Deep Learning for Understanding Consumer Histories

See how we're delivering better consumer experiences with recurrent neural networks.

Recurrent neural networks (RNNs), *one of the major classes of deep learning methods, offer us two benefits: (1) By processing raw consumer histories, RNNs provide accurate prediction models without requiring tedious feature engineering efforts. (2) Furthermore, we can interpret their reasoning when providing predictions for individual consumers.*

Model building is an ongoing process for us: We are constantly working on increasing the accuracy of our models to improve consumer experiences, often by incorporating new data sources. Likewise, understanding the reasoning underlying the predictions of our models is a growing concern for us, for monitoring purposes, as well as convincing our collaborators of the accuracy of our products.

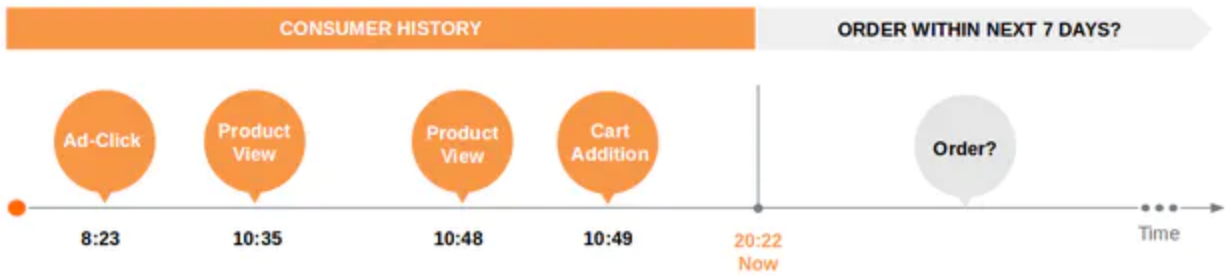
In this blog post, we will show that RNNs are helpful for both flexible model building and understanding model predictions. We thank our colleagues at Zalando Research for inspiring discussions and valuable feedback.

Predicting order propensities

There are a variety of goals that advertising campaigns are targeted at, like creating awareness or retargeting consumers. For example, the latter could involve estimating the propensity of consumers to order within several days. In this blog post we take this as our running example and focus on predicting order probability.

A side remark: Our models are based on histories of anonymous cookies. (That is, we **do not use** *customer data.) For ease of readability, we speak of consumers instead, but cookies are what we really refer to.*

The **consumer history** captures important clues that can be utilized for prediction: Has the consumer ordered recently? Did they visit Zalando just yesterday? Have they added something to their wishlist?



Customer histories are **sequences of events**. For each event, we know its type (product view, cart addition, order, etc.), its timestamp, and further information such as the viewed product or the fashion category the consumer is currently engaging with.

The central question when building predictive models is: **How do we get this sequential information into a machine learning model?**

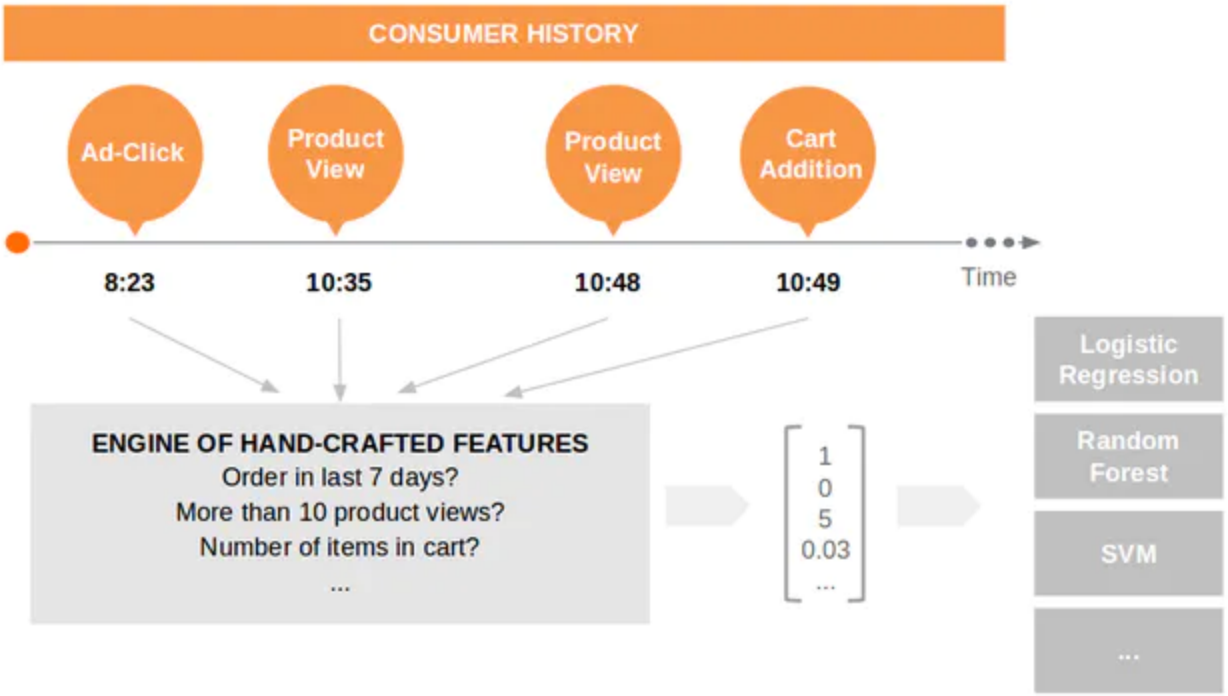
Approach 1: Traditional machine learning with handcrafted features

