On Deriving Optimal Cadences via Engagement Score Maximization

Roi Ceren
Data Scientist
SalesLoft
roi.ceren@salesloft.com

Scott Mitchell
Chief Technical Officer
SalesLoft
scott.mitchell@salesloft.com

Abstract

We propose a framework for analyzing the behaviors that users of the SalesLoft platform enact when interacting with their prospective customers, or contacts. The framework identifies the series of behaviors (*interactions*) users generate when following a defined Cadence, or scheduled pattern of interaction, that maximize the amount of *time effort* a contact spends in response. We adopt the perspective of decision theory, treating the underlying environment as almost completely unobservable. As a result, our approach is entirely model-free, leveraging reinforcement learning by directly exploring the space of behaviorally-derived Cadences using action-values derived from a large corpus of empirical samples in our database. We validate the model by comparing expected scores of high- and low-performing users of our platform and present several general insights from our observed results.

1 Introduction

In the domain of sales, Cadences are a powerful structural tool for organizing and scheduling a salesperson's actions when reaching out to target parties who may hold an interest in their product or service. These parties, referred to as *contacts*, are nuanced in their interest. Depending on their budget, organizational needs for the seller's product (which may be influenced by industry or their organization's size), and their position in the company, a contact may respond differently depending on the method of interaction.

One of the primary features of the SalesLoft platform is the management, view, and execution of Cadences. Contemporary literature on modern sales is rife with best practices for the structure of these Cadences, many of them based solely on expert understanding, often succumbing to pure conjecture or potentially erroneous correlation. Alternatively, we desire mathematically well-founded insights into the behavioral outcome of Cadence use. Thus, we pursue an approach that instead directly draws from the data generated from the *users* of our platform.

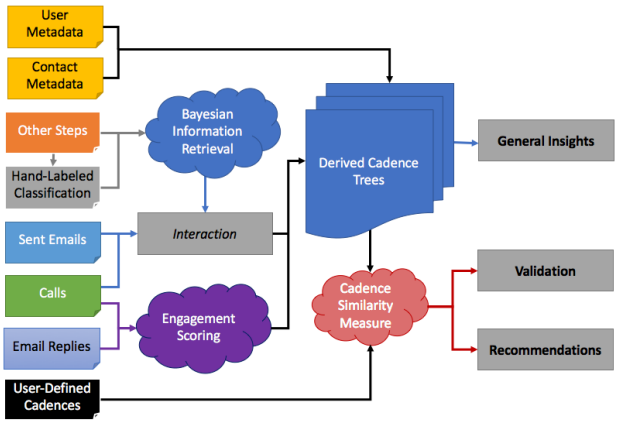

Figure 1: High level workflow illustrating the Derived Cadence framework

As a result, we propose leveraging the massive data generated in the use of our SalesLoft platform in a novel ensemble framework, based on probabilistic first principles. We establish a model-free reinforcement learning (RL) method for a generalized Markovian process with the goal of identifying fruitful, high-valued Cadences, based purely on direct action-values, computed from the amount of time contacts spend responding to user actions. Instead of relying on the Cadences that our user's follow, we base our framework on the behavior they elicit, and label the resulting scored behavior as Derived Cadences. This machine learning framework will serve as our departure point for extracting the maximum amount of insight from the noisy, largely unobservable domain of sales. Figure 1 illustrates the process of learning Derived Cadences and how they may be used.

The work proceeds as follows. In Sec. 2 we discuss the form of our data set and engagement scoring. Section 3 then defines how we model user behavior, learn scores for the model, how we quantify descriptive Other steps for the model, and a local search mechanism for finding similar Derived Cadences from an origin Cadence. Section 4 then validates the model and discusses general insights that can be learned from our analytical framework.

2 Problem Domain

Users of the SalesLoft platform generate Call, Email, and Other steps when executing Cadences. However, this does not comprise all actions a user may take. Users may, and often do, deviate from a Cadence, sending one-off emails or follow-up calls to a contact. As a result, we do not generate samples from Cadences, but instead the underlying *behavior* a user actually executes when pursuing a contact. We label these behaviors as *interactions* (also contextually known as a *touch*), where an *interaction sequence* is the series of all actions between a user and their contact.

By examining actions instead of scheduled Cadence steps, we retrieve the event that occurred when executing the step. Understanding these events affords us metrics for evaluating the effectiveness of the user's interactions. To this end, we interpret the amount of *time effort* as a real number computed from contact responses to a user's action, with the aggregate labeled the *Engagement Score*. Engagement Score is so named as a reference to the goal of seeking contact engagement, or the point in which a contact's attention has been established.

