Applying Machine Learning to Build a Website Interface Adaptation System

Egor Mateshuk, Alexander Chernyshev

Moscow Institute of Physics and Technology
9, Institutskij, Dolgoprudny, Moscow region, Russia, 141700
e-mail: e.mateshuk@gmail.com

Abstract. In this article we present the architecture and model of a website interface optimization system. We describe how we use clustering and genetic algorithms to automatically select a website interface with the highest conversion from website visitor to website user. In particular, we describe an algorithm for streamed clustering, which allows for real-time analysis of high traffic website users.

Keywords: A/B-testing, machine learning, genetic algorithm, clustering.

1. Introduction

In the modern world almost every company of a significant size has its own website. A vast majority of these companies treats them as an instrument for attracting potential clients. However, there is no perfect and easy way to make the website achieve the fullest possible effect. Because of that, the task of optimizing the website is taken on by an employee or a whole division of the company. They handle not only the technical side of things, but also the work of increasing the conversion of site visitors into paying customers via website interface optimization.

To achieve that, manual A/B testing methods are often used (also known as split testing) [1].

Limitations: there is a constant need of involving developers to create versions, results must be analyzed manually, and data unique to a specific visitor is not taken into account to produce tailored optimizations. All of this results in a long process of optimization and low efficiency.

Compared to the previous approach, there is no need to constantly involve developers to generate site variant. The site manager can build new variants themselves using a visual website designer. However, the optimizing efficiency is still low due to the fact, that the website is being tailored to the “average” user, without taking individual variability into account.

2. Proposed method

The basis of the approach we are proposing is automating and improving the process of website optimization using machine learning algorithms.
The following stages will be automated:
1. Generating website variants for later testing.
2. Gathering information on website visitors.
3. Displaying the most relevant variant based on gathered data.
4. Collecting information on user behavior on the website for further system learning.

Our method makes the whole process easier while remove developers from the equation. The site variants are generated based on hypothesis that can be specified by the site manager himself in an online editor. This leads to savings in time and money required to improve conversion.

Additionally, a new avenue to improve conversion becomes available when using our system. In regular split-test at the end of each iteration you have a single site variant, which has the best average conversion result. However, this approach doesn’t take into account that the same variant can have different conversion for different groups of users. Selecting a single variant may increase the average conversion, but for smaller groups conversion might actually decrease. Our system allows for automatically selecting the optimal website with the highest conversion for every group of users, and not only the biggest one.

3. System Architecture

Figure 1 describes system architecture in general.

4. Generating website variants

Different website variants are described by different combinations of optimization parameters. One example of such parameters is the ordering of website elements or the contents of the website header. We will be using genetic algorithms [2] to generate these variants — this will allow us to avoid brute forcing all possible combinations.

Broadly speaking, a genetic algorithm works in the following way: a population is formed from the site variants, after which these variants are tested by presenting them to the site audience, after which old variants are pruned and new ones are generated. The process is then repeated. We need to figure out the following questions: how to prune the results and how to generate new ones.

Obviously, the fitness function in this case will be the conversion on a variant matching its “chromosome”. However, we need to bear in mind that variants with low conversion for the whole flow of website visitors might have higher conversion for some groups of users. Therefore, the fitness function should be not the average conversion for a vari-
ant, but, rather, the maximum conversion among groups of users (we will discuss determining the boundaries of these groups in the next section)

\[ f(j) = \max_i c_{ij}, \]

where \( c_{ij} \) is the conversion of users from class \( i \) for website variant \( j \), while \( f(j) \) is the fitness of variant \( j \).

![Figure 1. System architecture](image)

Generating new variants to add to the population is achieved via mutation (random changes in variant parameters) and crossover (taking more than one parent variant and producing a child variant from them). Here is how crossover is implemented:

Let \( A_1 \) be the first parent variant, \( B_1 \) — the second parent variant;
\[ A_1 = \{a_1, a_2, \ldots, a_n\}, \]
where \( a_1, a_2, \ldots, a_n \) is a combination of optimization parameters. Likewise \( B_1 = \{b_1, b_2, \ldots, b_n\} \).

Now position numbers from 1 to \( n \) are randomly selected. These positions receive parameters from the first parent variant, other positions receive parameters from the second parent variant. A new variant is \( C_1 = \{a_1, b_2, a_3, a_4, b_5, \ldots, b_n\} \).
Parent selection will be conducted via roulette-wheel selection (also known as fitness proportionate selection) [3]. In this method variants are selected via N “spins” of the roulette-wheel, where N is the size of the variant population. The wheel of the roulette contains a sector for each member of the population. The size of sector j is proportional to the probability of becoming a parent of the new variant. We will call that probability \( P(j) \) and calculate it using the following formula:

\[
P(j) = \frac{\max_i c_{ij}}{\sum_{j=1}^{N} \max_i c_{ij}}
\]

where \( N \) is the size of the variant population.

