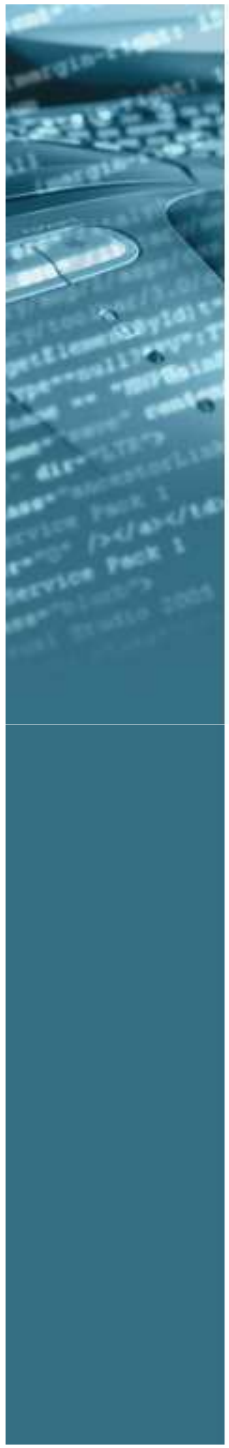


CAROUSEL

Distinctive features

noesis

www.noesis-research.com

Carousel in a nutshell

The simplest problem in the world

Game rules:

- you are paid for clicks
- you have 1 slot and 2 ads
- you have to perform 10,000 impressions

Too simple to be realistic, but useful for discussion.

The naïve solution

First statement:

maximizing revenue means choosing the better ad as quickly as possible, then showing it for the remaining impressions.

A popular strategy, with some hidden traps.

It is natural to split the campaign in two phases:

1. *Exploration*: assign 1,000 impressions to each ad
2. *Exploitation*: assigning the remaining 8,000 imps to the winner in phase 1.

. . .

When is exploration enough?

If you consider the problem not too naïvely, problems arise.

Why 1,000 imps for ad? Why not 500 or 2,000?

If the exploration phase is too short, you risk to fail in discovering the better ad.

If it is too long, you uselessly waste imps, then clicks, then money.

There is a trade-off:

exploration-exploitation dilemma

If the world changes

Another problem: the worst ad can become the better during the exploitation phase.

Maybe it is a matter of lifetime cycle, maybe simply of time of the day or day of the week, maybe the traffic is changing and moving to other visitors segments.

Normal solution: sometimes repeat the exploration phase.

Again, when do we have to repeat, and how long is each exploration phase?

Terrible problem: new ads enter the system …

Exploration vs. exploitation

The problem is to balance exploration and exploitation.

You need parameter tuning in lack-of-knowledge conditions.

When exploring, you pay opportunity costs, because you could choose an ad with a better CTR.

When exploiting, you pay opportunity costs, because you could miss to discover that another ad really has a better CTR.

No simple solution

The point is that you don’t have a simple fixed rule to apply in advance.

The right time for exploration and the right length of exploration are impossible to plan: they must be discovered during operations.

Optimizing clicks is ***real time online learning*** task.

It is real time because you have to decide in milliseconds.

It is online because you see data one at a time.

Scientists know it is extremely challenging.

. . .

For statisticians only

Yes, you can.

You can fix a confidence level, sampling until you reach the required confidence that ad is the best one, then stop exploration and start exploitation.

Many patents implement this strategy.

Yet, it does not scale to large real problems, and is difficult to implement, especially when the world changes (and the real world does).

Moreover, fixing the confidence level is a not trivial parameter tuning task.

It is a portfolio problem

Classical solutions, both simplistic and sophisticated ones, generally miss a key point:

*this is an **economic** problem, more specifically an **investment** problem.*

You have to build a portfolio of options coping with an uncertain world

- uncertain because you don't see real values
- uncertain because values change

Yes, this sounds excessive for one slot and two ads toy-world, but when you move to real world…

. . .

Knowledge is money

Assign the next impression to the best-until-now ad sounds reasonable. Indeed, many systems implement this strategy. But is far from optimal in practice.

It looses significant amounts of revenue.

Sometimes, it causes disasters.

Assigning the next impression to a non-leader ad might be a good idea, because it gains knowledge about the world.