Email replies are processed as the minute time the contact takes to write an email [1][1]. Calls are merely the recorded minute duration of the call, if they connect to a contact. We additionally weight email replies to connected calls by a factor of 3.81[2]. Therefore, Engagement Score of a touch $ES(t)$ is a piecewise function defined by

$$ES(t) = \begin{cases} \frac{length(t)}{240} & \text{if } t = Email \\ \frac{duration(t)}{3.81} & \text{if } t = Call \end{cases} \quad (1)$$

where $length(t)$ is the character size of the email with signatures and quoted bodies removed and $duration(t)$ is the minute length of the call. Note that Other steps are not scored. However, Engagement Score is conditionally dependent on the entire interaction that precedes it. As an example, an Other step to research a contact before emailing influences the quality of the email, and commensurately improves the likelihood of a contact response.

As with many real-world experiments, much of our data set is not formatted properly for the extraction of event data. As a result, we preprocess the sample corpus to exclude noisy data, creating the most well-formed corpus possible. Only using 6.11% of our total data set, our processed corpus still includes over 14.5 million interactions, collapsed into 2.9 million interaction sequences.

[1]Average writing speed selected from "fast" cohort

[2]This ratio was computed in an experiment on SalesLoft internal data correlating average call connect times to email replies, stratified by the day of the action and from interactions that resulted in a Salesforce Opportunity closing and being won.

3 Markovian Model for Optimizing Behaviorally-Derived Cadences

It is well known that the underlying mechanics determining the response behavior from a contact is complex [2]. As a result, attempting to model these mechanics is both computationally complex and often impossible to learn without access to unknowable information, such as contact deal size, their company's organizational chart, or behavioral disposition. Consequently, a promising approach in problems of limited information involves instead learning from direct action-values [3] (in our domain, Engagement Score of each touch). With these action-values, we establish a RL technique to learn and search for optimal interactions.

We represent a sequence of interactions as a tuple $\mathcal{X} = \{\langle d_0, t_0\rangle, \langle d_1, t_1\rangle, ..., \langle d_z, t_z\rangle\} = \{\langle d_i, t_i\rangle\}_{i=0}^{z}$, where $d \in (0, \infty)$ is the day a touch $t \in T$, where $T = \{Email, Call, Other\}$, is scheduled on and $z = |\mathcal{X}|$ is the length of the interaction sequence. As noted in Sec. 2, users deviate from their defined Cadences, so we instead learn from the underlying sequence of day-touch tuples they exhibit when interacting with a contact. After applying Eq. 1, we then define a *scored interaction sequence* as $\mathcal{X}_{ES} = \{\langle d_i, t_i, ES_i\rangle\}_{i=0}^{z}$. We then process the interactions with the intent to remove potentially unnecessary steps, covered in Sec. 3.2. We then aggregate the scored interactions with others of similar structure, compute the mean Engagement Score and its variance, and label the resulting scored structure as behaviorally-*Derived Cadences* (DCs), $\mathcal{X}_{DC} = \{\langle d_i, t_i, \bar{ES}_i, S_i\rangle\}_{i=0}^{z}$.

Due to the obfuscation of contact information, we treat the problem as a generalized Markov process where the state (here, the disposition of a contact influencing engagement) is fully hidden, thus no observation of the state is given. Since the state is unknown, our departure point is online, model-free RL. Our goal is to create a Derived Cadence tree, representing the entire policy space (where a policy here is contextually an arbitrary-length DC) of empirically sampled interactions. Policies are evaluated online, leveraging RL to acquire action-values in the DC.

3.1 Learning from Interactions

In the SalesLoft platform, a Cadence is simply defined as a series of outgoing email, call, and "other" steps, each scheduled for a particular day in a cycle. The Other step is mutable, and can represent social touches (such as LinkedIn connections or tweets), an indicator for a need to research a contact, or some other internal organizational process. Consequently, a policy in our Markovian framework represents the above environment: each action in a DC may be one of the above types, with other steps factored into 13 distinct categories.

In generalized Markovian models, such as the Markov Decision Process (MDP) or Partially Observable MDP (POMDP), rewards may be predicated on additional stimuli from the environment, such as its observable state [4] or noisy observation from the state [5]. However, in our domain, we do not have exposure to the underlying mechanics of the state, nor do we have any indication of the contact's disposition. In such settings, we may condition the learning on the entire sequential history of interactions [6], treating each Engagement Score stimuli as a product of the series of day-touch events that precede it.

