Adaptive Sequence Submodularity

Anonymous Author(s)
Affiliation
Address
email

Abstract

In many machine learning applications, one needs to interactively select a sequence of items (e.g., recommending movies based on a user's feedback) or make sequential decisions in a certain order (e.g., guiding an agent through a series of states). Not only do sequences already pose a dauntingly large search space, but we must also take into account past observations, as well as the uncertainty of future outcomes. Without further structure, finding an optimal sequence is notoriously challenging, if not completely intractable. In this paper, we view the problem of adaptive and sequential decision making through the lens of submodularity and propose an adaptive greedy policy with strong theoretical guarantees. Additionally, to demonstrate the practical utility of our results, we run experiments on Amazon product recommendation and Wikipedia link prediction tasks.

1 Introduction

The machine learning community has long recognized the importance of both sequential and adaptive decision making. The study of sequences has led to novel neural architectures such as LSTMs [26], which have been used in a variety of applications ranging from machine translation [51] to image captioning [55]. Similarly, the study of adaptivity has led to the establishment of some of the most popular subfields of machine learning including active learning [47] and reinforcement learning [52].

In this paper, we consider the optimization of problems where both sequences and adaptivity are integral part of the process. More specifically, we focus on problems that can be modeled as selecting a sequence of items, where each of these items takes on some (initially unknown) state. The idea is that the value of any sequence depends not only on the items selected and the order of these items but also on the states of these items.

Consider recommender systems as a running example. To start, the order in which we recommend items can be just as important as the items themselves. For instance, if we believe that a user will enjoy the Lord of the Rings franchise, it is vital that we recommend the movies in the proper order. If we suggest that the user watches the final installment first, she may end up completely unsatisfied with an otherwise excellent recommendation. Furthermore, whether it is explicit feedback (such as rating a movie on Netflix) or implicit feedback (such as clicking/not clicking on an advertisement), most recommender systems are constantly interacting with and adapting to each user. It is this feedback that allows us to learn about the states of items we have already selected, as well as make inferences about the states of items we have not selected yet.

Unfortunately, the expressive modeling power of sequences and adaptivity comes at a cost. Not only does optimizing over sequences instead of sets exponentially increase the size of the search space, but adaptivity also necessitates a probabilistic approach that further complicates the problem. Without further assumptions, even approximate optimization is infeasible. As a result, we address this challenge from the perspective of *submodularity*, an intuitive diminishing returns condition that appears in a broad scope of different areas, but still provides enough structure to make the problem tractable.

Submitted to 33rd Conference on Neural Information Processing Systems (NeurIPS 2019). Do not distribute.

Research on submodularity, which itself has been a burgeoning field in recent years, has seen comparatively little focus on sequences and adaptivity. This is especially surprising because many problems that are commonly modeled under the framework of submodularity, such as recommender systems [21, 58] and crowd teaching [48], stand to benefit greatly from these concepts.

While the lion's share of existing research in submodularity has focused on *sets*, a few recent lines of work extend the concept of submodularity to *sequences*. Tschiatschek et al. [53] were the first to consider *sequence submodularity* in the general graph-based setting that we will follow in this paper. They presented an algorithm with theoretical guarantees for directed acyclic graphs, while Mitrovic et al. [44] developed a more comprehensive algorithm that provides theoretical guarantees for general hypergraphs.

In their experiments, both of these works showed that modeling the problem as sequence submodular (as opposed to set submodular) gave noticeable improvements. Their applications could benefit even further from the aforementioned notions of adaptivity, but the existing theory behind sequence submodularity simply cannot model the problems in this way. While adaptive *set* submodularity has been studied extensively [12, 20, 22, 24], these approaches still fail to capture order dependencies.

Alaei and Malekian [1] and Zhang et al. [59] also consider sequence submodularity (called string-submodularity in some works), but they use a different definition, which is based on subsequences instead of graphs. On the other hand, Li and Milenkovic [38] have considered the interaction of graphs and submodularity, but not in the context of sequences.

Other Related Work Amongst many other applications, submodularity has also been used for variable selection [35], data summarization [31, 39, 43], sensor placement [33], neural network interpretability [14], network inference [23], and influence maximization in social networks [28]. Submodularity has also been studied extensively in a wide variety of settings, including distributed and scalable optimization [5–7, 17, 18, 37, 42, 43], streaming algorithms [3, 10, 11, 19, 34, 45, 46], robust optimization [9, 27, 36, 49, 54], weak submodularity [13, 15, 16, 30], and continuous submodularity [2, 4, 25, 50, 57].

Our Contributions The main contributions of our paper are presented in the following sections:

- In Section 2, we introduce our framework of *adaptive sequence submodularity*, which brings tractability to problems that include both sequences and adaptivity.
- In Section 3, we present our algorithm for adaptive sequence submodular maximization. We present theoretical guarantees for our approach and we elaborate on the necessity of our novel proof techniques. We also show that these techniques simultaneously improve the state-of-the-art bounds for the problem of sequence submodularity by a factor of $\frac{e}{e-1}$. Furthermore, we argue that any approximation guarantee must depend on the structure of the underlying graph unless the exponential time hypothesis is false.
- In Section 4, we use datasets from Amazon and Wikipedia to compare our algorithm against existing sequence submodular baselines, as well as state-of-the-art deep learning-based approaches.

2 Adaptive Sequence Submodularity

As discussed above, sequences and adaptivity are an integral part of many real-world problems. This means that many real-world problems can be modeled as selecting a sequence σ of items from a ground set V, where each of these items takes on some (initially unknown) state $o \in O$. A particular mapping of items to states is known as a **realization** ϕ, and we assume there is some unknown distribution $p(\phi)$ that governs these states.

For example in movie recommendation, the set of all movies is our ground set V and our goal is to select a sequence of movies that a particular user will enjoy. If we recommend a movie $v_i \in V$ and the user likes it, we place v_i in state 1 (i.e. $o_i = 1$). If not, we put it into state 0. Naturally, the value of a movie should be higher if the user liked it, and lower if she did not.

Formally, we want to select a sequence σ that maximizes $f(\sigma, \phi)$, where $f(\sigma, \phi)$ is the value of sequence σ under realization ϕ. However, ϕ is initially unknown to us and the state of each item in the sequence is revealed to us only after we select it. In fact, even if we knew ϕ perfectly, the set of

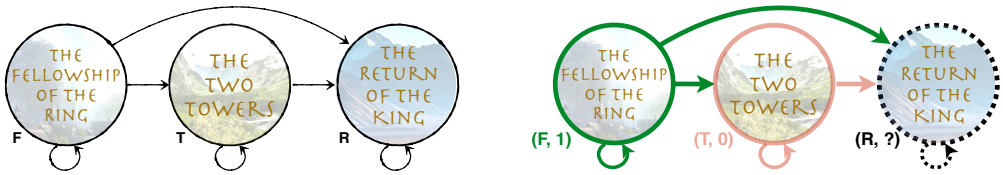


Figure 1: (a) shows an underlying graph for a movie recommendation problem. The vertices are movies and edges denote the additional value of watching certain movies in certain orders. (b) extends this to the adaptive case, where both the vertices and the edges take on a state. The user has reported that she liked the Fellowship of the Ring (so it is placed in state 1), but she did not like The Two Towers (so it is placed in state 0). The state of the last movie is still unknown. In this example, the state of an edge is equal to the state of its starting vertex.

all sequences poses an intractably large search space. From an optimization perspective, this problem is hopeless without further structural assumptions.

Our first step towards taming this problem is to follow the work of Tschiatschek et al. [53] and assume that the value of a sequence can be defined using a graph. Concretely, we have a directed graph $G = (V, E)$, where each item in our ground set is represented as a vertex $v \in V$, and the edges encode the additional value intrinsic to picking certain items in certain orders. Mathematically, selecting a sequence of items σ will induce a set of edges $E(\sigma)$:

$$E(\sigma) = \big\{(\sigma_i, \sigma_j) \mid (\sigma_i, \sigma_j) \in E, i \leq j\big\}.$$

For example, consider the graph in Figure 1a and consider the sequence $\sigma_A = [F, T]$ where the user watched The Fellowship of the Ring, and then The Two Towers, as well as the sequence $\sigma_B = [T, F]$ where the user watched the same two movies but in the opposite order.

$$E(\sigma_A) = E\big([F, T]\big) = \big\{(F, F), (T, T), (F, T)\big\}$$
$$E(\sigma_B) = E\big([T, F]\big) = \big\{(T, T), (F, F)\big\}$$

Using the self-loops, this graph encodes the fact that there is certainly some intrinsic value to watching these movies regardless of the order. On the other hand, the edge (F, T) encodes the fact that watching The Fellowship of the Ring before The Two Towers will bring additional value to the viewer, and this edge is only induced if the movies appear in the correct order in the sequence.

With this graph based set-up, however, we run into issues when it comes to adaptivity. In particular, the states of items naturally translate to states for the vertices, but it is not clear how to extend adaptivity to the *edges*. We tackle this challenge by assigning a state $q \in Q$ to each edge strictly as a function of the states of its endpoints. That is, similarly to how a sequence σ induces a set of edges $E(\sigma)$, a realization ϕ for the states of the vertices induces a realization ϕ^E for the states of the edges. As we will discuss later, the analysis for this approach will necessitate some novel proof techniques, but the resulting framework is very flexible and it allows us to fully redefine the adaptive sequence problem in terms of the underlying graph:

$$f(\sigma, \phi) = h\big(E(\sigma), \phi^E\big) \text{ where } \sigma \text{ induces } E(\sigma) \text{ and } \phi \text{ induces } \phi^E.$$

The last necessary ingredient to bring tractability to this problem is submodularity. In particular, we will assume that $h\big(E(\sigma), \phi^E\big)$ is *weakly adaptive set submodular*. This is a relaxed version of standard adaptive set submodularity that can model an even larger variety of problems, and it is a natural fit for the applications we consider in this paper.