This new knowledge can bring money allowing better future decisions. ***Maybe*** …

Carousel's way

Carousel is a real time online learning system based on microeconomic principles.

It decides investments in knowledge acquisition by means of apparently sub-optimal moves, maximizing the long-term expected monetary values in an uncertain ever changing world.

Its strategy evolves continuously in an endless effort to scout out potentially raising stars while exploiting already known cash-cows.

Moving to real world

*The simplest problem in the **real** world*

Game rules:

- you are paid for clicks
- you have 1 slot and 5 ads
- you have to perform a two-week campaign

Too simple to challenge Carousel, but realistic (indeed, real).

A case-study

A real problem for Lycos.

Despite its simplicity, it is complex enough to put the naïve approach in troubles.

Ads CTRs oscillates during the day (and the week) and have different trends.

The initial leader ad declined more quickly than others, and two leadership changes were observed in the long term trend, besides several surpasses in the short term (e.g. in the night).

A case-study (continued)

After two weeks, Carousel obtained a total performance of 99.53% of the best ad.

These means that if you could have known in advance which ad would be the best, you would have obtained nearly the same revenue!

The 0.47% difference is the cost of exploration.

Strategy for an ad network

- plan A: to get a wizard crystal ball and guess the future.
- plan B: to get Carousel.

A case-study (continued)

How counterintuitive it could be, at certain moments Carousel had a performance higher than 100% that of the best ad.

No non-sense. Imagine ad Blue is the best in the first week, and ad Red in the second.

Carousel exploits Blue, then quickly discover the change of leadership, then exploits Red. It picks the best from both, and the campaign beats every ad of its.

Carousel can beat the crystal ball, after all.

Scaling

Many slots, many ads

Now imagine to have 1,000 slots and 10,000 ads.

The naïve solution quickly becomes out-of-question.

If you sample ads uniformly, you loose too much money in exploration (bad investment).

You can also see some ads disappear before you have finished to evaluate them!

Intuition falls short in such contexts.

For example, not to evaluate many ads could be an effective strategy, though strange.

Time segmentation

Now imagine to have 1,000 slots, 10,000 ads and 4 time bands (morning, afternoon, evening, night).

Time matters. Imagine you have a banner of a bank offering loans and a banner of a girl offering chats. Try to guess which has the better CTR in the office-time and in night-time … (it is a real case).

Time segmentation is really challenging for classical solutions. If you don't manage to learn continuously, you leave a lot of money on the table. If you learn but in a not optimal way, you pay a price anyway.

Time exalts the exp-exp dilemma.

Sample fragmentation

One can think to overcome the time segmentation problem simply running a different strategy for each different time band.

The approach is correct in principle but too slow to learn in practice. You would have, say, 4 optimization processes running separately using different samples.

That is too data-demanding and learning happens over too long a time, due to lack of knowledge generalization.

The key point is that you need ***exploiting exploration*** in a more sophisticated way.

User segmentation

Now imagine to have 1,000 slots, 10,000 ads, 4 time bands and 200 user segments.

You are in trouble.

Two hundred segments are realistic. It is enough to have a 2-valued attribute Gender, a 5-valued Age and a 20-valued Region. Only 3 attributes in the user database. It is easy to have even more segments.

If you aggregate segments, you are wasting your database. Otherwise, you cope with a terrible sampling problem.

Behavioral targeting

Demographic profiles are enough. Behavioral profiles are much worse.

Behavioral profiling can generate an unlimited number of profiles. It is impossible to sample each possible profile with an adequate number of cases, unless wasting a lot of impressions in exploration.

We must have a way to learn about a huge variety of profiles using a limited number of empirical data.

Contextual targeting

You can also want to exploit page content for choosing the right ad to be shown in a slot.

Like behavioral targeting, contextual targeting can generate an unlimited number of profiles. Again, we need somewhat able to generalize knowledge from observed situations to unobserved ones.

A complex example

News recommendation on Repubblica.it

The second Italian newspaper. On Web 400 million page views per month.

Problem: for each page containing an article, recommend linked articles.

For Carousel this is similar to ad recommendation.

It is equipped with a behavioral and a contextual engine.