To avoid the curse of dimensionality when exploring DCs, our policy space is initially empty. We learn possible structures online as we sample interactions from our corpus. When sampling an interaction sequence that does not exist in the policy space, we create it with the Engagement Score initialized to the time effort values in the initial sample, described in Sec. 2. For each subsequent sample of the interaction sequence, the Score and its variance is updated via moving average and moving variance [7]. The moving average update is defined as

$$\bar{ES}_n = \bar{ES}_{n-1} + \frac{ES_n - ES_{n-1}}{n}$$

for score ES, mean $\bar{ES}$, and sample density n. As noted above, ES is initialized $\bar{ES}_0 = ES_0$ and $S_0 = 0$. Variance is updated as

$$M_n = M_{n-1} + \frac{ES_n - M_{n-1}}{n}$$

$$V_n = V_{n-1} + \frac{ES_n - M_{n-1}}{ES_n - M_n} \tag{2}$$

where variance for $0 < n \leq Z$ is computed as $S_n = \frac{V_n}{n-1}$, and $M_0 = ES_0, V_0 = 0$. The aggregate Engagement Score at the ith interaction, where $0 \leq i \leq z$ is:

$$\bar{ES}_i(\mathcal{X}) = \sum_{j=0}^{i} \mathcal{X}(\bar{ES}_j)$$

Every contiguous subset of an interaction sequence (constrained to those that begin with the first step) is treated as a separate sample. This is done to identify potentially fruitful core sections of the interactions, where subsequent actions may exceed a desired time or step limit without providing a significant portion of the overall Engagement Score.

Since we aggregate scores based on structure, we can simply represent the space of possible DCs as a tree. Each vertex is a touch and aggregate Engagement Score, and each edge is the relative amount of days that transpire between vertices. Since the Engagement Score is conditionally dependent on the vertices that precede it, no child of a parent may be the child of another vertex in the tree. With this as our underlying DC model, it affords us simple tree search mechanisms as a way to search the space of possible Cadences.

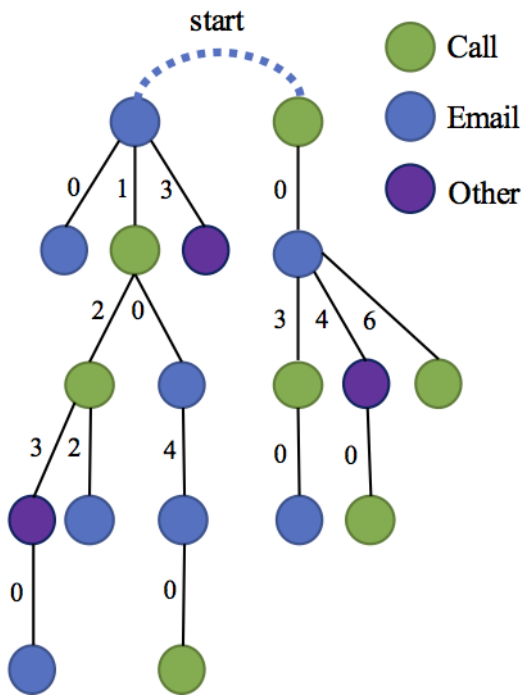

Figure 2: Example small DC tree after ingesting only 10 interactions. For brevity, ES and variance are omitted.

We illustrate an example Derived Cadence tree in Fig. 2. This example tree follows the structure of a DC tree for users in companies in the Salesforce-defined Computer Software industry. This tree illustrates one of the insights we present in Sec. 4.2, the behavior of a multi-channel *double tap* on the same day.

3.2 Engagement Pruning

An important observation about Engagement Score is that it must assume that the contact has not engaged prior to the scored interaction step $\langle d_k, t_k, ES_k \rangle$. We examine two pruning mechanisms to shorten the resultant DC to only the interactions that *lead to* an engagement. First, we prune all actions that follow the timestamp of the SalesLoft Success [3] button.

The second pruning mechanism is more involved: we first examine the relative change in Engagement Scores stratified by the absolute day the touch is defined on. We

[3] https://salesloft.com/resources/blog/2016/01/9-features-in-7-days/#success

compute both the average change and the variance, generating a confidence interval over the change.