In order to formally define weakly-adaptive submodularity, we need a bit more terminology. To start, we define a **partial realization** ψ to be a mapping for only some subset of items (i.e., the states of the remaining items are unknown). For notational convenience, we define the domain of ψ, denoted $dom(\psi)$, to be the list of items v for which the state of v is known. We say that ψ is a **subrealization** of ψ', denoted $\psi \subseteq \psi'$, if $dom(\psi) \subseteq dom(\psi')$ and they are equal everywhere in the domain of ψ. Intuitively, if $\psi \subseteq \psi'$, then ψ' has all the same information as ψ, and potentially more.

Given a partial realization ψ, we define the marginal gain of a set A as

$$\Delta(A \mid \psi) = \mathbb{E}\Big[h\big(dom(\psi) \cup A, \phi\big) - h\big(dom(\psi), \phi\big) \mid \psi\Big],$$

where the expectation is taken over all the full realizations ϕ such that $\psi \subseteq \phi$. In other words, we condition on the states given by the partial realization ψ, and then we take the expectation across all possibilities for the remaining states.

Definition 1. *A function $h : 2^E \times Q^E \to \mathbb{R}_{\geq 0}$ is **weakly adaptive set submodular** with parameter γ if for all sets $A \subseteq E$ and for all $\psi \subseteq \psi'$ we have:*

$$\Delta(A \mid \psi') \leq \frac{1}{\gamma} \cdot \sum_{e \in A} \Delta(e \mid \psi).$$

This notion is a natural generalization of weak submodular functions [13] to adaptivity. The primary difference is that we condition on subrealizations instead of just sets because we need to account for the states of items. Note that in the context of this paper h is a function on the edges, so we will condition on subrealizations of the edges ψ^E. However, these concepts apply more generally to functions on any set and state spaces, so we use ψ in the formal definitions.

Definition 2. *A function $h : 2^E \times Q^E \to \mathbb{R}_{\geq 0}$ is **adaptive monotone** if $\Delta(e \mid \psi) \geq 0$ for all partial realizations ψ. That is, the conditional expected marginal benefit of any element is non-negative.*

Figure 1b is designed to help clarify these concepts. It includes the same graph as Figure 1a, but now we can receive feedback from the user. If we recommend a movie and the user likes it, we put the corresponding vertex in state 1 (green in the image). Otherwise, we put the vertex in state 0 (red in the image). Vertices whose states are still unknown are denoted by a dotted black line.

Next, in our example, we need to define a state for each edge in terms of the states of its endpoints. In this case, we will define the state of each edge to be equal to the state of its start point. In Figure 1b, the user liked The Fellowship of the Ring, which puts edges (F, F), (F, T), and (F, R) in state 1 (green). She did not like The Two Towers, so edges (T, T) and (T, R) are in state 0 (red), and we do not know the state for The Return of the King, so the state of (R, R) is also unknown. We call this partial realization ψ_1 for the vertices, and the induced partial realization for the edges ψ_1^E.

Suppose our function h counts all induced edges that are in state 1. Furthermore, let us simply assume that any unknown vertex is equally likely to be in state 0 or state 1. This means that the self-loop (R, R) is also equally likely to be in either state 0 or state 1. Therefore, $\Delta\big((R, R) \mid \psi_1^E\big) = \frac{1}{2} \times 0 + \frac{1}{2} \times 1 = \frac{1}{2}$.

On the other hand, consider the edge (F, R). Under ψ_1, we know F is in state 1, which means (F, R) is also in state 1, and thus, $\Delta\big((F, R) \mid \psi_1^E\big) = 1$. However, if we consider a subrealization $\psi_2 \subseteq \psi_1$ where we do not know the state of F, then it is equally likely to be in either state and $\Delta\big((F, R) \mid \psi_2^E\big) = \frac{1}{2} \times 0 + \frac{1}{2} \times 1 = \frac{1}{2}$. Therefore, for this simple function we know that $\gamma \leq 0.5$.

3 Adaptive Sequence-Greedy Policy and Theoretical Results

In this section, we introduce our Adaptive Sequence-Greedy policy and present its theoretical guarantees. We first formally define **weakly adaptive sequence submodularity**.

Definition 3. *A function $f(\sigma, \phi)$ defined over a graph $G(V, E)$ is **weakly adaptive sequence submodular** if $f(\sigma, \phi) = h\big(E(\sigma), \phi^E\big)$ where a sequence σ of vertices in V induces a set of edges $E(\sigma)$, realization ϕ induces ϕ^E, and the function h is weakly adaptive set submodular. Note that if h is adaptive monotone, then f is also adaptive monotone.*

Formally, a policy π is an algorithm that builds a sequence of k vertices by seeing which states have been observed at each step, then deciding which vertex should be chosen and observed next. If $\sigma_{\pi,\phi}$ is the sequence returned by policy π under realization ϕ, then we write the expected value of π as:

$$f_{\text{avg}}(\pi) = \mathbb{E}\big[f(\sigma_{\pi,\phi}, \phi)\big] = \mathbb{E}\Big[h\big(E(\sigma_{\pi,\phi}), \phi^E\big)\Big]$$

where again the expectation is taken over all possible realizations ϕ.

Our Adaptive Sequence Greedy policy π (Algorithm 1) starts with an empty sequence σ. Throughout the policy, we define ψ_σ to be the partial realization for the vertices in σ. In turn this gives us the partial realization ψ_σ^E for the induced edges.

At each step, we define the valid set of edges $\mathcal{E}$ to be the edges whose endpoint is not already in σ. The main idea of our policy is that, at each step, we select the valid edge $e \in \mathcal{E}$ with the highest

expected value $\Delta(e \mid \psi_\sigma^E)$. For each such edge, the endpoints that are not already in the sequence σ are concatenated ($\oplus$ means concatenate) to the end of σ, and their states are observed (updating ψ_σ).

Algorithm 1 Adaptive Sequence Greedy Policy π

Input: Directed graph $G = (V, E)$, weakly adaptive sequence submodular $f(\sigma, \phi) = h\big(E(\sigma), \phi^E\big)$, and cardinality constraint k
Let $\sigma \leftarrow ()$
while $|\sigma| \leq k - 2$ **do**
  $\mathcal{E} = \{e_{ij} \in E \mid v_j \notin \sigma\}$
  if $\mathcal{E} \neq \varnothing$ **then**
    $e_{ij} = \arg\max_{e \in \mathcal{E}} \Delta(e \mid \psi_\sigma^E)$
    if $v_i = v_j$ **or** $v_i \in \sigma$ **then**
      $\sigma = \sigma \oplus v_j$ and observe state of v_j
    else
      $\sigma = \sigma \oplus v_i \oplus v_j$ and observe states of v_i, v_j
    end if
  else
    break
  end if
end while
Return σ

Theorem 1. *For adaptive monotone and weakly adaptive sequence submodular function f, the Adaptive Sequence Greedy policy π represented by Algorithm 1 achieves*

$$f_{avg}(\pi) \geq \frac{\gamma}{2d_{\text{in}} + \gamma} \cdot f_{avg}(\pi^*),$$

where γ is the weakly adaptive submodularity parameter, π^* is the policy with the highest expected value and d_{in} is the largest in-degree of the input graph G.

As discussed by Mitrovic et al. [44], using a hypergraph H instead of a normal graph G allows us to encode more intricate relationships between the items. For example, in Figure 1a, the edges only encode pairwise relationships. However, there may be relationships between larger groups of items that we want to encode explicitly. For instance, if included, the value of a hyperedge (F, T, R) in Figure 1a would explicitly encode the value of watching The Fellowship of the Ring, followed by watching The Two Towers, and then concluding with The Return of the King.

We can also extend our policy to general hypergraphs (see Algorithm 2 in Appendix B.3). Theorem 2 guarantees the performance of our proposed policy for hypergraphs.

Theorem 2. *For adaptive monotone and weakly adaptive sequence submodular function f, the policy π' represented by Algorithm 2 achieves*

$$f_{avg}(\pi') \geq \frac{\gamma}{rd_{\text{in}} + \gamma} \cdot f_{avg}(\pi^*),$$

where γ is the weakly adaptive submodularity parameter, π^* is the policy with the highest expected value and r is the size of the largest hyperedge in the input hypergraph.

In our proofs, we have to handle the sequential nature of picking items and the revelation of states in a combined setting. Unfortunately, the existing proof methods for sequence submodular maximization are not linear enough to allow for the use of the linearity of expectation that captures the stochasticity of the states. For this reason, we develop a novel analysis technique to guarantee the performance of our algorithms. Surprisingly, these new techniques improve the theoretical guarantees of the non-adaptive Sequence-Greedy and Hyper Sequence-Greedy [44] by a factor of $\frac{e}{e-1}$. **Proofs for both theorems are given in Appendix B.**

General Unifying Framework One more theoretical point we want to highlight is that weakly adaptive sequence submodularity provides a general unifying framework for a variety of common submodular settings including, adaptive submodularity, weak submodularity, sequence submodularity, and classical set submodularity. If we have $\gamma = 1$ and the state of all vertices is deterministic, then we have sequence submodularity. Conversely, if the vertex states are unknown, but our graph only

has self-loops, then we have weakly adaptive set submodularity (and correspondingly adaptive set submodularity if $\gamma = 1$). Lastly, if we have a graph with only self-loops, full knowledge of all states, and $\gamma = 1$, then we recover the original setting of classical set submodularity.

Tightness of Theoretical Results We acknowledge that the constant factor approximation we present depends on the maximum in-degree. While ideally the theoretical bound would be completely independent of the structure of the graph, we argue here that such a dependence is likely necessary.

Indeed, getting a dependence better than $O(n^{1/4})$ in the approximation factor (where n is the total number of items) would improve the state-of-the-art algorithm for the very well-studied densest k subgraph problem (DkS) [8, 32]. Moreover, if we could get an approximation that is completely independent of the structure of the graph, then the exponential time hypothesis would be proven false[1]. In fact, even an almost polynomial approximation would break the exponential time hypothesis [40]. Next, we formally state this hardness relationship. **The proof is given in Appendix C**.