A traditional way to apply machine learning models on sequential data is **feature engineering**. Although feature engineering receives negligible attention in research papers, it can be the most critical and laborious task in a practitioner's daily work.

"At the end of the day, some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used." (Pedro Domingos)

We can conceive many sensible features for our prediction task at hand. For example: How many products did the consumer view yesterday? Did the consumer order within the last week? What's the current item count in the consumer's cart? And many, many more.

For a given set of features and a given consumer history, we calculate the corresponding feature vector. This is simply a large vector of numbers. The feature vector is provided as input to a **vector-based machine learning model**. There is a large zoo of machine learning models -- logistic regression, random forests, support-vector machines, you name it -- and almost all of them are vector-based. However, often it is not model choice, but the feature engineering process which has the greatest influence on prediction accuracy.



Feature engineering is an art in itself and requires domain knowledge, as well as data science intuition. Additional data preprocessing steps often have decisive effects on model performance. For instance, preprocessing may be required if we have real-valued, ordinal, and categorical features at the same time; if features have different value ranges; or if we want to ensure model robustness with model regularization. A standard approach in real-time bidding is using logistic regression on a large set of binary features. Binary features are created by binning and binarizing the original input features, a transformation that can be done in many different ways. For example, a feature for order counts could be converted into buckets for zero orders, one order, two-five orders, and six plus orders.

While the exact choice of feature representation has decisive effects on model performance, it can typically be determined only in experiments on historical data and in A/B tests. Choosing appropriate feature sets is time-consuming, tedious work.

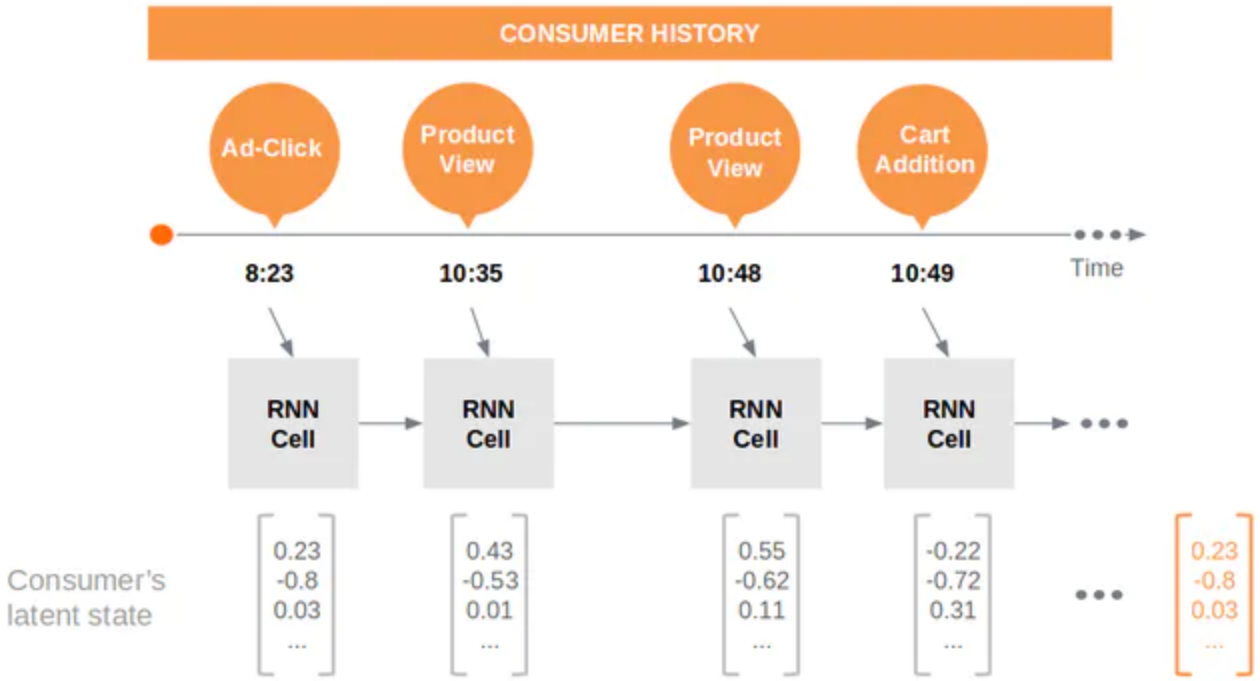
Tedious work which RNNs have the potential to circumvent.