Behavioral targeting on Repubblica.it

The number of user profiles is unlimited: no fixed segments, but 65 numbers for user, representing time spent on topics

Sport 12%, Politics 8 %, …

Extremely appreciated by the customer:

self-evolving segmentation and targeting

The privacy authority ordered to suspend the BT for a while, in order to ensure correct service.

Carousel continued its work without BT.

Carousel ***does*** perform BT, it ***is not*** a BT system.

Contextual targeting on Repubblica.it

The number of page profiles is unlimited: no fixed topic classification, but a complex internal representation (proprietary text mining technology):

self-evolving contextualization and targeting

We experimented suppressing text mining.

Carousel continued its work without CT.

Carousel ***does*** perform CT, it ***is not*** a CT system.

Folksonomies on Repubblica.it

Carousel is able to handle tags labeling articles.

Proprietary tag mining technology.

This is a precious alternative to text based targeting.

Experimented on Kataweb.it, another site of Repubblica, planning to build folksonomies.

The tags can equally well label users.

Or ads.

Again:

self-evolving tag targeting

Again, Carousel can work without it.

What's the performance?

Carousel is giving a 7% CTR on articles.

When a user reads an article, Carousel suggests about 10 articles, and 7 times on 100 the user follows a suggested link.

This performance is considered extremely good, much greater than expectation, much greater than other technologies proven in Republica.it.

Carousel even with no side-information (no behavioral, no contextual) beats by far the previous contextual targeting in use.

Exploiting exploration

Value= money + knowledge

Carousel sees revenue as the ultimate goal, knowledge as an important intermediate one (because knowledge will become money).

When exploring, Carousel makes an investment.

Its algorithms maximize the sum of expected monetary value and expected knowledge value.

Sampling as investment

In the evening a visitor lands on page containing slot #123. The segment is

Gender = female, Age = 30-40, Region = Italy

Carousel sees this ***situation*** (in its jargon):

<123, evening, female, 30-40, Italy>

It has to select an ad.

Even if ad A is more promising of ad B ***in that situation*** (greater CTR, greater bid), nevertheless B might be the best one in the Carousel's philosophy, because it is useful to learn about its performance in that situation.

Sampling as investment (continued)

In order to understand whether A or B is the better, Carousel performs an investment evaluation.

Ad A has CTR = 2% and bid = $1.00, so it has expected value 2 cents.

Ad B has CTR = 1% and bid = $1.5, so the expected value is 1.5 cents.

The balance is:

half-a-cent (opportunity cost, investment in knowledge) vs. one more case in the sample of B in that situation (knowledge gain).

Knowledge generalization

Carousel evaluates its own knowledge about relationships among situations and ads.

So it evaluate investment in knowledge acquisition.

Key point: Carousel evaluate how much new knowledge it can gain about that ad in that situation, but at the same time how much new knowledge is gained about ***all*** situations, not only the present one.

This is a deep issue. Knowledge generalization is not only a scientific pearl. It is the key feature for effective ***targeting***.

Similarity search

Carousel is equipped with a Noesis' proprietary technology for ***similarity search***, developed in applications far from the advertising world.

This technology allows Carousel to perform reasoning like this:

The ad A has got a click for the last impressions on page P for user of segment S. This is good news for A w.r.t. P (of course), but also for A in itself, for P in itself, for A w.r.t. to each page P' similar to P, for A' similar to A w.r.t. P or P'.

So Carousel is able to exploit knowledge and exploration in a very effective way, giving a decisive edge over other technologies too data-demanding.

The game Carousel plays

The goal of the game

Carousel continuously learns how to maximize value, intended as the *overall revenue of all campaigns in the long term*.

It plays an ever lasting game. Each move brings money and knowledge, which in turn will bring money.

The classical model is:

expected value = ctr × revenue

revenue = bid – ask

The Carousel's model is:

expected value = ctr × revenue + future added value

The core game

Carousel learns statistical correlations among situations (page, segment, context, time …) and actions (showing ads, news, videos, products, prices …).

Proprietary algorithms come from Artificial Intelligence.

Mixing correlations learnt with economics (bid, ask, budget, goal) it maximizes value (as defined in Carousel).