Theorem 3. *Assuming the exponential time hypothesis is correct, there is no algorithm that approximates the optimal solution for the (adaptive) sequence submodular maximization problem within a $n^{1/(\log\log n)^c}$ factor, where n is the total number of items and $c > 0$ is a universal constant independent of n.*

4 Experimental Results

4.1 Amazon Product Recommendation

Using the Amazon Video Games review dataset [41], we consider the task of recommending products to users. In particular, given the first g products that the user has purchased, we want to predict the next k products that she will buy. Full experimental details are given in Appendix D.1. **Dataset and code are attached in the supplementary material.**

We start by using the training data to build a graph $G = (V, E)$, where V is the set of all products and E is the set of edges between these products. The weight of each edge, w_{ij}, is defined to be the conditional probability of purchasing product j given that the user has previously purchased product i. There are also self-loops with weight w_{ii} that represent the fraction of users that purchased product i.

We define the state of each edge (i, j) to be equal to the state of product i. The intuitive idea is that edge (i, j) encodes the value of purchasing product j after already having purchased product i. Therefore, if the user has definitely purchased i (i.e., product i is in state 1), then they should receive the full value of w_{ij}. On the other hand, if she has definitely not purchased i (i.e., product i is in state 0), then edge (i, j) provides no value. Lastly, if the state of i is unknown, then the expected gain of edge (i, j) is discounted by w_{ii}, the value of the self-loop on i, which can be viewed as a simple estimate for the probability of the user purchasing product i. See Figure 2a for a small example.

We use a probabilistic coverage utility function as our monotone weakly-adaptive set submodular function h. Mathematically,

$$h(E_1) = \sum_{j \in V} \Big[1 - \prod_{(i,j) \in E_1} (1 - w_{ij})\Big],$$

where $E_1 \subseteq E$ is the subset of edges that are in state 1.

We compare the performance of our Adaptive Sequence-Greedy policy against Sequence-Greedy from Mitrovic et al. [44], the existing sequence submodularity baseline that does not consider states. To give further context for our results, we compare against Frequency, a naive baseline that ignores sequences and adaptivity and simply outputs the k most popular products.

We also compare against a set of deep learning-based approaches (see Appendix D.3 for full details). In particular, we implement adaptive and non-adaptive versions of both a regular Feed Forward Neural Network and an LSTM. The adaptive version will update its inputs after every prediction to reflect whether or not the user liked the recommendation. Conversely, the non-adaptive version will simply make k predictions using just the original input.

We use two different measures to compare the various algorithms. The first is the **Accuracy Score**, which simply counts the number of recommended products that the user indeed ended up purchasing.

[1] If the exponential time hypothesis is true it would imply that P $\neq$ NP, but it is a stronger statement.

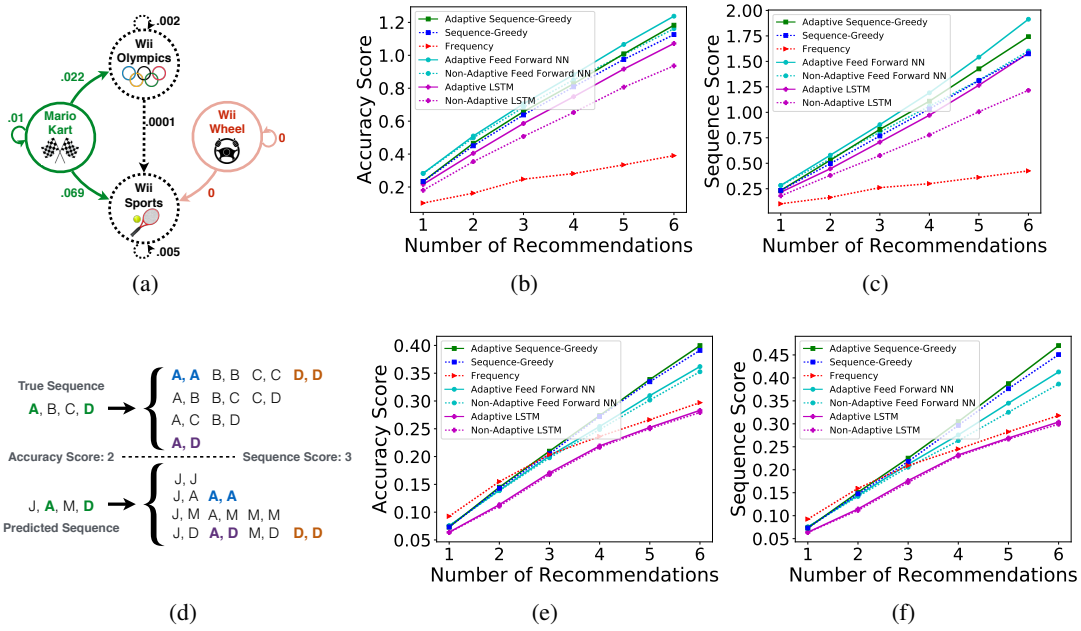


Figure 2: (a) shows a small subset of the underlying graph with states for a particular user. (b) and (c) show our results on the Amazon product recommendation task. In all these graphs, the number of given products g is 4. (d) gives an example illustrating the difference between the two performance measures. (e) and (f) show our results on the same task, but using only 1% of the available training to show that our algorithm outperforms deep learning-based approaches in data scarce environments.

While this is a sensible measure, it does not explicitly consider the order of the sequence. Therefore, we also consider the **Sequence Score**, which is a measure based on the Kendall-Tau distance [29]. In short, this measure counts the number of ordered pairs that appear in both the predicted sequence and the true sequence. Figure 2d gives an example comparing the two measures.

Figures 2b and 2c show the performance of the various algorithms using the accuracy score and sequence score, respectively. These results highlight the importance of adaptivity as the adaptive algorithms consistently outperform their non-adaptive counterparts under both scoring regimes. Notice that in both cases, as the number of recommendations increases, our proposed Adaptive Sequence-Greedy policy is outperformed only by the Adaptive Feed Forward Neural Network. Although LSTMs are generally considered better for sequence data than vanilla feed-forward networks, we think it is a lack of data that causes them to perform poorly in our experiments.

Another observation, which fits the conventional wisdom, is that deep learning-based approaches can perform well when there is a lot of data. However, when the data is scarce, we see that the Sequence-Greedy based approaches outperform the deep learning-based approaches. Figures 2e and 2f simulate a data-scarce environment by using only 1% of the available data as training data. Note that the difference between the adaptive algorithms and their non-adaptive counterparts is less obvious in this setting because the adaptive algorithms use correct guesses to improve future recommendations, but the data scarcity makes it difficult to make a correct guess in the first place.

Aside from competitive accuracy and sequence scores, the Adaptive Sequence-Greedy algorithm provides several advantages over the neural network-based approaches. From a theoretical perspective, the Adaptive Sequence-Greedy algorithm has provable guarantees on its performance, while little is known about the theoretical performance of neural networks. Furthermore, the decisions made by the Adaptive Sequence-Greedy algorithm are easily interpretable and understandable (it is just picking the edge with the highest expected value), while neural networks are generally a black-box. On a similar note, Adaptive Sequence-Greedy may be preferable from an implementation perspective because it does not require any hyperparameter tuning. It is also more robust to changing inputs in the sense that we can easily add another product and its associated edges to our graph, but adding another product to the neural network requires changing the entire input and output structure, and thus, generally necessitates retraining the entire network.

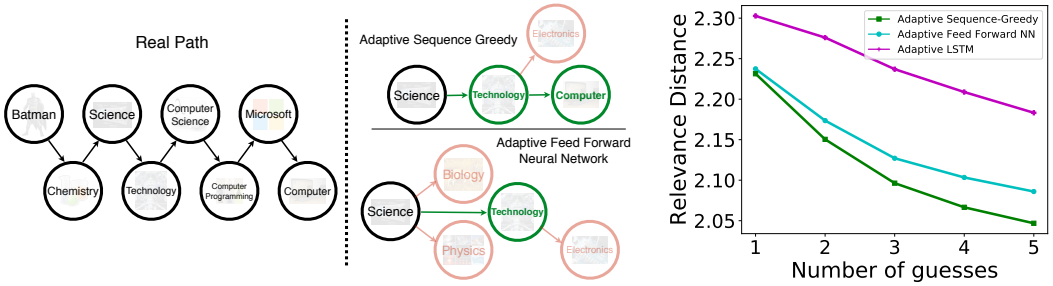
4.2 Wikipedia Link Prediction

Using the Wikispeedia dataset [56], we consider users who are surfing through Wikipedia towards some target article. Given a sequence of articles the user has previously visited, we want to guide her to the page she is trying to reach. Since different pages have different valid links, the order of pages we visit is critical to this task. Formally, given the first $g = 3$ pages each user visited, we want to predict which page she is trying to reach by making a series of suggestions for which link to follow.

In this case, we have $G = (V, E)$, where V is the set of all pages and E is the set of existing links between pages. Similarly to before, the weight w_{ij} of an edge $(i, j) \in E$ is the probability of moving to page j given that the user is currently at page i. In this case, there are no self-loops as we assume we can only move using links, and thus we cannot jump to random pages. We again define two states for the nodes: 1 if the user definitely visits this page and 0 if the user does *not* want to visit this page.

This application highlights the importance of adaptivity because the non-adaptive sequence submodularity framework cannot model this problem properly. This is because the Sequence-Greedy algorithm is free to choose any edge in the underlying graph, so there is no way to force the algorithm to pick a link that is connected to the user's current page. On the other hand, with Adaptive Sequence-Greedy, we can use the states to penalize invalid edges, and thus force the algorithm to select only links connected to the user's current page. Similarly, we only have the adaptive versions of the deep learning baselines because we need information about our current page in order to construct a valid path (Appendix D.3 gives a more detailed explanation).