Approach 2: Recurrent neural networks learn features

Most machine learning models are vector-based and require feature engineering to deal with sequences. In contrast, **recurrent neural networks** (RNNs) work directly on sequences as inputs.

RNNs make up one of the major classes of **deep learning** methods (the other one being convolutional neural networks (CNNs), mostly used for vision; for example, Zalando uses CNNs on product images to improve product recommendations and for logistics). It might be fair to say that research on deep learning has not been overly marginalized in recent years. Still, even with healthy hype-averse skepticism, its potential is great also in the ad-tech and e-commerce realm as we try to show in this blog (and as other recent work shows).

We feed consumer histories directly into RNNs. RNNs are made up of a sequence of computational cells. Each cell takes the input at a given time-step, in our case the event type and its timestamp (and potentially additional information like the brand of a viewed product). Cells maintain latent vectors of real-valued numbers. These numbers describe the consumer state until the respective time-step, as it is relevant for the prediction problem. The dimensionality of the latent vector is a design decision. In our example scenario, we use low-dimensional vectors to describe the consumer's propensity to order (typically, 10-15 dimensions). We use long short-term memory cells (LSTMs) (invented at TU Munich 20 years ago), which (together with descendants like GRU) underlie the recent success of RNNs.



The latent cell state at the last time-step (shown in orange) is used for the final prediction. This prediction can use a simple linear logistic regression layer or a sequence of non-linear layers as in a neural net.

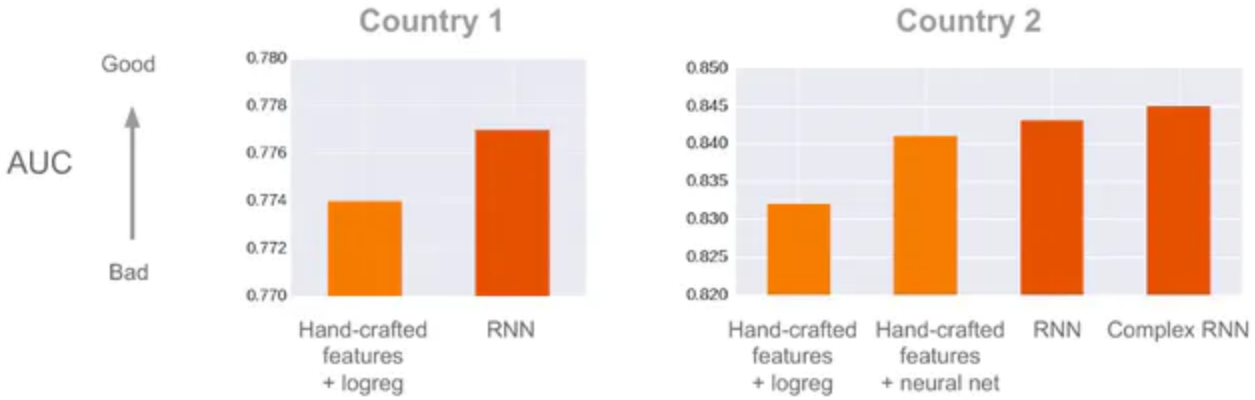
RNN cells contain a large number of parameters. These parameters are used in cascades of matrix multiplications to calculate latent cell states from inputs. During **training**, the parameters are adapted to find signals for prediction in the consumer history. This results in appropriate ways to calculate latent states: the latent states are prediction features that are **learned** from raw inputs. **No further feature engineering is required.**

But do RNNs work in practice?

To see whether RNNs live up to their promise, we performed experiments on historical consumer data from *Spring 2016* in two European countries, involving millions of Zalando visits. Data was split along time into training and test sets.

