the **current state of the algorithm** and a value for the **bid**, and depending on the auction result **it returns the new state of the algorithm**.

```
def Transition(state,bid):
    priceToBeat = getNewPriceToBeat()
    newState = state
    if bid > priceToBeat:
        newState[spent] = state[spent] + priceToBeat
        newState[imps] = state[imps] + 1
        if userClicks():
            newState[clicks] = state[clicks] + 1
    newState[nbAuct] = state[nbAuct] + 1
    return newState
```

Bidding Strategies

A bidding strategy is a function which takes as input a **state** and returns the **bid** the algorithm stands ready to bid for the next auction.

The outcome for an execution of a given bidding strategy is the value of the KPI at the end of the period (a function of the final state).

```
def GetOneOutcome(SomeBiddingStrategy):
    state = {nbAuct:0,spent:0,imps:0,convs:0}
    while AreMoreAuctions():
        bid = SomeBiddingStrategy(state)
        state = Transition(state,bid)
    return ComputeKPI(state)
```

The Goal

Our goal is to find the best bidding strategy.

?!

But the outcomes are random... How we define **The best** bidding strategy?

Statistically!

Which in ‘code’ means **we want the strategy yielding the best results after a large number of trials.** Thus, we define:

```
def StrategyAverageScore(BiddingStrategy):
    cumulateScore = 0
    for i in range(NBOFTRIALS):
        cumulateScore += GetOneOutcome(BiddingStrategy)
    return cumulateScore/NBOFTRIALS
```

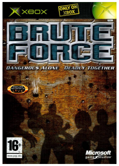
The Goal (in code)

We want to build a function `BestBiddingStrategy` corresponding to the input maximizing the output of the function `StrategyAverageScore`.

Possible solutions

- Option #1: Test ALL the possible combinations `state->bid` and choose the map which yields a higher `StrategyAverageScore`.

- Option #2: Choose a parametric form for the `BiddingStrategy` function and train it with real-life trials (reinforcement learning).



- Option #3: Choose a parametric form for the randomness then use the dynamic programming principle. This allows a deeper analysis and the possibility to extend the framework to pricing problems.

The Dynamic Programming principle

Idea (a sort of back-propagation procedure):

- Assign to each intermediary state a score representing **the best average KPI we can get at the end for a strategy visiting that state**. This is called the **value function**.
- For the final states (last auction) the value function is fully known (there is no more randomness concerning the KPI).
- For each intermediary state, each different possible bid will define different probabilities for the next transition.
- The best bid for that state is the bid yielding the probability maximizing the average of the value function one step ahead.
- The value function for the current state is the average of the value function for the next step ahead using the optimal bid.

The Dynamic Programming principle

The Dynamic Programming principle allows:

- Finding the optimal bidding strategy regardless of the randomness.
- Obtaining runtime estimates of the final KPI (the value function).
- Dramatically reduces the complexity compared to other approaches.
- Online learning using Bayesian analysis (forward propagation)
- It is code-friendly, it allows mathematical and economic analysis...

Pricing Performance-Based Contracts

The optimization framework allows to treat the pricing problem, i.e. **giving a target KPI, which is the fair price to pay for the algorithm?**.

The main difference between *optimization* and *pricing* is:

- **Optimization:** optimize KPI with fixed spent.
- **Pricing:** minimize spent with fixed KPI.

Mathematically the problems are equivalent.

Pricing Guaranteed Performance

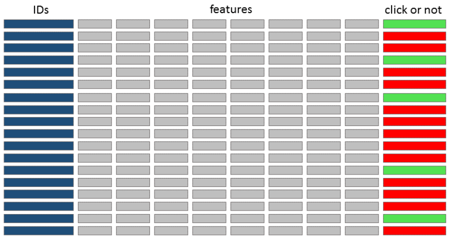
Conceptually however the problems are quite different:

- **Optimization:** Our focus is to find the bidding strategy.
- **Pricing:** Our focus is the bidding strategy **AND** the value function (which quantifies the fair price) **AND** the probability of the algorithm not delivering the target KPI (i.e. a notion of financial risk).

The strategy and the value function are found with the same techniques as in the optimization problem. The risk analysis is accomplished using repeated simulations of strategies (a.k.a. Monte-Carlo simulation).

Click-Through-Rate Prediction

A classic problem in RTB is click-through-rate (CTR) prediction.



Advantage: decisions based on data at a user-level using a machine learning procedure (free from the 'homogeneous-segment' assumption)

Disadvantage: it does not take in account the auction mechanics aspects. It does not provide solutions for dynamic optimization and pricing.

Optimization + CTR Prediction

Ways to merge these two approaches:

- **Offline:** Learning using user data, segment audiences in terms of their CTR probability, an optimization algorithm per audience.
- **Online:** Re-write the problem considering a ‘distribution of CTR probabilities’ evolving using the data gathered during runtime.

This is still an open problem.

More Open Problems

- Beyond clicks: Optimization with delayed conversions.
- Pricing with learning: Pricing under the learning framework.
- Sensitivity analysis: Variation on macro outputs (price, risk, ...) with respect to input parameters (auction rate, CTR, ...).
- Dynamic parameters, stochastic parameters, change point detection...
- Optimization with estimation in real time (Bayesian updates).

For more information: **Optimal Real Time Bidding Strategies**, Fernandez-Tapia, Guéant, Lasry (2016).

The End