Figure 3a shows an example of predicted paths, while Figure 3b shows our quantitative results. More detail about the relevance distance metric is given in Appendix D.2, but the idea is that the it measures the relevance of the final output page to the true target page (a lower score indicates a higher relevance). The main observation here is that the Adaptive Sequence Greedy algorithm actually outperforms the deep-learning based approaches. The main reason for this discrepancy is likely a lack of data as we have 619 pages to choose from and only 7,399 completed search paths.



(a) (b)

Figure 3: (a) The left side shows the real path a user followed from *Batman* to *Computer*. Given the first three pages, the right side shows the path predicted by Adaptive Sequence Greedy versus a deep learning-based approach. Green shows correct guesses that were followed, while red shows incorrect guesses that were not pursued further. (b) shows the overall performance of the various approaches.

5 Conclusion

In this paper we introduced adaptive sequence submodularity, a general framework for bringing tractability to the broad class of optimization problems that consider both sequences and adaptivity. We presented Adaptive Sequence-Greedy—a general policy for optimizing weakly adaptive sequence submodular functions. In addition to providing a provable theoretical guarantee for our algorithm (as well as a discussion about the tightness of this result), we also evaluated its performance on an Amazon product recommendation task and a Wikipedia link prediction task. Not only does our Adaptive Sequence-Greedy policy exhibit competitive performance with the state-of-the-art, but it also provides several notable advantages, including interpretability, ease of implementation, and robustness against both data scarcity and input adjustments.

References

[1] Saeed Alaei and Azarakhsh Malekian. Maximizing sequence-submodular functions and its application to online advertising. *arXiv preprint arXiv:1009.4153*, 2010.

[2] Francis Bach. Submodular functions: from discrete to continous domains. *arXiv preprint arXiv:1511.00394*, 2015.

[3] Ashwin Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Streaming submodular maximization: Massive data summarization on the fly. In *Knowledge Discovery and Data Mining (KDD)*, 2014.

[4] Wenruo Bai, William Stafford Noble, and Jeff A. Bilmes. Submodular Maximization via Gradient Ascent: The Case of Deep Submodular Functions. In *Advances in Neural Information Processing Systems*, pages 7989–7999, 2018.

[5] Eric Balkanski and Yaron Singer. The adaptive complexity of maximizing a submodular function. In *Symposium on Theory of Computing, STOC*, pages 1138–1151, 2018.

[6] Eric Balkanski, Aviad Rubinstein, and Yaron Singer. An Exponential Speedup in Parallel Running Time for Submodular Maximization without Loss in Approximation. In *Symposium on Discrete Algorithms (SODA)*, pages 283–302, 2019.

[7] Rafael Barbosa, Alina Ene, Huy Nguyen, and Justin Ward. The power of randomization: Distributed submodular maximization on massive datasets. In *International Conference on Machine Learning (ICML)*, 2015.

[8] Aditya Bhaskara, Moses Charikar, Eden Chlamtac, Uriel Feige, and Aravindan Vijayaraghavan. Detecting high log-densities: an $O(n^{1/4})$ approximation for densest k-subgraph. In *Symposium on Theory of Computing (STOC)*, pages 201–210, 2010.

[9] Ilija Bogunovic, Slobodan Mitrovic, Jonathan Scarlett, and Volkan Cevher. Robust Submodular Maximization: A Non-Uniform Partitioning Approach. In *International Conference on Machine Learning (ICML)*, 2017.

[10] Niv Buchbinder, Moran Feldman, and Roy Schwartz. Online submodular maximization with preemption. In *Symposium on Discrete Algorithms (SODA)*, 2015.

[11] Amit Chakrabarti and Sagar Kale. Submodular maximization meets streaming: Matchings, matroids, and more. *IPCO*, 2014.

[12] Yuxin Chen and Andreas Krause. Near-optimal Batch Mode Active Learning and Adaptive Submodular Optimization. In *International Conference on Machine Learning (ICML)*, 2013.

[13] Abhimanyu Das and David Kempe. Submodular meets Spectral: Greedy Algorithms for Subset Selection, Sparse Approximation and Dictionary Selection. In *International Conference on Machine Learning (ICML)*, pages 1057–1064, 2011.

[14] Ethan Elenberg, Alexandros Dimakis, Moran Feldman, and Amin Karbasi. Streaming Weak Submodularity: Interpreting Neural Networks on the Fly. In *Advances in Neural Information Processing Systems*, 2018.

[15] Ethan R. Elenberg, Rajiv Khanna, Alexandros G. Dimakis, and Sahand N. Negahban. Restricted strong convexity implies weak submodularity. *CoRR*, abs/1612.00804, 2016.

[16] Ethan R. Elenberg, Alexandros G. Dimakis, Moran Feldman, and Amin Karbasi. Streaming Weak Submodularity: Interpreting Neural Networks on the Fly. In *Advances in Neural Information Processing Systems*, pages 4047–4057, 2017.

[17] Alina Ene and Huy L. Nguyen. Submodular Maximization with Nearly-optimal Approximation and Adaptivity in Nearly-linear Time. In *Symposium on Discrete Algorithms (SODA)*, pages 274–282, 2019.

[18] Matthew Fahrbach, Vahab S. Mirrokni, and Morteza Zadimoghaddam. Submodular Maximization with Nearly Optimal Approximation, Adaptivity and Query Complexity. In *Symposium on Discrete Algorithms (SODA)*, pages 255–273, 2019.

[19] Moran Feldman, Amin Karbasi, and Ehsan Kazemi. Do Less, Get More: Streaming Submodular Maximization with Subsampling. In *Advances in Neural Information Processing Systems*, pages 730–740, 2018.

[20] Kaito Fujii and Shinsaku Sakaue. Beyond Adaptive Submodularity: Approximation Guarantees of Greedy Policy with Adaptive Submodularity Ratio. In *International Conference on Machine Learning (ICML)*, 2019.

[21] Victor Gabillon, Branislav Kveton, Zheng Wen, Brian Eriksson, and S. Muthukrishnan. Adaptive submodular maximization in bandit settings. In *Advances in Neural Information Processing Systems*, 2013.

[22] Daniel Golovin and Andreas Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, 42: 427–486, 2011.

[23] Manuel Gomez Rodriguez, Jure Leskovec, and Andreas Krause. Inferring networks of diffusion and influence. In *Knowledge Discovery and Data Mining (KDD)*, 2010.

[24] Alkis Gotovos, Amin Karbasi, and Andreas Krause. Non-monotone adaptive submodular maximization. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2015.

[25] S. Hamed Hassani, Mahdi Soltanolkotabi, and Amin Karbasi. Gradient Methods for Submodular Maximization. In *Advances in Neural Information Processing Systems*, pages 5843–5853, 2017.

[26] Sepp Hochreiter and Jrgen Schmidhuber. Long short-term memory. In *Neural Computation*, 1997.

[27] Ehsan Kazemi, Morteza Zadimoghaddam, and Amin Karbasi. "Scalable Deletion-Robust Submodular Maximization: Data Summarization with Privacy and Fairness Constraints. In *International Conference on Machine Learning (ICML)*, 2018.

[28] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Knowledge Discovery and Data Mining (KDD)*, 2003.

[29] Maurice Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

[30] Rajiv Khanna, Ethan R. Elenberg, Alexandros G. Dimakis, Sahand N. Negahban, and Joydeep Ghosh. Scalable Greedy Feature Selection via Weak Submodularity. In *Artificial Intelligence and Statistics (AISTATS)*, pages 1560–1568, 2017.

[31] Katrin Kirchhoff and Jeff Bilmes. Submodularity for data selection in statistical machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[32] Guy Kortsarz and David Peleg. On Choosing a Dense Subgraph (Extended Abstract). In *Foundations of Computer Science (FOCS)*, pages 692–701, 1993.

[33] A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. In *Journal of Machine Learning Research*, volume 9, 2008.

[34] Andreas Krause and Ryan G Gomes. Budgeted nonparametric learning from data streams. In *International Conference on Machine Learning (ICML)*, 2010.

[35] Andreas Krause and Carlos Guestrin. Near-optimal nonmyopic value of information in graphical models. In *Uncertainty in Artificial Intelligence (UAI)*, 2005.

[36] Andreas Krause, H Brendan McMahon, Carlos Guestrin, and Anupam Gupta. Robust submodular observation selection. *Journal of Machine Learning Research*, 2008.

[37] Ravi Kumar, Benjamin Moseley, Sergei Vassilvitskii, and Andrea Vattani. Fast greedy algorithms in mapreduce and streaming. In *Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2013.

[38] Pan Li and Olgica Milenkovic. Inhomogeneous hypergraph clustering with applications. *Advances in Neural Information Processing Systems*, 2017.

[39] Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In *Association for Computational Linguistics (ACL)*, 2011.

[40] Pasin Manurangsi. Almost-polynomial Ratio ETH-hardness of Approximating Densest K-subgraph. In *Symposium on Theory of Computing (STOC)*, pages 954–961, 2017.

[41] J. McAuley, C. Targett, J. Shi, and A. van den Hengel. Image-based recommendations on styles and substitutes. In *SIGIR Conference on Research and Development in Information Retrieval*, 2015.

[42] Vahab Mirrokni and Morteza Zadimoghaddam. Randomized composable core-sets for distributed submodular maximization. In *Symposium on Theory of Computing (STOC)*, 2015.

[43] Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. Distributed submodular maximization: Identifying representative elements in massive data. In *Advances in Neural Information Processing Systems*, 2013.

[44] Marko Mitrovic, Moran Feldman, Andreas Krause, and Amin Karbasi. Submodularity on hypergraphs: From sets to sequences. *Artificial Intelligence and Statistics (AISTATS)*, 2018.

[45] Marko Mitrovic, Ehsan Kazemi, Morteza Zadimoghaddam, and Amin Karbasi. Data summarization at scale: A two-stage submodular approach. In *International Conference on Machine Learning (ICML)*, 2018.

[46] Ashkan Norouzi-Fard, Jakub Tarnawski, Slobodan Mitrovic, Amir Zandieh, Aidasadat Mousavifar, and Ola Svensson. Beyond 1/2-Approximation for Submodular Maximization on Massive Data Streams. In *International Conference on Machine Learning (ICML)*, pages 3826–3835, 2018.