As a baseline, we use a logistic regression model built on a large number of fine-tuned handcrafted features, denoted *Handcrafted features + logreg*. These features were determined in ongoing labor-intensive feature engineering efforts. The model has been used in our production system previously.

We use a variety of metrics for benchmarking. Here, we present the results for the area under curve (AUC) metric on the test sets:



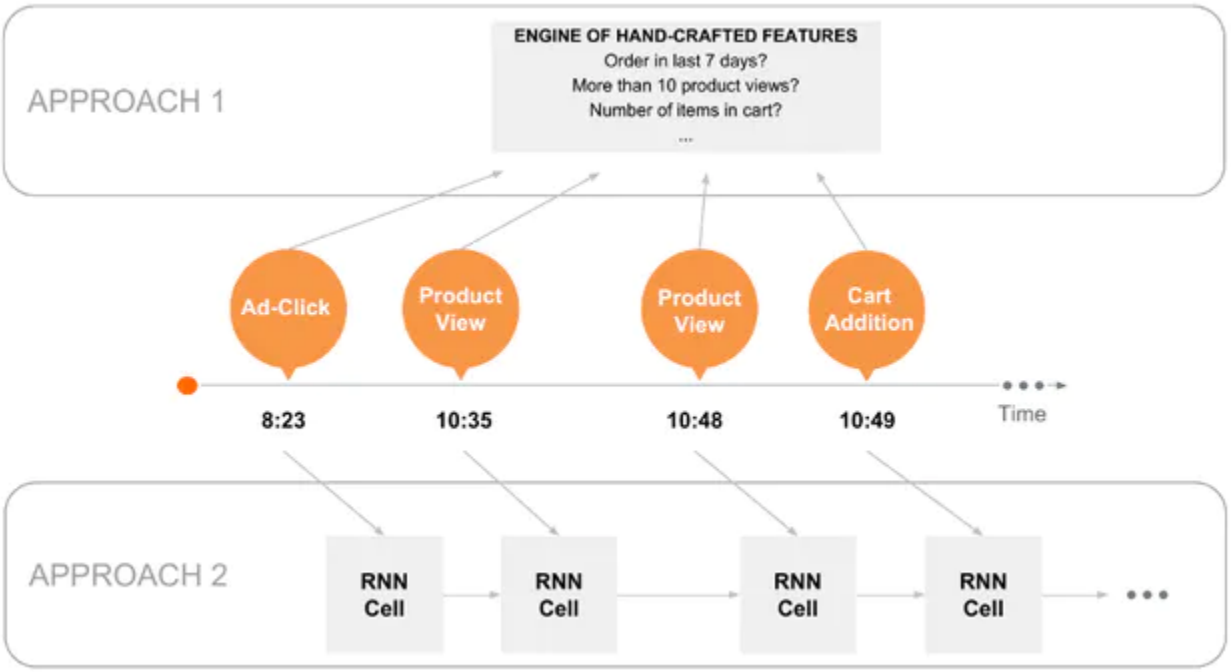
Random predictions have an AUC of 0.5. For *Country 2* (right diagram), we also present results with non-linear predictors: Both *Hand-crafted features + neural net* and *Complex RNN* use an additional hidden layer with rectified linear units (ReLUs) for prediction.

The results show that RNNs achieve about the same or better performance than the models relying on fine-tuned handcrafted features.

Comparing both approaches

Model tuning, training, and prediction is easier and faster with handcrafted features and vector-based machine learning methods like logistic regression. And it's no wonder, as most of the job was done beforehand by the human domain expert. This expert handcrafted informative features and thus, can take most of the credit for prediction accuracy.

In contrast, RNNs learn everything from scratch. Therefore, they require more training time and more involved model tuning. At the same time, it is machine learning that drives full model accuracy for RNNs.