With this approach members of the variant population with higher fitness have a higher chance of being selected than those with a lower fitness.

The selection of variants for the final population will happen in two stages:

First, we select the variant with the highest conversion for each cluster. After that we select additional variants based on the fitness function value.

5. Website visitor analysis

To select optimal interface parameters we first need to assign users to a number of groups, and then analyze how these groups behave with different variants of the website. We will use clustering to find the boundaries of these groups.

Traditional clustering algorithms have difficulties with dealing with large amounts of data, which is critically important in our case. Due to that we are proposing to aggregate data beforehand using Kohonen self-organizing maps [4].

After that, a traditional clustering algorithm (such as k-means or PAM [5–8]) will be applied to the aggregate data.

The results of the work of our module are as follows:

1. Initial data is collected initialize the Kohonen map.
2. Initial clustering is done. All network nodes are assigned a cluster number.
3. When a new user arrives, the closest node is calculated. The number of the corresponding cluster is then passed on to the “Conversion optimizer” module.
4. User data is fed to the Kohonen map as input.
5. For each n visitors we re-cluster them using the PAM algorithm.
6. Steps 3–5 are repeated for each new website visitor.

As can be seen from the algorithm, the module passes information on the cluster corresponding to the user on the first looped step (step 3 overall). The computation of this step can also be easily parallelized. This can be achieved by parallelizing the calculation
of distance from the vectors to each network node, and then selecting the minimal distance. If the distance calculation is parallelized, the time complexity is $O(1)$. The complexity of minimal distance calculation is not higher than $O(n)$, where $n$ is the number of network nodes. Therefore, we have an $O(n)$ algorithm on step one. After that, all of the following processing is done in other modules of the system, which means that we can take our time with steps 4 and 5. This allows us to support a large number of websites with a large number of visitors.

6. Conversion optimization

Input data for this module is the result of work of two other modules — variant generator and website user analyzer. Consequently, the input data is an array of website variants and the cluster number that corresponds to a user.

The goal of the module is to select a variant that maximizes the probability of a conversion action occurring. To achieve this we build a graph of relations between clusters and site variants. Each graph edge will have a weight that equals the visitor conversion on the corresponding variant. After that the probabilities of selecting each variant are defined as follows:

$$P_{ij} = \frac{c_{ij}}{\sum_{k=1}^{N} c_{ik}},$$

where $P_{ij}$ is the probability of user in cluster $i$ selecting variant $j$, $c_{ij}$ is the conversion of users from cluster $i$ on site variant $j$, and $N$ is the total variant number. The higher the conversion of a specific variant, the higher is the probability of it being selected.

In the end all data for a single website will be contained in matrix of connections between clusters and variants.

We need to bear in mind that occasionally the variants will change as a result of the genetic algorithm working. We need to somehow inherit the conversion values to avoid re-learning them from scratch. In this case we will need to rely on parent variants conversion — we can take the average between the two corresponding values.

The final result produced by this module is getting the data on user’s behavior on the website and updating the variant weights accordingly.

7. Results

We have described a system that can be used to offer each user a personalized version of a specific website with the highest expected conversion based on data we know about the user. Many of the aspects that need to be defined to fully describe the system
are not talked about in this paper. However, the aim of this paper was to provide a broad overview of the system, and further details will be described in future papers.

References


Система повышения эффективности интерфейсов веб-сайтов с использованием методов машинного обучения

Е. О. Матешук, А. С. Чернышев
Московский физико-технический институт (государственный университет) 141700, Московская обл., Долгопрудный, Институтский пер., 9
e-mail: e.mateshuk@gmail.com

Аннотация. В этой статье представлена архитектура и модель для оптимизации интерфейса веб-сайтов. Описано использование кластеризации и генетических алгоритмов для автоматического выбора интерфейса веб-сайта с самой высокой конверсией посетителей сайта. В частности, изложен алгоритм кластеризации, который позволяет в режиме реального времени проводить анализ трафика пользователей веб-сайтов.

Ключевые слова: A/B-тесты, машинное обучение, генетические алгоритмы, кластеризация.

Литература


Авторы:
Егор Олегович Матешук — аспирант кафедры технологического предпринимательства, Московский физико-технический институт (государственный университет)
Александр Сергеевич Чернышев — аспирант кафедры технологического предпринимательства, Московский физико-технический институт (государственный университет)