Let the set of all scored interactions in the corpus ($\vec{\mathcal{X}}_{ES}$) satisfying the condition where an interaction ($\phi = \langle d, t, ES \rangle$) occurs on particular day d be $\vec{\mathcal{X}}_d = \{x | x \in \mathcal{X} \wedge \exists \phi.(\phi \in x \wedge d \in \phi)\}$. That is, $\vec{\mathcal{X}}_d$ is the subset of interaction sequences within the total set $\mathcal{X}$ that satisfy the requirement of a touch being executed on day d. Let $\vec{\mathcal{X}}_d^k$ be the kth interaction sequence of the set. Lastly, let the function for retrieving a value $v \in \{d, t, ES\}$ in the kth interaction sequence be $\vec{\mathcal{X}}_d^k(v)$. For example, retrieving the Engagement Score follows the notation $\vec{\mathcal{X}}_d^k(ES)$.

Then, the average expected Engagement Score slope for a given day is

$$\hat{ES}(d) = \sum_{k=0}^{|\vec{\mathcal{X}}_d|} \sum_{i=1}^{|\vec{\mathcal{X}}_d^k|} \mathcal{I}(\vec{\mathcal{X}}_d^k(d), d) \left(\frac{\vec{\mathcal{X}}_d^k(ES) - \vec{\mathcal{X}}_d^{k-1}(ES)}{|\vec{\mathcal{X}}_d| \cdot \vec{\mathcal{X}}_d^{k-1}(ES)} \right)$$

where $\mathcal{I}(a, b)$ is an indicator function returning 1 if $a = b$ and 0 otherwise. Specifically, the above computes the slope change of the ES of the interaction step where the specific day occurs compared to the step that precedes it, and then averages it across the entire set. This method is executed in practice at the ingestion stage, and maintains a matrix of Engagement Score slopes by day.

$S^{\hat{ES}(d)}$ is the variance over $\hat{ES}(d)$, and may be computed as in Eq. 2 or using canonical techniques. We identify the *first instance of interaction* $\langle d_k, t_k, ES_k, 0 \rangle$ where $0 < k \leq z$ such that

$$\frac{(ES_k - ES_{k-1})}{ES_{k-1}} \geq \hat{ES}(d_k) + 3\sqrt{S^{\hat{ES}(d_k)}}$$

The above function uses a conservative estimation for the potential slope increase, and only considers those that exceed the normal average slope increase for that day with very high probability.

With this, we can identify the point in which a contact has responded with a disproportional increase in time effort. We then prune the interaction to the kth element, $\mathcal{X}_{ES} = \{\langle d_i, t_i, ES_0 \rangle\}_{i=0}^{k}$ and ingest it into the DC engine as in Sec. 3.1.

3.3 Quantifying the Other Step

Since Other steps in a Cadence are mutable and text-based, acquiring a discrete action in the model requires additional steps. We tackle this problem in a supervised way, by first deciding on a set of possible categories, learn a Bayesian feature vector over words in these categories, and classify the other step portion of our corpus.

In an internal effort, we consulted with the customer success organization within SalesLoft and identified several categories our users create Other steps for. For example, 51.73% of our Other steps are related to LinkedIn, and are used in at least 3 ways: send a connection request, send an InMail, or comment on a contact's post. Besides LinkedIn, some users research their contacts, move them to other Cadences, or reach out via Twitter.

We then set out to label a small but sizeable proportion of this data set. SalesLoft employees across the company hand-labeled 1,189 samples from the corpus, using a set of categories that include the above examples. SalesLoft employees could, and often did, disagree on the classification of an Other step, so we utilize first-past-the-post voting and use the label that has quorum. Disagreement may arise due to the fact Other steps can be ambiguous, as different users may use terms differently, or or Other steps can define more than one discrete action (e.g. both a LinkedIn Mail and a Twitter touch).

Following that effort, we apply basic information retrieval tactics on each Other step's name and description, such as stop word removal and stemming, and create a feature vector of the resultant phrases via a heavily modified Naive Bayes method [8]. We compute the probabilistic relationship between a category and the words used within them from our labeled set. These vectors are then used to classify the remaining, unlabeled data, and leveraged as a discrete action in the model.

The Other step classifier resulted in a 73% accuracy via 10-fold cross-validation. The lower accuracy may be due to the ambiguity or multi-type phenomena described above. A potential way to improve the model is to support *mixed types*, which capture multi-type Other steps [9].

3.4 Cadence Similarity Measure

While users of the SalesLoft platform may define Cadences that thousands of contacts are run through, it is common to deviate slightly from a defined Cadence or render it incomplete. The latter may arise from the user accomplishing the goal of the Cadence early, as their contact may *engage* or be *disqualified* before the Cadence is complete. As a result, the Derived Cadence engine defined in this work may not have enough samples to confidently report the expected Engagement Score of a given defined Cadence.