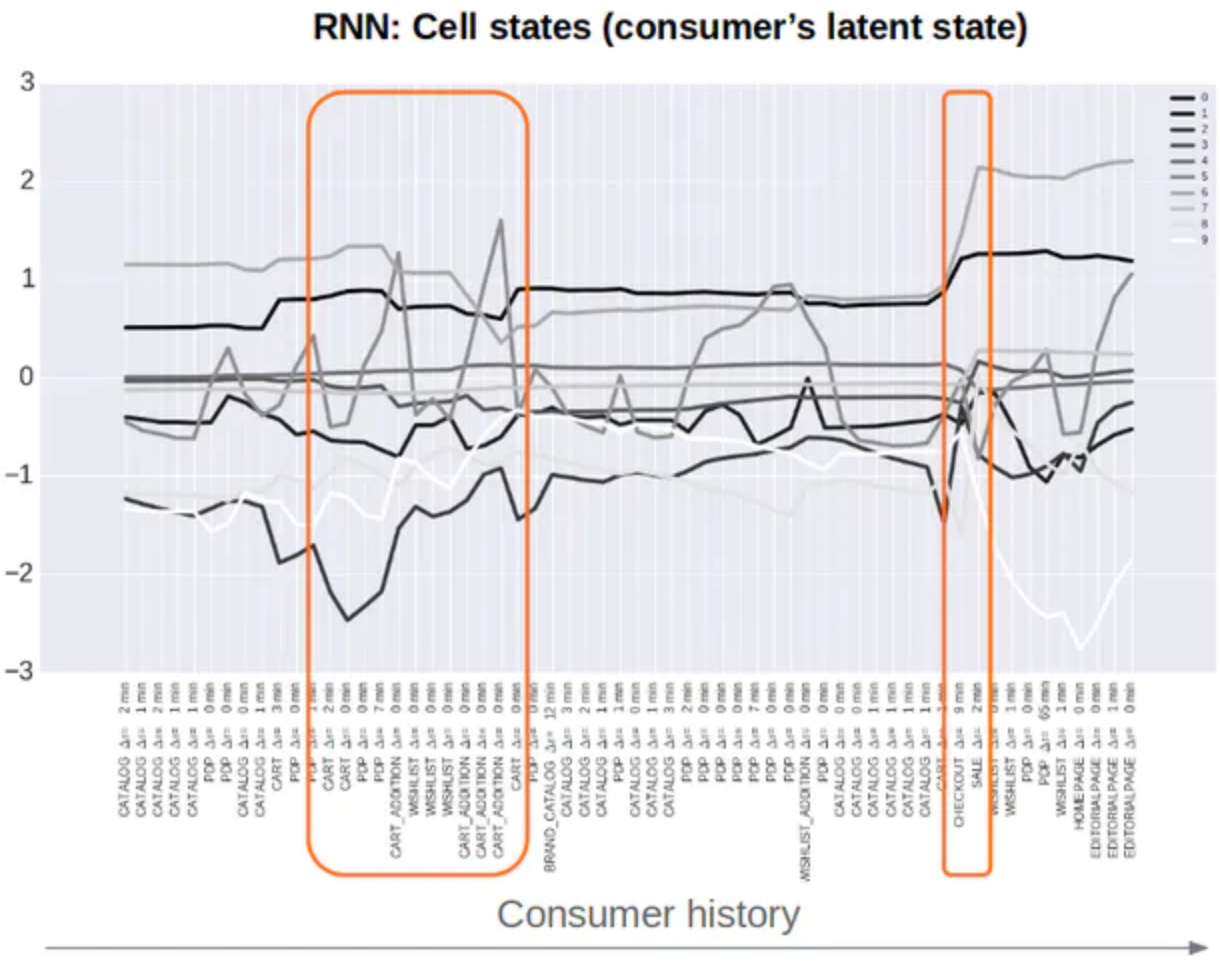
Nowadays, a growing concern is **model interpretability**. Contrary to popular belief, it can be more difficult than often assumed to interpret logistic regression models, due to correlated features and feature binarization.

Likewise, the parameters of RNN models are hard to make sense of. In contrast, we can interpret the prediction process of an RNN for a given consumer history, as we'll see in the following.

Visualizing RNN predictions to understand consumer behavior and improve service offers