[47] Burr Settles. Active learning. In *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 2012.

[48] Adish Singla, Ilija Bogunovic, Gábor Bartók, Amin Karbasi, and Andreas Krause. Near-optimally teaching the crowd to classify. In *International Conference on Machine Learning (ICML)*, 2014.

[49] Matthew Staib and Stefanie Jegelka. Robust Budget Allocation via Continuous Submodular Functions. In *International Conference on Machine Learning (ICML)*, pages 3230–3240, 2017.

[50] Matthew Staib, Bryan Wilder, and Stefanie Jegelka. Distributionally Robust Submodular Maximization. *CoRR*, abs/1802.05249, 2018.

[51] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, 2014.

[52] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction, 2nd edition. In *MIT Press*, 2018.

[53] Sebastian Tschiatschek, Adish Singla, and Andreas Krause. Selecting sequences of items via submodular maximization. In *AAAI Conference on Artificial Intelligence*, 2017.

[54] Vasileios Tzoumas, Konstantinos Gatsis, Ali Jadbabaie, and George J. Pappas. Resilient monotone submodular function maximization. In *Conference on Decision and Control (CDC)*, 2017.

[55] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[56] Robert West, Joelle Pineau, and Doina Precup. An Online Game for Inferring Semantic Distances between Concepts. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2009.

[57] Laurence A. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 1982.

[58] Yisong Yue and Carlos Guestrin. Linear submodular bandits and its application to diversified retrieval. In *Advances in Neural Information Processing Systems*, 2011.

[59] Zhenliang Zhang, Edwin K. P. Chong, Ali Pezeshki, and William Moran. String Submodular Functions with Curvature Constraints. *IEEE Transactions on Automatic Control*, 61(3):601–616, 2016.

A Table of Notations

Table 1

E	Ground set of elements.
$e \in E$	An individual element from E.
ϕ	A realization, i.e., a function from elements to states.
ψ	A partial realization to encoding the current set of observations.
$\text{dom}(\psi)$	Domain of a partial realization ψ is defined as $\text{dom}(\psi) = \{e : \exists o \text{ .s.t. } (o, e) \in \psi\}$.
Φ, Ψ	A random realization and a random partial realization, respectively.
$\sim$	For a realization ϕ and a partial realization ψ: $\phi \sim \psi$ means $\psi(e) = \phi(e)$ for all $e \in \text{dom}(\psi)$.
$p(\phi)$	The probability distribution on realizations.
$p(\phi \mid \psi)$	The conditional distribution on realizations: $p(\phi \mid \psi) \triangleq \Pr[\Phi = \phi \mid \Phi \sim \psi]$.
π	A policy, which maps partial realizations to items.
$E(\pi, \phi)$	The set of all edges induced by π when run under realization ϕ.
h	An objective function of type $h : 2^E \times O^E \to \mathbb{R}_{\geq 0}$.
$\Delta(e \mid \psi)$	The conditional expected marginal benefit of e conditioned on ψ.
k	Budget on the number of selected items.

B Proofs

In this section, we prove Theorems 1 and 2. Towards this goal, we first state some necessary definitions and notations, and present a few results regarding weakly adaptive submodular functions.

B.1 Weakly Adaptive Sequence Submodular

Notation The random variable Φ denotes a random realization with respect to the distribution $p(\Phi = \phi)$ over the items (or equivalently vertices of the graph).[2] For a set A, its partial realization (i.e., items in A and their corresponding states) is shown by $\psi_A = \{(e, O(e)) \mid e \in A\}$, where $O(e)$ gives the state of e. For a partial realization ψ, we define $\text{dom}(\psi) = \{e : \exists\, o \text{ s.t. } (o, e) \in \psi\}$. We use Ψ_A to denote a random partial realization over a set A. Note that the distribution of random variable Ψ_A is uniquely defined by the distribution of random variable Φ. A partial realization ψ is consistent with a realization ϕ (we write $\phi \sim \psi$) if they are equal, i.e., they are in the same state, everywhere in the domain of ψ. For the ease of notation, we define $h(\psi) \triangleq h(\text{dom}(\psi), O(\psi))$, where $O(\psi)$ is the state of items in the realization ψ. We also define $h_{avg}(A) \triangleq \mathbb{E}_\Phi(h(A)) \triangleq \mathbb{E}_\Phi[h(\Phi_A)]$ which is the expected utility of set A (and states of its elements) over all possible realizations of A under the probability distribution $p(\Phi = \phi)$. We define $\Delta(e \mid \psi) = \mathbb{E}_{\Phi \sim \psi}[h(\Psi_{\{e\}} + \psi) - h(\psi)]$ which is the conditional expected marginal benefit of item e conditioned on having observed the subrealization ψ. Note that the random variable $\Psi_{\{e\}}$ is the state of item e with respect to the probability distribution $p(\Phi = \phi \mid \Phi \sim \psi)$. Similarly, we define $\Delta(A \mid \psi) = \mathbb{E}_{\Phi \sim \psi}[h(\Psi_A + \psi) - h(\psi)]$ which is the expected marginal gain of set A to the partial realization ψ. Assume $E(\pi_\phi)$ is the set of edges induced by the set of items policy π selects under the realization ϕ. The expected utility of policy π is defined as $f_{avg}(\pi) \triangleq h_{avg}(E(\pi)) = \mathbb{E}_\Phi[h(E(\pi_\Phi)]$, where the expectation is taken with respect to $p(\Phi = \phi)$. For a list of all the notations used in the paper refer to Table 1 in Appendix A.

Next, we restate the definitions for weakly adaptive set submodular and adaptive monotone functions.

Definition 1. *A function $h : 2^E \times Q^E \to \mathbb{R}_{\geq 0}$ is **weakly adaptive set submodular** with parameter γ if for all sets $A \subseteq E$ and for all $\psi \subseteq \psi'$ we have:*

$$\Delta(A \mid \psi') \leq \frac{1}{\gamma} \cdot \sum_{e \in A} \Delta(e \mid \psi).$$

Definition 2. *A function $h : 2^E \times Q^E \to \mathbb{R}_{\geq 0}$ is **adaptive monotone** if $\Delta(e \mid \psi) \geq 0$ for all partial realizations ψ. That is, the conditional expected marginal benefit of any element is non-negative.*

[2] Note that there is a one to one correspondence between a realization ϕ over the vertices and a realization ϕ^E over the edges.

Definition 1 is the generalization of both weak submodularity [13] and adaptive submodularity [22] concepts.

Next, we state a few useful claims regarding weakly adaptive submodular functions.

First note for all ψ and for every set $A \subseteq E \setminus \text{dom}(\psi)$, from Definition 1 and the fact that $\psi \subseteq \psi$, we have

$$\Delta(A \mid \psi) \leq \frac{1}{\gamma} \cdot \sum_{e \in A} \Delta(e \mid \psi). \tag{1}$$

Lemma 1. *For all ψ and $A \subseteq B \subseteq E \setminus dom(\psi)$, we have*

$$\Delta(B \mid \psi) - \Delta(A \mid \psi) \leq \frac{1}{\gamma} \cdot \sum_{e \in B \setminus A} \Delta(e \mid \psi).$$

Proof. We have

$$\begin{aligned}\Delta(B \mid \psi) - \Delta(A \mid \psi) &= \sum \Pr[\Psi_A = \psi' \mid \Phi \sim \psi] \cdot \sum \Pr[\Psi_{B \setminus A} = \psi'' \mid \Phi \sim \psi + \psi'] \\ &\qquad \cdot (h_{avg}(\psi + \psi' + \psi'') - h_{avg}(\psi + \psi')) \\ &= \sum \Pr[\Psi_A = \psi' \mid \Phi \sim \psi] \cdot \Delta(B \setminus A \mid \psi + \psi') \leq \frac{1}{\gamma} \cdot \sum_{e \in B \setminus A} \Delta(e \mid \psi),\end{aligned}$$

where the inequality is derived from the definition of weakly adaptive set submodular functions (see Definition 1) and the fact that $\sum \Pr[\Psi_A = \psi' \mid \Phi \sim \psi] = 1$. □

Corollary 1. *For all ψ, $e^* = \arg\max_{e \in E} \Delta(e \mid \psi)$ and two random subsets $A \subseteq B \subseteq E \setminus dom(\psi)$ whose randomness might depend on the realization, we have*

$$\mathbb{E}[\Delta(B \mid \psi) - \Delta(A \mid \psi) \mid \Phi \sim \psi] \leq \frac{\mathbb{E}[|B \setminus A| \mid \Phi \sim \psi]}{\gamma} \cdot \Delta(e^* \mid \psi).$$

Proof. By taking expectation over the guarantee of Lemma 1, we get

$$\begin{aligned}\mathbb{E}[\Delta(B \mid \psi) - \Delta(A \mid \psi) \mid \Phi \sim \psi] &\leq \frac{1}{\gamma} \cdot \mathbb{E}\left[\sum_{e \in B \setminus A} \Delta(e \mid \psi) \mid \Phi \sim \psi\right] \\ &\leq \frac{1}{\gamma} \cdot \mathbb{E}\left[\sum_{e \in B \setminus A} \Delta(e^* \mid \psi) \mid \Phi \sim \psi\right] \\ &= \frac{\mathbb{E}[|B \setminus A| \mid \Phi \sim \psi]}{\gamma} \cdot \Delta(e^* \mid \psi),\end{aligned}$$

where the second inequality follows from the fact that e^* is the element with the largest expected gain. □

The following observation is an immediate consequence of Definition 2.