We decompose the process of scoring low-sample or unseen Cadences into two steps: (1) explore the space of neighboring (identical in all day-action pairs except for one) DCs, and (2) establish a way to measure the distance between a defined Cadence (called the *origin*) and a DC (the *neighbor*). Each has a dramatic impact on runtime, as the neighborhood grows combinatorially with regards to the length of the origin.

We define a *neighborhood* as the set of Cadences $\vec{\mathcal{N}}$ that differ from the origin, defined by a set of touches $\mathcal{O}$, by a single day-touch pair, to a maximum of 5 days differ-

Algorithm 1 $nbr(\mathcal{O})$: Generate neighborhood

Require: Origin day-touch set $\mathcal{O}$

1: $\vec{\mathcal{N}} \leftarrow \emptyset$
2: **for** $\langle d_k, t_k \rangle \in \mathcal{O}$ **do**
3: $\quad p \leftarrow -\infty$
4: $\quad q \leftarrow \infty$
5: $\quad$ **if** $k > 0$ **then**
6: $\quad\quad p \leftarrow d_{k-1}$
7: $\quad$ **if** $k < |\mathcal{O}|$ **then**
8: $\quad\quad q \leftarrow d_{k+1}$
9: $\quad$ **for** $i \in \{\max[p, d-5], \min[q, d+5]\}$ **do**
10: $\quad\quad$ **if** $i \neq d$ **then**
11: $\quad\quad\quad \vec{\mathcal{N}} \leftarrow \vec{\mathcal{N}} \bigcup (\mathcal{O} - \langle d_k, t_k \rangle) + \langle i, t \rangle$
12: $\quad\quad$ **if** $t = Email$ **then**
13: $\quad\quad\quad \vec{\mathcal{N}} \leftarrow \vec{\mathcal{N}} \bigcup (\mathcal{O} - \langle d_k, t_k \rangle) + \langle i, Call \rangle$
14: $\quad\quad$ **if** $t = Call$ **then**
15: $\quad\quad\quad \vec{\mathcal{N}} \leftarrow \vec{\mathcal{N}} \bigcup (\mathcal{O} - \langle d_k, t_k \rangle) + \langle i, Email \rangle$
16: **return** $\vec{\mathcal{N}}$

Algorithm 2 $dist(\mathcal{O}, \mathcal{N})$: Distance metric between origin and neighbor

Require: Day-touch sets $\mathcal{O}, \mathcal{N}$

1: $dist \leftarrow \infty$
2: $\mathcal{O}^- \leftarrow \mathcal{O} - (\mathcal{O} \cap \mathcal{N})$
3: $\mathcal{N}^- \leftarrow \mathcal{N} - (\mathcal{O} \cap \mathcal{N})$
4: $\vec{\mathcal{O}}^- \leftarrow \binom{\mathcal{O}^-}{\min[|\mathcal{O}^-|, |\mathcal{N}^-|]}$
5: $\vec{\mathcal{N}}^- \leftarrow \binom{\mathcal{N}^-}{\min[|\mathcal{O}^-|, |\mathcal{N}^-|]}$
6: **for** $\{\mathcal{O}_i, \mathcal{N}_i\} \in \vec{\mathcal{O}}^- \bigtimes \vec{\mathcal{N}}^-$ **do**
7: $\quad dist_i \leftarrow \left(1 + \sum_{\substack{\langle d_\mathcal{O}, t_\mathcal{O} \rangle \\ \in \mathcal{O}^- - \mathcal{O}_i}} d_\mathcal{O} + \sum_{\substack{\langle d_\mathcal{N}, t_\mathcal{N} \rangle \\ \in \mathcal{N}^- - \mathcal{N}_i}} d_\mathcal{N} \right)^2 - 1$
8: $\quad$ **for** $\{\langle d_\mathcal{O}, t_\mathcal{O} \rangle, \langle d_\mathcal{N}, t_\mathcal{N} \rangle\} \in \{\mathcal{O}_i, \mathcal{N}_i\}$ **do**
9: $\quad\quad$ **if** $t_\mathcal{O} = t_\mathcal{N}$ **then**
10: $\quad\quad\quad dist_i \leftarrow dist_i + |d_\mathcal{O} - d_\mathcal{N}|$
11: $\quad\quad$ **else**
12: $\quad\quad\quad dist_i \leftarrow dist_i + 2 \cdot (1 + |d_\mathcal{O} - d_\mathcal{N}|)$
13: $\quad dist \leftarrow \min[dist, dist_i]$
14: **return** $dist$

ent than the original schedule. We exclude Engagement Scores, allowing comparison between Cadences and DCs. Algorithm 1 defines the neighborhood generation process. Lines 6 and 8 ensures the transformation for a day is bound by the scheduled touches before and after it. Line 2 iterates over all day-touch tuples in the origin, exploring neighbors up to 5 days earlier or later. Every neighbor that differs by this range, and those that differ by the touch defined on that day, are added. The computational complexity of generating a neighborhood grows w.r.t. the length of the origin Cadence and number of touches, $O(|T|z)$.