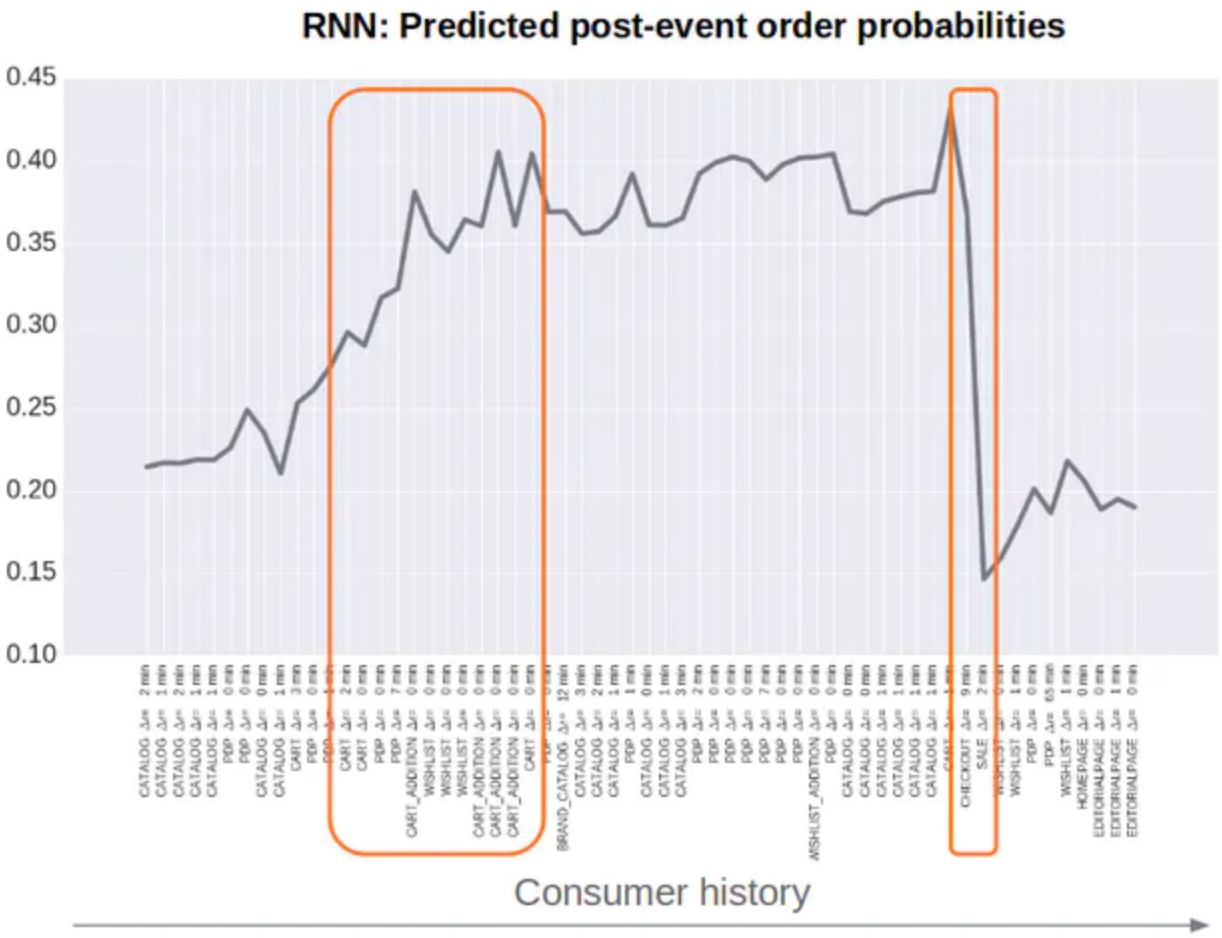
Understanding the consumer journey is a holy grail in e-commerce. Analyzing the reasoning of prediction models provides insights into customer behavior and ways to improve the service experience of consumers. In addition, visualizing the prediction process is helpful to assure oneself that a model does what it is intended to do. RNNs can be a great tool for these purposes.

The following graph shows the latent cell states of a trained RNN when fed with the history of a specific consumer (*PDP* = product detail page view, *SALE* = order, *CART* = cart view):



While the cell values cannot be interpreted by themselves, we can see how they change for the given inputs over time. In particular, the model is sensitive to specific event patterns (emphasized in orange boxes): The RNN has learned to detect consumer behavior indicative of future orders.