Other games

Carousel was explicitly designed to learn from scratch, without any auxiliary source of information.

Hence, it is ***not***

- a behavioral targeting engine
- a contextual targeting engine
- a collaborative filtering engine
- anything in this family

But it really performs behavioral, contextual and collaborative.

There is no contradiction

Autonomic data mining

Carousel performs a sort of implicit, proactive data mining, which we call autonomic to underline the absence of human intervention.

You can view Carousel moves as:

- experiments of an automated scientist
- surveys of an automatic marketer
- investments of an automatic trader

all in real time on a micro-scale (to the single impressions if required).

The value of the game

Lift in simple campaigns

Leonardo.it, large Italian entertainment site.

Lift: 200% to 400%.

Carousel CTR is three to five times the previous one, obtained with simple uniform rotation.

Without any additional resource (no user profile, no page context, no ad classification).

Small campaigns (6 to 15 alternative ads).

The clickthrough rates were rather low, below 0.5%.

Lift in simple campaigns (continued)

These conditions are not favourable for Carousel, which cannot show its full power.

The benefits of any optimizer become more evident when there are many alternative ads.

That is enough for Carousel vs. Random.

What about Carousel vs. smart algorithms?

Carousel vs. Google

Carousel has been tested on a Google Adwords campaign (based on search, not content).

While the campaign was regularly running on Adwords, very simple data were taken from Google Analytics (impressions and clicks for each creative during the campaign's progress) and given to Carousel.

A parallel campaign was driven setting on at each moment the Carousel recommended ad, off the others.

Carousel vs. Google (continued)

It turned out that Carousel is systematically faster than Google Optimizer in finding the best creatives.

A small and simple campaign contained 3 ads, with clickthrough of 0.39%, 0.22% and 0.11% after about 150,000 impressions.

Google obtained a clickthrough of 0.33% while Carousel reached 0.37%, which is very close to a “crystal ball” performance.

This was realized using Carousel by hand, so not in true real time.

Experiments with synthetic data

The trial vs. Google is interesting, but not scientifically strong.

To be rigorous we need repeated experiments on synthetic dataset. Only in this way we can draw scientifically robust evidence.

We used datasets generated with a model adopted by Yahoo! researchers for exactly the same reasons.

The model give ads CTR realistic distributions.

Carousel vs. Random

Experiments confirmed evidence gained in the real life case (leonardo.it). In the Yahoo! model a lift of 3-5 times against ad random delivery is normal.

The specific lift depends on the specific ads CTR distribution.

At one extreme, you have a flat distribution: there is no room for optimization of any sort. At the other extreme you have a narrow peak: you are searching for a needle in the haystack.

More complex the problem, higher the Carousel's lift.

Carousel vs. Naïve algorithm

The popular Naïve algorithm is:

- sample all ads uniformly
- pick the winner
- keep it
- after a while, restart

Not used in experiments: in practice it is difficult simply to reasonably choose when and how long to explore.

Carousel vs. Greedy algorithm

The second industry standard algorithm is *Greedy*:

- select the ad best performing until now

That's all. Not so trivial as it could appear. Better then the naïve one.

Experiments with 200 ads give a Carousel's lift of 80% in basic version with no additional information.

Moreover, the Carousel performance is much more stable, so it is much more reliable for real use, where psychology matters.

The needle in the haystack

We experimented a campaign with 100 ads.

The best ad has a CTR of 1%, the second one of 0.9%, the third one of 0.8%, all the others of 0.5%.

The campaign lasts 500.000 impressions.

Random rotation: CTR = 0.5%.

Greedy: CTR = 0.83% (with high volatility).

Carousel: CTR = 0.94%, (the average between the best and the second ads).

Standard algorithms loose information

Beyond performance figures, standard algorithms like Naïve or Greedy have a major drawback:

it is really difficult to use them together with additional information like behavioral or contextual.

In practice, when used in real complex contexts they loose available, costly and precious information.

They are not feasible solutions for a serious business implementation.

Carousel vs. scientific literature algorithms

We implemented many algorithms from the scientific literature, applying them to the ad recommendation problem.

Carousel outperforms all of them.

Figures are reserved.

Yet, the real point is not performance data. It is that Carousel is designed to work with and without segments, behavior and contexts, which is a decisive advantage in practice.