While a Touch's scheduled Day is quantitative, and thus straightforward to compute the distance for, the Touch *type* (Email, Call, or Other) is not. Algorithm 2 annotates the distance computation. As a first step, we eliminate all day-touch pairs from the origin and neighbor that are identical on lines 2 and 3. Next, lines 4 and 5 extract every combination of steps within each set, limited to a subset the size of the smallest original excluded set. Line 6 then explores the Cartesian product of these steps, with the goal of identifying the lowest-distance combination of the subsets. Line 7 gets the cost of the excluded day-touch pairs from whichever set, origin or neighbor, is longer. The subsequent loop explores the pairs in the joint set and applies a cost function that sums the day difference and, if the touches differ, doubles the cost. Lastly, if a new minimum pairing has been found, the distance is updated in line 13. Since this only produces transformations on a single day-touch tuple, the method may be nested to produce multiple transformations, at the increased expense of a much larger local neighborhood.

Finding ideal neighbors is quite difficult considering the combinatorial space of Cadences. As a result, a local search mechanism is ideal. We leverage the well-known methodology of *exploring starts* [4], a RL approach that picks random parts of a strategy and transforms the action on that part. If the origin Cadence is not in the DC engine, our method takes the origin, takes a day-touch pair, and checks if there is a DC that matches the origin where the day-touch pair takes place from 5 days before to 5 days after the origin's definition, and additionally checks if another Touch type matches. It does this for every day-touch pair in the origin. If it exhausts this set, and still cannot find a neighbor, it takes every combination of origin Cadences that is one less touch in length. It continues the above process until it identifies a candidate set of DCs.

The candidate set is then scored using the distance metric defined in this subsection. Depending on the relative distance to the origin, the approximate score of the origin is weighted towards the candidate DC. The approximate score is simply defined as follows:

$$\bar{ES}(\mathcal{O}) = \frac{\sum_{\mathcal{N} \in nbr(\mathcal{O})} \bar{ES}(\mathcal{N}) \cdot w(\mathcal{O}, \mathcal{N}, nbr(\mathcal{O}))}{\sum_{\mathcal{N} \in nbr(\mathcal{O})} w(\mathcal{O}, \mathcal{N}, nbr(\mathcal{O}))} \quad (3)$$

where

$$w(\mathcal{O}, \mathcal{N}, \vec{\mathcal{N}}) = 1 + \max_{\mathcal{N}_i \in \vec{\mathcal{N}}} dist(\mathcal{O}, \mathcal{N}_i) - dist(\mathcal{O}, \mathcal{N})$$

As a result, the approximate score is weighted towards the minimum distance, where the weight monotonically decreases as the neighboring DC grows farther away.

4 Evaluation and Discussion

We examine the result of ingesting the entire corpus of low-noise interaction data. We identify a few nuances dependent on the user's industry, but primarily interest ourselves in the results across all industries. As ingesting the entire corpus of interaction samples results in 3.2 million distinct DCs, we include a number of constraints to improve the scalability of the tree search.

Primarily, we optimize over two constraints: a limitation on a number of days and touches to appear in the resultant DCs. For the purposes of industry, we learn a new DC tree, thus exploring a separate policy space. For other team-defining features (e.g. employee count), we learn separate action-values, but compute ES using all of them, weighting the sample contact's features proportionally higher. For the time and touch constraints, we modify the ES of a DC using an exponentially decaying loss function, allowing DCs that differ from the constraints slightly to be weighted higher than those that are much different.

The goal of our framework is not to discover universal rules for good Cadences, but instead to leverage empirically sampled behavior as a mechanism for extracting insight. To this end, we provide some high-level validation of our model to provide credibility, and follow with some general insights we observe in high-ranking DCs.

4.1 Model Validation

We selected six example *teams* of users from two large companies and four small. Each category is split between high and low performers. High performers are defined as teams in companies who are experiencing corporate growth (both in employee count and recurring revenue), while low performers exhibit the inverse or stagnant growth. Teams are then evaluated by the aggregate score of their Cadences (either the computed ES if the Cadence exists in the DC tree, Eq. 3 otherwise). The aggregate score is weighted to the overall proportion of contacts the team has run through that Cadence.