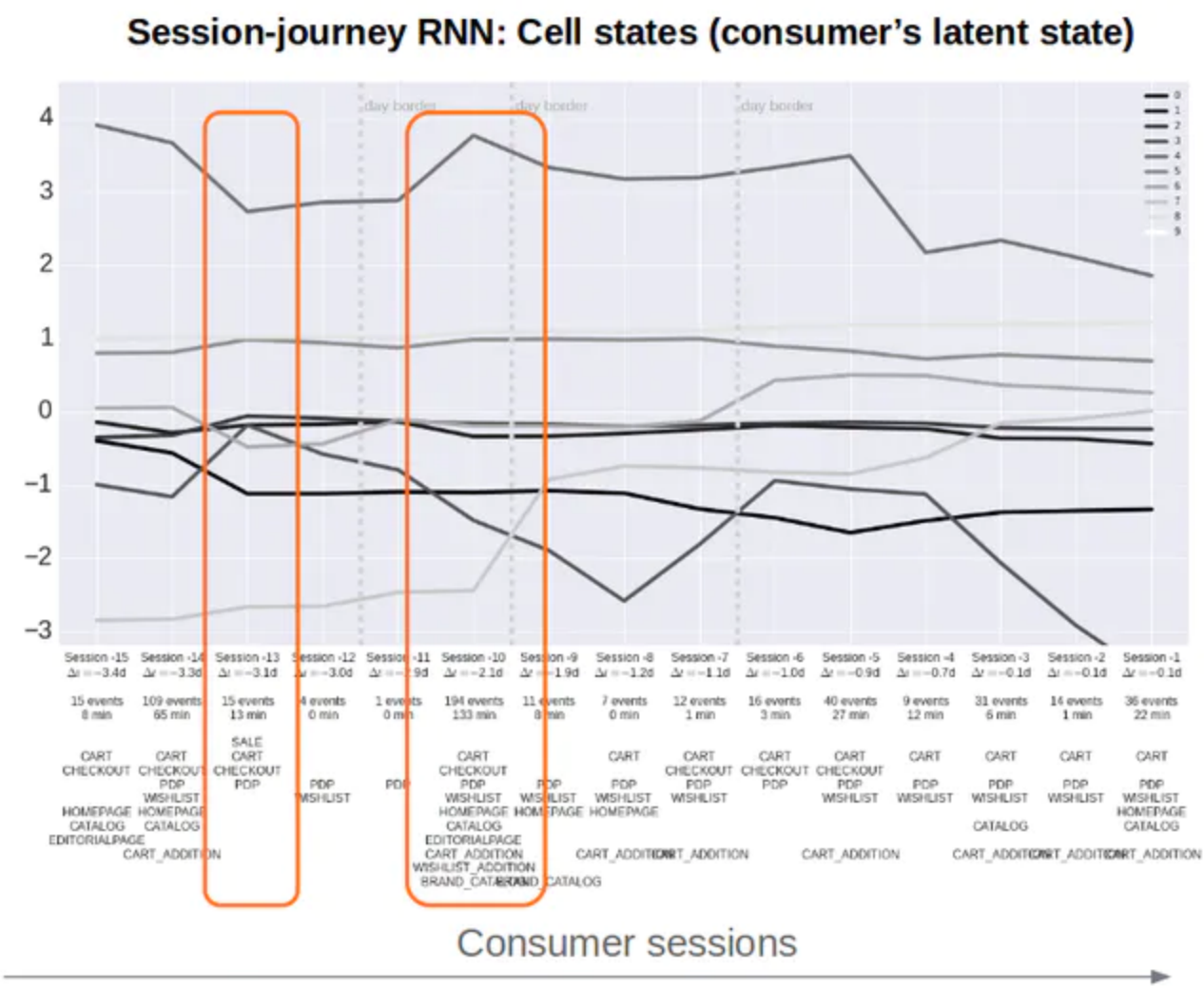
How wishlist views and cart additions increase order probability, while orders decrease it (in the context of the consumer's previous events) can be seen in the corresponding predicted probabilities:



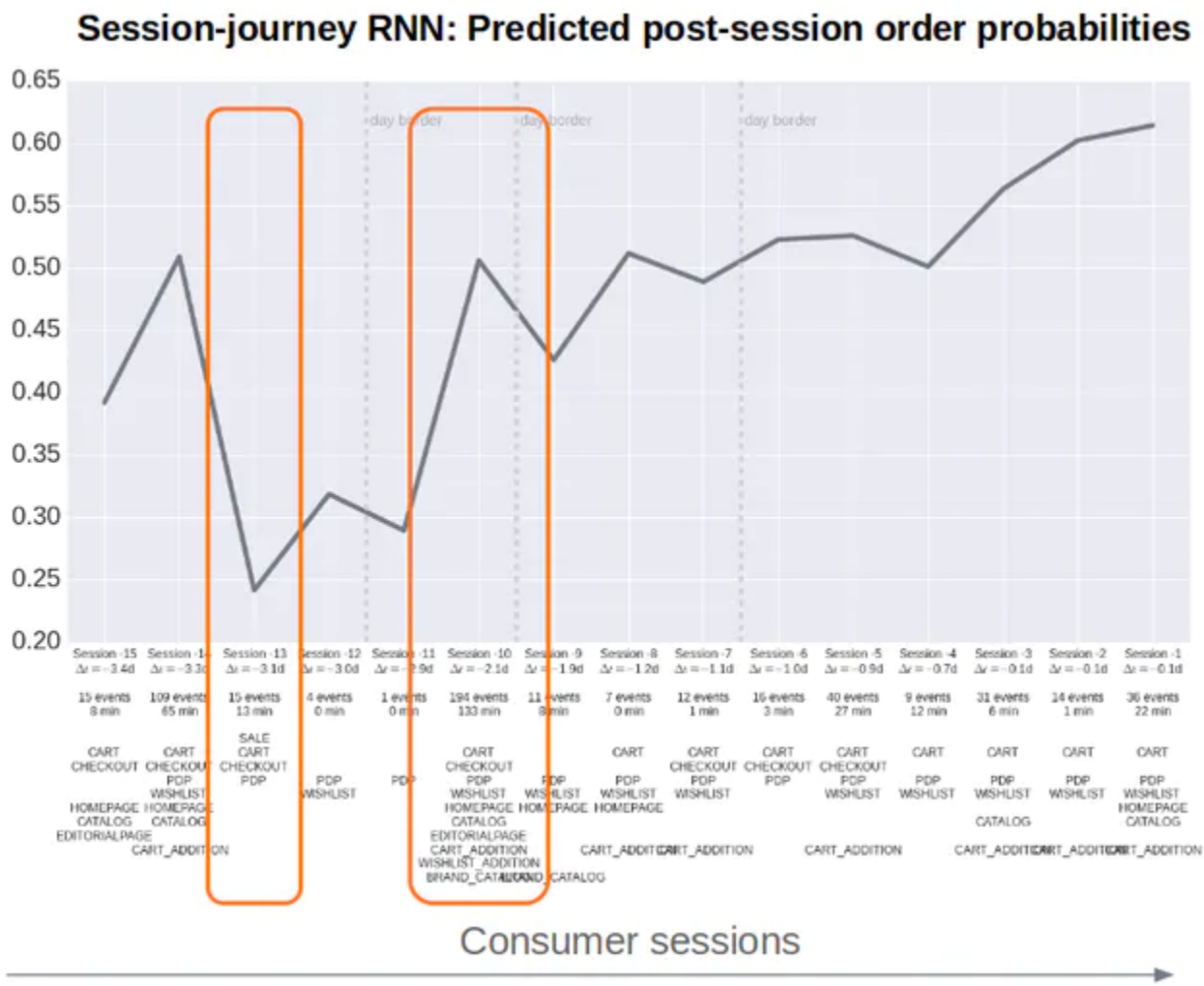
In e-commerce, we often want to make sense of the **session journey** of a consumer, the sequence of their **visits**. In our context, we would like to understand which sequences of sessions lead to orders. A session is simply a collection of events, with no events before or after for a specified duration. Instead of individual events, we can feed sessions into an RNN. More precisely, session inputs specify which event types happened within the session and how long the session was in minutes and in number of events. (This is a mild form of feature engineering.) We use the same RNN architecture, but training with session inputs results in a different RNN model, namely one that understands sessions instead of events.

Concerning empirical performance in our experiments, AUC values were similar for simple RNNs trained with event and session-journey inputs (the same until the third position after the decimal point; we show the event-journey RNN results in the AUC graph above). For the complex RNNs, we only experimented with the session-journey representation (for which we show the result in the AUC graph).

The following diagram depicts the resulting cell states for an exemplary consumer session journey:



The 10th to last session of the consumer was intense with 194 events (the kind of session we love to see at Zalando). Intuitively, this shows strong interest in ordering. Indeed, this is reflected in the predicted order probability which jumps from 29% to 51%:



These visualizations allow us to assess in a quantitative way how consumer actions affect model predictions. They deepen our understanding of how consumers interact with Zalando and shift vague assumptions about this process on a firm empirical basis.

In principle, the progress of predicted order probabilities could also be visualized with vector-based machine learning methods like logistic regression, but in a cumbersome and inefficient way: We would need to re-calculate all hand-engineered features at every single time-step. This would be a highly redundant process: The features at time-step *t* would represent the complete history until *t* and not only what happened between *t-1* and *t*. In contrast, all calculations and probabilities come for free in the prediction process of an RNN, as RNNs model sequences in a natural, direct way. More importantly, with vector-based machine learning methods, we would still need to interpret the hand-engineered features at every time-step to make sense of the model. If you have hundreds or thousands of features, and features are correlated and have been preprocessed, this is usually a complex and confusing task.

Wrap up and next steps

When you have a detailed understanding of your domain or when you're building your first predictive models, feature engineering with logistic regression, random forests and the like are a great way to start.

The future of modeling sequential data, however, seems to belong to deep learning methods: They do most of the feature engineering for you while achieving superior accuracy. RNNs process sequences in a natural way and are therefore particularly promising for modeling consumer behavior in e-commerce. They are a versatile tool if prediction tasks are diverse and when feature engineering becomes more cumbersome.

At the same time, RNNs are great in explaining the reasoning in the prediction for individual consumers, again due to their natural modeling of sequences.

In practice, rolling out and maintaining deep learning models is more challenging than using traditional machine learning models. Nevertheless, the gains may be well worth the effort.