Scaling algorithms

Scalability

Carousel algorithms scale well to large or very large systems.

Carousel performs two functions:

- Learning
- Selection

Learning is O(n) i.e. linear time, selection is O(1) i.e. constant time.

Scalability (continued)

Selection of the ad to be delivered is done in strict real time (milliseconds). It is constant time, so it is scale insensitive.

Learning delivering rules is done at intervals (seconds to minutes to hours, depending on cases). It is linear time, so only really huge applications could cause problems.

Moreover, Carousel uses techniques for:

- task factorization with assignment to many servers;
- data aggregation reducing dimensionality.

Two layers architecture

A back-end layer performs learning.

A front-end layer performs selection.

Both can be split on many servers.

The back-end hardly requires splitting.

The front-end was split onto 3 servers for Repubblica without any problem.

Both layer can reside on a same server (Lycos case).

Learning is fast

Test on a 2 processor Quadcore 2.33 Mhz 8 Gb RAM.

With 60,000 slots and 30,000 ads the learning task requires about an hour in the worst case.

The worst case is when each slot can host each ad, without any kind of filtering and preference.

If each slot can host 3,000 ads, the time reduces proportionally to 6 minutes.

Selection is fast

Test on a 1 processor Quadcore 2.33 Mhz 2 Gb RAM.

The selection module serves 12,000 requests per second.

Carousel as ad server

Interacting with browsers

Carousel can interact with browsers (with scripts), operating as a true ad server.

Both learning and selection are performed on Noesis' servers.

Carousel as web service

Interacting with customer servers

Carousel can interact with customer servers operating as a web service.

Noesis' servers perform the learning task.

Then Carousel sends the customer a collection of selection rules.

The customer use them to delivery ads.

Then the customer sends Carousel statistics about performance obtained.

This version is for advanced users willing to maintain the traffic management in their servers.

Using the web service

Carousel offers two main service:

- Get_Rules
- Set_Feedback

There are also several services for configuration purposes.

Periodically (seconds, minutes, hours) the customer calls the two main services.

Getting rules

The Get_Rules service sends the customer a collection of rules like

if time = evening and segment = >female, 30-40, Italy> and page = 2,381

then show ad #239 with probability 7%, ad 708 with probability 5 % … and so on.

The customer uses these rules as guideline for delivery. Implementing the rules is really an easy task for the customer.

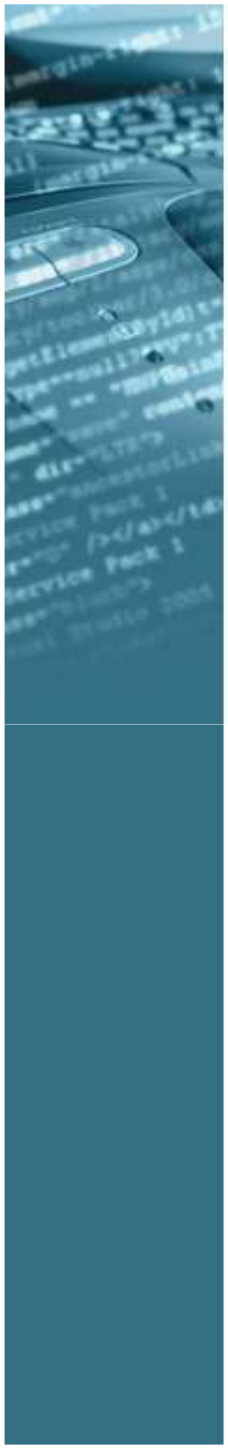
Setting feedback

The Get_Rules service receives from the customer a collection of statistics like

when time = evening and segment = >female, 30-40, Italy> and page = 2,381

Then ad #239 got 1587 impressions and 12 clicks, ad 708 got 1207 impressions and 9 clicks … and so on.

Carousel uses these figures as guideline for learning. Implementing the figures is really an easy task for the customer.



CAROUSEL

For more information, please contact Noesis:

Corso Italia 89
I56125 Pisa - Italy
+39 050 9911080
carousel@noesis-research.com
www.noesis-research.com

noesis

www.noesis-research.com

noesis-research