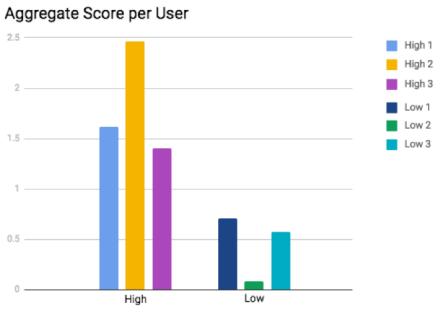

Figure 3: Aggregate Scores per User

Figure 3 illustrates the aggregate score of the six example users, stratified by performance. On average, high performers exhibit 1.3 times more Engagement Score over their Cadences than low performers.

Interestingly, the low performing users had extremely odd Cadences, ones that varied dramatically from the DC Engine's commonly encountered ones, and had the highest variance in distances when computing approximate scores. This would indicate that low performing users often deviate from their defined Cadences, and these deviations are also poorly scored. From another perspective, high performing users exhibit much more consistent behavior with commensurately higher-valued Cadences.

4.2 General Insights

With DCs established as correlating with real-world performance, we identify several general valuable insights observed by our examination of high score DCs. Each of these insights is positively correlated with maximizing the Engagement Score between a user and their contact.

Multi-Channel Double Tap: When examining DCs near 45 days and 13 touches in length, over 80% of the top 100 Cadences include a Call and an Email on the same day, and almost always on the first day. Often, a double tap will occur multiple times over the course of the Cadence, appearing an average of 2.96 times over the 80+ Cadences. 88.7% of high-valued double taps start with a Call.

Avoid Rounding Days: From the 15th Day onwards, users commonly define steps on every 5th subsequent day disproportionately. Figure 4 illustrates this phenomenon, with 5th day definitions highlighted in red. Figure 5 shows the average distributions for 3600 Derived Cadences, weighted by their overall Engagement Score. The range for both figures is between 15 and 50 days. Fig. 5 illustrates what high-value DC day distributions look like.

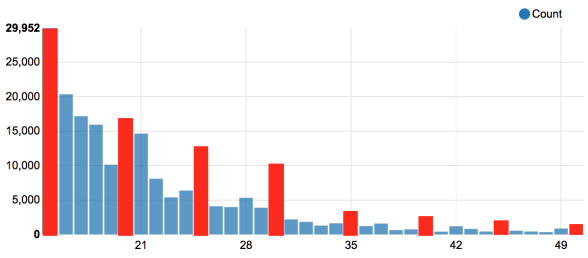

Figure 4: Defined Cadences adhere to round-day schedules, unlike high-valued Derived Cadences.

Effort Up Front: While also apparent in our users' defined Cadences, DCs illustrate that effort is better spent up front, near the start of contact, and tapering off in later days. An expanded view of Fig. 5 supports this finding, illustrated in Fig. 6.

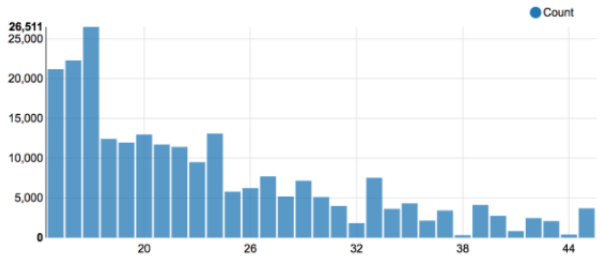

Figure 5: Derived Cadences illustrate more distributed days for each touch, with a few density spikes.

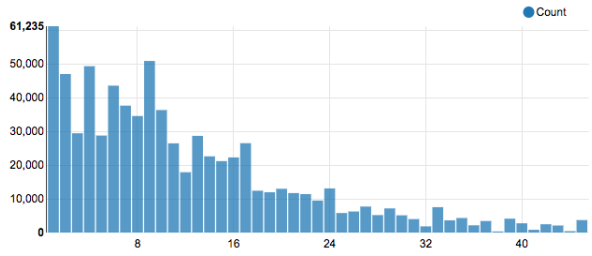

Figure 6: Expanded view of Fig. 5

Timing Matters: The day a user executes a touch is of significant importance. To validate this assumption, we took 100 of our most valuable DCs, and examined the effect on score when doubling, halving, or evenly spreading the delay between touches for each DC.