Observation 1. *For any two (possibly random) subsets $A \subseteq B \subseteq E$, we have*

$$\mathbb{E}_\Phi(h(A)) \leq \mathbb{E}_\Phi(h(B)).$$

Lemma 2. *Assume h is adaptive monotone and weakly adaptive set submodular with a parameter γ with respect to the distribution $p(\phi)$, and π is a greedy policy which picks the item with the largest expected marginal gain at each step, then for all policies π^* we have*

$$h_{avg}(\pi) \geq \left(1 - e^{-1/\gamma}\right) \cdot h_{avg}(\pi^*).$$

Proof. The proof of this lemma follows the same line of argument as the proof of [22, Theroem 5]. □

B.2 Proof of Theorem 1

In this section, we first restate Theorem 1 and then prove it.

Theorem 1. *For adaptive monotone and weakly adaptive sequence submodular function f, the Adaptive Sequence Greedy policy π represented by Algorithm 1 achieves*

$$f_{avg}(\pi) \geq \frac{\gamma}{2d_{\text{in}} + \gamma} \cdot f_{avg}(\pi^*),$$

where γ is the weakly adaptive submodularity parameter, π^* is the policy with the highest expected value and d_{in} is the largest in-degree of the input graph G.

We assume the function h is weakly adaptive set submodular (with a parameter γ) and monotone adaptive submodular. Furthermore, we assume π^* is the optimal policy. It means π^* maximizes the expected gain over the distribution Φ.

Let $\ell = \lceil k/2 \rceil$. For every $0 \leq s \leq \ell$, let π_s be the set of items picked by the greedy policy π after s iterations (if the algorithm does not make that many iterations because the set $\mathcal{E}$ became empty at some earlier point, then we assume for the sake of the proof that the algorithm continues to make dummy iterations after the point in which $\mathcal{E}$ becomes empty, and in the dummy iterations it picks no items). The observed partial realization of edges after s iterations of the algorithm is represented by ψ_s. The random variable representing ψ_s is Ψ_s. We define $f_{avg}(\pi_s) \triangleq h_{avg}(E(\pi_s))$, i.e., it is the expected value of items picked by the greedy policy π after s iterations. For every $1 \leq s \leq \ell$, we also denote by e_s and $\mathcal{E}_s$ the values assigned to the variables e_{ij} and $\mathcal{E}$, respectively, at iteration number s. Finally, we assume e_s is a dummy arc with zero marginal contribution to h if iteration number s is a dummy iteration (i.e., the algorithm makes in reality less than s iterations).

Observation 2. *For every $0 \leq s_1 \leq s_2 \leq \ell$, conditioned on the partial realization ψ_{s_1}, i.e., the policy has already made its first s_1 iterations, we have $\mathcal{E}_{s_1} \supseteq \mathcal{E}_{s_2}$ and $E(\pi_{s_1}) \subseteq E(\pi_{s_2})$.*

Proof. Both properties guaranteed by the observation follow from the fact that: for all possible realization $\phi \sim \psi_{s_1}$, we have that π_{s_1} is a (possibly trivial) prefix of π_{s_2}. □

Lemma 3. *For every $1 \leq s \leq \ell$, $f_{avg}(\pi_s) - f_{avg}(\pi_{s-1}) \geq \mathbb{E}_{\Psi_{s-1}}[\Delta(e_s \mid \Psi_{s-1})]$.*

Proof. Consider a fixed sub-realization ψ_{s-1}. If e_s is a dummy arc, then $\pi_s = \pi_{s-1}$, and the observation is trivial. Otherwise, notice that the membership of e_s in $\mathcal{E}_{s-1}$ guarantees that it does not belong to $E(\pi_{s-1}) = \text{dom}(\psi_{s-1})$, but does belong to $E(\sigma_s)$. Together with the fact that $E(\pi_{s-1}) \subseteq E(\pi_s)$ by Observation 2, we get $E(\pi_{s-1}) + e_s \subseteq E(\pi_s)$; which implies, by the adaptive monotonicity of h,

$$\begin{aligned} f_{avg}(\pi_s) - f(\pi_{s-1}) &= \mathbb{E}_{\Phi \sim \psi_{s-1}}[f_{avg}(\pi_s)] - h(\psi_{s-1}) \\ &\geq \mathbb{E}_{\Phi \sim \psi_{s-1}}[h(\psi_{s-1} + e_s)] - h(\psi_{s-1}) \\ &= \Delta(e_s \mid \psi_{s-1}). \end{aligned}$$ □

Note that we condition on the fact that the first $s-1$ steps of the policy π are performed, therefore we have $f_{avg}(\pi_{s-1}) = h(\psi_{s-1})$. By taking expectation over all the possible realizations of the random variable Ψ_{s-1} the lemma is proven.

Lemma 4. *Conditioned on any arbitrary partial realization ψ, we have $\mathbb{E}_{\Phi \sim \psi}[|E(\pi^*)|] \leq (k-1)d_{\text{in}}$.*

Proof. The optimal policy under each realization of the random variable Φ chooses at most k items. Each one of these k items (except the first one) will have at most d_{in} incoming edges. Therefore, the expected number of edges is at most $(k-1)d_{\text{in}}$. □

Lemma 5. *For every $1 \leq s \leq \ell$, we have*

$$\mathbb{E}_{\Phi}[h((E(\pi^*) \cap \mathcal{E}_{s-1}) \cup E(\pi_{s-1}))] \leq$$
$$\mathbb{E}_{\Phi}[h((E(\pi^*) \cap \mathcal{E}_s) \cup E(\pi_s))] + \frac{1}{\gamma} \cdot \mathbb{E}_{\Phi}[|E(\pi^*) \cap (\mathcal{E}_{s-1} \setminus \mathcal{E}_s)| \cdot \Delta(e_s \mid E(\pi_{s-1}))].$$

Note that the expectation is taken over all the possible realizations of the random variable Φ.

Proof. The lemma follows by combining the two inequalities of Eq. (2) and Eq. (3).

$$\begin{aligned}
&\mathbb{E}_\Phi[\Delta(E(\pi^*)\cap\mathcal{E}_{s-1}\mid E(\pi_{s-1}))]-\mathbb{E}_\Phi[\Delta(E(\pi^*)\cap\mathcal{E}_s\mid E(\pi_{s-1}))] \qquad (2)\\
&=\sum \Pr[\Psi_{s-1}=\psi_{s-1}]\cdot\left[\mathbb{E}_{\Phi\sim\psi_{s-1}}[\Delta(E(\pi^*)\cap\mathcal{E}_{s-1}\mid\psi_{s-1})-\Delta(E(\pi^*)\cap\mathcal{E}_s\mid\psi_{s-1})]\right]\\
&\overset{(a)}{\leq}\frac{1}{\gamma}\sum \Pr[\Psi_{s-1}=\psi_{s-1}]\cdot\mathbb{E}_{\Phi\sim\psi_{s-1}}[|E(\pi^*)\cap(\mathcal{E}_{s-1}\setminus\mathcal{E}_s)|\cdot\Delta(e_s\mid\psi_{s-1})]\\
&=\frac{1}{\gamma}\cdot\mathbb{E}_\Phi[|E(\pi^*)\cap(\mathcal{E}_{s-1}\setminus\mathcal{E}_s)|\cdot\Delta(e_s\mid E(\pi_{s-1}))].
\end{aligned}$$

To see why inequality (a) is true, note that for every given sub realization ψ_{s-1} we have: (i) if e_s is a dummy edge, then $(E(\pi^*)\cap\mathcal{E}_{s-1})\cup E(\pi_{s-1})=(E(\pi^*)\cap\mathcal{E}_s)\cup E(\pi_s)$, which makes (a) trivial, or (ii) when e_s is not dummy, (a) results from Corollary 1.

$$\begin{aligned}
&\mathbb{E}_\Phi[h((E(\pi^*)\cap\mathcal{E}_{s-1})\cup E(\pi_{s-1}))]-\mathbb{E}_\Phi[h((E(\pi^*)\cap\mathcal{E}_s)\cup E(\pi_s))] \qquad (3)\\
&\leq\mathbb{E}_\Phi[h((E(\pi^*)\cap\mathcal{E}_{s-1})\cup E(\pi_{s-1}))]-\mathbb{E}_\Phi[h((E(\pi^*)\cap\mathcal{E}_s)\cup E(\pi_{s-1}))])\\
&=\sum \Pr[\Psi_{s-1}=\psi_{s-1}]\cdot\mathbb{E}_{\Phi\sim\psi_{s-1}}[h((E(\pi^*)\cap\mathcal{E}_{s-1})\cup\psi_{s-1})-h(E(\pi^*)\cap\mathcal{E}_s)\cup\psi_{s-1})]\\
&=\sum \Pr[\Psi_{s-1}=\psi_{s-1}]\cdot\mathbb{E}_{\Phi\sim\psi_{s-1}}[\Delta((E(\pi^*)\cap\mathcal{E}_{s-1})\mid\psi_{s-1})-\Delta(E(\pi^*)\cap\mathcal{E}_s)\mid\psi_{s-1})]\\
&=\mathbb{E}_\Phi[\Delta(E(\pi^*)\cap\mathcal{E}_{s-1}\mid E(\pi_{s-1}))]-\mathbb{E}_\Phi[\Delta(E(\pi^*)\cap\mathcal{E}_s\mid E(\pi_{s-1}))]. \qquad \square
\end{aligned}$$

Lemma 6. $\mathbb{E}_\Phi[h((E(\pi^*)\cap\mathcal{E}_\ell)\cup E(\pi_\ell))]]\leq\frac{1}{\gamma}\cdot\mathbb{E}_\Phi[|E(\pi^*)\cap\mathcal{E}_\ell|\cdot\Delta(e_\ell\mid\Psi_{\ell-1})]+f_{avg}(\pi_\ell)$.