Double Time	*Halve Time*	*Even Spread*
-97.47%	-90.69%	-36.13%

Multi-Channel Matters: We examine 2 possible scenarios for determining multi-channel effectiveness. In each, we analyze the total loss introduced with each mutation of the original Cadence. First, we *remove* all multi-channel touches, leaving behind only the originally defined single channel. Second, we *transform* all touches to a particular channel. Again, we test against our top 100 Cadences.

Remove Calls	*Remove Emails*	*Transform to Emails*	*Transform to Calls*
-98.57%	-93.71 %	-76.49%	-91.43%

According to these results, having long, pure-email Touch Cadences introduced the least amount of Engagement Score loss, but each mutation produces staggeringly lower performance than the commensurate original high-score Cadence.

Don't Wait, Don't Push: Proper pacing of touches is highly contextual to the contact and goal of a Cadence, but we observe a distinct pattern of delays between touches over time. Early in an interaction, frequent contact and occasional double-taps are common. As time goes on, however, the best Cadences slow down. Figure 7 shows the computed distribution for average delays between touches.

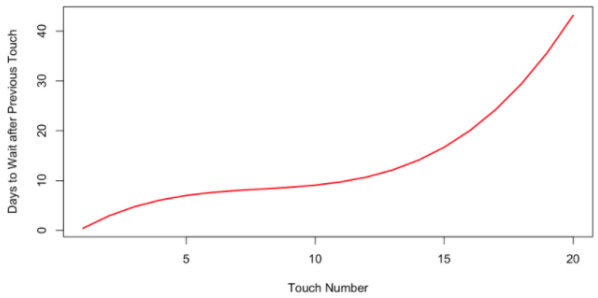

Figure 7: Average delays between Touches

5 Conclusion

In this work, we propose a novel ensemble method for identifying an optimal schedule of actions in the context of the SalesLoft platform via maximization of the expected time effort from a user's contact. Users often deviate from their defined Cadences in our platform. Consequently, we leverage the actual behavior that users execute, dubbed interactions, defined by the series of emails, calls, and Other steps they generate when reaching out to a contact. Since the mechanics that determine a contact's engagement rate are largely hidden, we instead directly utilize the amount of time a contact spends responding to a user, labeled Engagement Score.

Consequently, we frame the problem of eliciting engagement from a user as a model-free reinforcement learning problem for a generalized Markovian model, utilizing Engagement Score as the action-value. Calls, Emails, and Other steps are empirically sampled from interactions. As Other steps are not a discrete set of categories (instead non-uniform descriptions), we apply a modified Bayesian model and learn categories from a hand-labeled corpus. Each sample informs a policy tree, where each vertex is an action and each edge is a time delay between steps. Each sequence derived from the tree is labeled a Derived Cadence, whose score is the aggregate Engagement Score for each vertex in the sequence.

We leverage the Derived Cadences in a number of ways. We first validate the model by utilizing a Cadence similarity measure, which casts Cadences into Euclidean space to determine their nearness, and verify that our most successful users, in terms of growth and revenue, have generally higher Engagement Scores than our more poorly performing users. We then extract a number of insights, including the relative effectiveness of multichannel Cadences, the value of double taps, and general time delay structures as the length of Cadences grow.

References

[1] C Marlin Brown. *Human-Computer Interface Design Guidelines*. Intellect Books, 1998.

[2] Eva Ascarza, Peter S Fader, and Bruce GS Hardie. Marketing models for the customer-centric firm. In *Handbook of Marketing Decision Models*, pages 301–303. Springer, 2017.

[3] Theodore J Perkins. Reinforcement learning for POMDPs based on action values and stochastic optimization. In *AAAI/IAAI*, pages 199–204, 2002.

[4] Richard S Sutton and Andrew G Barto. *Introduction to Reinforcement Learning*. MIT Press, 1998.

[5] Leslie Pack Kaelbing, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *arXiv preprint cs/9605103*, 1996.

[6] Steven D Whitehead. Reinforcement learning for the adaptive control of perception and action. Technical report, DTIC Document, 1992.

[7] Donald Ervin Knuth. *The Art of Computer Programming*, volume 2. Pearson Education, 3rd edition, 1997.

[8] Jason D Rennie, Lawrence Shih, Jaime Teevan, and David R Karger. Tackling the poor assumptions of Naive Bayes text classifiers. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 616–623, 2003.

[9] Hanhuai Shan and Arindam Banerjee. Mixed-membership Naive Bayes Models. *Data Mining and Knowledge Discovery*, 23(1):1–62, 2011.