Proof. We have

$$\begin{aligned}
&\mathbb{E}_\Phi[h((E(\pi^*)\cap\mathcal{E}_\ell)\cup E(\pi_\ell))-h(\pi_\ell)]\\
&=\sum \Pr[\Psi_\ell=\psi_\ell]\cdot\mathbb{E}_{\Phi\sim\psi_\ell}[h(E(\pi^*)\cap\mathcal{E}_\ell)\cup\psi_\ell)-h(\psi_\ell)]\\
&\overset{(a)}{\leq}\frac{1}{\gamma}\sum \Pr[\Psi_\ell=\psi_\ell]\cdot\mathbb{E}_{\Phi\sim\psi_\ell}[|E(\pi^*)\cap\mathcal{E}_\ell|\cdot\Delta(e_\ell\mid\psi_{\ell-1})]=\frac{1}{\gamma}\cdot\mathbb{E}_\Phi[|E(\pi^*)\cap\mathcal{E}_\ell|\cdot\Delta(e_\ell\mid\Psi_{\ell-1})].
\end{aligned}$$

To see why inequality (a) is true, note that for every given sub realization ψ_ℓ we have: (i) if e_ℓ is a dummy edge, then $\mathcal{E}_\ell=\varnothing$, which makes inequality (a) trivial, and (ii) if e_ℓ is not a dummy edge then we conclude inequality (a) from the definition of weakly adaptive set submodular functions (see Definition 1).

The lemma follows by combining this inequality with the observation that $f_{avg}(\pi_\ell)=\mathbb{E}_\Phi[h(\pi_\ell)]$. $\square$

To combine the last two lemmata, we need the following observation.

Observation 3. *For every* $2\leq s\leq\ell$, $\mathbb{E}_\Phi[\Delta(e_{s-1}\mid E(\pi_{s-2}))]\geq\gamma\cdot\mathbb{E}_\Phi[\Delta(e_s\mid E(\pi_{s-1}))]$.

We are now ready to prove Theorem 1.

Proof of Theorem 1. Combining Lemmata 5 and 6, we get

$$\begin{aligned}
f_{avg}(\pi^*)-f_{avg}(\pi_\ell)=&\mathbb{E}_\Phi[h((E(\pi^*)\cap\mathcal{E}_1)\cup E(\pi_0))]-f_{avg}(\pi_\ell)\\
\leq&\frac{1}{\gamma}\cdot\sum_{s=1}^{\ell}\mathbb{E}_\Phi[|E(\pi^*)\cap(\mathcal{E}_{s-1}\setminus\mathcal{E}_s)|\cdot\Delta(e_s\mid E(\pi_{s-1}))]\\
&+\mathbb{E}_\Phi[\Delta((E(\pi^*)\cap\mathcal{E}_s)\cup E(\pi_s))]-f_{avg}(\pi_\ell)\\
\leq&\frac{1}{\gamma}\sum_{s=1}^{\ell}\mathbb{E}_\Phi[|E(\pi^*)\cap(\mathcal{E}_{s-1}\setminus\mathcal{E}_s)|\cdot\Delta(e_s\mid E(\pi_{s-1}))]\\
&+\frac{1}{\gamma}\cdot\mathbb{E}_\Phi[|E(\pi^*)\cap\mathcal{E}_\ell|\cdot\Delta(e_\ell\mid\Psi_{\ell-1})]
\end{aligned}$$

$$= \frac{1}{\gamma} \cdot \sum_{s=1}^{\ell-1} \mathbb{E}_{\Phi}[|E(\pi^*) \cap (\mathcal{E}_0 \setminus \mathcal{E}_s)| \cdot [\Delta(e_s \mid E(\pi_{s-1})) - \Delta(e_{s+1} \mid E(\pi_s))]]$$
$$+ \frac{1}{\gamma} \cdot \mathbb{E}_{\Phi}[|E(\pi^*) \cap \mathcal{E}_0| \cdot \Delta(e_\ell \mid E(\pi_{\ell-1}))], \quad (4)$$

where the first equality holds since the fact that σ_0 is an empty sequence implies $E(\sigma_0) = \varnothing$ and $\mathcal{E}_0 = E$, and the second equality holds since $\mathcal{E}_s \subseteq \mathcal{E}_{s-1}$ by Observation 2 for every $1 \leq s \leq \ell$. We now observe that for every $1 \leq s \leq \ell$, π_s contains at most $2s$ vertices. Since each one of these vertices can be the end point of at most d_{in} arcs, we get

$$|E(\sigma^*) \cap (\mathcal{E}_0 \setminus \mathcal{E}_s)| \leq |\mathcal{E}_0 \setminus \mathcal{E}_s| \leq 2sd_{\text{in}}$$

Additionally, by Lemma 4,

$$|E(\sigma^*) \cap \mathcal{E}_0| \leq |E(\sigma^*)| \leq (k-1)d_{\text{in}} \leq 2\ell d_{\text{in}}.$$

Plugging the last two inequalities into Inequality (4) yields

$$f_{avg}(\pi^*) - f_{avg}(\pi_\ell) \leq \sum_{s=1}^{\ell-1} \frac{2sd_{\text{in}}}{\gamma} \cdot \mathbb{E}_{\Phi}[\Delta(e_s \mid E(\pi_{s-1})) - \Delta(e_{s+1} \mid E(\pi_s))] +$$
$$\frac{2\ell d_{\text{in}}}{\gamma} \cdot \mathbb{E}_{\Phi}[\Delta(e_\ell \mid E(\sigma_{\ell-1}))]$$
$$= \sum_{s=1}^{\ell} \frac{2d_{\text{in}}}{\gamma} \cdot \mathbb{E}_{\Phi}[\Delta(e_s \mid E(\pi_{s-1}))] \leq \frac{2d_{\text{in}}}{\gamma} \cdot \sum_{s=1}^{\ell} [f_{avg}(\pi_s) - f_{avg}(\pi_{s-1})]$$
$$= \frac{2d_{\text{in}}}{\gamma} \cdot [f_{avg}(\pi_\ell) - f_{avg}(\pi_0)] \leq \frac{2d_{\text{in}}}{\gamma} \cdot f_{avg}(\pi_\ell),$$

where the second inequality holds due to Lemma 3 and the last inequality follows from the non-negativity of f. Rearranging the last inequality, we get

$$f_{avg}(\pi_\ell) \geq \frac{\gamma}{2d_{\text{in}} + \gamma} \cdot f_{avg}(\pi^*),$$

which implies the theorem since $f_{avg}(\pi_\ell)$ is a lower bound on the expected value of the output sequence of Algorithm 1 because σ_ℓ is always a prefix of this sequence. □

B.3 Proof of Theorem 2

In this section, we first restate and then prove Theorem 2 which guarantees the performance of our proposed policy applied to hypergraphs.

Theorem 2. *For adaptive monotone and weakly adaptive sequence submodular function f, the policy π' represented by Algorithm 2 achieves*

$$f_{avg}(\pi') \geq \frac{\gamma}{rd_{\text{in}} + \gamma} \cdot f_{avg}(\pi^*),$$

where γ is the weakly adaptive submodularity parameter, π^* is the policy with the highest expected value and r is the size of the largest hyperedge in the input hypergraph.

In the proof of this theorem we use the same notation that we used in Section B.2 for analyzing Algorithm 1, with the exception of $\mathcal{E}_s$, which is now defined as $\mathcal{E}_s = \{e \in E \mid \sigma_s \cap V(e) \text{ is a prefix of } e\}$, and ℓ, which is now defined as $\lfloor k/r \rfloor$.

The following lemma is a counterpart of Lemma 4.

Lemma 7. $|E(\sigma^*)| \leq (k - r + 1)d_{\text{in}}$.

Proof. For a realization ϕ, every arc of π^* must end at a vertex of π^* which is not one of the first $r-1$ vertices. The observation follows since π^* contains at most $k - r + 1$ vertices of this kind, and at most d_{in} arcs can end at each one of them. □

Algorithm 2 Adaptive Hyper Sequence Greedy

Require: Directed hypergraph $H(V, E)$, γ-adaptive and adaptive-monotone function $h : 2^E \times O^E \to \mathbb{R}_{\geq 0}$ and cardinality parameter k
Let $\sigma \leftarrow ()$
while $|\sigma| \leq k - r$ **do**
  $\mathcal{E} = \{e \in E \mid \sigma \cap V(e)$ is a prefix of $e\}$
  if $\mathcal{E} \neq \varnothing$ **then**
    $e^* = \arg\max_{e \in \mathcal{E}} \Delta(e \mid \psi_\sigma)$
    for every v $\in e^*$ in order **do**
      if $v \notin \sigma$ **then**
        $\sigma = \sigma \oplus v$
      end if
    end for
    Identify the state of all edges in $\mathcal{E}' = \{e \in E \mid$ all elements of $V(e)$ belong to σ and appear in the same order$\}$
    $\psi_\sigma = \psi_{\mathcal{E}'}$
  else
    break
  end if
end while
Return σ

One can observe that the proofs of all the other observations and lemmata of Section B.2 are unaffected by the differences between Algorithm 1 and Algorithm 2, and thus, these observations and lemmata can be used towards the proof of Theorem 2.

Proof of Theorem 2. The proof of this theorem is identical to the proof of Theorem 1 up to two changes. First, instead of getting an upper bound of $2sd_{\text{in}}$ on $|\mathcal{E}_0 \setminus \mathcal{E}_s|$ for every $1 \leq s \leq \ell$, we now get an upper bound of rsd_{in} on this expression because σ_s might contain up to rs vertices rather than only $2s$. Second, instead of getting an upper bound of $2\ell d_{\text{in}}$ on $|E(\sigma^*)|$, we now use Lemma 7 to get an upper bound of $(k - r + 1)d_{\text{in}} \leq r\ell d_{\text{in}}$ on this expression. □

C Proof of Theorem 3

The approximability of the sequence submodular maximization, as a generalization of the densest k subgraph problem (DkS) [32], is an open theoretical question with important implications. In this section, we prove Theorem 3.

In the DkS problem the goal is to find a subgraph on exactly k vertices that contains the maximum number of edges. DkS as a generalization of the k-clique problem is NP-hard and the best polynomial algorithm for DkS achieves a $n^{1/4+\epsilon}$ approximation factor[3] for an arbitrary $\epsilon > 0$ [8]. Furthermore, there exists no polynomial time algorithm that approximates DkS within an $O(n^{1/(\log\log n)^c})$ factor unless 3-SAT has a subexponential time algorithm [40].

Lemma 8. *Any algorithm with an α approximation factor to the sequence submodular maximization problem solves the densest k subgraph problem (DkS) with at most an α approximation factor.*

Proof. To prove this lemma, we show that for each instance of DkS over a directed graph $G(V, E)$ we can build an instance of the sequence submodular maximization problem over a directed graph $H(V, E')$ such that solving the latter problem also solves the former one. We assume all vertices and edges have a single state. Therefore, the problem translates to the non-adaptive sequence submodular scenario.

Graph H is built from graph G by replacing each edge $e = (u, v)$ in E by two directed edges (u, v) and (v, u). We define $h(S) = |S|$, which is linear and therefore submodular. Finally, the sequence

[3]Note that in this section we define the approximation factor as the ratio of the the optimal solution to the solution provided by the algorithm.

submodular function f is defined as $f(\sigma) = h(E(\sigma)) = |E(\sigma)|$. It remains to show that for every subset of vertices S the value of function f for an arbitrary permutation σ_S of S is equivalent to the size of subgraph G_S induced by those vertices in graph G. This is true because for every edge $(u, v) \in G_S$ we have two corresponding edges in the directed graph H and based on the order of u and v exactly one of them is considered in $E(\sigma_S)$.

As a result, maximizing the function f with a cardinality constraint k is equivalent to solving the DkS problem. Thus, any algorithm with an α approximation factor to the sequence submodular maximization problem solves DkS with at least an α approximation factor. □

Manurangsi [40] showed that any algorithm with an $O(n^{1/(\log\log n)^c})$ approximation factor to the DkS problem (for a constant $c > 0$) would prove the exponential time hypothesis is false. Next, we directly state the result of [40].

Theorem 4 (Manurangsi [40], Theorem 1)**.** *There is a constant $c > 0$ such that, assuming the exponential time hypothesis, no polynomial-time algorithm can, given a graph G on n vertices and a positive integer $k \leq n$, distinguish between the following two cases:*

- *There exist k vertices of G that induce a k-clique.*
- *Every k-subgraph of G has density at most $n^{-1/(\log\log n)^c}$.*

To sum-up, Theorem 3 is proved from the combination of the two following facts:

1. If there is an algorithm with an approximation within a $n^{1/(\log\log n)^c}$ factor to the sequence submodular maximization problem, from the result of Lemma 7, we know that it would solve the DkS problem with at most the same factor.
2. If there is an algorithm with a $n^{1/(\log\log n)^c}$ approximation factor to the DkS problem, it could distinguish the two cases of Theorem 4 and would prove the exponential time hypothesis to be false.

D Additional Experimental Details

D.1 Amazon Product Recommendation

In this application, we consider the task of recommending products to users. In particular, we use the Amazon Video Games review dataset [41], which contains 10,672 products, 24,303 users, and 231,780 confirmed purchases. We furthered focused on the products that had been purchased at least 50 times each, leaving us with a total of 958 unique products.

Although we are using a different dataset, the experimental set-up closely follows that of the movie recommendation task in Tschiatschek et al. [53] and Mitrovic et al. [44]. We first group and sort all the data so that each user u has an associated sequence σ_u of products that they have purchased. These user sequences are then randomly partitioned into a training set and a testing set using a 80/20 split. Note that we 5 trials to average our results.

Using the training set, we build a graph $G = (V, E)$, where V is the set of all products and E is the set of edges between these products. Each product $i \in V$ has a self-loop (i, i), where the weight (denoted w_{ii}) is the fraction of users in the training set that purchased product v_i. Similarly, for each edge (i, j), the corresponding weight w_{ij} is defined to be the conditional probability of purchasing product j given that the user has previously purchased product i.

For each sequence σ_u in the test set, we are given the first g products that user u purchased, and then we want to predict the next k products that she will purchase. After each product is recommended to the user, the state of the product is revealed to be 1 if the user has indeed purchased that product, and 0 otherwise. At the start, the g given products are known to be in state 1, while the states of the remaining products are initially unknown.

As described in Section 2, the states of the edges are determined by the states of the nodes. In this case, the state of each edge (i, j) is equal to the state of product i. The intuitive idea is that edge (i, j) encodes the value of purchasing product j after already having purchased product i. Therefore, if

the user has definitely purchased product i (i.e., product i is in state 1), then they should receive the full value of w_{ij}. On the other hand, if she has definitely not purchased product i (i.e., product i is in state 0), then edge (i, j) provides no value. Lastly, if the state of product i is unknown, then the expected gain of edge (i, j) is discounted by w_{ii}, the value of the self-loop on i, which can be viewed as a simple estimate for the probability of the user purchasing product i. See Figure 2a for a small example.

We use a probabilistic coverage utility function as our monotone adaptive submodular function h. Mathematically,

$$h(E_1) = \sum_{j \in V} \Big[1 - \prod_{(i,j) \in E_1} (1 - w_{ij})\Big],$$

where $E_1 \subseteq E$ is the subset of edges that are in state 1.

D.2 Wikipedia Link Prediction

We use the Wikispeedia dataset [56], which consists of 51,138 completed search paths on a condensed version of Wikipedia that contains 4,604 pages and 119,882 links between them. We further condense the dataset to include only articles that have been visited at least 100 times, leaving us with 619 unique pages and 7,399 completed search paths.

One natural idea for scoring each algorithm would be to look at the length of the shortest path between the predicted target and the true target. However, the problem with this metric is that all the popular pages have relatively short paths to most potential targets (primarily since they have so many available links to begin with). Hence, under this scoring, just choosing a popular page like "Earth" would be competitive with many more involved algorithms.

Instead, we define a measure we call the *Relevance Distance*. The relevance distance of a page i to a target page j is calculated by taking the average shortest path length to j across all neighboring pages of i. A lower distance indicates a higher relevance. For example, if our target page is *Computer Science*, both *Earth* $\rightarrow$ *Earth Science* $\rightarrow$ *Computer Science* and *University* $\rightarrow$ *Education* $\rightarrow$ *Computer Science* have a shortest path of length 2. However, the relevance distance of *Earth* to *Computer Science* is 2.68, while the relevance distance of *University* to *Computer Science* is 2.41, which fits better with the intuition that *University* is logically closer to *Computer Science*.

D.3 Deep Learning Baseline Details

D.3.1 Feed Forward Neural Network

For both experiments, the input to the Feed Forward Neural Network is a size $|V|$ vector X. That is, there is one input for each item in the ground set. In the Amazon product recommendation task in Section 4.1, $X_i = 1$ if the user is known to have purchased product i and 0 otherwise. Similarly, for the Wikipedia link prediction task in Section 4.2, $X_i = 1$ if the user is known to have visited page i and 0 otherwise.

The output in both cases is a size $|V|$ soft-maxed vector Y. In Section 4.1, Y_i can be viewed as the probability that product i will be the user's next purchase. In Section 4.2, Y_i can be viewed as the probability that user will visit page i next.

For the Amazon product recommendation task in Section 4.1, each user u in the training set has an associated sequence σ_u of products she purchased. Each such sequence was split into $|\sigma_u| - 2$ training points by taking the first g products as input and the $(g + 1)$-th product as the output for $g = 1, \ldots, |\sigma_u| - 1$. For each user u in the testing set, we would take the first $g = 4$ products she purchased and encode them in the vector X as described above. We would then input this vector into our trained network and output the vector Y. In the non-adaptive case we cannot get any feedback from the user, so we simply output the products corresponding to the k highest values in Y.

In the adaptive case, we would look at the largest value Y_j in our output vector and output this as our first recommendation. We then check if the corresponding product appeared somewhere later in the user's sequence σ_u. If yes, then we would update our input X so that $X_j = 1$ and re-run the network to get our next recommendation. If not, we would simply use the next highest value in Y_j as our next recommendation (since the input doesn't change). This was repeated for k recommendations. This is supposed to mimic interaction with the user where we would recommend a product, and then see

whether or not the user actually purchases this product. Note that we only considered values Y_j such that $X_j = 0$ because we did not want to recommend products that we knew the user had already purchased.

The main difference for the Wikipedia task in Section 4.2 is that, in the testing phase, we cannot simply output the top k values in Y as we did above because they likely will not constitute a valid path. Instead, we only have an adaptive version that is similar to what was described above. We find the highest value Y_j such that $X_j = 0$ (i.e. the user had not already been to this page) and a link to page j actually exists from our current page. We output this page j as our recommendation for the user's next page. We then check if the user actually visited our predicted page j at some point in their sequence of pages. If yes, we would update X so that $X_j = 1$ and re-run the network. If not we would look to the next highest value in the output Y. This was repeated for k guesses. Note that if we reached the true target page, we would stop making guesses.

In terms of architecture, we used a single hidden layer of 256 nodes with ReLU activations. We use a batch size of 1024 at first and then go down to a batch size of 32 when we are in the low data regime (i.e. only using 1% of the available training data). We used an 80/20 training/validation split to guide our early stopping criterion during training (with minimum improvement of 0.01 and patience of 1). We used categorical cross-entropy as our loss function.

D.3.2 LSTM

The main difference between the LSTM and the feed forward network is in the input. The input to the LSTM is a sequence of one-hot encoded vectors instead of just a single vector. That is, for the LSTM, each vector in the sequence had exactly one index with value 1.

We experimented with using a long sequence of input vectors and padding with all-zero vectors, but we found better results using a fixed small sequence length g and then "pushing" the sequence back when updating. For example, if our current input was a sequence of vectors $[v_1, v_2, v_3]$ and we wanted to update it with a new vector v_4, the updated input would be $[v_2, v_3, v_4]$.

The adaptive LSTM followed the same set-up as the non-adaptive LSTM, but with the same adaptive update rules described above for the feed-forward neural network.

For all experiments, we used a single hidden layer of 8 LSTM nodes. The other hyperparameters are all the same as described for the Feed Forward network above, except we start at a batch size of 256 instead of 1024 (before also going down to a batch size of 32 in the low data regime).